A Multi-Armed Bandit Model Selection for Cold-Start User Recommendation

Crícia Z. Felício
Instituto Federal do Triângulo Mineiro
Uberlândia, MG, Brazil
cricia@iftm.edu.br

Klérisson V. R. Paixão
Universidade Federal de Uberlândia
Uberlândia, MG, Brazil
klerisson@ufu.br

Celia A. Z. Barcelos
Universidade Federal de Uberlândia
Uberlândia, MG, Brazil
celiazb@ufu.br

Philippe Preux
Université de Lille & CRIStAL
Villeneuve D'ascq, France
philippe.preux@inria.fr

ABSTRACT

How can we effectively recommend items to a user about whom we have no information? This is the problem we focus on in this paper, known as the cold-start problem. In most existing works, the cold-start problem is handled through the use of many kinds of information available about the user. However, what happens if we do not have any information? Recommender systems usually keep a substantial amount of prediction models that are available for analysis. Moreover, recommendations to new users yield uncertain returns. Assuming that a number of alternative prediction models is available to select items to recommend to a cold user, this paper introduces a *multi-armed bandit* based model selection, named PdMS. In comparison with three baselines, PdMS improves the performance as measured by the *nDCG*. These improvements are demonstrated on real, public datasets.

CCS CONCEPTS

•**Information systems → Recommender systems;** *Retrieval effectiveness;* Personalization;

KEYWORDS

Recommender system; Cold-start problem; Model selection; Multi-armed bandits

1 INTRODUCTION

Given a cold user, such as a new Netflix' client, how can we effectively recommend movies to him? In most existing works, a typical recommender system will request initial ratings [11, 49] and/or it will harvest the World Wide Web looking for the user's tastes [22, 44] to bootstrap the system. But apart from that, what happens if we do not have the right information to build the user's profile? So, how can we estimate the tastes of a new user without prior side information? The focus of this paper is on offering better recommendations for such cold users.

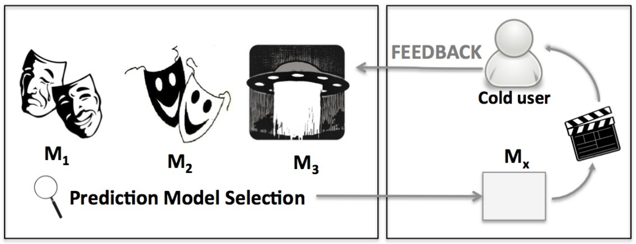


Figure 1: PdMS relies on feedback to learn how to appropriately select a consensual prediction model to deliver more accurate recommendations to cold users.

Different prediction models are used to deal with distinct stages of a user experience. For example, a particular model works better in earlier stages when the recommender system does not know much about the user. Then, as the number of interactions increases, an other model may become more effective, and therefore, the recommendation system switches to the more powerful model. Switching methods [9] were designed to handle the cold-start problem. The idea is to switch from one model to an other once the system has enough data about the user, so that the user is no longer cold.

While the concept of switching models [6] is not new, the availability of several cold-start methods provides enriched resources to *model selection*. Applied to the cold-start stage, a model selection method may be seen as a framework to alternate among prediction models to find the most suitable one at each time, as the user warms-up. Few works have sought to empirically assess the efficacy of a model selection, specifically at the cold-start stage [8, 55].

In this paper, we propose PdMS (Figure 1), an effective *Prediction Model Selection* method to deal with cold users. Many proposed solutions to deal with this problem use side information, which may be of different kinds, such as social information [2, 13], user click behavior [37, 53], location-based information [12, 41], user's visual perception [16, 17], and, more broadly speaking, contextual information [1]. Particularly, we delve into user feedback, but without prior information about the user.

UMAP'17, July 9–12, 2017, Bratislava, Slovakia
© 2017 ACM. 978-1-4503-4635-1/17/07...$15.00
DOI: http://dx.doi.org/10.1145/3079628.3079681

The first insight is to explore how a model selection can be useful at all to provide better recommendations to cold users. The reason we choose to select prediction models instead of building new ones is that these models have been tuned based on users already encountered by the recommender system: these models have already captured a lot of information about the users of the system so that it is *a priori* a good idea to try to frame any new user into an already known category; so, a cold user might be best served by one already tuned model available in the system [20].

The second insight is to consider our goal as a multi-armed bandit (MAB) problem [3]. Recommender systems dealing with a new user repeatedly propose items yielding uncertain returns. In this situation, we face a problem of sequential decision-making under uncertainty: an action amounts to selecting a particular model to serve a particular user; actions have to be selected to find a suitable prediction model for the user at hand, and to investigate other ones (explore vs. exploit). The idea is to be benefit from the relative performance of various prediction models [14], an idea also reminiscent of ensemble methods. Therefore, a model selection that maximizes the recommendation gain might be more precise.

Our primary contributions are:

- We show how to formalize the model selection problem faced by the recommendation system as a multi-armed bandit problem.
- We introduce PdMS, an effective approach to deal with cold users, *i.e.* users for whom no prior side information is available.
- We empirically put PdMS to the test on four real, public datasets.

This paper is organized as follows. First, we further motivate the need for a new cold-start method and its main concept (Section 2). Next, we present the approach and how it works (Section 3). Section 4 describes our experimental settings. Section 5 discusses the results of the experiments. Then, Section 6 discusses related work, and Section 7 concludes the paper.

2 BACKGROUND

As a motivating example, let us consider the domain of movie recommendations. Based on the ratings already collected we can apply a recommendation algorithm and make predictions for the movies not already rated using the personalized preferences from each user. User's preferences might indicate a preferred movie genre, director, actors, etc, and be represented by a prediction model.

Now, let us assume a new (cold) user u is looking for movies he would like to watch. A common situation is that the new user has not yet given any rating and no other information is available about him. However, he is admitted into the system, without providing any profile information. The new user probably has similar tastes to some other users in the system. Then for the initial recommendations, one way to recommend to him can be use a prediction model which is dealing correctly with other users.

Assume that for each recommended movie the cold user gives a feedback that will be used to make the next recommendation. So, the recommender system will first explore the different prediction models and then use feedback to select the most appropriate model for this user, at his current stage of interaction with the recommender system.

Before going any further, we introduce some terminology and notations on recommender systems, and on multi-armed bandits.

Let U be a set of m users and I be a set of n items. We assume that each user $u \in U$ and each item $i \in I$ has a unique identifier. The user-item rating matrix is $R = [r_{u,i}]_{m \times n}$, where each entry $r_{u,i}$ is either the rating given by user u on item i, or unknown. In a live system, as new ratings are entered by users, the rating matrix R depends on time t, so that the notation should be R_t; in the paper, we often drop this t index as long as the meaning of sentences is not in jeopardy. The recommendation task is based on the prediction of the missing values of the user-item rating matrix. Then, prediction models are used to rank items and recommend the k top-ranked.

A Multi-Armed Bandit problem is a sequential decision problem where an algorithm continually chooses among a set of possible actions (arms) which we assume to be finite in this paper. At each time step t, an arm a is selected and pulled which leads to a reward $X_a(t)$. This reward is distributed according to a certain unknown law. Here, we consider that the goal is to learn, as fast as possible, through repeated arm pulls, the arm that returns the maximum expected reward.

In this work, we assume a set of prediction models as the arms from Multi-Armed Bandit problem. Then, a bandit algorithm is sequentially applied to choose among the prediction models either the best performing one at the moment (exploitation), or an other model to better estimate its performance (exploration). We rely on the $UCB1$ algorithm [3] and ϵ-Greedy algorithm [54] to implement our model selection. $UCB1$ maintains the mean reward of each arm (prediction model) a, denoted by $\bar{X}_a$. Each time arm a is played, the mean reward $\bar{X}_a$ is updated. The number of pulls of arm a is denoted by n_a. In both notations, t is implicit.

In UCB1, each arm is initially pulled a couple of times to estimate its mean reward. Then, at turn t, UCB1 selects arm $a(t)$ according to Equation (1).

$$a(t) = \arg \max_{s=1\ldots k} \left(\bar{X}_{a_s} + \sqrt{\frac{2 \ln t}{n_{a_s}}} \right) \tag{1}$$

The chosen arm is the one maximizing the sum of the mean reward $\bar{X}_a$, and a confidence term $ba(t) := \sqrt{\frac{2 \ln t}{n_a}}$. At each t, the arm to be played is selected as the one maximizing the balance between immediate profit and the gathering of useful information.

UCB1 is particularly appealing because of its theoretical guarantees: as the number of pulls increases, we know that the number of sub-optimal arms being pulled grows as the logarithm of the number of pulls, that is very slowly; we also know that it can not grow slower than that.

UCB1 with tunned constant is a variant of UCB1 where the confidence term is represented as $\sqrt{\frac{C \ln t}{n_a}}$, where C is the exploration constant to tune.

ϵ-Greedy also maintains the mean reward of each arm (prediction model) a, denoted by $\bar{X}_a$. At each round, the ϵ-Greedy algorithm selects the arm with the highest mean reward with probability $1 - \epsilon$, and selects an arm uniformly at random with probability ϵ.

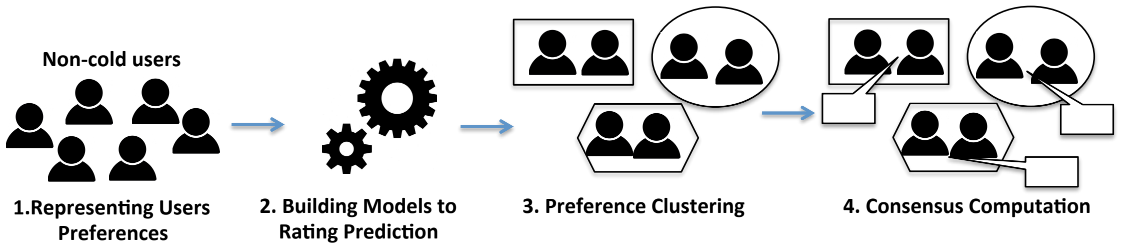

Non-cold users
1.Representing Users Preferences
2. Building Models to Rating Prediction
3. Preference Clustering
4. Consensus Computation

Figure 2: Overview of PdMS model building illustration.

The ϵ-decreasing greedy variant consists in slowly decreasing ϵ along time. Hence, the algorithm moves from purely random ($\epsilon = 1$) to purely greedy ($\epsilon = 0$). The theoretical guarantees of this selection strategy are not as good as for UCB1. However, it is known to often perform very-well in practice.

3 PdMS APPROACH

To exemplify the problem, let us consider a toy example: a user-item matrix is represented in Table 1a. We apply a matrix factorization algorithm to complete it, namely the BiasedMF[1] algorithm [32]: this step is detailed in section 3.1. We obtain the predicted ratings as shown in Table 1b.

At this point, we depart from the standard approach. Keeping in mind that this is just a toy example given for the sake of explaining our approach, we consider that each row of Table 1b is a prediction model.

When a new user comes into the system, we can choose one of the 5 prediction models to make recommendations for him. However this is not reasonable in a real-world recommendation system, where we have millions of users. Then, our first challenge is to reduce the number of prediction models available to the switch process. Considering the rows of the rating matrix, it is possible to cluster them: each cluster is made of users that basically assign the same ratings to the same items. It is a common assumption in matrix factorization that the rows of rating matrices can be clustered (that's actually the reason why rating matrices have low rank).

According to this intuition, we define PdMS as an algorithm made of two phases: (i) computing and updating prediction models and (ii) recommendation. Phase (i) is presented in the next section; it is based on our prior work [20]. Phase (ii) is then presented in the subsequent section.

3.1 Model Computing and Updating

To define the set of prediction models, we apply three steps: Rating prediction, Preference clustering, and Consensus computation. We illustrate this process in Figure 2, where the first step consists in obtain the user-item rating matrix of non-cold users and step 2, 3, and 4 are explained as follows.

Rating prediction. From the user-item rating matrix, composed only for non-cold users, we use a matrix factorization technique to get a matrix of predicted ratings R'. R' is expressed as a product of latent factors, $R' = PQ^T$, where P is the user latent factor matrix, and Q is the item latent factor matrix. The predicted rating of the item i_k by user u_j is $R'_{u_j,i_k} = predict(u_j, i_k, P, Q)$, the details of this function depending on the completion method being used.

As an example, Table 1b shows the predicted rating matrix R' obtained from the user-item matrix of Table 1a, as completed using the BiasedMF algorithm. With BiasedMF, the prediction function is $predict(u_j, i_k, P, Q) = \mu + b_{u_j} + b_{i_k} + P_{u_j} Q^T_{i_k}$, where μ is the overall average rating, b_{u_j} is the deviation from μ of user u_j ratings, b_{i_k} is the deviation from μ of item i_k ratings, P_{u_j} is the u_j^{th} row of matrix P which are the latent factors for user u_j, and $Q^{\text{th}}_{i_k}$ row of matrix Q which are the latent factors for item i_k. Finally, given the predicted rating matrix R', the preference vector for a user u_j is defined as the predicted ratings for user u_j, $\theta_j = R'_{u_j}$.

Preference clustering. Given a predicted rating matrix R', we can cluster users according to their preference vectors, that are the rows of R'. A distance function and a clustering algorithm C are used. After clustering, we have a set of clusters $C = \{C_s\}$, where each cluster C_s contains a set of users with similar preferences.

Consensus computation. For each cluster C_s, we apply a consensus operator $\mathcal{A}$ to get the consensual preference vector $\hat{\theta}_s$ of cluster C_s. In this paper, the consensus operator is the average, that is $\hat{\theta}_{s,k}$ is the average predicted rating for item k, the average being computed on the subset of users belonging to cluster C_s. We obtain $M = \{M_1 = (C_1, \hat{\theta}_1), \ldots, M_K = (C_K, \hat{\theta}_K)\}$, the set of prediction models where each M_s is composed of a cluster of users C_s and its consensual preference vector $\hat{\theta}_s$. Table 1c shows the result of clustering the predicted rating matrix rows (Table 1b) in 2 clusters and presents their consensual preference vectors $\hat{\theta}_1$ and $\hat{\theta}_2$.

Model Update. In a live recommendation system, due to the computational cost, the predicted rating matrix R' is computed from times to times only. At any time, R' is computed using a rating matrix R_{t_p} available at some earlier time $t_p \leq t$. As new ratings are entered into the system, these new ratings may be compared to those that have been predicted; so we compare the known entries of R_t at the current time t with the corresponding predictions available in R_{t_p}; when the divergence between the observed ratings and the predicted ratings becomes too large, it is time to update the model. To be more specific, let $diff(t) = \sum_{(u,i)\text{known at time } t} |r_{u,i} - r'_{u,i}|$ After each update on R at any time $t > t_p$, it is straightforward to update $diff(t)$ incrementally. Then, we decide on updating M once $diff(t)$ reaches a certain threshold.

[1]The name BiasedMF comes from the LibRec library.

Table 1: (a) Example of a user-item rating matrix R. "-" means that the user has not rated the item. (b) Predicted rating matrix R'. (c) Consensual preference vector.

(a)

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	5	2	4	-	5	1	-
u_2	4	-	5	-	5	-	1
u_3	2	5	3	5	-	-	-
u_4	1	-	2	-	2	-	-
u_5	-	-	3	4	1	-	-

(b)

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	4.6	2.09	4.23	4.24	4.84	1.07	1.0
u_2	4.2	3.8	4.42	5.0	4.86	2.28	1.2
u_3	1.97	4.84	3.22	4.87	2.68	2.68	1.61
u_4	1.19	3.24	2.17	3.56	1.92	1.23	1.0
u_5	1.77	3.16	2.81	4.07	1.14	1.6	1.56

(c)

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	4.6	2.09	4.23	4.24	4.84	1.07	1.0
u_2	4.2	3.8	4.42	5.0	4.86	2.28	1.2
$\hat{\theta}_1$	4.4	2.94	4.32	4.62	4.85	1.67	1.1
u_3	1.97	4.84	3.22	4.87	2.68	2.68	1.61
u_4	1.19	3.24	2.17	3.56	1.92	1.23	1.0
u_5	1.77	3.16	2.81	4.07	1.14	1.6	1.56
$\hat{\theta}_2$	1.64	3.74	2.73	4.16	1.91	1.83	1.39

3.2 Recommendation

From the previous steps, the recommendation system has a predicted rating for each item for the current user to recommend to. Now, it has to decide which items will be recommended.

For that purpose, we propose PdMS as a method to select a prediction model and determine the items to recommend to a user. PdMS applies a multi-armed bandit algorithm to that end.

PdMS: After getting the prediction models, we sort the consensual preference vectors according to their ratings. So, for each $\hat{\theta}_s$ we have a $\hat{\theta}'_s$ that represents the consensual preference vector in a sorted order. The idea is to recommend first the items with the highest ratings in each model. We hypothesize that this strategy can contribute to learn users preference faster. For instance, the correspondent $\hat{\theta}'_1$ to $\hat{\theta}_1$ in Table 1c will have the sorted list of items equal to $\{i_5, i_4, i_1, i_3, i_2, i_6, i_7\}$, while for $\hat{\theta}'_2$ we will have $\{i_4, i_2, i_3, i_5, i_6, i_1, i_7\}$. At each time t a recommendation for a user u is made according to a bandit algorithm $\mathcal{B}$ as follows:

(1) Select a prediction model M_s using the bandit algorithm $\mathcal{B}$;
(2) Select i^* from $\hat{\theta}'_s$ the next top-rated item not yet recommended to u;
(3) Recommend item i^* to user u;
(4) Receive a feedback from u;
(5) Compute the reward;
(6) Update the prediction model statistics of $\mathcal{B}$.

Note that we consider that PdMS approach will be applied to make recommendations to users that are initially new users who remain cold until they have provided a certain amount of preferences/feedback on recommended items. The threshold to determine when a user is not cold anymore can be different for distinct systems. Arguably, a user quickly loses the interest of a recommender system if he is compelled to provide many ratings [15, 22, 38]. Experiments have demonstrated that several state-of-art systems are able to provide recommendations with reasonable quality after getting 15 ratings from the cold user [11, 50]. In this paper we measure the method performance after 5, 10, 15 and 20 recommendations using the *nDCG* metric. As soon as the user is not cold anymore the system can switch to a personalized prediction model.

Computing the Reward . The goal of step 5 in PdMS is to compute the reward using the feedback of the recommendation. It is computed applying the following method:

Feedback-based reward: We define the reward measure in Equation (2) based on the feedback of the recommendation. In this way, when a user gives a higher feedback for a recommended item, we will have a high reward. This method is easily adapted to implicit feedback, such as a clicks, views, purchases, *etc.*

$$X_{M_s}(t) = \frac{r_{u,i}}{r_{max}} \quad (2)$$

Where r_{max} represents the largest rating in the dataset and $r_{u,i}$ is the feedback of user u for the recommended item i according to the selected model M_s. For a implicit feedback we can consider that the value of $r_{u,i}$ is binary. For example, in a music recommender system we will have $r_{u,i} = 1$ if the user listens to a recommended song and $r_{u,i} = 0$ otherwise.

Example: Consider that the system will offer recommendations for two cold users, user u_a and u_b based on the consensual prediction models in 1c. Using PdMS UCB1 approach, it will initially make a random selection of prediction models. Suppose that for u_a the system first recommends the item i_5, according to $\hat{\theta}'_1$ and receives a feedback equal to 4. If $r_{max} = 5$, we will have a reward equal to 0.8 for prediction model $\hat{\theta}'_1$.

In the next recommendation using $\hat{\theta}'_2$ the item i_4 is recommended and receives a feedback equal to 1, then the reward to $\hat{\theta}'_2$ will be 0.2. At this point, for u_a we have a mean reward of 0.8 to $\hat{\theta}'_1$ and 0.2 to $\hat{\theta}'_2$ with one trial to each prediction model. Applying Equation (1), the next u_a recommendation will be i_1 using $\hat{\theta}'_1$ that represents the arm with the highest mean reward at that moment.

For the second user u_b, suppose the system will first select $\hat{\theta}'_2$ and recommend item i_4 receiving a feedback of 5, and then recommend item i_5 receiving a feedback of 2. The mean reward for user u_b is 0.4 to $\hat{\theta}'_1$ and 1 to $\hat{\theta}'_2$. In this case, the item i_2 will be the next recommendation of user u_b using $\hat{\theta}'_2$.

Considering a scenario where users are not logged-in or that we cannot identify the users, the PdMS system will see them like cold users. The first recommendation will be offered with the random selection of prediction models. Then, we rely on the user feedback to make the trade-off between exploitation/exploration. So that, even not logged user can take advantage of tunned prediction models already in the system.

With this running example, it is reasonable to assume that a new user can be framed into already known categories, which might foster the initial experience of the new user.

4 EXPERIMENTS

In this section, we give an overview of the datasets used in the experiments. Then, we describe the evaluation protocol used, and the other methods that we compare to ours.

4.1 Datasets

We evaluate PdMS on 4 real movies datasets. Table 2 summarizes their main features.

Table 2: Dataset features.

Dataset	Users	Items	Ratings	Sparsity (%)
Facebook	498	169	49,729	40.9
FilmTrust	1,508	2,071	35,494	98.86
Movielens	943	1,682	100,000	93.7
Flixster	1,323	1,175	811,726	47.78

Facebook Dataset [19] a survey from Facebook's users requested to rate a list of movies. Ratings range from 1 to 5.

Filmtrust Dataset [26] is about rate and sharing movies. Ratings range from 0.5 (min) to 4 (max).

Movielens Dataset [28] collected by the GroupLens Research Project contains movies ratings from users that rated at least 20 movies in a 1 to 5 range.

Flixster Dataset *[29]* collected from a social movie site, contains movies ratings in a 0.5 to 5 range.

4.2 Evaluation Criteria

As we focus on cold users, we adopt the leave-one-out protocol [52]: at each round, we train on all users but one which is used as a test user. As we do not use any information about the test user, it is a cold user. We call this protocol a **0-rating protocol**.

Ranking quality is measured computing the Normalized Discounted Cumulative Gain (nDCG) metric, see Equation (3). $DCG(u)$ is the discounted cumulative gain of predicted ranking for a target user u, $DCG^*(u)$ is the ground truth and N is the number of users in the result set. $DCG^*(u)$ is defined by equation (4). In that equation, $r_{u,1}$ is the rating (according to the user feedback) of the item first recommended for user u, t is the recommendation time, $r_{u,t}$ is the user feedback for the item recommended in turn t and T is the size of the ranked list.

$$NDCG = \frac{1}{N}\sum_{u}\frac{DCG(u)}{DCG^*(u)} \tag{3}$$

$$DCG(u) = r_{u,1} + \sum_{t=2}^{T}\frac{r_{u,t}}{\log_2 t} \tag{4}$$

4.3 Comparison Methods

To assess the effectiveness of our PdMS model, we compare it to the following baselines:

Global Average: A standard "popular" baseline, which recommends using the global average rate for an item.

Most Popular: Another standard baseline that rank the items based on the number of ratings received and recommend the top-ranked.

Random-MS: The Random-MS is a baseline we proposed that selects at random a prediction model. It receives as input the set of prediction models M and makes a recommendation for a user u at each time t according to the steps:

(1) Randomly select a prediction model M_s;
(2) Randomly select an item i not recommended yet from $\hat{\theta}_s$;
(3) Recommend item i to user u;
(4) Receive a feedback from u.

Parameter Settings. We use LibRec [24], which provides an implementation of Global Average. The implementation of Random-MS and PdMS are built on top of BiasedMF algorithm in LibRec library. We cluster BiasedMF prediction models and in the recommendation process we include the implementation of Random-MS, PdMS UCB1, PdMS UCB1 with tuned constant, PdMS ϵ-Greedy and PdMS ϵ-decreasing Greedy algorithm.

Experiments were executed with 10 latent factors and 100 iterations. We executed Random-MS 5 times and get the average result. With PdMS, we also experimentally test several cluster size. Then we set the optimal number of clusters to 4 clusters for FilmTrust, 3 clusters for Facebook , Movielens and Flixster. Beside this, we apply K-means (using the Euclidean distance measure) as the clustering algorithm.

For PdMS ϵ-Greedy we test ϵ in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. The optimal ϵ value is 0.3 for Facebook dataset and 0.2 for Filmtrust, Movielens and Flixster dataset. PdMS ϵ-Decreasing greedy start with $\epsilon = 0.9$ and we use the function $\frac{0.5\epsilon}{t}$ to decrease ϵ value at each new recommendation.

For PdMS UCB1 with tunned constant we test different values to C where C in $\{0.05, 0.1, 0.2, 0.5, 0.8, 1, ..., 16\}$ and obtained $C = 0.05$ as the optimal value in all datasets.

5 RESULTS AND DISCUSSION

This section reports our results and further discussions. We aim to answer the following questions:

Q1: How effective is PdMS to offer initial recommendations to cold users?

Q2: Are the PdMS results reliable, or random and noisy?

Table 3 presents the *nDCG* at rank size of 5, 10, 15, and 20 per method. Note that PdMS has two variants, UCB1 and ϵ-Greedy.

5.1 Recommendation Effectiveness (Q1)

Comparing the results of PdMS to Global Average, we note the following improvements regarding *nDCG*@5: 13.1% on Facebook, 5.9% on Filmtrust, 12.4% on Movielens and 11.6% on Flixster. The

Table 3: We compute the *nDCG* of three baseline approaches and four PdMS variants across four datasets. The four last rows represents the results of PdMS.

(a) Facebook

	Method	Rank size			
		@5	@10	@15	@20
	Global Average	0.728	0.734	0.740	0.749
	Most Popular	0.813	0.808	0.813	0.814
	Random-MS	0.714	0.719	0.725	0.732
PdMS	ϵ-Greedy	0.858	0.851	0.849	0.849
PdMS	ϵ-decreasing greedy	0.858	0.852	0.851	0.849
PdMS	UCB1	0.858	0.851	0.849	0.848
PdMS	UCB1 tuned constant	0.859	0.854	0.851	0.850

(b) Filmtrust

	Method	Rank size			
		@5	@10	@15	@20
	Global Average	0.800	0.808	0.814	0.821
	Most Popular	0.819	0.824	0.832	0.839
	Random-MS	0.805	0.811	0.815	0.818
PdMS	ϵ-Greedy	0.859	0.858	0.861	0.862
PdMS	ϵ-decreasing greedy	0.859	0.859	0.860	0.861
PdMS	UCB1	0.859	0.857	0.860	0.861
PdMS	UCB1 tuned constant	0.858	0.859	0.860	0.861

(c) Movielens

	Method	Rank size			
		@5	@10	@15	@20
	Global Average	0.735	0.749	0.765	0.780
	Most Popular	0.786	0.795	0.807	0.822
	Random-MS	0.729	0.739	0.755	0.772
PdMS	ϵ-Greedy	0.858	0.859	0.865	0.874
PdMS	ϵ-decreasing greedy	0.859	0.859	0.866	0.874
PdMS	UCB1	0.857	0.858	0.865	0.874
PdMS	UCB1 tuned constant	0.858	0.859	0.866	0.875

(d) Flixster

	Method	Rank size			
		@5	@10	@15	@20
	Global Average	0.708	0.718	0.721	0.724
	Most Popular	0.766	0.767	0.769	0.769
	Random-MS	0.706	0.704	0.705	0.707
PdMS	ϵ-Greedy	0.824	0.821	0.820	0.819
PdMS	ϵ-decreasing greedy	0.823	0.822	0.821	0.821
PdMS	UCB1	0.822	0.821	0.819	0.817
PdMS	UCB1 tuned constant	0.823	0.822	0.821	0.820

comparison against Random-MS is similar with the following improvements: 14.5% on Facebook, 5.4% on Filmtrust, 13% on Movielens and 11.8% on Flixster. Most Popular presents better results than the others two baselines and compared with this method our approach achieves the following *nDCG*@5 improvements: 4.6% on Facebook, 4% on Filmtrust, 7,3% on Movielens and 5.8% on Flixster.

From Table 3, we first observe that PdMS consistently outperforms all baselines in all four datasets. Note that on Filmtrust, PdMS gain is smaller when compared to Global Average. That might be so because of the rating distribution. 68.14% of Filmtrust ratings are greater or equal to 3, see Figure 3b. On the other hand, a smaller gain of PdMS in all datasets when compared with Most Popular, mainly to Facebook and Filmtrust dataset, indicates that in general the most popular items receive the greater ratings.

In particular, the difference between PdMS variants are small, and we now assess their significance.

5.2 Randomness Analysis (Q2)

We examine whether the recommendations generated by PdMS are significantly better than those made by baseline methods and its variants on the different datasets by performing a null hypothesis test. We express *H0* as: Recommendations offered from Global Average, Random-MS, Most Popular and PdMS (ϵ-Greedy, UCB1, ϵ-Greedy Decreasing and UCB1 with tunned constant) are distributed identically on all 4 datasets.

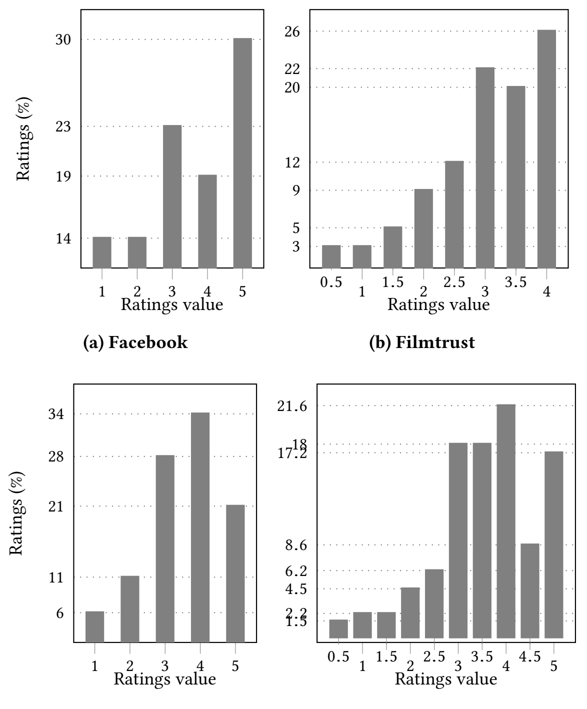
We checked the normality of the results with a Shapiro-Wilk test, and their homogeneity of the results (*nDCG*) using a Bartlett test. The tests reject both assumptions of normality, and homogeneity. Then, to check whether the null hypothesis holds, we run Kruskal-Wallis tests on the *nDCG* results, using the 95% confidence level, (*i.e.*, p-value < 0.05). The Kruskal-Wallis test is a non-parametric test to assess whether samples originate from the same distribution.

Between two baselines, Global Average and Random-MS there is no statistically significant difference. However, Most Popular outperforms both. The difference between PdMS and all baselines are significant, the p-value of Kruskal-Wallis test is less than 2.210^{-16}, therefore we reject the null hypothesis $H0$. In conclusion, PdMS performs significantly better than the 3 other baselines, but the four PdMS variants have similar results.

5.3 Limitations and Future Work

PdMS design decisions suggest a set of limitations, many of which we hope to address in future work.

First, we considered only one type of collaborative filtering algorithm, the Matrix Factorization. It is not clear whether conclusions generalize beyond this setting. Future work could compare it to state of the art systems. However, the approaches that are compared are all based on the same completed matrix, so that they should all suffer in the same way from the result of the matrix factorization. Second, we compare the two exploration model, ϵ-Greedy and UCB1, and their variants ϵ-decreasing greedy and UCB1 with tuned constant. However, we did not experiment optimization of bandit algorithms. Future work could also perform experiments with optimized methods.



(a) Facebook

(b) Filmtrust

(c) Movielens

(d) Flixster

Figure 3: Rating distribution per dataset.

Another threat arises from the evaluation criteria. We rely on *nDCG* score to access our approach, mainly because we were investigating the recommendation quality. Overall it presents high accuracy levels, but we might check other metrics to ensure a fair evaluation [31]. For example, user coverage study would be required to reveal whether our approach can offer recommendation to a large audience; and likewise, catalog coverage [51].

We also intend to work in a mechanism to identify when a user is not cold anymore. The idea is to harness the available models in the systems as much as possible, without sooner resort to typical recommendation models.

6 RELATED WORK

This section reviews prior works on user cold-start problem. Additionally, concerning the goal of the paper, we distinguish between two threads of related work. First, we elaborate on approaches that are coping with model selection for recommendation. Second, we survey approaches focusing on applications of Bandits in recommender systems.

User Cold-Start Recommendation

The history of cold-start problem in academia and industry to recommender systems is long [30]. A number of studies have proposed methods to improve certain aspects of the cold user problem.

One trend of these studies has been towards the application of multiple matrix factorization techniques for cold-start problems. The challenge is to predict the rating of deliberated held out cells in the user-item matrix. Then, a myriad of machine learning techniques have been applied such as clustering [39], classification [27], regression [45], or singular value decomposition [48]. Barjasteh et al. proposed a two-stage algorithm to factorize the rating matrix and then factorizes user similarity matrix and item similarity matrix [4, 5]. However, for massive scale of users, computing the similarity matrix might be prohibitive due to the computational resource power demanded and real-time constraints.

Even achieving high accuracy, the matrix completion task has shown to be insufficient to make the best recommendations. Other works cope with cold-start problems by the design of user interactions, basically, a survey given during the recommender system sign-up process. The idea is to ask new members to rate set of movies, instead of one movie at a time [11, 23].

Another trend of research has been towards designing new prediction models. The typical approach is to use side information to build a prediction model [1], specially using social information. For instance, the work of Ma et al. is matrix factorization based approach that incorporates contextual social information into prediction models [40]. In the same direction, Pereira and Hruschka proposed a hybrid system to recommend for users without ratings, but they rely on demographic information [47]. Social recommenders are often expensive due to the need of model computation whenever new user ratings arrive. Zhao et al. explored the combination of user-item relationship with item content to learning the preferences of user in more scalable way [57]. Wongchokprasitti et al. studied the possibility to use user models built by one system to another to address the cold start problem [56]. Peng et al. present a method to better weight the impact of user's attributes, preferences and item's popularity in multi-level regression model [46]. Guo et al. focused on both user modeling and trust modeling [25]. They proposed an improved recommender model based on trusted neighbors. Meo et al. found that centrality metrics are the most reliable way to spot reputable users in trust networks [43].

Herein, we advocate that the growing amount of prediction models opens an important problem of model selection. Specifically, we investigate how to handle a cold user without prior side information. Moreover, our approach exploits models already built in the system.

Model Selection in Recommender System

Cold user eventually warms up (or leaves the system). From that perspective, model selection methods have been applied to recommender systems as a switching mechanism between stages of a user experience [9].

Our work is inspired by techniques for selecting models, however, we are interested in selecting better models within the cold-start stage. There are some common themes between our approach and the work of Billsus and Pazzani [6]. They proposed a news recommender system that leverages explicit feedback from the user to build and update the user model. We also rely on user's feedback. However we use feedback to analyze which model among a set of consensual ones might be the best one.

Our approach builds on the same understand from Ekstrand and Riedl [14], different prediction models unveil distinct results for the same user. While their focus is on switch hybrids systems, we proposed a solution to switch among consensual prediction models existing in the same system. Specially within cold-start stage, Braunhofer et al. also proposed a switching mechanism, but dependent on contextual information [8].

Bandits in Recommender System

In the following, we review approaches which applied Bandits algorithms in recommender systems.

Li et al. reported on personalized recommendation of news articles as a contextual bandit problem [35]. They propose LinUCB, an extension to the UCB algorithm. It selects the news based on mean and standard deviation. It also has a factor α to control the exploration/exploitation trade-off. Moreover, Caron and Bhagat incorporate social components into bandit algorithms to tackle the cold-start problem [10]. They designed an improved bandit strategy to model the user's preference using multi-armed bandits. Several works model the recommendation problem using a MAB setting in which the items to be recommended are the arms [7, 21, 42]. In a different way, Lacerda et al. model users as arms to recommend daily-deals [33, 34]. They consider strategies for splitting users into exploration and exploitation. Li et al. proposed to double cluster users and items using bandits. We also rely on clustering users, but to reach a consensual model that could be leverage by our model selection strategy [36].

In our prior work [18], we presented a research proposal with the goal of analyzing existing prediction models taken from recommender systems to better understand their nature. Our study reported here, relies on and extends our previous study design. The goal of PdMS is the selection of existing prediction models that might offer better recommendations for cold users. Our MAB setting is also different, since the arms are consensual prediction models. Besides that, our approach requires no prior effort from the user.

7 CONCLUSIONS

In this paper, we showed how a careful model selection can provide better recommendations to full cold-start user. Furthermore, our approach, PdMS, performed reasonably well even with no prior side information, but exploring feedback to frame new users into known categories. It achieves 85% accuracy levels of *nDCG*@5. To sum-up, our contributions are:

- A formalization of the model selection as a multi-armed bandit problem (Sections 2 and 3).
- PdMS, which is an effective approach to recommend for users without prior side information (Section 3).
- An empirical evaluation of PdMS against four real, public datasets (Section 4 and 5).

Looking forward, PdMS envisions recommender systems in which substantial amount of prediction models is available for analysis, making possible a new wave of intelligent recommender systems.

Acknowledgments

We would like to thank the Brazilian research agencies CAPES, CNPq and FAPEMIG. K.V.R. Paixão is sponsored by a scholarship from CAPES. Ph. Preux' research is funded by Contrat de Plan État Région Data, French Ministry of Higher Education and Research, and CNRS; he also thanks the support of Inria, and SequeL.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-Aware Recommender Systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, Boston, MA, 217–253.

[2] D. H. Alahmadi and X. J. Zeng. 2015. Twitter-Based Recommender System to Address Cold-Start: A Genetic Algorithm Based Trust Modelling and Probabilistic Sentiment Analysis. In *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI'15).* 1045–1052.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2 (2002), 235–256.

[4] Iman Barjasteh, Rana Forsati, Farzan Masrour, Abdol-Hossein Esfahanian, and Hayder Radha. 2015. Cold-Start Item and User Recommendation with Decoupled Completion and Transduction. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15).* ACM, New York, NY, USA, 91–98.

[5] Iman Barjasteh, Rana Forsati, Dennis Ross, Abdol-Hossein Esfahanian, and Hayder Radha. 2016. Cold-Start Recommendation with Provable Guarantees: A Decoupled Approach. *IEEE Trans. on Knowl. and Data Eng.* 28, 6 (June 2016), 1462–1474.

[6] Daniel Billsus and Michael J. Pazzani. 2000. User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction* 10, 2-3 (Feb. 2000), 147–180.

[7] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Gançarski. 2012. A Contextual-Bandit Algorithm for Mobile Context-Aware Recommender System. In *Proceedings of the 19th International Conference Neural Information Processing, Doha, Qatar, November 12-15, Part III (ICONIP '12).* Springer Berlin Heidelberg, Berlin, Heidelberg, 324–331.

[8] Matthias Braunhofer, Victor Codina, and Francesco Ricci. 2014. Switching Hybrid for Cold-starting Context-aware Recommender Systems. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14).* ACM, New York, NY, USA, 349–352.

[9] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* 12, 4 (Nov. 2002), 331–370.

[10] Stéphane Caron and Smriti Bhagat. 2013. Mixing Bandits: A Recipe for Improved Cold-start Recommendations in a Social Network. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis (SNAKDD '13).* ACM, New York, NY, USA, Article 11, 9 pages.

[11] Shuo Chang, F. Maxwell Harper, and Loren Terveen. 2015. Using Groups of Items for Preference Elicitation in Recommender Systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15).* ACM, New York, NY, USA, 1258–1269.

[12] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. 2012. Fused Matrix Factorization with Geographical and Social Influence in Location-based Social Networks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI'12).* AAAI Press, 17–23.

[13] Julien Delporte, Alexandros Karatzoglou, Tomasz Matuszczyk, and Stéphane Canu. 2013. Socially Enabled Preference Learning from Implicit Feedback Data. In *Machine Learning and Knowledge Discovery in Databases: European Conference, Prague, Czech Republic, Proceedings, Part II*, Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 145–160.

[14] Michael Ekstrand and John Riedl. 2012. When Recommenders Fail: Predicting Recommender Failure for Algorithm Selection and Combination. In *Proceedings of the Sixth ACM Conference on Recommender Systems (RecSys '12).* ACM, New York, NY, USA, 233–236.

[15] Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2014. Active Learning Strategies for Rating Elicitation in Collaborative Filtering: A System-wide Perspective. *ACM Trans. Intell. Syst. Technol.* 5, 1, Article 13 (Jan. 2014), 33 pages.

[16] Crícia Z. Felício, Claudianne M. M. de Almeida, Guilherme Alves, Fabíola S. F. Pereira, Klérisson V. R. Paixão, and Sandra de Amo. 2016. Visual Perception Similarities to Improve the Quality of User Cold Start Recommendations. In *Advances in Artificial Intelligence: 29th Canadian Conference on Artificial Intelligence, Proceedings*, Richard Khoury and Christopher Drummond (Eds.). Springer International Publishing, Cham, 96–101.

[17] Crícia Z. Felício, Claudianne M. M. de Almeida, Guilherme Alves, Fabíola S. F. Pereira, Klérisson V. R. Paixão, Sandra de Amo, and Celia A. Z. Barcelos. 2016. VP-Rec: A Hybrid Image Recommender Using Visual Perception Network. In *IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI'16).* 70–77.

[18] Crícia Z. Felício, Klérisson V. R. Paixão, Celia A. Z. Barcelos, and Philippe Preux. 2016. Multi-Armed Bandits to Recommend for Cold-Start User. In *Proceedings of the 4rd Symposium on Knowledge Discovery, Mining and Learning.* 182–185.

[19] Crícia Zilda Felício, Klérisson Vinícius Ribeiro Paixão, Guilherme Alves, Sandra de Amo, and Philippe Preux. 2016. Exploiting social information in pairwise preference recommender system. *Journal of Information and Data Management (JIDM)* 7, 2 (2016), 99–115.

[20] Crícia Zilda Felício, Klérisson Vinícius Ribeiro Paixão, Celia Aparecida Zorzo Barcelos, and Philippe Preux. 2016. Preference-Like Score to Cope with Cold-Start User in Recommender Systems. In *IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI'16)*. 62–69.

[21] Sertan Girgin, Jérémie Mary, Philippe Preux, and Olivier Nicol. 2012. Managing advertising campaigns — an approximate planning approach. *Frontiers of Computer Science* 6, 2 (2012), 209–229.

[22] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. 2010. On Bootstrapping Recommender Systems. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. ACM, New York, NY, USA, 1805–1808.

[23] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.* 6, 4, Article 13 (Dec. 2015), 19 pages.

[24] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. LibRec: A Java Library for Recommender Systems. In *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization (UMAP 2015), Dublin, Ireland, June 29 - July 3, 2015.*

[25] Guibing Guo, Jie Zhang, and Daniel Thalmann. 2012. A Simple but Effective Method to Incorporate Trusted Neighbors in Recommender Systems. In *Proceedings of the 20th International Conference on User Modeling, Adaptation, and Personalization (UMAP'12)*. Springer-Verlag, Berlin, Heidelberg, 114–125.

[26] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2013. A Novel Bayesian Similarity Measure for Recommender Systems. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13)*. AAAI Press, 2619–2625.

[27] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2016. A Novel Evidence-Based Bayesian Similarity Measure for Recommender Systems. *ACM Trans. Web* 10, 2, Article 8 (May 2016), 30 pages.

[28] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages.

[29] Mohsen Jamali and Martin Ester. 2010. A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 135–142.

[30] Dietmar Jannach, Paul Resnick, Alexander Tuzhilin, and Markus Zanker. 2016. Recommender Systems — Beyond Matrix Completion. *Commun. ACM* 59, 11 (Oct. 2016), 94–102.

[31] Daniel Kluver and Joseph A. Konstan. 2014. Evaluating Recommender Behavior for New Users. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 121–128.

[32] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. ACM, New York, NY, USA, 426–434.

[33] Anisio Lacerda, Rodrygo L. Santos, Adriano Veloso, and Nivio Ziviani. 2015. Improving Daily Deals Recommendation Using Explore-then-exploit Strategies. *Inf. Retr.* 18, 2 (April 2015), 95–122.

[34] Anisio Lacerda, Adriano Veloso, and Nivio Ziviani. 2013. Exploratory and Interactive Daily Deals Recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 439–442.

[35] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 661–670.

[36] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative Filtering Bandits. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 539–548.

[37] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized News Recommendation Based on Click Behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*. ACM, New York, NY, USA, 31–40.

[38] Benedikt Loepp, Tim Hussein, and Jüergen Ziegler. 2014. Choice-based Preference Elicitation for Collaborative Filtering Recommender Systems. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3085–3094.

[39] Meilian Lu, Zhen Qin, Yiming Cao, Zhichao Liu, and Mengxing Wang. 2014. Scalable News Recommendation Using Multi-dimensional Similarity and Jaccard-Kmeans Clustering. *J. Syst. Softw.* 95 (Sept. 2014), 242–251.

[40] Hao Ma, Tom Chao Zhou, Michael R. Lyu, and Irwin King. 2011. Improving Recommender Systems by Incorporating Social Contextual Information. *ACM Trans. Inf. Syst.* 29, 2, Article 9 (April 2011), 23 pages.

[41] Augusto Q. Macedo, Leandro B. Marinho, and Rodrygo L.T. Santos. 2015. Context-Aware Event Recommendation in Event-based Social Networks. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, New York, NY, USA, 123–130.

[42] Jérémie Mary, Romaric Gaudel, and Philippe Preux. 2015. Bandits and Recommender Systems. In *Revised Selected Papers of the First International Workshop on Machine Learning, Optimization, and Big Data*. Vol. 9432. Springer-Verlag, Inc., New York, NY, USA, 325–336.

[43] Pasquale De Meo, Katarzyna Musial-Gabrys, Domenico Rosaci, Giuseppe M. L. Sarnè, and Lora Aroyo. 2017. Using Centrality Measures to Predict Helpfulness-Based Reputation in Trust Networks. *ACM Trans. Internet Technol.* 17, 1, Article 8 (Feb. 2017), 20 pages.

[44] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. 2010. You Are Who You Know: Inferring User Profiles in Online Social Networks. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM '10)*. ACM, New York, NY, USA, 251–260.

[45] Seung-Taek Park and Wei Chu. 2009. Pairwise Preference Regression for Cold-start Recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys '09)*. ACM, New York, NY, USA, 21–28.

[46] Furong Peng, Xuan Lu, Chao Ma, Yuhua Qian, Jianfeng Lu, and Jingyu Yang. 2017. Multi-level preference regression for cold-start recommendations. *International Journal of Machine Learning and Cybernetics* (2017), 1–14.

[47] Andre Luiz Vizine Pereira and Eduardo Raul Hruschka. 2015. Simultaneous co-clustering and learning to address the cold start problem in recommender systems. *Knowledge-Based Systems* 82 (2015), 11 – 19.

[48] Huseyin Polat and Wenliang Du. 2005. SVD-based Collaborative Filtering with Privacy. In *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC '05)*. ACM, New York, NY, USA, 791–795.

[49] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach. *SIGKDD Explor. Newsl.* 10, 2 (Dec. 2008), 90–100.

[50] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach. *SIGKDD Explor. Newsl.* 10, 2 (Dec. 2008), 90–100.

[51] Alan Said and Alejandro Bellogín. 2014. Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 129–136.

[52] Claude Sammut and Geoffrey I. Webb (Eds.). 2010. *Leave-One-Out Cross-Validation*. Springer US, Boston, MA, 600–601.

[53] L. Song, C. Tekin, and M. van der Schaar. 2016. Online Learning in Large-Scale Contextual Recommender Systems. *IEEE Transactions on Services Computing* 9, 3 (May 2016), 433–445.

[54] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

[55] Liang Tang, Yexi Jiang, Lei Li, and Tao Li. 2014. Ensemble Contextual Bandits for Personalized Recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 73–80.

[56] Chirayu Wongchokprasitti, Jaakko Peltonen, Tuukka Ruotsalo, Payel Bandyopadhyay, Giulio Jacucci, and Peter Brusilovsky. 2015. User Model in a Box: Cross-System User Model Transfer for Resolving Cold Start Problems. In *Proceedings of the 23rd International Conference on User Modeling, Adaptation and Personalization (UMAP'15)*. Springer International Publishing, Cham, 289–301.

[57] Z. Zhao, H. Lu, D. Cai, X. He, and Y. Zhuang. 2016. User Preference Learning for Online Social Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (Sept 2016), 2522–2534.