Noname manuscript No.
(will be inserted by the editor)

Recommending Items in Pervasive Scenarios: Models and Experimental Analysis

Luca Becchetti · Ugo Colesanti · Alberto Marchetti-Spaccamela · Andrea Vitaletti

Received: date / Accepted: date

Abstract In this paper, we propose and investigate the effectiveness of fully decentralized, collaborative filtering techniques. These are particularly interesting for use in pervasive systems of small devices with limited communication and computational capabilities. In particular, we assume that items are tagged with smart tags (such as passive RFIDs), storing aggregate information about the visiting patterns of users that interacted with them in the past. Users access and modify information stored in smart tags transparently, by smart reader devices that are already available on commercial mobile phones. Smart readers use private information about previous behavior of the user and aggregate information retrieved from smart tags to recommend new items that are more likely to meet user expectations. Note that we do not assume any transmission capabilities between smart tags: Information exchange among them is mediated by users' collective and unpredictable navigation patterns.

Our algorithms do not require any explicit interaction among users and can be easily and efficiently implemented. We analyze their theoretical behavior and assess their performance in practice, by simulation on both synthetic and real, publicly available datasets. We also compare the performance of our fully decentralized solutions with that of state-of-the-art centralized strategies.

Keywords Decentralized recommendations; Collaborative filtering; Resource constrained devices

1 Introduction

Nowadays, the pervasive deployment of tiny devices with minimum storage and limited or no computational capabilities appears a realistic perspective; one major obstacle are

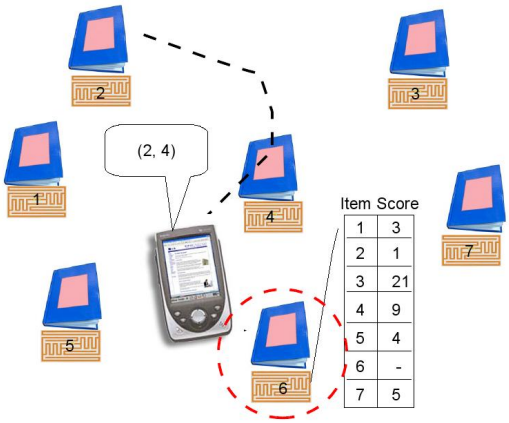
L. Becchetti · U. Colesanti · A. Marchetti-Spaccamela · A. Vitaletti
Dipartimento di Informatica e Sistemistica "A. Ruberti"
SAPIENZA Universitá di Roma
via Ariosto 25
00185 Roma, Italy.
Tel.: +39-06-77274025
Fax: +39-06-77274002
E-mail: {becchett,colesanti.alberto,vitale}@dis.uniroma1.it

the strict energy constraints of battery powered devices. We refer to a class of passive devices (i.e. not powered by batteries) that has emerged in the last decade, the most prominent examples being RFID and NFC tags. An RFID tag is an object that can be applied to or incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves. Passive RFID tags have no internal power supply and draw power from the radio waves emitted by the reader. Near Field Communication or NFC, is a short-range high frequency wireless communication technology which enables the exchange of data between devices within short distance aimed at usage in mobile phones. The limited costs and the pervasiveness of these devices are paving the way for new pervasive solutions: mobile ticketing in public transport, mobile payment, smart shopping, social applications.

In this paper we consider fully decentralized collaborative filtering strategies for item recommendation in pervasive systems. For example, the NFC consortium proposes smart posters for shopping by tagging items of interest posted in billboards, or any other form of advertising, with a passive tag, from which a user can exchange data by touching it with her NFC-enabled handset; namely, the user can buy the item associated with the tag and receive information on the item or even recommendations on other items of potential interest on the basis of the succinct information obtained interacting with smart posters. The distinguishing features of the above technologies entail a model of decentralized computation in which low capability devices observe a local stream of events and have to maintain summary information about overall system behavior, while obeying stringent memory, computational and communication constraints. In this scenario, distributed, local algorithms appear a natural choice to address computational, communication and storage restrictions of the scenarios outlined above.

The importance of recommender systems has been widely recognized in e-commerce systems as a tool to suggest products to customers, providing relevant information in shopping (e.g. Amazon, eBay). In order to recommend new items three main approaches have been proposed: collaborative filtering, content-based filtering and hybrid methods [12]. We consider Collaborative Filtering (CF for brevity) that classifies users and products in terms of their past interactions.

We assume that items of interest are advertised by smart tags (e.g., RFIDs or NFCs) distributed in an area, e.g., a city. In the sequel we will use the terms *smart tag* and *item* interchangeably and the the term *smart reader* to denote a smart tag enabled reader device (e.g. an NFC enabled smart phone). Each user is characterized by a (unknown) ranking of items, describing her preferences; a smart reader stores her history (e.g., the set of items previously visited by the user) during the visit and has some computational capability. When a user interacts with item i the user's smart reader reads details on the item. We also assume that i stores a suitable summary of the histories of users that visited i in the past: smart readers interact with smart tags, by *transparently* (to the user) reading and updating the information stored by the latter. We stress that we do not assume any transmission capabilities between smart tags, which are assumed to be passive devices. When item i is visited by user j, her smart reader can recommend a new item (or a set of items) of potential interest to j, using current summary at i and j's history. The recommendation of an item is good if the proposed item is likely to meet j's preferences. The above scenario is technologically realistic and it is closely related to architectures proposed for smart shopping carts [23] and smart shelves [15]. We remark that it also complies with privacy issues, since



Scenario: every item has an associated smart tag with a unique integer ID. The user is visiting item 6 after visiting items 2 and 4. The smart reader uses aggregate statistics concerning item 6 and user's history to assign scores to items other than 6 and to provide a recommendation (e.g., item 1 has score 3, while item 3 has score 21 and is thus the top item in the summary).

Fig. 1 Exemplifying scenario: a smart library

only aggregated information is disclosed from which it is not possible to infer private information concerning specific users.

Results of the paper

Our main result is to show that, even under the stringent constraints outlined above, simple heuristics allow to effectively profile users and provide good recommendations in the scenario described above. The recommendation algorithm we consider is simple: upon visiting item i, user j is recommended one item (or a subset of items) scoring highest among those not yet visited by j. The core of the whole problem is defining scores that i) can be efficiently estimated and updated locally by the smart reader upon visiting new items and ii) that effectively reflect the users' unknown preferences. Note that scores and rankings can statistically depend on users' visit patterns in complex ways. We tackle this issue by defining suitable models of user behavior.

On the theoretical side, we provide asymptotically tight lower and upper bounds on the number of examples required by the recommendation algorithm to compute good estimations of item scores and thus provide good recommendations. While theoretical analysis gives bounds that might be unfeasible in practical applications, our experiments compare the performance of our algorithms with that of a centralized, state-of-art recommendation algorithm that knows the overall system history. The tests are based on both synthetic data and real data sets provided by Netflix, a popular on-line DVD rental service; they show that the performance of our algorithm is very close to that of the centralized one, in terms of standard metrics normally adopted to assess the performance of Collaborative Filtering algorithms.

We observe that our algorithms can be considered as an application of stigmergy to recommendation algorithms. Stigmergy is a form of self-organization where traces left in the environment by the action of agents trigger the execution of subsequent actions, by the same or a different agent, thus allowing spontaneous and indirect coordination between agents [8]. Stigmergy has been exploited in tracking objects tagged by RFIDs [26] and routing messages in mobile wireless ad-hoc networks [32]; however, to the best of our knowledge, this is the first paper presenting a recommendation system for pervasive systems based on this interaction paradigm.

Related work.
In the last years recommendation systems have been recognized as an important research area and much work has been done both in industry and academia on developping new approaches. As a result, a number of recommender applications are used in a variety of e-commerce systems, e.g., for recommending books by Amazon [25, 1], movies by MovieLens [28], DVDs by Netflix [2]. A survey of the main approaches to recommendation applications can be found in [4]. On the other hand, future mass deployment of pervasive networks opens the possibility of new scenarios for recommendation systems. For example, we refer to MyGROCER [23], a recent proposal for a ubiquitous computing environment for supermarkets based on a smart shopping cart that exploits shopper's identity to provide a personalized service. As observed in [34], improvements are necessary to extend recommendation systems to new scenarios, *"including [...] products to purchase in a store made by a smart shopping cart"*.

Collaborative Filtering allows to extract useful information without requiring cooperation and identification of users and, for this reason, has emerged as the most effective approach to tackle the privacy issues and for mass deployment. We briefly review the main related results, referring to [4, 9, 20] for a thorough survey of literature on Collaborative Filtering. One of the main approaches to Collaborative Filtering, adopted in [9, 16], relies on the computation of similarity indices among items and on using them for prediction of user likely preferences. Namely, an $n \times m$ item-user matrix R stores binary information on users' choices: $R(i, j)$ is 1 if the j-th customer has purchased item i and zero otherwise. Using matrix R, items are classified and the user is suggested a set of items similar to items in U where, intuitively, two items are similar when most users that find one interesting tend to find the other relevant as well.

In many cases users can clustered in groups: two users in the same group have similar preferences. Singular Value Decomposition (SVD) [27] was shown to be useful to cluster users; we remark that SVD is computationally intensive and requires centralized information and often requires additional conditions for its applicability that are not met in practical cases. In [7, 17] the goal is to approximately recover the latent structure of users' preferences. However, proposed solutions require extensive data on each user and a centralized, expensive computation. Kumar et al. [22, 24] study the off-line problem where preferences are identified with past choices; items are clustered and each user has a probability distribution over clusters: a user first chooses a cluster by her distribution and then chooses a product uniformly at random from that cluster. The goal is to recommend an item from the user's preferred cluster. A different approach is based on the use of ranking-based evaluation measures for the evaluation of regression models [31]; this is motivated by the fact that ranking can be the main underlying goal.

The contributions above consider centralized settings. Distributed recommendation strategies have also been considered in the recent past. In [11] the authors propose to partition item-user matrix R into smaller matrices: each new smaller matrix contains the ratings of all the users on the items belonging to a certain topic or domain, e.g., the movies having a particular genre. However, they assume that these systems can communicate with each other using a simple request/response protocol. In [14], the authors explicitly consider the limitation imposed on CF by mobile devices, and incrementally update R by connecting near-by devices over Bluetooth without the need for constant connection to a central server. In [6], a distributed solution is proposed to on-line recommendation in which a user is in search of an item she likes; the algorithm is randomized: at each step, the user either selects an item uniformly at random or asks

another user about her preferences. Although the above solution are distributed, we remark that active cooperation between users is required. A similar remark applies to [5] where the goal of the users is to learn their complete preference vector (approximately) while minimizing the cost of probing.

Distributed CF has also been considered for P2P networks. We briefly discuss two representative approaches; [36] considers recommendation in P2P file sharing systems suing a Distributed Hash Table to allocate the database of user past transaction among the nodes of the network. [35] uses a similar approach, but the storage and update of user information is performed differently and is determined by the navigation of users. Both strategies require explicit communication among nodes of the network to maintain information on past user transactions.

2 Models terminology and notation

We consider a set of n *smart tags*, passive devices tagging *items* in a shop (e.g. a library) or in a museum; every item has a unique integer identifier $i \in [n]$, where $[n] = \{1, \ldots, n\}$. To make terminology simpler, in the sequel we use the term item extensively when referring to the smart tags attached to them, since the scenario and the solutions we consider are oblivious to the nature of tagged items. There are m *users* and each user visits the shop over time carrying a *smart reader*, i.e., a device able to read and update information stored at smart tags. Every user j enters the system, visits a subset of the items and then leaves the system. We call this a *session*. In general, by *visiting an item* we mean an active and detectable interaction between a smart reader and the smart tag tagging an item (e.g., purchasing a tagged item).

The identities of users are not stored, hence multiple visits of the same user to the shop are not individually tracked. However, we emphasize that information about multiple visits of the same user is stored in aggregate form at smart tags, as we shall see further. We assume that a user visits each item at most once during her permanence in the system, since this captures typical visiting patterns in many cases. The alternative model in which a user may perform multiple visits to the same item during the same session can be of interest in different scenarios and can be more easily modelled using random walks (see, e.g., [19]). Note also that we assume that computation entirely occurs at the smart reader (e.g., a phone-like device), whereas smart tags only store the outcome of the computation. For this reason, we think of smart readers as passive devices (e.g., RFIDs), which are not battery operated. The results we present also apply to scenarios in which smart tags play an active role in computation. Clearly, in this case energy issues at smart tags can be no longer neglected.

Modelling user behavior in the system entails two aspects: i) describing the way in which users select items of potential interest and ii) the order of visits of items that determines the way in which information about users' past visits is spread across the pervasive system.

i) Cluster based item selection. We assume that every user $j = 1, \ldots, m$ has an associated vector $\mathbf{w}(j) = (w_{1j}, \ldots, w_{nj})$, w_{ij}, called *user profile* in the sequel, describing j's potential interest for item i. Note that $\mathbf{w}(j)$ is unknown to the system. In particular, $w_{ij} \leq 1$ gives the *absolute probability* that user j will select item i. Hence, $\sum_{i=1}^{n} w_{ij} \neq 1$ in general. The selection of items visited by user j proceeds as follows: for every $i = 1, \ldots n$, j visits item i with probability w_{ij}, independently of other items

and of other users. Note that, by this definition, there is a user-dependent non zero probability that a user will visit no items.

Following a common assumption in the literature [4,22,24], we assume items are partitioned into disjoint *clusters*, $C_1, C_2, \ldots, C_s$, e.g., corresponding to different topics or categories. We further assume that, for item i and user j, w_{ij} satisfies $w_{ij} = p_{kj} w_i$, $i \in C_k$; p_{kj}, the *weight* of cluster k for user j, denotes the preference of user j for items in cluster C_k, and w_i, the *cluster weight* of item i, denotes the *popularity* of item i within cluster C_k, assumed to be the same for all users (i.e. $p_{kj} = p_{kj'}$ if j and j' belong to the same cluster). Put simply, this means that, if items were books for example, our model states that different users may have a different degree of preference for the topic 'Science fiction', but their preferences for science-fiction books mainly depend on item popularity. Note that, since p_{kj} is the absolute probability that user j visits cluster C_k, $\sum_{k=1}^{s} p_{kj} \neq 1$ in general.

We also assume that each item is aware of the cluster it belongs to. Namely we assume that each smart tag contains, among others, a unique label identifying the cluster it belongs to. Though a restriction, this assumption is perfectly realistic is many scenarios, such as the smart poster application we consider or also a bookshop, an e-shop or a supermarket, where items are (physically or virtually) arranged in groups defined by some notion of similarity (e.g., topic or use).

ii) Order of visit. We assume a *weighted visit model*. Namely, if S is the set of possible items and user j has already visited a subset X of items, then the probability that the next item visited is i, $i \in (S - X)$, is $w_{ij}/(1 - \sum_{r \in X} w_{rj})$ (for items in X this probability is 0); note that this probability is proportional to w_{ij} and depends on the sum of the total weights of the already visited items. [1] It follows that, the probability that a user visits a given subset of the items follows a distribution that is a special case of Fisher's noncentral hypergeometric distribution[2] [18].

Note that, while different users' visits are independent, the next items visited by a user clearly depends on the items he/she previously visited induced by her preferences. To consider the bookshop example, many people are likely to be first attracted by popular, recently published books in their fields of interest. Experimental evidence discussed in Subsection 5.3 strongly supports this assumption, at least for the Netflix recommendation dataset.

Remarks. The above model is intended to strike a balance between simplicity and soundness. It is clear that this choice brings some simplification with respect to the scenarios of potential interest. The recommendation based on items' popularity can be sensitive to changes of users' visit patterns over time. Also, assuming that rankings inside clusters only depend on items' popularities may be unrealistic in some scenarios. Furthermore, visit patterns might depend on different (e.g. geometric and physical) constraints, such as the (physical or virtual) structure of the shop. Finally, in the description above, we have "artificially" separated the selection and the visit phases, since this way of looking at the model is useful in the analysis.

[1] Note that, consistently, $\sum_{i \in S-X} \frac{w_{ij}}{1-\sum_{r \in X} w_{rj}} = 1$.

[2] Fisher's noncentral hypergeometric distribution arises in a "sampling balls from urns" model, in which each urn has an associated color and contains a number of balls of that color. Furthermore, balls are extracted according to weights that only depend on their colors. Our case is the special one in which every urn contains exactly one ball.

A few comments about independence of user visits are also in order. An aspect that is not taken into account in our model is the effect of recommendations themselves on future user behavior. We note that this is in fact a general problem in collaborative filtering and other recommendation approaches. Tackling this aspect easily brings to hardly tractable models. Furthermore, it can be hard to assess the soundness of such a model on publicly available datasets, since these (such the Netflix one) typically provide no information about the impact of recommendations possibly provided by the system on user behavior.

Another important point is that the structure itself of the shop may in many cases "shape" the probabilistic distributions of user visits, intuitively making them look more "similar". Assessing whether this introduces dependences in user visit patterns is in general hard and problem-dependent. We assume independence for simplicity, at the same time noting that this does not preclude the possibility of similar trends in user visiting patterns, induced by the shop structure.

We conclude by noting that experimental evidence on a real Netflix dataset, shows that at least in the scenario the data refer to, even this simple model captures important trends. For example, experimental evidence reported in Section 5 supports the model we consider, showing that a significant correlation exists between the order of users' visits and items' popularities within the same movie cluster.

3 Recommendation algorithms

As we have already observed, new items are recommended to the user when she "visits" an item, e.g., as her smart reader reads the information contained in the smart tag attached to an item picked up in the shop. As we pointed out earlier, recommendation is performed locally by the smart reader itself, on the basis of information contained in the smart tag and of current user's history information stored in his/her smart reader. In the rest of this section, we address the following points: i) which is the nature of the information carried by the user and by smart tags; ii) how this information is maintained and updated; iii) how it is used to predict a user's likely preferences; iv) the rule followed to provide recommendations. We next discuss points i), iii) and iv) above. Point ii) poses most challenges; it will be briefly outlined in this section and addressed in detail in Section 4.

User histories and item summaries. We assume that each user carries a vector $\mathbf{H}(j)$ (called *user history in the sequel*), whose components contain information about items visited by j earlier in her visit. More in detail, if j visited s at time t, $\mathbf{H}_s(j)$ contains $F_s(t)$, i.e, the number of users that visited s until time t, and the identifier of the cluster s belongs to. On the other hand, each smart tag also stores aggregate information about past user visits. More in detail, consider a generic item r belonging to cluster C_k. At any time t, r stores $F_r(t)$ and, for every other $s \in C_k$, r stores $N_{sr}(t)$, i.e., the number of users that visited r after visiting s. As we see more in detail in Section 4, this information is read by smart readers to update their histories, while r's summary is updated as new users visit it. This information is propagated among cluster by stigmergy, as we discuss more in detail in the next section.

Summarizing, each item maintains a counter of the number of users that visited the item in the past, its cluster identifier and, in the worst case, a counter for every other item in its same cluster. We assume each smart tag stores this information, that thus

becomes accessible to users as they visit the corresponding items. Note also that this information can be stored at every item (i.e., its smart tag) using a constant number of bits. We call the aggregate information stored at an item its *summary*. Note that items maintain no private information about specific users.

It is clear that, since each user only visits a small subset of a potentially large set, $\mathbf{H}(j)$ should be stored in compact form in a practical implementation, which we actually do. We only consider the definition given above to the purpose of simplifying notation. Furthermore, it is clear that even storing $O(n)$ bits at every tag can be overly expensive. Streaming techniques [30] can provide the necessary tools to address these issues. Since this is mainly an implementation aspect, it will be the focus of future work, our primary goal in this paper being to assess the feasibility of distributed recommendation algorithms using the models we consider.

Predicting user preferences. Upon visiting item r belonging to cluster C_k, the generic user j is recommended items of potential interest among those belonging to C_k, so that recommendations are cluster-based. Recalling the models discussed in Section 2, in order to achieve this goal it is necessary to estimate $w_{sj} = p_{kj} w_s$, for every $s \in C_k$ other than r. Since, for a given user j, p_{kj} is the same for all items in C_k, this amounts to estimating w_s, for every $s \in C_k$. The problem is that user profiles are unknown to the system and it is unfeasible to estimate them accurately from aggregate information about past user behavior. Fortunately, in order to provide recommendations, it is not necessary to know the values of these weights, but only their relative order.

In fact, we show in the next section that user histories and item summaries as defined above provide enough information for j's reader device to locally, accurately and efficiently compute a suitable monotonic function $f(\cdot)$ of cluster weights such that, for items $i, s \in C_k$, $f(w_i) \geq f(w_s)$ if and only if $w_i \geq w_s$. In practice, upon visiting r, j's smart reader computes a vector $\mathbf{R}(r,t)$, whose i-th entry $\mathbf{R}_i(r)$ is j's estimation of $f(w_i)$ at time t, for every $i \in C_k$. In the rest of this paper, we drop t from $\mathbf{R}(r,t)$ whenever clear from context. Also, we set $f(w_i) = w_i^2 \sum_{j=1}^{m(t)} p_{kj}^2$, where $m(t)$ is the number of users that entered the system up to time t. $f(w_i)$ is monotonically increasing in w_i and thus can be used to rank items, as stated by the following lemma, whose proof is straightforward and therefore omitted.

Lemma 1 *For items i and r function f verifies $f(w_i) \geq f(w_r)$ if and only if $w_i \geq w_r$.*

A crucial aspect is the probabilistic nature of the available information: since the set of items and the order of visit of a user are random variables, $\mathbf{H}(j)$ is the outcome of a random process. As we see in next section, this implies that $\mathbf{R}(r)$ is generated from statistics over the histories of users that visited r in the past. This implies that the information available at item r does not allow to exactly compute function $f(\cdot)$; in fact, one of our contributions are algorithms to compute $\mathbf{R}(r)$, so that it is a good estimate of $f(\cdot)$.

Cluster-based recommendations. The general recommendation algorithm is the obvious one and is modular with respect to how ranking of items is computed: upon visiting r belonging to cluster C_k, j's smart reader recommends the $\hat{T}$ *top* ranking items in $\mathbf{R}(r)$ that i) belong to C_k and ii) have not yet been visited by j. The overall behavior of the smart reader is summarized in Figure 2, with `UPDATE(`r`, `j`)` implementing the core operation of item ranking.

Require: Parameter $\hat{T}$: number of recommendations
1: When user j having history $\mathbf{H}(j)$ visits item r then
2: $\mathbf{R}(r)$ = UPDATE(r, j)
3: Recommend $\hat{T}$ top elements of $\mathbf{R}(r)$ not present in $\mathbf{H}(j)$

Fig. 2 Recommendation algorithm.

In the algorithm above, $\mathbf{R}(r)$ is read from the smart tag, updated and then it is used by the smart reader to recommend $\hat{T}$ top scoring elements. Then, the updated version of $\mathbf{R}(r)$ is stored back on the smart tag, replacing the older one. The key issue of how $\mathbf{R}(r)$ is maintained and updated is discussed in detail in Section 4.

Remarks. The algorithm described above provides *cluster-based* recommendations. I.e., upon visiting an item, a user is recommended a set of items belonging to the same cluster as the current one. Of course, the above strategy can be easily generalized to recommend the top $\hat{T}$ items, regardless of the cluster they belong to. In this paper, we focus on the important case in which we recommend items belonging to the same cluster[3]. We also note here that the general approach we consider is that of item-based recommendations, which has proved effective in practice (see for example [16, 25,9] or [4] for a more general survey). Our main goal here is to carry this approach, appropriately adapted, over to the fully decentralized, stigmergy-based scenario we envision.

In some cases, it may be interesting to consider strategies that also recommend items not belonging to the set of the $\hat{T}$ top ones, so as to diversify the basket of recommendations and thus increase the chance of serendipity. Proceeding this way may increase the probability of recommending items that do not match the user profile and can negatively affect the average quality of the recommendations provided. In Section 5, we test a simple randomized strategy, which recommends $\hat{T}$ items, each with probability proportional to its estimated weight in the user profile. This strategy essentially reproduces the probabilistic behavior of users in our model.

4 Prediction

We next discuss how $\mathbf{R}(r)$ is computed when the user visits the generic item r. Before, we give some additional notation that will be used in the sequel. In particular, we denote by $m(t)$ the number of users that entered the system up to time t. Recall that $F_i(t)$ denotes the number of users that visited item i up to time t. Considered two items i and r, we set $V_{ir}(j) = 1$ if j visits i *and* r *in this order*, 0 otherwise. Finally, we define $N_{ir}(t) = \sum_{j=1}^{m(t)} V_{ir}(j)$. If the user visits item r at time t, her smart reader reads r's summary and then, for every $i \neq r$ it computes $\mathbf{R}(r)$, where:

$$\mathbf{R}_i(r) = \left(1 + \frac{F_i(t)}{F_r(t)}\right) N_{ir}(t).$$

Here, $\mathbf{R}(r)$ is our estimator of $f(w_i)$ computed by the user's smart reader upon visiting r. Note that $F_r(t)$ and $N_{ir}(t)$ are available at r, while $F_i(t)$ is information

[3] Notice that a user may visit items belonging to different clusters over time and thus be recommended items belonging to different clusters, even if the recommendation strategy is cluster-based.

```
UPDATE(r, j)
Require: node r, agent j
 1: r maintains a vector w(r) of estimates of nodes' weights
 2: F_r = F_r + 1 {A new agent is visiting r}
 3: H_r(j) = F_r {j must initialize H_r(j)}
 4: for i: 1 ... n do {and i ≠ r}
 5:    if H_i(j) > 0 then {j visited i}
 6:       N_ir = N_ir + 1
 7:       R_i(r) = (1 + H_i(j)/F_r) N_ir
 8:    end if
 9: end for
10: return R(r)
```

Fig. 3 Update algorithm.

carried and provided to r by the visiting agent itself (see Figure 4, also observe that, by definition, $N_{ir}(t) > 0$ implies $F_r(t) > 0$). The estimator above translates into the implementation of the `UPDATE(·, ·)` routine described in Figure 3, to dynamically update $\mathbf{R}(r)$ at a generic node r whenever a new agent j visits the node. In particular, whenever agent j visits node r, r initially updates its local counter (line 2), since it is receiving a new visit, while j records in its history the number of agents that visited r prior to its visit (line 3). This information will be provided by j to nodes it visits in the sequel, if any. Finally (`for cycle`), for every item i previously visited by j, the user's smart reader updates $\mathbf{R}_i(r)$ using the information carried by j.

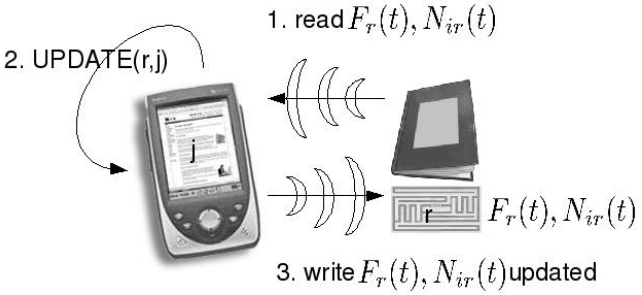


Fig. 4 Reader-tag interaction.

Computational aspects and memory requirements. The core of the computational complexity of the algorithm we propose lies in the update procedure of a smart tag's information upon a user's visit, described in Figure 3. The computational cost is clearly linear in the number of items for which the tag stores information (`for` cycle). It should be noted that this number will in general be smaller (possibly much smaller) than the total number n of items in the shop. On the other hand, in each iteration of the cycle, the smart reader is required to perform simple computations, which are compatible with many state-of-art devices (e.g., smart phones). Another issue (related to the former) concerns the amount of memory required at each smart tag. In the worst case, this will be the total number of items in the shop. Here, the limit comes from constraints imposed by the current state-of-art in passive devices. For example, commercial RFIDs

can have capacities as large as 8Kbytes at the price of 15$, but 32KByte devices are ready for the market [3]. In our approach and without employing data streaming techniques, each smart tag needs to store one vector with n integer components (N_{*r}) and the counter F_r (note that $\mathbf{R}(r)$ is computed by the smart reader on the fly). Assuming 4 Bytes for each component, this implies the possibility of storing the necessary information for about 2000 items using an 8KByte RFID and four times so much in the forseeable future. In fact, N_{*r} is a vector of frequency counts and it could be maintained in small (polylogarithmic) space using streaming techniques [13].

Analysis. The main result of this section is the proof that, over time and for every $i \neq r$, $\mathbf{R}_i(r)$ provides an increasingly accurate estimation of $f(w_i) = w_i^2 \sum_{j=1}^{m(t)} p_{kj}^2$. In order to analyze the accuracy of the estimator above and the rationale behind, we assume that, if user j visits item i at time t_1 and item r at time t_2, no other users visit i in the interval $(t_1, t_2]$. This assumption is only done for the purpose of the analysis and is not required by our algorithm. In the analysis, it is equivalent to assuming that users visit the system one at the time, i.e., if user j visits i at time t_1 and r at time $t_2 > t_1$, we have $m(t_1) = m(t_2)$. Note that we are interested in the system behavior as t grows and more and more users visit the smart shop, so as this approximation becomes increasingly accurate. The following result holds:

Theorem 1 *If i, r belong to the same cluster C_k for some k, $\mathbf{R}_i(r)$ becomes an increasingly accurate estimate of $f(w_i)$. In particular, accuracy becomes arbitrarily high as the number of users visiting r increases over time.*

Proof of Theorem 1. In the sequel, we set $\overline{f}_i(t) = \mathbf{R}_i(r)$, the estimate of $f(w_i)$ maintained at time t at node r. The proof of Theorem 1 is implied by proving the following statement:
If $i, r \in C_k$ for some k:

$$f(w_i) = \left(1 + \frac{\mathbf{E}[F_i(t)]}{\mathbf{E}[F_r(t)]}\right) \mathbf{E}[N_{ir}(t)]\,.$$

Furthermore, for every r and i belonging to the same cluster C_k, whenever t is large enough that $\sum_{j=1}^{m(t)} p_{kj}^2 \geq \frac{3(w_i+w_r)}{\epsilon^2 w_i^2 w_r} \ln \frac{6}{\delta}$ with $\epsilon \leq 1/5$:

$$\mathbf{P}\left[(1-3\epsilon)f(w_i) \leq \overline{f}_i(t) \leq (1+4\epsilon)f(w_i)\right] \geq 1-\delta.$$

We first give the following Lemma that will be useful later:

Lemma 2 *For every $i \in C_k$: $\mathbf{E}[F_i(t)] = w_i \sum_{j=1}^{m(t)} p_{kj}$. Furthermore, for every δ, $\epsilon > 0$, as soon as t is such that $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3}{\epsilon^2 w_i} \ln \frac{2}{\delta}$:*

$$\mathbf{P}[|F_i(t) - \mathbf{E}[F_i(t)|] > \epsilon \mathbf{E}[F_i(t)]] \leq \delta.$$

Proof We obviously have $F_i(t) = \sum_{j=1}^{m(t)} X_i(j)$ and $\mathbf{P}[X_i(j) = 1] = p_{kj} w_i$, where $X_i(j) = 1$ if user j visits item i, 0 otherwise. This immediately gives $\mathbf{E}[F_i(t)] = w_i \sum_{j=1}^{m(t)} p_{kj}$. Furthermore, agents visits are independent of each other. Hence, applying Chernoff bound to $\sum_{j=1}^{m(t)} X_i(j)$ [29] yields the result.

In the sequel, we denote by $\mathcal{S}(j)$ the set of items visited by user j during its permanence in the system. If $l \leq |\mathcal{S}(j)|$, $\mathcal{S}_{<l}(j)$ denotes the subset of the first $l-1$ items visited by j. The following lemma holds:

Lemma 3 *For every S such that $\{i,r\} \subseteq S$, with $i,r \in C_k$ for some k, for every j:*

$$\mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S] = \frac{w_i}{w_i + w_r}.$$

Proof Denote by $Y_l(j)$ the item visited at the l-th step of j's visit, where $Y_l(j) = \emptyset$ if $l > |S|$. We have:

$$\mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S] = \sum_{l=1}^{|S|-1} \mathbf{P}[(\mathcal{S}_{<l}(j) \cap \{i,r\} = \emptyset) \wedge (Y_l(j) = i) \,|\, \mathcal{S}(j) = S]$$
$$= \sum_{l=1}^{|S|-1} \mathbf{P}[Y_l(j) = i \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) \cap \{i,r\} = \emptyset)]\, \mathbf{P}[\mathcal{S}_{<l}(j) \cap \{i,r\} = \emptyset \,|\, \mathcal{S}(j) = S],$$

where the first equality follows since, given $\mathcal{S}(j) = S$, with $\{i,r\} \subseteq S$, $V_{ir}(j) = 1$ is equivalent to stating that i is visited at some step where r has not been visited yet. On the other hand, denote by $\mathcal{S}_l(i,j,S)$ the set of all subsets of S that i) contain $l-1$ elements and ii) do not contain $\{i,j\}$. We have:

$$\begin{aligned}
&\mathbf{P}[Y_l(j) = i \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) \cap \{i,r\} = \emptyset)] \\
&= \sum_{W \in \mathcal{S}_l(i,r,S)} \mathbf{P}[Y_l(j) = i \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) = W)] \cdot \\
&\cdot \mathbf{P}[\mathcal{S}_{<l}(j) = W \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) \cap \{i,r\} = \emptyset)],
\end{aligned}$$

where the equality follows since $\mathcal{S}_{<l}(j) = W \in \mathcal{S}_l(i,j,S)$ implies $\mathcal{S}_{<l}(j) \cap \{i,r\} = \emptyset$. On the other hand:

$$\mathbf{P}[Y_l(j) = i \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) = W)] = \frac{w_{ij}}{1 - \sum_{f \in W} w_{fj}},$$

by the definition of the weighted visit process described above. Analogously:

$$\mathbf{P}[Y_l(j) = r \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) = W)] = \frac{w_{rj}}{1 - \sum_{f \in W} w_{fj}},$$

and

$$\begin{aligned}
&\mathbf{P}[Y_l(j) = r \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) \cap \{i,r\} = \emptyset)] \\
&= \sum_{W \in \mathcal{S}_l(i,j,S)} \mathbf{P}[Y_l(j) = r \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) = W)] \cdot \\
&\cdot \mathbf{P}[\mathcal{S}_{<l}(j) = W \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) \cap \{i,r\} = \emptyset)].
\end{aligned}$$

This implies that the expressions of $\mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S]$ and $\mathbf{P}[V_{ri}(j) = 1 \,|\, \mathcal{S}(j) = S]$ are the same up to multiplying factors, which are $w_{ij} = p_{kj} w_i$ and $w_{rj} = p_{kj} w_r$ respectively. Therefore:

$$\frac{\mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S]}{\mathbf{P}[V_{ri}(j) = 1 \,|\, \mathcal{S}(j) = S]} = \frac{w_i}{w_r}.$$

Furthermore, $\mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S] + \mathbf{P}[V_{ri}(j) = 1 \,|\, \mathcal{S}(j) = S] = 1$, since $\{i,r\} \in S$ and, therefore, $(\mathcal{S}(j) = S)$ implies the event $(V_{ir}(j) = 1 \vee V_{ri}(j) = 1)$. This yields the result.

Notice that, at every node r and at any time t, we are in fact observing the variable $N_{ir}(t) = \sum_{j=1}^{m(t)} V_{ir}(j)$. As to $\mathbf{P}[V_{ir}(j) = 1 \,|\, r \in \mathcal{S}(j)]$, we have:

Lemma 4 *If $i, r \in C_k$:* $\mathbf{P}[V_{ir}(j) = 1] = \frac{p_{kj}^2 w_i^2 w_r}{w_i + w_r}$.

Proof We have:

$$\mathbf{P}[V_{ir}(j) = 1 \,|\, r \in \mathcal{S}(j)] = \sum_{S: r \in S} \mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S]\, \mathbf{P}[\mathcal{S}(j) = S \,|\, r \in \mathcal{S}(j)]$$

$$= \sum_{S: \{i,r\} \subseteq S} \mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S]\, \mathbf{P}[\mathcal{S}(j) = S \,|\, r \in \mathcal{S}(j)]$$

$$= \frac{w_i}{w_i + w_r} \sum_{S: \{i,r\} \subseteq S} \mathbf{P}[\mathcal{S}(j) = S \,|\, r \in \mathcal{S}(j)] = \frac{w_i}{w_i + w_r} \mathbf{P}[\{i, r\} \subseteq \mathcal{S}(j) \,|\, r \in \mathcal{S}(j)] = \frac{p_{kj} w_i^2}{w_i + w_r},$$

where the second inequality follows since $V_{ir}(j) = 0$ deterministically if $i \notin \mathcal{S}(j)$, the third follows from Lemma 3 and the fifth follows since the events $(i \in \mathcal{S}(j))$ and $(r \in \mathcal{S}(j))$ are statistically independent. The claim then follows since $\mathbf{P}[r \in \mathcal{S}(j)] = p_{kj} w_r$.

Lemma 5 *If $i, r \in C_k$:* $\mathbf{E}[N_{ir}(t)] = \frac{w_i^2 w_r}{w_i + w_r} \sum_{j=1}^{m(t)} p_{kj}^2$.
Furthermore: $\mathbf{P}[|N_{ir}(t) - \mathbf{E}[N_{ir}(t)]| > \epsilon \mathbf{E}[N_{ir}(t)]] \leq \delta$, *as soon as t is large enough that* $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3(w_i + w_r)}{\epsilon^2 w_i^2 w_r} \ln \frac{2}{\delta}$:

Proof The first claim follows immediately from Lemma 4. The second claim follows from a simple application of Chernoff bound [29] to the variable $N_{ir}(t)$.

The following holds:

Lemma 6 *If at most* 1 *agent visits the system at any time t and $i, r \in C_k$ for some k:*

$$f(w_i) = w_i^2 \sum_{j=1}^{m(t)} p_{kj}^2 = \left(1 + \frac{\mathbf{E}[F_i(t)]}{\mathbf{E}[F_r(t)]}\right) \mathbf{E}[N_{ir}(t)]\,.$$

Proof The proof follows immediately, by observing that $\mathbf{E}[F_r(t)]\,/\mathbf{E}[F_i(t)] = w_r/w_i$ from Lemma 2 and substituting $w_r = w_i \mathbf{E}[F_r(t)]\,/\mathbf{E}[F_i(t)]$ in the expression of $\mathbf{E}[N_{ir}(t)]$ in Lemma 5.

Lemma 6 shows that the estimator we are using is in fact a simple plug-in estimator for $f(w_i)$. We can finally prove the claim of the theorem, i.e., that the approximation of $f(w_i)$ becomes more and more accurate over time.

In the sequel of this proof we drop t from the notation, since it is understood from context. We also recall that F_i, F_r and N_{ir} are each the sum of binary independent variables by the independence of the agents' visits. Hence, if $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3(w_i + w_r)}{\epsilon^2 w_i^2 w_r} \ln \frac{6}{\delta}$, simple applications of Lemma 2 and Lemma 5 allow to conclude that each of the following events occurs with probability at most $\delta/3$: i) $|F_i - \mathbf{E}[F_i]| > \epsilon \mathbf{E}[F_i]$; ii) $|F_r - \mathbf{E}[F_r]| > \epsilon \mathbf{E}[F_r]$; iii) $|N_{ir} - \mathbf{E}[N_{ir}]| > \epsilon \mathbf{E}[N_{ir}]$. Hence, with probability at least $1 - \delta$ we have:

$$\overline{f}_i \leq \left(1 + \frac{1 + \epsilon}{1 - \epsilon} \frac{\mathbf{E}[F_i]}{\mathbf{E}[F_r]}\right) (1 + \epsilon) \mathbf{E}[N_{ir}] < (1 + 4\epsilon) \left(1 + \frac{\mathbf{E}[F_i(t)]}{\mathbf{E}[F_r(t)]}\right) \mathbf{E}[N_{ir}(t)]\,,$$

where the first inequality follows since we have $N_{ir} \leq (1+\epsilon)\mathbf{E}[N_{ir}]$, $F_i \leq (1+\epsilon)\mathbf{E}[F_i]$ and $F_r \geq (1-\epsilon)\mathbf{E}[F_r]$, while the second inequality holds if $\epsilon \leq 1/5$. Analogously:

$$\overline{f}_i \geq \left(1 + \frac{1-\epsilon}{1+\epsilon}\frac{\mathbf{E}[F_i]}{\mathbf{E}[F_r]}\right)(1-\epsilon)\mathbf{E}[N_{ir}] > (1-3\epsilon)\left(1 + \frac{\mathbf{E}[F_i(t)]}{\mathbf{E}[F_r(t)]}\right)\mathbf{E}[N_{ir}(t)]\,,$$

where the first inequality follows from a similar argument as above, while the second inequality follows from trivial manipulations. Recalling Lemma 6 we complete the proof of Theorem 1.

Convergence. The result of Theorem 1 also describes the convergence properties of our algorithms. It is possible to prove that these bounds are asymptotically tight. A complete analysis is not the purpose of this paper. For the sake of completeness, we briefly address the simpler aspect of the estimation of $\mathbf{E}[F_r(t)]$ at a generic node r. Note that accurately estimating $\mathbf{E}[F_r(t)]$ is crucial for our estimator. It is possible to prove the following theorem:

Theorem 2 *Assume the cluster based model. For every $i \in C_k$ and for every $0 < \delta < 1$, $\Theta(\frac{1}{w_i}\ln\frac{1}{\delta})$ visits are necessary and sufficient to estimate $\mathbf{E}[F_i(t)]$ accurately.*

Proof The definition of our cluster-based model immediately implies that, if x users visit C_k, then $\mathbf{E}[F_i(t)] = w_i x$. In particular, every user has an equal probability w_i of visiting the item i, independently of the others. We next prove that

$$\mathbf{P}[|F_i(t) - \mathbf{E}[F_i(t)]\,| \geq \epsilon\mathbf{E}[F_i(t)]] = \mathbf{P}[F_i(t) \leq (1-\epsilon)w_i x] > \delta,$$

whenever x is small enough. The case $F_i(t) > (1+\epsilon)w_i x$ is handled similarly. We have:

$$\begin{aligned}
&\mathbf{P}[F_i(t) \leq (1-\epsilon)w_i x] \geq \mathbf{P}[F_i(t) \leq \lfloor(1-\epsilon)w_i x\rfloor] \\
&= \sum_{y=0}^{\lfloor(1-\epsilon)w_i x\rfloor}\binom{x}{y}(1-w_i)^{x-y}w_i^y > \sum_{y=0}^{\lfloor(1-\epsilon)w_i x\rfloor}(1-w_i)^{x-y}w_i^y \\
&= (1-w_i)^x\frac{1-\left(\frac{w_i}{1-w_i}\right)^{\lfloor(1-\epsilon)w_i x\rfloor+1}}{1-\frac{w_i}{1-w_i}} > (1-w_i)^x,
\end{aligned}$$

where the fourth inequality follows from simple manipulations, while the last inequality follows whenever $w_i/(1-w_i) < 1$. which is always the case whenever $w_i < 1/2$. This implies that

$$\mathbf{P}[|F_i(t) - \mathbf{E}[F_i(t)]\,| > \epsilon\mathbf{E}[F_i(t)]] > (1-w_i)^x.$$

Considered any $0 < \delta < 1$, simple manipulations hows that $(1-w_i)^x \geq \delta$, whenever

$$x\ln\left(1 + \frac{w_i}{1-w_i}\right) \geq \ln\frac{1}{\delta}.$$

Now, if we assume $w_i > 1/2$, we have $w_i/(1-w_i) < 1$ and the inequality above is true only if

$$x\frac{w_i}{1-w_i} \geq \ln\frac{1}{\delta},$$

which holds whenever

$$x \geq \frac{1-w_i}{w_i}\ln\frac{1}{\delta},$$

thus proving the claim.

5 Experimental analysis

The experimental part of this paper focuses on assessing the soundness of our model (Section 5.3) and the effectiveness of our recommendation algorithms (Section 5.4). As to the second issue, we compared the performance of our solution with a standard centralized method [16].

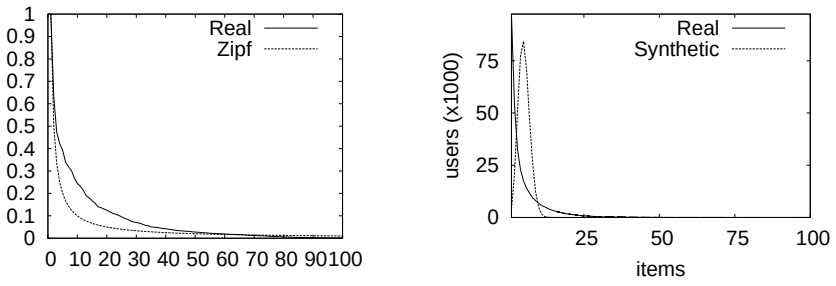
5.1 Performance Metrics for Recommendations

We evaluate the performance of the system along two main axes: the ability to infer a ranking in user preferences and the quality of recommendations. In particular, we evaluate the former in terms of ranking similarity, while the latter is evaluated in terms of standard measures of quality used in information retrieval [10], in particular hit ratio, precision and recall.

Ranking similarity. As described in Section 2, a user's preferences are described in terms of a vector of weights (i.e., the user profile), its i-th component measuring the degree of potential interest of the i-th item to the user. As already remarked in Section 3, to the purpose of recommending items we are not interested in estimating the components of the user profile, but only their relative order, i.e., their ranking. In particular, if user j visits items r and is recommended a list of items of potential interest, the quality of recommendation depends on how close the ranking of items estimated by j's smart reader is close to the real, unknown one determined by j's profile. To measure how close the real and the estimated rankings are, we use a standard measure of the distance between rankings, i.e., Kendall's τ coefficient (KT for short). KT measures the degree of similarity between two rankings and it is defined as $\tau = 4P/(n(n-1))-1$, where $P = \sum_i P_i$. Here, if x is the i-th item in the first ranking, P_i denotes the number of items that follow x in both rankings (i.e. actual ranking and estimated one). KT enjoys the following properties: i) if the agreement between the two rankings is perfect (i.e., the two rankings are the same) the coefficient has value 1; ii) if the disagreement between the two rankings is perfect (i.e., one ranking is the reverse of the other) the coefficient has value -1; iii) for all other arrangements the value lies between -1 and 1, and increasing values imply increasing agreement between the rankings.

HitRatio($\hat{\mathrm{T}}$). We recall that user j's smart reader, recommends the $\hat{T}$ top items in $\mathbf{R}(r)$ belonging to the same cluster as r and that are not present in $\mathbf{H}(j)$. There is a hit if at least one of the $\hat{T}$ recommended items will be eventually visited by j. *HitRatio($\hat{\mathrm{T}}$)* is defined as the ratio between the number of hits and the overall number of recommendations given.

Precision($\hat{\mathrm{T}}$) and Recall($\hat{\mathrm{T}}$). Precision and recall are standard measures of the accuracy in providing relevant documents. Define by D the set of items (called *corpus*) and by D_j the set of relevant items for user j and let $(d_1, d_2, .., d_t)$ be the recommendations provided by the visited item and $(r_1, r_2, .., r_t)$ where $r_i = 1$ if $d_i \in D_j$ and 0 otherwise. Then:



(a) weight distribution

(b) visit length

Fig. 5 Validation of our model

$recall(\hat{T}) = \frac{1}{|D_j|} \sum_{1 \leq i \leq \hat{T}} r_i$.
I.e., recall is the fraction of all relevant items included in the recommendation. Furthermore:

$precision(\hat{T}) = \frac{1}{\hat{T}} \sum_{1 \leq i \leq \hat{T}} r_i$.
I.e., precision is the fraction of the top $\hat{T}$ recommendations that are actually relevant.
In the following we assume that D_j is the set of items that will be visited in the future by a user.

5.2 Datasets

We validated our model through experiments on the Netflix movie dataset[4]. This dataset is made of 17770 files, one for each movie. Each file stores a set of ratings represented as a list of tuples <userId, rating, date>, for an overall number of 100480507 ratings made by 480000 users.

In our cluster based model, movies are clusterized by genre. Unfortunately, movies' genre is not provided by Netflix, but we extracted this information from the MovieLens dataset[5].

5.3 Model Validation

Distribution of weights. In our experiments on cluster-based recommendation, we considered the Comedy genre, which is one of the most populated clusters. In particular, we uniformly sampled 100 items out of 744 available comedy movies. Considered an item i from the above sample, we used its popularity, obtained by dividing the number of users that visited i by the total number of users, as a proxy for w_i. We did not use users' ratings directly to compute weights, since we did not observe a clear connection between the a priori choice of the movie based on its popularity and user's taste and the a posteriori rating of the movie. Our model, reasonably fits actual user behavior,

[4] Available at http://www.netflixprize.com/.
[5] Available at http://www.grouplens.org/node/73

if we assume a Zipf's distribution of the weights, with exponent equal to 1 (see figure 5(a)).

Visit patterns. As far as the order of visits is concerned, we calculated the correlation coefficient between the cluster weights and the order of visit on real data and we obtained a value of -0.6. This strongly supports the use of the weighted visit model. In contrast, we observed that the distribution of the number of visited items per user predicted by our model using a Zipf's distribution for item weights differs with respect to real data (see Figure 5(b)). In fact, Figure 5(b) shows that the number of items visited also follows a Zipf law. This is due to our simplifying assumption that users have the same profile within each cluster (though having different preferences for the same cluster). This aspect could be easily taken into account in the model, by suitably redefining weights in the cluster-based model. Since the purpose of the probabilistic model we consider is to infer a ranking in user preferences and not to describe user behavior in its complexity, we opted for simplicity and neglected this issue.

5.4 Performance

Algorithms. As a benchmark to evaluate the quality of our recommendation algorithm *alg*, we compare its performance to that of a centralized recommendation algorithm *deshp* and a baseline algorithm *rnd* that simply recommends an item chosen uniformly at random among the ones not yet visited by the user. Furthermore we considered *prob*, a variant of *alg*, where instead of recommending the $\hat{T}$ top items, each item is recommended with a probability proportional to its weight.

More formally, considering item r and denoted by Q_f the first f items already recommended to user j in the current interaction with r, the $f+1$-th recommendation is for item $i \notin \{r\} \cup \mathbf{H}(j) \cup Q_f$ with probability $\frac{\sqrt{\mathbf{R}_i(r)}}{\sum_{l \notin \{r\} \cup \mathbf{H}(j) \cup Q_f} \sqrt{\mathbf{R}_l(r)}}$ [6]. The possibility of also selecting items not belonging to the set of the $\hat{T}$ ones, addresses the issue that the top $\hat{T}$ recommendations are probably the most accurate, but they might in part correspond to very popular or obvious choices. Providing some degree of diversification may alleviate this potential issue at the cost of a loss in accuracy of prediction. Algorithm (*prob*) tries to achieve some degree of diversification in a "controlled" way, by essentially reproducing the probabilistic behavior of users as predicted by our model.

Algorithm (*deshp*) is a state-of-art centralized recommendation algorithm, based on conditional probability similarity and it is described in [16, Subsection 4.1]. The algorithm was implemented adopting the optimizations suggested in [16] and tuning parameters for best performance under the datasets we consider.

Note that, differently from our algorithm, (deshp) can access the whole dataset of user histories. This allows to define a similarity value among any pair of items i and j, such that they were both visited by at least one user. For our decentralized algorithm, this is not the case. For example, if x users visited first i and then j in this order and y users visited them in the inverse order, estimating the similarity between i and j using (deshp) would require knowledge of $x+y$ at both i and j, which is unfeasible in the scenario we envision. Furthermore, the performance of (deshp) depends by the

[6] Recall from Section 3 that $\mathbf{H}(j)$ and $\mathbf{R}_i(r)$ are respectively j's history and the i-th component of r's summary and that $\mathbf{R}_i(r)$ is proportional to w_i^2, up to a factor which is constant for items belonging to the same cluster.

choice of a frequency scaling parameter α [16, formula (2), page 152], which can have "a significant impact on the recommendation quality" [16, par. 6.2.1.4, page 164]. In our experiments, we optimized the choice of α for the specific dataset we considered, but this would be unfeasible in practice in the decentralized scenario we consider. On the other hand, our model assumes visiting patterns that probabilistically depend on item popularities within a topic and statistically infers them. This simple model seems to capture important trends in user behavior that somewhat compensate the lack of information mentioned above, as experimental evidence suggests.

Ranking similarity. Our first goal is to evaluate the ranking similarity between the actual cluster weights and the estimated ones by means of KT computed over 10000 users. It is worth noting that, each item i can estimate only a subset S_i of the other item's weights, depending on the number of users that visited the system and their visiting patterns. We calculate the KT of each item with cardinality $|S_i| \geq 2$. As predicted by our analysis, figure 6 shows that KT tends to one as the number of users increases and, more importantly, that ranking similarity is significantly high (0.7 for synthetic and 0.8 for real data) already after collecting statistics over a very small number of users (i.e. 500). Note that this relatively small number of users, supports the feasibility of our proposal in practice.

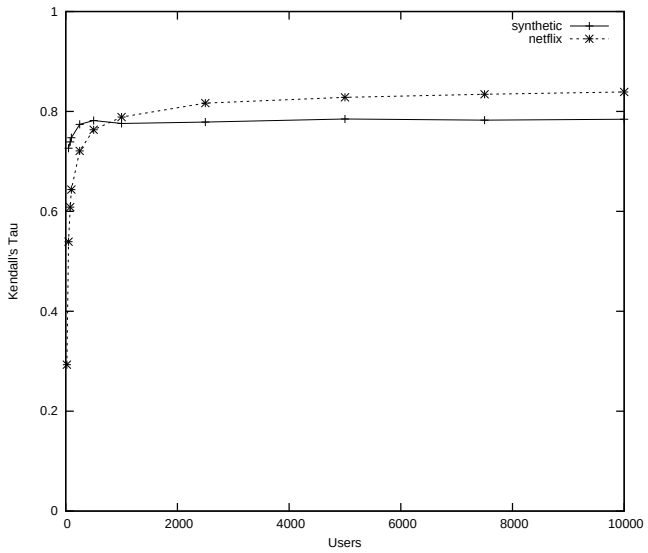
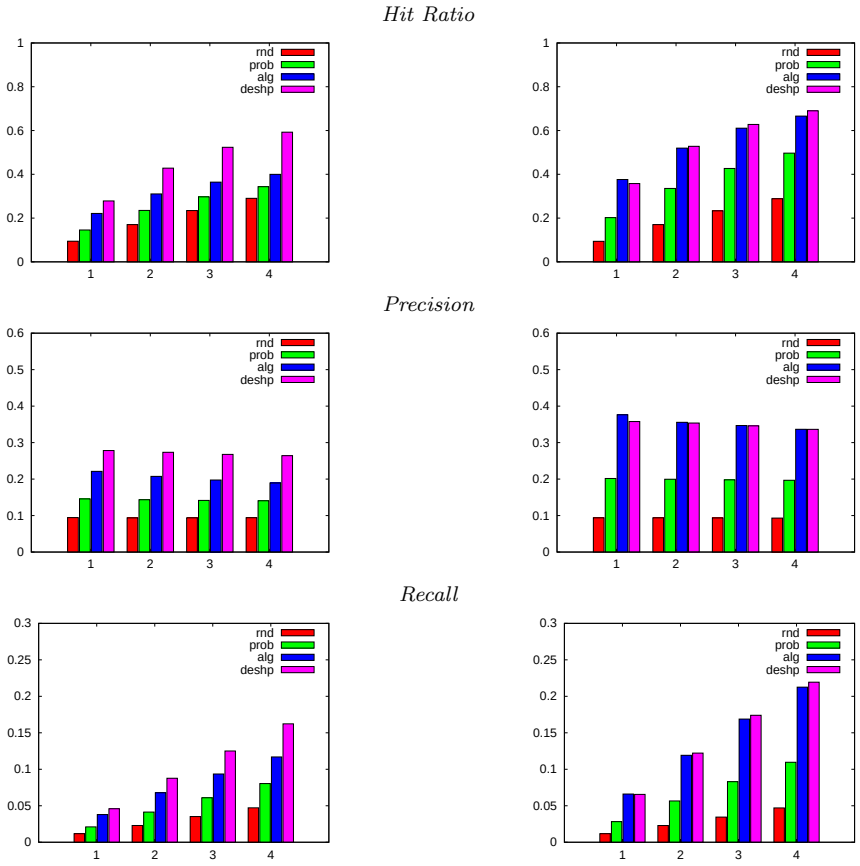


Fig. 6 Kendall's τ: cluster-based, 100 items.

Quality of recommendation. We evaluated hit ratio, precision and recall of our recommendation algorithm on both a real Netflix dataset and synthetic data generated according to our model. Each performance index has been computed by averaging the results over 10 independent runs with 100 items and 10000 users. To generate synthetic inputs, we assume the weighted visit model with item weights within a cluster distributed according to Zipf's law; since we are considering the single cluster of comedy movies, we can assume that p_{kj}, the *cluster preference*, is one for all users j.



(a) training 100 users **(b) training 2000 users**

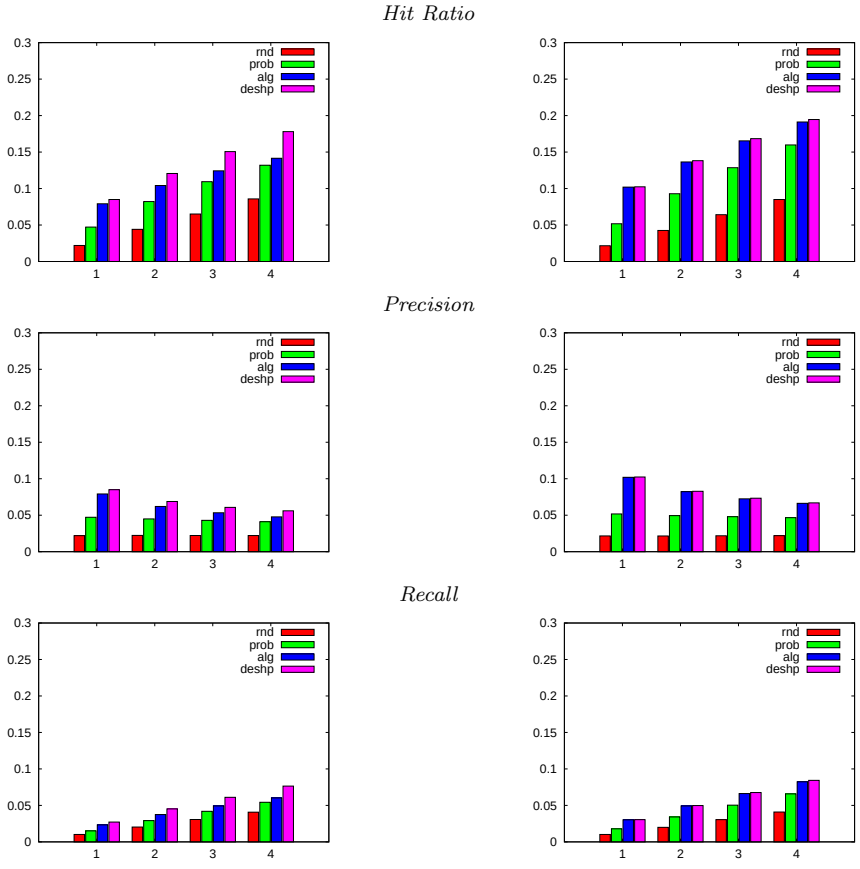
Fig. 7 Netflix dataset. x coordinate is the number of provided recommendations ($\hat{T}$).

We distinguish two phases in the execution of our algorithms: the *training phase*, during which user profiles are computed (for *alg* and *prob*, item weights are estimated) and the *recommendation phase*, in which recommendations are actually given. In light of the results above, we limit the training phase to very few users. In particular, we consider 100 (tp=100) and 2000 (tp=2000) users, so as to better evaluate how the considered metrics improve as the length of the training phase increases.

For hit ratio, the performance of *alg* is always sensibly better than *rnd* and *prob* and close to *deshp* as the length of the training phase increases (see figure 7). HitRatio($\hat{T}$) of *alg* and *deshp* on real data is between 4 and 2.5 times better than *rnd* (up to 2 for *prob*) when tp=2000. It is interesting to note that the absolute performance of the algorithms is worse on synthetic data (compare figures 8 and 7). This fact can be explained considering the average length $\tilde{v}$ of the number of visited items per users. In fact, $\tilde{v}$ on real data is about 7 while in synthetic data is about the half. Since the probability of a hit for a user clearly also depends on the number of visits of the user, it follows that the higher the $\tilde{v}$, the higher is the hit ratio (similar considerations can be made for precision and recall). For precision and recall, the performance of

our fully decentralized algorithm is very close to the centralized one and both of them significantly out-perform *rnd*. The precision of *rnd* is constant and does not depend on the number of suggestions provided (i.e. $\hat{T}$), as it can be easily proved considering that the random recommendation process is governed by a hypergeometric distribution. The precision of *alg* tends to decrease as the number of recommendation increases; this is expected since, as we observed previously, the accuracy of recommendations depends on the popularity of the items and increasing the number of recommended items forces the algorithm to choose items of decreasing popularity, so more unlikely to meet user expectations. Finally, observe that, as expected *prob* exhibits performance that are worse than those of *alg* and *deshp*, but still well above the baseline *rnd*.

Remark. It should be noted that the values of precision and recall that we obtain on real datasets are indeed relatively high, since they refer to the prediction of really existing links and not to the judgement given by the user about the quality of the recommendations provided. In fact, the data we have do not allows us to directly infer the impact of recommendations on user behavior.



(a) training 100 users **(b) training 2000 users**

Fig. 8 Synthetic data. x coordinate is the number of provided recommendations ($\hat{T}$).

6 Discussion

The main contribution of this paper is a model of user behavior that seems to capture important trends in real user data derived from commercial recommendation systems, enabling recommendation strategies that are fully decentralized and seem suitable to meet average user expectations. The proposed model is simple enough as to allow the statistical estimation of parameters from real user activity logs. The resulting recommendation strategy achieves a performance that is comparable to that of state-of-art centralized solutions.

Since available data did not allow to assess the impact of recommendations directly, results on the quality of recommendation (i.e., precision and recall) have been obtained in a worsening scenario, i.e., checking the extent to which items that were judged of potential interest for a user by the system were actually chosen by that user, which of course leaves out items of potential interest that did not appear in the user log.

A number of issues remain, which will hopefully encourage further research in the area. As to the model, while physical constraints can be incorporated into the model, as discussed at the end of Section 2, other aspects to address remain. A first issue has to do with item popularity. Popularity can indeed change, sometimes rapidly, over time. A best selling book might be much less popular within the next month, as the initial wave of interest fades. Proposing simple ageing mechanisms, while keeping the model simple enough is an interesting point. On the other hand, we have proposed strategies that work well when items are clustered, as described in Section 2. Extending the approach of this paper to the general case is an interesting issue. Of course simple heuristics (e.g., recommending the most popular items) can be easily derived from our approach, but more sophisticated strategies based on more solid theoretical foundations are needed. Another important extension is to include the social context in providing recommendations; social context points on the users community such as friends, neighbors and colleagues. According to the social dimension, adapting retrieval aims at leveraging the search according to implied preferences of the users community rather than just the individual. Social context is used in recommender systems based on collaborative filtering techniques [33, 21] and it will be important to include it in our approach.

We conclude noting that, as experimental evidence also suggests, our fully decentralized approach is competitive with state-of-the-art centralized solutions and it is technologically realistic.

7 Acknowledgments

Partially supported by EU STREP Project ICT-215270 FRONTS and by FIRB project RBIN047MH.

References

1. Amazon web site. url: http://www.amazon.com, 2008.
2. Netflix web site. url: http://www.netflix.com, 2008.
3. Airbus signs contract for high-memory rfid tags. rfid journal. url: http://www.rfidjournal.com/article/view/7323, 2010.

4. Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.
5. Noga Alon, Baruch Awerbuch, Yossi Azar, and Boaz Patt-Shamir. Tell me who i am: an interactive recommendation system. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 1–10, New York, NY, USA, 2006. ACM.
6. Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Mark R. Tuttle. Improved recommendation systems. In *SODA*, pages 1174–1183. SIAM, 2005.
7. Yossi Azar, Amos Fiat, Anna R. Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *STOC*, pages 619–626, 2001.
8. Ozalp Babaoglu, Geoffrey Canright, Andreas Deutsch, Gianni A. Di Caro, Frederick Ducatelle, Luca M. Gambardella, Niloy Ganguly, Márk Jelasity, Roberto Montemanni, Alberto Montresor, and Tore Urnes. Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst.*, 1(1):26–66, 2006.
9. Sarwar Badrul, Karypis George, Konstan Joseph, and Reidl John. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.
10. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.
11. Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. Distributed collaborative filtering with domain specialization. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 33–40, New York, NY, USA, 2007. ACM.
12. Byron Leite Dantas Bezerra1 and Francisco de Assis Tenorio de Carvalho. Symbolic data analysis tools for recommendation systems. *Knowl. Inf. Syst.*, 2010, on-line.
13. Graham Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Transactions on Database Systems*, 30(1):249–278, 2005.
14. Rickard Cöster and Martin Svensson. Incremental collaborative filtering for mobile devices. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1102–1106, New York, NY, USA, 2005. ACM.
15. Christian Decker, Uwe Kubach, and Michael Beigl. Revealing the retail black box by interaction sensing. In *ICDCS Workshops*, pages 328–333, 2003.
16. Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.
17. Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *STOC*, pages 82–90, 2002.
18. Agner Fog. Sampling methods for wallenius' and fisher's noncentral hypergeometric distributions. *Communications in Statistics - Simulation and Computation*, 37(2):241 – 257, 2008.
19. Marco Gori and Augusto Pucci. Itemrank: a random-walk based scoring algorithm for recommender engines. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2766–2771. Morgan Kaufmann Publishers Inc., 2007.
20. Zan Huang, Daniel Zeng, and Hsinchun Chen. A comparison of collaborative-filtering recommendation algorithms for E-commerce. *IEEE Intelligent Systems*, 22(5):68–78, 2007.
21. Cane Wing ki Leung, Stephen Chi fai Chan, and Fu-Lai Chung. A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. *Knowl. Inf. Syst.*, 10(3):357–381, 2006.
22. Jon M. Kleinberg and Mark Sandler. Convergent algorithms for collaborative filtering. In *ACM Conference on Electronic Commerce*, pages 1–10. ACM, 2003.
23. Panos Kourouthanasis, Diomidis Spinellis, Giorgos Roussos, and Giorgos Giaglis. Intelligent cokes and diapers: MyGrocer ubiquitous computing environment. In *First International Mobile Business Conference*, pages 150–172, July 2002.
24. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Recommendation systems: A probabilistic analysis. *J. Comput. Syst. Sci.*, 63(1):42–61, 2001.
25. Greg Linden, Brent Smith, and Jeremy York. Industry report: Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Distributed Systems Online*, 4(1), 2003.
26. Marco Mamei and Franco Zambonelli. Pervasive pheromone-based interaction with rfid tags. *ACM Trans. Auton. Adapt. Syst.*, 2(2):4, 2007.
27. Carl D. Meyer, editor. *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

28. Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Intelligent User Interfaces*, pages 263–266. ACM, 2003.
29. Michael Mitzenmacher and Eli Upfal. *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, January 2005.
30. Muthukrishnan. Data streams: Algorithms and applications. In *Foundations and Trends in Theoretical Computer Science, Now Publishers or World Scientific*, volume 1. 2005.
31. Saharon Rosset, Claudia Perlich, and Bianca Zadrozny. Ranking-based evaluation of regression models. *Knowl. Inf. Syst.*, 12(3):331–353, 2007.
32. M. Roth and S. Wicker. Termite: ad-hoc networking with stigmergy. *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, 5:2937–2941 vol.5, Dec. 2003.
33. Lynda Tamine-Lechani, Mohand Boughanem, and Mariam Daoudontact. Evaluation of contextual information retrieval effectiveness: overview of issues and research. *Knowl. Inf. Syst.*, 2009, on-line.
34. W. Wade. A grocery cart that holds bread, butter, and preferences. *New York Times*, Jan(16), 2003.
35. J. Wang, J. Pouwelse, R. Lagendijk, and M. J. T. Reinders. Distributed collaborative filtering for peer-to-peer file sharing systems. In *21st Annual ACM Symposium on Applied Computing*, pages 1026–1030, 2006.
36. Bo Xie, Peng Han, Fan Yang, Ruimin Shen, Hua-Jun Zeng, and Zheng Chen. DCFLA: A distributed collaborative-filtering neighbor-locating algorithm. *Information Sciences*, 177(6):1349–1363, 2007.