Mining Trust Values from Recommendation Errors

John O'Donovan, Barry Smyth
Adaptive Information Cluster
School of Computer Science and Informatics
University College Dublin
Ireland

john.odonovan@ucd.ie, barry.smyth@ucd.ie

Increasing availability of information has furthered the need for recommender systems across a variety of domains. These systems are designed to tailor each user's information space to suit their particular information needs. Collaborative filtering is a successful and popular technique for producing recommendations based on similarities in users' tastes and opinions. Our work focusses on these similarities and the fact that current techniques for defining which users contribute to recommendation are in need of improvement.

In this paper we propose the use of trustworthiness as an improvement to this situation. In particular, we define and empirically test a technique for eliciting trust values for each producer of a recommendation based on that user's history of contributions to recommendations.

We compute a recommendation *range* to present to a target user. This is done by leveraging under/overestimate errors in users' past contributions in the recommendation process. We present three different models to compute this range. Our evaluation shows how this trust-based technique can be easily incorporated into a standard collaborative filtering algorithm and we define a fair comparison in which our technique outperforms a benchmark algorithm in predictive accuracy.

We aim to show that the presentation of absolute rating predictions to users is more likely to reduce user trust in the recommendation system than presentation of a range of rating predictions. To evaluate the trust benefits resulting from the transparency of our recommendation range techniques, we carry out user-satisfaction trials on BoozerChoozer, a pub recommendation system. Our user-satisfaction results show that the recommendation range techniques perform up to twice as well as the benchmark.

Keywords:
Recommender systems, collaborative filtering, profile similarity, trust

1. INTRODUCTION

A large amount of research effort has been directed at the well-known problem of information overload, whereby the task of locating relevant information quickly is becoming ever more arduous with increasing volumes of online information [20],[21]. This is the domain of Recommender Systems, which are deployed in a variety of applications to assist the user in locating the right information at the right time. These systems can be found in diverse areas such as virtual shopping assistants [6], restaurant or movie recommenders [4], TV program recommendations [22] etc. Recommenders have proven themselves successful at harnessing information normally reserved for social interaction, and leveraging this to provide tailor made solutions for individual users based on the tastes and opinions of similar users. To achieve this solution, recommenders employ ideas from research areas such as user-profiling, information filtering and machine learning.

The majority of research into recommendation strategies focusses on two main approaches. Content-based recommenders [3],[4],[13] operate by comparing textual descriptions

of recommendable items (eg, products or services). Therefore, a content-based movie recommender requires movies to have adequate textual descriptions before they can be incorporated into the system. The content-based approach takes information such as genre, actors, director etc. and matches this against the learned preferences of the user to arrive at a suitable recommendation. These systems rely heavily on knowledge engineering, are not suited to domains without a high quality textual description, and do not facilitate serendipitous recommendations, in that the system will always try to recommend items that are similar to those seen and liked before. This is not an accurate reflection of the changing tastes of real world users.

The alternative approach to the recommendation problem is a collaborative one. [7],[20], [15] The collaborative filtering (CF) approach attempts to model the real-world scenario of looking to friends for recommendations [20]. In this way, the need for specific item-knowledge is eliminated, giving this approach a much more broad and flexible application base. CF systems rely on the availability of rich user profiles containing lots of rating data, so that sufficient profile overlap can be attained to build up user peergroups. Peers or *recommendation partners* for a target user are generated by choosing profiles that have similar or highly correlated ratings histories with that user.

This work focusses on the CF approach to the recommendation task, in ways to improve the accuracy of predictions made by CF approaches, and in ways to increase user-satisfaction and trust in these predictions. We propose to modify the selection process for recommendation partners. Currently this selection is usually based on a profile-profile similarity metric, calculated using some similarity function such as Pearson's correlation coefficient [20]. We introduce a second metric of *trustworthiness* upon which to base this selection decision. For example: People look to their friends for recommendations about items, but some of these friends might not know a lot about that particular item, or may have an esoteric or uncharacteristic opinion about that item, which leads him/her to give consistently skewed or bad recommendations about that particular item. Our proposal is that a recommendation partner should be both similar to the target user, and trustworthy in that the contribution that they have made to previous recommendations has been a positive one. We can model this trustworthiness both at the profile level and at the item level. The former allows us to say how trustworthy a recommendation producer is on a general level, while the latter allows us to provide a specific trust score for a producer with respect to the particular item that user is involved in recommending. For our evaluations in this paper, we use recommendation error history to elicit these trust values. We describe three methods of incorporating these values into a regular CF algorithm, and show results of an empirical comparison of these techniques. We also show that our trust-based recommendation strategy outperforms a standard benchmark system which uses Resnick's prediction algorithm [20] over a standard CF dataset, and that the resulting output is more trusted by users in live trials.

2. BACKGROUND

Trust, reputation and reliability are factors that influence decision-making across the board. Recent research in social networking, virtual communities and recommender systems has focussed on the issue of trust. The majority of recommender system research is carried out on carefully compiled experimental data and does not factor in real world problems such as malicious users. The increasing interest in issues of trust has raised some conflicting opinions on not only its application and relevance, but even its basic definition. Marsh's work in [10] goes some way towards formalising trust in a computational sense, and highlighting different definitions and applications of trust from both a social and technological viewpoint.

2.1. *Trust in Recommender Systems*

There are a number of recent research publications which deal with the issue of how to use trust and reputation models in the recommendation process. This paper focusses on

ways to automatically infer trust and reputation from ratings data (using a history of errors in contributions to recommendations), and on ways to integrate these models into the recommendation process to produce more reliable recommendations. Other research has focussed on a related issue, but using more invasive methods to acquire trust and reputation models. For example, the work of Massa et al. [2,11] builds a trust model directly from trust data provided by users as part of the popular *Epinions.com* service. *Epinions.com* is a web site that allows users to review various items (cars, books, music, etc.). In addition they can assign a trust rating to reviewers based on the degree to which they have found them to be helpful and reliable in the past. Massa argues in [11] that this trust data can be extracted and used as part of the recommendation process, especially as a means to relieve the sparsity problem that has hampered traditional collaborative filtering techniques. The sparsity problem refers to the fact that on average two users are unlikely to have rated many of the same items, which means that it will be difficult to calculate their degree of similarity and so limits the range of recommendation partners that can participate in a typical recommendation session. It is proposed in [11] that it is possible to compare users according to their degree of connectedness in the trust-graph encoded by *Epinions.com*. The basic idea is to measure the distance between two users in terms of the number of arcs connecting the users in the trust-graph encoded by the *Epinions.com* trust data. They show that it is possible to compare far more users according to this method than by conventional forms of ratings similarity and argue that because of this, trust-based comparisons facilitate the identification of more comprehensive communities of recommendation partners. However, it must be pointed out that while the research data presented does demonstrate that the trust data makes it possible to compare far more users to each other it has not been shown that this method of comparison maintains recommendation accuracy.

Montaner et al. [14] contemplate the availability of large numbers of virtual recommendation agents as part of a distributed agent-based recommender paradigm. Their main innovation is to consider other agents as personal entities which are more or less reliable or trustworthy and, crucially, that trust values can be computed by pairs of agents on the basis of a conversational exchange in which one agent solicits the opinions of an other with respect to a set of items. Each agent can then infer a trust value based on the similarity between its own opinions and the opinions of the other. Thus this more emphasises a degree of proactiveness in that agents actively seek out others in order to build their trust model which is then used in the *opinion-based* recommendation model. This approach is advantageous from a hybrid recommender perspective, in that agents can represent individual techniques and they can be combined using opinions based on trust.

2.2. *Trust and Recommendation Interfaces*

In this work we consider the effects a recommendation system interface has on the user. In [21], Sinha and Swearingen present a user-study of five music recommender systems to assess the effects that look and feel of a recommender has on user-satisfaction. They found that users are generally more confident when the system is transparent. The work of Cosley et al. in [5] is especially relevant, showing (through a technique called "re-rating") that users' opinions can actually be manipulated by a recommender system. They present evidence that users tend to rate toward the prediction that the system shows. Cosley et al. [5] also presents an experiment to examine users' reactions to different prediction displays. They find that users tend to rate fairly consistently across different scales.

3. A Computational Model of Trust

As with previous work in [17] we define two separate profile types in the recommendation process: A *consumer* profile refers to one receiving the item rating, whereas a *producer* refers to a profile that has been selected as a recommendation partner (a contributor to

the recommendation) for the consumer and that is participating in the recommendation session. So, to generate a predicted rating for item i for some consumer c, we will typically draw on the services of a number of producer profiles, combining their individual recommendations according to some suitable function, such as Resnick's formula, for example. (see Equation 1). Our benchmark algorithm uses this standard prediction formula (see also [20].) In this formula $c(i)$ is the rating to be predicted for item i in consumer profile c and $p(i)$ is the rating for item i by a producer profile p who has rated i. In addition, $\overline{c}$ 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. One advantage of using this common CF algorithm is that we can compare our approach more easily with existing systems, as seen in our work in [17].

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - \overline{p}) sim(c,p)}{\sum_{p \epsilon P_i} |sim(c,p)|} \tag{1}$$

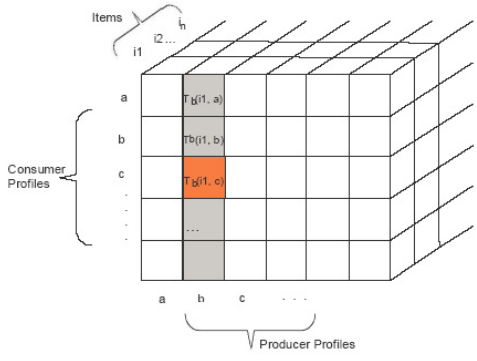

Fig. 1. Calculation of Error Scores from Rating Data.

3.1. *Eliciting Trust From Recommendation Error*

Standard collaborative filtering algorithms use some similarity function such as Pearson's correlation to compute the similarity between user profiles in the system. We don't believe that the standard method alone is sufficient to produce optimal recommendations. We propose that there is another important element which must be considered in the recommendation process. We refer to this as *trust*. In everyday terms, if a person has successfully recommended lots of items in the past, they should be awarded more trust than someone who has consistently made poor recommendations in the past.

We define two distinct levels of trust. *Item Level* trust is a score for a producer's trustworthiness with respect to the recommendation of specific items. Eg: "John's trustworthiness for recommending a Toyota Corolla". This is a context-specific trust, as defined in [1]. *Profile Level* trust is a less fine-grained metric, representing a recommendation producers trust as a whole, without respect to specific items. Eg: "John's trustworthiness". This score is simply an average over the Item Level trust scores for every item in the users profile.

In our work in [17] we calculate trust values at both the Item and Profile levels. Essentially these values summarise the proportion of correct recommendations that a producer has been involved in, according to a pre-defined error bound.

Here, we compute profile and item level error in a similar manner. Consider some producer profile p and some arbitrary item i which has been rated by p. We calculate the average error on p's contributions to recommendations of item i over a training set of profiles. Each ratings prediction for an item, i, by producer p for a consumer c, is within a finite error ϵ of c's actual rating $c(i)$; see Equation 4.

In Resnick's approach to CF, producers involved in recommendation would be operating in tandem with a number of other recommendation partners, so it may not be possible to assess whether or not producer p's contribution had a positive or negative effect on the final recommendation. To overcome this problem we isolate p's contribution to the overall recommendation by making p the sole recommendation partner for consumer c for a recommendation on item i. For example, in Figure 1, error scores for item *i1* are generated for producer b by using the information in profile b *only* to generate predictions for each consumer profile.

$$T_p i, c = |p(i) - c(i)| = \epsilon \tag{2}$$

Equation 2 shows how each box in Figure 1 contains a separate error score based on each individual profile p being used as the sole recommendation partner for each consumer c and item i.

$$RecSet(p) = \{(c_1, i_1), ..., (c_n, i_n)\} \tag{3}$$

The recommendation set for a producer profile p on a specific item i is show in the shaded area of Figure 1, and given by Equation 3. It is worth noting that in a deployed recommender system operating in real time, these values can be generated on-the-fly as new ratings are made in the system. For our experiments, we record 6 different error metrics which we show in the next section can be readily incorporated into the mechanics of a generic CF system to produce a *recommendation range* to be presented to the user.

(1) *PError* - The average prediction error over any time p has been involved in producing a recommendation.

(2) *IError* - The average prediction error over any time p has been involved in producing a recommendation for each item i.

(3) *PUnderest* - The average underestimate over all the times p is involved any recommendation, and there has been an underestimate in the prediction.

(4) *IUnderest* - The average underestimate over all the times p is involved in recommendation of each specific item, and there has been an underestimate in the prediction.

(5) *POverest* - The average overestimate over all the times p is involved in recommendation of any item, and there has been an overestimate in the prediction.

(6) *IOverest* - The average overestimate over all the times p is involved in recommendation of each specific item, and there has been an overestimate in the prediction.

$$IError(p, i) = \frac{\sum_{i \in R} |p(i) - c(i)|}{n} \tag{4}$$

$$PError(p) = \frac{\sum_{i \in P} (IError)}{n} \tag{5}$$

Equation 4 shows our calculation of the basic error for producer profile p at the item level (for item i). n represents the number of recommendations used in the calculation. $p(i)$ is the rating predicted on item i using producer p as the sole recommendation partner for consumer c. $c(i)$ is the actual rating which consumer c gave to item i. This is a fine grained approach to trust value elicitation. A more coarse metric $PError$ is given by Equation 5, which denotes the average $IError$ over all the items in profile p.

$$IUnderest(p, i) = \frac{\sum i \in Rp(i) - c(i)}{n} : ifp(i) - c(i) > 0 \tag{6}$$

$$IOverest(p,i) = \frac{\sum i \in Rp(i) - c(i)}{n} : if p(i) - c(i) < 0 \quad (7)$$

The case statement in Equation 6 defines the computation of average underestimate at the item level. Here, n represents the number of times there was an underestimate in the predictions generated using producer p for consumer c on item i. A similar computation is denoted in Equation 7 to get the average overestimate in p's contributions to recommendations of individual items.

To calculate these values at the profile level, we use Equation 8 which is simply an average of the item level underestimates over every item in profile p.

$$PUnderest(p) = \frac{\sum i \in P(IUnderest)}{n} \quad (8)$$

We have shown the processes involved in building our trust models based on recommendation error histories. The next step is to demonstrate how these values are manifested in the recommendation process to arrive at a more reliable and transparent recommendation solution.

3.2. *Using Error Ranges in Recommendation*

As a result of these calculations, for every producer profile and every item in these profiles we have the producer's normal rating for this item plus the average degree to which this user tends to underestimate ratings, the average degree to which he overestimates ratings, and the overall average error for this producer and item pair. Now, how can we use this in recommendation? One obvious direction to take is to produce a *recommendation range*.

For each time we wish to predict a rating for some new target user t and item i we get 3 ratings values: r_1, r_2, r_3 which basically gives us a rating range [r_2, r_3] and an expected rating, r_1, rather than just a single rating. For example, we can now say to the target user: "My predicted rating for item i is 3.2, but no less than 2.5 and no more than 4" This is showing more information to the target user. Intuitively, this is a better recommendation strategy than simply predicting a set individual value for the user.

We propose three such recommendation strategies:

3.2.1. *Addition and Subtraction*

This is the approach loosely described above. For each recommendation r we generate for an item i, we produce r_1, r_2, r_3 respectively in the form of Equation 9

$$R = \{(r - avg(IUnderest))...r...(r + avg(IOverest))\} \quad (9)$$

The second addition/subtraction approach is more basic and used as our benchmark in comparison with the other techniques. In this approach, instead of using the average under and overestimates, we simply produce our recommendation range by discounting the standard rating by the average error to get r_1 and increment the standard rating by the average error to arrive at our upper bound r_3. This is shown in Equation 10

$$R = \{(r - PError)...r...(r + PError)\} \quad (10)$$

In this equation, R is the recommendation range and r is the standard Resnick rating.

3.2.2. *Modified Resnick Approach*

A more interesting way to produce a recommendation range from our error values is to use them *inside* the standard Resnick prediction formula. The basic equation is given in 1, and our modified version in 11, below. Instead of discounting the underestimates after a recommendation has been produced (as with the approach above), we discount the underestimates from all of the neighbours ratings as they are calculated in the standard

formula. This produces a lower-bound rating r_u. A similar approach is used to increment neighbours ratings by their pre-calculated average overestimate to produce our upper-bound value r_o.

$$r_u = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - IUnderest(p,i) - \overline{p}) sim(c,p)}{\sum_{p \epsilon P(i)} |sim(c,p)|} \tag{11}$$

The formula for computing r_o, is that in 11 but with a $+IOverest$ in place of the $-IUnderest$. These results yield another recommendation range:

$$R = \{(r_u)...r...(r_o)\} \tag{12}$$

4. EVALUATION

In this work we have forwarded a new technique for building a model of user trust based on their history of errors in contribution to recommendations. We have proposed several techniques for integrating these into a standard recommendation algorithm as an additional metric to be considered along with a standard correlation function. In our discussion we will consider the general benefits of producing a recommendation range using these error metrics. The following section describes a specific set of experiments designed to provide us with a better grasp on the empirical improvements our technique can make to the standard CF prediction strategy.

4.1. *ACCURACY TESTS*

In this experiment we use the standard MovieLens dataset [20] which contains 943 profiles of movie ratings. Profile sizes vary from 18 to 706 with an average size of 105. We divide these profiles into two groups: 80% (753 profiles) are used as the producer profiles and the remaining 20% (190 profiles) are used as the consumer (test) profiles.

Our implementation falls into two phases: Firstly, building the trust-error model, and then using the model in recommendation.

4.1.1. *Building the Error-Trust Model*

In a deployed recommender system, error values can be computed in real time, as users enter ratings to the system. In this experiment however, we build the trust values separately. For each producer profile in our training set, we temporarily assign that profile to position of *consumer*, and using the standard Resnick prediction formula, we generate predictions for that consumer, using each of the other producer profiles in turn as the sole recommendation partner. We assess the proximity of the predicted ratings to the actual ratings by employing a standard leave-one-out test, where we hide n% of the consumers profile and try to predict the hidden ratings. We compare the predicted ratings to the known real rating and record the error. These errors fall into either underestimate or overestimate categories.

This procedure is carried out for every producer-item pair in the system, and the result is an average error for each producer-item pair over all of the times that producer was involved in the production of a recommendation of that item. This is our item-level model. In order to get our more coarse grained profile-level model, we average these error values again, but this time over each item in a producers profile. This provides us with an average error per producer (profile level error).

Table 1 gives us an overview of the type of recommendation errors we found for each of our strategies. Noting that IError and PError values are for overall average error at the item and profile levels respectively, and do not necessarily have to be the sum of the mean individual under and overestimates at that level. The largest average error is at the item level at 0.937 on the Movielens rating scale of 1 to 5. This is as expected from our recommendation results in the next section, in which the recommendation ranges

Table 1. Analysis of Recommendation Error Data in MovieLens.

Algorithm	Average (0-5)	Maximum (0-5)	Minimum (0-5)
IError	0.938	3.489	0
PError	0.943	1.804	0.679
IOverest	0.865	3.833	0
IUnderest	0.745	3.489	0
POverest	0.931	2.217	0.328
PUnderest	0.868	1.865	0.162

produced based on these errors outperforms the others in accuracy tests, most likely due to the broader range produced from these values. We note that the profile level errors are much less extreme, *POverest*, for example having a minimum error of 0.328 in comparison with minimum *IOverest* at 0 error. This result is exactly as expected since our profile level errors are averaged over every item in a users profile. A similar trend is the case for the other profile / item level comparisons.

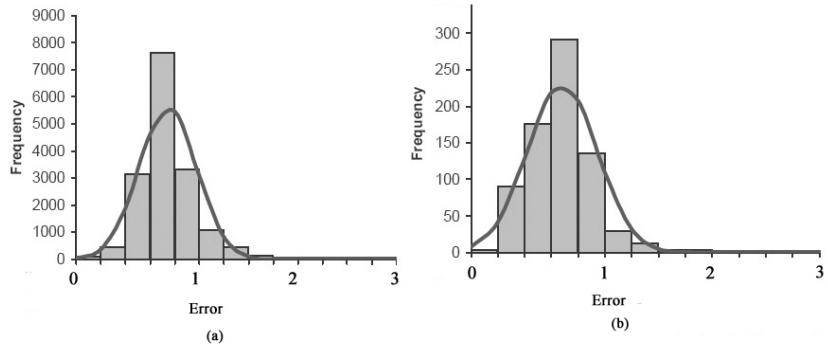

Fig. 2. The distribution of producer error values. (a) Item-level distribution, (b) Profile-level distribution.

Figure 2 depicts the distribution of the error values at item and profile levels respectively. Both follow a normal distribution. Profile level errors are centered more closely around the mean of 0.94, having a standard deviation of 0.13, compared with 0.3 around a mean of 0.94 for the item level errors. This variance should have a notable effect when they are used to generate our recommendation ranges in the next section, yielding potentially more broadness for the item level approach. Space restrictions prevent us from showing histograms for each individual technique described, but the others also follow normal distributions, with a similar variance between profile and item levels.

4.2. *Predictive Accuracy Experiment.*

The overarching goal of this research is to improve predictive accuracy for standard CF based recommendation algorithms. This experiment looks at the accuracy of our technique from a % correct perspective. For the 190 profiles in our test set, we generated recommendation ranges for each rated item using the three techniques described in section 3, using both item level and profile level error values separately for our modified

resnick approach, and average error value for the addition / subtraction approach. For this set we defined accuracy as a consumers actual rating falling within the predicted recommendation range. Note that training and test data are completely independent sets. Results of this experiment are presented in Figure 4. It is clear that the modified resnick approach using item level error values outperforms the other techniques, beating its closest rival $avgErr$ by 4.5%. This may be due to the greater variance in the item level errors. The worst performer in fact is the modified resnick approach using profile level errors, at 14% worse performance than its item level counterpart. This suggests that the variance in errors does have a notable effect on predictive accuracy for collaborative filtering.

4.3. *Comparison with a Benchmark Resnick Algorithm*

Due to the *recommendation range* nature of our predictions, we must define a new method to evaluate the performance of our technique with respect to a benchmark (Resnick) algorithm. To this end, we define a scalar constant which is the average of the absolute differences between our upper bound r_o and our lower bound r_u, (which is 1.85). We propose that a fair comparison would be to assume a standard Resnick prediction to be 'correct' if it falls within half of this distance either side of the real rating. This is only a pseudo-comparison, as it is impossible to compare these techniques directly.

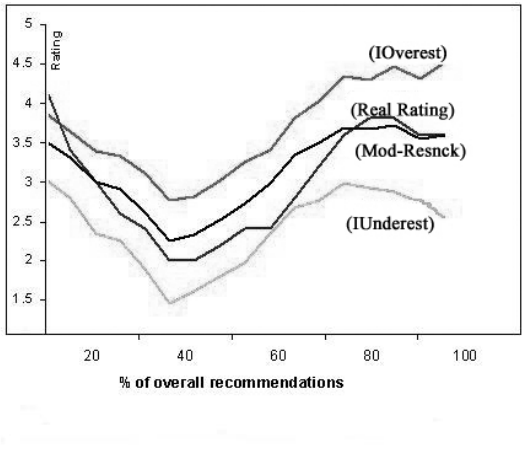

Fig. 3. Distribution of the recommendation ranges and actual ratings for the modified Resnick technique.

Figure 3 is a trend graph of the of the recommendation ranges for our best performer, the modified Resnick approach using item level errors. Here we can see the upper bounds r_o and lower bounds r_u of the recommendation range generated by using $IUnderest$ and $IOverest$ respectively in the standard Resnick prediction formula, see Equations 1 and 11. The real ratings provided by our test profiles tend to remain within the range, even when the standard Resnick prediction tends away from the real rating. This outcome has a positive effect not only on predictive accuracy, but also on users overall trust in the recommender system as a whole, reducing the mal-effects of false positives in recommendations.

From Figure 4, it is clear that the range technique performs 15% better than the benchmark using our fair comparison. This does show that the direction in which the error models pull the r_o and r_u ratings is having a positive effect on predictive accuracy.

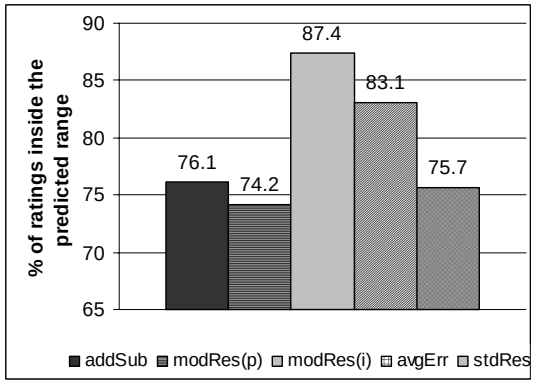

Fig. 4. % Ratings inside recommendation range.

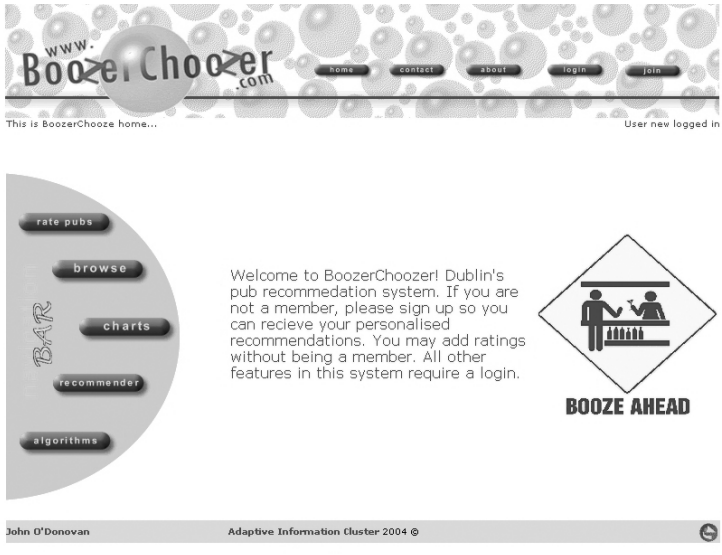

Fig. 5. Introductory Screenshot of the BoozerChoozer System.

5. Evaluation

Empirical evaluations of our trust-based recommendation strategies have been presented, and it was shown that there are performance benefits on a static dataset. We further our evaluation by presenting an assessment of the performance of our algorithms in a live system. The previous section defined a "fair" comparison with a standard, single-point recommendation algorithm. We note however that a defining a true fair comparison

between a discrete recommendation and a continuous recommendation range is a difficult task.

In this section, we focus on assessing the impact of recommendation ranges on real world users. Intuitively, the more transparent a recommender system is, the more a user would be inclined to trust output from that system, provided of course that the system is basing its recommendations on reasonable assumptions. The effects of transparency on user satisfaction in recommendation systems has been dealt with extensively in work by O'Mahony et al [19], and Burke [4], and it has been shown by Sinha and Swearingen in [21] that "users like and feel more confident about recommendations they perceive as transparent." It is not possible to empirically evaluate the effects of providing recommendation ranges to users directly, without defining assumptions such as our comparison in the previous section. To overcome this problem we deployed each algorithm in a live recommendation system in order to directly assess the (subjective) impact that a recommendation range approach has on users, in the form of live user trials.

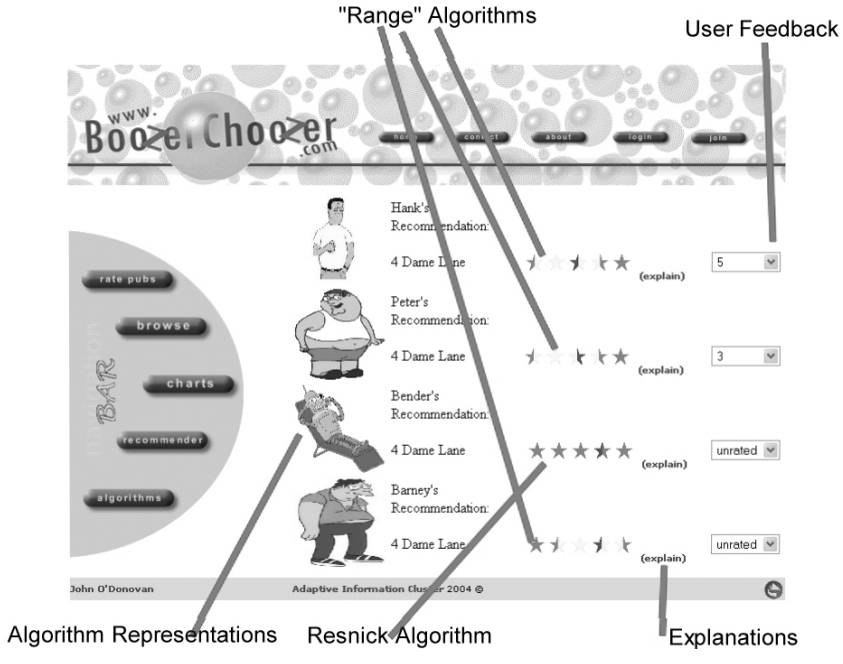

Fig. 6. Screenshot of a Recommendation Page.

5.1. *BoozerChoozer*

BoozerChoozer is a live online recommender system, designed to provide personalised recommendations of Dublin city pubs to it's users. Currently, the user base is small, consisting of less than 100 users from a similar demographic. To overcome issues such as cold-start and sparsity in the system, the number of items in the system has been restricted to 100, and each user has been asked to rate as many of these as possible at registration. At present, there are approximately 2000 ratings, meaning the dataset is about 80% sparse.

In this system, users can sign up and become members to receive personalised recommendations. Figure 5 shows a screenshot of the system. Apart from the recommendation

pages, users can learn about the algorithms which drive the system, browse individual venue details, or view leaderboard details, a dynamically updated list of the highest and lowest rated venues (shown in Figure 9).

5.2. *User Trials*

User-satisfaction is evaluated by explicitly asking users to rate recommendations from each algorithm in a series of live recommendation sessions. Similar trials have been carried out to assess subjective features of recommenders in [12], where an assessment of a critiquing system is presented. In the BoozerChoozer trial, a user was asked to rate more than 20 items in the system to build a profile to enable the algorithms to provide good quality recommendations. Once a reasonable profile has been built, users are asked to select as many items as possible for recommendation, provided that they already have an idea of the rating that they would give for each item. The system then provides four different recommendations for that item, one from each of our test algorithms, *addSub(), modRes(i), modRes(p), resnick()*, and participants are asked to assign a rating to each recommendation. The algorithms are disguised in the form of well known cartoon characters, so a test user has no idea which algorithm generated which recommendation. Figure 6 shows a screen-shot of the recommendation page for the test, displaying the four characters, the predictions from each, a link to explanations for each and a facility for the user to rate the recommendation. The benchmark Resnick algorithm is distinct in Figure 6, in this case the recommendation is denoted by half a gold star in the position of the Resnick prediction. For the other algorithms (highlighted in Figure 6, a recommendation range is represented as a series of contiguous gold stars, and the discrete prediction (r) is shown as half a red star somewhere in this series. For all of the algorithms, the remainder of the scale outside of the recommendation range is denoted as greyed out stars. This technique was chosen as it is portable to any size recommendation scale, and users can immediately see the size of the scale, as well as the predictions.

This simple trial aims to show that the recommendation range technique is preferred by users as it displays more information and is less of a black-box than standard CF techniques. In total, 40 users participated in the trial, averaging 8 recommendations each. Figure 7 shows the results of the trial. It is immediately apparent that users prefer recommendation ranges over a discrete recommendation. Each of the range techniques beats the Resnick technique by a convincing margin, our best performing technique performs twice as well as the benchmark.

Obviously, a users satisfaction with presented recommendations will rely more heavily on the content of the recommendation than on the delivery mechanism. From the results presented in Figure 7, we can see that the benchmark algorithm performed almost as well as both the *addSub()* and the *modRes(p)* techniques. In our satisfaction trial however, *all* of the range techniques performed substantially better than the benchmark, yielding an average relative benefit of 34.8% over the standard technique. This result shows that there is a strong case for presentation of recommendation ranges over discrete recommendations.

6. DISCUSSION

6.0.1. *Trust & CF Robustness*

The work of O'Mahony et. al [18] and Levien [9] outlines recommender systems from the viewpoints of *accuracy*, *efficiency*, and *stability*. The trust-error models defined in this paper can not only be used to increase recommendation accuracy, they can be utilised to increase the overall robustness of CF systems. [18] defines several attack strategies that can adversely skew the recommendations generated by a K-NN CF system. They show empirically in [18] that a CF system needs to attend to each of these factors in order to succeed well. There are many motivations for users attempting to mislead recommender

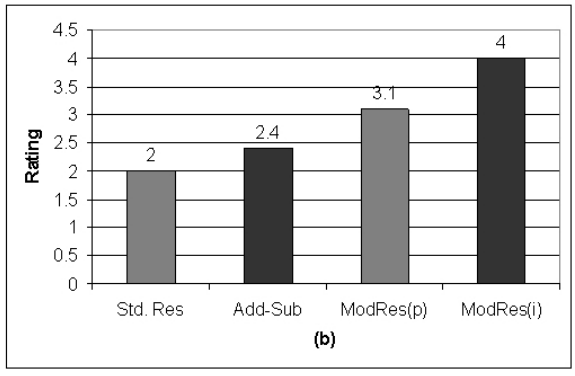

Fig. 7. Average rating of recommendations from each algorithm in a user survey

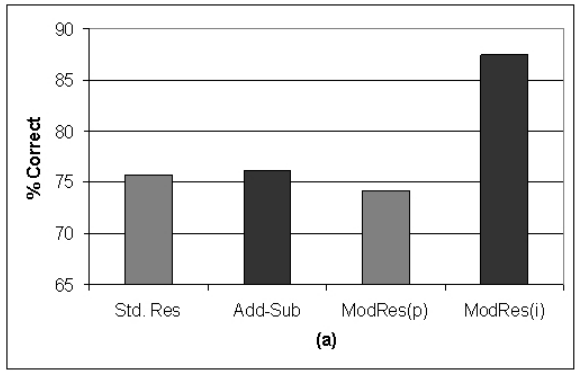

Fig. 8. Results of accuracy analysis showing % of actual ratings falling inside recommendation range.

systems, including profit and malice, as outlined in [8]. The trust-error metrics presented in this paper can help a system detect malicious users, as they will most likely tend to have higher errors in their contribution histories as producers. Using a weighting scheme we describe in [17] we can render their contributions to recommendation ineffective based on their reputation values. work by Sinha and Swearingen shows the importance of transparency in recommender system interfaces, [21]. Our recommendation range prediction method shows more information to the user about what the system knows, thereby increasing user trust in the recommender. In a similar vein, we can also mention the trustworthiness of the contributing producers to the consumer.

7. CONCLUSIONS

To conclude, here we have proposed two approaches to building up trust values based on error models, one that operates with respect to a specific user-item combination, eg: 'Bob's reputation for recommendation Toyota Landcruisers', and another that operates at a user level; eg: 'Bob's trustworthiness as a recommender'. We presented three techniques for integrating these models into a standard CF algorithm to produce recommendation *ranges* to present to a user, and defined a fair method for comparing our approach to the benchmark. Our empirical results indicate that trust-aided recommen-

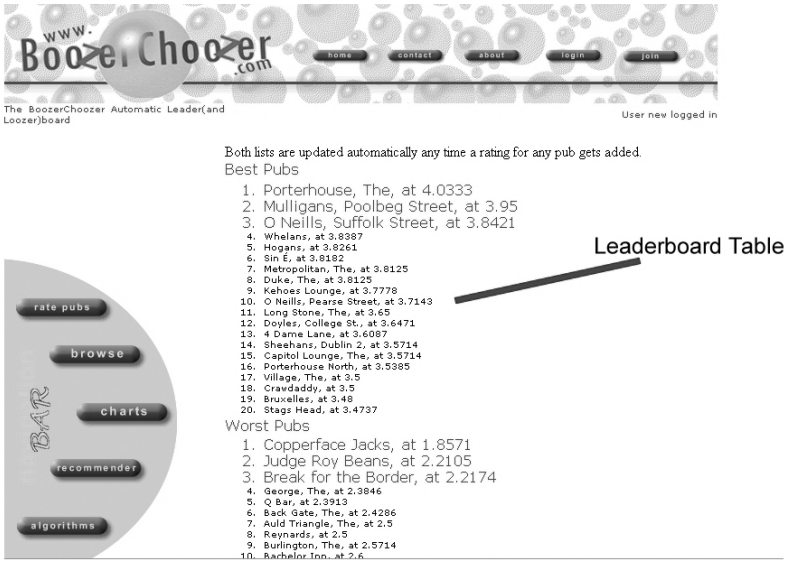

Fig. 9. Leaderboard charts.

dation based on error models increases predictive accuracy over the benchmark by 15%. We have also shown that there is a clear increase in the trust a user places in recommendations when they are presented in the form of a recommendation range derived from our error model.

8. ACKNOWLEDGMENTS

This material is based on works supported by Science Foundation Ireland under Grant No. 03/IN.3/I361. Preliminary work on mining trust from recommendation error was presented as a paper [16] at the FLAIRS 2005 conference in Clearwater, Florida, in May 2005.

References

1. Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *New Security Paradigms 1997*, pages 48–60, 1997.
2. Paolo Avesani, Paolo Massa, and Roberto Tiella. A trust-enhanced recommender system application: Moleskiing. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1589–1593, New York, NY, USA, 2005. ACM Press.
3. Keith Bradley, Rachael Rafter, and Barry Smyth. Case-based user profiling for content personalisation. In *AH '00: Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 62–72, London, UK, 2000. Springer-Verlag.
4. Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.
5. Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. Is seeing believing?: how recommender system interfaces affect users' opinions. In

Proceedings of the conference on Human factors in computing systems, pages 585–592. ACM Press, 2003.

6. Chrysanthos Dellarocas. Building trust on-line: The design of reliable reputation reporting mechanisms for online trading communities. *eBusiness@MIT, July 2001.*
7. Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
8. Shyong K. Lam and John Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pages 393–402. ACM Press, 2004.
9. Raph Levien. Attack resistant trust metrics. *Ph.D Thesis, UC Berkeley.*
10. S. Marsh. Formalising trust as a computational concept. *Ph.D. Thesis. Department of Mathematics and Computer Science, University of Stirling*, 1994.
11. Paolo Massa and Bobby Bhattacharjee. Using trust in recommender systems: an experimental analysis. *Proceedings of 2nd International Conference on Trust Managment, Oxford, England*, 2004.
12. Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. Experiments in dynamic critiquing. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 175–182, New York, NY, USA, 2005. ACM Press.
13. P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations, 2002.
14. Miquel Montaner, Beatriz Lopez, and Josep Lluis de la Rosa. Developing trust in recommender agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 304–305. ACM Press, 2002.
15. John O'Donovan and John Dunnion. A framework for evaluation of collaborative recommendation algorithms in an adaptive recommender system. In *Proceedings of the International Conference on Computational Linguistics (CICLing-04), Seoul, Korea*, pages 199–203. Springer-Verlag, 2004.
16. John O'Donovan and Barry Smyth. Eliciting trust values from recommendation errors. In *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS 05)*, pages 289–294, Clearwater Beach, Florida, USA, May 15–17 2005.
17. 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.
18. Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guenole Silvestre. Collaborative recommendation: A robustness analysis, url: citeseer.ist.psu.edu/508439.html.
19. Michael P. O'Mahony, Neil Hurley, and Guenole C. M. Silvestre. An attack on collaborative filtering. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 494–503. Springer-Verlag, 2002.
20. 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.
21. 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.
22. Barry Smyth and Paul Cotter. A personalized television listings service. *Commun. ACM*, 43(8):107–111, 2000.