Alleviating the Sparsity Problem in Recommender Systems by Exploring Underlying User Communities

Aline Bessa Alberto H. F. Laender Adriano Veloso Nivio Ziviani
{alinebessa,laender,adrianov,nivio}@dcc.ufmg.br

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil

Abstract. *Collaborative Filtering*, one of the main Recommender Systems' approach, has been successfully employed to identify users and items that can be characterized as similar in large datasets. However, its application is limited due to the *sparsity problem*, which refers to a situation where information to infer similar users and predict items is missing. In this work, we address this by (i) detecting underlying user communities that aggregate similar tastes and (ii) predicting new relations within communities. As a consequence, we alleviate some of the major consequences of this problem. As shown by our experiments, our method is promising. When compared to a user-based *Collaborative Filtering* method, it provided gains of 20.2% in terms of sparsity decay, for instance, while improving RMSE values in only 2.8%.

1 Introduction

Recommender systems have been a popular topic of research since it became clear that people of widely varying backgrounds would be able to query the same underlying data [1]. Variations of recommender algorithms have been applied to (i) provide an automatic and intelligent mechanism to filter out the excess of information available to users and (ii) make personalized recommendations for information, products, and services during a live interaction. One of the most successful approaches to build recommender systems is called *Collaborative Filtering* (CF). It uses the known preferences of a group of users to predict unknown preferences for other users. Some of these predictions are then used for recommending items.

In a typical CF scenario, there is a bipartite network $U \times I$ where nodes are comprised by users U and items I. A link between a user u and an item i exists if and only if u rated i, and the weight of this link is the rating itself. From $U \times I$ one can infer a network $U \times U$, where nodes are users U and a link between two nodes u and v exists if and only if they co-rated at least one of the items in I. The weight of such link is given by the similarity between u and v's co-ratings, using a metric such as cosine or Pearson correlation. Traditional CF systems

predict ratings for users, with respect to items they did not rate yet, according to a scheme defined by the following equation:

$$\hat{r}_{u,i} = \frac{\sum_{v \in N_i(u)} r_{v,i} \times sim_{u,v}}{\sum_{v \in N_i(u)} |sim_{u,v}|} \quad (1)$$

where $\hat{r}_{u,i}$ is the predicted rating user u will give to item i, $N_i(u)$ is the set of all neighbors of u in $U \times U$ that rated item i, $sim_{u,v}$ is the similarity between u and its neighbors v in $U \times U$ (link weights), and $r_{v,i}$ is the actual rating each one of these neighbors gave to i (i.e., can be found in $U \times I$).

Given that even very active users rate just a few of the total number of available items and, respectively, even very popular items are rated by only a few of the total number of users, $U \times I$ and $U \times U$ are very sparse. High sparsity levels pose a big challenge to the quality of predictions, as well as to the number of predictions that a recommender system can actually compute. Because of sparsity, the confidence of predicted ratings may be questionable, since they are based on a rather little amount of evidence.

Other important structural problems related to data sparsity are *reduced coverage* and *neighbor transitivity* [2]. The former occurs when the system gets unable to generate recommendations for many items, as a consequence of the small ratio between users' ratings and the total number of items. The latter happens when users with similar taste are not identified as such because the available data is not enough to infer that.

Many approaches have been proposed to alleviate the data sparsity problem. Dimensionality reduction techniques, such as SVD and PCA, create a more dense representation of $U \times I$ [3–5]. However, when certain users or items are discarded, useful information related to them may get lost, which is not ideal and can degrade predictions' quality. Hybrid algorithms, exploring other user information, such as user profiles, were also studied [6]. Ziegler, Lausen and Schmidt-Thieme [7] generated profiles based on the taxonomic classification of products that costumers have chosen, and then used them for alleviating sparsity. Schein et al. [8] proposed a latent model for dealing with cold start recommendations, which combined both content and collaborative evidence. Nonetheless, hybrid solutions depend on the existence of content-based information, which is not always possible.

Other option – the one we explore in this work – involves making either $U \times U$ or $U \times I$ more dense via link prediction. Huang, Cheng and Zeng [9] tackle the problem via an associative retrieval framework and spreading activation algorithms to explore transitive associations among users. Papagelis, Plexousakis and Kutsuras [1] also explore transitive relations associating source and target users connected by paths with length bigger than 1. Yildirim and Krishnamoorthy [10] infer new relations via an item-oriented random walk algorithm, infering transition probabilities among items and modeling random walks on the item space to compute predictions.

Most link prediction methods are based on a 2-hop transitivity in $U \times U$ – i.e., if user u is connected with user v, and user v is connected with user w, then a link might be predicted between u and w. The intuition behind it is that users u and w are likely to have similar preferences. Links between nodes connected by a path with length bigger than 2 (3-hop, 4-hop etc.) are not considered by most methods. The reason is that, empirically, it is known that preferences are not so transitive [1].

$U \times I$ and $U \times U$ present a significant Small-World effect [11]. In particular, it means that virtually every two users in $U \times U$ are connected by short paths, despite of their taste differences. Quite distinct items and users in $U \times I$ are also connected by very short paths. As we will see later, predicting transitive links in the Small-World $U \times U$, without considering the inherent taste differences between users, leads to poor recommendations.

Our proposed method to tackle this problem involves two major steps (steps 2 and 3 in Fig. 1): (i) a community detection algorithm is applied for aggregating users with similar tastes in communities; (ii) only links between users within a same community are determined using different link predictors and weighted using a simple transitive schema. To the best of our knowledge, this is the first link prediction method for data sparsity that makes use of communities, in a divide-and-conquer fashion, ignoring either demographical data or explicit social ties. Sahebi and Cohen [12] proposed a community-based method for recommendations, but in a rather different scenario: they explore links of a social network, where users *explicitly decide* to connect with each other. In our case, users are linked if they co-rated items, without any knowledge about the specific taste of other users. It is worth pointing out that these differences configure networks with different semantics and properties.

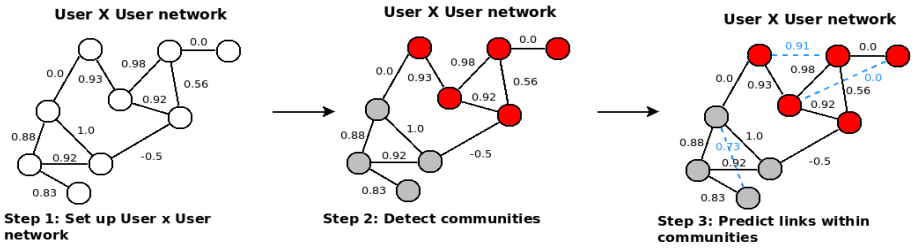

Fig. 1. A graphical representation of our proposed method.

The present work is structured as follows. Sections 2 and 3 detail each step of our method. Section 4 presents experiments and results. Finally, Section 5 discusses our conclusions and points out some future work lines.

2 Detecting Communities in the Users Network $U \times U$

The first step of our method involves detecting communities of users in $U \times U$. The detected communities must represent users with similar tastes. By construction, high link weights in $U \times U$ indicate taste similarity between users. Therefore, to detect similar taste communities, we should employ an algorithm that not only aggregates densely connected nodes, but guarantees that the connections among them are as high weighted as possible. The method proposed in [13] matches these prerequisites. For the sake of simplicity, this method, proposed by Blondel, Guillaume, Lambiotte and Lefbvere, will be hereafter refered to as BGLL.

Besides dealing with link weights in the desired way, BGLL scales very well with the number of nodes in the network, which is particularly interesting from a recommender systems' viewpoint. BGLL also works hierarchically, providing some clustering options with more or less granularity. There is at least one work in the field of recommender systems that already applies this algorithm, intending to find user communities for group recommendation [14]. Nevertheless, this work is not related with the data sparsity problem.

BGLL detects communities via modularity optimization. Modularity, a well known function in the complex networks field, is normally used to measure the quality of a partition. Basically, it searches for a network partition C that maximizes the Q value defined by:

$$Q = \frac{1}{2m} \sum_{i,j} [A_{i,j} - \frac{k_i k_j}{2m}] \delta(C_i, C_j) \tag{2}$$

where $A_{i,j}$ represents the weight of the link between i and j, $k_i = \sum_j A_{i,j}$ is the sum of the weights of the links attached to vertex i, C_i is the community to which vertex i is assigned, the δ function $\delta(u, v)$ is 1 if $u = v$ and 0 otherwise and $m = \frac{1}{2} \sum_{i,j} A_{i,j}$. Finally, $Q \in [-1, 1]$.

BGLL is based in two steps that are repeated iteratively. Initially, all nodes belong to their own community (N nodes and N communities). One looks through all nodes, from 1 to N, in an ordered way. Each node looks among its neighbors is observed, the node in question does not change of community. This step is performed iteratively until a local maximum of modularity is reached (each node may be considered several times). Once a local maximum has been attained, a new network, whose nodes are the detected communities, is built. The weight of the links between communities is the total weight of the links between the nodes of these communities.

The two steps are repeated iteratively, thereby leading to a hierarchical decomposition of the network. In this work, we will only take into account the communities corresponding to the last determined level of the hierarchy - i.e., the least granular ones. We explain this decision further in Section 4.

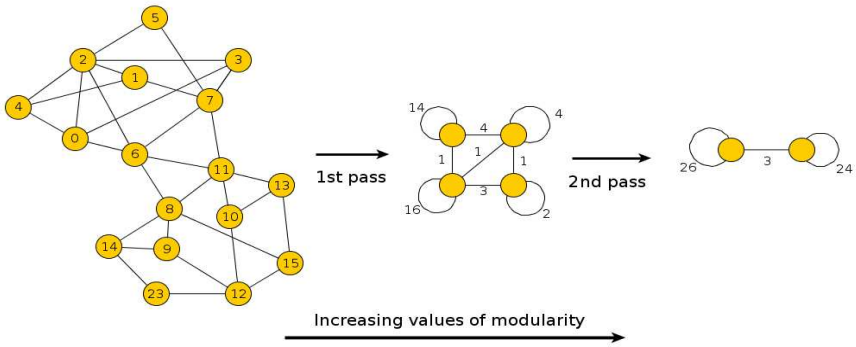

Fig. 2. A graphical representation of hierarchical communities found by BGLL.

3 Predicting Links in the Users Network $U \times U$

There are several techniques for predicting links in a network. Liben-Nowell and Kleinberg [15] have proposed an interesting taxonomy for them, separating the techniques into two main categories: (i) based on node neighborhoods; (ii) based on the ensemble of all paths.

In this work, we will not explore the second category, given that it generates links between nodes not so close. As mentioned previously, it is empirically known that preferences are not that transitive - e.g., if node u is similar to node v, node v is similar to node w and node w is similar to node y, one cannot state that u and y are also similar.

The techniques considered for this work are *Common neighbors*, *Jaccard's coefficient*, *Adamic/Adar*, and *Preferential attachment* [15]. All of them assign a connection $score(x, y)$ to pairs of non-connected nodes $\langle x, y \rangle$. All computed scores can then be ordered, generating a prediction list L_p. Below, we present the definitions adopted in this work for each predictor. For this, let $\Gamma(i)$ denote the set of neighbors of a node i.

Common neighbors This predictor determines the number of common neighbors between two nodes. Then:

$$score(x, y) = |\Gamma(x) \cap \Gamma(y)| \tag{3}$$

Jaccard's coefficient This predictor measures how likely a neighbor of node x is to be a neighbor of y and vice-versa. Then:

$$score(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{4}$$

Adamic/Adar This predictor assigns high weight to common neighbors z of x and y which themselves have few neighbors $\Gamma(z)$. Then:

$$score(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{|\Gamma(z)|} \tag{5}$$

Preferential attachment This predictor makes use of the empirical evidence that relations are correlated with nodes' neighborhood sizes. Then:

$$score(x, y) = |\Gamma(x)| \times |\Gamma(y)| \tag{6}$$

4 Experimental Results

In this work we applied our method to a dataset sampled from the movies' recommender site MovieLens[1]. It comprises 819 users and 1,180 items. The relations between these two entity types are ratings varying from 1 to 5. In this dataset, there are 7,851 rating relations between users and items, generating a bipartite graph $U \times I$: one partition corresponds to users (U) and the other corresponds to items (I). A second network $U \times U$ is induced from $U \times I$ following the traditional user-based CF approach outlined in Section 1.

We computed the similarity between users in $U \times U$ applying Pearson correlation [16] to their normalized co-ratings. Normalizing ratings is important to remove the noise created by the way each user rates – in other words, when it comes to assigning a rating to an item, each user has its own personal scale, and normalizing these scales leads to fairer Pearson correlations [16]. Since we are using normalized ratings for computing Pearson correlations, we also adapted the way we compute predictions for consistency. Thus, instead of using the traditional Equation 1, we adopted the following:

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N_i(u)} (r_{v,i} - \bar{r}_v) \times sim_{u,v}}{\sum_{v \in N_i(u)} |sim_{u,v}|} \tag{7}$$

After setting $U \times U$ up, we applied a C++ BGLL implementation for detecting user communities in it[2]. BGLL generated a hierarchy of four levels. The lower level (level 0) had 819 communities, one corresponding to each node. The higher one (level 3) had 627 communities. In this work, we only explored the communities reported in level 3, because the ones in levels 0, 1 and 2 were too fine-grained.

From 627 communities, only 11 were associated to more than one node. In addition, only 187 out of 819 users belonged to communities with more than one node. Analysing the users assigned to one-node communities, we discovered that

[1] MovieLens data, http://www.grouplens.org/.

[2] `http://sites.google.com/site/findcommunities`

all their links in $U \times U$ have 0.0 weight. This is the reason why BGLL never changes their communities, associating them with relevant neighbors: all their neighbors are equally relevant, given that all their links have the same weight.

In a certain sense, this reinforces the idea that there is a challenging long tail behind recommender systems: while few people have more regular preferences, being easy to aggregate, most users have very hard-to-describe tastes, limitating their allocation within communities. On the other hand, we believe that although the network topology does not help these users much, perhaps other clustering approaches can help finding good communities for at least a part of them. One possibility involves comparing attributes from their rated items and some of their own profile attributes.

For the purpose of this work, we only take into account the 11 communities with more than one user. These communities will be referred to as *relevant communities*.

Table 4 lists some of the most important movies rated in each community. Although some movies are assigned to multiple communities, such as "Contact", it seems they do reflect different user tastes. In a general sense, Table 4 indicates a certain degree of homophily among users[3] [17].

Table 1. Some relevant movies per community.

Community ID*	Movies
131	Jerry Maguire, Toy Story, Babe, Winnie the Pooh and the Blustery Day
136	Four Weddings and a Funeral, Forrest Gump, The Sound of Music, Dead Poets Society
26	The Birdcage, The Silence of the Lambs, The Godfather, Mighty Aphrodite
21	Alien, Contact, The Blues Brothers, Twelve Monkeys
174	Star Wars, Return of the Jedi, The Fugitive, The Silence of the Lambs
172	The Shawshank Redemption, Pulp Fiction, Raiders of the Lost Ark, Contact
170	Twelve Monkeys, The Full Monty, Fargo, Good Will Hunting
286	Citizen Kane, Psycho, Three Colors: Red, Platoon
52	Independence Day, Mission: Impossible, L.A. Confidential, Twelve Monkeys
63	Speed, Jurassic Park, Apollo 13, The Net

[3] In the context of this work, homophily (i.e., "love of the same") means the bias/tendency of individuals, indirectly associated via recommendations, to bond with similar others.

Inside every relevant community, we generated lists of new links L_p using all different predictors presented in Section 3. The weight of a predicted edge is given by:

$$sim(u,v) = \frac{1}{|N_{u,v}|}[\sum_{y \in N_{u,v}} sim(u,y) \times \sum_{y \in N_{u,v}} sim(y,v)] \quad (8)$$

where $N_{u,v}$ is the set of common neighbors of u and v. Given that we weight actual links in $U \times U$ using Pearson correlation, all similarities will range from -1.0 to 1.0. As a consequence, a multiplication between average similarities will always be, at most, equal to the highest similarity in question. This property is desirable in our scenario because we expect predicted similarities – which are transitive – to be lighter than actual similarities – which are naturally induced from $U \times I$. There are some other ways of estimating the weight of an edge in $U \times U$ [1, 9, 18], and we intend to explore them in the future.

After predicting and weighting links, we added the 10%, 20%, 30%, 40%, and 50% best – i.e., those with higher similarities – to $U \times U$. For the sake of completion, we also analysed the recommender system generated when we add 100% of the predicted links (Full L_p).

We compared our results with two baselines: the traditional user-based CF system discussed in Section 1 [19] and a user-based CF system that explores all $U \times U$ original relations *plus all* $U \times U$ *2-hop relations* [18]. As we outlined previously, there are many works that explore these relations, even though they apply very different strategies for link prediction. For the sake of simplicity, we will abbreviate the traditional user-based system as TUB, and the transitive-based system will be abbreviated as {1,2}-hopub. Notice that {1,2}-hopub is different from Full L_p, given that their links were predicted *within the entire network*, instead of only *between nodes within a same community*.

To assess the quality of our proposed solution, we do not measure neither the detected communities nor the predicted links quality directly. Alternatively, given that data sparsity challenges Recommender Systems' effectivity, we will evaluate our method through typical metrics of this field. Besides being more relevant with respect to the posed problem, using typical metrics eases the task of comparing our solution with chosen baselines.

The selected metrics are *Root mean-square-error (RMSE)*, *Coverage*, *Sparsity Decay (SD)*, *Recall*, and *Precision* [16]. Below, we present the adopted definitions for these metrics.

RMSE Given a test set T of user-item pairs $\langle u, i \rangle$, RMSE measures the divergence between real ratings $r_{u,i}$ and predicted ratings $\hat{r}_{u,i}$. The real ratings are known from offline experiments that do not interfere in $\hat{r}_{u,i}$ computations.

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{\langle u,i \rangle \in T} (\hat{r}_{u,i} - r_{u,i})^2} \quad (9)$$

Coverage In our context, coverage is the percentage of items for which the system is able to generate a recommendation. Let $|I|$ be the total of considered items and $|R(I)|$ be the number of items that are recommended at least once. Then:

$$Coverage = \frac{|R(I)|}{|I|} \tag{10}$$

Sparsity Decay (SD) It measures the improvement the system does with respect to relations between users and items. Let O be the set of original ratings in $U \times I$, and P be the set of predicted ratings between users and items. Then:

$$SD = \frac{|O|}{(|O| + |P|)} \tag{11}$$

Comparing two recommender systems A and B working over a fixed dataset, the one that presents the lowest SD is making more predictions. If both systems present very similar RMSEs, then the one that makes more predictions should be preferred, given that it increases the system's capacity of making different recommendations.

Given a test set T of user-item pairs $\langle u, i \rangle$, let TP be the number of user-item relations (ratings) that actually exist in T and that were predicted by the system; let FP be the number of user-item relations (ratings) that were predicted by the system although they do not exist in T; and let FN be the number of of user-item relations (ratings) that actually exist in T but were not predicted by the system. Then:

$$Recall = \frac{|TP|}{|TP| + |FN|} \tag{12}$$

$$Precision = \frac{|TP|}{|TP| + |FN|} \tag{13}$$

Tables 2-6 contain the obtained results for different considered percentages of links (i.e., L_p).

Evaluation metric	**Common neighbors**	**Jaccard's coefficient**	**Adamic / Adar**	**Pref. Attachment**	**TUB**	**{1,2}-hopub**	**Full L_p**
RMSE	1.045	**1.036**	1.038	**1.051**	1.010	1.367	1.082
Coverage	0.863	0.863	0.863	0.863	0.863	0.886	0.863
SD	0.246	**0.258**	0.246	**0.236**	0.297	0.007	0.192
Recall	0.274	**0.270**	0.274	**0.276**	0.254	0.570	0.285
Precision	0.006	0.006	0.006	**0.005**	0.007	0.003	0.004

Table 2. Results obtained with the 10% best links in L_p. L_p is different for each considered metric: *Common neighbors*, *Jaccard's coefficient*, *Adamic/Adar*, and *Preferential Attachment*. The baselines are listed for comparison.

Evaluation metric	**Common neighbors**	**Jaccard's coefficient**	**Adamic / Adar**	**Pref. Attachment**	**TUB**	**{1,2}-hopub**	**Full L_p**
RMSE	1.068	**1.039**	1.067	**1.081**	1.010	1.367	1.082
Coverage	0.863	0.863	0.863	0.863	0.863	0.886	0.863
SD	0.223	**0.237**	0.222	**0.208**	0.297	0.007	0.192
Recall	0.281	**0.275**	0.281	**0.284**	0.254	0.570	0.285
Precision	0.005	0.005	0.005	0.005	0.007	0.003	0.004

Table 3. Results obtained with the 20% best links in L_p. L_p is different for each considered metric: *Common neighbors*, *Jaccard's coefficient*, *Adamic/Adar*, and *Preferential Attachment*. The baselines are listed for comparison.

Evaluation metric	**Common neighbors**	**Jaccard's coefficient**	**Adamic / Adar**	**Pref. Attachment**	**TUB**	**{1,2}-hopub**	**Full L_p**
RMSE	1.072	**1.066**	1.071	**1.126**	1.010	1.367	1.082
Coverage	0.863	0.863	0.863	0.863	0.863	0.886	0.863
SD	0.212	**0.222**	0.210	**0.191**	0.297	0.007	0.192
Recall	0.282	**0.280**	0.282	**0.292**	0.254	0.570	0.285
Precision	0.005	0.005	0.005	**0.004**	0.007	0.003	0.004

Table 4. Results obtained with the 30% best links in L_p. L_p is different for each considered metric: *Common neighbors*, *Jaccard's coefficient*, *Adamic/Adar*, and *Preferential Attachment*. The baselines are listed for comparison.

Results in bold correspond to the best and worst values found for each metric in each experimental scenario. It is worth pointing out that, when compared against TUB, none of the proposed solutions - *Common neighbors*, *Jaccard's coefficient*, *Adamic/Adar*, *Preferential attachment*, and *Full* L_p - improved global coverage. This means that our method did not generate predictions for any item that was not already being covered by TUB. Even though, we generated many more predictions for already covered items, as indicated by SD - i.e., although the items are the same, there are predictions relating them to many more users, which is a highly desirable feature in a recommender system. One can state that {1,2}-hopub outperforms all other competitors in terms of global Coverage and low SD, but the quality tradeoff is not acceptable: all RMSE values are consistently higher when compared against TUB.

Concretely, the most important metric is RMSE. If the error is not low, increasing the number of predictions is not worth it: one could always recommend random items if quality (low error) does not matter. Of course, this is not the case and that is why we are particularly interested in the results obtained with *Jaccard's coefficient*. This predictor presents an interesting tradeoff between SD and RMSE. For practical purposes, using the 20% best links in the L_p related with *Jaccard's coefficient* improves the number of predictions significantly (SD goes from TUB's 0.297 to 0.237) without adding much noise in quality (RMSE goes from TUB's 1.010 to 1.039).

Evaluation metric	Common neighbors	Jaccard's coefficient	Adamic / Adar	Pref. Attachment	TUB	{1,2}-hopub	Full L_p
RMSE	1.080	**1.076**	1.079	**1.130**	1.010	1.367	1.082
Coverage	0.863	0.863	0.863	0.863	0.863	0.886	0.863
SD	0.205	**0.211**	0.204	**0.179**	0.297	0.007	0.192
Recall	0.283	**0.282**	0.283	**0.293**	0.254	0.570	0.285
Precision	0.005	0.005	0.005	**0.004**	0.007	0.003	0.004

Table 5. Results obtained with the 40% best links in L_p. L_p is different for each considered metric: *Common neighbors*, *Jaccard's coefficient*, *Adamic/Adar*, and *Preferential Attachment*. The baselines are listed for comparison.

Evaluation metric	Common neighbors	Jaccard's coefficient	Adamic / Adar	Pref. Attachment	TUB	{1,2}-hopub	Full L_p
RMSE	1.080	**1.079**	1.082	**1.139**	1.010	1.367	1.082
Coverage	0.863	0.863	0.863	0.863	0.863	0.886	0.863
SD	0.200	**0.204**	0.200	**0.171**	0.297	0.007	0.192
Recall	0.284	**0.283**	0.284	**0.297**	0.254	0.570	0.285
Precision	0.004	0.004	0.004	0.004	0.007	0.003	0.004

Table 6. Results obtained with the 50% best links in L_p. L_p is different for each considered metric: *Common neighbors*, *Jaccard's coefficient*, *Adamic/Adar*, and *Preferential Attachment*. The baselines are listed for comparison.

As expected, RMSE increases when we add more links – perhaps, it could be alleviated if the weight of predicted links were computed in a different fashion, but we do not believe that it would solve this issue completely. Also, for all studied scenarios, adding more links consistently improved the number of predictions, generating lower SDs. Finally, as expected, higher RMSE values are highly correlated with lower SD values – in other words, generating more predictions necessarily decreased quality.

Another analysed metrics were precision and recall. It is easy to observe that all precision values are consistently smaller than recall values. It indicates that, even if a significant fraction of the predicted user-item relations actually exist (recall), most real relations are missing (precision). There is not a balance between these two aspects, which may mean that the analysed recommenders are not fitting user tastes very well, but it would take further investigations to assert it. Despite this behavior, we want a solution that minimizes the difference with respect to original TUB's precision and recall. Once again, *Jaccard's coefficient* is a good option. For the sake of comparison, there is at least one work that analyses precision and recall in link prediction based recommender systems, also presenting quite low values for both metrics [20].

5 Conclusions and Future Work

In this work, we proposed a method for alleviating the sparsity problem in CF recommender systems, keeping a commitment with good quality predictions. Concretely, we analysed the impact of using community detection before predicting links in the CF Recommender Systems scenario. To the best of our knowledge, this is the first attempt in this direction. The results indicate that predicting links *only within users belonging to the same community* helps lowering prediction errors, when compared with a fully-transitive baseline, {1,2}-hopub.

The proposed method is also positive in terms of alleviating some of the drawbacks related to the *sparsity problem*: via communities, we delimitate users with similar taste and, when we link them, we are presenting a solution to the *neighbor transitivity* problem. Although our method did not increase global coverage, when compared with original TUB system, it improved the number of predictions for already covered items, which is highly desirable. Finally, with more links connecting users in $U \times U$, we increase the amount of information used by the system for making predictions, increasing its confidence.

The proposed method, nonetheless, does not present a solution to the *cold start problem* [8]. In particular, users that never rated an item do not connect with any other user. Therefore, they do not belong to any relevant community, and links involving them will never be predicted. If there were more available information for grouping users, such as demographical data, *cold start users* would not be isolated. It is likely that additional information would also be useful for grouping long-tail users - those that did not fit in any community due to their very particular movie taste.

In the future, we intend to (i) explore different methods for detecting communities and predicting links, (ii) compare different link weighting schemes, (iii) apply the proposed solution to bigger networks, (iv) compare our method with more robust baselines, (v) consider the usage of social/demographical data, and (vi) predict links on demand only for users who suffer from sparsity.

Acknowledgements

This work was partially sponsored the Brazilian National Institute of Science and Technology for the Web (grant MCT/CNPq 573871/2008-6), and by the authors' individual grants and scholarships from CAPES, CNPq and FAPEMIG.

References

1. Papagelis, M., Plexousakis, D., Kutsuras, T.: Alleviating the sparsity problem of collaborative filtering using trust inferences. In: Proceedings of the Third International Conference on Trust Management. (2005) 224–239
2. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence **2009** (2009)

3. Billsus, D., Pazzani, M.J.: Learning collaborative information filters. In: Proceedings of the 15th International Conference Machine Learning. (1998) 46–54
4. Landauer, T., Littman, M., Research, B.C.: Computerized cross-language document retrieval using latent semantic indexing (1994) US patent no. 5301109.
5. Goldberg, K.Y., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm. Journal Information Retrieval **4**(2) (2001) 133–151
6. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Proceedings of the 18th National Conference on Artificial Intelligence. (2002) 187–192
7. Ziegler, C.N., Lausen, G., Schmidt-Thieme, L.: Taxonomy-driven computation of product recommendations. In: Proceedings of the 13th ACM Conference on Information and Knowledge Management. (2004) 406–415
8. 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. (2002) 253–260
9. Huang, Z., Chen, H., Zeng, D.: Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Transactions on Information Systems **22**(1) (2004) 116–142
10. Yildirim, H., Krishnamoorthy, M.S.: A random walk method for alleviating the sparsity problem in collaborative filtering. In: Proceedings of the 2nd ACM International Conference on Recommender Systems. (2008) 131–138
11. Barabasi, A.L.: Linked: How Everything is Connected to Everything Else and What it Means for Business, Science and Everyday Life. Plume (2003)
12. Sahebi, S., Cohen, W.: Community-based recommendations: a solution to the cold start problem. In: Proceedings of the Workshop on Recommender Systems and the Social. (2011)
13. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics **2008**(10) (2008)
14. Boratto, L., Carta, S., Chessa, A., Agelli, M., Clemente, M.L.: Group recommendation with automatic identification of users communities. In: Proceedings of the 3rd International Workshop on Distributed Agent-Based Retrieval Tools. (2009) 547–550
15. Liben-Nowell, D., Kleinberg, J.M.: The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology **58**(7) (2007) 1019–1031
16. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B., eds.: Recommender Systems Handbook. Springer (2011)
17. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual Review of Sociology **27**(1) (2001) 415–444
18. Aggarwal, C.C., Wolf, J.L., lung Wu, K., Yu, P.S.: Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (1999) 201–212
19. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering **17**(6) (2005) 734–749
20. Huang, Z., Li, X., Chen, H.: Link prediction approach to collaborative filtering. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries. (2005) 141–142