Socially Enabled Preference Learning from Implicit Feedback Data

Julien Delporte[1], Alexandros Karatzoglou[2], Tomasz Matuszczyk[3], and Stéphane Canu[1]

[1] INSA de Rouen, LITIS, Rouen, France
firstname.lastname@insa-rouen.fr
[2] Telefonica Research, Barcelona, Spain
alexk@tid.es
[3] Tuenti, Barcelona, Spain
tomasz@tuenti.com

Abstract. In the age of information overload, collaborative filtering and recommender systems have become essential tools for content discovery. The advent of online social networks has added another approach to recommendation whereby the social network itself is used as a source for recommendations i.e. users are recommended items that are preferred by their friends.

In this paper we develop a new model-based recommendation method that merges collaborative and social approaches and utilizes implicit feedback and the social graph data. Employing factor models, we represent each user profile as a mixture of his own and his friends' profiles. This assumes and exploits "homophily" in the social network, a phenomenon that has been studied in the social sciences. We test our model on the *Epinions* data and on the *Tuenti* Places Recommendation data, a large-scale industry dataset, where it outperforms several state-of-the-art methods.

1 Introduction

Online social networks (OSN) provide users with new forms of interaction that currently shape the social lives of millions of people. The main ingredient of the success of OSN's is the ease with which friendships, groups and communities arise. These groups often arise among like-minded users, i.e. users that share the same interests. To explain our inexorable tendency to link up with one another in ways that reinforce rather than test our preferences sociologists in the 1950s, coined the term "homophily" a Greek word meaning love of the same.

Fundamental to online social networks and their commercial success is the commercial exploitation of this phenomena. The principle of homophily is used to recommend products and services through the social graph, i.e. if your friends like an item it will be recommended to you. In effect, the social graph is used as the recommendation engine. Leveraging the social graph to serve the user with potentially useful services (e.g. places, videos, coupons, etc.) can improve the

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part II, LNAI 8189, pp. 145–160, 2013.

satisfaction the involvement and the time the user spends on the network. Most recommendation algorithms work by modeling the bipartite graph of user-item preferences. In effect, an implicit social network among users who share the same taste is built and exploited. Methods based on this principle are often referred to as Collaborative Filtering (CF) methods.

Notation. Before going any further with CF some notations have to be introduced. The data from which recommendations can be produced is typically derived from interactions between users $i \in \mathcal{U}$ and items $j \in \mathcal{M}$ with a response $Y_{ij} \in \mathcal{Y}$. The data for n total users interacting with a collection of m items can be thought of as a sparse $n \times m$ so-called user-item matrix $Y \in \mathcal{Y}^{|\mathcal{U}| \times |\mathcal{M}|}$ where $|\mathcal{U}|$ denotes the cardinal of set $\mathcal{U}$. In this context, $Y_{ij} = 1$ indicates the existence of an interaction (purchase, rating, etc.) between user i and item j. In this sense, $Y_{ij} = 0$ is special since it does *not* indicate that a user dislikes an item but rather that data is missing. We thus only have implicit information on which items a user likes. To thus avoid an estimator that is overly optimistic with regards to user preferences we need to take into account unobserved entries ($Y_{ij} = 0$) as some form of negative feedback. Moreover from the social network graph we know the set of friends $\mathcal{F}_i \subset \mathcal{U}$ of user i.

A class of CF methods often used in recommender systems is memory or similarity based methods [1] that work by computing similarity measures (*e.g.* Pearson correlation) between users. Another common approach to collaborative filtering and recommendation is to fit a factor model (e.g. [2]) to the data. For example by extracting a feature vector U_i, M_j for each user and item in the data set such that the inner product of these features minimizes an explicit or implicit loss functional following a probabilistic approach). The underlying idea behind these methods is that both user preferences and item properties can be modeled by a number of latent factors.

The basic idea in matrix factorization approaches is to fit the original Y matrix with a low rank approximation $F = UM$ where matrix U contains the user features and M the item features. More specifically, the goal is to find such an approximation that minimizes the sum of the squared distances $\sum_{ij}(Y_{ij} - F_{ij})^2$ between the known entries in Y and their predictions in F. Combining the two approaches, i.e. direct recommendation over the social graph and recommendations using a collaborative filtering method can yield significant advantages both in terms of the quality of the recommendations but also in terms of computational efficiency and speed.

In most recommendation domains the data come in the form of implicit feedback (purchases, clicks, etc.) in contrast to explicit feedback such as ratings where a user explicitly expresses his positive, neutral or negative attitude towards an item. A key challenge in modeling implicit feedback data is defining negative feedback, since in this case the observed data (user-item interactions) can only be considered as a form of positive feedback. Moreover for non-observed $user - item$ interactions we cannot be certain if the user did not consider the items or if the user considered the items and simply chose not to interact with

the items (reflecting a negative feedback). Hence we cannot ignore these entries as this would lead to a model that would be overly optimistic with regard to user preferences.

The Socially Enabled Collaborative Filtering model denoted as *SECoFi* introduced here has several novel aspects:

- We develop a collaborative filtering model that also directly models the social interactions and quantifies the influence/trust between each users based on the implicit feedback data from the user and his friends.
- We develop a way to quantify and use this influence in the proposed collaborative filtering model, to our knowledge this is the first model to do so without precomputing any affinity or similarity measures among users.
- *SECoFi* scales linearly to the number of user-item interactions and is tested on a large-scale industry dataset with over 10M users where it outperforms state-of-the-art socially enabled collaborative filtering methods.
- We extensively test *SECoFi* on two datasets (*Tuenti, Epinions*) and compare it to three state-of-the-art socially-enabled collaborative filtering methods and a matrix factorization method.

2 Socially Enabled Collaborative Filtering

The main idea behind factor models is to fit a model of a d dimensional latent user $U \in \mathbb{R}^{|\mathcal{U}|\times d}$ and item factors $M \in \mathbb{R}^{d\times|\mathcal{M}|}$ so that the scores between a user and an item calculated by the inner product between the corresponding rows of the user i and item j latent factor matrices $F_{ij} = U_i M_j$ can be used to provide recommendations typically by displaying the top N scoring items to the user. The latent factors U and M are typically computed by minimizing some objective functions that either stem from regularized loss functions [3,4,5] or are derived from probabilistic models [6]. In both cases the objectives are of the form:

$$L(F, Y) + \Omega(F) \tag{1}$$

where $L(F, Y)$ is typically a loss function such as Frobenius norm of the error $\|F - Y\|_F^2$ and $\Omega(F)$ is a regularization term preventing from overfitting. A typical choice is the Frobenius norm of the factors $\|M\|_F^2 + \|U\|_F^2$ [7].

2.1 Friends Influence

The key challenge of this work is to include the influence of the social graph in a matrix factorization model. We choose to model the users preferences as a mixture of his own and those of his friends. To this end we change the score function to include the influence of the friendship network, and thus the score function becomes:

$$F_{ij} = U_i M_j + \sum_{k\in\mathcal{F}_i} \frac{\alpha_{ik}}{|\mathcal{F}_i|} U_k M_j \tag{2}$$

where α_{ik} is a weight parameter that encodes how much friend k influences user i. This weight α takes value between 0 and 1.

As we presume "homophily" in the social network it is reasonable to assume that some of the users latent preferences might not have been expressed in the user-item data but could instead be encoded in the users friendship network.

Moreover the score function in Equation 2 encodes the fact that the user is "influenced" by his friendship network and the weight α_{ik} quantifies the amount of influence each individual friend k has on the user i. OSN users tend to have dozens of friends and we can expect that the user might have similar preferences to only a fraction of his friends. Moreover it should be noted that the influence is not necessarily symmetric as a user might be "influenced" by a friend but might not be exerting influence on his friend in the same manner.

Given this score function and the objective function in Equation 1, we can devise an objective function with respect to the U, M factors and the influence weights α_{ik}. We define the matrix A such that $A_{ik} = \alpha_{ik}, \forall i, \forall k \in \mathcal{F}_i$, 0 otherwise.

$$\min_{U,M,A} J = \sum_{(i,j)\in\mathcal{Y}} c_{ij}\big(U_i M_j + \sum_{k\in\mathcal{F}_i} \frac{\alpha_{ik} U_k M_j}{|\mathcal{F}_i|} - Y_{ij}\big)^2 + \Omega_{U,M,A} \tag{3}$$

where $\Omega_{U,M,A} = \lambda_1 \|U\|_F^2 + \lambda_2 \|M\|_F^2 + \lambda_3 \|A\|_F^2$ is a regularizer term and c_{ij} is a constant defined to give more weight to the loss function when dealing with observed entries $Y_{ij} = 1$ than when $Y_{ij} = 0$.

2.2 Optimization

Although Equation 3 is not *jointly convex* in U, M, and A, it is still convex in each of this factors whenever the remaining two are kept fixed. Since we are dealing with implicit feedback data, we cannot give the same importance to information we know to be true, (i.e. the user clicked/purchased an item represented as a 1 in the Y matrix and thus showed and interest in it), and to information we do not know the real meaning (i.e. the user had no interaction with the item thus a 0 in the Y matrix and thus we are unsure about the potential interest). Note that in contrast to factor models for explicit data (i.e. ratings) where learning is performed only over the observed ratings in this case we perform the optimization over the whole matrix Y including the unobserved entries as a form of weak negative feedback.

We optimize the objective function in Equation 3 using the following block Gauss-Seidel process: fixing alternatively two of the three parameters (U, M or A), we update the third parameter. When two out of three parameters are fixed the remaining problem is a basic and convex quadratic least-square minimization that can be efficiently solved. So the optimization process consists in efficiently updating, at each iteration, alternatively the user matrix U, the item matrix M and the weight matrix A.

To get the proper updates for each of the three parameters (U_i, M_j and $\alpha_{ii'}$), we need to calculate the partial derivative of the objective in Formula 3 according to the corresponding factor matrices.

Update U. To compute the update for the factor vector of a single user i, U_i, we calculate $\frac{\partial J}{\partial U_i}$ the derivative of the objective with respect to the users factors and set it to 0. We can then analytically solve this expression with respect to U_i. To formulate the update it is convenient to write the equations in a matrix form. To this end we define a diagonal matrix $C^i \in \mathbb{R}^{|\mathcal{M}|\times|\mathcal{M}|}$ such that $C^i_{jj} = c_{ij}$. c_{ij} encodes the confidence we have in each entry y_{ij} in the Y matrix, i.e. observed entries clicks/purchases etc. get high confidence and thus a higher weight $c_{ij} = 1 + \beta y_{ij}$ where e.g. $\beta = 20$ while when $y_{ij} = 0$ i.e. no action has been taken by user i on item j, $y_{ij} = 0$ and thus $c_{ij} = 1$.

$$U_i = \left(Y_{i\bullet}C^i M^T - \frac{A_i U M C^i M^T}{|\mathcal{F}_i|}\right)(MC^i M^T + \lambda_1 I)^{-1} \tag{4}$$

In this update rule, the real problem is not the inversion of the $d \times d$ (which is in $O(d^3)$) matrix but the computation of MC^iM^T (which seems to be at first glance $O(|\mathcal{M}| \times d^2)$). Note that MC^iM^T is an operation quadratic in $|\mathcal{M}|$ the number of items. Computing this product is too expensive even for the smallest datasets since it has to be done for each user. In the spirit of [8] we can replace MC^iM^T by $MM^T + M(C^i - I)M^T$. Computing MM^T is independent of the user i and thus can be calculated once before each iteration (and not for each user i), and by cleverly choosing c_{ij}, the product $M(C^i - I)M^T$ can be computed efficiently. Since $c_{ij} = 1 + \beta y_{ij}$, the diagonal terms of $C^i - I$ will be zero for each j where $y_{ij} = 0$. we can thus just compute $M_{\mathcal{Y}_i}(C^i - I)_{\mathcal{Y}_i}M^T_{\mathcal{Y}_i}$, where $\mathcal{Y}_i$ is the set of items of user i. Because matrix Y is by it's nature very sparse, we have $|\mathcal{Y}_i| \ll |\mathcal{M}|$. This leads to a computational complexity of $O(|\mathcal{Y}_i| \times d^2)$ which is linear in the number of items user i had interactions.

Update M. To update matrix M, we use a matrix U' defined by $U'_i = U_i + \sum_{k\in\mathcal{F}_i} \frac{\alpha_{ik}U_k}{|\mathcal{F}_i|}$ for each user i. Using U' the loss function becomes:

$$L(U, M, A) = \sum_{i,j} c_{ij}(U'_i M_j - Y_{ij})^2$$

The partial derivative calculation is pretty much straightforward and can be easily written in a matrix notation if we use a diagonal matrix C^j, defined by $C^j_{ii} = c_{ij}$[1]. The update rule of M_j is as follows:

$$M_j = (U'^T C^j U' + \lambda_2 I)^{-1} U'^T C^j Y_{\bullet j} \tag{5}$$

To compute the expensive product, we propose to reuse the trick described above by writing it $U'^T C^j U' = U'^T U' + U'^T_{\mathcal{Y}_j}(C^j - I)_{\mathcal{Y}_j}U'_{\mathcal{Y}_j}$, where $\mathcal{Y}_j$ is the set of the users that have purchased/consumed item j . Just like in the paragraph concerning the update of U, we compute $U'^T U'$ once before the iteration over all items. The computational complexity of the update is $U'^T_{\mathcal{Y}_j}(C^j - I)_{\mathcal{Y}_j}U'_{\mathcal{Y}_j}$ is $O(|\mathcal{Y}_j| \times d^2)$.

[1] Note that $C^j \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{U}|}$ while $C^i \in \mathbb{R}^{|\mathcal{M}|\times|\mathcal{M}|}$.

Update A. One approach for updating A consists in working row by row, i.e. update $A_{i\bullet}$ for each user i. Since $A_{i\bullet}$ has the same sparsity structure as the adjacency matrix of the social graph we only need to compute the values $A_{i\mathcal{F}_i}$. By using the same procedure as above and setting the partial derivative of the objective to 0 we get:

$$A_{i\mathcal{F}_i} = (Y_{i\bullet}C^i M^T U_{\mathcal{F}_i}^T - U_i M C^i M^T U_{\mathcal{F}_i}^T) \left(\frac{U_{\mathcal{F}_i} M C^i M^T U_{\mathcal{F}_i}^T}{|\mathcal{F}_i|} + \lambda_3 \right)^{-1} \tag{6}$$

Note again that the computational cost for calculating the product $U_i M C^i M^T U_{\mathcal{F}_i}^T$, is limited since we can employ here the same trick we used in the update rules for U and M. The main computational bottleneck is in the computation of the inverse of the matrix which is of size $|\mathcal{F}_i| \times |\mathcal{F}_i|$, implying a complexity in $O(|\mathcal{F}_i|^3)$ i.e. the computation scales cubically to the number of friends per user. Depending on the social network, if we have $d \ll |\mathcal{F}_i|$ for a significative fraction of users, this update rule could be problematic.

Another approach for the update of α, is to compute them not in a user-by-user fashion but relationship-by-relationship, i.e. update $\alpha_{ii'}$ for given user i and friend i'. By calculating the gradient and setting it to zero, we reach the following update rule:

$$\alpha_{ii'} = \left(Y_{i\bullet} - U_i M - \sum_{k \in \mathcal{F}_i k \neq i'} \frac{\alpha_{ik} U_k M}{|\mathcal{F}_i|} \right) C^i M^T U_{i'}^T \left(\frac{U_{i'} M C^i M^T U_{i'}^T}{|\mathcal{F}_i|} + \lambda_3 \right)^{-1} \tag{7}$$

In this case, we just have to invert a scalar. And we can use the same trick as in the update of U to compute the product MC^iM^T. This can indeed be rewritten as $MC^iM^T = MM^T + M_{\mathcal{Y}_i}(C^i - I)_{\mathcal{Y}_i} M_{\mathcal{Y}_i}^T$, where $\mathcal{Y}_i$ is the set of the items liked/purchased by the user i.

Given that the complexity of computing Equation 7 is linear to the number of friends of i, while the complexity of Equation 6 is polynomial to the number of friends of i we choose to use 7. Finally note that the α parameters provide a relative measure of the influence (or trust) of a given user on his friends.

Given the optimization procedures for U, M and A we iterativly update each of the factor matrices by keeping the other two factor matrices fixed. We repeat this procedure until convergence.

Prediction. Using Equation 2 at prediction time can be slow since it requires extensive memory access due to the need to retrieve the friends from the social graph. To speedup the computation of the scores at prediction time we can simply precompute the mixed user factors $U_i' = U_i + \sum_{k \in \mathcal{F}_i} \frac{\alpha_{ik}}{|\mathcal{F}_i|} U_k$. The score computation then becomes $F_{ij} = U_i' M_j$.

3 Related Work

Much of the current work on OSN data and collaborative filtering models utilize the social graph data in order to impose additional constrains on the modeling process. Some methods [9,10,11] leverage the OSN graph information in factor models by adding an additional term to the objective function of the matrix factorization that penalizes the distance $|U_i - \frac{1}{|\mathcal{F}_i|}\sum_{k\in\mathcal{F}_i} U_k|^2$ between the factors of friends. This forces profiles among users that are friends to be similar. In [9] a refinement to this approach was presented whereby the penalization of the distance between friends was proportional to a Pearson correlation similarity measure $|U_i - \frac{\sum_{k\in\mathcal{F}_i} sim_{ik}U_k}{\sum_{k\in\mathcal{F}_i} sim_{ik}}|^2$computed on the items the users had consumed. This enforced even greater similarity among friends that have consumed the same items.

Another approach [12,13] that builds on [14] adds the OSN information by minimizing a second binary loss function $\sum_{k\in\mathcal{F}_i} L(S_{ik}, U_iU_k)$, where S the adjacency matrix of the graph, in the objective function that penalizes mistakes in predicting friendship. These models also leverage side information (i.e. user, item features) in the model. A similar method utilizing both a social regularization and a social loss function approach was introduced in [15].

In [16] a trust ensemble model is introduced, the user is modeled as an ensemble of his own and his friends preferences. While the functional form of this model has similarities with the approach introduced in the current work there are two crucial differences: 1) their method only deals with explicit feedback data (ratings) while we focus on implicit feedback data which is the norm in industry applications, 2) they precompute the weight of the influence or trust of friends on the users based on the ratings, while in *SECoFi* the interaction weights are computed in the model. This allows us to compute the interaction weights even when the users do not actually share a common subset of items. We demonstrate in the **Experiments** section that these are essential components for the performance of the model.

The matrix factorization approach for implicit feedback data introduced in [8] relies on using a least squares loss function and uses a trick that exploits the sparse structure of the data (dominated by non-observed entries) to speed up the optimization process. This approach though does not include any OSN information. An approach that leverages the social network for apps recommendation was introduced in [17]. Approaches such as [18] and [19] exploit geolocation information and context to recommend places to user. The focus of the current work is on the OSN integration for place recommandation.

4 Experiments

Tuenti Places. In the experiment section we use data from the places service of the Tuenti OSN. Tuenti is Spain's leading online Social Network in terms of traffic. Over 80% of Spaniards aged 14-27 actively use the service and today counts more than 14M users and over a billion daily page views. Early 2010,

Table 1. Summary of the data used for the experiments

	Users	Places/Items	Edges in SN
Tuenti	10M	100K	700M
Epinions	50K	140K	500K

a feature was added to the Tuenti web platform whereby users could tell their friends where they were, and which places they particularly enjoyed. These places where added to the users profile.

The Data. To test *SECoFi*, we used the *Tuenti* place-user interaction matrix, as the matrix Y, that contains all the places the users have added to their profile. We also used the social network $\mathcal{F}$, i.e. the friendship matrix of *Tuenti* users. The data contains about 10 million users and approximately 100,000 places. Both of the matrices are very sparse, as each user has on average 4 places in his profile and 60 friends. The social graph among the *Tuenti* users contains approximately 700M edges that is each user has on average 70 friends. Note that this is an industry-scale dataset where the user/places graph takes up 2GB of storage space and the social graph data 22GB.

The *Epinions* data contains about 50k users and approximately 140,000 articles. Here users form a social graph (500k edges) based on the trust they show on each others reviews/ratings. Unlike the *Tuenti* data, the *Epinions* data is in the form of ratings with values between 1 and 5. We replace the rating values by 1 to convert the data to implicit feedback (just like in the KDD cup challenge 2007 *who-rated-what?*).

In contrast to the *Tuenti* data the relationships of the users are much more well defined in the *Epinions* data in that they reflect trust in another users opinion. Social relationships as the ones in the *Tuenti* data capture a much wider range of relationships between users e.g. family relationships, neighbours, classmates etc. which might not always translate into trust/influence.

Evaluation Protocol. For the evaluation procedure we adopt a similar strategy to [20]. We split the dataset into two parts, a training set to learn our model and a test set for evaluation. The test set contains the last 25% of places or items added to each users profile, and the training set contains all the remaining places/items that where added in the users profile. For each user we draw randomly some unobserved entries $Y_{ij} = 0$ assuming that these places/items are irrelevant to the user. We use these randomly chosen unobserved entries for training some of the methods in comparison (see Section 4). We used this protocol for both datasets.

We trained the model to compute a score F_{ij} for each user i place j in the test set along with the randomly drawn irrelevant items and rank the items for each user according to their scores. In recommendation algorithms we ultimately care about the ranking of the items, we thus use ranking metrics for the evaluation.

A popular list-wise ranking measure for this type of data is the Mean Average Precision metric (MAP) which is particularly well suited to recommendations ranking since it puts an emphasis in getting the first items in the ranked list right. MAP can be written as in equation 8.

$$\text{MAP} = \frac{1}{|\mathcal{U}|} \sum_{i=1}^{|\mathcal{U}|} \frac{\sum_{k=1}^{|\mathcal{M}|} P(k) Y_{ik}}{|\mathcal{Y}_i|} \tag{8}$$

Where $P(k)$ is the precision at the cut-off k. We also compute the RANK metric described in [8] to evaluate the performance of the different models. In contrast to the MAP metric, here smaller values indicate better performance. As we have no rating data, the RANK metric can be written, as follows:

$$\text{RANK} = \frac{\sum_{i,j} Y_{ij} rank_{ij}}{|\mathcal{Y}|} \tag{9}$$

Where $rank_{ij}$ is the percentile-ranking of the item j for a given user i.

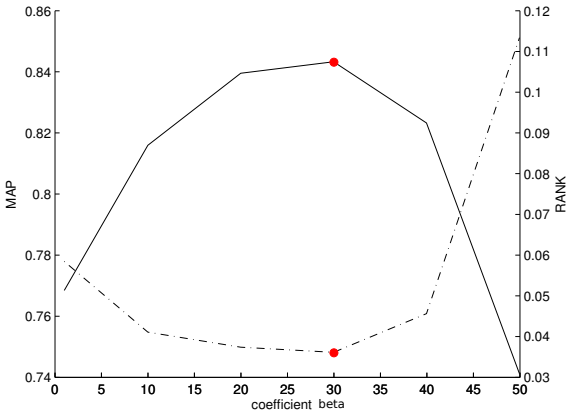


Fig. 1. The MAP and RANK metrics with respect to the value of the coefficient β

Methods in Comparison. The first method we compare against is a matrix factorization method based on alternating least squares optimization described in [8]. This method is tailored to implicit feedback data, but does not take the social graph into account. We can gauge based on the comparison with this method how much the use of the social data improves the recommendation performance. In the remaining we denote this method as *iMF*.

The second method we compare against is of [12], which takes advantage of the social graph along with contextual information to perform their recommendation. The resulting model is used to predict both items and friends for a given user. As the focus here is on the social aspect we do not use any contextual information but only the social graph. Thus adapting their objective function to our evaluation environment we end up optimizing:

$$\min_{U,M} \sum_{(i,j)\in\mathcal{Y}} L(U_i M_j, Y_{ij}) + \sum_{i,i'\in\mathcal{F}_i} L(U_i U_{i'}^T, S_{ij}) + \Omega_{U,M} \tag{10}$$

Where S represents the social graph (in which $S_{ii'}$ is 1 if the users i and i' are friends, 0 otherwise), and where L and Ω are respectively the loss function and the regularizer. The method was tested with several different loss function, we picked the one that gave the best results, the logistic loss function and used a simple $l2$-norm for the regularization term. Following [12] a stochastic gradient descent algorithm was used to optimize this objective. For the rest of the experiment section, we will denote this method as *LLA*.

The third method we compare against was introduced in [9] and takes the social data into account by penalizing the $l2$ distance between friends in the objective function. Two ways are proposed to penalize the distance between friends, we choose the one that gave them the best performance, i.e the one denoted individual-based regularization. The objective function minimized in [9] is the following:

$$\min_{U,M} \sum_{(i,j)\in\mathcal{Y}} (U_i M_j - Y_{ij})^2 + \sum_{i,i'\in\mathcal{F}_i} \mathrm{sim}(i,i') \|U_i - U_{i'}\|_F^2 + \lambda_1 \|U\|^2 + \lambda_2 \|M\|^2$$

Where $\mathrm{sim}(i,i')$ is a similarity score between a user i and a user i'. This similarity can be computed using vector space similarity or a Pearson correlation coefficient. Also here a stochastic gradient descent algorithm is used to optimize the objective function. In the remainder of this section we denote this method as *RSR*.

The last method we compare against is the one described in [16]. The focus of that work is on explicit feedback (ratings) and the social trust matrix A is precomputed. The model is then trained by optimizing a simple loss among the factors U and V, using a user-item rating dataset. We fit their method to the implicit feedback problem by, precomputing and fixing the matrix A at the beginning, and using the ponderation trick on the objective (with the use of the coefficients c_{ij}) to make implicit feedback learning possible. We will denote this method as *Trust Ensemble*. We also compare *SECoFi* to a baseline : the average predictor, which will recommend the overall most popular places to each user.

Computational Complexity. We first validate the efficiency of *SECoFi* by measuring the time needed to execute one iteration of the *SECoFi* method using a varying portion of the training data. We expect the method to show linear scalability in terms of the users and the observed entries in the user/item dataset.

To this end we use the *Epinions* data and run one iteration of the algorithm for each random data split. Those tests has been performed using a single Intel i5 core. The resulting timing information is displayed in Figure 2. Note the linear growth in the running-time of the method given the different data splits.

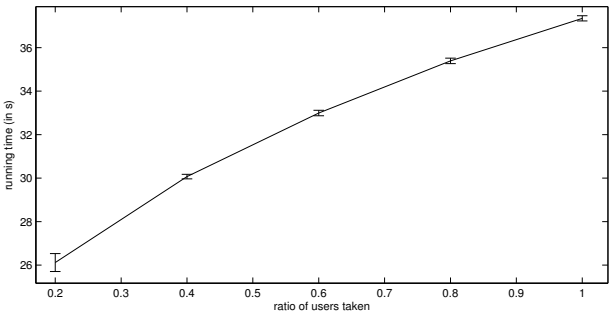


Fig. 2. The running time of one iteration of *SECoFi* given different random data splits 20%, 40% etc. over the *Epinions data*

Results on the *Tuenti* Data. We performed cross-validation for model selection. We randomly initialize U and M drawing from a uniform distribution between 0 and 1. For the initialization of the friendship weight α_{ij}, we found out empirically that the best performance is achieved by initializing with $\alpha_{ij} = 1$. We also estimate that the optimal value of the parameter β (used in the coefficient c_{ij}) is $\beta = 30$, according to the MAP and the RANK metrics (see Figure 1). We used this value of β for all the experiments.

We validate the performance of *SECoFi* also over a range of values of the number of factors d parameter (1, 5, 10, 15 and 20) on *Tuenti*. We repeated the experiments several (10) times for each method and report the mean values of the runs along with the standard deviations. We run experiments for different values of d for each method, results are shown in Figure 3.

We observe that even for a small number of factors, *SECoFi* outperforms the alternative social *LLA* and *RSR* enabled methods both in terms of MAP and RANK (over 17% improvement for the MAP and over 14% for the RANK). Moreover *SECoFi* is significantly better than *iMF* in terms of MAP, and for higher values of d our method becomes statistically equivalent to *iMF* in terms of RANK. Note that for recommendations where only a small number of items k is shown to the user the importance of MAP is bigger then RANK since MAP is a top-biased evaluation measure, i.e. placing items at the top of the list is more important then lowering the overall ranking of the all the items. *SECoFi* clearly outperforms in terms of MAP and RANK the *Trust Ensemble* method. Surprisingly *iMF* seems to outperform the alternative socially-enabled LLA and RSR methods in the comparison. One of the reason for this might be the strong

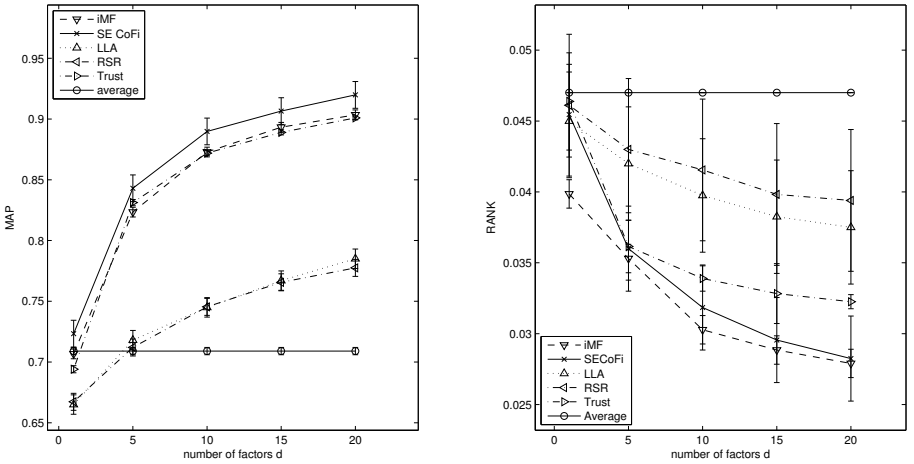


Fig. 3. The MAP and RANK metrics of the various methods on the *Tuenti* data depending on the number of factors d

sparsity of the data which bodes well with methods that take all the non-observed entries into account.

The relative performance between the methods does not depend strongly on the number of latent variables used. Except for *Trust Ensemble* method which was statistically equivalent to our method for small dimension, but we clearly see the difference for bigger dimensions. Indeed, *SECoFi* outperforms *Trust Ensemble* as well in terms of MAP as of RANK for a number of factors $d \geq 10$. *SECoFi* outperforms the other methods for all the values of d we tested with.

We thus confirm that the relative performance of our model does not depend on d for most of the alternative methods, we also observe that the relative performance *SECoFi* method with regards to *Trust Ensemble* is enhanced with higher numbers of factors. We also observe that the optimal regularization parameters for *SECoFi* were always the same, independent of the value of d. This eases the model selection process particularly compared to SGD based methods where both a learning rate and a regularizer need to be tuned. Moreover it seems that methods that are based on alternated least-square (ALS) optimization perform better predictions than those that use SGD. Note that the SGD-based methods subsample the unobserved entries to avoid biasing the estimator.

Experiments on the *Epinions* Data. We repeat the experimental evaluation of *SECoFi* on the publicly available *Epinions*[2] dataset. We follow the same procedure as described for the *Tuenti* data, the experiment results for the different methods on the *Epinions* data are shown in Figure 4.

[2] `http://snap.stanford.edu/data/soc-Epinions1.html`

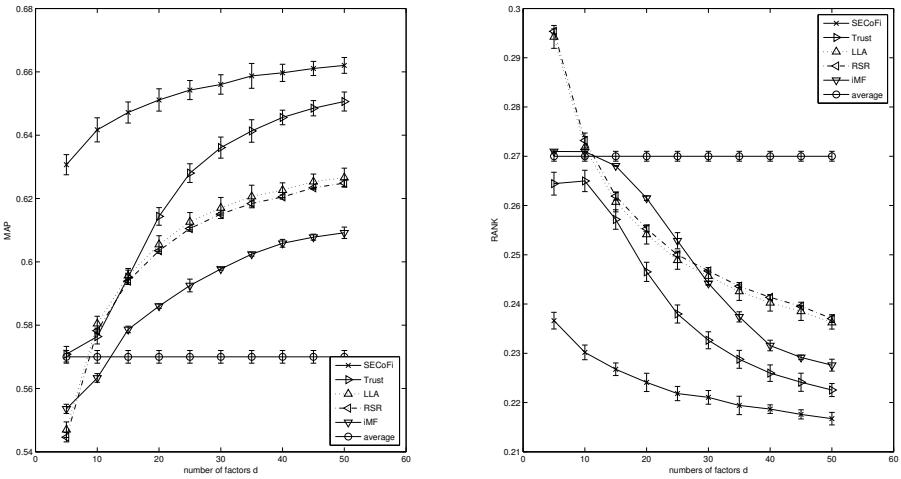
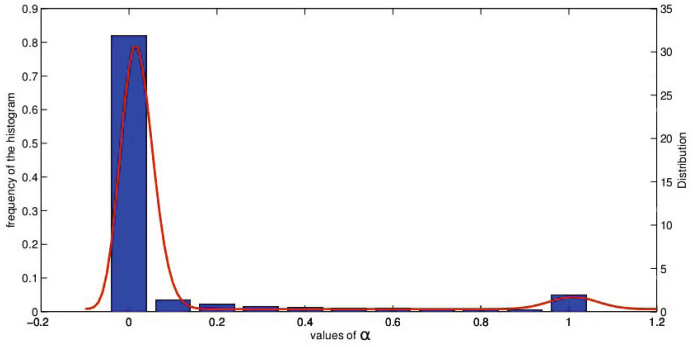


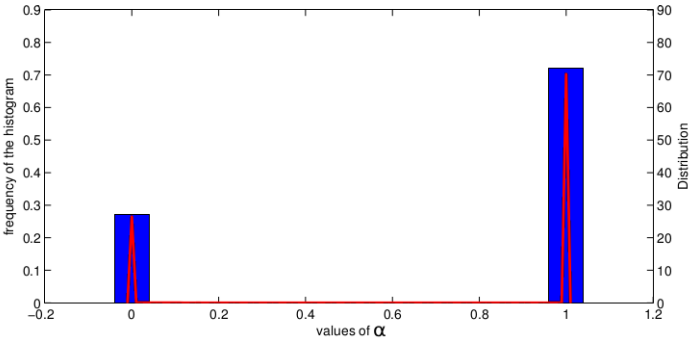
Fig. 4. The MAP and RANK metrics of *SECoFi* and the remaining methods on the *Epinions* dataset

From the results we can draw similar conclusions to the experiments with the *Tuenti* data: learning the friendship weights A during the optimization process significantly improves the performance over methods that just use the social network information as proposed by [16] without quantifying these relationships. Note that *SECoFi* outperforms the second best method *Trust Ensemble* by 2.4% in terms of MAP and by 4.1% in terms of RANK, while *SECoFi* outperforms the remaining methods in comparison by more then 6% both in terms of MAP and RANK. We observe that ALS based methods that take all the "unobserved entries" of the data into account perform better then SGD-based approaches that sample the space of "unobserved entries". Moreover *SECoFi* performs relatively well even utilizing a smaller numbers of factors d. This can be particularly useful in recommendation engines that need to be compact in terms of memory usage e.g. on a smartphone.

In Figure 5 we plot the distribution of the values of α for the two datasets. Recall that the values of alpha encode the degree of influence or trust among users. We observe that for both of the datasets there is a bimodal distribution. For the *Epinions* dataset, most of the values are between 0 and 1 (99%) and 70% of the values are around 1, signalling strong trust relationships among users. For Tuenti, fewer values of α are around 1, and most of the values are close to 0. While there is still some significant influence/trust among users it is less prevalent than in the *Epinions* dataset. This reflects the nature of the data: in the *Epinions* dataset the social network of the users is based on the trust that the users put on each others opinions/ratings while the social relationships on the *Tuenti* network are of much broader scope and can range from close friendships to simple acquaintances, thus we also expect that a smaller fraction of these



(a)



(b)

Fig. 5. Distribution of the values of α on *Tuenti* (a) and *Epinions* (b) datasets. The distributions are bimodal.

relationships will reflect trust/influence. Note also that *SECoFi* outperforms the competing methods to a higher degree on the *Epinions* data, another indication that the social information in this dataset provides more information on the preferences of the users.

Another important point is that *SECoFi* depends less on the "quality" of the users Social Network. In fact *iMF*, which does not utilize OSN information, is the best runner up in the experiments on the *Tuenti* dataset. This can be attributed to the more relaxed definition of friends in a general purpose social network such as *Tuenti* where we can expect that not all friends share the same taste and preferences with the user. Alternative approches relying on a non-adaptive contribution of friends (*RSR*, *LLA*) suffer more in this context, while learning the weights α helps *SECoFI* to keep only the useful part of the users social network with respect to the recommendations.

5 Conclusions

We presented a method minimizing a novel objective function that takes advantage of the social graph data to perform personalized item recommendation on implicit feedback data. *SECoFi* outperforms alternative state-of-the-art methods in terms of ranking measures and also provides the added benefit of quantifying the influence/trust relationships among users. The latter can be particularly helpful when providing group recommendations e.g. when inviting a group of users to an event etc. We can moreover use the computed α_{ij} values to perform friend recommendation, by leveraging the fact that these values represent a measure of shared interest and taste among users which quantifies the "homophily" effect.

Acknowledgements. This work is funded as part of a Marie Curie Intra European Fellowship for Career Development (IEF) awards held by Alexandros Karatzoglou (CARS, PIEF-GA-2010-273739).

References

1. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS) 22(1), 143–177 (2004)
2. Basilico, J., Hofmann, T.: Unifying collaborative and content-based filtering. In: Proc. Intl. Conf. Machine Learning, pp. 65–72. ACM Press, New York (2004)
3. Srebro, N., Rennie, J., Jaakkola, T.: Maximum-margin matrix factorization. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 17. MIT Press, Cambridge (2005)
4. Takacs, G., Pilaszy, I., Nemeth, B., Tikk, D.: Scalable collaborative filtering approaches for large recommender systems. Journal of Machine Learning Research 10, 623–656 (2009)
5. Weimer, M., Karatzoglou, A., Le, Q., Smola, A.: Cofirank - maximum margin matrix factorization for collaborative ranking. In: Advances in Neural Information Processing Systems 20 (NIPS). MIT Press, Cambridge (2008)
6. Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using markov chain monte carlo. In: Proceedings of the 25th International Conference on Machine learning, ICML 2008, pp. 880–887. ACM, New York (2008)
7. Srebro, N., Shraibman, A.: Rank, trace-norm and max-norm. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 545–560. Springer, Heidelberg (2005)
8. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Proc. of ICDM 2008, pp. 263–272. IEEE Computer Society, Washington, DC (2008)
9. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM 2011, pp. 287–296. ACM, New York (2011)
10. 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, RecSys 2010, pp. 135–142. ACM, New York (2010)

11. Li, W.-J., Yeung, D.-Y.: Relation regularized matrix factorization. In: Proceedings of the 21st International Joint Conference on Artifical Intelligence, IJCAI 2009, pp. 1126–1131. Morgan Kaufmann Publishers Inc., San Francisco (2009)
12. Yang, S.-H., Long, B., Smola, A., Sadagopan, N., Zheng, Z., Zha, H.: Like like alike: joint friendship and interest propagation in social networks. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, pp. 537–546. ACM, New York (2011)
13. Cui, P., Wang, F., Liu, S., Ou, M., Yang, S., Sun, L.: Who should share what?: item-level social influence prediction for users and posts ranking. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, pp. 185–194. ACM, New York (2011)
14. Agarwal, D., Chen, B.-C.: Regression-based latent factor models. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 19–28. ACM, New York (2009)
15. Noel, J., Sanner, S., Tran, K.-N., Christen, P., Xie, L., Bonilla, E.V., Abbasnejad, E., Penna, N.D.: New objective functions for social collaborative filtering. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 859–868. ACM, New York (2012)
16. Ma, H., King, I., Lyu, M.R.: Learning to recommend with social trust ensemble. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, pp. 203–210. ACM, New York (2009)
17. Pan, W., Aharony, N., Pentland, A.: Composite social network for predicting mobile apps installation. In: Proceedings of the 25th Conference on Artificial Intelligence (AAAI 2011), San Francisco, CA (2011)
18. Zheng, Y., Zhang, L., Ma, Z., Xie, X., Ma, W.Y.: Recommending friends and locations based on individual location history. ACM Transactions on the Web (TWEB) 5(1), 5 (2011)
19. Scellato, S., Noulas, A., Mascolo, C.: Exploiting place features in link prediction on location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1046–1054. ACM (2011)
20. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proc. of the 4th ACM Conference on Recommender Systems (RecSys 2010), pp. 39–46 (2010)