2008

# Trust-Networks in Recommender Systems

Kristen Mori
*San Jose State University*

# TRUST-NETWORKS IN RECOMMENDER SYSTEMS

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Kristen Mori

December 2008

# Abstract

Similarity-based recommender systems suffer from significant limitations, such as data sparseness and scalability. The goal of this research is to improve recommender systems by incorporating the social concepts of trust and reputation. By introducing a trust model we can improve the quality and accuracy of the recommended items. Three trust-based recommendation strategies are presented and evaluated against the popular MovieLens [8] dataset.

# Table of Contents

# List of Figures

# List of Tables

# 1. Introduction

With the explosive growth of the Internet, information overload has become an increasing problem. Users searching for products or content have an endless number of Web pages to navigate. Recommender systems [14] have shown to be an important solution to the information overload problem. The job of the recommender system is to provide the consumer with a selection of products or content which suit her needs.

This research will analyze current recommender systems and the improved trust-enabled recommender systems. I argue that incorporating the social concepts of trust and reputation into recommender systems can lead to an increase in accuracy and quality of recommendations. I propose a novel method for calculating trust and reputation between users, and new trust-based recommendation strategies which incorporate the trust metrics.

# 2. Recommender Systems

Recommender systems have become a widely used tool for Web applications. In an environment where there are an infinite number of Web sites for consumers to choose from, the competition is fierce. If a Web site can offer a consumer, an automated and intelligent system which generates personalized recommendations, this would definitely provide a competitive advantage. There are many different types and uses for recommender systems. The recommended items can range from online-shopping products (books, movies, clothing, etc.), to a suggested course for a student, or a particular kind of medical care for a patient [15]. Recommender systems use various types of information to generate a recommendation, such as, past purchase records, click stream analysis, user profiles, explicit ratings of items, or social network information [15]. Recommender systems use various methods to process the input data, and output recommendations to the user. These systems have been categorized into two groups: content-based and collaborative filtering [14].

Content-based recommender systems operate by comparing descriptions of content for recommendable items, and hence are limited by their need for rich textual descriptions [3]. For instance, a content-based movie recommender will typically rely on information such as genre, actors, directors, etc. and match this against the learned preferences of the user in order to select recommendations. This method requires that a significant amount of domain knowledge can be obtained.

The collaborative filtering recommendation strategy uses a different approach. It is built on the assumption that a good way to find interesting content for a user, is to find other users who have similar interests, and then recommend items that those similar users liked [7]. The main characteristic of collaborative filtering is that it accumulates the user's ratings of products, identifies users with common ratings, and offers recommendations

based on inter-user comparison [5]. In other words, recommendations for a target user are based on the behavior and the evaluations of the other users. In this research, we will focus on collaborative filtering and how to improve its ability to make accurate recommendations.

# 3. Collaborative Filtering

## 3.1 Overview of Collaborative Filtering

Generally, the task in collaborative filtering is to predict the ratings of a particular user, which we will refer to as the target user. If the system is able to accurately predict what the target user will rate an item, then the system can recommend desirable items. For collaborative filtering, we need a database of votes or ratings from a population of users. Each rating in the database corresponds to the rating from single user on a specific item.

There are three main categories of collaborative filtering algorithms [7]: memory-based algorithms, model-based algorithms, and hybrid algorithms. Memory-based algorithms use statistical techniques over the entire database of ratings to identify similar users, who are then used to make recommendations [5]. A model-based algorithm uses the ratings database to learn a probabilistic model, such as cluster models or Bayesian network models, and then uses the model to produce recommendations [7]. Hybrid collaborative filtering algorithms use both memory-based and model-based algorithms.

We shall focus on memory-based algorithms for this research. The traditional memory-based collaborative filtering technique is to identify similar users, and then the similar users predict the rating for a target user, *c* on an item, *i*. These similar users are chosen because they share similar or highly correlated rating histories with the target user [3]. If two users have a highly correlated rating history, then the user should be able to accurately predict how the target user will rate an item in the future.

It is typical to distinguish between the two types of user profiles in the context of a given recommendation session. The target user or *consumer*, *c*, refers to the user profile receiving the item recommendation, and the *producer, p,* refers to the profile that has been selected as a recommendation partner for the consumer [3].

It is important to note that the entire collaborative filtering process is completely transparent to the users. The actual users of the system simply provide ratings and receive recommendations. It is the collaborative filtering recommender system, which identifies similar users and utilizes their past ratings to make recommendations to other users. We can assume that the users do not interact with one another, and they do not provide recommendations for each other.

## 3.2 Correlation

Measuring the correlation or similarity between users plays a central role in the traditional collaborative filtering recommender system. The first step in the recommendation process is to compute similarity values between each pair of users. The most widely used formula to compute similarity is the Pearson correlation [7]:

$$sim(c, p) = \frac{\sum_{i \in I} (r_{c,i} - \overline{r_c})(r_{p,i} - \overline{r_p})}{\sqrt{\sum_{i \in I} (r_{c,i} - \overline{r_c})^2 \sum_{i \in I} (r_{p,i} - \overline{r_p})^2}} \tag{3.1}$$

Let *sim(c, p)* represent the similarity between users *c* and *p*, where *I* is the set of items rated by both users, $r_{c,i}$ is the rating user *c* gave to item *i*, $r_{p,i}$ is the rating user *p* gave to item *i*, $\overline{r_c}$ is the average rating of user *c*, and $\overline{r_p}$ is the average rating of user *p*. The results obtained from the Pearson correlation formula range from -1 for negative correlation to +1 for positive correlation. This formula measures the extent to which there is a linear relationship between two variables [1]. For the purposes of collaborative filtering, we look at all co-rated items between user *c* and user *p*. If two users have rated many items in common, then they will have a high similarity value. If two users have not rated any item in common, then a similarity value cannot be computed.

## 3.3 Neighborhood Formation

Once similarity values have been calculated between every pair of users, we want to identify all users who are the most similar to the target user. The idea is that the most similar users will also be the best recommendation partners. This technique is referred to as, *k*-Nearest-Neighbors (*k*NN) [5, 6]. Only the top-*k* most similar users will be added to the target user's neighborhood. The neighborhood for the target user is responsible for making recommendations. This network of users will remain static until additional ratings are added to the database, at which point the similarity values should be recalculated and the users in the network could potentially change.

## 3.4 Rating Prediction

Remember that the goal of collaborative filtering is to predict the ratings of the target user. If the system is able to accurately predict what the target user will rate an item, then the system can recommend desirable items to the target user.

The final step in the collaborative filtering process is to generate the rating prediction for the target user on a specific item, *i*. Once the neighborhood for the target user has been

formed, we aggregate the rating information of each neighbor to generate the prediction value. Resnick et al. [16] have proposed a widely used method for computing a prediction value, which has been commonly referred to as the Resnick formula:

$$P_{c,i} = \overline{r_c} + \frac{\sum_{p \in M} sim(c, p)(r_{p,i} - \overline{r_p})}{\sum_{p \in M} | sim(c, p) |} \tag{3.2}$$

Let $P_{c,i}$ represent the predicted rating, where $M$ is the set of all users who belong to target user's neighborhood and have rated item $i$. Target user, $c$, for whom predictions are being computed, does not belong to $M$. $sim(c, p)$ is defined by Eq. (3.1), $r_{p,i}$ is the rating user $p$ gave to item $i$, $\overline{r_p}$ is the average rating of user $p$, and $\overline{r_c}$ is the average rating of user $c$.

In the Resnick formula, each neighbor contributes their rating of item $i$, and each contribution is weighted according to the specific degree of similarity the neighbor shares with the target user [6]. It is important to note that a neighbor must have rated item $i$ in order to participate in the rating prediction. Thus, if none of the neighbors have rated item $i$, then a rating prediction cannot be computed. This is a major limitation to traditional collaborative filtering, which will be discussed in the next section.

The rating predictions determine which items are recommended to the target user; therefore the quality of recommendations relies on the accuracy of the predictions. In the normal operation of a recommender system, a set of rating predictions are sorted and the items with the highest values are used for recommendation purposes. To measure the accuracy of the recommendations, the target user must submit an item rating, which can then be compared to the predicted rating.

Figure 1 gives an overview of the traditional collaborative filtering process.

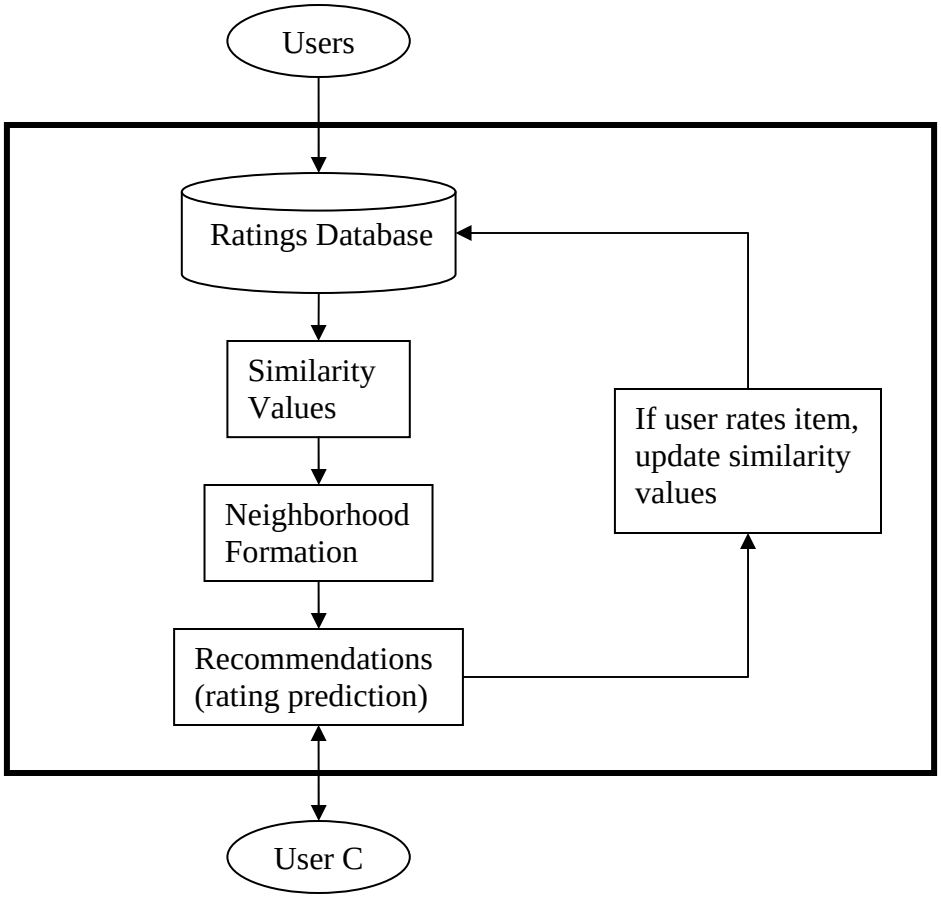


**Figure 1: Overview of Traditional Collaborative Filtering**

## 3.5 Limitations of Collaborative Filtering

Collaborative filtering suffers from several weaknesses. A major problem with traditional collaborative filtering is *data sparseness*. A real application will be comprised of millions of products and millions of users. However, a typical user will only rate a small number of items. The first step in collaborative filtering is computing similarity values for each pair of users. The process of computing similarity involves comparing the ratings of two users; therefore it is needed that the two users rated at least some items in common. With a large dataset it is very unlikely two random users have rated any items in common; therefore a similarity value is not computable in most cases [4]. As a result, the recommender system may not be able to make predictions for a user and/or the accuracy of the recommendations may be poor.

The data sparsity problem is particularly evident in *cold-start* users. This is the second limitation to collaborative filtering, which is referred to as the *cold-start* problem [4].

This problem refers to new users, or users who have provided few ratings. In these cases it is difficult or impossible to provide a recommendation based on the profile similarity. The collaborative filtering system is unable to calculate similarity values and build a neighborhood for the target user. This is a significant problem, since it is important to provide good recommendations to new users in order to retain them as customers.

*Scalability* is another weakness in traditional collaborative filtering. The basis for making accurate recommendations relies on finding similar users. A typical algorithm must compute a similarity value for each pair of users. The computation overhead will quickly increase as the number of users increases, leading to poor scalability.

### 3.6 Improving Collaborative Filtering

The goal of this research is to improve the accuracy and quality of recommendations that collaborative filtering provides. The proposed trust-based recommendation strategy will modify the way that neighbors are selected and weighted during the recommendation process. By introducing a trust and reputation metric, we are developing new relations between the users, thus increasing the connectivity of the user base and alleviating the data sparseness problem. The trust and reputation values will be used to identify the best recommendation partners. The similarity between users should not be the sole factor in neighborhood formation. The trustworthiness of a partner should be taken into consideration, because a recommendation partner may have similar ratings to a target user, but they may not be a reliable predictor for a given item. A neighbor should be trustworthy in the sense that they have a history of making reliable recommendations [3].

## 4. Background and Related Work

### 4.1 Social Trust

In the real-world, when looking for a recommendation of a restaurant or movie, we often turn to our friends, on the basis that we have similar food or movie preferences. However, a particular friend may not be reliable when it comes to recommending a particular type of movie. For example, you may ask Alice to suggest a romantic-comedy, but you would not ask her to recommend an action-thriller movie. There is a measurement of trustworthiness which we have in our friends' recommendations. This suggests that similarity alone will not always provide the most reliable recommendation partners. Our recommendation partners should have similar tastes and preferences, but they should also be trustworthy.

Social trust is very complex and depends on many factors, which makes is difficult to model in a computational system. Some factors which influence trust are: past experience with a person, relationship with the person, opinions of the actions a person has taken,

psychological factors impacted by a lifetime of history and events, rumor, and influence by others' opinions [2]. Much work has been done to formalize the concept of social trust into computing environments.

## 4.2 Properties of Trust

There are three main properties of trust that are relevant to developing trust-based computational models: transitivity, asymmetry, and personalization [2]. This research attempts to model each social property in the computing environment to accurately reflect the notion of social trust.

The idea of transitivity is that social trust can be passed between people. For example, Alice highly trusts Bob, and Bob highly trusts Chuck, even though Alice does not know Chuck, she could still derive some sense of trustworthiness for Chuck. However, trust is not perfectly transitive in the mathematical sense, because it would not be the case that Alice highly trusts Chuck, a person she has no previous interactions with. There has been much research in modeling the transitivity of trust, also referred to as trust propagation.

Guha et al. [18] developed a formal framework of trust propagation schemes. Their framework assumes that users explicitly state trust values in other users. They have also introduced the notion of distrust and the propagation of distrust, which has not been done in previous research. They conducted experiments on the Epinions.com dataset, and concluded that a small number of expressed trust statements per user, allows the system to predict trust between any two people in the system with high accuracy.

The asymmetric property of trust is very important. If two people are involved in a relationship, the trust which they hold for one another will not necessarily be identical. Because individuals are so unique in their terms of their personal experiences, backgrounds, and histories, it is easy to understand the asymmetric nature of trust. In the application to collaborative filtering, trust differs from similarity, in that similarity is symmetric. This is an important difference because trust allows users to form additional connections which were not possible with similarity values.

The last property is personalization of trust. Trust is a subjective, personal opinion. Two people often have very different opinions about the trustworthiness of the same person. Personalization plays an important role in making recommendations to a user. It is the personalization of trust which greatly affects the accuracy of a recommendation. The proposed model in this work, attempts to compute trust values on a very fine-grained, personalized level.

## 4.3 Related Work

There has been an increasing amount of research being performed in the area of trust-based recommender systems; however, it is still in the early stages of development.

Golbeck et al. [2] applies the concept of trust to Web-based social networks. They describe how trust can be computed and how it can be used in applications. Specifically, they propose two algorithms for inferring trust relationships between individuals who are not directly connected to the network. The researchers apply their technique to the TrustMail application, which is an email client that uses the trust algorithms to sort email messages in the user's inbox depending on the ratings in the trust network.

Pitsilis et al. [19] explored trust in decentralized recommender systems. They developed a model which uses quantitative and qualitative parameters to build trust relationships between entities based on their common choices. Trust propagation was used to extend trust relationships beyond the direct neighbors. Their recommender system was tested on various lengths, or "hops," of trust propagation. The experimental results showed a significant decrease in data sparseness and prediction error.

Massa et al. [4] introduce a trust-aware recommendation architecture which relies on users explicitly stating trust values for other users. They also use a trust propagation technique applied to the network of users, to estimate a trust weight that can be used in place of the similarity weight. An evaluation of their system on the Epinions.com dataset reveals that their system is successful in lowering the mean error on predictive accuracy for cold start users. The authors also define a trade-off situation between recommendation coverage and accuracy in the system.

O'Donovan et al. [3] propose a system which is similar to this work. They present algorithms for calculating a profile-level trust and item-level trust. Their trust metrics compute the percentage of correct recommendations that the user has contributed. The item-level trust represents the percentage of times that a user correctly recommended a particular item. Their experimental results show a positive impact on the overall prediction error rates.

This work differs from the above research in several ways, most notably, in the derivation of trust values. Rather than having the user explicitly state trust values, the recommender system will implicitly calculate trust between users. Users may not want to expend effort on assigning trust values to other users, but they should still receive the benefit of quality recommendations. The proposed trust metric in this work is a more refined measurement of determining the accuracy of a recommendation. Furthermore, the proposed system also incorporates a reputation value when choosing recommendation partners.

# 5. Computational Model for Trust and Reputation

## 5.1 Defining Trust and Reputation

Various definitions have been presented for trust and reputation. For example, Josang et al. [10] have defined trust as "the extent to which one party is willing to depend on something or somebody in a given situation, even though negative consequences are possible." Mui et al. [20] define trust as "a subjective expectation an agent has about another's future behavior based on the history of their encounters."

For the context of this work, trust is defined as the reliability of a user to deliver accurate recommendations in the past [3]. If a user has a history of making accurate recommendations, they can be viewed as more trustworthy than a user who made poor recommendations. A user's past behavior will determine their trustworthiness.

The reputation for a person is defined as "what is generally said or believed about a person's character or standing" [10]. The main difference between trust and reputation is that trust reflects a subjective view of a user's trustworthiness, whereas reputation reflects the view of the whole community. For this work, reputation is defined as "how the entire community views an individual user."

## 5.2 Trust Metric

For this research, trust has been defined as the ability of a user to provide accurate recommendations. Trust values will be calculated between each pair of users and the trust values are asymmetric. For instance, Alice may view Bob as very trustworthy and Bob may view Alice as untrustworthy. The trust values which a target user holds for all other users will vary over time; this represents the personalization of trust.

Over the normal operation of the recommender system, many items will be recommended to a target user, but the target user will only provide ratings to some of the recommended items. Once a rating is received from the target user, we can determine the accuracy of the recommendations by comparing the predicted rating to the user's actual rating. Through the process of comparing predicted and actual ratings, the trust values between users are updated accordingly.

As discussed in Sec. 3.4, a rating prediction for a target user is generated using the target user's neighborhood and applying the Resnick formula, Eq. (3.2). This means that all users in the neighborhood will be contributing to the rating prediction. However, the trust values are supposed to represent the trust that the target user holds for a specific user, *p*. Therefore, when calculating the trust value, we are only interested in the accuracy of user *p*'s contribution to the overall rating prediction. To capture user *p*'s contribution, the

Resnick formula is modified to generate a rating prediction where user *p* is the sole contributor:

$$P_{c,i} = (\overline{r_c} - \overline{r_p}) + r_{p,i} \tag{5.1}$$

Let $P_{c,i}$ represent the rating prediction generated from user *p*, on item *i*, for target user, *c*. Where, $\overline{r_c}$ is the average rating of user *c*, $\overline{r_p}$ is the average rating of user *p*, and $r_{p,i}$ is the rating user *p* gave to item *i*.

Once user *p*'s prediction has been generated, the item-level trust, Eq. (5.2), measures the accuracy of the predicted rating in comparison to the actual rating:

$$T_c(p,i) = 1 - \frac{|P_{c,i} - r_{c,i}|}{z_{\max} - z_{\min}} \tag{5.2}$$

Let $T_c(p,i)$ represent the item-level trust value, where *p* is the user who provided the rating prediction on item *i*, and *c* is the target user. $P_{c,i}$ is the predicted rating made by user *p*, $r_{c,i}$ is the target users actual rating, $z_{max}$ is the top of the rating scale, and $z_{min}$ is the bottom of the rating scale. In a collaborative filtering environment which uses a 5-star rating scale, $z_{max}$ would be 5 and $z_{min}$ would be 1. The results for the item-level trust range from [0, 1], where a larger value means the prediction was more accurate.

Remember that trust is defined as the ability of a user to provide accurate recommendations. The item-level trust measures accuracy of user *p*'s prediction for a particular item. An overall trust value for user *c* and *p* is calculated by taking the cumulative average of all item-level trust scores:

$$T_c(p) = \sum_{i \in I} T_c(p,i) \Big/ M \tag{5.3}$$

Let $T_c(p)$ represent the overall trust which user *c* holds for user *p*, where *I* is the set of items that user *p* recommended to user *c*, $T_c(p,i)$ is the item-level trust for each item *i*, and *M* is the total number of items in set *I*. The results for the overall trust value range from [0, 1], where a high value means user *p* has provided very accurate recommendations in the past.

The overall trust score indicates the accuracy of user *p*'s recommendations in the past; it also indicates whether or not the target user should trust user *p* in the future. For this reason, the trust metric provide valuable information when choosing recommendation partners. Additionally, the overall trust value will be updated each time the target user

provides a rating to a recommended item. The more ratings that the target user provides, means the trust values become more reliable.

## 5.3 Reputation Metric

Reputation is measured by how the entire community views an individual. Every user will have a reputation value, which is calculated as the harmonic mean between the average trust score and the experience of the user. To calculate the reputation value for user *p*, we need to average the trust values that every member in the community holds for user *p*:

$$T_{avg}(p) = \frac{\sum_{c \in N} T_c(p)}{|N|} \tag{5.4}$$

Let $T_{avg}(p)$ represent the average trust value that the community has for user *p*, where *N* is all set of all users in the community, and $T_c(p)$ is the overall trust value that user *c* holds for user *p*.

The trust value between user *c* and *p* represents the accuracy of *p*'s recommendations to user *c*. Therefore, the average trust value will indicate *p*'s recommendation accuracy across the entire community. It is possible for user *p* to have a very high average trust value, even if the total number of recommendations is very low. For instance, Alice and Bob could have identical average trust values of 0.95, yet Alice has contributed 200 recommendations and Bob has only contributed 10 recommendations. Even though their average trust values are the same, Alice would be a more reputable user in the community. For this reason, we need to take into account the number of recommendations that user *p* has contributed to the community; this will be calculated as Experience:

$$Exp(p) = N_p / N_{\max} \tag{5.5}$$

Let *Exp(p)* represent the experience of user *p*, where $N_p$ is the total number of recommendations that user *p* has contributed, and $N_{max}$ is the maximum number of recommendations contributed by a user. The results for experience will range from [0, 1], where a larger value means the user has more experience.

Once the average trust score and experience have been calculated, the reputation value for user *p* can be computed:

$$rep(p) = \frac{2(Exp(p))(T_{avg}(p))}{Exp(p) + T_{avg}(p)} \tag{5.6}$$

Let *rep(p)* represent the reputation value for user *p*, where *Exp(p)* is user *p*'s experience, and $T_{avg}(p)$ is the average trust for user *p*. The advantage to using the harmonic mean is that it is robust to large differences between the inputs [3], so a high value will only be calculated if both the average trust and experience values are high.

The reputation values will be visible to the entire community. Reputation will be particularly valuable for new users, who have not provided any ratings. Without a rating history the recommender system cannot generate similarity values or trust values. In this case, the recommender system could utilize the most reputable users for generating predictions and recommending items.

## 5.4 Trust Propagation

As discussed in Sec. 3.5, data sparseness is a major problem in traditional collaborative filtering. When the ratings database is sparse, it means that we are unable to calculate similarity values between users; hence, we are unable to identify recommendation partners and generate predictions. Trust propagation is a method which attempts to lessen the detrimental effects of the data sparseness problem.

The main idea behind trust propagation is to help derive the trustworthiness of users with whom the target user has had no previous experience. By propagating trust values we can find trustworthy recommendation partners for the target user, even if they have no co-rated items. The goal is to reduce the uncertainty about users in the community, while increasing the coverage [17]. To propagate trust values, we still need to form a neighborhood for the target user. Once we have direct neighbors, we build a trust network by propagating the trust values of the direct neighbors, to other members in the community. The following formula is used to propagate trust values:

$$Ta(c) = Ta(b) * Tb(c) + (1 - Ta(b)) * (1 - Tb(c)) \tag{5.7}$$

Let *Ta(c)* represent the propagated trust value that user *a* holds for user *c*, where *a* and *c* have had no previous experience. *Ta(b)* is the trust value that user *a* has in user *b*, and *Tb(c)* is the trust value that user *b* has in user *c*.

# 6. Trust-based Recommendation Strategies

There are several ways to incorporate the trust and reputation metrics into the recommendation process. I propose three different trust-based recommendation strategies. Each strategy will modify the way that neighbors are chosen, and how those neighbors are used to generate a rating prediction.

## 6.1 Trust Strategy

The Trust strategy requires that trust values are available for each pair of users. The neighborhood for the target user, *c*, is formed by finding the most trustworthy users, i.e. the users with the highest trust value, $T_c(p)$. The rating prediction is based on a modification of the Resnick formula, Eq. (3.2). In the modified formula, the similarity value is replaced with the trust value and all other variables remain the same:

$$P_{c,i} = \overline{r_c} + \frac{\sum_{p \in N} T_c(p)(r_{p,i} - \overline{r_p})}{\sum_{p \in N} |T_c(p)|} \tag{6.1}$$

## 6.2 Trust & Reputation Strategy

The Trust & Reputation strategy requires that trust, reputation, and similarity values are available for all users. The neighborhood for the target user, *c*, is formed by identifying the community members who have the highest combined value of trust, reputation, and similarity. By performing a simple summation of the three values, it gives an overall view into the past history of each user. If a user has a high trust value, high reputation value, and low similarity value, we need to take into account all three values in order to choose the best recommendation partners.

Once the neighborhood is formed, the rating prediction is generated using a modification of the Resnick formula. First, the trust value and reputation value are combined to produce a compound weighting, Eq. (6.2). This combined trust/reputation value, replaces the similarity value in the Resnick formula, all other variables remain the same, Eq. (6.3).

$$w(c, p) = \frac{2(rep(p))(T_c(p))}{rep(p) + T_c(p)} \tag{6.2}$$

$$P_{c,i} = \overline{r_c} + \frac{\sum_{p \in N} w(c,p)(r_{p,i} - \overline{r_p})}{\sum_{p \in N} |w(c,p)|} \tag{6.3}$$

## 6.3 Trust Propagation Strategy

The Trust Propagation strategy uses the same neighborhood formation technique as the Trust & Reputation method. All users in the neighborhood become the set of direct neighbors, *D*, for target user. Each direct neighbor, $d_i$, will propagate their trust values to their top-*m* most trustworthy neighbors using Eq. (5.7). The propagated trust values will be stored and used during the prediction process. These users become the secondary neighbors in the trust-network for the target user.

Once the trust-network is formed, the rating prediction is generated using Eq. (6.2) and (6.3). The only difference is that the secondary neighbors use the propagated trust values as input to Eq. (6.2).

Figure 2, gives an overview of the Trust Propagation strategy. Trust, reputation and similarity values are calculated from the historical ratings data. These values are used to build a neighborhood for the target user. The members in the neighborhood become the direct neighbors, who then propagate their trust values to other members in the community. This forms a trust-network for the target user, which is used to generate rating predictions and identify recommendable items.

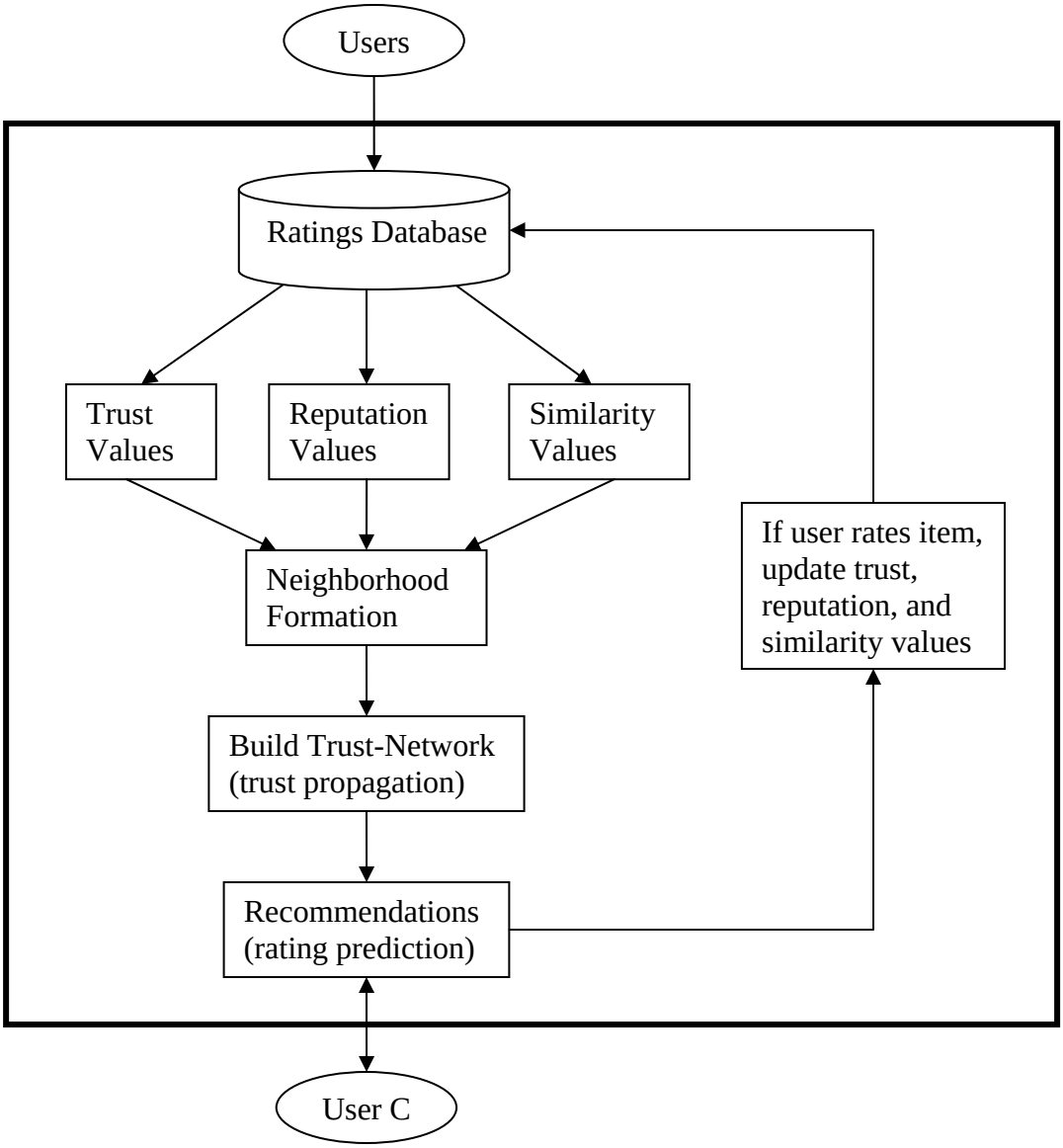


**Figure 2: Overview of Trust Propagation Strategy**

## 6.4 Neighborhood Formation

As discussed in Sec.3.3, traditional collaborative filtering uses the *k*-Nearest-Neighbors approach to identify the recommendation partners. This method identifies the top-*k* users based on the similarity, trust, and/or reputation values. These top-*k* users form the neighborhood for the target user. These neighbors are then used to generate rating predictions.

A second approach to neighborhood formation is to identify the target item first, and then form the neighborhood based on the particular item. The target item is the item we need to generate a prediction for. The idea is to filter out all users who have not rated the target item. If a user has not rated the target item, they are excluded from participating in the prediction process. This neighborhood formation method is called *k*-Nearest-Recommenders (*k*NR) [6]. The first step is to query the user database to identify all users who have rated the target item, and then the top-*k* neighbors are chosen based on the similarity, trust, and/or reputation values.

The *k*NR method will only consider users who have the required information to make a prediction. This differs greatly from the *k*-Nearest-Neighbor approach, where the target user's neighborhood remains static. The *k*NR method requires more computational overhead, because it dynamically builds a new neighborhood for the target user for each rating prediction. The benefit to *k*NR, is that once the neighborhood is formed, it is guaranteed that a rating prediction can be generated. In contrast, a neighborhood built with the *k*NN approach, must first ensure that at least one neighbor has rated the target item, and only then can a rating prediction be generated.

Both neighborhood formation methods will be implemented and tested during the evaluation of the trust-based recommendation strategies.

# 7. Evaluation

I argue that incorporating the social concepts of trust and reputation into recommender systems can lead to an increase in accuracy and quality of recommendations. The goal of the recommender system is to accurately predict what a user will rate an item. If the recommender system can predict a user's rating, then the recommender system can suggest items which the user will rate highly. This section presents experiments conducted for evaluating the performance of the proposed trust-based recommendation strategies.

## 7.1 Similarity Strategy

This strategy simulates the traditional collaborative filtering approach and will be used as a baseline. From the historical rating data, similarity values are calculated between each pair of users, Eq. (3.1). The neighborhood is formed by identifying the top-*k* most similar users. Finally, rating predictions are generated using the Resnick formula, Eq. (3.2).

## 7.2 Experimental Setup

In this experiment we have chosen to use the standard MovieLens dataset [8]. This dataset is the most widely used dataset for collaborative filtering research [9], thus, it will allow us to compare our results to the published research results of others. The MovieLens dataset was collected by the GroupLens Research Project at the University of Minnesota. The dataset consists of 100,000 ratings, assigned by 943 users on 1,682 movies. The users assign numerical ratings in the range of 1 to 5. The numerical ratings can be interpreted as follows: 1 – bad, 2 – average, 3 – good, 4 – very good, 5 – excellent. Each user profile has rated a minimum of 20 movies, with an average number of ratings of 106.

For each experiment, the MovieLens dataset is divided into a training set and test set. The ratings in the training set are used as explicit ratings from the user, which are contained as part of the historical rating database. The ratings in the test set are treated as unseen ratings that the system would attempt to predict. For each experiment iteration, the four recommendation strategies will generate a rating prediction on each item in the test set. By comparing the predicted rating with the actual rating we can measure the prediction accuracy of each recommendation strategy.

Before evaluating the accuracy of our proposed trust-based recommender system, we need to build up the trust values, reputation values, and similarity values. Usually the trust values would be built on-the-fly during the normal operation of the recommender system, but for our experiments we need to construct these values in advance using only the training data. The trust values are calculated by running a *leave-one-out* [3] training session. In the training dataset, we hide one rating at a time and then have each user make an individual rating prediction. By comparing the hidden rating with the predicted rating we can calculate a trust score between the two users.

Once the trust values have been constructed, we can calculate the reputation values, as described in Sec. 5.3. The similarity values are calculated using Pearson's correlation formula.

## 7.3 Metrics

The goal of the recommender system is to accurately predict what a user will rate an item. Therefore, the effectiveness of the recommender system is measured by the accuracy of the predictions that it makes. We will use two evaluation metrics to determine the overall effectiveness of the recommender system, namely the mean absolute error (MAE) and the total coverage of the system.

MAE measures the average absolute deviation between a predicted rating and the user's true rating [1]. A small value of MAE implies high prediction accuracy. MAE is a widely used and accepted metric for evaluating the performance of recommender systems [1]. In Eq. (7.1), $p_i$ represents the predicted rating, $r_i$ represents the user's actual rating, and $N$ represents the total number of items for which the recommender system made a prediction. MAE is a very simple, yet effective way to measure the prediction accuracy of the recommender system. It has been studied and reported that every 1 one-hundredth reduction in the MAE, provides a significant improvement in the quality of the recommendations [1].

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \tag{7.1}$$

Prediction accuracy alone does not fully express the effectiveness of the recommender system. We also need to take into account the usefulness of the recommender system by calculating the total coverage of the system. Coverage is the measure of the percentage of items for which a recommender system can provide predictions [1]. A recommender system may not be able to make predictions on every item. For instance, if a user has rated very few items, or if an item has been rated by very few users. These are the two main scenarios when a recommender system may be unable to generate a prediction for a user-item pair. A recommender system which has high prediction accuracy, but only on a small set of items, would not be very useful. Calculating coverage will give further insight into the usefulness of the recommender system. There are several ways to calculate coverage [1], for this research we compute coverage as number of items for which the recommender system can generate predictions, over the total number of item predictions that are requested. In Eq. (7.2), $p_i$ represents the prediction that the recommender system generated on item $i$, and $S$ represents the set of items for which the recommender system is generating a prediction.

$$Coverage = \frac{|\{p_i \mid i \in S\}|}{|S|} \tag{7.2}$$

Each recommendation strategy will generate a prediction for each rating in the test dataset, $S$. If the recommender system could not make a prediction, this will be captured during the calculation of coverage. Only items, for which the recommender system was able to generate a prediction, will be included in the calculation of MAE. Together, these metrics provide a measurement for the overall effectiveness of the recommender system.

## 7.4 Dataset Views

When evaluating the performance of recommender systems, the dataset is an important consideration. Collaborative filtering recommender systems incorporate learning algorithms which operate on statistical data [1]. The performance will vary based on the amount of learning data available. As the quantity of learning data increases, the quality of the predictions should increase as well. Different recommendation strategies will reach an acceptable MAE value at different rates. Some strategies may only need a small amount of data to make decent predictions, while other strategies may need a large amount of data [1].

In addition to the amount of data, the type of user is also significant. Several researchers have shown that an algorithm will perform differently for different types of users [4]. For example, in a real-world application there are users who will use the application very frequently and there are users who will use the application very rarely.

We will test the recommendation strategies on five different types of input data. Each type of input data will provide a different view and insight into how the recommender system performs. First, we will look at new users, who are users that provided only 5 ratings. For this scenario, the recommender system will be making rating predictions for a target user, based on very little knowledge about that user. Second, we will look at heavy users, who are users that provided 50 or more ratings. For the remaining dataset views, we will randomly split the data into the training set and test set, with each view having a different percentage of training data. The three views will be split as follows: 80% training/20% test data, 50% training/50% test data, and 10% training/90% test data. These dataset views will give an indication for the learning rate of the recommendation strategies.

To validate our results, we will perform each experiment five times. Therefore, five distinct dataset are created for each of the five dataset views: “new user” dataset, “heavy user” dataset, 10/90 data split, 50/50 data split, and 80/20 data split. This creates a total of 25 distinct datasets with each dataset being composed of a training set and a test set. Each recommendation strategy will make rating predictions for each item in the test set. The results will be averaged over the five experiment runs for each dataset view.

As was discussed in Sec. 6.4, the *k*-Nearest-Neighbor and *k*-Nearest-Recommender neighborhood formation methods will both be implemented and tested during the evaluation of the trust-based recommendation strategies. With five datasets for each of the five dataset views, four recommendation strategies, and two neighborhood formation techniques, there will be a total of 200 experiments.

Based on what other research [6] has found, and my own preliminary experiments, the *k*-value for this experimentation is set at $k = 60$.

## 7.5 Results

The first set of results was generated using the *k*-Nearest-Neighbor neighborhood formation technique.

The MAE results from the 80/20 data split are shown in Table 1. The average MAE was calculated from the five experiment runs. As you can see, the MAE gradually improved as the recommendation strategy became more complex. The Trust Propagation strategy performed the best with an average MAE of 0.7619. Propagating trust values only improved the MAE slightly over the Trust & Reputation strategy. This is because 80% of the data was used for training the recommendation algorithm. Propagating trust values has a more significant impact when there is less data available.

| | **Similarity** | **Trust** | **Trust & Reputation** | **Trust Propagation** |
|---|---|---|---|---|
| **80/20 Test 1** | 0.9076 | 0.8507 | 0.7917 | 0.7763 |
| **80/20 Test 2** | 0.9025 | 0.8467 | 0.7804 | 0.7630 |
| **80/20 Test 3** | 0.9061 | 0.8420 | 0.7755 | 0.7573 |
| **80/20 Test 4** | 0.9050 | 0.8453 | 0.7708 | 0.7534 |
| **80/20 Test 5** | 0.8996 | 0.8464 | 0.7798 | 0.7595 |
| **Average** | **0.9041** | **0.8462** | **0.7796** | **0.7619** |

**Table 1: MAE values for the 80% training/20% test, dataset view.**

Table 2 summarizes the coverage results from the 80/20 data split. The average coverage was calculated from the five experiment runs. Similar to the MAE, the coverage also improved as the recommendation strategy became more complex. The Trust Propagation strategy performed the best with an average coverage of 99.07%. The Trust Propagation strategy significantly improved the coverage when compared to the Similarity strategy, which only provided coverage of 57.59%. This means that the Similarity strategy could only generate predictions for 57.59% of the items in the test set, indicating that the recommender system would be far less useful on a wide array of items.

| | **Similarity** | **Trust** | **Trust & Reputation** | **Trust Propagation** |
|---|---|---|---|---|
| **80/20 Test 1** | 58.05% | 72.45% | 96.62% | 99.13% |
| **80/20 Test 2** | 57.63% | 71.79% | 96.27% | 99.01% |
| **80/20 Test 3** | 57.46% | 71.87% | 95.54% | 98.92% |
| **80/20 Test 4** | 57.21% | 71.60% | 95.83% | 99.24% |
| **80/20 Test 5** | 57.62% | 71.46% | 95.11% | 99.04% |
| **Average** | **57.59%** | **71.83%** | **95.87%** | **99.07%** |

**Table 2: Coverage calculations for the 80% training/20% test, dataset view.**

Instead of showing the results of the five experiment runs for each of the five dataset views, I consolidate the average MAE and average coverage for each dataset view into tables 3 and 4, respectively.

| | **Similarity** | **Trust** | **Trust & Reputation** | **Trust Propagation** |
|---|---|---|---|---|
| **New Users** | 0.8901 | 0.8664 | 0.8535 | 0.8506 |
| **Heavy Users** | 0.9316 | 0.8648 | 0.7856 | 0.7665 |
| **10/90 data split** | 0.9429 | 0.9368 | 0.9214 | 0.8596 |
| **50/50 data split** | 0.9417 | 0.8853 | 0.8122 | 0.7781 |
| **80/20 data split** | 0.9041 | 0.8462 | 0.7796 | 0.7619 |
| **Average** | **0.9221** | **0.8799** | **0.8305** | **0.8033** |

**Table 3: Average MAE values for different recommendation strategies on different dataset views.**

Looking at each individual recommendation strategy, we can see how the performance of a recommendation algorithm depends on the input data. The results from the 10/90, 50/50, and 80/20 data splits, indicate how fast the algorithms "learn" and begin to generate more accurate predictions. With only 10% training data the Similarity strategy performs poorly with an MAE of 0.9429. By increasing the training data to 80% the similarity-based algorithm provides an average error of 0.9041. There is not a huge improvement in prediction accuracy, but this is an expected result, since one of the known drawbacks of using similarity values is that sparse data means unreliable neighborhood formation. This means the recommender system is unable to find highly similar users for the target user, thus the rating prediction is less accurate.

The trust-based recommendation algorithms also benefit from more training data, however, the improvement in prediction accuracy is much more significant. The Trust & Reputation strategy provides the largest reduction in MAE when the training data increases from 10% to 80%. But it is the Trust Propagation strategy which is able to provide the best MAE with the least amount of data. For the 10/90 and 50/50 data splits the Trust Propagation strategy significantly outperforms all other recommendation strategies. This clearly indicates that there is a benefit to expanding the trust-network to include users in the community, who the target user may not have had previous interactions with.

When comparing the two most complex recommendation strategies, Trust & Reputation vs. Trust Propagation, we see some interesting results. When there is a larger amount of training data the usefulness of propagating trust values decreases. This happens because more data means more accurate trust and reputation values. The Trust & Reputation strategy is able to form a reliable trust-network without having to propagate trust values. For the 10/90 data split, the MAE for Trust & Reputation is 0.9214 and the MAE for Trust Propagation is 0.8596. As mentioned above, the Trust Propagation strategy performs very well with a small amount of data. For the 80/20 data split, the MAE for Trust & Reputation is 0.7796 and the MAE for Trust Propagation is 0.7619. With a large amount of data, the Trust & Reputation strategy performs very well and the benefit of propagating trust values is minimal.

The new user and heavy user dataset views provide insight into the performance of each recommendation strategy on a specific type of user. For the heavy user dataset we see the same trend as the other dataset views. As the recommendation strategy increases in complexity, the MAE value decreases which represents an improvement in prediction accuracy.

The new user dataset shows some interesting results. First, we see a very minimal difference in the MAE value across each of the recommendation strategies. The MAE values range from 0.8901 to 0.8506. These results show that it is very difficult to make accurate recommendations for a user who has provided little information. Utilizing the trust values does provide an improvement over the similarity-based algorithm. And with the inclusion of reputation values, the Trust & Reputation strategy provides a slight improvement in prediction accuracy. When there is little knowledge about the user, it is the reputation which provides invaluable information about the community. A recommender system cannot always generate a similarity-network or trust-network, but the recommender system can always generate a network of reputable community members. Remember that similarity and trust values are calculated between pairs of users, but reputation values are globally available for the entire community to see.

Another interesting result from the new user dataset is the performance of the Similarity strategy. The Similarity strategy provided its most accurate predictions on the new user dataset. This was an unexpected result, because a recommender system should perform better as the amount of data increases. The heavy users have provided 50 or more ratings, and therefore should have more reliable similarity values with other community members. The new users have provided 5 ratings and the similarity values would be less reliable. Since the Similarity strategy performed worse for the heavy users, it reinforces the fact that similarity is not a good measurement for choosing recommendation partners. The MAE for the new user dataset and the heavy user dataset was and 0.8901 and 0.9316, respectively.

The coverage values, shown in Table 4, follow a similar pattern to the MAE values. As the recommendation strategy increases in complexity, the coverage increases as well. The

most notable result from the coverage calculations is how poorly the traditional collaborative filtering approach performs and the improvement we see with the Trust Propagation strategy. For the 10/90 data split, the similarity-based algorithm provides a coverage of 65.22% and the Trust Propagation strategy improves the coverage to 94.26%. The 50/50 and 80/20 data splits also provide similar results. There are two main reasons why a recommender system would not be able to generate a rating prediction for a particular user-item pair: the system cannot identify similar or trustworthy neighbors for the target user or none of the neighbors have rated the target item. The first issue arises during the neighborhood formation process. If a user has provided no ratings or very few ratings, it is difficult to identify similar or trustworthy neighbors. The second issue arises during the rating prediction process. If the recommender system is able to form a network, one of the neighbors must have rated the target item or else a prediction cannot be generated.

| | **Similarity** | **Trust** | **Trust & Reputation** | **Trust Propagation** |
|---|---|---|---|---|
| **New Users** | 93.40% | 96.42% | 98.34% | 98.79% |
| **Heavy Users** | 61.91% | 75.20% | 93.84% | 98.32% |
| **10/90 data split** | 65.22% | 68.52% | 74.92% | 94.26% |
| **50/50 data split** | 53.39% | 66.66% | 92.21% | 98.56% |
| **80/20 data split** | 57.59% | 71.83% | 95.87% | 99.07% |
| **Average** | **66.31%** | **75.73%** | **91.03%** | **97.80%** |

**Table 4: Average coverage values for different recommendation strategies on different dataset views.**

With the introduction of trust values, we see a slight improvement in coverage over the Similarity strategy. Both the similarity values and trust values are calculated between each pair of users based on co-rated items. The difference is that trust values are measuring the ability of a neighbor to make accurate recommendations in the past. The reputation values offer a different perspective of the community. The reputation value measures how the entire community views an individual and co-rated items do not matter. The most reputable community members can always be identified and used as recommendation partners.

With the Trust strategy we see some unexpected results. The logical expectation is to see the coverage improve as the amount of training data increases. This expectation holds

true for all strategies, except the Similarity and Trust strategy. For the 10/90 data split the coverage is 68.52% and for the 50/50 data split the coverage is 66.66%. It is a marginal difference, but it is also indicative of something more. The coverage for the new user dataset is also much higher than expected, at 96.42%. What these results show, is that building a trust-network with the most trustworthy users, does not always have a positive impact on coverage. Building a network of highly trustworthy user, does not mean that the trust-network will be able to make predictions on a wide array of items. There is an element of unpredictability as to which items the trust-network will need to make predictions on. This problem of unpredictable coverage could be solved by ensuring that members of the trust-network cover a higher percentage of items. However, this method would require a more complex and resource intensive algorithm.

The Trust Propagation strategy is an alternative way to reliably increase the coverage without the need for a complex algorithm. For each dataset view, the Trust Propagation strategy provides a coverage of 98-99%, except for the 10/90 data split, where the coverage is 94.26%. The 94% coverage for the 10/90 data split is still significantly better than the next highest coverage value of 74.92%.

The new user and heavy user datasets provide interesting coverage results, specifically for the Similarity strategy. The Similarity strategy can generate a rating prediction for 93.40% of the items for new users and only 61.91% of the items for heavy users. This is an unexpected result, since a user who has provided more ratings should receive more accurate recommendations on a wider array of items. A deep analysis of the data revealed some interesting findings. First, I chose user id 1, as the target user in the heavy user dataset. User 1 had 50 ratings in the training set. The similarity-network for user 1 contained 60 neighbors who each had a similarity value of 1.0, the highest possible value. The fact that so many community members had a similarity value of 1.0 reveals another weakness of similarity. Similarity is calculated based on co-rated items, which means that user X and user Y can both have a similarity value of 1.0 with user 1, even though user X co-rated ten items with user 1, and user Y co-rated one item with user 1. Both user X and user Y are considered highly similar to user 1. If two users only rated one item in common the similarity value is not very reliable, yet the recommendation algorithm does not take that into account.

Going back to the analysis of user 1, I find that user 1's similarity-network had an average of 1.27 co-rated items. When looking at the entire community, the average number of co-rated items with user 1 was 7.33. Even though there were community members who co-rated many items with user 1, the similarity-network was full of members who co-rated the fewest items. In addition, the average number of ratings for the users in the similarity-network was 32.97. While the average number of ratings for the entire community was 93.46. This is the reason for the low coverage value on the heavy user dataset. The similarity-network is composed of community members who rated the fewest number of items. This means the network will be unable to make a rating prediction for a wide array of items.

In the analysis of the new user dataset, target user 5 had many similar users in the community. Specifically, there were 89 community members who had a similarity value of 1.0. Similar to the example above, user 5's similarity-network had an average of 1.13 co-rated items. However, the average number of ratings for the users in the similarity-network was 92.02, much higher than the previous example. This is why the coverage for the new user dataset is much higher than the coverage for the heavy user dataset.

To give a visual indication of the performance of the recommendation strategies, Figure 3 shows the average MAE for each recommendation strategy, and the relative benefit compared the traditional similarity-based strategy. As the recommendation algorithms increase in complexity, the relative benefit increases as well. The Trust strategy provides a 5% improvement, the Trust & Reputation strategy provides a 10% improvement, and the Trust Propagation strategy provides a 13% improvement in prediction accuracy.

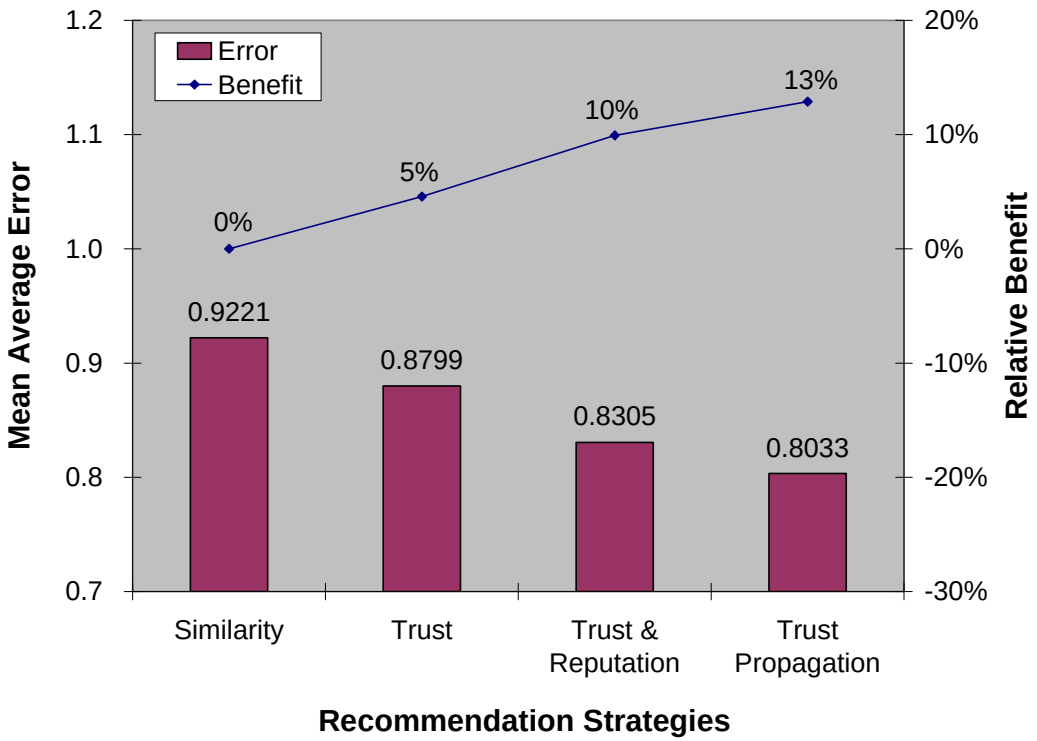


**Figure 3: The average prediction error and relative benefit (compared to Similarity) of each recommendation strategy.**

The second set of results was generated using the *k*-Nearest-Recommender neighborhood formation technique. Recall that *k*NR is more complex and dynamically builds the network depending on which item the recommender system is generating a prediction for. This is different from the *k*NN strategy where each user has a static neighborhood. The

neighborhood for the target user stays the same, no matter what the target item is. The MAE and coverage results from the *k*NR method are shown in tables 5 and 6, respectively.

| | **Similarity** | **Trust** | **Trust & Reputation** | **Trust Propagation** |
|---|---|---|---|---|
| **New Users** | 0.8677 | 0.8335 | 0.8346 | 0.8364 |
| **Heavy Users** | 0.7791 | 0.7472 | 0.7400 | 0.7473 |
| **10/90 data split** | 0.8892 | 0.8367 | 0.8417 | 0.8370 |
| **50/50 data split** | 0.7847 | 0.7513 | 0.7550 | 0.7578 |
| **80/20 data split** | 0.7674 | 0.7401 | 0.7420 | 0.7493 |
| **Average** | **0.8176** | **0.7818** | **0.7827** | **0.7856** |

**Table 5: Average MAE values for different (*k*NR) recommendation strategies on different dataset views.**

The major difference that we see in the *k*NR results is that the MAE values do not vary greatly across the recommendation strategies. As discussed in Sec. 7.3, every 1 one-hundredth reduction in the MAE provides a significant improvement in the quality of the recommendations. Therefore, as we compare the Similarity strategy and the Trust strategy, we see an average reduction in error of 0.036, or 3 one-hundredths. This means the Trust strategy will significantly improve the quality of recommendations over the Similarity strategy.

Another interesting result, is that we do not see a consist pattern of the more complex recommendation algorithms providing a higher prediction accuracy. For the heavy user dataset, the Trust strategy has average error of 0.7472. The Trust & Reputation strategy performs slightly better, 0.7400, and the Trust Propagation strategy performs slightly worse, 0.7473. For the 10/90 data split, we see the opposite result. The Trust Propagation strategy performs slightly better than the Trust & Reputation strategy. These results indicate that the neighborhood formation process is a critical component to the recommender system. For the *k*NR method, we first determine what the target item is, and then we dynamically identify direct neighbors depending on the item. Only community members who have rated the target item will be considered as a potential neighbor. With this method, the trust-network that is formed with the Trust strategy is more reliable in generating accurate predictions, than the reputation-network or the propagated-trust-network. This makes logical sense, because we are dealing with a small

subset of community members, and the trust values would still be representative of the trustworthiness of the user. On the other hand, the reputation values are a measure of how an individual user compares to the rest of the community. If we are only evaluating a small subset of the community, then the reputation values are a far less reliable indicator, as to how reputable the user is. Furthermore, the propagated trust values are no longer improving the quality of the trust-network. The *k*NR method is already considering all members who have rated the target item; therefore propagated trust values provide no additional benefit. This is why we see the Trust Propagation strategy performing worse than the Trust strategy over all dataset views. We see that the Trust strategy consistently outperforms the more complex recommendation algorithms. Also, as the amount of training data increases the prediction accuracy increases as well. This is an expected result, since more training data will improve the reliability of the trust values.

The *k*NR method for neighborhood formation is more complex, and would require more resources; however, it does provide a significant reduction in error over the *k*NN method. This presents a trade-off situation, in which the implementers of the recommender system would need to decide if prediction accuracy or minimizing data processing overhead was more important.

| | **Similarity** | **Trust** | **Trust & Reputation** | **Trust Propagation** |
|---|---|---|---|---|
| **New Users** | 97.52% | 99.79% | 99.79% | 99.79% |
| **Heavy Users** | 99.49% | 99.69% | 99.69% | 99.69% |
| **10/90 data split** | 79.45% | 95.14% | 95.14% | 95.14% |
| **50/50 data split** | 99.43% | 99.66% | 99.66% | 99.66% |
| **80/20 data split** | 99.72% | 99.83% | 99.83% | 99.83% |
| **Average** | **95.12%** | **98.82%** | **98.82%** | **98.82%** |

**Table 6: Average coverage values for different (*k*NR) recommendation strategies on different dataset views.**

When using the *k*NR method, the coverage values are no longer a major consideration in the evaluation of various recommendation strategies. As you can see, the coverage values are exactly the same across each of the trust-based recommendation strategies. Propagating trust values is no longer providing an advantage. Propagating trust increases the network size, to increase the item coverage. However, the *k*NR method already

guarantees the best possible coverage for the recommendation strategies. The *k*NR method determines what the target item is, and then dynamically identifies direct neighbors depending on the item. The *k*NR method will query for all community members who rated the target item, before forming the neighborhood. This means that once the neighborhood is formed, the network can definitely generate a rating prediction. There are few cases where the *k*NR method is unable to find at least one neighbor for the target user; this is evident in the high coverage values.

When comparing the Similarity strategy to the Trust Strategy, we see a slight improvement in the coverage. The biggest advantage is gained when there is a small amount of training data. For the 10/90 data split, the Similarity strategy provides 79% coverage, while the Trust strategy provides 95% coverage.

To give a visual indication of the performance of the *k*NR recommendation strategies, Figure 4 shows the average MAE for each strategy, and the relative benefit compared to the traditional similarity-based strategy. As the recommendation algorithms increase in complexity, the relative benefit decreases. The Trust strategy provides a 4.4% improvement, the Trust & Reputation strategy provides a 4.3% improvement, and the Trust Propagation strategy provides a 3.9% improvement in prediction accuracy.

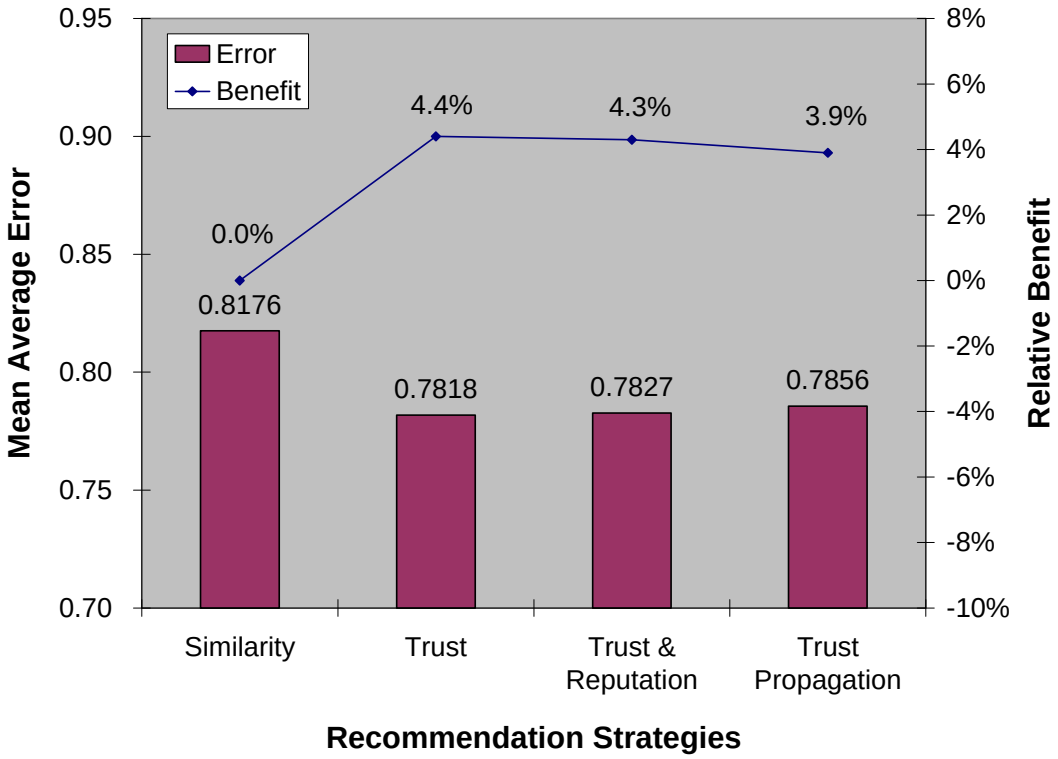


**Figure 4: The average prediction error and relative benefit (compared to Similarity) of each (*k*NR) recommendation strategy.**

## 8. Summary of Results

In this section I discuss the most important results observed from the experiments.

For the traditional *k*NN approach to neighborhood formation, we see that the trust-based recommendation strategies provide a significant improvement in prediction accuracy. Specifically, the Trust Propagation method provided the best coverage and lowest MAE, across all dataset views. The Trust Propagation strategy had a 13% reduction in error, compared to the traditional similarity-based recommendation strategy. With regard to coverage, the Similarity strategy completely failed in comparison to the trust-based strategies, and was unable to generate predictions for a large portion of the test set. The Trust Propagation method provided an average coverage of 97.80%.

For the *k*NR approach to neighborhood formation, the more complex recommendation strategies do not provide any benefit. Specifically, the Trust & Reputation and Trust Propagation methods both performed worse than the simple Trust strategy. In addition, the coverage is far less of a concern when using the *k*NR method. The nature of the *k*NR method provides a high coverage across all recommendation strategies. It is important to note that the MovieLens dataset is a very dense dataset. All users have provides at least 20 ratings and the average number of ratings is 106. The Epinions.com dataset is another widely used dataset for collaborative filtering research. The Epinions.com dataset greatly differs from the MovieLens dataset [4]. For instance, 52.82% of the users are new users, who provided less than 5 ratings, and 45% of the ratings are 5-stars [4]. The ratings in the MovieLens dataset are more balanced and since the average user provided a large number of ratings, it is easier to find trustworthy neighbors.

We can compare the performance of our proposed trust-based recommender system, to the published research results of others. In order to perform this comparison, we need to ensure that the same dataset was used and the same performance metrics were used. It has been studied and reported that collaborative recommender systems reach a “magic barrier” where natural variability may prevent us from getting much more accurate [1]. The barrier is around the MAE values of 0.72-0.74 [1]. Other researchers have reported similar MAE values: Sarwar et al. [11] got 0.72, Li and Kim [12] got 0.735, Campos et al. [9] got 0.7357, and Bharadwaj et al. [13] got 0.72. We can conclude that our model is very competitive with other published research results. For the 80/20 data split our trust-based recommender system provided a MAE of 0.74.

## 9. Future Work

Several researchers have shown that certain algorithms will perform better or worse, depending on the dataset. For instance, Massa and Avesani in [4] performed experiments using a dataset derived from Epinions.com. Their analysis found that the Epinions dataset

differs greatly from the MovieLens dataset. As future work, the proposed trust-based model should be further validated by experimenting with more datasets.

## 10. Conclusion

Trust is a concept that is starting to receive increasing attention by the research community and is being implemented in many online systems [4]. In this paper I argue that incorporating trust and reputation into recommender systems can lead to an increase in accuracy and quality of recommendations. Based on this idea, I proposed a trust model which implicitly quantifies the degree of trust a user holds for another user, and proposed new trust-based recommendation strategies which incorporate the new model into the standard collaborative filtering recommender system.

I evaluated three trust-based recommendation strategies against the standard MovieLens dataset. For a baseline comparison, I used the traditional similarity-based collaborative filtering strategy. The empirical results indicate that trust, reputation, and trust propagation are very effective tools for improving recommender systems. Specifically, Trust Propagation was the best performing strategy reducing the average error by 13% compared to the benchmark.

# 11. References


[1] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst*. 22 (1). pp. 5–53. 2004.

[2] J. Golbeck, J. Hendler. Accuracy of metrics for inferring trust and reputation in semantic web-based social networks. In *Proceedings of EKAW'04*, LNAI 2416, p. 278, 2004.

[3] J. O'Donovan, B. Smyth. Trust in Recommender Systems. *Proceedings of the 10th international conference on Intelligent user interfaces.* January 2005.

[4] P. Massa, P. Avesani. Trust-aware Recommender Systems. *RecSys'07.* October 19–20, 2007.

[5] E. Aimeur, F.S.M. Onana. Better Control on Recommender Systems. *Proceedings of the 8th IEEE International Conference on E-Commerce Technology*. 2006.

[6] N. Lathia, S. Hailes, L. Capra. Trust Based Collaborative Filtering. *In IFIPTM 2008: Joint iTrust and PST Conferences on Privacy, Trust management and Security*. Trondheim, Norway. June 2008.

[7] J.S. Breese, D. Heckerman, C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *14th Conf. on Uncertainty in Artificial Intelligence*. pp. 43–52. 1998.

[8] MovieLens dataset. GroupLens Research Project, University of Minnesota.

[9] L.M. de Campos, J.M. Fernández-Luna, J.F. Huete. A collaborative recommender system based on probabilistic inference from fuzzy observations. *Fuzzy Sets and System*s. 159:1554 – 1576, 2008.

[10] A. Josang, R. Ismail, C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*. 43:618–644, 2007.

[11] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: *Proc. ACM World Wide Web Conf., 2001*, pp. 285–295, 2001.

[12] Q. Li, B. Kim, Clustering approach for hybrid recommender system, in*: IEEE/WIC Proc. Internat. Conf. onWeb Intelligence*, pp. 33–38, 2003.

[13] K.K. Bharadwaj, M.Y.H. Al-Shamri. Fuzzy computational models for trust and reputation systems. *Electronic Commerce Research and Applications*, In Press, Corrected Proof, Available online 14 August 2008.

[14] P. Resnick, H.R. Varian. Recommender systems. *Communications of the ACM*, 40 (3) pp. 56–58, 1997.

[15] C. Cornelis, J. Lu, X. Guo, G. Zhang. One-and-only item recommendation with fuzzy logic techniques. *Information Sciences*. 177:22, pp. 4906-4921, Nov. 2007.

[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. *In Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, Chapel Hill, North Carolina, pp. 175–186, 1994.

[17] J. Weng, C. Miao, A. Goh. Improving Collaborative Filtering with Trust-based Metrics. *ACM SAC*. 2006.

[18] R. Guha, R. Kumar, P. Raghavan, A. Tomkins. Propagation of Trust and Distrust. *WWW2004*, May 17–22, 2004, New York, New York, USA.

[19] G. Pitsilis, L. Marshall. A Trust-enabled P2P Recommender System. *In Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06).* 2006.

[20] L. Mui, M. Mohtashemi, A. Halberstadt. A Computational Model of Trust and Reputation. *Proceedings of the 35th Hawaii International Conference on System Sciences.* 2002.