# Qualitative Analysis of User-based and Item-based Prediction Algorithms for Recommendation Systems

Manos Papagelis[1, 2], Dimitris Plexousakis[1, 2], Ioannis Rousidis[2] and Elias Theoharopoulos[1,2]

[1] Institute of Computer Science, Foundation for Research and Technology - Hellas

P.O. Box 1385, GR-71110, Heraklion, Greece

{papaggel, dp, theohar}@ics.forth.gr, rousidis@csd.uoc.gr

[2] Department of Computer Science, University of Crete

P.O. Box 2208, GR-71409, Heraklion, Greece

## ABSTRACT

Recommendation systems employ prediction algorithms to provide users with items that match their interests. In this paper, we describe and assess several prediction algorithms, some of which are novel in that they combine user-based and item-based similarity measures derived from either explicit or implicit ratings. We compare both statistical and decision-support accuracy metrics of the algorithms against different levels of data sparsity and different operational thresholds. The first metric evaluates the accuracy in terms of average absolute deviation, while the second evaluates how effectively predictions help a user select high-quality items. Our experimental results indicate better performance of item-based predictions derived from explicit ratings in relation to both metrics. Finally, we present theoretical extensions of our work that permit to achieve incremental update of similarity measures and to identify virtual, self-organized, online communities.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – information filtering, retrieval models; H.3.4 [**Information Storage and Retrieval**]: Systems and Software – Performance evaluation (efficiency and effectiveness);

## General Terms

Algorithms, Measurement, Performance, Experimentation.

## Keywords

Collaborative filtering, social filtering, similarity measures, content analysis, item-based prediction algorithms, user-based prediction algorithms, recommendation systems, community coherence grade, online communities.

## 1. INTRODUCTION

Recommendation systems [1] have been a popular topic of research ever since the ubiquity of the web made it clear that people of hugely varying backgrounds would be able to access and query the same underlying data. The initial human-computer interaction challenge has been made even more challenging by the observation that customized services require sophisticated data structures and well thought-out architectures to be able to scale up to thousands of users and beyond.

In recent years, such systems are extensively adopted by both research and e-commerce applications in order to provide an intelligent mechanism to filter out the excess of information available and to provide customers with the prospect to effortlessly find out items that they will probably like according to their logged history of prior transactions.

The use of efficient and accurate prediction algorithms is vital for a system that aims to provide recommendations to its users. If we define a *prediction* to be a number value that expresses the predicted likelihood that a user will "like" an item, then a *recommendation* is defined as the list of N items with respect to the top N predictions from the set of items available. Improved prediction algorithms indicate better recommendations. This explains the essentiality of exploring and comprehending the broad characteristics and potentials of prediction algorithms and the reason why this work concentrates on this research direction.

Recommendation algorithms are classified into *content-based* and *collaborative filtering* based, while hybrid techniques have been proposed as well. Content-based algorithms [3, 4] take into account the content of documents and filter in the ones that better match with user's preferences and logged profile. Collaborative-filtering based algorithms [1, 3, 5] aim to identify users that have relevant interests and preferences by calculating similarities and dissimilarities between user profiles. The idea behind this method is that it may be of benefit to one's search for information to consult the behaviour of other users who share the same or relevant interests and whose opinion can be trusted.

The challenges for recommendation algorithms expand to three key dimensions, identified as *sparsity*, *scalability* and *cold-start*. Sparsity occurs, as only a few of the total number of items available in a database are rated by the users. Scalability refers to the need of providing high quality recommendations promptly, even when the number of users and items in database scales up to thousands and beyond. Finally, cold-start describes the case in which a number of items cannot be recommended. This problem applies to new and obscure items and is particularly detrimental to users with eclectic taste [6, 7].

## 2. SIMILARITY MEASURES

In this section, we present a set of similarity measures based on the Pearson correlation coefficient, a metric of relevance between two vectors [8]. When the values of these vectors are associated with a user's model then we call it *user-based similarity*, whereas when they are associated with an item's model then we call it *item-based similarity*. The similarity measure can be effectively used to balance the ratings significance in a prediction algorithm and therefore to improve accuracy.

There are several similarity algorithms that have been used: cosine vector similarity, Pearson correlation, Spearman correlation, entropy-based uncertainty measure and mean-squared difference. In [9] Breese et al. suggest that Pearson correlation performs better than cosine vector similarity, while in [10] Herlock et al. suggest that Pearson's correlation performs better than Spearman's correlation, entropy-based uncertainty and mean-squared difference for collaborative filtering. According to these remarks we decide on Pearson correlation to calculate item-based and user-based similarities taking advantage of both explicit and implicit ratings.

An *explicit rating* identifies the preference of a user to a specific item. A user is prompted by the system's interface to provide ratings for items so as to improve his model. The more ratings the user provides, the more accurate the recommendations provided to him are. Ratings range from 1 to 10 with 1 expressing greatest aversion to the item and 10 expressing greatest liking to the item. Explicit ratings are logged by the system and form the user's model.

*An implicit rating* [11, 12] identifies the preference of a user to specific categories[1]. We use here the term "implicit" somewhat excessively so as to express that a user is never actually prompted to express his preference to categories. We take advantage of the fact that an item belongs to some categories and we develop a user model based on category preference. If the explicit rating of a user concerning a specific item belonging to a set of categories is considered "good" then his model is updated so as to include the preference and vice versa. A rating is considered as "good" when it is greater than or equal to a threshold.

Before describing the algorithms we introduce some definitions that facilitate the explanation process. First of all, we define

- A set of $m$ users $U = \{u_x : x = 1, 2, ..., m\}$
- A set of $n$ items $I = \{i_x : x = 1, 2, ..., n\}$
- A set of $p$ categories $C = \{c_x : x = 1, 2, ..., p\}$
- A set of $q$ explicit ratings $R = \{r_x : x = 1, 2, ..., q \wedge q \le m * n\}$
- A set of $t$ implicit ratings $R' = \{r'_x : x = 1, 2, ..., t \wedge t \le m * p\}$

We also define three matrices that derive from our data, the user-item matrix, the user-category matrix and the item-category bitmap matrix.

*User-item matrix* is a matrix of users against items that have as elements the explicit ratings of users to items. Some of the user-matrix cells are not filled, as there are items that are not rated by some users.

*User-category matrix* is a matrix of users against item categories that have as elements, values that show the number of times a user has rated positively or negatively for a category. For each category we keep two columns, one for positive ratings and one for negative ratings.

*Item-category bitmap matrix* is a matrix of items against categories that have as elements the value 1 if the item belongs to the specific category and the value 0 otherwise.

Similarity is calculated over the parts of two vectors that derive from one of these matrices as it is depicted by the shadowed parts in figure 1.

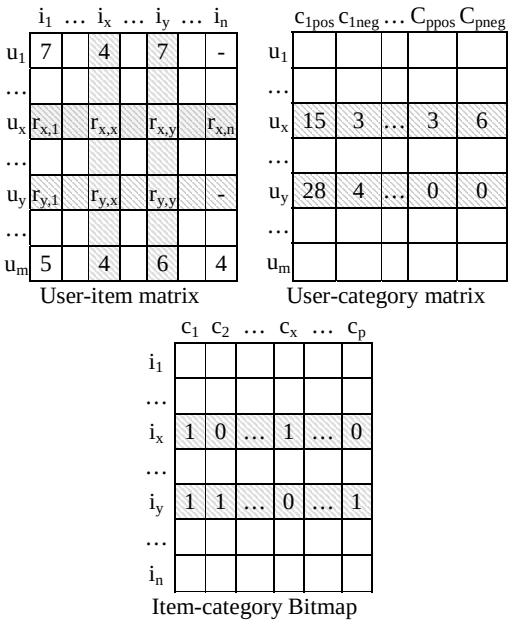



**Figure 1**. Matrices for calculating similarities

## 2.1 User-based Similarity

### *2.1.1 Based on Explicit Ratings*

If we define the subset of items that users $u_x$ and $u_y$ have co-rated as $I' = \{i_x : x = 1, 2, ..., n' \wedge n' \le n\}$, where $n$ is the total number of items in the database, then the similarity between the two users is defined as the Pearson correlation of their associated rows in the user-matrix and is given by equation 1.

$$\kappa_{x,y} = sim(u_x, u_y) = \frac{\sum_{h=1}^{n'} (r_{u_x,i_h} - \overline{r_{u_x}})(r_{u_y,i_h} - \overline{r_{u_y}})}{\sqrt{\sum_{h=1}^{n'} (r_{u_x,i_h} - \overline{r_{u_x}})^2} \sqrt{\sum_{h=1}^{n'} (r_{u_y,i_h} - \overline{r_{u_y}})^2}} \quad (1)$$

### *2.1.2 Based on Implicit Ratings*

The values of the user-category matrix elements are used to calculate the preference of the user $u_x \in U$ to the category $c_x \in C$ and the result is considered as an implicit rating $r'_{u_x,c_x} \in R'$ to that category, where $r'_{u_x,c_x} = \frac{c_{x_{pos}}}{c_{x_{pos}} + c_{x_{neg}}} * 10$

[1] Items in the database belong to categories.

and $c_{x_{pos}}$ , $c_{x_{neg}}$ are respectively the number of positive and negative ratings[2] that user $u_x$ has implicitly given to category $x$.

The similarity between the two users is defined as the Pearson correlation of their implicit ratings to categories $c \in C$ and is given by equation 2.

$$\lambda_{x,y} = sim(u_x, u_y) = \frac{\sum_{h=1}^{p}(r'_{u_x,c_h} - \overline{r'_{u_x}})(r'_{u_y,c_h} - \overline{r'_{u_y}})}{\sqrt{\sum_{h=1}^{p}(r'_{u_x,c_h} - \overline{r'_{u_x}})^2}\sqrt{\sum_{h=1}^{p}(r'_{u_y,c_h} - \overline{r'_{u_y}})^2}} \quad (2)$$

Where $p$ is the number of categories available.

## 2.2 Item-based Similarity

### *2.2.1 Based on Explicit Ratings*

If we define the subset of users that have rated both items $i_x$ and $i_y$ as $U' = \{u_x : x = 1,2,...,m' \wedge m' \le m\}$, where $m$ is the total number of users in database then the similarity between the two items is defined as the Pearson correlation of their associated columns in the user-matrix and is given by the equation 3.

$$\mu_{x,y} = sim(i_x, i_y) = \frac{\sum_{h=1}^{m'}(r_{u_h,i_x} - \overline{r_{i_x}})(r_{u_h,i_y} - \overline{r_{i_y}})}{\sqrt{\sum_{h=1}^{m'}(r_{u_h,i_x} - \overline{r_{i_x}})^2}\sqrt{\sum_{h=1}^{m'}(r_{u_h,i_y} - \overline{r_{i_y}})^2}} \quad (3)$$

### *2.2.2 Based on Item-category Bitmap*

There is also an alternative way to calculate the correlation between two items taking into account the categories in which they belong. The similarity between the two items is defined as the Pearson correlation of their associated rows in the item-category bitmap matrix and is given by equation 4.

$$\nu_{x,y} = sim(i_x, i_y) = \frac{\sum_{h=1}^{p}(v_{c_h,i_x} - \overline{v_{i_x}})(v_{c_h,i_y} - \overline{v_{i_y}})}{\sqrt{\sum_{h=1}^{p}(v_{u_h,i_x} - \overline{v_{i_x}})^2}\sqrt{\sum_{h=1}^{p}(v_{u_h,i_y} - \overline{v_{i_y}})^2}} \quad (4)$$

where $p$ is the number of categories and $v_{c_h,i_x}$ is a value equal to 1 if the item $x$ belongs to category $h$ and equal to 0 otherwise.

# 3. PREDICTION ALGORITHMS

Prediction algorithms [9] try to guess the rating that a user is going to provide for an item. We will refer to this user as *active user* $u_a$ and to this item as *active item* $i_a$. These algorithms take advantage of the logged history of ratings and of content associated with users and items in order to provide predictions.

[2] Ratings are considered as positive or negative when they are greater or lower than a threshold respectively

## 3.1 Random Prediction Algorithms

The random prediction algorithm represents the worst case of prediction algorithm[3], since instead of applying a sophisticated technique to produce a prediction it generates a random one. We refer to the random prediction algorithm so as to have a reference point at how much better results we obtain by the utilization of more sophisticated ones.

## 3.2 User-based Prediction Algorithms Description

User-based prediction algorithms are based on user's average rating and an adjustment to it, as given by equation 5.

$$prediction = user\_average + adjustment \quad (5)$$

*Adjustment* is most often a weighted sum that integrates user-based or item-based similarity measures. Since prediction arises as the sum of the two, improvements can be considered in both operators. Next, we present the classic user-based collaborative filtering prediction algorithm and we suggest some improvements taking advantage of the different user-based and item-based similarity algorithms described in the earlier section.

### *3.2.1 User-based with Explicit Ratings ($CF_{UB\text{-}ER}$)*

This is the classic user-based collaborative filtering prediction algorithm and comes up as the sum of the active user's average rating, regarding the whole set of items that the active user has rated, and an adjustment. The adjustment is a weighted sum of the other users' ratings concerning the active item and their similarity with the active user. The algorithm is given by the equation 6.

$$CF_{UB-ER} = p_{u_a,i_a} = \overline{r}_{u_a} + \frac{\sum_{h=1}^{m'} k_{a,h}(r_{u_h,i_a} - \overline{r}_{u_h})}{\sum_{h=1}^{m'}|k_{a,h}|} \quad (6)$$

where $m'$ is the number of users that have rated the item $i_a$ and $\overline{r}_{u_a} = \frac{\sum_{h=1}^{n} r_{u_a,i_h}}{n}$, i.e. the user's average rate, and $n$ is the number of items that the active user has rated.

### *3.2.2 User-based with Explicit Ratings and Category Boosted ($CF_{UB\text{-}ER\text{-}CB}$)*

Instead of calculating the active user average rating over the total number of rated items, we can calculate the active user average rating over the subset of rated items that belong to the same categories as the active item. The algorithm is given in equation 7.

$$CF_{UB-ER-CB} = p_{u_a,i_a} = \overline{r}_{u_a} + \frac{\sum_{h=1}^{m'} k_{a,h}(r_{u_h,i_a} - \overline{r}_{u_h})}{\sum_{h=1}^{m'}|k_{a,h}|} \quad (7)$$

[3] Actually, this is not absolutely true. We can artificially produce worse prediction algorithms than the random-based one but in order to do this we also need some kind of logged information

where $m'$ the number of users that have rated the item $i_a$ and $\overline{r}_{u_a} = \frac{\sum_{h=1}^{n} r_{u_a,i_h}}{n}$, i.e. the user's average rate, where $n$ is the number of items that the active user has rated and at least one of the categories that $i_a$ and $i_h$ belong to are common

### *3.2.3 User-based with Implicit Ratings ($CF_{UB\text{-}IR}$)*

Instead of using the user-based explicit ratings similarity $\kappa$, we can use the user-based implicit ratings similarity $\lambda$ in order to calculate the similarity between the active user and the other users. The prediction algorithm is given by the equation 8.

$$CF_{UB-IR} = p_{u_a,i_a} = \overline{r}_{u_a} + \frac{\sum_{h=1}^{m'} \lambda_{a,h}(r_{u_h,i_a} - \overline{r}_{u_h})}{\sum_{h=1}^{m'} |\lambda_{a,h}|} \quad (8)$$

where $m'$ is the number of users that have rated the item $i_a$ and $\overline{r}_{u_a} = \frac{\sum_{h=1}^{n} r_{u_a,i_h}}{n}$, i.e. the user's average rate, and $n$ is the number of items that the active user has rated.

## 3.3 Item-based Prediction Algorithms Description

We refer to item-based prediction algorithms as the algorithms that are based on item's average rating and an adjustment to it, as given by equation 9.

$$prediction = item\_average + adjustment \quad (9)$$

*Adjustment* is most often a weighted sum that integrates user-based or item-based similarity measures. Since prediction arises as the sum of the two, improvements can be considered in both operators. Next, we suggest an item-based collaborative filtering prediction algorithm based on explicit ratings and also an item-based collaborative filtering prediction algorithm based on implicit ratings. In both cases we use the item-based similarity algorithms described in the earlier section.

### *3.3.1 Item-based with Explicit Ratings*

Item-based collaborative filtering prediction algorithm comes up as the sum of the active item's average rating, regarding the whole set of users that have rated it, and an adjustment. The adjustment is a weighted sum of the ratings that the active user has given to other items and their similarity with the active item. This algorithm can be seen as a reversed user-based collaborative filtering prediction algorithm and is given in equation 10.

$$CF_{IB-ER} = p_{u_a,i_a} = \overline{r}_{i_a} + \frac{\sum_{h=1}^{n'} \mu_{a,h}(r_{u_a,i_h} - \overline{r}_{u_a})}{\sum_{h=1}^{n'} |\mu_{a,h}|} \quad (10)$$

where $n'$ the number of items that user $u_a$ have rated and $\overline{r}_{i_a} = \frac{\sum_{h=1}^{n} r_{u_h,i_a}}{n}$, i.e. the item's mean rate, where $n$ is the number of users that have rated the active item.

### *3.3.2 Item-based with Implicit Ratings*

Instead of using the item-based explicit ratings similarity $\mu$, we can use the item-based implicit ratings similarity $\nu$ in order to calculate the similarity between the active item and the other items. The prediction algorithm is given by equation 11.

$$CF_{IB-IR} = p_{u_a,i_a} = \overline{r}_{i_a} + \frac{\sum_{h=1}^{n'} \nu_{a,h}(r_{u_a,i_h} - \overline{r}_{u_a})}{\sum_{h=1}^{n'} |\nu_{a,h}|} \quad (11)$$

where $n'$ is the number of items that user $u_a$ have rated and $\overline{r}_{i_a} = \frac{\sum_{h=1}^{n} r_{u_h,i_a}}{n}$, i.e. the item's average rating, and $n$ is the number of users that have rated the active item.

# 4. EXPERIMENTAL EVALUATION AND RESULTS

## 4.1 Data Set

Our experimental data comes from our movie recommendation system, named MRS (http://marouli.csd.uoc.gr:8989/web/rs).. The MRS database consists of 2068 ratings provided by 114 users to 641 movies, which belong to at least 1 of 21 categories. Therefore the lowest level of sparsity for our tests is defined by $\frac{114 \times 641 - 2068}{114 \times 641} \simeq 0.9717$. For our tests we run all the prediction algorithms over a pre-selected 300-item set extracted randomly by the set of 2068 actual ratings. We encourage interested readers to visit the web site of the system and obtain a more detailed view.

## 4.2 Metrics

There are two key dimensions on which the quality of a prediction algorithm can be measured, namely *coverage* and *accuracy*.

### *4.2.1 Coverage*

Coverage is a measure of the percentage of items for which a recommendation system can provide predictions. A basic coverage metric is the percentage of items for which predictions are available. Coverage can be reduced by defining small neighborhood sizes or by sampling users to calculate predictions. All experimental results demonstrated in this paper had coverage slightly less than perfect (99% in our experiments) for a high level of sparsity. A prediction is impossible to be computed in case that very few people rated an item or in case that the active user has zero correlations with other users.

### *4.2.2 Accuracy*

Several metrics have been proposed for assessing the accuracy of a collaborative filtering system. They are divided into two main

categories: *statistical accuracy metrics* and *decision-support accuracy metrics*.

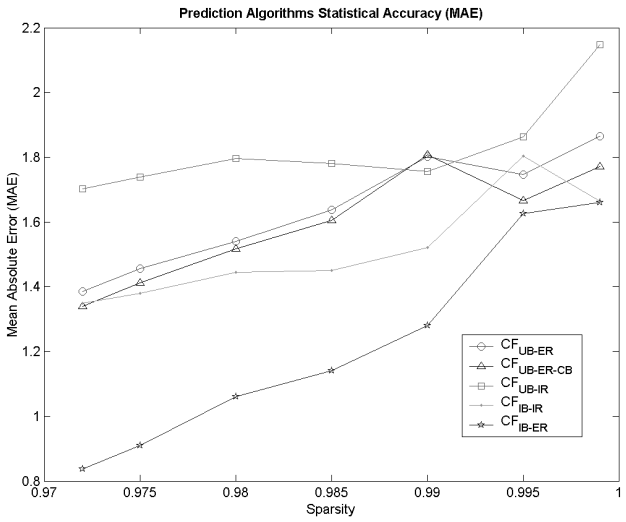
### *4.2.2.1 Statistical Accuracy Metrics*

Statistical accuracy metrics evaluate the accuracy of a filtering system by comparing the numerical prediction values against user ratings for the items that have both predictions and ratings. Some of them frequently used are *Mean Absolute Error (MAE)*, *Root Mean Squared Error (RMSE)* and *correlation* between ratings and predictions [10]. All of the above metrics were computed on result data and generally provided the same conclusions.

For our statistical accuracy measure, we use Mean Absolute Error (MAE). MAE is a measure of the deviation of predicted ratings from their actual ratings. Formally, if we define $n$ as the number of actual ratings in an item set, then MAE is defined as the average absolute difference between the $n$ pairs $< p_h, r_h >$ of predicted ratings $p_h$ and the actual ratings $r_h$ and equals to

$$MAE = \frac{\sum_{h=1}^{n} |p_h - r_h|}{n} \qquad (12)$$

The lower the MAE, the more accurate the predictions are permitting to provide better recommendations. We calculated the MAE for the different prediction algorithms and for different levels of *Sparsity*. The results are shown in figure 2.




| | | Sparsity Levels | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **0.972** | **0.975** | **0.98** | **0.985** | **0.99** | **0.995** | **0.999** |
| **Prediction Algorithms** | **$CF_{UB-ER}$** | 1.385 | 1.457 | 1.541 | 1.637 | 1.801 | 1.746 | 1.865 |
| | **$CF_{UB-ER-CB}$** | 1.34 | 1.412 | 1.518 | 1.606 | 1.807 | 1.667 | 1.771 |
| | **$CF_{UB-IR}$** | 1.703 | 1.739 | 1.796 | 1.781 | 1.755 | 1.863 | 2.147 |
| | **$CF_{IB-ER}$** | 0.838 | 0.91 | 1.06 | 1.14 | 1.28 | 1.626 | 1.66 |
| | **$CF_{IB-IR}$** | 1.35 | 1.38 | 1.445 | 1.45 | 1.521 | 1.804 | 1.665 |
| | **Random** | 3.166 | 3.515 | 3.414 | 3.024 | 3.256 | 3.174 | 3.398 |

**Figure 2**. Prediction Algorithms Statistical Accuracy

As far as statistical accuracy is concerned, we reach the following outcomes about the performance of the prediction algorithms.

- Item-based prediction algorithms perform better than user-based algorithms
- Implicit rating based algorithms (in the sense that they have been defined for these tests) perform much worse than explicit rating based algorithms
- Our item-based algorithm, based on explicit ratings ($CF_{IB-ER}$) seems to be very sensitive to sparsity levels. As sparsity reduces, the MAE of the algorithm goes down, which means that accuracy is increased. $CF_{IB-ER}$ performs as much as 39,5% better than classic Collaborative Filtering prediction algorithm $CF_{UB-ER}$ when sparsity level is 97,2%
- $CF_{UB-ER-CB}$ increases the accuracy precision of $CF_{UB-ER,}$ as it calculates the user average based only on the subset of items that belong to the same categories as the active item. However, this increment is insignificant taking into consideration the extra computation needed to include category information.

Experimental results indicate that item-based algorithms provide more accurate recommendations than user-based algorithms. In particular, our item-based algorithm based on explicit ratings, behaves much better as sparsity increases in comparison to all other algorithms presented. Our agent provides predictions of rating with an average MAE lower than 1 (i.e. 0.838). In our system, ratings range from 1 to 10, while in other common systems (GroupLens, EachMovie dataset) ratings range from 1 to 5. In order to obtain a clear comparative view of our MAE results, one needs to divide the results with a factor of 2. This in the best case leads to 0,419 for MAE, which is especially satisfactory for providing high-quality recommendations.

### *4.2.2.2 Decision-support Accuracy Metrics*

Decision-support accuracy metrics evaluate how effectively predictions help a user select high-quality items. Some of them frequently used are *reversal rate*, *weighted errors*, *Precision-Recall Curve (PRC) sensitivity* and *Receiver Operating Characteristic (ROC) sensitivity* [13]. They are based on the observation that, for many users, filtering is a binary process. Consequently, prediction algorithms can be treated as a filtering procedure, which distinguishes "good" items from "bad" items.

For our decision support accuracy measure, we use the ROC sensitivity. ROC sensitivity is a measure of the diagnostic power of a filtering system. Operationally, it is the area under the receiver operating characteristic (ROC) curve-a curve that plots the sensitivity and the 1-specificity of the test. Sensitivity refers to the probability of a randomly selected "good" item being accepted by the filter. Specificity is the probability of a randomly selected "bad" item being rejected by the filter. The ROC curve plots sensitivity (from 0 to 1) and 1 – specificity (from 0 to 1), obtaining a set of points by varying the quality threshold. The ROC sensitivity range between 0 to 1, where 0.5 is random and 1 is perfect.

If we denote the predicted rate as $PR$, the actual rate as $AR$ and the quality threshold as $QT$, then the following possible cases are defined by the filter for one item

- True Positive (TP) when $PR \geq QT \wedge AR \geq QT$
- False Positive (FP) when $PR \geq QT \wedge AR < QT$

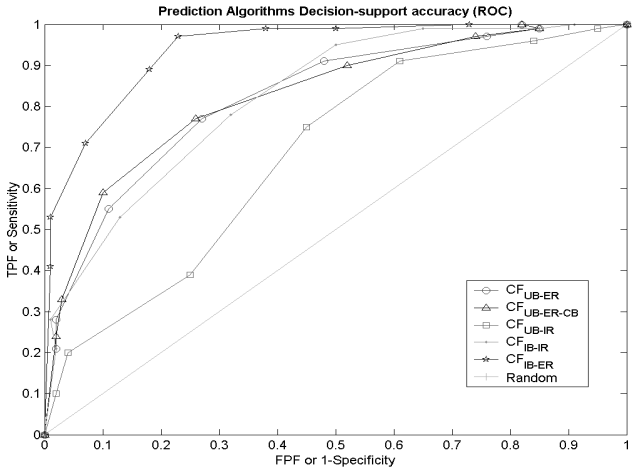
- True Negative (TN) when $PR < QT \wedge AR < QT$
- False Negative (FN) when $PR < QT \wedge AR \geq QT$

For a set of items sensitivity is defined as the True Positive Fraction (TPF) and the 1-specificity as the False Positive Fraction (FPF), where

$sensitivity = TPF = \frac{tp}{tp + fn}$, where $tp$, $fn$ is the number of true positive and false negative occurrences over the set of items respectively.

$1 - specificity = FPF = \frac{fp}{fp + tn}$, where $tn$, $fp$ is the number of true negative and false positive occurrences over the set of items respectively.

We calculated the ROC curve for the different prediction algorithms and for quality thresholds ranging between 1 and 9, while the sparsity level was equal to 0,972. We use the symbolism ROC-threshold to represent the point on the ROC curve for the specific quality threshold value. The results are shown in figure 3.




| | | | Quality Threshold | | | |
|---|---|---|---|---|---|---|
| | | | **ROC-6** | **ROC-7** | **ROC-8** | **ROC-9** |
| **TPF or Sensitivity** | **Prediction Algorithms** | **$CF_{UB-ER}$** | 0.77 | 0.55 | 0.28 | 0.21 |
| | | **$CF_{UB-ER-CB}$** | 0.77 | 0.59 | 0.33 | 0.24 |
| | | **$CF_{UB-IR}$** | 0.75 | 0.39 | 0.2 | 0.1 |
| | | **$CF_{IB-ER}$** | 0.89 | 0.71 | 0.53 | 0.41 |
| | | **$CF_{IB-IR}$** | 0.78 | 0.53 | 0.28 | 0.21 |

**Figure 3.** Prediction Algorithms Decision support accuracy (ROC sensitivity)

The following remarks can be made about the performance of the prediction algorithms as far as decision-support accuracy is concerned.

- Item-based prediction algorithms perform better than user-based algorithms
- Implicit rating based algorithms (In the sense that they have been defined for these tests), perform much worse than explicit rating based algorithms
- Our $CF_{IB-ER}$ performs 95% better than classic collaborative filtering $CF_{UB-ER}$ for ROC-9, 89% better for ROC-8 and 29% better for ROC-7. This means that if we define as "good" items the ones that have average rating more than 9 or 8 or 7 repsectively and as "bad" items the ones that have average rating less than 9 or 8 or 7 respectively, then $CF_{IB-ER}$ predicts and therefore recommends items with 95% or 89% or 29% respectively more accuracy than classic collaborative filtering $CF_{UB-ER}$.
- To obtain a clear view of the overall performance of each algorithm one need to calculate the area under the ROC curve. It is clear from the figure 3 that our $CF_{IB-ER}$ performs much better than any other algorithm examined.

# 5. EXTENSIONS

In this section, we present preliminary theoretical approach of future research directions. Our first objective is to bring recommendation algorithms on the Web that can be proved accurate and require soft computations. Subsequently, we try to investigate how similarity measures can be used in order to identify online, virtual, self-organized communities of users that share similar interests.

## 5.1 Incremental calculation of similarity measures

Recommendation algorithms are based on similarity measures, as described in the previous section. However, these measures require expensive computations. Each time a recommendation is requested by an active user, the algorithm needs to calculate the similarity between the active user and all other users, based on their co-rated items, so as to pick the ones with similar behavior. Next, the algorithm recommends to the active user, items that are highly rated by the already selected most similar users.

In this extension, we aim to achieve incremental update of the user similarities, and as a result to be able to recommend items in short time. In order to achieve this, we propose a caching scheme that permits to update the user similarities after each single rating is provided or an older one is updated. At the time that a recommendation is requested, cached information permits to provide recommended items to users *on the fly*, as only trivial computations are required. Our approach maintains a high–level of quality as recommendation algorithms employ the total of the information that is logged into the system and not just a part of it. Our method permits highly scalable and accurate recommendation algorithms to effectively be brought on the Web.

The similarity between user $u_x$ and $u_y$ for the subset of items they have co-rated, is given by equation 1, where $n'$ is the number of the co-rated items, $r_{u_x,i_h}$ is user's $u_x$ rating for item $i_h$ and $\overline{r_{u_x}}$, $\overline{r_{u_y}}$ are the averages of items rated by $u_x$, $u_y$ respectively. Whenever a user $u_x$, rates an item or updates the value of an already submitted rating, his user model is updated and thus all similarity measures between him and the other users need to be re-calculated. The new similarity between a user $u_x$ and a user $u_y$ is given by the equation below:

$$sim(u_x,u_y)=\frac{\sum_{h=1}^{n''}(r_{u_x,i_h}-\overline{r_{u_x}'})(r_{u_y,i_h}-\overline{r_{u_y}})}{\sqrt{\sum_{i=1}^{n''}(r_{u_x,i_h}-\overline{r_{u_x}'})^2}\sqrt{\sum_{i=1}^{n''}(r_{u_y,i_h}-\overline{r_{u_y}})^2}},$$

where $\overline{r_{u_x}'}$ is the new average of user $u_x$ and $n''$ is the new number of co-rated items. If a new item rated by the active user is also rated by user $u_y$, then the number of co-rated items is incremented by one, so $n''=n'+1$. In the case that user $u_y$ has not rated this item or a rating update occurs, the number of co-rated items remains the same, so $n''=n'$. Our objective is to express the new similarity between two users as a sum of the old similarity and an *increment*. To smooth on the progress of this task we adopt the following notation for the old Pearson Correlation similarity:

$$A=\frac{B}{\sqrt{C}\sqrt{D}}, \text{ where } A=sim(u_x,u_y),$$

$$B=\sum_{h=1}^{n'}(r_{u_x,i_h}-\overline{r_{u_x}})(r_{u_y,i_h}-\overline{r_{u_y}}),$$

$$C=\sum_{i=1}^{n'}(r_{u_x,i_h}-\overline{r_{u_x}})^2 \text{ and } D=\sum_{i=1}^{n'}(r_{u_y,i_h}-\overline{r_{u_y}})^2$$

and the following notation for the new Pearson Correlation similarity:

$$B'=\sum_{h=1}^{n''}(r_{u_x,i_h}-\overline{r_{u_x}'})(r_{u_y,i_h}-\overline{r_{u_y}})=B+e$$

$$C'=\sum_{i=1}^{n''}(r_{u_x,i_h}-\overline{r_{u_x}'})^2=C+f$$

$$D'=\sum_{i=1}^{n''}(r_{u_y,i_h}-\overline{r_{u_y}'})^2=D+g$$

where $e$, $f$ and $g$ are the increments that need to be added to each factor after the submission of a new or the update of an already existing rating. Actually, we split the similarity measure into three factors, independently calculate the new values of each factor and then combine their updated values so as to yield the value of the new similarity. Combination of $B'$, $C'$ and $D'$ values yields the value of the new similarity, which equals to:

$$A'=\frac{B'}{\sqrt{C'}\sqrt{D'}}\Leftrightarrow A'=\frac{B+e}{\sqrt{C+f}\sqrt{D+g}}$$

There are two cases that we need to examine. The first occurs when the active user submits a new rating and the second when the active user updates an already submitted rating. Furthermore, for each one of these cases we distinguish between two instances. As similarities are defined between the active user and a user $u_y$, the first instance refers to the case that the active item is co-rated by user $u_y$ and the second when it is not. We finally conclude on four cases that need to be examined.

The factors that may be affected during these computations are the average of the active user, the average of the user $u_y$ and the total number of their co-rated items. Taking these factors into consideration we calculate the factors $e$, $f$, $g$. Results are displayed on Table 1 of the Appendix.

In order to achieve our method we propose a caching scheme that permits to calculate the incremental factors with trivial calculations. We need to cache the values of $B$, $C$ and $D$ for all couple of users. We also need to cache the average rating of each user and the number of items that he/she has rated. Part of the cached information needs to be updated every time a user submits a new rating or updates an existing one. In Table 2 of the Appendix we explain how incremental factors $e$, $f$, $g$ are calculated.

## 5.2 Formation of Communities

In this paragraph, we present ideas about how similarity measures can be employed, in order to identify self-organized, online, virtual communities and sub-communities in an information system.

### *5.2.1 The "Community Coherence Grade"*

The similarity measures described earlier are detecting relationships between users and they can consequently be functional in the formation, description and behavioral explanation of communities [14]. In order to examine the tightness or looseness of these communities we introduce the "Community Coherence Grade" (CCG), which is defined as the average correlation between user $u_a$ and the rest members $m'$ of the community. A user $u_h$ is considered as a member of user's $u_a$ community if the correlation between them is greater than a threshold. CCG is given by the equation 13.

$$CCG_{u_a}=\frac{\sum_{h=1}^{m'}sim(u_a,u_h)}{m'} \quad (13)$$

In our approach, communities are self-organized. This means that there is not any intentional registration to a community, and there is not any kind of external moderation. Membership to a community or not is decided without any human intervention and is merely based on user's rating behavior. These communities are also considered virtual as they exist just from the current user's perspective and dynamic as any user's single rating can result in an adjustment of the community's state.

### *5.2.2 Detecting Sub-communities*

We also describe a filtering algorithm that facilitates the detection of users that are relevant according to just a part of their profile and consider them as sub-community. We apply criteria that typify the degree of preference to specific categories in a five-level scale (0-20, 20-40, 40-60, 60-80, 80-100 %). For example

we can detect the sub-community, which likes Comedy (e.g. 80-100%) but dislikes Drama (e.g. 0-20%). The algorithm calculates the users that satisfy all the criteria by applying a Boolean AND operation to the criteria specified by the filtering procedure. The algorithm steps are shown below.

1. Find the categories for which criteria have been specified
2. For each category find which one out of the five preferences has been specified
3. Find the users of the system that have rated at least once
4. For each user decide whether he satisfies the criteria
   a. For each category calculate the positive percentage that corresponds to the user.
   b. If the percentage satisfies the criteria then continue with the next category.
   c. If a user satisfies the criteria of all categories then is returned by the algorithm
5. Continue with step 4 until there are no remaining users

The complexity of the algorithm is linear on the number of categories on which criteria have been specified and the number of users on the system. Detection of sub-communities is valuable when recommending items to users with eclectic state.

## 6. CONCLUSIONS

Recommendation systems are effectively used to filter out excess information and to provide personalized services to users by employing sophisticated, well though-out prediction algorithms. We described how explicit ratings could be utilized in order to implicitly provide user's preference to specific categories. We designed, implemented and tested a number of prediction algorithms based on either user or item similarity and we thoroughly evaluated their performance in relation to statistical and decision-support accuracy. Our analysis shows that item-based are much better than user-based predictions. Category-boosted predictions can lead to slightly better predictions when combined with explicit ratings, while implicit ratings (in the sense that we have defined them here) perform much worse than explicit ratings.

In the sequence, we presented preliminary theoretical extensions of our future research direction. We present a methodology that permits to efficiently bring recommendation algorithms on Web. The novelty of our approach yields in incremental calculation of the similarity measures between users, contrary to dimensionality reduction techniques previously adopted. Although there are no experimental results to depict the efficiency of our work at the moment, we expect that our approach will do better than the existing recommendation algorithms in both performance and accuracy aspects. Finally, we drew the direction to another interesting aspect of similarity measures, as they can be effectively used to identify self-organized, virtual, online communities and we introduced the CCG term to define the relevance between community members.

## 7. REFERENCES


1. Paul Resnick and Hal R. Varian. Recommender Systems (introduction to special section). Communications of the ACM, 40(3), March 1997
2. R.B. Allen, "User Models: Theory, method and Practice". International Journal of Man-Machine Studies, 1990
3. M. Balabanovic and Y. Sholam. "Combining Content-Based and Collaborative Recommendation". Communications of the ACM, 40 (3), March 1997.
4. Ning Zhong, Jiming Liu, Yiyu Yao, "Web Intelligence: Algorithmic Aspects of Web Intelligent Systems", eds., Springer-Verlag, 2003, pp. 323-345
5. Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl, "Explaining Collaborative Filtering Recommendations". Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 2000).
6. Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, "Methods and Metrics for Cold-Start Recommendations". Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002).
7. Prem Melville, Raymond J. Mooney and Ramadass Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations". Proceedings of the National Conference of the American Association Artificial Intelligence (AAAI-2002)
8. Karl Pearson, "Mathematical contribution to the theory of evolution: VII, on the correlation of characters not quantitatively measurable". Phil. Trans. R. Soc. Lond. A, 195,1-47, 1900.
9. John S. Breese, David Heckerman and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)
10. Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers and John Riedl, "An Algorithmic Framework for Performing Collaborative Filtering". Proceedings of the 22nd ACM SIGIR conference on Research and development in information retrieval, (SIGIR 1999)
11. Jon Kleinberg, Christos H. Papadimitriou, Prabhakar Raghavan, "On the Value of Private Information". Proceedings of 8th Conference on Theoretical Aspects of Reasoning about Knowledge (TARK VIII), 2001
12. David Nichols, "Implicit Rating and Filtering". Proceedings of 5th DELOS Workshop on Filtering and Collaborative Filtering, 1997, pp. 31-36
13. Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller and John Riedl, "Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System". Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 1998).
14. Nick Papadopoulos and Dimitris Plexousakis, "The Role of Semantic Relevance in Dynamic User Community Management and the Formulation of Recommendations". Proceedings of 14th International Conference on Advanced Information Systems Engineering, (CAiSE 2002).
15. Chun Zeng, Chun-Xiao Xing,Li-Zhu Zhou. "Similarity measure and instance selection for collaborative filtering". Proceedings of 12th international conference on World Wide Web (WWW 2003).
16. Thomas Hofmann. "Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis". Proceedings of the 26th ACM SIGIR conference on Research and development in information retrieval (SIGIR 2003)

# Appendix

**Table 1.** Summary of incremental factors that need to be calculated after each rating so as to achieve incremental update of similarity measure

| | | |
|---|---|---|
| Submission of a new rating $r_{u_a,i_a}$ for an item $i_a$ by the active user $u_a$ | | |
| In case that item $i_a$ has been rated by user $u_y$ | e | $e = (r_{u_a,i_a} - \overline{r_{u_a}'})(r_{u_y,i_a} - \overline{r_{u_y}}) - \sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$ |
| | f | $f = (r_{u_a,i_a} - \overline{r_{u_a}'})^2 + \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}})$ |
| | g | $g = (r_{u_y,i_a} - \overline{r_{u_y}})^2$ |
| In case that item $i_a$ has not been rated by user $u_y$ | e | $e = -\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$ |
| | f | $f = \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}})$ |
| | g | $g = 0$ |
| Update of an existing rating $r_{u_a,i_a}$ for an item $i_a$ by the active user $u_a$ | | |
| In case that item $i_a$ has been rated by user $u_y$ | e | $e = dr_{u_a,i_a}(r_{u_y,i_a} - \overline{r_{u_y}}) - \sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$ |
| | f | $f = dr_{u_a,i_a}{}^2 + 2dr_{u_a,i_a}(r_{u_a,i_a} - \overline{r_{u_a}'}) + \sum_{h=1}^{n'} d\overline{r_{u_a}'}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}'}(r_{u_a,i_h} - \overline{r_{u_a}})$ |
| | g | $g = 0$ |
| In case that item $i_a$ has not been rated by user $u_y$ | e | $e = -\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$ |
| | f | $f = \sum_{h=1}^{n'} d\overline{r_{u_a}'}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}'}(r_{u_a,i_h} - \overline{r_{u_a}})$ |
| | g | $g = 0$ |

**Table 2.** Calculation of operands that appear in incremental factors

| Operand | Calculation |
|---|---|
| $m$ | Cached Information (The number of the items that a user has rated is cached) |
| $\overline{r_{u_a}}$ | Cached information (Average ratings of all users are cached) |
| $\overline{r_{u_a}}'$ | The new average of the active user is calculated incrementally based on the cached values of the previous average and the number of already rated items. If a new rating is submitted the new average equals to $\overline{r_{u_a}}' = \frac{r_{u_a,i_a}}{m+1} + \frac{m}{m+1}\overline{r_{u_a}}$. if an update of an existing rating is submitted then the new average equals to $\overline{r_{u_a}}' = \frac{dr_{u_a,i_h}}{m} + \overline{r_{u_a}}$. In both equations, $m$ is the number of items that the active user had rated before his new rating to $i_a$. |
| $\overline{r_{u_y}}$ | Cached information (Average ratings of all users are cached) |
| $r_{u_a,i_a}$ | The rating of the active user to the active item is provided through the system's interface |
| $d\overline{r_{u_a}}$ | The difference of user's previous and current average rating is calculated as described earlier based on the cached value of $\overline{r_{u_a}}$, $m$ and the new or updated rating $r_{u_a,i_a}$ |
| $r_{u_y,i_a}$ | The rating of the user $u_y$ to the item $i_a$ is found by querying the database |
| $\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}})$ | The sum of all ratings of active user $u_a$ is found using an aggregation query |