Trust-based recommendation with privacy

Simon Meyffret — Lionel Médini — Frédérique Laforest

Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205, F-69621, France
Université Lyon 1, LIRIS, UMR5205, F-69622, France
simon.meyffret@liris.cnrs.fr

ABSTRACT. *Recommender Systems are widely used to achieve a constantly growing variety of services. Along with social networks recommendation systems have emerged that take into account friendship or trust between users. In this article, we propose several recommendation algorithms that respect data privacy constraints in order to be implemented on decentralized and dynamic architecture. They do not need any global knowledge on the network, they limit data exchange to trusted friendship relations and protect private data from being spread over the network.*
We present an evaluation of different recommendation algorithms, in order to compare ours to centralized ones, such as those used in online marketing services. Several concerns are considered: specific accuracy measures are defined, as well as the amount of required knowledge on the overall dataset. We ran two series of simulation tests that are herein presented and discussed.

RÉSUMÉ. *Les systèmes de recommandation sont de plus en plus largement utilisés. Avec les réseaux sociaux, ces systèmes tiennent désormais compte des relations d'amitié ou de confiance entre les utilisateurs. Dans cet article, nous proposons des algorithmes de recommandation respectant la confidentialité des données et pouvant être implantés sur des architectures décentralisées et dynamiques. Ces algorithmes n'ont besoin d'aucune connaissance globale du réseau, limitent l'échange de données aux relations d'amitié et de confiance entre les pairs et évitent la diffusion des données privées sur le réseau.*
Nous avons réalisé une évaluation de différents algorithmes de recommandation, afin de comparer notre proposition aux algorithmes centralisés. Pour cela, nous avons défini des mesures spécifiques de précision des résultats et identifié l'ensemble des connaissances nécessaires pour chaque algorithme. Nous avons mené deux séries de tests de simulation qui sont présentés et discutés.

KEYWORDS: *Trust-based recommender systems, decentralized algorithms, data privacy protection*

MOTS-CLÉS : *Systèmes de recommandation basés sur la confiance, algorithmes décentralisés, protection des données personnelles*

1. Introduction

Internet and pervasive environments tend to connect everyone to everything. Users can access a lot of media/documents/resources easily and rapidly, through various types of devices (computers, notebooks, smartphones...). There are too many items to make an informed choice on one's own. Information overload is an ever growing problem that Recommender Systems (RS) aim at coping with. RS ease item selection by providing users with limited lists of preselected items (online music websites, cameras...) which are supposed to fit their needs. Users select the most appropriate item in this limited list according to their needs and preferences. For this, RS are based on users and items profiles. Most commonly, RS use *ratings* to preselect items regarding a user by ranking these items and selecting the top$-k$ items. Usually, users can explicitly rate items to indicate if they like them or not, whereas some RS use traces (items reviews, items visualization duration...) in order to construct implicit ratings.

Whatever the way to construct ratings, it is quite impossible to get a full ratings matrix between all users and all items. RS need to predict ratings in order to complete the matrix. Classical RS are of two kinds [Balabanović et Shoham1997]. Content-based RS compare items to recommend items similar to the ones already rated by the user. Collaborative Filtering RS (CF) compare users in order to find those with similar profiles and use their ratings to recommend items. Both are based on many users and/or items descriptions that are often centralized. Trust RS use trust relations between users to predict items scores; trust propagation among users is used to widen the prediction capabilities. This technique also requires information on all users.

Privacy concerns require new techniques for RS that would drastically limit the disclosure of information on users and trust relations. In a more and more mobile and ad hoc network environment, the centralization of information is not conceivable any more. Our aim is to show that a well chosen limited vision of the ratings is still efficient in score propagation and that using the user social network is relevant for RS.

A social network (SN) consists of a finite set or sets of actors and the relation or relations defined between them. An actor is a social entity (discrete individual, corporate or collective social unit) [Wasserman et Faust1994]. In this paper, an actor refers to any social entity in the SN.

Our overall objective is to make a RS suitable for decentralized architectures with privacy consideration. Such architectures bring constraints that make classical RS unadapted. First, since the local engine located on a peer cannot access a centralized database containing all actors and items profiles, it has to cope with data spread among peers in a decentralized architecture. Second, RS should use only information that actors have accepted to share and respect their privacy. Trust relations are the basis of this mechanism: actors accept sharing information with the people they have explicitly defined as trustees. Finally, peers can connect or disconnect to and from the network instantly and unpredictably, without destabilizing the RS.

As we use the social network of actors to propagate items score, our approach computes social scoring. In our architecture, only trusted actors can communicate together. The trust value is explicitly set by actors in their social network. We use the term "friends" to refer to actors connected in the social network. Every actor keeps his/her own private profile. He/she can share it on demand if he trusts the requester. Possibly, actors can copy part of friends profile locally to limit network requests and anticipate peers disconnection. It is adapted to peer to peer architectures, since peers (actors) require only information from their trusted friends, scores are propagated from peers to peers in order to predict missing ratings.

We have implemented metrics to compare and evaluate different RS, as explained in the state of the art (section 2). We also propose a predictive accuracy metrics which increases the importance of unusual ratings (*i.e.* far away from the mean ratings of an item). Finally, we will discuss accuracy of this approach regarding existing RS.

2. State of the art

This work concerns three subjects: RS using collaborative filtering (CF); social network and trust and evaluation metrics. This state of the art summarizes main works of our knowledge in these three directions.

CF use users (actors) profiles in order to compute similarity between users and then aggregate similar users' ratings. Two classical aggregation functions are used [Adomavicius2005]:

$$r_{a,i} = \frac{\sum_{a' \in A_i} sim(a, a') \times r_{a',i}}{\sum_{a' \in A_i} sim(a, a')} \tag{1}$$

$$r_{a,i} = \overline{r_a} + \frac{\sum_{a' \in A_i} sim(a, a') \times (r_{a',i} - \overline{r_{a'}})}{\sum_{a' \in A_i} sim(a, a')} \tag{2}$$

Where $sim(a, a')$ is the similarity between a and a' and $r_{a,i}$ is the rating given by a to item i and A_i is the set of users having rated item i, sim is traditionally a Pearson coefficient correlation between users' ratings.

Most CF use all ratings in order to select similar users and then propagate missing scores between them. Although global CF are successfully used by most e-commerce websites, they require to know pieces of information on every user: purchase history, items views (which item, how many times, how long...), users information (gender, city, work)... Data is aggregated and centralized by the system, which knows everything. This approach is only suitable for a centralized architecture. This kind of approach is not adapted to the preservation of user privacy, and cannot work in a decentralized architecture or peer to peer network.

Thanks to their social network, people share data and opinions anytime with their family, friends or simple relations. Smartphones bring this connectivity to another level where they can share every single detail immediately on their network, without

intermediaries, from anywhere. The social relation between actors used here is trust. In the literature, one can find multiple definition of trust [McKnight et Chervany1996, Hasan2010]. In this paper, trust is defined as the belief of an actor in the usefulness of information provided by an other actor [Lee2009].

Trust-based RS use social networks of trust to improve the collaborative recommendations [O'Donovan et Smyth2005, Massa et Avesani2007, Ma *et al.*2009]. Moreover, CF methods suffer some drawback that can be addressed by trust-based RS. As resumed in [Lee2009], CF is not well-protected against malicious [Mehta *et al.*2007] or peculiar [Schafer *et al.*2007] users. CF compares all users together to compute similarity, which requires important off-line calculation. Traditional problems as the cold start user or sparsity can be resolved by trust-based RS: users do not need to have ratings to use the system, they only need to trust other users through specific or existing SN [Massa et Avesani2007, Pitsilis et Knapskog2009].
Different kinds of trust metrics are used in RS. Global trust metrics as PageRank ([Page *et al.*1999]) or eBay feedback are widely used, they concern a unique and global value of trust by user. Local metrics define a trust value from one user to another. Moreover, the trust can be subjective or objective (reputation). Subjective local metrics are defined by each user to indicate who is trustee or not, it is user-centric and user-controlled, which allows some privacy guaranties.
Traditionally, trust is propagated in the network in order to compute new links between users [Hang *et al.*2009, Massa et Avesani2007]. Existing trust RS propagate trust: the systems adds new trust relations in the network, based on some "transitivity" property of trust relations. We believe that trust must be controlled by the user and can only be explicitly set by users. Moreover, to compute those links, the system must know all trust values between users. SN clustering is also used to improve recommendation [DuBois *et al.*2009], but here again trust relations are created between users who are not directly related in the SN.

[Herlocker *et al.*2004] indicates several metrics to evaluate RS, including predictive accuracy metrics and rank accuracy metrics. Predictive accuracy metrics measure how close predicted ratings are to existing ratings. The main one is the Mean Absolute Error (Eq. 3). p_i is the predicted rating for item i, r_i is the existing rating for item i and N is the total number of items. The Root Mean Square Error (RMSE) is used to accentuate larger errors (Eq. 3). The lower are the metrics, the nearer are predicted ratings from existing ones, so the better is the RS.

$$MAE = \frac{\sum_{n=1}^{N} |p_n - r_n|}{N} \quad RMSE = \sqrt{\frac{\sum_{n=1}^{N} (p_n - r_n)^2}{N}} \tag{3}$$

Rank accuracy metrics compare predicted (p) item ranks with existing ones (r). The Pearson correlation measures the linear relation between two variables. Spearman's rank correlation measures the relation between items ranks. The same formula (Eq. 4) is applied in both cases but Pearson uses items ratings whereas Spearman uses

items ranks [Herlocker *et al.*1999]. The greater the metrics are, the more the ratings (or ranks) are correlated, so the better the RS is.

$$\rho = \frac{\sum_{n=1}^{N}(r_n - \bar{r})(p_n - \bar{p})}{N \times \sqrt{\sum_{n=1}^{N}(r_n - \bar{r})^2} \times \sqrt{\sum_{n=1}^{N}(p_n - \bar{p})^2}} \tag{4}$$

Coverage metrics are also good indicators of the overall quality of a RS. They indicate how many ratings a RS can predict, since having a recommendation is almost as important as having a good recommendation for the users. In this paper, we used three coverage metrics. *Count* is the count of scores actually predicted. *Actors count* is the count of actors having at least one score predicted. *Actors scores mean* is the mean of scores count by actor: $\frac{count}{actor\_count}$.

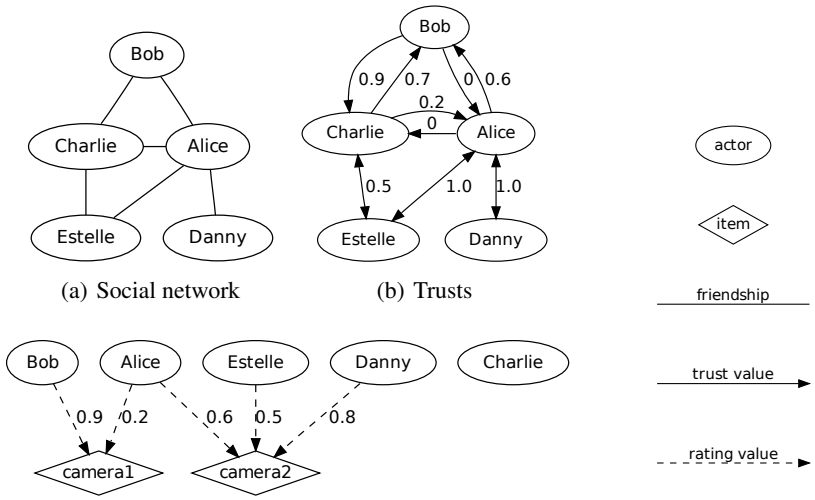
Ratings take their values in a limited set of values (*e.g.* $[\![1, 5]\!]$), so RS can be seen as classifiers where the classes are the possible ratings. Precision, recall, f-measure, true positive or false positive rate are some measures used in classification. However our classes are not independent: a predicted 2 value for a rating of 1 is better than a predicted 3.
In information retrieval, a document is relevant or not regarding a query. Precision and recall are frequently used. In recommendation, the result is not bound to a query, so it is not appropriate. However, [Basu *et al.*1998] took the first quartile of items (the best rated) and defined this set as the relevant set, the remainder being the irrelevant set. Then precision and recall are computed with predicted scores. But since ratings range is quite small (typically from 1 to 5), most ratings have the same value, and it becomes difficult to take the top$-k$ best items if the twenty first have the same rating.

3. Example and definitions

We illustrate this article with a simple, yet adequate, example. Alice, Bob, Charlie Danny and Estelle are five actors. The social network provides friendship relations between actors (figure 1(a)). The trusts values shared between friends vary between 0 and 1. They are given in figure 1(b). The trust relation is defined in section 3: it is a weighted and oriented relationship from an actor to one of his/her friends.

Figure 1(c) shows ratings by actors on items. Items are two cameras: $camera_1$ and $camera_2$. The rating values set by actors on items are represented as weights on edges. Actors do not rate items in the same way. Bob really likes $camera_1$ whereas Alice prefers $camera_2$. Danny and Estelle only rated $camera_2$ and Charlie did not rate any camera.

In our work, recommendation of cameras takes into account friends ratings and trust. To do so, our social RS propagates scores through the social network in order to predict Bob's score for $camera_2$, Danny and Estelle's scores for $camera_1$ and Charlie's score for the two cameras.



(a) Social network (b) Trusts

(c) Ratings

Figure 1. *Social network, trust and ratings example*

Let A be the set of actors and a denote an actor. Let I be the set of items and i denote an item. In our motivating example:

- $A = \{Alice, Bob, Charlie, Danny, Estelle\}$
- $I = \{camera_1, camera_2\}$

The trust relation from actor a to his/her friend f is noted $t_{a,f}$ with $t_{a,f} \in [0, 1]$. This relation is only defined between friends. Let T be the set of trust triples $(a, f, t_{a,f})$ with $(a, f) \in A^2$ and $t_{a,f} \in [0, 1]$. For each $(a, f) \in A^2$ can be associated zero or one trust $t_{a,f}$. The greater $t_{a,f}$ is, the more actor a trusts actor f's scoring, and then the more actor f's preferred items are valuable for a. This relation is not symmetric: $(a, f, t_{a,f}) \in T \not\Rightarrow (f, a, t_{f,a}) \in T$ and not transitive: $(a, b, t_{a,b}) \in T \wedge (b, c, t_{b,c}) \in T \not\Rightarrow (a, c, t_{a,c}) \in T$. Our social RS does not infer trust relations from existing ones. In Fig.1(b) Bob trusts Charlie's recommendations at 0.9 but Charlie trusts Bob only at 0.7. Friendship does not imply trust between actors: Alice and Charlie are friends but share low trust.

Let T_a be the set of a's trusted friends: $T_a = \{f \in A | \exists (a, f, t_{a,f}) \in T \wedge t_{a,f} > 0\}$. Example on our motivating scenario:

- $T = \{(Alice, Bob, 0.6), (Bob, Alice, 0), \cdots, (Estelle, Charlie, 0.5)\}$
- $T_{Alice} = \{Bob, Danny, Estelle\}$
- $T_{Estelle} = \{Alice, Charlie\}$

The ratings graph between actors and items (Fig. 1(c)) is a bipartite graph. Ratings are edges between actors (first partition) and items (second partition). Moreover, each

actor is the root of a one-depth tree which contains all items rated by this actor. The rating given by actor a to item i is written $r_{a,i}$. Let R be the set of ratings triples $(a, i, r_{a,i})$ with $a \in A$, $i \in I$ and $r_{a,i} \in [0, 1]$. 0 means that the actor does not like this item. 1 means he/she does. For each $(a, i) \in A \times I$ can be associated zero or one rating $r_{a,i}$.

Let $A_i = \{a \in A | \exists (a, i, r_{a,i}) \in R\}$ be the set of all actors who have rated item i. Example on our motivating scenario:

- $R = \{(Bob, camera_1, 0.9), (Alice, camera_1, 0.2), \cdots, (Danny, camera_2, 0.8)\}$
- $A_{camera_1} = \{Alice, Bob\}$
- $A_{camera_2} = \{Alice, Estelle, Danny\}$

4. Social scoring

In this work, we propose to use social scoring to predict a missing rating that could have been made by an actor on an item. We call such missing ratings scores. Different algorithms can be used to predict scores based on trust relationships. In this article, we define and present four different algorithms that are presented below.

4.1. *Immediate Social Scoring*

In order to compute scores for a specific actor, immediate social scoring uses the actor's friends ratings to evaluate the actor score on an item. No data is transmitted between actors if they are not friends. It is a way to ensure privacy for actors ratings.

The immediate social score of an item i by an actor a is the actor's rating if it is defined, otherwise it is a combination of a's friends ratings weighted by trust. Of course, if the actor has no friend having rated item i, then the score is not computed ($\perp$), whatever the social scoring algorithm used is.

$$
\begin{aligned}
score_1 : A \times I &\to [0, 1] \cup \{\perp\} \\
(a, i) &\mapsto score_1(a, i)
\end{aligned}
$$

$$
score_1(a, i) = \begin{cases} r_{a,i} & \text{if } \exists r_{a,i} \\ \dfrac{\sum_{f \in T_a \cap A_i} t_{a,f} \times r_{f,i}}{\sum_{f \in T_a \cap A_i} t_{a,f}} & \text{if } \nexists r_{a,i} \wedge T_a \cap A_i \neq \emptyset \\ \perp & \text{otherwise} \end{cases}
$$

4.2. $k-$*Depth Social Scoring*

This algorithm extends the previous formula to use $k-$depth friends of friends instead of immediate vicinity. This introduces a kind of transient, secured and anonymous transitivity of friendship at depth k. $score_k$ gets user's rating if exists. If not, it asks the user's trusted friends to provide their ratings (if any) or to predict their score, using their friends' ratings and so on, to depth k. To ease the next algorithms definitions, we introduce $\mathcal{A}^k_{a,l,i} = \{f \in T_a | f \neq l \wedge score_k(f,i,a) \neq \perp\}$ as the set of actors f different of l and trusted by a where $score_k(f,i,a)$ is defined. $\mathcal{A}^0_{a,l,i} = A_i \cap T_a, \quad \forall l \in A$.

$$score_k : A^2 \times I \to [0,1] \cup \{\perp\}$$
$$(a,l,i) \mapsto score_k(a,l,i)$$

$$score_k(a,l,i) = \begin{cases} r_{a,i} & \text{if } \exists r_{a,i} \\ \dfrac{\sum_{f \in \mathcal{A}^{k-1}_{a,l,i}} t_{a,f} \times score_{k-1}(f,i,a)}{\sum_{f \in \mathcal{A}^{k-1}_{a,l,i}} t_{a,f}} & \text{if } \nexists r_{a,i} \wedge \mathcal{A}^{k-1}_{a,l,i} \neq \emptyset \\ \perp & \text{otherwise} \end{cases}$$

To minimize cycles in the formula, we have introduced a "last requesting node" l: a receives a score request from his/her friend l and if $t_{a,l} > 0$, a will not ask back l's score for this item. Then, a becomes the last requesting node for his/her friends. a's friends will not know that l is the original requester. If f is a friend of a and l, he/she might ask l's score for the item. We have a limited cycle since k decreases at every hop and since due to small world effect [Milgram1967], k must be low (2 or 3 maximum). If one wants to use $k > 3$, then a node with a pending request should return $\perp$ if he/she receives the same request again. The cycle remains at a low level and the confidentiality increases since no information is shared between non trusted actors. By definition, we set $score_k(a,i) = score_k(a,a,i)$ for the first requester.

The $k-$depth social scoring shares data only between immediate friends. Scores are aggregated before being transmitted to the requester so the requester does not know from whom the replied score is computed. However full scores are exchanged between friends. It may still be too much information exchange regarding privacy policies, this is why we introduce the next social scoring.

4.3. *Relative Social Scoring*

The $k-$depth social scorer computes absolute scores, *i.e.* whatever the actor behaviour, it returns a score computed only using actor or friends ratings. The following definition computes a relative score *i.e.* the difference, for each friend, between the score and the friend's ratings mean. Friends relative scores are aggregated and added to the actor's ratings mean.

This approach has two main advantages. First, it does not transmit absolute scores, but only relative scores: this means less information and therefore more privacy. Second, it takes into account actors rating behaviour (low or high ratings in general). But this approach has a main drawback: if an actor has rated no item, no mean is available. In this case, 0.5 is used as a default mean.

$$\begin{aligned}
\delta score_k &: A^2 \times I \to [0,1] \cup \{\bot\} \\
&(a,l,i) \mapsto \delta score_k(a,l,i) \\
\delta score_k(a,l,i) &= \begin{cases} r_{a,i} - \overline{r_a} & \text{if } \exists r_{a,i} \\ \dfrac{\sum_{f \in \mathcal{A}_{a,l,i}^{k-1}} t_{a,f} \times \delta score_{k-1}(f,i,a)}{\sum_{f \in \mathcal{A}_{a,l,i}^{k-1}} t_{a,f}} & \text{if } \nexists r_{a,i} \wedge \mathcal{A}_{a,l,i}^{k-1} \neq \emptyset \\ \bot & \text{otherwise} \end{cases}
\end{aligned}$$

Each actor adds his/her ratings mean to the received relative score to compute the absolute score:

$$\Delta score_k(a,i) = \delta score_k(a,a,i) + \overline{r_a}$$

This scoring shares even less data than the previous social scoring. Peers only share the difference between their ratings and the computed mean. They indicate that they like an item because it has a better rating than their mean.

4.4. *Correlative Social Scoring*

Since the trust relation used in the social network is manually defined by actors, it is subjective. We have introduced a correlation between actors to refine this coefficient. Unlike traditional approaches, the correlative social scorer does not compute this correlation between all actors to make a global social network, but it computes this correlation only between friends. The trust coefficient is then modulated with a similarity coefficient. This coefficient is a classic Pearson correlation coefficient, as in section 2 eq. 4, denoted by ρ. For $\rho_{a,f}$ computation between two friends a and f, only items rated by both friends are used. If the two friends have no item in common, the coefficient is not defined. The correlative social scoring take into account trust and correlation to compute scores, subsequently friends with similar tastes are "promoted" in the recommendation process. ρ is computed using $r_{a,i} - \overline{r_a}$, then we can use absolute or relative correlative social scoring. We will only define the relative social scoring here, but both scorers are implemented and evaluated in the section 5.

$$\delta score_k^\rho : A^2 \times I \to [0,1] \cup \{\bot\}$$
$$(a,l,i) \mapsto \delta score_k^\rho(a,l,i)$$

$$\delta score_k^\rho(a,l,i) = \begin{cases} r_{a,i} - \overline{r_a} & \text{if } \exists r_{a,i} \\ \dfrac{\sum_{f \in \mathcal{A}_{a,l,i}^{k-1}} t_{a,f} \times \rho_{a,f} \times \delta score_{k-1}^\rho(f,i,a)}{\sum_{f \in \mathcal{A}_{a,l,i}^{k-1}} t_{a,f} \times \rho_{a,f}} & \text{if } \nexists r_{a,i} \wedge \mathcal{A}_{a,l,i}^{k-1} \neq \emptyset \\ \bot & \text{otherwise} \end{cases}$$

And finally:

$$\Delta score_k^\rho(a,i) = \delta score_k^\rho(a,a,i) + \overline{r_a}$$

5. Evaluation

In order to evaluate the social approach explained in section 4, we have implemented several scoring algorithms: scorers presented in section 4, but also other non social scorers that are introduced in section 5.4. The dataset is presented in section 5.1. We have used different measures to compare the scoring algorithms, as shown in section 5.3. The evaluation is made using the dataset, but we ran two different evaluation campaigns as explained in section 5.2.

5.1. *Epinions dataset*

[Massa et Bhattacharjee2004] proposes a dataset derived from Epinions [1]. This dataset contains items rated by actors and trust values between actors. There are very few datasets alike available, most social datasets do not contain ratings and most ratings dataset do not include social networks. The dataset is anonymized: actors and items are numbers, so content-based approaches are not available here. A rating is an integer value between 1 (does not like) and 5 (likes). The dataset is very sparse (only 0.01 % ratings). The ratings are distributed as shown in figure 2. Half of the ratings are 5 and only the quarter is either 1, 2 or 3. The average is 4 and the standard deviation is 1.2. Nearly 60 % of the actors made five ratings or less. 70 % of the items have only one or two ratings.

Rating	1	2	3	4	5	total
Count	43 228	50 678	75 525	194 340	301 053	664 824
Percent	6.5 %	7.6 %	11.4 %	29.2 %	45.3 %	100 %

Figure 2. *Epinions ratings distribution*

1. www.epinions.com

An actor trusts or not another actor. The trust value is 1 (trusts) or 0 (does not trust). An actor trusts 10 actors on average. Most of the actors trust only a few actors and are trusted by few actors. Nearly 50 % of the actors trust three actors or less.

Since the ratings range from 1 to 5, we have normalized the dataset using a linear operation to stretch $[1, 5]$ to $[0, 1]$.

5.2. *Evaluation campaigns*

We ran two different kinds of evaluation campaign: the training set and the leave-one-out evaluation.

Training set: This is a classical method of evaluation: the Epinions dataset is split into two datasets: the evaluation dataset and the training dataset. The training dataset is used by algorithms to predict scores in the evaluation dataset. Then we measure the difference between predicted scores and real ratings contained in the evaluation dataset.
To process this campaign, we shuffle all the ratings, then we split the ratings in two groups. We considerate one experimentation for each shuffle. Then for each experimentation, we split the ratings with different sizes: 10 % - 90 %, 20 % - 80 %, 50 % - 50 %. The first group becomes the evaluation dataset (we try to predict these ratings), the second becomes the training dataset (we use these ratings to predict the others). We then run this experience for each ratio.

Leave One Out: In this campaign, we take the whole Epinions dataset. For each rating, we remove it, try to predict it, measure the difference between the predicted one and the real one and we finally put it back in the dataset. There is no random selection so this campaign is reproducible.

5.3. *Measures*

In section 2, we have selected some measures that are relevant for our problem. Predictive and rank accuracy metrics and coverage metrics are described in the state of the art. In addition to these metrics, here are two rank accuracy metrics based on individual actors and two weighting predictive accuracy metrics.

5.3.1. *Rank Correlation*

This measure allows to explore how predicted scores rank regarding the real ones. In other words, if each item has the same rank using ratings or scores, then the scorer is perfect, even if scores are different from ratings [McLaughlin et Herlocker2004].

Pearson Correlation Coefficient (PCC), as explained in section 2 (Eq. 4), is a correlation coefficient computed between items scores and items ratings, for each (actor, item, score) triple where the score is defined.

Actor Pearson Correlation Coefficient (APCC): for each actor, the Pearson correlation coefficient is computed between items scores and items ratings for this specific actor. The mean of all actors coefficient is then returned. Each actor coefficient has the same weight, whatever the number of ratings or scores from this actor is.

Spearman's Rank Correlation Coefficient (SRCC), as explained in section 2 (Eq. 4), is a correlation coefficient computed between items scores ranks and items ratings ranks, for each (actor, item, score) where the score is defined.

Actor Spearman's Rank Correlation Coefficient (ASRCC): for each actor, the Spearman's rank correlation coefficient is computed between items scores ranks and items ratings ranks for this specific actor. The mean of all actors coefficient is then returned. Each actor coefficient has the same weight, whatever the number of ratings or scores from this actor is.

5.3.2. *Error*

Error metrics are used to measure the distance between ratings and scores.

Mean Absolute Error and *Root Mean Square Error*, *c.f.* section 2 (Eq. 3).

Weighted Absolute Error and *Root Weighted Square Error* are similar to MAE and RMSE respectively, except that each error is weighted by the difference between the rating and ratings mean. That is, ratings equal to the mean do not count, and the further the rating is to the mean, the more it becomes important. This metric measures the accuracy for outlier ratings.

$$WAE = \frac{\sum_{n=1}^{N} \left(|p_n - r_n| \times |r_n - \bar{r}| \right)}{N} \qquad RWSE = \sqrt{\frac{\sum_{n=1}^{N} \left((p_n - r_n) \times (r_n - \bar{r}) \right)^2}{N}}$$

5.4. *Evaluated scorers*

We have defined four trust-based scorers in section 4. We aim to compare those with the existing scorers shown in figure 3. The less the scorer know, the best it is for

Scorer	Definition	Description	Knowledge
all ratings	$\overline{r}$	mean of all ratings r, *i.e.* 4	global
item ratings	$\overline{r_i}$	mean of all ratings r_i of item i	global
some item ratings	$\overline{r_i^k}, r_i^k \in \binom{r_i}{k}$	mean of k random ratings of i	local (k nearest peers)
$k-$depth social scorer	score_k	*c.f.* section 4.2	local trust and ratings
relative social scorer	Δscore_k	*c.f.* section 4.3	local trust and Δratings
correlative social scorer	score_k^ρ	*c.f.* section 4.4	local trust and ratings
relative correlative social scorer	$\Delta\text{score}_k^\rho$	*c.f.* section 4.4	local trust and Δratings

Figure 3. *Evaluated scorers*

privacy considerations. Knowledge associated with each scorer indicates how much information the scorer needs to compute scores. Global knowledge indicates that the

scorer needs all ratings existing in the social network. It can only use ratings from one specific item, but access all actors' profile. Local knowledge indicates a more limited knowledge on ratings. It can be randomly local, like for P2P architecture when a peer asks the k nearest nodes. It can be local trust, like for social scorers that access data from a specific set of actors, here the actor's friends. Local ratings knowledge indicates that only a subset of ratings is used by the scorer.

5.5. *Results*

Among the scorers described in figure 3, we have selected in this section the most interesting ones for the evaluation. Coverage metrics are indicated in figure 4 and accuracy measures in figure 5.

Scorer	Count	ActorCount	ActorScoreMean
$\overline{r_i^{10}}$	58114	19661	2.955801
$\overline{r_i}$	58114	19661	2.955801
$score_2$	37374	11229	3.328346
$\Delta score_2$	37260	11116	3.351925
$score_3^\rho$	27664	5535	4.998013
$score_1^\rho$	10761	3554	3.027856
$\Delta score_3^\rho$	27664	5535	4.998013
$\Delta score_1^\rho$	10761	3554	3.027856

(a) 90 % Training

Count	ActorCount	ActorScoreMean
115410	25935	4.449971
115410	25935	4.449971
73217	14361	5.098322
72943	14097	5.174363
54373	6168	8.815337
20187	4568	4.419221
54373	6168	8.815337
20187	4568	4.419221

(b) 80 % Training

Scorer	Count	ActorCount	ActorScoreMean
$\overline{r_i^{10}}$	275960	33520	8.232697
$\overline{r_i}$	275960	33520	8.232697
$score_2$	164632	17397	9.463241
$\Delta score_2$	163347	16408	9.955327
$score_3^\rho$	124960	6495	19.239415
$score_1^\rho$	38222	5378	7.107103
$\Delta score_3^\rho$	124921	6483	19.269011
$\Delta score_1^\rho$	38201	5366	7.119083

(c) 50 % Training

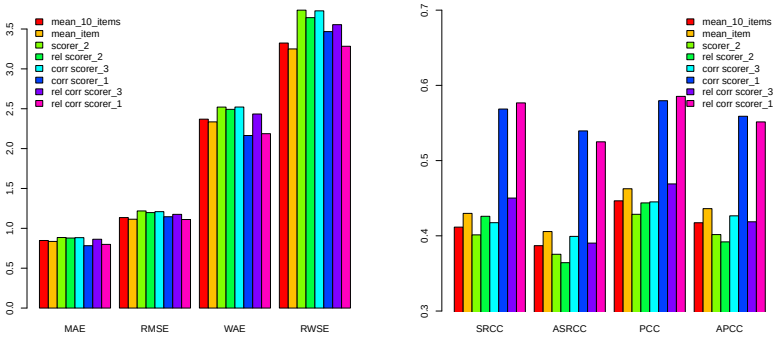
Count	ActorCount	ActorScoreMean
586359	39588	14.811534
586359	39588	14.811534
383218	21625	17.721064
381820	20227	18.876749
283010	6673	42.411209
114839	6673	17.209501
283010	6673	42.411209
114839	6673	17.209501

(d) Leave one out

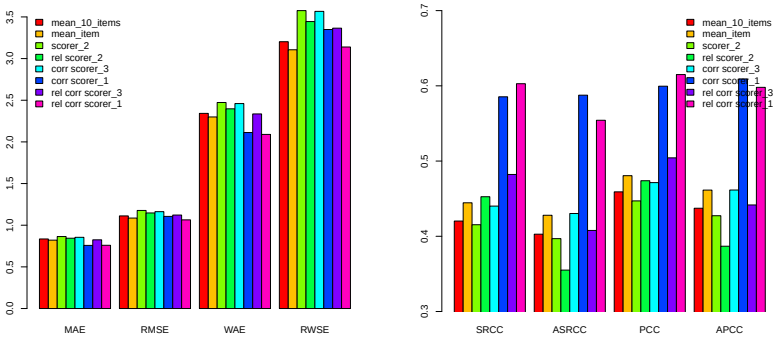
Figure 4. *Coverage metrics*

The training set campaigns show that even with a low number of ratings for the training set, social scorers perform good. The leave-one-out campaign, which provides the scorers with the maximum amount of existing ratings, is also well performed. It is even better performed with user centered rank accuracy metrics, which means that social scorers predict good personalized ratings. Figure 5 shows that our social scorers are more accurate than the global scorers both in terms of predictive and rank accuracy. Using a social network to predict ratings is efficient, and using only trust relation without creating new ones to preserve privacy does not reduce accuracy. The $\Delta score_k^\rho$ scorer is the best regarding these metrics.

Logically, for our social scorers, the greater k is, the better the coverage is, since more actors ratings are taken into account. However, even with $k = 2$, global scorers predict more ratings than social scorers. Since the formers know more information than the latter, we expected this result.



(a) Training set with 50 % of ratings (similar to 80 % and 90 %)



(b) Leave one out

Figure 5. *Predictive and rank accuracy metrics*

6. Conclusion

This paper focuses on recommender systems algorithms (aka. scorers), for trust-based systems with privacy concerns for decentralized architectures. Many of existing scorers use global knowledge on users and items profiles, and are not applicable to decentralized architectures. Regarding scorers built on social networks, trust-based algorithms do not always respect privacy since trust is often propagated through the network. We introduce several algorithms that can be implemented on decentralized network nodes and are privacy-aware. They use a limited vision of the network (the node where they are implemented and its direct trusted friends). They maintain a high level of privacy, since they do not propagate trust and only rely on trusted peers to propagate scores. Moreover, our relative scorers do not even share complete infor-

mation on users' scores. We present a comparative evaluation of these scorers using different metrics for predictive and rank accuracy and coverage. This evaluation shows that our scorers using immediate social relations ($k = 1$) have a very good accuracy but a low coverage. To enlarge this coverage, we propagate scores through the social network using only trusted peers with a limited depth (k). The evaluation results show that coverage increases fast using only friends of friends ($k = 2$). Even if our scorers still need to be compared to more accurate trust-based systems such as those used on e-commerce online services, it then appears that it is possible to build efficient decentralized, privacy-aware algorithms for recommender systems.

We remain conscious of the limits of our evaluation process. We have used the Epinions dataset to evaluate our approach - even if it has several drawbacks - since it is the only available dataset containing both trust and rating information. Other datasets should be interesting: lastFM with social relations between users and user ratings on music; FilmTrust [DuBois *et al.*2009] with trust relations and films ratings. However no public release of the two latters is currently available. The sparsity of the datasets (only 0.01 % ratings) remains a problem for local scorers, since they cannot find ratings for an item that has not been rated by a user's close environment. Content-based approaches that take advantage of similarities between items could be tested, in conjunction with our social scorers, to enlarge their coverage. In addition, since ratings are opinions, RS can use fuzzy logic to predict a score regarding existing ratings (I rather like, I rather don't). Today, our aggregating function uses a weighted scores mean; ordered weighted aggregation operators as presented in [Yager1988] could enhance the score prediction by taking into account the fuzziness involved by humans' ratings and relations.

7. References

[Adomavicius2005] Adomavicius G., "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions", *IEEE transactions on knowledge and*, vol. 17, num. 6, 2005, p. 734–749.

[Balabanović et Shoham1997] Balabanović M., Shoham Y., "Fab: content-based, collaborative recommendation", *Communications of the ACM*, vol. 40, num. 3, 1997, p. 66–72, ACM.

[Basu *et al.*1998] Basu C., Hirsh H., Cohen W., "Recommendation as classification: Using social and content-based information in recommendation", *Proceedings of the National Conference on Artificial Intelligence*, JOHN WILEY & SONS LTD, 1998, p. 714–720.

[DuBois *et al.*2009] DuBois T., Golbeck J., Kleint J., Srinivasan A., "Improving recommendation accuracy by clustering social networks with trust", *Recommender Systems & the Social Web*, , 2009, p. 1–8, Citeseer.

[Hang *et al.*2009] Hang C., Wang Y., Singh M., "Operators for propagating trust and their evaluation in social networks", *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, International Foundation for Autonomous Agents and Multiagent Systems, 2009, p. 1025–1032.

[Hasan2010] Hasan O., "Privacy Preserving Reputation Systems for Decentralized Environments", PhD thesis, Institut National des Sciences Appliquées de Lyon, 2010.

[Herlocker *et al.*1999] Herlocker J., Konstan J., Borchers A., Riedl J., "An algorithmic framework for performing collaborative filtering", *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 1999, p. 230–237.

[Herlocker *et al.*2004] Herlocker J., Konstan J., Terveen L., Riedl J., "Evaluating collaborative filtering recommender systems", *ACM Transactions on Information Systems (TOIS)*, vol. 22, num. 1, 2004, p. 5–53, ACM.

[Lee2009] Lee D., "Does Trust Influence Information Similarity?", *Proceedings of the ACM RecSys' 09 Workshop09 Workshop*, , 2009, p. 3–6.

[Ma *et al.*2009] Ma H., King I., Lyu M., "Learning to recommend with social trust ensemble", *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, New York, New York, USA, 2009, ACM, p. 203–210.

[Massa et Avesani2007] Massa P., Avesani P., "Trust-aware recommender systems", *Proceedings of the 2007 ACM conference on Recommender systems - RecSys '07*, , 2007, Page 17, ACM Press.

[Massa et Bhattacharjee2004] Massa P., Bhattacharjee B., "Using trust in recommender systems: an experimental analysis", *Trust Management*, , 2004, p. 221–235, Springer.

[McKnight et Chervany1996] McKnight D., Chervany N., "The meanings of trust", 1996.

[McLaughlin et Herlocker2004] McLaughlin M., Herlocker J., "A collaborative filtering algorithm and evaluation metric that accurately model the user experience", *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2004, p. 329–336.

[Mehta *et al.*2007] Mehta B., Hofmann T., Nejdl W., "Robust collaborative filtering", *Proceedings of the 2007 ACM conference on Recommender systems - RecSys '07*, , 2007, Page 49, ACM Press.

[Milgram1967] Milgram S., "The small world problem", *Psychology today*, vol. 2, num. 1, 1967, p. 60–67, New York.

[O'Donovan et Smyth2005] O'Donovan J., Smyth B., "Trust in recommender systems", *Proceedings of the 10th international conference on Intelligent user interfaces*, New York, New York, USA, 2005, ACM, p. 167–174.

[Page *et al.*1999] Page L., Brin S., Motwani R., Winograd T., "The PageRank Citation Ranking: Bringing Order to the Web.", 1999.

[Pitsilis et Knapskog2009] Pitsilis G., Knapskog S., "Social Trust as a solution to address sparsity-inherent problems of Recommender systems", *Recommender Systems & the Social Web*, , num. October, 2009, Page 33.

[Schafer *et al.*2007] Schafer J., Frankowski D., Herlocker J., Sen S., "Collaborative filtering recommender systems", *The adaptive web*, Springer-Verlag, 2007, p. 291–324.

[Wasserman et Faust1994] Wasserman S., Faust K., *Social network analysis: Methods and applications*, Cambridge Univ Pr, 1994.

[Yager1988] Yager R., "On ordered weighted averaging aggregation operators in multicriteria decisionmaking", *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 18, num. 1, 1988, p. 183–190, IEEE.