ExcUseMe: Asking Users to Help in Item Cold-Start Recommendations

Michal Aharon
Yahoo Labs, Haifa, Israel
michala@yahoo-inc.com

Oren Anava
Technion, Haifa, Israel
oanava@tx.technion.ac.il

Noa Avigdor-Elgrabli
Yahoo Labs, Haifa, Israel
noaa@yahoo-inc.com

Dana Drachsler-Cohen
Technion, Haifa, Israel
ddana@cs.technion.ac.il

Shahar Golan
Yahoo Labs, Haifa, Israel
shaharg@yahoo-inc.com

Oren Somekh
Yahoo Labs, Haifa, Israel
orens@yahoo-inc.com

ABSTRACT

The item cold-start problem is of a great importance in collaborative filtering (CF) recommendation systems. It arises when new items are added to the inventory and the system cannot model them properly since it relies solely on historical users' interactions (e.g., ratings). Much work has been devoted to mitigate this problem mostly by employing hybrid approaches that combine content-based recommendation techniques or by devoting a portion of the user traffic for exploration to gather interactions from random users.

We focus on pure CF recommender systems (i.e., without content or context information) in a realistic online setting, where random exploration is inefficient and smart exploration that carefully selects users is crucial due to the huge flux of new items with short lifespan. We further assume that users arrive randomly one after the other and that the system has to immediately decide whether the arriving user will participate in the exploration of the new items.

For this setting we present *ExcUseMe*, a smart exploration algorithm that selects a predefined number of users for exploring new items. *ExcUseMe* gradually excavates the users that are more likely to be interested in the new items and models the new items based on the users' interactions. We evaluated *ExcUseMe* on several datasets and scenarios and compared it to state-of-the-art algorithms. Experimental results indicate that *ExcUseMe* is an efficient algorithm that outperforms all other algorithms in all tested scenarios.

1 Introduction

Recommendation systems aim to present users with the most relevant items (e.g., movies, songs, advertisements, etc.) by predicting the user interest. Typically, they base their predictions on predefined features and user activity. User activity refers to numerical ratings or binary interactions users provide that reflect their interest in certain items. Techniques that rely on features are known as *content-based* [23] while techniques that rely solely on user activity are known as *collaborative filtering* (CF). CF is widely-used in recommenders due to its high accuracy, good scalability, and ability to execute without content analysis for feature extraction.

Though CF is employed in many industrial recommenders, there is still active research on the question of how to cope with new users or items [2, 3, 4, 15, 22]. This challenge, known as *the cold-start problem*, arises since the system does not have relevant interactions for the new entity (user or item) and thus cannot model it properly. The user cold-start problem, in which a new user joins the system, is commonly addressed by interviewing the new user and asking her to rate several key items [9, 14, 24]. Unfortunately, the item cold-start problem is trickier because items cannot be interviewed and typically there are no users willing to rate every new item.

To model new items, CF recommenders select users for exploration and record their interactions with the new items. However, recommender systems typically have only a handful of slots to present items, for both exploration and recommendation, and since their goal is to present recommendations, the exploration process must be: (i) efficient, namely require a few impressions for new items, and (ii) accurate, namely obtain good results under some quality measure, for example under the *root mean squared error* (RMSE). Not only are CF recommenders expected to meet these two goals but they also face the challenge that users arrive in an online fashion, namely systems do not know which users will arrive and when. This enforces the exploration process to decide whether to present to users new items immediately upon their arrival.

In this paper, we present *ExcUseMe*, an algorithm for selecting users in online explorations of CF recommenders that rely on binary interactions. *ExcUseMe* aims to select users that are likely be interested in the new item and are thus expected to provide feedback. To detect such users, *ExcUseMe* attempts to **exc**avate the interested **use**rs' **me**an characteristics. It begins with selecting users with distinct tastes until obtaining the first feedback. Once a user provides feedback, *ExcUseMe* selects users that are similar to this user and are thus more likely to provide feedback on the new item.

We compare *ExcUseMe* to state-of-the-art algorithms on several datasets and scenarios. Our scenarios simulate real-world settings by varying the number of users available for exploration and the number of participants. Experimental results indicate that *ExcUseMe* obtains the best RMSE in all tested scenarios namely it is an efficient and accurate algorithm. In addition, results show that *ExcUseMe* converges towards selecting users that are interested in the new items, thus *ExcUseMe* also provides better user experience during exploration compared to the other algorithms.

Main Contributions The contributions of this paper are:

- Definition of an online exploration framework for selecting users for the item cold-start problem, inspired by real-world settings (Sec. 3). To the best of our knowledge, we are the first to consider such realistic framework for this problem.
- An efficient and accurate algorithm, named *ExcUseMe*, for selecting users in online explorations (Sec. 4).

RecSys'15, September 16–20, 2015, Vienna, Austria.

DOI: http://dx.doi.org/10.1145/2792838.2800183.

- Evaluation of the exploration framework and the *ExcUseMe* algorithm. Experiments show that *ExcUseMe* is an efficient and accurate algorithm that outperforms state-of-the-art algorithms in several datasets and scenarios (Sec. 5).

The rest of the paper is organized as follows. Sec. 2 provides background and related work, Sec. 3 presents the problem definition and the exploration framework, Sec. 4 describes the *ExcUseMe* algorithm, Sec. 5 presents evaluation results, and Sec. 6 concludes.

2 Background and Related Work

In this section, we provide background and survey related work. We first describe the collaborative filtering recommendation technique (Sec. 2.1), then its inherent cold-start problem (Sec. 2.2), and finally we discuss related aspects of online algorithms (Sec. 2.3).

2.1 Collaborative Filtering

Collaborative filtering (CF) is one of the most widely-used approaches for constructing recommender systems, commonly implemented via *latent factor models* (LFM) or *neighborhood methods*. The difference between these techniques is that LFM models users and items in the same (latent factor) space and predicts whether a user is interested in an item based on their relationship in that space, whereas neighborhood methods predict the interest based on the relationship of the user to other users or the relationship of the item to other items [15]. For example, the item-item approach [19, 26] predicts the tendency of a user to an item by inspecting her ratings of items that tend to be rated similarly by other users.

Latent Factor Models In this paper, we follow the LFM approach. In this approach, items and users are associated with *latent factor vectors* and *biases* [17]. The vectors capture the users' and items' characteristics through latent factors, and the biases capture either the likelihood that the user will be interested in an arbitrary item (the *user bias*) or the likelihood that an arbitrary user will be interested in the item (the *item bias*). The latent factor vectors and biases are inferred from the observed interactions, which are assumed to be drawn from patterns reflecting the users' tastes.

Matrix Factorization To infer the vectors and biases we employ a common realization of the LFM which is based on low-rank matrix factorization [17]. This approach is appealing due to its high predictive accuracy and good scalability and it provides a substantial expressive power that allows modeling specific data characteristics such as temporal effects [16], item taxonomy [6], and attributes [1].

2.2 The Cold-Start Problem

An inherent requirement of CF is to have historical user-item interactions. Thus, when a new entity (user or item) appears and there is no relevant historical interactions, the CF recommender cannot model the new entity reliably. This is known as the *user cold-start problem* if the new entity is user, or the *item cold-start problem*, otherwise. While users and items are represented similarly in the latent vector space (usually), these two problems are essentially different. The main difference is that new users can be interviewed by the recommender to bootstrap their modeling. Another difference is that in most settings the number of users is much larger than the number of items, hence a typical item usually gets more ratings than an individual user provides, which may affect the modeling. We next review past work that has addressed these challenges.

The User Cold-Start Problem Modeling new users' preferences is typically obtained by short interviews during which the users rate several items from carefully constructed *seed sets*. Seed sets may be constructed based on *popularity*, *contention*, and *coverage* [10, 14, 24, 25]. In [24] the idea of constructing seed sets adaptively based on user responses (in adaptive interviews) was recognized as beneficial. Since then it was employed in many works, commonly using decision trees [9, 18, 25].

The Item Cold-Start Problem A common approach to mitigate the item cold-start problem is by providing additional attributes of the new items to the recommender systems [1, 11, 12, 21]. This approach is known as the *hybrid* approach since it combines CF with content based methods. The authors of [11, 12] propose a hybrid approach based on *Boltzmann machines*. The authors of [1] propose a regression-based latent factor model in which the items' and users' latent vectors are obtained from low-rank matrix decomposition of a matrix whose products are weight matrices and attribute matrices. The authors of [21] improve this work by solving a convex optimization problem to estimate the weight matrices.

Other works have addressed a different setting in which there are few ratings to the new items but there is no item content or context information. They showed that new items' latent factor vectors could be estimated by a linear combination of the raters' latent factor vectors and their ratings (without retraining the model) [2, 3, 15, 22]. A common approach to obtain these ratings for new items to bootstrap their modeling, applied by large-scale commercial recommenders, devotes a portion of the user traffic for *random exploration* of the new items. A recent work [4] addresses this problem of obtaining few ratings for new items in an offline setting in which all users are available for selection and their modeling is known. The authors first showed how to estimate the new item's latent vector and bias from ratings and the raters' latent vectors and biases to obtain the optimal RMSE. They then formalized the problem of selecting the raters as an optimization problem where the goal was to minimize the RMSE obtained by the estimated new item's vector and bias. This problem was solved via an *optimal design approach* and the solution was shown to be an approximation of the optimal solution. While this work considers a different setting than ours since it was applied in an offline setting and considered only positive interactions (ratings) while ignoring missing interactions, to the best of our knowledge, it is the closest work to ours.

2.3 Online Algorithms

In this paper, we consider the setting of online explorations and thus we apply *online computations*. In online computation settings, the input sequence is unknown and must be processed upon its arrival. A well-studied problem in this field is the *secretary problem* in which an unknown series of secretaries arrive one by one where each is assigned with a value revealed upon her arrival, and the goal is to pick the secretary with the highest value. However, the algorithm must decide upon the arrival of a new secretary whether to hire her (and end the process) or decline her (which is an irrevocable decision) and wait for a future, possibly better, secretary. Due to this limitation, the goal in such algorithms is to obtain a *good* secretary, e.g., one whose value is guaranteed not to be too far from the best secretary's value.

The secretary problem has been extensively studied as well as its variants. A relevant variant to our setting is the k *secretary problem* [5, 8, 13] in which the k best secretaries need to be hired. Our work can be phrased as the k-secretary problem: the series of secretaries are the series of users arriving and available to present them the new item; the value of k is the predefined number of users that may be selected; and the value of the users (the secretaries) is determined by the *ExcUseMe* algorithm. To select the k users (secretaries) we follow the work of [5] that showed that it is beneficial to split the set of arriving users to k portions and select one user from each portion; further details are provided in Sec. 3.

3 Definitions and Exploration Framework

In this work, our goal is to compute latent factor vectors and biases for new items by carefully selecting a few users to present them with the new items and then computing the vectors and biases from their interactions (feedback or lack of feedback). We begin this section with a formal definition of this problem. Then, we explain how to optimally compute such a vector from users' interactions. Thereby, the problem at hand boils down to selecting users to provide interactions for the new item (which is the problem *ExcUseMe* addresses). Finally, we describe the online exploration framework which resembles realistic online recommender settings.

The Item Cold-Start Problem We assume we are given a recommender system whose data is:

- N users: $\mathbf{U} = \{u_1, u_2, \ldots, u_N\}$,
- M "old" items: $\mathbf{Q} = \{q_1, q_2, \ldots, q_M\}$, and
- $N \cdot M$ interactions: $I : \mathbf{U} \times \mathbf{Q} \rightarrow \{0, 1\}$, where $I(u, q)$ equals 1 if u provided feedback to q, or 0 otherwise.

We further assume that the system has learned from these interactions a CF-model that captures characteristics of users and items via real-valued column vectors of dimension d (where d is typically small) and real-valued biases. Based on this model, the system estimates the interest of a user u in an item q as follows:

$$\widetilde{I}(u, q) = V_u^T \cdot V_q + b_u + b_q + \mu \quad (1)$$

where V_u and V_q are the latent vectors, b_u and b_q are the biases, and μ is the average feedback rate: $\mu = \frac{1}{|\mathbf{U} \times \mathbf{Q}|} \cdot \sum_{u \in \mathbf{U}, q \in \mathbf{Q}} I(u, q)$.

In this setting we define the item cold-start problem. Given:

- a new item i,
- a candidate set U_A ($\subseteq \mathbf{U}$) revealed in an online fashion, and
- a budget of k users for exploring i,

select k users from U_A to provide interactions with i and accordingly compute a latent factor vector and bias for i obtaining the lowest root mean squared error (RMSE), where:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathbf{U}|} \cdot \sum_{u \in \mathbf{U}} (\widetilde{I}(u, i) - I(u, i))^2} \quad (2)$$

Computing Vectors and Biases from Interactions To compute the new item's latent factor vector V_i and bias b_i from a set of interactions we assume we are given k users $\mathbf{U_i} = \{u_{j_1}, ..., u_{j_k}\}$, their vectors, biases, and interactions with the new item: $I_i : \mathbf{U_i} \rightarrow \{0, 1\}$. We then follow the approach in [4] and define V_i and b_i to be those minimizing the mean squared error, namely:

$$(V_i, b_i) = \underset{V_i', b_i'}{\text{argmin}} \sum_{u \in \mathbf{U_i}} ((V_u^T \cdot V_i' + b_u + b_i' + \mu) - I_i(u))^2 \quad (3)$$

This equation can be solved analytically, yielding the solution:

$$(V_i, b_i) = (\sum_{u \in \mathbf{U_i}} V_u' \cdot V_u'^T)^{-1} (\sum_{u \in \mathbf{U_i}} (I_i(u) - b_u - \mu) \cdot V_u') \quad (4)$$

where V_u' is the concatenated column vector $(1, V_u)$. As noted in [4], the left term in Eq. (4) might be non-invertible, however, in practice a regularization term $\lambda \cdot (||V_i'||^2 + b_i'^2)$ is added to Eq. (3) resulting in a different and invertible term: $(\lambda \cdot \mathcal{I} + \sum_{u \in \mathbf{U_i}} V_u' \cdot V_u'^T)^{-1}$.

The Online Exploration Framework To model realistic settings we assume that the candidate set U_A is revealed in an online fashion, i.e., users arrive one by one. Upon the arrival of a user, an immediate decision is required on whether she is chosen for exploring the new item. If a user u is selected, a binary interaction $I_i(u)$ is revealed, indicating if u is interested in the new item i.

Algorithm 1: The Item Cold-Start Exploration Framework

Input: A new item i, a candidate set U_A, a budget k, and a score function F_s.
Output: The set of selected users and their interactions D_i.

```
D_i = ∅
phase= 0.5 · |U_A|/k
initialize the auxiliary data structure A
for i = 0; i < k; i++ do
    s_m=−∞
    for j = 0; j< phase; j++ do // The learning phase
        Upon the arrival of a new user u from U_A:
        s_m = max(s_m,F_s(u,A))
    for j = 0; j< phase; j++ do // The selection phase
        Upon the arrival of a new user u from U_A:
        if F_s(u,A)≥s_m or j == phase − 1 then
            I_i(u) = getInteraction(u, i)
            D_i = D_i ∪ {(u, I_i(u))}
            update the auxiliary data structure A
            break
return D_i
```

To select users in such online setting, we follow the approach presented in [5]. In this approach, k users are selected in k intervals where each interval is divided into a *learning phase* and a *selection phase*. During the learning phase, users are not selected but instead evaluated via a score function, F_s, and the best score is recorded. After the learning phase is completed, the selection phase begins and the first user whose score is not lower than the recorded score is selected. If no such user arrives, the last user of the selection phase is selected. Once a user is selected, the next interval begins. The size of each interval is at most U_A/k thus enabling equal intervals.

Algorithm 1 summarizes this approach. It receives the new item i, the candidate set U_A that is revealed in a stream, the budget k, and the score function F_s. The algorithm's goal is to select k users from U_A and return the set D_i of selected users and their interactions with i. In case F_s requires an auxiliary data structure A, A is initialized at the beginning and updated after each user selection.

4 The ExcUseMe Approach

In this section, we present the *ExcUseMe* algorithm that evaluates users for exploring new items. It is defined by its score function and auxiliary data structure required by the exploration framework.

The Score Function F_s ***and the*** **UseMe** ***Vector*** *ExcUseMe* aims to discover (*excavate*) the mean latent vector of the users that are interested in the new item. To this end, it maintains a vector that converges to this average latent vector, which we call the *UseMe* vector, and it is the auxiliary data structure described in the previous section. The *UseMe* vector, denoted by V_{UseMe}, is initialized with zeros and after users provide interactions it is updated to the vector V_i computed in Eq. (4) (Sec. 3) based on all revealed interactions.

The score function F_s returns the sum of the user bias and the dot product of the user's latent factor vector and V_{UseMe}, namely: $F_s(u, V_{UseMe}) = V_u^T \cdot V_{UseMe} + b_u$. The dot product captures the similarity of u to the *UseMe* vector and thus the likelihood that u will provide feedback to i, while the user's bias captures the likelihood that u will provide feedback to an arbitrary item, as the bias is positively correlated to the number of old items for which u provided feedback. In the first iteration, F_s boils down to the user bias, namely it guides to select users who tend to provide feedback.

The Rationale Behind **ExcUseMe** *ExcUseMe* is guided by two principles: (i) to excavate the new item's vector it is required to reveal as much feedback as possible (because user feedback is sparse), and (ii) the user whose score is maximal is assumed to be

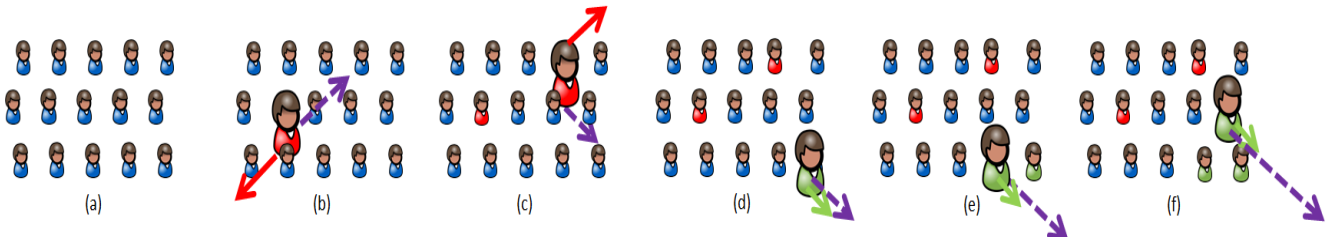

Figure 1: A schematic illustration of ExcUseMe converging toward users who show interest in the new item (see Section 4 for details).

the most interested in the new item. Driven by these two guidelines, *ExcUseMe* selects the users that are believed to be more likely to provide feedback and thus will help to excavate the *UseMe* vector. In addition, *ExcUseMe* leverages the co-existence of the users' and items' latent vectors in the same latent space by computing V_{UseMe} as if it were the new item's vector while treating it as capturing the mean latent vector of the interested users.Hence, since V_{UseMe} and the estimated latent vector of the new item's are computed in the same manner (see Eq. 4), not only does *ExcUseMe* converge towards selecting the interested users but also it is a semi-greedy algorithm to estimate the new item's vector.

Example To illustrate why *ExcUseMe* converges towards selecting the interested users, consider Figure 1. To simplify presentation, assume an offline setting. In 1(a), a set of candidates is revealed and in 1(b) the first selected user is the one with the maximal positive interactions. Assume this user does not provide feedback to the new item. The *UseMe* vector (in dashed arrow) is updated to capture this negative interaction, and the next user with the highest score is in the opposite direction of that user's vector. Then, in 1(c), the second selected user does not provide feedback either and the *UseMe* vector is updated to be orthogonal to the two users' vectors. In 1(d), the third selected user is the first to provide feedback and thus the *UseMe* vector roughly remains in the same direction as this user's vector. The following selected users are assumed to have vectors roughly in the same direction as well. In 1(e) and 1(f), the two selected users are also assumed to provide feedback, which strengthens the direction of the *UseMe* vector towards the interested users, thus converging towards the mean vector of the interested users.

Complexity We next show that *ExcUseMe* complexity is $O(1)$. A single score computation of F_s in *ExcUseMe* is $O(d)$ where d is the dimension of the latent factor vectors. The *UseMe* vector is updated incrementally: after selecting a user, the terms $V'_u \cdot V_u'^T$ and $(I_i(u) - b_u - \mu) \cdot V'_u$ are computed and added to the two terms in Eq. (4) (Sec. 3), then the left factor inverse is computed, and finally the dot product is computed. Computing these two terms is $O(d)$, computing the inverse is $O(d^2)$ since this term is a symmetrical matrix, and the final dot product is also $O(d^2)$. Since d is constant (and typically small), *ExcUseMe* overall complexity is $O(1)$.

5 Evaluation

In this section, we evaluate the effectiveness of *ExcUseMe* using several datasets. For each dataset we define 200 randomly selected items as "new items". The remaining items are used to train the initial model. After obtaining a mature initial latent factor model, we evaluate the performance of the different exploration algorithms: we execute them under the exploration framework for all 200 new items (independently), where users selected by the algorithms return interactions based on the datasets (these interactions are not used during the initial model training), then the new items' latent vectors and biases are computed as described in Eq. (4), and finally they are evaluated based on three metrics. Experiments were implemented in Java and ran on a Sony Vaio with Intel(R) Core(TM) i7-3612QM processor and 8GB RAM.

We next provide more details on the evaluation and present experimental results. We begin with describing the datasets (Sec. 5.1), then explain how the initial model is constructed (Sec. 5.2), continue with providing the evaluation metrics (Sec. 5.3), then list the baseline algorithms (Sec. 5.4), and finally describe in more detail the experiments and present their results (Sec. 5.5).

5.1 Datasets

Here, we describe the datasets and the data pre-processing steps.

Datasets The datasets we use are: (i) *MovieLens1M* from `grouplens.org`, (ii) *Netflix* from `netflixprize.com`, and (iii) *Yahoo! Music* from `webscope.sandbox.yahoo.com`. *MovieLens1M* and *Netflix* contain movie ratings in a 1–5 scale, and *Yahoo! Music* contains song ratings in a 0–100 scale.

Data Filtering To avoid entities which are less indicative as they have too few ratings, we consider only users with at least 10 ratings and items with at least 20 ratings. Also, to avoid users which are noisy and unreliable we removed those with more than 300 ratings.

Data Statistics After filtering the data, *MovieLens1M* contained $488,616$ rates from $5,085$ users to $2,388$ items; *Netflix* contained $638,343$ rates from $6,657$ users to $6,685$ items; and *Yahoo! Music* contained $441,954$ rates from $9,458$ users to $2,640$ items.

Obtaining Binary Interactions Since all datasets contain numerical ratings while we require binary interactions, we interpreted the ratings as follows. We say user u provided feedback on item i if the dataset contains a numerical rating u provided to i. Otherwise, we say u did not provide feedback. This definition considers also low rates as feedback. We believe this interpretation is valid since even a low rate indicates that the user chose to interact with that item, in contrast to items she ignored and did not provide feedback at all.

5.2 Constructing the Initial Model

We next describe how we constructed the initial latent factor model that captures the characteristics of the users and the "old" items.

The Model To model the users and old items, we apply regularized matrix factorization [27] where the goal is to minimize the mean squared error. Namely, given the interactions $I : \mathbf{U} \times \mathbf{Q} \to \{0, 1\}$, the estimated interest $\widetilde{I} : \mathbf{U} \times \mathbf{Q} \to \{0, 1\}$ (see Eq. (1), Sec. 3), and a regularization parameter λ, the goal is to minimize:

$$\sum_{\substack{u \in \mathbf{U} \\ q \in \mathbf{Q}}} \left(\widetilde{I}(u,q) - I(u,q)\right)^2 + \lambda\left[\sum_{u \in \mathbf{U}} (||V_u||^2 + b_i^2) + \sum_{q \in \mathbf{Q}} (||V_q||^2 + b_q^2)\right] \tag{5}$$

where the variables are the average feedback rate, μ, the latent factor vectors, and the biases.

Model Training To optimize the cost function (Eq. (5)) we set the latent factor dimension d to 10 and λ to 0.1 (these parameters were optimized using a grid search over a validation set), and applied stochastic gradient descent (SGD) [17]. The SGD step sizes were determined according to the AdaGrad approach [7] that adapts the steps based on the aggregated gradient of each factor in the vectors. SGD was executed for 1000 iterations, after which further iterations would negligibly improve the cost function.

RMSE Model for Binary Interactions Typically, in settings considering binary interactions models are optimized based on the *logistic regression* approach [20] that minimizes the log-loss function instead of RMSE. We implemented both approaches to construct the model and observed they achieved similar results for our purposes. Since RMSE allows to analytically compute the new item's latent vector and bias (see Sec. 3), we chose it over the log-loss.

5.3 Evaluation Metrics

We next present the evaluation metrics.

RMSE Measures the root mean squared error between the estimated and actual scores. RMSE was described in Eq. (2) (Sec. 3) and considered all users in the system. However, it is common to consider only a test set of users and we follow this approach in the experiments. Namely, given a test set of users, $\mathbf{U}'(\subseteq \mathbf{U})$, the new item i's interactions $I_i : \mathbf{U}' \rightarrow \{0,1\}$ and the estimated scores $\widetilde{I}_i : \mathbf{U}' \rightarrow \{0,1\}$, RMSE equals: $\sqrt{\frac{1}{|\mathbf{U}'|} \cdot \sum_{u \in \mathbf{U}'} (\widetilde{I}_i(u) - I_i(u))^2}$.

Probability of Receiving Positive Interactions To estimate accurately a new item's latent factor vector from binary interactions it is crucial to receive positive interactions. Positive interactions are strong indications to the new items' characteristics and thus having only negative interactions is likely to result in inaccurate estimated latent vectors. Thus, we measure the probability of revealing positive interactions, namely the fraction of new items for which the algorithm revealed at least one positive interaction.

Number of Positive Interactions While revealing one positive interaction is crucial for estimating the new item's vector, having multiple positive interactions is typically more desirable. Having only few users that provided positive interactions may lead to estimating a vector biased towards these users and thus the estimated vector would not capture accurately the new item's characteristics. Moreover, an algorithm that selects on average more interested users provides better *user experience* because in real recommender systems users are unaware that they participate in exploration and expect to receive only recommendations. Thus, we measure the average positive interactions the algorithm reveals for new items.

5.4 Baseline Algorithms

We next describe the algorithms compared to *ExcUseMe* by describing their score functions required by the exploration framework (none of them require auxiliary data structures). We consider two algorithms commonly used by recommenders (*Random* and *Frequent Users*), a state-of-the-art approach in an offline setting (*Anava et al.*), and a complementary approach to ours (*Distance*).

Random Users are randomly picked with a probability of $k/|U_A|$ where k is the number of users that participate in the exploration and U_A is the set of available users.

Frequent Users In this approach, users that provided more positive interactions to "old" items (items that were already modeled) are believed to be more likely to provide positive interactions to new items. Thus, the score is the number of items to which the user provided positive interactions, namely: $F_s(u) = \sum_{q \in \mathbf{Q}} I(u,q)$.

Anava et al. This approach, proposed by [4], is an approximation algorithm for an offline setting in which *all* users in the recommender system may be considered and when selecting the next user *all* users may be examined. This algorithm was adapted to our online setting by defining F_s to be the function used in [4] to select the next user: $F_s(u) = 1/\text{Trace}\left(\left(\mathcal{P}_{B\setminus u}\mathcal{P}_{B\setminus u}^{\top}\right)^{-1}\right)$, where B is the set of users not selected yet and $\mathcal{P}_{B\setminus u}$ is a matrix whose columns are the latent factor vectors of the users in $B \setminus \{u\}$. This approach was shown to outperform other well-known offline approaches, and thus we consider this approach as the representative of offline approaches.

Distance This approach picks the first user randomly and afterwards selects the user with the farthest vector from the selected users' vectors, namely: $F_s(u) = min\{||V_u - V_{u'}||^2 \mid u' \in S\}$, where S is the set of selected users and the distance is the L^2 norm. This approach is in some sense complementary to *ExcUseMe* because it explores the user space by locating the user which is the most different from the previously selected users. This is in contrast to *ExcUseMe* that converges towards selecting users which are very similar to each other.

5.5 Experimental Setting and Results

Here, we describe the experiments that evaluate all algorithms.

Experimental Setting In each experiment, we fix a new item i and a budget k and let the algorithms select k candidates to learn i's latent vector and bias. As described in Sec. 3, users are revealed gradually and after a user is revealed the algorithms have to decide immediately whether to ask her for feedback on i.

Simulating Real-World Scenarios In real-world scenarios, not only do recommendation systems have to decide immediately upon the user arrival whether to ask for feedback, but they are also exposed to only $n\%$ users. This is because they wish to model the new item's quickly and thus cannot wait for all users to arrive.

To simulate this, in each experiment we randomly selected $n\%$ users, shuffled them, and let the algorithms select users from this set of users. The values of n were 10, 25, 50, and 100. Since we select the available users randomly, to increase the statistical significance of the results, we repeated each experiment 15 times.

RMSE Evaluation To evaluate the estimated items' latent vectors and biases, we measured RMSE (Sec. 5.3). To this end, before each experiment, 10% of the users were put aside as a test set to compute RMSE and were not selected to the $n\%$ available users.

Figure 2 shows the RMSE obtained on the different datasets and different values of n. The graphs show the RMSE as a function of the budget k where each point is the average RMSE of the 200 new items' latent vectors and biases, where the RMSE of a single new item is the average RMSE over the 15 repeating experiments.

Figure 2 shows that in all tested scenarios *ExcUseMe* outperforms all other approaches. The improvement of *ExcUseMe* over other approaches is most significant for low budgets and when there are more available users to consider for exploration (large n). Thus, we conclude that (i) *ExcUseMe* allows a fast exploration that expedites the convergence towards the new item's latent factor vector, and that (ii) *ExcUseMe* provides a score function that is adaptable to the number of available users (n) and as this number increases, the RMSE may only decrease. This is in contrast to other approaches which do not demonstrate this property. To illustrate it, consider the graphs showing the RMSE of the *Netflix* dataset, when the budget is k=20, and the percentage of available users is n=50 and n=100: only the *ExcUseMe* approach obtains a better RMSE when there are 100% available users. These two conclusions imply that recommender systems may obtain a short exploration with

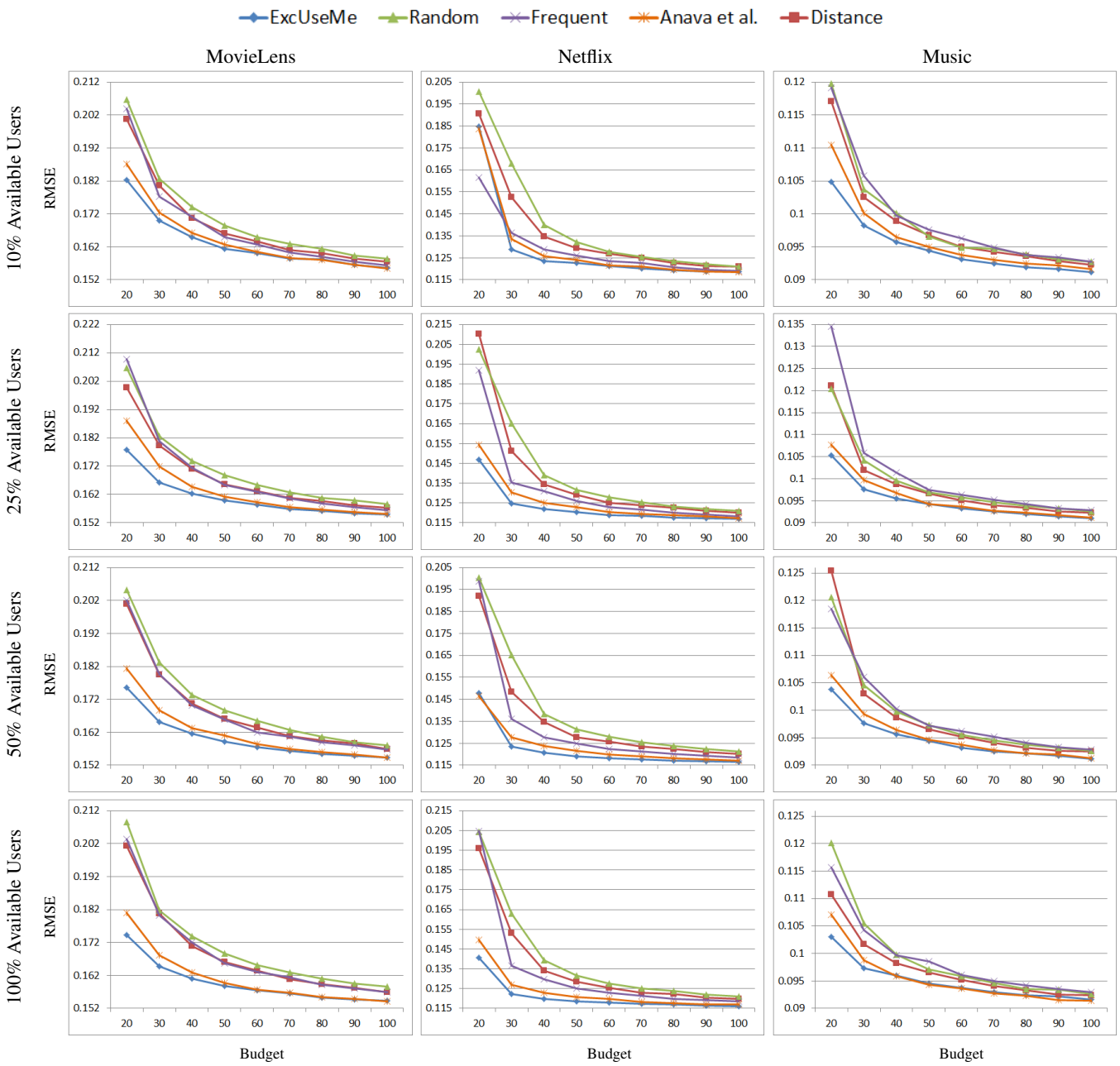

Figure 2: The RMSE value on all datasets for different percentages of available users.

ExcUseMe, both in terms of the number of available users, thus not having to wait until enough candidates arrive, and in terms of the budget size. In addition, *ExcUseMe* adapts well to the size of available users which may be beneficial for recommenders in which at certain times (e.g., evenings) users connect more frequently and then more users are available (i.e., n increases).

Probability of Receiving Positive Interactions We next study how well the different algorithms succeed in selecting users that provide positive interactions. As discussed in Sec. 5.3, having at least one positive interaction for new items is crucial to learn their characteristics accurately. Figure 3 shows the results for the *MovieLens1M* dataset for different percentages of available users. The results for the other datasets were similar and are thus omitted. The graphs show the average probability of receiving at least one positive interaction for the new items averaged over 15 repeating experiments.

The results indicate that *ExcUseMe*, *Frequent*, and *Anava et al.* behave similarly which explains why they obtain better RMSE than *Random* and *Distance*. In addition, as expected, the results show that as the budget increases the probability of receiving positive interactions increases consistently for all approaches. Results also indicate that as the number of available users (n) increases, the probability to receive positive interactions increases for all approaches except *Random*. We believe this is the result of longer learning and selection phases in the exploration framework that help to select better users, a property relevant to all approaches but *Random*.

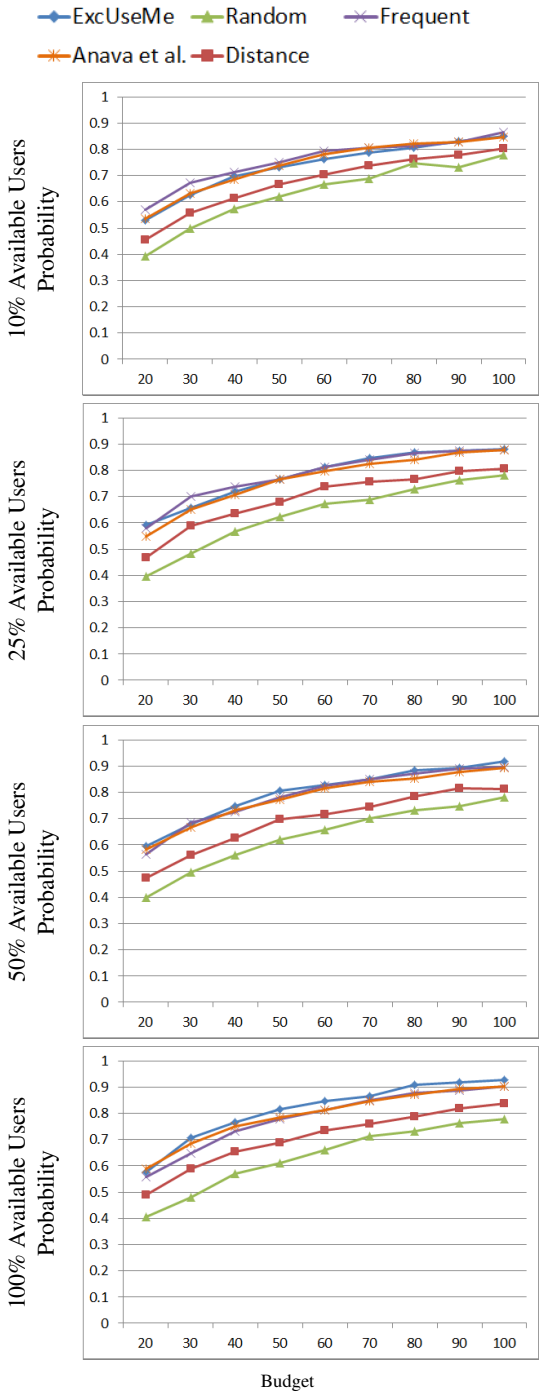

Figure 3: Probability of receiving positive interactions.

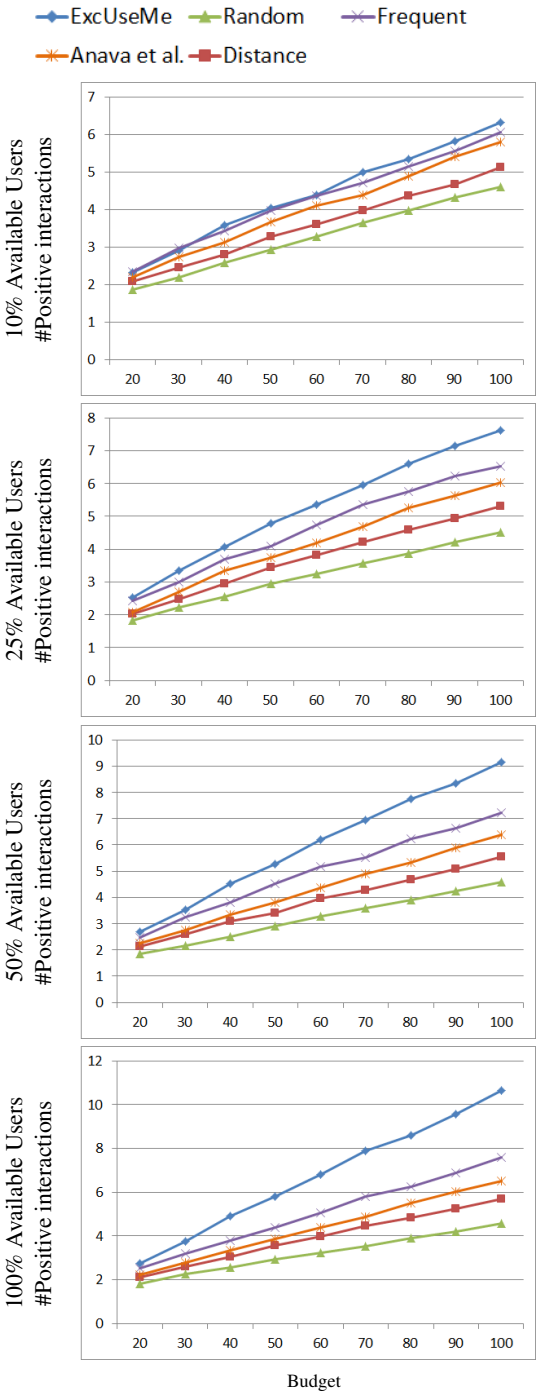

Figure 4: Number of positive interactions for items.

Average Number of Positive Interactions Lastly, we study the average number of positive interactions the approaches reveal for the new items. As discussed in Sec. 5.3, revealing many positive interactions is important to accurately model the new items and it also provides better user experience. Figure 4 shows the average number of positive interactions revealed for the new items on the *MovieLens1M* dataset for different percentages of available users (experiments on the other datasets obtained similar results).

Results indicate that *ExcUseMe* consistently and significantly reveals more positive interactions compared to all other approaches under all budgets for all percentages of available users. Further, as the budget increases and/or when there are more candidates available, the number of positive interactions that *ExcUseMe* reveals increases more significantly than the other approaches. Namely, *ExcUseMe* provides better user experience than the other approaches.

In addition, we believe that the fact that *ExcUseMe* reveals more positive interactions than the other approaches is the reason for obtaining better RMSE. Specifically, this explains why *ExcUseMe* outperforms *Frequent* and *Anava et al.*, which demonstrated a similar behaviour to *ExcUseMe* in revealing at least one positive interaction for new items.

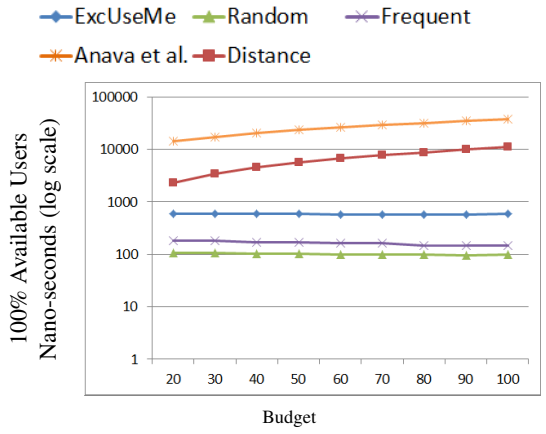

Figure 5: Computational Time.

Computational Complexity To evaluate the computational complexity of the *ExcUseMe* score function, we measured the time a single computation takes. The computational time may be affected by the budget (for example, the *Anava et al.* and *Distance* approaches base their scores on previously selected users), and thus we measured time as a function of the budget. However, the computational time is unaffected by the dataset or the number of available users and thus we only show the results on the *MovieLens1M* dataset where all users are available for selection (i.e., $n = 100$). Results are presented in Figure 5 that shows the average time in a *log scale* when time was measured in nano-seconds. Results indicate that while *ExcUseMe* requires longer computations compared to *Random* or *Frequent* (as expected), it is only by a factor of less than 4 compared to *Frequent* and less than 5 compared to *Random*. In contrast, *Distance* and *Anava et al.* require expensive computations and their performance becomes worse as budget increases.

6 Conclusions

We presented *ExcUseMe*, a novel and simple algorithm for selecting users to cope with the item cold-start problem, an inherent problem in collaborative filtering recommender systems. *ExcUseMe* assumes an initial model capturing the characteristics of the users and accordingly selects the users that are most likely to be interested in the new items. To evaluate *ExcUseMe*, we applied it in an online setting that captures two realistic aspects: a limited number of available users and a limited number of users that may participate in the exploration. We compared *ExcUseMe* against state-of-the-art algorithms and experimental results show that *ExcUseMe* is efficient and obtains the best RMSE in all tested scenarios. In addition, *ExcUseMe* converges towards selecting users that are likely to be interested in the new items, thus provides better user experience while smartly exploring the new items.

7 References

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD '09*.

[2] M. Aharon, A. Kagian, Y. Koren, and R. Lempel. Dynamic personalized recommendation of comment-eliciting stories. In *RecSys '12*.

[3] N. Aizenberg, Y. Koren, and O. Somekh. Build your own music recommender by modeling internet radio streams. In *WWW '12*.

[4] O. Anava, S. Golan, N. Golbandi, Z. Karnin, R. Lempel, O. Rokhlenko, and O. Somekh. Budget-constrained item cold-start handling in collaborative filtering recommenders via optimal design. In *WWW '15*.

[5] M. Bateni, M. Hajiaghayi, and M. Zadimoghaddam. Submodular secretary problem and extensions. *ACM Trans. Algorithms*, 9(4), 2013.

[6] G. Dror, N. Koenigstein, and Y. Koren. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item. In *RecSys '11*.

[7] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2011.

[8] M. Feldman, J. Naor, and R. Schwartz. Improved competitive ratios for submodular secretary problems (extended abstract). In *APPROX-RANDOM '11*.

[9] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *WSDM '11*.

[10] N. Golbandi, Y. Koren, and R. Lempel. On bootstrapping recommender systems. In *CIKM '10*.

[11] A. Gunawardana and C. Meek. Tied boltzmann machines for cold start recommendations. In *RecSys '08*.

[12] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *RecSys '09*.

[13] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE '10*.

[14] A. Kohrs and B. Mérialdo. Improving collaborative filtering for new-users by smart object selection. In *ICME '01*.

[15] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08*.

[16] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. of the ACM*, 53(4), 2010.

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

[18] S.-L. Lee. Commodity recommendations of retail business based on decision tree induction. *Expert Systems with Applications*, 37(5), 2010.

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

[20] P. McCullagh and J. A. Nelder. Generalized linear models (Second edition). 1989.

[21] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys '09*.

[22] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD '07*.

[23] M. J. Pazzani and D. Billsus. Content-based recommendation systems. *The Adaptive Web*, 4321, 2007.

[24] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *IUI '02*.

[25] A. M. Rashid, G. Karypis, and J. Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *SIGKDD Explor. Newsl.*, 10(2), 2008.

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW '01*.

[27] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *ICDMW '08*.