Towards Conversational Recommender Systems

Konstantina Christakopoulou*
University of Minnesota
United States
christa@cs.umn.edu

Filip Radlinski, Katja Hofmann
Microsoft
Cambridge, UK
{filiprad, katja.hofmann}@microsoft.com

ABSTRACT

People often ask others for restaurant recommendations as a way to discover new dining experiences. This makes restaurant recommendation an exciting scenario for recommender systems and has led to substantial research in this area. However, most such systems behave very differently from a human when asked for a recommendation. The goal of this paper is to begin to reduce this gap.

In particular, humans can quickly establish preferences when asked to make a recommendation for someone they do not know. We address this cold-start recommendation problem in an online learning setting. We develop a preference elicitation framework to identify which questions to ask a new user to quickly learn their preferences. Taking advantage of latent structure in the recommendation space using a probabilistic latent factor model, our experiments with both synthetic and real world data compare different types of feedback and question selection strategies. We find that our framework can make very effective use of online user feedback, improving personalized recommendations over a static model by 25% after asking only 2 questions. Our results demonstrate dramatic benefits of starting from offline embeddings, and highlight the benefit of bandit-based explore-exploit strategies in this setting.

Keywords

online learning; recommender systems; cold-start

1. INTRODUCTION

Recommendation is an everyday process that frequently touches people's lives. Hence, it has seen tremendous research interest (such as [9, 14]). Most work in recommendation falls into two broad classes: *Collaborative Filtering* starts with a set of user/item affinity scores and assumes that two users who agree about one item are more likely to agree about another item; *Content-Based Filtering* models users by the characteristics of the items they like or dislike. We note that neither model represents how real people make recommendations, particularly in a cold-start setting where the person making a recommendation does not know a lot about the person asking for one.

Consider what would happen if a conference attendee in your home town, whom you have never met before, asked for a recommendation on where to eat dinner today. Most likely, you would start with one or two clarifying questions, perhaps whether the person likes seafood, or whether they have a car. These questions would depend on the context; for instance if there are great restaurants around the corner, then whether they have a car would be irrelevant.

We argue that such an interaction can be represented using online learning, where two types of learning are occurring. First, the person making the recommendation is learning about the preferences of the person asking. However, the attributes learned will be *contextual*, based on the likely follow-on answers (such as the car question earlier). Second, the person making the recommendation is learning about which questions allow them to quickly reach a good recommendation in the current context. In this paper, we present a recommendation system that exhibits these two types of learning. Further, the learning is online: It immediately impacts future recommendations for this user, rather than requiring a batch reprocessing of information learned.

We present a bandit-based approach for online recommendation, applied to restaurant recommendation so as to ground it in a specific application. Our approach builds on top of prior work, such as [33], but learns to adapt the recommendation space (user-item embedding) to its users throughout their interactions. We use generalized Thompson Sampling to systematically sample questions to ask the user and to incorporate observed feedback. We further propose and compare a range of alternative question selection strategies to identify characteristics of approaches that most effectively learn users' preferences. We use a matrix factorization approach [21, 29] to learn and adapt the embedding.

Our main contributions are four-fold. (1) We propose a novel view of human-like recommenders that converse with new users to learn their preferences. (2) We successfully demonstrate a fully online learning approach for recommendation – both using absolute and relative feedback. (3) We propose a systematic approach to incorporating offline data to initialize online learning recommenders, and demonstrate performance improvements, even using weakly labeled offline data. (4) We propose a set of item selection strategies for deciding what question to ask to a cold-start user to

*This work was done during an internship at Microsoft.

KDD '16, August 13 - 17, 2016, San Francisco, CA, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4232-2/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2939672.2939746

most quickly infer preferences, and demonstrate benefits of bandit-based strategies. Our extensive experiments on both synthetic and real data evaluate each step of our approach.

Importantly, we note that this work is applicable to a wide variety of recommendation scenarios. Here, we focus on one such application, restaurant recommendation, where (1) we study search logs to understand the space of real user needs; (2) we use the insights to collect preference data in a user study; (3) we use online behavioral data to initialize the recommendation system; (4) we use both synthetic and user study data to evaluate our approaches.

We now present related work (Section 2), followed by an analysis of real-live restaurant search (3). We describe our model (4) and empirical setup (5), and validate the model on synthetic data (6). Finally, we evaluate on real data to further empirically analyze our learning framework (7).

2. RELATED WORK

This paper builds on work in multiple active research areas. We review recent work in the most closely related areas.

Offline Recommenders. The wide interest in personalized recommendations has sparked substantial research in this area [14]. The most common approaches are content-based approaches [24] and collaborative filtering (CF) [9, 21]. Collaborative filtering, which powers most modern recommenders, uses an a-priori available set of user-item ratings to learn the interdependencies among users and items. It predicts a user's rating on an item either via the neighboring items' ratings (neighbor-based [9, 28]) or by inferring latent factors in a low-dimensional embedding (latent factor-based [21, 29]). The majority of the work in recommendation has focused on offline recommenders, i.e., building an offline model based on past user-item interactions, periodically retrained to incorporate new observations.

Online Recommenders. Recently, it has been recognized that offline recommender approaches (i) suffer from the cost of retraining the model, (ii) are built to optimize offline performance which does not necessarily match online user behavior and (iii) fail to capture preference drifts. These realizations have initiated developments towards building continuously learning (online) recommenders.

A recent line of work poses the problem in a *contextual bandit* formulation [16, 31, 32], where items are seen as *arms*, users as *contexts*, and the goal is to *explore* the arm space in order to *exploit* the best performing arm for a given context. Most work in this area relies on the availability of user/item features and assumes that the reward of an arm for a given context is a linear/logistic function of the concatenated feature of arm-context [16, 31]. [32] uses Gaussian processes to allow feedback sharing among similar contexts and arms.

Using the CF point of view, [4] introduced an ϵ-greedy online user-user *neighbor-based* CF method. The first *latent-factor* online recommender was introduced in [33], which uses bandit strategies on top of Probabilistic Matrix Factorization (PMF) [21]. This is the most closely related work to ours. However, while [33] fixes the item latent factors to those learned offline, formulating the problem as a linear bandit, our method *fully online* learns *all* user and item parameters, including the biases; [33] can be seen as special case of our framework. In [13], the authors extend Thompson Sampling for PMF with a Kalman filter to track changing preferences over time. This work is orthogonal to ours.

Preference Elicitation. The problem of eliciting user feedback has long been of interest for a variety of tasks (e.g [8]). To elicit the preferences of an existing or new user (cold-start), a range of methods have been proposed, varying from interview-based strategies (e.g [30]), to asking users to rate some items, to active learning [26], entropy minimization [27], picture-based [22], and explore-exploit strategies on top of a latent factor model [33]. Our work is the first to elicit users' preferences by utilizing either absolute or relative feedback in a fully online setting.

Interactive Recommenders. There have been many works (critique-based [7], constraint-based [10], dialog, utility-based recommenders [19]) emphasizing the importance of interactivity in recommenders so that the user has a more active role over the recommendations. However, these works rely on prior modeling of the items' features, preventing the flexibility in adaptation to a different domain; thus a comparison with them is out of the scope of this work.

In contrast, our work, in the same spirit as a recent line of work [18, 11] initiated by [33], learns online the *latent factors* from PMF and uses *these* to do interactive preference elicitation. These methods, although close in principle, have significant differences from our work.[1] Briefly, in [18], the authors focus on set-based feedback and propose a progressive building of the user's latent factor in each question. In contrast, our work focuses on absolute and relative feedback on (pairs of) items, and uses a preference elicitation phase fully integrated with the online updates of the PMF model. The method of [11] focuses on choice-based feedback and updates online only the user's latent factor. Neither of [11, 18] balance the exploration-exploitation tradeoff.

3. UNDERSTANDING REAL USERS

A recommendation system that aims to satisfy real people should reflect how real people look for recommendations. Therefore, we start by analyzing restaurant-related search behavior in a commercial Web search engine. Our goals are two-fold. First, to gain insight into our target domain, namely understand the criteria people use to choose restaurants[2]. Second, to identify questions we must ask users to construct ground truth data for evaluating our system.

Given a large sample of all queries issued between 07/2014 and 07/2015, we filtered for those that contain *restaurant* or *dining*, and terms such as *for*, *near(by)*, *next (to)*, *close (to)*, *with* and *in*. Let $\mathcal{Q}$ be the most frequent 10,000 such queries.

3.1 Query Annotation with Entities

We tagged the top several hundred queries from $\mathcal{Q}$ as containing a location, name, cuisine, food type, or terms such as *best* and *menu*. The dictionary of tags was not pre-specified to avoid biasing our annotations due to prior beliefs about categories people *should* be looking for. Rather, our methodology used content analysis [25] to develop a coding that captures the dominant behaviors involved.

We found that 39% of queries specify a location, 19% a restaurant name, 9% cuisine constraints, 7% have the term *best* or other superlative. More rarely, queries specified food ethnicity (2%), an adjective describing the restaurant (2%), dish name (2%), decoration, etc. The most common term co-

[1]We plan to extend our framework to set and choice-based feedback, to allow the comparison with [18, 11].

[2]Keeping in mind that these criteria to some degree reflect users' perception of the search engine and its capabilities.

Table 1: Contextual terms in restaurant related queries.

IN	NEAR	FOR	WITH
nyc	me	wedding receptions	party room
chicago	by	large group	small private room
las vegas	times square	rich	piano
houston	here	dessert	live music
miami	lax	your 21st birthday	someone
los angeles	miami airport	steak at lunch time	a view
atlanta	airport	gluten free food	play area
brooklyn	fenway park	valentine day in dallas	banquet room

Table 2: Restaurant Feature Dictionaries (note: frequency counts have been rescaled)

Cuisine	Freq.	Adj.	Freq.	Food	Freq.
mexican	475	good	27	seafood	139
chinese	407	famous	13	sushi	40
italian	381	nice	12	steak	32
thai	174	romantic	11	bbq	29
indian	144	upscale	6	fish	24
japanese	125	small	6	tapas	19

occurences are location and name in the same query (29%), cuisine and location (10%), and *best* and location (8%).

3.2 Understanding Common Phrases

The above statistics show that location is an important factor in restaurant search. Hence we zoom in to discover the specific ways in which people constrain locations. Similarly, the context under which users search for a restaurant is a second common constraint. We analyzed the specific terms appearing after a variety of prepositions, and constructed a dictionary of the most frequent contextual constraints.

A sample of these is shown in Table 1. For example, the top places people search for restaurants *in* are nyc, chicago, las vegas. Using the preposition 'near' to indicate location, the majority of terms shows users want a restaurant *near me*. Queries can also be very specific, e.g. searching for a restaurant near points of interest (*times square*), airports (*miami airport, lax*), postal codes or entire addresses. The contexts under which users search for restaurants vary from an occasion (*wedding reception*), to dietary restrictions, to type of meal or a group of people. People also search for restaurants *with live music, piano, a view* etc.

3.3 Restaurant Feature Dictionaries

As user queries can be very specific (e.g., combining constraints, 'adjective & dish & great & location'), we now study the terms that people use to describe restaurants. We annotated the top 1,013 phrases appearing before the word 'restaurant' or 'dining' in 5,000 queries from $\mathcal{Q}$ with a number of traits. The traits identified are related to food (cuisine, meal, dietary restrictions, quality e.g. *organic, healthy*, buffet, menu), rating (e.g michelin star), atmosphere (view, *fancy* or *romantic*), time/location (opening hours, decade theme, time of the day) etc.

For every trait, we collected the most common terms. Table 2 shows a few common examples, giving us a glimpse into the most searched cuisines, food types, and adjectives. Though not shown, top amenities searched are *garden, jazz, patio, ocean* and top restaurant types are *bar, fast food, cafe.*

3.4 Outlook

Besides motivating our application scenario, this search log analysis forms the basis of the user study in Section 7. Given this understanding of *what to ask* people who look for a restaurant, we also focus on *how to ask it.* In this work, among the various combinations of the feedback type (explicit/implicit, absolute/relative/list) on the available content (explicit features/items/latent features), we elicit users' preferences using *explicit feedback* from *absolute* and *relative* questions about *explicit items* (e.g. restaurants).

4. MODEL

We next present our approach for determining 'what to ask' and 'how to ask' as the key pieces of a conversational recommender, starting with a high level picture of the entire algorithm (Section 4.1). Then, we describe the pieces in detail as we require (i) a model exploiting implicit structure among items and users to efficiently propagate feedback (Section 4.2); (ii) an explore-exploit approach to probe the space of items, to allow continuous learning (Section 4.3) and (iii) a feedback elicitation mechanism for selecting absolute (Section 4.3) and relative questions (Section 4.4).

4.1 Overview

Our recommendation pipeline can be summarized as:

1. Pick a model (Absolute/Pairwise) (Sec. 4.2) and preference elicitation mechanism: `Abs` (Sec. 4.3) / `Abs Pos` / `Abs Pos & Neg` / `Pairwise` (Sec. 4.4).
2. Initialize model parameters using offline data.
3. A new user arrives. Now iterate for a few questions[3]:
   (a) Mechanism selects a question to ask
   (b) User answers the question
   (c) All model parameters are updated
   (d) Remove the question from the allowed questions
4. System presents the final recommended list

The inner loop represents the 'human in the loop' present in all interactive systems, i.e., we make an intervention which affects the user and thus the future system decisions.

4.2 Latent Factor Recommendation

Consider how people make recommendations when a friend asks them to suggest a restaurant. They strategically ask each question with the goal of eliminating or confirming strong candidates for dining out. Similarly, when designing a conversational recommender that asks questions about explicit items, a principled way of selecting items is desired.

The implicit structure of the item space that allows us to learn quickly is motivated by *collaborative filtering.* Items that have been co-rated similarly (liked/disliked) by users will lie close in a low dimensional embedding. For our model we use a simplified version of the Matchbox Recommender model [29] – equivalent to Probabilistic Matrix Factorization (PMF) [21]. This is a generative model, in that it specifies a probabilistic procedure by which the observed likes/dislikes of users on items are generated on the basis of latent variables. The model variables are learned so that the model can explain the observed training data.

Formally, throughout the paper we use the convention that i denotes the index over M users, forming the set $\mathcal{I}$, and j denotes the index over N items, forming the set $\mathcal{J}$. Every user $i \in \mathcal{I}$ is modeled by a user bias variable $\alpha_i \in \mathbb{R}$, accounting for users who tend to like/dislike most of the items, and a d-dimensional trait vector $\mathbf{u}_i \in \mathbb{R}^d$. The trait vector represents the latent embedding of user i in a d-dimensional

[3]A study of the right stopping criterion is left for the future.

space, where $d \ll M, N$. Every item $j \in \mathcal{J}$ is modeled with latent variables $\beta_j \in \mathbb{R}$ (the item bias that accounts for item popularity) and a trait vector $\mathbf{v}_j \in \mathbb{R}^d$ that represents the latent embedding of item j in the *same* d-dimensional space.

Given that both users and items trait vectors lie in the same latent space, the similarity between a user i and an item j can be measured with the inner product of their corresponding trait vectors $\mathbf{u}_i^T \mathbf{v}_j$. We now present two models for estimating the latent variables, depending on the type of observations we obtain from users, i.e., absolute or pairwise.

Absolute Model. First, let us assume that we have observed tuples of the form (user i, item j, 1/0).[4] The model estimates the *affinity* of user i to item j based on the biases and traits. The generative procedure is:

1. User i has traits $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I})$, bias $\alpha_i \sim \mathcal{N}(0, \sigma_2^2)$.
2. Item j has traits $\mathbf{v}_j \sim \mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I})$, bias $\beta_j \sim \mathcal{N}(0, \sigma_2^2)$.
3. (a) The (unobserved) affinity is

$$y_{ij} = \alpha_i + \beta_j + \mathbf{u}_i^T \mathbf{v}_j. \qquad (1)$$

Observations are modeled as the noisy estimate $\hat{y}_{ij} \sim \mathcal{N}(y_{ij}, \epsilon_{ij})$, where ϵ_{ij} models the affinity variance, accounting for noise in user preferences. This yields an observation of whether the user likes an item ($\hat{r}_{ij}$):

$$\hat{r}_{ij} = \mathbf{1}[\hat{y}_{ij} > 0]. \qquad (2)$$

The hyper-parameters σ_1, σ_2 model the variance in traits and biases. The model variables are learned by maximizing the log-posterior over the item and user variables with fixed hyper-parameters, given the training observations.

Pairwise Model. People are often better at giving relative preferences over items instead of absolute judgments. Such preferences yield tuples of the form (user i, item j, item h) when user i prefers item j over item h (both j and h are indices over $\mathcal{J}$.) We can adapt the generative model to such ground truth data by modifying the third step of the *Absolute* model to yield a pairwise model:

3. (b) For each observation (i, j, h) compute

$$\text{noisy difference: } \hat{y}_{ijh} = \hat{y}_{ij} - \hat{y}_{ih} \qquad (3)$$

where the noisy item affinities are defined as before. Using $\hat{y}_{ijh}$, we estimate if user i prefers j over h:

$$j \succ_i h : \hat{r}_{ijh} = \mathbf{1}[\hat{y}_{ijh} > 0] \qquad (4)$$

4.3 Continuous Learning Recommender

To generate recommendations, most latent factor based recommender systems use an offline trained model based on past interactions, such as the one described above, that they periodically update to incorporate new data. The parameters for new users are typically initialized based on a combination of explicit attributes and past ratings (e.g [1]). The items with the highest predicted noisy affinity mean comprise the recommendations presented to the user.

In contrast, recent work has moved towards continuously learning recommenders [4, 16, 33]. This optimizes for *online* performance using explore-exploit (EE) strategies. We present our fully online updated recommendation approach with the embedded preference elicitation in Algorithm 1. Following we detail the components of this algorithm for the case of asking absolute questions (`Abs`). In Section 4.4 we show how we extend this framework for relative questions.

[4]We use the convention that 0 denotes dislike and 1 like.

Algorithm 1 Preference Elicitation Algorithm

```
Input: $\forall i \in \mathcal{I} : \mathbf{u}_i, \alpha_i$, $\forall j \in \mathcal{J} : \mathbf{v}_j, \beta_j$ (offline embedding)
1: A new user $i$ arrives. Initialize prior based on Eq. 5.
2: $\forall j \in \mathcal{J}$, infer noisy affinity $y_{ij}$ (Eq. 1)
3: while fewer than allowed questions have been asked do:
4:     Pick item(s) for absolute (relative) question (Sec. 4.3/ 4.4).
5:     Incorporate feedback according to (i) Abs (Sec. 4.3), (ii)
       Abs Pos, (iii) Abs Pos & Neg, or (iv) Pairwise. (Sec. 4.4)
6:     Update $\mathbf{u}_i$, $\alpha_i$, $\mathbf{v}$, $\beta$ (Section 4.3.3) and infer the noisy
       affinity distribution $\mathbf{y}_i$ (Eq. 1) or noisy difference (Eq. 3).
7: end while
```

4.3.1 Initialization from Offline Data

We propose to initialize online learning models using an initial embedding, that is learned offline. We hypothesize that such an initial embedding will allow the system to learn new user's preferences more quickly. Effective learning from few questions is crucial for conversational recommenders.

We start by learning the *offline embedding* of items from logged observations. Then, we initialize the prior of every item $j \in \mathcal{J}$ by setting the trait $\mathbf{v}_j$ and bias β_j from the corresponding offline posterior – assuming that all items in the online phase appeared in the offline data.

For the initialization of the user parameters, we focus on the case when the user is new to the system. Without additional information, we can assume that the new user is similar to the average offline user. We implement this by using as trait and bias the mean value over all offline users:

$$\mathbf{u}^{cold} \sim \mathbf{E}_{i=1,\dots,M}[\mathbf{u}_i] \qquad \alpha^{cold} \sim \mathbf{E}_{i=1,\dots,M}[\alpha_i] \qquad (5)$$

4.3.2 Question Selection Strategies

When a new user initiates interaction with a continuous recommender, the system asks a few questions to learn about the user's preferences. During this phase, it is important to select questions that lead to learning effectively (i) the user's preferences and (ii) the questions' quality, so that the number of questions asked can be minimized and interactions remain enjoyable. This task can be modeled as an item selection task. Here, we propose approaches that capture characteristics of *active learning* and *bandit learning*. Active learning approaches capture the intuition that learning is fastest when the system queries for labels that provide a high amount of new information [26]. Bandit learning approaches balance the need to learn new information with a focus on what has already been learned [3, 5]. In the context of conversational recommenders, such a balance may help focus questions on the most relevant part of the latent space, while still considering that highly preferred items may lie in as of yet unexplored areas of the space.

The model's confidence in its belief over the user's preferences on items $j \in \mathcal{J}$ at a given time is captured by the current variances of the posterior of the noisy affinities y_j^{cold}. As we ask about an item j^* and observe the user's feedback, the variance of the inferred noisy affinity of this item *and of the nearby items* in the learned embedding is reduced. Also, the means of these items' inferred noisy affinities change. It is this property that allows us to search the space of items quickly. Hence, while in the classic multi-armed bandit scenario the bandit algorithms converge only after all arms are sufficiently explored, in our setting the collaborative structure allows for faster convergence.[5]

[5]A formal regret analysis lies beyond the scope of this work.

Table 3: Question selection strategies evaluated.

Greedy: $j^* = \arg\max_j y_{ij}$
A trivial *exploit*-only strategy: Select the item with highest estimated affinity mean.

Random: $j^* = \text{random}(1,N)$
A trivial *explore*-only strategy.

Maximum Variance (MV): $j^* = \arg\max_j \epsilon_{ij}$
A *explore*-only strategy, variance reduction strategy: Select the item with the highest noisy affinity variance.

Maximum Item Trait (MaxT): $j^* = \arg\max_j \|\mathbf{v}_j\|_2$
Select the item whose trait vector $\mathbf{v}_j$ contains the most information, namely has highest L2 norm $\|\mathbf{v}_j\|_2 = \sqrt{v_{j1}^2 + v_{j2}^2 + \ldots + v_{jd}^2}$.

Minimum Item Trait (MinT): $j^* = \arg\min_j \|\mathbf{v}_j\|_2$
Select the item with trait vector with least information.

Upper Confidence (UCB): $j^* = \arg\max_j y_{ij} + \epsilon_{ij}$
Based on UCB1 [3]: Pick the item with the highest upper confidence bound, namely mean plus variance (95% CI)

Thompson Sampling (TS) [5]: $j^* = \arg\max_j \hat{y}_{ij}$
For each item, sample the noisy affinity from the posterior. Select item with the maximum sampled value.

We compare a number of approaches for question selection that reflect the considerations discussed above, and several baselines. All approaches are listed in Table 3. Each selects an item j^* to obtain user feedback on. While the first few are self-explanatory baselines, we discuss three in more detail. (1) MaxT approximates maximizing information gain, in the vein of active learning: As user-item ratings are observed, the PMF model adds information to the corresponding user and item trait vectors. In the opposite extreme case, if all item trait elements are close to 0, the corresponding item carries no information. As MinT does the opposite from MaxT, it is hypothesized to have low performance and is employed to establish a lower bound on question selection performance. (2) UCB [3] is a popular bandit algorithm that selects the items with the highest confidence bound to avoid missing preferences for promising items. (3) Thompson Sampling (TS) is a bandit algorithm that balances exploration and exploitation by selecting items using a sampling strategy [5]. It samples from its current posterior belief over noisy affinities, and then acts optimally according to this sampled belief. TS focuses on items with high mean affinity, but is also likely to select items with high variance.

4.3.3 Online Updating

After posing a question to the user, the observed response needs to be incorporated into to the recommender to allow for continued learning. As shown in [23], the questions can be incorporated into the model by setting the probability of the question to 1 and incorporating the user's response following standard probability theory.

The user's response thus becomes a new observation that the system uses to update the posterior distributions of *all* latent model parameters related to the incoming user i and the item j asked about, i.e., $\alpha_i, \beta_j, \mathbf{u}_i, \mathbf{v}_j$ (but affecting only user i's interaction session). Due to space constraints, we refer the reader to [29] for the specific Expectation Propagation updates. To select the next question for user i, we use the updated posteriors as the new priors to infer the user's noisy affinity distribution $\hat{y}_{ij}$ for all items $j \in \mathcal{J}$, denoted by $\hat{\mathbf{y}}_i$. As the system keeps asking questions to user i and incorporates his/her feedback, the beliefs about the user, and the item in the question, change. This allows the model to move towards the true underlying affinity distribution. All online updates were implemented in Infer.NET [20].

`Abs`: *Absolute Model, Absolute Questions*

So far we have presented the entire framework for the case where the system poses absolute questions. Before turning to relative feedback, we describe the high-level approach for asking absolute questions (`Abs`). Using TS for illustration purposes, `Abs` asks user i about the item with the largest sampled noisy affinity as inferred by the *Absolute* model:

$$j^* = \arg\max_{j \in \mathcal{J}} \hat{y}_{ij} \tag{6}$$

Based on whether the user (dis)liked item j^*, a new observation (i, j^*,1/0) is incorporated into the Absolute model.

4.4 Extension to Relative Preferences

An alternative is for the system to ask for a preference about a pair of items, i.e., does the user prefer item A (j^*) or item B (h^*)? Therefore, we present here the extension of our framework to the case of asking *relative questions.*

We consider three separate formulations (referred to as `Abs Pos`, `Abs Pos & Neg` and `Pairwise`) for selecting relative questions and incorporating feedback, to identify the best way of asking relative questions. Importantly, in every such formulation there are two choices: (i) the underlying model (*Absolute* vs. *Pairwise*) and (ii) how the user's response to the question is incorporated back into the model. For the first choice, `Abs Pos` and `Abs Pos & Neg` use the Absolute model, while `Pairwise` uses the Pairwise model. The second choice is applicable only for `Abs Pos` and `Abs Pos & Neg`, as they represent two ways of incorporating relative feedback into the absolute model.

4.4.1 Absolute Model, Relative Questions

First, we present `Abs Pos` and `Abs Pos & Neg`. Both use the same mechanism to generate the question "A vs. B" for user i:

1. Select item A as in `Abs` (Equation 6).
2. Virtual observation: Assume user i did not like A.
3. Virtual update: Incorporate the tuple (i, A, 0) into the *Absolute* model, infer the posteriors for all model parameters and set them as the virtual new prior.
4. Select item B, again according to `Abs`, but this time using the virtual prior as prior.

The insight behind this mechanism of constructing the relative question is that the two items the user is asked to give a relative preference on should be relatively far apart in the latent embedding, so that (i) the system can learn users' preferences effectively and (ii) the user is not forced to choose among very similar items. This diversity enforcing mechanism is inspired by the approach in [6].

The two methods introduced here differ only in the way the feedback is incorporated into the Absolute model. `Abs Pos` incorporates only positive information while `Abs Pos & Neg` incorporates both positive and negative information. For example, assume that the user preferred item B to A. Then, `Abs Pos` incorporates only the observation (i, B, 1), interpreting the relative preference on the preferred item B as a like for B. In contrast, `Abs Pos & Neg` incorporates two observations: (i, B, 1) for the preferred item and (i, A, 0) for the less preferred item. This can be seen as a variant of the "sparring" approach to dueling bandits [2], which samples item pairs and updates the model for both items as if an absolute reward were observed.

4.4.2 Pairwise Model, Relative Questions

The third method for selecting relative questions, `Pairwise`, uses the *Pairwise* model that directly takes pairwise preferences as input, to generate the relative question and incorporate the observations. Thus, the user's relative feedback is incorporated into the model without any intermediate transformation, i.e., as an observation (i, B, A) when B is preferred over A.

The `Pairwise` method picks A exactly as in `Abs`, and for item B, inspired by the dueling bandit approach in [34], it picks the item with the largest probability of being preferred to item A. We instantiate the latter by selecting the item with the maximum noisy difference from item A (j^*):

$$\text{item B} = h^* = \arg\max_{h \in \mathcal{J}} \hat{y}_{ihj^*} \tag{7}$$

For the selection of item B, any question selection strategy besides TS illustrated here (except for MinT, MaxT), can be employed exactly as in Table 3, with the difference that the noisy *difference* distribution should be used.

Incorporating the 'Neither' Option.

Preliminary experiments showed that when the method asks the user to give a relative preference on two items that he dislikes, forcing him/her to choose one could mislead the model about the user's preferences. Thus, we adjusted all methods so that in such a case the user can specify that he likes neither. We implemented this by (i) incorporating two dislikes in `Abs Pos & Neg` and (ii) omitting the update in `Abs Pos` and `Pairwise`.

5. EXPERIMENTAL SETUP

We now describe our overall empirical setup, used for the experiments described in the next two sections.

Setting. One main difficulty of evaluating conversational recommenders is that it requires the ability to have access to user reactions to any possible system action. We address this requirement using generative user models. The first user model is constructed synthetically and is used to validate our model (Section 6). The second is instantiated from real user preferences, collected via a user study (Section 7).

All experiments consist of an offline and an online phase. During the *offline phase*, the model is presented with data where M users interact with N items. In the subsequent *online phase*, the model is used to interact with cold-start users, asking questions using the pool of the offline N items.

We varied the number of questions from 0 to 15 and report the model's performance after each question. In practice, recommendations could be given after fewer questions, could be integrated with initial recommendations, or could be spread out over several interactions with a given user.

Research Questions. Our experiments are designed to address the following research questions:

RQ 1. Can our model adapt to the user's preferences?
RQ 2. Does our model learn effectively under either absolute or relative feedback?
RQ 3. Which relative question method performs better?
RQ 4. Is absolute or relative feedback more effective?
RQ 5. Does the offline initialization step help?
RQ 6. Which question selection strategy is more effective?

To answer each one of these questions, we need some measure of evaluating the effectiveness of our framework. Given that the goal of preference elicitation is a good recommendation list adhering to the user's preferences, we use *Average Precision@k* (*AP@k*) as our evaluation metric.

Metric. *AP@k* is a widely used, precision-oriented metric [15] for capturing accuracy in the top k (we set $k = 10$). Formally, for user i, we obtain the user's predicted recommendation list by sorting all items by decreasing mean of inferred noisy affinities $\mathbf{y}_i$. We evaluate this list by looking at the ground truth $\mathbf{r}_i^{\text{true}}$, i.e., capturing whether the user liked/disliked each item. *AP@k* is defined as the average of precisions computed at each liked position in the top k items of the user's ranked list. $P@\ell$ (*Precision@*ℓ) is the fraction of liked items out of the top $\ell + 1$ ranked items. Thus,

$$AP@k = \sum_{\ell=0}^{k-1} \frac{P@\ell \cdot r_{i[\ell]}^{\text{true}}}{\min(k, \#\text{ of liked items})} \tag{8}$$

where $[\ell]$ represents the index of the item present in rank ℓ, with $[\ell] = 0$ corresponding to the index of the top recommended item. Higher value (closer to 1) of *AP@k* implies better recommendation list. In our results, we report the average and 95% confidence intervals of AP@10 over all cold-start users. Results in additional metrics, such as ratio of correctly ranked pairs and mean reciprocal rank, were omitted as they showed similar trends as *AP*@10.

6. LEARNING SYNTHETIC USER TYPE PREFERENCES

We begin our experiments with an intentionally simplistic example of restaurant recommendation, as real world high-dimensional data is difficult to visualize. The example is meant to demonstrate concepts of our model and to illustrate the effectiveness of our approach to unlearn initial prior beliefs and tailor recommendations to specific user types, answering *RQ1* affirmatively.

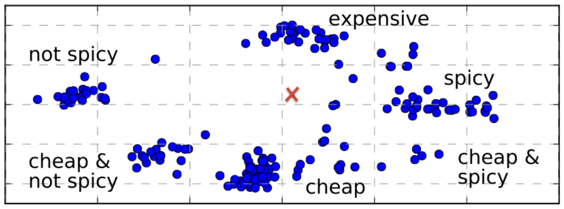
In our framework, we first use observations to learn an offline embedding for users and items in the same low-d space. Here, we generated the offline observations by considering types of users and restaurants as follows:

Restaurant types	%
expensive	15%
cheap & spicy	5%
cheap & not-spicy	10%
only cheap	35%
only not-spicy	15%
only spicy	20%

User types	%
Like expensive	20%
Like spicy	15%
Like not-spicy	25%
Like cheap	30%
Like only not-spicy	5%
Like only spicy	5%

We generated $N = 200$ restaurants, and $M = 200$ users. For each offline user, according to their type, we sampled 10 items from their liked category as likes and 10 items from the rest of the categories as dislikes. We used this intentionally simple synthetic setup to evaluate various parameter choices, and we show results for $\sigma_1^2 = 10$, $\sigma_2^2 = 0.5$, $\epsilon = 0.1$. To allow visualization, we considered only two latent traits (d=2) for this example. We see that in the learned embedding, the first trait indicates spiciness, while the second the price.

In the same space, an embedding for users is also learned (not shown to avoid clutter). The average trait vector over



all users is shown with a red cross. This becomes the initial trait vector for the cold-start user. Considering also the learned items' biases and the average user bias (not shown here), the system constructs an initial estimate of the noisy affinity distribution of the incoming user about all items.

Based on the offline observations, the learned prior for this affinity distribution favors user types which were popular in the offline data. The task of selecting restaurants for online users resembling the mean offline user is easy, as the prior already captures valuable information. As Fig. 1, *bottom right* panel shows, even with no questions, a close to perfect recommendation list can be given for *Liking not-spicy* users. Similar is the trend for the *Liking cheap* users (not shown).

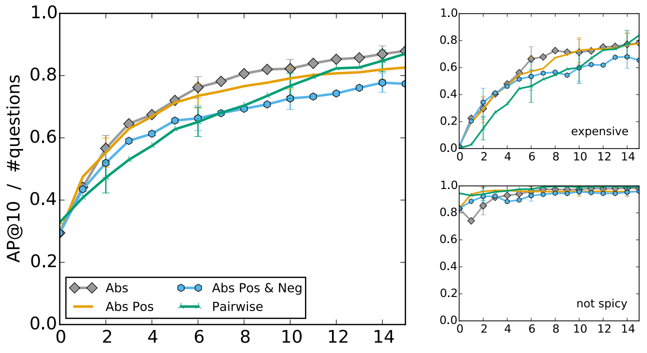


Figure 1: Results on synthetic restaurant data, across user types (*left*), and two of the user types (*right*).

In contrast, when the user is of a type that was rarely seen during the offline phase (e.g., expensive, shown in Fig. 1, *top right* panel), the online recommender has to collect observations that move away from more popular types. The trends for *Liking spicy*, *Liking only-spicy*, and *only not-spicy* user types are similar to the *Liking expensive*. For these types, the model (all four approaches) starts with AP@10 close to 0, but after every question asked, unlearns the initial wrong prior beliefs, and learns the specific user types' preferences.

For the results reported, we considered 60 cold-start users of each type and used TS for the question selection.

All methods learn effectively across all user types, with minor differences (Fig. 1, *left*) answering *RQ 2* positively.

7. RESULTS ON REAL DATA

Having positively answered *RQ 1* and *RQ 2* above, we turn our attention to a real-world setting to address all research questions in the context of 'where to dine in Cambridge, UK'. For the offline initialization of our framework, we use real users' online search behavior in a commercial search engine (Section 7.1). We use the insights from Section 3 to design a user study that is used to collect real restaurant preferences for Cambridge (Section 7.2). The collected responses serve as a basis for evaluating our online recommendation methods. We sketch a novel two-step approach to infer ratings for all restaurants, apart from those asked in the study (Section 7.3). We extensively evaluate our choices for the recommendation pipeline (Section 7.4).

7.1 Search Data for Offline Initialization

We start by describing the data which serve as our offline user-restaurants observations, based on which we learn the embedding used to warm start the question-asking phase.

This data is obtained by identifying restaurant review pages for restaurants in Cambridge (UK) on a major restaurant review service provider. Next, we filter the query and click logs from a major commercial search engine to find (anonymous) cookies that mark a single PC accessing a sequence of these restaurant pages. Each cookie is taken to be a distinct user, and all visits on restaurant review pages are considered to be indicating the user liking the restaurant[6].

In particular, taking logs from 26 March to 26 April 2015, we identified 3,549 cookies (users) who accessed at least one of the 512 distinct Cambridge restaurant review pages identified on the review service provider. This resulted in an index of Cambridge restaurants, each one visited by at least one user. Augmenting each of the restaurants with all known links and metadata associated with it in a proprietary restaurant index, and selecting a further three months back in each of those users' search histories, we recorded every interaction of these users with these restaurant links. During the four month search history of the users, we recorded interactions with 289 unique restaurants out of the 512 Cambridge restaurants. The total number of unique user – restaurant interactions recorded is 9330.

Thus, our offline data consists of $M = 3549$ users, $N = 289$ restaurants, and 9330 positive observations (1). To introduce dislikes (0) to the rating matrix as well, for every user i who has liked n_i^+ items, we sampled uniformly at random $n_i^- = \min(10, n_i^+)$ restaurants as dislikes.

Parameter Setting. To learn the offline embedding, we set the hyper-parameters to the combination that achieved the highest pairwise accuracy in the offline observations: $d = 4$ (i.e., 4 latent traits), $\sigma_1^2 = \sigma_2^2 = 10$, $\epsilon = 0.1$.

7.2 User Study as Basis for Online Evaluation

One of the issues of evaluating a conversational recommender is that one needs to know the user's ground truth on the space of all items (and questions). To obtain this, one needs to implement an interactive recommender asking questions to real users and receiving their responses. As an intermediate step, we conducted a user study and used the collected responses as ground truth for online users.

In the user study conducted, each participant filled in an anonymous web questionnaire about their preferences on a pool of restaurants in Cambridge, UK. The participants were asked "would you consider restaurant X for your next Friday night dinner?", labeling each restaurant with a binary label (yes/no). These responses comprise our ground truth.

For the pool of Cambridge restaurants we carefully selected ten restaurants, diverse in various features (as identified in Section 3). We recruited twenty eight individuals for the study. Given the anonymity of the questionnaire (in order to encourage truthfulness in the responses), demographic information was not recorded. However, the larger

[6]Though URL visitation is a weak positive signal, the experiments indicate it is a reasonable proxy for interest.

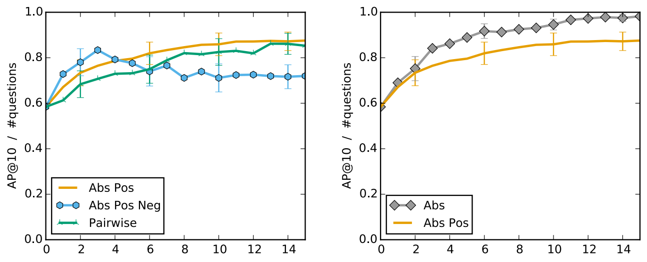


Figure 2: Differences between relative feedback models (*left*); and comparing absolute and relative feedback (*right*).

pool of individuals from which the participants were drawn (65 people working in a research lab) varied in factors such as age, job level, income, time spent in Cambridge.

Each participant was presented with the questionnaire about the same restaurants, but with varying order to avoid presentation bias. The participants were advised to visit the restaurant webpage when unfamiliar with the restaurant.

7.3 Obtaining Ground Truth

In our user study, we obtained restaurant labels for 10 out of the 289 Cambridge restaurants present in the offline data, for 28 participants. However, for reliable evaluation we need labels for the entire item space (rather than limiting our methods to ask about just these 10 restaurants). Therefore, we introduce an approach to fill in complete ground truth labels. Also, to increase the diversity of the user space, inspired by [17], we used bootstrapping to obtain 50 cold-start users based on the 28 participants' ground truth.

In particular, for each cold-start user:

1. Randomly sample one of the 28 participants.
2. Observe the sampled user's labels on the pool of 10 restaurants asked in the user study.
3. Infer user's traits $\mathbf{u}_i$ (prior= learned embedding in 7.1).
4. *Sample* $\hat{\mathbf{u}}_i \sim \mathbf{u}_i$. Set this to be the new prior of $\mathbf{u}_i$.
5. With this prior, infer the ratings $\mathbf{r}_i$ distribution.
6. *Sample* ratings from their distribution $\hat{\mathbf{r}}_i \sim \mathbf{r}_i$.

In this way, for each bootstrapped user we obtain a complete rating list for all 289 restaurants that is consistent with the user study labels of some user, yet is perturbed to account for variety in real user populations.

As far as we are aware, this approach for filling in the missing ratings is novel. It gives us ground truth as close to real as possible given the resources available. Alternatives would be exhaustive labels (not feasible for our study), or rejection/importance sampling (only effective when leveraging logged exploration data from large-scale systems, e.g [16]).

7.4 Results

Having obtained the offline embedding and the online users' ground truth for all items, we now present our experiments on a real restaurant recommendation scenario with the focus of answering the remaining research questions RQ 3 - RQ 6. Each of these questions investigates a separate component of our continuously learning recommender system.

Which method for relative questions is better? (RQ 3)

Recall that we proposed three approaches for modeling relative feedback. The first two incorporate feedback in an absolute model, (a) by incorporating information that the user liked the preferred item (ignoring the non-preferred one, `Abs Pos`), and (b) by incorporating both positive (preferred) and negative (non-preferred) information (`Abs Pos & Neg`). Alternatively, we construct a pairwise model (`Pairwise`). The results of the three methods' comparison are shown in the *left* panel of Figure 2. These results were obtained using TS for question selection and start from the offline embedding.

We see no significant difference among the methods during the first few questions. After only 2 questions, all methods significantly improve over the initial performance of .584 to respectively .734 (`Abs Pos`), .780 (`Abs Pos & Neg`), and .684 (`Pairwise`). However, as we ask more questions, `Abs Pos & Neg` forces negative observations on liked restaurants, thus causing the method to degrade the ranking. Overall, `Abs Pos` is the most effective method for relative questions.[7]

Are absolute or relative questions better? (RQ 4)

To answer this question, we compare the performance of the absolute-question asking method (`Abs`) with the best relative-question asking method (`Abs Pos`). The results are shown in the *right* panel of Figure 2. Until 2 questions, both methods have almost identical performance. But, after 5 questions, `Abs` performs significantly better than the relative feedback method, and achieves close to perfect performance after 15 questions ($AP@10 = .975$). We hypothesize that this result can be explained by the fact that our offline embedding favored absolute feedback.

Although our result shows that very high accuracy can be achieved when users provide accurate feedback on absolute questions, in practice this may not always be possible. Psychological effects such as anchor bias [12] can lead users to implicitly compare items, lowering the quality of absolute feedback. When this is the case, our result shows that high performance can be achieved with relative feedback as well.

Future work could involve hybrid models that automatically learn whether absolute or relative feedback, or a combination, is more accurate in a given setting.

Does offline initialization help? (RQ 5)

Next, we investigate the impact of model initialization, i.e., initializing the online recommender with an initial *offline embedding*, compared to starting from a generic prior (using the optimized hyper-parameters specified in 7.1). We present the results of this comparison in Figure 3 both for the absolute (`Abs`) and the best relative feedback model (`Abs Pos`), in the left and right panel correspondingly.

Our hypothesis is that the offline embedding learned from the weakly labeled data of a search log captures sufficient information to helpfully constrain continued learning, even if it does not exactly match the structure that underlies online users. Indeed, Figure 3 demonstrates great performance improvements when initializing the models from this embedding over generic prior initialization. By placing the new users as the average offline user, performance increases from .217 to .584, even without asking any questions. As the recommender collects more responses, performance continues to improve in both cases.

One observation is that the uninitialized system can ultimately achieve high performance, and for `Abs Pos (Prior)` even pass the offline initialized system. However, this is only achieved after many questions (here: 14). The phenomenon

[7]Studying the effect of alternatives for introducing dislikes in the offline data on `Abs Pos`'s success is left for future work.

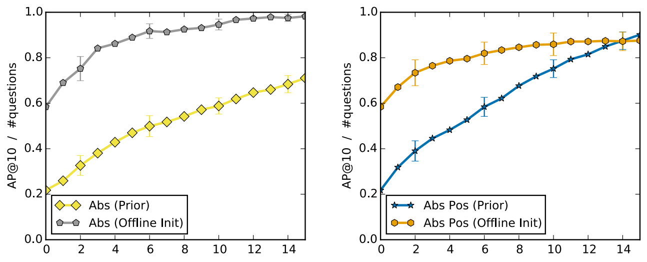


Figure 3: Impact of offline initialization on performance for absolute (`Abs`, *left*) and relative feedback (`Abs Pos`, *right*).

is nevertheless interesting, as it may point to a bias-variance trade-off. Starting from the generic prior allows the system to eventually perfectly fit a given user's preferences, however, learning takes a long time because there is no initial structure to constrain learning outcomes. We conclude that using offline initialization is highly beneficial.

Which question selection strategy is best? (RQ 6)

In Figure 4 we report the comparison results of the various question selection strategies of Section 4.3.2 (except `MaxV` whose performance almost coincides with `UCB`), for the `Abs` and `Abs Pos` methods initialized with the offline embedding.

For the `Abs` model (Figure 4, *top*), we observe that: (i) as expected, lowest AP@10 is achieved by `MinT`, (ii) `Random` learns slowly, likely because it fails to focus on more promising items for effective learning, and (iii) all remaining strategies perform equally well. We hypothesize that `Greedy` performs well thanks to the offline embedding, along with the online updating of all parameters after each response.

Turning to `Abs Pos` (Figure 4, *bottom*), the best performing strategies are those that encourage more diversity across the questions of the interactive session, namely the bandit-based ones and `Random`. `Greedy` and `MaxT` are the worst performing ones, following `MinT`. Our insight why this is the case is that they tend to select questions *A vs B*, followed by *A vs C*, etc. when A is preferred. Given that there is no construction encouraging B and C to be diverse (such as sampling or taking into account uncertainties), the questions focus on comparing parts of the embedding which are similar across questions, thus preventing truly effective learning.

Overall, we find that the bandit-inspired strategies perform the most robustly, achieving top performance across models. In the classic bandit setting, these approaches systematically balance the need to explore new solutions with the need to reap the rewards of what has already been learned. Here, we find that similar principles allow these strategies to collect user feedback that balances the need to not discard any areas of the latent space prematurely with the need to focus questions on the most promising areas. This novel insight is important because it shows that bandit-based question selection strategies can lead to benefits that go beyond the typical bandit problem.

Discussion. All our results show that our methods enable effective learning from interactions with their users, under either feedback type, answering positively *RQ 1* and *RQ 2*.

We show substantial recommendation performance improvements over the performance we would get without adapting to the user; by 25% after only 2 questions.

Although our underlying model can be augmented with external features [29], one key advantage is it does not need

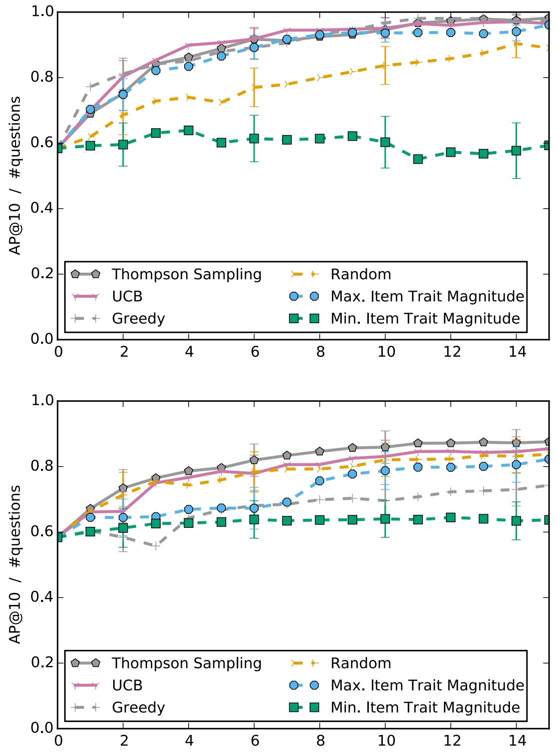


Figure 4: Performance of question selection strategies for absolute (`Abs`, *top*) and relative (`Abs Pos`, *bottom*) models.

them. It learns online both the user's and the items' latent traits. Together with [18] and [33], our findings corroborate the effectiveness of latent-feature interactive recommenders.

A novel insight is that taking into account the uncertainties in the learning of *both* the item and the user embedding, we can adapt to the specific user's preferences, under a certain context. This answers to the question posed by [33] and is key for allowing the system to learn both the user's profile and the questions' effectiveness in a contextual manner.

We have demonstrated effective learning from feedback present in many interactive settings. Thus, our approach is a good candidate for online learning across domains.

8. CONCLUSIONS

In this paper we proposed a novel view of recommendation as an interactive process. Like human recommenders, we envision recommender systems that can converse with new users to learn to know their preferences.

We develop such a conversational recommender, using restaurant recommendation as our motivating example. We start by examining restaurant related queries issued in a commercial search engine. Using these insights, we conducted a user study to elicit ground truth for evaluating our system. We propose a conversational recommender model that is theoretically well anchored in probabilistic matrix factorization models [21, 29]. We show how such models can be extended to support continuous learning. We empirically evaluate our approach using the ground truth based on real dining preferences elicited through our user study.

Our results have important implications for the development of conversational recommender systems. First, we demonstrated that best performance can be achieved with absolute questions. However, even in settings where only relative feedback is available, effective learning is possible. Second, we proposed a systematic approach to initializing conversational recommenders with an offline learned embedding, boosting greatly the performance even when only weakly supervised data is available. Third, we identified question selection strategies that can elicit feedback for very effective learning. Together, these insights pave the way towards conversational recommenders.

A promising future direction is to extend conversational recommenders to use reinforcement learning approaches, for capturing longer-term dependencies [17]. The modular structure of our framework allows various choices in a plug-and-play-manner, considering different feedback types, underlying probabilistic models etc., with the goal of building a suite of conversational recommenders for a variety of settings.

Acknowledgments. We would like to thank Tom Minka and Yordan Zaykov for the very insightful discussions on the model and evaluation. We also thank the participants of our user study.

9. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, 19–28, 2009.

[2] N. Ailon, Z. Karnin, and T. Joachims. Reducing dueling bandits to cardinal bandits. In *ICML*, 856–864, 2014.

[3] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. In *JMLR*, 3:397–422, 2003.

[4] G. Bresler, G. H. Chen, and D. Shah. A latent source model for online collaborative filtering. In *NIPS*, 3347–3355, 2014.

[5] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NIPS*, 2249–2257, 2011.

[6] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR*, 429–436, 2006.

[7] L. Chen and P. Pu. Critiquing-based recommenders: survey and emerging trends. In *User Modeling and User-Adapted Interaction*, 22(1-2):125–150, 2012.

[8] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas and P. N. Yianilos. The Bayesian image retrieval system, PicHunter: theory, implementation, and psychophysical experiments. In *Image Processing, IEEE transactions*, 9(1):20–37, 2000.

[9] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, 271–280, 2007.

[10] A. Felfernig, G. Friedrich, D. Jannach and M. Zanker. Developing Constraint-based Recommenders. In *Recommender systems handbook*, 187–212, 2011.

[11] M. P. Graus and M. C. Willemsen. Improving the user experience during cold start through choice-based preference elicitation. In *RecSys*, 273–276, 2015.

[12] D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. In *Econometrica*, 263–291, 1979.

[13] J. Kawale, H. H Bui, B. Kveton, L. Tran-Thanh, and S. Chawla. Efficient Thompson Sampling for Online Matrix-Factorization Recommendation. In *NIPS*, 1297–1305, 2015.

[14] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. In *Computer*, (8):30–37, 2009.

[15] H. Li. A short introduction to learning to rank. In *IEICE TIS*, 94(10):1854–1862, 2011.

[16] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 661–670, 2010.

[17] E. Liebman, M. Saar-Tsechansky, and P. Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. In *AAMAS*, 591–599, 2015.

[18] B. Loepp, T. Hussein, and J. Ziegler. Choice-based preference elicitation for collaborative filtering recommender systems. In *CHI*, 3085–3094, 2014.

[19] T. Mahmood and F. Ricci. Improving recommender systems with adaptive conversational strategies. In *Hypertext*, 73–82, 2009.

[20] T. Minka, J. Winn, J. Guiver, and D. Knowles. Infer .net 2.5. *Microsoft Research Cambridge*, 2012.

[21] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, 1257–1264, 2007.

[22] J. Neidhardt, R. Schuster, L. Seyfang, and H. Werthner. Eliciting the users' unknown preferences. In *RecSys*, 309–312, 2014.

[23] P. A. Ortega and D. A. Braun. Generalized thompson sampling for sequential decision-making and causal inference. In *Complex Adaptive Systems Modeling*, 2(1):2, 2014.

[24] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, 325–341, 2007.

[25] J. Ritchie and J. Lewis. Qualitative research practice. In *Sage*, 2003.

[26] N. Rubens, D. Kaplan, and M. Sugiyama. Active learning in recommender systems. In *Recommender systems handbook*, 735–767, 2011.

[27] T. Salimans, U. Paquet, and T. Graepel. Collaborative learning of preference rankings. In *RecSys*, 261–264, 2012.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 285–295, 2001.

[29] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*, 111–120, 2009.

[30] M. Sun, F. Li, J. Lee, K. Zhou, G. Lebanon, and H. Zha. Learning multiple-question decision trees for cold-start recommendation. In *WSDM*, 445–454, 2013.

[31] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li. Personalized recommendation via parameter-free contextual bandits. In *SIGIR*, 323–332, 2015.

[32] H. P. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *RecSys*, 225–232, 2014.

[33] X. Zhao, W. Zhang, and J. Wang. Interactive collaborative filtering. In *CIKM*, 1411–1420, 2013.

[34] M. Zoghi, S. A. Whiteson, M. de Rijke, and R. Munos. Relative confidence sampling for efficient on-line ranker evaluation. In *WSDM*, 73–82, 2014.