UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Preference Elicitation as an Optimization Problem

Sepliarskaia, A.; Kiseleva, J.; Radlinski, F.; de Rijke, M.

DOI
10.1145/3240323.3240352

Publication date
2018

Document Version
Final published version

Published in
12th ACM Conference on Recommender Systems

Link to publication

Citation for published version (APA):
Sepliarskaia, A., Kiseleva, J., Radlinski, F., & de Rijke, M. (2018). Preference Elicitation as an Optimization Problem. In *12th ACM Conference on Recommender Systems: October 2-7, 2018 : RECSYS : Vancouver, BC : 2018* (pp. 172-180). Association for Computing Machinery. https://doi.org/10.1145/3240323.3240352

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:07 Oct 2026

Preference Elicitation as an Optimization Problem

Anna Sepliarskaia
University of Amsterdam
Amsterdam, The Netherlands
a.sepliarskaia@uva.nl

Julia Kiseleva
University of Amsterdam
Amsterdam, The Netherlands
y.kiseleva@uva.nl

Filip Radlinski
Google
London, UK
filiprad@google.com

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
derijke@uva.nl

ABSTRACT

The new user coldstart problem arises when a recommender system does not yet have any information about a user. A common solution to it is to generate a profile by asking the user to rate a number of items. Which items are selected determines the quality of the recommendations made, and thus has been studied extensively. We propose a new elicitation method to generate a *static preference questionnaire* (SPQ) that poses relative preference questions to the user. Using a latent factor model, we show that SPQ improves personalized recommendations by choosing a minimal and diverse set of questions. We are the first to rigorously prove which optimization task should be solved to select each question in static questionnaires. Our theoretical results are confirmed by extensive experimentation. We test the performance of SPQ on two real-world datasets, under two experimental conditions: *simulated*, when users behave according to a *latent factor model* (LFM), and *real*, in which only real user judgments are revealed as the system asks questions. We show that SPQ reduces the necessary length of a questionnaire by up to a factor of three compared to state-of-the-art preference elicitation methods. Moreover, solving the right optimization task, SPQ also performs better than baselines with dynamically generated questions.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Personalization*; Collaborative filtering;

KEYWORDS

Cold start problem; Mixed initiative search and recommendation; Preference elicitation

ACM Reference Format:
Anna Sepliarskaia, Julia Kiseleva, Filip Radlinski, and Maarten de Rijke. 2018. Preference Elicitation as an Optimization Problem. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240323.3240352

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
RecSys '18, October 2–7, 2018, Vancouver, BC, Canada
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5901-6/18/10.
https://doi.org/10.1145/3240323.3240352

1 INTRODUCTION

Millions of online e-commerce websites are built around personalized recommender systems (e.g., [5]). Together, they offer broad and varied items, ranging from books and accommodation to movies and dates. The main goal of recommender systems is to predict unknown ratings or preferences based on historical user interactions and information about items, such as past user ratings and item descriptions. Collaborative filtering (CF) is one of the most effective techniques to build personalized recommender systems. They collect user item ratings and derive preference patterns. The main advantage of approaches based on CF is that they do not require domain knowledge and can easily be adapted to different recommender settings.

However, CF-based approaches only work well for users with substantial information about their preferences. When this information is not available, as for new users, the recommender system runs into the *new user cold-start problem*: it cannot produce reliable personalized recommendations until the "cold" user is "warmed-up" with enough information [11, 16].

Most well-known and effective approaches to the new user cold-start problem are questionnaire-based [16, 26]. These elicit information from new users via a questionnaire where users provide absolute ratings for some items [9, 22, 24]. However, obtaining such explicit ratings suffers from a calibration issue [17]: the same rating, of say three stars, may mean completely different things for different people. Moreover, users may change their opinion about an item after having seen other items. For instance, it has been shown that a user is likely to give a lower rating to an item if the preceding one deserved a very high rating [1, 20]. On top of that, recommender systems typically provide ranked lists of items, but for producing a good ranking pairwise preferences between items are preferable to learning absolute relevance scores [4].

Finally, we note that recent research has shifted towards predicting relative preferences rather than absolute ones [18, 31]. For such relative methods, it is natural to ask users to complete a questionnaire by answering relative questions rather than by providing absolute ratings. Therefore, *preference elicitation* questionnaires have been proposed [6, 26]. A *preference elicitation method* (PEM) thus asks questions about pairwise preferences between items.

An example preference elicitation question is shown in Figure 1 below, where a user is asked to indicate which of two movies she prefers. However, it is difficult to create these questionnaires, because it is unclear what criteria should be used to select items for questions. Moreover, a greedy approach, which is common in elicitation procedures, is computationally expensive in PEMs, because the cardinality of the preference question space is proportional to the square of the number of items.

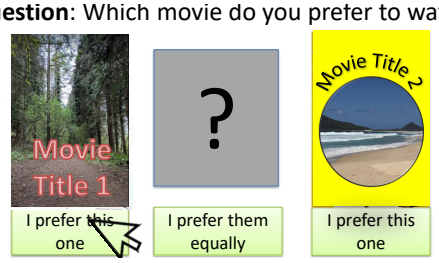

Figure 1: An example of a relative question in a questionnaire aimed at eliciting preferences.

In this work, we address some of these challenges with the following **main research question**:

> *How to optimally generate a preference questionnaire, consisting of relative questions, for new users that will help to solve the new user cold-start problem?*

To answer this question, we develop a new preference elicitation method, called *static preference questionnaire* (SPQ), that can be used in several domains, such as book or movie recommendation. In particular, inspired by [2], we formulate the elicitation procedure in SPQ as an optimization problem. The advantage of static *preference* questionnaires, like the approaches of [6, 26], is that they work with relative rather than absolute questions. The main advantages of our SPQ over previous work is the absence of any constraint to the questions except that they should be informative, as was done in [6], and the fact that it optimizes the expectation of the loss function of the underlying latent factor model (LFM).

Thus, the proposed method SPQ is a preference elicitation method that solves the following optimization problem: Select items for relative questions to predict with maximum accuracy how the user will respond to other relative questions.

Our contributions are three-fold: (1) We formulate the task of generating preference questionnaires, consisting of relative questions for new users, as an optimization problem (Section 2). (2) We show how to solve this optimization problem in a theoretically optimal way (Section 3). (3) We verify that the theoretical results are supported by experimental results on two datasets: By using SPQ, the length of the questionnaire can be reduced independently of the experimental conditions (Section 5).

2 PROBLEM FORMULATION

2.1 Assumptions

Suppose we have a system that contains a history of user feedback for items in a rating matrix $R \in \mathbb{R}^{n \times m}$, where n is the number of users and m is the number of items.[1] The value of entry r_{ui} describes the feedback of user u on item i. The feedback can be (a) explicit, e.g., a user rating (usually from 1 to 5) or a binary value, or (b) implicit feedback like purchases or clicks. Usually, R is sparse [27], since most users rate only a small portion of the items. The task for a recommender system is to predict missing ratings in R and recommend to a user the most attractive item.

An important aspect of our work, as well as previous work [2, 6], is the use of a *latent factor model* (LFM) [27] to solve this task. We assume that LFM is already trained and that each rating r_{ui} complies with the following noisy model:

$$r_{ui} = \mu + b_i + b_u + v_i^T v_u + \epsilon_{ui}, \quad (1)$$

where (1) μ is a global bias, (2) v_i, b_i are the latent factor vector and bias of item i, (3) v_u, b_u are the latent factor vector and bias of user u, and (4) ϵ_{ui} is small random noise. The dimensions in the latent vectors represent item's characteristics. If the user likes particular item's attributes, the corresponding dimensions in her latent factor vector are large. Large values in the item's latent vector correspond to the item's most valuable characteristics. Thus, if an item satisfies a user's taste, the value of $v_i^T v_u$ in Eq. 1 is large. The *user bias* is the rating that a user predominantly provides. The intuition behind the item bias is its popularity. The LFM predicts that most users will like popular items more, despite their diverse preferences. The parameters listed above are assumed to be given to us; we refer to them further as *ground truth*.

The **main problem** that we address is to solve the new user cold-start problem, having ground truth parameters of all items and of all users who have provided some feedback already.

Table 1: Notation used in the paper.

Symbol	Gloss
$\mathbb{I}$	Set of all items
$\mathbb{P}$	Set of all pairs of items: $\mathbb{I} \times \mathbb{I}$
$\mathbb{B}$	Seed set of questions
$\mathbb{F}$	Subset of questions from $\mathbb{B}$
i, j	Items
u	User
q	Question
$\mathbb{V}_{\mathbb{S}}$	Latent presentation of questions from set $\mathbb{S}$
v_i	Latent factor vector of item i
$v_{(i,j)}$	Latent presentation of the question
v_u^*	True latent factor vector of user u
$\hat{v}_u$	Predicted latent factor vector of user u
R	Matrix of ratings provided by users
$\hat{r}_{uij}$	Predicted preference of user u of item i over item j
r^*_{uij}	True preference of user u of item i over item j

2.2 Problem definition

To address the new user cold-start problem, we propose a *preference elicitation method* (PEM) that produces a questionnaire with relative questions; an example such question is in Fig. 1. We call our method *static preference questionnaire* (SPQ). The main goal of SPQ is to minimize the number of questions N that is required to derive a ranked list of personalized recommendations. SPQ minimizes N by generating a list of questions that maximize the gain that a recommender system observes after adding answers for q.

The LFM uses SPQ as follows: (1) using the completed questionnaire, LFM gets $\hat{v}_u$ while predicting the ground truth parameters v_u^*; then (2) LFM uses $\hat{v}_u$ to predict users' relative preferences about all pairs of the items. Thus, the performance of SPQ is measured in terms of the binary classification quality of item pairs. For a given user u and a pair of items (i, j) the classifier estimates a user u's preference of item i over item j. The quality of the classifier is defined as the deviation of the predicted preference value $\hat{r}_{uij}$ from the true preference value r^*_{uij}. We assume that r^*_{uij} can take the values -1, 1, and 0: -1 and 1 reflect that the user prefers one item over another, while 0 corresponds to the situation when she treats them equally (again, for an example, see Fig. 1). Similar to Anava

[1] The notation we use is presented in Table 1.

et al. [2], we define the preference elicitation task as an optimization problem. Given a budget N, SPQ chooses a seed set $\mathbb{B} \subseteq \mathbb{P}$, consisting of N pairs of items, that minimizes the expected loss of the corresponding binary classifier:

$$\arg\min_{\mathbb{B}} \mathbb{E}\left(\sum_{(i,j)\in\mathbb{P}} (\hat{r}_{uij} - r^*_{uij})^2\right). \tag{2}$$

The intuition behind this optimization problem is to force a model to provide a high-quality list of recommendations, asking a predefined number of questions. To provide a high-quality ranked list, a model should correctly classify pairwise preferences between items [4]. That it is exactly what Eq. 2 expresses. In Section 5, we will prove how minimizing Eq. 2 corresponds to improving the quality of the final list of recommendations.

To summarize, we have introduced assumptions for our model and formulated the task of finding a seed set of relative questions as an optimization problem. Next, we will describe an iterative greedy procedure to obtain a near-optimal solution for this problem.

3 METHOD

Eq. 2 is difficult to optimize directly, so we will reformulate it in Section 3.2. To this end, we need two preliminary procedures: (1) one that predicts users' answers to preference questions, and (2) one for modeling a user's latent vector representation after receiving their completed questionnaire. We provide these next.

3.1 Preliminaries

3.1.1 Predicting users' answers. We aim to estimate a user u's preference for item i over item j, given her predicted latent representation $\hat{v}_u$, using the equation:

$$\hat{r}_{uij} = \hat{v}_u \cdot v_{(i,j)} + b_{(i,j)}, \tag{3}$$

where $v_{(i,j)}$ is the latent representation of the question, $v_i - v_j$, and $b_{(i,j)} = b_i - b_j$ is a bias of the question. We rewrite this in matrix form. The predicted answers to a set of preference questions $\mathbb{S}$ are:

$$\hat{r}_{u\mathbb{S}} = \hat{v}_u \cdot \mathbb{V}_{\mathbb{S}} + b_{\mathbb{S}}, \tag{4}$$

where the columns of matrix $\mathbb{V}_{\mathbb{S}}$ are the latent vector representations of the questions from $\mathbb{S}$ and $b_{\mathbb{S}}$ is a column that consists of the biases of the questions.

3.1.2 Modeling users' latent vector presentations. To model the user parameters after receiving a completed questionnaire, which consists of a seed set $\mathbb{B}$, we solve the following problem:

$$\arg\min_{v_u} \sum_{(i,j)\in\mathbb{B}} (v_u \cdot v_{(i,j)} + b_{(i,j)} - r^*_{uij})^2, \tag{5}$$

where r^*_{uij} is an answer given by the user while completing the questionnaire. This problem can be solved using linear regression.

3.2 Static Preference Questionnaire (SPQ)

Anava et al. [2] show that if the procedure for predicting a user's answers and modeling their latent presentation is exactly like we describe above in Eq. 4 and Eq. 5, then the optimization problem Eq. 2 can be formulated as a simpler yet equivalent task:

$$\arg\min_{\mathbb{B}} \mathrm{tr}[(\mathbb{V}_{\mathbb{B}}\mathbb{V}_{\mathbb{B}}^T + \epsilon \cdot \mathbb{E})^{-1}]. \tag{6}$$

Here, $\mathrm{tr}(M)$ denotes the trace of matrix M and $\mathbb{E}$ is the identity matrix. We propose a greedy iterative algorithm that chooses the next question by minimizing Eq. 6 in each iteration. More precisely, given the subset $\mathbb{F}$ of $\mathbb{B}$ consisting of questions that have already been chosen for our questionnaire, the procedure selects the next question q that minimizes the following:

$$\mathrm{tr}[(\mathbb{V}_{\mathbb{F}\cup\{q\}}\mathbb{V}_{\mathbb{F}\cup\{q\}}^T + \epsilon * \mathbb{E})^{-1}]. \tag{7}$$

The proposed optimization problem can be reduced to a simpler one, which is stated and proven in Theorem 1 below.

Before presenting Theorem 1, we need to introduce some additional notation used in Theorem 1 and our algorithm. Define $\mathbb{A} = (\mathbb{V}_{\mathbb{F}}\mathbb{V}_{\mathbb{F}}{}^T + \epsilon * \mathbb{E})^{-1}$. Also, $e_1, e_2, \ldots, e_n$ are orthonormal eigenvectors of $\mathbb{A}$ with eigenvalues: $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_i > \epsilon^{-1}$. We assume that the number of questions in the questionnaire is less than the dimension of the latent space; as a consequence, the subspace of eigenvectors with eigenvalues ϵ^{-1} is non-empty; we denote it as E_ϵ.

Theorem 1. *Let $\epsilon > 0$ be arbitrarily small. The optimization problem (7) is equivalent to finding a question q with latent representation $v_q = \sum a_i e_i + e_\epsilon$ that minimizes*

$$\frac{\lambda_1 a_1^2 + \cdots + \lambda_n a_n^2 + 1}{\|e_\epsilon\|^2}, \tag{8}$$

where $e_\epsilon \in E_\epsilon$ and a_i is a coordinate of v in the basis $\langle e_1, \cdots, e_n, e_\epsilon\rangle$.

Proof. To begin, note that $\mathbb{V}_{\mathbb{F}\cup\{q\}}\mathbb{V}_{\mathbb{F}\cup\{q\}}^T + \epsilon * \mathbb{E} =$

$$\mathbb{V}_{\mathbb{F}}\mathbb{V}_{\mathbb{F}}^T + v_q * v_q^T + \epsilon * \mathbb{E} = \mathbb{A}^{-1} + v_q * v_q^T. \tag{9}$$

Since v_q is a column vector, using the Sherman-Morrison formula [14] we obtain:

$$\mathrm{tr}(\mathbb{V}_{\mathbb{F}\cup\{q\}}\mathbb{V}_{\mathbb{F}\cup\{q\}}^T + \epsilon * \mathbb{E})^{-1} = \mathrm{tr}\left(\mathbb{A} - \frac{\mathbb{A}v_q v_q^T \mathbb{A}}{1 + \mathrm{tr}(\mathbb{A}v_q v_q^T)}\right)$$

$$= \mathrm{tr}(\mathbb{A}) - \frac{\mathrm{tr}(\mathbb{A}v_q v_q^T \mathbb{A})}{1 + \mathrm{tr}(\mathbb{A}v_q v_q^T)} = \mathrm{tr}\,\mathbb{A} - \frac{\sum_{i=1}^n \lambda_i^2 a_i^2 + \epsilon^{-2}\|e_\epsilon\|^2}{1 + \sum_{i=1}^n \lambda_i a_i^2 + \epsilon^{-1}\|e_\epsilon\|^2}. \tag{10}$$

Thus minimizing (7) is equivalent to maximizing the following:

$$\max\left(\frac{\sum_{i=1}^n \lambda_i^2 a_i^2 + \epsilon^{-2}\|e_\epsilon\|^2}{1 + \sum_{i=1}^n \lambda_i a_i^2 + \epsilon^{-1}\|e_\epsilon\|^2}\right)$$

$$\sim \max\left(\epsilon^{-1} + \frac{\sum_{i=1}^n \lambda_i(\lambda_i - \epsilon^{-1})a_i^2 - \epsilon^{-1}}{1 + \sum_{i=1}^n \lambda_i a_i^2 + \epsilon^{-1}\|e_\epsilon\|^2}\right)$$

$$\sim \max\left(\epsilon^{-1} + \frac{\sum_{i=1}^n \lambda_i(\lambda_i \cdot \epsilon - 1)a_i^2 - 1}{\epsilon(1 + \sum_{i=1}^n \lambda_i a_i^2) + \|e_\epsilon\|^2}\right)$$

$$\sim \max\left(\frac{\epsilon \cdot (\sum_{i=1}^n \lambda_i^2 a_i^2) - (\sum_{i=1}^n \lambda_i a_i^2 + 1)}{\epsilon(1 + \sum_{i=1}^n \lambda_i a_i^2) + \|e_\epsilon\|^2}\right)$$

$$\sim^{\epsilon\to 0} \max\left(\frac{-(\sum_{i=1}^n \lambda_i a_i^2 + 1)}{\|e_\epsilon\|^2}\right) \sim \min\left(\frac{(\sum_{i=1}^n \lambda_i a_i^2 + 1)}{\|e_\epsilon\|^2}\right). \qquad \square$$

To understand the intuition behind optimizing Eq. 8, we need to understand the connection between the eigenvectors of $\mathbb{A}$ and the questions in $\mathbb{F}$, which we address next.

Lemma 3.1. *If the questions in $\mathbb{F}$ are represented by linearly independent vectors $v_{(i_1,j_1)}, \ldots, v_{(i_m,j_m)}$, then $e_1, e_2, \ldots, e_n$ are vectors from a subspace $U = \langle v_{(i_1,j_1)}, \ldots v_{(i_m,j_m)}\rangle$. Also, the number of eigenvalues that are different from ϵ is n. That is, $n = m$.*

Proof. Eigenvectors are the same for $\mathbb{A}$ and $\mathbb{A}^{-1}$, so we prove this lemma for $\mathbb{A}^{-1}$. To prove the first part of the lemma it is enough to show that U and $U^\perp$ are invariant subspaces for $\mathbb{A}^{-1}$ and $U^\perp \subseteq E_\epsilon$, where $U^\perp$ is the subspace that consists of vectors orthogonal to U. Note that $U^\perp$ is a subspace of E_ϵ. Indeed, if $v \in U^\perp$, then

$$\mathbb{A}^{-1}v = \left(\mathbb{V}_{\mathbb{F}}\mathbb{V}_{\mathbb{F}}{}^T + \epsilon \cdot \mathbb{E}\right)v = \left(\sum_l v_{(i_l,j_l)}v_{(i_l,j_l)}^T + \epsilon \cdot \mathbb{E}\right)v$$

$$= \sum_l v_{(i_l,j_l)} v_{(i_l,j_l)}^T v + \epsilon \cdot v = \sum_l \langle v_{(i_l,j_l)}, v\rangle v_{(i_l,j_l)} + \epsilon \cdot v$$
$$= \epsilon \cdot v. \quad (11)$$

Also, U is an invariant subspace of A^{-1}. To see this, assume that $u = \sum_l x_l v_{(i_l,j_l)}$ is a vector from U. Then $A^{-1} =$

$$\left(\sum_l v_{(i_l,j_l)} v_{(i_l,j_l)}^T + \epsilon * \mathbb{E}\right) u = \sum_l v_{(i_l,j_l)} \langle v_{(i_l,j_l)}, u\rangle + \epsilon u \in U. \quad (12)$$

To prove the last part of the lemma, it is enough to show that $\mathbb{A}^{-1}|_U$ is non-degenerate. $\mathbb{A}^{-1}|_U : U \to U$ is an operator that changes the vectors from U in the same way as $\mathbb{A}^{-1} : \mathbb{A}^{-1}|_U \cdot u = \mathbb{A}^{-1} \cdot u$ for any vector $u \in U$. From Eq. 12 it follows that $\mathbb{A}^{-1}|_U$ is a Gram matrix [29] of the vectors $v_{(i_1,j_1)}, \ldots, v_{(i_n,j_n)}$. But $v_{(i_1,j_1)}, \ldots, v_{(i_n,j_n)}$ are independent vectors and consequently their Gram matrix is non-degenerate. □

One implication of Lemma 3.1 is that E_ϵ is equal to $U^\perp$. Also, if we choose the basis of U to be $\{\sqrt{\lambda_1}e_1, \ldots, \sqrt{\lambda_n}e_n\} = \{e'_1, \ldots, e'_n\}$ and decompose the latent presentation of the new question by this basis $v_q = \sum_l a_l e'_l + e_\epsilon$, then the numerator of Eq. 8 will simply be

$$\sum_l {a_l}^2 + 1, \quad (13)$$

which is the length of the part of v_q that is parallel to U in this basis, increased by one.

To understand the intuition behind the denominator of Eq. 8 we need another piece of notation: we denote the projection of vector v on E_ϵ as v_ϵ. Suppose question q is about the preference of i over j, then the denominator of Eq. 8 equals

$$\|v_{q,\epsilon}\|^2 = \|(v_i - v_j)_\epsilon\|^2 = \|v_{i,\epsilon} - v_{j,\epsilon}\|^2. \quad (14)$$

That means that the denominator of Eq. 8 is a Euclidean distance between the projection of v_i and v_j or a length of $v_{q,\epsilon}$. Consequently, in order to add one more question to $\mathbb{F}$, we should find a vector with a large component from $U^\perp$ and with a small component from U. Thus, if the questions from $\mathbb{F}$ are linearly independent, then the question that will be added to $\mathbb{F}$ should be linearly independent of all previous questions, and the extended questionnaire will consist of linearly independent questions.

Now we are ready to present Algorithm 1, *static preference questionnaire* (SPQ), for generating preference questionnaires.

Algorithm 1 Static Preference Questionnaire (SPQ).

Input: $\{e'_1, \ldots, e'_n\}$ basis of U; $\mathbb{E}$ the set of all items and $r > 0$
for $i \in I$ **do**
  $v_i = v'_i + v_{i,\epsilon}$, where $v'_i \in U, v_{i,\epsilon} \in U^\perp$
$\mathbb{S} = \{i \in \mathbb{E} : \|v'_i\| < r\}$
$\mathbb{S}_\perp = \{v_{i,\epsilon} \text{ for } i \in \mathbb{S}\}$
Find the farthest points $(i, j) \in \mathbb{S}_\perp$ by Euclidean distance.
return $v_{(i,j)}$

To minimize the numerator of Eq. 8, we pick a small number r ($r = 1$ in our experiments) and choose the set of items $U_{<r}$ on which Eq. 13 is smaller than it. To maximize the denominator of Eq. 8, the algorithm chooses among $U_{<r}$ the question q with max $\|v_{q,\epsilon}\|^2$. As pointed out above, this problem is equivalent to finding the diameter of the projections of points from $U_{<r}$ to E_ϵ. The diameter finding problem does not have an exact solution in linear time [8], thus we use a well-known approximation to find it; see Algorithm 2 [7].

Algorithm 2 Computing the diameter of a set of points [7].

Input: The set of points $\mathbb{S}$
Choose a random point $p \in \mathbb{S}$
Find the farthest point from p: p_1 by Euclidean distance
Find the farthest point from p_1: p_2 by Euclidean distance
return p_1, p_2

4 EXPERIMENTAL SETUP

4.1 Research questions

We assume that a good questionnaire consists of a small number of questions in order not to bother the user too much. At the same time, information received after completing the questionnaire should be enough to provide good recommendations and to understand the user's preferences. Thus, our research questions are the following:

RQ1 The final goal of SPQ is to provide enough information to build a high-quality list of personalized recommendations for a new "cold" user. Does minimizing Eq. 2 correspond to a better quality of the recommendations?

RQ2 What is the impact of the preference elicitation procedure used on the final quality of recommendations?

4.2 Experimental methodology

4.2.1 Datasets. We conduct experiments on the MovieLens dataset, with explicit ratings, and an Amazon book dataset, with implicit feedback. As a ground-truth answer r^*_{uij} to the question of the preference between two elements i and j, that have received different ratings $r_{ui} \neq r_{uj}$, we use 1 if $r_{ui} < r_{uj}$ and -1 otherwise. We processed the datasets subject to the following constraints: (1) in the dataset there is a sufficient amount of information in order for LFM to be reliable; (2) personalization really improves the quality of the recommendation of the items in the data set; (3) users in the dataset tend to evaluate items they like and items they do not like .

MovieLens. The original MovieLens dataset [15] contains 27,000 movies, 20 million ratings, that are scored by 138,000 users. These 5-star ratings were collected between January 9, 1995 and March 31, 2015. The dataset is processed as follows. First, following [22] we are interested in the ability of algorithms to distinguish good recommendations from bad ones. Therefore, we binarize the ratings as in [22]: ratings that are scored by 4 stars and above become positive, others become negative. Then, for the constructed dataset to satisfy the first constraint mentioned above, we only retain movies that have been rated by at least five users. In addition, for the created dataset to satisfy the second constraint, we keep only movies for which the ratio between the number of negative ratings and the positive ratings received by these movies is less than three. Finally, we keep only users for whom the ratio between the number of negative and positive ratings given by each of these users is less than three. Thereby the dataset satisfies the third constraint.

The final dataset consists of 10 million ratings, produced by 89,169 users, for 11,785 movies.

Amazon books. This dataset contains ratings for books as part of users' reviews [23]. One drawback of collecting ratings this way is that users rate and write reviews only for books that they bought and read. So, the descriptions and genres of the negatively rated books can potentially satisfy user's preferences. Therefore, we create a new dataset using the reviews from Amazon. In the constructed dataset a rating is positive if a user wrote a review about the book, and consequently, bought and read it.

Negative ratings are created by uniformly sampling from books that the user did not read. For each user, the number of positive ratings equals the number of negative ratings. In order for the data set to satisfy the first constraint (that LFM is reliable), we only include the 30,000 most popular books. We only keep users who rated from 20 and 1,000 books. This leaves us with a dataset with 3,200,000 ratings, from 33,000 users for 30,000 books; all books included in the dataset received at least 5 reviews. The dataset satisfies the second and third constraint because users and items have diverse ratings as the data was obtained by negative sampling.

4.2.2 Baselines. We are the first to propose a preference elicitation method that creates a *static* questionnaire containing relative questions about user preferences. To the best of our knowledge, there are no direct baselines to compare with. Therefore, we compare SPQ with three baselines designed for solving a similar but not identical task. However, they all optimize the Presicion@k measure, which we also use to compare the methods. Moreover we use them in the setting for which they were designed. As baselines we use two state of the art methods that create a preference questionnaire, and one method that creates an absolute questionnaire:

- Bandits [6] and Pair-Wise Decision Trees (PWDT) [26] that create a dynamic questionnaire by asking relative questions;
- Forward Greedy [2] that creates a static questionnaire but contains absolute questions.

In addition, we compare SPQ with a random baseline that selects relative questions randomly.

Importantly, the main goal of this paper is to create a *preference questionnaire* in an optimal way, thereby avoiding problems from which absolute questionnaires suffer. Thus, the most appropriate baselines for SPQ are methods that create preference questionnaires.

Bandits. Christakopoulou et al. [6] present a bandit-based method for creating a dynamic questionnaire by asking relative questions in an online setting. They propose the following algorithm for picking two items for asking relative questions: (1) select the first item that the recommender algorithm predicts to be the best for the user in the current stage; (2) virtual observation and update: assume that the user does not like the first item and virtually update the model including this knowledge; (3) select a comparative item that the algorithm, using the virtually updated model, predicted to be the best for the user in the current stage.

PWDT. Rokach and Kisilevich [26] propose the PWDT algorithm, which solves the same problem as Bandits, i.e., it creates a dynamic questionnaire asking relative questions. PWDT minimizes the weighted generalized variance in each step. To this end, the algorithm needs to brute force all pairs of items. To avoid this, the authors suggest to first cluster items, using the k-means algorithm with cosine similarity. Then they create artificial items, which are the centroids of these clusters, and use only these artificial items to choose the next question. The PWDT algorithm is dynamic, which means that it must perform every step fast; it remembers all answers and questions that it has received and asked earlier, using a lazy decision tree [10].

Forward Greedy. Anava et al. [2] propose a Forward Greedy (FG) approach that creates a static questionnaire consisting of absolute questions. Users are asked to rate one item per question. FG has the same assumptions as SPQ (listed in Section 2.1) and it also solves optimization problem which similar to our formulation in (2). Anava et al. [2] prove an upper bound on the error of the Backward Greedy (BG) algorithm and show that FG and BG achieve similar results on realistic datasets. Therefore, we only use FG for comparisons. Our goal is to provide a method for creating a static *preference questionnaire* in an optimal way, thereby avoiding problems from which absolute questionnaires suffer. However, it is useful to compare the proposed method with a reasonable absolute baseline to understand if the quality of the preference questionnaire is comparable to the quality of absolute questionnaires in terms of the final list of recommendations.

Random. Finally, we compare SPQ with a baseline that randomly picks pairs of items for questions. We include this comparison to better understand the importance of carefully selecting questions.

Algorithm 3 Evaluation of an elicitation procedure.

Input: Set of items $\mathbb{I}$, the set of cold users $\mathbb{U}_{cold}$
$Loss = 0$
divide $\mathbb{I}$ into two subsets of equal size randomly: $\mathbb{I}_{test}$, $\mathbb{I}_{train}$
for each $u \in \mathbb{U}_{cold}$ **do**
  create a questionnaire, consisting of questions about items from $\mathbb{I}_{train}$
  receive u's completed questionnaire
  predict v_u: $\hat{v}_u$
  choose test questions about items from $\mathbb{I}_{test}$
  using $\hat{v}_u$ predict answers on test questions: $Answers_u$
  increase the loss $Loss \rightarrow Loss + Loss(Answers_u)$
return $Loss$

4.2.3 Training latent factor models (LFMs). We stress that training LFM is not the task of this paper; instead, we just use its parameters: μ, v_i, b_i, v_u, b_u (presented in Eq. 1) for (1) the ground truth and (2) our recommender system. We factorize users' ratings matrix R using LFM (Eq. 1), which is trained to minimize the Root Mean Squared Error (RMSE) using Stochastic Gradient Descent (SGD). We first randomly pick one thousand users as "cold start users". Then we train LFM on all ratings by all other users, who are "warm" users. To choose a combination of hyper-parameters, namely the latent dimension and regularization, we create a validation set that consists of ten random ratings for each "warm" user who has rated at least twenty items. We set the hyper-parameters to the values that achieve the best accuracy on the validation set.

In order to achieve statistically meaningful results, we perform this procedure ten times. I.e., the following is repeated ten times: (1) choose the subset of cold-start users randomly; (2) train LFM using the ratings obtained from all other users; (3) create a questionnaire; and (4) measure its quality.

4.3 Experimental conditions

We randomly divide all items into two equal subsets: (1) the training items that are used to select questions; (2) the test items that are used to measure performance. For all experimental conditions we use a standard procedure to measure the performance [9], as described in Algorithm 3: (1) we receive a completed questionnaire for each user; (2) we receive predictions of the users' latent presentation using Eq. 3 and ground truth presentation of the question ($v_{(i,j)}$ and $b_{(i,j)}$); and (3) we use this presentation to predict the user's ratings on the chosen subset of the test items. We have two experimental conditions, *real* (Section 4.3.1) and *simulated* (Section 4.3.2), which differ in how they receive users' answers and choose test questions.

4.3.1 Real condition. As described above, we first split all items into two equal-sized subsets randomly: (1) training items that are

Algorithm 4 Choosing a subset from the test items in the Simulated condition.

Input: Set of test items $\mathbb{I}_{test}$, user u's ground truth vector v_u
initialize subset from the test items $\mathbb{I}_{testSubset} = \emptyset$
using (1) receive all ratings by u to $\mathbb{I}_{test}$
get the extended recommendation list $L = \{i_1, \ldots, i_n\}$ by sorting items according to their ratings
divide items into 50 groups $S_1, \ldots S_{50}$:
$S_k = \{i_{(k-1)*(n/50)+1}, \ldots, i_{(k-1)*(n/50)+n/50}\}$
for each group S in $S_1, \ldots S_{50}$ **do**
  choose the random item $i \in S$
  add i to $\mathbb{I}_{testSubset}$
return $\mathbb{I}_{testSubset}$

used to select questions; and (2) test items that are used to measure the quality. In this condition, for each user, we create questions only about items that she has rated. We also measure the quality of the elicitation procedure using the subset of the test items that the user has rated. Consequently, we can infer users' answers using the ratings that are available in the dataset. Thus, in this setup, we do not simulate answers and do not need any assumptions to receive users' answers. However, optimal items may not be available. Also, despite the fact that we use the same elicitation procedure for all users, the questionnaires will be different for different users because they rate different subset of items. To avoid this problem and ask all users the same questions, we also perform a simulated experiment.

4.3.2 Simulated condition. In order to ask all users the same questions and to choose the questions in an optimal way we require the mechanism for obtaining answers to the questions. However, we cannot obtain this information directly from ratings in the datasets that we use, because users rate different subsets of items and only a small fraction of all items. Thus, following [6], in the *simulated condition* we simulate users' answers using Eq. 3. This way of obtaining users' answers is motivated by the following assumption: all ratings comply with the LFM model that is given to us. We use this simulation procedure to get answers for both sets of questions from (1) the questionnaire; and (2) the subset of test questions. It is important to point out that, like SPQ, Bandits, PWDT, and FG also rely on the assumptions about pretrained LFMs described in Section 2.1. Thus, using the simulation condition does not favor any method, baseline or otherwise.

Algorithm 4 provides detailed information on how to choose a subset from the test items to generate the test questions. First, for a given user we divide all test items into 50 groups according to her preference. Then we pick one item per group. This procedure picks only 50 items instead of thousands of test items; moreover, the chosen 50 items are very diverse.

4.4 Evaluation methodology

To answer our research questions we use different metrics.

4.4.1 RQ1. To understand whether it is right to optimize Eq. 2, we should provide two measures: (1) one that reflects how well Eq. 2 is optimized; and (2) another one that reflects the quality of the final personalized list of recommendations. We choose (1) Precision of the classification of all preference questions (PCPF) between any two test items: PCPF is equal to the fraction of correctly classified pairwise preferences between items; and (2) Precision@10 of the recommendation list (P@10). We would like to observe if the algorithm with the lowest PCPF has the highest P@10.

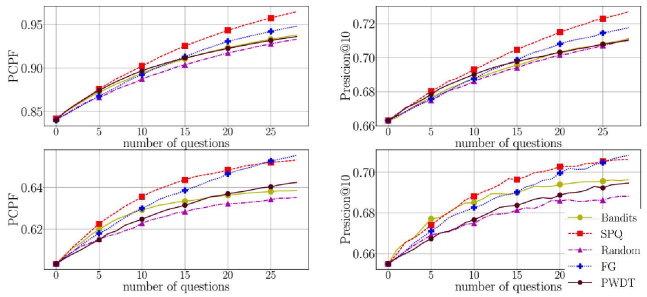

Figure 2: Results on the MovieLens movie dataset. (Top): Simulated condition. (Bottom): Real condition. (Left): PCPF. (Right): P@10.

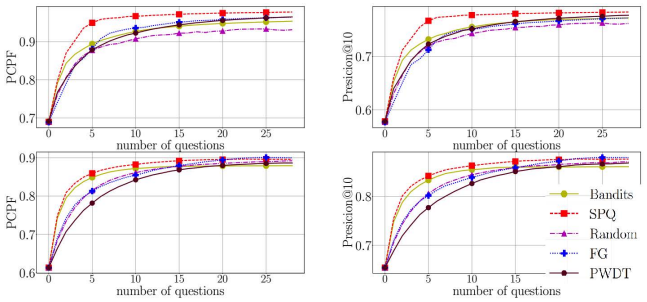

Figure 3: Results on the Amazon books dataset. (Top): Simulated condition. (Bottom): Real condition. (Left): PCPF. (Right): P@10.

4.4.2 RQ2. The goal of a recommender system is to provide to a user the most relevant recommendations. That is why we use ranking measures to evaluate all methods, specifically P@10, to compare the quality of the final personalized lists. Precision@k is the fraction of relevant items among the top-k recommendations. It is one of the most popular measure for the recommender systems. Our final evaluation measures were computed by averaging Precision@k over all users in the test set.

5 RESULTS

Our experimental results are shown in Fig. 2 and 3 for the MovieLens movie dataset and the Amazon book dataset, respectively.

Importantly, the maximum Precision@10 score depends on the dataset and LFM: for some users, there are fewer than 10 positive ratings. We chose the range of the plots for Precision@10 measure from (1) its minimum value for the non-personalized recommendation; (2) until its maximum value, the value achieved by calculating the Precision@10 for the ideal ranking in which items that a user likes are all on the top. The maximum value for Precision@10 in the real condition for the MovieLens and Amazon Books datasets are 0.71 and 0.90, respectively. And the maximum value for Precision@10 in the simulated condition for the MovieLens and Amazon Books datasets are 0.75 and 0.89, respectively.

5.1 RQ1

To answer RQ1 we should understand if the property listed in Section 4.4.1 holds for every pair of algorithms and all conditions.

For pragmatic reasons we answer RQ1 by comparing methods that achieve state of the art results in terms of Precision@10. Let's turn to the plots shown in the left and center columns of Fig. 2 and 3. We observe that for every pair of methods A and B, and every budget size N (≤ 29), if method A significantly outperforms B in terms of PCPF, then method B does not significantly outperform A in terms of P@10. Hence, we were correct in optimizing Eq. 2 to produce better lists of recommendations for a user after receiving her completed questionnaire.

5.2 RQ2

To answer RQ2 we turn to Fig. 2 and 3 (right-hand side).

5.2.1 MovieLens movie dataset. All algorithms provide the same performance for questionnaires with a length less than 5: the differences in performance in terms of P@10 between the elicitation methods are not statistically significant ($p > 0.1$) for any pair of methods. In all cases except one, SPQ works significantly better than other methods if the budget is large (P@10; $p < 0.01$); only FG achieves the same performance as SPQ in the Real condition for a large budget ($N > 20$). In the Real condition with a medium size budget SPQ significantly outperforms all other algorithms.

5.2.2 Amazon book dataset in the Real condition. SPQ achieves near-perfect performance only with budgets that are larger than 15; for the smaller budget it has a similar but statistically significantly better performance than Bandits; for budgets larger than 15, SPQ has a similar performance as FG. FG is statistically better than SPQ when the budget is more than 25, but the difference with the quality that is achieved by this algorithm in terms of Precision@10 is less than 0.5%. This finding can be explained by the fact that for long questionnaires (with $N > 15$), in the Real condition, the following property holds: a completed absolute questionnaire provides the same information that is contained in the dataset about a user. Hence, for a long questionnaire the FG method achieves the best result for this experimental condition. But this does not imply that for real users it would lead to the same top performing results.

To conclude our answer to RQ2, the importance of the elicitation method depends on the dataset. We distinguish two cases. The first case is datasets like the MovieLens movie dataset, where a non-personalized list of recommendations based on item popularity achieves good results. In that case for a small budget, such as $N = 5$, it is not important to carefully choose the questions. But for a larger budget ($5 < N \leq 15$), it is important to choose a method other than Random. For long questionnaires ($N > 15$), SPQ is the best choice. The second case is for datasets such as the Amazon book dataset, for which a non-personalized recommendation list has far worse quality than a personalized one. In this case, it is always beneficial to use SPQ to create a questionnaire.

There are two important properties of the elicitation method that should be considered when choosing a method. First, the function the elicitation method minimizes to select a question. We concluded that if the method directly optimizes the loss function, then it works better than others. Moreover, the difference becomes noticeable if the questionnaire is long enough: SPQ and FG achieve better performance than PWDT, despite the fact that they are static, while PWDT is dynamic. A possible explanation is that PWDT optimizes a function that is different from the loss function, namely weighted generalized variance. Second, constraints on selecting items to create a question. The Bandits method selects items that, as it predicts, will be liked by the user. This constraint affects performance, especially when the questionnaire is long ($N > 15$) because at this time Bandits already predicts some user preferences (but not all) and starts to ask questions about the elements that satisfy these preferences. However, the problem that we solve in this paper has no limitations on the choice of items. Instead, methods that do not have this limitation still ask very informative questions and, therefore, achieve better performance: SPQ selects informative questions and achieves better performance than Bandits, while being static.

5.2.3 Amazon book dataset in the Simulated condition. SPQ is statistically significantly better than others for a budget that is larger than 3 ($p < 0.01$). Moreover, it achieves a high performance after asking only 5 questions, while other algorithms achieve similar results only for a budget that is larger than 15. Thus, compared to other elicitation methods, the length of the questionnaire can be reduced by a factor of three for the same recommendation performance. In all other cases, LFM achieves a near perfect result (on the validation set, RMSE < 0.45).

6 RELATED WORK

6.1 Cold start problem

In collaborative filtering (CF), ratings and implicit information are used to provide recommendations [28]. Unlike content-based methods, CF does not require feature engineering. Two popular approaches are clustering [32] and latent factor models (LFMs) [27]. In LFM, items and users are represented as vectors in a latent space that is automatically inferred from observed data patterns. The dimensions of the space represent information about user preferences and properties of items. The recommendation algorithm predicts ratings using the representations. An influential realization of this approach is matrix factorization (MF) [27]. MF, like all CF methods, cannot provide reliable ratings for users with limited history and we run into the new user cold-start problem [3, 19].

Previous research suggests different approaches to the new user cold-start problem. Li et al. [21] suggest to model representative vectors, not only recommending the best items (exploit), but also getting information about user preferences (explore). Such approaches are usually used for recommender domains that are dynamic: items appear very fast, and usually, new items are relevant (for example, news and ad recommendation). For a broader range of domains, a popular method for solving the new user cold-start problem involves questionnaire-based[2] approaches where new users are asked a seed set of questions [6, 11, 24–26].

6.2 Questionnaire-based approaches to the new user cold-start problem

6.2.1 Static questionnaire. The standard procedure for creating a static questionnaire chooses seed items independently of new users, and then users rate these items [24]. The underlying algorithmic problem is how to build a seed set that can yield enough preference information to build a good recommender system. One approach is to formulate it as an optimization problem; Anava et al. [2] find a mathematical expression for the expectation of the number of wrong predictions of ratings after a user has answered a questionnaire. Rashid et al. [24]'s method selects questions that are not only informative but also satisfy the constraint that users know the

[2]In the literature, such methods are also called "interview-based." To avoid ambiguity, we stick with "questionnaire-based."

items in the questions and are therefore able to make fast decisions. The set of questions can be chosen based on diversity and coverage of the set [9, 22]. That is, latent vectors of the selected questions should have the largest length yet be as orthogonal as possible to each other, i.e., the parallelepiped spanned by these latent vectors should have maximum volume. Fonarev et al. [9], Liu et al. [22] propose to use Maxvol [13] to find such items.

In line with [2], we propose a method that creates a static questionnaire by solving an optimization problem. But SPQ is the first one to do it for relative questions of the kind shown in Fig. 1.

6.2.2 Dynamic questionnaire. In a dynamic[3] questionnaire the next question depends on previous questions and answers. A popular way to adapt questions to users' answers is to build a decision tree [12, 16]. In each node of the tree, there is a question. The children of the node are the next questions depending on the answer to the question in the current node. Thus, each node has as many children as there are answers to the current question. Consequently, dynamic methods can be memory inefficient, but usually, they provide better results than static ones. The method presented in this paper, while static and memory efficient, achieves better results than dynamic baselines. We leave a dynamic variant as future work.

6.2.3 Asking relative questions. Another way to solve the new user cold-start problem is to ask relative questions. As stated in [17], this approach has all the advantages of pairwise preference approaches: it is faster and easier for a user to answer relative questions, and users' relative preferences are more stable over time. Jones et al. [17] also show that relative questions can deal with the rating calibration issue. However, the number of possible relative questions is proportional to the square of the number of items and therefore it is computationally expensive to select optimal preference questions. Thus, only a few papers have so far used this approach [6, 26].

There are several solutions to avoid brute-forcing all pairs of items. Rokach and Kisilevich [26] suggest to cluster items before selecting questions. However, valuable information about items may be lost in the clustering. Moreover, this algorithm can be memory inefficient due to the need to memorize all answers of all users. Christakopoulou et al. [6] present several bandit-based algorithms for online recommendations, including asking relative questions in an online setting.

To summarize, we propose a method that creates a static questionnaire consisting of relative questions, which allows us to build a personalized list of recommendations for a new user. Similarly to [2], we formulate our task as an optimization problem, but we ask relative questions [6, 26] instead of absolute ones. Therefore, we cannot perform the kind of brute-force search of all available questions that is performed in [2]. Hence, we propose a new solution for the optimization task defined in [2]. Our method differs from [26], as we offer a procedure that selects the most informative questions without clustering items, thus avoiding losing any information. Moreover, our method solves the optimization problem that minimizes the expectation of misclassified personalized preferences of one item over another, while the method in [26] minimizes the weighted generalized variance, which is not directly related to the loss that we minimize. The main advantage of SPQ over [6] is that we optimize informativeness of questions without constraints.

[3]In the literature, such methods are also called "interactive." To avoid ambiguity, we stick with the term "dynamic."

7 CONCLUSION AND FUTURE WORK

We have proposed a static preference questionnaire generation method, called SPQ, that extends earlier work [2, 6, 11, 24–26] on approaching the new user cold-start problem for recommender systems by asking a set of seed questions. Our main research question is *How to optimally generate a preference questionnaire, consisting of relative questions, for new users that will help to solve the new user cold-start problem?* Generating a short questionnaire that is sufficiently informative so as to derive a high-quality list of personalized recommendations, is crucial for a recommender system to be able to engage new "cold" users. We are the first to address the problem of generating a list of relative questions as an optimization problem. The use of relative questions is beneficial [17] because: (1) users find it easier to provide feedback through relative questions than through absolute ratings; (2) absolute ratings suffer from calibration issues; and (3) answers on relative questions are stable over time.

We have demonstrated theoretically how to solve the optimization problem and how to avoid brute-forcing all possible questions as is implemented in [2]. Also, we have shown experimentally that minimizing the proposed SPQ objectives leads to a high-quality list of personalized recommendations. We have performed experiments on two datasets: the MovieLens movie dataset and the Amazon book dataset. These datasets differ in the type of user feedback and in the diversity of items. Also, we have considered two experimental conditions to evaluate elicitation procedures: (1) experiments in *Simulated* shows results in the ideal situation when users behave according to the latent factor model (LFM); and (2) experiments in the *Real* condition do not rely on any user rating model. We have compared SPQ with multiple state-of-the-art baselines [2, 6, 26] that solve similar but different tasks. SPQ outperforms all baselines on both datasets independently of the length of the questionnaire, and it achieves a statistically better performance than the baselines for questionnaires whose length exceeds 5. On the Amazon book dataset in the real condition, SPQ was able to reduce the length of the questionnaire by a factor of three. Moreover, SPQ, a static method, has demonstrated better results than the dynamic baselines [6, 26]. SPQ outperforms the dynamic methods because it optimizes the loss function and has no constraints on the questions, while dynamic baselines either optimize a substitute function instead of the LFM loss function or have constraints on the questions that are not necessary for the problem solved in this paper.

In future work, we plan to work on the following important aspect to improve a questionnaire. SPQ may ask very informative questions about items that a user might not like. To overcome this problem, we plan to make SPQ dynamic to make sure that we ask attractive questions for the user while receiving almost the same amount of information.

Code and data

To facilitate reproducibility of the results in this paper, we are sharing the code to run our experiments at https://github.com/Seplanna/pairwiseLearning.

Acknowledgements

This research was partially supported by the Google Faculty Research Awards program and the Microsoft Research Ph.D. program. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Xavier Amatriain, Josep M Pujol, and Nuria Oliver. 2009. I like it... I like it not: Evaluating user ratings noise in recommender systems. In *UMAP*. Springer, 247–258.

[2] Oren Anava, Shahar Golan, Nadav Golbandi, Zohar Karnin, Ronny Lempel, Oleg Rokhlenko, and Oren Somekh. 2015. Budget-constrained item cold-start handling in collaborative filtering recommenders via optimal design. In *WWW*. ACM, 45–54.

[3] Lucas Bernardi, Jaap Kamps, Julia Kiseleva, and Melanie J. I. Müller. 2015. The continuous cold start problem in e-commerce recommender systems. In *Proceedings of the Workshop on New Trends on Content-Based Recommender Systems co-located with ACM Conference on Recommender Systems*. 30–33.

[4] Christopher JC Burges. 2010. From RankNet to LambdaRank to LambdaMART: An overview. *Learning* 11, 23-581 (2010), 81.

[5] Census Bureau 2015. E-Stats: Measuring the electronic economy. https://www.census.gov/econ/estats/.

[6] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *KDD*. ACM, 815–824.

[7] Ömer Eğecioğlu and Bahman Kalantari. 1989. Approximating the diameter of a set of points in the Euclidean space. *Inform. Process. Lett.* 32, 4 (1989), 205–211.

[8] Daniele V. Finocchiaro and Marco Pellegrini. 2002. On computing the diameter of a point set in high dimensional Euclidean space. *Theoretical Computer Science* 287, 2 (2002), 501–514.

[9] Alexander Fonarev, Alexander Mikhalev, Pavel Serdyukov, Gleb Gusev, and Ivan Oseledets. 2016. Efficient rectangular maximal-volume algorithm for rating elicitation in collaborative filtering. arXiv preprint arXiv:1610.04850.

[10] Jerome H. Friedman, Ron Kohavi, and Yeogirl Yun. 1996. Lazy decision trees. In *AAAI/IAAI, Vol. 1*. 717–724.

[11] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. 2010. On bootstrapping recommender systems. In *CIKM*. ACM, 1805–1808.

[12] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. 2011. Adaptive bootstrapping of recommender systems using decision trees. In *WSDM*. ACM, 595–604.

[13] Sergei A. Goreinov, Ivan V. Oseledets, Dimitry V. Savostyanov, Eugene E. Tyrtyshnikov, and Nikolay L. Zamarashkin. 2010. How to find a good submatrix. In *Matrix Methods: Theory, Algorithms, Applications*, Vadim Olshevsky and Eugene E. Tyrtyshnikov (Eds.). World Scientific, Hackensack, NY, 247–256.

[14] William W. Hager. 1989. Updating the inverse of a matrix. *SIAM Rev.* 31, 2 (1989), 221–239.

[15] F. Maxwell Harper and Joseph A. Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2016), 19.

[16] Fangwei Hu and Yong Yu. 2013. Interview process learning for top-n recommendation. In *RecSys*. ACM, 331–334.

[17] Nicolas Jones, Armelle Brun, and Anne Boyer. 2011. Comparisons instead of ratings: Towards more stable preferences. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, 451–456.

[18] Saikishore Kalloori, Francesco Ricci, and Marko Tkalcic. 2016. Pairwise preferences based matrix factorization and nearest neighbor recommendation techniques. In *RecSys*. ACM, 143–146.

[19] Julia Kiseleva, Alexander Tuzhilin, Jaap Kamps, Melanie J. I. Muller, Lucas Bernardi, Chad Davis, Ivan Kovacek, Mats Stafseng Einarsen, and Djoerd Hiemstra. 2016. Beyond movie recommendations: Solving the continuous cold start problem in e-commerce recommendations. *CoRR* 1607.07904 (2016).

[20] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.

[21] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*. ACM, 661–670.

[22] Nathan N. Liu, Xiangrui Meng, Chao Liu, and Qiang Yang. 2011. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In *RecSys*. ACM, 37–44.

[23] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. ACM, 43–52.

[24] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. 2002. Getting to know you: learning new user preferences in recommender systems. In *IUI*. ACM, 127–134.

[25] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning preferences of new users in recommender systems: an information theoretic approach. *ACM SIGKDD Explorations Newsletter* 10, 2 (2008), 90–100.

[26] Lior Rokach and Slava Kisilevich. 2012. Initial profile generation in recommender systems using pairwise comparison. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 6 (2012), 1854–1859.

[27] Ruslan Salakhutdinov and Andriy Mnih. 2008. Probabilistic matrix factorization. In *NIPS*, Vol. 1. 1257–1264.

[28] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The Adaptive Web*. Springer, 291–324.

[29] Hans Schwerdtfeger. 1950. *Introduction to Linear Algebra and the Theory of Matrices*. Noordhoff.

[30] Anna Sepliarskaia, Filip Radlinski, and Maarten de Rijke. 2017. Simple personalized search based on long-term behavioral signals. In *ECIR (LNCS)*. Springer, 95–107.

[31] Amit Sharma and Baoshi Yan. 2013. Pairwise learning in recommendation: experiments with community recommendation on linkedin. In *RecSys*. ACM, 193–200.

[32] Lyle H. Ungar and Dean P. Foster. 1998. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*. AAAI, 114–129.