Weight-and-Relate: A Group Recommender Algorithm and Prototype

Kevin Li

Research Paper submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master's Degree in Computer Science

Ottawa-Carleton Institute for Computer Science
Computer Science Faculty
University of Ottawa

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LEGEND

Acronym	Description
WaR	Weight-and-Relate
GUI	Graphical User Interface
API	Application Programming Interface
MAE	Mean Absolute Error

ABSTRACT

In electronic commerce, recommender system is a powerful tool that assists consumers in locating and determining desired items. This paper introduces the weight-and-relate recommendation algorithm that targets both individual and group environments, while incorporating dynamic social interaction evolutions. The weight-and- relate algorithm is based primarily on the collaborative filtering concept to predict ratings, merging profiles, and merging recommendations in order to generate appropriate recommendations. A weight-and-relate prototype was developed to assists in experimenting with the algorithm by providing environment adjustment capabilities. The algorithm has been evaluated with the objective of prediction accuracy, utilizing the mean absolute error value as the metric, and experimental results indicated the algorithm is acceptable. Comparisons to previous research are discussed and future directions are outlined.

ACKNOWLEDGEMENTS

Thanks for Professor Thomas Tran for assisting in completing the research project, and to the GroupLens Research Project for releasing rating datasets.

Introduction

Recommender system is a powerful tool that contributes to the success of any electronic commerce (E-Commerce) business. By assisting the consumers in identifying items that satisfies their individual interests, recommender systems can transform potential seekers into buyers. Sophisticated recommender systems often employ multiple policies to present recommendations that most resemble the consumer's interests.

The majority of recommender systems employed to-date target single-consumer settings, where the resulting recommendations are geared towards the interests of an individual. Situations where multiple consumers requiring a single set of recommendations are often overlooked. For example, a group of people going to a movie together may need recommendations on a movie that everyone would enjoy. As a result, the concept of group recommendation system emerged to support providing recommendations to groups of consumers instead of individuals. Group recommender systems are usually practical for domains in which multiple consumers can enjoy together, such as music or literatures, and impractical for requirements geared towards individuals. Compared to individual recommender systems, the complexity of the group system amplifies with the involvement of multiple users.

As with all other recommender systems, the primary challenge for group recommender system is the ability to produce accurate recommendations. Although many research efforts have been placed on achieving accurate individual recommendations [2, 3, 7, 27, 28, 31, 32, 34, 35], the fundamental differences with group recommender systems necessitate an entire new spectrum of research due to its inherited multi-user nature. For the most part, group recommender systems need to produce recommendations that satisfy the group as one while considering the differences, affections, and impacts of each member of the group.

The goal of this research is to design a recommendation algorithm, named weight-and-relate (WaR), which will satisfy the requirement of producing recommendations for a group of users, while addressing the interaction and influence factors between individuals in the group. An experimental prototype is also to be developed that implements the WaR algorithm to demonstrate the capabilities of the algorithm.

Chapter 1. Background and Related Work

Section 1.A. Recommender Systems

As the Internet introduced a global information society with growing amount of information, recommender systems emerged to assist users in locating and identifying specific pieces of information. Recommender systems are applied to multiple domains, as previous research has studied the taxonomy of recommender agents on the Internet, classifying the different recommender systems in different domains and categorizing each of their properties [2]. Another study looked into the recommender systems provided by different E-Commerce websites, discussed the techniques that are used by each recommender system, and listed some research challenges in the field of recommender systems [3]. In addition, there have been papers that studied the impacts that recommender systems have on end-user's opinions [4].

Over the years, various algorithms were developed to meet the demands of recommender systems. Most algorithms employ the concepts of content-filtering, collaborative-filtering, context- filtering, or a combination of all three concepts [1, 5, 6, 25, 28]. As there are advantages and drawbacks in each concept, many papers had proposed hybrid approaches hoping to inherit the benefits and fulfill the weaknesses, including combining collaborative filtering and knowledge-based approaches [1, 25, 28], improving collaborative-filtering by applying various algorithmic frameworks [5, 27], and personalizing recommendations by learning user profiles [6, 29].

The idea of content-filtering approach is to consult a knowledge base of product domains and identify products that most closely match the requirements. This approach strictly relies on the user-defined requirements to generate recommendations. Hence, two exact same requests made by two different users will result in the exact same set of recommendations. The content-filtering approach is usually suitable for situations where the features of the products serve a more important role.

Opposite to content-filtering, the idea of collaborative- filtering is to consult a knowledge base of user information, and identify users that have similar interests as the requesting user. Recommendations are then generated based on the preferences of the similar-minded counterparts. In this concept, difference users requesting for recommendations will likely receive different sets of recommendations even if their requirements are the exact same. This approach is suitable for products with similar features but with different properties such that different users would have different preferences.

Similar to collaborative-filtering, the idea of context- filtering is to not rely on user requirements, but instead consult a knowledge base of context information. Such information may include location, season, age, gender, race, etc. Recommendations are then generated based on matching the user contexts against the product contexts. This approach is suitable for products that are applicable for only certain contexts, such as a snowboard is usually more popular during the winter season [26].

The WaR algorithm described and implemented in this paper is based on the concept of collaborative-filtering; however, it is possible to operate under a hybrid approach and further incorporate both content and context filtering concepts.

Section 1.B. Group Recommender Systems

Despite the overwhelming employment of individual recommender systems in E-commerce, there are also a small number of group recommender systems available. However, the systems usually target specific audience groups, specific product domains, or specific environments.

In most existing group recommender systems, the focus has been placed on several particular product domains, such as the music domain with MusicFX [10, 13, 14], the tourists domain with Intrigue [17] and Travel Decision Forum [8, 9], and collaborative web browsing with Let's Browse [15] and I-Spy [21, 22]. One of the more popular product domains for group recommender systems is movies. PolyLens is a prototype implementation of group recommender system, based on the collaborative-filtering concept, which recommends movies for a group of users [11]. The prototype is built on top of MovieLens [12], which itself is a recommender system that recommends movies to individuals. PolyLens places emphasis on the idea of "groups", namely the nature of a group, group formation and evolution, personal privacy within group settings, generate recommendations for the groups, and interfaces to support group recommender systems. To generate recommendations, PolyLens merged and sorted all members' recommendation sets based on least happy member. Nine-months of field trials were conducted for the PolyLens system with around 800 experimental users, 7000 of group recommendation requests made, and over 110000 movies recommended. Several important social interaction findings were discovered during the field trials, including the need of better recommendation algorithm [11].

Chapter 2. Weight-And-Relate Algorithm

The intention of this research is to design a recommendation algorithm that is capable to be utilized for both individual and group recommender systems, while demonstrating its capabilities with a prototype implementation. The WaR is primarily based on the concept of collaborative filtering, while learning social interaction evolutions to accomplish group recommendations.

As with other collaborative filtering techniques, this algorithm requires that a previously populated dataset, containing several necessary datasets, is available to use. The first dataset is the set of items, which are things that may be recommended to the end user. The second dataset is the set of users, each representing a real-life individual. The last dataset is each user's rating values for the items, based on each user's truthful preferences. A rating value is a numeric value ranging from 0.5 to 5 with 0.5 increments, where 0.5 being lowest preference and 5 being highest preference. A study by Cosley et al. [4] shows this rating value scheme is the most favourable for end users. In order to be as precise as possible, the WaR continues to maintain all additional decimal values when generating recommendations and only strips down to a single decimal place when presenting recommendations to end users.

Each user is modelled with a username and optionally personal information. Although personal information is not utilized in the WaR algorithm, it may be beneficial when incorporating other recommendation techniques, such as context-filtering. In addition, each user has a user profile that maintains a collection of item-to-rating-value mappings, which essentially contains all rating values for the items that the user has previously rated.

The WaR algorithm is modularized into three sub-algorithms: prediction algorithm, individual algorithm, and group algorithm. Each sub-algorithm contains different functionalities that, when combined, will produce the desired recommendations. This section explains the three sub-algorithms and discusses the implementation and interaction details for each.

Section 2.A. Prediction Algorithm

The function of the prediction algorithm is to output a prediction for a particular item for a particular profile. Based on the input profile and target item, this algorithm generates a prediction value for the item, ranging from 0.5 to 5.0.

The input profile is required to contain all applicable item-to-rating-value mappings that are pre-defined (by other algorithms). Upon receiving the request, the prediction algorithm first retrieves the user profiles for all users in the dataset. For each retrieved profile, a difference factor against the input profile is calculated by taking the average of the ratings differences between the two profiles. Specifically, the difference factor *DF* is calculated by:

Let Pa be the input profile.

Let Pb be the retrieved profile.

Let $I = \{I_1, I_2, I_3 ... I_N\}$ be the set of common items between Pa and Pb, where C = number of common items.

Let $f(i, P)$ be the rating of item i in profile P.

Then the difference factor *DF* between profiles Pa and Pb is:

$$DF = \frac{\sum_{x=1}^{N} | f(Ix, Pa) - f(Ix, Pb) |}{C}$$

Equation 1 Difference Factor between Two Profiles

An important aspect in calculating the difference factor is to define the *similarity-threshold* value. For any two profiles, their similarity is defined by the number of common items between the two profiles, or items that both profiles have previously rated. This similarity value must exceed the *similarity-threshold* value for the retrieved profile to be considered as a legitimate profile to use in next step (described below). In essence, the *similarity-threshold* value acts as a gate to eliminate profiles with low difference factors simply because it has minimal common items when compared against the target profile. Determining the appropriate *similarity-threshold* value is achieved through experimentation and explained in Section 5.A.

Once the difference factors for all legitimate profiles have been determined, the algorithm proceeds to collect each profile's rating value for the target item, if it exists, in the order of lowest to highest difference factor. Since lower difference factor value implies greater resemblance to target user profile, the difference factor values are inverted to formulate a profile weight value for each profile. The algorithm collects up to a *k* rating values, where *k* is defined by a *nearest-k-threshold* value. Thus, to calculate the resulting prediction value for the target item:

Let P be the set of profiles that has been selected, where k = number of profiles.

Let Rp be the rating value of profile p for the target item.

Let Dp be the difference factor of profile p.

Let $Dtotal$ be the total difference factors of all k profiles.

Let Wp be the profile weight of profile p, where

$$Wp = 1 - \frac{Dp}{Dtotal}$$

Equation 2 Individual Profile Weight Value

Let $Wtotal$ be the total profile weights of all k profiles.

Then the prediction rating value $Rpred$ for the target item is

$$Rpred = \sum_{p=1}^{k} (Rp \times \frac{Wp}{Wtotal})$$

Equation 3 Prediction Rating Value

The resulting value is the output prediction value for the target item, as generated by the prediction algorithm. In essence, the *nearest-k-threshold* value is used to avoid collecting rating value data from excessive amount of profiles, as there are often cases where too much data is unnecessary and may end up causing fluctuation in the result data. Determining the appropriate *nearest-k-threshold* value is achieved through experimentation and explained in Section 5.A.

Section 2.B. Individual Algorithm

The function of the individual algorithm is to output a list a recommendations for a particular user. Based on a target user, the algorithm returns a list of items that it has predicted the target user would most prefer, determined based on the predicted rating values for each item (the rating values are directly proportional to the user's preference for the items). The detailed implementation of the algorithm is as follows:

Filtering Stage: upon receiving the recommendation request, the algorithm first retrieves the user profile for the target user. Then, a list of candidate items to recommend is generated by filtering through all items in the dataset. In this current stage, the individual algorithm does not apply any filtering mechanisms and the list of candidate items is essentially the list of all items in the dataset.

Item Evaluation Stage: once the user profile and candidate items has been determined, the algorithm then calculates the rating values by utilizing the prediction algorithm iteratively, passing the user profile and one candidate item as the inputs in each iteration. This step is repeated for all candidate items in order to generate predicted rating values for all items. Once the rating values for all candidate items are calculated, the items are then sorted by rating values (from highest to lowest), and the top items are returned as recommendations.

Section 2.C. Group Algorithm

The function of the group algorithm is to output a list of recommendations for a group. Based on the users in the group, the algorithm returns a list of items that it has predicated the target group would most prefer, determined based on the predicted rating values for each item (the rating values are directly proportional to the group's preference for the items). In addition, the algorithm considers and incorporates the social interaction factors when generating the recommendations. The detailed implementation of the algorithm is as follows:

User Evaluation Stage: upon receiving the recommendation request, the algorithm first determines each user's influence within that particular group. To achieve this, the algorithm requires that a user-relation-function to be maintained permanently. For each unique pair of users *i* and *j*, the user-relation-function defines the degree of influence, *Mij*, that the user *i* imposes on user *j*. The degree is represented by assigning a numerical value to both users, ranging from 0.0 to 2.0, where the two values sum to 2.0. If two users have equal influence on one another, both users will be assigned with a degree of 1.0. If one user has imposes more influence on the other user, the first user will be defined with a higher degree than the second user (while sum of the two values is still maintained at 2.0).

The algorithm utilizes the user-relation-function to determine a weight-coefficient value for each user in the group. The weight-coefficient values are calculated as follows:

Let U be the group of users, where N = number of users in the group.

Let u be an arbitrary user where $u \in U$.

Let Mij be the degree of influence that user i imposes on user j (Mij = 1.0 if $i = j$).

Then the weight coefficient Wu of user u is

$$Wu = \frac{\sum_{x=1}^{N} Mux}{N}$$

Equation 4 Weight Coefficient

Once the weight-coefficient values are determined, they are converted to a percentage-share value for each user. The percentage-share values are calculated by:

Let U be the group of users, where N = number of users

Let u be an arbitrary user where $u \in U$

Let Wu be the weight coefficient of user u

Then the percentage-share PSu of user u is

$$PSu = \frac{Wu}{\sum_{x=1}^{N} Wx}$$

Equation 5 User Percentage Share

The percentage-share values from each user in the group sum to 100%. The percentage-share values essentially indicate the amount of influence that the associated user has within that particular group, displayed as a percent format, where a higher percentage represents more influence.

Filtering Stage: similar to the individual recommendation algorithm, a list of candidate items is also generated based on all items in the dataset without any filtering mechanisms applied. At this point, the data for the group, the percentage-share values, and the list of candidate items are established, and the group algorithm can now generate recommendations for the group using these data as inputs. The algorithm employs two merger procedures to generate the recommendations for the group, called "Profile Merger" and "Recommendation Merger"

The Profile Merger procedure generates recommendations by merging all user profiles into a single profile, then employs a similar process as the individual recommendation algorithm but uses the merged profile instead. Therefore, the key processing is to produce a merged profile that is representative for the group of users.

Merge Stage: to merge all user profiles into a single profile is essentially merging the rating values for each item, and this occurs in two steps: first step determines if the item should be included, and second step calculates the merged rating value for the item.

To determine if an item should be included, a merge-percentage value is calculated by summing the percentage-share values of the users that has rated this particular item. The resulting merge-percentage value has to exceed a merge-threshold value for the item to be included in the merged profile. Specifically:

Let $I = \{i_1, i_2, .. i_M\}$ be the set of items, where M = number of items

Let i be an arbitrary item where $i \in I$

Then the rating merge percentage RMi of item i is

$$RMi = \sum_{u=1}^{N} f(u)$$

$$\text{where } f(u) = \begin{cases} PSu & \text{if user } u \text{ has rated item } i \\ 0 & \text{otherwise} \end{cases}$$

Equation 6 Item Rating Merge Percentage

In essence, the merge-threshold value prevents inclusion of items that has not been sufficiently rated by enough members in the group. A reasonable merge-threshold value depends on the nature and intent of the group. A low merge-threshold value could result in recommendations that tailor to only some users in the group and not others, whereas a high merge-threshold value would eliminate such recommendations. However, higher merge-threshold value would also imply that individual's rating values are more likely to be filtered during the merge phase, resulting in less accurate recommendations. As such, the algorithm does not enforce a fixed merge-threshold value; instead, the value can be dynamically modified by the end users.

After the rating merge percentages are determined, the group's rating values for the items are then calculated by adjusting and averaging each user's individual rating value based on percentage-shares and merge-percentage values:

Let $RVui$ be the rating value of user u for item i

Then the merged rating value Ri of item i is

$$Ri = \sum_{u=1}^{N} \frac{PSu}{RMi} \times RVui$$

Equation 7 Item Merged Rating Value

Item Evaluation Stage: after the merged rating values for all items are calculated, they are placed within an empty profile which is now the merged profile. The algorithm then leverages the individual algorithm to generate recommendations, but applies the merged profile as the input instead of an individual's profile. The resulting recommendations from the individual algorithm are then considered as the recommendations for the group.

In the group algorithm, the social interaction factors are incorporated through the use of user percentage share values. As the percentage share values represents the amount of influence each member has in the group, including them in profile merger calculations effectively gives higher priorities to users with higher percentage share values, or higher social standings within the group.

As an example of profile merger, consider the following scenario:

Number of users = 6

Merge Threshold = 40%

User weight coefficients and ratings:

User (***u***)	**Weight Coefficient** (***Wu)***	**Item A Rating** (***RVuA***)	**Item B Rating** (***RVuB***)
User 1	0.66	5	1
User 2	0.88	4	2
User 3	1.13	N/A	5
User 4	1.60	3	N/A
User 5	1.20	3	5
User 6	0.53	N/A	N/A
TOTAL	**6.00**		

Table 1 User Weight Coefficients and Ratings Example

Based on the above example, the following is calculated during ratings merge:

User Percentage Share values:

User (*u)*	**User Percentage Share (*PSu)***
User 1	0.66/6 = 11.00%
User 2	14.67%
User 3	18.83%
User 4	26.67%
User 5	20.00%
User 6	8.83%
TOTAL	**100%**

Table 2 User Percentage Share Example

Merge Percentage values:

Item *i*	**Merge Percentage (*RMi)***
Item A	11+14.67+26.67+20.0 = **72.34%**
Item B	**64.50%**

Table 3 Merge Percentage Example

Merged Ratings Values:

Item (*i)*	**Merged Rating Values (*Ri)***
Item A	(11/72.34 x 5) + (14.67/72.34 x 4) + (26.67/72.34 x 3) + (20/72.34 x 3) = **3.51**

Item B	**3.64**

Table 4 Merge Rating Values Example

In this example, the ratings values for both Item A and Item B are included in the merged profile, since the merge percentage values exceeds the merge threshold value. Furthermore, the ratings are calculated based on only the users that have ratings values against the items and the percentage shares of each of these users.

The Recommendation Merger procedure generates recommendations by first applying the individual recommendation algorithm for all users in the group, then merging the resulting recommended item sets into one set of items. This final set of items is then the recommended items for the group. Therefore, the key processing is to produce a merged recommendation set based on the multiple individual recommendation sets.

The concept of merging recommendation sets is very similar to merging profiles. Each profile contains a collection of item-to-rating-value mappings, and the recommended sets essentially contain a collection of item-to-rating-value mappings as well. Hence, the same process as merging profiles is applied to merging recommendations and hence is not further elaborated.

Once the final set of items is established with the rating values determined, the items are then sorted by rating values (from highest to lowest), and the top items are returned as recommendations.

Chapter 3. Weight-and-Relate Prototype

The intention of the WaR prototype is to demonstrate the capabilities and effectiveness of the WaR algorithm, in terms of generating recommendations in both individuals and groups environments. As such, the prototype is focused on providing features that assist in conducting experiments with the WaR algorithm; often times such features are uncommon in consumer-based recommender systems. On the other hand, the prototype is limited in other standard recommender system functionalities such as user management, security management, etc. as these are not essential in evaluating the algorithm itself. As an example, the prototype provides features to directly modify the user-relation-function values, while not providing any user identification mechanism.

The WaR prototype follows the traditional three-layer architecture, consisting of an Interface Layer, an Application Layer, and a Database Layer. The prototype is implemented using Java programming language with a MySQL server acting as the informational database.

Section 3.A. Interface Layer

The interface layer consists of a Graphical User Interface (GUI) component that supports interaction with the end users, written using the Java Swing programming language. The interface layer is strictly limited to user interaction capabilities and contains no knowledge of the WaR algorithm; instead, the requests are delegated to the application layer for processing. The GUI component is composed of four main screens: User screen, User Relation screen, Recommender screen, and Settings screen.

The User screen supports creating new users, searching for existing users, deleting users, and adding users to a group. There is currently no multiple groups support and all users are added to the same single group for generating recommendations. Figure 1 illustrates a screenshot of the User screen. The usernames are represented in integer values to avoid exposing any personal information.

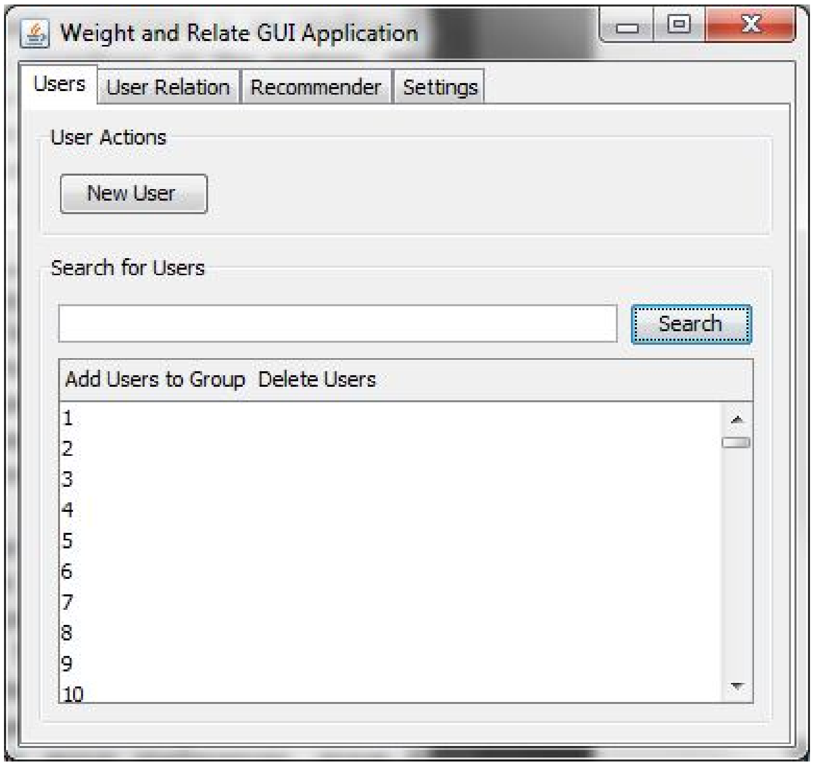
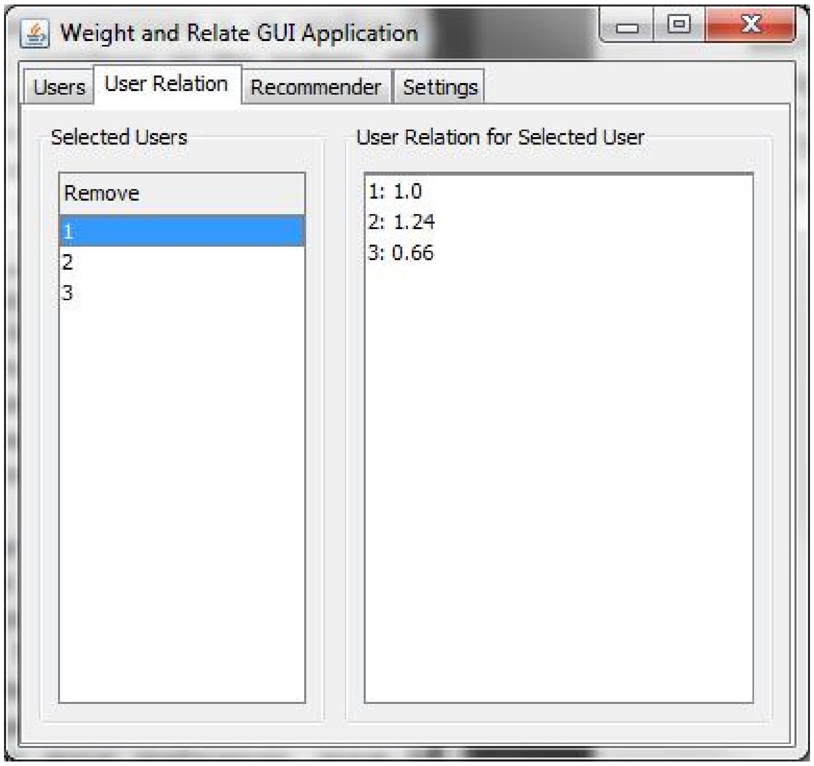


Figure 1 User Screen Example

Once users have been added to the group, the User Relation screen supports direct visualizations and modifications of the user-relation-function values between users in the group. The purpose of this screen is to support experimentation of the WaR algorithm under different user relationships. In a consumer-based system, this type of capability is usually not exposed to the end users; instead, the user-relation-function values are adjusted internally by the system through user feedbacks. Figure 2 illustrates a screenshot of the User Relation screen. By selecting a particular user in the group, his user-relation-function values for all users in the group will appear as numeric values.

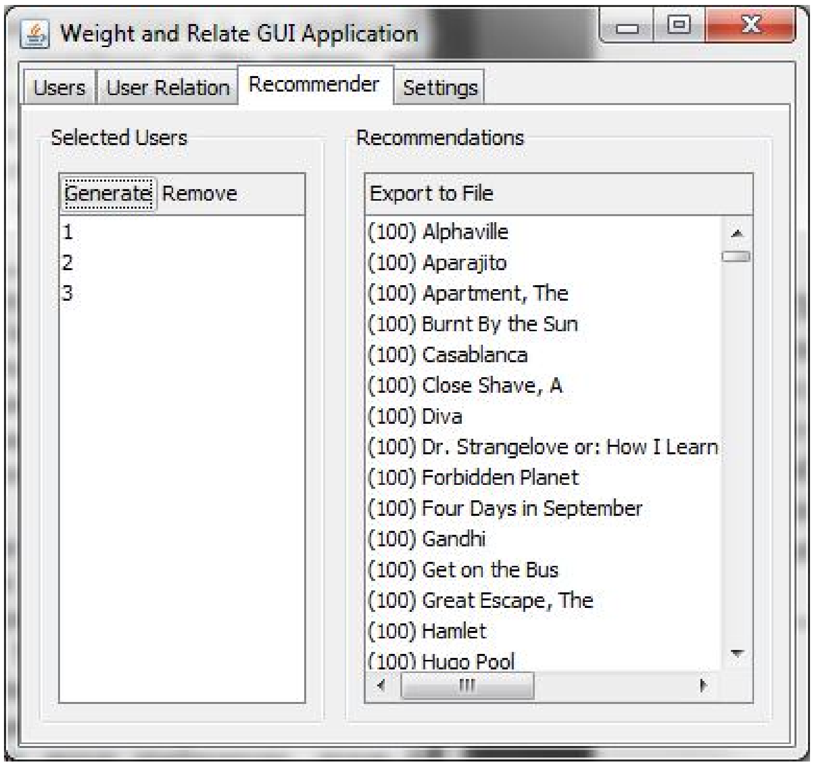


Weight and Relate GUI Application
Users
User Relation
Recommender
Settings
Selected Users
User Relation for Selected User
Remove
1
2
3

```
1: 1.0
2: 1.24
3: 0.66
```

Figure 2 User Relation Screen Example

The Recommender screen supports generating recommendations for the selected user(s) in the group. The generated recommendations are sorted in the order of the group's highest preference to lowest preference, determined by the underlying WaR algorithm. The screen also provides an “export to file” capability, which exports the generated recommendations to a file on the file system. The exported content also includes all users in the group, the user-relation-function values between the users, and the parameters used by the WaR algorithm to generate the recommendations. The purpose of this export utility is to assist in investigations and comparisons of the generated recommendation under different parameter settings, which is also uncommon in consumer-based systems. Figure 3 illustrates a screenshot of the Recommender screen. The recommendations are shown in conjunction with its recommended percentage value.

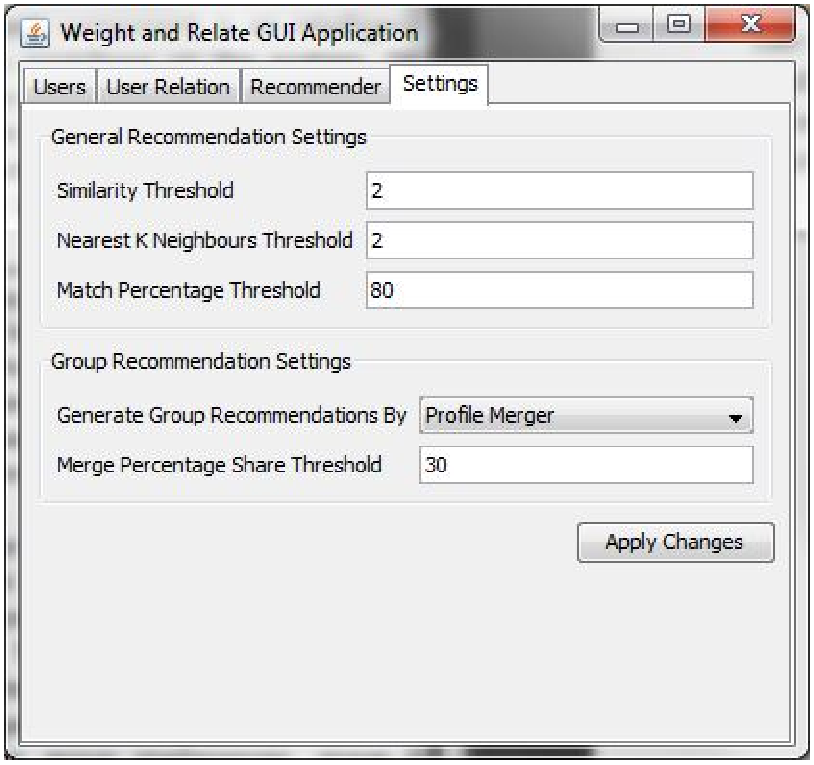


1
2
3

(100) Alphaville
(100) Aparajito
(100) Apartment, The
(100) Burnt By the Sun
(100) Casablanca
(100) Close Shave, A
(100) Diva
(100) Dr. Strangelove or: How I Learn
(100) Forbidden Planet
(100) Four Days in September
(100) Gandhi
(100) Get on the Bus
(100) Great Escape, The
(100) Hamlet
(100) Hugo Pool

Figure 3 Recommender Screen Example

The Settings screen supports adjusting the different parameters that are used by the WaR algorithm, including the similarity threshold value, the nearest-k-neighbors threshold value, the match percentage threshold value, the group recommendation merger procedure, and the merger procedure's merge percentage share threshold values. Similar to the User Relation screen, this adjustment capability is usually not exposed to the end users and instead maintained within the system internally. Figure 4 illustrates a screenshot of the Settings screen. Once applied, the settings will be automatically applied the next time the WaR algorithm is executed.



General Recommendation Settings

Similarity Threshold	2
Nearest K Neighbours Threshold	2
Match Percentage Threshold	80

Group Recommendation Settings

Generate Group Recommendations By	Profile Merger
Merge Percentage Share Threshold	30

Figure 4 Settings Screen Example

Section 3.B. Application Layer and Database Layer

The Application layer contains the core business logic of the prototype, including implementation of the WaR algorithm and management of users. It provides multiple application programming interfaces (API) to the Interface layer for accessing information and sending recommendation queries. It also interacts with the Database layer, which manages all data used by the system, to properly process and evaluate recommendation queries using the WaR algorithm.

Within its database, the WaR prototype stores data with regards to user information, product information, previous user ratings, and user-relation-function values. Data contained within the MovieLens dataset of 100k ratings, as collected by the GroupLens Research Project [23], were imported into the WaR prototype database in order to perform experimentations.

Chapter 4. Evaluation and Results

Section 4.A. Objective: Prediction Accuracy

The prediction accuracy objective is evaluated by the Mean Absolute Error (MAE) metric. The MAE metric is defined as the average absolute difference between the predicted rating values and the actual rating values, where lower MAE indicates more accurate predictions [33]. The MAE metric is used to evaluate the accuracy of the prediction algorithm.

In the WaR algorithm, the prediction algorithm serves the most basic yet essential role. It is the base component that allows recommendations to be generated for both individuals and groups. Therefore, the MAE metric essentially measures the accurateness of the entire WaR algorithm.

Although it is desirable for the MAE value to be 0 for all predictions, however, it is not realistic due to the nature of predicting. Furthermore, a recent study by McNee et al. [30] argued that algorithms with lowest MAE values may not always be desirable for a recommender system, and other metrics have to be considered. Thus, it is expected that the WaR algorithm should not perform significant worse than other collaborative filtering algorithms. The MAE metric results obtained from the WaR algorithm will be compared against other collaborative algorithms.

To properly evaluate the prediction accuracy metric for the WaR algorithm, it is desirable to compare the collected MAE value against other collaborative algorithms. The following table illustrates the reference MAE values resulted from experiments performed with other collaborative filtering algorithms, namely combining collaborative filtering with personal agents by Good et al. [31], combining content filtering with collaborative filtering by Melville et al [32], analyzing user-item matrix with collaborative filtering by Sarwar et al. [34], and clustering items for collaborative filtering by O'Connor et al. [35]. When there are multiple MAE values presented due to different conditions, the lowest MAE value is used for comparison. It must be noted that the conditions and environments are different in each experiment; hence the comparison of the results should only be used as a reference.

Reference Collaborative Filtering Algorithms	**Lowest MAE Value**
Combine with Personal Agents [31]	0.8303
Content-Boosted [32]	0.962
Item-Based [34]	0.726
Clustering-Items [35]	0.7594

Table 5 Reference MAE Values

Section 4.B. Experiment Environment and Results

The MovieLens dataset of 100k ratings collected by the GroupLens Research Project [23] was used to evaluate the WaR algorithm for the prediction accuracy objective. The dataset contains 100,000 ratings from 943 users on 1682 movies, and each user has rated at least 20 movies. The dataset was imported into a MySQL server database and the recommendation algorithm was implemented using the Java programming language.

The goal of the experiments was to determine the lowest MAE value using the static user data collected by GroupLens. Two variables are altered in the experiments: similarity-threshold and nearest-k-threshold. The values used for the two variables are 2, 4, 6, 8, 10, 20, 40, 60, 80, and 100.

Each experiment is executed with a unique combination of the two variables. The experiments calculate and average the MAE values for all modelled users. The following table shows 5 experiment results with lowest MAE values and 5 experiment results with the highest MAE values.

	Similarity Threshold	Nearest K Threshold	MAE Value	Number of Ratings Compared
Low	**2**	**6**	**0.584572**	**98414**
	2	4	0.587457	98895
	4	4	0.587499	98883
	2	8	0.588124	97848
	4	2	0.588553	99328
High	80	100	0.706129	11755
	80	80	0.706964	17964
	100	100	0.708142	5108

	100	40	0.708552	25568
	100	60	0.714145	15394

Table 6 MAE Values obtained from Experiments

Figure 5 and Figure 6 show the MAE value when using all combinations of similarity-threshold and nearest-k-threshold with values 2, 4, 6, 8, and 10. The combinations with values of 20, 40, 60, 80, and 100 are not shown since the MAE values simply increase as the threshold values increase.

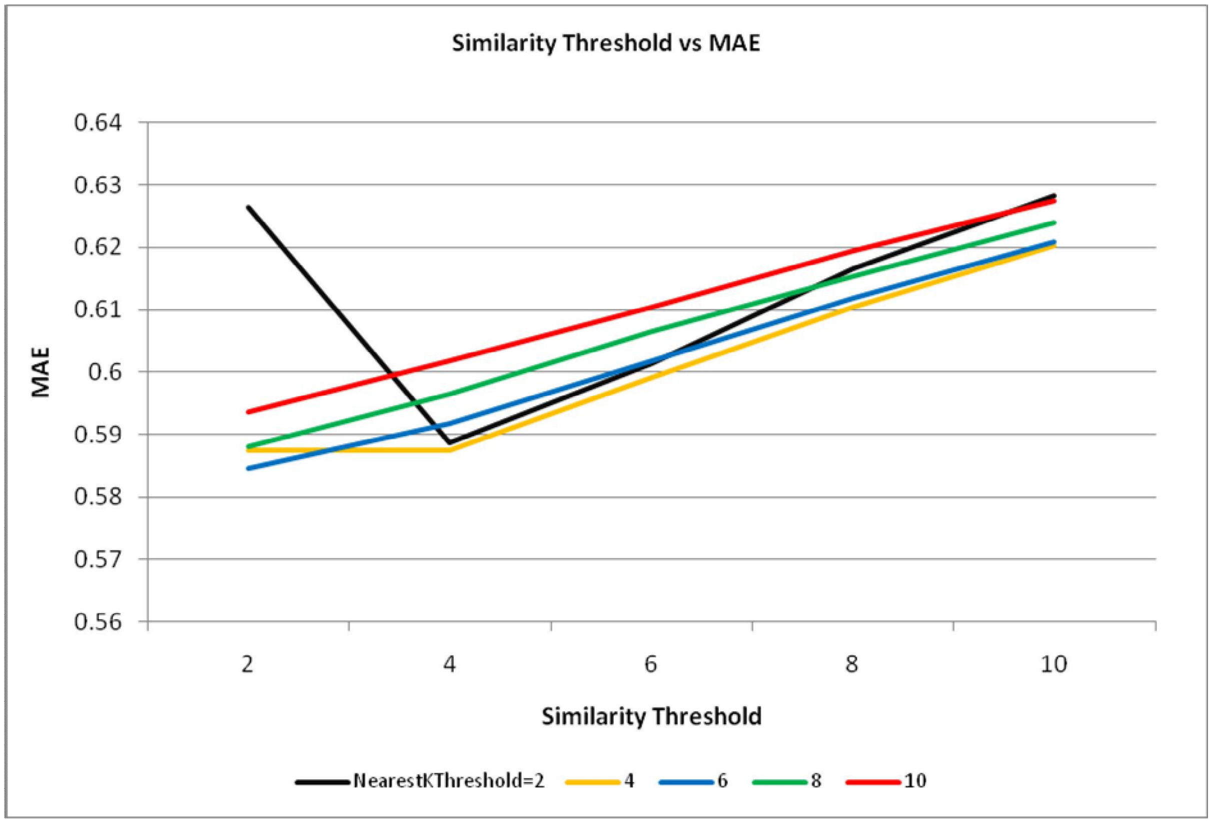


Figure 5 Similarity Threshold vs MAE Chart

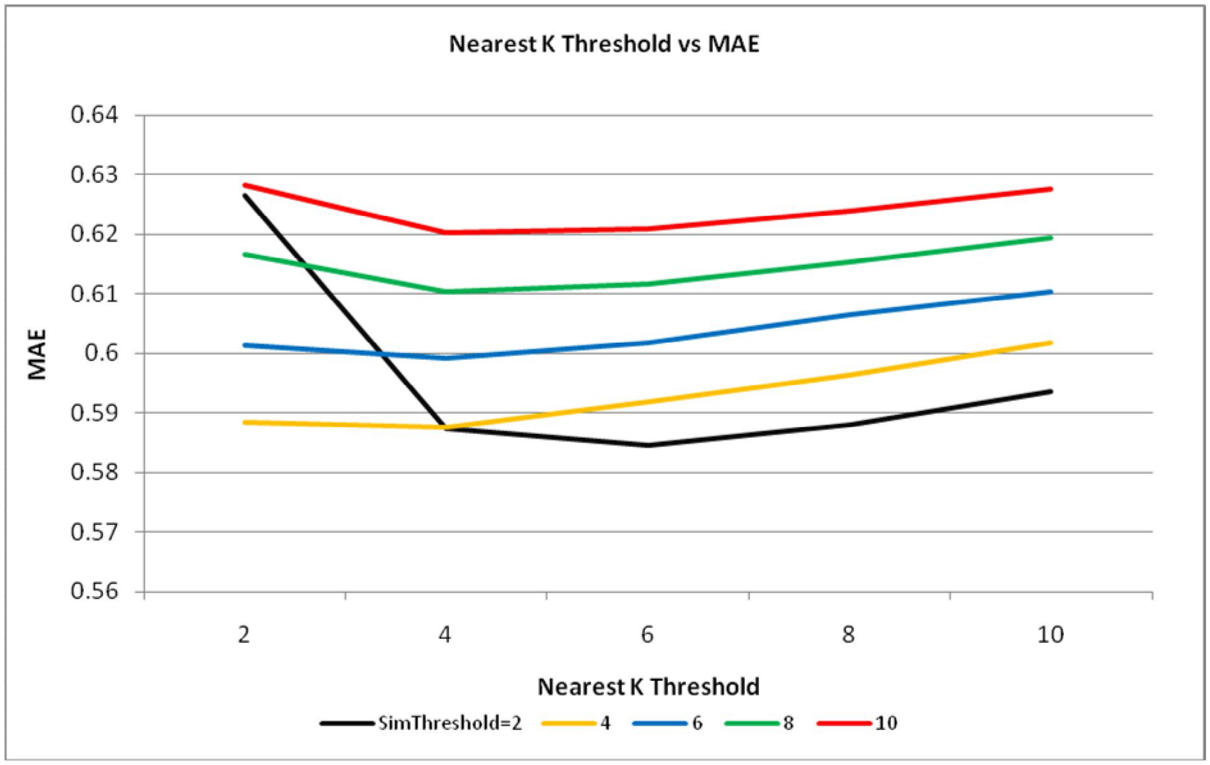


Figure 6 Nearest K Threshold vs MAE Chart

Section 4.C. Evaluating the Prediction Accuracy Objective

In the user experiments, the lowest MAE value collected is 0.584572, for which 98414 ratings were compared. The highest MAE value collected is 0.714145, for which 15394 ratings were compared. The low MAE value obtained from the user experiments compares favorably to other collaborative algorithms with at least a 0.16 value difference (illustrated in Section 4.A). Therefore, it is concluded that the resulting data is acceptable, although not perfect, and the prediction accuracy objective is satisfied.

Chapter 5. Discussion

Section 5.A. Prediction Algorithm

The prediction algorithm utilizes two variables as part of its execution: similarity-threshold and nearest-k-threshold. The similarity-threshold is used to eliminate profiles with low difference factors simply because of minimal common items with the target profile. The nearest-k-threshold is used to avoid collecting rating value data from excessive amount of profiles. When evaluating the prediction accuracy metric, part of the experiments were to also determine to appropriate values for similarity-threshold and nearest-k-threshold.

In the user experiments, the results indicated that a combination of (2, 6) for the similarity-threshold and nearest-k-threshold yields that lowest MAE value, followed closely by (2, 4) and (4, 4). On the other hand, the combination of (100, 60) yields the highest MAE value, followed by (100, 40) and (100, 100). Thus, from the user experiments, it can be concluded that using lower values for similarity-threshold and nearest-k-threshold would produce much accurate predictions. Therefore, it is suggested to control both similarity-threshold and nearest-k-threshold values within a range of 2 to 10, as such range produces optimal results during the experiments.

Section 5.B. Individual Algorithm

The individual algorithm is based on a specific collaborative filtering implementation called nearest-k-neighbours technique. A common approach when implementing the nearest-k-neighbours is to first find the nearest neighbours, then find items to recommend based on the items such neighbours has previously rated. This approach has preferable performance as the search scope is limited – however, accuracy suffers since common rated items between the neighbours are not always guaranteed.

A different approach is taken with the implementation of the WaR individual algorithm. Instead of locating a static set of nearest neighbours, each item has its own set of nearest neighbours. Therefore, the predicted ratings are generated based on the nearest neighbours associated with each item. Using this approach, it is guaranteed that each predicted rating is based on previous ratings from at least k neighbours. Although the performance of the algorithm will likely degrade due to the expanded search space, the accuracy should improve by incorporating a definite number of ratings.

Section 5.C. Group Algorithm

The user-relation-function serves a key role for the group algorithm to capture social interactions, as the function defines the degree of influence between users and the impact each user has on the final recommendations. The tasks of maintaining and updating the user-relation-function are delegated to the recommender system that employs the WaR algorithm. The WaR prototype simply allows direct modifications of the user-relation-functions as its intention is experimentation only. In consumer-based systems, the tasks are achieved through user feedbacks. When the recommendations are presented, users may provide feedbacks on the recommendations in the form of rating values, both as a group and as individuals. Based on these feedbacks, it can be assumed that users with individual rating values closer to group rating values would have larger impact within the group. The user-relation-function can then be updated accordingly to slightly shift the degree of influence to users with larger impacts.

In both Profile Merger and Recommendation Merger, the merge-percentage value is used to adjust the accuracy of the final recommendations. Increasing the merge-percentage implies that a smaller but more accurate set of recommendations will be generated, whereas decreasing the value yields the opposite outcome. The merge-percentage values are structured to be dynamic and user-defined such that the generated set of recommendations can be controlled externally.

Section 5.D. Comparison to Other Group Recommender Systems

Many previously proposed group recommender algorithms have restricted the product domain or user audiences. Let’s Browse and I-Spy for web browsing and searching, MusicFX for music selection, PolyLens for movie recommendation, and Travel Decision Forum and Intrigue for tour itineraries and attractions. On the other hand, the WaR algorithm is developed to be a general algorithm that can be applied to any domains or audiences.

To compute group recommendations, many previous algorithms choose the collaborative filtering technique by forming some type of group preference model based on the individuals in the group. Often the group preference model is computed by an aggregation algorithm using individuals' preferences as the inputs. Once the candidate recommendations are generated based on the group preference model, each of the candidates are again compared against each individual's preferences for potential eliminations. Examples include the PolyLens and MusicFX.

The WaR algorithm employs two variations of the aggregation concept. The profile merger technique is similar to the group preference model approach described above. However, the recommendation merger is not usually employed in previous algorithms. This procedure avoids the process of forming a group preference model, thus eliminating the potential model aggregation failure.

A difference between the previous recommender algorithms with respect to forming the group preference model is the ability to view and adjust the model. Algorithms such as Travel Decision Forum and Intrigue allow the members of the group to examine and negotiate an appropriate group preference model before actually issuing the recommendation request. This is often desirable as users are the one that knows the most about their preferences; however, this leads to user privacy concerns. On the other hand, algorithms that do not provide the ability to adjust the model, such as PolyLens, might generate less accurate recommendations but users do not have to worry about their privacy.

The WaR algorithm operates differently than previously-mentioned algorithms in that it allows implicit adjustment of the group preference model through the use of user-relation-function and weight coefficients; users with the higher weight coefficient will have a larger impact on the resulting recommendations. This approach may not generate as accurate recommendations as the examining and negotiating process, but it is faster in terms of setting up the model since only the request initiator is involved in the process.

A deficiency of the WaR algorithm is its GUI component, specifically the WaR prototype. Many of the previous group recommender systems already provides sophisticated interface components, such as the Travel Decision Forum with the multiple user discussion support. As the WaR algorithm is in its infant stages, the features of the WaR prototype is comparatively lacking in all areas. However, once the algorithm has matured, more appropriate GUI systems can be designed and developed to use the full potential of the WaR algorithm.

Conclusion

While many recommender systems have focused on serving individuals, this paper presented the WaR algorithm, based on collaborative filtering, that supports both individuals and groups settings. The algorithm relied on the ability to predict ratings accurately, and it was the primary objective in evaluating the algorithm. The results obtained from experiments were positive, where most of the data indicated acceptable values of predicted ratings. In addition to designing the algorithm, a WaR prototype was implemented to demonstrate the capabilities of the algorithm. Many features that are normally not seen in consumer-based systems are exposed in the prototype to assist in experimenting with the algorithm.

The next crucial step is evaluating the group algorithm, which will involve conducting experiments with real users and groups. The primary objective for the group algorithm evaluation is to determine the accuracy of the group recommendations, experimenting with both profile merger and recommendation merger.

The WaR algorithm currently utilizes only the collaborative-filtering concept to generate recommendations. Incorporating other concepts can further improvement the accuracy of the algorithm. The WaR algorithm currently does not apply any filtering mechanisms in the filtering stage, and the set of all modelled items are used in the item evaluation stage. Additional filtering mechanisms can be utilized to produce an appropriate set of candidate items that are then used in the item evaluation stage. The filtering mechanisms effectively remove any unnecessary items from consideration, which should improve prediction accuracy as such items are not evaluated. However, the unnecessary items need to be defined appropriately by the filtering mechanisms.

Many different filtering mechanisms can be incorporated into the WaR algorithm. The simplest example could be removing any item that the target user/group has already rated. More complicated examples could be to remove any items that do not match user-specified requirements of item features, or filter any items that do not match user's location or personal data. Ultimately, the algorithms developed for either the content-filtering or context-filtering concepts can be applied at the filtering stage in the WaR algorithm.

When the WaR algorithm is thoroughly evaluated, it can then be employed by any recommender systems that require either individual or group settings. The algorithm is meant to be flexible and not restricted to any specific audiences, domains, or environments.

Bibliography

[1] T. Tran. Combining Collaborative Filtering and Knowledge-Based Approaches for Better Recommendation System. *Journal of Business and Technology*, 2007.

[2] M. Montaner, B. Lopez, and J. Lluis De La Rosa. A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review 19*, pages 285-330, 2003.

[3] J.B. Schafer, J. A. Konstan, and J. Riedl. E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery*, pages 115-153, 2001.

[4] D. Cosley, S. K. Lam, I. Albert, J. A. Konstan, and J. Riedl. Is Seeing Believing? How Recommender Interfaces Affect User's Opinions. *In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 585-592, 2003.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230-237, 1999.

[6] F. Abbattista, M. Degemmis, N. Fanizzi, O. Licchelli, P. Lops, G. Semeraro, and F. Zambetta. Learning User Profiles for Content-Based Filtering in E-Commerce. In *Proceedings of AI Workshop (AIIA-02)*, 2002.

[7] R. Burke. Integrating Knowledge-Based and Collaborative Filtering Recommender Systems. *In Proceedings of the AAAI-99 Workshop on AI and Electronic Commerce,* 1999.

[8] A. Jameson, S. Baldes, and T. Kleinbauer. Enhancing Mutual Awareness in Group Recommender Systems. *Proceedings of the IJCAI 2003 Workshop on Intelligent Techniques for Web Personalization,* 2003.

[9] A. Jameson. More Than the Sum of Its Members: Challenges for Group Recommender Systems. *Working Conference on Advanced Visual Interfaces.* 2004.

[10] MusicFX, "MusicFX: Adapting Music to Please Most of the People All of the Time", Joe McCarthy [online], [cited Sept. 1, 2009], available from the World Wide Web: <http://interrelativity.com/joe/projects/MusicFX.html>.

[11] M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl. PolyLens: A Recommender System for Groups of Users. *Proceedings of the 7th European Conference on Computer Supported Cooperative Work,* 2001.

[12] MovieLens, "MovieLens – Movie Recommendations", GroupLens Research [online], [cited Sept. 1, 2009], available from the World Wide Web: <http://www.movielens.org/login>.

[13] J.F. McCarthy and T.D. Anagnost. MusicFX: An Arbiter of Group Preferences for Computer Supported Collaborative Workouts. *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, 1998.

[14] M.V. Nagendra Prasad and J.F. McCarthy. A Multi-Agent System for Meting Out Influence in an Intelligent Environment. *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, 1999.

[15] H. Lieberman, N.W. Van Dyke, and A.S. Vivacqua. Let's Browse: A Collaborative Web Browsing Agent. *Proceedings of the 4th international conference on Intelligent user interfaces*, 1998.

[16] G. Adomavicius and A. Tuzhilin. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[17] L. Ardissono, A. Goy, G. Petrone, M. Segnan, and P. Torasso. Intrigue: Personalized Recommendation of Tourist Attractions for Desktop and Handset Devices. *Applied Artificial Intelligence: An International Journal*, 2003.

[18] A. Jameson and B. Smyth. What a Difference a Group Makes: Web-Based Recommendations for interrelated Users, http://www.win.tue.nl/~laroyo/2L340/resources/group-recommenders-jameson-smyth.pdf (2009).

[19] R. Burke. Hybrid Recommender Systems: Surveys and Experiments. *User Modeling and User-Adapted Interaction*, 2002.

[20] K. McCarthy, L. McGinty, B. Smyth, and M. Salamo. The Needs of the Many: A Case-Based Group Recommender System. *Advances in Case-Based Reasoning*, 2006.

[21] B. Smyth, J. Freyne, M. Coyle, P. Briggs, and E. Balfe. I-SPY – Anonymous, Community-Based Personalization by Collaborative Meta-Search. *Proceedings of the 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 2003.

[22] B. Smyth, E. Balfe, J. Freyne, P. Briggs, M. Coyle, and O. Boydell. Exploiting Query Repetition and Regularity in an Adaptive Community-Based Web Search Engine. *User Modeling and User-Adapted Interaction*, 2005.

[23] GroupLens Research, “MovieLens Data Sets”, GroupLens Researhc [online], [cited Jul. 29, 2010], available from the World Wide Web: <http://www.grouplens.org/node/73>.

[24] P. Kearney, M. Shapcott, S. Anand, D. Patterson. Evaluating the use of Semantics in Collaborative Recommender Systems: A User Study. *Intelligent Techniques for Web Personalization and Recommender Systems in E-Commerce*, pages 46-53, 2007.

[25] G. Shani, A. Meisles, Y. Gleyzer, L. Rokach, and D. Ben-Shimon. A Stereotypes-Based Hybrid Recommender System for Media Items. *Intelligent Techniques for Web Personalization and Recommender Systems in E-Commerce*, pages 76-83, 2007.

[26] H. Stormer. Improving E-Commerce Recommender Systems by the Identification of Seasonal Products. *Intelligent Techniques for Web Personalization and Recommender Systems in E-Commerce*, pages 92-99, 2007.

[27] Z. Zhang, D. Zhang, and Z. Guo. Improving Memory-based Collaborative Filtering Using a Factor-based Approach. *Intelligent Techniques for Web Personalization and Recommender Systems in E-Commerce*, pages 110-117, 2007.

[28] J.B. Schafer, J. A. Konstan, and J. Riedl. Recommender Systems in E-Commerce. *Proceedings of the 1st ACM conference on Electronic commerce,* pages 158-166, 1999.

[29] W. Lee, Chih. Liu, and Cheng Lu. Intelligent agent-based systems for personalized recommendations in Internet Commerce. *Expert Systems with Applications*, 2002.

[30] S. McNee, J. Riedl, and J. Konstan. Being Accurate is not good enough: How Accuracy Metrics have hurt Recommender Systems. *CHI '06 extended abstracts on Human factors in computing systems*, 2006.

[31] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining Collaborative Filtering with Personal Agents for Better Recommendations. *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, 1999.

[32] P. Melville, R. Mooney, and R. Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations. *Eighteenth national conference on Artificial intelligence*, 2002.

[33] J. Herlocker, J. Konstan, L. Terveen, and J. Riedel. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 2004.

[34] B. Sarwar, G. Karypis, J. Konstan, and J. Riedel. Item-Based Collaborative Filtering Recommendation Algorithms. *Proceedings of the 10th international conference on World Wide Web*, 2001.

[35] M. O'Connor and J. Herlocker. Clustering Items for Collaborative Filtering. *In Proceedings of the ACM SIGIR Workshop on Recommender Systems*, 1999.

[36] A. Jameson, B. Smyth. Recommendation to Groups. *The adaptive web: methods and strategies of web personalization,* 2007.

[37] S. Amer-Yahia, S.B. Roy, A. Chawla, G. Das, C. Yu. Group Recommendation: Semantics and Efficiency. *Proceedings of the VLDB Endowment*, 2009.