Copyright
by
James Adrian Edwards
2009
The Report Committee for James Adrian Edwards
Certifies that this is the approved version of the following report

Quantifying the Multi-user Account Problem for Recommender Systems

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Joydeep Ghosh

Inderjit Dhillon
Quantifying the Multi-user Account Problem for Collaborative Filtering Based Recommender Systems

by

James Adrian Edwards, MA;BS

Report
Presented to the Faculty of the Graduate School of The University of Texas at Austin in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering

The University of Texas at Austin December 2009
Acknowledgements

I would like to acknowledge Salim Amirdache and Thomas Hughes for their contributions to a previous paper entitled “Modeling Multi-user accounts in Netflix rating data”. The ideas discussed therein formed the basis for the current report.

December 2009
Abstract

Quantifying the Multi-user Account Problem for Collaborative Filtering Based Recommender Systems

James Adrian Edwards, MSE
The University of Texas at Austin, 2009

Supervisor: Joydeep Ghosh

Identification based recommender systems make no distinction between users and accounts; all the data collected during account sessions are attributed to a single user. In reality this is not necessarily true for all accounts; several different users who have distinct, and possibly very different, preferences may access the same account. Such accounts are identified as multi-user accounts. Strangely, no serious study considering the existence of multi-user accounts in recommender systems has been undertaken. This report quantifies the affect multi-user accounts have on the predictive capabilities of recommender system, focusing on two popular collaborative filtering algorithms, the kNN user-based and item-based models. The results indicate that while the item-based model is largely resistant to multi-user account corruption the quality of predictions generated by the user-based model is significantly degraded.
# Table of Contents

Chapter 1: Introduction .................................................................................................................1

Chapter 2: Recommender Systems and Multi-user accounts ......................................................5

Chapter 3: Recommender Systems, History & Models .................................................................9
    User-user Similarity Weighting Measure ..............................................................................10
    User-user Significance weighting .....................................................................................11
    User-user Neighborhood selection ....................................................................................11
    User-user Prediction calculation .........................................................................................12
    Item-item Similarity Weighting Measure ............................................................................14
    Item-item Neighborhood selection .....................................................................................16
    Item-item Prediction calculation .........................................................................................16

Chapter 4: Tools, Data and Metrics Used in Model Evaluation ...............................................18
    kNN recommender systems ...............................................................................................18
    Multi-User training datasets ..............................................................................................20
    Evaluation Metrics .............................................................................................................24

Chapter 5: Evaluation of the Affects Multi-user Accounts have on kNN Recommender Systems .................................................................30
    Evaluation of the RMSE Metric .........................................................................................30
    Evaluation of the MRP_N Metric .......................................................................................35

Chapter 6: Conclusion .................................................................................................................43

Appendix A: The Netflix Prize Dataset ......................................................................................45

Appendix B: NFDB Database Format .........................................................................................47

Glossary ........................................................................................................................................49

References ....................................................................................................................................51

VITA
List of Tables

Table 1: Accuracy (RMSE) for the Netflix Probe Data ...................................................... 20
Table 2: MRP for N=10 ...................................................................................................... 36
List of Figures

Figure 1: Distribution of #Ratings/Movie of the Netflix Prize dataset; Movies are Ranked by Rating Count .................................................................22

Figure 2: Distribution of the Number of Ratings per Account for the Truncated Netflix Prize dataset ...........................................................................23

Figure 3: RMSE vs. %corruption for user-user and item-item models ...........31

Figure 4: Change in RMSE delta vs. %corruption .........................................................32

Figure 5: ROC Curve Using SE as Account Type Classifier ....................34

Figure 6: Item-item reconstruction precision distribution for ‘pure’ accounts .....37

Figure 7: User-user reconstruction precision distribution for ‘pure’ accounts .....37

Figure 8: Item-item reconstruction precision distribution for ‘corrupt’ accounts .39

Figure 9: Figure 10 Item-item reconstruction precision distribution for ‘corrupt’ accounts ................................................................................40

Figure 10: Reconstruction precision distribution vs. corruption proportion..........41
List of Equations

Equation 1: Pearson Correlation .............................................................. 11
Equation 2: Rating prediction .................................................................. 12
Equation 3: Cosine similarity .................................................................. 15
Equation 4: Adjusted cosine similarity ..................................................... 15
Equation 5: Item-item model prediction calculation .................................. 16
Equation 6: RMSE for a set of N rating predictions ..................................... 25
Equation 7: Precision and Recall .............................................................. 27
Chapter 1: Introduction

Recommender systems are used by online commercial sites to help site users find products that fit their preferences. These systems use user account profiles that contain preference indicators for specific products or types of products to assess the interest a customer may have in a product offering. Recommender systems generally perform user profiling in two different ways (Tso & Schmidt-Thieme, 2006):

1) Anonymous profiling.
2) Profiling with user identification.

Recommender systems that use anonymous profiling do not require user identification. They usually do require the user to fill out a short questionnaire customized to the product category, however. These systems ask about a user's needs up-front, and generally rely on product attributes to recommend similar products to the anonymous user. Anonymous recommender systems are usually focused on product attributes, and do not consider long-term user preferences to make recommendations. Such systems do not utilize accounts, so the multi-user account problem (described later) is not defined. Such systems are not considered in this report.

The recommender systems that do utilize user identification often require a user to login each time the system is accessed. These systems can accumulate user preference indicators gathered over time during distinct account sessions. Whenever a customer logs into the system, his or her past account profile is looked up and used for recommending new products. New users often have to provide some up-front information before they can use the system. This usually occurs in the initial account session, and the data
obtained during that session is used to provide base-line data for classifying the new user. Subsequent account sessions provide more data, allowing the recommender system to further refine its classification of the user, and thus its ability to provide more accurate recommendations.

Identification based recommender systems make no distinction between the users and accounts; all the data collected during account sessions are attributed to a single user. In reality this is not necessarily true for all accounts; several different users who have distinct, and possibly very different, preferences may access the same account. Beyond requiring session authentication, there is currently no way a recommender system can ensure that data associated with an account comes from a single user. If the account authentication mechanism is compromised, intentionally or otherwise, the recommender system will aggregate data from several users and treat the aggregate as the profile of a distinct individual.

Multi-user accounts could obviously affect a recommender system's ability to make accurate predictions about an individual that uses the account. The preferences of the other account users will be added to the preferences of the current user. This interference will defeat the primary purpose of the recommender system; to predict what the current user is interested in and provide access to information and products that match those interests. In addition, multi-user accounts may also affect recommender systems beyond the local effects inflicted on compromised accounts. This may depend on the method the system uses to make recommendations and the number and makeup of the multi-user accounts.

As an example, consider the impact multi-user accounts could have on recommender systems based on collaborative filtering techniques. Collaborative filtering based recommender systems try to predict the utility of items for a particular
user based on the items previously rated by other users, or similar items previously rated by the user. They rely solely on past user behavior. A common form of collaborative filtering is the neighborhood-based approach, or k-Nearest-Neighbors\(^1\) (Bell & Koren, Lessons from the Netflix Prize Challenge, 2007). The user-user \(k\)-NN methods identify pairs of items that tend to be rated by similar or like-minded users with similar histories of rating or purchasing, in order to predict ratings for unobserved user-item pairs. In order to estimate the unknown rating for a user-item pair, a set of neighbors is selected that tend to rate similarly to the user, and have also rated the item in question. The predicted rating value for that user of the specified item is calculated as the weighted average of the nearest neighbor ratings. The closely related item-item \(k\)NN methods use item similarities to make user-item predictions.

Neighborhood-based methods are popular because they are intuitive and simple to implement. They do not require tuning many parameters or an extensive training stage, and also provide an intuitive justification for the computed predictions. However, these models become less effective if the individual identity of the user making the ratings is compromised. Consider, for example, an account that contains a record of the ratings, but that has been shared by two individuals with greatly different tastes. This account, identified as a single user, could create false correlations between the two neighborhoods that the actual individuals would normally belong to individually. Large numbers of these multi-user accounts could cause many of these false correlations, ultimately affecting the accuracy of the prediction model.

Researchers active in the recommender field are aware of the impacts data scarcity and of Shilling (Tso & Schmidt-Thieme, 2006) attacks have on recommender

---

\(^1\) kNN for short.
systems, and mitigating the effects of these problems are subjects of great interest and considerable study. Strangely, the recommender system community has largely ignored the Multi-user Account Problem. To the knowledge of the author, no serious study considering the existence of multi-user accounts in recommender systems has been undertaken.

This report attempts to quantify the Multi-user Account Problem by evaluating the impact such accounts have on two well known collaborative filtering models, the user-user and item-item k-Nearest-Neighbor models. This is done by generating synthetic datasets containing know proportions of multi-user contamination and observing the changes in the predictive accuracy of the different models as the multi-user contamination percentage is increased. Two accuracy metrics are used to evaluate the predictive capabilities of the models, the widely used Root Mean Squared Error and a classification accuracy metric based on precision and recall called the Mean Reconstruction Precision.
Chapter 2: Recommender Systems and Multi-user accounts

Non-anonymous recommendation systems are generally divided into two categories. One is called content-based filtering; the other is collaborative filtering. Content-based systems perform recommendations by matching an item’s described attributes with the preferred attributes associated with a user profile. The user profile is often created and updated automatically in response to feedback on the desirability of items that have been presented to the user. In a content-based filtering system, each user is assumed to operate independently and item attributes are pre-defined.

In collaborative filtering systems, the representation, or attributes, of an item is based on the evaluation of that item by other users in the system. One of the main characteristics in collaborative filtering, compared to content-based filtering, is that the method knows nothing about the items’ true content or what their attributes may be. Collaborative filtering systems use data from other users to “train” a prediction model; predictions are specific to a user but are based on the collaborative input of all previous users. This means that predictions only rely on the preference values of previous users, such as ratings, to generate the recommendations.

Collaborative filtering systems provide several advantages over content-based systems (Herlocker J. L., Konstan, Borchers, & Riedel, 1999): (1) they are easily automated to support filtering of items whose content or attributes are not easy to analyze; (2) they have the ability to filter items based on quality and taste; and (3) they have the ability to provide users with serendipitous recommendations. A serendipitous recommendation is a recommendation for an item that a user would not normally have
considered, but which is based on the recommender systems predicted attractiveness or utility of the item to the user.

Collaborative filtering systems do suffer from several disadvantages (Adomavicius & Tuzhilin, 2005), however. The **New User Problem**, also associated with content-based systems, occurs because recommender systems must first learn a user’s preferences from the ratings that the user gives. A new user will have little associated data, so predictions for a new user will be inaccurate until a sufficient amount of profile information can be amassed. A similar problem is the **New Item Problem**, which occurs when a new item is added to a collaborative filtering based recommender system. Until the new item is rated by a substantial number of users the recommender system will not be able to accurately predict a user’s affinity toward it, and thus will not be able to recommend it.

In any recommender system, the number of ratings already obtained is usually very small compared to the number of ratings that need to be predicted. The success of the collaborative recommender system depends on the availability of a large number of users and associated item ratings. There are several ways to overcome the problem of rating sparsity, including using demographic user profile information to perform “demographic filtering” and dimensionality reduction to impute missing values and/or reduce the dimensionality of sparse rating matrices.

Recommender systems based on collaborative filtering also face more insidious challenges than those posed by the technical issues of new entities and data sparsity. Because such systems provide recommendations based on user collaboration it is possible for unscrupulous item producers to **shill** a recommender system in an attempt to influence the popularity of their product. These users inject false opinions into the system to either “push” one or more items in the system in order to have them recommended to more
users or to “nuke” a set of items to cause them to be recommended to fewer users. Shilling is the general term used to describe such practices and shilling attacks pose a serious threat to users and operators of collaborative filtering based recommender systems and can greatly affect the accuracy of a system’s predictions (Lam & Riedl, 2004).

The disadvantages and vulnerabilities of collaborative filtering recommender systems described in the previous paragraphs are well known and widely studied by the recommender system research community. Another vulnerability that is not well studied is the existence of multi-user accounts and the impact such accounts have on recommendations. Multi-user accounts are individual accounts that are used by multiple users to rate various items. Because automated collaborative filtering recommendation systems primarily use the similarities between users or items to make predictions and generally identify a user with an account id, the existence of accounts used by two or more users with greatly different preferences could cause correlations between accounts and items that would normally not exist. These “false” correlations have an obvious impact on the recommender systems ability to make predictions for the current user of the compromised account (local effect), but they could also impact the systems ability to accurately make predictions for single user accounts that are not directly compromised (global effect). Users of non-compromised accounts could start to receive recommendations based on their similarities with one user of a multi-user account, but for items associated with a different user of the contaminated account. These “serendipitous” recommendations will usually be badly received, and may reduce a users confidence in the recommender system.

While less insidious than a shilling attack, the existence of multi-user accounts have the potential to greatly affect the accuracy of a recommender system’s predictions
and thus its ability to make relevant recommendations. The casual use of someone else’s account to browse or purchase items seems like it would be a more common user behavior than using an account to generate a shilling attack, so high levels of multi-user account corruption should not be discounted. The impact such accounts have on a recommender system obviously depends on the percentage of accounts that are compromised but can also depend on the prediction algorithm.

In this report the Multi-user Account Problem denotes the affect multi-user accounts have on the predictive capabilities of a recommender system. Because content-based recommender systems consider each user in isolation, the affect of multi-user accounts on the prediction accuracy of the system will be local, i.e. only predictions for the set of multi-user accounts will be affected. This report focuses primarily on recommender systems that use collaborative filtering to make predictions, and the affects that multi-user accounts have on the accuracy of their predictions.
Chapter 3: Recommender Systems, History & Models

The first recommender system to use collaborative filtering was the Information Tapestry project at Xerox PARC. Tapestry was designed to support both content-based filtering and collaborative filtering. Collaborative filtering was supported through reactions, called annotations, users had to an electronic document. Other users could filter documents based on these annotations and solicit recommendations on document content using a query language called TQL. In 1992 the developers of the Tapestry project presented the system in the paper “Using collaborative filtering to weave an information Tapestry” (Goldberg, Nichols, Oki, & Terry, 1992).

Just a few years later the concept of collaborative filtering had already been applied in dozens of publicly available systems, several proprietary systems, and even some commercially available systems. The GroupLens project was established in 1992 by Paul Resnick and associates at the University of Minnesota, with the primary goal of examining automated collaborative filtering systems. GroupLens was built upon the Tapestry concept but incorporated a distributed network, allowing more users to interact with the system. In 1994 Resnick et al. automated the collaborative filtering process for GroupLens and introduced an automated collaborative filtering algorithm based on k-Nearest-Neighbor (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994).

Researchers at MIT created Ringo, a social information filtering system that allowed users to make recommendations for music. The Firefly recommendation system grew out of the Ringo project in 1996. Firefly was a collaborative filter that anonymously gathered users preferences and used the information to suggest web sites. Firefly was ultimately bought by Microsoft in 1998 and eventually became Microsoft Passport.
These first generation recommender systems used correlation-based and cosine-based techniques to compute similarities between users, and generally used a variation of the kNN algorithm originally created by Resnick (Lam & Riedl, 2004). The user-based, kNN algorithms have become known as the user-user algorithm and are described in detail in a paper written by Jonathan L. Herlocker et al. (Herlocker, Konstan, Borchers, & Riedel, 1999). In user-user neighborhood-based methods, a subset of appropriate users are chosen based on their similarity to the active user, and a weighted aggregate of their ratings is used to generate a prediction for the active user. All user-user algorithms have the following basic steps:

1. Weight all users that have rated the item of interest with respect to their similarity with the active user.
2. Select a subset, or neighborhood, of users to use as a set of predictors based on similarity.
3. Normalize ratings and compute a prediction from a weighted combination of the selected neighbors’ ratings.

This algorithm relies on the definition of a user similarity metric, an algorithm for selecting nearest neighbors and a method for computing a prediction. An empirical analysis conducted by Herlocker et al. found that the most accurate recommender systems based on user-user models utilized the following algorithmic components:

**USER-USER SIMILARITY WEIGHTING MEASURE**

Similarity weighting measurements are best performed using either Spearman correlation or Pearson correlation, as opposed to other know similarity measures (cosine, etc.). The Pearson correlation between two users is calculated using the following equation:
The score, or rating, of item $i$ by user $u$ is represented by $r_{ui}$. The summations occur over all items that are common between the two users, represented by the set $I(u,v)$, and the average ratings for each user, denoted by $R_u$ and $R_v$, are calculated only considering the items the two users have in common.

**USER-USER SIGNIFICANCE WEIGHTING**

The incorporation of significance weighting by devaluing correlations that are based on a small number of co-rated items provides a significant gain in the prediction accuracy of the system. In other words, if users had less than a threshold number, $N$, of similarly rated items the similarity measure was weighted by the factor $n/N$, where $n$ is the number of similarly rated items. If $n$ was greater than $N$ the similarity measure was un-weighted. Values of $N$ between 20 and 50 generally provided the best result.

**USER-USER NEIGHBORHOOD SELECTION**

Best-k neighbors (the k most similar users) proved to be the best approach to selecting neighbors to form a neighborhood. A value of 20 to 30 for $k$ is optimal in most cases, with accuracy declining for values of $k$ greater than 50.
**User-user Prediction Calculation**

Rating predictions could be calculated in various ways, but the following equation provides good accuracy with adequate performance:

\[
P_{u,i} = \bar{R}_u + \frac{\sum_{v \in N(u,i)} S_{u,v} (r_{v,i} - \bar{R}_v)}{\sum_{v \in N(u,i)} S_{u,v}}
\]

Equation 2: Rating prediction

Here the average rating for user \( u \) and user \( v \) is over all ratings associated with that user. The set \( N(u,i) \) is the user neighborhood for item \( i \) relative to the active user \( u \), i.e. the set of like-minded neighbors of \( u \) for item \( i \). Pearson’s correlation has been used in Equation 2 as the weighting factor, but other algorithms may use other similarity weightings.

By the year 2000 there were several collaborative filtering systems based on the user-user model in active use, both on the web and in commercially available software. They were quickly beginning to show their age, however. The radical growth in the amount of available information and the number of users started to pose some serious challenges for recommender systems, especially systems that were web-based and popular. Unfortunately, the user-user model (by this point it was being called the “traditional” model) didn’t cope well with increasing user count. Generating a prediction in the user-user model requires the construction of the user neighborhood, which involves ranking the similarity of the active user with all other system users who have rated the item that the prediction will be generated for. The amount of work required to
generate the neighborhood increases linearly with the number of participants in the system. The problem of increasing user count was exacerbated by new demands on how fast a recommender system should produce recommendations. By the year 2000, recommender systems were expected to produce high quality recommendations concurrently for millions of users, at a rate approaching several individualized recommendations per user per second. Pre-computing user similarities proved infeasible for the user-user model; a recommendation system with N users requires on the order of \( N^2/2 \) similarities. A modest system with 100 thousand users would have to store over 5 billion similarities. Doubling the system to 200 thousand users would require more than 40 billion stored similarities. Calculating, storing and efficiently accessing that many entries are daunting tasks, even with today’s hardware.

In 2001 Sarwar et al. (Sarwar, Karypis, Konstan, & Riedl, 2001) proposed a solution using the same correlation-based and cosine-based techniques used in user-user models. The authors of this new method recognized that the computational bottleneck in the user-user model was the neighborhood formation process, primarily because of all the user-user similarities calculated during that process. In their new model, now known as the item-item model, predictions are based on the similarity of items instead of the similarity of users. Because the number of items associated with a recommender system is usually orders of magnitude less than the number of users, using item-item similarities greatly reduces the number of similarity calculations. Also, item profiles are mostly static, especially when compared with user profiles, which can undergo constant change in some systems. The approximately static nature of the item profiles suggested that the item-item similarities should be pre-computed, further increasing the performance and scalability of the algorithm.
The basic *item-item* method searches the set of items the active user has rated and computes how similar these items are to the item the prediction will be generated for. The top \( k \) most similar items are selected, forming a \( k \)-Nearest-Neighborhood of items, and are used to calculate the utility of the target item for the active user. The prediction is generally computed by taking a weighted average using the active user’s ratings on these similar items. Like the *user-user* model, the *item-item* method has three basic steps:

1. Calculate the similarity of all items that have been rated by the active user with the item of interest.

2. Select a subset, or neighborhood, of items to use as a set of predictors based on similarity.

3. Calculate a prediction by computing a weighted average over all similar items that the active user has rated.

Also like its *user-user* cousin, the *item-item* algorithm relies on the definition of an item similarity metric, an algorithm for selecting nearest item neighbors and a method for computing predictions. Sarwar et al. provided the following guidelines for *item-item* recommender systems:

**ITEM-ITEM SIMILARITY WEIGHTING MEASURE**

The standard metric for *item-item* based models is the cosine similarity measure. The cosine similarity between two items is calculated by:
Here, $U(i,j)$ is the set of users that have rated both items $i$ and $j$, $r_{ui}$ is the rating of the $u$-th user for item $i$ and $R_i$ and $R_j$ are the average ratings of the $ith$ and $jth$ items.

The adjusted cosine similarity was designed to offset a drawback inherent in the basic cosine similarity used on item column vectors; the differences in rating scale between different users are not taken into account by the basic cosine metric. For the adjusted cosine similarity metric, the similarity between items $i$ and $j$ is given by:

$$S_{i,j} = \frac{\sum_{u \in U(i,j)} (r_{ui} - \bar{R}_i)(r_{uj} - \bar{R}_j)}{\sqrt{\left(\sum_{u \in U(i,j)} (r_{ui} - \bar{R}_i)^2\right)\left(\sum_{u \in U(i,j)} (r_{uj} - \bar{R}_j)^2\right)}}$$

Equation 3: Cosine similarity

Here, $U(i,j)$ is the set of users that have rated both items $i$ and $j$, $r_{ui}$ is the rating of the $u$-th user for item $i$ and $R_u$ is the average of the $u$-th users ratings.

Empirical studies show that the use of the adjusted cosine similarity measure usually provides the most accurate results for the item-item model. However, the cosine
similarity metric does have a big advantage over the adjusted cosine metric; it doesn’t require the computation of the average user ratings. Other authors have proposed different similarity measures (Deshpande & Karypis, 2004) but the cosine similarity measure remains a good choice for the item-item model, and is the measure that is used most often.

**ITEM-ITEM NEIGHBORHOOD SELECTION**

The Best-k item neighbors (the k most similar items) method was selected by default. Values of k over 20 yielded similar accuracy to k=20.

**ITEM-ITEM PREDICTION CALCULATION**

The best method of prediction calculates the sum of the ratings given by the active user over the items most similar to the target item. Each rating is weighted by the corresponding similarity between the target item and the associated neighbor:

\[ P_{u,i} = \frac{\sum_{j \in sim(i)} s_{i,j} \times r_{u,j}}{\sum_{j \in sim(i)} |s_{i,j}|} \]

Equation 5: Item-item model prediction calculation

Here, \( sim(i) \) is the set of the \( i-th \) items nearest neighbors and \( r_{u,j} \) is the rating the active user has made for the \( j-th \) item.

The item-based algorithms can provide better computational performance than traditional user-based collaborative methods while, at the same time, providing comparable or better prediction quality than the best available user-based algorithms.
Since its introduction the *item-item* model was rapidly incorporated into commercial recommender systems, especially systems that required excellent performance at high scale. One of the most famous recommendation systems was implemented soon after the introduction of the *item-item* method: Amazon.com Recommendation (Linden, Smith, & York, 2003). Amazon.com Recommendation utilizes the *item-item* model and incorporates a matrix of item similarities for over 100-thousand distinct items. It also supports tens of millions of users. With the success of Amazon.com Recommendation other web based recommender systems began to incorporate the *item-item* model. It quickly started to supplant the *user-user* model as the preferred method for implementing recommender systems, and has since become the most widely used recommendation algorithm.

Many other recommendation algorithms using collaborative filtering concepts have been developed as well, including models based on singular value decomposition, Bayesian networks and factor analysis. However, the *user-user* and *item-item* kNN models have enjoyed greater success, and have essentially dominated the recommender system landscape for the last 20 years. While recent research using the Netflix Prize dataset shows that kNN based approaches are easily beaten in terms of speed and accuracy by simple factor models (Bell & Koren, Lessons from the Netflix Prize Challenge, 2007) the kNN methods still work very well in an ensemble of methods (Bell, Toscher and Jahrer, The BigChaos Solution to the Netflix Grand Prize 2009). In fact, the algorithm that ultimately won the Netflix Prize challenge utilized several kNN variants to weight the residuals of other models. The kNN methods are also attractive because they are intuitive and relatively simple to implement, while still providing reasonable accuracy and scalability. It is likely that kNN models will be in use, at least in some capacity, for years to come.
Chapter 4: Tools, Data and Metrics Used in Model Evaluation

The process of implementing, collecting and defining the correct set of algorithms, datasets and evaluation metrics is a task any researcher studying recommender systems must ultimately address. Correct completion depends on the interplay between the models being analyzed, the properties of the selected datasets and which capabilities of the recommender system are most interesting to the researcher.

Evaluating the impact multi-user account have on the kNN models required the development, collection or adoption of several tools, datasets and metrics appropriate to the problem. The basic list includes:

1. Reliable implementations of the kNN user-user and item-item algorithms
2. Multi-user datasets with know contamination distributions
3. Evaluation metrics appropriate to the problem

This chapter describes these three components, presenting how they were developed, generated or decided upon.

KNN RECOMMENDER SYSTEMS

Both recommender systems under consideration were expected to train using a data model that contained more users than items and that used a 5-point ordinal scale for ratings. Rating entries can take one of 6 values, from 0 to 5, where 0 indicates that the item is unrated by the associated user. This data model is consistent with the Netflix Prize dataset, which is why it was selected for evaluation.

The recommender systems were developed in C, and utilized the most common attributes of the user-user and item-item models described in Chapter 3. Specifically, the user-user model utilized Pearson Correlation as the similarity measure and Equation 2 to
calculate predictions. Significance weighting was implemented with \( N=20 \), and the neighborhood size used in all prediction runs was \( k=20 \).

The implementation of the \textit{item-item} model utilized the popular cosine similarity (Equation 3) as its similarity measure and used the standard formula given in Equation 5 to calculate predictions. It also had a neighborhood size of \( k=20 \) but, since the items similarities calculated using the evaluation dataset almost always had a large number of common factors between items, no significance weighting was performed. This implementation of the \textit{item-item} model did not pre-compute item similarities, but this only reduced the performance profile of the implementation, not its accuracy.

Neither model was particularly accurate, but both performed respectably on the standard Netflix Prize\textsuperscript{2} dataset. Both implementations used the standard Netflix Prize training set to train while generating predictions for the probe set. The following table shows the published accuracy of various recommender systems on the Netflix Prize dataset, in relation to the implemented models used in this report:

\textsuperscript{2} See Appendix A for a description of the Netflix Prize dataset.
This table uses the metric RMSE to evaluate the accuracy of the predictions the recommender system generated for the corresponding dataset. The qualifying and probe sets have comparable statistical qualities, so the accuracy comparisons between the different models are relevant. The user-user and item-item based implementations had comparable accuracies to the trivial model, and were therefore deemed acceptable for the purposes of this evaluation.

### Multi-User Training Datasets

Evaluating how multi-user accounts affect recommender systems required training datasets that mixed standard accounts and multi-user accounts with known

<table>
<thead>
<tr>
<th>Recommender System</th>
<th>Dataset</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-user algorithm</td>
<td>Probe</td>
<td>0.9847&lt;sup&gt;3&lt;/sup&gt;</td>
</tr>
<tr>
<td>Item-item algorithm</td>
<td>Probe</td>
<td>1.0095&lt;sup&gt;4&lt;/sup&gt;</td>
</tr>
<tr>
<td>Trivial algorithm</td>
<td>Qualifying</td>
<td>1.0540</td>
</tr>
<tr>
<td>Cinematch</td>
<td>Qualifying</td>
<td>0.9514</td>
</tr>
<tr>
<td>“BellKor's Pragmatic Chaos” solution</td>
<td>Qualifying</td>
<td>0.8554</td>
</tr>
<tr>
<td>“The Ensemble” solution</td>
<td>Qualifying</td>
<td>0.8554</td>
</tr>
</tbody>
</table>

Table 1: Accuracy (RMSE) for the Netflix Probe Data

---

<sup>3</sup> Calculation based on approximately 70,000 predictions (5% of probe set).

<sup>4</sup> See footnote above.
proportions. Obtaining an appropriate natural dataset for the evaluation of a recommender system is always difficult, but obtaining a dataset that has a known distribution of multi-user accounts is not possible. Therefore, synthetic datasets had to be generated that were appropriately contaminated.

It is widely recognized that synthetic data (Tso & Schmidt-Thieme, 2006) can be used to detect flaws in prediction algorithms that cannot be accurately modeled with real datasets. Usually synthetic datasets are generated using statistical properties that are desirable for evaluating the algorithm being studied, and the researcher varies these properties to study the system’s response. Unfortunately, purely synthetic datasets do not accurately model (Herlocker J. L., Konstan, Terveen, & Riedl, 2004) the nature of real users and real data, and were not desirable for this evaluation. Instead, several synthetic datasets mimicking multi-user contamination were constructed by joining together random pairs of ‘pure’ accounts obtained from a large natural dataset (the Netflix Prize dataset). The subset of contaminated accounts was then combined with a disjoint group of ‘pure’ accounts, forming a synthetic dataset that contained a known percentage of multi-user accounts.

Accounts that had rated between 100 and 300 of the first two thousand most often rated movies were filtered from the Netflix Prize dataset, and the synthetic accounts comprising both the ‘pure’ and ‘corrupted’ subsets were randomly selected from those accounts. Only the ratings attributed the top two thousand most rated movies were retained for the synthetic accounts.

Figure 1 shows the distribution of rating count for each movie in the Netflix Prize dataset. The movies are ranked by total rating count\(^5\).

\(^5\) This should not be confused with the average rating of the movie. A movie with a low average rating can still have a high rating count.
By ranking the movies and only retaining ratings from the first two thousand movies, only 21% of the total number of ratings is eliminated. The sparsity of the dataset is also reduced; the number of non-zero entries in the truncated set contains 8.3% of the possible ratings, as opposed to 1.17% if all items in the dataset are retained.

**Figure 2** depicts the distribution range of the truncated dataset that accounts used to construct the synthetic dataset are selected from. The selection criteria were chosen to target accounts that have rated a substantial amount of the most popular movies. The accounts are intended to describe the rating behavior of a normal, but active individual user. This ensures that the kNN algorithms under evaluation are trained with accounts (both ‘pure’ and ‘corrupted’) that all have a significant number of ratings (each account selected rated between 10% and 30% of the items under consideration).
For each synthetic dataset fifty thousand accounts were selected at random from the target\(^6\) rating count distribution range. A percentage of those fifty thousand accounts were ‘corrupted’ by randomly selecting a second, but disjoint, subset of accounts from the target range and then merging the ratings from the accounts of the disjoint subset with the ratings of the accounts belonging to the corruption subset. The merging was done in pairs, i.e. one account from the disjoint subset was randomly paired with an account from the corruption subset and their ratings were merged. If both accounts forming the merged pair had rated the same item, the ratings of the primary account were retained. The result was a ‘corrupted’ account, the was, on average, weighted towards the opinions of the original account owner. Merging accounts in such a way simulated the addition of bi-

---

\(^6\) This was the only requirement placed on the selected accounts. No attempt was made to increase the accuracy of the prediction models by analyzing the truncated dataset for *shill* accounts, accounts with large numbers of extreme ratings, etc.
user accounts, i.e. one account used by two distinct ‘natural’ personalities, one possibly more active than the other. Only bi-user accounts were considered in this evaluation.

Five synthetic datasets with fifty thousand accounts were created, each with an increasing percentage of corrupted accounts. The corruption percentage was increased in 5% increments so corruption percentages ranged from 0% to 30%. New accounts comprising the ‘corrupted’ and ‘disjoint’ subsets were randomly selected for each of the five datasets, i.e. subset selections were never reused, but the initial fifty thousand ‘pure’ accounts were selected only once, and formed the basis of all five of the synthetic datasets. For comparison, a non-mixed version of each dataset was also constructed, and included ratings from the original, initially ‘pure’, accounts and ratings from the second account that did not overlap the original account. The non-mixed version of the dataset retains the identity of the new accounts used in the mixed version to corrupt the original accounts. The prediction results generated using the non-mixed versions of the datasets are used as reference values to quantify the effects of multi-user contamination, and represent the accuracy of the prediction model when user mixing is not present.

**Evaluation Metrics**

Most of the published evaluations of recommender systems have focused on the prediction accuracy of the system under examination. It is assumed that if a user could examine all the items available, they could place those items in an ordering of preference. An accuracy metric empirically measures how close a recommender system’s predicted ranking of items for a user differs from the user’s true ranking of preference. While some studies have considered non-accuracy based metrics\(^7\) (Herlocker J. L., Konstan, Terveen,  

\(^7\) Such as coverage, learning rate and user confidence
& Riedl, 2004), accuracy based metrics remain the most widely used tools for recommender system evaluation.

The root mean squared error, or RMSE, measures the squared deviation between predicted ratings and the users’ true ratings. The total RMSE for a set of N rating predictions is given by:

\[ RMSE = \sqrt{\frac{\sum_{i=1}^{N} (p_i - r_i)^2}{N}} \]

Equation 6: RMSE for a set of N rating predictions

The RMSE is an expectation, and therefore a number, not a random variable. It may be a function of an unknown parameter, but it does not depend on any random quantities. While particular values of RMSE other than zero are meaningless in and of themselves, they may be used for comparative purposes. Two or more statistical models may be compared using their RMSEs as a measure of how well they explain a given set of observations: The unbiased model with the smallest RMSE is generally interpreted as best explaining the variability in the observations.

Calculation of the RMSE for a recommender system is usually performed by testing a set of known ratings, called the probe set, against predictions generated by the recommender system. The recommender system neglects the ratings that are part of the probe set during training.

The RMSE metric has the disadvantage of heavily weighting outliers. This is a result of the squaring of each term, which effectively weights large errors more heavily than smaller errors. Still, RMSE is simple to calculate and has well-know statistical properties. The Netflix Prize competition used total RMSE on their qualifying set as the
sole criteria for determining a winner, and of all the accuracy metrics it is one of the easiest to apply and one of the most widely recognized.

While **RMSE** is a good metric for determining the overall predictive quality of a recommender system, it may be completely inappropriate for use in some evaluations. Researchers recognized early on that when recommenders are used to support decisions it can be more valuable to measure how often the system leads its users to wrong choices. Unfortunately, **RMSE** isn’t a good indicator of this. Consider, for example, a dataset with a five-point rating scale; are users sensitive to a change in mean absolute error of 0.01? Will such a change lead to more appropriate predictions, i.e. will the top-N items selected by the system more accurately reflect what is desirable to the user? These observations suggest that algorithmic evaluations of collaborative filtering systems may need more sophisticated metrics than **RMSE**.

Another category of accuracy metrics, called classification accuracy metrics, measures the frequency with which a recommender system makes correct or incorrect decisions about the desirability of an item to a user. One example is the ROC metric, which attempts to determine the extent to which an information filtering system can successfully distinguish between signal (in this case relevance) and noise. The ROC metrics make an assumption of binary relevance, i.e. items are relevant or they aren’t. The ROC metric is presented as a curve, or plot, of recall versus fallout\(^8\), where the points on the curve correspond to different values of a cutoff value. Items that the system ranks above the cutoff value are recommended by the system. Items below the cutoff value are discarded. Items recommended are either successful recommendations (relevant) or unsuccessful recommendations (non-relevant). If all relevant items appear before all non-

\(^8\) Fallout is the proportion of non-relevant items that are retrieved, out of all non-relevant items available.
relevant items in the recommendation list, the ROC curve is perfect, i.e. 100% of the relevant items will be recommended before any of the non-relevant items are.

Another set of classification metrics, called Precision and Recall, are the most popular metrics used for evaluating information retrieval systems. The Precision is defined as the ratio of relevant items selected to the number of items selected. Recall represents the probability that a relevant item will be selected, and is defined as the ratio of relevant items that were selected to all relevant items available. The computation of Precision and Recall are given as:

\[
\text{Precision} = \frac{N_{rs}}{N_s} \\
\text{Recall} = \frac{N_{rs}}{N_r}
\]

Equation 7: Precision and Recall

Here, \(N_{rs}\) is the number of selected and relevant items, \(N_s\) is the total number of selected items and \(N_r\) the total number of relevant items. The Precision and Recall metrics are set based metrics; they evaluate the quality of an unordered set of retrieved items.

This evaluation uses the Precision and Recall metrics to measure the impact on item ranking; more specifically, how the ranking of a selected list of items is changed when multi-user accounts are present. To calculate the mean Precision and Recall of a recommender system the following steps are performed:

1. A set of previously unrated items, \(M_u\), is selected for a user, \(u\), and ranked using the predictions generated by the model using the uncorrupted, i.e. unmixed, dataset as the training dataset.
2. The top-N ranked items are selected, and are identified as the relevant items.

3. The same set of items, \( M_u \), is re-ranked using the corrupted dataset as the training dataset.

4. The top-N ranked items are selected, and are identified as the selected items.

5. The intersection of the relevant set and the selected set of user \( u \) form the items that are both relevant and selected. The Precision and Recall for that user’s top-N list are calculated.

6. The previous 5 steps are repeated for a number of different users, and the mean values of the Precision and Recall are calculated.

This procedure is applied to ‘pure’ accounts and ‘corrupted’ accounts separately, in order to quantify the different affect multi-user corruption has on the list generation of each account type.

The mean Precision of a recommender system using a corrupted training dataset represents its ability to reconstruct a top-N item list that would be generated if no corruption was present. The mean Recall of a recommender system represents its ability to recall how many of the items that were relevant are still relevant if corruption is present. In this evaluation, since order isn’t considered and the number of retrieved documents is set to some constant \( N \), the Precision and Recall metrics collapse to the same metric\(^9\). This metric is defined as the Mean Reconstruction Precision of top-N item lists, or \( \text{MRP}_N \), and represents the average precision the system has in reconstructing a valid top-N item list for a user with corruption present.

\(^9\) That is the evaluation process is setting \( N_r = N_s \), so Precision = Recall.
Note that $\text{MRP}_N$ only represents the list reconstruction capabilities of a recommender system when faced with corruption; it doesn’t measure how valid the original top-N item lists are. This report uses $\text{RMSE}$ to measure the affect multi-user accounts have on the overall accuracy of a model, i.e. the average quality of its predictions. The $\text{MRP}_N$ is used to quantify the affect multi-user accounts have on top-N list generation.
Chapter 5: Evaluation of the Affects Multi-user Accounts have on kNN Recommender Systems

To evaluate the impact multi-user accounts have on kNN recommender systems an empirical analysis of the prediction accuracy of the two differing models, trained using synthetic datasets, was performed. The synthetic datasets were generated using rating data from the Netflix Prize dataset, and were constructed to simulate datasets corrupted by multi-user accounts at varying proportions. The procedure detailed in Chapter 4 was used in all cases. The RMSE and the MRP
 were applied to the prediction results of each model. The results are used to evaluate the different affects multi-user accounts have on each model. The response of each system’s prediction capabilities to an increasing amount of multi-user contamination determines the vulnerability of that model to the Multi-user Account Problem.

EVALUATION OF THE RMSE METRIC

The first metric considered was the RMSE. To facilitate the calculation of the RMSE, a probe set was generated using account-item pairings of already known ratings. Specifically, the probe set contained fifty thousand account-item pairings\textsuperscript{10}, randomly selected without replacement, from ratings that the original accounts had previously made, i.e. only the original ratings from both ‘pure’ and ‘corrupted’ accounts were considered. Both models generated predictions for each account-item pairing in the set, using the different corrupted datasets to train. The known ratings of the target items were not used in the prediction process, and were compared against generated predictions to calculate the RMSE of both the models. Figure 3 plots the RMSE of both models versus

\textsuperscript{10} This comprised around 0.5\% of the known ‘uncorrupted’ ratings in the un-corruption training set, a ratio that is comparable to ratio the Netflix probe set has (1\%) to the total Netflix Prize dataset.
corruption percentage. For comparison, the results for the non-mixed datasets are included.

Figure 3: RMSE vs. %corruption for user-user and item-item models

The RMSE results for both types of datasets indicate that the corrupted, or mixed, datasets caused a significant degradation in the overall quality of the predictions generated by both models. Further, the non-mixed version generally showed slightly increasing accuracy, attributable to the increase in the amount of training data injected by the corruption process. Figure 4 plots the difference in the RMSE between predictions generated using the mixed and non-mixed datasets at each corruption level.
The RMSE delta of the user-user model increases linearly with corruption, indicating that the overall quality of predictions generated by the user-user model are degraded by the presence of multi-user accounts. The RMSE delta increases, on average, by 0.003 for every 5% increase in corruption. At the 15% corruption level the RMSE has increased by almost 0.01, or approximately 1% over the RMSE of the predictions generated by the non-mixed datasets. It is well known that small improvements in RMSE for a recommender system translate into significant improvements in the quality of the top-N recommended items, so multi-user corruption can cause considerable degradation to the quality of the user-user model’s prediction capabilities.

The RMSE of the item-item model is much more resistant to multi-user account corruption, remaining relatively flat as the corruption percentage increases. It isn’t until 20% of the accounts are corrupted that the RMSE increases by any significant amount, and then by only 0.005, or 0.5%. Corruption levels over 20% show no additional increase in RMSE.
The resistance of the *item-item* model’s RMSE to increasing contamination can be explained by an examination of **Equation 3** (cosine similarity) and **Equation 5** (prediction calculation). The prediction calculation only considers ratings belonging to the active user’s account; interactions with other accounts only occur through the selection of a neighborhood of items using *item-item* similarities. If multiple users of a single account with different preferences only rated items they are interested in, and generally ignored uninteresting items, they would effectively be two different accounts when viewed from the context of *item-item* similarities (column similarities), particularly when calculated using the cosine similarity\(^{11}\). The only obvious way a multi-user account will cause inaccurate similarities between items is if one of the users is actively down-rating the items the other user is interested in. This seems unlikely, if not impossible.

Assuming it is true the *item-item* model has the ability to effectively decouple the ratings in multi-user accounts when training, it seems reasonable that the predictive capabilities of the model would increase with increasing corruption. Comparing the RMSE values associated with the predictions generated using the mixed and non-mixed datasets confirm this, at least for corruption levels under 20%. For low corruption levels the *item-item* model effectively decouples user data belonging to multi-users accounts, at least in terms of predictive accuracy. To the *item-item* model more data is just more data to train with, regardless of the account it is associated with.

The local effects multi-user accounts have on themselves may explain the increases in RMSE seen at corruption levels above 20%. However, the *item-item* model only uses items that are most similar to the item that is targeted for prediction, so one would expect the decoupling of users to also apply locally, effectively inoculating

\(^{11}\) The adjusted cosine similarity (**Equation 4**) does use the average of the ratings of other users to calculate similarity. Consequently, *item-item* models using the adjusted cosine similarity may be more vulnerable to the effects caused by multi-user accounts.
corrupted accounts from themselves. Still, as the percentage of corrupted accounts increased past 15%, so did the RMSE delta between the mixed and non-mixed datasets.

To try and evaluate if the main source of error above 15% comes from the ‘corrupted’ accounts, a ROC plot was created for the predictions used in the RMSE calculations to try and determine if the SE, or squared error, of each prediction could be used to classify ‘corrupted’ and ‘pure’ accounts. Prediction results using the 20% corrupted dataset to train the prediction model are the only results considered for the item-item model. For comparison, the corresponding classification curve for the user-user model is also included.

![Classification of Accounts Using MSE](image)

**Figure 5: ROC Curve Using SE as Account Type Classifier**

As both plots indicate, the SE of predicted ratings is ineffective as an account classifier, i.e. ‘corrupted’ accounts are no more likely to exhibit predominantly higher SE
values than ‘pure’ accounts are, at least for the probe set considered here. Correspondingly, local effects do not dominate the RMSE metric for this probe set. Predictions for both the ‘corrupted’ and ‘pure’ accounts have approximately the same distributions of SE, and any affect on the RMSE comes from a global increase in all of the individual SE values.

**Evaluation of the MRPₙ Metric**

The MRPₙ results were calculated using the procedure outlined in Chapter 4; one thousand accounts were randomly selected for each corrupted dataset, and fifty ratings were generated for each account on items that had previously not been rated. The fifty items were randomly selected for each account separately, and were replaced before selection for the next account occurred. These account-item pairings comprise the MRPₙ probe set for the targeted corrupted dataset, and predictions were made using the corrupted dataset to train the prediction models. The items for an individual user were then ranked by the prediction values, and the top ten items were retained.

For comparison, the same method was applied to the probe sets using the non-mixed datasets to train the prediction models. The final results were two thousand top-10 lists, two for each account. The MRPₙ was calculated by averaging the reconstruction precision for each of the one thousand accounts.

**Table 2** lists the calculated MRPₙ for both models at varying degrees of corruption. Results for the MRPₙ are reported separately for the ‘pure’ and ‘corrupted’ accounts.
Table 2: MRP for N=10

<table>
<thead>
<tr>
<th>Calculated</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>item-item (corrupted)</td>
<td>0.5264</td>
<td>0.5184</td>
<td>0.5072</td>
<td>0.4941</td>
<td>0.498</td>
<td></td>
</tr>
<tr>
<td>user-user (corrupted)</td>
<td>0.6528</td>
<td>0.6283</td>
<td>0.6106</td>
<td>0.6094</td>
<td>0.5719</td>
<td></td>
</tr>
<tr>
<td>item-item (pure)</td>
<td>0.8035</td>
<td>0.7371</td>
<td>0.7343</td>
<td>0.7126</td>
<td>0.7064</td>
<td></td>
</tr>
<tr>
<td>user-user (pure)</td>
<td>0.9021</td>
<td>0.8246</td>
<td>0.8029</td>
<td>0.7726</td>
<td>0.7489</td>
<td></td>
</tr>
</tbody>
</table>

The top-10 lists generated from the predictions of both models are markedly changed, even for small amounts of corruption. At a 5% corruption level the user-user model has a MRP_N of 0.9 for pure accounts, indicating that, on average, one of the items that would normally be in the top-10 list has been replaced. At 30% corruption two and a half items, on average, are replaced. Surprisingly, the affect account corruption has on the MRP_N values of the item-item model are greater, even at lower corruption levels. At 5% corruption, two items, on average, have already been knocked off of the top-10 lists belonging to the ‘pure’ accounts. This increases to three items at a 30% corruption level.

Both models generate comparable distributions for the reconstruction precision values associated with the ‘pure’ accounts. Figure 6 (item-item) and Figure 7 (user-user) are histograms of the reconstruction precision values for the ‘pure’ accounts that were generated at 30% corruption.
Figure 6: Item-item reconstruction precision distribution for ‘pure’ accounts

Figure 7: User-user reconstruction precision distribution for ‘pure’ accounts
The user-user model actually displays slightly better reconstruction precision for the ‘pure’ accounts than the item-item model does, both in terms of the $\text{MRP}_N$ and the ‘tightness’ of the distribution. Because the RMSE of the item-item model displays considerable resistance to multi-user corruption, it is hard to argue the top-10 list changes measured by $\text{MRP}_N$ are, necessarily, bad modifications. However, it is obvious that even small levels of corruption significantly affect the top-10 lists produced by the item-item model. The item-item model may display better overall predictive accuracy when faced with corruption than the user-user model does, but top-10 recommendation lists generated by the item-item model are affected just as much as are lists generated by the user-user model.

The item-item results for the ‘corrupted’ accounts are less surprising. Because the corrupted datasets only contain bi-user accounts generated using approximately\(^{12}\) equal amounts of each individual user’s rating data, it seems reasonable to expect the top-10 list generated for a randomly selected set of items to contain, on average, five items that are desirable to the original user and five items that are desirable to the ‘corrupting’ user. Figure 8 is a histogram of the item-item precision value distribution for ‘corrupt’ accounts, at 30% corruption. The normality of the distribution indicates that, on average, half of the items from the original list will be retained and that for most lists a significant percentage of the original items are likely to be retained. It is possible that the item-item model treats both the original user and the ‘corrupting’ user separately, replacing approximately five items in the list that are attractive to the ‘corrupting’ user, but retaining the best five original items for the original user. This conclusion would support

\(^{12}\)Because ratings already made by the original user are retained during the corruption process, the original user will, on average, have more associated data.
the theory that the *item-item* model has the ability to decouple users belonging to a corrupted account, effectively allowing accurate predictions for both users.

For corrupted accounts the *user-user* model generated top-10 lists that had an MRP$_N$ around 0.6, so usually 6 of the original items in the uncorrupted Top-10 lists are retained. This remained relatively constant with increasing corruption levels. **Figure 9** is a histogram of the *user-user* precision value distribution for ‘corrupt’ accounts, at 30% corruption.
The user-user model doesn’t treat both users in a bi-user account equally, and seems to favor the most active user. The distribution of the reconstruction precision also appears to be more random than the item-item model, exhibiting a large amount of variation.

To evaluate the relationship between list reconstruction precision and the proportion that an account is corrupted, several new datasets were created using various corruption proportions for the ‘corrupted’ accounts. The original accounts all contained between 100 and 120 ratings. Adding an additional number of ratings from a second account to the original account varied the corruption proportions for those accounts. The number of new ratings varied between 0% and 80% of the original number of ratings, and four datasets were generated to represent 10% increments of the corruption proportion,
i.e. the first dataset contained accounts that had between 0% and 10% of their ratings attributed to the second account, the second contained accounts that had between 10% and 20% of their ratings attributed to the second account, etc. For all the datasets only 10% of the accounts were corrupted at the targeted proportion.

The following graph shows the average $\text{MRP}_N$ for ‘corrupted’ accounts versus the corruption proportion. The graph includes error bars that represent the spread of the distribution for a particular average corruption proportion. The standard deviation is used in all cases to represent the spread of the distribution.

![Graph showing reconstruction precision distribution vs. corruption proportion](image)

**Figure 11: Reconstruction precision distribution vs. corruption proportion**

Both models exhibit significant changes in the top-10 lists generated for a random set of items even for small amounts of proportional corruption, indicating a strong local affect. For the *item-item* model the affect is partially moderated, in that the distribution
exhibits lower variation than the user-user model and the average values roughly depend on the corruption proportion.

For the user-user model list reconstruction is much more volatile. Even small levels of corruption exhibit large variations in the $\text{MRP}_N$ value. This is true even though the $\text{MRP}_N$ values for corrupted accounts don’t clearly depend on the corruption proportion for such large levels of corruption. Top-10 lists generated by the user-user model are much more likely to be ‘rewritten’ for corrupted accounts than lists generated using the item-item model.
Chapter 6: Conclusion

This report presents an empirical study of how two popular recommender systems respond to increasing levels of multi-user account contamination. The models under consideration were the venerable user-user kNN model and the widely used item-item kNN model. The results presented in this report indicate that the existence of multi-user accounts can have profound affects on both models, but in very different ways.

The user-user model displays decreasing levels of overall (RMSE) accuracy as the percentage of contaminated accounts increase, indicating that the existence of multi-user accounts strongly influence its ability to construct relevant recommendations. In addition, users that belong to multi-user accounts will receive greatly different recommendations than they would if they were the only user. While this seems like common sense, it is a subject not generally addressed by the designers of recommender systems.

The results for the item-item model were more interesting, both because the model itself if more relevant to the current state-of-the-art recommender systems and because of the content of the evaluation. The overall accuracy of the item-item model is largely unaffected by multi-user accounts. This report attributed this phenomenon to the item-item model’s ability to de-couple the data provided by two users belonging to a single account, especially for lower levels of corruption. To the item-item model, more data is more data to train with, regardless of the owner.

Regardless, the recommendation lists generated by the item-item model can still be significantly affected by account corruption. The affects are both local and global, i.e. accounts that are corrupted will be given a different set of recommendations than they
would have been given if no corruption was present, but ‘pure’ accounts will also see different recommendations, possibly caused by cross-correlations with corrupted accounts. While the affects are much less than those exhibited by the predictions generated by the user-user model, they are still substantial.

The Multi-user Account Problem can have a significant impact on the both models, and could impact many of the newer recommender models currently being developed. Consequently, the Multi-user Account Problem should not be ignored as a potential source of error in recommender systems.
Appendix A: The Netflix Prize Dataset

The dataset consists of 5-star ratings on 17770 movies and 480189 anonymous accounts. Netflix collected the dataset over a period of approximately 7 years. There are a total of 100480507 ratings; the probe set, comprised of 1408395, is a subset of them. The quiz set is an unknown 50% random subset of the qualifying set. The probe set has the same statistical properties as the qualifying set.

Two separate random sampling processes were employed to compose first the entire Prize dataset and then the quiz, test, and probe subsets used to evaluate the performance of recommender systems. The complete Prize dataset was formed by randomly selecting a subset of all accounts that provided at least 20 ratings between October, 1998 and December, 2005. All the ratings for a specific account were retrieved. To protect information about the Netflix subscriber base, a perturbation technique was then applied to the ratings in that dataset. The perturbation technique was designed to not change the overall statistics of the Prize dataset. Selecting, for each of the randomly selected accounts in the complete Prize dataset, a set of the most recent ratings, formed the qualifying set. These ratings were randomly assigned, with equal probability to three subsets: quiz, test, and probe. Selecting the most recent ratings reflects the Netflix business goal of predicting future ratings based on past ratings. The training set was created from all the remaining ratings and the probe subset; the qualifying set was created from the quiz and test subsets. The training set ratings were released to contestants; the qualifying ratings were withheld and form the basis of the contest scoring system. Based on considerations such as the average number of ratings per account and the target size of
the complete Prize dataset, the account’s 9 most recent ratings were selected to assign to the subsets. However, if the account had fewer than 18 ratings, only the most recent one-half of their ratings were selected to assign to the subsets.

The rating data found in the training dataset can be thought of as a large matrix, $R$, with 17,770 item (movie) columns and 480,189 account (user) rows. This matrix has 8,532,958,530 possible elements. The ratings data provided in the training dataset provide only 1.17% of those elements.
Appendix B: NFDB Database Format

The delivery format of the Netflix Prize rating dataset is 17770 text files, each containing the complete rating history of a single movie. The sizes of these files range from 70 bytes to 4.8 Mbytes, and depend on the number of ratings each of the movies had received. Searching the binary file format isn’t efficient, particularly when accessing information regarding a particular user\textsuperscript{13} (every text file would have to be scanned), so a simple database was developed, dubbed the NFDB\textsuperscript{14} database format, to allow quick searches on both the user id, and the movie id. Similar techniques were widely used by participants in the Netflix Prize contest to access rating entries.

The database consisted of two flat files, each recording entries as 64bit values (the unsigned long data type was used as storage). The item file, denoted with the suffix *_.mov.dat, encoded entries using the item id as the first 16bits. The user id, session id and rating followed. The exact bit format for an entry in the item file is:

| 1-16bit movie id | 1 | 32bit user id | 1 | 1-12bit session id | 1 | 1-4bit rating |

The user file, denoted with the suffix *_.usr.dat, encoded entries using the user id as the first 32bit, then the session id, movie id and rating. The exact bit format for an entry in the account file is:

| 1 | 32bit user id | 1 | 1-12bit session id | 1 | 1-16bit movie id | 1 | 1-4bit rating |

Each file was sorted by the value of the entries, creating two sorted lists, one with a major index by movie id, the other with a major index by user id. This allows fast access to all the ratings for a moving, using the item file, or all the ratings from a user, using the account file. Binary search algorithms are employed to locate entries.

\textsuperscript{13} The term user and account are being used interchangeably in this Appendix.
\textsuperscript{14} Or Netflix database format.
The NFDB data system provides efficient search algorithms exposed through a simple API interface. By utilizing the NFDB API, the user-user and item-item implementations are capable of generating more than 2 predictions per second\textsuperscript{15} even while using the entire Netflix Prize dataset. If item-item similarities had been pre-calculated, the item-item could have performed almost three orders of magnitude faster.

\textsuperscript{15} This was performed on a 2.35 GHz processor. The entire dataset (1.6 Giga-bytes) could be loaded into memory.
Glossary

Collaborative filtering - The process of filtering for information or patterns using techniques involving collaboration among sources.

k-Nearest-Neighbors (kNN) - a method for classifying objects based on closest training examples in the feature space.

Shilling – The act of entering fake information into a recommender system in order to affect the rating scores for targeted items.

Root Mean Squared Error (RMSE) - a frequently used measure of the differences between values predicted by a model or an estimator and the values actually observed from the thing being modeled or estimated.

Mean Reconstruction Precision (MRP_n) – A metric that determines the average number of items that are replaced on a top-N list when several lists generated by two different prediction algorithms are compared.

Similarity Measure – A defined metric that measures the similarity between two entities in a feature space.

Best-k neighbors – A neighbor selection rule for kNN algorithms. Best-k selection keeps the best k (some number) rated objects to form the neighborhood.
‘pure’ account – Accounts that contain rating data from a single user.

‘corrupted’ account – Accounts that contain rating data from several users. Bi-user accounts are the only types considered in this report.
VITA

James Adrian Edwards was born in Austin, Texas on December 16, 1969, the son of James Quincy Edwards and Judith Ann Edwards. He received the degree of Bachelor of Science from the University of Oregon in December 1992 and his Master of Arts in Physics from The University of Texas in December 1996. The author currently lives in Austin, Texas and is a full time employee of International Business Machines.

Permanent email address: jamise_2000@yahoo.com

The author typed this report