JÜRİ VE ENSTİTÜ ONAYI

Cihan Kaleli'nin "**Privacy-preserving Distributed Collaborative Filtering**" başlıklı **Bilgisayar Mühendisliği** Anabilim Dalındaki, Doktora Tezi 04.05.2012 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	**İmza**
Üye (Tez Danışmanı)	**: Doç. Dr. HÜSEYİN POLAT**	**…..…….**
Üye	**: Doç. Dr. YUSUF OYSAL**	**…..…….**
Üye	**: Doç. Dr. AYDIN AYBAR**	**…..…….**
Üye	**: Doç. Dr. AHMET UYAR**	**..…….**
Üye	**: Yard. Doç. Dr. GÜRKAN ÖZTÜRK**	**…..…….**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …… tarih ve …… sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ABSTRACT
Ph.D. Dissertation
PRIVACY-PRESERVING DISTRIBUTED COLLABORATIVE FILTERING
Cihan KALELİ
Anadolu University
Graduate School of Sciences
Computer Engineering Program

Supervisor: Assoc. Prof. Dr. Hüseyin POLAT
2012, 162 pages

In order to provide accurate and dependable recommendations, online vendors need to have adequate data; however, due to the nature of online shopping and increasing amount of e-commerce sites, data collected for collaborative filtering purposes might be distributed among various companies, even competing ones. Those online vendors holding distributed data might want to offer predictions based on integrated data collaboratively. However, concerns regarding protecting private data, financial fears due to revealing valuable assets, and legal regulations imposed by various organizations prevent them from alliance.

In this dissertation, various solutions are proposed to enable online vendors' collaboration for estimating recommendations on vertically or horizontally distributed data while preserving their confidentiality. The proposed solutions mainly employ randomized and cryptographic techniques for protecting privacy. To improve online performance, which may become worse due to collaboration, preprocessing methods such as clustering, dimensionality reduction, and trust are utilized. The recommended methods are analyzed in terms of privacy. Also, superfluous loads caused by privacy concerns are examined. Finally, real data-based trials are performed for evaluating the proposed schemes in terms of the quality of predictions. The analyses and experimental outcomes demonstrate that the methods preserve confidentiality, cause insignificant overheads, and offer accurate recommendations.

Keywords: Privacy, Distributed Data, Collaborative Filtering, SOM, Naïve Bayesian Classifier, Random Projection, and Performance.

ÖZET
Doktora Tezi
GİZLİLİĞİ KORUYARAK DAĞITIK ORTAK SÜZGEÇLEME
Cihan KALELİ
Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Hüseyin POLAT
2012, 162 sayfa

Doğru ve güvenilir öneriler üretebilmek için sanal alışveriş siteleri yeterli veriye ihtiyaç duyarlar. Fakat sanal alışverişin doğası gereği ve e-ticaret sitelerinin sayısındaki artış nedeniyle ortak süzgeçleme amacıyla toplanmış veriler çeşitli siteler arasında dağıtık olmuş olabilir. Bu dağıtık veriye sahip e-ticaret siteleri bütünleştirilmiş veri üzerinden ortak öneriler sunmak isteyebilirler. Fakat gizli verilerini koruma düşüncesi, finansal korkular ve yasal zorunluluklardan dolayı bu siteler işbirliği yapmak istemeyebilirler.

Bu doktora tez çalışmasında, yatay veya dikey dağıtık veri üzerinden veri sahiplerinin gizliliklerini koruyarak öneriler üretmek için işbirliğini sağlayacak çeşitli çözümler önerilmiştir. Gizliliği korumak için rasgele karıştırma ve kriptografi tabanlı çözümler kullanılmıştır. İşbirliği nedeniyle kötüleşebilecek çevrim içi performansı artırmak için kümeleme, boyut indirgeme ve güven tabanlı benzerlik gibi ön işleme metotları kullanılmıştır. Önerilen yöntemler gizlilik açısından analiz edilmiştir. Ayrıca, gizlilik endişesi nedeniyle ortaya çıkan ilave yükler irdelenmiştir. Son olarak, çeşitli gerçek veriye dayalı deneyler yapılmış ve önerilen çözümler doğruluk açısından incelenmiştir. Yapılan analizler ve deney sonuçları önerilen çözümlerin gizliliği koruduğunu, önemsenmeyecek miktarda ilave yükler getirdiğini ve kaliteli öneriler üretebildiğini göstermiştir.

Anahtar Kelimeler: Gizlilik, Dağıtık Veri, Ortak Süzgeçleme, SOM, Basit Bayes Sınflandırıcı, Rastgele Projeksiyon ve Performans.

ACKNOWLEDGEMENTS

I am truly indebted and thankful to my advisor Assoc. Prof. Dr. Hüseyin Polat for the support and the guidance he showed me throughout my research. I am sure it would have not been possible without his help. It has been a pleasure to work with him. His wisdom, knowledge, and commitment to the highest standards inspired and motivated me.

I am thankful to Assoc. Prof. Dr. Yusuf Oysal, Assoc. Prof. Dr. Aydın Aybar, Assoc. Prof. Dr. Ahmet Uyar, and Assist. Prof. Dr. Gürkan Öztürk on my dissertation committee for their valuable contributions.

I would like to thank my research friends, İbrahim Yakut and Alper Bilge, for their scientific support.

I also would like to thank my parents. I am very grateful for their efforts on me.

Finally, I am very grateful to my wife for her endless support and patience during my study.

Cihan Kaleli

May, 2012

CONTENTS

ABSTRACT ..... **I**

ÖZET ..... **II**

ACKNOWLEDGEMENTS ..... **III**

LIST OF TABLES ..... **VII**

LIST OF FIGURES ..... **VIII**

ABBREVIATIONS ..... **X**

1. INTRODUCTION **1**

1.1. Collaborative Filtering 1

1.2. Challenges of Collaborative Filtering 4

1.3. Solutions for Eliminating Challenges 6

1.4. Privacy-Preserving Distributed Data Mining 10

1.5. Problem Definition 12

1.6. Utilized Collaborative Filtering Algorithms 14

1.7. Utilized Privacy-Preserving Techniques 16

1.8. Contributions 18

1.9. Organization of the Dissertation 19

2. PRIVACY-PRESERVING SOM-BASED RECOMMENDATIONS ON DISTRIBUTED DATA **20**

2.1. Introduction 20

2.2. SOM Clustering and Collaborative Filtering 21

2.3. Private SOM-based Recommendations on VDD 22

2.4. Private SOM-based Recommendations on HDD 29

2.5. Privacy Analysis 33

2.6. Supplementary Cost Analysis 38

2.7. Accuracy Analysis: Experiments 41

2.8. Conclusions 54

3. PRIVACY-PRESERVING RANDOM PROJECTION-BASED RECOMMENDATIONS ON DISTRIBUTED DATA **55**

3.1. Introduction 55

3.2. VDD-based Recommendations with Privacy 57
3.3. HDD-based Recommendations with Privacy 61
3.4. Privacy Analysis 64
3.5. Supplementary Costs Analysis 66
3.6. Accuracy Analysis: Experiments 67
3.7. Conclusions 76

4. PRIVACY-PRESERVING TRUST-BASED RECOMMENDATIONS ON DISTRIBUTED DATA 78

4.1. Introduction 78
4.2. Trust-based Collaborative Filtering Algorithm 79
4.3. Trust-based Recommendations on VDD 80
4.4. Trust-based Recommendations on HDD 85
4.5. Privacy Analysis 90
4.6. Supplementary Costs Analysis 92
4.7. Accuracy Analysis: Experiments 93
4.8. Conclusions 98

5. PRIVACY-PRESERVING NAÏVE BAYESIAN CLASSIFIER-BASED RECOMMENDATIONS ON DISTRIBUTED DATA 99

5.1. Introduction 99
5.2. Protecting Active User's Data 100
5.3. Privacy-Preserving NBC-based HDD Schemes 101
5.4. Privacy-Preserving NBC-based VDD Schemes 104
5.5. Privacy Analysis 106
5.6. Supplementary Cost Analysis 107
5.7. Accuracy Analysis: Experiments 109
5.8. Conclusions 116

6. PRIVACY-PRESERVING NAÏVE BAYESIAN CLASSIFIER-BASED P2P COLLABORATIVE FILTERING 118

6.1. Introduction 118
6.2. P2P NBC-based Collaborative Filtering with Privacy 119
6.3. Privacy Attacks 122

6.4. Privacy Analysis 124
6.5. Supplementary Costs Analysis 126
6.6. Accuracy Analysis: Experiments 127
6.7. Conclusions 131

7. CONCLUSIONS AND FUTURE WORK 133

REFERENCES 136

LIST OF TABLES

2.1. Data Sets 41
2.2. MAEs with Varying γ Values 50
3.1. Effects of Collaboration in VD2RP Scheme 70
3.2. Effects of Varying *d* Values (ML) 71
3.3. Effects of Varying *d* Values (EM) 71
3.4. PNFA's Hit Ratio (%) vs. *k* & *z* 74
3.5. Overall Performance of the Proposed Schemes 76
5.1a. Overall Performance by Integrating HDD (Jester, *n* = 100) 110
5.2a. Overall Performance by Integrating HDD (EM, *n* = 100) 110
5.1b. Overall Performance by Integrating HDD (Jester, *n* = 250) 111
5.2b. Overall Performance by Integrating HDD (EM, *n* = 250) 111
5.1c. Overall Performance by Integrating HDD (Jester, *n* = 500) 111
5.2c. Overall Performance by Integrating HDD (EM, *n* = 500) 111
5.1d. Overall Performance by Integrating HDD (Jester, *n* = 1,000) 112
5.2d. Overall Performance by Integrating HDD (EM, *n* = 1,000) 112
5.3a. Overall Performance by Integrating VDD (EM, *m* = 400) 113
5.3b. Overall Performance by Integrating VDD (EM, *m* = 800) 113
5.3c. Overall Performance by Integrating VDD (EM, *m* = 1,648) 113
5.4. Effects of Adding Random Users 114
5.5. Effects of Adding Random Ratings 115
5.6. Overall Performance with Varying β_g Values 116

LIST OF FIGURES

1.1. Basic Scheme of VDD 13
1.2. Basic Scheme of HDD 13
1.3. Basic Scheme of *k-nn* Algorithm 15
2.1. SOM-based Collaborative Filtering Scheme 22
2.2. NMAEs with Varying Number of Collaborating Vendors 43
2.3. NMAEs with Varying θ Values for Train Data 44
2.4. NMAEs with Varying θ Values for Test Data 45
2.5. NMAEs with Varying *n* Values 46
2.6. MAEs with Varying *n* Values 46
2.7. Coverage with Varying *n* & *z* Values 48
2.8a. MAEs with Varying *n* & *z* Values (Jester) 49
2.8b. MAEs with Varying *n* & *z* Values (ML) 49
2.9a. MAEs with Varying δ Values (Jester) 51
2.9b. MAEs with Varying δ Values (ML) 52
2.10a. Joint Effects of Varying γ and δ Values on MAEs (Jester) 53
2.10b. Joint Effects of Varying γ and δ Values on MAEs (ML) 53
3.1. Example of VDD-based recommendations with privacy with 3 Parties 62
3.2. Example of HDD-based recommendations with privacy with 3 Parties 64
3.3. Coverage with Varying *n* Values 68
3.4. Effects of Collaboration in HD2RP Scheme 69
3.5. Effects of PNFA on Accuracy (ML) 73
3.6. Effects of PNFA on Accuracy (EM) 73
3.7. Accuracy vs. Varying β Values 75
4.1. Effects of Collaboration on Accuracy with Varying *z* Values 95
4.2. ARE Values with Varying *z* Values 96
4.3. Effects of Collaboration on Reliability with Varying *z* Values 97
4.4. Effects of Varying α Values on Accuracy 98
6.1. CA with Varying α_a Values 128
6.2. F1 with Varying α_a Values 128
6.3. CA with Varying α_g Values 129
6.4. F1 with Varying α_g Values 130

6.5. CA with Varying n_p Values 131

ABBREVIATIONS

a	:	Active User
A	:	Active User Vector
c	:	Number of Clusters
CA	:	Classification Accuracy
CF	:	Collaborative Filtering
D	:	Rating Data
D′	:	Perturbed Rating Data
EM	:	EachMovie Data Set
F1	:	F-Measure
HDD	:	Horizontally Distributed Data
HE	:	Homomorphic Encryption
HF		Hiding Failure
IC	:	Initial Company
k	:	Number of Nearest Neighbors
KN	:	Kohonen Network
m	:	Number of Items
MAE	:	Mean Absolute Error
MC	:	Master Company
ML	:	MovieLens Data
n	:	Number of Users
NBC	:	Naïve Bayesian Classifier
NG	:	Number of Groups
NMAE	:	Normalized Mean Absolute Error
NN	:	Set of Nearest Neighbors
PCA	:	Principle Component Analysis
PCC	:	Pearson Correlation Coefficient
PPCF	:	Privacy-Preserving Collaborative Filtering
PPDDM	:	Privacy-Preserving Distributed Data Mining
P2P	:	Peer to Peer
R	:	Random Matrix
RP	:	Random Projection

RPT	:	Randomized Perturbation Techniques
RRT	:	Randomized Response Techniques
q	:	Target Item
SOM	:	Self-Organizing Map
SVD	:	Singular Value Decomposition
$t_{a \to u}$	:	Trust between user a and u
VDD	:	Vertically Distributed Data
z	:	Number of Parties

Dedicated to my dear son Kerem Utku.

1. INTRODUCTION

Technological developments simplify individuals' lives. It is possible to carry out daily tasks such as searching an unknown topic, following written and visual media, constructing friendship, shopping etc. through the Internet. Since people are interested in conveniences of the Internet, there are lots of competing web sites; and it causes information overload that diminishes people's decision-making ability. Therefore, it is inevitable to be confused while choosing the right information. To overcome the information overload problem, information filtering schemes are proposed. As the information filtering systems are utilized in various applications, recommender systems, which are implemented in commercial and non-profit web sites to predict the user preferences, utilize such schemes. Collaborative filtering (CF) is the most popular information filtering method employed in recommender systems.

1.1. Collaborative Filtering

CF aims to help people choose right products while shopping online. CF phrase is firstly introduced by developers of one of the first recommender systems called Tapestry (Goldberg et al., 1992). Many customers choose various products to buy, books to read, music CDs to listen, foods to eat, and so on over the Internet. Hence, there are many online vendors and they employ CF techniques to enhance their customers' satisfaction. CF depends on users' preferences about various items and assumes that who agree in the past tend to agree in the future (Grcar, 2004). The main functions of CF algorithms include analyzing user data and extracting useful information for further predictions.

CF algorithms perform on a database including preferences of users about various products to generate recommendation requested by an existing or new user for a selected item. The database, which is generally large-scale, consists of n users' preferences on m items. The preferences can be collected either explicit indications given by users, or implicit clues from log-based data. Since online vendors have huge amount of items, it is not possible to have users' preferences for all m items; thus, the database is sparse.

There are two general classes of CF algorithms, which are memory or model-based schemes (Breese et al., 1998). The former operates on entire user-

item matrix and it is based on correlation between users. However, the latter utilizes the database for constructing an underlying model of user preferences and predictions are inferred from the model. These two different kinds of classes have their own advantages and disadvantages. Memory-based algorithms produce more accurate predictions than model-based ones. On the other hand, model-based algorithms' online performance is better than memory-based ones.

The first algorithms proposed for CF (Goldberg et al., 1992; Resnick et al., 1994; Shardanand and Maes, 1995) are in class of memory-based algorithms. In this kind of algorithms, similarity scores between users are calculated and *k* nearest neighbors (*k-nn*) are selected according to similarity values for any user requesting prediction. Predictions are generated by weighting *k* users' ratings proportionally to their similarity to the user. To compute correlation between users, there are various similarity metrics. One of them and the most popular one in CF processes is Pearson correlation coefficient (PCC) employed by Resnick et al. (1994). Besides correlation-based similarity, researchers also utilize vector cosine-based similarity (Sarwar et al., 2001). Researchers study on examining algorithmic framework of CF and they make suggestions about using right components in memory-based CF (Breese et al., 1998; Herlocker et al., 1999). Sarwar et al. (2001) introduce a different way to employ memory-based algorithms. They propose to use item-based CF algorithm in which user-item matrix is analyzed to identify relationships between items. Firstly, the correlation between items are determined and then relationships are employed when recommendations are generated.

Generally, ratings are discrete numbers from a pre-defined range; however, it is possible to utilize binary ratings in CF. Therefore, Miyahara and Pazzani (2002) develop an algorithm based on naïve Bayesian classifier (NBC) for generating recommendations in binary data. Jin et al. (2004) present an algorithm to automatically determine appropriate weights for different items for CF. Chandrashekhar and Bhasker (2007) introduce a new memory-based approach. Unlike existing memory-based CF approaches, their approach exploits the predictable portions of even some complex relationships between users while selecting the mentors for a user through the use of the novel notion of selective

predictability, which can be measured using the entropy measure. Su and Khoshgoftaar (2009) make a detailed survey about CF covering development in memory-based CF algorithms.

Model-based CF algorithms depend on training data and learning model. The algorithms are proposed to handle with shortcomings of memory-based algorithms. Data mining tasks such as classification, clustering, dimension reduction, and regression are employed for constructing models. Bayesian belief networks are also used in recommendation process of CF (Heckerman et al., 2001; Su and Khoshgoftaar, 2006). These networks are capable of handling missing value problems (Su and Khoshgoftaar, 2009). To collect similar users or items into the same group, researchers propose various clustering-based CF algorithms (Ungar and Foster, 1998; Chee et al., 2001; Roh et al., 2003; Xue et al., 2005). By clustering, they aim to improve scalability of recommender systems. To decrease dimension of available data, dimension reduction-based methods are proposed. Principle component analysis (PCA) and singular value decomposition (SVD) are examples of dimension reduction methods utilized in CF applications (Sarwar et al., 2000; Goldberg et al., 2001). Vucetic and Obradovic (2005) propose a regression-based method for CF. Their method computes similarities between users by building a linear model.

In addition to memory and model-based CF algorithms, researchers also propose hybrid methods that combine content-based filtering with CF (Pazzani, 1999). Content-based recommender systems make recommendations by analyzing the content of textual information. This kind of recommender systems gather require information from documents, URLs, web logs, item descriptions, profiles about users' tastes, and preferences (Pazzani, 1999). To join both recommender systems' advantages, researchers front to propose hybrid CF algorithms. Melville et al. (2002) introduce a content boosted CF algorithm to generate more accurate recommendations. Although combining content-based filtering with CF increases recommendation accuracy, these hybrid models have an increased complexity of implementation (Pazzani, 1999; Popescul et al., 2001; Burke, 2002). Besides content-based models, methods joining memory and model-based CF algorithms are also proposed. A hybrid probabilistic memory-based CF combines memory

and model-based techniques to reduce complexity with decent accuracy (Yu et al., 2004). Pennock et al. (2000) propose a hybrid algorithm to retain both methods advantages. Su et al. (2007) propose hybrid CF algorithms, sequential mixture CF and joint mixture CF, each combining advice from multiple experts for effective recommendation generation.

1.2. Challenges of Collaborative Filtering

Recommender systems take an important role in e-commerce. However, there is no completely perfect system that serves recommendation without any drawbacks. Although CF is the most popular and widely used information filtering method in e-commerce, it has some important challenges. To produce high quality recommendation, problems of CF must be overcome. Main challenges of CF are listed below:

Data Sparsity: CF systems operate on large-scale data due to having huge amount of items. Therefore, collected data for CF purposes is sparse and it causes several challenges. One of the challenges is called *cold start* problem, which generally occurs when a new user is inserted to the database. Since there is not enough information about the user, forming neighborhood step in recommendation process becomes hard. Also, it is not possible to produce referral for a new item due to lack of users having rating for that item. This is called *coverage* problem, where *coverage* shows the percentage of items that recommender system can provide predictions. Consequently, data sparsity problem is caused by having inadequate data and it must be overcome to produce accurate and dependable referrals (Ahn, 2008).

Individuals' Privacy: The proposed algorithms for filtering purposes need customers' preference data to produce recommendations. Hence, the individuals share their data with e-companies (servers), so there are several risks for individuals' privacy (O'Cranor, 2003). Such risks occur due to unauthorized use of the users' confidential data by vendors. Since data holders have their customers' preference data, they might attempt to get benefit by unsolicited marketing. Another risk is that users' profiles might be used in criminal cases. Customer preference data are valuable assets for companies, and when companies fall into a financially difficult situation, they might sell this asset to get rid of

bankruptcy. According to privacy survey (Westin, 1999), the privacy fundamentalists concern about any use of their data and they are generally unwilling to provide their data to web sites. The pragmatic people are also concerned about usage of their private data, but less than the fundamentalists. If privacy anxieties are not dispelled, users might hesitate to provide actual preference data. Besides central server-based CF applications, users have confidentiality concerns in peer-to-peer (P2P) network-based CF applications, too. Although users have full control on their private data in P2P networks, it still requires having a privacy protection mechanism while collaborating with other people. Consequently, to feel comfortable while utilizing recommendation services, privacy concerns of individuals must be alleviated. If users' anxieties are overcome, they send their actual preferences; thus, CF algorithms produce more accurate recommendations.

Privacy Concerns of Data Holders: In CF, not only individuals have concerns about their privacy, but also vendors might have the same concerns. In order to overcome problems caused by having inadequate data, data holders might need data belonging to another vendor. In the case of collaboration of companies, data holders might hesitate to work together due to privacy, financial, and legal concerns. Since collected data are valuable assets, revealing them might cause losing competitive edge. According to reports published by the Organization for Economic Co-operation and Development (OECD) (2000; 2005), exposing of customers' privacy is very serious issue, and the companies are obliged to protect the data. In order to enable collaboration of companies, their confidentiality must be protected. Hence, privacy is an important challenge for both individuals and e-companies.

Scalability: Since e-commerce is independent from space and it is easy to exhibit huge amount of product through the Internet, product scope of e-commerce grow enormously. In addition to products, millions of people are able to do shopping through e-companies. Hence, CF algorithms face with serious scalability problems. Complexity of CF algorithms is generally in the order of $O(nm)$ and it is not an easy task forming neighborhood of users online from such

huge data. Receiving predictions in a reasonable time for users is an inevitable requirement; thus, CF service providers must overcome scalability problems.

Other Challenges: Like sparsity, privacy, and scalability, CF systems suffer from *synonymy*, *gray sheep*, and *shilling attacks* (Su and Khoshgoftaar, 2009). Synonymy problem occurs when the same or similar items in different companies have different names. Sometimes, a user's or a group of users' opinions are not common with remaining users. This situation is called gray sheep and it is a challenge for service providers to form such users' neighborhood. In order to sabotage a recommender system, malicious users or companies, especially rival ones, employ shilling attacks. Their aim is breaking the success of recommendation system.

1.3. Solutions for Eliminating Challenges

Researchers study the aforementioned challenges of CF schemes because such schemes are widely used in recommender systems. They propose several solutions alleviating such weaknesses. In this section, the solutions are explained in the following.

Data Sparsity: To alleviate data sparsity problem, several approaches are suggested. Since data in CF have large dimensions, researchers propose methods for reducing dimension of the data. By reducing dimension, their goal is removing effects of insignificant users or items. Besides SVD and PCA (Billsus and Pazzani, 1998; Goldberg et al., 2001), latent semantic indexing is also employed as a reduction process (Hofmann, 2004). Another method for overcoming sparsity problem is proposed by Chen et al. (2009). In their method, orthogonal non-negative matrix tri-factorization is applied to CF to alleviate sparsity via matrix factorization. As mentioned previously, hybrid methods are presented to improve CF results. Ziegler et al. (2004) propose a hybrid method to exploit useless information to address the sparsity problem in CF. Melville et al. (2002) introduce a content-boosted CF algorithm to alleviate cold start problem.

Traditional similarity measures cannot compute correlation between users if data are sparse. To cope with this challenge, Ahn (2008) propose a new similarity metric, which alleviates cold start problem. In addition to Ahn's study, Bobadilla et al. (2011) also present a new similarity metric to measure similarity between

users. According to their results, the method improves prediction quality. Another method proposed by Kim et al. (2010) to deal with sparsity is deriving recommendations from user-created tags. With improvement of social networks, trust metric is introduced by researchers (Hwang and Chen, 2007; Massa and Avesani, 2007; Lathia et al., 2008; Walter et al., 2009). It is investigated that there is a relationship between similarity and trust; and shown that trust can be used instead of similarity. According to researchers' results, trust-based CF handles problems caused by sparse data.

Individuals' Privacy: The notion of privacy-preserving CF (PPCF) is firstly introduced by Canny (2002a; 200b). To achieve privacy, Canny proposes to use some cryptographic approaches. Polat and Du (2003; 2005) employ randomized perturbation techniques (RPT) to achieve PPCF. In their schemes, users perturb their data by adding randomly created numbers to their numerical ratings. Since the users perturb their data, data collectors cannot learn the original ratings. Polat and Du (2006) utilize randomized response techniques (RRT) to perturb users' data while still producing binary ratings-based referrals with decent accuracy. The users either send their true ratings or the exact opposite of their ratings with a probability. In another study, Polat and Du (2007) propose a PPCF scheme based on inconsistently masked data. Each user variably disguises their private data using different methods. Their scheme is still able to offer predictions from inconsistently disguised data. Zhang et al. (2006) introduce a two-way communication privacy-preserving scheme for CF in which users perturb their ratings for each item based on the server's guidance instead of using an item-invariant perturbation. Parameswaran and Blough (2007) propose a framework for obfuscating sensitive information in such a way that it protects individual secrecy and also preserves the information content required for CF. Kaleli and Polat (2007b) propose a method for producing private referrals using NBC-based CF. They propose to use RRT for preserving users' confidentiality. According to their empirical results, the method is able to produce predictions with decent accuracy. Yakut and Polat (2007) investigate how to achieve predictions using Eigentaste without greatly exposing users' privacy. Ahmad and Khokhar (2007) propose an architecture, which attempts to restore user trust in these services by introducing

the notion of distributed trust. This essentially means that instead of trusting a single server, a coalition of servers is trusted. Distributions of trust make the proposed architecture fault resilient and robust against security attacks. Aïmeur et al. (2008) introduce a recommender system called Alambic, which is a hybrid recommender system combining content-based, demographic, and CF techniques and preserves users' privacy. In another study, Shokri et al. (2009) suggest that individuals modify their preference vector by merging it with a profile of a similar user before sending it to server. Calandrino et al. (2011) increases robustness of CF according to profile inference attacks. Xie et al. (2007) work on efficiency problem of distributed predictions in P2P systems and propose a scheme, which is an efficient neighbor-location algorithm for distributed database predictions. Berkovsky et al. (2005; 2006; 2007) also present solutions for P2P recommendation systems in which users obfuscate their data and get private referrals. Lathia et al. (2007) offer a solution, which computes similarity between users by estimating the number of concordant, discordant, and tied pairs of ratings between two users on distributed data without exposing parties' privacy. Ahn and Amtriain (2010) introduce private and fully distributed CF settings. They develop Rich the Internet Application (RIA) based on linked data. In addition to P2P network solutions, authors propose private solutions for social networks. Dokoohaki et al. (2010) introduce a solution for producing private recommendations on a trust-aware social network. Their method preserves individuals' privacy in a social network. Machanavajjhal et al. (2011) present and quantify a trade-off between accuracy and privacy of any social recommendation algorithm that is based on any general utility function. Researchers (Li et al., 2011) study producing privacy-preserving recommendations for online social communities. In the study, an interest group-based privacy-preserving recommender system called Pistis is proposed. Pistis generate recommendations based on aggregated judgments of group members and local personalization. Erkin et al. (2011) present an efficient privacy-preserving recommender system for a social trust network.

Data Holders' Privacy: Researchers introduce private solutions while data are partitioned between two parties. Kaleli and Polat (2007a) propose privacy-

preserving schemes to produce NBC-based recommendations on partitioned data between two online vendors. Polat and Du (2008) present a solution, which enables two data owners produce binary ratings-based top-*N* recommendations on partitioned data without violating their privacy. Their methods help recommender systems offer top-*N* lists. Yakut and Polat (2010) introduce SVD-based referrals on partitioned data between two parties only while protecting data holders' privacy. Yakut and Polat (2012a; 2012b) study arbitrarily partitioned data-based recommendations while preserving data owners' privacy.

Besides partitioned data-based solutions, researchers study methods for producing recommendations from distributed data among multiple parties. Zhang and Chang (2006) introduce a method based on secure multi computation and random oracle to produce recommendations from distributed data. Zhan et al. (2008) solve confidentiality problem of data holders by employing scalar product protocol to produce recommendations from distributed data. In another work, Zhan et al. (2010) address how to avoid privacy disclosure in collaboration of recommender systems on horizontally distributed data (HDD). In this study, major cryptology approaches are compared and an efficient privacy-preserving method based on the scalar product protocol is proposed. Hsieh et al. (2008) introduce an ElGamal homomorphic encryption (HE)-based method to merge recommender system databases. Weighted slope one predictor for item-based CF is proposed by Basu et al. (2011a; 2011b). They propose solutions for both vertical and horizontal distributed data among multiple parties. Basu et al. (2012) suggest a solution for collaboration of data holders on HDD within a cloud. The study presents a performance case-study on implementing the building blocks of a PPCF scheme in Java on the Google App Engine (GAE/J) cloud platform.

Scalability: Dimension reduction-based solutions, proposed to solve data sparsity problem, also overcome scalability problem. In addition to the solutions presented previously, similar methods are proposed to overcome scalability. To enhance online performance of memory-based CF algorithms, Sarwar et al. (2001) propose an item-based CF algorithm. Clustering is also among the methods that are applied to CF to improve online efficiency. Data analyzers can explore large amounts of data in order to discover useful information using clustering (Berry

and Linoff, 2000). Hence, researchers offer predictions online based on the clustered data. Ungar and Foster (1998) introduce clustering methods for CF. O'Connor and Herlocker (2001) show that items can be clustered instead of users in CF to improve performance. Roh et al. (2003) propose a method for producing self-organizing map (SOM) clustering-based referrals. Their method improves accuracy and efficiency of memory-based recommendation algorithm. Xue et al. (2005) employ smoothing-based clustering to increase online performance of recommendation process. In another study, discrete wavelet transform (DWT) is utilized to reduce dimension of data to enhance scalability of memory-based CF (Russell and Yoon, 2008).

In order to improve scalability of PPCF, researchers introduce solutions. The limitation of privacy-preserving NBC-based scheme is significantly improved by studies employing preprocessing techniques (Kaleli and Polat, 2009; Bilge and Polat, 2010). Also, Bilge and Polat (2011) enhance the central server-based PPCF schemes utilizing preprocessing of profiling users. They aim to overcome the sparsity and scalability of the proposed schemes by profiling user data. In another study, the same authors enhance both accuracy and online performance of PPCF by reducing data with DWT (Bilge and Polat, 2012).

1.4. Privacy-Preserving Distributed Data Mining

Privacy of data holders is not a challenge in CF applications only. Since there are lots of different data mining tasks, privacy-preserving distributed data-based functionalities have been also receiving increasing attention. Researchers introduce private methods for carrying out several data mining tasks on distributed data while preserving data holders' privacy. In this section, brief literature review about privacy-preserving distributed data mining (PPDDM) is given.

Clifton et al. (2002) suggest a toolkit of components that can be combined for specific PPDDM applications. The authors present some components as a toolkit, and show how they can be used to solve several PPDDM problems. Vaidya and Clifton (2004) address the classification problem while data are vertically distributed; and they introduce a private NBC algorithm. Yang and Wright (2006) provide an efficient privacy-preserving protocol to learn Bayesian

networks from vertically partitioned data between two parties. They present a privacy-preserving protocol to construct a Bayesian network efficiently from two parties' joint data. In another study, Xiong et al. (2007) present a framework for mining horizontally partitioned databases using a privacy-preserving *k-nn* classifier. To classify distributed data among multiple parties with *k-nn* algorithm, Zhang et al. (2009) introduce a private algorithm. Gambs et al. (2007) propose two algorithms for constructing a boosting classifier on distributed data with privacy. Dung et al. (2010) develop a cryptographic solution for classification rules learning methods for vertically and horizontally distributed data.

To accomplish clustering task over distributed data, Vaidya and Clifton (2003) study on a private *k*-means clustering for vertically partitioned data. Lin et al. (2005) propose a technique employing EM mixture technique for clustering distributed data. To enhance accuracy of clustering result while preserving parties' privacy, Jagannathan et al. (2006) introduce an algorithm for *k*-means clustering method for distributed data. This algorithm works on distributed data and guarantees privacy concerns. Kaya et al. (2007) propose a privacy-preserving distributed clustering protocol for HDD based on a very efficient homomorphic additive secret sharing scheme. Inan et al. (2007) propose private methods for constructing the dissimilarity matrix of objects over HDD. They propose to utilize this matrix for clustering, database joins, and record linkage. Lodi et al. (2010) introduce a private clustering method working on P2P networks. To achieve privacy in distributed clustering, Hajian and Azgomi (2011) use Haar wavelet transforms and scaling data perturbation to achieve privacy.

In order to accomplish other data mining tasks in such distributed environments, researchers propose various solutions. Kantarcioglu and Clifton (2004) discuss privacy-preserving association rules over HDD. To determine global association rules on distributed data privately, Shen et al. (2006) propose a data distortion-based method. In another work, Wang et al. (2010) introduce a private framework to find collaborative recommendation association rules from horizontally partitioned data. Zhong (2007) addresses the way of finding frequent item sets in distributed data among multiple parties without revealing data holders' private data. Emekci et al. (2007) enables different kinds of data owners

to perform decision tree learning algorithm ID3 on their distributed data without disclosing their private data. In another study, Vaidya et al. (2008) propose a technique for ID3 algorithm over VDD with privacy. Yi et al. (2006) proposes a trust-based privacy-preserving method for data sharing in P2P systems utilizing trust relations between peers in P2P system. Liu et al. (2006) suggest utilizing random projection (RP) in distributed data mining applications. They explore to use multiplicative RP matrices to preserve data holders' privacy. RP is used to compute the correlation matrix of privacy sensitive data, where data owners are not trusted by utilizing statistical data (Kargupta et al., 2003). Vaidya and Clifton (2009) propose a method to compute score of vertically partitioned elements. Their method requires revealing only little extra information. Since there can be malicious users in PPDDM schemes, Duan and Canny (2009) propose solutions for dealing with those malicious users. To handle outliers in distributed data, Dung and Bao (2011) study a private solution.

1.5. Problem Definition

Recommender systems collect users' preferences about various entities; and create an $n \times m$ user-item matrix **D**, where n and m represent number of users and items, respectively.

Data collected for CF purposes might be distributed vertically or horizontally among z companies. In vertically distributed data (VDD), each party has the same n users' ratings for disjoint sets of m_g items, where $g = 1, 2, \ldots, z$. While **D** has the size $n \times (m_1 + m_2 + \ldots + m_z)$, the distributed user-item matrix $\mathbf{D_g}$ has the size $n \times m_g$, which is held by each party g. Note that z is a small constant and $z << m$. More formally, **D** can be written in terms of $\mathbf{D_g}$ matrices, as follows: $\mathbf{D} = [\mathbf{D_1}\ \mathbf{D_2} \ldots \mathbf{D_z}]$. VDD is schematized in Figure 1.1.

In HDD, each company holds disjoint sets of n_g users for the same m items, where $g = 1, 2, \ldots, z$. Thus, the integrated data has the size $(n_1 + n_2 + \ldots + n_z) \times m$, while split data held by each vendor g, has the size $n_g \times m$. Notice again that z is a small constant and $z << n$. **D** can be written in terms of $\mathbf{D_g}$ matrices, as follows: $\mathbf{D} = \begin{bmatrix} \mathbf{D_1} \\ \mathbf{D_2} \\ \vdots \\ \mathbf{D_z} \end{bmatrix}$. HDD scheme is displayed in Figure 1.2.

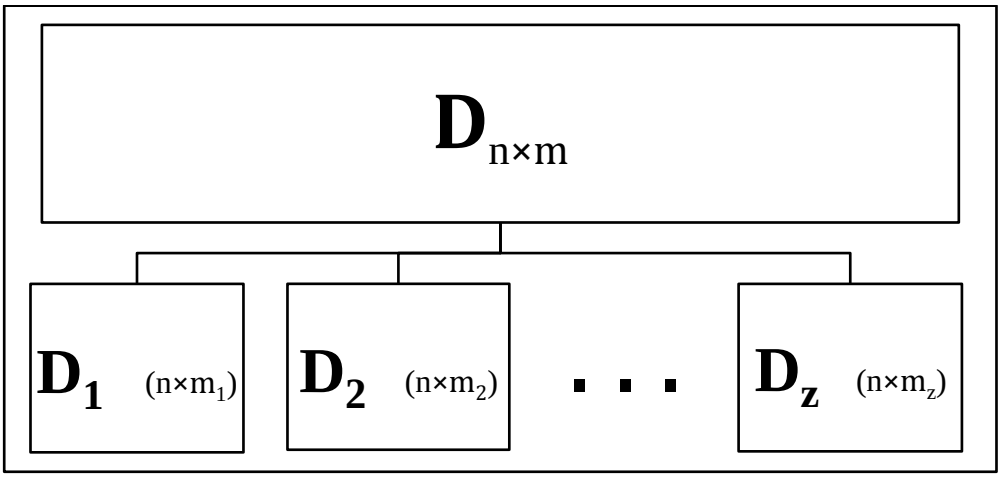

Figure 1.1. Basic Scheme of VDD

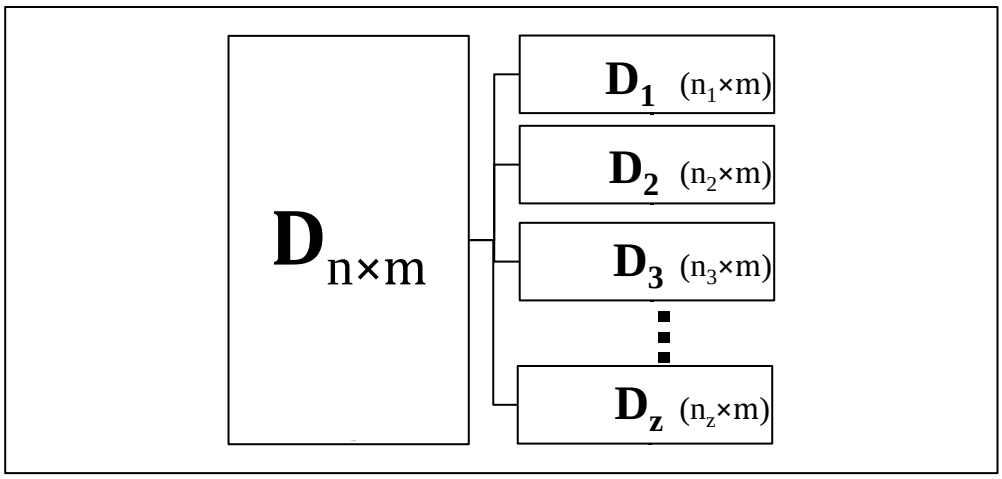

Figure 1.2. Basic Scheme of HDD

Those online vendors holding distributed data or inadequate data might want to collaborate to alleviate challenges (cold start, sparsity, lower quality recommendations) caused by insufficient data. However, due to privacy, legal, and financial concerns, e-companies might hesitate to work in partnership. Since users' ratings about different entities are considered the companies' confidential data, privacy concerns about revealing these data to other companies make data owners uncomfortable about partnership. Besides privacy concerns, e-companies hesitate to collaborate because of financial fears. If the vendors reveal private data, such data can be utilized by rival companies to profile customers and used to direct future sales campaigns. Thus, collected data are valuable assets; revealing

them might cause losing competitive edge. In addition to financial and confidentiality concerns, collaboration without protecting private data might cause legal problems. According to reports published by OECD (2000; 2005), e-companies are obligate to preserve their customers' privacy. Consequently, the main problem is *how do those corporations holding insufficient data, due to distributed data (vertically or horizontally); provide referrals (binary and numerical) on their integrated data with a decent performance without disclosing their confidential data to each other*?

In this dissertation, privacy of parties is defined, as follows: *The parties should not be able to learn the true rating values and the rated and/or unrated items held by each other while providing recommendations based on their distributed data*. In the suggested methods, true ratings are considered confidential because they can be used to profile customers and they represent users' opinion about commercial goods. Moreover, rated entities (accordingly unrated products) are also regarded as private because special offers and discounts can be made for those unsold merchandises. E-companies do not want to reveal such data to others while conducting collaborative works. Hence, in distributed data-based recommendation processes, the ratings and the rated items held by each retailer are considered private. Unlike them, entity IDs, like user and item IDs, can be viewed as public. The company, which returns the predictions to *a*, is called the master company (MC). Since a customer sends her corresponding data to related parties, her data also become confidential for such companies, as well. Thus, they are obliged to estimate predictions collaboratively without divulging the customer's data to each other.

1.6. Utilized Collaborative Filtering Algorithms

In this dissertation, two widely used CF algorithms are studied to provide predictions on distributed data with privacy. One of them is used to provide predictions using numeric data while the other is used to produce binary ratings-based recommendations.

The *k-nn*-based CF algorithm proposed by Herlocker et al. (1999) is utilized for producing referrals based on numeric ratings. In this algorithm, when an active user (*a*) asks a prediction for a target item (*q*), referred to as p_{aq}, she sends her

known rating vector (**A**) and a query (q). The system computes the correlations between a and each user in the database. It then forms the neighborhood for a. To form neighborhood, the algorithm use two different approaches. First one is selecting the most similar k users as neighbors. If algorithm utilizes the second approach, users having a similarity score which is bigger than a predefined threshold value (τ) are included a's neighborhood. Finally, p_{aq} is estimated on her neighbors' data using a recommendation algorithm. The basic scheme is described in Figure 1.3.

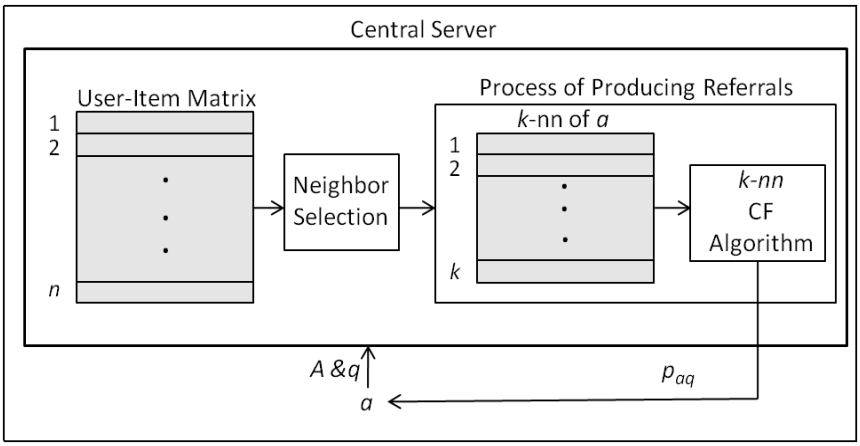

Figure 1.3. Basic Scheme of *k-nn* Algorithm

Correlations between various entities can be estimated using the PCC, as follows:

$$w_{au} = \frac{\sum_{j \in J}(v_{aj} - \overline{v_a})(v_{uj} - \overline{v_u})}{\sigma_a \, \sigma_u} \tag{1.1}$$

where v_{aj} and v_{uj} are the ratings of a and user u for item j, J is the set of commonly rated items by both users, and $\overline{v_a}$ and $\overline{v_u}$ are the a's and user u's mean votes. Besides finding similarities with PCC, it is also possible to use adjusted cosine measure (Sarwar et al., 2001) in recommendation process. By employing adjusted cosine, correlation between users is computed, as follows:

$$w_{au} = \frac{(v_{aj} - \overline{v_J}) \cdot (v_{uj} - \overline{v_J})}{\sqrt{\sum_{j \in J}(v_{aj} - \overline{v_J})^2} \sqrt{\sum_{j \in J}(v_{uj} - \overline{v_J})^2}} \tag{1.2}$$

where $\cdot$ is the scalar dot product. In this dissertation, both similarity measures are employed. After estimating the similarities, the best k users are selected as nearest neighbors. p_{aq} finally can be calculated, as follows:

$$p_{aq} = \overline{v_a} + \frac{\sum_{u \in NN}(v_{uq} - \overline{v_u})w_{au}}{\sum_{u \in NN} w_{au}} \quad (1.3)$$

where *NN* is the set of *a*'s neighbors who rated *q*.

Miyahara and Pazzani (2002) propose an approach for CF based on NBC, where they define two classes, *like* and *dislike*. Since customers vote items as *like* (1) or *dislike* (0), the sparse user ratings matrix includes Boolean values indicating whether the user rated items as 1 or 0. In the user ratings matrix, other users correspond to features and the matrix entries correspond to feature values. The naïve assumption states that features are independent, given the class label. Thus, the probability of an item belonging to $class_j$ (c_j), where *j* is *like* or *dislike*, given its *y* feature values, can be written, as follows:∝

$$p(c_j|f_1, f_2, \ldots, f_y) \propto p(c_j) \prod_i^y p(f_i|c_j) \quad (1.4)$$

where ∝ shows proportionality and both $p(c_j)$ and $(f_i|c_j)$ can be estimated from training data and f_i corresponds to the feature value of *q* for user *i*. To assign *q* to a class, the probability of each class is computed, and the example is assigned to the class with the highest probability.

1.7. Utilized Privacy-Preserving Techniques

In this section, privacy-preserving measures that are utilized to achieve privacy are explained.

Randomization Techniques: Randomized methods have been widely used in PPCF schemes for data masking. Randomization-based approaches are useful when aggregate data are interested in. There are three randomization schemes applied to CF algorithms for preserving privacy, as follows:

i. *Randomized Perturbation Techniques* (*RPT*): RPT method attempts to preserve privacy by modifying data items with a randomization process (Agrawal and Srikant, 2000). RPT perturbs a specific value by adding a random number. If the value *x* is wanted to be hidden, then to perturb *x* with RPT, a random number *r* is added to it and new perturbed value is used instead of *x*. Adding random numbers, which are drawn from over a specified range and distribution like uniform or Gaussian, perturbs the original data. In place of utilizing actual preferences of users, perturbed

data values are employed in CF process. Such techniques are suitable for numeric ratings-based schemes and widely used in PPCF.

ii. *Randomized Response Techniques* (*RRT*): RRT is first introduced by Warner (1965) as a technique to estimate the percentage of people in a population that has a specific attribute. The interviewer asks each respondent two related questions, the answers to which are opposite to each other. Respondents are able to choose one of the questions to answer using a randomizing device. With probability θ, respondents choose the first question while with probability 1- θ; they select the second question to answer. Although the interviewer learns the responses, she does not know which question was answered. RRT can be used to mask users' binary ratings. Ratings vectors usually contain binary ratings for rated items and blank cells for unrated items. An example of ratings vector for a user u is V_u = (1 1 0 – 0 1 – 0 0 1 1), where "–" means not rated. To mask V_u using RRT, u uniformly randomly chooses a random number over the range [0, 1]. If it is less than or equal to θ, then user u sends V_u. Otherwise, the user sends the false data. In other words, she sends the exact opposite of the ratings, which is called V_u', where V_u' = (0 0 1 – 1 0 – 1 1 0 0). With probability θ, true data is sent while false data is sent with probability 1- θ. Since users only know random numbers, the data collector does not know whether the received data is true or not. Users might decide to put their ratings into multiple groups; and mask ratings in each group independently to increase privacy level.

iii. *Random Filling* (*RF*): In addition to perturbing real ratings, it is also crucial to mask rated and/or unrated item cells. To do so, PPCF schemes also utilize RF in which some uniformly randomly chosen cells are filled with some default votes or noise data.

***Homomorphic Encryption* (*HE*)**: HE is a form of encryption, where a specific algebraic operation performed on the plaintext is equivalent to another algebraic operation performed on the cipher-text. Depending on one's viewpoint, this can be seen as either a positive or negative attribute of the cryptosystem. HE schemes are malleable by design. The homomorphic property of various

cryptosystems can be used to create secure voting systems, collision-resistant hash functions, and private information retrieval schemes and enable widespread use of cloud computing by ensuring the confidentiality of processed data. In this dissertation, an efficient HE scheme proposed by Paillier (1999) for protecting data owners' confidential data is utilized. Suppose that ξ is an encryption function and K is a public key, there are two plain texts x_1 and x_2, which are desired to be encrypted. Paillier's HE scheme provides performing addition or multiplication operations on encrypted data values, as follows: $\xi_K(x_1) \times \xi_K(x_2) = \xi_K(x_1 + x_2)$ and $x_1 \times \xi_K(x_2) = \xi_K(x_1 \times x_2)$.

Private Neighborhood Formation Algorithm **(PNFA)**: In order to sort similar users according to their weights without divulging the weights, a sorting algorithm is proposed. Multiple companies can sort users while sharing some virtual weights rather than actual weights. Although collaborating companies are able to estimate partial similarity weights, they do not want to share them to find the sorted list of them. PNFA allows them to sort the similar users without exposing true weights.

1.8. Contributions

There are limited numbers of studies for performing CF on partitioned data between two parties only. However, data might be distributed among multiple parties. In this dissertation, various schemes are proposed to provide distributed data-based CF services with privacy. To achieve confidentiality, techniques like RPT, RRT, RF, HE, PNFA, and so on are employed. Privacy analyses show that the proposed schemes are secure and they do not violate privacy requirements. In order to improve online performance, that might get worse due to distributed data-based CF with privacy, clustering, data reduction, and trust methods are utilized. It is shown that such approaches improve online performance. To test accuracy of the proposed schemes, real data-based experiments are performed. Empirical analyses show that it is still possible to produce recommendations with a decent accuracy from distributed data without jeopardizing data owners' privacy. Also, additional costs caused by collaboration and privacy-preserving measures are analyzed in the suggested schemes. The analyses show that additional costs are negligible and the companies are still able to produce referrals efficiently.

In order to improve online performance, SOM clustering is employed. This solution provides producing SOM-based recommendations from distributed data while considering data holders' privacy. Protocols for clustering VDD or HDD privately with SOM are introduced (Kaleli and Polat, 2012a; 2012b). Also methods for constructing an off-line recommendation model for parties are provided and protocols for utilizing the off-line model to produce online recommendations are explained.

It is shown that RP can be utilized in order to improve online performance of CF systems. RP is also a very powerful tool to enhance privacy. Thus, it is investigated that RP can be carried out to preserve confidentiality of parties while they collaborate. In this solution, a protocol which is used for sorting distributed weights privately in order to obtain *k-nn* of any user in CF process is introduced. Finally, online protocols for producing RP-based recommendations are provided.

In this dissertation, it is shown that a trust network can be constructed from HDD or VDD without revealing collaborating parties' privacy. The trust network is created off-line; hence, performance problems caused by online neighborhood forming step of CF schemes can be handled. In order to relieve privacy concerns, secure protocols for multi-party schemes are proposed (Kaleli and Polat, 2011).

In the case of binary ratings, protocols enabling data holders to employ NBC-based CF algorithm on their distributed data privately are introduced. P2P-based scheme is a special kind of HDD in which $n = z$. In this dissertation, a solution for producing NBC-based recommendations for users in a P2P network without jeopardizing their privacy is studied (Kaleli and Polat, 2010).

1.9. Organization of the Dissertation

In Chapter 2, producing SOM-based recommendations from distributed data is studied. In the following chapter, distributed CF solutions based on RP-based dimensionality reduction are introduced. In Chapter 4, trust-based distributed CF schemes are described. To produce binary recommendations from distributed data, privacy-preserving solutions are introduced in Chapter 5. In Chapter 6, a special kind of HDD is studied and P2P CF based on binary ratings with privacy is explored. Finally, in Chapter 7, conclusions are presented and future research directions are explained.

2. PRIVACY-PRESERVING SOM-BASED RECOMMENDATIONS ON DISTRIBUTED DATA

In this chapter, providing SOM clustering-based predictions on distributed data without deeply jeopardizing their confidentiality is investigated. Users are first grouped into various clusters off-line using SOM clustering while preserving the collaborating parties' privacy. With privacy concerns, predictions are produced based on distributed data using *k-nn* CF algorithm. The proposed privacy-preserving schemes are analyzed in terms of privacy and supplementary costs. The analyses show that the methods offer recommendations without greatly exposing data holders' privacy and cause negligible superfluous costs caused by privacy concerns. To evaluate the schemes in terms of accuracy, real data-based experiments are performed. Empirical results demonstrate that the schemes are still able to provide truthful predictions.

2.1. Introduction

To enhance performance of CF systems, clustering methods can be employed. SOM clustering is one of such methods. Roh et al. (2003) propose to employ SOM clustering for improving *k-nn* CF algorithm's efficiency. SOM is an unsupervised competitive learning method used for clustering and data visualization. It works well on dividing an input data into closest clusters. According to the study in (Roh et al., 2003), SOM clustering approach alleviates online computational complexity and improves scalability of the recommendation process.

In this thesis, SOM-based methods for providing predictions on distributed data without greatly jeopardizing data holders' privacy are introduced. Firstly, users are grouped into various clusters using SOM clustering off-line. After determining *a*'s cluster, those users in that cluster are considered the best similar *k* users to *a*. Since off-line costs are not critical for the overall success of CF, the schemes perform some computations off-line. As explained before, in one hand, those companies holding HDD or VDD might want to collaborate to overcome the problems caused by scarce data. On the other hand, they do not want to work together due to privacy, legal, and financial concerns. The schemes make it possible for such data holders to provide predictions on integrated data without disclosing their confidential data to each other. Using the methods, the parties can overcome

coverage and accuracy problems through partnership. Additionally, since they do not reveal their private data to each other, they do not face with privacy, financial, and legal issues.

2.2. SOM Clustering and Collaborative Filtering

Roh et al. (2003) state that SOM-based CF scheme provides higher quality predictions than other comparative models. In addition to providing high quality recommendations, SOM has capability of clustering large-scale databases and handles with high dimensional data in recommender systems (Mao & Jain, 1996). SOM is introduced by Kohonen (1995). The SOM architecture consists of two fully connected layers: an input layer and a Kohonen layer. The steps of SOM clustering algorithm are described in the following (Gan et al., 2007):

i. Determine values of initial constants: η_0, σ_0, τ_1, and τ_2, as follows: $\eta_0 = 0.1$, $\sigma_0 = 3/2$, $\tau_1 = 1{,}000/\log \sigma_0$, and $\tau_2 = 1{,}000$, which are configured by Haykin (1999). Note that η is the learning rate, σ represents the radius of the lattice, and τ_1 and τ_2 are the time constants.

ii. Find the winning Kohonen layer neuron. In this step, a random object $\boldsymbol{x}$ is selected from input data $\mathbf{X}$ and the winning Kohonen Neuron (KN_i) is determined by the computed minimum Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{W_j}$ using Eq. (2.1), as follows. Notice that $\boldsymbol{W_j}$ represents initial weights chosen randomly among objects in $\mathbf{X}$ for $j = 1, 2, \ldots, H$, where H shows number of neurons in Kohonen layer and s shows an iteration:

$$KN_i^{(s)} = \min ||x^{(s)} - W_j^{(s)}|| \tag{2.1}$$

where ||x|| shows norm of x.

iii. Update the weight vectors of all neurons by using Eq. (2.2), as follows:

$$W_j^{(s+1)} = W_j^{(s)} + \eta(s) h_{ji}(s) (x - W_j^{(s)}) \tag{2.2}$$

where $h_{ji}(s)$ is the neighborhood function. $\eta(s)$ and $h_{ji}(s)$ are computed using Eq. (2.3) and Eq. (2.4), as follows:

$$\eta(s) = \eta_0 \exp(-s/\tau_2),\ s = 0, 1, 2, \ldots, \tag{2.3}$$

$$h_{ji}(s) = \exp\left(-\frac{d_{i,j}^2}{2\sigma^2(s)}\right) \text{ and } \sigma(s) = \sigma_0(-s/\tau_1) \tag{2.4}$$

iv. Repeat from step 2 until there is no noticeable change in the future maps.

In SOM-based CF scheme, as shown in Figure 2.1, users are clustered off-line, where *nc* shows number of clusters. To obtain p_{aq}, *a* sends **A** and *q* to the server that first computes the distances between **A** and each cluster center using Euclidean distance measure. It then assigns *a* to the closest cluster. Those users in that cluster are considered *a*'s neighbors. After estimating p_{aq}, it finally sends it to *a*.

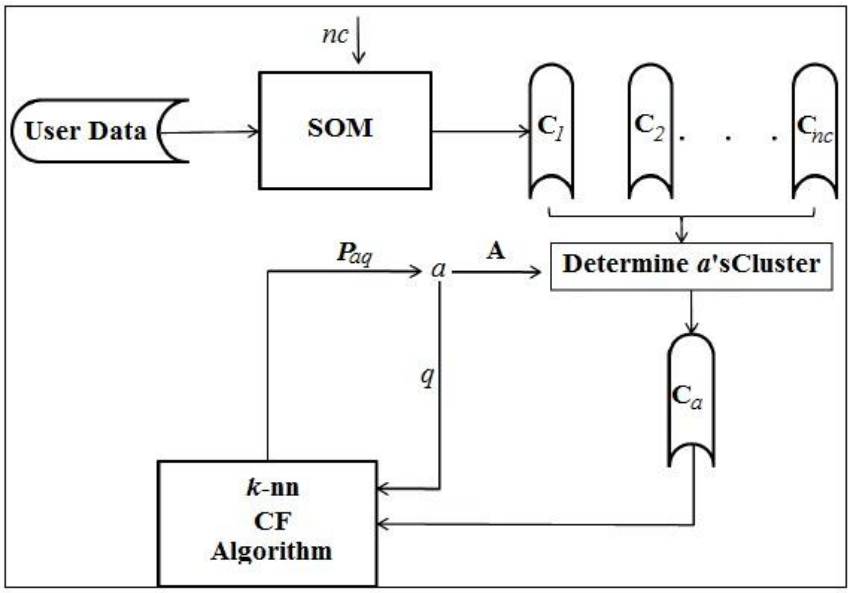

Figure 2.1. SOM-based Collaborative Filtering Scheme

2.3. Private SOM-based Recommendations on VDD

Since *a* sends her data to the MC holding *q*, the MC is responsible for protecting *a*'s data, as well. To provide predictions on VDD without deeply violating data owners' privacy, various protocols are proposed, as follows:

SOMP: Collaborating companies first need to cluster data using SOM clustering off-line. The protocol, called private SOM clustering on multi-party vertically distributed data (SOMP), enables them to cluster distributed data without exposing their privacy, as follows:

i. The parties decide *nc* and decide the initial constant values.

ii. One of the parties then uniformly randomly chooses *nc* users whose data form initial weight (W_j) values and a user as *o*, among the users in its database; and informs the others. Due to VDD, each party knows the

values of the initial weights $\boldsymbol{W_j}$s and *o* it holds and does not know the values that the other parties hold.

iii. To determine KN_i, the parties need to estimate distances between *o* and each $\boldsymbol{W_j}$ using the Euclidean distance measure. Since data are vertically distributed, each company $g = 1, 2, \ldots, z$ can compute $\|o_g - W_{gj}\|$ values using the data items it holds for all $\boldsymbol{W_j}$s. They then exchange such aggregate values. After calculating the sums of such aggregate values, they take their square roots and find the distances. Each party then determines the KN_i. Note that distance between any two vectors, $\boldsymbol{X}$ and $\boldsymbol{Y}$, whose elements are vertically distributed between *z* companies can be estimated, as follows using the Euclidean distance measure:

$$\|X - Y\| = \sqrt{\sum_{j=1}^{n_1}(x_j - y_j)^2 + \sum_{j=n_1+1}^{n_2}(x_j - y_j)^2 + \cdots + \sum_{j=n_{z-1}+1}^{n_z}(x_j - y_j)^2}$$

where n_g values represent the average number of elements held by company *g*. As seen from the above equation, due to VDD, the parties compute partial sums and exchange them.

iv. Using Eq. (2.2), each party updates the corresponding parts of the $\boldsymbol{W_j}$s without sharing any information with other parties due to VDD.

v. They repeat steps 3 and 4 until no noticeable change in the future map. Each party ends up with the corresponding parts of the cluster centers and each user's cluster.

vi. The parties finally exchange such distributed centers with each other so that each company now has the whole cluster centers.

DPP: Data holders need to compute item and user mean votes and vector lengths. In such computations, the parties can derive confidential data. In order to prevent them from gaining private data, the vendors perturb their data. On one hand, such data masking should hide confidential data. On the other hand, it still allows generating correct outcomes. Data perturbation protocol (DPP) can be used by each party *g* to mask its data set ($\mathbf{D_g}$) containing users' preferences about various items, as follows:

i. Uniformly randomly chooses a random value, θ_g, over the range (0, 100].

ii. Selects a random value, β_g, over the range (0, θ_g] uniformly randomly.

iii. Uniformly randomly picks β_g percent of the unrated cells.

iv. Finally, fills such chosen cells with fake ratings and obtains $\mathbf{D'_g}$.

Data collected for CF purposes are usually sparse. Using some methods (average, the most probable value, and so on), which handle missing values, the parties fill some of the empty cells of their databases. The DPP includes using fake ratings as part of data disguising process. Either the parties can use non-personalized votes (user, item, or overall mean votes) or personalized ratings estimated using a CF algorithm. It is proposed to use personalized votes to fill blank cells because personalized ratings are more likely represent users' preferences than non-personalized ratings, where such ratings can be computed using the algorithm proposed by Herlocker et al. (1999). Each company can determine personalized or even non-personalized ratings using the available ratings it holds without the help of other companies. Thus, each vendor can estimate them off-line without revealing any information to others and they can mask their data using the DPP off-line.

UMRP: In SOM-based CF on VDD, *k-nn*-based CF algorithm with adjusted cosine similarity measure is employed. As seen from Eq. (1.2) and Eq. (1.3), the parties need to compute item and user mean ratings for normalizing votes. Like personalized ratings estimation, each party can find the item mean votes of those items it holds without the help of other parties because it knows the required data. In other words, due to VDD, each party has the ratings for any item it holds. Thus, each company can determine non-personalized item mean ratings for each item it holds off-line without disclosing data to other parties. However, the parties need to collaborate with each other to estimate user mean ratings due to the nature of data distribution. For a user *u*, her average rating can be computed, as follows:

$$\overline{v_u} = \frac{\sum_{j \in J} v_{uj}}{|J|} \tag{2.5}$$

where *J* shows the set of ratings *u* has, v_{uj} represents *u*'s rating for item *j*, and |*J*| shows the number of ratings *u* has. As seen from Eq. (2.5), $\overline{v_u}$ can be calculated in a distributive manner using distributive measures like sum and count. Since data are vertically distributed among *z* parties, Eq. (2.5) can be written, as follows:

$$\overline{v_u} = \frac{\sum_{j \in J_1} v_{uj} + \sum_{j \in J_2} v_{uj} + \cdots + \sum_{j \in J_z} v_{uj}}{|J_1| + |J_2| + \cdots + |J_z|} \quad (2.6)$$

where J_g shows the set of user u's ratings held by the party g. As seen from Eq. (2.6), in order to find $\overline{v_u}$ values for n users, data owners need to exchange partial sums like $\sum_{j \in J_g} v_{uj}$ and $|J_g|$ values. The vendors can estimate user mean votes off-line using the following protocol, called user mean ratings protocol (UMRP), as follows, without deeply jeopardizing their privacy:

i. They first perturb their data sets using the DPP, which elaborated previously.

ii. They then compute interim aggregate results, sum and count, ($\sum_{j \in J_g} v_{uj}$ and $|J_g|$) for all n users.

iii. Next, they exchange such partial aggregates with other parties.

iv. After receiving the required data from other parties, they finally compute mean votes for all n users using Eq. (2.6).

VLP: As seen from Eq. (1.2), the parties need user ratings vector lengths in order to estimate similarity weights. For any user u, her ratings vector length ($\|X_u\|$) can be computed, as follows:

$$\|X_u\| = \sqrt{\sum_{j \in J} v_{uj}^2} \quad (2.7)$$

As seen from Eq. (2.7), vector length can be calculated in a distributive manner using distributive measure sum. Since data are vertically distributed among z parties and ratings are normalized, Eq. (2.7) can be written, as follows:

$$\|X_u\| = \sqrt{\sum_{j \in J_1} (v_{uj} - \overline{v_j})^2 + \sum_{j \in J_2} (v_{uj} - \overline{v_j})^2 + \cdots + \sum_{j \in J_z} (v_{uj} - \overline{v_j})^2} \quad (2.8)$$

As seen from Eq. (2.8), each party g needs to compute interim aggregate sum values, $\sum_{j \in J_i} (v_{uj} - \overline{v_j})^2$, for all n users. The companies can estimate vector lengths off-line using the following protocol, called vector lengths protocol (VLP), while preserving their privacy:

i. They first mask their data sets using the DPP, as described before.

ii. Then, they calculate item mean ratings and normalize their ratings by subtracting item mean votes.

iii. Next, they compute interim aggregate results, $\sqrt{\sum_{j \in J_g}(v_{uj} - \overline{v_j})^2}$ values, for all *n* users.

iv. After that they exchange them with other parties.

v. After receiving the necessary data from others, they finally compute vector lengths for all *n* users using Eq. (2.8).

The protocols described so far are used during off-line phase. In other words, data holders cluster their distributed data, mask their data, estimate personalized and non-personalized ratings, and compute vector lengths off-line using the proposed protocols. Note that the MC collaboratively estimates p_{aq} based on *a*'s neighbors' data using Eq. (1.3), which can be written, as follows:

$$p_{aq} = \overline{v_a} + \frac{\sum_{u \in NN} w_{au} \times v'_{uq}}{\sum_{u \in NN} w_{au}} = \overline{v_a} + P_{aq} \tag{2.9}$$

Similarly, Eq. (1.3) can be written, as follows because the vendors know item mean ratings and vector lengths and they can normalize their ratings:

$$w_{au} = \sum_{j \in J} v''_{aj} \times v''_{uj} \tag{2.10}$$

where v''_{aj} and v''_{uj} represent the normalized ratings of user *a* and *u*, respectively. Notice that the ratings can be normalized by first subtracting corresponding item mean vote and dividing the result by user ratings vector length. P_{aq} in Eq. (2.9) can be written, as follows:

$$P_{aq} = \frac{\sum_{u \in NN} \left[\sum_{j \in J} v''_{aj} \times v''_{uj}\right] \times v'_{uq}}{\sum_{u \in NN} \sum_{j \in J} v''_{aj} \times v''_{uj}} = \frac{\sum_{j \in J} v''_{aj} \left[\sum_{u \in NN} v''_{uj} \times v'_{uq}\right]}{\sum_{j \in J} v''_{aj} \left[\sum_{u \in NN} v''_{uj}\right]} \tag{2.11}$$

Since data are vertically distributed among *z* parties, Eq. (2.11) can be written, as follows:

$$P_{aq} = \sum_{g=1}^{z} \frac{\sum_{j \in Jg} v''_{aj} \left[\sum_{u \in NN} v''_{uj} \times v'_{uq}\right]}{\sum_{j \in Jg} v''_{aj} \left[\sum_{u \in NN} v''_{uj}\right]}. \tag{2.12}$$

As seen from Eq. (2.12), the parties can compute $P_N = \left[\sum_{u \in NN} v''_{uj} \times v'_{uq}\right]$ and $P_D = \left[\sum_{u \in NN} v''_{uj}\right]$ values for all target items off-line in order to enhance online performance. This is like constructing a model off-line to improve online efficiency. Notice that *q* can be any of *m* items. Also note that items are distributed among *z* parties due to vertical distribution. The following protocols called P_N protocol (P_NP) and P_D protocol (P_DP) are proposed to be utilized off-line to construct a model.

P_NP: P_N values can be estimated for all target items off-line, as follows: For each party g = 1, 2, …, z, they do the followings assuming the g^{th} party is the MC. For each target item q = 1, 2, …, m_g (m_g is the number of items held by g):

i. The MC and the other parties mask their data sets using the DPP, as explained before.

ii. The MC then encrypts each value in the column vector of q using ξ with its K_{MC}.

iii. It then sends such encrypted values $\left(\xi_{K_{MC}}(v'_{uq})\right)$ to other parties.

iv. For all items j = 1, 2, …, m_g held by each collaborating party g, each collaborating vendor computes $\xi_{K_{MC}}(v'_{uq}) \times v''_{uj}$ and obtains $\xi_{K_{MC}}(v'_{uq} \times v''_{uj})$ values using the HE property.

v. Each party then permutes $\xi_{K_{MC}}(v'_{uq} \times v''_{uj})$ values for each item they hold using a row permutation function (F_{gr}).

vi. Then, the collaborating vendors permute the items they hold using a column permutation function (F_{gc}).

vii. They then send the encrypted and the permuted (both row and column) results to the MC.

viii. Since the MC knows the K_{MC}, it can decrypt the encrypted results.

ix. For each item j, the MC finds column sums (the P_N values) from permuted ones. Note that the order does not matter to find them.

x. The MC then sends the permuted P_N values to the corresponding parties.

xi. Since each collaborating vendor knows the column permutation functions, they order the received P_N values and obtain the nominator part of the model.

P_DP: After estimating P_N values, the parties should compute P_D values, as well. They can compute P_D values for all target items, as follows: For each vendor g = 1, 2, …, z, they do the followings based on the mask data: For each target item q = 1, 2, …, m,

i. Calculate normalized ratings using the item mean votes and vector lengths determined previously and obtain v''_{uj} values for all j = 1, 2, …, m and u = 1, 2, … , n. Note that $v''_{uj} = \left(v_{uj} - \overline{v_j}\right) / \sqrt{\sum_{j \in J} v_{uj}^2}$.

ii. Determine only those rows containing ratings for q (*NN*). Such users' data are used to generate prediction for q. The parties know which rows have ratings for q due to the $P_N P$. However, they do not know actual ratings due to encryption and rated items due to filled cells.

iii. Find column sums, $P_D = \sum_{u \in NN} v''_{uj}$ values, for $j = 1, 2, \ldots, m$.

PRP: After the parties construct the model (estimating P_N and P_D values off-line), they can now start providing referrals online. After the MC receives **A** and q from a, it first assigns a to the closest cluster by estimating distances between **A** and each cluster center. Since all parties know cluster centers, the MC can easily compute such distances. Finally, the MC initiates the final recommendation estimation process and returns p_{aq} to a after collaborating with other parties. The proposed scheme, called private recommendation protocol (PRP), can be described, as follows:

i. The MC disguises a's ratings vector using the DPP like data owners do for masking their databases.

ii. It normalizes her data and obtains v''_{aj} values, where $v''_{aj} = \left(v_{aj} - \overline{v_j}\right) / \sqrt{\sum_{j \in J} v_{aj}^2}$.

iii. After encrypting each v''_{aj} value using ξ and K_{MC}, it sends the corresponding values to the collaborating companies.

iv. Each collaborating party similarly inserts item average normalized ratings into uniformly randomly selected empty cells of the received normalized data. In other words, they similarly perturb the received data.

v. They then encrypt such default values using ξ and K_{MC}.

vi. Each collaborating party g finds $\xi_{K_{MC}}(v''_{aj}) \times \left[\sum_{u \in NN} v''_{uj} \times v'_{uq}\right] = \xi_{K_{MC}}\left(v''_{aj} \times \left[\sum_{u \in NN} v''_{uj} \times v'_{uq}\right]\right)$ and $\xi_{K_{MC}}(v''_{aj}) \times \left[\sum_{u \in NN} v''_{uj}\right] = \xi_{K_{MC}}\left(v''_{aj} \times \left[\sum_{u \in NN} v''_{uj}\right]\right)$ for all j in J_a, which is the set of a's ratings including the fake ones.

vii. After computing such encrypted aggregated results for nominator and denominator, each collaborating vendor g permutes them using F_g and sends them to the MC.

viii. The MC decrypts them and finds the overall sums for nominator and denominator. It finally estimates p_{aq} and sends it to a.

2.4. Private SOM-based Recommendations on HDD

Data holders firstly cluster the HDD to construct a model for further recommendations while preserving privacy. To cluster the data, companies employ private distributed SOM clustering protocol (PDSOM).

PDSOM: The details of the protocol are given below:

i. Data owners decide nc and determine the sequence of active company (Initial Company-IC) in clustering operation.

ii. The first IC initializes $\boldsymbol{W_j}$ vectors for all $j = 1, 2, \ldots, nc$ by selecting random rating values and decides the constant parameters.

iii. The IC starts clustering operation by selecting a random user among its users. After finding winning neuron in Kohonen layer using Eq. (2.1), it updates $\boldsymbol{W_j}$ vectors using Eq. (2.2). It then increases s by one.

iv. It repeats step 2 until all users it holds are assigned to a cluster. It finally sends the updated $\boldsymbol{W_j}$ vectors and s to the second party in the sequence.

v. Since the updated value of s and $\boldsymbol{W_j}$ vectors are enough to continue clustering, the second party repeats step 2 as IC does. When all users it holds are assigned to a cluster, it sends new s and updated $\boldsymbol{W_j}$ vectors to the next party.

vi. After receiving s and $\boldsymbol{W_j}$ vectors, each party updates them like IC does. When all parties assign their users to a cluster, an epoch is completed. The last party then sends the updated $\boldsymbol{W_j}$ vectors to the IC.

vii. Steps 3 - 6 are repeated until no noticeable change in the future map.

The IC initially chooses $\boldsymbol{W_j}$ vectors and updates them s times. Since it sends the updated $\boldsymbol{W_j}$ vectors to the next party, that company cannot learn the true ratings and the rated and/or unrated items held by the IC. Although number of users held by the IC is known by the second company, it cannot derive any information about data held by the IC, because it does not know which users have rated which items and the distances between users and $\boldsymbol{W_j}$ vectors. Similar argument is also true for other companies. The parties only exchange the updated $\boldsymbol{W_j}$ vectors and the updated s values. The parties after the second one cannot learn the number of users held by

the previous company without colluding with the one coming before the previous one in the sequence. Without revealing useful information to each other, the parties can determine their users' clusters using PDSOM protocol off-line.

To compute similarities between users, z-scores are employed in PCC (Herlocker et al., 1999), as follows:

$$w_{au} = \sum_{j=1}^{J} zs_{aj} zs_{uj} \tag{2.13}$$

where zs_{aj} and zs_{uj} show z-scores (zs) values of item j of users a and u, respectively, J shows commonly rated items set, and w_{au} is similarity weight between users a and u. The zs values can be found, as follows: $zs_{uj} = (v_{uj} - \overline{v_u})/\sigma_u$, where $\overline{v_u}$ and σ_u show the mean and the standard deviation of user u's ratings, respectively. After similarity computation, predictions again are produced using Eq. (1.3).

To request a referral, a asks a prediction for q from MC. Once the MC receives a's data and her query, it provides the prediction after estimating it through collaboration with other parties online. Note that each party knows the $\boldsymbol{W_j}$ vectors and the clusters of their users. The MC first determines a's cluster by calculating the distances between a and each cluster center. It assigns a to the closest cluster. The users' data in that cluster is then used to estimate a prediction for a using Eq. (2.13) and Eq. (1.3). As seen from such equations, zs_{uj} and deviation from mean ratings of q (v_{duq}) are needed. Notice that $v_{duq} = (v_{uq} - \overline{v_u})$, where v_{uq} is user u's ratings for item q. Thus, in order to improve online performance, each party computes both zs_{uj} and v_{duq} values off-line and stores them. They can compute such values because they have the data required to calculate them. Eq. (1.3) can be written as $p_{aq} = \overline{v_a} + P$, where P is:

$$P = \frac{\sum_{u=1}^{k}\left[\sum_{j=1}^{J} zs_{aj} zs_{uj}\right] v_{duq}}{\sum_{u=1}^{k}\sum_{j=1}^{J} zs_{aj} zs_{uj}} = \frac{\sum_{j=1}^{J} zs_{aj}\left[\sum_{u=1}^{k} zs_{uj} v_{duq}\right]}{\sum_{j=1}^{J} zs_{aj}\left[\sum_{u=1}^{k} zs_{uj}\right]} \tag{2.14}$$

As seen from Eq. (2.14), since the MC is able to compute zs_{aj} values and $\overline{v_a}$ value, it can estimate p_{aq}. However, since data are horizontally distributed among z parties, P can be written, as follows:

$$P = \frac{\sum_{j=1}^{J} zs_{aj}\left[\sum_{u=1}^{k_1} zs_{uj} v_{duq} + \sum_{u=1}^{k_2} zs_{uj} v_{duq} + \cdots + \sum_{u=1}^{k_z} zs_{uj} v_{duq}\right]}{\sum_{j=1}^{J} zs_{aj}\left[\sum_{u=1}^{k_1} zs_{uj} + \sum_{u=1}^{k_2} zs_{uj} + \cdots + \sum_{u=1}^{k_z} zs_{uj}\right]} \tag{2.15}$$

where k_1, k_2, …, k_z show the number similar users held by the first, second, …, and the z^{th} party, respectively, who rated q. The MC, which is asked by a for recommendation, needs aggregate data from other z-1 companies to estimate recommendations. If the MC sends a's data to other parties, they can easily compute the required data. However, since a's ratings are valuable and will be added to the MC's database; and her data will be used for prediction generation in the following queries, it does not send them to collaborating companies. Thus, such companies should compute $\sum_{u=1}^{k} zs_{uj} v_{duq}$ and $\sum_{u=1}^{k} zs_{uj}$ aggregate values for all j = 1, 2, …, m-1, and send them to the MC without greatly jeopardizing their privacy. Therefore, the parties follow the following private distributed *k-nn* CF protocol (PDKNN) to offer predictions.

PDKNN: The details of the protocol are, as follows:

i. The MC assigns a to the closest cluster. It sends target cluster (C_a) and q to other parties.

ii. Each party including the MC computes $\sum_{u=1}^{k} zs_{uj} v_{duq}$ and $\sum_{u=1}^{k} zs_{uj}$ aggregate values based on those users' data that are in C_a. They then send them to the MC.

iii. The MC then is able to estimate P using Eq. (2.15) after collecting required aggregate data from other parties.

iv. It finally estimates p_{aq} and returns it to a.

The parties can succeed recommendation process based on HDD. However, PDKNN has the following shortcomings:

i. Since the MC has the required partial results for the C_a to estimate p_{aq}, it can use them for producing referrals for those active users who will be in that cluster and ask prediction for q.

ii. The MC can collect aggregate data values for fake active users over a time in order to derive information about other parties' databases.

To overcome the aforementioned shortcomings, the parties follow the following steps to compute aggregate values in the step 2 of PDKNN protocol, where the new protocol is called as the IPDKNN (Improved PDKNN).

IPDKNN: The details of the protocol are, as follows:

i. Each party g uniformly randomly selects a random number (β_g) over the range (0, γ]. They then uniformly randomly choose β_g percent of the users who did not rate q, where the probability of selecting any user is proportional to the number ratings she has due to accuracy concerns. In other words, the chance of selecting a user with more ratings is bigger than the chance of selecting a user with fewer ratings.

ii. Each party then fills selected users' cells for q with non-personalized ratings (v_d). Since v_d values are estimated based on available ratings, when selecting users, giving higher probability to those users with more ratings makes sense. The parties generate v_d values using the distribution of users' ratings, which can be considered as a Gaussian distribution with mean (μ) and standard deviation (σ).

iii. Before calculating $\sum_{u=1}^{k} zs_{uj} v_{duq}$ and $\sum_{u=1}^{k} zs_{uj}$ aggregate values for all m-1 items, each party uniformly randomly selects some of its zs_{uj} values, removes their values, and replaces with zero. For this purpose, each data holder g uniformly randomly selects a random number (α_g) over the range (0, δ]. They then uniformly randomly choose α_g percent of their zs_{uj} values, remove their values, and replace with zero.

iv. Each party then estimates the required aggregate values based on its modified database. They finally send them to the MC.

When the parties follow the aforementioned protocols, they preserve their privacy against each other and such protocols force them to collaborate whenever a asks a prediction from one of them. They can produce accurate and dependable predictions. In one hand, the parties increase the amount of data involved in aggregate data computation by inserting v_d values in some q's empty cells. On the other hand, the amount of data involved in such computations is reduced due to removed zs_{uj} values. In each query, data owners choose different β and α so that unpredictable randomness is added to their databases. Each party will compute different partial results for a cluster in different recommendation processes. Therefore, they collaborate with each other to answer queries until they have enough data to offer accurate and dependable recommendations by themselves.

2.5. Privacy Analysis

In this section, the proposed schemes are analyzed in terms of privacy according to privacy definition.

Analysis of the VDD Protocols: Note that each party tries to hide their true ratings and the rated items from each other. Thus, throughout the proposed privacy-preserving schemes, actual votes and the rated items are considered private; while *s*, cluster centers, user mean ratings, and vector lengths are considered public. The DPP and the P_DP are secure because data owners do not exchange any data during such protocols. Each party itself can conduct them without the help of others. It is impossible to derive information about each other's data through them. However, the parties exchange data while performing other protocols.

SOMP is secure due to following reasons: First, as explained previously, the parties determine the initial constant values without exchanging any data they hold. Second, each party g computes $\Sigma_j(o_{gj} - w_{gj})^2$ based on the corresponding parts of any user's data (object o) and weight vector it holds to estimate the distance between any user vector and each weight vector. They then exchange such aggregate values. Since j, o_{gj}, and w_{gj} values are known by the party g only, the parties cannot derive any useful information from such aggregate sum values. Third, after every iteration, each data holder updates the corresponding part of $\boldsymbol{W_j}$ it holds without collaboration by utilizing Eq. (2.3). Therefore, other parties cannot obtain any useful information during updating step. And finally, at the end of the clustering, the parties reveal $\boldsymbol{W_{gj}}$ vectors updated s times to each other to get the entire $\boldsymbol{W_j}$ vectors. Since $\boldsymbol{W_{gj}}$ vectors are updated by using Eq. (2.3) based on each party's data, the data owners cannot get any useful information about each other's data even if some vendors collude.

In UMRP, the parties exchange sum and count aggregates. Since each party disguises its data set using the DPP, the parties cannot learn true rating values from such sum values. Such aggregate values are the sum of the true ratings and inserted personalized votes. Even if the number of filled cells is guessed, the collaborating parties cannot estimate their sum because such personalized ratings are known by the party only that generates them off-line. However, any party can guess the filled cells or the rated items held by a collaborating party based on the count value came

from that party. The probability of guessing the correct β_g is 1 out of 100. After guessing it, the party can estimate number of filled cells or the rated items (m_{rg}). Finally, the probability of guessing which items are rated is 1 out of $Comb_{m_{rg}}^{count}$, where $Comb_w^f$ represents the number of ways of picking w unordered outcomes from f possibilities. Therefore, for any company, the probability of guessing the rated items based on the received count is 1 out of $100 \times Comb_{m_{rg}}^{count}$.

VLP includes utilizing DPP and working on aggregates. The parties exchange aggregated sums estimated from filled vectors. The collaborating companies cannot learn true ratings and the rated items from received aggregates because such values are the sum of the squares of normalized votes. Moreover, the item means are used for normalization are known by the parties only who own such items. Thus, the VLP is secure.

In P_NP, privacy is achieved through DPP, encryption, and permutation. Each party utilizes HE to hide the true values of their private data. Since it is shown that HE is secure (Paillier, 1999), the parties cannot derive information from encrypted values. Although the MC inserts fake ratings into q's votes, collaborating parties can guess the number of filled cells and the rated items of q held by the MC. The probability of guessing the correct β_g is 1 out of 100. Then, number of filled cells of q, m_{qg}, can be estimated. Finally, the probability of guessing which items are rated is 1 out of $Comb_{m_{qg}}^{q_i'}$, where q_i' shows the number of ratings including the fake ones of q. The MC can guess the rated items held by each collaborating party after receiving encrypted data. For the MC, estimating the probability of guessing the rated items of one item held by a collaborating company can be explained, as follows: The probability of guessing the correct β_g is 1 out of 100. The MC then can estimate number of filled cells of that item, m_{rg}. The probability of guessing the correct position of the item vector is 1 out of $m_i!$ Similarly, the probability of guessing the correct position of the values in a column vector is 1 out of $Comb_{m_q}^{q_i''}!$, where q_i'' shows the number of encrypted values that the MC sends and m_q shows the number of values in the received column vector because such values are computed between commonly rated items only. Finally, the probability of guessing which items are rated is 1 out of $Comb_{m_{rg}}^{m_q}$. The probability of guessing the true

values of normalized ratings can be estimated similarly. However, since only the results on commonly rated items are exchanged, the MC cannot learn other values.

The security of the PRP depends on inserting default values into *a*'s ratings vector, encryption, and permutation. The P_N and P_D values are parts of model distributed among multiple parties; and they are considered private. The parties including the MC can use privacy attacks to derive data during the PRP. To overcome such attacks, fake values are introduced. First of all, the MC disguises *a*'s data to prevent the collaborating parties acting as an *a* in multiple scenarios to derive its data. The basic idea of this type of attack, called the colluded collaborating companies (3C), is to change only one rating of those items held by the MC in each prediction process by acting as an *a*. After multiple interactions, the collaborating parties can learn the P_N and P_D values held by the MC. Second, the collaborating parties also masks *a*'s ratings vector due to the similar reasons. This time the MC acts maliciously to learn the P_N and P_D values held by a victim company. The MC changes only one rating of those items held by the target or the victim party in each prediction process by acting as *a*. After multiple interactions, the MC can learn such values held by the victim. And finally, in a different type of 3C attack, the victim can be one of the collaborating parties other than the MC. Similarly, the colluding parties can learn the victim's data. In each recommendation process or PRP conducted online, the parties mask *a*'s data differently so that they can defend themselves against the aforementioned attacks. The collaborating parties cannot learn the MC's data encrypted using the K_{MC}, because decryption key is known by the MC only. The MC can compute the number of inserted fake values by a collaborating party *g*, referred to as m_{fg}, and it knows the number of items held by that party, referred to as m_g. Thus, the probability of guessing which item cells are filled is 1 out of $Comb_{m_{fg}}^{m_g}$. However, due to permutation, the server first needs to determine the correct positions. The probability of guessing such positions is 1 out of $m_{ag}!$, where m_{ag} shows the number of data values including the fake ones in the corresponding part of *a*'s ratings vector held by the party *z*.

Analysis of the HDD Protocols: To analyze the proposed approach in terms of privacy, possible privacy attacks caused by the parties and vulnerabilities are determined. Then, proposed solutions against such attacks and weaknesses are

introduced; and how private the scheme is shown. Such attacks and vulnerabilities can be described, as follows:

i. *A_1: Parties can coalesce for capturing a target party's data:* In this attack, *z*-1 companies can coalesce for capturing a target site's private data. Since the target party acts as the MC, it collects partial results, combines them with its own ones, and returns a recommendation to *a*. Therefore, other parties can derive its partial results from the output prediction value for *a* in multiple scenarios. One of the corrupted parties can act as an *a* and requests referrals for some target items in several settings to derive data about the target party. After they get at least *m*-1 recommendations for the same item, they are able to figure out target party's partial results. Once they determine such partial results calculated for different target items, such corrupted parties can derive information about the target items' ratings. If it is conducted, A_1 can jeopardize the MC's privacy. To defend itself against A_1, the MC also should utilize IPDKNN protocol like auxiliary parties do. Since the MC adds arbitrariness to its private data in each recommendation computation process, corrupted parties cannot get exact partial results to derive information about the MC's database.

ii. *A_2: Paying-off:* This attack is similar to A_1. Instead of acting as an *a* in multiple states, *z*-1 corrupted parties can bribe *a* to gather the target party's useful information for themselves. Moreover, since the databases are updated periodically by inserting active users' ratings and removing some old ratings, the spoiled companies utilize such bribed users' data in order to derive information about the target party's data. As explained previously, if data owners including the MC follow the IPDKNN protocol, they can defend themselves against A_2. Since each party including the MC can bribe any active user, any induced user can be corrupted again by offering more incentives. Therefore, this attack might become expensive and data obtained through this attack are more likely questionable.

iii. *V_1: Not able to return any result:* To be part of the recommendation process, each party should return partial results. To do that, each assisting party must have at least one user who rated *q*. One or more parties might

not be able to return any partial results to the MC. This phenomenon happens when such auxiliary parties face with extreme cases in which they may have no rating for *q*. Such cases definitely leak information about secondary parties' data. In other words, the MC learns that such parties, that are not able to return any partial results, do not have any rating for *q*. This absolutely violates data owners' privacy. If the MC knows the users held by that party who did not send any partial results, it can offer special discounts to them for selling *q*. In order to resolve V_1, each party should return results to the MC even if they do not have any rating for *q*. If such parties follow the proposed IPDKNN protocol properly, they can easily overcome this weakness.

iv. V_2*: Missing values in aggregate values vector:* This limitation occurs when some assisting parties face with another type of extreme case. When such parties do not have any ratings for items other than *q*, aggregate values for them will be 0. If this is the case, similar to the V_1, the MC concludes that they do not have any ratings for such items. It then can exploit this information to make financial benefits. This vulnerability can be easily fixed. When any party faces with V_2, they just fill some of the randomly chosen cells of such items with non-personalized votes. Thus, the MC cannot learn whether they have any items without any ratings.

The proposed scheme contains distributed data-based SOM clustering, which preserves data owners' confidentiality. The parties are able to cluster their users without exchanging private data. They exchange updated weight vectors and *s* values. Thus, the parties cannot learn useful information about each other data during clustering. Each party learns the number of users held by the previous party in the sequence. However, that information does not cause any privacy, financial, or legal problems. Even if they have such information, they cannot use it to determine true ratings and rated and/or unrated items held by the assisting company.

As explained above, proposed scheme is able to preserve data owners' confidentiality against the aforementioned privacy attacks and vulnerabilities. If each party including the MC follows the proposed protocols, they defend

themselves against A_1 and A_2; and they are able to overcome vulnerabilities V_1 and V_2. Due to the randomness they add, the MC and the auxiliary parties preserve their privacy. In addition, the scheme utilizes normalized values (*zs* and deviation from mean values), which are computed by each party alone. Since computations are performed on such normalized values, even if they are derived, it becomes difficult to learn true ratings because attacking party does not know the mean and standard deviation values. Also, the parties exchange aggregate values rather than individual data items. Finally, due to randomness, which added in each prediction generation process, the MC cannot keep the interim results collected from other parties for future recommendation processes. It must collaborate with other companies for upcoming predictions.

2.6. Supplementary Cost Analysis

Due to privacy protection measures, extra costs like storage, communication, and computation costs are inevitable because privacy, accuracy, and performance conflict with each other. Note that off-line computation and communication costs are not critical for overall performance. Therefore, it is better to conduct as many computations as possible off-line in order improve online efficiency. However, in order to provide new recommendations after users provide new ratings, the collaborating parties need to update their databases by inserting new votes. In other words, to get new ratings involved in prediction process so that new referrals can be estimated, the companies should update their model (clustering users conducted off-line) periodically. They then provide recommendations online using the up-to-date model and the ratings.

Analysis of the VDD-based Scheme: Supplementary costs due to privacy are important for efficiency. Extra storage, communication and computation costs are analyzed. The proposed scheme introduces additional storage costs, as follows: Storage costs for saving cluster centers are in the order of $O(mnc)$ and $O(zmnc)$ in the original and the proposed schemes, respectively; where notice that *nc* shows number of clusters. Thus, storage costs needed to save cluster centers increase by *z* times because each company needs to save cluster centers. Due to the non-personalized ratings used in the UMRP, additional storage costs are in the order of $O(nc)$. The parties estimate a model off-line by utilizing the P_NP and P_DP. Due to

storing P_N values estimated using the P_NP, extra storage costs are in the order of $O(m^2)$. Similarly, additional costs are in the order of $O(m^2)$ due to storing P_D values.

Due to privacy concerns, the scheme introduces extra communication costs. In the SOMP, number of communications the parties need to make is in the order of $O(ez^2)$, where e shows the number of epochs. In the UMRP and the VLP, number of communications are both in the order of $O(z^2)$, where the parties exchange interim aggregate results. Additional communication costs conducted during the P_NP are in the order of $O(zm)$. The P_DP does not cause any communication costs. Notice that since all protocols except PRP are performed off-line, supplementary communication costs due to privacy concerns are not critical for online performance. However, the PRP is conducted online and number of communications made during online phase is imperative. During the PRP, there are $2(z-1)$ communications made online. Although the number of extra communications made online is in the order of $O(z)$, they are made simultaneously between the MC and each collaborating company.

The proposed scheme introduces extra computations. However, not all of these computations are performed online and directly affect the performance of recommendation process. Also note that, since the parties conduct computations simultaneously, execution time for total computations in all protocols will be determined by the party, which owns the maximum number of items (m_{max}). All protocols except the PRP are performed off-line, their computation costs are not that critical for recommendation process. The SOMP does not cause any extra computation cost. During the DPP, the computations costs for choosing random values and selecting some of the unrated item cells are negligible. However, additional computation costs due to estimating personalized ratings are in the order of $O(m^2n)$. Due to the UMRP and the VLP, there are no supplementary computation costs. In the P_NP, extra computation costs are inevitable due to HE. Number of encryptions and decryptions performed in the P_NP are $\bar{n} \times m \times (m + 1)$ and $\bar{n}m^2$, respectively, where $\bar{n}$ shows the average number of ratings for each item including the inserted false ones. Unlike the P_NP, there are no supplementary computation costs due to the P_DP. The PRP is performed online and causes extra

computation costs due to HE. Number of encryptions and decryptions conducted online during the PRP are $3\bar{m}$ and $2\bar{m}$, respectively, where $\bar{m}$ shows the number of ratings including the inserted ones in *a*'s rating vector. Although supplementary online computation costs are inevitable, again notice that such computations are performed in parallel by the collaborating parties. To determine the running times of cryptographic algorithms, benchmarks for the CRYPTO++ toolkit from www.cryptopp.com/ can be used (Canny, 2002a). According to study conducted by Yakut and Polat (2012a)**,** it takes 80 milliseconds to perform an encryption and decryption using the abovementioned benchmarks.

Analysis of the HDD-based Scheme: To improve online performance, the parties compute normalized values off-line and store them. Due to their storage, extra storage cost is in the order of $O(nm)$. Since each party uses the cluster centers, they save them in z matrices with size $nc \times m$. Accordingly, additional storage cost is in the order of $O(m)$ because z and nc are constants. Although the scheme seems to cause further storage costs; however, the parties should save such information after calculating off-line to improve online performance even if they offer referrals by themselves.

Providing recommendations is an online process. Performing CF tasks efficiently is imperative. For this reason, the proposed scheme must not introduce significant extra computation costs that might harm the efficiency of CF schemes. Due to clustering, which is conducted off-line, additional costs are not critical. During online phase, data owner insert some default votes, which increases the amount of computations, while they remove some of the z-scores, which decreases the amount of computations. Generally speaking, applying these two different randomness processes surpass their effects on the amount of computations. Furthermore, data used in CF are distributed in the scheme and online computations are done simultaneously. Traditional *k-nn*-based scheme's online computation time (without the use of clustering) is in the order of $O(nm)$, while the proposed scheme improves online computation costs by $z \times nc$ times without considering communication costs due to clustering and simultaneous computations.

The proposed scheme introduces extra online communication costs. In traditional CF schemes, *a* sends a message to a server, which returns a single

prediction. Hence, number of communications is two only. In the distributed scheme, *a* sends the same message to the MC as in traditional systems. However, the MC sends C_a and *q* to *z*-1 companies so that they return partial results (two vectors containing *m*-1 aggregate values). In other words, number of communications is 2*z* in the scheme or in the order of *O*(*z*), where *z* is a constant. Therefore, communication costs increase by *z* times. Supplementary costs can be considered negligible because auxiliary parties simultaneously communicate with the MC.

2.7. Accuracy Analysis: Experiments

To show how accurate the proposed scheme-based predictions are, several experiments are conducted using real data sets. Three well-known data sets called MovieLens (ML), Jester, and EachMovie (EM) were constructed for CF purposes. ML was collected by GroupLens research team at the University of Minnesota. ML and EM (Billsus and Pazzani, 1998) have ratings for movies while Jester data set contains ratings for jokes (Gupta et al., 1999). These data sets are described in Table 2.1. Since Jester data set is very dense, it is not utilized in VDD scheme's experiments. To measure accuracy performance of HDD, experiments are performed on Jester and ML data sets while VDD scheme's experiments are performed on ML and EM data sets.

Table 2.1. Data Sets

Name	**Item**	**Size (*n* × *m*)**	**Total Votes**	**Density (%)**	**Range**	**Type**
Jester	*Joke*	24,983 × 100	1,810,455	72.47	[-10, 10]	*Continuous*
ML	*Movie*	6,040 × 3,900	1 million	4.22	[1, 5]	*Discrete*
EM	*Movie*	72,916 × 1,628	2.8 millions	2.29	[0, 1]	*Discrete*

To measure the quality of the referrals, mean absolute error (MAE) and normalized mean absolute error (NMAE) are used, which happen to be the most popular statistical accuracy measures. They are widely used in CF systems. To compute NMAE, MAEs are normalized by dividing them the difference between the maximum and the minimum ratings. The lower the MAE and the NMAE, the

more accurate the outcomes are. They measure how close the predictions with privacy concerns to the true ones. Since collaboration among multiple parties increases the amount of data involved in prediction process for HDD, the parties are able to generate predictions for more items and they might overcome cold start problem. Thus, to show how collaboration affects the number of items for which predictions could be provided, coverage metric is utilized, which is the percentage of items for which a CF algorithm can provide referrals. Coverage can be calculated, as follows: $Coverage = v_{res}/v_{test}$, where v_{res} and v_{test} stand for the number of predictions returned and the number of test ratings, respectively. Finally, statistical *t*-tests are applied in order to show that the results are statistically significant and they are not occurred by chance. First of all, a *t* value is computed. Then, a *p*-value from *t*-distribution table is found. If the *p*-value chosen for some significance level (usually 0.10, 0.05, or 0.01) is less than the calculated *t* value, then it is concluded that the improvements are statistically significant and they are not happened by chance.

Before performing the experiments, firstly, users having at least 50 items are determined. Then, they are uniformly randomly divided into two disjoint sets, training and test sets. For each test user, five rated items are uniformly randomly chosen. After withholding their true ratings, their entries are replaced with null; and recommendations are provided for them using the train users' data. It is assumed that data are distributed among *z* companies, where *z* might in [1, 10]. Thus, in VDD, any data owner *g* has *n* train users with about *m*/*z* items while it has *n*/*z* users' ratings for *m* items in HDD. In the following, experiments for VDD are explained.

Experiment 1: To show how accuracy changes due to collaboration among *z* parties, experiments using ML and EM data sets are conducted. 500 and 1,000 users are used for testing and training, respectively, where *m* is 1,600 for both data sets. To determine the best *k* neighbor, optimum cluster number is set at 3. As shown by Roh et al. (2003), clustering data into three clusters happens to give the best results. After clustering train data, for each test users, five predictions are generated for randomly chosen five rated items based on split data only and

integrated data. *z* is changed from 1 to 7. NMAEs are displayed for both data sets in Figure 2.2 after computing overall averages.

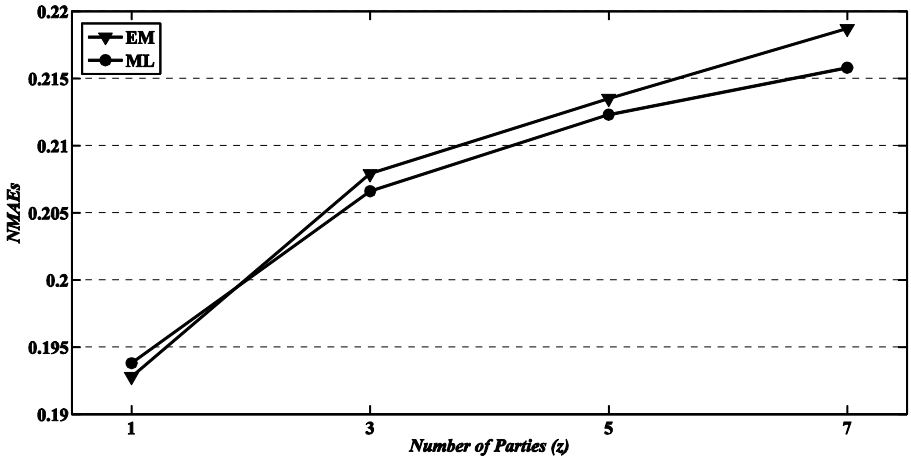

Figure 2.2. NMAEs with Varying Number of Collaborating Vendors

As seen from Figure 2.2, the quality of the recommendations improves with collaboration of parties. In other words, if data owners provide predictions on their integrated data, they offer more accurate recommendations. When data are distributed among multiple parties, amount of data held by each party is not enough for precise referrals. However, as seen from the results, data owners are able to offer better recommendations if they collaborate with each other. With increasing *z* values or decreasing amount of ratings held by each party, accuracy diminishes. For example, when users' ratings are distributed among seven parties for EM data set, the NMAE is about 0.2187 for each individual party, while it is about 0.1928 when they collaborate. In other words, due to collaboration, accuracy improves by about 10%. For ML data set, if data are distributed among seven parties, the NMAE is about 0.2157. However, accuracy improves to 0.1938 if the parties collaborate. For both data sets, accuracy develops due to providing referrals on combined data. Thus, integrating split data definitely enhances accuracy. To show if the improvements due to collaboration are statistically significant, *t*-tests are conducted. The values of *t* are 3.29 and 2.95 for EM and ML, respectively, where *z* is 7. Those values are statistically significant for 99% confidence intervals for both data sets.

Experiment 2: The effects of randomly inserted fake ratings into the train data are investigated. Note that the parties can disguise their data using the DPP protocol. Again, 500 users for testing and 1,000 users for training for both data sets are used. θ values are varied from 0 to 100 and personalized votes estimated from available data by each party are employed as fake ratings. The experiments are run 100 times. After calculating overall averages for both data sets, the outcomes are presented for varying θ values in Figure 2.3.

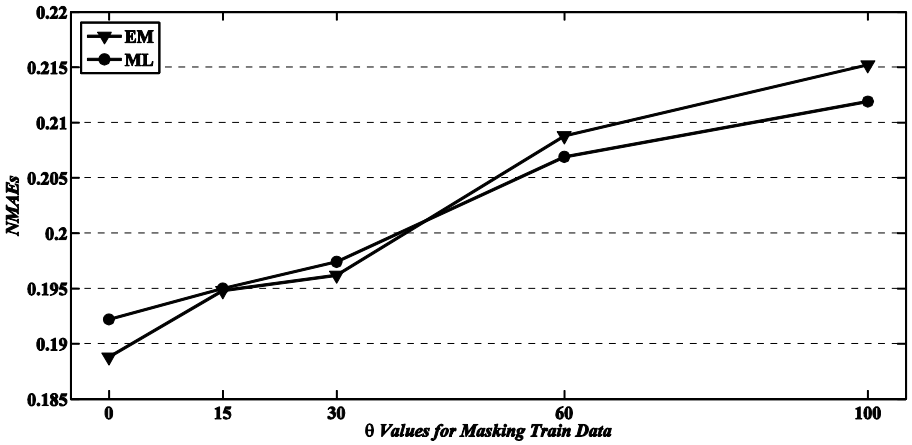

Figure 2.3. NMAEs with Varying θ Values for Train Data

The results in Figure 2.3 show that accuracy diminishes with increasing number of fake ratings for both data sets. Since ML is very sparse and it has more items than EM, with increasing θ values, more fake ratings are added. As a result, accuracy losses due to such inserted votes are greater for ML than for EM with larger θ values. However, according to the results, it is still possible to produce accurate recommendations when data owners hide their ratings by inserting fake ratings according to θ values. Note that since privacy and accuracy are conflicting goals, the parties can determine the optimum θ values that they use according to the privacy and accuracy levels they want. Thus, 30 can be chosen as the optimum θ value for masking train data.

Experiment 3: To show the effects of adding fake ratings into the test users' rating vectors, another set of trials are performed. The same methodology is used. θ values are varied from 0 to 100 while using 500 users for testing and 1,000 users for training. Train users again are clustered into three clusters. Trials are run 100

times for both data sets. Overall averages of NMAE values are shown for varying θ values in Figure 2.4.

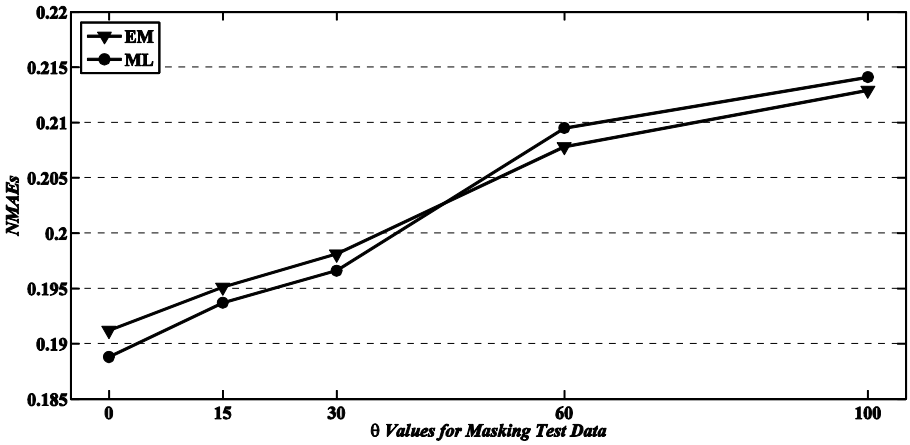

Figure 2.4. NMAEs with Varying θ Values for Test Data

According to results shown in Figure 2.4, accuracy diminishes with increasing number of fake ratings inserted into the test users' data, as expected. With increasing θ values, randomness increases; and that makes accuracy worse. However, accuracy losses are small for smaller θ values (values less than or equal to 30) and that helps the parties provide accurate recommendations. As previously mentioned, the optimum θ value for masking test data can be determined. In the following experiments, θ is set at 30 to mask test data.

Experiment 4: After showing the effects of varying θ values used to mask train and test data separately, trials are run to show the joint effects of such values while varying n. Both θ values are set at 30 and experiments are conducted using 500 users for testing while varying n from 250 to 1,000 for both data sets. Predictions are estimated for randomly selected five rated items for each test user. NMAEs values are presented for both data sets in Figure 2.5 and the MAEs for ML in Figure 2.6.

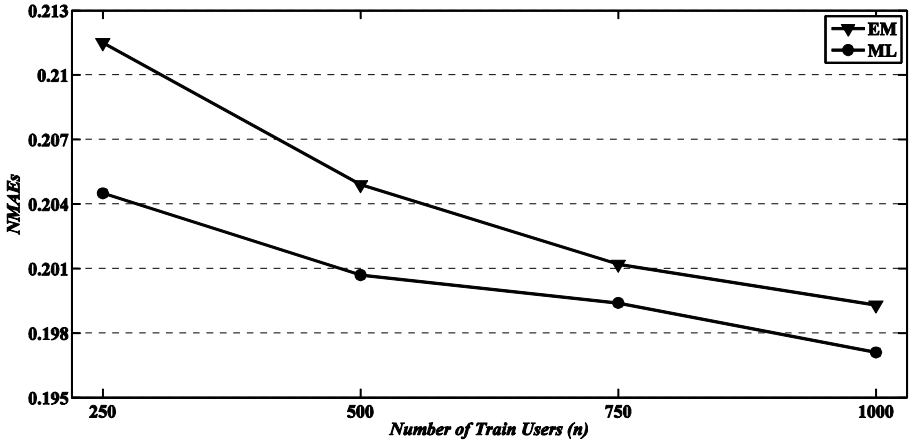

Figure 2.5. NMAEs with Varying *n* Values

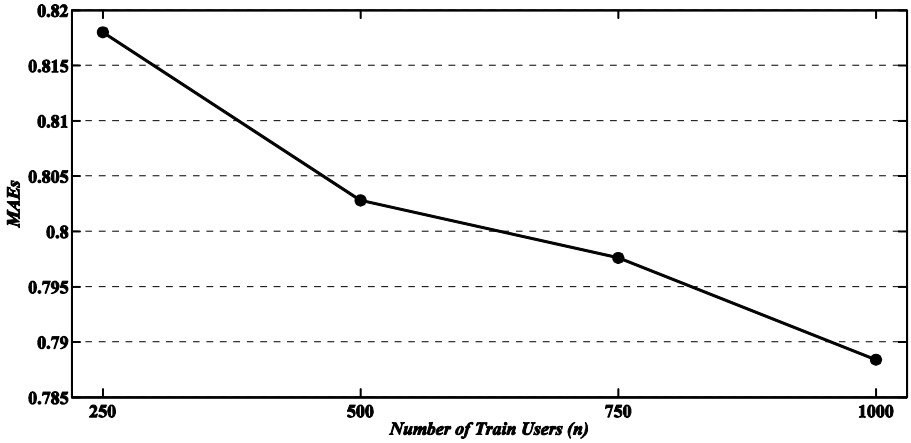

Figure 2.6. MAEs with Varying *n* Values

In Figure 2.5 and Figure 2.6, the NMAE and the MAE values show that the proposed solution produces accurate predictions while preserving privacy with increasing *n* values. According to the results, for EM data set, if data are distributed among seven parties, the accuracy of recommendations is about 0.2187 without privacy concerns when *n* is 1,000. However, when data owners collaborate and produce predictions using the proposed scheme, accuracy improves to 0.1989, where *n* is 1,000 and θ values are 30. The results are similar for ML, as well. When data are distributed among seven parties and the parties collaborate without jeopardizing their privacy, the MAE is about 0.7882. If they do not collaborate, accuracy decreases to 0.8004. As shown previously, accuracy improves due to collaboration and decreases due to privacy concerns. However, for larger *n* values (*n* is 1,000), accuracy gains due to collaboration outweigh accuracy losses due to

privacy concerns. To show if the improvements due to collaboration with privacy concerns are statistically significant, *t*-test is conducted. The *t* value is about 2.72 for EM when *n* is 1,000 and *m* is 1,600, where data are distributed among seven parties. Similarly, the *t* value is 1.98 for MLM when *m* is 3,900. Those values are statistically significant for 99% and 95% confidence intervals for EM and ML, respectively. Thus, although accuracy becomes worse due to privacy concerns, accuracy gains are still significant due to distributed CF with privacy concerns. The introduced scheme can produce accurate recommendations without deeply jeopardizing data owners' privacy.

After describing the experiments for VDD, the conducted experiments for HDD are explained in the following.

Experiment 1: Trials are performed to demonstrate how coverage changes with varying *n* and *z* values. In other words, how collaboration affects overall performance is tested firstly. It is hypothesized that the parties are able to provide predictions for more items if they integrate their split data through collaboration. To verify this hypothesis, experiments are performed while changing *n* from 250 to 1,000 and *z* from 1 to 10. It is assumed that if there is at least one rating for *q* and at least two commonly rated items between *a* and those users who rated *q*, the CF system can provide referrals for *q*. Coverage values are found for data owners based on data sets they own only (split data) and combined data (collaboration) for both data sets. Since Jester is a dense data set (the density of the set is about 72%), coverage is 100% even if *n* is 250 and *z* is 10. However, since ML is a very sparse set (the density is about 4%), coverage is significantly affected by varying available data. In Figure 2.7, average coverage values for ML with varying *n* and *z* values are presented.

As expected, coverage significantly improves with increasing *n* values for sparse data set ML. If *n* increases, amount of ratings involving in recommendation process also increases; that makes coverage better. As seen from Figure 2.7, due to integrating split data, coverage enhances. When 250 users' data horizontally distributed among 10 parties, coverage is about 42%. If they integrate their data through collaboration, coverage increases to 78%. Note that when *z* is 1, data are held by a single party. In other words, when $z = 1$, it means that the parties decide

to collaborate. For sparse data sets, collaboration among various parties definitely improves coverage.

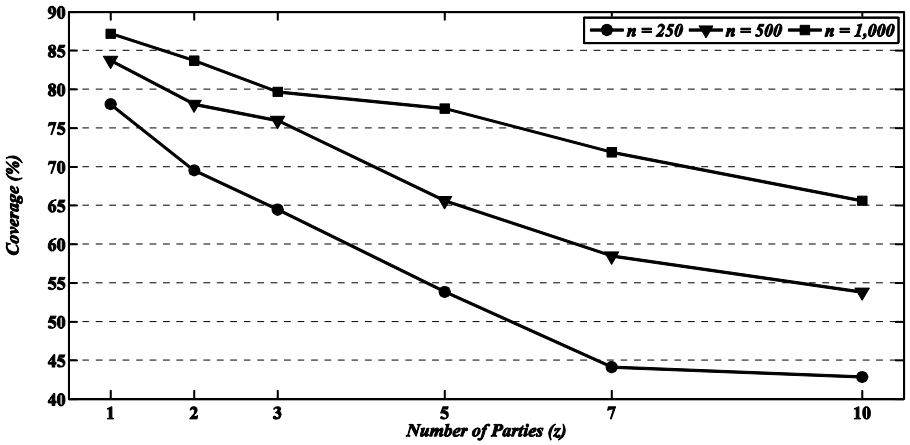

Figure 2.7. Coverage with Varying *n* & *z* Values

Experiment 2: To show how accuracy changes due to collaboration among multiple parties, experiments are conducted using both data sets. It is wanted to compare the results with and without collaboration. 500 users for testing are used while 250, 500, or 1,000 users are used for training, where *z* is varied from 1 to 10. Notice again that when $z = 1$, it means that data owners collaborate and provide predictions on integrated data. If $z = 2, 3$, and so on, then it means that data are partitioned between two, three parties, and so on, respectively. With increasing *z* values from 2 to 10, the parties provide predictions on their split data only. Number of users held by each party decreases with increasing *z* values. Train data is clustered using SOM clustering into three clusters. Predictions for test items are estimated firstly for all test users using the data held by each party only. Then, predictions are estimated for the same items using the integrated data. MAE values are computed for both cases; and displayed them in Figure 2.8a and Figure 2.8b for Jester and ML data sets, respectively.

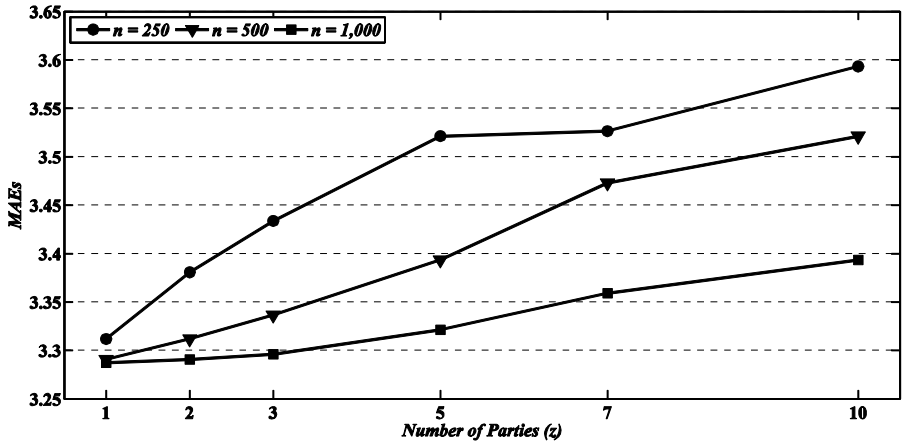

Figure 2.8a. MAEs with Varying *n* & *z* Values (Jester)

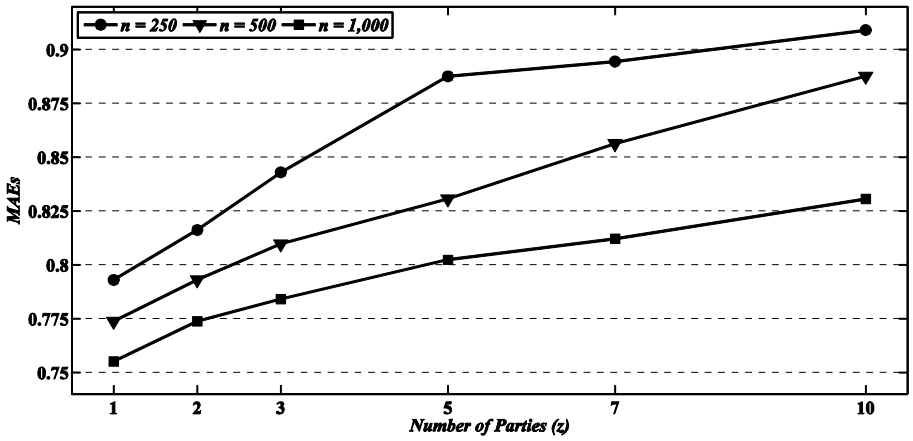

Figure 2.8b. MAEs with Varying *n* & *z* Values (ML)

As seen from Figure 2.8a and Figure 2.8b, when data owners decide to collaborate, they achieve the best results (the outcomes for $z = 1$). The MAE values improve with decreasing *z* values. In other words, if data owners provide predictions on their integrated data via collaboration, they offer more accurate recommendations. Similarly, accuracy enhances with increasing *n* values, as expected. When data are distributed among various parties, each party uses its data to provide predictions. Since available data decrease, accuracy becomes worse. If they decide to collaborate, they are able to use more data for referral generation. That makes accuracy better. When *n* is 1,000 and data are distributed among 10 parties, the MAE is about 0.83 for ML, while it is about 0.75 if they collaborate. Thus, integrating split data definitely enhances the quality of the predictions. For

both data sets, accuracy develops due to providing referrals on combined data. In order to show whether the improvements due to collaboration are statistically significant or not, *t*-tests are conducted. For example, for ML, the improvements are statistically significant for n = 1,000 and z = 10 because the value of t is 5.94, which is still greater than the value of t for significance level = 0.01 in the *t*-table. Similarly, for Jester, the value of t is 9.51, which is still greater than the value of t in the *t*-table for significance level being 0.01. For both data sets, the improvements are still statistically significant for z = 5 for significance level = 0.01.

Experiment 3: In the following trials, it is assumed that data are distributed between 10 vendors. To protect each data owners' privacy and let them collaborate for future recommendations, they add randomness to train data. As explained previously, data owners add default ratings into randomly chosen some of the unrated cells of q. Each party g selects a random number β_g over the range (0, γ] to fill randomly chosen β_g percent of unrated cells of q with default votes. Trials are performed for both data sets while varying γ from 0 to 100 to show how accuracy changes with various amount of randomness. 1,000 and 500 users are used for training and testing, respectively for both data sets. Since uncertainty is added, trials are performed 100 times. After the outcomes are computed in terms of MAE values, they are shown in Table 2.2 for both data sets. Note that γ being 0 represents the case without privacy concerns.

Table 2.2. MAEs with Varying γ Values

γ	**0**	**6.25**	**12.50**	**25**	**50**	**100**
Jester	3.2880	3.2980	3.3120	3.3140	3.3220	3.3360
ML	0.7552	0.7560	0.7576	0.7604	0.7652	0.7864

As seen from Table 2.2, with increasing γ values, the quality of the recommendations generally becomes poorer. As expected, adding randomness to original data makes accuracy worse. However, the results are still promising because default votes are inserted in order to add randomness. Such votes are non-personalized ratings and they might represent users' true preferences. For ML,

increasing γ values make accuracy worse. For Jester, although the MAE values become poorer with increasing γ values, accuracy losses due to inserting default votes are very small compared to ML. The parties can append uncertainty into their partial results by inserting non-personalized votes into unrated cells of q without greatly sacrificing on accuracy.

Experiment 4: Besides inserting default ratings into unrated cells of q, data owners randomly select some of their z-scores and then remove them. To do this, each party g uniformly randomly selects α_g over the range (0, δ]. Removing some of the z-scores affects accuracy because the amount of ratings involved in recommendation process decreases. Therefore, in order to show how removing various amounts of z-scores affect the outcomes, experiments are run while varying δ from 0 to 100. In the experiments, uniformly randomly selected α_g percent of the z-scores from train sets are removed. Again the same train and test users as in the previous trial are used. After MAE values are computed, they are displayed in Figure 2.9a and Figure 2.9b for Jester and ML data sets, respectively. Note again that δ being 0 means that the parties do not remove any z-scores.

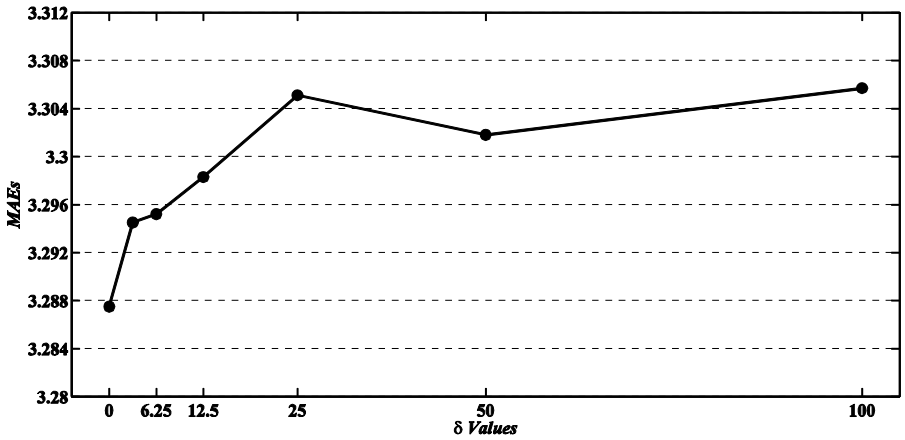

Figure 2.9a. MAEs with Varying δ Values (Jester)

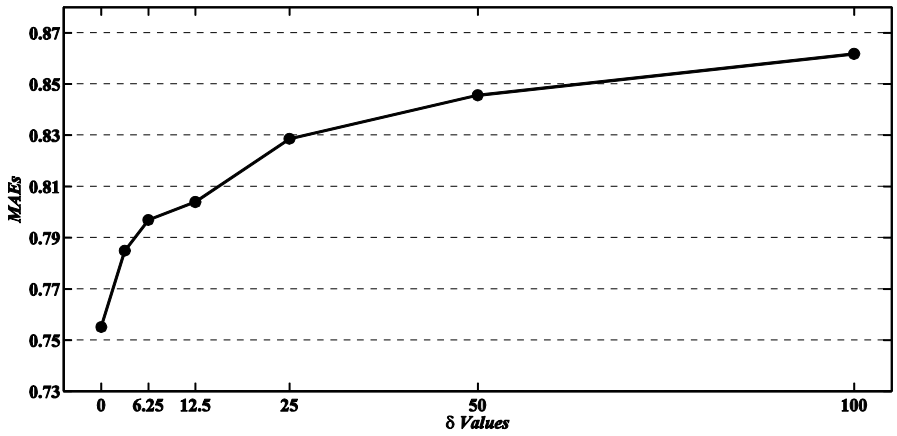

Figure 2.9b. MAEs with Varying δ Values (ML)

As seen from Figure 2.9a and Figure 2.9b, MAE values become worse with escalating δ values. However, compared to losses in ML, accuracy losses in Jester due to removing z-scores are very small. The reason for this phenomenon can be explained with the density of Jester. Since Jester is much more dense set than ML, removing some of the z-scores does not significantly affect the quality of the predictions. Even if δ is 100, the MAE increases from 3.2875 to 3.3057 for Jester. Although MAE values become worse with increasing δ values for ML, the results are hopeful when δ is smaller than 12.5. To achieve better results in terms of accuracy, 3.125 is determined as the optimum value of δ for both data sets. However, the parties can use different δ values in order to achieve required levels of privacy and accuracy.

Finally, experiments to show the joint effects of γ and δ values with varying n values are conducted. Although the optimum values of γ and δ in the previous trials are determined, they are varied from 0 to 12.5 to show how overall performance changes with various γ and δ values. Also, n is varied from 250 to 1,000. The same 500 test users are used. After the trials for both data sets are performed, MAE values are computed; and they are displayed in Figure 2.10a and Figure 2.10b for Jester and ML data sets, respectively.

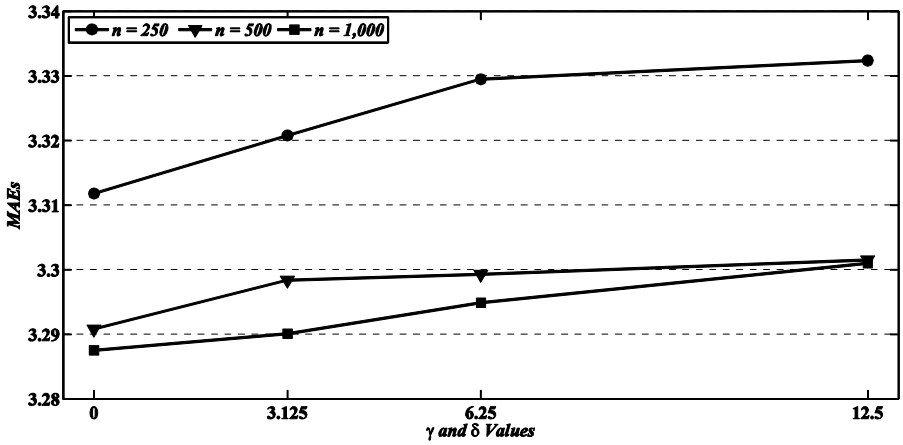

Figure 2.10a. Joint Effects of Varying γ and δ Values on MAEs (Jester)

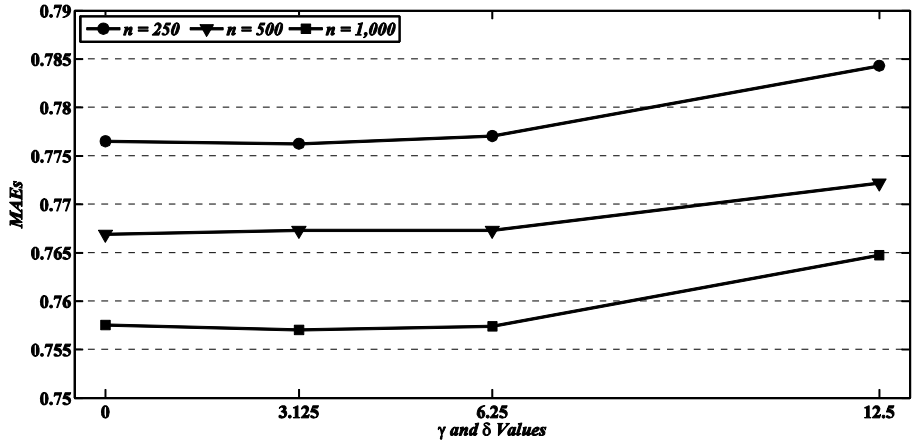

Figure 2.10b. Joint Effects of Varying γ and δ Values on MAEs (ML)

Figure 2.10a and Figure 2.10b show that the joint effects of such measures on accuracy are negligible. It is still possible to offer accurate recommendations with privacy concerns. As shown previously, due to collaboration among various parties even competing companies, improvements in accuracy are statistically significant. On the other hand, privacy-preserving measures cause losses in accuracy. Such losses should not surpass the gains due to alliance. Compared to the enhancements, accuracy losses are smaller.

To show whether privacy-preserving distributed scheme develops accuracy significantly or not, *t*-tests are performed. The results on split data are compared with the ones on the proposed scheme. For ML data set, for example, the improvements are statistically significant for n = 1,000, $\gamma = \delta$ = 3.125, and z = 10

because the value of t is 5.63, which is still greater than the value of t for significance level = 0.01 in the t-table. In the same case, for Jester, the value of t is 7.62, which is still greater than the value of t in the t-table. Even if z is 5, the improvements are still statistically significant for both data sets. As the t-tests show, the distributed data-based scheme with privacy improves accuracy. When data holders offer predictions on their split data only, accuracy diminishes due to the insufficient amount of ratings. However, if they collaborate, accuracy enhances even if they apply privacy-preserving measures because of increasing amount of available ratings.

2.8. Conclusions

In this chapter, privacy-preserving schemes are described in order to offer accurate recommendations using the SOM-based CF algorithm from vertically or horizontally distributed data. The schemes help data owners produce referrals based on their combined data without greatly jeopardizing their privacy. Several protocols or building blocks are presented to achieve privacy. It is shown that the protocols are secure and the scheme is able to offer predictions while preserving data owners' privacy. Due to collaboration and privacy-preserving measures, additional costs are inevitable. Hopefully, it is demonstrated that supplementary costs due to the schemes are negligible and the companies are still able to produce referrals efficiently. Various experiments are performed to show how accuracy enhances due to collaboration. The results demonstrate that the quality of the predictions significantly improves due to collaboration. Since accuracy diminishes due to privacy concerns, trials are conducted to investigate how privacy concerns affect the outcomes. As shown by the experimental results, the companies are able to offer recommendations with decent accuracy when they mask their data to achieve privacy. To sum up, the parties holding VDD or HDD can use the proposed schemes to produce accurate predictions efficiently without deeply jeopardizing their privacy.

3. PRIVACY-PRESERVING RANDOM PROJECTION-BASED RECOMMENDATIONS ON DISTRIBUTED DATA

This chapter presents the solution proposed for enabling data owners' partnership in recommendation process while preserving their privacy using RP. Since RP is a dimension reduction method, besides confidentiality defending, it also reduces data in recommendation process and improves performance of collaboration. The solution eliminates the aforementioned anxieties of data owners, so that the vendors can work in partnership. Data owners can produce recommendations privately from vertically or horizontally distributed data by utilizing the proposed schemes.

3.1. Introduction

RP is a powerful and computationally simple dimensionality reduction method. Employing it does not cause considerable overhead costs. Online performance, which is very critical for overall performance of recommender systems, can be significantly enhanced when user-item matrix is reduced by using RP. Likewise, distributed data can be reduced by using it. Moreover, since it contains randomness in its nature, reducing dimensionality through RP also improves privacy protection level of private data. If the original data is projected onto a random subspace using RP, original form of data is changed; however, its statistical characteristics are preserved. Therefore, RP hides original data while reducing it. Due to these dual gains, RP is a good choice to reduce dimensions of distributed data with privacy.

RP projects m-dimensional data into a d-dimensional subspace, where $d << m$. The main idea behind random mapping in RP depends on the Johnson-Lindenstrauss lemma (Bingham and Mannila, 2001), which proposes that if points in any vector space are projected to a randomly selected subspace, the distance between the points will be nearly the same. To reduce data, RP is utilized a random $m \times d$ matrix $\mathbf{R}$ whose columns have unit lengths (Bingham and Mannila, 2001). If the original data is $\mathbf{D}_{n\times m}$, which can be reduced to $\mathbf{RD}_{n\times d}$ by using $\mathbf{R}_{m\times d}$, as follows:

$$\mathbf{RD}_{n\times d} = \mathbf{D}_{n\times m} \times \mathbf{R}_{m\times d} \tag{3.1}$$

To select elements of r_{ij} of **R**, there are different approaches. **R** is usually generated from elements having Gaussian distribution; but, according to Achlioptas (2001), a simple distribution can be used, as follows to choose r_{ij}:

$$r_{ij} = \begin{cases} \sqrt{3} & \text{having probability } \frac{1}{6} \\ 0 & \text{having probability } \frac{2}{6} \\ -\sqrt{3} & \text{having probability } \frac{1}{6} \end{cases} \quad (3.2)$$

This distribution still satisfies the Jahnson-Lindenstrauss lemma (Bingham and Mannila, 2001). Bingham and Manila (2001) compare the experimental results of utilizing two different distributions; and they suggest using the Gaussian distribution-based **R** matrix if data are dense and the distribution proposed by Achlioptas if data are sparse. Since data used in recommendation algorithms are generally sparse, the distribution proposed by Achlioptas (2001) is utlized.

RP is one of the widely used methods in various data mining applications. RP-based dimension reduction provides decent accuracy in clustering and estimating similarities (Kaski, 1998; Achlioptas, 2001). Bingham and Mannila (2001) apply RP to reduce dimension of image and text data. They show that RP-based data reducing gives good results for image and text data. Fern and Brodley (2003) investigate how RP can be used for clustering high dimensional data. Liu et al. (2006) suggest utilizing RP in distributed data mining applications. They explore to use multiplicative RP matrices to preserve data holders' privacy. Their technique is successful for applying to different types of privacy-preserving data mining applications over horizontally or vertically partitioned data. Oliveira and Zaïane (2007) utilize RP to cluster distributed data while preserving the parties' privacy. In their method, they reduce dimension of data with RP. In another study, RP is used to compute the correlation matrix of privacy sensitive data (Kargupta et al., 2003).

In this chapter, solutions for producing recommendations on multi-party distributed data without disclosing data owners' confidential data to each other are proposed. The proposed methods help cooperating firms hide the true ratings and the rated entities against each other. To carry out privacy, RP is suggested. The goal is to offer accurate referrals efficiently while preserving privacy. Hence, the suggested solutions should bring equilibrium among such divergence goals. The

methods are flexible rather than rigid schemes. The solutions are evaluated in terms of privacy; and shown that they do not violate data owners' confidentiality. Online performance analysis shows that supplementary loads caused by privacy measures are insignificant. Real data-based experiments are conducted to assess the schemes in terms of the quality of the recommendations. The outcomes demonstrate that the results are promising. The schemes are able to provide accurate referrals efficiently while preserving privacy.

3.2. VDD-based Recommendations with Privacy

Computations performed for estimating predictions can be grouped as off-line and online calculations. The detail descriptions of them are given in the following.

Off-line Computations: Collaborating sites first need to reduce their split data using RP in such a way so that confidential data remain private. The following private dimension reduction protocol on random projection is proposed (PD2RP):

i. Decide the value of d (dimension of reduced data).

ii. Each party g generates a random matrix $\mathbf{R_g}$ with the size $m_g \times d$, using the distribution proposed by Achlioptas (2001). Note that the random matrix $\mathbf{R} = \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \\ \vdots \\ \mathbf{R_z} \end{bmatrix}$.

iii. Then, each party g estimates $\mathbf{RD_g} = \mathbf{D_g} \times \mathbf{R_g}$, where $\mathbf{RD} = \mathbf{RD_1} + \mathbf{RD_2} + \ldots + \mathbf{RD_z}$.

iv. To allow continual collaboration, the parties exchange partial $\mathbf{RD_g}$ with each other in such a way so that the $\mathbf{RD_{n\times m}}$ is vertically distributed among z parties (Each party g holds $\mathbf{RD_{gv}}$ with the size $n \times m_g$, where m_g, on average, equals d/z). The first party keeps $\mathbf{RD_{11}}$ (with the size $n \times m_1$), sends $\mathbf{RD_{12}}$ (with the size $n \times m_2$), $\mathbf{RD_{13}}$ (with the size $n \times m_3$), and so on to the second, third, and other parties, respectively. Also, the second party keeps $\mathbf{RD_{22}}$ (with the size $n \times m_2$), sends $\mathbf{RD_{21}}$ (with the size $n \times m_1$), $\mathbf{RD_{23}}$ (with the size $n \times m_3$), and so on to the first, third, and other parties, respectively. All collaborating vendors do the same thing.

v. At the end, each party g adds the partial $\mathbf{RD_g}$ matrices, gets $\mathbf{RD_{gv}}$; and

stores them. Thus, **RD** is distributed among *z* parties.

After reducing the user-item matrix, the vendors are supposed to estimate the user mean and standard deviation values based on the condensed data because such aggregates are needed during similarity computations. They are estimated off-line so that online performance improves. Due to the nature of data distribution, reduced data are distributed among *z* parties. Hence, the corporations need each other to estimate such values in a distributive manner. The user mean values on reduced data, referred to as $\overline{r_u}$, can be estimated, as follows, where it is assumed that $m = m_g$:

$$\overline{r_u} = \frac{\sum_{g=1}^{z}\sum_{j=1}^{m} r_{ugj}}{d} \tag{3.3}$$

where r_{ugj} shows the value of reduced vote for item *j* of user *u* held by the party *g*. As seen from Eq. (3.3), each party *g* finds partial sum values ($\sum_{j=1}^{m} r_{ugj}$) for each user and sends them to other parties. After receiving required data from others, each party estimates user mean values on reduced data. Similarly, the users' ratings standard deviation on reduced data (σ_{ru}) can be estimated, as follows:

$$\sigma_{ru} = \sqrt{\frac{1}{d}\sum_{g=1}^{z}\sum_{\mathrm{j}=1}^{m}(r_{ugj} - \overline{r_u})^2} \tag{3.4}$$

Each party can easily normalize the reduced data by subtracting user mean values from corresponding data items because they know such user mean values estimated formerly. They then find the squares of each normalized data items for all users. Next, each party sends the partial sum values for all items to all other collaborating parties. They finally estimate σ_{ru} values for all users using Eq. (3.4).

Online Computations*:* Due to VDD, when *a* wants a prediction for *q*, she sends corresponding parts of her ratings vector **A**, referred to as $\mathbf{A_g}$, to the related parties. She also sends the query *q* to MC holding the ratings of *q*. The process continues, as follows:

i. *a* sends $\mathbf{A_g}$ to each collaborating vendor *g*.

ii. After receiving $\mathbf{A_g}$, each party *g* reduces *a*'s data and obtains $\mathbf{RA_g} = \mathbf{A_g} \times \mathbf{R_g}$.

iii. The parties then are supposed to collaboratively estimate *a*'s ratings

mean and standard deviation based on reduced data. They can achieve such task as they do for estimating mean and standard deviation values for other users off-line.

iv. Parties try to compute PCC given in Eq. (1.1) between users and *a*. Each party g finds $w_{aug} = \frac{\sum_{j \in Jg}(v_{ajg}-\overline{v_a})(v_{ujg}-\overline{v_u})}{\sigma_a\, \sigma_u}$ values for all users, where $w_{au} = \frac{\sum_{j \in J1}(v_{aj1}-\overline{v_a})(v_{uj1}-\overline{v_u})}{\sigma_a\, \sigma_u} + \frac{\sum_{j \in J2}(v_{aj2}-\overline{v_a})(v_{uj2}-\overline{v_u})}{\sigma_a\, \sigma_u} + \dots + \frac{\sum_{j \in Jz}(v_{ajz}-\overline{v_a})(v_{ujz}-\overline{v_u})}{\sigma_a\, \sigma_u}$. Although sending such partial similarity weight values to MC does not reveal any confidential data; however, if MC receives them, it does not need other parties when the same active user asks predictions for those items held by MC. Thus, the parties do not send such partial sums to MC.

v. Since MC does not have the similarity weights, it cannot determine *a*'s neighborhood alone. *a*'s neighbors can be determined using the private neighborhood formation algorithm (PNFA), explained in the following subsection.

vi. After forming the neighborhood using the PFNA, MC can estimate p_{aq} employing the private recommendation protocol (PRP), explained in the following subsection.

vii. MC finally returns that prediction to *a*.

***Private Neighborhood Formation Algorithm* (*PNFA*)**: Collaborating companies are able to estimate partial similarity weights between *a* and each user in their databases because they have the required data. Due to the wishes for continual collaboration, they do not want to send such aggregate values to MC. On the other hand, *a*'s neighborhood should be formed in order to compute predictions. The parties can determine *a*'s neighbors using the following scheme, called PNFA:

i. Each vendor *g* computes partial similarity weights (w_{aug}) for all users u = 1, 2, …, n.

ii. Then, each company *g* sorts the users according to w_{aug} values in descending order.

iii. They then associate each user in the ordered list with a virtual weight

starting from the first one, as follows: Since there are *n* users, the first user in the list is associated with the virtual weight $n \times (max_{wp}\text{-}min_{wp})$, the second one gets $(n\text{-}1) \times (max_{wp}\text{-}min_{wp})$, and so on. The last user in the list is associated with $1 \times (max_{wp}\text{-}min_{wp})$, where $max_{wp}\text{-}min_{wp}$ is the difference between the maximum and the minimum partial similarity weight values.

iv. Next, each party exchanges such virtual weights together with user IDs with other retailers.

v. For all users, each corporation then finds the sum of the virtual weights assigned by each party.

vi. The companies sort the users according to the sum of the virtual weights. In case of equality, prefer those users with lower IDs.

vii. The first *k* users are finally chosen as neighbors for *a*. Notice that each corporation ends up with the same neighborhood.

***Private Recommendation Protocol* (*PRP*)**: After neighborhood formation, MC can collaboratively estimate p_{aq}. Note that the similarity weights of *a*'s neighbors are distributed among the collaborating companies. Due to the distributed weights, Eq. (1.3) can be written, as follows:

$$p_{aq} = \overline{v_a} + \frac{\sum_{u \in NN}(v_{uq} - \overline{v_u})w_{au}}{\sum_{u \in NN} w_{au}} = \overline{v_a} + \frac{\sum_{u \in NN}(v_{uq} - \overline{v_u})(w_{au1} + w_{au2} + \ldots + w_{auz})}{\sum_{u \in NN}(w_{au1} + w_{au2} + \ldots + w_{auz})}. \quad (3.5)$$

As seen from Eq. (3.5), in order to estimate predictions, either MC sends $(v_{uq} - \overline{v_u})$ values to collaborating parties or cooperating vendors send w_{aug} values to MC. The following protocol is proposed to estimate p_{aq} without violating data owners' privacy:

i. MC finds $V_{uq} = (v_{uq} - \overline{v_u})$ values for all *a*'s neighbors who rated *q*. Notice that the user *u* is one of *a*'s neighbors and rated *q*.

ii. It then encrypts V_{uq} values using a HE scheme and obtains $\xi_{K_{MC}}(V_{uq})$ values. Next, MC sends them to the collaborating parties.

iii. Using HE, each party *g* computes $\xi_{K_{MC}}(Q_g) = \prod_{u=1}^{k_a} \xi_{K_{MC}}(V_{uq}))^{w_{aug}}$, where k_a is the set of *a*'s neighbors who rated *q*.

iv. Each cooperating vendor *g* also estimates $\xi_{K_{MC}}(W_g) = \prod_{u=1}^{k_a} \xi_{K_{MC}}(w_{aug})$ using HE property.

v. Then, each party sends such encrypted partial sum values to MC, which can decrypt them using its private key and obtains Q_g and W_g values.

vi. MC finally computes $p_{aq} = \overline{v_a} + \frac{\sum_g Q_g}{\sum_g W_g}$, and returns it to a.

While conducting the PRP, the vendors should not be able to derive information about each other's data. However, since MC sends encrypted data of those users who rated q only, other parties can figure out which neighbors of a did not rate q. That violates the privacy definition. PRP can be improved if some arbitrariness is added using non-personalized votes, as follows:

i. MC determines the number of a's neighbors who did not rate q and unrated cells of q, which are called blank cells.

ii. It uniformly randomly selects a θ value over the range [1, β], where β is an integer between 1 and 100 and called performance parameter.

iii. Then, it uniformly randomly chooses θ percent of the blank cells.

iv. It finally fills such chosen blank cells with item average default rating, which is a non-personalized vote.

Note that off-line computations are not that critical for the overall performance. Therefore, the parties try to perform as many computations as possible off-line to enhance online performance. An example that showing distributed computations and data exchanges in our proposed VDD-based recommendations with privacy scheme is given in Figure 3.1.

3.3. HDD-based Recommendations with Privacy

In addition to vertical distribution, data can be distributed horizontally among multiple vendors. The proposed scheme can be similarly used to offer HDD-based predictions. The computations carried by the collaborating vendors can be divided into two main phases, as follows:

Off-line Computations: The first thing that should be executed by the vendors is reducing the distributed data without disclosing confidential data to each other using RP. They can achieve such task, as follows:

i. Decide the value of d (dimension of reduced data).

ii. Construct a random matrix, $\mathbf{R_{m \times d}}$ having the distribution proposed by Achlioptas (2001).

iii. Each party g finds $\mathbf{RD_g} = \mathbf{D_g} \times \mathbf{R}$; and stores the partial reduced data $\mathbf{RD_g}$

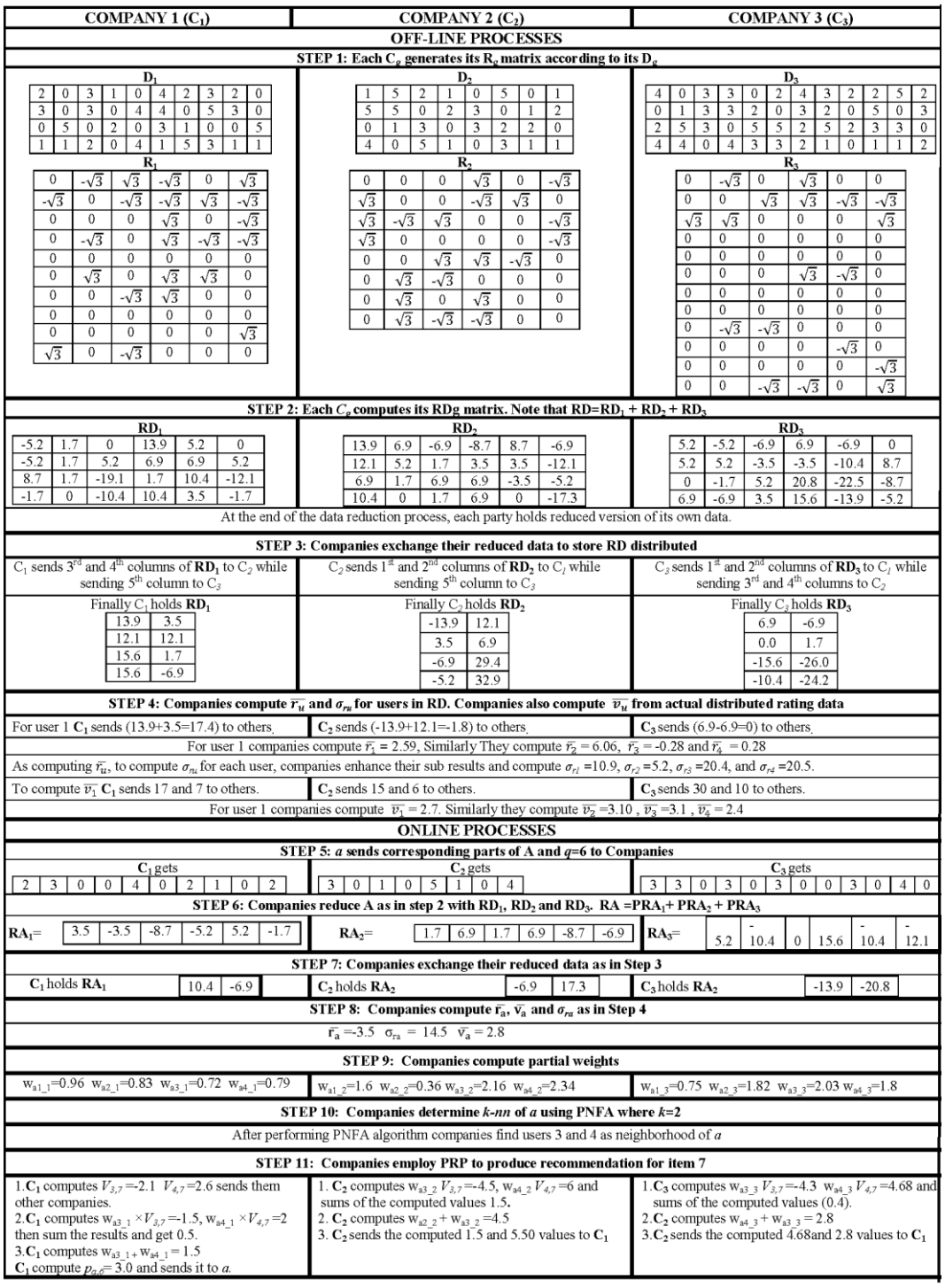

Figure 3.1. Example of VDD-based recommendations with privacy with 3 Parties

After estimating reduced data without disclosing the confidential data, the companies can now compute the means and the standard deviations of users' ratings, which are required for recommendation estimation. Due to the nature of the distribution, each vendor is able to compute such values without the help of other companies based on reduced data that it holds. Thus, each party finds out the mean and standard deviation of those users' ratings values held by it; and stores them.

Online Computations: During an online interaction, *a* sends **A** and *q* to one of the collaborating companies to get a recommendation for item *q*. Due to the HDD, Eq. (1.3) can be written, as follows:

$$p_{aq} = \overline{v_a} + \frac{\sum_{u \in NN_1}\left((v_{uq}-\overline{v_u})w_{au}\right) + \cdots + \sum_{u \in NN_z}\left((v_{uq}-\overline{v_u})w_{au}\right)}{\sum_{u \in NN_1} w_{au} + \sum_{u \in NN_2} w_{au} + \cdots + \sum_{u \in NN_z} w_{au}} \qquad (3.6)$$

where NN_1, NN_2, …, NN_z are the sets of *a*'s neighbors held by each company. p_{aq} can be collaboratively estimated while preserving privacy, as follows:

i. MC reduces **A**; and obtains **RA** = **A**×**R**.

ii. It forwards **RA** and *q* to the collaborating parties.

iii. Each party including MC computes the similarity weights between *a* and each user in their databases based on the reduced data using Eq. (1.1).

iv. They then choose those users as neighbors whose similarity weights satisfy a τ value.

v. Next, each party g calculates partial aggregate results, $\sum_{u \in NN_1}\left((v_{uq} - \overline{v_u})w_{au}\right)$, $\sum_{u \in NN_2}\left((v_{uq} - \overline{v_u})w_{au}\right)$, …, $\sum_{u \in NN_z}\left((v_{uq} - \overline{v_u})w_{au}\right)$; and $\sum_{u \in NN_1} w_{au}$, $\sum_{u \in NN_2}\left((v_{uq} - \overline{v_u})w_{au}\right)$, …, $\sum_{u \in NN_z} w_{au}$ values.

vi. The collaborating vendors send such aggregate partial sums to MC, which estimates p_{aq} using Eq. (3.6); and sends it to *a*.

Notice that the collaborating parties are not able to derive truthful information from *a*'s reduced data due to random projection. Due to the nature of data distribution, the parties are able to estimate the users' mean ratings and their ratings' standard deviation without exchanging any data. Prediction estimation performed online is achieved by exchanging aggregate data. An example that

showing distributed computations and data exchanges in our proposed HDD-based recommendations with privacy scheme is given in Figure 3.2.

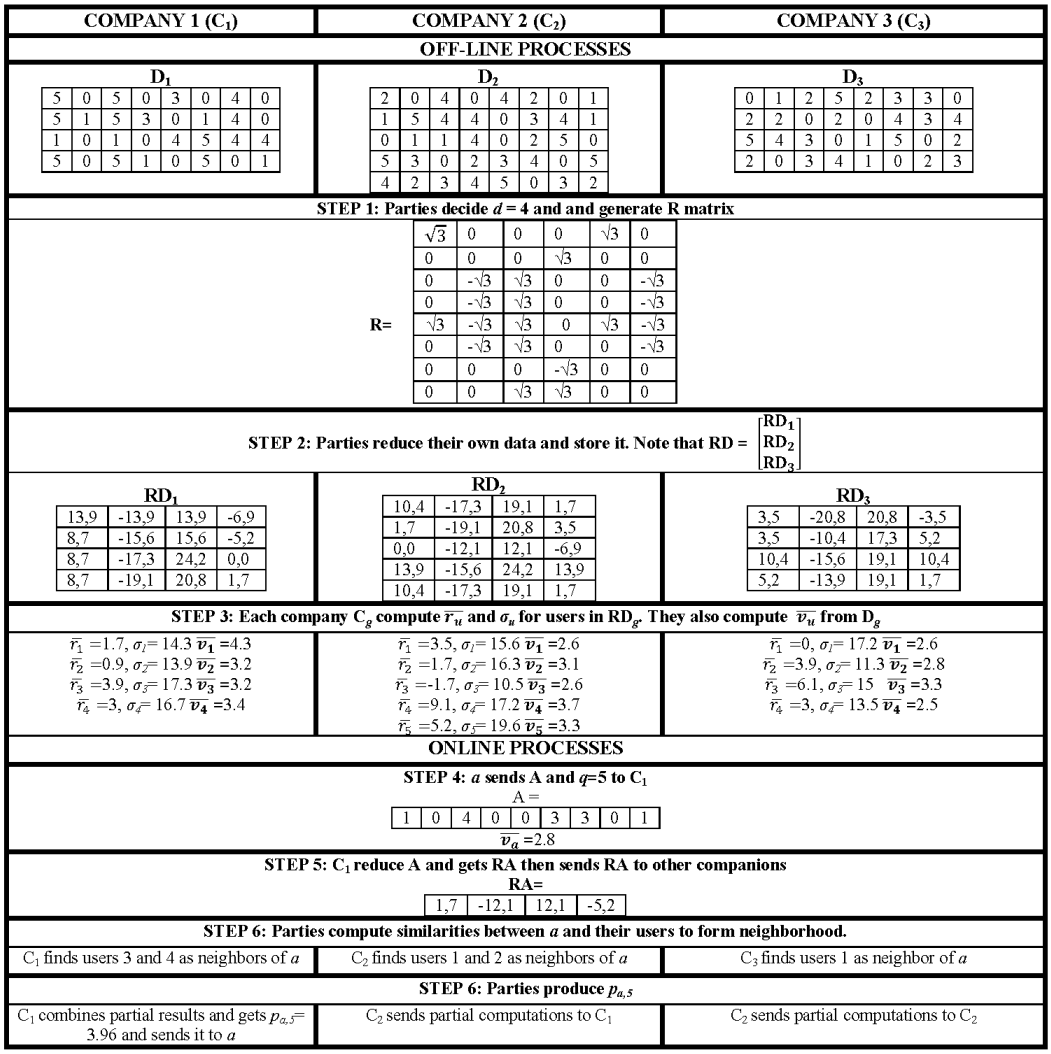

Figure 3.2. Example of HDD-based recommendations with privacy with 3 Parties

3.4. Privacy Analysis

In this section, the proposed schemes are analyzed in terms of privacy. It is shown that the schemes do not violate data owners' confidentiality in terms of the criteria determining how an algorithm enforces the main goals of PPDM (Bertino et al., 2008). The criteria consist of privacy level indicating how sensitive information can be acquired from hidden data, referred to as hiding failure (HF), which is measured as a percentage of sensitive information that is still discovered after perturbation of private data; and data quality showing how hidden data are still useful for data mining purposes (Oliveira et al., 2002; Bertino et al., 2008). RP-

based data perturbation's privacy level is analyzed deeply by Liu et al. (2006). According to their results, RP-based data perturbation technique guarantees original data's confidentiality. Their results prove that, even if the parties know **R**, it is impossible to acquire exact values of the confidential data hold by any party and they cannot identify any parties' private data by a random guess of **R**, unless **R** is disclosed.

Due to the property of underlying data masking method, VD2RP scheme preserves data owners' confidentiality. In VD2RP, each data owner produces its own random matrix $\mathbf{R_g}$. Thus, the parties cannot derive any information about other companies' data even if the resulted **RD** matrix is vertically distributed, because each data owner holds its corresponding part of **RD**. As explained by Liu et al. (2006), by having resulted **RD** matrix, data owners cannot identify the original data even if they guess the random matrix randomly. Therefore, utilizing RP-based data reduction process on VDD does not violate data owners' privacy. Note that the companies compute means and standard deviations of the users in **RD** matrix by exchanging partial aggregates. Although they exchange such consolidated results, it does not jeopardize HF, because they are acquired from **RD** not from the sensitive data. Hence, computing user means and standard deviations from **RD** does not violate data owners' privacy and HF. The PNFA algorithm can similarly be analyzed. In this algorithm, the parties exchange virtual weights rather than true weight values. It is not possible to derive information from such virtual weights. Utilizing PNFA algorithm only discloses virtual weights; and that does not decrease privacy level of VD2RP and jeopardize HF. Another algorithm employed in VD2RP is PRP algorithm in which HE is utilized to preserve data owners' privacy. The parties cannot derive sensitive information from the encrypted values. In PRP, MC inserts fake ratings to q's votes, thus, privacy level of PRP depends on how collaboration parties can guess the number of actual ratings and fake ratings. Remember that MC adds fake ratings to θ percent of blank cells. Probability of guessing the correct θ value is 1 out 100. Then, guessing probability of the exact position of filled cells in vector of item q is $Comb_{q_a}^{q_f}$, where q_f shows the number of fake ratings while q_a shows all encrypted values send by MC. Since adding fake ratings increases inconsistency of the

received encrypted values, privacy level of VD2PR increases. Data quality in VD2RP is affected by adding fake ratings to *q*'s rating vector while utilizing RP. However, such ratings are considered non-personalized votes and they might represent users' preferences. PRP method preserves data quality.

Due to the findings presented by Liu et al. (2006), the proposed HD2RP scheme is able to preserve data owners' confidentiality. Since the resulted **RD** in HD2RP is horizontally distributed without any disclosure, the privacy level of the scheme is enforced. Moreover, in HD2RP method, all parties compute mean and standard deviations of their users in **RD** without any collaboration; and the parties send aggregate partial sums to MC only while producing recommendations online. Thus, it can be concluded that the HF of HD2RP is zero, which is the main goal of a privacy-preserving algorithm. Revealing the aggregate partial sums does not leak any information about true ratings and the rated items. Therefore, privacy level is high. Although HF is zero and the privacy level is high in HD2RP scheme in general, extreme cases like in the following can violate data owners' confidentiality. If any party has no users having rating for *q*, it then cannot send any partial sum to MC; and MC concludes that such users provide no ratings for *q*. To overcome this exception, the party has no users having rating for *q* can select random users and fill their cells for *q* with user mean values as proposed in PRP. Finally, since after RP-based data perturbation, the reduced data is still appropriate for performing data mining tasks such as clustering, correlation computation, etc. (Liu et al., 2006), data performed in HD2RP has still enough quality to compute correlations between users to compute weights in recommendation process.

3.5. Supplementary Costs Analysis

Supplementary costs due to privacy are important for efficiency. Unlike online costs, off-line costs are not that critical for the overall success of prediction schemes. Thus, extra storage, communication, and computation costs are analyzed. VD2RP scheme introduces extra storages costs in the order of $O(nm_g)$ and $O(n)$ for $\mathbf{RD_g}$ matrices and users' mean and standard deviation values, respectively. Similarly, HD2RP method also introduces some extra storage costs for each data owner *g* for storing $\mathbf{RD_g}$ matrices and user means and standard

deviations, which are in the order of $O(n_g m)$ and $O(n_g)$, respectively.

Due to privacy concerns, the proposed schemes introduce extra communication costs. In VD2RP, during off-line phase, each party sends the corresponding part of their $\mathbf{RD_g}$ matrix to related party making z-1 communications. Therefore, total number of communications happens to be $z(z$-1). In each communication, amount of data transferred is in the order of $O(nm_g)$. To compute users' mean and standard deviations in **RD**, data owners again make $z(z$-1) communications. Since such communications are made off-line, they do not make online performance worse. During online computations, number of extra communications caused by PNFA is $2z(z$-1). In PRP algorithm, data holders make $4(z$-1) communications. In HD2RP, there is no additional communications performed off-line. During online phase, MC makes z-1 communications to send **RA** to other parties. To obtain interim results from collaborating parties, extra z-1 communications are made. Since all communications are made parallel during online phase, their effects to online efficiency of proposed scheme become smaller.

The proposed schemes introduce extra computation costs due to preserving data owners' confidentiality. In VD2RP protocol, during online step, reducing a's data introduce extra computations. Since data owners make this reduction simultaneously, execution time for this computation will be determined by the party, which owns the maximum number of items, referred to as m_{max}. Thus, additional costs are in the order of $O(dm_{max}^2)$. Due to employing PNFA, supplementary computation costs are in the order of $O(nlogn)$ for sorting n users. Additional costs due to addition are negligible in PNFA. During PRP, encryptions and decryptions are conducted. In HD2RP, reducing a's data requires extra computations, which are in the order of $O(dm^2)$.

3.6. Accuracy Analysis: Experiments

In order to test the accuracy of the proposed schemes, several experiments are conducted using ML and EM data sets. To measure the quality of the recommendations, NMAE measure is employed. Since collaboration among multiple parties increases the amount of data involved in recommendation process, the parties can generate predictions for more items. In other words,

coverage enhances significantly. In order to demonstrate how collaboration improves the coverage, coverage metric is utilized. Finally, *t*-tests are applied in order to evaluate the schemes in terms of statistical significance.

Given the entire data sets, firstly those users who rated at least 50 items from ML and EM are determined; then, they are uniformly randomly divided into two disjoint sets, training and test sets for each data set. Finally, 1,000 and 500 users are uniformly randomly selected for training and testing from training and test sets, respectively. For each test user, five rated items are uniformly randomly chosen as test items. After withholding their true ratings, they are replaced with null; and tried to provide recommendations for them using the train users' data. It is assumed that data are distributed among *z* companies, where *z* might be one, three, five, or 10. Each data owner owns about *n*/*z* number of train users in HD2RP and *m*/*z* number of items in VD2RP. In HD2RP, it is decided to select those users as neighbors whose similarity weight with *a* is greater than 0.3, as proposed in (Herlocker et al., 1999). The experiments are run using MATLAB R2009b on a computer, which is Intel Core2Duo, 24.0 GHz with 4 GB RAM.

Experiment 1: First trials are performed to demonstrate how coverage changes with varying *z* values because amount of data involving in recommendation process increases with decreasing *z* values. Since coverage results for ML data set is shown in the previous section, the results for EM is shown only. Experiments are performed while varying *n* from 100 to 1,000. In other words, *z* is varied from 10 to one. In Figure 3.3, the average coverage values are presented for EM data set with varying *n* values.

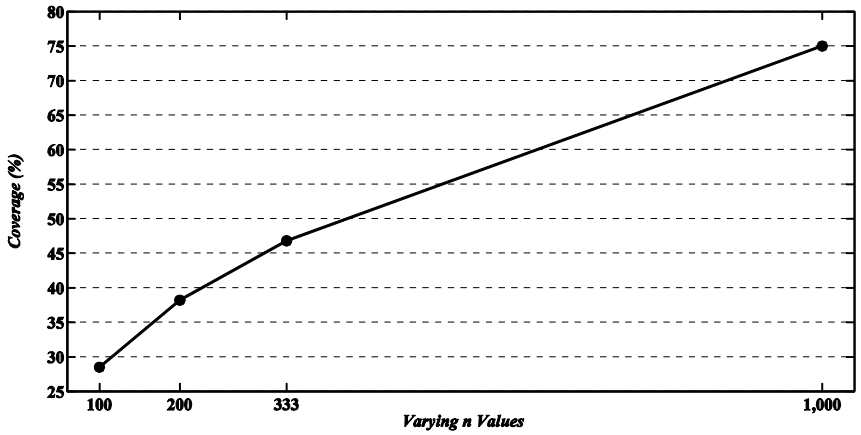

Figure 3.3. Coverage with Varying *n* Values

As expected, coverage significantly improves with increasing *n* values or decreasing *z* values, thus, by collaboration, the parties are able to provide predictions for more items. If *n* increases, amount of ratings involving in recommendation process also increases; that makes coverage better. As seen from Figure 3.3, when there are 1,000 users whose data are partitioned among 10 parties (on average, each party holds 100 users), average coverage for each party is about 27%. If they integrate their data, coverage values increase to 75%. Hence, it can be concluded that, for sparse data sets, collaboration among various parties definitely improves coverage. VD2RP scheme helps online vendors provide referrals for more items.

Experiment 2: Online vendors might be able to provide more accurate predictions if they combine their HDD. To show how precision changes due to combining HDD, trials are conducted using both data sets while varying *z* values. 1,000 users for training and 500 users for testing are used. Predictions are produced for all test items from train data. The experiments are run 100 times. During producing recommendation process, *k* is set at 40 as proposed by Herlocker et al. (1999). Then, recommendations are estimated from 1,000 train users (integrated data) and the split data among three, five, or 10 parties to show the effects of collaboration. The NMAE values are computed for both data sets and displayed them in Figure 3.4 for both data sets.

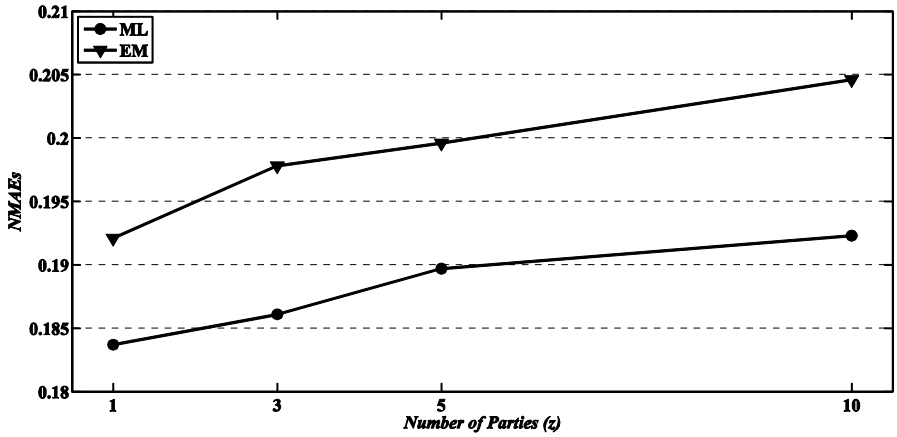

Figure 3.4. Effects of Collaboration in HD2RP Scheme

As seen from Figure 3.4, NMAE values improve due to collaboration of parties for both data sets. If data are distributed among 10 parties (n = 100) and they produce recommendations from their own data only, NMAE values are 0.1923 and 0.2046 for ML and EM, respectively. On the other hand, if these 10 parties collaborate (n = 1,000), their recommendations' accuracy improves to 0.1837 and 0.1921 for ML and EM, respectively. Hence, alliance definitely enhances the quality of the referrals. To show if the improvements due to partnership are statistically significant, *t*-tests are conducted. The values of t are 3.27 and 2.45 for ML and EM, respectively, where z is five. Those values are statistically significant for 99% confidence intervals for both data sets.

Experiment 3: To show how accuracy changes due to collaboration when data are vertically distributed, trials are performed using both data sets. Again 1,000 train users and 500 test users are used for experiments. Five test items are randomly selected for each test user. The experiments are run 100 times, where k is set at 40. After computing overall averages, the NMAEs are shown in Table 3.1. Note again that the last column in Table 3.1 shows the results for integrated data. First, second, and third columns represent the results for z being 10, five, and three, respectively.

Table 3.1. Effects of Collaboration in VD2RP Scheme

z	**1**	**3**	**5**	**10**
ML	0.1837	0.1876	0.1940	0.2042
EM	0.1921	0.2006	0.2072	0.2551

According to the results presented in Table 3.1, data owners provide more accurate results if they collaborate. With decreasing number of cooperating vendors (increasing number of m values), the quality of the referrals enhances. In case of split data, amount of data held by a single party only is not sufficient for offering precise recommendations. However, as seen from the empirical outcomes, the parties are able to offer better recommendations if they collaborate with each other. When 10 companies decide to work in groups, accuracy enhances from 0.2042 to 0.1837 for ML. The improvements for EM for the same case are even better. The outcomes verify that the hypothesis is true. To show how significant such improvements are, *t*-tests are employed. t values are found as 6.39

and 5.34, which are statistically significant for 99% confidence intervals, for ML and EM, respectively when *z* is 10.

Experiment 4: The proposed approach utilizes RP to achieve privacy and improve online performance through data reduction. Since it reduces high-dimensional data to *d*-dimensional, it is vital to determine the optimum value of *d* experimentally. To show how varying *d* values affect accuracy and find out its optimum value, trials are performed using both data sets while changing *d* values. 1,000 and 500 users for training and testing are used, respectively. The experiments are run 100 times for each *d* value with different **R** matrix; and computed user similarities from reduced data while producing referrals for each test item. Overall averages of NMAE are computed values and the online time in seconds required for estimating predictions. The outcomes are displayed in Table 3.2 and Table 3.3 for ML and EM, respectively.

Table 3.2. Effects of Varying *d* Values (ML)

d	**100**	**250**	**500**	**1,000**	**2,000**	**3,900**
NMAE	0.1882	0.1854	0.1845	0.1840	0.1836	0.1836
T(sec)	10	11	14	18	27	60

Table 3.3. Effects of Varying *d* Values (EM)

d	**50**	**100**	**200**	**400**	**800**	**1,600**
NMAE	0.2074	0.2048	0.1944	0.1994	0.1976	0.1921
T(sec)	8	8	9	10	15	27

As expected, online performance enhances with decreasing *d* values for both data sets. Due to data reduction, online efficiency of RP-based scheme improves. In addition to the enhancements in *T*, the results show that predictions with decent accuracy can be provided on reduced data. As seen from Table 3.2, the NMAE is 0.1836 for *d* = 3,900, while it is 0.1882 for *d* = 100. Although accuracy becomes worse by amount of 0.0046, online performance improves by a factor of six for the same case. The reason for enhancements in time is the fact that user-user similarities are estimated on reduced data. Similarly, the phenomenon why there are no noteworthy losses due to data reduction can be explained in terms of the nature of RP, which keeps statistical information of actual data. The similar

findings are obtained for EM, as well, as seen from Table 3.3. Although accuracy slightly becomes worse with decreasing *d* values, online performance improves. In terms of preciseness and online efficiency, 500 and 200 are selected as the optimum values of *d* for ML and EM, respectively.

After performing experiments to determine the effects of collaboration and figuring the optimum values of *d* for both data sets, various trials are conducted to demonstrate how the privacy-preserving measures affect the overall performance. In the following experiments, 1,000 and 500 train and test users are used, respectively and *d* is set at its optimum values.

Experiment 5: To show the effects of privacy solution in HD2RP scheme, experiments are performed using both data sets. τ value is proposed to use for determining neighbors instead of selecting *k* most similar ones, where it is set at 0.3, as proposed by Herlocker et al. (1999). Similarly, *d* is set at its optimum values for both data sets. The NMAE values of 0.1851 and 0.1956 for ML and EM are obtained, respectively. The results show that collaboration increases accuracy of produced recommendations. To show the significance of the improvements, *t*-tests are conducted and the results showed that the improvements are statistically significant for at least 95% confidence intervals.

Experiment 6: In the VD2RP scheme, PNFA is proposed to utilize in order to determine *a*'s neighbors. Thus, experiments are conducted to evaluate the effects of PNFA on accuracy for both data sets. In these experiments, *k* is varied from 20 to 160. Also the values of *z* are changed from one to 10. Note that when *z* is one, it means that the data held by a single party. In other words, the results for *z* being one are the base results for assessing PNFA. The trials are conducted 100 times. After finding overall averages, the outcomes are displayed in Figure 3.5 and Figure 3.6 for ML and EM, respectively.

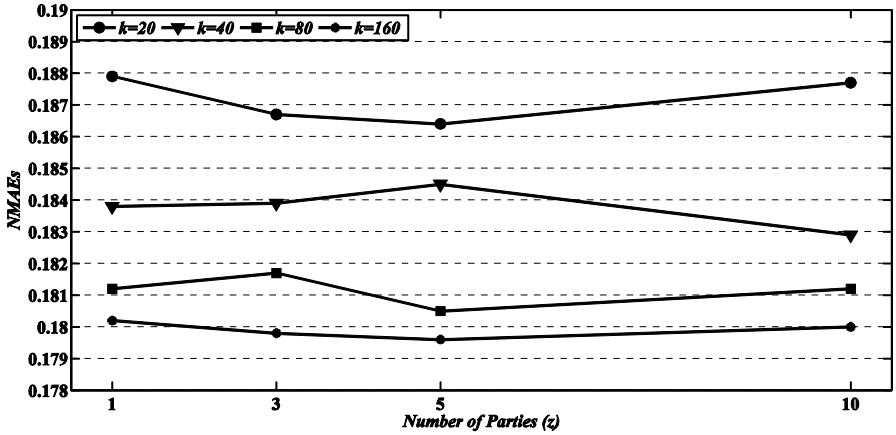

Figure 3.5. Effects of PNFA on Accuracy (ML)

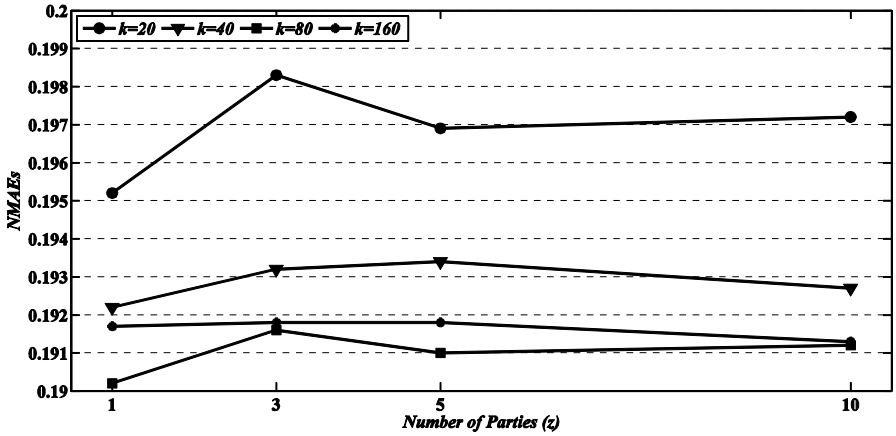

Figure 3.6. Effects of PNFA on Accuracy (EM)

According to experiment results shown in Figure 3.5 and Figure 3.6, the outcomes are very stable for varying *z* values for both data sets. In other words, PNFA algorithm is able to form consistent neighborhoods even if the number of collaborating parties increases. As seen from Figure 3.5 and Figure 3.6, the results usually become worse with decreasing *k* values. For both data sets, the outcomes are very promising for *k* values of 80 and 160. The best results occur when *k* is 80 for EM, while they are the best when it is 160 for ML. Compared to the base results (where *z* is one), the quality of the recommendations on PNFA are comparable for both data sets for larger *k* values. The proposed PNFA algorithm can be used to determine *a*'s neighbors without sacrificing on accuracy while preserving confidentiality.

***Experiment* 7**: After scrutinizing PNFA's effects on accuracy, trials are conducted to show how talented PNFA is to determine the best neighbors. In traditional *k-nn* prediction algorithms, the most similar *k* users are chosen as neighbors. Using PNFA, cooperating companies determine such neighbors without jeopardizing their privacy. 1,000 users are used for training while 500 users are used for testing. The trials are run 100 times while varying *z* and *k* values. The hit ratio representing the percentage of actual neighbors chosen by PNFA as the best similar users is calculated. In case of one party, *k* users are selected as neighbors. It is shown that what percentages of such *k* users are also chosen by PNFA, as well with varying *z* values. After calculating overall averages, the hit ratios are displayed as percent for both data sets in Table 3.4.

Table 3.4. PNFA's Hit Ratio (%) vs. *k* & *z*

z		**1**	**3**	**5**	**10**
***k*=20**	**ML**	100	69.58	62.63	59.68
	EM	100	66.92	63.43	61.33
***k*=40**	**MLM**	100	72.47	68.09	65.47
	EM	100	71.84	69.73	66.60
***k*=80**	**MLM**	100	76.42	73.98	70.65
	EM	100	77.46	76.21	72.50
***k*=160**	**MLM**	100	82.26	80.42	76.19
	EM	100	84.94	82.20	77.96

As seen from Table 3.4, PNFA's ability to determine the actual neighbors decreases with increasing *z* values, while it improves with increasing *k* values for both data sets. In conventional *k-nn* recommendation algorithms, the best 40 users are selected as neighbors. When *k* is 40 and *z* is three, PNFA is able to choose about 72% of actual neighbors for both data sets. If there are five vendors, 68% and 70% of true neighbors are selected by PNFA for ML and EM, respectively. Although the hit ratios of PNFA cannot be considered as promising in general; however, it is previously shown that accuracy losses due to PNFA are insignificant. Another set of experiments are performed to verify why PNFA is able to provide predictions with decent accuracy even though it has somehow

poor hit ratios. In a *k-nn* prediction algorithm, the best *k* users' data are used for estimating predictions. With increasing *k* values up to some point, changes in accuracy are very stable. Almost the same level of accuracy can be obtained even if it is used let say, uniformly randomly chosen 40 of the best 100 users. To verify this, experiments are conducted using both data sets while uniformly randomly selecting 40 nearest neighbors from the most similar 100 users to each test user. The trials are run 1,000 times and computed the overall averages. The NMAE values of 0.1838 and 0.1960 for ML and EM are obtained, respectively. The first 40 best similar are used users as neighbors; corresponding NMAE values are 0.1937 and 0.1921. For ML, almost the same results are obtained. Similarly, for EM, there are only 0.0039 deviations. Hence, it is concluded that similar outcomes can be offered even if some of the selected best users are used rather than the first *k* best users. This result strengthens the approach why utilizing PNFA does not cause significant accuracy losses.

Experiment 8: One of the controlling parameters that are used in the VD2RP scheme is *β.* In order to show the effects of *β,* experiments are conducted using both data sets while varying *β* from zero to 100. 1,000 and 500 users are used for training and testing, respectively. Predictions are estimated for all test items, where the trials are run 100 times. *k* is set at 40 and *d* is set at its optimum values for both data sets. After computing the overall averages, the final NMAE values are presented for both data sets in Figure 3.7.

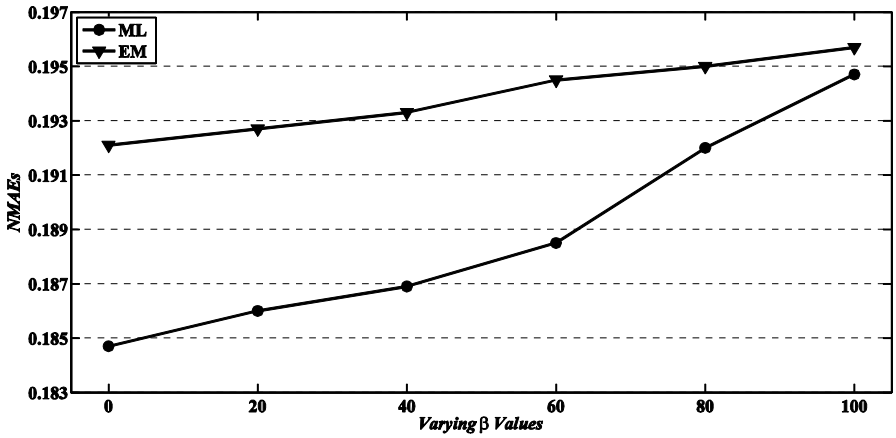

Figure 3.7. Accuracy vs. Varying *β* Values

As expected and seen from Figure 3.5, the quality of the recommendations diminishes with increasing β values due to augmenting randomness. However, such losses are very stable for both data sets for β values less than or equal to 60. Although accuracy becomes worse with increasing β values, confidentiality level improves. Since accuracy and privacy are conflicting goals, the parties can decide the value of β according to their requirements. In this sense, the scheme is flexible and allows data owners to determine the values of controlling parameters.

Experiment 9: Finally, trials are performed to show the joint effects of using PNFA and β using both data sets while varying z. β is set at 40, k is set at 40, and d is set at its optimum values. The best neighbors are selected utilizing PNFA and masked q's ratings based on β. The NMAEs are compared on split data only (No Collaboration-NC) and the ones on VD2RP. After running the trials 100 times, the overall averages are computed and displayed in Table 3.5 for both data sets.

Table 3.5. Overall Performance of the Proposed Schemes

z	**3**		**5**		**10**	
Scheme	**NC**	**VD2RP**	**NC**	**VD2RP**	**NC**	**VD2RP**
ML	0.1876	0.1849	0.1940	0.1861	0.2042	0.1882
EM	0.2006	0.1945	0.2072	0.1969	0.2551	0.2039

Note that if data owners collaborate without privacy concerns, NMAEs are 0.1837 and 0.1921 for ML and EM, respectively. According to results in Table 3.5, the proposed schemes provide more accurate results than the results on split data only. Due to collaboration without any privacy concern, accuracy enhances. Since confidentiality and preciseness are conflicting, accuracy slightly becomes worse due to the privacy-preserving measures. However, the results are still better than the ones on split data only. t values are found as 2.34, 5.13, and 9.89 for z is three, five, and 10, respectively for ML. They are 1.57, 2.06, and 8.47 for EM. According to t-test results, the improvements in accuracy for both data sets are statistically significant at least 95% confidence level.

3.7. Conclusions

Privacy-preserving schemes based on RP are presented to provide referrals on distributed data. To achieve data reduction and confidentiality, RP is proposed to utilize. In the schemes, it is assumed that data are vertically or horizontally

distributed between more than two parties. Collaboration is imperative. However, due to some reasons, vendors do not want to disclose their private data to each other. Schemes are proposed to offer recommendations on distributed data without deeply violating privacy.

It is shown that the schemes do not violate data owners' privacy. They can be used to offer predictions on distributed data without disclosing private data. Due to privacy measures, additional costs are inevitable. It is demonstrated that supplementary costs introduced by the schemes are negligible. The empirical results on two real data sets show that accuracy significantly improves due to collaboration. Although privacy measures make preciseness slightly worse, the gains are still statistically significant.

4. PRIVACY-PRESERVING TRUST-BASED RECOMMENDATIONS ON DISTRIBUTED DATA

Providing trust-based recommendations has been receiving increasing attention. Due to data sparsity in recommender systems, it is challenging to estimate similarities between any two entities and forming good neighborhoods. Researchers propose alternative approaches like examining trust between any two users from their rating profile. Although it is trivial to provide trust-based predictions when data collected for recommendation purposes held by a central server; however, it becomes a challenge to offer trust-based recommendations on distributed data between multiple parties due to privacy, legal, and financial reasons. Without privacy-preserving measures, data holders do not feel comfortable to estimate predictions on distributed data collaboratively. This chapter covers trust-based solutions enabling data owners' collaboration when producing recommendations on VDD or HDD.

4.1. Introduction

Traditionally, CF systems employ similarity metric to compute correlation between users and this correlation can be computed if they have commonly rated items. With increasing number of items, it is less likely to find co-rated products. To associate any two entities, researchers propose to utilize trust rather than similarity (Hwang and Chen, 2007; Massa and Avesani, 2007; Lathia et al., 2008; Walter et al., 2009). In order to investigate the relationship between trust and similarity, Ziegler and Goldbeck (2007) present frameworks and make empirical analysis. Exploiting trust in CF handles with problems caused by sparse data. Since it is possible to compute indirect trust value between any two users, it is possible to obtain correlation between them by trust propagation even if they have no commonly rated items (Hwang and Chen, 2007; Massa and Avesani, 2007). The methods proposed so far to estimate recommendations on trust networks can be used by central server-based CF systems (Hwang and Chen, 2007; Lathia et al., 2008). In addition to overcoming sparse data problems, utilizing trust networks for predictions also improves online performance because trust values between users can be computed off-line.

In social networks, reputations and implicit trust information are utilized while performing CF algorithms. Researchers introduce several solutions, which provide producing recommendations in a social network. Massa and Avesani (2004; 2005; 2007) introduce the implicit trust-based CF algorithms and its real world application. O'Donovan and Smyth (2005) show the importance of trustworthiness of users and present two computational models of trust. Walter et al. (2008) present a model, which uses agents to reach social network information and relationships to produce trust-based recommendations. Yuan et al. (2010) show that trust network is a small-world network and nodes are highly clustered. They propose solutions to overcome weaknesses of trust-aware recommender systems. Liu and Lee (2010) develop a solution to increase effectiveness of recommendation systems by utilizing user rating data with social network relationship. Besides implicit trust-based studies, researchers introduce methods for extracting trust values between users by utilizing their rating profiles (Hwang and Chen, 2007; Lathia et al., 2008). They also show that trust can be propagated so that it is possible to compute trust between two users having no commonly rated items.

In this dissertation, solutions are introduced for producing trust-based recommendations on VDD or HDD without jeopardizing data owners' confidentiality. The schemes first construct a trust network of the users off-line from distributed data while preserving their privacy. They then form neighbors for each user in the database off-line without greatly violating confidentiality. They finally estimate predictions with privacy in a distributive manner online. The solutions are evaluated in terms of privacy and it is shown that they preserve data owners' confidentiality. Online performance analysis displays that supplementary loads caused by privacy measures are insignificant. Real data-based experiments are conducted to assess the scheme in terms of the quality of the predictions. The outcomes demonstrate that the results are promising.

4.2. Trust-based Collaborative Filtering Algorithm

According to Hwang and Chen (2007), trust between two users having commonly rated items is computed by employing a simple prediction formula introduced by Resnick et al. (1994), which is shown in Eq. (4.1):

$$p_{aj}^{u} = \overline{v_a} + \left(v_{uj} - \overline{v_u}\right) \quad (4.1)$$

where $\overline{v_a}$ and $\overline{v_u}$ refer to the mean ratings of users *a* and *u*, respectively, and v_{uq} is the rating of item *j* given by *u*. Hwang and Chen (2007) employ this simple prediction calculation for computing trust between users, as follows:

$$t_{a\to u} = \frac{1}{\#(I_a \cap I_u)} \sum_{j\in(I_a \cap I_u)} \left(1 - \frac{\left|p_{aj}^{u} - v_{aj}\right|}{b}\right) \quad (4.2)$$

where I_a and I_u refer to the set of rated items of *a* and *u*, respectively, and *b* is the size of the rating range and $t_{a\to u}$ is trust value, which shows how much *a* trusts *u*. The authors propose trust propagation metric to compute trust between users having no co-rated items, as follows (Hwang and Chen, 2007):

$$t_{s\underset{v}{\to}h} = t_{s\to v} \oplus t_{v\to h} = \frac{\#(I_s \cap I_v) t_{s\to v} + \#(I_v \cap I_h) t_{v\to h}}{\#(I_s \cap I_v) + \#(I_v \cap I_h)} \quad (4.3)$$

where users *s* and *h* do not have commonly rated items but there is a user *v* who has co-rated items with both *s* and *h.* For each user *v,* the computations are performed and average of the inferred trusts is assigned for $t_{s\to h}$. After computing trust between users recommendations are produced, as follows:

$$p_{aq} = \overline{v_a} + \frac{\sum_{u\in NN} \left(v_{uq} - \overline{v_u}\right) \times t_{a\to u}}{\sum_{u\in NN} t_{a\to u}} \quad (4.4)$$

4.3. Trust-based Recommendations on VDD

In the proposed distributed data-based approaches, there are off-line and online computation phases. In both parts, data holders conduct some computations in a distributive manner while preserving their privacy. Overall steps of the proposed scheme can be described, as follows:

I. Off-line

i. Constructing Trust Network (Estimating Trusts)

a. Direct

b. Indirect-Trust Propagation

ii. Forming Neighborhoods

II. Online: Estimating Recommendations

In VDD scheme, since data is vertically distributed among multiple parties, the parties can construct sub-trust networks from their own data. Although data owners can construct sub-trust networks by themselves, they need to know each

user's rating mean and number of commonly rated items by any two users. Data holders can collaboratively compute them and then exchange them. Since such values are aggregates, sharing them with collaborating parties does not violate privacy constraints. Thus, the parties first calculate the mean values and the number of commonly rated items between any two users off-line with privacy.

Computing Row Mean with Privacy: For a user *u*, her average rating ($\overline{v_u}$) can be computed, as follows:

$$\overline{v_u} = \frac{\sum_{j \in I} v_{uj}}{|I|} \tag{4.5}$$

where *I* shows the set of ratings *u* has, v_{uj} represents *u*'s rating for item *j*, and |*I*| shows the length of set *I*. Since data are distributed among *z* parties vertically, Eq. (4.5) can be written, as follows:

$$\overline{v_u} = \frac{\sum_{j \in I_1} v_{uj} + \sum_{j \in I_2} v_{uj} + \cdots + \sum_{j \in I_z} v_{uj}}{|I_1| + |I_2| + \cdots + |I_z|} \tag{4.6}$$

where I_g shows the set of user *u*'s ratings held by the party *g*. As seen from Eq. (4.6), each party *g* finds interim aggregate results ($\sum_{j \in I_g} v_{uj}$ and $|I_g|$ values) for all *u* = 1, 2,…, *n*. After exchanging them with all collaborating parties, each company then computes $\overline{v_u}$ values for each user *u*.

Computing Number of Commonly Rated Items with Privacy: Data owners can collaboratively compute number of co-rated items between any two users *a* and *u* ($\#_{au}$), as follows:

$$\#_{au} = \#(I_a \cap I_u)_1 + \#(I_a \cap I_u)_2 + \cdots + \#(I_a \cap I_u)_z \tag{4.7}$$

After computing interim aggregates, the parties exchange them and find $\#_{au}$ values for all any two users in their databases. Notice that the parties cannot derive ratings and the rated items held by each other from such exchanged interim aggregate values during the above computations. The parties can now compute trust values between any two users based on commonly rated items explained in the following.

***Privacy-Preserving Direct Trust Computation on VDD* (*PPDTCV*)**: If any two users have commonly rated items, trust between them can be calculated using direct trust computation given in Eq. (4.1) and Eq. (4.2). Since the parties know user mean votes, they can compute p_{aj}^{u} values by themselves. On the other hand,

due to VDD, they can compute direct trust value between users a and u ($t_{a\to u}$), using Eq. (4.2), as follows:

$$t_{a\to u} = \frac{1}{\#_{au}}\left[\sum_{j\in(I_a\cap I_u)_1}\left(1-\frac{\left|p_{aj}^{u}-v_{aj}\right|}{b}\right)+\sum_{j\in(I_a\cap I_u)_2}\left(1-\frac{\left|p_{aj}^{u}-v_{aj}\right|}{b}\right)+\cdots+\sum_{j\in(I_a\cap I_u)_z}\left(1-\frac{\left|p_{aj}^{u}-v_{aj}\right|}{b}\right)\right] \quad (4.8)$$

Remember that data holders have $\#_{au}$ values estimated before. Due to vertical distribution, each party g can compute $\frac{1}{\#_{au}}\sum_{j\in(I_a\cap I_u)_g}(1-\frac{\left|p_{aj}^{u}-v_{aj}\right|}{b})$ values by itself. Note that if $\#_{au}$ is zero, then corresponding partial trust is also zero. After computing such interim aggregates, each party g then saves them in an $n \times n$ adjacency matrix $\mathbf{AD_g}$, which shows partial trust values between users. Note that computing partial direct trust values does not violate data holders' confidentiality. When they sum such partial trust values without exposing their privacy, they obtain the trusts between users. In other words, $\mathbf{AD} = \sum_{g=1}^{z}\mathbf{AD_g}$.

If partial trust values for any two users a and u are zero, then it means that these two users have no co-rated items and there is no direct edge between them in trust network. To compute trust between users having no commonly rated items, data owners utilize trust propagation metric given in Eq. (4.3), as follows, where they use adjacency matrix in order to exploit direct trust values:

***Privacy-Preserving Trust Propagation on VDD* (*PPTPV*)**: The parties can compute trust propagation, as follows:

i. First, collaborating companies determine those user pairs having no co-rated items by scanning their adjacency matrices.

ii. After determining such user pairs, the parties then search out possible paths between such users. They can find the possible paths from intersection of each user's directly connected users. For example, suppose that there is no edge between users a and u. Also, assume that there is an edge between users a and u_1; and an edge between u_1 and u. Then, the possible path might be $a\to u_1\to u$.

iii. Next, data owners compute interim trust values for such disjoint users in a distributive manner using Eq. (4.3), as follows:

$$t_{s\underset{v}{\rightarrow}h} = t_{s\rightarrow v} \oplus t_{v\rightarrow h} = \frac{\sum_{g=1}^{z}\left((\#(I_s \cap I_v)_g \, t_{s\rightarrow v_g} + \#(I_v \cap I_h)_g \, t_{v\rightarrow h_g}\right)}{\sum_{g=1}^{z}\left((\#(I_s \cap I_v)_g + \#(I_v \cap I_h)_g\right)} \quad (4.9)$$

iv. Finally, they perform the previous step for each user providing a path between any two disjoint users; and each data owner *g* stores average of these partial indirect trust values into the corresponding cells of the adjacency matrix $\mathbf{AD_g}$.

Since each data owner g knows $\#(I_s \cap I_v)_g$ and $\#(I_v \cap I_h)_g$ values, and partial $t_{s\rightarrow v_g}$ and $t_{v\rightarrow h_g}$ values in their own trust network; they can compute partial indirect trust values between user *s* and *h* if user *v* provides a path between them.

Now each party *g* has $\mathbf{AD_g}$ including partial trust values between any two users in the network. In other words, trust network, constructed off-line, is distributed among them. Once they have partial trust values, they now can form neighborhoods for each user in the network using such sub aggregates without jeopardizing their privacy.

***Privacy-Preserving Neighborhood Formation on VDD* (*PPNFV*)**: The second step in a traditional trust-based prediction scheme is forming neighborhoods. After computing partial trust values between users, the parties can now determine each user's neighbors by selecting the most trusted *k* users based on trust values. Note that trust values are distributed among *z* parties. The following protocol is proposed, referred to as distance-based private sorting algorithm (DPSA), in order to sort users based on distributed trust values without disclosing partial trusts. In the following, how to find neighbors for any user *a* is explained, where it is assumed that *a* is the first user in the adjacency matrix **A**:

i. Each vendor *g* computes the average of the partial trust values between *a* and each user *u* in the trust network, as follows:

$$\overline{t_{ag}} = \frac{\sum_{u=1}^{n-1} t_{a\rightarrow u_g}}{n-1}$$

In other words, each party g computes mean of the first row of their adjacency matrix $\mathbf{AD_g}$.

ii. Then, each company g normalizes $t_{a\to u_g}$ values using deviation from approach and obtains $t'_{a\to u_g} = t_{a\to u_g} - \overline{t_{ag}}$ values. These differences show how much partial trust values deviate from average.

iii. After each vendor g calculates $t_{ag} = \sum_{u=1}^{n-1} \left| t'_{a\to u_g} \right|$ values, they then normalize $t'_{a\to u_g}$ values, as follows:

$$t''_{a\to u_g} = \frac{t'_{a\to u_g}}{t_{ag}}$$

iv. After exchanging $t''_{a\to u_g}$ values with collaborating companies, each party g finds the sum of the normalized trust values between a and for all users $u = 2, 3, \ldots, n$, as follows:

$$t''_{a\to u} = \sum_{g=2}^{z} t''_{a\to u_g}$$

v. Finally, they sort these sums in descending order and select the first k users as the most trusted users for a.

During off-line process, the parties utilize DPSA protocol n times to determine neighbors for all n users. DPSA helps data owners sort distributed trust values. Note that each party g first normalizes its partial trusts using deviation from mean approach. They then normalize them again by dividing them with the sum of the normalized partial sums. Also, notice that partial sums range from 0 to 1. This type of normalization does not change the order of the sum of such distributed values.

Private Recommendation Protocol on VDD (PRPV): After off-line computations, the parties can now provide recommendations online using private recommendation protocol (PRP). In this protocol, HE scheme is utilized for protecting data owners' confidential data. The parties follow the following steps to estimate a trust-based prediction to a for item q (p_{aq}), as follows:

i. a sends a prediction request for q to the MC. Suppose that the first party is MC.

ii. MC first informs collaborating parties about a.

iii. Then, MC computes $V_{uq} = \left(v_{uq} - \overline{v_u}\right)$ values for all u = 1, 2, …, k users who are a's neighbors.

iv. It then encrypts V_{uq} values using an HE scheme and its public key K_{MC}; and obtains $\xi_{K_{MC}}(V_{uq})$ values. After that, it sends them to the collaborating parties.

v. Using an HE scheme, each party g computes $\xi_{K_{MC}}(Q_g) = \prod_{u=1}^{k}(\xi_{K_{MC}}(V_{uq}))^{t_{a\to u_g}}$ and $\xi_{K_{MC}}(T_g) = \prod_{u=1}^{k}(\xi_{K_{MC}}(t_{a\to u_g}))^{1}$ values, where k is the set of a's neighbors and $t_{a\to u_g}$ is the partial trust value held by g.

vi. Then, each party sends such encrypted partial aggregates to MC.

vii. Meanwhile, MC computes $Q_1 = \sum_{u=1}^{k}(V_{uq} \times t_{a\to u_1})$ and $T_1 = \sum_{u=1}^{k} t_{a\to u_1}$ values.

viii. MC then decrypts the received encrypted aggregates using its corresponding private key and obtains Q_g and T_g values for all g = 2, 3, …, z.

ix. Finally, MC computes $p_{aq} = \overline{v_a} + \frac{\sum_{g=1}^{z} Q_g}{\sum_{g=1}^{z} T_g}$, and returns it to a.

4.4.Trust-based Recommendations on HDD

In HDD, data holders first compute direct and indirect trust values among their own users by employing Eq. (4.1) – (4.3). Thus, they construct their own users' trust network. Since data holders have users' whole rating data, they do not need to collaborate at this stage. Unlike VDD, HDD does not require collaboration while computing $\overline{v_u}$ for any user u, because required data are held by the related vendor. After constructing their own trust networks for those users held by the same company, any pair of parties follow the below protocol for computing direct trust among the users held by them.

***Privacy-Preserving Direct Trust Computation on HDD* (*PPDTCH*)**: Suppose that A and B represent any two vendors want to construct trust network between users held by them. A and B follow this protocol to compute trust values between any pair of users a and u held by A and B, respectively. Since trust

computation has distributive property, collaborating parties can compute partial trust values based on vertically split data. Therefore, each party divides their own data into two halves vertically, referred to as left (*L*) and right (*R*). *A* holds sub trust computed from *L*, while *B* holds the one inferred from *R*. This way, the trust values are distributed between them. Hence, Eq. (4.2) can be written as follows:

$$t_{a\to u} = t_{u\to a} = t_{aL\to uL} + t_{uR\to aR} = \frac{1}{\#(I_a \cap I_u)} \left(\begin{array}{c} \sum_{j\in(I_{aL} \cap I_{uL})} \left(1 - \frac{\left|p_{aj}^{u} - v_{aj}\right|}{b}\right) + \\ \sum_{j\in(I_{aR} \cap I_{uR})} (1 - \frac{\left|p_{uj}^{a} - v_{uj}\right|}{b}) \end{array} \right) \quad (4.10)$$

where $t_{aL\to uL}$ and $t_{uR\to aR}$ show the sub-trust values computed from *L* and *R* respectively. Notice that user *a*'s ratings are held by *A*, while user *u*'s ratings are held by *B* due to horizontal distribution. I_{aL}, I_{uL}, I_{aR}, and I_{uR} refer to the sets of rated items of corresponding part of *a* and *u* in *L* and *R*. If Eq. (4.10) is extended, then Eq. (4.11) is obtained, as follows:

$$t_{a\to u} = \frac{1}{\#(I_a \cap I_u)} \left(\sum_{j\in(I_{aL} \cap I_{uL})} \left(1 - \frac{\left|\overline{v_a} + (v_{uj} - \overline{v_u}) - v_{aj}\right|}{b}\right) + \sum_{j\in(I_{aR} \cap I_{uR})} (1 - \frac{\left|\overline{v_u} + (v_{aj} - \overline{v_a}) - v_{uj}\right|}{b}) \right) \quad (4.11)$$

If Eq. (4.11) is simplified by assigning $v'_{uj} = \left(\frac{v_{uj} - \overline{v_u}}{b}\right)$ and $v'_{aj} = \left(\frac{v_{aj} - \overline{v_a}}{b}\right)$, Eq. (4.12) is get, as follows:

$$t_{a\to u} = \frac{1}{\#(I_a \cap I_u)} \left(\sum_{j\in(I_{aL} \cap I_{uL})} \left(1 - \left|v'_{aj} - v'_{uj}\right|\right) + \sum_{j\in(I_{aR} \cap I_{uR})} \left(1 - \left|v'_{uj} - v'_{aj}\right|\right) \right) \quad 4.12)$$

In *PPDTCH* protocol, the parties perform the required computations in Eq. (4.12) to compute $t_{a\to u}$ between each pair of users *a* and *u* in their database, as follows:

i. The parties need to determine commonly rated items of *a* and *u*. Hence, *A* and *B* employ efficient and secure protocol for determining set intersection of two parties private data without jeopardizing their privacy proposed by Sang and Shen (2009). The protocol helps two data owners determine the common items in their databases without violating their privacy.

ii. For the left part of their data, the parties perform the following steps:

a. *A* computes v'_{aj} values based on the data in left part of its data; and sends them to *B* after encrypting them with its own public key K_A using a HE function ξ.

b. *B* similarly computes v'_{uj} values based on the data in left part of its data. It then multiplies them with -1 and encrypts the results with K_A using an HE function ξ. After that *B* computes $\xi_{K_A}(v'_{aj}) \times \xi_{K_A}(v'_{uj}) = \xi_{K_A}(v'_{aj} - v'_{uj})$ for each item *j* using the HE property to find the required values in encrypted form in Eq. (4.12).

c. *B* permutes the encrypted values found in previous step using a permutation function F_B. Then, *B* uniformly randomly chooses θ percent of them and multiplies them with -1 using HE scheme. Note that multiplying -1 does not change the outcome of taking absolutes, but rather improves privacy.

d. Next, *B* sends them to *A*.

e. Since *A* knows the related decryption key, it decrypts the received encrypted values and finds their absolutes. *A* then subtracts each value from 1 and computes $\sum_{j \in (I_{aL} \cap I_{uL})}(1 - |v'_{aj} - v'_{uj}|)$; it then finally divides it by $\#(I_a \cap I_u)$.

iii. For the right part of their data, the parties perform the same steps while they switch their roles.

Any two party, holding different users' data and want to estimate the trust between them, can conduct PPDTCH protocol, as explained before. After they employ this protocol, trust values between two users held by different parties are partitioned between two data owners. When there are commonly rated items between any two users, trust values between them can be computed using PPDTCH protocol. However, there might be users who do not share any commonly rated items, which might lead incomplete trust network. Therefore, data owners search their sub-trust networks and determine the pair of users having no trust value. In this case, for such users, they then follow trust propagation protocol described in the following.

***Privacy-Preserving Trust Propagation on HDD* (*PPTPH*)**: When there are common ratings between any two users, the trust value between them can be estimated using PPDTCH protocol without violating their privacy. If there is no any commonly rated item between them, they then follow the privacy-preserving trust propagation on HDD (PPTPH) protocol, as follows:

i. By scanning the sub-trust networks constructed after direct trust computation, the parties determine those user pairs having no co-rated items.

ii. After determining such user pairs, the parties then search out all possible paths between them. They can find those users who have common ratings with these two users. Since there are different numbers of users having common ratings with them, there might be various paths depending on such number. Therefore, they determine all possible paths for any two users having no common ratings.

iii. According to number of users sharing commonly rated items between those two users, Eq. (4.3) can be written in a distributive manner. For example, if it is assumed that user u_1 constructs a path between *a* and *u*, then *A* computes the following partial trust value using Eq. (4.3):

$$t_{aL\rightarrow uL} = \frac{\#\left(I_a \cap I_{u_1}\right) t_{aL\rightarrow u_1 L} + \#\left(I_{u_1} \cap I_u\right) t_{u_1 L\rightarrow uL}}{\#\left(I_a \cap I_{u_1}\right) + \#\left(I_{u_1} \cap I_u\right)}$$

where $t_{aL\rightarrow uL}$shows partial trust value computed from left half of rating vectors of *a* and *u*.

iv. Similarly, *B* computes the following:

$$t_{aR\rightarrow uR} = \frac{\#\left(I_a \cap I_{u_1}\right) t_{aR\rightarrow u_1 R} + \#\left(I_{u_1} \cap I_u\right) t_{u_1 R\rightarrow uR}}{\#\left(I_a \cap I_{u_1}\right) + \#\left(I_{u_1} \cap I_u\right)}$$

where $t_{aR\rightarrow uR}$ shows partial trust value computed from right half of rating vectors of *a* and *u*.

v. After finding all necessary trust propagations, the parties now have the complete trust network. Notice that when *a*, u_1, and *u* are held by the same party, it is an easy task to compute trust propagation. However, when *a* and *u* are held by any two parties, *A* and *B*; and u_1 is held by one

of them, the parties can estimate partial trust values between a and u_1 or u_1 and u using the PPDTCH protocol.

***Privacy-Preserving Neighborhood Formation on HDD* (*PPNFH*)**: After computing direct and indirect trust values between any two users, each party g has partial trust values between its users and other users in the network. Once parties have partial trust values, they now need to determine the *NN* to use in online recommendation process. Since data distribution configuration in HDD is different from VDD, the private neighborhood formation protocol proposed for VDD does not work for HDD. In traditional CF algorithms, there are two widely used schemes for selecting the best similar users. The first method chooses k most similar users while the second one chooses those users whose similarity weights are greater than τ. In the HDD-based schemes, the neighborhood is formed by utilizing the latter approach. The idea behind setting a threshold for trust values is identical with the one proposed by Herlocker et al. (1999). Since trust values between users are partitioned, the parties follow the steps below in order to determine whether the sum of partial trust values is greater than or equal to τ or not without jeopardizing their privacy. The scheme is explained in terms of A only. Other companies can follow the same steps.

i. Suppose that the collaborating parties agree on an interval over which the parties can uniformly randomly select the related threshold value τ. Since the trust values are between 0 and 1, the parties can choose the threshold values over the range [0.5, α], where $\alpha < 1$. Thus, A first uniformly randomly selects τ_A over the range [0.5, α].

ii. To determine the neighbors of its users, A considers each user in its database as an active user a. A subtracts all partial trust values for each user a in its data from τ_A. There might be three possible cases for each user u in **D** being a neighbor of a:

a. If the result is less than or equal to zero, then that partial trust is already greater than or equal to τ_A; it means that u is one of a's neighbors.

b. If the result is greater than zero, then A and the party having the partial trust value of user u employ the solution for Yao'

millionaires' problem proposed by Shundong et al. (2008) to determine whether partial trust value in other party is greater than or equal to subtracted value in *A*. If partial result is greater than or equal to the subtracted value, then *u* belongs to *a*' neighborhood.

c. Otherwise, *u* does not belong to *a*'s neighborhood.

iii. *A* repeats step 2 for all users in its database.

All parties follow the same steps for determining their users' neighborhoods. In order to improve privacy, the parties might utilize different thresholds for each user.

The computations conducted so far are done off-line. Compared to online costs, off-line costs are not that critical for the overall performance. The parties now can provide predictions based on trust values.

***Private Recommendation Protocol on HDD* (*PRPH*)**: In off-line computations, data holders make required computations for online recommendations; and when an active user *a* requests a prediction for an item *q*, the parties follows the following steps:

i. *a* sends a prediction request for *q* to the MC.

ii. MC first informs the collaborating parties about *a* and *q*; and computes $\sum_{u \in NN} \left(v_{uq} - \overline{v_u}\right) \times t_{a \to u}$ and $\sum_{u \in NN} t_{a \to u}$ for each user *u* in *NN* in its own data.

iii. Each party g computes $\sum_{u \in NN} \left(v_{uq} - \overline{v_u}\right) \times t_{aL \to uL_g}$ and $\sum_{u \in NN} t_{aL \to uL_g}$ if it holds information for *L* and *a*; otherwise, it computes $\sum_{u \in NN} \left(v_{uq} - \overline{v_u}\right) \times t_{aR \to uR_g}$ and $\sum_{u \in NN} t_{aR \to uR_g}$. It then sends the partial results to MC.

iv. MC computes p_{aq} from received interim results and sends the prediction to *a*.

4.5. Privacy Analysis

In this section, the proposed schemes are analyzed in terms of privacy. It is explained why the schemes do not jeopardize data holders' confidentiality in

terms of the criteria determining how an algorithm enforces the main goals of PPDM, HF and data quality.

In the scheme for VDD, firstly mean values are estimated based on distributed data. During privacy-preserving mean computation, data owners exchange aggregate data (sum and number of ratings) without introducing any randomness. Given an aggregate, which is the sum of m_r values, it is not possible to learn such values. Since the parties also exchange m_r, the probability of guessing the rated items is 1 out of $Comb_{m_r}^{m_t}$, where m_r and m_t are the number of rated items and total items held by one company. While estimating commonly rated items between any two users, the parties again switch aggregate data with no randomness. From number of commonly rated items, the parties can guess the rated items with a probability given above. Since such probabilities are very low, it can be concluded that HF is also very low. Data quality does not change, because data owners do not add randomness.

As explained previously, during partial trust computations (either direct or indirect) in VDD, vendors do not exchange any data. Therefore, HF is zero and data quality does not change. In DPSA protocol, data owners exchange normalized values. It is not possible to derive true partial trust values from received normalized quantities without knowing the average and sum of the absolute values of the deviations. Hence, DPSA does not violate the privacy constraints. The parties can only learn the sorting of partial trust values. This does not help them find out exact partial trusts. Finally, the scheme utilizes PRPV, which is basically based on HE. There is no information leakage during PRPV. Moreover, due to encryption, which does not add randomness to original data, data quality is preserved.

In HDD solution, the parties employ PPDTCH protocol to compute direct trust values in which a secure set intersection method proposed by Sang and Shen (2009) is utilized. The method is based on evaluating a randomized polynomial Y whose roots set contains the intersection. The authors propose to employ building blocks against possible attacks. According to their analysis, the secure set intersection protocol does not violate data holders' confidentiality. On the other hand, at the end of the secure set intersection protocol, the parties learn their

users' commonly rated items, which can be named as hiding failure of PPDTCH protocol. Other computations in PPDTCH are computed using an HE; and since HE is secure, it does not introduce any hiding failure and also it does not affect data quality. Also, permutation and multiplication by -1 prevents revealing sum of v'_{uj} or v'_{aj} values, which can be harmful for data holders. Besides strengthen privacy level of users, these operations do not affect final results of computations.

In PPTPH protocol, the parties work on their private data, thus, there is no privacy risk in this protocol. PPNFH protocol depends on the selected τ value range, which is determined empirically and to enhance privacy level of parties, a wide range is determined for τ. With increasing range and precision, it becomes difficult to guess exact τ values. The parties employ a secure solution to Yao's millionaires' problem (Shundong et al., 2008) for determining the greater value from given two private numbers. The solution for Yao's millionaires' problem utilizes pseudo-random sequence, guessing permutation function of the sequence very low and it cannot be broken in polynomial time. Thus, employing the Yao's millionaires' problem solution (Shundong et al., 2008) does not cause any hiding failure and also as parties do not add any randomness, data quality is preserved. However, depending on selection of τ, accuracy of scheme might be affected. In the experiments section, the effects of different τ values on accuracy are shown. Although PRPH protocol does not include any randomness or encrypted computations, it does not violate data owners' confidentiality. Since the parties send aggregates, which are the sum of multiple values, it is not possible to derive any information from a single aggregate, as explained previously.

4.6. Supplementary Costs Analysis

The methods proposed for both VDD and HDD cause extra storage costs because each company need to save partial trust values. Hence, supplementary storage costs are in the order of $O(n^2z)$, where note that z is a small constant. In a traditional trust-based recommendation scheme, a asks a prediction and the system returns a result. Thus, total number of communications during online phase is two only. In the proposed methods, on the other hand, MC sends a message to z-1 companies and they return some aggregates to MC during PRPV and PRPH protocols. Hence, additional communication costs due to the proposed

scheme are 2*z*-2 or in the order of $O(z)$. In other words, communication costs increase by a factor of *z*-1 times. Although the method introduces extra communication costs during online process, making such communications simultaneously reduces their effects to online performance.

Collaborating companies compute trust values, construct trust network, and form neighborhoods off-line in the scheme. The protocols run in online phase are PRPV and PRPH protocols. In a central server-based scheme, $O(k)$ multiplications are conducted during online phase, where k is the number of a's neighbors. In PRPV, however, there are $O(kz)$ multiplications, $O(kz)$ exponentiations, $O(k)$ encryptions, and $O(z)$ decryptions performed online. Note that z is a constant and multiplications and exponentiations are performed concurrently by collaborating parties. Since both k and z are small constants, additional computation costs due to cryptographic functions are small. In PRPH, all computations are performed parallel, thus, $O(k)$ computations in a central server-based scheme are done by z parties in parallel.

Due to privacy, additional costs are inevitable. Protecting privacy and providing predictions efficiently are two conflicting goals. Improving one makes the other worse. The proposed schemes cause additional online costs. However, they are negligible and they still make it possible to offer recommendations efficiently. Moreover, providing trust-based predictions while preserving privacy outweigh the performance losses caused by the proposed schemes. To sum up, the proposed schemes still are able to offer predictions efficiently even if they introduce inevitable extra costs.

4.7. Accuracy Analysis: Experiments

To evaluate the overall performance of the method, several experiments are conducted using ML data set. To measure the quality of the recommendations, MAE is employed. Also, average relative error (ARE) is utilized to show the percentage of improvements due to collaboration. In addition to MAE and ARE, a reliability measure is used, as follows: Reliability can be defined as the average number of commonly rated items by a and train users in the trust network. In order to demonstrate how collaboration improves coverage, coverage metric is

utilized. Finally, *t*-tests are applied in order to evaluate the schemes in terms of statistical significance.

Firstly, those users who rated at least 50 movies are determined. 1,000 users are uniformly randomly chosen among them. Leave-one-out methodology is used in the experiments. In this methodology, each user in the data set acts as *a* and the remaining users are used as training users. For each test user, five rated items are uniformly randomly selected as test items, their entries are replaced with null, withheld their true votes; and predictions are produced for them. After recommendations are estimated, they are compared with the true withheld votes and MAE and ARE values are calculated. The experiments are performed 100 times using randomly selected 1,000 users and test items. According to Hwang and Chen's empirical results (Hwang and Chen, 2007), the best MAE is obtained when the most trusted 70 users are assigned as the nearest neighbors; and one, two, or three trust propagation level provides similar results, one level is used. Thus, in the experiments, one level and the most trusted 70 users are used.

Experiment 1: Experiments are performed to show how collaboration affects the quality and the reliability of predictions. When data are horizontally distributed, it becomes a challenge to form good neighborhoods and find enough similar users. Likewise, when data are vertically distributed, trust values estimated on available data might be unreliable and they may not be accurate enough due to scarce available ratings. Then, neighbors determined based on such values might be untrustworthy. Finally, the quality of recommendations generated on selected neighbors' data might be inaccurate and unreliable. Data are assumed to be distributed among *z* companies, where *z* is varied from 1 to 10. As a result, each data owner owns about 3,900/*z* number of items belonging to 1,000 users for VDD and 1,000/*z* users for HDD when data are distributed. The same methodology is followed and overall averages of MAE values are computed. Results are displayed in Figure 4.1 for both VDD and HDD.

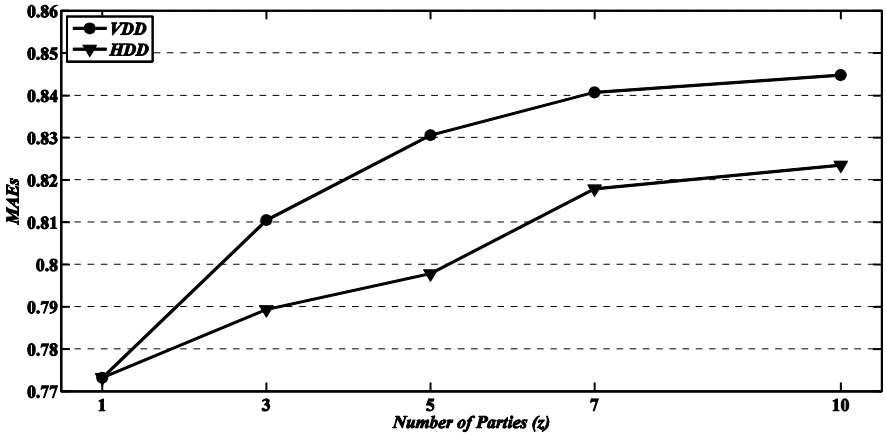

Figure 4.1. Effects of Collaboration on Accuracy with Varying *z* Values

As seen from Figure 4.1, the quality of recommendations improves with decreasing *z* values. In other words, collaboration among vendors significantly enhances accuracy. Notice that when data are distributed among 10 parties vertically, MAE is about 0.8448 while it is about 0.7732 if they decide to provide referrals on integrated data. Similarly, MAE improves from 0.8235 to 0.7732 if 10 parties decide to collaborate when data are distributed among them horizontally. Thus, it can be concluded that collaboration makes accuracy better. Accuracy improvements are stable while changing *z* from 10 to 7. However, such improvements are significant if *z* is changed from seven to five, three, or one. To show if the improvements due to partnership are statistically significant, *t*-tests are conducted. The values of *t* are 3.75 and 2.62 for VDD and HDD, respectively, where *z* is five. Those values are statistically significant for 99% confidence intervals for both partitioning cases. The *t*-test results for other *z* values are similar and they show that the schemes through collaboration significantly improve accuracy of recommendations while preserving data owners' privacy.

In addition to MAE, ARE values are estimated as percent. The percentage of improvements due to collaboration is wanted to show. For this purpose, overall averages of AREs are computed and displayed in Figure 4.3 for both VDD and HDD. As seen from Figure 4.2, working jointly helps the quality of predictions improve. For smaller *z* values, AREs are larger. With increasing *z* values, improvements due to collaboration become stable. When *z* is changed from five to one (or five companies decide to collaborate), accuracy improves by about 6.91%

and 3.08% for VDD and HDD, respectively. To sum up, if data owners decide to provide trust-based predictions on integrated data, preciseness definitely improves due to collaboration.

Reliability values with varying *z* values in VDD schemes are also estimated. Notice that reliability does not change when data are horizontally distributed. Hence, experiments are performed for VDD only to evaluate the schemes in terms of reliability. The same methodology is followed in which *z* is varied from one to 10. After computing reliability values, they are displayed in Figure 4.3. As seen from Figure 4.3 and as expected, number of commonly rated items increases if the parties decide to collaborate. When data are held by a single party (or the parties collaborate), reliability enhances significantly. It becomes more likely to find larger number of commonly rated items. Trust values estimated direct or trust propagation then become more reliable. When data are vertically distributed, it becomes a challenge to find commonly rated products. Thus, besides accuracy, collaboration also improves reliability.

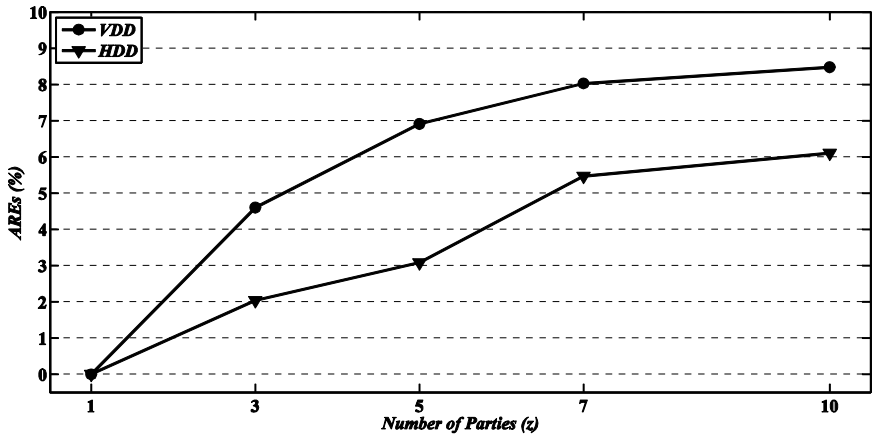

Figure 4.2. ARE Values with Varying *z* Values

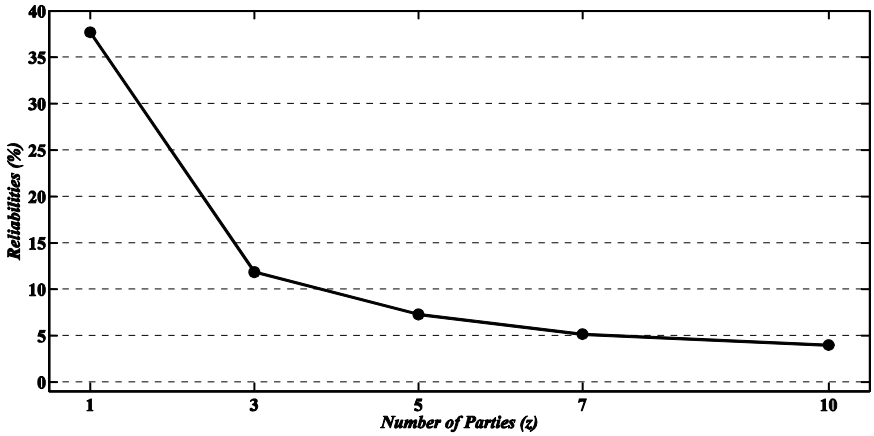

Figure 4.3. Effects of Collaboration on Reliability with Varying *z* Values

Experiment 2: Due to the nature of the privacy-preserving measures that are employed in the schemes, in general, do not affect accuracy. The only privacy-preserving measure that might affect the quality of the predictions is utilizing variable τ values in HDD-based schemes. Hence, finally, experiments are conducted to demonstrate how varying τ values affect accuracy. Notice that τ is uniformly randomly selected over an interval by data owners to protect their privacy when data are horizontally distributed. As explained previously, τ values can be uniformly randomly chosen over the range [0.5, α]; and observed the changes in accuracy. The same methodology is followed and overall averages are computed for varying α value from 0.5 to 0.9. Results are displayed in Figure 4.4. According to results shown in Figure 4.4, varying α value from 0.5 to 0.7 does not affect accuracy too much. The changes on accuracy are stable. However, for α values larger than 0.7, accuracy significantly becomes worse. Hence, it is suggested choosing τ values between 0.5 and 0.7. Since selecting those values as τ does not diminishes accuracy, it is possible to produce recommendations based on a trust threshold value.

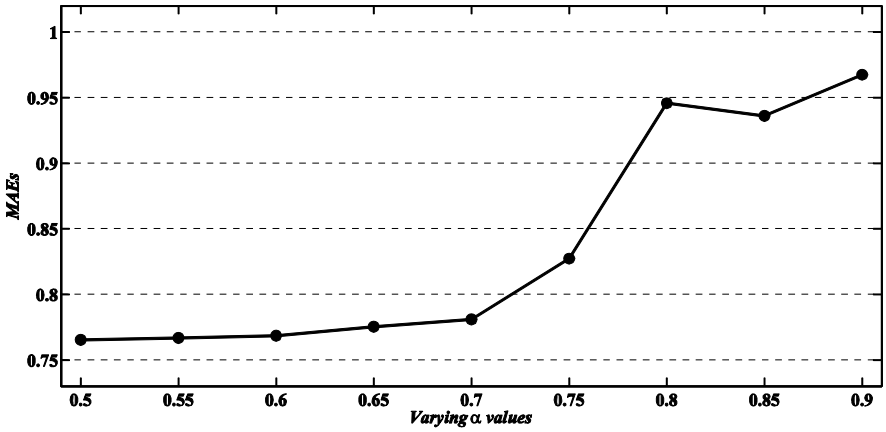

Figure 4.4. Effects of Varying α Values on Accuracy

4.8. Conclusions

Privacy-preserving schemes are presented in order to provide trust-based recommendations from distributed data. The methods help data owners construct trust networks off-line without violating their confidentiality. They can also determine the neighborhoods for each user in their trust network off-line. Like off-line computations, the vendors can provide predictions collaboratively during online phase without deeply jeopardizing their privacy. The proposed protocols do not significantly affect accuracy. However, since preciseness and performance are conflicting goals, they introduce some supplementary costs. On the other hand, creating trust network off-line improves online performance of recommendation process and due to simultaneous computations; the vendors can offer referrals efficiently using the scheme. Empirical results show that accuracy and reliability significantly improves due to working jointly.

5. PRIVACY-PRESERVING NAÏVE BAYESIAN CLASSIFIER-BASED RECOMMENDATIONS ON DISTRIBUTED DATA

In addition to numerical ratings, CF systems utilize binary preferences in which 1 represents *like* and 0 represents *dislike*. To produce recommendations from binary data, NBC-based CF algorithm is introduced (Miyahara and Pazzani, 2002). In this chapter, privacy-preserving solutions for generating NBC-based recommendations from VDD or HDD are proposed. In the introduced methods, confidentiality of data holders is preserved through various protocols. To analyze the overall performance of the proposed schemes, experiments are performed on real data; and suggestions are provided. Empirical results show that it is still possible to produce true NBC-based predictions without deeply jeopardizing data owners' secrecy on distributed data.

5.1. Introduction

To provide true CF services efficiently, various techniques have been proposed. NBC is among such techniques applied to CF (Miyahara and Pazzani, 2002). It is one of the most successful machine learning algorithms in many classification domains (Miyahara and Pazzani, 2002). Despite its simplicity, it is shown to be competitive with other complex approaches, especially in text categorization and content-based filtering tasks. Moreover, it is stable with respect to small changes to training data and it does not require large amounts of data before learning.

In this chapter, providing NBC-based predictions from horizontally or vertically partitioned data among multiple parties while protecting their privacy are investigated. Estimated recommendations should be accurate and provided efficiently without greatly jeopardizing data owners' secrecy. However, accuracy, privacy, and efficiency are conflicting goals. Thus, a good balance among them is wanted to achieve. The proposed privacy-preserving schemes eliminate data owners' privacy, legal, and financial concerns. Through privacy measures, online vendors are able to hide their true ratings and the rated items. Moreover, since each e-company has responsibility of protecting their own users' privacy, sharing their data with another company might cause legal problems (Oliveira et al., 2004). However, if they utilize the proposed schemes, they can get rid of legal issues. Finally, competing companies try to derive data from each other for

providing true and dependable predictions so that they can recruit new customers and make more money. The proposed schemes prevent them from deriving information about each other's databases so that they do not worry about financial loses. Real data-based trials are performed to assess the schemes. Then, empirical results are analyzed and some suggestions are provided. Due to privacy measures and distributed computations, extra storage, computation, and communication costs are inevitable. Since off-line costs are not critical, the schemes are scrutinized in terms of online extra costs only. Also, it is shown that the schemes are secure. In other words, online vendors' privacy is not deeply violated while offering true referrals.

5.2. Protecting Active User's Data

To get predictions from the MC, a should send **A** and q to it. Moreover, the MC should share her data with the collaborating companies to estimate a recommendation. However, once the MC obtains a's ratings, such data become its private data. Like it tries to hide the ratings it holds and the rated items while estimating recommendations through collaboration, it is also responsible for protecting a's ratings and the rated items. The following protocols are suggested to hide both a's rated items and her true ratings provided to the MC. Hiding rated items (HRI) protocol is described, as follows:

i. The MC first finds the number of empty cells (e) in A.

ii. It selectively or uniformly randomly chooses β over the range (1, 100].

iii. It then uniformly randomly selects a value, λ, over the range (1, β].

iv. Then, it can fill randomly selected λ percent of these e number of empty cells in a's ratings vector, where $h = e\lambda /100$.

v. The MC finally fills such randomly selected cells with fake ratings.

Although two methods are proposed to use, random and default vote, to fill such cells, since default votes (v_ds) are non-personalized ratings, filling empty cells with v_ds is more insightful. In random method, the MC uniformly randomly selects 1s and 0s; and fills empty cells with them. In default vote method, the MC determines v_ds for each item using the ratings it holds and fills them with corresponding v_d. In HDD, the MC finds the number of 1s (t) and 0s (s) for each item. It then compares t and s values. If $t > s$ then v_d for that item is 1, it is 0

otherwise. In VDD, the MC determines v_ds based on a's corresponding data. For each distribution, similarly, it finds the number of 1s (t) and 0s (s) in a's ratings vector's corresponding part. It then compares t and s values. If $t > s$ then v_d for that partition is 1, it is 0 otherwise. The MC can determine v_ds off-line in HDD because it owns required data to calculate them. Since off-line performance is not critical, it can compute them off-line and store them. Note that with increasing randomness, accuracy diminishes while confidentiality improves. With increasing h values, randomness increases; thus, accuracy diminishes. The values of h depend on how much privacy and accuracy the parties want. Also remember that the MC follows such steps in each prediction computation process to be secure against previously mentioned attacks other than the extreme cases.

After masking the rated items, the MC should perturb a's known ratings, as well. To prevent other parties from learning a's true rating values, RRT is utilized. RRT can be applied to perturb a's data, referred to as RRT protocol, as follows:

i. The MC first decides number of groups (*NG*).

ii. It then divides **A** into *NG* groups.

iii. For each group, it uniformly randomly selects two random values (θ_j and r_j) over the range (0, 1], where $j = 1, 2, \ldots, NG$.

iv. For each group j, it then compares θ_j and R_j. If $\theta_j < R_j$, it sends true data; otherwise, it sends the exact opposite of the ratings. In other words, when $\theta_j > r_j$, it reverses 1s into 0s and 0s into 1s in **A**.

Since the MC disguises each group j independently, the received data are true with probability θ_j for each group j. After disguising a's data, the MC can now send them to the collaborating companies.

5.3. Privacy-Preserving NBC-based HDD Schemes

Since data are horizontally distributed, Eq. (1.4) can be written, as follows:

$$p(c_j|f_1, f_2, \ldots, f_y) \propto p(c_j) \prod_i^y p(f_i|c_j) =$$
$$p(c_j) \prod_{i=1}^{y_1} p(f_i|c_j) \prod_{i=1}^{y_2} p(f_i|c_j) \ldots \prod_{i=1}^{y_z} p(f_i|c_j) \quad (5.1)$$

where y_g represents the known feature values of q held by company g, where $g = 1, 2, \ldots, z$. As seen from Eq. (5.1), since each company g knows the users' ratings for q they hold, they can easily compute $p(f_i|c_j)$ conditional probabilities for both

like and *dislike* classes and for all $g = 1, 2, \ldots, y_g$; and compute $\prod p(f_i|c_j)$ values for both classes if they know a's ratings. However, the MC sends them masked data, as explained previously. Thus, conditional probabilities should be computed first. They can be calculated, as follows:

$$p(f_i|c_j) = \frac{\#(f_i|c_j)}{\#c_j} \tag{5.2}$$

where $\#(f_i|c_j)$ shows the number of similarly rated items of c_j as the feature value of q for corresponding user; and $\#c_j$ represents the number of commonly rated items as j. Since a's data are grouped into NG groups, conditional probabilities can be computed, as follows:

$$p(f_i|c_j) = \frac{\#P_1(f_i|c_j) + \#P_2(f_i|c_j) + \cdots + \#P_{NG}(f_i|c_j)}{\#P_1(c_j) + \#P_2(c_j) + \cdots + \#P_{NG}(c_j)} \tag{5.3}$$

where $\#P_s(f_i|c_j)$ shows the number of similarly rated items of c_j as the feature value of q for corresponding user's data in group P_s; and $\#P_s(c_j)$ is the number of commonly rated items between a's and corresponding user's data in group P_s as j. In each group, since the received data can be true or false, each party computes $\#P_s(f_i|c_j)$ and $\#P_s(c_j)$ values twice: one for assuming the received data are true and one for assuming the received data are false. Once $p(f_i|c_j)$ values are computed for both *like* and *dislike* classes, the MC can easily calculate $p(c_j)$ values and determine the prediction for a on q because it owns a's ratings vector and the required probability values. The scheme producing a prediction for an a on q on distributed data among z parties while preserving privacy can be described, as follows:

i. a sends **A**, q to one of the companies from which she wants recommendation. This company acts as the MC. Suppose that the z^{th} party is the MC.

ii. As explained before, the MC first masks **A** and obtains **A′**. It then sends **A′** (including how **A** is divided into how many sub-vectors) and q to other z-1 companies, which have decided to collaborate with the MC beforehand.

iii. Since each company owns the known f_i values of q, for each class *like* and *dislike* and f_i values of q, they can compute $\#P_s(f_i|c_j)$ and $\#Ps(c_j)$

values assuming the received data are true; and $\#P'_s(f_i|c_j)$ and $\#P'_s(c_j)$ values assuming the received data are false.

iv. After each party calculates the required interim aggregate values (AV_s) for both *like* and *dislike* classes and for all known ratings of q, they then send them to the MC.

v. After receiving such values, the MC selects and finds the required data to find the conditional probabilities because it knows what groups include true data and which ones contain false data.

vi. It finally computes $p(c_j)$ values and estimates p_{aq}; and sends it back to *a*.

The MC does not know which users rated q, commonly rated items, and true ratings. Thus, it cannot learn any information about other companies' data from the received interim aggregate values in one prediction process. However, it can get information in consecutive recommendation process. Therefore, once each party receives data from the MC, they can use the HRI protocol to fill uniformly randomly chosen some of the empty cells of the received data with v_ds. Using the available ratings, such column default ratings can be determined by each party off-line like the MC does. In each prediction process, each party fills different numbers of empty cells so that the MC cannot derive data about their ratings and rated items from interim results.

When any party has no ratings for q, it is not able to compute conditional probabilities and cannot send any result. The MC then concludes that any user held by that party does not rate item q. Similarly, when the parties send interim results based on the users' data who rated q, the MC can learn how many ratings are available for q. To overcome this challenge and prevent the MC from learning how many users rated q, any company g first determines the density of the data set (d_g) it holds. It then selectively or uniformly randomly chooses a random number (Q_g) over the range (1, $4d_g$]. It then uniformly randomly selects a random number (V_g) over the range (1, Q_g]. It then uniformly randomly chooses V_g percent of its users and fills their corresponding cells for q with their row v_ds. On average, it sustains the density of the data set. Finally, the company computes interim aggregate results based on filled data and sends the result to the MC. Therefore, the MC cannot learn whether any user rated q or not, how many users rated it; and

which user rated it. Since online performance is critical, the parties can compute v_ds off-line.

When any company does not have any user rated q but having no commonly rated items with a, it selectively or uniformly randomly chooses a random number (X_g) over the range (1, $4d_g$] in order to maintain the same density of its data set. It then uniformly randomly selects a random number (W_g) over the range (1, X_g]. It then uniformly randomly chooses W_g percent of corresponding users' empty cells and fills them with corresponding item v_ds. Finally, the company computes interim aggregate results based on filled data and sends them to the MC.

5.4. Privacy-Preserving NBC-based VDD Schemes

In VDD scheme, a sends her ratings vector **A** and a query q to the company, which owns q. That company acts as the MC. Since data is vertically split, all ratings of q are held by the MC. After receiving **A** and q, the MC first masks **A**. It then sends the corresponding groups and q to those companies, which have decided to collaborate beforehand. They perform required computations and send results to the MC. Finally, the MC finds prediction by combining results from other companies and sends it back to a.

In order to determine whether a will like q or not, conditional probabilities should be computed first, as described in Eq. (5.2). Since data are vertically split and the **A** is grouped into NG groups by the MC, such conditional probabilities can be computed using Eq. (5.3).

After receiving a's corresponding data and q from the MC, the companies not having q should be able to compute $\#P_s(f_i|c_j)$ and $\#P_s(c_j)$ values for both classes twice because they do not know whether data in P_s are true or false in such a way to prevent the MC deriving information from their data. The privacy-preserving scheme computes such values based on vertically distributed data among z parties can be described, as follows:

i. a sends **A** and $\boldsymbol{q}$ to the company having q (the MC).

ii. The MC first finds the corresponding parts of **A** for each party according to items they hold. It then perturbs the corresponding parts of A and gets $\mathbf{A}'$ for each party. It then sends masked a's ratings vector's corresponding parts ($\mathbf{A}'_i$) to those z-1 companies, which have decided to join the

prediction computations beforehand.

iii. Since such companies do not know feature values of q, which is held by the MC only, they compute $\#P_s(f_i|c_j)$ and $\#P_s(c_j)$ values for all features. Moreover, they do compute them twice, for $f_i = 1$ and $f_i = 0$ because they do not know the feature values of q. However, since $P(f_i = 1|c_j) + P(f_i = 0|c_j) = 1$, they do compute such values for each classes for only f_i being 1 or 0. Furthermore, since such parties do not know the received data in each group are true or false, they compute such interim aggregate results twice. After computing such values for both *like* and *dislike* classes, they send them to the MC.

iv. After receiving such values, the MC selects the required data because it knows the feature values of q and the actual content of each group. It then calculates the conditional probabilities for both classes.

v. It finally estimates p_{aq} and sends it back to a.

Since a sends **A** to the MC*;* and it masks it using the HRI and RRT protocols, it knows how many 1s, 0s, and empty cells are in **A′**. Such information may help the MC to derive information about other z-1 companies' data. Therefore, such companies must compute $\#P_s(f_i|c_j)$ and $\#P_s(c_j)$ values in such a way to prevent the MC deriving data from their databases. To achieve such goal, each party should selectively or uniformly randomly fill some of the empty cells in the received ratings vector. Thus, each party can utilize the HRI protocol to reach their goal. As v_ds, the parties can use column or item v_ds because they own necessary ratings due to vertical partitioning. Since off-line costs are not critical, they can find such votes off-line to improve online performance.

In both PPHDD and PPVDD schemes, the parties cannot calculate conditional probabilities directly using their own data due to partitioning and masked **A**. The parties compute interim aggregated results and send them to the MC, which combines such results using Eq. (5.3) to produce recommendations for active users.

5.5. Privacy Analysis

In the schemes, the parties should not be able to learn true ratings and the rated items held by each other. Note that a's data are held by the MC**.** Nothing except q and NG can be public for all parties.

The parties other than the MC are not able to learn a's true ratings and rated items due to the filled cells and disguising using RRT. However, they can guess the randomly selected unrated items. The probabilities of guessing the correct β and λ are 1 out of 100 and 1 out of β, respectively. After guessing them, they can compute h. The probability of guessing the w randomly selected cells among e empty cells is 1 out of $Comb_w^e$. Since the parties do not know the v_d values, the probability of guessing the inserted v_d values for one item is 1 out of 2. Thus, the probability of guessing the randomly selected empty cells and their ratings is 1 out of $(100 \times \beta \times (1/2) \times h \times Comb_w^e)$. Since RRT is used to disguise **A**, the parties need to determine θ_j values to guess the actual ratings in each group. Since MC selects θ_j values over the range (0, 1], the probability of guessing the true θ_j values is 1 out of 10^l, where l shows the number of digits used for precision. With probability θ_j, the received data are true, otherwise they are false.

Calculating item or user v_ds does not reveal any useful information. Each party including the MC can find such votes without the help of other parties. Thus, such computations do not greatly jeopardize companies' privacy.

In both schemes, the parties insert some fake ratings into the received data from the MC. The collaborating companies are able to disguise a's data in such a way to achieve required levels of privacy and accuracy. In the recommendation generation processes, the MC does not know the rated items and the true rating values held by each party due to randomly selected empty cells and v_ds. Since the parties utilize the HRI like the MC does to disguise a's data, the MC can guess the randomly selected unrated items. Thus, as explained before, the probability of guessing the randomly selected empty cells and their ratings is 1 out of $(100 \times \beta_i \times (1/2) \times w \times Comb_w^e)$. In case of the extreme cases and hiding the users who rated the q in PPHDD scheme, the parties similarly mask the available ratings of q. The MC does not know W_g and V_g values, which are known by party g only. Moreover, it does not know fake ratings and which users' data involved in

computations. Therefore, the probability for the MC to figure out who rated q can be determined similarly.

Due to partial interim results, it is difficult for the MC to derive information from such results. In both PPHDD and PPVDD schemes, even if the MC knows a's ratings, since only commonly rated items between a and other users are used for recommendation computations; it will not be able to derive any data held by other companies. Due to the HRI protocol utilized in each recommendation process, acquiring any information about any unknown user cannot be successful even if the MC utilizes consecutive queries. When some of the companies coalesce to acquire the other company's data, they cannot succeed it due to the HRI protocol and v_ds.

5.6. Supplementary Cost Analysis

Accuracy, privacy, and performance are conflicting goals. Due to privacy measures, proposed schemes introduce extra off-line and online costs. Since off-line costs are not critical for overall performance, online supplementary costs of proposed schemes are analyzed. Such costs can be storage, computation, and communication costs in terms of both number of communications and amount of transferred data.

First of all, the costs of the NBC algorithm based on a central server are discussed. The storage cost to save ratings collected from n users for m items is in the order of $O(nm)$. Similarly, when predictions generated on $n \times m$ user-item matrix, the computation cost of producing NBC-based recommendations is in the order of $O(nm)$. Since a sends her data and gets back a prediction, the number of communications is 2 or in the order of $O(1)$; and the amount of data transferred between the CF system and a is in the order of $O(m)$.

In the proposed multi-party schemes, additional storage costs are expected due to v_ds. In the PPHDD scheme, each company saves the item and user v_ds. Thus, supplementary storage cost due to v_ds is in the order of $O(m)$ for each company assuming that number of users they hold is less than m. Similarly, in the PPVDD scheme, extra storage cost is in the order of $O(m_g)$ for each company g. Compared to $O(nm)$, additional costs can be considered negligible.

In PPHDD schemes, after receiving data from the MC, the companies perform computations simultaneously. Therefore, the computation cost is in the order of $O(n_{max}m)$, where n_{max} shows the maximum number of users involving in the recommendation process in one of the companies' database. When extreme cases occur, the companies fill some of the target item's cells (let say w cells) with corresponding v_ds. If w is bigger than N_{max}, then the computation cost will be in the order of $O(wm)$; otherwise it will not change. Note that since a's data are masked; collaborating companies should perform computations twice. However, due to concurrent executions, running time of multi-party scheme is still better than the running time of centralized scheme. In PPVDD schemes, since the companies other than the MC does not know the feature values of q, they compute required values for both f_g is being 1 and 0. They also calculate such values for all features because they do not know which features are known. Furthermore, they fill some of a's ratings vector's empty cells with v_ds. And finally, they perform each computation twice because a's data are masked. However, like in PPHDD scheme, since prediction computations are conducted simultaneously by all companies in the PPVDD scheme, as well, the computation cost is in the order of $O(nm_{max})$, where m_{max} shows the maximum number of items involving in the recommendation process in one of the companies' database.

In privacy-preserving multi-party schemes, a communicates with the MC, while the MC exchanges data with the z-1 companies. Therefore, the numbers of communications are $2z$ for both PPHDD and PPVDD schemes. In other words, they are in the order of $O(z)$, where notice that z is a constant. Although z is defined less than m or n, for practical purposes, it is a small constant. Also note that it can be said that the communications between the MC and the other parties are conducted simultaneously. In PPHDD schemes, the maximum amount of data to be transferred is a's rating vector plus q. Therefore, the amount of data transferred in a communication message will be the same with the central server-based scheme. In PPVDD schemes, since the MC sends a's rating vector's corresponding parts to the other companies, there is no additional communication costs per messages in terms of amount of data transferred.

5.7. Accuracy Analysis: Experiments

To evaluate the schemes in terms of accuracy and online performance, various experiments based on real data sets are performed. Unlike off-line performance, online performance is critical for the overall success of CF. Therefore, the schemes are investigated in terms of online efficiency. Accuracy shows how precise privacy-preserving schemes-based predictions are. Similarly, online performance points how much effort is employed for producing private referrals using the schemes. Performance and preciseness are very important for customers' pleasure. Therefore, several experiments are conducted for testing the proposed schemes' effects on them.

Experiments are run using Jester and EM data sets. To measure accuracy, classification accuracy (CA) and F1 metric are used. CA is the ratio of the number of correct classifications to the number of classifications. F1 is a weighted combination of precision (P) and recall (R), which are categorized as classification accuracy metrics by Herlocker et al. (2004), where $F1 = (2PR) / (P + R)$. The higher the CA and F1, the better the results are. To determine the schemes' online performances, T is defined in seconds as online time required offering recommendations. Coverage is also used as a metrical indicator to show the effectiveness of the NBC-based CF on multi partitioned data. A basic measure of coverage is the percentage of items for which predictions are available. Low number of users and neighbors results in low coverage.

Firstly the numerical ratings are transformed into binary ones (Miyahara and Pazzani, 2002). For EM data set, items are labeled as 1 if the numerical rating for the item is bigger than 0.5 or 0 otherwise. Items are labeled as 1 if the numerical rating for the item is above 2.0 or 0 otherwise in Jester. For training and test sets, 3,000 and 2,000 users are selected randomly, respectively, among those users who have rated at least 50 and 60 items from Jester and EM, respectively. Training data is divided up to five groups. Five rated items are randomly selected from test users' ratings vectors as test items for both schemes. Trails are run trials using MATLAB 7.6.0 on five computers, which are Intel Core2Duo, 2.0 GHz with 1 GB RAM.

Experiment 1: Trials are performed to show how integrating different amounts of horizontally or vertically distributed data among varying numbers of companies affect the results. *n*, *m*, and *z* values are varied to show how accuracy and *T* change.

By combining HDD, it is more likely to find large enough neighborhoods for more accurate and reliable referrals. Experiments are performed using both data sets based on randomly chosen 100, 250, 500, and 1,000 train users, where it is assumed that data are held by 1, 3, or 5 parties. Randomly selected 500 users are used for testing. For each test user, five rated items are randomly selected, withheld their ratings and produced recommendations for them based on split data alone and integrated data. The outcomes are shown in Table 5.1a through Table 5.1d and in Table 5.2a through Table 5.2d for Jester and EM, respectively. In the tables, Dist. means Distributed and Integ. means Integrated. In Table 5.1a and Table 5.2a, how overall performance changes are shown with varying *z* values for Jester and EM, respectively when *n* equals 100.

Table 5.1a. Overall Performance by Integrating HDD (Jester, *n* = 100)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**63.76**	62.68	**63.76**	62.43	**63.76**	62.05	**63.76**
F1 *(%)*	**62.78**	61.18	**62.78**	61.04	**62.78**	61.01	**62.78**
T(sec)	**0.76**	0.37	**0.44**	0.27	**0.38**	0.18	**0.23**
n_i	**100**	50	**100**	33	**100**	20	**100**

Table 5.2a. Overall Performance by Integrating HDD (EM, *n* = 100)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**65.61**	64.25	**65.61**	63.25	**65.61**	62.86	**65.61**
F1 *(%)*	**66.98**	66.04	**67.38**	65.78	**67.38**	64.76	**67.38**
T(sec)	**2.45**	1.36	**1.53**	0.96	**1.23**	0.67	**0.84**
n_i	**100**	50	**100**	33	**100**	20	**100**

In Table 5.1b and Table 5.2b, similarly, overall performance changes are shown with varying *z* values for Jester and EM, respectively when *n* = 250.

Table 5.1b. Overall Performance by Integrating HDD (Jester, n = 250)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**65.24**	64.90	**65.24**	64.17	**65.24**	63.48	**65.24**
F1 *(%)*	**63.85**	63.64	**63.85**	63.26	**63.85**	62.74	**63.85**
T(sec)	**1.98**	0.82	**0.95**	0.60	**0.74**	0.37	**0.48**
n_q	**250**	125	**250**	83	**250**	50	**250**

Table 5.2b. Overall Performance by Integrating HDD (EM, n = 250)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**66.14**	65.22	**66.14**	64.42	**66.14**	64.13	**66.14**
F1 *(%)*	**68.35**	67.34	**68.35**	66.16	**68.35**	65.73	**68.35**
T(sec)	**6.30**	2.53	**2.95**	1.85	**2.26**	1.42	**1.76**
n_q	**250**	125	**250**	83	**250**	50	**250**

Accuracy and online computation time changes are demonstrated with varying k values for Jester and EM in Table 5.1c and Table 5.2c, respectively when n = 500. Finally, accuracy and online computation time changes are demonstrated with varying k values for Jester and EM in Table 5.1d and Table 5.2d, respectively when number of users is 1,000.

Table 5.1c. Overall Performance by Integrating HDD (Jester, n = 500)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**66.13**	65.57	**66.13**	64.88	**66.13**	63.65	**66.13**
F1 *(%)*	**64.31**	63.97	**64.31**	63.94	**64.31**	62.53	**64.31**
T(sec)	**4.31**	1.71	**1.86**	1.12	**1.43**	0.68	**0.85**
n_q	**500**	250	500	167	**500**	100	**500**

Table 5.2c. Overall Performance by Integrating HDD (EM, n = 500)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**67.12**	66.03	**67.12**	65.82	**67.12**	65.18	**67.12**
F1 *(%)*	**69.32**	68.21	**69.32**	67.68	**69.32**	66.91	**69.32**
T(sec)	**13.07**	6.27	**6.74**	3.11	**3.46**	2.53	**2.84**
n_q	**500**	250	**500**	167	**500**	100	**500**

Table 5.1d. Overall Performance by Integrating HDD (Jester, n = 1,000)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**67.12**	66.45	**67.12**	65.64	**67.12**	64.35	**67.12**
F1 *(%)*	**65.34**	64.64	**65.34**	64.21	**65.34**	63.67	**65.34**
T(sec)	**8.45**	4.17	**4.84**	2.20	**2.87**	1.34	**1.52**
n_q	**1,000**	500	**1,000**	333	**1,000**	200	**1,000**

Table 5.2d. Overall Performance by Integrating HDD (EM, n = 1,000)

	$C_{\#}$						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**68.16**	66.59	**68.16**	66.27	**68.16**	66.17	**68.16**
F1 *(%)*	**70.58**	68.94	**70.58**	68.29	**70.58**	68.37	**70.58**
T(sec)	**28.19**	13.68	**14.15**	9.54	**9.87**	6.18	**6.51**
n_q	**1,000**	500	**1,000**	333	**1,000**	200	**1,000**

As seen from Table 5.1a-5.1d and Table 5.2a-5.2d, collaboration between different parties helps e-companies improve the quality of the recommendations. The outcomes demonstrate that with increasing amount of data, accuracy improves. It becomes easy to form dependable and precise neighborhoods through collaboration; that leads to improved recommendations. Overall gains for EM are larger than the ones for Jester because Jester is denser compared to EM. As seen from the results, T improves with increasing z values. Since computations are conducted simultaneously by the companies involving in the CF process, T is expected to improve with increasing number of parties participating in the distributed computations. To sum up, if parties collaborate and join in distributed CF, they can produce more accurate results more efficiently than if they perform such tasks on their individual data only.

Experiment 2: By integrating VDD, similarities between a and other users become more dependable and accurate. When online vendors collaborate over VDD, it is more likely to have more commonly rated items between users because number of items increases. To validate how overall performance changes with varying amounts m or integrating VDD, several experiments are conducted. Since Jester has limited number of items (it has only 100 items), EM data set is employed only in these experiments. 1,000 train users are used while varying m

from 400 to 1,648. For each of the 500 test users, five predictions are generated for randomly chosen five rated items based on split data alone and integrated data. It is assumed that data held by one party or VDD among 1, 3, or 5 companies. The results are shown in Table 5.3a through Table 5.3c.

Table 5.3a. Overall Performance by Integrating VDD (EM, m = 400)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**64.16**	62.69	**64.16**	62.15	**64.16**	61.31	**64.16**
F1 *(%)*	**65.02**	64.14	**65.02**	63.49	**65.02**	62.19	**65.02**
T(sec)	**4.89**	2.37	**5.32**	1.89	**4.26**	1.27	**2.98**
m_q	**400**	200	**400**	133	**400**	80	**400**

Table 5.3b. Overall Performance by Integrating VDD (EM, m = 800)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**65.64**	64.45	**65.64**	63.16	**65.64**	62.53	**65.64**
F1 *(%)*	**67.98**	65.36	**67.98**	64.81	**67.98**	63.27	**67.98**
T(sec)	**11.99**	5.18	**12.67**	3.59	**9.35**	1.72	**4.11**
m_q	**800**	400	**800**	267	**800**	160	**800**

Table 5.3c. Overall Performance by Integrating VDD (EM, m = 1,648)

	z						
	1	**2**		**3**		**5**	
		Dist.	**Integ.**	**Dist.**	**Integ.**	**Dist.**	**Integ.**
CA *(%)*	**68.16**	65.56	**68.16**	64.24	**68.16**	63.83	**68.16**
F1 *(%)*	**70.58**	67.84	**70.58**	66.69	**70.58**	66.17	**70.58**
T(sec)	**28.12**	11.83	**25.19**	6.74	**14.52**	4.67	**10.65**
m_q	**1,648**	824	**1,648**	549	**1,648**	330	**1,648**

As seen from Table 5.3a-5.3c, if data owners collaborate when data collected for CF purposes are vertically distributed, they are able to provide more precise recommendations in less time. The reason why accuracy improves can be explained, as follows: By increasing number of companies participating in CF process when their data are vertically distributed, amount of items and commonly rated ones increases, which help e-companies find more trustworthy and accurate similarities between users. Dependable and truthful similarities then lead to better

predictions. Improvement in T can be explained with distributed computations. With increasing z values, T gets better because number of companies joining in distributed computations increases and each party works on fewer amounts of data. Note again that communication costs are ignored while calculating z values. In conclusion, the results show that when data collected for CF are distributed vertically among various parties, such parties can produce more precise recommendations in less time if they collaborate.

Experiment 3: When a company is not able to return any result to the MC, it should use the proposed schemes in order to prevent the MC from deriving information about its data. In case of the having no rating for q, the company uniformly randomly chooses some of the users it holds and fills their cells for q with row v_ds, thus, they add random users to recommendation process. Since such case occurs when data are sparse, experiments are performed using EM only because almost 97 percent of the ratings are missing. 500 train and 500 test users are uniformly randomly selected among those users who rated more than 60 items. Since such users as train users are selected, it is ended up with a data set whose density is about 4%. Therefore, d is set at 4. For each test user, five referrals are generated for five rated items. It is assumed that data are distributed among five companies and the ratings of q in one of the companies are removed in randomly selected one of the companies. Then the proposed scheme is utilized. In order to show how accuracy changes with varying randomness, Q_g is changed from d to 8d. Trials are run 100 times and the overall averages are displayed in Table 5.4.

Table 5.4. Effects of Adding Random Users

Q_g	**0**	**2*d***	**4*d***	**8*d***
CA *(%)*	66.24	67.52	67.32	67.16
F1 *(%)*	68.25	69.85	69.68	69.51

In Table 5.4, Q_g is 0 means that the predictions are generated on four parties' data. In other words, the party having no rating for q does not join the CF process. As seen from Table 5.4, the quality of the referrals improves when the fifth vendor joins the prediction process. As expected, recommendations generated based on all five companies' data are more accurate than the ones on

four parties' data. Since inserted row v_ds are non-personalized ratings, filling empty cells of q with them and letting the fifth company join the distributed computations makes accuracy better. Although with increasing Q_g values from $2d$ to $8d$, accuracy slightly becomes worse, they are still better than when Q_g is 0.

Experiment 4: In addition to the adding random users, one of the companies might have users who rated q but not any common ratings with a. To overcome this challenge, parties add random ratings to the users' rating vector having rating for q. In order to investigate how overall performance changes when the proposed scheme is utilized to overcome this problem, various trials are performed. It is assumed that data are distributed among five companies and one of them faces with the problem. Therefore, the ratings of those users who rated q are removed in randomly selected one of the companies, where only the ratings of q are kept. Then some of their randomly chosen cells are filled with corresponding column v_ds as explained previously. X_g is varied from d to $8d$ in order to show how accuracy changes with varying randomness. The experiments are performed 100 times and shown the results in Table 5.5.

Table 5.5. Effects of Adding Random Ratings

X_g	**0**	**2*d***	**4*d***	**8*d***
CA *(%)*	66.24	68.10	68.21	68.36
F1 *(%)*	68.25	70.30	70.32	70.39

As seen from Table 5.5, both CA and F1 values improves when the fifth vendor participates in the distributed computations. With increasing X_g values, the quality of the referrals slightly increases. This is due to the similar reasons that are explained for the previous experiments. To sum up, the results show that the PPHDD schemes preserve data owners' privacy while allowing them to offer accurate predictions.

Experiment 5: To prevent the collaborating companies from deriving information about a's data, the MC fills uniformly randomly selected some of a's unrated cells with v_{ds}. Similarly, to avoid data owners from deriving data from privacy attacks in PPVDD schemes, the parties that need to send sub-results to the MC insert v_{ds} into some of the randomly selected empty cells of a's ratings vector.

Although such fake ratings are non-personalized ratings, filling such empty cells with them might affect accuracy. To show how different β_g values affect the quality of the referrals, experiments are performed using both data sets. Since the results are similar, results based on EM data set are shown only. 1,000 and 500 train and test users are used, respectively, where the users who rated more than 60 items randomly selected among them. Referrals are estimated for randomly selected five rated items for each test user. The experiments are run for 100 times, overall averages of CA and F1 values are calculated; and they are displayed in Table 5.6.

Table 5.6. Overall Performance with Varying β_g Values

B_g	0	25	50	75	100
CA *(%)*	68.16	69.52	68.40	67.86	67.14
F1 *(%)*	70.58	71.23	70.63	69.72	69.48

In Table 5.6, β_g is 0 means that the parties do not utilize any privacy-preserving scheme. As seen from Table 5.6, when β_g is less than or equal to 50, the results with privacy measures are better than the ones without privacy concerns. However, with increasing randomness from β_g is being 25 to 100, accuracy becomes worse. When β_g is bigger than 50, the results with privacy concerns are worse than the results without privacy concerns. Although this is the case, accuracy losses are at most 1% only. Generally speaking, privacy-preserving schemes for VDD still generate promising results. Moreover, data owners can determine the value of β_g based on how much accuracy and privacy they want. The proposed schemes preserve online vendors' privacy when they join the distributed CF processes while still allowing them to offer recommendations with decent accuracy.

5.8. Conclusions

Online vendors, especially newly founded ones, might have insufficient data for dependable and precise predictions. They might face with cold start problem. To overcome such problem and produce better CF services, such companies might decide to collaborate. However, due to privacy, legal, and financial reasons, such parties may not want to collaborate.

In this chapter, privacy-preserving schemes are proposed to achieve NBC-based recommendations when data are horizontally or vertically distributed among more than two parties. The proposed schemes, PPHDD and PPVDD, make it possible for data owners to offer accurate and dependable referrals without greatly jeopardizing their privacy. Since efficiency is another goal that must be achieved by the schemes, they are analyzed in terms of online supplementary costs. It is demonstrated that online additional costs due to privacy concerns are small and the schemes are still able to produce recommendations efficiently. In order to evaluate the general performances of the schemes, real data-based experiments are performed. The results show that distributed computations or collaborations among multiple parties improve accuracy. Moreover, the proposed schemes produce referrals with decent accuracy while protecting online vendors' privacy. Data owners are able to use different values of privacy protection measures in order to achieve required levels of privacy and accuracy. According to their requirements, they adjust the values of such measures.

6. PRIVACY-PRESERVING NAÏVE BAYESIAN CLASSIFIER-BASED P2P COLLABORATIVE FILTERING

In this chapter, a privacy-preserving scheme is proposed for achieving recommendations with privacy, where data is distributed among *n* peers. In other words, a special kind of HDD is covered. In this kind of data distribution, individuals control their own preference data. There is no central server holding users' data, where $z = n$. Individuals utilize P2P networks to get recommendations; thus, they collaborate via a P2P network. Researches propose several privacy-preserving P2P network-based CF schemes work on numerical ratings. However, there is no study covering P2P solutions for binary rating-based CF applications. Hence, in this chapter, an NBC-based CF method working on P2P networks is proposed. To overcome privacy concerns of users, the method employ RRT method. The introduced scheme is analyzed in terms of accuracy, privacy, and efficiency. Real data-based results show that the schemes offer accurate NBC-based predictions with privacy eliminating central server.

6.1. Introduction

CF and PPCF schemes proposed so far are generally executed with a central server. Since users must send their ratings vectors to a central server for producing recommendations, the server controls all of the ratings. Therefore, there is a threat for users' privacy. The server performs all computations and provides many recommendations to loads of users during an online interaction. Using a central server for CF purposes is not desirable for privacy and performance reasons. With increasing popularity of the Internet, users are able to construct small networks to exchange and share various information over the Web. Decentralized approaches have advantageous like workload sharing, easy information exchange, eliminating control of central servers, controlling their own data, and so on. Therefore, P2P network is proposed to use to generate NBC-based referrals. In a close community, users can construct a P2P network. The members of the network might academicians in a university, practitioners in a hospital, students in a school, workers in a company, etc. Anyone's friends in her messenger list might be a good example for a P2P network. Peers in the network help each other to produce recommendations. Users in the P2P network might have ratings vectors

for different categories like movies, books, music CDs, etc. Using such ratings, they are able to involve predictions processes for their peers.

In this chapter, how to produce NBC-based CF services while preserving users' privacy without using a central server are investigated. A P2P network among various peers is proposed to use to generate recommendations without violating their privacy. The privacy risks that might occur among such peers while providing predictions over a P2P network are studied. To overcome such risks, privacy-preserving schemes are proposed. Due to privacy concerns, accuracy might be affected. To show how overall performance of the proposed schemes changes with privacy concerns, real data-based experiments are conducted. The scheme is analyzed in terms of accuracy, privacy, and efficiency. Finally, the findings are displayed and suggestions are provided.

6.2. P2P NBC-based Collaborative Filtering with Privacy

NBC is one of the most successful learning algorithms. Although it is applied to CF productively; however, with increasing number of users/items, its performance degrades significantly. Kaleli and Polat (2007b) show how to provide NBC-based predictions while preserving users' privacy. However, in their scheme, a central server collects disguised users' ratings. It controls all of the data and conducts all computations online to offer recommendations. Since data is valuable asset, the server can use them for malicious purposes. Users mask their data in the same way in order to have consistently masked data. They propose to use one- and multi-group schemes. With increasing number of groups, however, their schemes' performances decrease considerably. In the proposed scheme, each user or peer controls her own data. The control of central server is broken. Moreover, recommendation computation workload is distributed among the peers. Therefore, performance improves noticeably. Unlike the schemes proposed by (Kaleli and Polat, 2007b), data is masked differently and the schemes are still able to provide predictions from such inconsistently perturbed data.

The introduced scheme should preserve users' privacy and provide accurate recommendations efficiently. Although it is a challenge to define privacy succinctly, it can be defined, as follows: All peers or users including active users should not be able to learn the true values of other peers' ratings. Moreover, since

learning rated items might be more damaging, they should not learn rated items. To achieve privacy, RRT is utilized. Due to privacy concerns and RRT, it is expected that accuracy might get worse. However, recommendations provided with privacy concerns should still be accurate. Therefore, predictions estimated from masked data while preserving privacy should be as close as possible to true ratings. In other words, accuracy losses due to privacy concerns should be small and make it possible to offer acceptable recommendations. It is desirable to offer many referrals to loads of users during an online interaction. Therefore, the schemes' online computation times should be small and make it possible to generate referrals to various peers. Compared to centralized privacy-preserving NBC-based CF schemes, decentralized schemes are expected to achieve higher performance because online computations are split among different peers. To produce a recommendation without privacy concerns over a P2P network, the peers follow the following steps:

i. When *a* or a peer who wants a prediction for *q*, she sends a request including *q* to other peers in the network.

ii. Those peers who rated *q* and want to involve CF process send their responses back to *a*.

iii. After determining train users or peers, *a* sends her ratings vectors to them. She then computes $p(c_j)$ values for both classes.

iv. Each peer then computes $p(f_i|c_j)$ values for each class based on *a*'s and their ratings and *q*. They then send the calculated values to *a*.

v. After receiving all values from all peers, *a* computes $p(f_i|c_j|f_1, f_2, \dots, f_n)$ probabilities for both classes using Eq. (1.4). She finally assigns her target item to the class with the highest probability.

With privacy concerns, *a* should not be able to learn the true ratings and the rated items of train users; and the train users should not learn *a*'s ratings and rated items. Note that it is also important to hide rated items because it might be more damaging to reveal which items are rated. To increase privacy level, *a* perturbs her private data differently for each peer. To mask her data, *a* follows the following steps before sending her data to the train peers:

i. *a* first finds number of rated (m_r) and unrated (m-m_r) items.

ii. She decides a random integer (α_a) between 0 and 100. She then uniformly randomly chooses another random integer (δ_a) over the range [0, α_a].

iii. She uniformly randomly selects δ_a percent of her unrated items' cells (w) and then fills half of them with 1s and the remaining cells with 0s. With increasing α_a values, a adds more randomness to her ratings vector. That might make accuracy worse while increasing privacy level. Therefore, she is able to decide α_a value in such a way to achieve a required balance between accuracy and privacy.

iv. a uniformly randomly chooses $n_p \times NG_t$ number of θ (θ_{agNG}) values over the range [0, 0.5] because complementary θ_{ag} values produce the same results in terms of the randomness, where n_p shows the number of peers sending response to a to involve prediction process, $g = 1, 2, \dots, n_p$, and NG_t shows the total number of groups.

v. To determine number of groups for each peer, a uniformly randomly selects n_p number of NG (NG_{ag}) values over the range [2, γ], where NG is a positive integer and γ should be a small number because as stated in (Kaleli and Polat, 2007b), when NG is bigger than 5, performance becomes worse.

vi. After determining NG_{ag} values for each peer, a divides her ratings into NG_{ag} groups. She then uniformly randomly selects NG_{ag} random numbers (r_{agNG}) for each peer over the range [0, 1]. Then, she compares r_{agNG} random values with corresponding θ_{ag} values. If r_{agNG} is bigger than θ_{ag}, then a reverses her ratings in the corresponding group; otherwise, she keeps the same ratings values, as explained previously.

vii. And finally, a sends her masked data to each peer together with the corresponding NG_{ag} values. Remember that a masks her data differently for each peer and NG_{ag} values are needed to compute probability values.

The peers cannot learn true ratings and rated items of a, because they do not know randomly selected blank cells and random numbers. Once they received perturbed ratings, q, and NG_{ag} values from a, each peer estimates $p(f_i|c_j)$ values, as follows: Since the train peers receive NG_{ag} values, they know each possible

situations of a's perturbed vector. For example, if NG_{ag} is 3, a's disguised ratings vector can have eight possible situations like TTT, TTF, TFT, etc, where T and F represent true and false data, respectively. For each possible case, the peers estimate $p(f_i|c_j)$ values for both classes. For NG_{ag} is 3, the peer g estimates eight $p(f_i|c_j)$ values for both classes by considering eight possibilities. After each peer computes such values for both classes by considering each possible case, they then send them to a. Since a knows how she disguised her data for each peer, she considers those values that represent true ratings for both classes and then disregards the remaining values. Moreover, a can easily compute $p(f_i|c_j)$ values. And finally, a calculates $p(f_i|c_j|f_1, f_2, \dots, f_n)$ values; and assigns q to the class with the highest probability.

Due to data disguising by grouping ratings into multiple groups using RRT, there is no accuracy losses, because the peers consider all possible cases and a uses the values for true ratings vector. However, since a fills some of her blank cells for unrated items with v_{ad}, accuracy might become worse because v_{ad} might not represent true votes for unrated items.

6.3. Privacy Attacks

In the proposed scheme, explained previously, although data is masked, the peers including the active peer might pose various privacy risks. Such risks might come from the train peers because they receive data from active peers and they might try to learn the true ratings and the rated items. The active peer might be able derive data from the results, which she receives from various train peers. In multiple scenarios, active peers can derive useful information about the train peers' data from $p(f_i|c_j)$ values. Privacy risks are divided into two groups, as follows:

Train Peers' Privacy Attacks: After receiving a disguised ratings vector from a, any train peer tries to learn the true ratings and the rated items. Since θ_{ag} and r_{ag} values, number of blank cells, and the blank cells are known by a only, each peer cannot learn the true ratings and the rated items by herself. Although each peer cannot do anything to a's data, however, they might decide to collaborate to derive a's data. With increasing number of peers involving the collaboration, the chances they might learn a's data increase.

The following solution is suggested against this privacy risk: *a* should select θ_{ag} values over the range [0, 1] in such way that the mean value of θ for those groups including the same items should be 0.5. In this way, when the train peers collaborate to learn the true ratings of the same items fall into the same group, the probability of the received ratings to be true will be 0.5. Since the ratings are binary, this value does not help them at all. The peers cannot increase the chance to learn the rated items by collaboration, because *a* fills the same unrated items' cells for all peers. Prediction generations continue as any peer asks referrals. The active peer asks recommendations for different items time to time. In each process, she sends her masked data to each train peer. However, this might violate her privacy. Therefore, the active peer starts with the masked ratings vector. In the subsequent each recommendation process, she removes the oldest ratings and adds new ratings. Note that customers frequently buy items like movies, CDs, books, and so on over the Internet. Their ratings vectors include more and more ratings as they purchase new items. As shown by (Kim et al., 2008), using the recent ratings leads better quality recommendations. Therefore, the peers might decide to remove the oldest ratings and utilize the newest ratings to obtain high quality predictions. Since the ratings vectors employed in recommendation processes are changed in each prediction generation process, this prevents the peers from deriving data about each other's ratings through multiple prediction processes.

Active Peer's Privacy Attacks: The attacks posed by an active user *a* can be briefly explained, as follows:

i. *a* can learn which peers rated the *q* and which did not because when she sends a request to her peers on the P2P network for producing predictions, only those peers who rated the *q* send responses to her to involve in the process. By asking predictions for various items in multiple scenarios, *a* can learn which peers rated which items.

To get meaningful and trustworthy predictions, it is not possible to completely resolve this attack, because those peers who rated *q* should involve in prediction process. However, the following solution is proposed based on randomness to partially overcome this problem: When

a request is sent by a, each peer on the network uniformly randomly selects a random value (λ_g) and a random number (r_g) over the range [0, 1]. If r_g is less than λ_g, then the peer g sends a response to a to participate in prediction process. On the average, half of the peers on the network join in referral computation. If they did not rate the q, they use their default vote (v_{gd}) to fill its entry. Since a does not know λ_g and r_g values, she cannot learn which peers rated or not rated the q.

ii. In multiple scenarios, a might decide to use fake ratings and by changing the values of a single cell in her ratings vector each time to derive data about other peers' data. Since a gets $p(f_i|c_j)$ values, which are computed on other peers' and a's data, by changing the value of a single cell each time, a is able to derive other peers' data.

The following scheme is introduced to overcome this attack: In each recommendation process, each peer first decides a random integer (α_g) between 0 and 100. They then uniformly randomly choose another random integer (δ_g) over the range [0, α_a]. They finally uniformly randomly select δ_g percent of their unrated items' cells and fill half of them with 1s and half of them with 0s like active peers. Since a does not know α_g and δ_g values and each time the peers utilize different values, she cannot derive information about the train peers' data in multiple scenarios.

iii. Unlike previous attack, a might be able to derive data from $p(f_i|c_j)$ values computed for both classes and all possible cases without using fake ratings and without making any malicious change in her ratings vector. She can compare the results of multiple recommendation processes to learn the true ratings and the rated items of other peers. However, she cannot derive useful information through this attack, because the train peers fill some of their randomly selected unrated items' cells with fake votes, as explained in the previous attack.

6.4. Privacy Analysis

The proposed scheme is analyzed in terms of privacy, as follows: The train peers do not figure out the rated items due to randomly filled unrated items' cells with

bogus ratings. Note that a masks the same vector differently and sends it to each peer. The peers might guess the randomly filled cells. The probability of guessing α_a is 1 out of 100 because it is determined over the range [0, 100]. After guessing it, the probability of guessing the correct δ_a is 1 out of α_a. After that they can figure out the number of filled cells (w) with the help of the blank cells in the perturbed vector when it is not totally filled. Since the peers know that half of the filled cells (w/2) filled with 1s and the remaining w/2 cells filled with 0s, the probabilities of guessing which cells filled with 1s and 0s are 1 out of $Comb_{w_1}^{m_1}$ and $Comb_{w_0}^{m_0}$, where m_1 and m_0 represent the numbers of 1s and 0s, respectively; w_1 and w_0 show the numbers of randomly filled cells with 1s and 0s, respectively; and $Comb_r^o$ shows the number of ways of selecting r outcomes from o possibilities. Therefore, the probability of figuring out the filled unrated items' cells is 1 out of $[100 \times \alpha_a \times Comb_{w_1}^{m_1} \times Comb_{w_0}^{m_0}]$. After that they can easily learn the rated items.

Each peer is not able to figure out a's true ratings by herself. They might decide to collaborate to increase the chance of guessing the true ratings. However, due to the underlying data disguising and selection of θ_{ag} values explained previously, they are not able to do that. Since the peers do not know the θ_{agNG} values and the random numbers (r_{agNG}), they cannot learn whether the received data is true or false in the same group. Even if they figure out the ratings in one group, they cannot learn the votes in other groups, because data is perturbed independently in different groups. Thus, with increasing number of groups, privacy enhances.

Remember that a receives $p(f_i|c_j)$ values rather than the peers' ratings vectors, the peers fill some of their randomly selected blank cells with fake ratings, they fill different cells in each process, and they estimate $p(f_i|c_j)$ values on filled vectors. Therefore, a is not able to learn the true ratings and the rated items through $p(f_i|c_j)$ values.

Finally, a might try to learn who rated the target item q or not. However, since the peers on the network decide to join the recommendation process or not

based on the comparisons on λ_g and r_g values and they are known by the peers only, *a* cannot figure out who really rated the *q* or not.

6.5. Supplementary Costs Analysis

Due to the proposed scheme, extra costs like storage, communication, and computation costs are expected. In order to reach higher performance, additional costs due to privacy concerns should be small. Unlike off-line costs, online costs are critical for overall performance. Therefore, overhead online costs due to privacy-preserving schemes should still make it possible to offer predictions efficiently.

The scheme introduces additional storage costs, however, they are negligible. The peers on the network should save their default votes. Extra storage costs due to privacy concerns for each peer are order of 1 because each peer saves her default rating in a 1×1 matrix. Therefore, extra storage costs for all peers to save their default ratings are negligible. However, *a* should additionally save her filled ratings vector in a $1 \times m$ vector. Moreover, she should save NG_{ag}, θ_{ag}, and r_{agNG} values. Therefore, on average, additional storage costs for *a* are order of $n_p \times \gamma$. Since γ is a small constant, due the privacy-preserving schemes, extra storage costs are small.

Due to privacy concerns, the proposed schemes do not cause any extra number of communications. However, amount of information exchanged between *a* and the train peers increases due to the proposed scheme. *a* additionally sends her filled ratings and NG_{ag} values to other peers. The train peers sends $p(f_i|classj)$ values for all possible cases rather than single values for both classes to *a*. Therefore, on average, overhead communication costs in terms of the amount of data are order of $2^{\gamma/2}$. Note again that γ is a small constant and *a* should be able to determine its value in such a way to achieve required levels of privacy and efficiency.

Supplementary computation costs due to data disguising are negligible. However, the train peers compute $p(f_i|classj)$ values, on average, $2^{\gamma/2}$ times for both classes rather than a single value. Therefore, additional computation costs are order of $2^{\gamma/2}$. Remember that that γ is a small constant and the overhead costs are split among all train peers in the P2P network-based schemes. In the schemes

proposed by (Canny, 2002a), with increasing *NG* values, computation costs increase exponentially, because the central server conducts all computations. However, in the schemes, all train peers join the computation and the costs are split among them. They perform such computations in parallel.

To sum up, supplementary costs due to privacy concerns are inevitable, because privacy, accuracy, and performance are conflicting goals. Extra costs are negligible and the introduced scheme is still able to provide predictions efficiently. Moreover, the peers can determine the values of data disguising parameters in such a way to reach required levels of privacy and performance.

6.6. Accuracy Analysis: Experiments

Due to privacy concerns, accuracy might become worse. To evaluate how various privacy concerns and their solutions on randomization affect the accuracy of P2P NBC-based schemes, various experiments are performed using Jester and EM. To measure the accuracy of the schemes, CA and F1 metric are used.

Firstly, numerical ratings are transformed into binary. Using the similar methodology in (Miyahara and Pazzani, 2002), if the numerical rating for the item is bigger than 0.5 are labeled as 1, or 0 otherwise in EM, while if the numerical rating for the item is above 2.0 are labeled as 1, or 0 otherwise in Jester. The users who rated more than 60 items from both data sets are selected. They are randomly divided into two disjoint sets, training and test. For each experiment, the required numbers of train and test users from train and test sets are randomly selected, respectively, based on the experiment requirements. For each test user (active user or peer), five rated items are randomly picked, they are replaced with null, and are tried to predict. Predicted votes are compared with true withheld ratings. Each trial is performed 100 times to obtain trustworthy results. After computing CA and F1 values, final overall outcomes are demonstrated. As stated in (Kim et al., 2008), using the latest ratings makes accuracy better. After a certain point, old ratings reflect the current ratings and leads to lower predictions quality. Therefore, using recent ratings enhance accuracy (Kim et al., 2008). In the proposed scheme, the most up to date ratings are proposed to use and the dated ratings are removed to prevent privacy attacks. Since the effects of using current ratings are explained in (Kim et al., 2008), experiments are not performed for this purpose.

Experiment 1: To show how filling some of *a*'s unrated items' cells with fake ratings affects accuracy, trials are performed using both data sets. 1,000 and 500 training and test users are employed, respectively, from both data sets. α_a is varied from 0 to 100 to show how different δ_a values affect accuracy. Note that when α_a is 0, *a* does not fill any unrated items' cells. Once α_a is determined, $\delta_{a\ is}$ uniformly randomly selected over the range [0, α_a]. After CA and F1 values are calculated for both data sets with varying α_a values, they are displayed in Figure 6.1 and Figure 6.2, respectively.

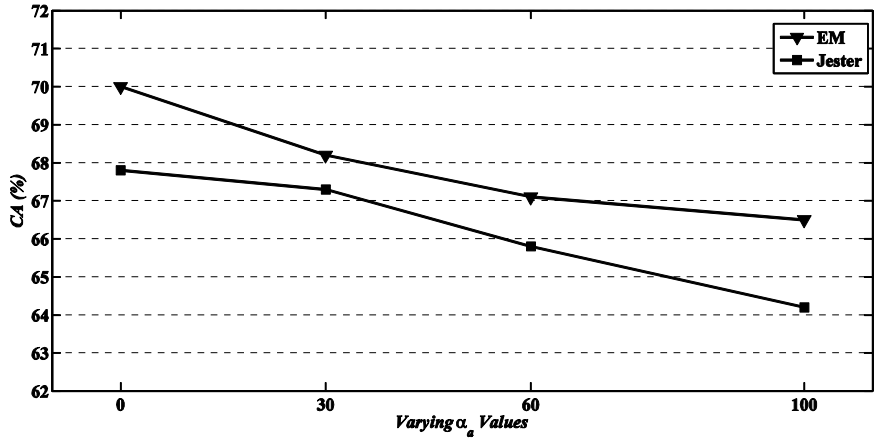

Figure 6.1. CA with Varying α_a Values

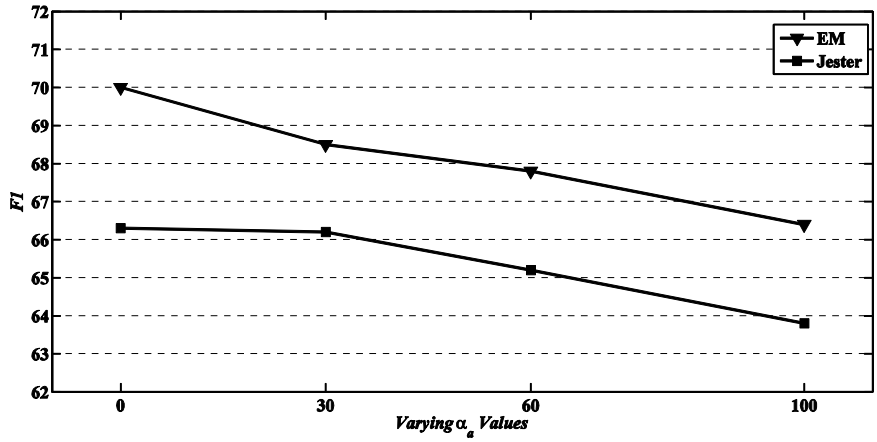

Figure 6.2. F1 with Varying α_a Values

As seen from Figure 6.1 and Figure 6.2, accuracy slightly becomes worse with increasing α_a values. Although the same number of 1s and 0s are inserted into some of the unrated items' cells, such fake ratings might not represent *a*'s true preferences about those unrated items. Therefore, it can be said that randomness boosts with augmenting α_a values; and it is expected that the quality

of the predictions becomes poorer. However, accuracy losses due to inserting fake ratings into some randomly selected unrated items' cells are negligible and it is still possible to offer referrals with decent accuracy. For Jester, when α_a is 60, CA losses are 2% only. When it is 100, they are more than 3%. In terms of F1 metric, the results are slightly better compared to CA. The results are similar for EM data set.

Experiment 2: In the second set of experiments, how inserting bogus votes into the train users' randomly selected unrated items' cells affects the results are demonstrated. Due to the same reasons explained in the first set of experiments, quality of the predictions might be affected by inserting false votes into train users' unrated items' cells. To show such affects, trials are performed using both data sets. Again the same 1,000 and 500 training and test users are utilized, respectively from both sets. α_g is varied from 0 to 100 to show how different δ_g values affect accuracy. Once again, after α_i is determined, δ_g is uniformly randomly selected over the range [0, α_i]. After CA and F1 values are calculated for both data sets, the outcomes are demonstrated in Figure 6.3 and Figure 6.4, respectively.

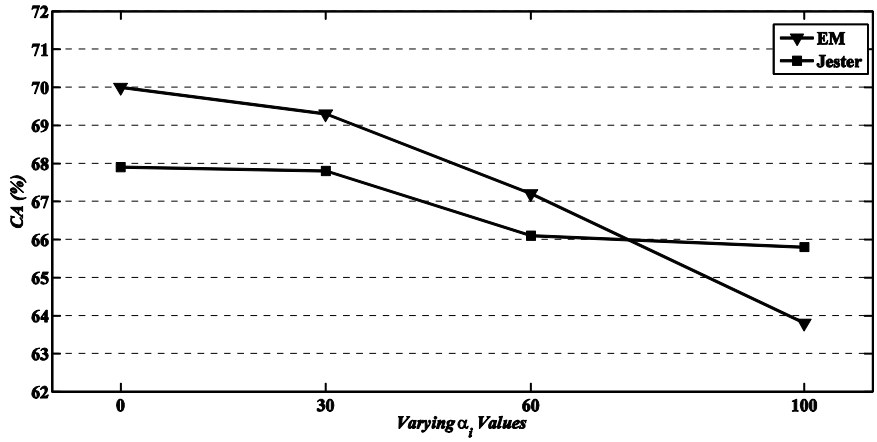

Figure 6.3. CA with Varying α_g Values

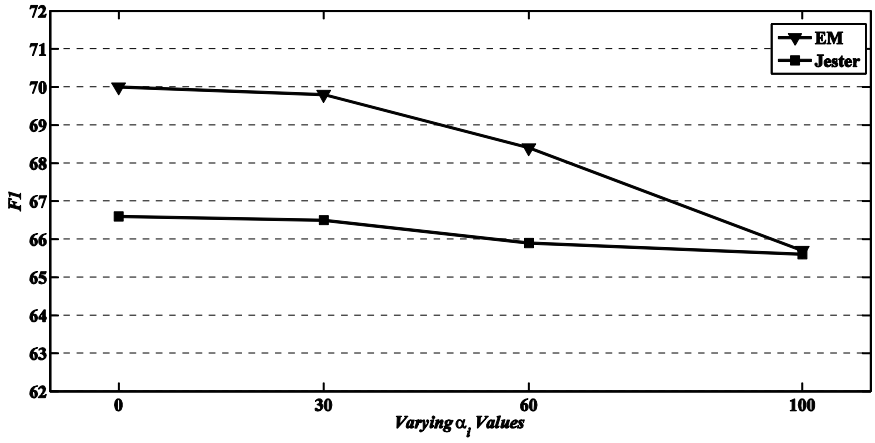

Figure 6.4. F1 with Varying α_g Values

As seen from Figure 6.3 and Figure 6.4; and as expected, the quality of the recommendations gets worse with increasing α_g values. For both data sets, the results in terms of CA and F1 are similar and get somewhat worse. Due to the same reasons explained previously, randomness increases with augmenting α_g values. As expected, accuracy worsens with increasing randomness. However, as seen from figures, accuracy losses due to privacy concerns are small and the proposed schemes still make it possible to offer accurate referrals.

Experiment 3: Finally, experiments are performed to show how different numbers of the peers involving the prediction computations affect the overall performance of the schemes. Remember that the peers decide whether to join the recommendation process or not based on the comparison of λ_g and r_g values. Experiments are conducted using both data sets while numbers of train peers (n_p) are varied from 100 to 2,000 for both data sets. 500 test users are employed. Overall averages of CA and F1 values are calculated. Since the results are similar, CA values are displayed only in Figure 6.5. Note that without privacy concerns, those peers who rated q involve in prediction generation. However, with privacy concerns, the peers join the recommendation process based on the comparison of λ_g and r_g values. If they did not rate q, they fill its cell with their default vote (v_{gd}).

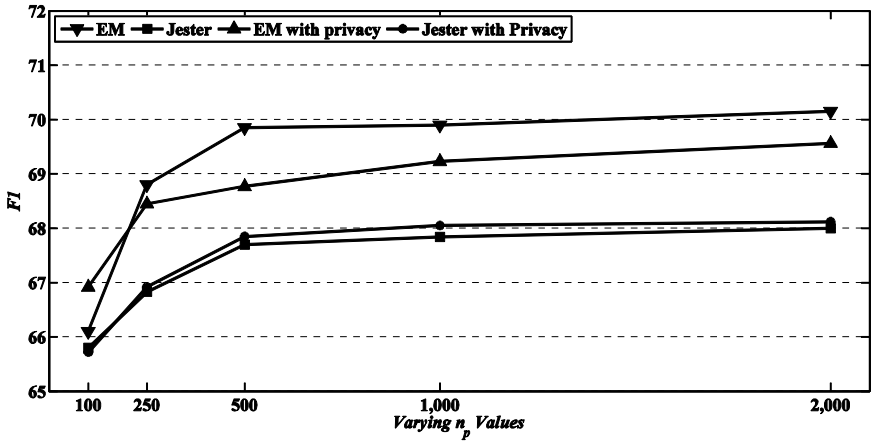

Figure 6.5. CA with Varying n_p Values

Without privacy concerns, for both data sets, the quality of the recommendations turns out to be better with increasing n_p values. Although with increasing number of peers involving in recommendation computations, CA enhances; however, after 500 peers, such improvements become stable. Once there are enough peers to generate referrals, increasing the number of train peers does not enhance the results too much. Although the results barely become worse with privacy concerns for EM data set, accuracy somewhat improves for Jester. It is expected that the quality of the predictions worsens with privacy concerns due to randomness. Improvements in Jester with privacy concerns might be explained the density of Jester. EM data set is sparse while Jester is a dense set; and 50% of all ratings are available. Therefore, v_{id} values can be estimated with decent accuracy and they may represent peers' true preferences. Moreover, number of peers involving recommendation processes increase. To sum up, the results are promising; and they make it possible to generate truthful predictions.

6.7. Conclusions

In this chapter, how to provide accurate P2P network-based referrals efficiently using NBC are investigated. Moreover, it is still possible to generate such recommendations while preserving the users' privacy is shown. Also, it is demonstrated that NBC-based predictions can be provided without a central server. Close community groups are able to construct P2P network for various purposes including CF. Without a central authority, the control of the central server is broken; and the recommendation computation workload is split among

the peers. Therefore, performance is expected to enhance significantly. Real data-based experiments are performed to evaluate the overall performance of the scheme. The results show that accuracy losses due to privacy concerns are negligible and make it possible to offer accurate recommendations. Also, the scheme is analyzed in terms of privacy and overhead costs. It is shown that the scheme is secure and the overhead costs are small.

7. CONCLUSIONS AND FUTURE WORK

In this dissertation, various solutions are proposed to overcome challenges caused by having inadequate data via collaboration among multiple parties while preserving their confidentiality. The proposed schemes are analyzed in terms of privacy, supplementary costs, and accuracy. Real data-based experiments are conducted. In order to test significance of the experimental outcomes, *t*-tests are performed.

According to experimental results, it is possible to produce accurate recommendations form distributed data (vertically or horizontally) among multiple parties without jeopardizing their confidentiality. It is shown that collaboration of parties handles *cold start* and *coverage* problems. Besides overcoming challenges caused by data sparsity, partnership increases accuracy of recommendations. Thus, e-companies can employ the proposed solutions until they have sufficient user data.

In order to provide privacy, randomized perturbation techniques, randomized response techniques, cryptographic methods, random filling, private sorting algorithm, and so on are employed. Although cryptographic methods do not affect accuracy of recommendations, randomized methods have impact on quality of referrals. Hence, several experiments are performed to determine effects of randomized techniques. The outcomes demonstrate that although utilizing randomized methods might decrease recommendation accuracy, they are useful for keeping confidentiality of data holders. Furthermore, increase in accuracy provided by collaboration is higher than losses in accuracy due to randomization. Consequently, the utilized privacy-preserving techniques provide confidentiality to parties while partnership increases accuracy significantly.

Another important obstacle against collaboration is scalability. In this dissertation, several solutions are proposed to enhance performance of partnership. Distributed recommendation processes are divided into two groups, off-line and online, and many of the required distributed computations are done off-line. Thus, in each solution, an off-line model is constructed. The solution in Chapter 2 employs self-organizing map clustering to determine the nearest neighbors of an active user in collaborative filtering. Besides clustering,

dimension reduction techniques are utilized to reduce dimension of data used in recommendations process. To reduce dimension, random projection is carried out. In Chapter 3, secure protocols for random projection-based distributed collaborative filtering are introduced. According to experimental results, random projection improves online performance of not only distributed recommendations but also improves scalability of general collaborative filtering schemes. A private secure multi-party protocol, which determines the nearest neighbors of active users from distributed values, is introduced. In Chapter 4, it is shown that it is possible to increase scalability by constructing a distributed off-line trust network. The trust values are employed instead of similarity in recommendation process. Again, secure multi-party protocols are proposed to compute distributed trust values. Although constituting off-line models improves performance of collaboration, note that the parties need to update their models periodically to insert new users' data into the models.

Besides numerical ratings-based solutions, methods for enabling collaboration of parties for producing binary recommendations are introduced in Chapter 5. The experimental outcomes are promising and collaboration enhances accuracy of referrals. Compared to numerical ratings, it is more common to collect binary ratings via market basket analysis. The proposed solutions help online vendors provide predictions on distributed binary ratings without jeopardizing their confidentiality.

In Chapter 6, a special kind of horizontally distributed data is studied. The proposed solution enables individuals to control their own rating data and get private binary recommendations. Since workload of recommendation process is distributed among the users in P2P network, scalability of the system enhances significantly.

There are remaining works to be done in order to show the effects of overlapped ratings in the distributed databases. In order to enhance cryptographic techniques-based solutions, new secure-multi party computation protocols might be suggested. Also, in order to improve online performance of collaboration, new methods can be proposed. Besides horizontal or vertical distributions, data might be distributed arbitrarily among multiple parties. Thus, solutions are needed to be

introduced enabling partnership when data distribution is arbitrary. Besides privacy of individuals in P2P networks, protocols for protecting privacy of users utilizing mobile devices to get recommendations should be studied.

REFERENCES

Achlioptas, D. (2001), "Database-friendly random projections," *Proceedings of the 20th ACM SIGMOD Symposium on Principles of Database Systems*, Santa Barbara, CA, USA, 274-281.

Agrawal, R. and Srikant, R. (2000), "Privacy-preserving data mining," *Proceedings of the 19th ACM SIGMOD International Conference on Management of Data*, Dallas, TX, USA, 439-450.

Ahmad, W. and Khokhar, A. (2007), "An architecture for privacy-preserving collaborative filtering on Web portals," *Proceedings of the 3rd* IEEE *International Symposium on Information Assurance and Security*, Manchester, UK, 273-278.

Ahn, H.J. (2008), "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem," *Information Sciences*, **178** (1), 37-51.

Ahn, J.W. and Amatriain, X. (2010), "Towards fully distributed and privacy-preserving recommendations via expert collaborative filtering and RESTful linked data," *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, Toronto, ON, Canada, 66-73.

Aïmeur, E., Brässard, G., Fernandez, J.M. and Onana, F.S.M. (2008), "Alambic: A privacy-preserving recommender system for electronic commerce," *International Journal of Information Security*, **7** (5), 307-334.

Avesani, P., Massa, P. and Tiella, R. (2005), "A trust-enhanced recommender system application: Moleskiing," *Proceedings of the ACM Symposium on Applied Computing*, Santa Fe, NM, USA, 1589-1593.

Basu, A., Kikuchi, H. and Vaidya, J. (2011a), "Privacy-preserving weighted Slope One predictor for item-based collaborative filtering," *Proceedings of the International Workshop on Trust and Privacy in Distributed Information Processing*, Copenhagen, Denmark.

Basu, A., Kikuchi, H. and Vaidya, J. (2011b), "Efficient privacy-preserving collaborative filtering based on the weighted Slope One predictor," *Journal of Internet Services and Information Security*, **1** (4), 26-46.

Basu, A., Vaidya, J., Dimitrakos, T. and Kikuchi, H. (2012), "Feasibility of a privacy preserving collaborative filtering scheme on the Google App Engine – A performance case study," *Proceedings of the 27th ACM Symposium on Applied Computing***,** Trento, Italy, 2012**.**

Berkovsky, S., Eytani, Y., Kuflik, T. and Ricci, F. (2005), "Privacy-enhanced collaborative filtering," *Proceedings of the Workshop on Privacy-Enhanced Personalization, at the International Conference on User Modeling*, Edinburgh, UK, 75-83.

Berry, M. and Linoff, G. (2000), "*Mastering data mining*," JohnWiley & Sons, New York.

Bertino, E., Lin, D. and Jiang, W. (2008), "A survey of quantification of privacy preserving data mining algorithms," *Privacy-Preserving Data Mining: Models and Algorithms*, C. C. Aggarwal and P. S. Yu, New York Springer**,** 183-202.

Bilge, A. and Polat, H. (2010), "Improving privacy-preserving NBC-based recommendations by preprocessing," *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, Toronto, ON, Canada, 143-147.

Bilge, A. and Polat, H. (2011), "An improved profile-based CF scheme with privacy," *Proceedings of the 5th IEEE International Conference on Semantic Computing*, Palo Alto, CA, USA, 133-140.

Bilge, A. and Polat, H. (2012), "An improved privacy-preserving DWT-based collaborative filtering scheme," *Expert Systems with Applications*, **39**, 3841-3854.

Billsus, D. and Pazzani, M.J. (1998), "Learning collaborative information filters," *Proceedings of the 15th International Conference on Machine Learning*, Madison, WI, USA, 46-54.

Bingham, E. and Mannila, H. (2001), "Random projection in dimensionality reduction: Applications to image and text data," *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, USA, 245-250.

Bobadilla, J., Ortega, F., Hernando, A. and Alcalá, J. (2011), "Improving collaborative filtering recommender system results and performance using genetic algorithms," *Knowledge-Based Systems*, **24** (8), 1310-1316.

Breese, J., Heckerman, D. and Kadie, C. (1998), "Empirical analysis of predictive algorithms for collaborative filtering," *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*, Madison, WI, USA, 43-52.

Burke, R.D. (2002), "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, **12** (4), 331-370.

Calandrino, J. A., Kilzer, A., Narayanan, A., Felten, E.W. and Shmatikov, V. (2011), "'You might also like:' Privacy risks of collaborative filtering," *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 231-246.

Canny, J. (2002a), "Collaborative filtering with privacy," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 45-57.

Canny, J. (2002b), "Collaborative filtering with privacy via factor analysis," *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, 238-245.

Chandrashekhar, H. and Bhasker, B. (2007), "Collaborative filtering based on the entropy measure," *Proceedings of the 9th IEEE International Conference on E-Commerce Technology*, Tokyo, Japan, 203-210.

Chee, S.H.S., Han, J. and Wang, K. (2001), "RecTree: An efficient collaborative filtering method," *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery*, Munich, Germany, 141-151.

Chen, G., Wang, F. and Zhang, C. (2009), "Collaborative filtering using orthogonal nonnegative matrix tri-factorization," *Information Processing & Management*, **45** (3), 368-379.

Chua, F.C.T. and Lim, E.P. (2010), "Trust network inference for online rating data using generative models," *Proceedings of the 16th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 889-898.

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X. and Zhu, M.Y. (2002), "Tools for privacy preserving distributed data mining," *SIGKDD Explorations Newsletter*, **4** (2), 28-34.

Cranor, L.F. (2003), "'I didn't buy it for myself' privacy and e-commerce personalization," *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, Washington, DC, USA, 111-117.

Dokoohaki, N., Kaleli, C., Polat, H. and Matskin, M. (2010), "Achieving optimal privacy in trust-aware social recommender systems," *Social Informatics*, **6430,** 62-79.

Duan, Y. and Canny, J. (2009), "How to deal with malicious users in privacy-preserving distributed data mining," *Statistical Analysis and Data* Mining, **2** (1), 18-33.

Dung, L.T., Bao, H.T., Binh, N.T. and Hoang, T.H. (2010), "Privacy-preserving classification in two-dimension distributed data," *Proceedings of the 2nd International Conference on Knowledge and Systems Engineering*, Washington, DC, USA, 96-103.

Dung, L.T. and Bao, H.T. (2011), "A distributed solution for privacy-preserving outlier detection," *Proceedings of the 3rd International Conference on Knowledge and Systems Engineering*, Washington, DC, USA, 26-31.

Emekci, F., Sahin, O.D., Agrawal, D. and El Abbadi, A. (2007), "Privacy-preserving decision tree learning over multiple parties," *Data & Knowledge Engineering*, **63** (2), 348-361.

Erkin, Z., Veugen, T. and Lagendijk, R.L. (2011), "Generating private recommendations in a social trust network," *Proceedings of the International Conference on Computational Aspects of Social Networks*, Salamanca, Spain, 82-87.

Fern, X. and Brodley, C. (2003), "Random projection for high dimensional data clustering: A cluster ensemble approach," *Proceedings of the 20th International Conference on Machine Learning*, Washington, DC, USA, 86-193.

Gambs, S., Kégl, B. and Aïmeur, E. (2007), "Privacy-preserving boosting," *Data Mining and Knowledge Discovery*, **14** (1), 131-170.

Goldberg, D., Nichols, D., Oki, B.M. and Terry, D. (1992), "Using collaborative filtering to weave an information Tapestry," *ACM Communications*, **35** (12), 61-70.

Goldberg, K., Roeder, T., Gupta, D. and Perkins, C. (2001), "Eigentaste: A constant time collaborative filtering algorithm," *Information Retreival*, **4** (2), 133-151.

Grcar, M. (2004), "User profiling: Collaborative filtering," *Proceedings of the 7th International Multiconference Information Society*, Ljubljana, Slovenia, 75-78.

Hajian, S. and Azgomi, M.A. (2011), "A privacy-preserving clustering technique for horizontally and vertically distributed datasets," *Intelligent Data Analysis*, **15** (4), 503-532.

Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R. and Kadie, C. (2001), "Dependency networks for inference, collaborative filtering, and data visualization," *Journal of Machine Learning Research*, **1**, 49-75.

Herlocker, J.L., Konstan, J.A., Borchers, A. and Riedl, J.T. (1999), "An algorithmic framework for performing collaborative filtering," *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, CA, USA, 230-237.

Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T. (2004), "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, **22** (1), 5-53.

Hsieh, C.L.A., Zhan, J., Zeng, D. and Feiyue, W. (2008), "Preserving privacy in joining recommender systems," *Proceedings of the International Conference on Information Security and Assurance*, Busan, Korea, 561-566.

Hofmann, T. (2004), "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, **22** (1), 89-115.

Hwang, C.S. and Chen, Y.P. (2007), "Using trust in collaborative filtering recommendation," *New Trends in Applied Artificial Intelligence*, **4570,** 1052-1060.

Hwang, S.Y. and Chen, L.S. (2009), "Using trust for collaborative filtering in e-commerce," *Proceedings of the 11th International Conference on Electronic Commerce*, Taipei, Taiwan, 240-248.

Inan, A., Kaya, S.V., Saygin, Y., Savas, E., Hintoglu, A.A. and Levi, A. (2007), "Privacy preserving clustering on horizontally partitioned data," *Data & Knowledge Engineering*, **63** (3), 646-666.

Jagannathan, G., Pillaipakkamnatt, K. and Wright, R.N. (2006), "A new privacy-preserving distributed *k*-clustering algorithm," *Proceedings of the 6th SIAM International Conference on Data Mining*, Bethesda, MD, USA, 494-498.

Jin, R., Chai, J.Y. and Si, L. (2004), "An automatic weighting scheme for collaborative filtering," *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Sheffield, UK, 337-344.

Johnson, W.B. and Lindenstrauss, J. (1984), "Extensions of lipschitz mapping into Hilbert space," *Proceedings of the Conference in Modern Analysis and Probability*, New Haven, CT, USA, 189-206.

Kaleli, C. and Polat, H. (2007a), "Providing naïve Bayesian classifier-based private recommendations on partitioned data," *Lecture Notes in Computer Science*, **4702,** 515-522.

Kaleli, C. and Polat, H. (2007b), "Providing private recommendations using naïve Bayesian classifier," *Advances in Intelligent Web Mastering*, **43,** 168-173.

Kaleli, C. and Polat, H. (2009), "Similar or dissimilar users? or both?," *Proceedings of the 2nd International Symposium on Electronic Commerce and Security*, Nanchang, China, 184-189.

Kaleli, C. and Polat, H. (2010), "P2P collaborative filtering with privacy," *Turkish Journal of Electric Electrical Engineering & Computer Sciences*, **8** (1), 101-116.

Kaleli, C. and Polat, H. (2011), “Privacy-preserving trust-based recommendations on vertically distributed data,” *Proceedings of the 5th IEEE International Conference on Semantic Computing*, Palo Alto, CA, USA, 376-379.

Kaleli, C. and Polat, H. (2012a), "SOM-based recommendations with privacy on multi-party vertically distributed data," *Journal of the Operational Research Society*, **63** (6), 826–838.

Kaleli, C. and Polat, H. (2012b), "Privacy-preserving SOM-based recommendations on horizontally distributed data," *Knowledge-Based Systems*, doi: 10.1016/j.knosys.2012.02.013.

Kantarcioglu, M. and Clifton, C. (2004), "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Transaction on Knowledge and Data Engineering*, **16** (9), 1026-1037.

Kargupta, H., Liu, K. and Ryan, J. (2003), "Privacy sensitive distributed data mining from multi-party data," *Intelligence and Security Informatics*, **2665,** 960-960.

Kaski, S. (1998), "Dimensionality reduction by random mapping: Fast similarity computation for clustering," *Proceedings of the IEEE International Joint Conference on, Neural Networks and World Congress on Computational Intelligence,*Anchorage, AK, USA, 413-418.

Kaya, S.V., Pedersen, T.B., Savas, E. and Saygin, Y. (2007), "Efficient privacy-preserving distributed clustering based on secret sharing," *Proceedings of the International Conference on Emerging Technologies in Knowledge Discovery and Data Mining*, Nanjing, China, 280-291.

Kim, H.N., Ji, A.T., Ha, I. and Jo, G.S. (2010), "Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation," *Electronic Commerce Research and Applications*, **9** (1), 73-83.

Kim, J.K., Kim, H.K. and Cho, Y.H. (2008), "A user-oriented contents recommendation system in peer-to-peer architecture," *Expert Systems with Appications*, **34** (1), 300-312.

Lathia, N., Hailes, S. and Capra, L. (2007), "Private distributed collaborative filtering using estimated concordance measures," *Proceedings of the ACM Conference on Recommender Systems*, Minneapolis, MN, USA, 1-8.

Lathia, N., Hailes, S. and Capra, L. (2008), "Trust-based collaborative filtering," *Trust Management II*, Springer, Boston, **263,** 119-134.

Li, D., Lv, Q., Xia, H., Shang, L., Lu, T. and Gu. N. (2011), "Pistis: A privacy-preserving content recommender system for online social communities," *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, Lyon, France, 79-86.

Lin, X., Clifton, C. and Zhu, M. (2005), "Privacy-preserving clustering with distributed EM mixture modeling," *Knowledge and Information Systems*, **8** (1), 68-81.

Liu, F. and Lee, H.J. (2010), "Use of social network information to enhance collaborative filtering performance," *Expert Systems with Applications*, **37** (7), 4772-4778.

Liu, K., Kargupta, H. and Ryan, J. (2006), "Random projection-based multiplicative data perturbation for privacy-preserving distributed data mining," *IEEE Transactions on Knowedge and Data Engineering*, **18** (1), 92-106.

Lodi, S., Moro, G. and Sartori, C. (2010), "Distributed data clustering in multi-dimensional P2P networks," *Proceedings of the 21st Australasian Conference on Database Technologies*, Darlinghurst, Australia, 171-178.

Machanavajjhala, A., Korolova, A. and Sarma, A.D., (2011), "Personalized social recommendations: Accurate or private," *Proceedings of the VLDB Endowment*, **4**, 440-450.

Massa, P. and Avesani, P. (2004), "Trust-aware collaborative filtering for recommender systems on the move to meaningful Internet systems," *Proceedings of the Federated International Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE*, Agia Napa, Cyprus, 492-508.

Massa, P. and Avesani, P. (2007), "Trust-aware recommender systems," *Proceedings of the ACM International Conference on Recommender Systems*, Minneapolis, MN, USA, 17-24.

Melville, P., Mooney, R.J. and Nagarajan, R. (2002), "Content-boosted collaborative filtering for improved recommendations," *Proceedings of the*

8th International Conference on Artificial Intelligence, Alberta, Canada, 187-192.

Miyahara, K. and Pazzani, M.J. (2002), "Improvement of collaborative filtering with the simple Bayesian classifier," *The Information Processing Society of Japan Journal*, **43** (11), 3429-3437.

O'Donovan, J. and Smyth, B. (2005), "Trust in recommender systems," *Proceedings of the 10th International Conference on Intelligent User Interfaces*, San Diego, CA, USA, 167-174.

O'Connor, M. and Herlocker, J.L. (2001), "Clustering items for collaborative filtering," *SIGIR Workshop on Recommender Systems*, New Orleans, MS, USA.

OECD (2000). Guidelines for Consumer Protection in the Context of Electronic Commerce.

OECD (2005). Guidelines on the Protection of Privacy and Transborder Flows of Personal Data.

Oliveira, S., Oliveira, S.R.M. and Zaïane, O.R. (2004), "Toward standardization in privacy-preserving data mining," *Proceedings of the 3rd Workshop on Data Mining Standards in conjuction with KDD 2004*, Seattle, WA, USA, 7-17.

Oliveira, S.R.M., Zaïane, O.R. (2002), "Privacy-preserving frequent itemset mining," *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining*, Maebashi City, Japan, 43-54.

Oliveira, S.R.M. and Zaïane, O.R. (2007), "A privacy-preserving clustering approach toward secure and effective data analysis for business collaboration," *Computers & Security*, **26** (1), 81-93.

Paillier, P. (1999), "Public-key cryptosystems based on composite degree residuosity classes," *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, Prague, Czech Republic, 223-238.

Parameswaran, R. and Blough, D.M. (2007), "Privacy preserving collaborative filtering using data obfuscation," *Proceedings of the IEEE International Conference on Granular Computing*, San Jose, CA, USA, 380-386.

Pazzani, M.J. (1999), "A framework for collaborative, content-based and demographic filtering," *Artificial Inteligence Review*, **13** (5), 393-408.

Pennock, D.M., Horvitz, E., Lawrence, S. and Giles, C.L. (2000), "Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach," *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Stanford, CA, USA, 473-480.

Polat, H. and W. (2003), "Privacy-preserving collaborative filtering using randomized perturbation techniques," *Proceedings of the 3rd IEEE International Conference on Data Mining*, Melbourne, FL, USA, 625-628.

Polat, H. and Du, W. (2005), "Privacy-preserving collaborative filtering," *International Journal of Electronic Commerce*, **9** (4), 9-35.

Polat, H. and Du, W. (2006), "Achieving private recommendations using randomized response techniques," *Proceedings of the 10th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, Singapore, 637-646.

Polat, H. and Du, W. (2007), "Effects of inconsistently masked data using RPT on CF with privacy," *Proceedings of the ACM Symposium on Applied Computing*, Seoul, Korea, 649-653.

Polat, H. and Du, W. (2008), "Privacy-preserving top-*N* recommendation on distributed data," *Journal of the American Society for Information Science and Technology*, **59** (7), 1093-1108.

Popescul, A., Ungar, L.H., Pennock, D.M. and Lawrence, S. (2001), "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments," *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, Seattle, WA, USA, 437-444.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J.T. (1994), "GroupLens: An open architecture for collaborative filtering of Netnews," *Proceedings of the ACM conference on Computer Supported Cooperative Work*, Chapel Hill, NC, USA, 175-186.

Roh, T.H., Oh, K.J. and Han, I. (2003), "The collaborative filtering recommendation based on SOM cluster-indexing CBR," *Expert Systems with Applications*, **25** (3), 413-423.

Russell, S. and Yoon, V. (2008), "Applications of wavelet data reduction in a recommender system," *Expert Systems with Applications*, **34** (4), 2316-2325.

Sang, Y. and Shen, H. (2009), "Efficient and secure protocols for privacy-preserving set operations," *ACM Transactions on Information System Security*, **13** (1), 1-35.

Sarwar, B.M., Karypis, G., Konstan, J.A and Reidl, J.T. (2001), "Item-based collaborative filtering recommendation algorithms," *Proceedings of the 10th International Conference on World Wide Web*, Hong Kong, 285-295.

Sarwar, B.M., Karypis, G., Konstan, J.A. and Riedl, J.T. (2000), "Application of dimensionality reduction in recommender system - A case study," *Proceedings of the Web Mining for E-Commerce*, Boston, MA, USA.

Shardanand, U. and Maes, P. (1995), "Social information filtering: Algorithms for automating 'word of mouth'," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Denver, CO, USA, 210-217.

Shen, H., Zhao, J. and Yao, R. (2006), "Privacy-preserving mining of global association rules on distributed dataset," *Lecture Notes in Computer Science*, **3975**, 654-656.

Shokri, R., Pedarsani, P., Theodorakopoulos, G. and Hubaux, J.P. (2009), "Preserving privacy in collaborative filtering through distributed aggregation of offline profiles," *Proceedings of the 3rd ACM International Conference on Recommender Systems*, New York, NY, USA, 157-164.

Shundong, L., Daoshun, W., Yiqi, D. and Ping, L. (2008), "Symmetric cryptographic solution to Yao's millionaires' problem and an evaluation of secure multiparty computations," *Information Sciences*, **178** (1), 244-255.

Su, X. and Khoshgoftaar, T.M. (2006), "Collaborative filtering for multi-class data using belief nets algorithms," *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, Washington, DC, USA, 497-504.

Su, X., Greiner, R., Khoshgoftaar, T.M. and Zhu, X. (2007), "Hybrid collaborative filtering algorithms using a mixture of experts," *Proceedings*

of the IEEE/WIC/ACM International Conference on Web Intelligence, Silicon Valley, CA, USA, 645-649.

Su, X. and Khoshgoftaar, T.M. (2009), "A survey of collaborative filtering techniques," *Advances in Artifical Intelligence*, **2009**, 1-19.

Ungar, L. and Foster, D. (1998), "Clustering methods for collaborative filtering," *Proceedings of the Workshop on Recommendation Systems*, Menlo Park, CA, USA.

Vaidya, J. and Clifton, C.W. (2003), "Privacy-preserving *k*-means clustering over vertically partitioned data," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 206-215.

Vaidya, J. and Clifton, C.W. (2004), "Privacy preserving naïve Bayes classifier for vertically partitioned data," *Proceedings of the 4th SIAM International Conference on Data Mining*, Lake Buena Vista, FL, USA, 522-526.

Vaidya, J., Clifton, C.W., Kantarcioglu, M. and Patterson, A.S. (2008), "Privacy-preserving decision trees over vertically partitioned data," *ACM Transactions on Knowledge Discovery and Data Mining*, **2** (3), 1-27.

Vaidya, J. and Clifton, C.W. (2009), "Privacy-preserving k^{th} element score over vertically partitioned data," *IEEE Transactions on Knowledge and Data Engineering*, **21** (2), 253-258.

Vucetic, S. and Obradovic, Z. (2005), "Collaborative filtering using a regression-based approach," *Knowledge and Information Systems*, **7** (1), 1-22.

Walter, F., Battiston, S. and Schweitzer, F. (2008), "A model of a trust-based recommendation system on a social network," *Autonomous Agents and Multi-Agent Systems*, **16** (1), 57-74.

Walter, F.E., Battiston, S. and Schweitzer, F. (2009), "Personalised and dynamic trust in social networks," *Proceedings of the 3rd ACM International Conference on Recommender Systems*, New York, NY, USA, 197-204.

Wang, S.L., Lai, T.Z., Hong, T.P. and Wu, Y.L. (2010), "Hiding collaborative recommendation association rules on horizontally partitioned data," *Intelligent Data Analysis*, **14** (1), 47-67.

Warner, S.L. (1965), "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, **60** (309), 63-65.

Westin, A. (1999), "'Freebies' and pivacy: What net users think," Opinion Research Corporation, Sponsored by Privacy & American Business.

Xie, B., Han, P., Yang, F., Shen, R.M., Zeng, H.J. and Chen, Z. (2007), "DCFLA: A distributed collaborative-filtering neighbor-locating algorithm," *Information Sciences*, **177** (6), 1349-1363.

Xiong, L., Chitti, S. and Liu, L. (2007), "Mining multiple private databases using a *k-nn* classifier," *Proceedings of the ACM Symposium on Applied Computing*, New York, NY, USA, 435-440.

Xue, G.R., Lin, C., Yang, Q., Xi, W., Zeng, H.J., Yu, Y. and Chen, Z. (2005), "Scalable collaborative filtering using cluster-based smoothing," *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Salvador, Brazil, 114-121.

Yakut, I. and Polat, H. (2007), "Privacy-preserving Eigentaste-based collaborative filtering," *Proceedings of the 2nd International Conference on Advances in Information and Computer Security*, Nara, Japan, 169-184.

Yakut, I. and Polat, H. (2010), "Privacy-preserving SVD-based collaborative filtering on partitioned data," *International Journal of Information Technology & Decision Making*, **9** (3), 473-502.

Yakut, I. and Polat, H. (2012a), "Arbitrarily distributed data-based recommendations with privacy," *Data & Knowledge Engineering*, **72**, 239-256.

Yakut, I. and Polat, H. (2012b), "Privacy-preserving hybrid collaborative filtering on cross distributed data," *Knowledge and Information Systems*, **30** (2), 405-433.

Yang, Z. and Wright, R.N. (2006), "Privacy-preserving computation of Bayesian networks on vertically partitioned data," *IEEE Transactions on Knowledge and Data Engineering*, **18** (9), 1253-1264.

Yi, L., Weichao, W., Bhargava, B. and Dongyan, X. (2006), "Trust-based privacy preservation for peer-to-peer data sharing," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, **36** (3), 498-502.

Yu, K., Schwaighofer, A., Tresp, V., Xu, X. and Kriegel, H.P. (2004), "Probabilistic memory-based collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, **16** (1), 56-69.

Yuan, W., Guan, D., Lee, Y.K., Lee, S. and Hur, S.J. (2010), "Improved trust-aware recommender system using small-worldness of trust networks," *Knowledge-Based Systems*, **23** (3), 232-238.

Zarghami, A., Fazeli, S., Dokoohaki, N. and Matskin, M. (2009), "Social trust-aware recommendation system: A T-index approach," *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web*, Milano, Italy, 85-90.

Zhan, J., Wang, I.C., Hsieh, C.L., Hsu, T.S., Liau, C.J. and Wang, D.W. (2010), "Privacy-preserving collaborative recommender systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, , **40** (4), 472-476.

Zhan, J., Wang, I.C., Hsieh, C.L., Hsu, T.S., Liau, C.J. and Wang, D.W. (2008), "Towards efficient privacy-preserving collaborative recommender systems," *Proceedings of the IEEE International Conference on Granular Computing*, Hangzhou, China, 778-783.

Zhang, F. and Chang, H. (2006), "Research on privacy-preserving collaborative filtering recommendation based on distributed data," *Chinese Journal of Computers*, 1487-1495.

Zhang, S., Ford, J. and Makedon, F. (2006), "A privacy-preserving collaborative filtering scheme with two-way communication," *Proceedings of the 7th ACM Conference on Electronic Commerce*, Ann Arbor, MI, USA, 316-323.

Zhang, F., Zhao, G. and Xing, T. (2009), "Privacy-preserving distributed *k-nn* mining on horizontally partitioned multi-party data," *Lecture Notes in Computer Science*, **5678**, 755-762.

Zhong, S. (2007), "Privacy-preserving algorithms for distributed mining of frequent itemsets," *Information Sciences*, **177** (2), 490-503.

Ziegler, C.N. and Golbeck, J. (2007), "Investigating interactions of trust and interest similarity," *Decision Support Systems*, **43** (2), 460-475.

Ziegler, C.N., Lausen, G. and Schmidt-Thieme, L. (2004), "Taxonomy-driven computation of product recommendations," *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, Washington, DC, USA, 406-415.