A Social Formalism and Survey for Recommender Systems

Daniel Bernardes, Mamadou Diaby*, Raphaël Fournier, Françoise Fogelman-Soulié, Emmanuel Viennet

Université Paris 13, Sorbonne Paris Cité, L2TI, 93430, Villetaneuse, France.
* Work4 Labs, 3 Rue Moncey, 75009, Paris, France

{daniel.bernardes, mamadou.diaby, raphael.fournier, soulie, emmanuel.viennet}@univ-paris13.fr

ABSTRACT

This paper presents a general formalism for Recommender Systems based on Social Network Analysis. After introducing the classical categories of recommender systems, we present our Social Filtering formalism and show that it extends association rules, classical Collaborative Filtering and Social Recommendation, while providing additional possibilities. This allows us to survey the literature and illustrate the versatility of our approach on various publicly available datasets, comparing our results with the literature.

Keywords

Recommender systems; Social Network Analysis; Collaborative Filtering; Social Recommenders.

1. INTRODUCTION

Recommender Systems (RSs) help users or groups of users deal with information overload by proposing to them items suited to their interests. The history of RSs started in the late 1990s with work by the GroupLens team at University of Minnesota [18] to recommend news, and by MovieLens in 1996 to recommend movies, which demonstrated that automated recommendations were very well received by users. Then Amazon, which had been incorporated in 1994, published its patent in 2001 and has been serving recommendations ever since, acting as a *de facto* reference showcase for the efficiency of RSs [33]. The Netflix competition (2006-2009) attracted over 41,000 participating teams [6] and turned RS into a hot topic among researchers.

The first papers on collaborative filtering showed how to use the opinions of similar users to recommend items to the active user [1]. Since then, research in RSs has become very active (see for example a recent RS survey including more than 250 references [8]) and RSs have been successfully used in many industry sectors to recommend items: movies (Netflix [6], MovieLens [41]), products (Amazon.com [33], La Boîte à Outils [47]), songs [3], jobs to Facebook users (Work4Labs.com [16]), books, friends, banners or content on a social site (Skyrock.com [44]) etc.

RSs exploit various sources of information: about users (their demographics), about products (their features) and about user interactions with the products [8; 25], either *explicit* (rating, satisfaction) or *implicit* (product purchased, book read, song heard, content clicked etc.) More recently, Social networks and social media (blogs, social tagging sites, etc) have emerged and become very active. A social site will allow users to construct profiles (public or semi-public), to share connections with other users and to view and traverse lists of connections made by others in the system [10]. Authors [8; 51; 56; 57] have thus proposed to also include information from social media (Facebook, Twitter, ...) because social relations obviously influence users' behaviors. There are two visions for social recommendation [56]:

- The narrow definition only considers RSs which combine users' data and data from their social relationships. It is the one most used in the literature. In this case, we need an *explicit social network* such as, for example, Facebook, to provide the social relationships;
- In a broader definition, a social RS is any recommendation system targeting social media (blogs, social tagging, video sharing, etc.) or even ecommerce. In this case, we might not have an explicit social network, but could still derive an *implicit social network*.

RSs implementations are based on various techniques [1]: content-based, collaborative filtering (both memory- and model-based), hybrid or social [56]. Performances are evaluated through various criteria [52].

In this paper, we propose a Social Filtering formalism (SF), based on Social Network Analysis (SNA), which allows us to describe, within the same formalism, both association rules, traditional Collaborative Filtering (CF) and Social Recommendation (SR), while providing additional ways to implement a RS, thus producing novel RSs.

The paper can thus be read as a survey on RSs, with experiments illustrating the various RSs presented. We have not tried, in this paper, to optimize hyper-parameters, but rather intended to present a wide repertoire of RSs tested in a uniform setting to allow for comparisons which are usually hard to make since, in the literature, each paper has its own settings and hyper-parameters choices.

The paper is organized as follows: in section 2, we introduce general concepts and notations, and we review traditional techniques in section 3. In section 4, we introduce our Social Filtering formalism and show in section 5 how it relates to conventional approaches. In section 6, we introduce evaluation metrics and various representative datasets. In section 7, we present extensive experimental results to illustrate the various RSs presented in the paper: we reproduce known results from the literature, and add new results, showing the benefits brought by our unifying formalism. Our conclusion identifies remaining issues and perspectives in section 8.

2. NOTATIONS

RSs use available data to generate lists of recommendations. Depending on the application, data can involve:

- **Usage**: the user visits a site and performs various actions on items shown (clicks, insertion-into-cart, purchases) which are collected in the site log files. Such actions provide feedback on the user's interest in a given item, which are essentially positive (the fact that the user did not act on the item at that time does not necessarily mean he did not like it). This data is often called *implicit* because the user did not generate it intentionally, but it comes as a side effect of the user's actions.
- **Ratings**: the user posts his evaluations or "likes" of items displayed on the site. These ratings can appear as stars, numbers or even comments. They can be positive or negative. Most users will not leave any evaluation and those who do might widely differ in their rating ways. Ratings are said to be *explicit* data, since the user intentionally acted to provide them.
- **Additional data**: in most cases, many more data sources exist on items and users.
  - Items: items such as products, contents or banners usually have associated attributes, such as for example the title and author of a book. Obviously these attributes are important to understand whether a user could be interested in the item.
  - Users: various types of information might be available, from a mere IP address to cookies to detailed attributes of registered customers (name, address, history of purchases, etc). Knowing the user in details should help building personalized recommendations for him/her. Additionally, information from social media (friends, followers, etc) might also be available.
  - Data from the RS: the RS itself produces ordered lists of items recommended to the user who might like them later (clicks, purchases, etc. will indicate that).

Usage and rating data can be represented by an *interaction* (or *preferences*) matrix R which encodes the actions of users on items. Table 1 below, for example, shows 4 users who rated 5 movies (ratings are shown by numbers). In the case where the user's interaction is a purchase or a click, matrix R is binary, with 1 indicating the item was purchased and 0 it was not. In cases where repeated consumption is possible, values in matrix R indicate the number of times the item was consumed, thus leading to R having the same structure as for ratings. In the following, we will use the word *consume* to indifferently mean rate, purchase or click.

	Monuments Men	**Django Unchained**	**Forrest Gump**	**Gran Torino**	**Pulp Fiction**
Amy	3			2	5
Paul		4			2
Rob	5	4	1		
Liz	2		3		

Table 1: Interaction matrix R in the case of ratings.

It should be noted that collecting implicit user's behavior is usually easier than requiring the user to provide explicit feedback or provide access to his / her social network. Most users purchase but very few items and rate even less: as few as 1% of users who consume an item might also rate it. As a result, matrix R is often very sparse, even more so in the case of ratings.

We will denote by L (resp. C) the total number of users or lines in R (resp. total number of items or columns). Matrix R is thus of dimensions L x C. Usually, matrix R is very large: a few millions $\times$ a few tens-hundreds of thousands.

3. STATE-OF-THE-ART

RSs have been studied for more than 15 years now [1; 8]. Traditional techniques are grouped into content-based, Collaborative Filtering, hybrid, and, more recently, social [8], which we describe in this section.

3.1 Content-based

Content-based RS [1; 35; 46] use items (or users) descriptions to define items' (or users) profiles. A user is then recommended items which profiles best match the items he best rated in the past, or items which users with most similar profiles best rated in the past. Sometimes, users only provide descriptions (instead of ratings), in this case items' profiles are constructed using these descriptions [16]. To implement Content-based RSs, we need a similarity measure (among items or users) and profiles comparison (for example k-nearest neighbors).

3.2 Collaborative Filtering

Collaborative Filtering is certainly the most widely used technique for implementing RSs. There exist two main groups of CF techniques: memory-based (or neighborhood methods [1]), and model-based (or latent factor models [42]).

As stated earlier in the introduction, CF methods use the opinion of a group of similar users to recommend items to the active user [1; 27; 32; 42; 54]. According to [42], the two key assumptions behind these systems are:

- Users who had similar tastes in the past will have similar tastes in the future;
- Users' preferences remain stable and consistent over time.

In the literature, there exist two main groups of CF techniques [59]: model-based or latent factor models [42; 34; 2] and memory-based or neighborhood methods [1; 32].

3.2.1 *Model-based methods*

Model-based RSs [11; 14; 29; 54; 60; 61] estimate a global model, through machine learning techniques, to produce unknown ratings. This leads to models that neatly fit data and therefore to RSs with good quality. However, learning a model may require lots of training data which could be an issue in some applications. In the literature many model-based CF systems have been proposed:

- Reference [11] proposes a probabilistic model in which ratings are integer valued; the model is then learnt using Bayesian networks;
- Reference [61] designs a CF system based on support vector machine (SVM) by iteratively estimating all missing ratings using an heuristic;
- Reference [29] develops a neural network-based collaborative method: a user-based and an item-based methods;
- Reference [28] describes various methods used for the Netflix prize[1] and in particular the matrix factorization methods which provided the best results.

One of the most efficient and best used model-based methods is matrix factorization [42; 59] in which users and items are represented in a low-dimensional latent factors space. The new representations of users ($\hat{U}$) and items ($\hat{I}$) are commonly computed by minimizing the regularized squared error [59]:

$$\min_{\hat{U},\hat{I}} \sum_{u,i} \left[\left(r_{u,i} - \hat{v}_u^T \hat{v}_i \right)^2 + \lambda_1 \parallel \hat{v}_u \parallel^2 + \lambda_2 \parallel \hat{v}_i \parallel^2 \right] \quad (1)$$

where λ_1 and λ_2 are regularization parameters, $r_{u,i}$ is the rating that user u gave to item i, $\hat{v}_u$ and $\hat{v}_i$ are the new representations of user u and item i respectively, $\hat{U}$ and $\hat{I}$ are the new representation sets of the sets of users and items, respectively. Once the new representations of users and items $\hat{v}_u$ and $\hat{v}_i$ have been computed, we can obtain the predicted rating $\hat{r}_{u,i}$ as follows:

$$\hat{r}_{u,i} = \hat{v}_u^T \hat{v}_i \quad (2)$$

Matrix factorization methods can be generalized to probabilistic models called Probabilistic Matrix Factorization [42; 50; 59]. These techniques are more suited to explicit feedback cases. They usually produce very good results but suffer from extremely large sizes of matrix R [3].

3.2.2 *Memory-based methods*

Memory-based CF techniques rely on the notion of similarity between users or items to build neighborhoods methods.

3.2.2.1 *Similarity.*

If a is the active user for whom we seek recommendations, u another user and i and j two items, we will denote:

- $I(a)$, $I(u)$ and $I(a\&u) = I(a) \cap I(u)$ the sets of items consumed by a, u, both a and u respectively.
- $U(i)$, $U(j)$ and $U(i\&j) = U(i) \cap U(j)$ the set of users who consumed i, j and both i and j, respectively.
- $\overrightarrow{l}(u)$ the line of matrix R for user u and $\overrightarrow{c}(i)$ its column for item i,
- $\bar{l}(u)$ the average of $\overrightarrow{l}(u)$ (average rating given by u or average number of items consumed by u) and $\bar{c}(i)$ the average of $\overrightarrow{c}(i)$ (i's average rating or average number of users who consumed i).

$$\bar{l}(u) = \frac{1}{C}\sum_{i=1}^{C} r_{ui} \qquad \bar{c}(i) = \frac{1}{L}\sum_{u=1}^{L} r_{ui}$$

The similarity between users a and u can be defined through many similarity measures, for example cosine, Pearson correlation coefficient (PCC) [1] or asymmetric cosine [3] similarities (equations (3), (4) & (5) below respectively):

$$\begin{aligned}\text{Sim}(a,u) &= \cos\left[\overrightarrow{l}(a), \overrightarrow{l}(u)\right] \\ &= \frac{\sum_{i=1}^{C} r_{ai} \times r_{ui}}{\sqrt{\sum_{i=1}^{C}(r_{ai})^2}\sqrt{\sum_{i=1}^{C}(r_{ui})^2}}\end{aligned} \quad (3)$$

$$\begin{aligned}\text{Sim}(a,u) &= \text{PCC}\left[\overrightarrow{l}(a), \overrightarrow{l}(u)\right] \\ &= \frac{\sum_{i\in I(a)\cap I(u)} \left[r_{ai} - \bar{l}(a)\right]\left[r_{ui} - \bar{l}(u)\right]}{\sqrt{\sum_{i\in I(a)\cap I(u)} \left[r_{ai} - \bar{l}(a)\right]^2}\sqrt{\sum_{i\in I(a)\cap I(u)} \left[r_{ui} - \bar{l}(u)\right]^2}}\end{aligned} \quad (4)$$

$$\begin{aligned}\text{Sim}(a,u) &= \text{asym-cos}_\alpha\left[\overrightarrow{l}(a), \overrightarrow{l}(u)\right] \\ &= \frac{\sum_{i=1}^{C} r_{ai} \times r_{ui}}{\left[\sum_{i=1}^{C} r_{ai}^2\right]^\alpha \times \left[\sum_{i=1}^{C} r_{ui}^2\right]^{1-\alpha}}\end{aligned} \quad (5)$$

Note that, in the binary case, asymmetric cosine for $\alpha = \frac{1}{2}$ is equivalent to cosine similarity. The similarity between items i and j can be defined in the same fashion: cosine, Pearson correlation coefficient or asymmetric cosine (equations (6), (7) & (8) below respectively):

$$\begin{aligned}\text{Sim}(i,j) &= \cos\left[\overrightarrow{c}(i), \overrightarrow{c}(j)\right] \\ &= \frac{\sum_{u=1}^{L} r_{ui} \times r_{uj}}{\sqrt{\sum_{u=1}^{L}(r_{ui})^2}\sqrt{\sum_{u=1}^{L}(r_{uj})^2}}\end{aligned} \quad (6)$$

[1]http://www.netflixprize.com

$$\begin{aligned}\mathrm{Sim}(i,j) &= \mathrm{PCC}\Big[\overrightarrow{c}(i), \overrightarrow{c}(j)\Big] \\ &= \frac{\sum\limits_{u\in U(i)\cap U(j)} \big[r_{ui}-\bar{c}(i)\big]\big[r_{uj}-\bar{c}(j)\big]}{\sqrt{\sum\limits_{u\in U(i)\cap U(j)} \big[r_{ui}-\bar{c}(i)\big]^2}\sqrt{\sum\limits_{u\in U(i)\cap U(j)} \big[r_{uj}-\bar{c}(j)\big]^2}}\end{aligned} \quad (7)$$

$$\begin{aligned}\mathrm{Sim}(i,j) &= \text{asym-cos}_\alpha\Big[\overrightarrow{c}(i), \overrightarrow{c}(j)\Big] \\ &= \frac{\sum\limits_{u=1}^{L} r_{ui}\times r_{uj}}{\Big[\sum\limits_{u=1}^{L} r_{ui}^2\Big]^\alpha \times \Big[\sum\limits_{u=1}^{L} r_{uj}^2\Big]^{1-\alpha}}\end{aligned} \quad (8)$$

It should be noted that both users and items similarity measures above take into account the actions on all items (resp. of all users): this is why these measures are called *collaborative.*

3.2.2.2 *Collaborative Filtering scores.*

CF techniques produce, for an active user a, a list of recommended items ranked through a *scoring function* (or aggregation function), which takes into account either users most similar to a (user-based CF) or items most similar to those consumed by a (item-based CF).

Let us thus denote $K(a)$ the *neighborhood* of a and $V(i)$ the neighborhood of item i. These neighborhoods can be defined in many ways (for example, N-nearest neighbors, for some given N, or neighbors with similarity larger than a given threshold, using user/item similarity).

The score functions are then defined for users and items as:

$$\begin{aligned}\mathrm{Score}(a,i) &= \sum_{u\in K(a)} r_{ui}\times f\big[\mathrm{Sim}(a,u)\big] \\ \mathrm{Score}(a,i) &= \sum_{j\in V(i)} r_{aj}\times g\big[\mathrm{Sim}(i,j)\big]\end{aligned} \quad (9)$$

where various functions f and g can be used [1]:

- For user-based CF: average rating (or popularity) of item i by neighbors of a in $K(a)$, weighted average rating and normalized average rating of nearest users weighted by similarity to a (from top to bottom in equation (10) below):

$$\begin{aligned}\mathrm{Score}(a,i) &= \frac{1}{\mathrm{card}\big[K(a)\big]}\sum_{u\in K(a)} r_{ui} \\ \mathrm{Score}(a,i) &= \frac{\sum\limits_{u\in K(a)} r_{ui}\times \mathrm{Sim}(a,u)}{\sum\limits_{u\in K(a)\cap U(i)} |\mathrm{Sim}(a,u)|} \\ \mathrm{Score}(a,i) &= \bar{l}(a) + \frac{\sum\limits_{u\in K(a)\cap U(i)} \big(r_{ui}-\bar{l}(u)\big)\times \mathrm{Sim}(a,u)}{\sum\limits_{u\in K(a)\cap U(i)} |\mathrm{Sim}(a,u)|}\end{aligned} \quad (10)$$

- For item-based CF: average rating by a of items neighbors of i in $V(i)$, weighted average rating, normalized average rating weighted by their similarity with i:

$$\begin{aligned}\mathrm{Score}(a,i) &= \frac{1}{\mathrm{card}\big[V(i)\big]}\sum_{j\in V(i)} r_{aj} \\ \mathrm{Score}(a,i) &= \frac{\sum\limits_{j\in V(i)} r_{aj}\times \mathrm{Sim}(i,j)}{\sum\limits_{j\in V(i)\cap I(a)} |\mathrm{Sim}(i,j)|} \\ \mathrm{Score}(a,i) &= \bar{c}(i) + \frac{\sum\limits_{j\in V(i)\cap I(a)} \big(r_{aj}-\bar{c}(j)\big)\times \mathrm{Sim}(i,j)}{\sum\limits_{j\in V(i)\cap I(a)} |\mathrm{Sim}(i,j)|}\end{aligned} \quad (11)$$

Another mechanism has been developed [3] to produce locality instead of explicitly defining neighborhoods. Functions f and g are defined so as to put more emphasis on high similarities (with high q, q'):

$$\begin{aligned}\mathrm{Score}(a,i) &= \sum_{u\in K(a)} r_{ui}\times \big[\mathrm{Sim}(a,u)\big]^q \\ \mathrm{Score}(a,i) &= \sum_{j\in V(i)} r_{aj}\times \big[\mathrm{Sim}(i,j)\big]^{q'}\end{aligned} \quad (12)$$

For $q = 0$, this is equivalent to average rating, and for $q = 1$, this is similar to weighted average rating.
We then rank items i by decreasing scores and retain the top k items $(i_1^a, i_2^a, ..., i_k^a)$ which are recommended to a, such that:

$$\mathrm{Score}(a, i_1^a) \geq \mathrm{Score}(a, i_2^a) \geq ... \geq \mathrm{Score}(a, i_k^a) \quad (13)$$

3.2.2.3 *Conclusion on CF techniques.*

Notice that while memory-based techniques produce ranked lists of items, model-based techniques predict ratings, through a score which can be used also to rank recommendations. In practice, all CF systems suffer from several drawbacks:

- New user/item: collaborative systems cannot make accurate recommendation to new users since they have not rated a sufficient number of items to determine their preferences. The same problem arises for new items, which have not obtained enough ratings from users. This problem is known as the *cold start* recommendation problem;

- Scalability: memory-based systems generally have a scalability issue, because they need to calculate the similarity between all pairs of users (resp. items) to make recommendations;

- Sparsity: the number of available ratings is usually extremely small compared to the total number of pairs user - item; as a result the computed similarities between users and items are not stable (adding a few new ratings can dramatically change similarities) and so predicted ratings are not stable either;

- Information: of course memory-based and model-based techniques use very limited information, namely ratings/purchases only. They could not use content on users or items, nor social relationships if these were available.

The representation of users and items in a low dimensional space in latent factor models mitigates the cold start recommendation problem but raises a scalability issue. In general, latent factor methods are known to generally yield better results than neighborhood methods [28; 42; 59].

3.3 Social Recommender Systems

Traditional RSs, and in particular model-based systems, rely on the (often implicit) assumption that users are independent, identically distributed (i.i.d). The same holds for items. However, this is not the case on social networks where users enjoy rich relationships with other members on the network. It has long been observed in sociology [40] that users' "friends" on such networks have similar taste (homophily). It is thus natural that new techniques [65] extended previous RSs by making use of social network structures. However, it was realized that the type of interaction taken into account could have a dramatic impact on the quality of the obtained social recommender [65]. In this section, we review three families of social recommender: one based on explicit social links, one based on trust and an emerging family based on implicit links.

3.3.1 Social Recommender Systems based on explicit social links

In this section, we assume that users are connected through explicit relationships such as friend, follower etc. Unsurprisingly, with the recent thrive of online social networks, it has been found that users prefer recommendations made by their friends than those provided by online RSs, which use anonymous people similar to them [53]. Most Social RSs are based on CF methods: social collaborative recommenders, like traditional CF systems, can be divided into two families: memory-based and model-based systems.

3.3.1.1 Memory-based Social Recommender.

Memory-based methods in social recommendation are similar to those in CF (presented in section 3.2.2), the only difference being the use of explicit social relationship for computing similarities.

- In [64; 65] the authors present their social-network-based CF system (SNCF), a modified version of the traditional user-based CF and test it on Essembly.com[2] which provides two sorts of links: friends and allies.
  - In [64], they use a graph theoretic approach to compute users' similarity as the minimal distance between two nodes (using Dijkstra's algorithm for instance), instead of using the ratings' patterns as in traditional CF; it is assumed that the influence will exponentially decay as distance increases. They show that this method produces results worse than traditional CF;
  - In [65], the user's neighborhood is just simply its set of friends in the network (first circle). This approach provides results slightly worse than the best CF. But the computation load is much reduced: from computing the similarity of all pairs of users to just looking for the user's friends. This is a dramatic improvement to the scalability issues of CF. They also show that if the allies are used instead of friends, then the results are as good as CF, but at a much reduced computation cost.
- In [24], authors observe on a dataset from Yelp[3] that friends tend to give restaurant ratings (slightly) more similar than non-friends. However, immediate friends tend to differ in ratings by 0.88 (out of 5), which is rather similar to results in [65]. Their experimental setup compares their model-based algorithm (probabilistic), a Friends Average approach (which only averages the ratings of the immediate friend), a Weighted Friends (more weight is given to friends which are more similar according to cosine-similarity), a Naive Bayes approach and a traditional CF method. All methods which use the influences from friends achieve better results than CF in terms of prediction accuracy.
- Reference [12] presents SaND (Social Network and Discovery), a social recommendation system; SaND is an aggregation tool for information discovery and analysis over the social data gathered from IBM Lotus Connections' applications. For a given query, the proposed system combines the active user's score, scores from his connections and scores between terms and the query;
- Reference [51] proposes two social recommendation models: the first one is based on social contagion while the second is based on social influence. The authors define the social contagion model as a model to simulate how an opinion on certain items spreads through the social network;
- Reference [19] proposes a group recommendation system in which recommendations are made based on the strength of the social relationships in the active group. This strength is computed using the strengths of the social relationship between pairwise social links (scaled from 1 to 5 and based on daily contact frequency).

3.3.1.2 Model-based Social Recommenders.

Model-based methods in social recommenders represent users and items into a latent space vector (as described in section 3.2.1) making sure that users' latent vectors are close to those of their friends.

- Reference [4] combined matrix factorization and friendship links to make recommendations: the recommendation score for the active user is the sum of the scores of his friends.
- Reference [38] proposes algorithms which yield better results than non-negative matrix factorization [31], probabilistic matrix factorization [42] and a trust-aware recommendation method [37]. It presents two social RSs:
  - A matrix factorization making sure that the latent vector of a given user is close to the weighted average latent vectors of his friends;

[2] http://www.essembly.com

[3] http://www.yelp.com

– A matrix factorization minimizing the difference between a user's and his friends' latent vectors individually.

Finally, a few social RSs combine social and content-based techniques. For example, [17] proposes two ways to aggregate users' preferences with those of their friends: enrich users' profiles with those of their friends or aggregate users' recommendation scores with those of their social relationships.

3.3.2 *Trust and influence-based social Recommender Systems*

As explained in [38] "trust relationships" are different from "social relationships" in many respects. Trust-aware RSs are based on the assumption that users have taste similar to other users they trust, while in social RSs, some of the active user's friends may have totally different tastes from him [38]. This was also observed in [65], with the differences between friends and allies, which represents a case where trust is explicitly provided by users.

In everyday life, people may ask other people (friends, relatives, someone they trust) for a recommendation. If the person cannot provide sufficient information, she may indicate another person whom she knows which could, and so on. The notion of trust network arises naturally: one tend to have faith in the opinion of people trusted by the people he trusts himself, transitively. Conversely, the notion of social influence has long been used in marketing, relying on the assumption that users are likely to make choices similar to their role-models [49]. The notion of influence can be seen as close to that of trust: when providing a friend with a referral, a trusted user influences her friend. It has long been known that this "word-of-mouth effect" can be used commercially, such as for example in viral marketing.

Recently, it was attempted to incorporate trust or influence knowledge into RSs. Beyond the mere expected increase in efficiency, computing trust may also alleviate recurrent problems of traditional RSs, such as data sparsity, cold start or shilling attacks (fake profile injections) to bias recommendations.

3.3.2.1 *Trust computation.*

The trust relationship is directional, i.e. the fact that user u_1 trusts user u_2 at some level t does not necessarily mean that u_2 trusts u_1 at the same or another level. Trust can be represented by a binary value, 0 for "not trusted user" and 1 for "trusted user", or through more gradual scales [20; 21; 39] or even with a probabilistic approach [15; 48]. Some models include an explicit notion of distrust [21; 67], but most of them ignore it.

For RSs, trust is computed over an explicit social network to increase the information available to generate recommendations. There exist two cases in the literature: either trust is provided explicitly in a trust network, or it has to be inferred.

In an explicit trust network, we propagate and aggregate trust to infer long chains of trust [67; 20]. Trust computation also requires an aggregation strategy, to combine estimates obtained from different paths from one user to another. Several operators may be used like minimum, maximum, (un)weighted sum and average. Different strategies may also be applied: propagate trust first, then aggregate; or aggregate first, then propagate (the latter allowing easier distributed computation).

In non-explicit trust networks, trust has to be inferred. For example, in [45], the author defines a profile and item-level trust, based on correct previous recommendations.

3.3.2.2 *Trust-enhanced Recommender Systems.*

In explicit trust networks, users provide the system with trust statements for their peers, be it on a gradual scale (Moleskiing [5]), allies (Essembly [65]) or lists of trusted and non-trusted people (Epinions [39]). Then, for the recommendation to a specific user u, trust is estimated between u and the relevant users, in order to weigh the recommendation computation, either through trust-based weighted mean (rating from user u for item i is an ordinary weighted mean of the ratings of users which have evaluated i, where the weights are the trust estimates for these users) or Trust-based CF (as in classical CF methods, replacing similarity-based weights by trust-based weights obtained via propagation and aggregation strategies as described above).

3.3.3 *Social Recommender Systems based on implicit social links*

Recently, a new type of social RSs has been introduced which rely not upon an explicit social network (as in section *3.3.1*) but upon networks which can be derived from users' behaviors and have thus been named *implicit networks*. Users will be – implicitly – connected if, for example, they take pictures in the same locations [23], they attend the same events or click on the same ads [44]. The implicit users' social network can then be used, as in section *3.3.1*) to build recommendations.

For example, [57] extracts from cooking recipes bipartite graph (*recipes*, *ingredients*) and sets the weight of the link in the ingredients network as the point-wise mutual information of the ingredients, extremities of the link. Authors then apply a discriminative machine learning method (stochastic gradient boosting trees), using features extracted from the ingredients network, to predict recipe ratings and recommend recipes. Results show that the structural features extracted from the ingredient networks are most critical for performances.

Similar approaches have been developed for RSs where no ratings are provided, but only information on whether objects were collected (product purchased, banner clicked, movie watched, song listened to).

- Reference [66] uses a resource-allocation process in the object network to define the weight on the links and an aggregate score : they show that their technique is more efficient on the MovieLens dataset than traditional CF;

- Reference [44] shows an example to recommend ads banners: for an active user, the similarity among banners is measured by the number of users who clicked on both. Then traditional item-based CF is applied. Authors show a 20-fold increase in number of cumulated clicks with the social RS compared to the random selection of 5 recommendations.

3.3.4 Conclusion

Social RSs are still relatively new. There is a lot of active research in this area and it should be expected that new results will extend the field of traditional systems to incorporate social information of all sorts. In particular, the field of social recommenders built on implicit social networks seems particularly promising and we will now dig deeper in this direction to produce our Social Filtering formalism.

4. SOCIAL FILTERING

Our **Social Filtering formalism** (SF) is based upon a bipartite graph and its projections (see [22; 66] for a discussion of bipartite graphs). A *bipartite graph* is defined over a set of nodes separated into two non-overlapping subsets: for example, users and items, items and their features, etc. A link can only be established between nodes in different sets: a link connects a user to the items she has consumed. The bipartite network is then projected into two (unipartite) networks, one for each set of nodes: a Users' and an Items' networks. In the projection (see Figure 1), two nodes are connected if they had common neighbors in the bipartite graph. The link weight can be used to indicate the number of shared neighbors. For example, two users are linked if they have consumed at least one item in common (we usually impose a more stringent condition: at least K items). The projected networks can thus be viewed as the network of users consuming at least K same items (users having the same preferences) and the network of items consumed by at least K' same users (items liked by the same people).

Projected networks can then be used to define neighborhoods [55] or recommendation algorithms which perform better than conventional CF on the MovieLens dataset [66]. This generic formalism extends these early contributions: we are able to reproduce results from various classical approaches, and we also provide new approaches, allowing more flexibility and potential for improved performances, depending on the dataset.

In the SF formalism, as in traditional CF, we build recommendations by defining neighborhoods and scoring functions.

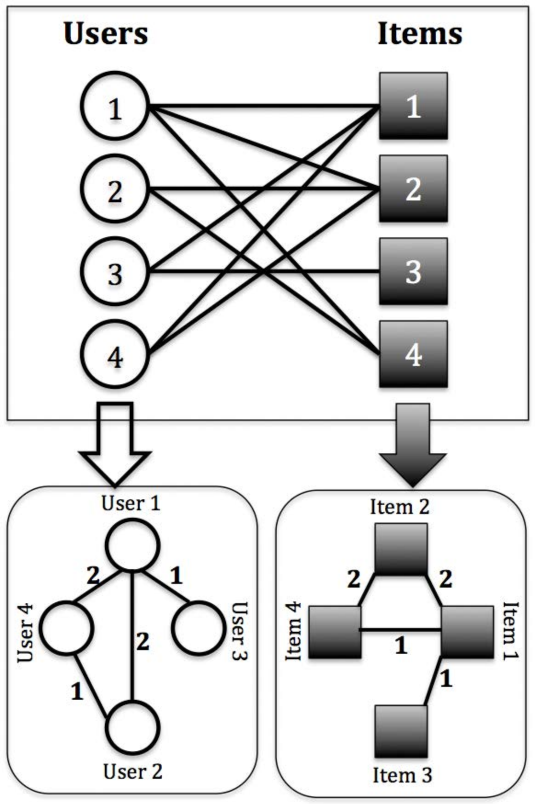

Figure 1: Bipartite graph and projections

4.1 Similarity

4.1.1 Support-based similarity

In the case of implicit feedback (binary interaction matrix R), in essence, the link between two users a and u (resp. i and j) represents an association rule $a \to u$ (resp. $i \to j$) with the link weight proportional to the rule support, where support of rule $a \to u$ (resp. $i \to j$) is defined as:

$$\text{Supp}(a \to u) = \frac{\#\text{ Items cons. by } a \text{ and } u}{\#\text{ Items}} = \frac{1}{C}\sum_{i=1}^{C} r_{ai}r_{ui}$$
$$\text{Supp}(i \to j) = \frac{\#\text{ Users who cons. } i \text{ and } j}{\#\text{ Users}} = \frac{1}{L}\sum_{u=1}^{L} r_{ui}r_{uj}$$

In the case of a non-binary matrix R (ratings), support is similarly defined. Hence, *Support* is defined in general as:

$$\begin{aligned} \text{Supp}(a \to u) &= \frac{1}{C}\sum_{i=1}^{C} r_{ai}r_{ui} \\ \text{Supp}(i \to j) &= \frac{1}{L}\sum_{u=1}^{L} r_{ui}r_{uj} \end{aligned} \tag{14}$$

Support is similar to cosine similarity (equations (3) and (6)), so that we can use support as a similarity measure. We will define support-based similarity of users (resp. items) as:

$$\begin{aligned} \text{Sim}(a, u) &= \text{Supp}(a \to u) \\ \text{Sim}(i, j) &= \text{Supp}(i \to j) \end{aligned} \tag{15}$$

4.1.2 Confidence-based similarity

In the case of implicit feedback, the confidence of link $a \to u$

(resp. $i \to j$) is defined, as for association rules, by:

$$\mathrm{Conf}(a \to u) = \frac{\text{\# Items cons. by } a \text{ and } u}{\text{\# Items cons. by } a} = \frac{\sum_{i=1}^{C} r_{ai} r_{ui}}{\sum_{i=1}^{C} r_{ai}}$$
$$\mathrm{Conf}(i \to j) = \frac{\text{\# Users who cons. } i \text{ and } j}{\text{\# Users who cons. } i} = \frac{\sum_{u=1}^{L} r_{ui} r_{uj}}{\sum_{u=1}^{L} r_{ui}}$$

For ratings (non-binary matrix R), confidence will be similarly defined. Hence, *Confidence* is defined in general as:

$$\begin{aligned} \mathrm{Conf}(a \to u) &= \frac{\sum_{i=1}^{C} r_{ai} r_{ui}}{\sum_{i=1}^{C} r_{ai}} \\ \mathrm{Conf}(i \to j) &= \frac{\sum_{u=1}^{L} r_{ui} r_{uj}}{\sum_{u=1}^{L} r_{ui}} \end{aligned} \tag{16}$$

This is again similar to cosine similarity. We can thus also use confidence as a similarity measure. Confidence-based similarity of users (resp. items) is defined as:

$$\begin{aligned} \mathrm{Sim}(a,u) &= \mathrm{Conf}(a \to u) \\ \mathrm{Sim}(i,j) &= \mathrm{Conf}(i \to j) \end{aligned} \tag{17}$$

4.1.3 *Asymmetric confidence-based similarity*

We might want to define similarity between a and u (resp. i and j) from both $a \to u$ and $u \to a$ links, which is not the case for confidence. Following [3], we thus define the following Asymmetric Confidence-based Similarity of users/items, where α is a parameter to be tuned by cross-validation:

$$\begin{aligned} \mathrm{Sim}(a,u) &= \left[\mathrm{Conf}(a \to u)\right]^{\alpha} \left[\mathrm{Conf}(u \to a)\right]^{(1-\alpha)} \\ \mathrm{Sim}(i,j) &= \left[\mathrm{Conf}(i \to j)\right]^{\alpha} \left[\mathrm{Conf}(j \to i)\right]^{(1-\alpha)} \end{aligned} \tag{18}$$

This measure is identical to asymmetric cosine, generalizes confidence similarity of link $a \to u$ (for $\alpha = 0$) and of link $u \to a$ (for $\alpha = 1$) and, in the case where matrix R is binary, cosine similarity as well (for $\alpha = 0.5$).

4.1.4 *Jaccard Index-based similarity*

Jaccard Index [36] measures the similarity of lists by counting how many elements they have in common. The Jaccard Index of users a and u (resp. items i and j) is defined (in the binary case) as:

$$\mathrm{Jaccard}(a,u) = \frac{Card\left[\overrightarrow{l}(a) \cap \overrightarrow{l}(u)\right]}{Card\left[\overrightarrow{l}(a) \cup \overrightarrow{l}(u)\right]}$$
$$\mathrm{Jaccard}(i,j) = \frac{Card\left[\overrightarrow{c}(i) \cap \overrightarrow{c}(j)\right]}{Card\left[\overrightarrow{c}(i) \cup \overrightarrow{c}(j)\right]}$$

According to the definition of the Jaccard index, we thus have for users (and similarly for items):

$$\mathrm{Jaccard}(a,u) = \frac{\sum_{i=1}^{C} r_{ai} \times r_{ui}}{\sum_{i=1}^{C} r_{ai} + \sum_{i=1}^{C} r_{ui} - \sum_{i=1}^{C} r_{ai} \times r_{ui}}$$
$$= \frac{\text{\#Items Cons. By a and u}}{(\text{\#Items Cons. By a}) + (\text{\#Items Cons. By u}) - (\text{\#Items Cons. By a and u})}$$

As above, Jaccard index will be similarly defined if matrix R is not binary. Hence, Jaccard index for users / items is defined as:

$$\begin{aligned} \mathrm{Jaccard}(a,u) &= \frac{\sum_{i=1}^{C} r_{ai} \times r_{ui}}{\sum_{i=1}^{C} r_{ai} + \sum_{i=1}^{C} r_{ui} - \sum_{i=1}^{C} r_{ai} \times r_{ui}} \\ \mathrm{Jaccard}(i,j) &= \frac{\sum_{u=1}^{L} r_{ui} \times r_{uj}}{\sum_{u=1}^{C} r_{ui} + \sum_{u=1}^{C} r_{uj} - \sum_{u=1}^{C} r_{ui} \times r_{uj}} \end{aligned} \tag{19}$$

We then define the Jaccard-based Similarity of users / items as:

$$\begin{aligned} \mathrm{Sim}(a,u) &= \mathrm{Jaccard}(a,u) \\ \mathrm{Sim}(i,j) &= \mathrm{Jaccard}(i,j) \end{aligned} \tag{20}$$

4.2 Neighborhoods

Now that we have defined various similarity measures, we can define the neighborhood $K(a)$ of user a (resp. $V(i)$ of item i): we only give the definition for users, items are similar) in any of the following ways (all users or only the Top N or only those with similarity measure larger than a threshold can be chosen):

- $K(a)$ is the first circle of user a in the Users graph, where neighbors of a in the circle can be rank-ordered by any of the previously defined similarity measures: support-based; confidence-based; asymmetric confidence-based; Jaccard Index-based.
- $K(a)$ is the local community of user a in the Users graph, where local communities are defined as in [43].
- $K(a)$ is the community of user a in the Users graph (where communities are defined, for example, by maximizing modularity [22]).

These last two cases are completely novel ways to define neighborhoods: they exploit the homophily expected in social networks (even implicit as here), where users with the same behavior tend to connect and be part of the same community. In CF, users in $K(a)$ (with cosine similarity) are such that they have consumed at least one item in common (otherwise their cosine would be 0); in the SF setting such users would be linked in the Users graph ($K \geq 1$). In the above community-based definitions of $K(a)$, users in $K(a)$ might not be directly connected to active user a. These definitions thus embody some notion of paths linking users, through common usage patterns.

4.3 Social Collaborative Filtering scores

We now define scoring functions as for Collaborative filtering (equations (10), (11) and (12) above) with the various social similarity measures (equations (15), (17), (18) and (20)) and the various neighborhoods we have defined.

- For user-based SF: average rating (or popularity) of item i by neighbors of a in $K(a)$, weighted average rating, normalized average rating of nearest users weighted by similarity to a, as in equation (10) above.

- For item-based SF: average rating by a of items neighbors of i in $V(i)$, weighted average rating, normalized average rating of nearest items weighted by their similarity with i, as in equation (11) above.

As in equation (12), scores with locality parameters q and q' can be used with any of the similarity measures defined above.
We then rank-order items i by decreasing scores as in equation (13) and retain the top k $(i_1^a, i_2^a, \dots, i_k^a)$ recommended to a.

5. SOCIAL FILTERING AND OTHER RSs

To implement the SF framework, we need to build the bipartite network and project it to unipartite networks, after choosing adequate parameters K and K' (Figure 1) and eliminating mega-hubs if necessary [44]. Then, depending upon the choice of similarity measure (equations (15), (17), (18) and (19)), neighborhoods and score function (equations (10), (11) or (12)) we obtain a RS which is equivalent to one of the following classical recommender systems:

- **Association rules** [9] of length 2 can be used for recommendation [47]. They are obtained from the item-based SF formalism with $K = K' = 1$, asymmetric confidence-based similarity with $\alpha = 0, N = 1$ ($V(i)$ is reduced to the first nearest neighbor) and local scoring function (equation (12)) with $q' = 1$.

- **CF** is obtained with $K = K' = 1$, asymmetric confidence with $\alpha = 0.5$ (cosine similarity) and the usual CF score functions (identical to those used by SF).

- **CF with locality** [3] is obtained with $K = K' = 1$, asymmetric confidence and score function as in equation (12).

- **Social RS**: if an explicit social network is available, such as a friendship network for example, then one can use that network as a Users' graph and proceed as in the SF framework. In [65], the authors use the first circle as neighborhood and show that their results are slightly worse than conventional CF, but at a much reduced computational cost.

- **Content-based RS**: instead of building a bipartite Users x Items graph, one could use the exact same methodology and build a bipartite Users x Users' attributes or Items x Items' attributes graph and recommend items liked by similar users or items similar to those consumed by the user, with a similarity measure based on the projected graphs.

As can be seen above, the SF formalism generalizes various well established recommendation techniques. However, it offers new possibilities as well: the Social Filtering formalism thus extends content-based, association rules and CF, with new similarity measures and new ways to define neighborhoods.

By only building once one bipartite graph and the projected unipartite graphs, we have at our disposal, in a unique framework, a full set of similarity measures, neighborhood definitions and scoring functions; thus allowing us to produce many different RSs, evaluate their performances and select the best.

We will illustrate in section 7 performances on various standard datasets, comparing our SF formalism to conventional techniques and showing original results produced within our SF framework.

6. EVALUATION

6.1 Performance metrics

A RS can be asked either to predict a rating (in the case of explicit feedback) or to rank items (in the case of implicit feedback). The two visions are quite different and globally correspond to model-based and memory-based approaches in CF respectively [1]. *Performance metrics* differ in these two cases [52].
Predictive case: we want to evaluate how close predicted rating $\hat{r}_{ij}$ is to actual rating r_{ij}. We thus use classical performance metrics from machine learning:

- AUC: Area Under the Curve [52].

- RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) defined as:

$$\begin{aligned} MAE &= \frac{1}{I}\sum_{i,j}|r_{ij} - \hat{r}_{ij}| \\ RMSE &= \sqrt{\frac{1}{I}\sum_{i,j}(r_{ij} - \hat{r}_{ij})^2} \end{aligned} \tag{21}$$

where I is the total number of tested ratings.
Ranking case: in that case, we want to evaluate whether recommended items were adequate for the user; for example, recommended items were later consumed. We thus have a Target set for each user which represents the set of items he consumed after being recommended. This can be implemented by splitting the available dataset into Training / Testing subsets (taking into account time stamps if available). In this case, metrics are those classically used in information retrieval:

- *Recall@k* and *Precision@k* are defined as:

$$\begin{aligned} \text{Recall@}k &= \frac{1}{L}\sum_a \frac{Card(R_a \cap T_a)}{Card(T_a)} \\ \text{Precision@}k &= \frac{1}{L}\sum_a \frac{Card(R_a \cap T_a)}{k} \end{aligned} \tag{22}$$

  where $R_a = (i_1^a, i_2^a, ..., i_k^a)$ is the set of k items recommended to a, T_a is the target set for a. We can also plot *Recall@k* as a function of the number of recommended items and compute the AUC for that curve.

- F_β-mesure: F_β is designed to take into account both recall and precision; F_1 is the most commonly used. F_β is defined as :

$$F_\beta@k = \frac{(1+\beta^2) \times prec@k \times recall@k}{(\beta^2 \times prec@k) + recall@k} \tag{23}$$

- MAP@k (Mean Average Precision) was used, for example, in the Million Song Dataset challenge[4]; it is defined as:

$$\text{MAP@}k = \frac{1}{L}\sum_{a=1}^{L}\frac{1}{k}\sum_{i=1}^{k}\frac{C_{ai}}{i}1_{ai} \tag{24}$$

where C_{ai} is the number of correct recommendations to user a in the first i recommendations ($Precision@i$ for user a) and $1_{ai} = 1$ if item at rank i is correct (for user a), 0 otherwise.

In practical situations, we are also interested in more *qualitative indicators* showing whether all users (resp. items) get recommendations (resp. are recommended) or which items are recommended: as we will see, some RSs might be better on performance indicators and poorer on these qualitative indicators and it will be the user's choice to trade performance decrease for qualitative indicators increase.

- **Users' coverage**: this is the proportion of users who get recommendations:

$$\text{UsersCoverage@}k = \frac{\text{\# Users in Test with k Reco}}{L_{\text{test}}} \tag{25}$$

where L_{test} is the number of users in the test set.

- When Users' coverage is partial, we want to know what the **average number of recommendations** was for the users with partial lists:

$$\text{AvNbRec@}k = \sum_{K=0}^{k-1} K\frac{\text{\# Users in Test with k Reco}}{L_{\text{test}} - \text{\# Users with k Reco}} \tag{26}$$

- **Items' coverage**: higher diversity (recommended items are varied) should result in more attractive recommendations [52]. We thus want to have a high coverage, i.e. a high proportion of items which get recommended. When this is evaluated on a test set (e.g. 10% of users), the Items' coverage will appear lower than if we were evaluating it on the full population of users. Yet, since all other indicators are evaluated on the test set, for homogeneity reasons, we will also report the evaluation of Items' coverage of the lists of recommended items for users in the test set.

$$\text{ItemsCoverage@}k = \frac{\text{\# Items in Reco Lists}}{C} \tag{27}$$

- **Head/Tail coverage**: if we rank items by decreasing popularity (number of users who purchased the item), we call *Head* the 20% of items with highest popularity and *Tail* the remaining 80% [47]. Recommending only most popular items will result in relatively poor performances and low diversity. We thus define the rate of recommended items in the *Head* and in the *Tail*:

$$\text{RateHead@}k = \frac{1}{L_{\text{test}}}\sum_{u\in\text{Test}}\frac{\text{\# Reco for u in Head}}{\text{\# Reco for u}}$$
$$\text{RateTail@}k = \frac{1}{L_{\text{test}}}\sum_{u\in\text{Test}}\frac{\text{\# Reco for u in Tail}}{\text{\# Reco for u}} \tag{28}$$

where the sum runs on the L_{test} users in the Test set.

[4]http://kaggle.com/c/msdchallenge

Many more indicators are described in [52], but we will only use these for the experiments described in section 7.

6.2 Data sets

To demonstrate the generality of our framework, we present results on various datasets traditionally used in the literature. These datasets, shown in Table 2 are characterized by the number of users, items, preferences (*implicit* shown by a count of usage or *explicit* shown by ratings) and, in some cases, existing *explicit* social relationships.

Dataset	Preferences	Preferences Type	Users	Items	Explicit Social
LastFM	92,834	Count	1,892	17,632	25,434
MovieLens1M	1 M	Ratings	6,040	3,883	–
Flixster	8.2 M	Ratings	1 M	49,000	26.7 M
MSD	48 M	Count	1.2 M	380,000	–

Table 2: Datasets.

- **Lastfm**: this dataset[5] contains 92,834 listening information counts of 17,632 artists by 1,892 users of the website Lastfm.com. There is an explicit friends' network with 25,434 links.
- **Flixster**: this dataset[6] contains 8.2 M ratings of about 49,000 movies by about 1 M users. There is an explicit friends' network with 26.7 M links.
- **MovieLens1M**: this dataset[7] contains 1,000,209 ratings of approximately 3,900 movies made by 6,040 users who joined MovieLens in 2000.
- **Million Song Dataset** (MSD): this dataset was used for a Kaggle challenge[4]. It contains 48 M listening counts of 380,000 songs by 1.2 M users.

Some performance results of RSs on these datasets are already available in the literature. See for example:

- Lastfm: user-based CF and inferred explicit trust network [63];
- Flixster: user-based CF and matrix factorization [26];
- MovieLens1M: user-based CF and local scoring function with asymmetric confidence in [3]; CF and matrix-factorization [62];
- MSD: CF and local scoring function with asymmetric confidence [3].

7. EXPERIMENTS

We have implemented our SF formalism on the various datasets presented above. The goal of these experiments is **not** to demonstrate that our formalism produces better results than other RSs: this would have required systematic runs of multiple splits and optimization of parameter choices;

[5]http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-2k.zip
[6]http://www.cs.ubc.ca/~jamalim/datasets/
[7]http://files.grouplens.org/datasets/movielens/ml-1m.zip

whereas we only performed one run without optimizing parameters.

We rather intend to show the *versatility* of our formalism which provides many different ways to assemble the various ingredients and produce old and new RSs. We also give details, which are not always present in the literature, about the settings we used for the experiments. To allow for comparison, our code is available in open source[8].

Our experiments will be presented in two sets:

- First, we show that our SF formalism produces results identical to those found by conventional techniques, demonstrating that this formalism can be particularized into some of the existing techniques. Since results reported in the literature use various evaluation metrics and settings (training/test splits for example), we have defined homogeneous settings and reproduced classical methods to compare them to our formalism;
- Then, we show cases where our SF formalism generates new RSs.

In the experiments reported below, we have used various settings which are not always explicit in the literature:

- **Binarization of inputs**
  - Counts: we split the interval of counts into 10 bins with the same number of elements. We then replace each count in matrix R by the bin index;
  - Ratings: we transform ratings into value from 1 to 10 (for example rating 1 to 5 stars is multiplied by 2);
  - Then we transform values $r \in [0, 10]$ into a binary value. We have used the same setting as [3]: 10 is coded as 1, all others as 0. Then all users, resp. items, with their entire line, resp. column, at 0 are eliminated, which is a very drastic reduction of the number of users and items.
- **Data split**: to evaluate performances, we split data into two data sets, one used for training, the other for testing. We implemented the same technique as in [3] for the MSD challenge: take all users, randomize and split: 90% users for training and 10% kept for testing.
  - For training, we use all transactions of the 90% users.
  - We test on the remaining transactions of the test users: we input 50% of the transactions of each test user, and compare the obtained recommendations list to the remaining 50%.
- **Choices of similarity measure, neighborhood and scoring function**. These will be varied, producing the various RSs we want to implement.

For reference, to compare performances obtained in our experiments, we have implemented three classical techniques:

- **Popularity**: we rank items by decreasing popularity (the sum of 1s in the matrix R item column); popularity served as baseline in the MSD challenge[4];
- **Bigrams**: we used the apriori algorithm for association rules [9] with length 2 and thresholds on support and confidence at 1% and did not try to further optimize the settings; items are ranked in decreasing confidence of the rule generating them;
- **NMF (non-negative matrix factorization)**: we used the code[9] associated to [28], with maximal rank 100 and maximum number of iterations 10,000 and did not try to further optimize the settings.

Performances shown in Table 3 provide a baseline: figures in bold indicate the best performance of the corresponding category. We run our simulations on an Intel Xeon E7-4850 2,00 GHz (10 cores, 512 GB RAM), shared with members of the team (so concurrent usage might have happened in some of the experiments, with impact on reported time). Computing time in hours is thus indicative only (0:01:00 is 1 min, 4:20:00 is 4 hours 20 min). Our formalism was implemented using state-of- the-art libraries, such as Eigen[10] for computing similarities.

As can be seen, bigrams are very efficient in terms of performances on all four datasets (see also [47]) and they require no parameters tuning (except the support and confidence threshold). But bigrams have low Users' and Items' coverage, with all recommended items in the Head.
In contrast, NMF are very sensitive to parameters (maximal rank and maximum number of iterations) and since we did no try to optimize these parameters, we obtain low performances here. In addition, NMF do not scale well with increasing size of datasets. On the other hand, NMF have best Users' and Items' coverage (note that after 4 days of computing time, we stopped NMF on MSD). These two techniques illustrate the trade-off one has to make in practice: fine tune parameters vs. default parameters to obtain optimal performances, and performances vs. coverage. Finally scalability is indeed a critical feature.

7.1 SFs reproduce classical RSs

We have implemented CF with the code provided by the author[11] in [3], in 2 versions:

- **Classical CF** [1] with cosine similarity and average score. Results are shown in Table 4, lines CF IB (item-based) and CF UB (user-based);
- **CF with locality** [3]: we exactly reproduced the results shown in [3] for $k = 500$ recommendations as in the MSD challenge (we do not show these results here). We show results in Table 4 for $k = 10$ recommendations (columns CF IB_Aiolli) for values $q = 5$ and $\alpha = 0$. These results are coherent with those presented in [3].

[8]`https://bitbucket.org/danielbernardes/socialfiltering`

[9]`https://github.com/kimjingu/nonnegfac-python`

[10]`http://Eigen.tuxfamily.org`

[11]Code on `http://www.math.uni.pd.it/~aiolli/CODE/MSD`

Data sets	**LastFM**			**Flixster**			**MovieLens1M**			**MSD**		
Methods	**Popularity**	**Bigrams**	**NMF**	**Popularity**	**Bigrams**	**NMF**	**Popularity**	**Bigrams**	**NMF**	**Popularity**	**Bigrams**	**NMF**
MAP @ 10	0.058	**0.144**	0.005	0.038	**0.098**	0.005	0.097	**0.149**	0.008	0.022	**0.135**	*
Prec @ 10	0.053	**0.082**	0.048	0.061	**0.120**	0.075	0.150	**0.206**	0.141	0.017	**0.042**	*
Recall @ 10	0.165	**0.260**	0.161	0.092	**0.126**	0.116	0.120	**0.155**	0.113	0.055	**0.175**	*
Users Full Coverage	**100%**	52.86%	**100%**	**100%**	74.00%	79.65%	**100%**	99.50%	**100%**	**100%**	0%	*
Users Partial Coverage	0%	47%	0%	0%	26.00%	20.35%	0%	0.50%	0%	0%	100%	*
– Avg. num. of recs.	-	3.4	-	-	2.3	2.8	-	5.0	-	-	1.9	*
Items coverage	0.46%	1.13%	**2.87%**	0.06%	0.33%	**0.60%**	0.62%	**3.47%**	**3.47%**	**0.008%**	0.001%	*
– Proportion in Tail	0%	0%	**1.95%**	**0%**	**0%**	**0%**	0%	**0.30%**	0.01%	**0.00%**	**0.00%**	*
– Proportion in Head	100%	100%	**98.05%**	**100%**	**100%**	**100%**	**100%**	**99.70%**	99.99%	**100.00%**	**100.00%**	*
Computation time	**0:01:00**	0:05:00	1:00:00	**0:01:00**	0:05:00	4:00:00	**0:01:00**	0:05:00	2:00:00	**0:14:52**	0:30:00	> 4 days

Table 3: Performances of reference techniques.

Note that in Table 4, the neighborhood chosen simply consists of all users / items with non null similarity (the case with $q = 5$ restores some locality). We found that limiting to a Top N neighbors (we just tested $N = 100$) usually resulted in decreased performances and coverage. Note that, with the datasets sizes we use, a relative difference of 1% is significant.

We have then implemented our SF formalism to reproduce classical RSs:

- **Association rules**: we obtained, as expected, the exact same performances as those shown in Table 3;
- **Item-based CF**: we again obtained, as expected, from the SF formalism, the exact same results as those in Table 4 (lines CF IB and CF UB with cosine similarity, all users/items for neighborhoods $K(a)$ and $V(i)$ and weighted average score).
- **CF with locality**: we implemented our SF framework, with settings described in section 5 ($K = K' = 1$). The projected networks (and not the explicit network in the case of Lastfm) can be used with different similarity measures and scoring functions, producing various RSs. We again obtained, as expected, from the SF formalism the exact same results as those in Table 4 (lines CF IB Aiolli and CF UB Aiolli).

When comparing CF IB, CF UB, with cosine or Aiolli asymmetric confidence, we do not find one technique systematically best:

- **For CF IB**, asymmetric confidence has better performances and poorer Items' coverage than cosine on datasets Lastfm and Flixster; but on MovieLens and MSD, cosine is better for performances.
- **For CF UB**, asymmetric confidence is better than cosine for all datasets except MovieLens.
- **CF IB outperforms CF UB** for all datasets except Flixster (both in terms of performance and Users' and Items' coverage).

Note that parameters ($\alpha = 0, q = 5$) have not been optimized the way they were in [3]: it would certainly be possible to choose, by cross-validation, better parameters. But, as we said, the purpose of these experiments is not to fully optimize settings.

Comparing Table 3 and Table 4 shows that bigrams seem to perform best (except for MovieLens), but their Users' coverage is poor (because the filter on support and confidence limits the number of rules: the threshold was not optimized). These results show that our formalism covers these various classical techniques: association rules, CF (item or user-based) and CF with locality as in [3].

7.2 SF produces new RSs

We now show in Tables 5 and 6 implementations of the SF formalism with various choices of similarity measures (equations, (15), (17), (18) and (20)) and neighborhood $K(a)/V(i)$. We use as score function the weighted average (equations (10) and (11) middle) or, for the asymmetric confidence similarity, the *local score* of equation (12). In our implementation, we filter the links in the Users' and Items' networks with support / confidence less than 1%.
We thus have three differences with the classical implementations of CF above: one is the choice of neighborhood (first circle or local community in the implicit Users or Items network, instead of most similar users/items for CF); next one is the filtering of links in SF which results in a reduction in the numbers of neighbors; and last one is the choice of various similarity measures such as support, confidence, or Jaccard (equations (15), (17) and (20)). Here are our main findings:

- **Item-based Social Filtering**:
  - Results for SF IB with 1st circle as neighborhood (in Table 5) show that the various similarity measures produce rather different performances. For all datasets, the best technique outperforms bigrams, while improving Users' coverage and Items' coverage. In addition, our implementation of SF IB, compared to CF IB, for the parameters of Table 4 ($\alpha = 0, q = 5$) shows improved performance for all datasets, except MovieLens. Note that Jaccard similarity usually produces poor performances but good Users' and Items' coverage and also succeeds in recommending items in the Tail. Except for MovieLens, where Jaccard similarity provides the best performances on all indicators.
  - Results for SF IB with local community as neighborhood are shown in Table 6. Because local communities are not really relevant in implicit networks, we just run experiments on Lastfm and Flixster. The use of local communities in these

Datasets	**LastFM**		**Flixster**		**MovieLens1M**		**MSD**	
Methods	**CF IB** (cosine, average)	**ACF IB** (α=0, q=5)	**CF IB** (cosine, average)	**ACF IB** (α=0, q=5)	**CF IB** (cosine, average)	**ACF IB** (α=0, q=5)	**CF IB** (cosine, average)	**ACF IB** (α=0, q=5)
MAP @ 10	0.066	**0.107**	0.082	0.082	**0.170**	0.147	0.075	**0.078**
Prec @ 10	0.054	**0.072**	0.090	**0.091**	**0.223**	0.199	**0.057**	0.053
Recall @ 10	0.154	**0.240**	0.164	0.164	**0.169**	0.150	0.158	**0.160**
Users Full Coverage	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.79%	**100.00%**
Users Partial Coverage	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.21%	**0.00%**
– Avg. num. of recs.	-	-	-	-	-	-	4.83	-
Items coverage	**22.16%**	12.44%	**15.24%**	5.40%	**10.61%**	6.80%	**42.02%**	22.88%
– Proportion in Head	**66.97%**	87.75%	**92.00%**	99.00%	**98.50%**	99.86%	**66.20%**	95.45%
– Proportion in Tail	**33.03%**	12.25%	**8.00%**	1.00%	**1.50%**	0.14%	**33.80%**	4.55%
Computation time	0:05:00	0:05:00	2:30:00	2:30:00	0:05:00	0:05:00	72:00:00	72:00:00
Methods	**CF UB** (cosine, average)	**ACF UB** (α=0, q=5)	**CF UB** (cosine, average)	**ACF UB** (α=0, q=5)	**CF UB** (cosine, average)	**ACF UB** (α=0, q=5)	**CF UB** (cosine, average)	**ACF UB** (α=0, q=5)
MAP @ 10	0.050	**0.071**	0.084	**0.089**	0.062	**0.105**	**0.069**	0.058
Prec @ 10	0.059	**0.062**	0.087	**0.093**	0.131	**0.161**	**0.044**	0.037
Recall @ 10	**0.185**	0.176	0.176	**0.188**	0.119	0.119	**0.144**	0.119
Users Full Coverage	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Users Partial Coverage	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
– Avg. num. of recs.	-	-	-	-	-	-	-	-
Items coverage	10.14%	**14.01%**	4.13%	**8.22%**	4.76%	**15.13%**	20.08%	**27.59%**
– Proportion in Head	89.76%	**80.62%**	99.70%	**99.43%**	99.76%	**95.34%**	96.97%	**82.22%**
– Proportion in Tail	10.24%	**19.38%**	0.30%	**0.57%**	0.24%	**4.66%**	4.03%	**17.77%**
Computation time	0:05:00	0:05:00	1:20:00	1:20:00	0:05:00	0:05:00	24:00:00	24:00:00

Table 4: Performances of Collaborative Filtering. CF and ACF are respectively the abbreviations of Collaborative Filtering and Asymmetric Cosine Collaborative Filtering.

cases results in a lower MAP, and generally degrades all performance metrics.

- **User-based Social Filtering**:
  - Results for SF UB in Table 5 vary depending on the dataset: sometimes the best SF UB (usually with asymmetric confidence) is better than CF UB or bigrams; sometimes it is worse. For MSD, SF UB has poorer MAP but better precision and recall than best CF IB or bigram.

- **Social RS**: we implemented our SF formalism using the explicit networks (Flixster and Lastfm) for Users' graph; the neighborhood is the 1st Circle of friends, Communities (computed with Louvain's algorithm [7]) or Local Communities (computed with our algorithm in [43]) and the score function is the average. Results in Table 7 show, as usual, various situations: on Lastfm 1st Circle has best performances, but Community better Users' coverage and Local Community better Items' coverage. For Flixster, Community is best, but 1st Circle has better Items' coverage and Local Community better Tail coverage.

7.3 Ensemble methods

As was demonstrated in many cases, and very notably in the Netflix challenge [6; 30], ensembles of RSs often get improved performances. We have thus implemented a few ensembles using a basic combination method, originated from [13], and used in [3]. To combine N recommenders, we assemble, for each user a, the N lists of k recommended items produced by the N RSs.

Let us denote $(i_1^n, \dots, i_k^n)$ the list of items recommended to a (omitted in the notation) by RS n (with $n = 1, ..., N$). Some of the lists might have less than k elements. For each list, we assign points to the items in each list as follows: 1st item gets k points, 2nd item $k-1$, etc. The lists are then fused and each item gets the sum of points in the various lists, with the ties resolved through a priority on RSs (the winner comes from the highest priority list).

In order to explore the potential of ensemble techniques, we tried combinations of pairs of RSs: we tested combinations of some of the best SF-based recommenders mentioned above (Tables 4, 5 and 6). Results shown in Table 8 are encouraging:

- For Lastfm, the best two-systems ensemble is CF-IB with Cosine similarity and weighted average score, combined with SF-IB with asymmetric confidence similarity and local score ($q = 1, \alpha = 0$), which gets a $MAP@10 = 0.16$. This performance is not better than the one obtained with a unique recommender.

- For Flixster, we obtain an improvement: combining SF-UB with asymmetric confidence similarity and local score ($q = 1, \alpha = 0$) and SF-UB with asymmetric confidence similarity and local score ($q = 5, \alpha = 0$) leads to a $MAP@10$ of 0.175, while the best performance was SF-UB with asymmetric confidence similarity and local score ($q = 5, \alpha = 0$) at $MAP@10 = 0.157$.

These preliminary results indeed confirm the potential of ensemble methods. They could certainly be enhanced by testing combinations of more systems, optimizing the parameters and implementing more adequate aggregation methods (Borda's aggregation method [13] seems rather popular in the literature, but can certainly be improved upon to merge recommendation results).

8. CONCLUSION

We have presented a simple and generic formalism based upon social network analysis techniques: by building once the projected Users' and Items' networks, the formalism allows reproducing a wide range of RSs found in the literature while also producing new RSs. This unique formalism thus provides very efficient ways to test the performances of many different RSs on the dataset at hand to select the most adequate in that case.

LastFM	**Item-based**							**User-based**						
Similarity Neighborhood = 1st circle	**Support**	**Confidence**	**Asym. Cos (q=1)**		**Asym. Cos (q=5)**		**Jaccard**	**Support**	**Confidence**	**Asym. Cos (q=1)**		**Asym. Cos (q=5)**		**Jaccard**
			α=0	**α=0.5**	**α=0**	**α=0.5**				**α=0**	**α=0.5**	**α=0**	**α=0.5**	
MAP @ 10	0.145	0.082	**0.161**	0.157	0.151	0.148	0.082	**0.062**	**0.062**	0.050	0.057	0.040	0.042	0.051
Prec @ 10	0.087	0.067	**0.087**	0.084	0.089	0.076	0.053	**0.130**	**0.130**	0.107	0.127	0.083	0.093	0.063
Recall @ 10	0.280	0.188	**0.280**	0.269	**0.283**	0.250	0.164	**0.208**	**0.208**	0.193	0.203	0.144	0.158	0.200
Users Full Coverage	56.45%	56.45%	56.45%	56.45%	56.45%	56.45%	**96.72%**	96.66%	96.66%	96.66%	96.66%	87.50%	96.66%	**96.72%**
Users Partial Coverage	43.00%	43.00%	43.00%	43.00%	43.00%	43.00%	3.28%	3.33%	3.33%	3.33%	3.33%	12.50%	3.33%	3.28%
– Avg. num. of recs.	3.9	3.9	3.9	3.9	3.9	3.9	3.8	9.0	9.0	9.0	9.0	7.3	9.0	4.8
Items coverage	0.92%	1.13%	0.96%	1.06%	0.96%	1.03%	**22.97%**	4.75%	4.75%	5.42%	5.25%	4.71%	5.82%	**10.32%**
– Proportion in Head	100%	100%	100%	100.00%	100.00%	100.00%	**60%**	99.33%	99.33%	94.66%	96.00%	**93.27%**	94.00%	95.20%
– Proportion in Tail	0%	0%	0%	0%	0%	0%	**40%**	0.66%	0.66%	5.33%	4.00%	**6.72%**	6.00%	4.79%
Computation time	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00

MovieLens1M	**Item-based**							**User-based**						
Similarity Neighborhood = 1st circle	**Support**	**Confidence**	**Asym. Cos (q=1)**		**Asym. Cos (q=5)**		**Jaccard**	**Support**	**Confidence**	**Asym. Cos (q=1)**		**Asym. Cos (q=5)**		**Jaccard**
			α=0	**α=0.5**	**α=0**	**α=0.5**				**α=0**	**α=0.5**	**α=0**	**α=0.5**	
MAP @ 10	0.127	0.091	0.133	0.154	0.144	0.144	**0.168**	0.097	0.084	0.093	0.075	**0.129**	0.088	0.062
Prec @ 10	0.172	0.151	0.182	0.208	0.196	0.203	**0.224**	0.225	0.209	0.203	0.175	**0.239**	0.191	0.130
Recall @ 10	0.133	0.104	0.141	0.159	0.148	0.150	**0.166**	**0.033**	0.031	0.030	0.025	0.032	0.027	0.119
Users Full Coverage	99.16%	99.16%	99.16%	99.16%	99.16%	99.16%	**100.00%**	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Users Partial Coverage	0.84%	0.84%	0.84%	0.84%	0.84%	0.84%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
– Avg. num. of recs.	5.0	5.0	5.0	5.0	5.0	5.0	-	-	-	-	-	-	-	-
Items coverage	1.70%	13.43%	1.92%	4.42%	2.48%	5.85%	**13.55%**	4.36%	4.39%	3.99%	4.08%	2.75%	4.08%	**5.45%**
– Proportion in Head	100%	100%	100%	100%	100%	100%	**98%**	96%	96%	95.63%	**95.00%**	100.00%	95.00%	99.73%
– Proportion in Tail	0%	0%	0%	0%	0%	0%	**2%**	4%	4%	4.38%	**5.00%**	0.00%	5.00%	0.27%
Computation time	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:15:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:30:00

Flixster	**Item-based**							**User-based**						
Similarity Neighborhood = 1st circle	**Support**	**Confidence**	**Asym. Cos (q=1)**		**Asym. Cos (q=5)**		**Jaccard**	**Support**	**Confidence**	**Asym. Cos (q=1)**		**Asym. Cos (q=5)**		**Jaccard**
			α=0	**α=0.5**	**α=0**	**α=0.5**				**α=0**	**α=0.5**	**α=0**	**α=0.5**	
MAP @ 10	0.079	0.074	0.080	0.084	**0.105**	0.096	0.078	0.124	0.107	**0.157**	0.141	**0.157**	0.134	0.041
Prec @ 10	0.100	0.096	0.102	0.104	**0.119**	0.117	0.086	0.279	0.250	0.279	0.250	**0.318**	0.250	0.069
Recall @ 10	0.113	0.097	0.115	0.112	0.126	0.117	**0.156**	0.005	0.005	0.005	0.005	0.006	0.005	**0.168**
Users Full Coverage	71.10%	71.10%	71.10%	71.10%	82.31%	71.10%	**100.00%**	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Users Partial Coverage	28.90%	28.90%	28.90%	28.90%	17.69%	28.90%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
– Avg. num. of recs.	2.4	2.4	2.4	2.4	3.6	2.4	-	-	-	-	-	-	-	-
Items coverage	0.29%	0.43%	0.29%	0.39%	0.29%	0.39%	**18.01%**	0.26%	0.27%	0.25%	0.25%	0.24%	0.25%	**6.60%**
– Proportion in Head	100%	100%	100%	100%	100%	100%	**95.46%**	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	**99.69%**
– Proportion in Tail	0%	0%	0%	0%	0%	0%	**4.54%**	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**0.31%**
Computation time	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	0:05:00	43:00:00	0:10:00	0:10:00	0:10:00	0:10:00	0:10:00	0:10:00	112:00:00

MSD	**Item-based**							**User-based**						
Similarity Neighborhood = 1st circle	**Support**	**Confidence**	**Asym. Cos (q=1)**		**Asym. Cos (q=5)**		**Jaccard**	**Support**	**Confidence**	**Asym. Cos (q=1)**		**Asym. Cos (q=5)**		**Jaccard**
			α=0	**α=0.5**	**α=0**	**α=0.5**				**α=0**	**α=0.5**	**α=0**	**α=0.5**	
MAP @ 10	**0.135**	0.134	0.135	0.134	0.135	0.134	0.078	**0.054**	0.054	0.052	0.053	0.049	0.049	0.028
Prec @ 10	**0.042**	0.042	0.042	0.042	0.042	0.042	0.059	**0.090**	0.090	0.089	0.090	0.083	0.084	0.038
Recall @ 10	**0.175**	0.175	0.175	0.175	0.175	0.175	0.166	**0.178**	0.178	0.175	0.178	0.168	0.166	0.116
Users Full Coverage	**0.00%**	0.00%	0.00%	0.00%	0.00%	0.00%	**99.58%**	**99.01**%	99.01%	99.01%	0.990	97.37%	0.990	86.70%
Users Partial Coverage	100.00%	100%	100%	100%	100%	100%	**0.42%**	0.99%	0.99%	0.99%	0.99%	2.63%	1.01%	13.30%
– Avg. num. of recs.	1.9	1.9	1.9	1.9	1.9	1.9	**5.1**	7.0	7.0	7.0	7.0	6.4	7.0	2.9
Items coverage	0.0012%	0.0012%	0.0012%	0.0013%	0.0013%	0.0012%	**46.30%**	7.81%	7.81%	8.74%	8.50%	8.94%	9.33%	**23.37%**
– Proportion in Head	100%	100%	100%	100%	100%	100%	**78%**	97.83%	97.84%	97.56%	97.62%	97.24%	97.39%	**95.80%**
– Proportion in Tail	0%	0%	0%	0%	0%	0%	**22%**	2.17%	2.16%	2.44%	2.38%	2.76%	2.61%	**4.20%**
Computation time	0:10:00	0:10:00	0:10:00	0:10:00	0:10:00	0:10:00	3:00:00	13:50:00	14:10:00	12:52:00	12:51:00	13:37:00	13:00:00	7:00:00

Table 5: Performances of Social Filtering (implicit) with 1st circle; Asym. Cos is the abbreviation for Asymmetric Cosine.

Datasets	**Lastfm**						**Flixster**					
Similarity Neighborhood = Local community	**Support**	**Confidence**	**Asym. Conf. q=1**		**Asym. Conf. q=5**		**Support**	**Confidence**	**Asym. Conf. q=1**		**Asym. Conf. q=5**	
			α=0	**α=0.5**	α=0	**α=0.5**			α=0	**α=0.5**	**α=0**	**α=0.5**
SF-IB												
MAP @ 10	0.151	0.089	0.151	**0.152**	0.149	**0.152**	0.064	0.058	0.066	0.066	**0.081**	**0.081**
Prec @ 10	**0.080**	0.064	**0.080**	0.076	**0.080**	0.073	0.079	0.076	0.082	0.080	0.096	**0.097**
Recall @ 10	**0.256**	0.198	**0.256**	0.243	**0.256**	0.240	0.073	0.065	0.076	0.072	**0.083**	0.079
Users Full Coverage	51.67%	51.67%	51.67%	51.67%	51.67%	51.67%	80.22%	80.22%	80.22%	80.22%	80.22%	80.22%
Users Partial Coverage	48.33%	48.33%	48.33%	48.33%	48.33%	48.33%	19.78%	19.78%	19.78%	19.78%	19.78%	19.78%
Av. nb of recommendations	4.00	4.00	4.00	4.00	4.00	4.00	3.13	3.13	3.13	3.13	3.13	3.13
Items coverage	0.74%	**0.92%**	0.74%	0.89%	0.82%	0.85%	0.30%	**0.35%**	0.30%	0.34%	0.29%	0.34%
Rate-Head	100.00%	100%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Rate-Tail	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Computation time	0:05:00	0:06:00	0:08:00	0:10:00	0:09:00	1:10:00	0:18:00	0:18:00	0:23:00	0:22:00	0:21:00	0:23:00

Table 6: Performances of Social Filtering (implicit) with Local Community.

Dataset	Lastfm			Flixster		
Perf. Indicator @10 SF-UB	**1st circle**	**Community**	**Local Comm.**	**1st circle**	**Community**	**Local Comm.**
MAP @ 10	**0.101**	0.083	0.078	0.019	**0.038**	0.016
Prec @ 10	**0.074**	0.064	0.053	0.037	**0.057**	0.027
Recall @ 10	**0.239**	0.182	0.203	0.046	**0.093**	0.036
Users Full Coverage	80.16%	**88.09%**	56.52%	84.16%	**99.87%**	61.45%
Users Partial Coverage	19.84%	11.91%	43.48%	15.85%	0.13%	38.55%
Av. nb of recommendations	4.76	4.66	4.375	3.85	3.2	3.71
Items coverage	13.54%	8.22%	**14.25%**	**8.01%**	0.37%	7.96%
Rate-Head	89.93%	93.77%	**83.96%**	99.53%	100.00%	**99.18%**
Rate-Tail	10.07%	6.22%	**16.04%**	0.47%	0.00%	**0.82%**
Computation time	0:05:00	0:05:00	0:05:00	0:05:00	58:00:00	0:05:00

Table 7: Performances of Social Filtering user-based on explicit networks.

Datasets	Lastfm	Flixster
Perf. Indicator @10	**(CF-IB Cosine, w. average) and (SF-IB Asym. Conf. q=1, α=0)**	**(SF-UB Asym. Conf. q=5, α=0) and (SF-UB Asym. Conf. q=1, α=0)**
MAP @ 10	0.160	0.175
Prec @ 10	0.089	0.307
Recall @ 10	0.298	0.006
Users Full Coverage	56.45%	100%
Users Partial Coverage	43.55%	0%
Av. nb of recommendations	3.9	-
Items coverage	0.96%	0.25%
Rate-Head	100%	100%
Rate-Tail	0%	0%
Computation time	0:01:00	0:06:00

Table 8: Ensemble of RSs.

As can be seen from our experiments, there is no unique silver bullet (so far!): for each dataset, one has to test and try a full repertoire of candidate RSs, fine tuning hyper-parameters (a topic we did not address in this paper) and selecting the best RS for the performance indicator he/she cares for. The richer the repertoire is, the more chances for the final RS to get best performances. This theoretical formalism thus provides a very powerful way to formalize and compare many RSs.

In addition, this integrated formalism enables the production of modular code, uncoupling the similarities and scoring functions computation steps. It also allows for elegant implementation of the recommendation engines, as demonstrated in our published open source code[12].

Computing the bipartite network and its projections requires significant computing resources. It can be considered as a set-up step for our recommender framework. This overhead is only worth it if one wants to produce an ensemble of RSs originating from various choices of parameters (similarity measures, neighborhoods and scoring functions) to make comparisons and select the best choice for the dataset at hand. In addition, computation of similarities is the bottleneck (since obviously it involves all pairs of users or items). However, some similarity measures (asymmetric confidence or Jaccard) are more costly than others (support, confidence and cosine) as the figures about computing time show in the various tables.

This work opens ways for future research in several directions:

- We have introduced various choices of similarity measures, neighborhood and scoring functions. Obviously, more choices can be designed and evaluated;

- Since it is easy to produce many RSs within the same framework, we could produce collections – or ensembles – of RSs working together to complement each other weaknesses. More ways for combining recommended lists will be needed to improve upon the Borda's mechanism discussed here. Inspiration from the literature on ensemble of rating-based RSs could certainly be useful.

- Since implicit and explicit social networks can be set into the same framework, further investigation is required on how to integrate implicit and explicit networks, thus producing hybrid Social recommenders;

- Recent work[13] allowing to merge user-based and item-based CF seem promising, and could certainly be framed into our formalism;

- The first phase in our formalism consists in projecting the Users and Items network. This step is computationally heavy. Hence, at the present time we cannot process extremely large datasets, such as for example in the MSD challenge. We thus intend to optimize our implementation of the Social Network projection. More generally, our code could be ported to distributed Hadoop-based environments to allow processing of larger datasets and parallel testing of the various hyper-parameters choices.

9. ACKNOWLEDGEMENTS

[12]`https://bitbucket.org/danielbernardes/socialfiltering`

[13]Personal communication of one of the authors [58].

This work has been partially supported by projects AMMICO and OFS (funded by BPI France). Access to public datasets and code is gratefully acknowledged.

10. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM, 2009.

[3] F. Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 273–280, New York, NY, USA, 2013. ACM.

[4] J. Aranda, I. Givoni, J. Handcock, and D. Tarlow. An online social network-based recommendation system. *Toronto, Ontario, Canada*, 2007.

[5] P. Avesani, P. Massa, and R. Tiella. A trust-enhanced recommender system application: Moleskiing. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1589–1593. ACM, 2005.

[6] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[7] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[8] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[9] C. Borgelt. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):437–456, 2012.

[10] D. M. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2008.

[11] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[12] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'El, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1227–1236. ACM, 2009.

[13] J.-C. de Borda. On elections by ballot. *Classics of social choice, eds. I. McLean, AB Urken, and F. Hewitt*, pages 83–89, 1995.

[14] L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. A collaborative recommender system based on probabilistic inference from fuzzy observations. *Fuzzy Sets and Systems*, 159(12):1554–1576, 2008.

[15] Z. Despotovic and K. Aberer. A probabilistic approach to predict peers' performance in p2p networks. In *Cooperative Information Agents VIII*, pages 62–76. Springer, 2004.

[16] M. Diaby, E. Viennet, and T. Launay. Toward the next generation of recruitment tools: an online social network-based job recommender system. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 821–828. ACM, 2013.

[17] M. Diaby, E. Viennet, and T. Launay. Exploration of methodologies to improve job recommender systems on social networks. *Social Network Analysis and Mining*, 4(1):1–17, 2014.

[18] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.

[19] M. Gartrell, X. Xing, Q. Lv, A. Beach, R. Han, S. Mishra, and K. Seada. Enhancing group recommendation by incorporating social relationship interactions. In *Proceedings of the 16th ACM international conference on Supporting group work*, pages 97–106. ACM, 2010.

[20] J. A. Golbeck. Computing and applying trust in web-based social networks. 2005.

[21] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pages 403–412. ACM, 2004.

[22] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral. Module identification in bipartite and directed networks. *Physical Review E*, 76(3):036102, 2007.

[23] M. Gupte and T. Eliassi-Rad. Measuring tie strength in implicit social networks. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 109–118. ACM, 2012.

[24] J. He and W. W. Chu. *A social network-based recommender system (SNRS)*. Springer, 2010.

[25] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[26] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

[27] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[28] J. Kim and H. Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.

[29] M. W. Kim, E. J. Kim, and J. W. Ryu. A collaborative recommendation based on neural networks. In *Database Systems for Advanced Applications*, pages 425–430. Springer, 2004.

[30] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[31] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[32] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SDM*, volume 5, pages 1–5. SIAM, 2005.

[33] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[34] N. N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 759–766. ACM, 2009.

[35] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[36] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[37] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210. ACM, 2009.

[38] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[39] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 492–508. Springer, 2004.

[40] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[41] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266. ACM, 2003.

[42] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[43] B. Ngonmang, M. Tchuente, and E. Viennet. Local community identification in social networks. *Parallel Processing Letters*, 22(01), 2012.

[44] B. Ngonmang, E. Viennet, S. Sean, F. Fogelman-Soulié, and R. Kirche. Monetization and services on a real online social network using social network analysis. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 185–193. IEEE, 2013.

[45] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174. ACM, 2005.

[46] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[47] B. Pradel, S. Sean, J. Delporte, S. Guérif, C. Rouveirol, N. Usunier, F. Fogelman-Soulié, and F. Dufau-Joel. A case study in a recommender system based on purchase data. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–385. ACM, 2011.

[48] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *The Semantic Web-ISWC 2003*, pages 351–368. Springer, 2003.

[49] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[50] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[51] S. Shang, P. Hui, S. R. Kulkarni, and P. W. Cuff. Wisdom of the crowd: Incorporating social influence in recommendation models. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 835–840. IEEE, 2011.

[52] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[53] R. R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends. In *DELOS workshop: personalisation and recommender systems in digital libraries*, volume 106, 2001.

[54] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[55] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[56] J. Tang, X. Hu, and H. Liu. Social recommendation: a review. *Social Network Analysis and Mining*, 3(4):1113–1133, 2013.

[57] C.-Y. Teng, Y.-R. Lin, and L. A. Adamic. Recipe recommendation using ingredient networks. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 298–307. ACM, 2012.

[58] K. Verstrepen and B. Goethals. Unifying nearest neighbors collaborative filtering. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 177–184. ACM, 2014.

[59] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[60] K. Wang and Y. Tan. A new collaborative filtering recommendation approach based on naive bayesian method. In *Advances in Swarm Intelligence*, pages 218–227. Springer, 2011.

[61] Z. Xia, Y. Dong, and G. Xing. Support vector machines for collaborative filtering. In *Proceedings of the 44th annual Southeast regional conference*, pages 169–174. ACM, 2006.

[62] B. Xu, J. Bu, C. Chen, and D. Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, pages 21–30. ACM, 2012.

[63] C. Yin, T. Chu, et al. Improving personal product recommendation via friendships' expansion. *Journal of Computer and Communications*, 1(05):1, 2013.

[64] R. Zheng, F. Provost, and A. Ghose. Social network collaborative filtering: Preliminary results. In *Proceedings of the Sixth Workshop on eBusiness WEB2007*, 2007.

[65] R. Zheng, D. Wilkinson, and F. Provost. Social network collaborative filtering. *Stern, IOMS Department, CeDER, Vol*, 2008.

[66] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115, 2007.

[67] C.-N. Ziegler and G. Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.