2009 International Conference on Computational Science and Engineering

A Visual Interface for Social Information Filtering

John O'Donovan*, Brynjar Gretarsson†,
Svetlin Bostandjiev‡, Tobias Höllerer¶
Department of Computer Science
University of California, Santa Barbara
USA
*jod@cs.ucsb.edu
†brynjar@cs.ucsb.edu
‡alex@cs.ucsb.edu
¶holl@cs.ucsb.edu

Barry Smyth§
CLARITY: Centre for Sensor Web Technologies
School of Computer Science and Informatics
University College Dublin
Ireland
§barry.smyth@ucd.ie

***Abstract*—Collaborative or "Social" filtering has been successfully deployed over the years as a technique for analysing large amounts of user-preference knowledge to predict interesting items for an individual user. The black-box nature of most collaborative filtering (CF) applications leave the user wondering how the system arrived at its recommendation. In this paper we introduce *PeerChooser*, a collaborative recommender system with an interactive interface which provides the user not only an explanation of the recommendation process, but the opportunity to manipulate a graph of their peers at varying levels of granularity, to reflect aspects of their current requirements. *PeerChooser's* prediction component reads directly from the graph to yield the same results as a benchmark recommendation algorithm. Users then improve on these predictions by tweaking the graph in various ways. *PeerChooser* compares favorably against the benchmark in live evaluations and equally well in automated accuracy tests.**

I. Introduction

The idea behind recommender systems is to provide a user with a useful recommendation. Many recommender systems use item-based [12] or user-based [11] collaborative filtering (CF). CF models the social process of asking a friend for a recommendation. However, without the right friends the process obviously fails. In the CF literature, such failures have been termed the "grey-sheep", "sparsity" and "early-rater" problems. In the real world it is difficult to choose who your peers will be, as modeled in most current CF systems. Imagine you want a movie recommendation and can choose a group of peers, in addition to your existing group. You might choose Stephen Spielberg, among others if you're in the mood for a Sci-Fi flick. However, what if you told Mr Spielberg your top 10 movies, and he hated all of them. With this knowledge, would you still want his contributions?

In this paper we show that not only is it possible to manipulate CF peergroups based on dynamic and informative feedback [10], but doing so greatly improves the users experience with the recommendation system. The process is fast and more importantly fun for the end user. Dynamic feedback is achieved in *PeerChooser* by allowing a user to see what their existing peergroup would recommend for the user's top-n favorite and least-favorite items. Users can then tweak the neighborhood graph to tune the predicted ratings on their movie list to the required level. In fact, an existing profile is not essential for the process to work well. By starting at an average position in the neighbourhood-space representing all possible peers, a user can then manipulate neighbor icons to optimize predictions on their own favorite movies. This technique can rapidly generate information upon which the system can base predictions on unseen items.

PeerChooser also serves as an explanation interface for CF, affording the user the opportunity to gain an understanding of the core trends in the data, and become more aware of where the final predictions are really coming from. Following from the work of Herlocker et al. in [2], we show that visualization provides an openness to the recommender system, which in turn enhances overall user trust in the system– something which is lacking in traditional recommender systems [9].

Furthermore, *PeerChooser*'s interactive components provide the user with the option of telling the system about current requirements. Mood and persona are important factors to be considered in the recommendation process [4], but they are also highly ephemeral phenomena in the context of the web, making them very difficult to profile. For example, a user may purchase books on Egypt prior to a vacation, but not want recommendations of books on ancient Egypt six months later.

The remainder of this paper is organized as follows: Section II presents a review of the related work in recommender systems and interfaces. Section III outlines the architecture of the *PeerChooser* recommender. Section III-A details the force directed layout used in *PeerChooser*. Section III-B explains how personalized neighborhood graphs are constructed within *PeerChooser*. Section III-C discusses the application of our system as a tool for visualizing trust relations between users. Section III-D explains the prediction mechanisms in *PeerChooser*. Section IV presents results of our empirical and live-user evaluations of the system.

II. Background

Recent research in recommender systems is taking the focus off large scale accuracy metrics such as mean absolute error and precision-recall, and placing more emphasis on general user satisfaction with the system. For example McNee et al.

978-0-7695-3823-5/09 $26.00 © 2009 IEEE
DOI 10.1109/CSE.2009.26

IEEE computer society

in [5] posit that the "usefulness" of a recommendation should be determined by a user's *current* need. Herlocker et al. [2] believe that "usefulness" of a recommendation is as important as accuracy. Ziegler et al. construct a similar argument in [15] and analyse the importance of topical diversity within recommendation lists. In this work, facets from visualization research have been incorporated into a novel recommender system to improve performance and the user's overall experience with the system.

Many synergies have been created by applying visualization and interactivity to existing applications, for example the application of friend-visualization on the social networking web site Facebook [3]. Through visualization we are creating an "explanation interface" for our recommender system. Some research has been carried out into the effects recommendation explanation has on the overall user experience with the system. A prominent work in this field is Herlocker's study of recommendation explanations [2]. Herlocker et al. evaluate a "white box" conceptual model of recommendation as opposed to the run-of-the-mill black box approach. They present a user study where 21 different recommendation interfaces are presented to users, explaining various types of information from within the recommender algorithm. Herlocker found that users preferred viewing histograms of their neighbors' ratings for the recommended item over all other approaches. This is in agreement with Middleton's findings [7] that "explanation interfaces lead to improved acceptance of a predicted rating."

More recent work by Freyne et al. explores the value of explanation in the web-search domain. Their work in [1] presents a social search application in which the "influence of community wisdom" is presented and used to explain ranking in a search engine. Results from their user survey show that users generally find the social visualization both interesting and relevant.

This brief examination of related work shows that a synergy can be achieved by combining information filtering algorithms with visualization tools to create an "explanation interface". However, the application we now present serves not only as an explanation, but as a fully interactive *control* over the neighborhood data upon which our algorithms operate.

III. Architecture

PeerChooser is a visualization-based collaborative filtering recommender system which allows users to explore, understand and visually interact with the underlying data in the system. The system brings a novel aspect of openness and accountability to the recommendation process, allowing users to familiarize themselves with the structure and trends within the data.

To date, collaborative filtering systems have relied heavily on what is termed the *similarity assumption* [9]: that similar profiles (similar in terms of their ratings histories) make good recommendation partners. We argue that in addition to the standard similarity assumption, users can benefit from incorporating facets of their current mood or persona into the recommendation process. This is facilitated in full by *PeerChooser*'s interactive neighborhood graphs.

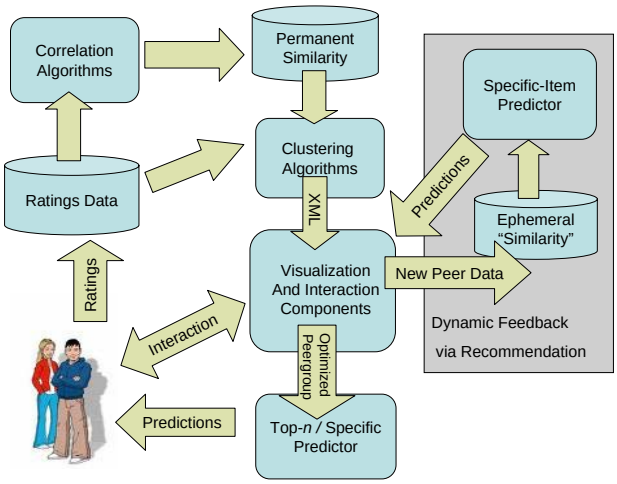

Fig. 1. Architecture of the PeerChooser Visual Interactive Recommender System.

Before presenting a discussion of the actual visualization of the CF process, we will discuss the high level architecture of the system. Figure 1 presents an outline of the *PeerChooser* architecture. User rating data is correlated to produce a model of similarity. For this work we use Pearson's correlation function as used in [9] to compute correlations between users. The resulting similarity model, ratings, and item genre information from the database are passed to a graph generator component. This is essentially a clustering algorithm which generates an XML representation of a personalized graph for one "active" user.

Section III-A describes the graph layout in detail. The graph generated is a spoke-model with the active user positioned at the center. Edges are shown between the active user and every other user in the system, up to a threshold number. These edges represent the Pearson correlation between the active user and each neighbor node. Edges are also drawn between non-active user nodes and genre nodes. These edges indicate an affinity for the associated genre based on underlying rating data. The following section details the processes by which these connections are determined.

As the user manipulates icons on the graph, dynamic predictions are generated by the system. These can be seen on the right panel in Figure 2. The active user can specify the items to be dynamically predicted by the system. In our user trials we suggest that this be a list of 5 liked and 5 disliked movies. Users play around with the graph until they are satisfied with the dynamic predictions on their movie list. When a user reports that they are satisfied with the layout, the personalized graph is saved for later use. The user can now receive specific, or top-n recommendation lists based on their personalized graph configuration.

The graph is currently scalable to approximately 1000 users, but fewer nodes are easier for a user to understand

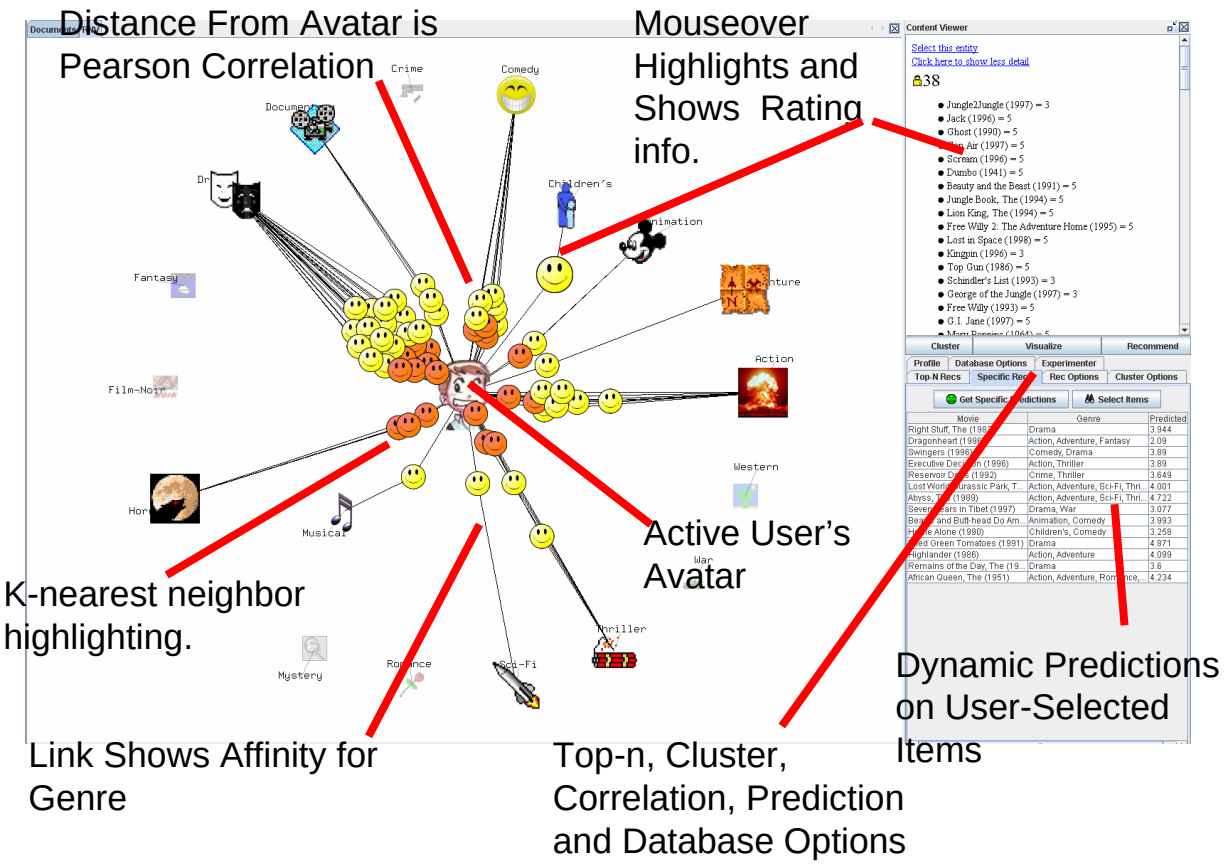

Fig. 2. Annotated Screenshot of PeerChooser's Interactive Interface.

and manipulate. Accordingly, the number of users shown is thresholded according to similarity with the active user. The *PeerChooser* graph is highly interactive, users can freely move nodes around, delete nodes or edges at an individual level or on a per cluster level. In doing this the active user is effectively changing the value of the similarity weight to be used in our CF prediction formula. For this work we use a standard Resnick prediction formula, shown in Formula 2. By moving an icon representing "comedy", for example, closer to the center of the graph, the user is essentially telling the system, "right now I feel more similar to comedy fans than fans of other genres." This allows a user to express current mood/requirements to the system quickly and easily. *PeerChooser* then overrides the existing similarity value by creating an *ephemeral similarity model* based on the new values.

A. Forces in the Visualization

PeerChooser uses a simple force-directed spring layout algorithm [14]. Figure 3 provides an overview of the node interactions in the layout algorithm. Edges between all nodes

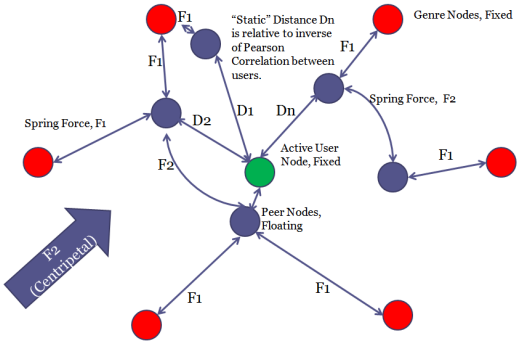

Fig. 3. Node interactions and forces in *PeerChooser*

are represented as springs. All the springs have a natural length, measured in the same units as the screen coordinate system, i.e: pixels, which they attempt to achieve constantly. Force $F1$ operates between genre nodes and non-active user nodes. Force $F2$ acts between non-active users' nodes and prevents them from collapsing in on each other. There is a

constant centripetal force that prevents nodes from drifting too far from the center. The key point of this layout is that the distances $D1, D2, ..., Dn$ are static and cannot be changed by other forces in the algorithm. Only mouse movements from the user can change these distances, which represent the correlation between the active user and the individual peer nodes. *PeerChooser* is a Java/OpenGL application and can automatically detect screen resolution and adjust its pixel map accordingly. If a spring in *PeerChooser* is shorter than its natural length it extends, pushing the nodes at either end of the edge apart. If a spring is longer than its natural length it contracts, pulling the nodes at either end together. The force exerted by the spring is proportional to the difference between its current length and its natural length. Nodes linked together tend to form clusters so a repulsive force is also added so there will still be distinguishing space between the nodes. To keep the nodes in the *PeerChooser* graph from drifting into too large a space, a weak centripetal force is applied to each node. Intra-node forces keep the graph from collapsing in on itself. The lengths are changed iteratively to obtain a well spaced out layout by minimizing the total energy. During initial implementation it was found that nodes tended to "quiver", i.e: jump back and forward rapidly between two or more approximately equal resting positions. To address this issue a resting-heuristic was introduced to settle each note once it had minimized its energy to within a threshold. The spring layout is initialized with a graph of nodes and edges. X/Y locations are assigned to each node in the network. The nodes are then moved iteratively to minimize the energy.

B. Building Neighborhood Graphs

We are interested in allowing a user to manipulate a neighborhood space to generate better recommendations and express current preferences. So far we have explained how we use Pearson's correlation coefficients to place users relative to the active user node in the graph. Without suitable information about these connected nodes the user would not gain from interacting with the graph. Ideally, we would like to show the correlation between each pair of users in the system and have this information displayed to the active user. However, expressing a high-dimensional neighborhood space in a 2-dimensional, or even 3-dimensional space is a complex task and difficult for a user to understand. To maintain ease of use for the end user we map the complex ratings data onto the space of movie genres and present this to the end user. An edge is formed from a user to a genre if their affinity for that genre is above a threshold value. The spring forces in *PeerChooser* cause a natural clustering of non-active user nodes based on genre affinity. Since each user is clustered to a genre, the active user can gain a lot of information about the neighborhood at a glance. Each genre cluster is labeled with the appropriate name. The clustering algorithm can associate with 1 to many genres, Figure 4 shows the visualization for a graph with 2 connections from user nodes to genre nodes (i.e, a user's favorite and second favorite genre).

The genre distribution in the MovieLens data has a strong bias towards three genres: Drama, Action, and Comedy. As a result, our initial clustered graphs tended to only show connections between non-active user nodes and nodes representing these three genres. To counter this problem and provide a more diverse set of neighbors we apply a scaling function to our edge drawing threshold based on number of occurrences of a genre in the database. Equation 1 shows how the system computes user to genre connectivity on a per-user basis. In Equation 1 G is the set of all genres, g represents one genre. Term $U_{liked,g}$ indicates the number of items user U liked in genre g. U_{total} represents the total number of ratings by U; g_{total} is the number of movies in genre g and c is a scaling constant.

$$Max_{(g \in G)}\left(\frac{U_{liked,g}}{U_{total}} + \frac{U_{liked,g}}{g_{total}} \cdot c\right) \quad (1)$$

C. Visualizing Trust Relations

In addition to providing an explanation of the recommendation process to the end user, *PeerChooser* enables the user not only to visualize *correlation*, but also the *trust-space* generated from the underlying ratings data.

A trust matrix from [9] built on the MovieLens dataset was incorporated into the visualization mechanism as a means to provide at-a-glance information to the end user on *trust* in conjunction with *similarity*. For this experiment, trust was computed using the *CITEM* algorithm from [9], and similarity was computed in the usual way, using Pearson's correlation over the raw rating data.

Figure 5 shows the *PeerChooser* application displaying trust and correlation in parallel. In this personalized graph, the active user is positioned at the center with a personalized avatar. Non-active nodes are positioned around this, again with edge length fixed proportional to the Pearson correlation between the users. Node size is a function of trust – smaller icons are less trustworthy and larger ones have higher trust. Using this graph the active user can easily discern the differences between similar users and trustworthy users. In this example, a highly trusted neighbor can be seen just below the active user's avatar. This neighbor is also highly correlated due to the close proximity to the active user node. Attached to the trusted peer is an icon representing the "horror" genre. In the right panel the top-n recommendation list contains the slightly esoteric movie "Hellraiser" in the top three, with a prediction of 4.1 from 5. This is most likely the influence of the trusted and correlated peer who likes horror movies.

D. Visualisation-Based Prediction

In the previous section we focused on PeerChooser's visualization interface, which allows the user to manipulate their direct neighbors and genre preferences; these manipulations are the user's *hints* for the recommender. In this section we will explain how these hints translate into actual recommendation influences. In *PeerChooser* the k nearest (user) nodes are selected as the neighborhood for the purpose of recommendation; these are the more shaded nodes in Figures 2 and 4. Our benchmark CF prediction algorithm uses Resnick's formula

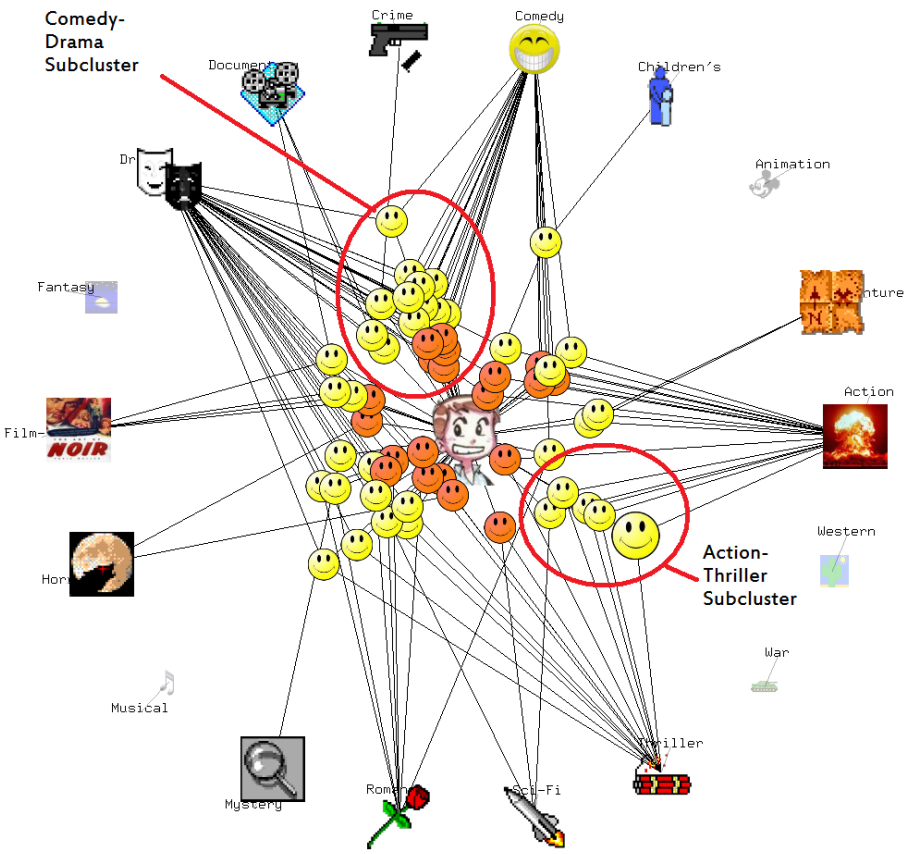

Fig. 4. *PeerChooser* multiple genre associations.

which is reproduced below as Equation 2; see also [11]. In this formula $c(i)$ is the rating to be predicted for item i for an active profile c (the user receiving the recommendation), and $p(i)$ is the rating for item i by a peer profile p who has rated i. $\overline{c}$ and $\overline{p}$ refer to the mean ratings for c and p respectively. The weighting factor $sim(c,p)$ is a measure of the *similarity* between profiles c and p, which is traditionally calculated as Pearson's correlation coefficient and has a range of -1 to +1. In our evaluation section we test the performance of this standard benchmark algorithm against our visualization-based approaches, which allow user hints to influence the similarity values used during recommendations as we shall now discuss.

$$c(i) = \overline{c} + \frac{\sum_{p \in P(i)} (p(i) - \overline{p}) sim(c,p)}{\sum_{p \in P_i} |sim(c,p)|} \quad (2)$$

To incorporate user hints into the recommendation process we simply replace the standard similarity values (based on user-user ratings correlations) with a new similarity value that is based on the inverse *Euclidean distance* between the active user node and each of the k peer nodes that have been manipulated by the user. This is our *ephemeral similarity* value and is given by Equation 3. Here, Euclidean distance between pixels on the graph is normalized to the Pearson's correlation range of (-1, +1), max_dist is the maximum possible distance between the active user node and a peer node, while $node\_dist$ is the distance between the active node (ie: the center of the graph) and each peer node. Equation 4 shows the original Resnick prediction formula using ephemeral similarity in place of the standard Pearson correlation. The nomenclature is similar to that in Equation 2 with $c(i)$ being the predicted rating for an active user c on item i.

$$eph\_sim(c,p) = (2(1 - \frac{node\_dist}{max\_dist}) - 1) \quad (3)$$

$$c(i) = \overline{c} + \frac{\sum_{p \in P(i)} (p(i) - \overline{p}) eph\_sim(c,p)}{\sum_{p \in P_i} eph\_sim(c,p)} \quad (4)$$

IV. Evaluation

The majority of experiments involving recommender system algorithms are based on some form of automated testing, for example, predicting ratings for some "hidden" subset of the rated data. This is only possible in the absence of

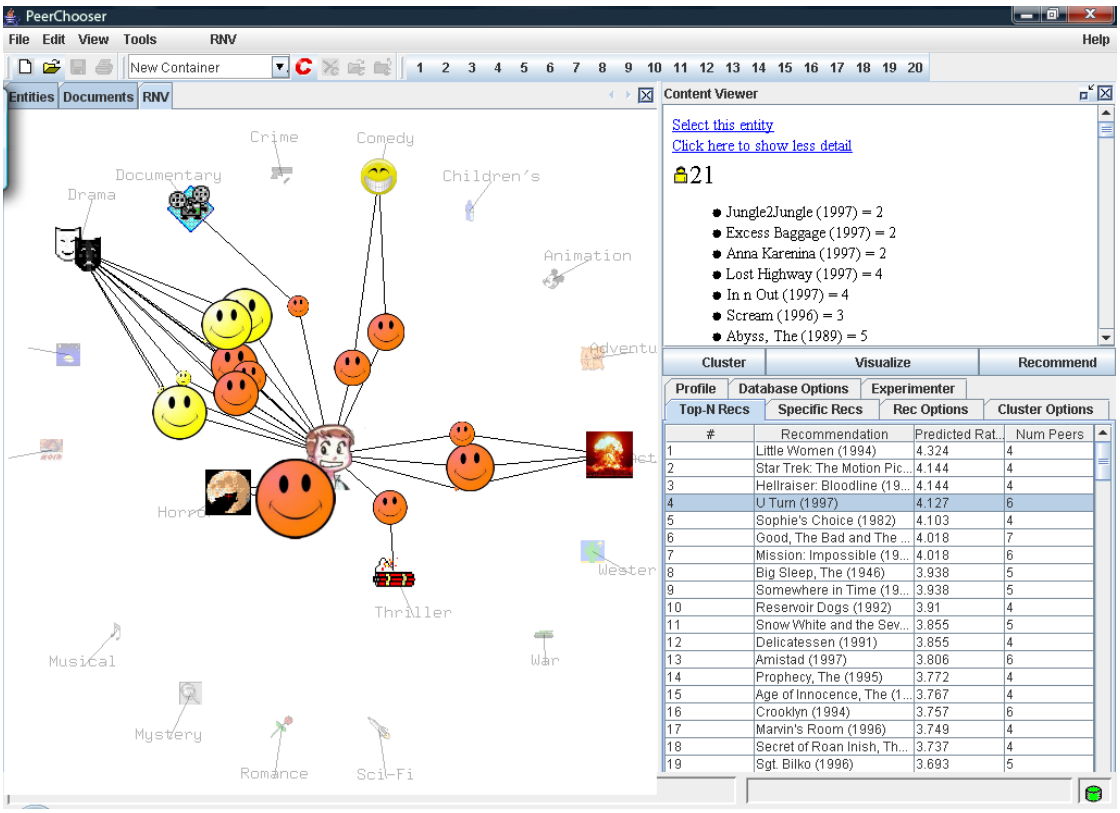

Fig. 5. The *PeerChooser* OpenGL application showing trust values as node size and correlation as edge length.

interactive components such as the ones of our *PeerChooser* system. Visualisation and interaction are additional "tiers" to the process of collaborative filtering. The visualization and interaction tiers bring many obvious advantages, however, it does prohibit the standard automated testing procedure since real people must explore the graph and interact with it to obtain results.

A. Procedure

To assess users' opinions of *PeerChooser's* interactive visualizations and to assess predictive accuracy we conducted an evaluation with twenty five participants which examined four techniques for generating recommendations with *PeerChooser*, two with interaction (hints) and two without. The source data for this survey was the smaller MovieLens[8] dataset, which contains 943 user ratings on 1682 items. Participants were asked to generate recommendations using the techniques listed below. Three values per recommendation were recorded: the predicted rating, the actual rating, and the average rating for that item across the entire database.

1) *Average Layout* - (non-interactive) The graph was laid out based on an "average user". This profile was created by taking the average rating for the 50 most rated items. Participants were then asked to rate recommendations generated for this user. This technique was expected to yield the worst results as it contained no personalized information.
2) *Profile-Based Layout* - (non-interactive) This is our benchmark CF algorithm. Users rated 30 items in the right hand panel. Correlations computed from these were used to generate predictions.
3) *Profile-Based Layout with Manipulation* - (interactive) Same as above but the user can manipulate the graph to provide information on current requirements.
4) *Profile-Based Layout with Manipulation and Feedback* - (interactive) Same as above except the user receives dynamic recommendations on their salient items with each graph movement. We expected this to exhibit the best performance.

In all cases, the system's predicted rating was not shown to participants until after they had provided their rating for each predicted item. This follows from work by Swearingen et al. in [13] which suggests that users tend to rate towards the machine-provided ratings. In all cases the number of neighbors was set to k=30, an optimal value for CF on our dataset reported in [6]. For associating or *clustering* non-active user nodes to genre nodes we used the scaling function from Equation 1 with a liked-item threshold of 3 (ie: liked-items had a rating of 4 or 5). For each test, the 400 most similar users based on Pearson's correlation were displayed on the graph. In all tasks where the user interacted with the graph,

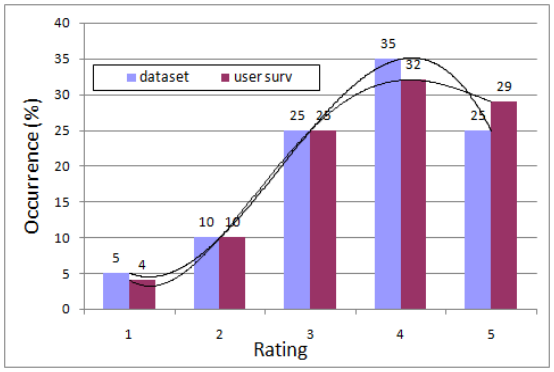

Fig. 6. Comparison of ratings distributions between existing MovieLens data and data from the user trials.

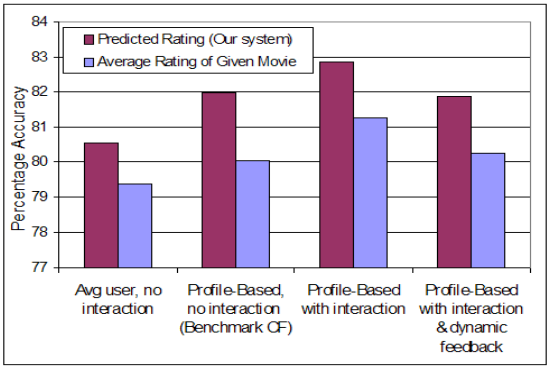

Fig. 7. Error results for each technique in the user trials compared with average predictions for each item recommended.

predictions were made using our distance-based version of Resnick's prediction formula (given by Equation 4); in all others the standard Resnick prediction formula (Equation 2) was used.

In the final task, *Profile-Based Layout with Manipulation and Feedback* the graph layout was as in the previous tasks, built from correlation over the profile data. Users were asked to select checkboxes next to the 5 movies they really liked and the 5 that they disliked. Users were then told to select a button marked "specific predictions" see the benchmark CF predictions on those items. Users were then told to take some time to manipulate the graph of connected peers to try and tweak the recommendations for their chosen items to a value that most suited their needs and preferences. With no time constraints imposed, all users in the survey reported that they had arrived at a satisfactory position within 2 mins. With each interaction users were provided with dynamic recommendations based on their "salient" item-set and according to the current neighborhood configuration. This allowed the users to dynamically assess the goodness of the evolving neighborhood space as each interaction was performed. When a user becomes satisfied with predictions generated from the "tweaked" graph on the salient item sets, a list of top-n recommendations was presented and the user was asked to rate them individually.

#	*QuestionDescription*
S1	I am familiar with recommender systems
S2	I am familiar with interactive computer graphics
S3	I am familiar with graph visualizations
S4	Which did you think was the most accurate
S5	I found the graph easy to understand
S6	Look at the 4 graphs and rate each one
S7	Did you prefer the visualization approach
S8	I gained knowledge about the underlying data from the visualization
S9	I felt that the labeling was appropriate
S10	I felt that the information in the right panel was helpful
S11	I felt that the *visualization* system gave me more control of the recommendation algorithm
S12	Would you like to see this interface on other domains (eg: Amazon.com)
S13	I felt that I benefitted from *interaction* with the system

TABLE I
LIKERT SCALE QUESTIONS FROM THE USER SURVEY.

This list did not contain any items previously rated by the user.

B. Recommendation Accuracy

To evaluate the accuracy of the techniques, mean absolute error was computed between the predicted rating and the user's actual rating for each of the methods. Results are presented in Figure 7 for each of the four techniques. As expected, the average predictions– that is, predictions based on the average user described earlier, exhibited the worst performance, producing accuracy of 80.5%. Predictions based on an average user (column 1 in Figure 7) have high accuracy, this is not surprising if we take a look at the rating distribution graph in Figure 6. Users tended to rate only movies they liked, and the average user was constructed from a list of the most commonly rated movies, which as it turns out were generally the most highly rated movies. This may be attributed to the fact that users tended to remember older movies in a good light. Our profile-based technique (column 3) with manipulation beats the benchmark (column 2) achieving a small relative increase of 1.05%. P-tests indicate that these differences are significant in each case with p <0.01. This small increase is an important result because it indicates that current information does help the recommendation process. Future work includes conducting this experiment with the more modern Netflix dataset and a larger user-base through a fully web-based user study.

The most surprising result was that the dynamic feedback technique performed worse than the other profile-based techniques. After much analysis of the graphs it was determined that users tended to *over-tweak* the system to achieve desired results for their salient item sets. In doing this, much of the existing correlation information was reduced and the resulting layout was overfitted to the specific item sets. A solution to this may be to ensure diversity within the salient item sets.

C. User Satisfaction

To assess the effects of user interaction with the system a pre and post study questionnaire was answered by each participant.

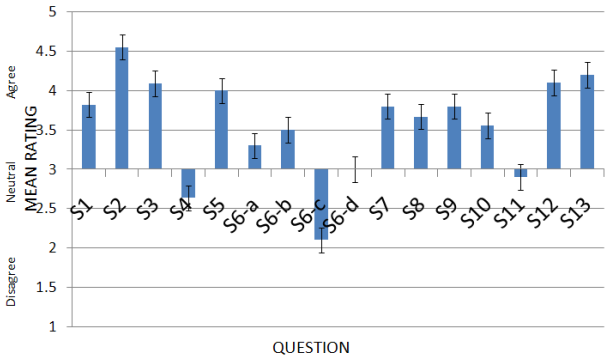

Fig. 8. Results of the Post-Study Questionnaire from the User Trials.

Table 1 lists each question from the survey and references the columns in Figure 8. $S1$ to $S3$ indicate that all participants had experience with graphical interfaces, recommenders and visualizations. $S4$ tells us that, in contrast to our empirical accuracy tests, users felt that manual rating provided more accurate results. This was an interesting result which may indicate that users are more comfortable with familiar, manual rating systems. The four columns marked $S6$ represent participants' opinions on a range of different graph representations of the data, with column 8 being an exceptionally poor display. $S7$ shows a clear preference for the visualization approach, with 75% of users preferring *PeerChooser* over the traditional approach. The benefit of our technique as a recommendation *explanation* is shown by $S8$, where the majority of users felt they gained knowledge of the data from their interaction with the graphs. $S9$ and $S10$ indicate that users felt that labeling and node information were appropriate. $S11$ shows us that there were mixed views about the control that the interface provided on the CF algorithms. This response may be due to insufficient familiarization time, since the technique does provide more access points to influence the CF process. $S12$ shows that more than 80% of participants agreed that they would like to see a *PeerChooser*-like interface on other domains such as Amazon.com. This is an encouraging result, given that there is a broad scope of applications for our technique. More importantly, $S15$ shows that more than 80% of participants felt they benefitted overall from interacting with the system.

V. CONCLUSIONS

Traditionally collaborative filtering systems have relied heavily on similarities between the ratings profiles of users as a way to differentially rate the prediction contributions of different profiles. We have shown how interactive visualization techniques can be incorporated into the mechanics of a standard collaborative filtering algorithm to greatly expedite the process of profile generation, helping to ease the "cold start" problem that exists in most collaborative filtering systems. Visualization enhances the user experience by providing a user with at-a-glance information about the underlying structure of the data upon which predicted ratings are generated. Interaction with the visualization affords the user the opportunity to express facets of current mood and requirements.

This paper evaluated our visualization-based approach against a benchmark collaborative filtering algorithm using mean error accuracy tests. A live user survey was carried out to assess a range of aspects of user interaction with the system from a HCI and predictive accuracy perspectives. Accuracy experiments indicate that there is a benefit to the visualization approach to profile generation for collaborative filtering over each of the four techniques tested.

VI. ACKNOWLEDGMENTS

This research was supported in part by Science Foundation Ireland under Grant No. 07/CE/I1147 as well as by NSF grants IIS-0840585, CNS-0722075, and IIS-0808772.

REFERENCES

[1] Jill Freyne, Rosta Farzan, Peter Brusilovsky, Barry Smyth, and Maurice Coyle. Collecting community wisdom: integrating social search & social navigation. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 52–61, New York, NY, USA, 2007. ACM Press.

[2] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Computer Supported Cooperative Work*, pages 241–250, 2000.

[3] Cliff Lampe, Nicole Ellison, and Charles Steinfield. A face(book) in the crowd: social searching vs. social browsing. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 167–170, New York, NY, USA, 2006. ACM Press.

[4] Hugo Liu, Pattie Maes, and Glorianna Davenport. Unraveling the taste fabric of social networks. *The Media Laboratory, Massachusetts Institute of Technology*.

[5] Sean M. Mcnee, Nishikant Kapoor, and Joseph A. Konstan. Don't look stupid: avoiding pitfalls when recommending research papers. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 171–180, New York, NY, USA, 2006. ACM Press.

[6] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations, 2002.

[7] Stuart E. Middleton. Exploiting synergy between ontologies and recommender systems.

[8] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM Press.

[9] John O'Donovan and Barry Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174. ACM Press, 2005.

[10] John O'Donovan, Barry Smyth, Brynjar Gretarsson, Svetlin Bostandjiev, and Tobias Höllerer. Peerchooser: visual interactive recommendation. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1085–1088, New York, NY, USA, 2008. ACM.

[11] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, Sharing Information and Creating Meaning, pages 175–186, 1994.

[12] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285–295, 2001.

[13] Rashmi Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI '02 extended abstracts on Human factors in computing systems*, pages 830–831. ACM Press, 2002.

[14] LLC TouchGraph. Touchgraph available at www.touchgraph.com.

[15] Cai-Nicolas Ziegler, Sean M. Mcnee, Joseph A. Konstan, and Georg Lausen. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, New York, NY, USA.