Item Reputation-Aware Recommender Systems

Ahmad Abdel-Hafez  
Queensland University of Technology  
Brisbane, Australia  
a.abdelhafez@qut.edu.au

Yue Xu  
Queensland University of Technology  
Brisbane, Australia  
yue.xu@qut.edu.au

Nan Tian  
Queensland University of Technology  
Brisbane, Australia  
n.tian@qut.edu.au

ABSTRACT
Recommender systems provide personalized advice for customers online based on their own preferences, while reputation systems generate a community advice on the quality of items on the Web. Both systems use users’ ratings to generate their output. In this paper, we propose to combine reputation models with recommender systems to enhance the accuracy of recommendations. The main contributions include two methods for merging two ranked item lists which are generated based on recommendation scores and reputation scores, respectively, and a personalized reputation method to generate item reputations based on users’ interests. The proposed merging methods can be applicable to any recommendation methods and reputation methods, i.e., they are independent from generating recommendation scores and reputation scores. The experiments we conducted showed that the proposed methods could enhance the accuracy of existing recommender systems.

Keywords  
Recommender System, Reputation System, User profile, Enrichment, Merging Ranked Lists, Personalized Reputation

1 INTRODUCTION
Today, recommender systems are an essential part of many Web 2.0 sites. Therefore, enhancing the accuracy of current recommender systems can significantly improve services provided by these websites and positively affect customer satisfaction [1]. Recommender systems suggest a list of items that are personalized based on the opinions of similar members in a target user's local community, while reputation systems provide the opinions of the whole community. The systems are similar in that they both collect user item data [2]. However, to our knowledge, only modest efforts have been made to incorporate item reputations in the recommendation process [2]. We suggest that combining item reputations with recommendations can enhance the accuracy of recommender systems.

Collaborative filtering and content-based filtering are two main filtering methods in recommendation making. The collaborative filtering (CF) [3] method exploits user ratings to identify other users with similar tastes to the target user, and then predicts items the target user might like based on the similar-users’ preferences. An item-to-item correlation system is applied in content-based filtering (CBF) [4]. Thus, the system recommends an item to the target user if the item content is similar to the content of an item the target user has previously liked or viewed. Recently, a third, hybrid system which combines both methods has emerged. In this paper, we made use of the user-based CF recommendation method for evaluation.

User-based CF recommender systems assume that people have similar tastes and will respond similarly to various items. Therefore, data from similar users is employed to generate recommendations for the target user. Item-based CF is a different approach that uses item similarities. This method detects similar items, rather than similar users. Similar items are those the system expects groups of users to prefer. In general, the CF method depends on the accuracy of the similarity functions to find the most similar users or items. A lack of sufficient data about users or items (e.g., in the case of cold start situations or sparse datasets) can negatively affect the accuracy of the recommendation. In these cases, the predicted items generated by CF may not reflect the relevance of the predicted items to the target user. This means that an item with no relevance to the target user may still earn high prediction value.

An item's reputation is calculated by a specific aggregation method based on ratings given by many users. The final aggregated value reflects the opinions of the whole community toward a specific item. High item-reputation scores can indeed reflect the quality of an item in the view of the whole community. Consequently, these scores can predict whether more (interested) users will like the item. However, if applied alone, reputation scores do not predict whether an individual user will like an item with high accuracy. This is because the reputation score does not consider the individual's specific preference; therefore, reputation scores are not personalized.

In this paper, we introduce methods to combine the recommender and reputation systems to enhance the accuracy of the top-N recommender results. Recommenders focus only on generating personalized results, without perceiving the global opinions of users about the recommended items. The reputation of an item reflects the quality of an item, which could affect a user opinion. We believe that adding the reputation awareness to the recommender systems has the potential to avoid it from recommending unsolicited items.
We use the weighted Borda Count method [5] to merge two sorted lists generated by recommender and reputation systems. However, the accuracy of this method is reduced by some noisy items that have low values generated by recommender and reputation systems and won't be recommended. It is because they will collect some votes from one or more of the candidate items on top which possibly will affect their ranking. We propose to use a recursive Borda Count called “Baldwin method” to solve this issue.

We conducted experiments to evaluate our method using a real dataset with different sparsity levels. The resulting accuracy of the proposed system was consistently better than the system that used only the CF method. The generality is one of the advantages of the proposed method, as any recommendation or reputation method can be used in conjunction. We employed a user-based CF method [6] and the Normal Distribution based Reputation with Uncertainty (NDRU) model [7]. We also tested other recommendation methods such as BiasedMF [8], SVD++ [8], and RegSVD [9].

In the rest of the paper we introduce the previous work in recommender and reputation systems in section 2. The detailed methods of the merging process are discussed in section 3. In section 4 we propose the personalized reputation system where we propose personalizing item reputation scores for each user, while keeping the global users' opinions. Section 5 describes the experiment and presents a discussion of the results. Finally we conclude in section 6.

2 Related Work

Recommender systems represent an essential component of many websites. Resnick and Varian suggested that recommender systems work similarly to word-of-mouth recommendations [10]. Resnick et al. introduced GroupLens, a system for the CF of net-news, in 1994 [3]. They defined the CF system as the one that helps people make choices based on the opinions of others. It worked, they said, by detecting users with similar tastes (neighbors) and then offering recommendations to the target user based on this neighbor data.

The CF approach is classified into model-based, memory-based, and hybrid approaches. Memory-based algorithms depend on user profiles to predict ratings or to generate the top-N recommended items [11]. The memory-based CF approaches can be classified into user-based and item-based approaches. The user-based approach generates a neighborhood of like-minded users (K-Nearest Neighbor [KNN]) based on profile similarity measures. Common similarity measures include the Pearson correlation coefficient (PCC) and the cosine similarity. These measures calculate predictions using weighted averages of the ratings given by other users in the neighborhood, where the weight is proportional to the similarity value between the target user and the neighborhood users. The same method can be applied for the item-based approach [10,12,13].

Model-based CF algorithms apply the user's earlier ratings to develop a model, which is then used to predict ratings for unrated items. The approaches used with the model-based CF include k-means clustering [14], the multiple multiplicative factor model [15], the Markov decision process [16], the restricted Boltzmann machine model [17], and the latent factor models based on the matrix factorization technique (i.e., singular value decomposition [SVD]) [8].

Reputation models use different methods to generate aggregated values that represent reputation scores; the Naïve model uses the average of the ratings of an item to measure the item's reputation, while many other models use the weighted average method as an aggregator to calculate item reputations based on item ratings. The weight can represent the user's reputation score, the time when the rating was given, or the distance between the current reputation score and the rating received [18,19].

Josang and Haller introduced a reputation model based on the Dirichlet distribution [20]. The authors used a cumulative vector $R_i$ to represent the aggregated ratings for agent $y$. $R_i = (R_i(1), ..., R_i(k))$ and $R_i(1)$ is the number of ratings of the level $i$, where $k$ is the number of possible different ratings level for an item. They added a decay factor to calculate the aggregate ratings, assuming that human agents change their behavior over time. They then calculated a single reputation score based on the multinomial probabilities derived from the aggregated ratings, which is defined in equation (1). $S_i(1)$ is the probability of rating $i$ that other agents give to agent $y$. The overall reputation is calculated by equation (2), which is the weighted sum of the rating probabilities with weights $v(i)$ evenly distributed in the range [0,1].

$$S_i(1) = \frac{R_i(1) + Ca(i)}{C + \sum_{j=1}^{k} R_i(j)} \quad |i = 1...k|$$

$$\alpha = \sum_{i=1}^{k} v(i) \times S_i(1), \quad v(i) = \frac{k-1}{k-1}$$

where $\alpha$ represents the overall reputation value, $S_i(1)$ represents the score vector of each rating level, $C$ is a constant value, and $a(i)$ is the base rate, which is equal to $1/k$.

Abdel-Hafez et al. [21] used the normal-distribution to generate weighted average reputation model which explicitly reflects the distribution of ratings of items. Their proposed Normal Distribution based Reputation with Uncertainty (NDRU) model is described as a weighted average method where the weights reflect the distribution of ratings in the overall score.

$$MR = \sum_{i=1}^{k} \left( l \times \left( \frac{n \times LW^i + C \times b}{C + n} \right) \right)$$

$k$ is the number of rating levels which represent the number of possible rating values that can be assigned to a specific item by a user. $n$ is the number of ratings per item. $LW^i$ is the summation of weights of every rating belongs to the rating level $i$, where the weights are generated using the normal-distribution. $C$ is a priori constant, here $C = 2$, and $b = \frac{1}{k}$ is a base rate for any of the $k$ rating values.

Recently, research has focused on improving the accuracy of recommender systems by combining the traditional recommendation methods with reputation systems [22]. Ku and Tai [22] proposed an exploratory framework to investigate the effects of recommendation and reputation systems on user purchase intentions toward recommended products. Their results showed that the opinions of other consumers influenced consumer attitudes about purchasing the recommended product through normative social influence. This revealed the effectiveness of recommendation systems that considered online reviews to influence consumers.
Jøsang et al. [2] suggested that combining reputation scores with recommendation scores would provide more accurate recommendations. They used the same belief model they had introduced in a previous work [23] to calculate reputation scores. The authors mentioned different methods for combining resulting scores, but they adopted the Cascading Minimum Common Belief Fusion (CasMin) method. They choose the smaller value between the belief of the system with higher belief value and the summation of belief and uncertainty of the other system to be the final belief score. This method ensured that the values from the recommender and reputation systems would need to be high to produce a high value in the CasMin fusion method.

3 A Reputation-Enhanced Recommender System

Our goal was to introduce a new reputation-aware recommender system that could enhance the accuracy of recommendations by filtering low-quality items based on reputation. The proposed method uses two ranked lists of items; the first list is generated by a recommender system, such as the user-based CF recommender system [6], and the second list is generated based on item reputations calculated using a reputation model, such as the NDRU reputation model [7]. The two ranked lists are then combined to enhance the accuracy of the recommendations. The proposed method is general, as it separates the implementation of the recommender system, the reputation system, and the merging process. In other words, we can apply any other recommendation method to generate the first list of items, and any other reputation model to generate the second list.

3.1 Definitions

The input of the proposed item-reputation-aware recommender system is user ratings. To make this model generalizable and applicable for any website, we intentionally did not use any other content information. The reputation and recommendation scores are generated from the available ratings and are considered input data. The following definitions for the input data are used throughout the paper.

- Users: $U = \{u_1, u_2, ..., u_{|U|}\}$ is a set of users who have rated at least one item.
- Items: $P = \{p_1, p_2, ..., p_{|P|}\}$ is a set of items that are rated at least one time by a user in $U$.
- Users-Ratings: this is a user-rating matrix defined as a mapping $ur: U \times P \rightarrow [0, r]$. If the user $u_i$ has rated the item $p_j$ with rating $a$, then $ur(u_i, p_j) = a$; otherwise, $ur(u_i, p_j) = 0$ such that $0 < a <= r$, and $r$ is the maximum rating.
- Item-Reputation Score: $S(p_j), p_j \in P$, where $S$ is a function representing the reputation method used to generate the reputation scores. The top $M$ items based on the reputation scores are defined as below:
  \[ TopM_{\text{Rep}} = \text{argmax}_{p_j \in P} S(p_j) \]
- Item Recommendation Score: $T(u_i, p_j), p_j \in P, u_i \in U$ , where $T$ is a function representing the recommendation method used to generate the recommendation scores. The top $M$ candidate items recommendations are generated based on the recommendation scores using equation (4).
  \[ TopM_{\text{Rec}} = \text{argmax}_{p_j \in P} T(u_i, p_j), u_i \in U \]  

3.2 Generating Recommendations by Merging the Two Ranked Lists

We propose two methods, the re-sorting and the weighted Borda-count methods, to combine the recommendation and reputation scores in order to generate the final top-$N$ recommendations. Before discussing the merging methods, we want to emphasize the differences between the two lists, as this was the justification behind the selection of the two methods. The recommender-generated lists represent personalized item recommendations for individual users. The reputation lists reflect the community opinion about items and are not related to individual user preferences. Therefore, we assumed that recommendation lists would be more accurate than using only impersonalized reputation lists. Thus, we prioritized the use of the recommender-generated lists over the use of the reputation-based lists and chose recommendation lists as the primary candidate recommendations.

3.2.1 Weighted Borda-Count Method

The Borda-count (BC) [24] method is a popular voting method that uses points to represent the multiple selections of a candidate; that is, if the list contains $N$ items, the first ranked item is given BC score of $N$ and the next one is $N-1$, and so on. If two items have the same rank, they get the same BC score and then the next item score reduced by 2. Every item that is outside the Top-$N$ list will receive a score of zero. Two ranked lists are merged by summing up the two BCs of the same item in the two lists. The final ranked list is sorted based on the BC sums of items. For an item $p \in P$, the sum of the BCs for this item is denoted $SBC(p)$.

The authors mentioned different methods for combining resulted recommendations. They used the same belief model they had adopted the Cascading Minimum Common Belief Fusion (CasMin) method. This value gives higher weight for the recommender system generated list. Algorithm 1 shows the process of generating the WBC scores for items to be recommended to a user (omit the notation for the user in Algorithm 1). Firstly we assign each item with WBC and BC scores equal to 0. Then we calculate the BC scores for the top-$M$ items in the recommender generated ranked list and the reputation generated ranked list. The final step is to calculate the WBC scores for all the items. The example provided in Figure 2 shows how this method works.

\[ TopN_{u}^{BC} = \text{argmax}_{p \in P} SBC(p) \]  

The recommender generated list and the reputation generated list have different influence on the top-$N$ recommendation accuracy. Therefore, they should be merged with different weights. We introduce a weight to the traditional BC method to emphasize difference between the two lists and to produce the best accuracy result. The proposed method is named Weighted Borda-count (WBC) method. Equation (6) shows how to calculate the WBC, by adding the weight $\alpha$ to the BC list generated by the recommender system and the weight $(1-\alpha)$ to the BC list generated by the reputation system, where $0 < \alpha < 1$. The Top-$N$ recommendations for the user $u$ based on the WBC method, are selected using the following two equations.

\[ WBC(p) = \alpha \times BC_{\text{Rec}}(p) + (1-\alpha) \times BC_{\text{Rep}}(p) \]  

Based on the experiment, the best results are achieved when $\alpha = 0.7$. This value gives higher weight for the recommender system generated list. Algorithm 1 shows the process of generating the WBC scores for items to be recommended to a user (omit the notation for the user in Algorithm 1). Firstly we assign each item with WBC and BC scores equal to 0. Then we calculate the BC scores for the top-$M$ items in the recommender generated ranked list and the reputation generated ranked list. The final step is to calculate the WBC scores for all the items. The example provided in Figure 2 shows how this method works.
3.2.2 Baldwin Method

In election systems, some candidates are only used to change the likelihood of the top candidates. Those candidates have no chance to win the elections but their presence can increase the chance of a specific candidate to win over another. It is because they will collect some votes from one or more of the candidates on top which possibly will affect their ranking. The Baldwin method is used to eliminate this dilemma by applying a recursive BC method [25]. First they calculate BC votes, then the candidate with the least number of votes is removed and the BC is calculated again without the removed candidate as he never existed. This process is repeated until only two candidates left.

This method is used to remove the noisy items that might change the ranking of top recommended items. In the first function call, the item with the lowest WBC score is moved from the original item list to the Baldwin final list. The recursive function will work again on the remainder of items as if the eliminated item never existed. The eliminated item will be added to the top of Baldwin’s final list, as more items come they push this item towards the end of the list. This process is repeated |P| times. In many cases the Baldwin method generates different rankings compared with the WBC method. Algorithm 2 shows the Baldwin ranking process.

Algorithm 2. Baldwin Method

Input: finalList = [], itemsLeft = P, n = |P|
Output: Sorted list of items.

1. int BaldwinMethod(int[] finalList, int[] itemsLeft, n) {
2. int[] sortedItems
3. if (n == 0)
4. return finalList
5. else
6. sortedItems = WBC(itemsLeft, n)
7. push sortedItems[n-1] to finalListn[0]
8. remove sortedItems[n-1] from itemsLeft
9. return BaldwinMethod (finalList, itemsLeft, n - 1)}

4 Personalized Item Reputation

An item’s reputation is the global community opinion about it. At a specific time, the ranking of items based on item reputation is the same for all users. This means that the top ranked items on the reputation-based list are not necessarily the items that a particular user likes. If the item recommendation is determined only based on item reputation, then the same items with the highest reputations will be recommended for all users. Similarly, when this list is combined with the recommender-generated list, the items at the top of the reputation list will dictate the recommendation list and will always have advantages over all other items for all users.

The other major problem with using the reputation-ranked list in recommendation systems is that items with high reputations can appear in the recommendation list despite that they are outside the scope of the individual user’s preferences. This causes a drop in recommendation accuracy. Therefore, we propose a personalized reputation for the items to tackle this problem. The idea was to build a user-preference profile based on previous user ratings, and then to use this profile to filter the items that were outside the preference scope.

4.1 Implicit Item Category

To produce the personalized reputation-based item list, we propose to cluster items based on user ratings or based on item categories. For clustering items based on user ratings, items that were rated by similar users are grouped in the same cluster. Each item cluster reflects certain common features shared by users with similar interests, and each cluster is called an “implicit item category”. In many application domains, the ontologies or taxonomies of the item/product categories are available; in such cases, we can use the provided ontology directly to group items based on item explicit categories.

In the experiment, we assumed that each implicit item category reflected a certain user preference for items. We could build an individual user’s preferences by collecting the categories of items the user had rated. We used only the positive ratings, as the items with negative ratings were not preferred. The implicit item category and user item preference are defined below:

- Implicit Item Categories C = \{C_1, C_2, ..., C_n\} is the set of categories, C_i \subseteq P and C_i \cap C_j = \emptyset.
- User Item Preference P_u = \{p \mid p \in P, ur(u, p) \geq \frac{r+1}{2}\}
contains the entire user’s preferred items, r is the maximum rating.

A user item preference P_u is a set of items that the user has rated positively. Ratings that are larger than or equal to \(\frac{r+1}{2}\) were considered positive ratings, where r was the maximum rating. Based on user item preferences, we defined user category preference as described below:

- User Category Preference F_u = \{C_i \mid C_i \subseteq C, (C_i \cap P_u) \neq \emptyset\} contains item categories to which the user's preferred or positively rated items belong. A user category preference F_u is a set of categories that are preferred by the user u.

The personalized reputation is defined as the degrading process for all the items in the reputation-ranked list that did not belong to the user preference. To apply the personalization to the reputation model, we degraded the reputation of all the items that belong to those categories which are not in the user preference. This step ensured that we did not recommend items outside the user's interest scope. The purpose of using reputation systems remained, as we did not change the reputation values of the other items, but kept the global community opinion. We only preserved or degraded the items based on the user's individual preferences. The derived resulting list is called personalized item reputation (PIR). The use of PIR guaranteed that the reputation-based ranked list was different for each group of users, which means that a greater variety of items would be considered compared to the number of
items considered using reputation without personalization. Equation (8) shows \( PIR_p \) calculation where \( S(p) \) is the reputation for the item \( p \).

\[
PIR_p = \begin{cases} 
S(p), & p \in C_i \cap C_i \in F_u, \\
0, & \text{Otherwise}
\end{cases}
\]  

(8)

### 4.2 User Preferences Enrichment

Using the PIR method raised a new concern regarding sparse datasets. Specifically, this was because it is common for a user to rate only a very small number of items. In this case, the number of categories in the user profile is low and, consequently, every item that belongs to other categories is degraded. We solved this problem by "enriching" the user’s categories if the number of categories in the user’s profile is less than a predefined minimum number. The minimum number of categories should be related to the average of ratings for a user. The user’s category preference profile is expanded with the categories which are popularly appearing in the neighbours’ profiles until the minimum number is reached. The popularity of categories are determined according to the number of times they appeared in the neighbours’ profiles. The result was an enriched personalized item reputation (EPIR) which was calculated exactly as the PIR but after performing the enrichment process described in Algorithm 3. The user neighborhood is defined below.

- **User Neighborhood** \( N_{u_i} = \{u_j | u_j \in \maxK{\{\text{sim}(u_i, u_j)\}} \} \) \( u_j \in U \) is the set of nearest \( K \) neighbors to user \( u_i \) in \( U \); where \( \maxK{\{\}} \) is required to obtain the top-K large values. We use the Pearson Correlation Coefficient similarity for generating the nearest neighbors.

**Algorithm 3. Enrichment Process**

**Input:** users’ category preferences \( F_u, u \in U \).

**Output:** enriched users’ category preferences \( F_u, |F_u| \geq \min \)

1. for \( C_j \in C \)
2. \( \text{frequency}[C_j] = 0 \)
3. for all users \( u_k \in N_{u_i} \)
4. for all categories \( C_j \in C \)
5. if \( C_j \in F_{u_k} \) and \( C_j \notin F_{u_i} \)
6. \( \text{frequency}[C_j] = \text{frequency}[C_j] + 1 \)
7. while \( |F_{u_i}| < \min \)
8. find \( C_j \) with \( \max(\text{frequency}[C_j]) \)
9. add \( C_j \) to \( F_{u_i} \)
10. \( \text{frequency}[C_j] = 0 \)

### 5 Experiment

We conducted the top-N recommender system experiment. We aimed to demonstrate that combining item reputation with user-based CF could enhance the accuracy of the top-N recommendations. The experiment is split into three parts: the first part aims to test the proposed ranked lists merging methods. We use the same recommender and reputation methods and we compare with the CasMin [2] method as a baseline. The second part of the experiment is to test the proposed personalized reputation method. We use the same recommendation and merging methods and we compare with the original reputation model as a baseline. The third part aimed to test if the proposed merging method and personalized reputation model can enhance the recommender accuracy for different recommenders used.

#### 5.1 Dataset

We used the MovieLens movie ratings dataset extracted from Grouplens.org. The dataset contained around 100,000 ratings on 1,682 movies provided by 943 users. We used this dataset in three different ways: 1) using all 2) using only 10%, and 3) using only 5% of the ratings. The purpose of the three tests was to observe the effects of this method on recommendation accuracy over dense and sparse datasets. The numbers of users and movies did not change in the three datasets; the only factor that changed was the number of ratings. Table 1 presents some of the statistics for each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MovieLens 5% (ML5)</th>
<th>MovieLens 10% (ML10)</th>
<th>MovieLens Complete (MLC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of ratings</td>
<td>6,515</td>
<td>13,077</td>
<td>100,000</td>
</tr>
<tr>
<td>Sparsity</td>
<td>0.99589</td>
<td>0.99175</td>
<td>0.93695</td>
</tr>
<tr>
<td>Min number of ratings per user</td>
<td>5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Max number of ratings per user</td>
<td>36</td>
<td>73</td>
<td>737</td>
</tr>
<tr>
<td>Average number of ratings per user</td>
<td>6.849</td>
<td>13.867</td>
<td>106.044</td>
</tr>
<tr>
<td>Min number of ratings per movie</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Max number of ratings per movie</td>
<td>59</td>
<td>114</td>
<td>583</td>
</tr>
<tr>
<td>Average number of ratings per movie</td>
<td>3.840</td>
<td>7.774</td>
<td>59.453</td>
</tr>
</tbody>
</table>

For each of the generated datasets, the ratings were selected randomly per user. However, we defined the minimum number of ratings selected for any user at 10 for the ML10 dataset and five for the ML5 dataset. This was because, when we split the dataset into training and testing sets, we wanted to ensure that there was at least; two items in testing for the ML10 dataset and 1 item for the ML5 dataset. For both datasets (ML10 and ML5), we generated 10 randomly selected additional datasets using the same method to perform a 10-fold experiment. We split each dataset into training and testing sets by randomly selecting 80% of each user's ratings into a training dataset and the rest into a testing dataset. For the MLC dataset, we performed a 5-fold experiment, where each time a different 20% of the dataset was selected for testing. We calculated the average of the results at the end. The sparsity for the datasets was calculated using equation (9).

\[
\text{Sparsity} = 1 - \frac{\# \text{ of Ratings}}{\# \text{ of Users} \times \# \text{ of Items}}
\]  

(9)

#### 5.2 Evaluation Metrics

We evaluated the top-N recommendation experiment with the globally used precision and recall metrics. The recommended item was considered a hit if it appeared in the user-testing dataset and the user has granted the item a rating \( \geq 3 \). We used the value of 3 because any rating \( < 3 \) in a 5-star scale system employed by this system indicates that the user did not like the item. Finally,
we used the F1-score metric to represent the results of both precision and recall.

5.3 Experiment Settings
We conducted the first two experiments in three runs for each dataset using the settings $N = 20$, $M = 60$, and the nearest neighbors $K = 20$. While the last experiment used only MLC dataset. The experiments comprised three parts: The first part is the reputation method. We used the user-based CF introduced in [6] for the first two experiments. In the last experiment we tested other recommender methods. The second part is the reputation model, and the third part is the ranked lists proposed merging methods.

5.4 Testing Proposed Merging Methods
In this experiment we tested three methods for merging the reputation and recommender ranked lists; the baseline method is the CasMin method proposed in [2], and our two proposed methods the weighted Borda-Count (WBC) and the Baldwin. The recommender used with the three merging methods was the user-based CF [6] and the reputation model was the NDRU reputation model which was calculated using equation (3) [7]. We chose this model because it can provide more accurate results no matter the sparsity of the dataset. However, any other reputation model can be used instead.

We used the CasMin method proposed in [2] as a baseline to compare with. The CasMin method is the only method we know to propose combining recommender and reputation scores. Table 2 shows the precision, recall, and F1-scores for the baseline and the proposed merging methods over the three datasets. The first thing we noticed from the results was that the Baldwin method produced the best results of the other merging methods. Because the Baldwin eliminate the noise produced by undesired items, and their effect on the overall ranking produced by the weighted Borda Count.

In our experiment, the CasMin method results were very low and worse than the user-based CF method. The authors of the CasMin method didn’t provide an experiment in their paper. The CasMin method invokes a strict rule that the difference between the results of recommender and reputation must be less than the uncertainty score of the method with lower value. In this case only the result is the higher score between the two methods, otherwise the result is the lower one added to its uncertainty value. We noticed that the CasMin method does not enhance the results of the basic user-based CF over datasets with different densities.

In contrast, the WBC method enhances the results of the basic user-based CF over the three datasets, but the Baldwin method performed better than the WBC. The reason is because the WBC method often incorporates noisy items that change the ranking of the top items in both ranked lists. However, the Baldwin method is used to eliminate items of this kind. Because the Baldwin method performed the best in the first experiment, we used this method in the following experiments.

5.5 Testing Proposed Personalized Item Reputation
The second part of the experiment comprised generating a ranked list of items using the personalized items reputation. We implemented NDRU reputation model [7] as a baseline for this experiment. We used the movie categories provided with the MovieLens dataset to generate user category preferences. The reputation methods tested were:

1. NDRU: the Normal Distribution based Reputation with Uncertainty model which is the baseline method.
2. PIR: the proposed personalized item reputation; we used the (NDRU) method as the basic reputation method, and then modify the ranked list based on each user preference as explained in section 4.
3. EPIR: an enriched version of the PIR. We first checked the number of categories rated by the user, and if the number was less than the determined number, we proceeded to the enrichment process. Based on the experiment we used

<table>
<thead>
<tr>
<th>Recommender-Reputation</th>
<th>Merging Method</th>
<th>ML5</th>
<th>ML10</th>
<th>MLC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1-score</td>
<td>Precision</td>
</tr>
<tr>
<td>CF</td>
<td>0.0061</td>
<td>0.0684</td>
<td>0.0112</td>
<td>0.0079</td>
</tr>
<tr>
<td>CF-NDRU</td>
<td>0.0004</td>
<td>0.0032</td>
<td>0.0007</td>
<td>0.0004</td>
</tr>
<tr>
<td>CF-NDRU</td>
<td>0.0075</td>
<td>0.0665</td>
<td>0.0134</td>
<td>0.0079</td>
</tr>
<tr>
<td>CF-NDRU</td>
<td>0.0077</td>
<td>0.0611</td>
<td>0.0137</td>
<td>0.0089</td>
</tr>
</tbody>
</table>

Table 3. Results of top-N recommendations accuracy for different reputation lists using Baldwin merging method.

<table>
<thead>
<tr>
<th>Recommender-Reputation</th>
<th>ML5</th>
<th>ML10</th>
<th>MLC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1-score</td>
</tr>
<tr>
<td>CF-NDRU</td>
<td>0.0077</td>
<td>0.0611</td>
<td>0.0137</td>
</tr>
<tr>
<td>CF-PIR</td>
<td>0.0182</td>
<td>0.1661</td>
<td>0.0328</td>
</tr>
<tr>
<td>CF-EPIR</td>
<td>0.0201</td>
<td>0.1812</td>
<td>0.0362</td>
</tr>
</tbody>
</table>
(6,12,24) as minimum numbers of categories for the ML5, ML10, and MLC datasets, respectively.

Table 3 shows the results for using different reputations ranked lists combined with the basic user-based CF using the Baldwin merging method over the three generated datasets. We noticed that when we sorted the final recommended items according to the personalized reputation scores the accuracy was improved over the non-personalized reputation scores. The only explanation was that the items in the top-M list generated by the CF did not belong to the set of categories that the user preferred. However, the personalized reputation system was able to filter those items so that better results could be obtained. This leads us to conclude that reputation scores are better to be personalized using users preferences before they are used for recommendation purposes.

It is now clear that the proposed personalized methods of reputation model generated better results than did the original reputation lists. We had two versions of this kind of reputation: the PIR and the EPIR. Using the ML5 and ML10 datasets, the EPIR produced slightly better results than did the PIR method. This means that the neighbor categories could be used to enrich the user categories by increasing the diversity of recommendations, while still producing more accurate results. When we used the MLC dataset, both methods produced exactly the same results. This was because no enrichment was required for these dense datasets. Moreover, the NDRU reputation model produced results different from those of the Naive method, which indicated that the reputation method should be carefully selected to enhance results.

5.6 Testing Merging Method and Personalized Reputation With Other Recommender Systems

In this experiment we intended to test the proposed methods with several available recommender systems to make sure that the accuracy enhancement is not restricted to the results of the basic user-based recommender system. We used the reputation-enhanced recommender method, EPIR, as the reputation method, and Baldwin method for merging. The dataset used was the complete MovieLens (MLC) dataset described above. We tested the proposed method using the SVD++, BiasedMF [8], and RegSVD [9] recommender systems. All the methods are tested twice with different number of factors. We wanted to check if the number of factors used affects the accuracy enhancement of the proposed method for the recommender systems used or not.

The experiment is done as 5-fold cross-validation with 20% testing dataset and 80% training. The maximum number of iterations used for all the methods is 100, and the all were testing using 5 and 20 factors. The item and user regularizes value is 0.1 in both the BiasedMF and SVD++, while it was set to 0.05 for the RegSVD method. The learn rate was 0.005, 0.07, 0.01 for the RegSVD, BiasedMF, and SVD++ respectively.

Table 4 shows the results of this experiment. We noticed that the proposed method of EPIR using the Baldwin merge method enhance the accuracy results of the implemented recommender systems. Both values of precision and recall are improved. When the number of factors increases, the proposed method still generates better accuracy results. This leads us to conclude that the top-N items in recommenders include noisy ones that can be filtered out using item reputation scores.

6 Conclusions and Future Work

In this paper, we presented a new method for enhancing the accuracy of top-N recommendations using reputation systems. We introduced a personalized reputation method to render the utility of using reputation to improve the performance of recommender systems. Based upon the evaluations, we have important findings to share. First, reputation models do not necessarily produce better results when they are incorporated with recommender systems. On the contrary, reputation models without personalization can reduce the accuracy of the recommendations. The second significant finding is that personalized reputation scores can be very helpful for improving the accuracy of recommender systems.

In the future we plan to investigate other methods for merging recommender and reputation lists. Other voting systems methods can be employed and tested. We believe that there is potential for different voting systems to generate good accuracy results if they are the results of several merging methods also can be combined to produce the best accuracy.

References


11. H. Liang. 2010. User profiling based on folksonomy information in Web 2.0 for personalized recommender systems. Doctor of Philosophy, Faculty of Science and Technology, Queensland University of Technology, QUT ePrints.


