electronics

MDPI

Article

AgreeRelTrust—A Simple Implicit Trust Inference Model for Memory-Based Collaborative Filtering Recommendation Systems

Ahmed Zahir *, Yuyu Yuan and Krishna Moniz

Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, School of Software, Beijing University of Posts and Telecommunications, Beijing 100876, China; yuanyuyu@bupt.edu.cn (Y.Y.); krishnamoniz@live.nl (K.M.)

* Correspondence: xahiru@gmail.com; Tel.: +86-131-204-4403

Received: 8 March 2019; Accepted: 9 April 2019; Published: 11 April 2019

Abstract: Recommendation systems alleviate the problem of information overload by helping users find information relevant to their preference. Memory-based recommender systems use correlation-based similarity to measure the common interest among users. The trust between users is often used to address the issues associated with correlation-based similarity measures. However, in most applications, the trust relationships between users are not available. A popular method to extract the implicit trust relationship between users employs prediction accuracy. This method has several problems such as high computational cost and data sparsity. In this paper, addressing the problems associated with prediction accuracy-based trust extraction methods, we proposed a novel trust-based method called AgreeRelTrust. Unlike accuracy-based methods, this method does not require the calculation of initial prediction and the trust relationship is more meaningful. The collective agreements between any two users and their relative activities are fused to obtain the trust relationship. To evaluate the usefulness of our method, we applied it to three public data sets and compared the prediction accuracy with well-known collaborative filtering methods. The experimental results show our method has large improvements over the other methods.

Keywords: implicit trust; collaborative filtering; recommendation system

1. Introduction

A recommendation system (RS) is a powerful tool that assists online users to find the information most relevant to their preferences. The enormous number of products and services available online makes it very difficult for users to find useful information. An RS serves as an automated solution providing a better user experience. For example, personalized suggestions for items on eBay [1] or videos on Youtube [2] are results of an RS.

The recommendation strategies are broadly classified into three categories: content-based recommendation, collaborative filtering (CF), and hybrid recommendation. Content-based methods rely on the similarity of the attributes of the items the user has interacted in the past. For instance, if a user on Youtube has viewed many mathematics lecture videos, then the Youtube recommendation system would suggest the user subscribe to channels categorized as education. In CF, the preferences of similar users are aggregated to predict a personalized recommendation [3]. The intuition is that users who had similar preferences in the past will have similar preferences in the future. Preference similarities are calculated as a correlation between users' rating profiles. For example, Youtube uses 'likes' provided by users to find other users who have similar preferences. On the basis of the similarity, the system offers movie recommendations to the users. Hybrid methods combine both content-based approaches and CF.

Many researchers have tried to improve the prediction accuracy of recommendations using social aspects [4–7]. It is natural to think that we are very likely to accept recommendations obtained from trusted partners. To this end, trust is often used to find similar preferences among users in CF. However, for various reasons, direct trust relationships are difficult to obtain. Instead, implicit trust relationships are inferred from the preferences.

Most of the trust inference methods require predictions to be calculated before generating the trust matrix, whereas others rely on a similarity matrix. Inspired by a model proposed by O'Donovan et al. [8], we propose AgreeRelTrust, in which trust is based on users' agreements and their relative activity and does not require prior predictions. The users' positive and negative agreements are used to generate trust between them. The significance of the study is as follows. (1) A novel model is proposed that uses trust to increase the prediction accuracy of memory-based CF recommendation systems. (2) A new method for inferring trust using explicit rating data is proposed. In the proposed method, the trust calculation does not require prior prediction; hence, it is more efficient than traditional trust-based methods. The method accounts for the direction of the user's preferences, which is a much closer representation of the real world. (3) The efficiency of the model is evaluated using three publicly available real-world data sets.

The rest of the paper is organized as follows. Next, we discuss the related work found in the literature. The following section details the problem statement and formalizes our model. The experimental section provides the details of the datasets and the evaluation metrics. In the results section, we compare the performance of the proposed method with the baseline method. We conclude the paper by highlighting possible future work references.

2. Related Work

Generally, CF algorithms are classified into two categories: (1) memory-based methods, which use profile similarity to make predictions; and (2) model-based methods, which build a model of users using profiles. The models are then used to make predictions. One of the simplest memory-based methods is k-nearest neighbor (in), in which the k-nearest neighbor profiles of the target user are chosen to produce recommendations. Matrix factorization is a model-based method and is well-known for its success after winning the famous Netflix competition [9]. The algorithm relies on model parameters and does not require iterating through the rating matrix to make predictions.

Although state-of-the-art recommendation systems are heavily dependent on deep learning models, which is a sub-category of model-based CF, memory-based CF systems are still widely used because of their simplicity. For a summary of recent works using deep learning in RSs, interested readers can refer to Signal et al. [10]. Memory-based CF methods have been widely researched and are available in almost all software implementations of recommendation frameworks, such as Librec [11] and Lenskit [12]. They represent one of the most common types of personalized RSs used in E-commerce platforms on the Web [13]. Therefore, in this work, we solely focus on memory-based methods.

The most common similarity measures used in memory-based methods include Pearson, Cosine, and Jacquard correlations [14,15]. Correlation-based CF systems suffer from various problems such as cold starting and data sparsity. Cold starting is a situation in which finding similar users is difficult because of a lack of ratings [16,17]. To address the issues with correlation-based similarity measures, trust is often used as a replacement. The intuition behind using trust is that users are more likely to accept recommendations from a trustworthy partner.

Trust in computer models has been successfully incorporated in various contexts, such as reputation systems, for example, Amazon.com [18], dynamic networks [19], and mobile environments [20]. For a detailed survey of various computational trust models, interested readers can refer to Jøsang et al. [21]. Researchers have incorporated many aspects of social relationships between users to increase the efficiency of recommendation systems. Trust has become one of the key avenues for this exploration. A growing amount of work is being completed on trust-based recommendation systems. The use of trust in recommendation systems has been shown to alleviate problems such as cold starting and data

sparsity [22]. Trust has been used to increase explainability, as well as to improve the robustness of CF recommendation systems [23].

In general, trust computation methods are broadly classified into two types: implicit trust, where the trust between two users is inferred from their rating profiles; and explicit trust, where existing social links are used for trust inference. The goal of both methods is to use the underlying trust relationship to aggregate user preferences so that more weight is assigned to trustworthy partners.

Although, intuitively, it is more logical to use explicit trust for prediction, in many real-world recommendation systems, obtaining social link data of users is difficult. For instance, many online shops do not require users to be registered to buy products. In addition, using social network data increases the risk of compromising users' privacy [24]. The number of ratings available is far greater than that of explicit trust links, which are often available in the form of binary values. It is possible to generate real values for the binary data; however, this could add noise to the data.

Similar to traditional brick-and-mortar businesses, trust plays a vital role in the success of e-commerce businesses [19]. Massa and Avesani [25] showed that incorporating trust increased the efficiency of a recommendation system. Several trust inference models have been proposed to increase the accuracy of recommendation systems using explicit trust links [17,22,26,27]. The availability of explicit trust information is relatively low compared with that of ratings. The focus of our study was implicit trust inference.

O'Donovan et al. [8] proposed a method that uses recommender's contributions to prediction accuracy as a measure of trust. For each pair of users, the recommender's contribution is calculated as the ratio of the number of successful recommendations and the total number of recommendations made by the same user. The predictions are completed using only the pair's ratings. A recommendation is successful only if the predicted rating is very close to the actual target rating. The higher the contribution to the prediction accuracy, the more the trust between two users increases. The Resnick prediction formula [28] is used to calculate the predictions. Instead of using the similarity matrix, the newly generated trust matrix is then used to produce final recommendations. The main disadvantage of this method is the time required for trust matrix generation. For each user, the algorithm requires a prediction to be made for all other users to calculate the absolute difference between the predicted rating and ground truth. Also, the method does not consider the direction of the two users' preference, but simply the accuracy using the mean absolute error (MAE). In contrast, our proposed trust model considers the direction of the agreement. Our trust calculation does not rely on making predictions.

Quasit et al. [29] proposed an implicit-trust based CF method, dubbed as hybrid user-item trust (HUIT), addressing the issues of data sparsity and cold start. The method combines predictions obtained using a user-based trust matrix with predictions obtained using an item-based trust matrix to make final predictions. Similar to O'Donovan et al. [8], the trust calculation involves making initial predictions. In addition to that, the trust matrices calculation in this method has a higher running time than that of O'Donovan et al. [8], because it is done twice; one for each trust matrix. On the other hand, in our method, trust is directly inferred from the ratings, making it much more efficient.

Addressing the issues of using similarity in k-nearest neighbor (kNN) CF, Lathia et al. [30] proposed a similar trust-based kNN CF method. In this method, the absolute prediction difference between two users is subtracted from one, before being divided by the maximum of the rating scale. The average of the result is taken as a trust value between the users. The trust value ranges from zero to one. If the target user does not have co-rated items, then the trust between them is zero. The predictions are then produced using the trust matrix rather than the similarity. This method assumes a recommender with negative similarity correlation is more trustworthy to the target user than those who have not yet rated the item. Similar to O'Donovan et al. [8], this method depends on a prediction being calculated to generate the trust matrix.

Similarly, Pitsilis and Marshall [31] derived trust by measuring the uncertainty in the similarity values. The users' inability to make accurate predictions is modeled as uncertainty. The similarity matrix is then scaled according to the user's belief and disbelief of the rating provider (also sometimes

referred to as the trustee). The sum of belief, disbelief, and uncertainty is one. Although the inclusion of belief in this model inclines to subjective probability, the essence of prediction depends on the correlation of users. In contrast with our method, this approach relies on a similarity matrix for trust value generation.

Li et al. [32] improvised the model proposed in O'Donovan et al. [8] by including preference similarity, recommendation trust, and social relations into the recommendation algorithm. In the recommendation trust analysis module, implicit trust is calculated in exactly the same manner as in O'Donovan et al. [8]. Therefore, we argue that replacing their trust module with ours would increase the performance of the recommendation system. In the next section, we discuss, in detail, the problems associated with the prediction-based methods similar to O'Donovan et al. [8].

3. Proposed Model

3.1. Problem Statement

Let $u(i),\ v(i) \in \{1,\ 2,\ 3,\ \ldots, r_{max}\}$ be the ratings of users $u,\ v \in U$ for item i $\in I$ in the rating matrix $R_{|U| \times |I|}$, and r_{max} be the maximum of the rating scale. Given the rating history of the users, the goal is to predict ratings as accurately as possible for the items that the user has not yet evaluated. Often, the predictions are calculated using the k-nearest neighbor method:

$$\hat{u}(i) = \frac{\sum_{v \in N_i^k(u)} (v(i) - \overline{v}) \times \text{similarity}(u,\ v)}{\sum_{v \in N_i^k(u)} \text{similarity}(u,\ v)}, \tag{1}$$

where $\hat{u}(i)$ is the predicted rating of item i for the user u, $\overline{v}$ is the mean of the ratings of all the items rated by the user v, and N_i^k is the set of k neighbors of u who have rated item i. The similarity between u and v using the Pearson correlation is given by

$$\text{similarity}(u,v) = \frac{\sum_{i \in I}(u(i) - \overline{u})(v(i) - \overline{v})}{\sqrt{\sum_{i \in I}(u(i) - \overline{u})^2 \sum_{i \in I}(v(i) - \overline{v})^2}}. \tag{2}$$

In Equation (2), the similarity between two users is calculated based on the correlation of their commonly rated items, and $\overline{u}$ is the mean of the ratings of all item rated by user u.

As discussed in the previous section, trust is often used as a replacement for similarity to improve the prediction accuracy. In the absence of social links, explicit rating data are used to infer implicit trust.

A common method, proposed by O'Donovan [8], to calculate implicit trust uses a target user's contribution to make successful recommendations to other users. This is measured as follows. The user u's recommendation of i for user v is considered *correct* if predicted rating $\hat{u}(i)$ is within α of the actual rating $v(i)$:

$$\text{correct}(i,\ u,\ v) \iff \left|\hat{u}(i) - v(i)\right| < \alpha. \tag{3}$$

Then, the set of a correct recommendation is defined as

$$\text{CorrectSet}(u) = \{(c_m, i_m) \in \text{RecSet}(u) : \text{correct}(c_m, u, i_m)\}, \tag{4}$$

where $\text{RecSet}(u) = \{(c_1,\ i_1),\ \ldots,\ (c_n,\ i_n)\}$ contains all the recommended n items to n users by the user u. The initial predictions are made using the Resnik formula:

$$\hat{u}(i) = \overline{u} + \frac{\sum_{v \in N_i^k(u)} (v(i) - \overline{v}) \times \text{similarity}(u,\ v)}{\sum_{v \in N_i^k(u)} \text{similarity}(u,\ v)}. \tag{5}$$

The final trust score is derived from the set of recommendations and the set of correct recommendations:

$$\text{Trust}(u) = \frac{\left|\text{CorrectSet}(u)\right|}{\left|\text{RecSet}(u)\right|}. \tag{6}$$

Then, the trust is replaced by the similarity in Equation (5) to make the final predictions:

$$\hat{u}(i) = \overline{u} + \frac{\sum_{v \in N_i^k(u)} (v(i) - \overline{v}) \times \text{Trust}(u, v)}{\sum_{v \in N_i^k(u)} \text{Trust}(u, v)}. \tag{7}$$

In the next sub-sections, we discuss the problems associated with this approach.

3.1.1. Limited Social Aspect

The trust formulation in Equation (5) can be seen as the usefulness of a user to another on receiving recommendations. However, it does not consider the agreement between both users. In the real world, the concept of trust relies on the degree of agreements between users. This contributes to the explainability of the recommendation system, as the process of recommending is much easier to explain to the users of the system. In this context, Equation (5) (trust) is not much different from Equation (2) (similarity), because it adds little information about the social aspect.

3.1.2. Sparsity of the Weight Matrix

In most recommendation systems, it is common to find more items than users, and users often rate a few items. Often, this leads some user pairs having no common ratings. If no items are co-rated items, we cannot calculate a similarity value. For users who have no common rating with any other user, Equation (5) would assign them mean ratings as a prediction. As all initial predictions are the same for all of these users, trust among any of them would be the same. Often, the trust value between them is 0, as α in the equation becomes very small. Even if the user has a common rating with some users, all the other users are ignored during the prediction, as similarity cannot be calculated for them using Equation (2). Therefore, the trust between the target user and all the ignored users would be the same; often, it is 0. This, in turn, would decrease the usefulness of the trust matrix in the final prediction calculation in Equation (6). If the trust value is 0, Equation (6) simply results in a mean rating. Thus, the computation of trust becomes a waste.

3.1.3. High Computational Cost

Making recommendations using the above trust method is a two-step process. First, a trust matrix is generated. Calculating the trust score for any specific pair of users involves generating predictions RecSet, solely using the pair's rating vectors. This step is further divided into similarity calculation and making initial predictions. This is achieved using Equations (1) and (2). Second, the final predictions are provided by replacing the similarity matrix with a trust matrix, as in Equation (6). The trust matrix generation process significantly increases the method's computation cost. The problem worsens as the number of users or items increases. As users rate very few items among the total number of available items, the similarity matrix is sparse in user-based CF. To increase the accuracy of the predictions, it is common to use item-based recommendations. Because of the large number of items used in the calculation, the inefficiency of this method is more apparent in the item-based recommendation.

In the next section, we detail our proposed model, AgreeRelTrust, in which we use the degree of agreement between the users and their relative activity to generate implicit trust. We assume that using trust in the prediction calculation will result in better accuracy.

3.2. AgreeRelTrust

The trust AgreeRelTrust between user u and user v is calculated by combining both users' agreements $A(u,v) \in A_{|U|\times|U|}$ and relative activities $RelA(u,v) \in RelA_{|U|\times|U|}$. The agreement $A(u,v)$ is

calculated as the ratio of agreements, that is, the sum of positive agreements and negative agreements in co-rated items:

$$A(u,v) = \frac{positiveAgreement(u,v) + negtiveAgreement(u,v)}{|R_u \cap R_v|}, \tag{8}$$

$$positiveAgreement(u,v) = \left|r : \ R_{(u,r)\in \mathrm{R}_v} \cap R_{(v,r)\in \mathrm{R}_u} \cap R_{(u,r)} \geq \beta \ \cap R_{(v,r)} \geq \beta\right|, \tag{9}$$

$$negtiveAgreement(u,v) = \left|r : \ R_{(u,r)\in \mathrm{R}_v} \cap R_{(v,r)\in \mathrm{R}_u} \cap R_{(u,r)} < \beta \cap R_{(v,r)} < \beta\right|, \tag{10}$$

where β separates positive and negative ratings; and R_u and R_v are the rating vectors of user u and v, respectively. The agreement value is in the range of [0,1], where 0 means no agreement and 1 is complete agreement. The positive agreements in Equation (9) are the number of items both users have liked; similarly, the negative agreements in Equation (10) contain items both users have disliked. The positive agreements do not overlap with negative agreements. As commonly rated items are always less than or equal to agreed items, the numerator is always less than or equal to the denominator. Thus, the agreement value is guaranteed to always be between 0 and 1. If both users do not have any common agreement, the agreement value would be 0.

For example, consider the case in which user *a* rated three items {2,1,3} and user *b* rated {3,1,2} for the same items. If the rating scale is from 0 to 5 with full stars, then 2.5 separates positives from negatives. In this case, user *a* has more disagreements than agreements with user *b*. However, in similarity-based systems, this would simply be taken as a positive correlation of 0.5 and is likely to be used in the prediction calculations for large values of *k*. However, using Equation (8), this would result in a much lower agreement of 0.33.

The assumption is that people are more likely to trust recommendations from people who have agreed with them in the past than from those who have disagreed with them. The agreement value is calculated by dividing the agreement by the total number of common ratings to normalize the value. We argue that the inclusion of agreement, as in Equation (8), would result in a much closer representation of the real world. Thus, this would yield better prediction accuracy.

In contrast to agreements, trust is asymmetric in nature. We incorporate this asymmetry in our model by combining the agreement with the relative activity (*RelA*) of users. This measure is also useful for addressing the issue of users with no common ratings. The relative activity of user u with respect to v is calculated as follows:

$$RelA(u,v) = \begin{cases} \frac{1}{1+e^{-ac}} & \text{if } |R_u| + |R_v| \ > 0 \ AND \ v \neq u \\ 0 \quad \text{else} & \end{cases}, \tag{11}$$

where $ac = \frac{|R_u|}{|R_u|+|R_v|}$.

Here, $|R_u|$ and $|R_v|$ are the lengths of the rating vectors of user u and user v, respectively. If none of the users have rated an item, then *ac* is set to 0. The value of *RelA* ranges from 0.5 to 0.731. The relative activity matrix *RelA* has the same dimension as *A*. The diagonal values are set to 0, so that these values do not increase in the final trust matrix. The final trust matrix is calculated as follows:

$$AgreeRelTrust \ = \ A^{\lambda} + \varepsilon \ RelA, \tag{12}$$

where λ and ε are hyper parameters that control how much of the agreement and the relative activity are to be included in the final trust value. Higher values of ε will increase the values of AgreeRelTrust matrix and, conversely, lower values will decrease the values of AgreeRelTrust. Therefore, it is important to choose an ε value that does not shift AgreeRelTrust values dramatically beyond positive 1 or negative 1. Similarly, higher λ values will shrink the contribution from the agreement, and thus overall AgreeRelTrust values. Equation (12) decreases the sparseness of the final trust matrix, as the relative activity of any two users would be greater than 0, as long as one of the users has rated an item,

although they may not have any common ratings. However, if both users in the pair have not yet rated any items, their trust value would still be 0. We think this is a much closer representation of real-world trust. If no information is available about the transacting partners, it is reasonable to assume that they have 0 trust among them. In this formulation, even if a user has provided the same rating for all items, the AgreeRelTrust values between the user and others would be different.

As agreement matrix A and relative activity matrix *RelA* have same the dimension, the indices of each paired value (item–item or user–user) in both matrices are the same. Therefore, as seen on lines 16 and 17 of Algorithm 1, the final combination of A and *RelA* is completed inside a single loop. The algorithm does not require any inputs other than the rating matrix and it is not involved in additional calculation, such as making initial predictions or the calculation of a similarity matrix.

Algorithm 1. Pseudocode for calculating the AgreeRelTrust algorithm.

```
1.  procedure GENERATETRUST (RatingMatrix, λ, ε, β)
2.      A = 0, RelA = 0
3.      n_users = length(RatingMatrix.row)
4.      for i = 0, i++, i < n_users do
5.          u = RatingMatrix.row[i]
6.          for j = i, j++, j < n_users do
7.              v = RatingMatrix.row[j]
8.              ac ← length(u)/(length(u) + length(v))
9.              RelA_{u,v} ← 1/1 + e^{−ac}
10.             RelA_{v,u} ← 1/1 + e^{ac−1}
11.                 if (u ≠ v) then
12.                     C_r ← getCommonItems(u, v)
13.                     Agr_pos ← length(C_r ≥ β)
14.                     Agr_neg ← length(C_r < β)
15.                     A_{u,v} ← (Agr_pos + Agr_neg) /length(C_r)
16.                     AgreeRelTrust (u, v) ← A^λ_{u,v} + ε × RelA_{u,v}
17.                     AgreeRelTrust (v, u) ← A^λ_{u,v} + ε × RelA_{v,u}
18.                 end if
19.             end for
20.         end for
21.         return AgreeRelTrust
22.  end procedure
```

One property of the *RelA* matrix is that complement values of *ac* in Equation (11) are symmetric on the diagonal. The entries in the lower triangular portion have the complement of those in the upper triangular portion. Thus, we improve the efficiency of the algorithm by only looping through the upper triangular portion of the matrix while calculating and assigning the values to the lower triangle of the matrix, as seen on lines 8 and 10 of the pseudocode in Algorithm 1. This would halve the time required to calculate the trust matrix.

Once the trust matrix is generated, we then make the predictions:

$$\hat{u}(i) = \overline{u} + \frac{\sum_{v \in N_i^k(u)} (v(i) - \overline{v}) \times \text{AgreeRelTrust}(u,v)}{\sum_{v \in N_i^k(u)} \text{AgreeRelTrust}(u,v)}, \tag{13}$$

where $\hat{u}(i)$ is the predicted rating. In Equation (13), k trustworthy friends are chosen to make the prediction.

4. Experiment

4.1. Data Sets

To evaluate our method, we used three publicly available datasets: (1) ML- 20m (GroupLens, University of Minnesota, Minniapolis, MN, USA), a 100,000 subset of MovieLens "ML20M, Recommended for New Research" dataset [33]; (2) (University of California, Barkeley, CA, USA), a 100,000 subset of Jester Dataset 2+ [34]; and (3) ML-100k (GroupLens, University of Minnesota, Minniapolis, MN, USA), MovieLens "ml-latest-small" full dataset [33]. The details are provided in Table 1.

Table 1. Details of the datasets.

Dataset	No. Ratings	No. Users	No. Items	Rating Scale
ML-20m	100,000	696	9000	1 to 5
ML-100k	100,836	610	9742	0.5 to 5
Jester	100,000	3359	128	−10 to 10

We used a slightly modified version of the Surprise, a well-known python framework for recommendations by Hug et al. [35], to implement our algorithm. For the link to code implementation, see the Supplementary Materials. Because of the high memory requirement of memory-based CF methods, the surprise framework cannot process the full datasets. Therefore, we used the abovementioned subsets (ML-20m and Jester). We used an Intel Core i5 2.7 GHz (Intel Corporation, Santa Clara, CA, USA) with an 8 GB RAM MacBook Pro (Apple Inc., Cupertino, CA, USA) to run our experiment.

We conducted experiments for different values of α, k, and ε and used Grid Search to select the best values for both datasets. β values depend on the maximum of the rating scale of the dataset (i.e., β is the midpoint of the rating scale). Table 2 summarizes the parameter values used for the experimental results presented in this paper.

Table 2. Parameter settings used for evaluation. CF—collaborative filtering.

Dataset	CF-Method-Type	k	ε	λ	β	α
ML-20m	user-based	40	−1	0.5	2.5	0.2
ML-20m	item-based	40	−0.6	0.5	2.5	0.2
ML-100k	user-based	40	−1	0.5	2.5	0.2
ML-100k	item-based	40	−1	0.5	2.5	0.2
Jester	user-based	40	−0.5	1	0	0.2
Jester	item-based	40	−0.6	0.5	0	0.9

4.2. Evaluation Metrics

The most commonly used accuracy metrics in CF are the mean absolute error (MAE) and the root mean square error (RMSE). Therefore, we adopted these two matrices to measure the prediction accuracy of our model. MAE treats all errors equally, whereas RMSE punishes higher deviation from the ground truth more. The RMSE is calculated as

$$\text{RMSE} = \sqrt{\frac{\sum_{\text{i}} (\hat{u}(i) - u(i))^2}{\left|\hat{R}\right|}}. \tag{14}$$

The mean absolute error (MAE) is calculated as

$$\text{MAE} = \frac{\sum_{\text{i}} \left|\hat{u}(i) - u(i)\right|}{\left|\hat{R}\right|}, \tag{15}$$

where $\hat{u}(i)$ is the predicted rating, $u(i)$ is the actual rating of the ith item, and $\hat{R}$ is the predicted rating matrix. MAE and RMSE are equal if there is no variance in errors.

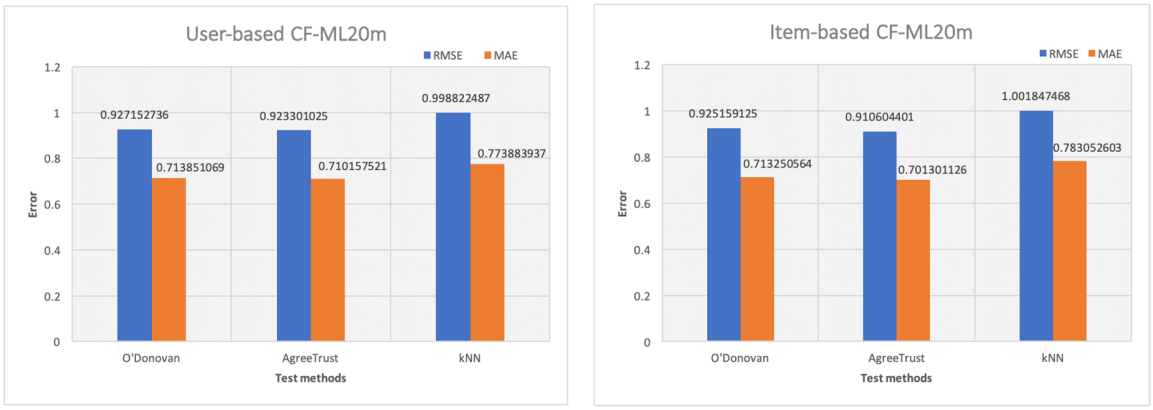
5. Results and Discussion

5.1. Prediction Accuracy

In this section, we compare the test results of our method (AgreeRelTrust) against the basic kNN (denoted kNN) and the baseline—a trust-based method discussed in Section 3.1. (denoted O'Donovan). For each dataset, we performed five-fold cross-validation on each method being tested, where one iteration was used as a test set and the remaining four iterations as training. The mean five-fold prediction accuracy for both metrics (RMSE and MAE) was recorded for all the methods.

5.1.1. ML-200m Dataset

Figure 1a shows the mean prediction accuracy results obtained for the ML-20M dataset under user-based CF for the three test methods. Figure 1b shows the results of item-based CF obtained from the same dataset.



(**a**) (**b**)

Figure 1. Result of three collaborative filtering (CF) methods for the ML-20m dataset: (**a**) user-based and (**b**) item-based. RMSE—root mean square error; MAE—mean absolute error; kNN—k-nearest neighbor.

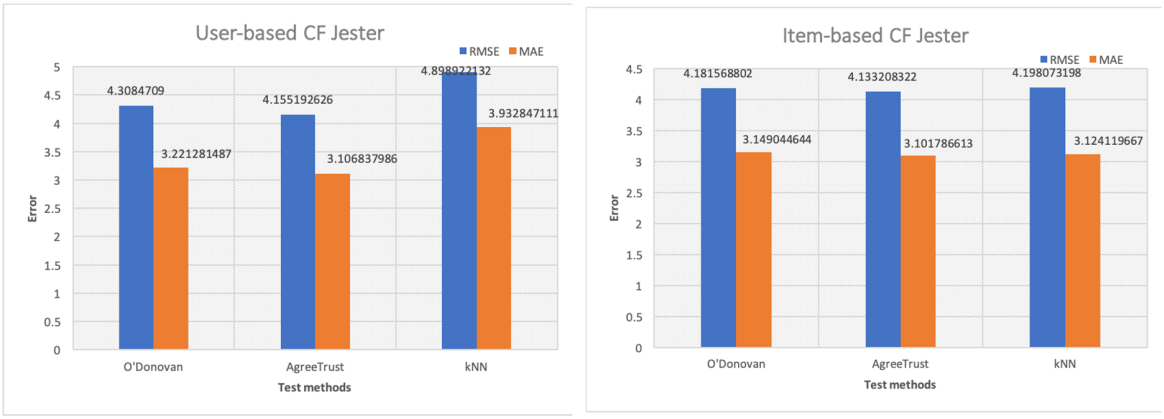
Regarding the prediction accuracy, for user-based CF, AgreeRelTrust outperformed all other methods. It achieved a marginal improvement of 0.42% in RMSE against the baseline (O'Donovan) and significantly against kNN by 7.56%. For the MAE, the respective improvements were 0.52% and 8.2%. Similarly, O'Donovan displayed improved accuracy of RMSE by 7.2% and MAE by 7.76%.

With respect to the accuracy for item-based CF, AgreeRelTrust ranked first. It exhibited improvements in RMSE ranging from 9.11% (against kNN) to 1.6% (against O'Donovan), and the corresponding improvements in MAE are 10.4% (against kNN) and 1.7% (against O'Donovan). O'Donovan, which ranked second, outperformed kNN in RMSE by 7.7% and in MAE by 8.9%.

Both trust-based methods, AgreeRelTrust and O'Donovan, have better accuracy for item-based CF than for user-based CF. However, for kNN, the opposite is true. For both item-based CF and user-based CF, trust-based methods have better prediction accuracies.

5.1.2. Jester Dataset

Figure 2 illustrates the mean accuracy results obtained from the Jester dataset for user-based CF for the three methods being studied. Figure 2b shows the same for item-based CF.



(**a**) (**b**)

Figure 2. Result of the three CF methods for the Jester dataset: (**a**) user-based and (**b**) item-based.

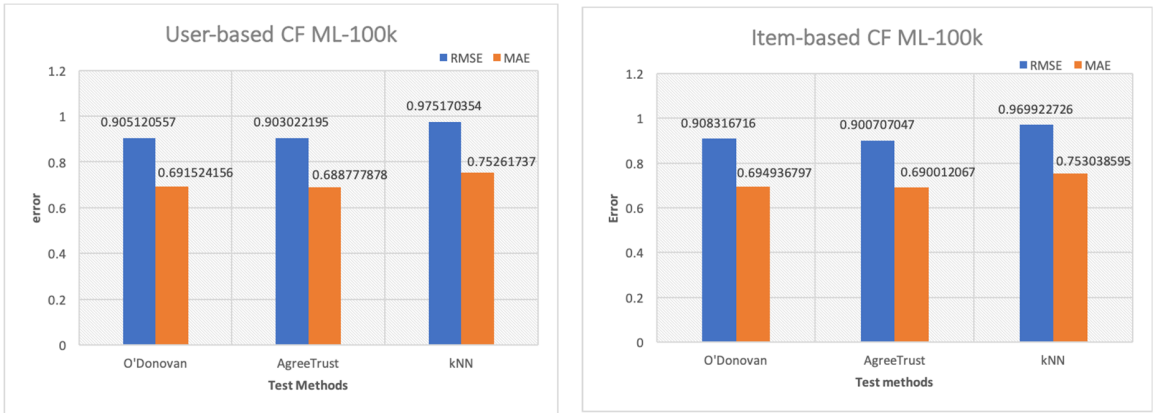
Similar to the results for the ML-20m dataset, as seen in Figure 2, we observed that the accuracy of AgreeRelTrust was better than that of kNN and O'Donovan for user-based CF. The overall accuracy of the test for the Jester dataset was worse than that of the ML-20m dataset. In general, the RMSE values were closer to four and MAE values were around three. In contrast to the ML-20m dataset, the Jester dataset contains many more users than items (only 128 items).

Regarding the accuracy for user-based CF, AgreeRelTrust produced a decrease of 3.56% in RMSE compared with that of O'Donovan. For the MAE metric, the decrease was marginal, at 1.56%. kNN performed the worst among all methods tested. AgreeRelTrust outperformed kNN with a significant improvement in RMSE by 15.2% and in MAE by 21%. O'Donovan, which ranked second, achieved improvements in RMSE by 12.1% and in MAE by 18.1% against kNN.

The performance for item-based CF of AgreeRelTrust against O'Donovan is similar to that of user-based CF. AgreeRelTrust achieved a decrease of 3.55% in RMSE, with a decrease of 1.55% in MAE. Similarly, AgreeRelTrust's improvements against kNN in RMSE and MAE were 1.55% and 0.71%, respectively. O'Donovan produced a slight improvement for item-based CF against kNN in RMSE of 0.39%; however, kNN was better in MAE by 0.79%.

5.1.3. ML-100k Dataset

Figure 3 illustrates the mean accuracy results obtained for the ML-100k dataset with user-based CF for the three methods being studied. Figure 3b shows the same for item-based CF.



(**a**) (**b**)

Figure 3. Results of the three CF methods for the ML-100k dataset: (**a**) user-based and (**b**) item-based.

The results of the 100k dataset, as seen in Figure 3, are similar to those of the other two datasets. Regarding the accuracy for user-based CF, AgreeRelTrust ranked first, with a slight increase of 0.23% in RMSE and of 0.4% in MAE against the O'Donovan method. The improvements against kNN were significant—7.4% for RMSE and 8.5% for MAE. O'Donovan beat kNN with improvements of 7.2% in RMS and 8.1% in MAE.

Regarding accuracy for item-based CF, AgreeRelTrust had an 0.84% lead in RMSE and an 0.71% lead in MAE against O'Donovan. It achieved significant improvements against kNN under item-based CF as well, at 7.14% for RMSE and 8.37% for MAE. O'Donovan's method, which ranked second, improved RMSE by 6.35% and MAE by 7.72% against kNN.

With respect to the overall prediction accuracy for both item-based and user-based CF, AgreeRelTrust ranked first and the baseline was the runner-up. The CF results for these three datasets show that the use of trust improves prediction accuracy. These results are consistent with results from previous studies [8].

5.2. Sparsity of Weight Matrix

To compare the sparsity of the weight matrices in all methods, in Figure 4, we graphed the number of empty cells in the weight matrix samples generated for all three methods during user-based CF for the ML-20m dataset against the number of users. For kNN, the weight matrix is the similarity matrix. AgreeRelTrust and O'Donovan use trust matrices for weighting. In the O'Donovan method, smaller values of α are supposed to increase the trust between users (decreasing the prediction error in initial predictions); therefore, we kept $\alpha = 0.01$. For high values of α, comparing the sparsity of the trust matrix is illogical, as this would include filling all the cells of the trust matrix by taking less trustworthy users. It was not necessary to change any parameter for AgreeRelTrust and kNN, as we were only comparing the sparsity of the trust matrix against other weight matrices.

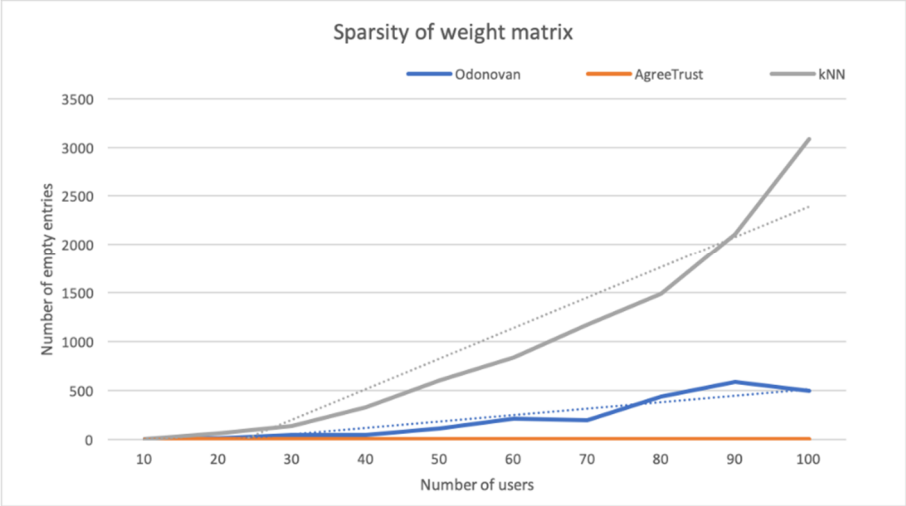


Figure 4. Sparsity of the weight matrices versus number of users.

As seen in Figure 4, an increase in the number of users increases the sparsity of the weight matrices. In the case of kNN, the average rate of increase in the sparseness of the similarity matrix is roughly five times that of the O'Donovan trust matrix. Similarly, the O'Donovan trust matrix has an average increase of sparseness five times that of AgreeRelTrust. AgreeRelTrust shows a flat zero, meaning one of the users in all the user-pairs in all the trust matrices has rated at least one item. Although a graph for item-based CF is not presented here, the trend would be similar. As discussed earlier, decreasing the sparsity is important as it impacts the accuracy of the predictions. In order to check the effect of the relative activity on the final AgreeRelTrust matrix, in Figure 5, we graphed the sparsity of the AgreeRelTrust matrix before and after adding the relative activity.

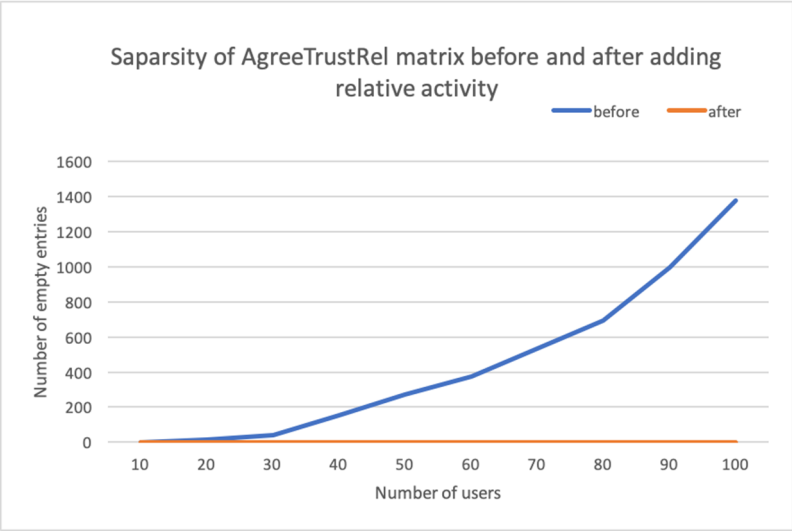


Figure 5. Sparsity of the AgreeRelTrust matrix before and after adding relative activity.

As seen in Figure 5, it is obvious that adding relative activity to the agreements decreases the sparsity of the final AgreeRelTrust matrix to 100%. This is expected because all users in ML-20m have rated at least 20 items. The decrease in sparsity of the final AgreeRelTrust matrix is important to increase the prediction accuracy as it would increase the number of non-zero trusted neighbors in Equation (13).

5.3. Complexity Analysis

The rating prediction calculation phases of the three methods discussed in this work are similar—they all have the same time complexity of $O(k)$, where k is the number of neighbors. The main difference between kNN and the other two methods is the replacement of the similarity matrix with a trust matrix. Therefore, we only focus on the complexity of trust and the similarity calculation.

The kNN method uses Pearson correlation, as in Equation (2), to calculate the similarity. The complexity of the similarity between a single pair of users is $O(2 \times r)$, where r is the length of the common ratings. For N pairs, the computational complexity is $O\left(N^2 \times 2 \times \bar{r}\right)$, where $\bar{r}$ is the average length of rating vectors. As the similarity matrix is symmetric, complexity can be reduced to half; thus, $O\left(N^2 \times \bar{r}\right)$.

In O'Donovan's method, trust matrix generation involves the initial prediction calculation for all the items for each user pair. This, in turn, requires the calculation of the similarities for the user-pairs; this task has a complexity of $O(2 \times r)$, and in general, $O(2 \times \bar{r})$. The prediction step involves calculating the target user average and weighing the common ratings. The average calculation is always dominated by weighting. Therefore, the overall complexity of this step is $O\left(\overline{r^2}\right)$. To calculate trust, RecSet is searched for correct items; thus, the complexity is $O(l)$, where l is the length of RecSet. The whole process is repeated for all the user pairs (N pairs). Thus, the overall complexity is $O\left(N^2 \times 2 \times \overline{r^3} \times l\right)$. For simplicity, we implemented O'Donovan's method using a leave-one-out strategy.

The first step in the AgreeRelTrust calculation involves obtaining the length of two rating vectors to calculate the relative activity. For a single user pair, the complexity of this step is $O(2 \times r)$, and in general, $O(2 \times \bar{r})$. The agreement calculation loops through both rating vectors to find positive and negative agreements. This is completed by looping through the common ratings of both users in the user pair; thus, $O(2 \times \bar{r})$. The process is repeated for all the N user-pairs. The total complexity is $O\left(2 \times N^2 \times \overline{r^2}\right) + \left(2 \times N^2 \times \overline{r^2}\right)$. Because of some symmetric properties of the agreement matrix and relative activity matrix, the complexity of the algorithm can be reduced to half. Thus, the overall complexity is $O\left(2 \times N^2 \times \overline{r^2}\right)$.

To analyze the time complexity of the three methods being studied, we ran user-based CF and recorded the time taken to compute weight matrices against the number of users. We used the ML-100k dataset for this test. As seen in Figure 6, kNN was the fastest method, which requires less than a millisecond for less than three users. kNN was 10 times faster than AgreeRelTrust and 1000 times faster than O'Donovan. AgreeRelTrust ranked second in the performance graph, being 100 times faster than O'Donovan. Notably, in this experiment, we did not perform full optimization on AgreeRelTrust. The performance can further be improved by parallelization.

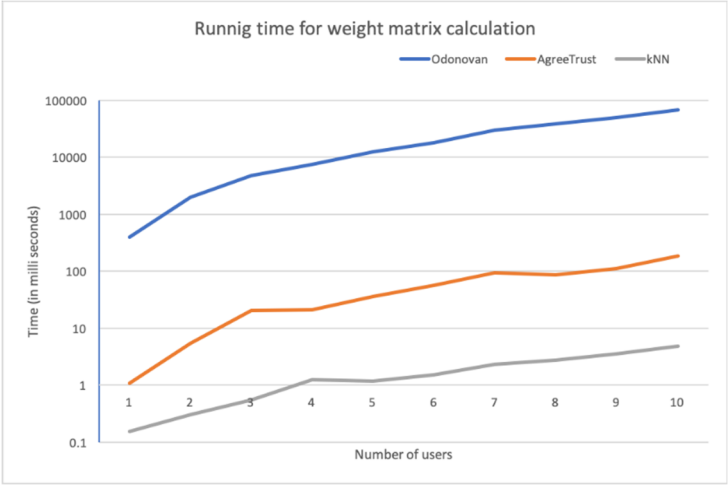


Figure 6. Running time to calculate weight matrices.

In real-world implementations, it is common to pre-calculate the similarity matrix offline. The same can be applied to the trust matrix calculation, which then can be used to make real-time predictions.

5.4. Issues and Limitations

In contrast with the similarity, the trust values in our method are not bounded between –1 and 1. This is because of the addition of the relative activity into the agreements. If two users completely agree, then their agreement would be 1. Adding any positive relative activity would result in a trust value greater than 1. In the case in which two users have no agreement (i.e., agreement is 0), and for values of ε less than –1, adding relative activity may result in trust values less than –1. One way to address this issue is to cap the trust values between 0 and 1 by replacing all the values greater than 1 with 1 and replacing the values less than 0 with 0. However, in our experiment, we minimized these occurrences by carefully adjusting the value of ε.

In this work, our model was designed to infer implicit trust from the explicit ratings. In many online recommendation systems, a user's explicit ratings are often not available. Instead, implicit data, such as clicks, are used to make recommendations. As is, AgreeRelTrust would fail for implicit binary data, as there is no accurate method of measuring negative agreements. We leave this problem for future work.

6. Conclusions

In this paper, we proposed AgreeRelTrust, an implicit trust inference model for CF recommendation systems. The basic idea is that the trust relationship between two users relies on their agreements and their relative activity. The agreement is the ratio of the number of agreed ratings (considering both positive and negatives) and the total number of co-rated items. Relative activity depends on a quantity that is based on the ratio of the difference in the number of ratings of two users over their total number of ratings. We have shown that, by experimental evaluations on three real benchmark datasets, our method improves prediction accuracy while reducing time complexity compared with other available

models. In contrast to most implicit trust inference methods that use rating as a primary source of trust, our model considers the direction of rating agreement between users.

The proposed model has a limitation of working only on explicit feedback. We plan to expand our model to consider implicit feedback. Modern recommendation systems rely heavily on deep learning models. However, not much work has been undertaken to incorporate trust into deep learning models. Therefore, we also plan to investigate the effect of incorporating trust into deep learning models.

Supplementary Materials: For the purpose of reproducibility, we published the code for our experiment at http://github.com/xahiru/agreerecom.

Author Contributions: The conceptualization of the original idea, formal analysis, and the performance of the experiment, as well as the original draft preparation, were completed by A.Z. Review and editing were completed by K.M. Y.Y. supervised the project.

Funding: This research was funded by National Natural Science Foundation of China, grant number 91118002.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ebay Inc. eBay. 2019. Available online: www.ebay.com (accessed on 29 March 2019).
2. Google LLC. YouTube. 2019. Available online: www.youtube.com (accessed on 29 March 2019).
3. Huang, Z.; Chen, H.; Zeng, D. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.* **2004**, *22*, 116–142. [CrossRef]
4. Javari, A.; Jalili, M. Cluster-based collaborative filtering for sign prediction in social networks with positive and negative links. *ACM Trans. Intell. Syst. Technol.* **2014**, *5*, 24. [CrossRef]
5. Manuja, K.; Bhattacharya, A. Using social connections to improve collaborative filtering. In Proceedings of the Second ACM IKDD Conference on Data Sciences, Bangalore, India, 18–21 March 2015; pp. 140–141.
6. Brandão, M.A.; Moro, M.M.; Lopes, G.R.; Oliveira, J.P.M. Using link semantics to recommend collaborations in academic social networks. In Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 833–840.
7. Ma, X.; Lu, H.; Gan, Z. Improving recommendation accuracy by combining trust communities and collaborative filtering. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, Shanghai, China, 3–7 November 2014; pp. 1951–1954.
8. O'Donovan, J.; Smyth, B. Trust in Recommender Systems. In Proceedings of the 10th international conference on Intelligent user interfaces, San Diego, CA, USA, 10–13 January 2005; pp. 167–174.
9. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
10. Singhal, A.; Sinha, P.; Pant, R. Use of deep learning in modern recommendation system: A summary of recent works. *arXiv*, 2017; arXiv:1712.07525.
11. Guo, G.; Zhang, J.; Sun, Z.; Yorke-Smith, N. LibRec: A Java Library for Recommender Systems. In Proceedings of the 23rd International Conference on User Modeling Adaptation and Personalization, Dublin, Ireland, 29 June 29–3 July 2015; Volume 4.
12. Ekstrand, M.D.; Ludwig, M.; Konstan, J.A.; Riedl, J.T. Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. In Proceedings of the Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; pp. 133–140.
13. Kim, H.-N.; Ji, A.-T.; Jo, G.-S. Enhanced Prediction Algorithm for Item-Based Collaborative Filtering Recommendation. In Proceedings of the E-Commerce and Web Technologies, Krakow, Poland, 5–7 September 2006; pp. 41–50.
14. Ricci, F.; Rokach, L.; Shapira, B. Recommender systems: introduction and challenges. In *Recommender Systems Handbook*; Springer: Berlin, Germany, 2015; pp. 1–34.
15. Guo, G.; Zhang, J.; Yorke-Smith, N. A Novel Bayesian Similarity Measure for Recommender Systems. In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence (IJCAI), Beijing, China, 3–9 August 2013; pp. 2619–2625.

16. Schein, A.I.; Popescul, A.; Ungar, L.H.; Pennock, D.M. Methods and metrics for cold-start recommendations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, 11–15 August 2002; pp. 253–260.
17. Jamali, M.; Ester, M. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 397–406.
18. Amazon.com Inc. Amazon. 2019. Available online: www.amazon.com (accessed on 29 March 2019).
19. Carbone, M.; Nielsen, M.; Sassone, V. A formal model for trust in dynamic networks. In Proceedings of the First International Conference on Software Engineering and Formal Methods, Brisbane, QLD, Australia, 22–27 September 2003; pp. 54–61.
20. Fullam, K.K.; Klos, T.B.; Muller, G.; Sabater, J.; Schlosser, A.; Topol, Z.; Barber, K.S.; Rosenschein, J.S.; Vercouter, L.; Voss, M. A Specification of the Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies. In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, Utrecht, The Netherlands, 25–29 July 2004; pp. 512–518.
21. Jøsang, A.; Ismail, R.; Boyd, C. A survey of trust and reputation systems for online service provision. *Decis. Syst.* **2007**, *43*, 618–644. [CrossRef]
22. Guo, G.; Zhang, J.; Thalmann, D. A simple but effective method to incorporate trusted neighbors in recommender systems. In Proceedings of the International Conference on User Modeling, Adaptation, and Personalization, Montreal, QC, Canada, 16–20 July 2012; pp. 114–125.
23. Pu, P.; Chen, L. Trust building with explanation interfaces. In Proceedings of the 11th International Conference on Intelligent User Interfaces, Sydney, Australia, 29 January–1 February 2006; pp. 93–100.
24. Korolova, A.; Motwani, R.; Nabar, S.U.; Xu, Y. Link privacy in social networks. In Proceedings of the 17th ACM Conference on Information and Knowledge Management, Valley, CA, USA, 26–30 October 2008; pp. 289–298.
25. Massa, P.; Avesani, P. Trust metrics in recommender systems. In *Computing with Social Trust*; Springer: Berlin, Germany, 2009; pp. 259–285.
26. Golbeck, J.; Hendler, J. Filmtrust: Movie recommendations using trust in web-based social networks. In Proceedings of the IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, 8–10 January 2006; Volume 96, pp. 282–286.
27. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 135–142.
28. Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of the 1994 ACM conference on Computer supported cooperative work, Chapel Hill, NC, USA, 22–26 October 1994; pp. 175–186.
29. Shambour, Q.; Lu, J. An effective recommender system by unifying user and item trust information for B2B applications. *J. Comput. Syst. Sci.* **2015**, *81*, 1110–1126. [CrossRef]
30. Lathia, N.; Hailes, S.; Capra, L. Trust-based collaborative filtering. In Proceedings of the IFIP International Conference on Trust Management II, Trondheim, Norway, 18–20 June 2008; pp. 119–134.
31. Pitsilis, G.; Marshall, L.F. *A Model of Trust Derivation from Evidence for Use in Recommendation Systems*; University of Newcastle upon Tyne, Computing Science: Newcastle upon Tyne, UK, 2004.
32. Li, Y.-M.; Wu, C.-T.; Lai, C.-Y. A social recommender mechanism for e-commerce: Combining similarity, trust, and relationship. *Decis. Support Syst.* **2013**, *55*, 740–752. [CrossRef]
33. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* **2015**, *5*, 19. [CrossRef]
34. Goldberg, K.; Roeder, T.; Gupta, D.; Perkins, C. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Inf. Retr.* **2001**, *4*, 133–151. [CrossRef]
35. Hug, N. Surprise, a Python Library for Recommender Systems. 2017. Available online: http://surpriselib.com (accessed on 29 March 2019).