Recommender systems are used to predict the interest of a customer on a particular item based on that customer’s ratings on other items. Many websites like Amazon and Netflix use such systems to recommend items of potential interest to their customers. In order to improve their recommendation service, the data owners sometimes publicly release all or part of their recommendation data i.e., the ratings of their customers on various items without any person specific detail like the customer names. Still, this released data could suffer from re-identification attacks compromising the customer’s privacy.

However, such releases in the past, like the one by Netflix, proved to be fruitful. So, in our work, we propose a technique to publish these recommendation datasets without compromising the privacy of the customers. The goal of this thesis is to provide better privacy and utility than the current solutions.
A Thesis

Submitted to the
Faculty of Miami University
in partial fulfillment of
the requirements for the degree of
Master of Computer Science
Department of Computer Science and Software Engineering
by
Jyothilakshmi Somasundaram
Miami University
Oxford, Ohio
2011

Advisor ________________________________
Dr. Keith Frikken

Committee Member ________________________________
Dr. William John Brinkman II

Committee Member ________________________________
Dr. Lukasz Opyrchal
# Contents

1 Introduction .................................................. 1

2 Background and Related Work ................................. 2
   2.1 Privacy in Relational Data .................................. 3
   2.2 Privacy in Transaction Data ................................. 4
   2.3 Privacy in Sparse Data ...................................... 5
   2.4 Privacy in Recommendation Data ......................... 6

3 General Overview .............................................. 6
   3.1 Problem Definition .......................................... 6
   3.2 Recommendation Dataset .................................... 6

4 Solution .......................................................... 8
   4.1 Initial processing of the data .............................. 8
   4.2 High Level Solution ......................................... 8
   4.3 Problems with implementing the solution ................... 9
   4.4 Variation to the solution .................................... 10
   4.5 Converting to an Assignment Problem ................. 10
   4.6 A quick example ............................................ 12

5 Quality Metrics .................................................. 13

6 System Design .................................................... 15

7 Experiments & Results ........................................ 18
   7.1 Ranking of algorithms ....................................... 18
      7.1.1 Datasets Used .......................................... 18
      7.1.2 Tools Used ............................................. 19
      7.1.3 Computing RMSE ...................................... 19
   7.2 Experiment Setup ............................................ 20
   7.3 Results & Analysis .......................................... 24
      7.3.1 Proposed Solution ..................................... 24
         7.3.1.1 Initial Results ................................... 24
         7.3.1.2 Non-zero Average .................................. 27
7.3.2 Random Users ................................................................. 30
  7.3.2.1 Flips & Delta ............................................................. 30
  7.3.2.2 Similar Results ....................................................... 32
  7.3.2.3 Increasing number of user sets ................................. 34
7.3.3 Scaling of the results ..................................................... 35
  7.3.3.1 For different percentages of users and items ............... 35
  7.3.3.2 For larger datasets ............................................... 35
7.3.4 Verifying the Evaluation Method ..................................... 37
8 Privacy & Utility of the Published Dataset ........................... 39
  8.1 Privacy ........................................................................... 39
  8.2 Utility ........................................................................... 40
9 Future Work ........................................................................ 41
10 Conclusion .......................................................................... 42
References .............................................................................. 43
## List of Tables

1. Examples for privacy models in relational data ............................................. 4
2. An example recommendation dataset showing the ratings of users to items ........ 7
3. Example cost matrix ...................................................................................... 12
4. The released recommendation dataset for the example in Figure 2 ...................... 13
**List of Figures**

1. An example to represent the privacy model described in [3] ........................................ 7
2. The bipartite graph corresponding to the recommendation dataset in Table 2 ................. 9
3. The two near isomorphic sub-graphs for the example in Figure 2 .................................. 13
4. The System Design ......................................................................................................... 16
5. The Common Format for Datasets ................................................................................... 17
6. Experiment Setup ............................................................................................................ 22
7. Proposed Experiment: RMSE values for the original and published datasets of the 100K Movielens dataset, and the corresponding difference in their RMSE values . 25
8. Proposed Experiment: Rankings for the original and published datasets of the 100K Movielens dataset, and the corresponding difference in their rankings . . . 26
9. Experiment with Non-Zero Average: RMSE values for the original and published datasets of the 100K Movielens dataset, and the corresponding difference in the RMSE values ................................................................. 27
10. Experiment with Non-Zero Average: % change in RMSE values for the original and published datasets of the 100K Movielens dataset .................................................... 28
11. Experiment with Non-Zero Average: Rankings for the original and published datasets of the 100K Movielens dataset, and the corresponding difference in the rankings . . 29
12. The delta values for the users in the 100K dataset .......................................................... 32
13. The flip values for the users in the 100K dataset ............................................................. 32
14. Experiment with Random Sets: The difference in RMSE values for the 100K dataset 33
15. Experiment with Random Sets: The difference in Rankings for the 100K dataset . . . 33
16. Experiment Averaging more Datasets: The average change between the original and the published datasets in terms of the rankings and RMSE values when averaging different number of datasets to obtain the final dataset ................................................. 34
17. Experiment varying the percentages of users and items: Shows the average change in the RMSE values and the rankings between the original and published datasets . 36
18. Experiment varying the dataset size: Shows the maximum and average change in the RMSE values and the rankings between the original and published datasets . . 36
19. Experiment with mismatching datasets: Shows the changes in the RMSE values between the mismatched datasets in terms of the maximum and average change . . 38
20. Experiment with mismatching datasets: Shows the changes in the rankings between the mismatched datasets in terms of the maximum and average change . . . 38
Experiment to determine the percentage of original users that a given user in the published dataset maps to
Acknowledgement

I would like to thank Dr. Keith Frikken for his support and guidance throughout this thesis process. I would also like to thank my committee members Dr. Brinkman and Dr. Opyrchal for willing to be in my committee, and their involvement in my thesis. I must also thank my family and friends for their help and encouragement through these two years of my Masters program.
1 Introduction

Many websites deploy recommender systems that attempt to recommend items (e.g., movies, music, books, web pages) to the consumers that might be of interest to them. Typically these systems seek to predict the interest of a user on a particular item based on the user’s interest on other items. To improve the prediction algorithms, data owners sometimes release all or part of their recommendation dataset, and this helps researchers create better recommendation algorithms. When releasing such data, it is essential that the privacy of the consumers be protected.

A classic example for such a recommendation dataset is the one released by Netflix, the world’s largest online DVD rental service, in 2006. The dataset contains over 100 million movie ratings of 500,000 Netflix subscribers, and a prize of $1 million was announced for improving their movie recommendation service. The released data was sanitized by removing the name of the subscribers and applying very little perturbation to the rest of the data. However, this released data suffered from a re-identification attack [9], where the identities of specific users were revealed. This attack used the Internet Movie Database1 as the source of background knowledge. It was demonstrated that 84% of the subscribers can be uniquely identified if the adversary knows 6 to 8 of their movie ratings outside the top 500 rated movies, which is not a significant limitation as more than 90% of the subscribers have at least five movies not in the top 500.

These re-identification problems have stifled such data releases. For example, Netflix planned for a sequel to the Netflix prize in 2009. However, in the wake of the vulnerability in the released dataset, the Federal Trade Commission (FTC) questioned about the Netflix member’s privacy and a lawsuit was filed by KamberLaw LLC against the Netflix prize sequel, leading to the withdrawal of the sequel.

However, it has to be noted that release of such data proved to be fruitful in creating better algorithms. Hence, we provide a technique for generating a recommendation dataset as a subset of

1IMDb. The Internet Movie Database. http://www.imdb.com
the original data available with the data owner, that could be publicly released and is secure from re-identification attack. The solution involves removing both the names of the users and also the item names (as these may be irrelevant for the algorithm development). Then, two subsets of the original dataset are generated, such that they have identical ratings. The two subsets have non-overlapping sets of consumers. The sets of items (movies) used in these subsets may or may not be disjoint, as this does not affect the privacy of the consumers. Then, when one of these subsets is published, it would be secure.

The rest of this document is organized as follows: Background and related work is reviewed in Section 2. The overview of the problem being considered is in Section 3. Our solution to this problem, and the problems and corresponding variations to it are explained in Section 4. In Section 5 we discuss on how we evaluate our solution detailing the various quality metrics used. The system design is presented in Section 6. The various experiments we performed, the tools that were used for it, and the analysis of the results obtained are detailed in Section 7. The discussion on how the dataset we published has better privacy and utility is in Section 8. Finally, the possible future works and conclusion are in Sections 9 and 10 respectively.

2 Background and Related Work

In this section, the prior work in the area of privacy in data publishing is discussed. The main goal of these works is to protect the privacy of the published data, while trying to preserve the utility of that data.

First, the privacy in relational data, the area in which many privacy models have been proposed is discussed. Next, the privacy in transaction data, which is more closer to recommendation data in terms of dimensionality than the relational data, and extends the privacy models of relational data is discussed. This is followed by the privacy model for handling sparse data. Finally, work in the area of privacy in recommendation data is explained.
Data when published is sanitized by removing person specific values like the name, and address. However, if it is still possible to associate the published data to a specific individual with a higher probability, then the data is said to suffer from re-identification attack.

One of the basic assumptions in most of these works is that there are two non-overlapping sets of attributes: quasi-identifier values, which can aid an adversary in re-identification, and sensitive values, which have to be protected from the attack.

### 2.1 Privacy in Relational Data

Privacy in data publishing has been studied extensively to prevent this re-identification attack in relational data. Table 1a shows a relational dataset. In this, the attributes Zip Code and Age are considered quasi identifiers, and the attribute Disease is considered to be sensitive.

Well established privacy models in this area include the k-anonymity [10], where a relational table is considered to be k-anonymous if for each record there are other k-1 indistinguishable records. This ensures that individuals cannot be uniquely identified by linking attacks, preventing identity disclosure. Table 1b shows the corresponding 3-anonymous table of Table 1a. However, this model suffers from the homogeneity and background knowledge attacks [8].

To overcome these attacks, another privacy model proposed for relational data is the L-diversity [8], in which a table is considered L-diverse if in each group at most 1/L of the tuples possess the same sensitive value. Thus, this prevents attribute disclosure. Table 1c shows the corresponding 3-diverse table of Table 1a.

These techniques are limited when publishing data in a recommendation system because, there are no disjoint sets, i.e., all values can be considered sensitive and as well serve as quasi-identifiers for re-identification. Even if the aforementioned privacy models are applied to the recommendation system, most of the items will be suppressed for such high-dimensional quasi-identifiers [1]. This is because there is an exponential number of combination of attributes that could be used in a re-identification attack.
Table 1: Examples for privacy models in relational data

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Zip Code</th>
<th>Age</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45056</td>
<td>22</td>
<td>Heart Disease</td>
</tr>
<tr>
<td>2</td>
<td>45057</td>
<td>29</td>
<td>Heart Disease</td>
</tr>
<tr>
<td>3</td>
<td>45058</td>
<td>24</td>
<td>Cancer</td>
</tr>
<tr>
<td>4</td>
<td>45059</td>
<td>28</td>
<td>Bronchitis</td>
</tr>
<tr>
<td>5</td>
<td>45060</td>
<td>32</td>
<td>Bronchitis</td>
</tr>
<tr>
<td>6</td>
<td>45061</td>
<td>22</td>
<td>Flu</td>
</tr>
</tbody>
</table>

(a) Original Table

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Zip Code</th>
<th>Age</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4505*</td>
<td>2*</td>
<td>Heart Disease</td>
</tr>
<tr>
<td>2</td>
<td>4505*</td>
<td>2*</td>
<td>Heart Disease</td>
</tr>
<tr>
<td>3</td>
<td>4505*</td>
<td>2*</td>
<td>Cancer</td>
</tr>
<tr>
<td>4</td>
<td>450**</td>
<td>[28-32]</td>
<td>Bronchitis</td>
</tr>
<tr>
<td>5</td>
<td>450**</td>
<td>[28-32]</td>
<td>Bronchitis</td>
</tr>
<tr>
<td>6</td>
<td>450**</td>
<td>[28-32]</td>
<td>Flu</td>
</tr>
</tbody>
</table>

(b) 3-anonymous Table

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Zip Code</th>
<th>Age</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4505*</td>
<td>2*</td>
<td>Heart Disease</td>
</tr>
<tr>
<td>2</td>
<td>4505*</td>
<td>2*</td>
<td>Cancer</td>
</tr>
<tr>
<td>3</td>
<td>4505*</td>
<td>2*</td>
<td>Bronchitis</td>
</tr>
<tr>
<td>4</td>
<td>450**</td>
<td>[29-32]</td>
<td>Heart Disease</td>
</tr>
<tr>
<td>5</td>
<td>450**</td>
<td>[29-32]</td>
<td>Bronchitis</td>
</tr>
<tr>
<td>6</td>
<td>450**</td>
<td>[29-32]</td>
<td>Flu</td>
</tr>
</tbody>
</table>

(c) 3-diverse Table

2.2 Privacy in Transaction Data

Another area in data publishing, that deals with high-dimensional data is the privacy in publishing transaction dataset. Transaction data includes an arbitrary set of items from a large universe and corresponds to a specific person. Some examples for this type of data include web search queries, purchase records, and click streams. Most of the work in this area adapt the concepts of relational data publishing to transaction data.

In [11], k-anonymity is extended to provide k"m"-anonymity. It assumes that the knowledge of an attacker is at most m items in a specific transaction. It prevents this attacker from distinguishing
a particular transaction from a set of k published transactions. The (h,k,p)-coherence [13] [14] addresses both the identity (limited by k) and attribute (limited by h) disclosure for an adversary with a knowledge of at most p items.

COAT [7] is an anonymization scheme that also provides k-anonymity. It balances between both privacy and utility by allowing the user to specify privacy and utility constraints as item sets. Using each set of items in the privacy constraint, no person can be linked to less than k transactions. At the same time, the number of transactions containing item sets in the utility constraint remains the same in the published data as the original data.

However, all these schemes have two basic assumptions: i) there is a small subset of private items that needs to be secured, and ii) the attacker’s knowledge is modeled as a finite set of items. Both of these assumptions do not hold for recommendation datasets, and thus these techniques do not apply to recommendation data.

2.3 Privacy in Sparse Data

Another property of the recommendation dataset that is not handled by the above-mentioned solutions is its sparsity. The real-world recommendation dataset is quiet sparse, which makes it difficult to design anonymization schemes. The existing schemes are not effective on sparse data, as the overlap between similar users is relatively small in such data.

An anonymization algorithm for sparse high-dimensional data is provided in [5] based on the bucketization approach, in which the sensitive values are released as a separate table and are combined with the quasi-identifier values using a grouping mechanism. However, this scheme too has the assumption that there is a small subset of the values which are to be considered sensitive.
2.4 Privacy in Recommendation Data

Research in the area of privacy in recommendation datasets started only recently, after the introduction of the re-identification attack on the Netflix dataset in [9]. One solution to this problem was proposed in [3], which provides an algorithm to publish recommendation dataset. The main idea of [3] is based on the anonymization techniques, and involves grouping similar user profiles together and averaging them. To reduce the sparsity in the dataset and to increase the overlap, the dataset is padded with predictive values before the anonymization.

Figure 1 represents the original ratings of users on four different movies and the anonymized graph corresponding to the released dataset. The dotted lines represent the ratings added to the dataset to enforce anonymity. As is evident from the figure, this results in a lot of information loss, as the padding and averaging both deform most of the data in the published dataset.

3 General Overview

3.1 Problem Definition

A recommender dataset consists of a set of consumers, a set of items, and the ratings of different users on various items. Given the original recommender dataset, the problem is to find a subset of this that could be published without compromising the privacy of the consumers, i.e., the published data is secure from the re-identification attack.

3.2 Recommendation Dataset

Before describing our solution, let us first formally define the recommendation dataset. The recommendation dataset consists of:

- Set of users, \( U = \{ U_1, U_2, U_3, \ldots, U_{|U|} \} \),
• Set of items, \( I = \{I_1, I_2, I_3, \ldots, I_m\} \),

• Set of possible ratings, \( R \). For example, in case of the Netflix dataset, \( R = \{0 - \text{Didn’t watch it}, 1 - \text{Hated it}, 2 - \text{Didn’t Like it}, 3 - \text{Liked it}, 4 - \text{Really Liked it}, 5 - \text{Loved it}\} \), and

• An \(|U| \times |I|\) rating matrix, \( M \). The matrix elements are from the set of the possible ratings, \( R \).

An example of a recommendation dataset is shown in Table 2.

We sometimes view the recommendation dataset as a bipartite graph \( G = \{U, I, E, W\} \), where the two subsets of vertices \( U \) and \( I \) correspond to the users and items respectively. There is an edge \((U_i, I_j)\) in \( E \) if the user \( i \) has rated item \( j \), and the weight function \( W: E \rightarrow \mathbb{R} \) assigns each edge a

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
<th>Item 6</th>
<th>Item 7</th>
<th>Item 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User 2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>User 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User 5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: An example recommendation dataset showing the ratings of users to items
rating in R. The bipartite graph corresponding to the recommendation dataset in Table 2 is shown in Figure 2. When there is no edge between a pair of vertices, it implies a 0 rating.

As is evident, this graph is not complete as not all users would have rated all the items, but could be made complete by adding an edge of weight 0.

4 Solution

4.1 Initial processing of the data

In our solution to this problem, both the names of the consumers and the items are removed from the recommendation dataset. Whereas the prior work [3], just removed the consumer names. The names of the items may be irrelevant for developing algorithms based on the dataset, however their presence provides more information to an adversary. The re-identification attack on the Netflix data was possible by cross-listing the movie names in the published data with those in the IMDb, and extracting the corresponding ratings [9]. Such an attack could be averted by our solution.

4.2 High Level Solution

At a high level, our solution finds two different sub-datasets of the original dataset. The set of users in the sub-datasets are disjoint. The rating of the users on the corresponding set of items in one sub-dataset is the same as that by the different set of users on the other set of items in the second sub-dataset. Either of this sub-datasets is what is published by our solution.

If this is done, then no published data corresponds to a particular user, and thus the user’s privacy is protected. This would also provide the data publishers with lawsuit protection, such as the one against Netflix² in which it was alleged that the users could be personally identified even with the proposedly anonymized dataset, as for each user in the published dataset there are at least

two real users corresponding to it.

4.3 Problems with implementing the solution

As mentioned before, the recommendation dataset can be viewed as a bipartite graph. Hence, the problem becomes that of identifying two isomorphic subgraphs of the given original graph. However, this dataset is sparse and it will be easy to find two subgraphs which would have no edges. Publishing a dataset with no edges would not serve the basic purpose of publishing, which is to aid in developing data mining algorithms.

To overcome this problem, we specify one subgraph which satisfies the publisher’s requirements of the released dataset and then find another subgraph that is isomorphic to the given one, if such a subgraph exists. However, this approach also has two problems.
1. The first is that, this is the subgraph isomorphism problem, which involves given a graph G and a graph H finding if there exists another subgraph in G that is isomorphic to H. Unfortunately, this subgraph isomorphism problem is NP-complete [4].

2. The next problem is that, an isomorphic subgraph may not exist.

4.4 Variation to the solution

We propose a variation of the above solution to address these problems. To overcome the first one, the data publisher specifies a subgraph and also another set of items. Thus, the goal is to find another set of users that leads to an isomorphic graph. Stated formally, a subgraph $H_1 = \{U_1, I_1, E_1, W_1\}$ of the graph $G$ is given, along with a set of vertices $I_2$ corresponding to a different set of items. The solution involves finding the set of vertices $U_2$ corresponding to the set of users disjoint from $U_1$, such that the graphs $H_1$ and $H_2 = \{U_2, I_2, E_2, W_2\}$ are isomorphic.

For the second problem, if there is no subgraph $H_2$ that is isomorphic to $H_1$, then the subgraph $H_2$ should be as close to $H_1$ as possible i.e., the goal is to minimize the number of changes that are needed to convert $H_2$ into a graph that is isomorphic to $H_1$.

4.5 Converting to an Assignment Problem

If we take into the edge weights, finding a subgraph closest to the given graph translates into finding a subgraph such that the distance between the two graphs is minimal. Though there are a number of possible distance functions, initially we use the simplest one.

Let’s say $I_1 = \{i_1, i_2, \ldots, i_{|I_1|}\}$ and $I_2 = \{i_1', i_2', \ldots, i_{|I_2|}'\}$, $U_1 = \{u_1, u_2, \ldots, u_{|U_1|}\}$ and $U_2 = \{u_1', u_2', \ldots, u_{|U_2|}'\}$. Then, the distance between two users $u_i \in U_1$ and $u_j' \in U_2$ is given by:

$$D_{i,j'} = \sum_{k=1}^{|I_1|} |M(u_i, i_k) - M(u_j', i_k')|$$

where, $M(i, k) \rightarrow$ Rating given by user $i$ for item $k$ (Section 3.2)
Then, the distance between the two graphs would be $\sum_{i=1}^{\lfloor U_1 \rfloor} D_{i,i'}$.

In case of a difference between the two subgraphs, the rounded average of the weights would be used in the released dataset. One of the aspects of this topic that will be explored is to study the effect of different distance functions.

If the distance between the two sets of users are considered for the different items, then this reduces to the assignment problem. Recall that the assignment problem is as follows: Given a set of users $U = \{U_1, U_2, U_3, \ldots U_m\}$, a set of tasks $T = \{T_1, T_2, T_3, \ldots T_n\}$, and a cost function $C: U \times T \to \mathbb{N}$ where $C(U_i, T_j)$ represents the cost of performing the task $T_j$ by user $U_i$, find a one-to-one assignment mapping $A: U \to T$ such that the total cost given by $\sum_{i=1}^{m} C(U_i, A(U_i))$ is minimal.

Suppose we are given an instance of the recommendation problem i.e., $G, H_1, I_2$ as described above, then we convert it into an assignment problem and solve it to obtain the set of users $U_2$ using the Algorithm 1

**Algorithm 1 SOLVE_RECOMMENDATION(G, H_1, I_2)**

1: $U = U_1$
2: $T = U - U_1$
3: for all $u \in U_1$ and $t \in (U-U_1)$ do
4: $C(u,t) \leftarrow D_{u,t} = \sum_{k=1}^{\lfloor I_1 \rfloor} |M(u, i_k) - M(t, i_k')|$
5: end for
6: $A = SOLVE_ASSIGNMENT(U, T, C)$ //The assignment, $A: U \to T$
7: $U_2 = \{A(u_1), A(u_2), \ldots A(u_{\lfloor U_1 \rfloor})\}$

**Proof of Correctness for Algorithm 1.** To prove that $U_2$ is the optimal solution, by way of contradiction, let us assume that $\exists U_2'$ from the assignment $A'$ which is the optimal solution. Then, the graph $H_2' = \{U_2', I_2, E_2', W_2'\}$ is closer to $H_1$ than $H_2$. So,
\[
\sum_{i=1}^{|U_1|} D_{U_1(i), U_2'(i)} < \sum_{i=1}^{|U_1|} D_{U_1(i), U_2(i)} \text{ i.e.,}
\]
\[
\sum_{i=1}^{|U_1|} C(U_i, A'(U_i)) < \sum_{i=1}^{|U_1|} C(U_i, A(U_i)) \text{ i.e.,}
\]
Cost of $A'$ < Cost of $A$

However, this is not possible as assignment $A$ is the solution for the assignment problem and has the minimal cost. Thus, $U_2$ is the optimal solution for the recommendation problem.

The Hungarian algorithm [6] solves the assignment problem with the time complexity of $O(n^4)$. The algorithm was later improved in [12] to solve the problem with $O(n^{2.22}r^{0.11})$ time complexity, where $n$ represents the number of rows in the matrix and $r$ the range of values in the matrix. However, in our solution the range of values is a constant, $|R|$ and hence the time complexity of the algorithm in our solution is $O(n^{2.22})$. Combining this complexity with that for the reduction of the recommendation problem to an assignment one, the complexity of our solution is $O(n^3)$.

### 4.6 A quick example

As an example, consider the dataset given in Table 2, let $U_1 = \{u_1, u_3\}$, $I_1 = \{i_1, i_5, i_6, i_7\}$ and $I_2 = \{i_4, i_2, i_8, i_5\}$. Then, $U_2$ would be $U - U_1 = \{u_2, u_4, u_5\}$ and the cost matrix $C$ representing the distance between the users in terms of item ratings is given in Table 3. By solving this using the Hungarian algorithm, we obtain the assignments as: $u_1 \rightarrow u_2$, $u_3 \rightarrow u_4$.

Figure 3 shows the two subgraphs of the original graph in Figure 2 with minimum difference between them. The difference between the two graphs is the weight of the edges (1,5) and (2,2). Thus, the rounded value of their average weights would be used in the released dataset, which is shown in Table 4.

<table>
<thead>
<tr>
<th></th>
<th>$u_2$</th>
<th>$u_4$</th>
<th>$u_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>2</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>$u_3$</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3: Example cost matrix
5 Quality Metrics

The primary goal of our solution is to aid in generating recommendation datasets which are published with long time gap between each release. Hence, the time taken for execution is not an important parameter to be considered for measuring the quality of the proposed solution, as long as it is efficient enough to be completed in a reasonable amount of time. Even a solution that would require a couple of weeks on a desktop is an acceptable one.

The important parameter to consider is the quality of the data published. The main purpose for publishing these datasets is to aid in creating better recommendation algorithms. So, the main parameter we have to consider is the quality of these recommendation algorithms developed using our published dataset in comparison with the same developed using the original dataset.

In essence, an algorithm A’s prediction must be similar when developed using either datasets.
In addition, an algorithm A which performs better than algorithm B in the original dataset, must also perform in a similar way in the published dataset. Our quality metrics are also based off of these two ideas.

The evaluation uses over a suite of recommendation algorithms. For all these algorithms we basically calculate two values in line with the quality parameters mentioned above:

1. RMSE (Root Mean Square Error): This reflects the precision of our system, by measuring the difference between the values predicted by our solution and the actual values.

2. Rank: This is the relative position of a particular algorithm with respect to the other algorithms in the suite based on its RMSE value.

We run these algorithms first on the subset of the original dataset which is the primary candidate for publishing, the one which then undergoes changes before publishing to guarantee privacy. Next the algorithm is run on the final dataset, the one obtained by our solution which can be published and is private.

From the above two experiments, we obtain two sets of RMSE values and rankings. For a given set of n algorithm, A = \{A_1, A_2, ... A_n\} lets say these values are RMSE = \{E_1, E_2, ... E_n\} and Ranks = \{R_1, R_2, ... R_n\} for the original dataset, and RMSE’ = \{E_1’, E_2’, ... E_n’\} and Ranks’ = \{R_1’, R_2’, ... R_n’\} for the published dataset.

These values are then compared to establish the quality of the published data relative to the original data. The metric data that we used for this are:

1. Difference RMSE: For a particular algorithm, this value is the difference of its RMSE values pertaining to the original and published datasets.

   Difference RMSE for A_i = |E_i - E_i’|

2. Difference Rank: For a particular algorithm, this value is the difference of its Rankings pertaining to the original and published datasets.

   Difference Rank for A_i = |R_i - R_i’|
3. % change in RMSE: For a particular algorithm, this value corresponds to the percentage of change in RMSE between the original and published datasets w.r.to the original RMSE.

\[
\% \text{ change for } A_i = \left( \frac{| E_i - E_i' |}{E_i} \right) \times 100
\]

4. Max Change in RMSE: For all the algorithm in the suite, this value gives the maximum change in the RMSE value for that run.

\[
\text{Max Change in RMSE} = \max_{i=1}^{n} \text{Difference RMSE for } A_i
\]

5. Max Change in Rank: For all the algorithm in the suite, this value gives the maximum change in the rankings for that run.

\[
\text{Max Change in Rank} = \max_{i=1}^{n} \text{Difference Rank for } A_i
\]

6. Average Change in RMSE: For all the algorithm in the suite, this value gives the average change in the RMSE value for that run.

\[
\text{Average Change in RMSE} = \left( \frac{\sum_{i=1}^{n} \text{Difference RMSE for } A_i}{n} \right)
\]

7. Average Change in Rank: For all the algorithm in the suite, this value gives the average change in the rankings for that run.

\[
\text{Average Change in Rank} = \left( \frac{\sum_{i=1}^{n} \text{Difference Rank for } A_i}{n} \right)
\]

6 System Design

The basic design of the system to implement our solution is presented here. Figure 4 represents the various stages in the processing of the input data to obtain the final table.

1. The input to the system could be any recommendation dataset that has to be anonymized. It is possible that each dataset be available in a different format. So, the initial step would be convert it into a common format that will be used by the other modules of the system. The implementation for this step will be different for different set of data.
The common format we used for storing the dataset is a sparse matrix implemented as an array of hash maps. Each element of the array corresponds to a particular user. The hash map contains the ratings given by that user to specific items. Since the matrix is sparse, for each user only the available ratings (in case of Netflix, 0 ratings are ignored) are stored with the key values corresponding to the item number.

For the Movielens dataset, we used the TIntByteHashMap. As there are only 10 possible ratings: \{0.5, 1, 1.5, \ldots, 4.5, 5\}, a byte is sufficient to hold them rather than a float.

For the example dataset in (refer to proposal) the corresponding dataset in the common format is shown in Figure 5.

2. The common format data contains the entire original dataset in the specified format. From this we have to extract the data that is required for obtaining the solution.

The three arrays: \(U_1, I_1, I_2\) would be the input to the system. From this we could compute the set of all possible users for \(U_2\) i.e., \(U_2' = U - U_1\).

\[^3\text{http://trove4j.sourceforge.net/javadocs/gnu/trove/TIntByteHashMap.html}\]
With these four 1-D arrays, we read the common format dataset and just store the required information in two matrices: \( M_1 = U_1 \times I_1, M_2 = U_2' \times I_2 \). It is not necessary to store the other information, as they won’t be required for obtaining the final dataset, and storing them would occupy more memory than required.

These matrices are also stored in the same format as in Figure 5.

3. Once we have the two matrices, we obtain the cost matrix, \( C \) for the assignment problem as discussed earlier in Section 4.5.

For a cost matrix to be used as an input to the assignment problem, it must be an \( n \times n \) matrix. However, in our case it is highly likely that \( |U_2'| > |U_1| \). In such a case, the matrix is padded with zero values to make both the dimensions to be of equal size.

4. Then the assignment problem is solved using an existing implementation of the Hungarian algorithm [2] to obtain the set of assignments \( A : U_1 \rightarrow U_2' \), as a 2D array. From this assignment, the other set of users, \( U_2 \) is computed (ignoring the assignments for zero values).

5. With the two sets of users computed, the final dataset to be published is computed. The ratings of every user in \( U_2 \) is compared with the corresponding one in \( U_1 \), and the final table is populated with their average. //Have to change it to reflect non-zero average, if required

The matrix corresponding to the final table is also stored as a 2-D array of size \( |U_1| \times |I_1| \), with zero values for ratings that are not available. It is also written to a data file that could
be used as the input for testing algorithms.

It is obvious that the same design can be easily adapted to figure out a set of items given two sets of users.

7 Experiments & Results

The experiments performed to measure the quality of our solution, the results obtained, and the corresponding improvements/changes made to our solution are detailed in this section.

7.1 Ranking of algorithms

The major metrics for the evaluation of our system involve obtaining the RMSE values and ranking of various recommendation algorithms as mentioned in Section 5. Here, we explain the various datasets and algorithms that were used for testing our system, and how the algorithms were ranked.

7.1.1 Datasets Used

Since, the Netflix dataset is not available anymore⁴, we have used the Movielens dataset⁵, which is also a movie rating dataset and is available publicly. The Movielens has three datasets:

- 100,000 ratings for 1682 movies by 943 users
- 1 million ratings for 3900 movies by 6040 users
- 10 million ratings for 10681 movies by 71567 users

We have used the first two datasets as such in our experiments. However, we were not able to use the entire 10 million ratings due to memory restrictions and have used around 4 million ratings. Apart from these real datasets, we have used some smaller synthetic datasets for testing.

⁴http://archive.ics.uci.edu/ml/datasets/Netflix+Prize
⁵http://www.grouplens.org/
7.1.2 Tools Used

Two tools were used that provided us with the set of algorithms to test with:

1. Weka:6

   It is a project at the University of Waikato to build state-of-the-art software for machine learning techniques. It consists of a collection of different machine learning for data mining tasks. It comes with a GUI for using the various tools on the data. However, we imported and used it in our own Java code.

   This requires the input file to be in the Attribute-Relation file format, .arff7. So, we also wrote code to convert the input file available in the .data format to the .arff file. It provides its own methods for evaluating an algorithm.

2. Cofi:8

   It is a Java based collaborative filtering library. It just provides the implementation of the various collaborative filtering algorithms. We had to write a wrapper around this code, that uses these algorithms, providing appropriate input and comparing the output produced.

   The library has an in-built parser for the Movielens dataset. For other datasets, we had to write our own parser.

7.1.3 Computing RMSE

The different algorithms in the suite of algorithm are ranked based on their RMSE values in a non-decreasing order. The formula for computing the RMSE for a given algorithm and a set of ratings is as follows:

6http://www.cs.waikato.ac.nz/ml/weka/
7http://www.cs.waikato.ac.nz/ml/weka/arff.html
8http://www.nongnu.org/cofi/
RMSE: Root Mean Square Error = $\sqrt{\left( \sum_{i=1}^{n} (x_{1,i} - x_{2,i})^2 \right) / n}$

where, $x_{1,i}$ → original value for rating $x_i$

$x_{2,i}$ → predicted value for rating $x_i$

$n$ → total number of available ratings

To obtain the RMSE value for a particular algorithm on a given dataset, we split the dataset into training and test data. The algorithm is initially trained using the training data, and then its RMSE value is computed by testing it with the test data as follows: for each of the available rating i.e., for every non-zero value in the test data, we remove that rating alone from the entire test data and then predict what that rating would be based on the other available ratings using the given algorithm. Comparing this value predicted by the algorithm with the actual value, helps us determine the quality of the particular algorithm.

The complete algorithm for computing the RMSE value for a particular trained algorithm using a given test data is provided in algorithm 2.

### 7.2 Experiment Setup

The experiment setup to implement the algorithm(s) discussed before in Section 4 is presented here. Figure 6 shows the different components of our system, their inputs and outputs, and how they interact with each other.

Given the original dataset, and the percentage of it that must be published, the system produces the final dataset containing the given percentage of users and items using the original, with comparable quality and privacy, and that could be published.

The input to the system is the original dataset, the total number of users and items in it, and the percentage of the original users and items to be available in the published dataset. The output is the final dataset that could be published, and the various quality metrics data.
Algorithm 2 COMPUTE_RMSE(Algorithm A, Test Data T)

1: RMSE = 0
2: Count = 0
3: for all User U ∈ T do
4:     for all non-zero rating, r of U do
5:         u’ ← Set of ratings of U - r
6:         r’ ← A’s prediction of r using u’ as the user’s available set of ratings
7:         if r’ < min of R then
8:             r’ ← min of R
9:         else if r’ > max of R then
10:            r’ ← max of R
11:       end if
12:       RMSE += (r − r’)²
13:       Count++
14:     end for
15: end for
16: RMSE = √(RMSE/Count)
1. Input: Total number of users and items, and their percentages in final dataset

   The number of users and items to be present in the final dataset is computed. Then, two sets of users and one set of item is picked at random.

   Output: Three lists of Ids: U₁, U₂, and I₁

2. Input: Three list of Ids, Original Dataset

   The test file and a training file are computed. The training file consists of the ratings of all users not in the first set of users for the computed list of items. The training file consists of the ratings for the computed list of items by the first set of users.

   Test = (U - U₁) x I₁

   Train 1 = U₁ x I₁
3. Input: Three list of Ids, Original Dataset

The algorithm as described in Section 4.6 is executed, and we obtain another set of items.

Output: List of Ids: I_2

4. Input: Four list of Ids, Original Dataset

The final dataset to be published is computed as the average of the two sets of ratings U_1 x I_1, and U_2 x I_2

Output: Final dataset to publish

5. Input: Final dataset

The published dataset is used as another training file (Train 2). The final dataset must be written in the format of the original file.

Output: Training file (Train 2)

6. Input: Test, Train 1, Train 2

The tool containing the suite of recommendation algorithms mentioned in Section 7.1.2 is first trained using the training file, Train 1. Then, the algorithms are tested using the test file, Test and we obtain the RMSE values for each algorithm using the algorithm 2.

The algorithms are then trained using the second training file, Train 2 and tested, yielding another set of RMSE values for the same set of algorithms.

Output: Two sets of RMSE values pertaining to the same set of algorithms

7. Input: Two set of RMSE values for a set of algorithms

The algorithms are assigned ranks based on their RMSE values in a non-decreasing order. Then, the other quality metrics data such as the difference in values, the maximum and average changes in them are computed using formulas listed in Section 5
7.3 Results & Analysis

The initial experiments correspond to the solution described in Section 4. The next set of experiments pertain to the analysis of the results of the initial experiment, and creating a better solution based on non-zero averaging. The analysis of the experiments with non-zero averaging leads us to the next set of experiments to figure out the overlap between the two sets of users, and creating solutions with random sets of users.

With all the above mentioned experiments designed to use the 100K Movielens dataset, and a particular percentage of users (15%) and items (30%) for evaluation, the next set of experiments are to reflect the effect of scaling both in terms of different percentages of users and items, and also different sizes of datasets.

Before drawing conclusion based on all these experiments, we must also establish the integrity of our evaluation technique. For this, we use experiments with expected bad results.

All these experiments, their results, and the analysis of these results are described in detail in the following sections.

7.3.1 Proposed Solution

7.3.1.1 Initial Results  We started with this experiment pertaining to the setup detailed in Section 7.2. We ran this experiment using the 100K Movielens dataset, the details of which was presented in Section 7.1.1 The goal of this experiment is to publish a subset of the original dataset containing 15% of the original users i.e., 141 users, and 30% of the original items i.e., 504 items.

The results pertaining to this experiment, sorted by their original rankings are shown in Figures 7 and 8. As seen in these graphs, though the RMSE values followed a similar pattern for most of the algorithms, the rankings varied a lot with the maximum change being 13.
Figure 7: Proposed Experiment: RMSE values for the original and published datasets of the 100K Movielens dataset, and the corresponding difference in their RMSE values.
Figure 8: Proposed Experiment: Rankings for the original and published datasets of the 100K Movielens dataset, and the corresponding difference in their rankings
7.3.1.2 Non-zero Average When we analyzed the reason behind this vast change in the rankings in our proposed solution, we figured out that the main reason for it was that we were just averaging the two sets of ratings irrespective of their original values. Since, the original matrix is a sparse matrix, a lot of averaged values contained zero values.

![Graph 1](image1)

![Graph 2](image2)

Figure 9: Experiment with Non-Zero Average: RMSE values for the original and published datasets of the 100K Movielens dataset, and the corresponding difference in the RMSE values

When an available rating is averaged with zero, its value just gets halved resulting in a consid-
erable distortion of data, which ultimately led to a much different set of rankings for the same set of algorithms. However, averaging a value with zero does not make sense in this context, as zero doesn’t represent a rating given by the user like the other numbers in the matrix from 1 to 5. It is just used as a filler to represent the non-existence of the rating.

For instance, if we were averaging 0 and 2 and coming up with a rating of 1 for our final dataset, then in this context 0 would mean that the user really hates that item. However, this is not the semantic of zero, and it was only be used to represent that the user has no opinion about that item. If we incorporate this semantic into our solution, then the distance between 0 and any other rating is essentially the same.

So, what we were doing in the initial experiment was averaging an existing rating with a non-existing rating without taking into account their semantics. This essentially means a problem in our averaging technique.

To overcome this, we computed the non-zero average of the two sets of ratings for the published dataset i.e., the average is computed only when both the ratings exist, else the only available non-

Figure 10: Experiment with Non-Zero Average: % change in RMSE values for the original and published datasets of the 100K MovieLens dataset
Figure 11: Experiment with Non-Zero Average: Rankings for the original and published datasets of the 100K Movielens dataset, and the corresponding difference in the rankings

zero rating is used as such. Thus, even in our final dataset zero would only mean the non-existence of a rating. With this averaging technique included in our original solution, we were able to obtain a similar pattern for both the RMSE values and the rankings for the given set of algorithms, between the original and published datasets.

The same experiment as described for the proposed solution above using the 100K dataset, and 15% of users and 30% of items, was repeated with the non-zero averaging technique replacing our
original averaging. The results obtained in terms of the RMSE values, the % change, and rankings sorted by their original rankings are shown in Figures 9, 10, and 11 respectively. As we could see in these graphs, both the RMSE values and the rankings follow a similar pattern. The maximum change in rank in this case is only 2 as opposed to 13 in our original, the maximum percentage change is 7.79%, and the maximum change in RMSE is only 0.093. Thus, we were able to produce a final dataset as a subset of the original dataset with comparable quality.

7.3.2 Random Users

7.3.2.1 Flips & Delta  As mentioned above, when we analyzed the results, we figured out that most of the averaging involved a zero value. This would imply that there isn’t a significant overlap between the corresponding ratings of the two sets of users. To make it more concrete we computed the following two values for each user in the final dataset:

1. Delta: The difference between the original and published values when the final value was obtained by averaging two existing ratings, and

2. Flip: The number of ratings in the final dataset which involve the averaging of an existing rating with a zero rating.

The detailed algorithm for computing these values is presented in algorithm 3.

Figures 12 and 13 show the delta and flip values for the final dataset for the experiment discussed above, computing final dataset with 15% of users, and 30% of items of the 100K dataset respectively.

As is evident from these graphs, the number of flips is much higher than the delta values. This in turn implies that though we were computing the closest users, the overall similarity between the two datasets that we were averaging to obtain the final dataset is rather low. The main reason behind it being that the two set of items are chosen at random, and there is no guarantee regarding their similarity.
Algorithm 3 \textsc{compute} \_\textsc{change}(\text{Set of User Ratings } U_1, \text{ Set of User Ratings } U_2)

1: \text{U} = \text{Final set of ratings}

2: \text{delta} = 0

3: \text{flips} = 0

4: \text{for all Items I } \in \text{ set of items in } U_1 \text{ do}

5: \quad \text{if } U_1[I] = 0 \text{ then}

6: \quad \qquad U[I] \leftarrow U_2[I]

7: \quad \quad \text{if } U_2[I] \neq 0 \text{ then}

8: \quad \quad \quad \text{flips}++

9: \quad \quad \text{end if}

10: \quad \text{else if } U_2[I] = 0 \text{ then}

11: \quad \qquad U[I] \leftarrow U_1[I]

12: \quad \quad \text{flips}++

13: \quad \text{else}

14: \quad \qquad U[I] \leftarrow (U_1[I] + U_2[I])/2

15: \quad \text{delta} += |U[I] - U_1[I]| + |U[I] - U_2[I]|

16: \quad \text{end if}

17: \text{end for}
7.3.2.2 Similar Results  Consequently, we ran experiments with all the four sets: \(U_1, U_2, I_1,\) and \(I_2\) chosen at random, as we had figured out that the overlap between them is not significant even if computed. Not so surprisingly, the result of this experiment was not much worser than the initial results, as there wasn’t any significant overlap between the two sets of users even when they were computed using the assignment algorithm.
Figures 14 and 15 show the results pertaining to the same experiment as in non-zero averaging with the 100K dataset and 15% of users, 30% of items, but this time using random selection of the two sets of users and items. As is evident from these graphs, this result though not at as good as the previous result, is comparable. The maximum change in ranking in this case increased to 4, and the maximum change in RMSE increased to 0.129 from 0.093.
The performance of this published dataset is also comparable to that of the original dataset, because the amount of distortion is really low (delta values). In either cases, we only add additional information to the existing dataset. However, the amount of additional information added is also low in most of the cases (flip values), as the dataset is itself really sparse. Thus, the performance of the algorithms and the corresponding RMSEs do not change significantly between the original and published datasets.

7.3.2.3 Increasing number of user sets  From the previous experiments, we can conclude that adding or distorting little information did not affect the quality of the final dataset significantly. To throw further light on this argument, we conducted experiments to learn the effect of adding more information to compute the final dataset.

We generated the final dataset by not just computing the non-zero average of two different datasets, but more than that. We conducted this experiment by computing the final dataset with 3, 4, and 5 different datasets.

![Averaging more datasets](image)

Figure 16: Experiment Averaging more Datasets: The average change between the original and the published datasets in terms of the rankings and RMSE values when averaging different number of datasets to obtain the final dataset
The result of this experiment in Figure 16, shows how the average difference in the rankings, and the RMSE values change with the addition of more information. We can see that the difference between the original and published datasets is greater with the addition of more information.

So, adding a few additional information doesn’t have much impact on the published dataset. However, if we keep adding more and more information to the existing dataset, the quality of the published dataset starts deteriorating.

7.3.3 Scaling of the results

All the results we discussed above pertained to the 100K Movielens dataset as the original dataset, and the final dataset was created to have 15% of the users and 30% of the items. However, we wanted to make sure that our solution scales to different percentages of the users and items in the final dataset, and also to different sizes of datasets

7.3.3.1 For different percentages of users and items
To analyze the effect of varying the percentages of the users and items of the original dataset in the final dataset, we conducted the experiments with non-zero averaging and with random users, but by varying these percentages alone.

Figure 17 shows how the results vary for the 100K dataset when these percentages were varied from 10 through 45. It shows the change in terms of the average RMSE values and the rankings respectively. It is evident from these graphs that there is no significant change either of the values when we alter the percentages of users and items. The maximum difference in RMSE values between all these different percentages is just 0.013, and that of the ranks is 0.57 (less than 1).

7.3.3.2 For larger datasets
To understand the performance of our solution on larger datasets, we conducted the experiments on different datasets apart from the 100K dataset like the 1M Movielens dataset, and the 10M Movielens dataset. We have used a random subset of the 10M dataset
Figure 17: Experiment varying the percentages of users and items: Shows the average change in the RMSE values and the rankings between the original and published datasets with approximately 3.5 million ratings. All these experiments aimed at publishing a subset of the original with 15% of the users and 30% of the items.

Figure 18: Experiment varying the dataset size: Shows the maximum and average change in the RMSE values and the rankings between the original and published datasets.

Figure 18 shows the results of this experiment. As we can see, the performance gets better
with the increase in the size of the dataset. This is an expected behaviour as the effect of distortion decreases with the increase in the available data. In case of the 10M dataset there is absolutely no change in ranking, and there is only a very little change in the RMSE values with the maximum change being 0.009. So, using our solution, for datasets of size comparable to that released by Netflix, the quality of the dataset published by our solution will be very similar to that of the original.

In case of the Netflix contest, the difference in RMSE between the different algorithms were in terms of 0.001. Since, we are able to obtain a performance with average change in RMSE being 0.004 by publishing just 80238 ratings with 3.5M ratings in the original dataset, it might be possible to obtain a performance with RMSE changes less than 0.001 with 100M ratings in the published dataset.

### 7.3.4 Verifying the Evaluation Method

Before making conclusions with our experiments, we wanted to verify the integrity our evaluation model, since we have been getting good results even with random selections of users and items. So, we ran experiments with expected bad results, like the ones with totally unrelated original and published datasets.

The first experiment compared the original dataset with a different dataset, while the second experiment compared the different dataset with the published dataset. For this experiment we used the dataset corresponding to the second set of users and items that was used in generating the final dataset, as the different mismatching dataset. These experiments also used the 100K dataset with 15% of users and 30% of items.

Figures 19 and 20 show the results of these experiments in terms of the change in RMSE values and the rankings respectively, comparing them with the results we had originally obtained using non-zero averaging.

As expected and evident from these graphs, the results pertaining to the mismatched datasets
Figure 19: Experiment with mismatching datasets: Shows the changes in the RMSE values between the mismatched datasets in terms of the maximum and average change

Figure 20: Experiment with mismatching datasets: Shows the changes in the rankings between the mismatched datasets in terms of the maximum and average change

are much worse compared to the previous ones we had obtained with similar datasets. The maximum change in ranking between the two datasets obtained using our solution was just 2, compared to 19 in this case, and the maximum change in RMSE is 0.7, when it was just 0.093 for our solution.

This verifies the results we had obtained using this evaluation model. Thus, when the results
pertaining to two datasets is similar in our evaluation, then it does imply that the quality of those recommendation datasets are similar too.

8 Privacy & Utility of the Published Dataset

8.1 Privacy

In our solution, two similar datasets are produced. However, the privacy provided is greater than that provided by the 2-anonymization in [3]. The dataset released by [3] consists of at least two users in one group with the same set of ratings, which implies that the users have similar if not the same ratings for the same set of items, which are explicitly specified.

However, no such inference can made from the dataset published using our solution. No specific detail about the items are published. Each published rating corresponds to two different users. Moreover, the items over which they are merged are different. The published dataset is the union of two totally different datasets, in terms of both the set of users and set of items.

To have a better understanding of the privacy of the dataset generated by our solution, we conducted experiments to determine the number of users in the original dataset that could potentially map with a given user in the published dataset. In this experiment, a user in the original dataset could map to a user in the published dataset, if the ratings corresponding to the published user is a subset of the ratings of the original user, given the amount of distortion the user’s ratings has undergone (in terms of flips and delta).

Using this experiment, for the 100K dataset, it was possible for a user in the published dataset to map to any of the original users. Even for the 1M dataset, the number of mapping users was large, with most of them mapping to well over 50% of the original users. Figure 21 shows these results for the 1M dataset. The lowest mapping was to 2.8% (170 of 6040) of the original users, with the highest of them matching to all the users. Hence, for a dataset generated using our solution, it is
not possible to map a published user to the original user with a higher probability.

Moreover, in this experiment we have used the amount of distortion for each user to obtain the mapping. However, even this information will not be publicly available. So, it will not be possible to obtain any reasonable mapping between the two sets of users. Thus, the dataset generated by our solution is private, and is not susceptible to the re-identification attacks.

8.2 Utility

The proposed solution generates a recommendation dataset with less information distortion than [3]. In [3], the original dataset is padded to reduce its sparsity before anonymizing. In our solution, there is no such padding involved. The sparse characteristic of the original dataset is also carried over to the dataset generated for publishing. The only information loss incurred in our solution is the averaging of the mismatched ratings, which also exists in [3].

The experiments whose results were discussed in Section 7.3, were mainly to ascertain the utility of the dataset being published. As these results suggested, the utility of the published dataset
is comparable to that of the original.

9 Future Work

Some of the works which could be undertaken and could potentially improve our solution are:

- The system uses a simple distance function. However, other distances like the Euclidean distance could be used, and the change in the solution’s quality could be noted.

- In the solution described above, a constant distance function is used to compute the cost matrix, which might not be appropriate in all the cases. For example, in case of the Netflix dataset the ratings 3-Liked it and 4-Really liked it, are closer than the ratings 2-Didn’t like it and 3-Liked it.

To incorporate this difference into the system a $|R_1| \times |R_2|$ difference matrix, $D$ could be supplied as an optional input parameter to the system, where $D_{i,j}$ would represent the difference between the ratings $R_i$ and $R_j$ which would be used in computing the cost matrix.

Apart from computing the cost matrix, the same could also be used in computing the information loss of the final dataset to determine the quality of the system.

- It is possible that the items in the original dataset are categorized. For example, in case of the Netflix dataset there are many categories like Action & Adventure, Classics, Comedy, Drama, Foreign, and Thrillers$^9$.

The system could be extended to include minimum number items/users from each of the different available categories. In this case it must be noted that while selecting two set of items, the corresponding items in the two sets must be from the same category.

---

$^9$http://www.netflix.com/AllGenresList
For the experiment in our privacy discussion, we had obtained the number of original users that a given published user could map to. However, this could be extended to obtaining mapping for a pair of users scaling it upto n users, for a better understanding of the privacy.

10 Conclusion

With the recommendation systems becoming more prevalent, the need for publishing the recommendation datasets is great. In this research, we have provided a solution that would help publish these recommendation datasets without much loss of quality and with ushered privacy. However, as mentioned earlier this is just the first step and there is scope for improvement.
References


