Trust-Based Collaborative Filtering

Neal Lathia, Stephen Hailes, Licia Capra

Abstract *k*-nearest neighbour (kNN) collaborative filtering (CF), the widely successful algorithm supporting recommender systems, attempts to relieve the problem of information overload by generating predicted ratings for items users have not expressed their opinions about; to do so, each predicted rating is computed based on ratings given by like-minded individuals. Like-mindedness, or similarity-based recommendation, is the cause of a variety of problems that plague recommender systems. An alternative view of the problem, based on trust, offers the potential to address many of the previous limiations in CF. In this work we present a varation of kNN, the trusted k-nearest recommenders (or kNR) algorithm, which allows users to learn who and how much to trust one another by evaluating the utility of the rating information they receive. This method redefines the way CF is performed, and while avoiding some of the pitfalls that similarity-based CF is prone to, outperforms the basic similarity-based methods in terms of prediction accuracy.

1 Introduction

Over the last decade, recommender systems have had a rising presence on the web, transitioning from novelty components of e-commerce portals to become focal points of many web services. Implemented throughout e-commerce, movie and music profiling web sites, the goal of recommender systems is the flipside of classical information retrieval; these systems aim to present users with interesting content based on their historical behaviour, rather than answering a specific query. Collaborative Filtering [1], or CF, has emerged as the dominant algorithm behind recommender systems, and, as the name describes, it uses the collaborative effort of an entire community of users to help each individual sift through the endless amounts

Neal Lathia, Stephen Hailes, Licia Capra
Department of Computer Science, University College London, London WC1E 6BT, UK
e-mail: n.lathia, s.hailes, l.capra @cs.ucl.ac.uk

of online content. The current assumptions of CF are that historically like-minded individuals will also share similar tastes in the future. Measuring similarity plays a central role; only the top-*k* most similar users are allowed to contribute their ratings, and each contribution is weighted according to the specific degree of similarity the neighbour shares with the current user.

Grounding the prediction engine of CF algorithms in similarity measures hides a number of pitfalls, which stem from the fact that user profiles are incredibly sparse and limited in breadth. When users have no profile, there is no way to measure their similarity to anyone else's; the *cold-start* problem arises and no predictions can be made for the user. When users do have a profile, the neighbours they are assigned often cannot provide information about new items of interest; prediction *coverage* problems appear. Lastly, users look to recommender systems to provide both useful and *serendipitous* (or "surprising") results, and although this is a very difficult quality to measure in an algorithm, the lack of this property finds its provenance in locking measurably similar users together. These three problems originate from the fact that user profiles are incredibly sparse. The set of items that a user has rated, and hence expressed an opinion about varies in size from one user to the next. Based on the ratings each user has provided and the varying degrees of overlap between user profiles, we are currently unable to know exactly which other users would be perfect recommenders. Since CF systems hold a highly incomplete picture of the participating users they need to deal with an immense amount of uncertainty, and the current similarity-based methods often fall short of the desired performance.

An alternative view of CF systems, based on trust, has the potential to address many of the problems outline above. As detailed in [2], trust has been applied to a wide range of scenarios, including (but not limited to) *access* problems, or deciding who should be trusted to access content or services [3], problems involving *sanctioning*, or punishing network nodes that misbehave in a given context [4], and *signaling*: or helping users decide whether to access a resource or not. The last problem is surprisingly similar to problem the CF aims to solve. Descriptions of information overload [5] tell us that there is simply too much content for users to find all the items that they will like, and recommender systems alleviate this problem by helping each user decide which content to access. However, trust broadens the limited view of similarity to encompass a wider range of characteristics; we define recommenders to be *trustworthy* if they are consistent sources of valuable information, which can be appropriately interpreted when predicting how much a user will rate an item. We therefore propose to tackle the problem of information overload as a trust-management problem, and introduce and evaluate the following:

- **Selecting Neighbours Based on Trust**: neighbour selection based on profile similarity is replaced by a utilitarian evaluation of the *value* that each user provides to others, and trust is awarded accordingly, as described in Section 2.1. In particular,
- **Lack of Information is Informative**: our method awards varying degrees of trust to all those who were potential recommenders for each item a user rates, and downgrades trust scores for users who could not provide any information. In

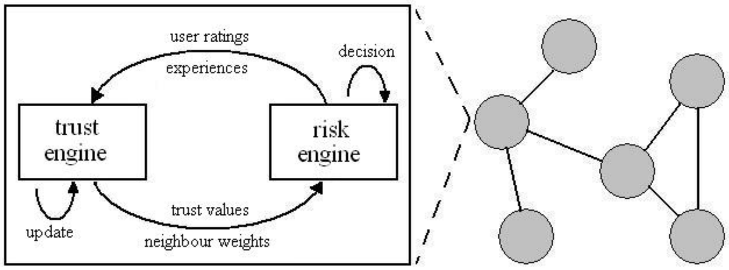

Fig. 1 A Trust Principal in a Web of Trust

other words, the presence (or lack) of information is used when computing trust values.

- **Retrieve Ratings from Recommenders, Not Neighbourhoods**: Limiting users to a top-*k* neighbourhood damages the prediction coverage that is possible. We propose to operate according to a top-*k* recommenders, or users who have the required information to make a prediction, in Section 2.2.
- **Divergent Ratings can be Useful**. The fact that two users completely disagree in their ratings does not imply that this information cannot be successfully used: an appropriate *interpretation* is required, as we discuss in Section 2.3.

2 Filtering as a Trust Problem

Can we use the properties of trust systems to address the difficulties faced by recommender systems? To answer this question, we need to understand how a trust system works, and then explore whether the methods it describes are applicable to CF or not. A trust system is a network of interacting peers, or trust *principals* [2]. Formal models of trust often divide the process of building a trust relationship with another entity into an interaction between two internal components, as we have depicted in Figure 1. The first is the *risk engine*, a component that decides whether to enter into a transaction with the entity or not. Assuming that it does enter into this transaction, then the risk engine will observe the results, and report these observations to the *trust engine*. The trust engine evaluates the interaction, and updates its trust in that entity according to rules defined in a trust policy. Each principal has a subjective view of the environment around it, and the entire community of principals forms a web of trust.

In CF, each user can be described as a trust principal, which needs to decide what content to access. To do so, the risk engine needs to decide whether to recommend an item to the end user, by generating predicted ratings. Each predicted rating is computed by collecting rating information from a subset of the other principals in the system. Which principals should be selected? Here the risk engine queries the trust engine, which maintains and updates a table of trusted peers. Traditional CF dictates that only provably similar neighbours should be considered, and the rest disregarded. Once the actual rating is received from the end user, the actual experience with the content is known, and the trust principal can look back on the prediction it made to not only see if it was correct, but if it also appropriately weighted the contributions it received from the surrounding principals, and can update its trust values accordingly. A CF environment, where a decision mechanism is needed in order to *select* an appropriate subset of users to act as recommenders, therefore, can be described as an instance of a trust based system. If CF is considered as an instance of a trust-management problem, the reverse approach can be adopted; we begin from the perspective of a trust-management system and construct a CF algorithm, by describing the operation of each component of the trust principal.

2.1 The Trust Engine

The first decision that must be made in a CF system is who to interact with; each user needs a defined neighbourhood of recommenders. This step is supported by the assumption that collecting information from everybody (and basing predictions on item *reputations*) will not be as useful as only aggregating the information from the "appropriate" sources; and thus involves deciding who these appropriate users will be. Traditionally, a user's neighbourhood has been populated with the top-k most similar users in the system. To do so, the common ratings in two user's profiles, r_a and r_b, are compared to each other, using measures such as the Pearson Correlation Coeffiecient [6]:

$$w_{a,b} = \frac{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)^2 \Sigma_{i=1}^{N}(r_{b,i} - \bar{r}_b)^2}} \tag{1}$$

The similarity $w_{a,b}$ between users a and b is computed as the degree of linearity between the ratings. Many other similarity measures have been proposed [1, 7], yet they all share the same characteristic; they rely on a non-empty intersection between two user's profiles in order to find a measure of similarity, and only measure similarity using co-rated items. This assumption is the primary cause for the cold-start problem (since users may have no historical profile), and can lead to poor prediction coverage (as the only rating information for certain items may belong to recommenders with zero-similarity). It also has been shown that the various similarity measures tend to disagree with one another; there is no way of finding the optimal method [8]. What we propose is a new method based on trust.

Transferring the current idea of CF into a trust based context means "I trust the best k users who can show that they have similar opinions to my own, and I do not trust anyone else." However, the broader approach put forward by trust management research, is also worthy of exploration. The equivalent quote would be "I trust the users who I have had a positive experience with, and *do not know* how much to trust the rest.” This quote introduces two new important concepts, which may offer an escape route from the pitfalls of CF algorithms:

- **Uncertainty**: Users should not necessarily be excluded from contributing to one another’s predicted ratings if they have no measurable value of similarity. They can still be sources of serendipitous and valuable information, if there is a means of dealing with the uncertainty.
- **Value**: Being the optimal recommender is not simply a matter of high similarity, but can be described according to two further qualities. The best neighbours will have the information necessary to participate in the user’s predicted ratings and positively influence the predicted rating towards the opinion that the user will provide. In other words, the neighbour’s opinion should be heavily weighted in the user’s predictions, and thus similarity will be an emergent property of the trust relationship, rather than being the cause of it.

When the user a enters a rating $r_{a,i}$ for an item i, the system examines all the rating information available for item i, and asks; how much *should* the user have trusted each of these recommenders? If we were considering this problem from the perspective of a user-item rating matrix, this process would iterate over item i’s column, and for each row (i.e. recommender) b make a utilitarian evaluation of the entry compared to the user’s rating r:

$$value(a,b,i) = \frac{-1}{5}|r_{a,i} - r_{b,i}| + 1 \tag{2}$$

If the recommender b has not rated item i, a trust score of 0 is returned. The equation, which assumes a five-star rating scale, awards the highest trust to users who rated the item exactly as the user did. As the distance between the user’s rating and the recommender’s rating increases, trust decreases linearly. However, if a recommender b has rated the item, the trust score will be positive, even if the recommender’s rating was the complete opposite of the user’s. This captures the fact that even if $r_{a,i}$ was $1*$, and $r_{b,i}$ was $5*$ (so the absolute difference between the ratings is 4), the recommender b was able to provide a with information regarding i, even if the opinion was very distant to a’s. A measure of value, therefore, departs significantly from similarity measures by awarding discordant ratings. The computed value is used to update the trust for recommender b, which is an average of the value contributed over all the n historical ratings:

$$trust(a,b,n) = \frac{\Sigma_{i=0}^{n} value(a,b,i)}{n} \tag{3}$$

The idea is to reward recommenders who can provide information, varying the reward according to the perceived quality of the information, and to downgrade recommenders who do not have any information available. The trust values that are assigned to other users in this system will range from 0 to $+1$. Similarity measures, on the other hand, range from -1 to $+1$; we therefore disregard the concept of dissimilarity or distrust [9], in favour of rewarding sources of information. The reward scheme we implemented is a linear function based on the absolute distance between the user's and recommender's ratings. We could have also implemented value functions with higher degrees, such that the amount of trust rewarded falls at a sharper rate as the distance increases. In this work, for the sake of simplicity and the encouraging results we obtain, we focus only on the linear case. Structuring the selection of neighbours based on learned trust has the added benefit that it need not be fully recomputed every time that a system update is performed; much like other learning algorithms [10], trust is learned incrementally as the user inputs more ratings.

Users are no longer weighted according to how like-minded they are, but will be weighted according to the quality of the information that they exchange with each other. Since the amount of information that a user provides to others may vary from how much information that user receives, weightings between user pairs are no longer guaranteed to be symmetrical. Furthermore, based on each users' initial disposition to trust the other nodes around it, users will no longer ignore the potentially useful information they receive from neighbors they have never interacted with, or a neighbor that they bear zero similarity to. Rating information will be weighted according to trust, a value that will reflect a history of interactions rather than a history of similar ratings.

2.2 Finding Opinions: k-Nearest Recommenders

Once the ranking of neighbours is complete, and trust values have been assigned to all users, a second decision must be made: CF needs to determine what neighbours will be acceptable sources of information. There are a variety of methods that can be implemented, which we define here:

- **Threshold-based**: All neighbours with weights above a pre-determind threshold are potential recommenders. Selecting such a threshold is a difficult decision, since it is dependent on the distribution of weights over the community [8], and hence is also dependent on the method that is used to rank neighbours.
- **k-Nearest Neighbours**: Traditional CF operates by limiting interactions to the top-*k* neighbours. The limitations are two-fold: on the one hand the user can have no more than *k* neighbours; on the other the algorithm only works if the user shares a measurable degree of similarity with these neighbours. If this is not the case, neighbours are assigned a zero-weighting, and any contribution from them is disregarded. Each user's neighbourhood can therefore be considered a *static* group of users, which may only change when the system is updated.
- **k-Nearest Recommenders**: A further change we introduce is a slight modification to *k*NN, by allowing users to search for the top-*k* neighbours who can actually provide information about the item that requires a predicted rating. Ranking the users is a job for the trust engine, but in order to maximise coverage we look to those who can provide information when making a prediction; a method we define as the k-nearest recommender, or *k*NR, strategy. Unlike *k*NN, each user's neighbourhood will be dynamic, and the selection of neighbours will be guided by the item that a prediction is being made for.

2.3 The Risk Engine

Now that we have outlined how the trust engine works, we have a means of selecting and updating each user's neighbourhood. This neighbourhood defines the breadth and quality of information that is available to generate predicted ratings for each user, and will hopefully be populated with neighbours who are good sources of valuable information, as we have constructed above. We now require a means of combining, or aggregating, the information that a user receives from his neighbours in order to generate a predicted rating. The predicted rating will determine whether the item should be recommended to the end user or not, and thus the aim is to minimise the *risk* of accessing (or even being recommended) uninteresting items. Traditional CF defines this task as a job for the prediction engine [6], we however chose to align our terminology with trust management research [2] and call it a risk engine. The role of this component will be to select the top-k neighbours $N(a,i)$ of user a who have rated the item i we want to make a prediction about, and can combine the ratings with trust values in one of two ways:

$$p_{a,i} = \frac{\Sigma_{b \in N(a,i)} r_{b,i} \times w_{a,b}}{\Sigma_{b \in N(a,i)} w_{a,b}} \tag{4}$$

This equation simply takes a weighted average of each neighbours rating, and has been used widely [11]. The second method converts each neighbours rating into an *opinion*, by seeing how much it deviates from that neighbour's mean rating, and can thus derive a predicted rating for a user that is centred around that user's mean, by taking a weighted average of neighbour opinions [6]:

$$p_{a,i} = \bar{r}_a + \frac{\Sigma_{b \in N(a,i)} (r_{b,i} - \bar{r}_b) \times w_{a,b}}{\Sigma_{b \in N(a,i)} w_{a,b}} \tag{5}$$

Both of these equations share a common assumption: they assume that the people who are rating items share a common interpretation of the rating scale, and thus two users who have input different valued ratings disagree with each other. Rating items, however, is a human activity, and is only guided by descriptive adjectives of what each value should represent (1 = poor, 5 = excellent). Each user of a recommender system can and will interpret these adjectives differently, and form their own mental model of the rating scale, to then use it subjectively, even if in general their opinions may tend to agree. We therefore introduce an additional characteristic of a trust-relationship: learning to interpret the opinion that we receive from a recommender. This idea has previously only been applied to trust management systems, by finding the *semantic distance* between a recommendation and one's own opinion [12]. The interpretation will serve to boost the prediction accuracy of Equations 4 and 5, and is best demonstrated by introducing an example. Our example uses actual rating information of two individuals in the MovieLens[1] movie rating dataset, collected by the GroupLens team at the University of Minnesota. The dataset consists of $100,000$

[1] http://www.grouplens.org/

	1*	2*	3*	4*	5*
1*	0	0	0	0	0
2*	0	0	0	2	0
3*	0	0	0	3	3
4*	0	0	0	3	1
5*	0	0	1	6	3

Table 1 Two-Way Contingency Table

	Lower	Same	Higher	Transpose
1*	0	0	0	1.00
2*	0	0	0	2.00
3*	0	0	1	4.00
4*	5	3	6	4.07
5*	4	3	0	4.43

Table 2 Interpreting Bob's Ratings

ratings, from 943 users on 1682 movies. Each user has rated at least 20 movies, using a five-star rating scale. The dataset has been subdivided into five disjoint training/test sets ($u1$, $u2$, ... $u5$), in order to use the training sets to set any necessary values (such as finding similarity measures), measure the predictive power of a CF method on the test set, and perform five-fold cross validation of experimental results.

2.3.1 An Example

Let us consider two users from the MovieLens $u1$ subset, the users with ids 1 and 10, who we respectively rename Alice and Bob. A quick look at their training set profiles shows that Alice has rated 135 items, and Bob has rated 94 items. The overlap, or number of co-rated items, between the two profiles, has size 22. For the sake of simplicity, let us assume that predictions for unrated items will be done with a single recommender (i.e. $k = 1$), and that the underlying neighbourhood selection procedure has determined that Bob is the best recommender for Alice. We can therefore proceed to make predictions of the items in Alice's test set, using the opinions expressed in Bob's profile.

The first point we can observe is that the overlap between Alice's test set and her recommender's training set has size 18. Therefore, we can only make predictions on 18 of her 137 test items, and so the coverage we achieve will be around 13%. Since we are using a single recommender, the amount that we weight Bob's opinions is unimportant; we simply go ahead and make 18 predictions, and only collect error measures from these items. In other words, in this case we are only interested in how well we perform when we can make a prediction, and disregard all uncovered items from the accuracy error measure.

If the predictions are made with Equation 4, the achieved mean absolute error (MAE) between the predictions and actual ratings is 0.888, while if we use the opinion-based Equation 5, we do slightly worse with a MAE of 0.948. However, both methods assume that Alice and Bob are using their rating scales in the same way. In other words, if both Alice and Bob agreed about the quality of a movie, they would both give it the same rating, and therefore the main task of CF algorithms becomes that of finding two users who not only rate the same items, but rate them in the exact same way as well. However, Alice and Bob may be using the rating scale differently from one another. Therefore, Alice's "excellent" rating may be equivalent to Bob's "great," and in the same way Bob's "very poor" (1 star) may be Alice's "poor" (2 stars). In general, they tend to agree about the items they rate, but the error remains high due to the misalignment between their use of the rating scale.

We can visualise the extent to which Alice and Bob's opinions diverge from one another by combining their ratings into a two-way contingency table, as shown in Table 1. Each entry in the table corresponds to the frequency that Alice and Bob gave a particular rating; a 6 in the last cell from the right of the bottom row tells us that on 6 separate occasions Bob's recommendation was 5 stars, and Alice's rating was 4 stars.

This table highlights a number of characteristics, such as the fact that Alice's training ratings are never lower than 3 stars, reflecting the positive skew of rating distributions that is common to CF datasets. This approach has been used to determine the extent to which two movie raters agree in their opinions [13], yet in this work we use it to interpret the information that we receive from a recommender. In fact, Table 2.3 can be reduced to a 5 × 3 table recording the experiences that Alice has compared to Bob's recommendations, and Alice can learn to transpose the ratings she receives from Bob in order to correctly interpret, or transpose, them based on the experiences she has shared with him.

The transpose of a recommender's rating is found by computing a weighted mean of the experiences shared with the recommender when receiving that rating. For example, when Alice receives a 5 star recommendation from Bob, she knows that of the seven times she received 5 stars from him before, four of those times her opinion was slightly less than 5. She thus interprets 5 stars from Bob as 4.43 stars on her own scale, as shown in Table 2. This is described by the following formula:

$$transpose(r) = \frac{(r - 1 \times lower_r) + (r \times same_r) + (r + 1 \times higher_r)}{lower_r + same_r + higher_r} \tag{6}$$

For the ratings that have no recorded historical experience the transpose remains the same as the rating. When a rating can be transposed, however, it replaces the actual rating in Equations 4 and 5. In other words, we learn to interpret the opinions we receive from recommenders as we interact with them. If we recompute the predictions on Alice's test set using Equation 4 and Bob's transposed ratings, the error in the predicted ratings falls from 0.888 to 0.710, while if we use Equation 5, the mean absolute error is reduced from 0.948 to 0.791. In other words, in this toy example, we were able to improve the recommendation accuracy of the items we could make predictions on by including the *interpretation* of the neighbour's rating into the trust relationship. Using these ideas, we can now perform a full scale evaluation on the entire MovieLens dataset.

3 Evaluation

To evaluate the performance of our method, we used the MovieLens dataset, the same dataset we used in Section 2.3.1. In this paper, we report the full results from the $u1$ subset and provide summarised results from the rest of the subsets.

	Trust-Learning		Weighted-PCC		Co-rated Proportion	
Neighbours	X-MAE	Covered (%)	X-MAE	Covered (%)	X-MAE	Covered (%)
1	0.8705	99.84	0.8723	45.44	0.9164	38.45
2	0.8226	99.84	0.8513	63.18	0.8946	54.02
5	0.7913	99.84	0.8195	81.32	0.8464	74.28
10	0.7821	99.84	0.7906	90.42	0.8139	85.24
30	0.7791	99.84	0.7565	96.80	0.7781	94.77
50	0.7794	99.84	0.7476	97.94	0.7673	97.02
100	0.7804	99.84	0.7434	99.0	0.7590	98.65

Table 3 MovieLens u1 XMAE and Coverage Results

	Trust-Learning		Weighted-PCC		Co-rated Proportion	
Dataset	X-MAE	Covered (%)	X-MAE	Covered (%)	X-MAE	Covered (%)
u1	0.7916	99.84	0.8195	81.32	0.8464	74.28
u2	0.7793	99.82	0.8159	79.16	0.8380	76.58
u3	0.7729	99.82	0.8158	79.84	0.8488	77.04
u4	0.7717	99.87	0.8247	80.24	0.8399	76.68
u5	0.7752	99.82	0.8263	81.52	0.8461	74.26
Average	0.7781	99.83	0.8204	80.42	0.8438	75.77

Table 4 MovieLens Subset Results, $k = 5$

One of the major problems concerning the evaluation of CF systems is the power of the error measures used; the primary focus of research to date is to achieve the highest possible mean accuracy. The primary error measures have therefore been the MAE, as used in the above example, and the root mean squared error (RMSE). In this work we collect error measures exclusively on the items that we can make predictions for, as we did in Section 2.3.1. We call this error exclusive-MAE, or X-MAE, values. This error measure must go hand in hand with coverage statistics, and thus leaves the trade-off decision between coverage and accuracy to those implementing recommender systems. The results are shown in Tables 3 and 4.

We compare the performance of our method to two similarity measures: the weighted-PCC in the second column, and the proportion of co-rated items in the third, which have both demonstrated to be accurate and successful means of similarity-based CF [8, 6]. When using similarity measures, we report results from using the traditional kNN strategy.

The immediate highlight of the results is the improvement in accuracy and coverage when very small neighbourhoods (k) are used; while co-rated similarity-based prediction achieves a MAE of 0.9164 on 38.45% of the dataset, the trust learning method nearly covers all predictions, 99.84%, and reduces the MAE to 0.8705. The improved coverage comes from the kNR strategy: we look for recommenders who have rated the item in order to generate a predicted rating, and even when doing so manage to maintain an improved level of accuracy. Coverage is not boosted to a maximum value since the dataset contains ratings for which there are no existing ratings.

k = 5		Number of Opinions				
Method	Total	1	2	3	4	5
Co-Rated	14,856	4,268	3,714	3,029	2,345	1,500
Weighted-PCC	16,263	4,119	4,020	3,686	2,863	1,575
Trust-Learning	19,968	38	53	97	65	**19,715**

Table 5 MovieLens *u*1 Neighbour Participation Results, for $k = 5$

k = 100		Number of Opinions				
Method	Total	1	(1-25)	(25-50)	(50-75)	(75-100)
Co-Rated	14,856	4,268	3,714	3,029	2,345	1,500
Weighted-PCC	16,263	4,119	4,020	3,686	2,863	1575
Trust-Learning	19,968	38	1,945	2,538	2,417	**13,030**

Table 6 MovieLens *u*1 Neighbour Participation Results, for $k = 100$

As the neighbourhood size increases, the accuracy results improve across all methods, and the difference between the methods is found in the coverage results. In fact, neither of the similarity measures achieve maximal coverage with fewer than 100 neighbours, which is a very large proportion of the 943-user dataset. These results also appear in the subset summary of Table 4: when k is 5, the trust method will, on average, cover a near-maximal proportion of the dataset with a higher level of accuracy than the similarity measures accomplish on smaller amounts of the same datasets. However, accuracy and coverage results are not the sole descriptors of the performance of CF algorithms, and other methods have been proposed [14]. In particular, little work has been done at understanding why one method may outperform another. For example, why does trust perform better than similarity with small neighbourhoods, but loses its advantage over similarity when neighbourhood sizes are larger?

One aspect that influences the accuracy of generated recommendations is the *participation* of recommenders. In other words, when the neighbourhood k is restricted to a specific value x (such as 5), does that mean that all predictions are made with x opinions? We show the result for $k = 5$ in Table 5. For each method we report the total number of predictions that were possible along with the individual number of predictions that were made using a varying number of opinions. We preferred using the actual values rather than proportions due to the the very low values found in the trust-learning method. As is visible in the table, restricting a user's neighbourhood in *k*NN methods does not guarantee that each prediction will be made using k opinions; on the contrary, the number of opinions used will vary. Only a very small proportion, about $1,500$ of between 14 and $16,000$ predictions, are made using contributions from the full neighbourhood. The alternative strategy adopted by our trust-learning method, *k*NR, guarantees that nearly all predictions will be made using the specified number of opinions. This may account for the improved accuracy that we measured.

However, this characteristic may also be the downfall of the trust-based method as neighbourhood sizes increase. In Table 6, we collected the same participation results for $k = 100$. Once again, the trust method includes a significantly larger number

Method	X-MAE	Covered (%)
trust *k*NN	0.8785	67.54
trust *k*NR	0.9191	99.84
trust *k*NR-T	0.8705	99.84
trust *k*NR-TO	0.8301	65.4
trust *k*NR-TOP	0.8182	59.24
trust *k*NR-TOPB (0.5)	0.8177	59.03

Table 7 MovieLens u1 Subset Results, $k = 1$

of opinions within its predicted ratings. To do so, it must include information from neighbours that it has a minimal, near-negligible history of interactions with. Ratings will therefore be difficult to interpret, or transpose correctly, and the rating will diminish the value of the predicted rating.

This sort of analysis begs the question as to whether k recommendations are necessary when making each prediction. It thus questions what the added value of each opinion will be to the specific *prediction* (rather than the considerations of value done in Section 2.1), and whether this added value can be computed when making the prediction. We leave this matter for future work.

4 Trust Results Summary

In this work we have outlined a number of techniques for performing CF from the point of view of a trust-management problem, including the *k*NR strategy, learning trust by evaluating recommender's *value*, and *transposing* (T) recommender's ratings by learning to interpret their opinions. The results we reported above explore the predictive performance of a trust-based CF algorithm when all of these components have been implemented. In this section we will take a brief look at how each of these components influences the performance. As the results in Table 7 show, the trust-learning method works relatively well with the *k*NN strategy. Changing to *k*NR boosts the coverage to a maximum, forsaking some of the accuracy. Adding the interpretation of recommender ratings (T), maintains the coverage while returning the accuracy to the level we found at first. There are a number of further characteristics that can be considered when generating recommendations, which we introduce below. The main conclusion remains that each of these individual components can be implemented separately; the decision should be based on the recommendation context, or domain, and the user search mode [15].

4.1 Positive Opinions, Potential Recommenders, and Initial Trust

In this section we describe three additional characterisics of trust relationships that can be applied to CF; each of them exerts its own influence on the accuracy and coverage performance metrics, and is included in Table 7.

- **O: Positive Opinions**: One choice that can be applied to our k-NR strategy is to limit the predictions that are made to those that include at least one *positive* opinion. The reason for this is that, while the goal of CF is to predict ratings accurately, in order to rank items and then decide whether to recommend the item to the user or not, recommender systems in general are more interested in finding the items that users *like*, or rather, ones that they would also have a positive opinion about. If predictions that do not include at least one positive opinion are disregarded, and marked as *uncovered*, the accuracy increases at the expense of prediction coverage.
- **P: Potential Recommenders**: Computing trust and similarity share a common trait. Both are subject to the size of the profile of the current user; as it increases the possible set of co-rated items with neighbours increases as well, and both trust and similarity measures become more reliable. A further consideration would be to reward trust to *potential* recommenders, by rewarding them with trust for items they have rated that the current user has not. This is an attempt to capture how much "knowledge" the neighbour's profile contains, and pair this with the value judgements that are computed on shared opinions. In Table 7, the reward value was set to 0.1.
- **B: Initial Trust**: The computation of trust we have described thus far requires the user to have rated items in order to find valuable neighbours. If the user has no such ratings, no predictions can be made at all, and the *cold-start* problem appears. This problem can be side stepped if users are allowed to set initial trust values in others [16], or trust values are set to an inital constant. How much should a principal trust another if they have never interacted before, or if the current user has no profile? This is a question of trust *bootstrapping*, a non-trivial problem that has been studied extensively [17] that is beyond the scope of our work here. Limiting ourselves to bootstrapping trust values in all neighbors to an initial value β, which will then be molded by the measures of value found as the user's profile grows requires a small modification of Equation 3:

$$trust(a,b,n) = \frac{\beta + \Sigma_{i=0}^{n} value(a,b,i)}{n+1} \tag{7}$$

  In terms of how this affects recommendations, the initial predictions that a system will be able to make (based on no historical profile) will be the same as items' reputation, since all neighbours will have equal weighting. Recommendations will become more accurate as the number of ratings the user inputs grows, trust values become more reliable, and the risk engine, as will be explored in Section 2.3, can operate more effectively. Table 7 reflects an experiment done with $\beta =$ 0.5.

5 Related Work

Computational trust is a topic that is experiencing a rising influence in the field of recommender systems. Not only does it provide an interesting metaphor that can be used to explain the implicit interactions that occur as CF algorithms operate, but also suggests methods that can improve on the work that has been done to date. Trust, however, is applicable to different aspects of recommender systems. In this short review we focus on the influence of trust on the algorithm deriving predicted ratings, and not, for example, on the interactions between users and the interface to the recommender system.

Amongst the first applications of trust to CF is the work done by Massa et al [16]; in order to overcome problems that arise from similarity measures, which require co-rated items in order to function, users can be asked to input trust values in other users. This method is particularly useful if a *trust propagation* algorithm is also implemented, and allows both the breadth and accuracy of predicted ratings to improve. Our work differs from [16] by performing trust evaluations based on *computed* added value, rather than pushing the decision to the end users, a scenario that may be more appropriate when user profiles need to are restricted in the interest of privacy. We also did not include trust propagation as we assume a centralised domain, where the user-rating matrix is available for complete analysis.

The trust-learning method that we propose more closely resembles the work done by O'Donovan and Smyth in [14]; they define a recommender's rating to be correct if the difference between it and the user's rating is less than a threshold value. Correctness is defined as a binary value, and, in doing so, begins to return to the domain of similarity. Our trust-learning method rewards sources of information, regardless of the opinion that was expressed, in order to differentiate between trustworthy recommenders (sources) and opinions (transposed ratings).

Trust has also been explored in decentralised recommender systems [18, 19]. As above, similarity relationsips can be viewed as a limited instance of trust relationships, thus allowing trust to be propagated over networks and then reconverted into similarity in order to reduce data sparsity. [18] extends this work, exploring whether trust models can be feasibly implemented in these systems, observing the effect of the trust model on bandwith usage, response time, and on user's patience.

All these solutions, however, focus on profile similarity. Our approach to the problem moves away from similarity by considering trust as a question of value rather than similarity. All users learn who their trusted recommenders are, and at different points in time they will have both varying levels of information with regards to their community. This idea deviates from the traditional assumption of similarity-based user interaction, towards encompassing further requirements of successful interaction, such as information sharing and honesty, into the recommendation process.

It is important to note that the implementations of trust in CF contexts are parallel to work being done by the clustering community, which include many interesting techniques that aim towards the same goals we stated above: finding a subset of the community of users that are the "appropriate" ones. Work such as [20] describes clustering from the perspective of message-passing, performing clustering by imitating the way people group together around common interests. Although lacking in the language used within trust-management research, these fields of work can offer a wide variety of important techniques that are equally applicable using the trust metaphor of CF.

6 Conclusion

In this work we have addressed the problem of learning how much to trust rating information that is received from other users in a recommender system. We thus transform the formation of predicted ratings from "how did *similar* users to me rate this item?" to "how much do those I *trust* like this item, and how should I *interpret* their opinion?" The results we have achieved not only offer a means for CF to be performed successfully under a new trust-based metaphor and perform within similar ranges of similarity-based methods, but offer novel ways of interpreting profile data when generating predicted ratings. Constructing a CF algorithm based on trust will thus include similarity as an emergent property of trust relationships, but not the cause of it. The performance of CF algorithms are known to be subjective to the dataset that they operate on; the contribution of this work can only be further validated by experimenting with more datasets. In particular, the optimal initial trust may vary between datasets, and designing a means of learning what this value is would be a great advantage.

The trust-based perspective and methodology also reduces the vulnerability that CF has to profile injection attacks; inserting a profile of ratings would not affect a target user's trust values in others unless the inserted ratings cover items that the target user has yet to rate. This is due to the fact that building trust relationships inherently includes a temporal component, and thus only users who have input ratings for an item prior to the active user inputting the rating will be considered for trust updates. Therefore an attacker can only target users who have yet to rate the items that meet the injected profile. It is possible to mimic this scenario in a controlled, experimental set up, but makes attacks much harder to design in the broader settings of online recommender systems. Our evaluation at this point has also implicitly assumed that all the users who are participating in the CF environment will not alter it or try to "game" the system in any way. Our future work will focus on removing these assumptions and observing the effect that these new scenarios will have on the recommendation process. Since we have defined interactions in this work on a user-pair basis, a malicious node can still attack each user in the system before all will have lost their trust in it. This entails defining suitable attack models for these kind of systems, and methods for users to defend themselves from malicious peers, by propagating trust values throughout the community, such as in techniques provided in [17].

References

1. J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. In *ACM Transactions on Information Systems*, volume 22, pages 5–53. ACM Press, 2004.
2. M. Carbone, M. Nielsen, and V. Sassone. A formal model for trust in dynamic networks. In *In Proceedings of Int. Conference on Software Engineering and Formal Methods, (SEFM)*, 2003.
3. Daniele Quercia, Manish Lad, Stephen Hailes, Licia Capra, and Saleem Bhatti. Strudel: Supporting trust in the dynamic establishment of peering coalitions. In *Proceedings of the* 21^{st} *ACM Symposium on Applied Computing*, pages 1870–1874, Dijon, France, April 2006.
4. L. Yan, S. Hailes, and L. Capra. Analysis of packet relaying models and incentive strategies in wireless ad hoc networks with game theory. In *Proc. IEEE 22nd International Conference on Advanced Information Networking and Applications (AINA08)*, GinoWan, Okinawa, Japan, March 2008. IEE Computer Society.
5. J.B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the ACM Conference on Electronic Commerce*, 1999.
6. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237, 1999.
7. G. Linden, B. Smith, and Y. York. Amazon.com recommendations: Item-to-item collaborative filtering. In *IEEE Internet Computing*, pages 76–80, 2003.
8. N. Lathia, S. Hailes, and L. Capra. The effect of correlation coefficients on communities of recommenders. In *To Appear in ACM SAC TRECK*, 2008.
9. R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pages 403–412, 2004.
10. K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2001.
11. R. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *IEEE International Conference on Data Mining (ICDM'07)*. IEEE, 2007.
12. A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd International Conference on System Sciences*, Hawaii, USA, 2000.
13. A. Agresti and L. Winner. Evaluating agreement and disagreement among movie reviewers. In *Chance*, volume 10, 1997.
14. J. O'Donovan and B. Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174. ACM Press, 2005.
15. I. Im and A. Hars. Does a one-size recommendation system fit all? the effectiveness of collaborative filtering based recommendation systems across different domains and search modes. In *ACM Transactions on Information Systems (TOIS)*, volume 26, November 2007.
16. P. Massa and P. Avesani. Trust-aware recommender systems. In *Proceedings of Recommender Systems (RecSys)*, 2007.
17. D. Quercia, S. Hailes, and L. Capra. Lightweight distributed trust propagation. In *Proceedings of the 7th IEEE International Conference on Data Mining*, Omaha, US, October 2007.
18. G. Pitsilis and L. Marshall. A trust-enabled p2p recommender system. In *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 59–64. IEEE, 2006.
19. G. Pitsilis and L. Marshall. *Trust as a Key to Improving Recommendation Systems, Trust Management*, pages 210–223. Springer Berlin / Heidelberg, 2005.
20. H. Geng, X. Deng, and H. Ali. A new clustering algorithm using message passing and its applications in analyzing microarray data. In *Proceedings of the Fourth International Conference on Machine Learning and Applications*, 2005.