MobHinter: Epidemic Collaborative Filtering and Self-Organization in Mobile Ad-Hoc Networks

Rossano Schifanella, André Panisson, Cristina Gena and Giancarlo Ruffo
Dipartimento di Informatica
Università degli Studi di Torino
Corso Svizzera, 185 - 10149, Torino. ITALY
{schifane,panisson,cgena,ruffo}@di.unito.it

ABSTRACT

We focus on collaborative filtering dealing with self-organizing communities, host mobility, wireless access, and ad-hoc communications. In such a domain, knowledge representation and users profiling can be hard; remote servers can be often unreachable due to client mobility; and feedback ratings collected during random connections to other users' ad-hoc devices can be useless, because of natural differences between human beings. Our approach is based on so called *Affinity Networks*, and on a novel system, called *MobHinter*, that epidemically spreads recommendations through spontaneous similarities between users. Main results of our study are two fold: firstly, we show how to reach comparable recommendation accuracies in the mobile domain as well as in a complete knowledge scenario; secondly, we propose epidemic collaborative strategies that can reduce rapidly and realistically the cold start problem.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Sociology*

General Terms

Algorithms, Design, Measurement

Keywords

Recommender Systems, Social Collaborative Filtering, Ad-Hoc Networks

1. INTRODUCTION

Social Networking services and self-organized communities are rapidly filling the Web with innovative and highly popular data-driven applications. Most of these applications are based on data sharing, and even if architectural models vary from rigid client/server implementations to pure peer-to-peer systems, users are invited to join a given community and distribute their own files, like photos (e.g. Flickr), videos (e.g., YouTube, Joost), music (e.g., Emule, Gnutella), blogs, personal data, and other information even just for being connected to other people (e.g., eBlogger, MySpace, Facebook).

Moreover, users can communicate each other by means of a broader and broader set of short and wide range wireless network interfaces (e.g., Bluetooth, Wi-Fi, Wi-Max, GSM, UMTS). Some analysts [1] expect that these applications will finally increase 3G data traffic. Nevertheless, there is an increasingly influential opposite trend, because ad-hoc communications can be exploited to collect relevant information during ordinary off-line user's activity, and/or when WAN connectivity is difficult (e.g., in the underground, in the airplane), expensive, or simply superfluous. In addition, smart vehicles have embedded computers, that are going to host vehicular computing and networking solutions addressed to safer driving, dynamic route planning, and a new generation of entertainment applications.

In such a scenario, adaptive recommendation systems can play a leading role for reducing information overload, and for filtering out useless information. Unfortunately, there are a number of drawbacks that make many approaches unfit to run. Firstly, knowledge representation can be hard, because (1) it is difficult to trace a user's profile if she/he accesses services presenting different identities and pseudonyms, and (2) flexible and less expressive languages based on user-defined tags (e.g., folksonomy) are preferred instead of rigid, even if exhaustive, taxonomies.

Secondly, mobile devices can be unable to access the Internet or a remote server. Therefore, the adopted architectural model should allow the recommendation system to operate in autonomy.

Thirdly, when the user's device is able to connect to other hosts in the proximity via ad-hoc connections, it grants access to a very limited portion of the broader community and to a small subset of all the available data. More significantly, information coming from random "rendez vous" can result uninteresting for a user, accordingly to her/his own preferences. This is an issue for collaborative filtering, because neighbors should be selected for similarities, and no assumption of this kind can be done on users that are in a context-free proximity.

Contributions and Road Map

Our proposal is based on a connection oriented collabora-

RecSys'08, October 23–25, 2008, Lausanne, Switzerland.

tive filtering approach that strongly mitigates domain representation's problems. Hence, after a brief survey on related works (Section 2), we introduce *Affinity Networks* in Section 3. This model represents a similarity graph that links users each other using a configurable affinity threshold. Section 4 presents *MobHinter* that allows the mobile device to identify affinity network neighbors from random ad-hoc communications. Collected information is used to incrementally refine locally calculated predictions, with no need of interacting with a remote server or accessing the Internet. Using a simulated environment and the well known MovieLens dataset, in Section 5 we show how recommendation accuracies experimented in the mobile domain, with sparse and incomplete knowledge, are comparable to predictions resulting in a scenario with complete knowledge available. Moreover, we empirically prove how epidemic filtering strategies greatly reduce the cold start problem. Finally, we point out some practical observations in Section 6.

2. RELATED WORKS

To authors knowledge, there has been few experience in designing collaborative filtering systems in distributed environments such as mobile ad-hoc networks.

A first attempt to deal with a decentralized environment is proposed in [2] where products and services are suggested in a marketplace populated by mobile customers. In the environment assistant agents act as peers serving the mobile customers. When a neighbor is looking for suggestions it broadcasts a query containing a vector with its votes on products and recommendations. When a peer receives the voting vector it calculates the proximity with the cached previous messages: if the proximity is higher than a threshold, then the peer send back the cached voting vector. If the proximity measure is lower, the query voting vector is broadcasted further to other peers.

In the PocketLens project [3] a collaborative filtering algorithm for finding neighbors in a peer-to-peer environment is proposed. The authors evaluate the algorithm in different architecture for finding neighbors: Central, Random, and Transitive Architectures. The two last architectures are distributed. In particular in the Transitive Architecture the most similar neighbors are incrementally discovered each time a new user is encountered. This approach is comparable to our proposal. In fact, the Transitive Architecture maintains a queue of "neighbor's neighbors". The intuition is that if you are one of my best neighbors, then it is likely that your neighbors are good for me as well. However, differently from our approach the data concerning neighbors are not locally maintained. If some of the neighbor's neighbors are off line while building the model, they are substituted with a pool of quality neighbors. What is also relevant for our approach in the PocketLens project is that in the peer-to-peer context the algorithm achieves an acceptable quality level with a minimum number of neighbors required.

Another work investigating both the issue of the availability of data needed to make predictions and the underlying social aspects of recommender systems is the one from Mirza et al. [4]. They start from the assumption that a recommender, in an indirect way, has the goal of bringing people together. Thus they propose a framework based on a mathematical model of the social network implicit in recommendation. Studying typical recommender dataset they found underlying graph structures. Therefore they analyse recommenders in term of jumping connections. Their goal is to propose a model able to calculate the minimum number of rating constraints and of jumps - between users and between users and artifacts - needed to generate accurate recommendations. However they do not contextualize the problem in distributed and sparse environments, so the problem of finding recommendations in ad-hoc and decentralized networks still remains open.

Castanos and Boyer [5] propose a collaborative filtering model which is totally distributed. Each peer of the network owns the users' public profiles. The goal of their algorithm is to build a group profile for every active user representing the preference of a virtual community of interests. The group profile is generated on the basis of the other public profiles similar to the public profile of the user. The group profile can also be stored on the user device, thus predictions can run locally when the user does not want to share her preferences. They have computed the MAE by simulating arrivals of peers by progressively adding new profiles, and they got predictions as good as using PocketLens algorithm.

The work of Splinder et al. [6] deals specifically with collaborative mobile ad-hoc networks. They introduce the notion of *shared social context* in order to help their distributed collaborative filtering system to establish similarity relationship between copresent users. The assumption behind their proposal is that if two users attend the same event, as arts festival events, it is likely that they have similar interests. Thus, without computing prior similarity, predictions are based on the set of users sharing at least one item, that is an event/location they have consumed during the same period of time. However this approach does not work in context where users are copresent by chance, such as on the subway, or on the bus, etc.

Another proposal is *Tribler* [7, 8], a personalized peer-to-peer television system that automatically recommends, records and downloads content from peers in a social network. This system is based on the concept of buddy tables and provides a distributed algorithm, called *BuddyCast*, to spread in the network the ratings database. Even though *Tribler* proposes a mechanism comparable to our proposal, there are some distinctive differences. In particular, in *Tribler* a user can mark a discovered node as a friend by way of the interface, implementing a sort of list of favorite users. This explicit declaration of friendship does not involve any topological property of the spontaneous similarity network. On the contrary, our proposal exploits implicit properties of the affinity networks, in order to implement an incremental and epidemic algorithm to spread ratings in a mobile community. Further approaches have been proposed in literature, like $PipeCF$ scheme using a *Distributed Hash Table* in order to spread ratings information [9, 10], or the $Vineyard$ [11] system where a set of independent recommender components co-operate in a given domain. In a mobile scenario, other proposals can be found in [12, 13].

3. AFFINITY NETWORKS

In order to model our domain in a more formal way, we have a set of users $U = \{u_1, u_2, \ldots, u_n\}$, and a set of objects $S = \{s_1, s_2, \ldots, s_l\}$ (e.g., pictures, videos, songs, favorite restaurants, ...). In many domains, we may assume that a *feedback rating* function is defined: $\text{rate} : U \times S \rightarrow R$, where R is a set of allowed rating values (e.g., $R = 1, \ldots, 5$). We assume a bijection between users and nodes in the system,

hence the user u_i denotes both the i-th node and the i-th user. We define $\mathcal{P}(S)$ as the power set of S, i.e. the set of all subsets of S. The function $f : U \rightarrow \mathcal{P}(S)$ maps users to objects of the entire collection (i.e., $\bigcup_{i=1}^{n} f(u_i) = S$). In other words, $f(u_i)$ is the set of items user u_i is related to; for instance, in a file sharing applications $f(u_i)$ can be the set of files u_i is storing, as well as for a video streaming service $f(u_i)$ can be the set of channels or podcasts u_i is subscribed to. In the peculiar domain that we investigate in this paper, an object is mapped to a user if such a user assigned a rate to the given object; i.e., $\forall u_i \in U \wedge \forall s_k \in S$: rate(u_i, s_k) is defined $\Longleftrightarrow s_k \in f(u_i)$.

To take advantage of the power of users relationships, we need to evaluate the *affinity* among users. For this purpose, we first introduce the *affinity function* $Aff : U^2 \rightarrow [0, 1]$, that returns a similarity evaluation for any pair of users.

This function is subject to the following properties:

1. $Aff(u_i, u_j)$ must be computable using information that u_i and u_j can exchange each other without interacting with a third party;
2. $Aff(u_i, u_i) = 1$;
3. $Aff(u_i, u_j) = 0$ means that, due to their knowledge of each other, u_i and u_j have nothing in common;
4. if $Aff(u_i, u_j) > Aff(u_i, u_k)$ then u_i is more similar to u_j than to u_k. This does not imply, in general, any relationships between u_j and u_k.

Affinities can be estimated taking into account many parameters: resources in common, similar behaviors, comparable ratings assigned to items, and so on.

Now we introduce the idea of *"Affinity Network"* that is represented by a graph $G^{\theta} = (U, E)$ such as:

$$e_{ij} \in E \Leftrightarrow Aff(u_i, u_j) \geq \theta. \quad (1)$$

As an instance of a previous successfully study adopting Affinity Networks, we analysed the Gnutella file sharing network in our works [14, 15]. Observe that a P2P system represents a difficult domain with many points in common with the mobile ad-hoc scenario discussed in this paper. In fact, there is not a central repository or a trusted third party that can collect profiles and ratings, and available content description is very poor (sometimes only file names and paths are used), and often misleading. Moreover, prediction accuracy estimation is complicated because users do not usually rate items.

Our previous analysis on the Gnutella network, lead to the identification of the *Small World* phenomenon in those peculiar Affinity Graphs. This means that observed networks showed small mean-shortest paths, and significantly high clustering coefficients. Such evidence depicts the Gnutella network as a set of strongly interconnected clusters, representing spontaneous thematic communities of users sharing kindred files. This has been exploited during the design of *DeHinter* [14], a P2P recommendation module recently implemented and integrated to Limewire [16].

3.1 Affinities on Rating

In a domain where ratings are available and feedbacks are largely returned by the users of the community, we can exploit them in order to find affinities. Even if one of the many well known similarity functions can be defined (e.g., Pearson [17] or Spearman [18] correlations), we use a computational inexpensive measure calculated as the percentage of similar feedbacks that two users assigned to a common set of objects:

$$Aff(u_i, u_j) = \frac{\sum_{s_k \in F_{ij}} \left(1 - \frac{|\text{rate}(u_i, s_k) - \text{rate}(u_j, s_k)|}{\max_R - \min_R}\right)}{|F_{ij}|} \quad (2)$$

where $F_{ij} = f(u_i) \cap f(u_j)$ is the set of files that both u_i and u_j rated, $\max_R$ and $\min_R$ are respectively the highest and the lowest value that can be used to rate an object. Notice that this formula is a generalized version of the one described in [14].

In order to improve readability of the paper and to give an example how recommendations can be triggered according to our approach based on affinity networks, we present a toy domain in the following paragraphs.

Let us suppose a set of three users $U = \{u_x, u_y, u_z\}$, and a collection of five objects $S = \{s_a, s_b, s_c, s_d, s_e\}$. Users rated objects accordingly their own preferences and tastes, that are reported in Table 1.

User u	List of pairs (Object s, rate(u, s)))
u_x	$\{(s_a, 3), (s_b, 2), (s_c, 1), (s_d, 5)\}$
u_y	$\{(s_b, 4), (s_c, 5), (s_e, 5)\}$
u_z	$\{(s_a, 3), (s_b, 1), (s_d, 5), (s_e, 1)\}$

Table 1: Three users rated five different objects

In this case, we have that $\max_R = 5$, $F_{xy} = \{s_b, s_c\}$, $F_{xz} = \{s_a, s_b, s_d\}$, and $F_{yz} = \{s_b, s_e\}$. In Table 2 affinities calculated with Equation (2) are shown.

Aff	u_x	u_y	u_z
u_x	1	0.25	0.92
u_y	0.25	1	0.12
u_z	0.92	0.12	1

Table 2: Affinities between users

If we set threshold $\theta = 0.5$, only users u_x and u_z will be connected in the affinity network. The idea is to spread ratings through affinity links, filtering out feedbacks coming from *distant* users. Our approach aims to exploit word-of-mouth behaviors in such networks, accepting implicit suggestions coming from neighbors and neighbors-of-neighbors with higher probability than receiving positive information coming from users that are some hops away in the affinity network. For instance, neither s_e will be suggested to u_x, nor s_c to u_z.

4. COLLABORATIVE FILTERING VIA AD-HOC COMMUNICATIONS

The mobility scenario has its own peculiar issues. Even if we can assume an "omniscient" server that stores and maintains all the knowledge provided for in our community, we need to grant a high level of autonomy to each host. In fact, in order to make our approach as much general as possible,

we allow the user to walk around, and to *meet* other users at random. Only devices in the proximity can exchange information, but we cannot claim that this provides a natural filtering based on the social contexts, because we are not assuming anything about the reasons why they are there in that moment, in contrast with other authors (e.g., [6]).

When a device is able to connect to the Internet, the user can access the rest of the community, and public his/her own data, profile and preferences. Searches and lookups can be managed by a central directory service, or in a Peer-to-Peer manner (e.g., by flooding or by means of a DHT based overlay network). If each peer u_i accesses the status of its affinity network's neighbors, then it can calculate predictions over an object s_k using one of the many collaborative filtering algorithms known in the literature. Given an affinity graph G^{θ}, let U_i be the set of neighbors of u_i. This means that if u_j is a neighbor of u_i, then they are similar according the affinity function (2), i.e., $Aff(u_i, u_j) > \theta$. Furthermore, node u_i may store all neighbors' referred items and ratings. Hence, if we define the list of feedback ratings of u_i as a set of pairs (object identifier, rating), then the user can keep the following information:

$$Fr_i = \{\forall s_k \in f(u_i) : (s_k, r_{ik})\} \tag{3}$$

where, for the sake of simplicity, $r_{ik} = rate(u_i, s_k)$. In this scenario we can say that peers share a uniform range of ratings and that the status of node u_i is very limited, because many data can be retrieved run-time contacting online neighbors. Therefore, u_i needs to keep references to its neighbors U_i, and list Fr_i.

In our experiments, in order to generate a *prediction* p for an object s_k analyzing ratings available to u_i, we used a slightly modified version of a well known user based nearest neighbor algorithm [19]:

$$p(u_i, s_k) = \overline{r_i} + \frac{\sum_{u_j \in U_i} \text{Aff}(u_i, u_j) \cdot (r_{jk} - \overline{r_j})}{\sum_{u_j \in U_i} \text{Aff}(u_i, u_j)} \tag{4}$$

where $\overline{r_i}$ and $\overline{r_j}$ are, respectively, the average rating of u_i and of neighbor u_j. Notice that the algorithm differs from its classical application for two aspects: (i) Equation (2) is used as affinity function instead of Pearson correlation. This implies that (ii) the number of nearest neighbors can differ for every user depending on how many neighbors are connected in the affinity network. Of course, other statistical and not statistical approaches could be used to generate predictions, but this is out of the scope of this paper. MobHinter is an epidemic protocol that can be applied even when only ad-hoc communications are available during a relatively big period of time. The statistical approaches used to calculate affinities and to generate predictions are not the core of the presented proposal.

If mobile devices cannot access the Internet and they cannot contact directly their neighbors in the affinity network, then the prediction simply cannot be calculated. Our hypothesis is that each node u_i can start with an empty set of neighbors, i.e., $U_i = \emptyset$. We suppose that for each neighbor $u_j \in U_i$, the node stores also its list of ratings Fr_j. Moreover, u_i maintains a list of *known hosts* Kn_i, which is a cache of nodes encountered during u_i's walks. Again, for each node $u_z \in Kn_i$, list Fr_z is stored as well.

When a device u_x finds another device u_y in the proximity, a handshake phase is started: u_x and u_y exchange each other their identities, their feedback ratings lists Fr_x and Fr_y, their neighbors lists U_x and U_y, and their Known hosts lists Kn_x and Kn_y. Then, both ad-hoc nodes can calculate the affinity function in autonomy. If they find that $\text{Aff}(u_x, u_y) > \theta$, then they can add a new node in their neighbors list; i.e., $U_x = U_x \cup u_y$ and $U_y = U_y \cup u_x$. Otherwise, they simply update their known hosts list; i.e., $Kn_x = Kn_x \cup u_y$ and $Kn_y = Kn_y \cup u_x$. Since affinity function is symmetrical, we are sure that both nodes will have the same results.

Trivially, we can foresee an approximately long *cold start* phase. Nevertheless, MobHinter exploits social networking style Word-of-Mouth strategies in order to spread neighborhood information across the discovered links of the affinity network. In fact, if node u_x is not a neighbor of u_y, it can happen that a similarity is found with one of his/her neighbors in U_y, or with one of known nodes in cache Kn_y. It is important to observe that, after the first ad-hoc session, few or no other ad-hoc interactions are needed. This is very important because a node can be out of sight in a few seconds after handshake. All the predictions and the affinities are calculated in autonomy with available data. In the following, we list three different spreading strategies. For improving readability, only actions at u_x side are described, because u_y behaves symmetrically.

PREF u_x keeps Fr_y and discards U_y and Kn_y.
If $\text{Aff}(u_x, u_y) > \theta$, u_x updates his neighbors list accordingly, i.e., $U_x = U_x \cup u_y$.

NEIGH u_x keeps Fr_y and U_y, and discards Kn_y.
Of course, If $\text{Aff}(u_x, u_y) > \theta$, then U_x is updated with neighbor u_y; moreover, $\forall u_z \in U_y$, $\text{Aff}(u_x, u_z)$ is calculated, and if the value is greater than θ, then u_z is referred as a new neighbor of u_x. This is possible because we assumed that each node keeps each neighbor's ratings Fr_z.

KN-NEIGH u_x keeps Fr_y, U_y and Kn_y.
Additionally to actions performed in the previous strategies, u_x looks for neighbors in Kn_y. When a new neighbor u_z is found, U_x is updated. It is possible to evaluate affinities because we assumed that each node keeps each known host's ratings Fr_z.

Predictions are calculated again after each meeting using Equation (4), considering all the neighbors in U_x and their ratings. The three strategies are increasingly more expensive in terms of bandwidth consumption, status storage, and computational resources usage. We want to analyze if the latter strategies are scalable in terms of the number of iterations, and if cold start mitigation worths the overhead. Such a simulative evaluation is provided in the next section, while a discussion on other practical issues is given in Section 6.

5. EXPERIMENTAL ANALYSIS

The best results that could be acquired for a given recommendation algorithm is reached when the system has complete knowledge about objects, users and users' preferences usually expressed by feedback ratings. Devices using MobHinter move into a disconnected environment, where epidemic dissemination of information is critical before accurate recommendations can be triggered to the user. This process

is simulated through a set of iterations that mimic meetings between users. This evaluation is made of two phases:

1. Run the adopted user-based neighbor recommender system in a simulated fully connected scenario, in order to produce a reference model and to find optimal prediction results;

2. Run the same recommendation system in a fully disconnected environment where only random ad-hoc meetings are allowed.

Aims of this evaluation is three-fold: (1) to estimate how many interactions are needed before converging to optimal values, accordingly to the three different strategies described in the previous section; (2) to estimate overhead w.r.t. the capacity of a modern mobile device; and (3) to analyze the scalability of the approach.

For all our experiments we refer to the dataset of MovieLens developed at the University of Minnesota. It is composed of a number of randomly selected users enough to obtain 100,000 ratings from the database (considering only users that had rated 20 or more movies). It can be represented as a sparse user-movie matrix with 943 users and 1,682 movies, with the rating values represented in the matrix as an integer value from 1 to 5. The sparsity level is, therefore, $1 - \frac{100.000}{943 \cdot 1682}$, which is 0.937.

5.1 Evaluation Methodology

In order to evaluate the accuracy of the predictions, we adopted two well known accuracy metrics: the *Mean Absolute Error* (MAE) and the *Root Mean Square Error* (RMSE). MAE measures the average absolute deviation between a predicted rating and the user's true rating. RMSE, that squares the error before summing it, emphasizes on large errors, that is, it amplifies the contributions of egregious errors, both false positives ("*trust busters*") and false negatives ("*missed opportunities*").

Accordingly to Sarwar et al. [20], a recommender system based on nearest neighbor algorithms may be unable to make any item recommendations for a particular user, if the data set is too sparse. As a result the accuracy of recommendations may be poor, and the estimation of preferences may well be undefined, when asked to estimate a preference for an item for which no preference is expressed in the user neighborhood. This situation occurs with high thresholds that limit too much the neighborhood. In this case, if we consider only estimations that are defined by the algorithm, we can have an unfair good result. To go around this problem, we decided that when the estimation is undefined by the algorithm, the final resulting estimation is the average of other preference scores from that user (we use a sort of "*default voting*" reflecting neutral preferences of the user).

To make an evaluation that includes all the data set, we used the k-fold cross-validation, with $k = 10$. In this evaluation, the ratings set is partitioned in 10 samples. Of the 10 samples, a single sample is retained as the *testing set*, and the remaining 9 samples are used as *training set*. This validation is repeated 10 times, ensuring that all ratings are used for testing. The results are then averaged to produce a single estimation.

The framework used was the Taste library[1], a collaborative filtering engine for Java. It is a very flexible framework, with most of the common collaborative filtering approaches available. It also provides a way to evaluate the accuracy of recommendation algorithms, including metrics as MAE and RMSE.

[1]http://taste.sourceforge.net

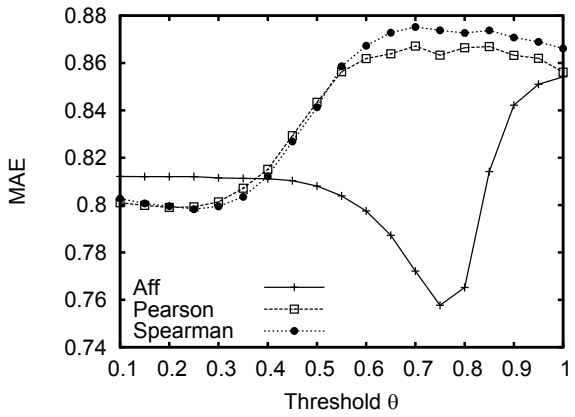

Figure 1: MAE in a complete knowledge scenario

5.2 Results in the Reference Scenario

We evaluated the prediction of ratings for the entire MovieLens data set, with the assumption that each node can find its neighbors in the affinity network and run the user-based neighbor algorithm as reported in Equation (4). Differently with the standard version of such formula, we considered neighbors of u_i that satisfy Equation (1), and not a constant number of nearest neighbors. We tested three different similarity functions, including Pearson correlation, Spearman correlation and the affinity function defined in Equation (2). The tested thresholds were in a scale from 0.1 to 1.0. The results for MAE are shown in Figure 1, and the results for RMSE have a similar behavior, with the best results for Spearman and Pearson correlation at 0.25, and the best results for function in Equation (2) at 0.75.

We found that the affinity function defined in Equation (2) produces best results (MAE of 0.7577 and RMSE of 0.9617) in correspondance with the optimal threshold $\theta = 0.75$. After analysing the optimal scenario made of a static fully connected affinity network, and with completely available knowledge, we estimated MAE and RMSE lower bounds, that are terms of reference for the ad-hoc experimental setup.

5.3 Simulation of Ad-Hoc Scenario

In the previous section we individuated the correlation function and the threshold θ that minimize MAE and RMSE in the reference scenario for a given user-based neighbor recommender system. In the following, we will present a simulative framework in order to model the dynamics of a population of mobile devices that interact each others through ad-hoc meetings. In this way, they share their local knowledge for producing personalized advices. We assume that each device starts without any a priori knowledge about the domain or the population of users. Afterward, it starts a *discovery phase* during which the device explores the world looking for new nodes. During this explorative phase we suppose that it does not access to the Internet or to any other remote server. Its only knowledge comes from ran-

dom meetings. The device is able to exchange data only through ad-hoc communications.

The simulation is composed by a sequence of *iterative steps* that model random meetings between users. If n is the number of users under observations, at each step, we simulate $n/2$ random meetings between different pairs (u_i,u_j) of peers. At last, for a node is not possible to meet more than one other host in a single iterative step.

The state of a user u_i, i.e., the amount of data the user has to store, is composed by two sets: (1) the set of ratings Fr_i (see 3) and the set of neighbors U_i. Of course, we have that $U_i = \emptyset$ at the beginning of the simulation. When users u_i and u_j meet each other, they can run one of the three selected strategies: *PREF*, *NEIGH* and *KN-NEIGH*.

In our simulation, we performed 80 iterative steps and, at each iteration, we computed for all *PREF*, *NEIGH* and *KN-NEIGH* strategies the MAE and RMSE accuracy metrics, under the evaluation framework discussed in Section 5.1. In all the simulations, we used the affinity function defined in Equation (2) with threshold $\theta = 0.75$. Figure 2 and Figure 3 show observed results, respectively, for the MAE and RMSE metrics.

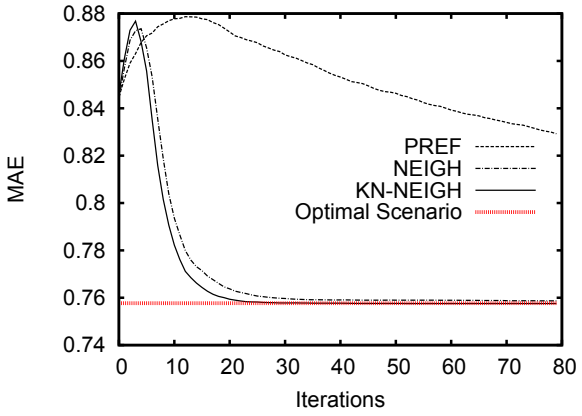

Figure 2: MAE estimation in the ad-hoc scenario

The *Optimal Scenario* line marks the value reached in the reference fully connected model (i.e., 0.757 for MAE and 0.961 for RMSE). In both figures is evident that the *PREF* strategy converges very slowly toward the lower bound because of the spreading of information is relied only to direct meetings. Exchanging the neighbors list U_i allows an epidemic form of ratings distribution that strongly improves accuracy of prediction. In fact, for the *NEIGH* strategy we can reach results comparable to the complete knowledge one in 50 iterative steps (e.g., MAE=0.768 and RMSE=0.99). Moreover, sharing of known hosts list Kn_i, considered with strategy *KN-NEIGH*, allows to maximize the epidemic diffusion of data and permits to obtains good results after only 25 random meetings (e.g., MAE=0.758 and RMSE=0.962).

The process of neighbors discovery is showed in Figure 4. At each iteration, we computed the average number of neighbors stored by users, i.e., $avg(|U_i|)$, and the standard deviation *stdev* (for clarity reasons in Figure 4 we show only *stdev* for the *KN-NEIGH* strategy).

Very interestingly, even if there is an appreciable difference between *KN-NEIGH* and *NEIGH* strategies, this is not comparable to the difference observed between *NEIGH* and *PREF* approaches. Our interpretation is that, as observed in many social networking analysis, the neighbors of my neighbors are likely to be my neighbors (also known as the *triangulation* property). Hence, epidemic distribution of ratings is greatly enhanced if information are spread though affinity links. Conversely, known hosts that are not my neighbors, have limited probability to be neighbors of my neighbors, especially in a sparse dataset like MovieLens. Nevertheless, if two users take into account Kn lists, trust busters and missed opportunities can be reduced in fewer steps.

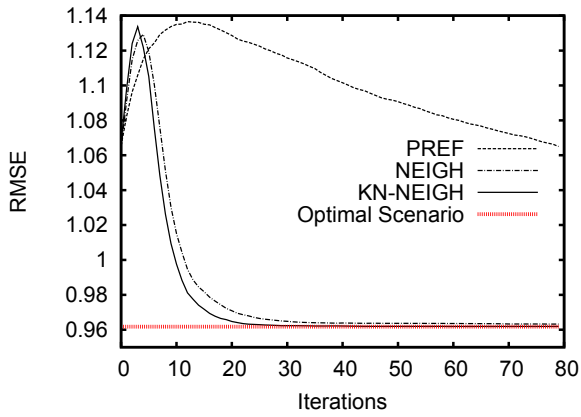

Figure 3: RMSE estimation in the ad-hoc scenario

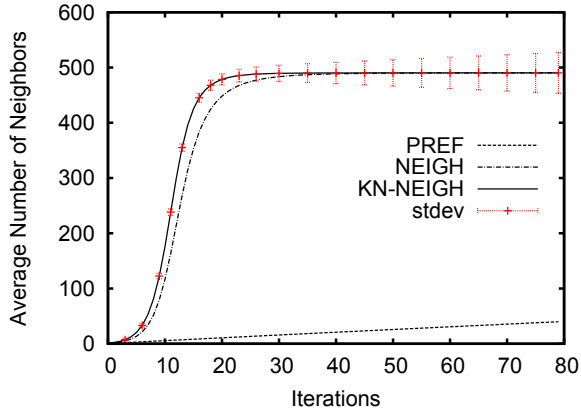

Figure 4: Average number of neighbors in the ad-hoc scenario

5.4 Overhead and Scalability considerations

Considering a simulative scenario with a population of 943 users, a collection of 1,682, and 80 iterative steps that emulate 80 sequential random meetings for each user's device, Figures 2 and 3 show also how many ad-hoc communications are needed before predicting approximately accurate recommendations. In fact, after 20 iterative steps (i.e., each device has met about 2% of the entire population), on average, we have that $0.757 < \text{MAE} < 0.763$ and $0.961 < \text{RMSE} < 0.97$ when *NEIGH* strategy is used, and that $0.757 < \text{MAE} < 0.759$ and $0.961 < \text{RMSE} < 0.964$, when *KN-NEIGH* is adopted instead. Furthermore, after 50 iterative steps (i.e., each device has met about 5% of the

$\|u_i\|$	10 bit
$\|s_k\|$	11 bit
$\|rate(u_i, s_k)\|$	3 bit

Table 3: Parameters for the Estimation of the Spatial Overhead

Step	*PREF*	*NEIGH*	*KN-NEIGH*
20	3895	18735	46189
30	5750	32462	58061
40	7605	37471	60102
50	9460	39511	60658

Table 4: Status overhead behaviors (in bytes)

MovieLens population), *KN-NEIGH* strategy makes MAE and RMSE converge to optimal minimum values.

Hence, in order to understand if the overhead is viable with modern mobile devices in terms of their available capacity and to evaluate scalability of the system, we need to estimate step by step the average device's status in terms of memory occupancy. Focusing again on the MovieLens dataset, we can define $|u_i|$, $|s_k|$ and $|rate(u_i, s_k)|$ as, respectively, the size in bit of a user's identifier, an object identifier and a rate (see Table 3).

The status of a MobHinter node u_i is made of the following elements: Fr_i, U_i, and Kn_i. The neighbors list and known hosts cache grows differently accordingly to the adopted strategy. It is important to recall that for each node u_j in U_i or in Kn_i, we need to store the node's identifier (i.e., $|u_j|$ bit), and ratings list Fr_j (i.e., $\forall(s_k, r_{jk}) \in Fr_j$, the device consumes $|s_k| + |rate(u_i, s_k)|$).

Focusing on iterative steps from 20th to 50th, for each different strategies, we have an average status for device as in Table 4.

Of course, *PREF* is the less expensive in terms of memory consumption, but the reader should observe that after 50 iterative steps, the average status of MobHinter node behaving accordingly to *KN-NEIGH* is about 60 KB, which is widely practical for actual mobile devices. Furthermore, this status size can represent an upper bound, because (as explained in the following section) we avoid to let indefinitely grow the lists; in fact, in the case study we showed that after an average of 50 random meetings, recommendations are accurate as well as in the optimal scenario.

6. DISCUSSION

In the previous sections we proved that MobHinter converges quickly to optimal recommendation predictions even if only random ad-hoc meetings are considered. MobHinter's field of applicability is very wide, and it includes movie selection, tourist attraction suggestion, personalized restaurant advices, and so on. Real world applications have pros and cons that go beyond simulative frameworks, and that cannot be realistically considered in a simplified model. In the following, we state some of the most relevant issues that may arise.

Synchronization: Even if we proved the practicality of our approach in a very extreme domain where nodes meet each other only using ad-hoc connections, we can realistically consider the existence of a remote service that can be accessed when Internet is available. During synchronization with such a service, a node can merge information collected during the ad-hoc phase with new information related to geographically remote users. When online, the device is able to refine the personalized predictions previously obtained, for example, accessing to content-based data repositories, downloading more specific information about items and users, recalculating affinities according to more accurate data. Even if very relevant to a practical implementation of MobHinter, these engineering issues are out of the scope of the paper.

Privacy and profile control: A MobHinter node has complete control over its user's profile and over recommendation calculations and predictions. In fact, every device collects all the information it needs for estimating ratings for unknown objects, and it has to exchange such information with other nodes. This greatly protects user's privacy, because even during a synchronization with a remote service, it has to pull all the data needed for calculation, by means of a (decentralized) directory system, e.g., for example, an user can query the system looking for potential neighbors that matches his preferences according a given affinity threshold. This can be executed, for example, by flooding the searches, or using a Distributed Hash Table overlay network that allows structured and scalable Peer-to-Peer systems. During this process, the user can mask himself behind a presented identity with no relationships with the real entity. The identity is characterized in terms of a set of preferences, and she/he can decide which information to be publicly available.

Pseudonyms - many identities - one entity problem: The reader can question if a massive usage of pseudonyms can add noise to the recommendation system. Even if this problem must be carefully analyzed before estimating the real degree of its potential impact, we can adopt a straightforward solution for filtering out redundant identities. Because Equation (4) is not significantly affected by popularity factors, a device can simply discard neighbors and/or known hosts with identical ratings lists, assuming that this is an evidence of a replicated identity.

Lists size issue: As shown in Figure 4, the size of the neighbors list, as well as the known nodes lists, can be very large in some cases, and in a real scenario with undefined number of users it can even grow indefinitely. In Section 5.3 we discussed how introduced overhead is not a problem with modern mobile devices, but anyway the problem can be tricky in some cases. This issue can be avoided just introducing a smart strategy for limiting the size of these lists, without removing very relevant references. Some preliminary results in this direction show that limiting the size of the neighbors list does not present significant implications in accuracy loss. In order to maintain the performance of the algorithms, it is necessary to choose a good strategy to discard neighbors when the list is at its maximum size, and recommendation predictions are very close to optimal accuracy. One good strategy for neighbors list is to keep the nodes with higher affinity value, discarding the hosts that have the lowest values. About the known hosts list, it should have limited size too, but as the cache of known hosts is maintained mainly to mitigate the cold start problem, then the adopted strategy is less critical. However, it is possible to compare behaviors of the system when different strategies are implemented, e.g., LRU, MRU, random deletions, and so on. For brevity reasons, this analysis has not been included in this version of the paper.

7. CONCLUSIONS AND FUTURE WORK

In this paper we focused on collaborative filtering techniques in mobile ad-hoc networks where spontaneous affinity relations are derived and exploited in order to push personalized suggestions by way of direct meetings between users. We referred to a scenario where devices exchange information with users in the proximity without accessing to any remote online directory services. We proposed epidemic collaborative strategies to spread ratings over self-organized communities of users. We simulated the ad-hoc environment and we found that the proposed approach converges very quickly to the prediction accuracy measured in the reference domain with an acceptable overhead.

The implementation and the evaluation of a prototype for the Google Android[2] platform is future work in order to experiment the *MobHinter* framework in a real scenario.

Acknowledgment

This work has been partially supported by the MIUR within the "PROFILES" project (PRIN) and by the *Torino Wireless Foundation* within the "RD-PVR" project. Panisson has been supported by the *"WWS Mobility Program"* founded by the CRT Foundation.

8. REFERENCES

[1] M. Reardon, "Mobile communities could fill 3g pipes," 2006. [Online]. Available: http://news.zdnet.com/2100-1035_22-6058001.html (last access: 2007/11/5)

[2] A. Tveit, "Peer-to-peer based recommendations for mobile commerce," in *WMC '01: Proceedings of the 1st international workshop on Mobile commerce*. New York, NY, USA: ACM Press, 2001, pp. 26–29.

[3] B. N. Miller, J. A. Konstan, and J. Riedl, "Pocketlens: Toward a personal recommender system," *ACM Trans. Inf. Syst.*, vol. 22, no. 3, pp. 437–476, 2004.

[4] B. J. Mirza, B. J. Keller, and N. Ramakrishnan, "Studying recommendation algorithms by graph analysis," *J. Intell. Inf. Syst.*, vol. 20, no. 2, pp. 131–160, 2003.

[5] S. Castagnos and A. Boyer, "Modeling preferences in a distributed recommender system," in *User Modeling*, ser. Lecture Notes in Computer Science, C. Conati, K. F. McCoy, and G. Paliouras, Eds., vol. 4511. Springer, 2007, pp. 400–404.

[6] A. de Spindler, M. C. Norrie, and M. Grossniklaus, "Collaborative filtering based on opportunistic information sharing in mobile ad-hoc networks," in *OTM Conferences (1)*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 4803. Springer, 2007, pp. 408–416.

[7] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips, "Tribler: a social-based peer-to-peer system." *Concurrency and Computation: Practice and Experience*, vol. 20, no. 2, pp. 127–138, 2008.

[8] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips, "Tribler: A social-based peer-to-peer system," in *IPTPS*, no. 2006-002, feb 2006. [Online]. Available: http://pds.twi.tudelft.nl/reports/2006/PDS-2006-002/PDS-2006-002.pdf

[9] B. Xie, P. Han, and R. Shen, "Pipecf: a scalable dht-based collaborative filtering recommendation system," in *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. New York, NY, USA: ACM Press, 2004, pp. 224–225.

[10] P. Han, B. Xie, F. Yang, and R. Shen, "A scalable p2p recommender system based on distributed collaborative filtering." *Expert Syst. Appl.*, vol. 27, no. 2, pp. 203–210, 2004.

[11] T. Oka, H. Morikawa, and T. Aoayama, "Vineyard : A collaborative filtering service platform in distributed environment," in *Proc. of the IEEE/IPSJ Symposium on Applications and the Internet Workshops*, 2004.

[12] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl, "Movielens unplugged: Experiences with an occasionally connected recommender system," in *2003 Conference on Intelligent User Interfaces (IUI'03)*. ACM, 2003. [Online]. Available: http://citeseer.nj.nec.com/584995.html

[13] K. Chorianopoulos, "Personalized and mobile digital tv applications," *Multimedia Tools Appl.*, vol. 36, no. 1-2, pp. 1–10, 2008.

[14] G. Ruffo, R. Schifanella, and E. Ghiringhello, "A decentralized recommendation system based on self-organizing partnerships." in *Networking*, ser. Lecture Notes in Computer Science, vol. 3976. Springer, 2006, pp. 618–629.

[15] G. Ruffo and R. Schifanella, "Evaluating peer-to-peer recommender systems that exploit spontaneous affinities," in *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2007, pp. 1574–1578.

[16] A. Panisson, G. Ruffo, and R. Schifanella, "X-hinter: a framework for implementing social oriented recommender systems," in *Hypertext 2008*. New York, NY, USA: ACM Press, 2008.

[17] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*. New York, NY, USA: ACM Press, 1994, pp. 175–186.

[18] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[19] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*, 2007, pp. 291–324. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72079-9_9

[20] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *WWW '01: Proceedings of the 10th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2001, pp. 285–295. [Online]. Available: http://dx.doi.org/10.1145/371920.372071

[2]http://code.google.com/android/