Partial ranking of products for recommendation systems

Sébastien Hémon, Thomas Largillier, and Sylvain Peyronnet

LRI; INRIA; Univ. Paris-Sud XI; 91405 Orsay, France

Abstract. A recommendation system (or recommender) is an algorithm whose goal is to recommend products to potential users. To achieve its task, it uses information about some user preferences.
We present recommenders that use information about the preferences of only a very small subset of users (called a committee) on a very small set of products called the witness products set. The main interest of our approach compared to previous ones is that it needs substantially less data for ensuring a very good quality of recommendation.

1 Introduction & related work

Recommendation systems aim to select products for a particular user from a list shared by all (available products for instance) according to known previous preferences of users. There are essentially two ways to recommend products. One consists in learning the preferences of a particular user based on the products (s)he liked before and recommending products similar to these ones (content based approach, see [1]). The other approach consists in recommending products to someone by choosing products that have been liked by users that seem to have the same preferences as the person to recommend (collaborative filtering, see [2]).

We propose a recommender that follows the collaborative filtering paradigm. Our algorithm recommends to a particular user a product that will be in the set of its favorite products with high probability. Such recommendations are done according to a categorization of users into equivalence classes with respect to the relation "having the same preferred products". Our algorithm is original since it produces a good recommendation with high probability without knowing the exact rating of each product by each user.

Most of the collaborative filtering work relies on an analysis of users' preferences that take their values in a continuous space (see [3, 2, 4]). In these papers, recommenders are based on the assumption that users can be categorized in classes that strongly differ one from each other. In [5], Kleinberg *et al* use mixture models to make a good recommender. In [6], the authors made a first attempt to design recommenders that are not based on specific assumptions about the internal behavior of a committee (a small set of users that rate a lot of products). Meanwhile, preferences need to be binary values. They are interpreted as "good" or "bad". This assumptions is clearly a drawback since the behavior of customers do not generally obey to such separate agreements. Moreover, the

need for the committee to evaluate all products is required. Several papers agree to consider such a need to be hardly reliable [3, 4, 2, 7, 6].

To the contrary, the essence of our method is to make user evaluate very few products with small discrete rating scale and consider that a committee would never be able to evaluate a large number of products. So our method does not use a bound on the number of products and ask for a committee to rank only a very small number of these products. It means that we do not need a rating of products, a strong assumption used in [4, 8, 9]). While [4] asks for a strong gap between user classes (to be said, orthogonality), we weaken this assumption by asking for user classes to be *sufficiently* different.

The structure of the paper is the following. We present in section 2 our framework. Section 3 sets the condition on products and users for the recommendation system to work. Section 4 presents our recommenders. Last, section 5 shows the effectiveness of our approach through a user satisfaction experiment.

2 Framework and principle of our method

2.1 Our framework : modeling of users and products

The goal of a recommendation system is to provide users with "good" products. In this paper, we consider that users belong to a set $\mathcal{U} = \{u_1 \cdots u_m\}$ of m distinct users and that products come from $\mathcal{P} = \{p_1 \cdots p_n\}$ a set of n distinct products. We also suppose that we are given, even implicitly, a function $f : \mathcal{U} \times \mathcal{P} \longrightarrow \mathbb{R}$ that gives for every couple of user/product a *utility*; f is then a so-called utility function. We can now define a recommendation system as:

Definition 1. *A recommendation system is a function $\mathcal{R} : \mathcal{U} \longrightarrow \mathcal{P}_r$, where $\mathcal{P}_r = \{X \subseteq 2^{\mathcal{P}}, |X| = r\}$. Thus, for each user u_i, $\mathcal{R}(u_i)$ is a set of r products.*

r is a fixed parameter ($r = 5$ in our experiments). Let $\mathcal{F}_r(u_i)$denotes the r favorite products (according to f) of user u_i, we have the following definition.

Definition 2. *A good recommendation occurs when we have $\mathcal{F}_r(u_i) \cap \mathcal{R}(u_i) \neq \emptyset$*

Our goal is to obtain an algorithm that gave good recommendations.

For convenience, we summarize the utilities in a $m \times n$ matrix M_f such that $M_f(i, j) = f(u_i, p_j)$. We denote respectively by $M_f(i, \star)$ and $M_f(\star, i)$ the i^{th} row and j^{th} column of the matrix M_f.

In order to design recommenders, we are interested in top values of a given row $M_f(i, \star)$. These values corresponds to the favorite products of user u_i. These top values are given by an injective function $rank : \mathcal{U} \times \mathcal{P} \longrightarrow [n]$, where $[n]$ denotes the set $\{1, \cdots, n\}$ for $n \in \mathbb{N}$. $rank(u_i, p_j)$ is the index of p_j in the sorted (according to the values of f) list of products for user u_i. For instance, $rank(u_1, p_2) = 3$ means that p_2 is the third preferred product of user u_1.

Using the function $rank$ we can define the notion of $r-equivalence$ for users.

Definition 3. *Two users u_i and u_j are said to be r-equivalent if and only if*

$$\forall p \in \mathcal{P} \quad rank(u_i, p) \leq r \Longleftrightarrow rank(u_j, p) \leq r$$

Intuitively, u_i and u_j are r-equivalent if they have the same r preferred products (but not necessarily with the same order of preference).

We also define a function $index : [n] \times \mathcal{U} \longrightarrow [n]$. $index(*, u_i)$ that indicates the permutation that sort products according to their rank for u_i. We then define an equivalence relation $\equiv_{ps}$ between users. It is called the *product sorting relation*. $\widetilde{\mathcal{U}}$ denotes the quotient space of $\mathcal{U}$ by $\equiv_{ps}$.

Definition 4. *Two users u_k and u_l are equivalent w.r.t. the relation $\equiv_{ps}$ iff*

$$\forall i \leq n \quad index(i, u_k) = index(i, u_l).$$

In this case we write $u_k \equiv_{ps} u_l$.

If necessary, we use a $m \times n$ matrix S, called the sort table, such that $S(i, j) = index(j, u_i)$. This will be only for the sake of clarity in the notations.

We are interested in good recommendations, so we need a notion of equivalence between users that considers only the favorite products of a given user.

Definition 5. *Let $r < n$, two users u_k and u_l are r-equivalent (that is have the same r favorite products) if and only if :*

$$\forall i \leq r \quad \exists j \leq r \quad index(i, u_l) = index(j, u_k)$$

When this happens, we write $u_k \equiv_r u_l$. Moreover this is an equivalence relation.

This relation is important for the rest of the paper since our goal is to deal with only a small numbers of products (here r) in order to give good recommendation. $\widehat{\mathcal{U}}$ denotes the quotient space of $\mathcal{U}$ by $\equiv_r$.

2.2 Principle of our method

We follow the modeling we just defined above. We want our recommendation system to output a good recommendation of r products, where r is a very small integer (typically $r = 5$ in our experiments). It is often admitted that, to get a good recommendation, users follow some kind of behavior which can be viewed as arbitrary distinct classes. In our method we made this natural and formal using the notion of product sorting equivalence and r-equivalence. Thus, we do not admit that users behave the same, exactly or modulo some randomized perturbations, but only tell that there is a model that, given a ranking of products for each user, naturally sort users into equivalence classes. In the following, we will consider cases where both quotient sets $\widehat{\mathcal{U}}$ and $\widetilde{\mathcal{U}}$ have small cardinality in regards to both m and n (numbers of users and products).

All of our work was done according to two assumptions. The first is that we have access to a few users who will rank a set of witness products and give their r favorite products. These users are known as "the committee". The second assumption is that we are authorized to ask every people about these witness products. This two hypothesis allow us to say that we use partial information in order to make its recommendation. The main issue is then to understand what should be the size of the committee, and how many witness products we need in order to be able to provide a good recommendation to users.

Our recommender is described in Fig.1. We first choose a committee and ask to each member of the committee to sort some products according to its preferences. Then we choose a set of witness products and ask all users to sort those products. Then we can cluster, with high confidence, users into equivalence classes according to their products sorting. The given clustering will likely attribute at least one member of the committee to each equivalence class. This peculiar user will be used to make recommendations to members of its class.

In the following we note by C the committee (thus $C \subseteq \mathcal{U}$) and by W the witness products ($W \subseteq \mathcal{P}$). As said previously, $\widehat{\mathcal{U}}$ and $\widetilde{\mathcal{U}}$ are the quotient sets of $\mathcal{U}$ with respect to, respectively, $\equiv_r$ and $\equiv_{ps}$. Classes of $\widetilde{\mathcal{U}}$ (resp. $\widehat{\mathcal{U}}$) are denoted by $\widetilde{u_i}$ (resp. $\widehat{u_i}$) for i ranging from 1 to $|\widetilde{\mathcal{U}}|$ (resp. $|\widehat{\mathcal{U}}|$). For the sake of clarity, we use θ as a notation for the cardinal of $\widetilde{\mathcal{U}}$ and θ_i for the cardinal of class $\widetilde{u_i}$.

The next section is devoted to the calculus of $|C|$ and $|W|$ in order to make a good recommendation. Our goal is to find W and C such that the following holds: let $u_i \in C$ and $u_j \in \mathcal{U}$. If $u_i \equiv_{ps} u_j$ for products from W then $u_i \equiv_r u_j$ on $\mathcal{U}$ with high probability. Note that it implies that with high probability we can have a good recommendation for u_j by giving him the favorite of u_i.

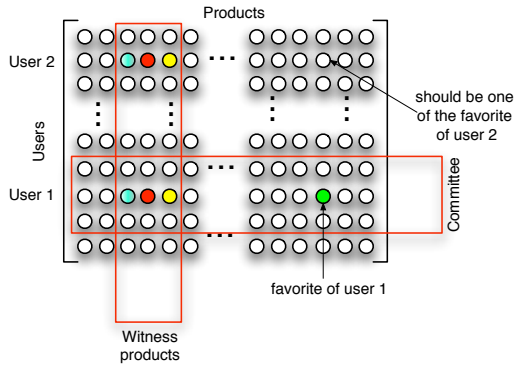

Fig. 1. Principle of our method

3 Recommendation under partial information

We now address the problem of the size of both the committee and the set of witness products in order to provide good recommendations with high probability. We first consider the problem of the size of the witness products set.

3.1 Cardinality of the witness products set

We consider that the relation between products is preserved, e.g. if a user prefers product p_i to product p_j, he will provide a higher utility value for p_i than for

p_j. This only means that users always sort products the same way. This may not be true that users evaluate utilities the same way, regardless of the proposed products. Sorting information may look weaker than utility in order to perform recommendations, but it is clearly more robust : context or even mood can change the actual value of utility. A users does not have the will to be rigorous but it will certainly keep its preferences. Thus, this is one of the assumption on which we build our algorithm.

We now look at bounds for $|W|$, assuming that θ is the number of classes of users, regarding the relation *having same product sorting* ($\equiv_{ps}$). We start by giving a combinatorial result (proof in appendix) which will ensure bounds for our recommendation system to work:

Proposition 1 (sub-permutation). *Let τ be a permutation over $[n]$ and W be a subset of $[n]$ of cardinality $|W|$. There exists a unique sub-permutation $\tau|_W$ which is the one-to-one mapping from W to $[|W|]$ that satisfies :*

$$\forall k, l \in W \quad \tau|_W(k) < \tau|_W(l) \Leftrightarrow \tau(k) < \tau(l) \tag{1}$$

Lemma 1. *Let $\{\tau_i; i \leq \theta\}$ be a family of θ distinct permutations over $[n]$. To distinguish all τ_i between them with their sub-permutations, using a single $W \subset [n]$ of cardinality $|W|$, we need at most $|W| = \inf\{n; 2(\theta - 1)\}$ elements.*

Proof. The proof is in the appendix.

Observe that the proof of the lemma defines an algorithm which computes the set W. Its time complexity is $\theta \cdot n$, which is just the input size. This algorithm is optimal among deterministic techniques. We will prove that we can get good recommendations with high probability when taking less than $2(\theta - 1)$ products.

We now rephrase the two last results in term of recommendation :

Proposition 2. *Let $W \subset \mathcal{P}$ be of cardinality $|W|$ and $\widetilde{\mathcal{U}} = \{\widetilde{u_1}, \cdots, \widetilde{u_\theta}\}$ be the quotient set of $\mathcal{U}$, w.r.t.* having the same product sorting *relationship, where $\theta \leq \lceil n/2 \rceil$. Suppose we are given $M_f(l_i, \star)$ for at least one user $u_{l_i} \in \mathcal{T}_i$ for all $i \leq \theta$ (or, which is enough, an ordering of all products by user u_{l_i}). Then we need at least $|W|$ products in the witness products set in order to make good recommendation to all users where $|W|$ is such that $(|W|)! \geq \theta$ and $|W| \leq 2(\theta - 1)$.*

Proof. We prove the proposition in two steps.
Lower Bound: It is sufficient to note that when extracting the ordering of $|W|$ products for a user u_j, the total number of possible different ordering is exactly $(|W|)!$. Thus, it becomes clear that if θ is the number of different classes of users among $\mathcal{U}$, we obviously need $|W|$ such that $(|W|)! \geq \theta$ in order to distinguish all different classes and, thus, making a *good recommendation* to u_j.
Upper Bound : From $M_f(l_i, \star)_{i \leq \theta}$, we can compute a family of functions $index(*, u_{l_i})$. For a given l_i, the function $index(*, u_{l_i})$ can be seen as a permutation over $[n]$.
We can now use proposition 1. We need a subset of $[n]$ of cardinality no more

than $2(\theta - 1)$ to distinguish each of these θ permutations with their associated sub-permutations. This means that we need $|W| \leq 2(\theta - 1)$ products to make every functions $index(*, u_{l_i})$ one different of each others.

Finding exactly the $|W|$ products that fit with the lower bound may be a very difficult task. If we choose these products uniformly at random they will be unlikely to ensure to distinguish between classes. Then, it would certainly be easier to choose more products for the witness products set in order to be able to have enough information. The upper bound means that it is always possible to make good recommendations, under condition that every class of user is known via one of its representant, with at most $2(\theta-1)$ known products per user. Thus, we add to our recommender a specific algorithm to select products that every people should evaluate.

Although we have these lower and upper bounds, we feel concerned with finding the number of products needed in W in order to achieve a fast but good recommendation. Proposition 3 below gives the solution.

Proposition 3. *Let $W \subseteq \mathcal{P}$ a set of products and $\widetilde{\mathcal{U}}$ be as before. Suppose we know for all $i \leq \theta$, $M_f(l_i, \star)$ for at least one user u_{l_i} such that $u_{l_i} \in \widetilde{u_i}$. Then if we pick uniformly at random $|W| = \lceil \sqrt{\theta} \rceil$ elements from $\mathcal{P}$, we can correctly sort, with high probability, every user in its class (e.g. equivalence class w.r.t. the $\equiv_{ps}$ relation) by looking the ordering of those $|W|$ products for each user.*

Proof. Let Dst be the event *every $M_f(l_i, \star)|_W$ is distinct*, when taking $|W|$ elements from $\mathcal{P}$ with uniform distribution. This event can be observed as distinguishing each class from $\widetilde{\mathcal{U}}$ with $|W|$ products. Hence, this means exactly that, taking one user from each class as a witness of his class, we do not want two distinct of these users to be represented asame (i.e. with the same ordering over these $|W|$ products). The probability of the event Dst is given by :

$$\Pr[Dst] = \prod_{i=0}^{\theta-1} \frac{(|W|)! - i}{((|W|)!)} \quad \Leftrightarrow \quad \ln(\Pr[Dst]) = \sum_{i=0}^{\theta-1} \ln\left(1 - \frac{i}{(|W|)!}\right)$$

Which leads to, using power series :

$$\ln(\Pr[Dst]) = \sum_{i=0}^{\theta-1} \sum_{j \geq 1} \frac{(-1)^j}{j} \left(\frac{i}{(|W|)!}\right)^j = \frac{-1}{(|W|!)} \sum_{i<\theta} i \quad + o(\tfrac{1}{(|W|)!})$$

Finally, we get: $\ln(\Pr[Dst]) = \frac{\theta - \theta^2}{2(|W|)!} + o(\frac{1}{(|W|)!})$, which leads to the approximation: $\Pr[Dst] \approx e^{-\frac{(|W|)^4}{2(|W|)!}}$. This is close to 1, thus the proposition holds.

From now on, we know how much products we have to choose in our witness products set in order to achieve good recommendation with high probability. We now address the problem of the number of users that must be in the committee.

3.2 Cardinality of the committee

In order to cluster users into classes that depend on the product sorting equivalence, we need a committee that will evaluate all products of the set W.

This committee is a subset C of $\mathcal{U}$ whose members must be chosen in a way or an other. After choosing how we sample users for being in the committee, we must evaluate how much of these users we need. As in the case of witness products, it is easier to have people chosen uniformly at random. We should then study the behavior of a recommendation system that pick users for its committee with uniform distribution. We thus define the notion of good (e.g. representative) committee.

Definition 6. *A committee C is representative of $\mathcal{U}$ if and only if :*

$$\forall \widetilde{u_i} \in \widetilde{\mathcal{U}}, \quad \exists u \in C \text{ such that } u \in \widetilde{u_i}$$

Our goal is then to pick enough users in order to have a representative committee, that is obtaining at least one member of each class of users (w.r.t. the relation *having same product sorting*) with high probability.

We then have to evaluate the probability of getting one member of each of such class when asking $|C|$ users from $\mathcal{U}$ at random with uniform distribution, where this last event will be written GC for *Good Committee*. It is given via the probability of the opposite event (i.e. not having a representative committee) :

$$\Pr[\neg GC] = \sum_{i=1}^{\theta} \left(1 - \tfrac{\theta_i}{m}\right)^{|C|}$$

We recall that θ_i stands for the cardinal of class $\widetilde{u_i}$ and $m = |\mathcal{U}|$.

We make some reasonable assumptions about θ_i and θ. It seems natural to assume that $\theta \ll m$, that is the number of classes, is small w.r.t. the number of users. We also suppose that every class of users contains a non negligible number of users. Formally, these facts can be expressed as $\theta = O(1)$ and $\theta_i = \Theta(m)$ with associated multiplicative constant $q_i < 1$ for all $i \leq \theta$. That is $\theta_i = q_i \cdot m$. In the following we consider, without loss of generality, that $q_1 < q_2 < \cdots < q_\theta$ so that the size of the $\widetilde{u_i}$ is increasing with i.

These assumptions are reasonable since, in practice, we are not interested in providing good recommendations to users that belong to not large enough classes. Providing users in small classes with good recommendations will increase massively the size of the committee only to satisfy few more users.

We can now compute the probability $\Pr[\neg GC]$, written ε from now on :

$$\varepsilon \approx \sum_{i=1}^{\theta} (1 - q_i)^{|C|} \leq \theta \, (1 - q_1)^{|C|} = \theta e^{|C| \ln(1 - q_1)}$$

This can be equivalently written as :

$$\ln\left(\tfrac{\varepsilon}{\theta}\right) \leq |C| \ln(1 - q_i) \Longleftrightarrow |C| \geq \tfrac{1}{q_1} \ln\left(\tfrac{\theta}{\varepsilon}\right)$$

Observe that, under our assumptions, we have $q_1 = \Theta(1)$. It leads to :

$$|C| \geq q\theta \ln\left(\tfrac{\theta}{\varepsilon}\right) = O(\theta \ln(\theta/\varepsilon))$$

We can now summarize this result into the following proposition :

Proposition 4. *Suppose that $\theta = O(1)$ and $\theta_i = \Theta(m)$ with associated multiplicative constant $0 < q_i < 1$ for all $i \leq \theta$. Then a committee C is representative of $\mathcal{U}$ with probability $(1 - \varepsilon)$ if and only if $|C| = O(\theta \ln(\theta/\varepsilon))$*

Note that, here, θ is a constant, so $|C| = O(\ln(1/\varepsilon))$. It means that the size of the committee only depends on the targeted precision of the recommender.

4 Recommenders

In this section, we use the results of the previous sections to design a family of recommendation systems. Each of these recommendation systems depends on how we collect the information needed for initializing the algorithm.

We present here two of these algorithms, probably the most naturals that can be constructed using our framework. First we present two different ways to collect the informations that are needed by our algorithm. Please note that θ, the number of equivalence classes of users w.r.t. the relation "having same product sorting", is a parameter of the two initialization process. The natural question is then how to choose the value of this parameter in order to make the algorithm usable ? In practice θ is a constant known via users' polls. But if we consider the more general case where $\theta = O(\ln m)$, our algorithm is still efficient.

The first initialization process is used if one has only access to values of the utility function for users from the committee C.

Initialization case 1
- Set $C = \{u_{h_i}; i \leq |C|\}$ by picking $|C| = q\theta \ln\left(\frac{\theta}{\varepsilon}\right) = \Theta(\theta \ln(\frac{\theta}{\varepsilon})$ users from $\mathcal{U}$.
- Set W by picking $|W| = \max(8\ ;\ \lceil\sqrt{\theta}\rceil)$ products from $\mathcal{P}$.
- For all $u_k \in C$, Extract $\mathcal{F}_r(u_k)$ from $\mathcal{P}$.
- For all $u_k \in C$, build the family of functions $index|_W(\star, u_k)$. This family is represented as a restriction to C and W of the matrix S defined in subsection 2.1. We denote this matrix as $S_{C,W}$

The second initialization process is used when the user of the recommender can ask to committee members their r favorite products in P without rating all products in P. In our practical experiment, we use this initialization method.

Initialization case 2
- Set $C = \{u_{h_i}; i \leq |C|\}$ by picking $|C| = q\theta \ln\left(\frac{\theta}{\varepsilon}\right) = \Theta(\theta \ln(\frac{\theta}{\varepsilon})$ users from $\mathcal{U}$.
- Set W by picking $|W| = \max(8\ ;\ \lceil\sqrt{\theta}\rceil)$ products from $\mathcal{P}$.
- Ask each user from C about his r favorite products. Build a $|C| \times r$ table containing this information.
- Ask every member of the committee to sort products of W. Use this to build the family of functions $index|_W(\star, u_k)$. This family is represented as a restriction to C and W of the matrix S defined in subsection 2.1. We denote this matrix as $S_{C,W}$.

Note that $S_{X,Y}$ will denotes the restriction of S to users from X and products from Y. Moreover, $S(i,j) = index(j, u_i)$. We now give the algorithm that uses either initialization case 1 or 2.

Algorithm

Input : u_i from $\mathcal{U}$ represented by its sorting on products from W. This can be seen as a vector $V_i = \big(index|_W(1, u_i), \ldots, index|_W(|W|, u_i)\big)$.

Output : The r-recommended products for u_i.

Behavior :

1. Compute $S_{C,W} \cdot V_i := {}^t w(i)$
2. Set $J = \{j \leq |C| : w(i)_j = \max_{l \leq |W|}\{w(i)_l\}\}$ where $w(i) = (w(i)_1 \cdots w(i)_{|W|})$
3. Take $j_0 \in J$ at random uniformly
4. Outputs the r favorite products of committee user u_{j_0} corresponding to ${j_0}^{th}$ row of matrix $S_{C,W}$ (This is $\mathcal{F}_r(u_{j_0})$).

For this algorithm, the following theorem holds :

Theorem 1. *The above algorithm gives good recommendation to a given user with probability at least* $1 - (\varepsilon + \eta)$*, where* $\eta = 1 - e^{-\frac{|W|^4}{2(|W|)!}} = o(1)$ *and* ε *is such that* $|C| = q\theta \ln\left(\frac{\theta}{\varepsilon}\right)$*. Its time complexity is* $O(\theta\sqrt{\theta}\ln\left(\frac{\theta}{\varepsilon}\right))$*.*

Proof. We have at least one user from each of the θ classes with probability $(1-\varepsilon)$ according to proposition 4. Let user u_i be the input of our algorithm. As there exists $j \leq \theta$ so that user $u_i \in \widetilde{u_j}$, we get that user u_i has a member of its class in C with the same probability $(1-\varepsilon)$. In this case, let us define $ref(i) = C \cap \widetilde{u_j}$. By proposition 3, since $|W| = \lceil\sqrt{\theta}\rceil$, two users from different classes have different corresponding rows in $S_{C,W}$ with probability $e^{-\frac{|W|^4}{2(|W|)!}} = 1 - \eta$.

Hence, there is a probability at least $1 - (\varepsilon + \eta)$ that the committee contains at least one user from each class (i.e. the committee is representative) and that its members have same representative in the matrix $S_{C,W}$ computed so that $S_{C,W}(k, \star) = V_i$ if and only if user $u_{h_k} \in ref(i)$. Thus, picking any of the user in $ref(i)$ will provide someone who share same product ordering as user u_i so that recommending his top r favorite products is a good recommendation for user u_i. It remains now to show that the set J is exactly $ref(i)$. For this we use the following lemma which is a direct application of the Cauchy-Schwarz inequality.

Lemma 2. *Let* τ *and* σ *be permutations over* $[N]$*,* $N \in \mathbb{N}^\star$*. We have that* $\sum_{i=1}^{N} \tau(i) \cdot \sigma(i)$ *is maximum if, and only if,* $\tau = \sigma$*.*

Hence, since every row l in $S_{\mathcal{U},W}$ can be seen as the restriction of the *index* function to the set W of witness products, this corresponds to enumerating every image of a sub-permutation. Thus, the vector ${}^t w(i)$ consists of the scalar product given by the previous lemma, so that $w(i)_k$ reaches its maximum if, and only if, $S_{C,W}(k, \star) = V_i$, permitting us to conclude.

The time complexity is in $O(|C| \cdot |W|) = O(\theta\sqrt{\theta}\ln\left(\frac{\theta}{\varepsilon}\right))$. Computational complexity for finding maximum coordinates of list $w(i)$ and taking j_0 at random among corresponding indexes can be neglected because of the O.

With the previous theorem, we have a recommendation system for recommending to one user a set of r products. We now see what happens if we use this algorithm for making recommendations to all users of $\mathcal{U}$.

Corollary 1. *When adding a loop to the beginning of the algorithm in order to make recommendation for every user in $\mathcal{U}$, the time complexity of the algorithm is $O(m+n)$ when using initialization case 1 and $O(m)$ when using case 2, assuming that we both have $|C| = O(1)$ and $|W| = O(1)$. Thus, the fact that* every user gets a good recommendation *happens with probability* $1-(\varepsilon+\eta)$.

5 Experiments

unknown / good	0		1		2		3		4		5	
	Rec	Ran	Rec	Ran	Rec	Ran	Rec	Ran	Rec	Ran	Rec	Ran
0	0%	7%	1.1%	5.2%	1.3%	1.6%	4.7%	8%	22.2%	23.7%	100%	100%
1	4.3%	20.7%	5.7%	22.4%	16%	31.1%	16.3%	37.1%	77.8%	76.3%	0	0
2	19.1%	17.2%	14.8%	24.1%	20%	39.3%	**79%**	**54.9%**	0	0	0	0
3	23.4%	24.1%	33%	31%	**62.7%**	**28%**	0	0	0	0	0	0
4	19.1%	13.8%	**45.4%**	**17.3%**	0	0	0	0	0	0	0	0
5	34.1%	17.2%	0	0	0	0	0	0	0	0	0	0

Table 1. Percentage of recommendations regarding the number of good and unknown recommended products

In order to validate the effectiveness of our approach we decided to make an experiment with actual products and users.We selected from a set of 4400 movies, 160 of them uniformly at random. We then extract uniformly at random from this data set 9 movies (our witness products set).

Our methodology was then the following. From the people that volunteer to appear in the committee, we extract (at random) 20 of them. It was then asked, through a web site, to the committee members to sort the 9 movies and then to choose their 5 favorite movies in the data set (these 5 movies were called the selection). We then asked as many people as possible to use our recommendation engine to see if it is effective. 270 people have used it so far, the experiment is ongoing so the presented results are only partial but still trustworthy. The protocol was the following, first a user is asked to sort the 9 witness movies and then we offer him two recommendations. The first one is provided by the recommendation system presented in this paper and the second recommendation is simply composed of 5 movies chosen uniformly at random in the data set. The users are then asked two questions for each recommendation, how many films do they like in the recommendation and how many films do they actually know in the recommendation.

Fig.2. It shows the percentage of recommendations that contains at least a given number of good recommended products (e.g. products liked by user). It first gives the percentage of good recommendations (according to the definition 2). Our method outperforms the random recommendation since we achieve a percentage greater than 95% while the random recommendation only achieves less than 85%. Moreover, we can see that our recommender also provides higher order good recommendations. On the other hand, random choices' effectiveness drops quickly and it often fails to provide users with more than 2 good products.

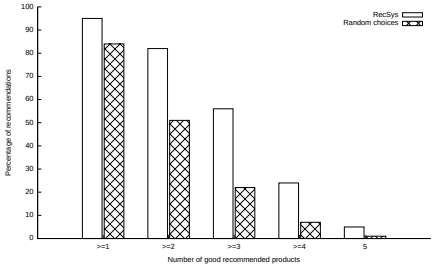

Fig. 2. Percentage of recommendations containing at least x good products

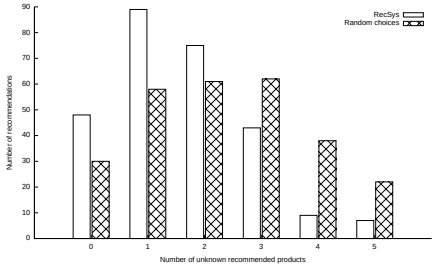

Fig. 3. Number of recommendations with x unknown products

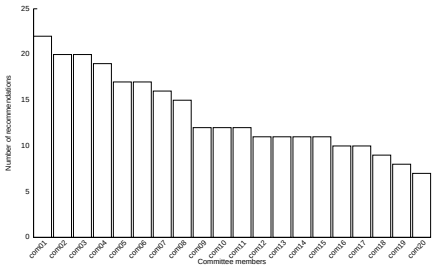

Fig. 4. Number of recommendations made by each committee member

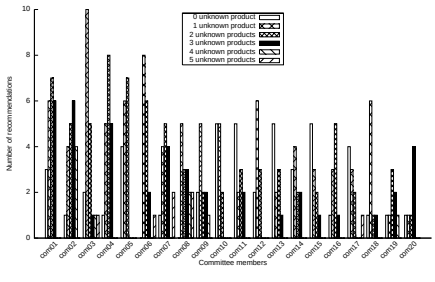

Fig. 5. Number of recommendations with x unknown products made by each committee member

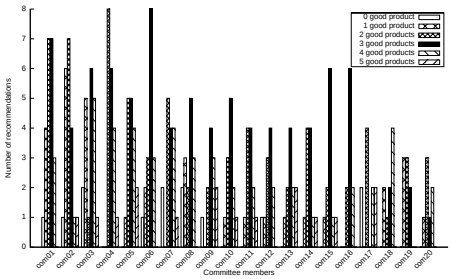

Fig. 6. Number of good recommendations made by each committee member

Fig.3. It shows the number of recommendations w.r.t. the number of unknown products recommended. A too large number (i.e. 4 or 5) of unknown products seems to indicate a poor quality of recommendation since we are here dealing with well known movies: if one does not know items recommended too him, it is likely that they are in fact movies he did not want to see. It is also important to note that if this number is too high it will decrease the user's confidence in the recommendation. We can clearly see on Fig.3 that this number decrease much faster with our algorithm than with the random choices.

Table 1. We compare the "quality" of the recommendations made by both techniques. Meaning we want to compare the number of good recommended products w.r.t. the number of unknown products in the recommendation. The columns

are indexed by the number nu of unknown products in the recommendations and the rows are indexed by the proportion of recommendations with ng good products. We see that when $(ng, nu) \in \{(2,3),(3,2),(4,1)\}$ our algorithm outperforms the random choices. These cases are interesting because they concern good recommendations where $(ng + nu = 5) \wedge (ng > 1) \wedge (nu < 5)$, thus the user is confident in the recommendation (he liked all the movies he knows in the recommendation) and will probably consult the unknown products.

Figs.4, 5 and 6. These figures consider each committee member separately. We see in Fig.4 that most recommendations are given by only a few users, but that there are no users that make zero recommendation. We also see that the percentage of good products and unknown products in each committee member is approximately the same for every member and does not depend on the number of recommendations.

Experiments show the effectiveness of our approach: it is possible to provide users with good recommendations with high probability but low complexity.

6 Conclusion

In this paper we have presented a new recommendation system based on weaker assumptions than previous ones. Our recommender is as efficient in terms of time complexity and probability of having a good recommendation as other recommendation systems. A user satisfaction experiment supports these results.

References

1. Allen, R.B.: User model : theory methods and practice. International Journal of Man-Machine Studies (1990) 511 – 543
2. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Communications of the ACM (1992) 61 – 70
3. Awerbuch, B., Azar, Y., Lotker, Z., Patt-Shamir, B., Tuttle, M.: Collaborate with strangers to find own preferences. Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures (2005) 263 – 269
4. Drineas, P., Kerenidis, I., Raghavan, P.: Competitive recommendation systems. Annual ACM Symposium on Theory of Computing, Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (2002) 82–90
5. Kleinberg, J., Sandler, M.: Using mixture models for collaborative filtering. Journal of Computer and System Sciences **74**(1) (2008) 49 – 69 Learning Theory 2004.
6. Awerbuch, B., Patt-Shamir, B., Peleg, D., Tuttle, M.: Improved recommendation systems. Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (2005) 1174 – 1183
7. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM (1997) 56–58
8. Mahoney, M.W., Maggioni, M., Drineas, P.: Tensor-cur decompositions for tensor-based data. In Eliassi-Rad, T., Ungar, L.H., Craven, M., Gunopulos, D., eds.: KDD, ACM (2006) 327–336

9. Kleinberg, J., Sandler, M.: Convergent algorithms for collaborative filtering. In: EC '03: Proceedings of the 4th ACM conference on Electronic commerce, New York, NY, USA, ACM (2003) 1–10

7 Appendix

Proof. (of proposition 1).

The existence is straightforward, so it remains to prove the unicity. Let $[n]$ and W be such fixed sets and suppose that there exists two different sub-permutation τ_1 and τ_2 of τ that satisfy the propriety (1). We then consider the smallest $i \in [|W|]$ such that $\tau_1(k) = \tau_2(l) = i$, with $k \neq l$ elements from W. It follows that $\tau_1(l) > \tau_1(k)$ and $\tau_2(k) > \tau_2(l)$: otherwise, $\tau_1(l)$ or $\tau_2(k)$ would be lower than i making k and l equal. But then, we have by (1) that $\tau(l) > \tau(k)$ and $\tau(k) > \tau(l)$, a contradiction. In any case, we are lead to $\tau_1 = \tau_2$ showing that the function is unique when W is fixed.

Proof. (of lemma 1).

By induction on the number θ of distinct permutations. The result stands for $\theta = 1$. As this is obvious for $\theta \geq n$, we will assume $\theta < n$.

Let us consider that $W \subset [n]$ is a set of cardinality at most $2(\theta-1)$ satisfying $\forall i, j \ \leq \theta \ \ \tau_i|_W \neq \tau_j|_W$. Recall that is means $\tau_i(a) < \tau_i(b)$ while $\tau_j(a) > \tau_j(b)$ for some a, b belonging to W, a and b being case-specific for each $(i; j)$ with $i \neq j$. We then compare $\tau_{\theta+1}|_W$ to the $\tau_i|_W$'s. If they are all different, then we are done. Else, there exists some $i \leq \theta$ so that $\tau_i|_W \neq \tau_{\theta+1}|_W$. Moreover, such an i is unique among $[\theta]$, otherwise it would contradict the assumption that W allows to distinguish between $\tau_1, \ldots, \tau_\theta$. As all the $\theta+1$ permutations are distinct, there exists a, b such that $\tau_i(a) < \tau_i(b)$ while $\tau_{\theta+1}(a) > \tau_{\theta+1}(b)$. Let $W' = W \cup \{a; b\}$. It must be the case that $\tau_i|_{W'} \neq \tau_{\theta+1}|_{W'}$. Finally, $\tau_{\theta+1}|_W \neq \tau_j|_W \Rightarrow \tau_{\theta+1}|_{W'} \neq \tau_j|_{W'}$ so that W' permits to distinguish every of the $\theta + 1$ permutations and we have that $|W'| \leq 2(\theta - 1) + 2 = 2\theta$, and the result follows.