Minimal Interaction Content Discovery in Recommender Systems

BRANISLAV KVETON, Adobe Research
SHLOMO BERKOVSKY, CSIRO

Many prior works in recommender systems focus on improving the accuracy of item rating predictions. In comparison, the areas of recommendation interfaces and user-recommender interaction remain underexplored. In this work, we look into the interaction of users with the recommendation list, aiming to devise a method that simplifies content discovery and minimizes the cost of reaching an item of interest. We quantify this cost by the number of user interactions (clicks and scrolls) with the recommendation list. To this end, we propose generalized linear search (GLS), an adaptive combination of the established linear and generalized search (GS) approaches. GLS leverages the advantages of these two approaches, and we prove formally that it performs at least as well as GS. We also conduct a thorough experimental evaluation of GLS and compare it to several baselines and heuristic approaches in both an offline and live evaluation. The results of the evaluation show that GLS consistently outperforms the baseline approaches and is also preferred by users. In summary, GLS offers an efficient and easy-to-use means for content discovery in recommender systems.

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI)**; **Interaction paradigms**; **Interactive systems and tools**; • **Theory of computation → Active learning**;

Additional Key Words and Phrases: Recommender systems, content discovery, user-recommender interaction, generalized linear search

ACM Reference Format:
Branislav Kveton and Shlomo Berkovsky. 2016. Minimal interaction content discovery in recommender systems. ACM Trans. Interact. Intell. Syst. 6, 2, Article 15 (July 2016), 25 pages.
DOI: http://dx.doi.org/10.1145/2845090

1. INTRODUCTION

Recommender systems are used in a variety of eCommerce sites and social networks. They typically act as decision support tools, helping users to cope with the overwhelming volume of choices, options, and services. To achieve this, recommenders select a set of best matching items—we denote this set as the recommendation list—out of a large set of candidate items. The selection process capitalizes on the available user data, which communicates the preferences and needs of the target user, and predicts the rating that the user would have assigned to the candidate items. These predicted ratings reflect the match of the candidate items to the target user and are the basis for selecting the items in the recommendation list.

Although many prior works targeted the development of algorithms for improving the accuracy of rating predictions and item selection [Ricci et al. 2011], relatively little research has investigated the presentation of recommendations to users [Felfernig et al. 2012], which is an important aspect of user-recommender interaction. The most popular

Authors' addresses: B. Kveton, Adobe Research, 321 Park Avenue, San Jose, CA; email: kveton@adobe.com; S. Berkovsky, CSIRO, Vimiera Road, Marsfield, NSW, Australia; email: shlomo.berkovsky@csiro.au.

DOI: http://dx.doi.org/10.1145/2845090

presentation of the recommendations is through a list of items, ordered according to their predicted rating. Sometimes the items are accompanied by thumbnails or other contextual information [Sharmin et al. 2012], or grouped according to some predefined criteria [Chen and Pu 2010], but the choice of the item to consume is done by the user on examination of the recommendation list.

Normally, either a fixed number of N items with the highest predicted ratings or all items above a threshold T are included in the recommendation list. However, for a sufficiently large recommendation list, the examination of the list and the selection of the item to consume can become a tedious task. To simplify the task and adjust to practical constraints of the system interface, the recommendation list is often split into small-size batches—for example, 10 items per screen for a desktop interface, and 3 to 4 items for a mobile app. Consider on the top of the list examination challenge the difficulty of selecting one item out of a set of potentially appealing recommendations, and the presentation of the recommendation list becomes an important consideration in achieving a good user experience in recommender systems [Knijnenburg et al. 2012].

In this work, we focus on minimizing the cost of item-selection user interactions with the recommender. In our context, the term *cost* embraces both the number of interactions, such as clicks, taps, or scrolls, and the duration of the recommendation list exploration. Thus, we generally refer to the minimization problem as minimal interaction search. We assume that the predicted item ratings are computed by existing recommendation techniques and leave the algorithmic side of rating computation beyond the scope of our work. Given the predicted ratings, we focus on the recommendation list of N top-scoring items, assuming that the item to be consumed by the user—the target item—is necessarily included in the list. In other words, we assume that at least one item in the recommendation list satisfies the needs of the user. Notwithstanding, we consider a situation in which the user may not know in advance the exact item that he or she is looking for and therefore cannot use the conventional textual search functionality. However, the user may be able to describe the features of the item of interest and answer questions about the item. A typical use case for such a search is “I would like to watch a movie like that Irish black comedy that I watched last week,” which for obvious reasons cannot be addressed by search techniques.

The simplest approach of searching for the target item in the recommendation list is linear search (LS). This assumes that the recommended items are ranked according to some criterion, such as alphabetically or according to their predicted rating, and the user scrolls through batches of items until the target item is discovered. An inherent limitation of LS is that it cannot incorporate any user feedback that may refine the search. For instance, the user may want to inform the recommender that she is not interested in watching comedies at the moment, which may eliminate a portion of the recommended items from the list and decrease the cost of the search. Thus, an immediate generalization of LS is linear search with item categories ($\mathtt{LS}_{\mathtt{cat}}$), which capitalizes on the availability of item categories. Here, the user is first shown a list of item categories and asked to choose the category that best matches the target item. Then the user proceeds with LS in a smaller list of recommended items that belong to the selected category.

However, finding the target item with $\mathtt{LS}_{\mathtt{cat}}$ may still require many user-recommender interactions due to the poor efficiency of LS following the category question. This observation naturally raises the question of iteratively repeating the category selection step. In other words, the user may be able to answer multiple questions and conduct a hierarchical search over a set of dynamically computed item categories. This search can be viewed as a question-answering game [Winterboer et al. 2011], where the recommended items are split at each step into K item categories that are presented to the user. The user selects the category that matches the target item. This question-answering game

Interaction 1 of GLS 8 categories shown (GS)	Interaction 2 of GLS 8 categories shown (GS)	Interaction 3 of GLS 8 categories shown (GS)	Interaction 4 of GLS 8 movies shown (LS)
Action Thrillers	Romance Classics	Contemporary Literature	Misery (1990)
Foreign Regions	Mobster	Crime Thrillers	**Primal Fear (1996)**
Ages 11-12	**Thrillers**	Mystery	General's Daughter (1999)
20th Century Period	Based on Bestsellers	Courtroom Dramas	Silence of the Lambs (1991)
Based on the Book	Crime Dramas	Sci-Fi Horror	Dolores Claiborne (1994)
Classics	Drama	Crime Dramas	Pelican Brief (1993)
Drama	Classic Dramas	**Based on Bestsellers**	Boys from Brazil (1978)
Something Else	Something Else	Something Else	Instinct (1999)

Fig. 1. Searching for a movie like *Silence of the Lambs* (1991) with GLS. The choices of the user are highlighted in bold.

continues until only a single target item is consistent with the answers to all asked questions. This form of a search is known as generalized search (GS) [Dasgupta 2005; Golovin and Krause 2011; Nowak 2011]. In large-scale problems, GS is expected to perform significantly better than both LS and LS_{cat} [Bhamidipati et al. 2013].

We extend the work of Kveton and Berkovsky [2015] and propose generalized linear search (GLS), a combination of GS and LS that leverages the benefits of both of these approaches. Since GS is a hierarchical search, given a recommendation list of N items, it may converge to the target item in $O(\log(N))$ interactions if the item categories are sufficiently diverse. However, in the worst case when each item belongs to its own category, GS needs $\Omega(N)$ interactions to discover the target item. On the contrary, LS is independent of the item categories and therefore is more appropriate for situations where some items are more likely to be selected by the user than others. Thus, we combine GS and LS, and propose a hybrid search strategy that starts with a number of question-answering iterations of GS and then adaptively switches to LS when GS becomes suboptimal. The pivotal question in the hybrid GLS is when to switch from GS to LS. We argue that GS should be applied as long as the *expected* cost of its user interactions is smaller than that of LS interactions, and the switch should occur afterward.

We illustrate GLS in Figure 1 in an example of searching for a movie like *Silence of the Lambs*. In the first interaction, GS suggests eight categories and the user selects the best matching category, "Based on the Book." Following this, the search space is reduced to the movies that belong to the Based on the Book category. In the second interaction, the expected cost of GS is smaller than that of LS. Therefore, GLS proceeds with another GS question and suggests another eight categories. The user selects "Thrillers." The third interaction is still GS, and the user selects "Based on Bestsellers." At this stage, the expected cost of GS is larger than that of LS. Therefore, GLS switches to LS, and in the fourth interaction the user is shown eight movies rather than eight categories. Note that the shown movies are consistent with the user's answers to the GS questions and are all Based on the Book, Thrillers, and Based on Bestsellers. The user finally selects "Primal Fear (1996)" as the target movie. Assuming that all interactions with GLS are of unit cost, the cost of the preceding search is four.

In this work, we formally prove that GLS performs at least as well as GS and compare GLS empirically with LS, GS, and several heuristics. We conduct offline evaluations of the expected cost of user-recommender interactions using two publicly available datasets from the movie and music domains, and show that GLS achieves smaller interaction costs than LS and GS individually. We also show that the cost of GLS interactions is never larger than the cost of the evaluated heuristics. The performance of the heuristics requires an a priori parameter tuning, whereas GLS is parameter free. We also conduct several live user studies that compare the binary variants of GLS, LS, and GS. The studies clearly show that GLS achieves shorter user-recommender interactions and is perceived as easy to use by our users. The users also indicate their preference toward

GLS when compared to LS and GS. Finally, we conduct a study of a tuned version of GLS that differentiates between the costs of LS and GS interactions. However, this study does not demonstrate significant differences with respect to the original variant of GLS.

To summarize, GLS successfully combines two search methods—GS and LS—in a provably optimal and efficient way. The presented combination has several notable properties. First, GLS is guaranteed to perform at least as well as each of its components (GS and LS), and we prove this. Second, GLS does not have tunable parameters and can be applied to a wide range of online systems and dynamic application domains. Third, GLS significantly outperforms GS and LS. We report performance improvements as large as 50% both in offline experiments and live user studies.

2. PRELIMINARIES

In this section, we present two policies—LS and $\texttt{LS}_{\texttt{cat}}$—for searching in a list of recommended items. We apply these policies to a movie recommendation problem and use the findings to motivate the rest of the work.

Before we proceed, let us formalize the problem of minimal interaction search. We assume that the user has a target item e^* in mind, which necessarily belongs to a set of N recommended items $E = \{1, \dots, N\}$. The user does not know the exact target item and therefore cannot search for the item using conventional textual search techniques. However, the user knows the desired features of the target item and can eliminate other items that do not have these features. The target item e^* is drawn from distribution π over items in E. The distribution π is generated by the recommender system using one of the existing rating prediction techniques and is known prior to user interaction with the recommender. However, the exact target item e^* is unknown. The goal of our work is to devise a method that presents items E to the user such that the cost of discovering target item e^* is minimized. As stated earlier, we quantify the cost through the number and duration of user-recommender interactions, such as clicks, scrolls, and questions. Let A be a method for searching items in E and $N_A(e^*)$ be the cost of interactions to discover the target item e^*. Then our goal is to devise a method that minimizes $\mathbb{E}_{e\sim\pi}[N_A(e)]$.

2.1. Linear Search

LS is a prevalent form of search in many commercial recommender systems. In LS, the recommended items in E are sorted according to a predefined criterion—for example, in a descending order of their predicted ratings. The items are then iteratively scanned by the user, usually in batches of L items, until the target item e^* is discovered. For the sake of concreteness and without loss of generality, we assume that the value of L is dictated by the number of items that can be displayed on the screen of a user device that shows the recommendations. Therefore, the cost of interaction to find item e^* is the number of screens that the user scans until e^* is discovered. The pseudocode of LS is shown in Algorithm 1.

We present a simulationcomparing three LS policies that differ only in the computation of the predicted item ratings. The first policy scores items by their number of ratings. We refer to this policy as *popularity*. The second policy scores items by their expected rating, which is computed by averaging the ratings of the item across all users. We refer to this policy as *expected rating*. In the last policy, the score of the item is the rating assigned by the user. The unrated items are assigned with a score of zero. This policy is unrealistic and cannot be implemented in practice, as the ratings of recommended items are unknown. However, the policy can be viewed as a lower bound on the cost of interaction in any LS policy, under the assumptions that the true ratings of items are known. Therefore, we refer to this policy as *perfect model*.

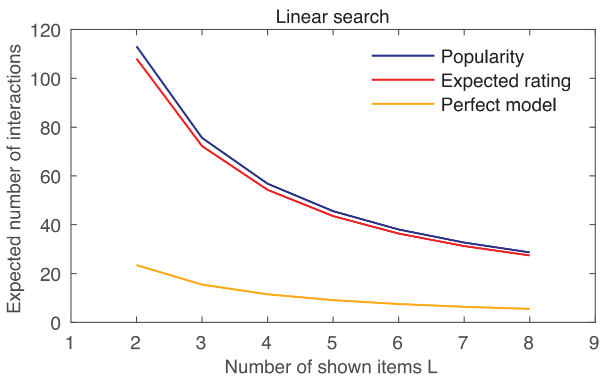


Fig. 2. Expected cost of three LS policies.

```
ALGORITHM 1: LS: Linear Search
Input:
  Items $V \subseteq E$
  Number of shown items $L$

Let $e_1, \ldots, e_{|V|}$ be an ordering of items $V$ such that:
  $\pi(e_1) \geq \ldots \geq \pi(e_{|V|})$
$\ell \leftarrow 0$
repeat
    $A \leftarrow \{e_i \in V : \ell L < i \leq (\ell + 1)L\}$
    Show items $A$ to the user
    $\ell \leftarrow \ell + 1$
until $(e^* \in A)$

Output: Target item $e^* \in A$
```

We evaluate the expected length of interactions on the MovieLens dataset, which is described in Section 4.1. The expected length of interaction is computed as follows. First, we randomly choose a user, proportionally to the number of positively rated items by the user. A user assigns a positive rating to item e when the user's rating for movie e is four stars or more. Second, we randomly choose a target item e^* from these positively rated items. All movies are scored according to the chosen policy: popularity, expected rating, or perfect model. Finally, we rank the movies according to their scores and simulate LS until the target item e^* is discovered. The expected length of interaction is estimated from 10^8 randomly chosen user-item pairs. We vary the number of shown items L from two to eight. The results of the simulation are reported in Figure 2.

We observe two trends. First, the length of interaction in all three policies decreases with the number of displayed items L. This trend is expected; as L increases and more items are shown in each batch, the expected length of interaction to find the target item should not increase. Second, we observe that the way of computing the item ratings has a strong impact on the expected length of interaction. In particular, the perfect model policy consistently outperforms the expected rating policy, which outperforms the popularity policy. Interestingly, the popularity policy performs only slightly worse than the expected rating policy, whereas the perfect model policy significantly outperforms the other two policies. On average, the perfect model policy discovers the target item in only about 20% of the interactions of the worst-performing popularity policy.

2.2. Linear Search with Item Categories

$\texttt{LS}_{\texttt{cat}}$ is a variant of LS that exploits additional information—the features of items. We leverage the features as follows. First, the user is asked to select the category that best matches the target item e^* out of K shown item categories. Following this category selection, all items that do not belong to the selected category are eliminated, whereas the items in the selected category are searched using LS. It should be highlighted that $\texttt{LS}_{\texttt{cat}}$ involves two types of user interaction. The first interaction is a one-off category selection. The remaining interactions are identical to LS, and the recommended items are scanned until the target item e^* is discovered.

$\texttt{LS}_{\texttt{cat}}$ is parameterized by the number of item categories K that are shown to the user in the first interaction. We represent each category as a set of items $q \subseteq E$ such that $e \in q$ if and only if item e belongs to category q. The problem of choosing K item categories that are shown to the user is an optimization problem. In an ideal world, these K categories would partition the set of items E into K disjoint sets of equal cardinality N/K each. In this case, the user's selection would reduce the set E to a set of N/K items, regardless of the selected category. However, such a partitioning is typically impossible in practice, as the item categories are not completely disjoint and the items are not distributed uniformly across the categories.

We partition the set E into K item categories by the greedy strategy of Bhamidipati et al. [2013]. For a given K, we greedily select $K - 1$ item categories $q_1, \dots, q_{K-1} \subseteq E$ whose cardinalities are closest to N/K. In addition, we define a special category $q_K \leftarrow \overline{(q_1 \cup \dots \cup q_{K-1})}$ that includes all items not belonging to any of the first $K-1$ categories. The key advantage of this strategy is that any item in E is guaranteed to belong to at least one category q_k out of K. Therefore, the user can always select a category that matches the target item e^*, or q_K when the target item does not belong to any of the first $K - 1$ categories. When a category is chosen, the set of candidate items is reduced to $q_k \cap E$ and we apply LS. The pseudocode of $\texttt{LS}_{\texttt{cat}}$ is shown in Algorithm 2.

ALGORITHM 2: $\texttt{LS}_{\texttt{cat}}$: Linear Search with Item Categories

```
Input:
  Number of item categories $K$ in a question
  Number of shown items $L$

Choose $K - 1$ categories $q_1, \dots, q_{K-1}$ given $E$ and $\pi$
$q_K \leftarrow \overline{(q_1 \cup \dots \cup q_{K-1})}$
Ask the user to choose one category
The user chooses category $q_k$
$e^* \leftarrow \texttt{LS}(q_k \cap E, L)$

Output: Target item $e^*$
```

We evaluate $\texttt{LS}_{\texttt{cat}}$ using the same methodology as in Section 2.1. We simulate $K = 2$, $K = 4$, and $K = 8$ item categories shown to the user and compare $\texttt{LS}_{\texttt{cat}}$ with these values of K to the popularity and perfect model policies of LS. The results of the simulation are reported in Figure 3. We observe that splitting items into K categories and then performing LS substantially reduces the cost of user interactions. A split into $K = 2$ categories decreases the length of interaction by about 30% for any number of shown items L. Notably, for $K = 8$, the expected cost of $\texttt{LS}_{\texttt{cat}}$ interaction is comparable to, and in fact slightly smaller than, the cost achieved by the perfect model LS policy. In other words, a single piece of information provided by the user—selection of one

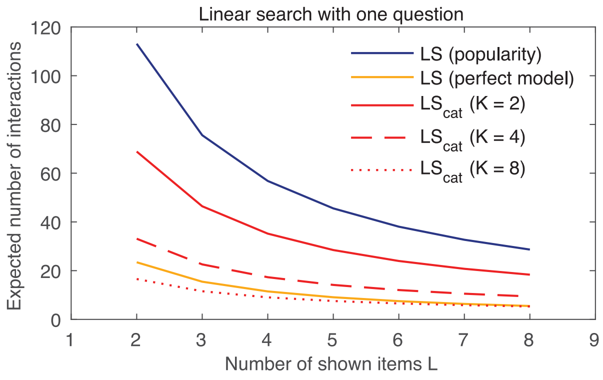


Fig. 3. Expected cost of $\texttt{LS}_{\texttt{cat}}$.

item category out of $K = 8$—reduces the expected cost of interactions as much as the unrealistic knowledge of the exact ratings of the recommended items.

3. MINIMAL INTERACTION SEARCH

We now present GS and our main contribution, GLS.

3.1. Generalized Search

Generalized binary search (GBS) [Dasgupta 2005; Golovin and Krause 2011; Nowak 2011] is a greedy algorithm for active learning, in which the goal is to identify an item of interest e^* by asking a sequence of binary questions. The question-answering policy that minimizes the expected number of asked questions is NP-hard to compute. GBS is its computationally efficient approximation. To optimize the search, GBS greedily asks questions that partition the *search space* $V \subseteq E$, the set of items that are consistent with the user's answers to up to that point, most evenly in expectation. A remarkable property of GBS is that it asks, in expectation, at most $\log(1/\pi_{\min})$ times more questions than the optimal search policy, where $\pi_{\min} = \min_{e \in E} \pi(e)$ is the probability of choosing the least likely item.

In this work, we consider GS, a generalization of GBS from binary to multiway questions [Bhamidipati et al. 2013]. The search for the target item e^* proceeds as follows. The user is presented with K item categories $q_1, \ldots, q_K \subseteq E$ and asked to choose the category that best matches e^*. As in $\texttt{LS}_{\texttt{cat}}$, these are $K - 1$ item categories and a special category of items that do not belong to any of the $K - 1$ categories. After the user chooses a category q_k, the search space V is reduced to the items in this category, $q_k \cap V$, whereas the remaining items get eliminated. Then the user is presented again with K categories, which may and typically do depend on the previously chosen categories. The user chooses again one category out of K, and this question-answering game continues until the cardinality of the search space V is reduced to one, in which case the target item e^* is discovered.[1] Each round of this game is considered as an interaction. The pseudocode of GS is shown in Algorithm 3.

GS can search high-dimensional spaces efficiently when item categories permit relatively uniform partitioning of the search space. This is not always possible. For instance, consider a recommendation problem with $N = 100$ items, where the probability that the user selects the most likely item is $\pi(e_1) = 0.99$, but this item cannot be separated

[1]If the user gives a wrong answer to one of the questions, GS may terminate with an item different from the target item e^*. In the analysis, we assume that the user answers all questions correctly, but we will address this practical consideration in the user study reported in Section 5.4.

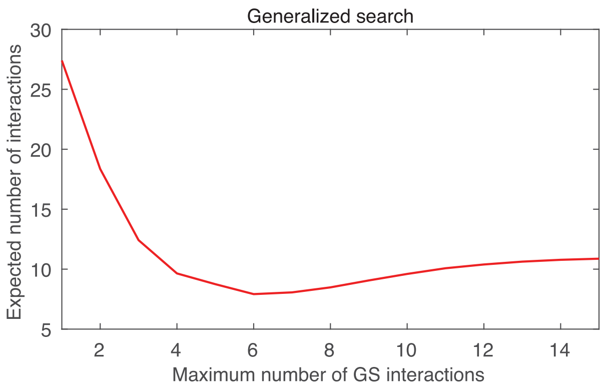


Fig. 4. Expected cost as a function of the number of GS questions.

ALGORITHM 3: GS: Generalized Search

```
Input:
  Number of item categories $K$ in a question

$V \leftarrow E$
repeat
    Choose $K - 1$ categories $q_1, \ldots, q_{K-1}$ given $V$ and $\pi$
    $q_K \leftarrow \overline{(q_1 \cup \ldots \cup q_{K-1})}$
    Ask the user to choose one category
    The user chooses category $q_k$
    $V \leftarrow q_k \cap V$
until $(|V| = 1)$

Output: Target item $e^* \in V$
```

from the other recommended items in a single question. In this case, LS can discover the target item in a single interaction with at least 0.99 probability because the highest-ranked item is the target item with 0.99 probability. On the other hand, GS is much less efficient than LS, as it needs to ask the user multiple questions in expectation to eliminate all but the most likely item.

In Section 2.2, we showed that a single question in LS_{cat} can gain as much information about the target item e^* as knowing the exact item ratings. A natural question to ask is, what if the user is asked to select a category more than once? Will the expected cost of interactions further decrease? To answer this question, we simulate a modified variant of GS with $K = 2$ item categories. We ask a fixed number of GS questions and switch to LS either after this number of questions or when the cardinality of the search space is one. The methodology for choosing users and items is identical to that in our earlier simulations. The cost of interaction in our proposed hybrid search is the sum of the costs of GS and LS interactions. The results of the simulation are reported in Figure 4.

Interestingly, we observe that the expected cost of interaction does not always decrease with the number of asked GS questions. More precisely, it decreases when the number of questions increases from 1 to 6 but grows after GS asks more than 6 questions. The expected cost at 6 questions is smaller by one than the cost at 9 questions, although the cost stabilizes at 10 interactions. We posit that this behavior can be explained by the observation that asking questions is beneficial only if the search space can be partitioned as uniformly as possible into K item categories. If this cannot be

ALGORITHM 4: GLS: Generalized Linear Search

```
Input:
  Number of item categories $K$ in a question
  Number of shown items $L$

$V \leftarrow E$
repeat
    Choose $K - 1$ categories $q_1, \ldots, q_{K-1}$ given $V$ and $\pi$
    $q_K \leftarrow \overline{(q_1 \cup \ldots \cup q_{K-1})}$
    Ask the user to choose one category
    The user chooses category $k$
    $V \leftarrow q_k \cap V$
until $((\texttt{costGS}(V, K) \geq \texttt{costLS}(V, L)) \vee (|V| = 1))$

Let $e_1, \ldots, e_{|V|}$ be an ordering of items $V$ such that:
  $\pi(e_1) \geq \ldots \geq \pi(e_{|V|})$
$\ell \leftarrow 0$
repeat
    $A \leftarrow \{e_i \in V : \ell L < i \leq (\ell + 1)L\}$
    Show items $A$ to the user
    $\ell \leftarrow \ell + 1$
until $(e^* \in A)$

Output: Target item $e^*$
```

done, it may be preferable to order the items in the search space according to their popularity, from the most popular to the least, and ask the user to discover the target item by iteratively scanning batches of items.

3.2. Generalized Linear Search

This observation motivates our work on hybridizing GS with LS in a way that adaptively leverages the strengths of both approaches and minimizes the cost of user interaction with the recommender. Specifically, we propose using GS up to the point where it is not efficient, as the search space V cannot be partitioned uniformly, and then switch to LS, which does not rely on the features of the items in the search space. The pivotal question in this hybridization is how to choose the sweet spot where GS should switch to LS. One suitable switching criterion may be when the number of items in the search space V is small. Another criterion may be when the probability mass in V is small. There are many ways of hybridizing GS and LS, and it is not clear which one should be applied.

We propose GLS, a novel hybrid method. In a nutshell, GLS switches from GS to LS when the *expected cost of interaction* to discover the target item by LS becomes smaller than that of continuing with GS. We choose this switching oracle for several reasons. First, this criterion reflects the fact that GS generally outperforms LS in minimizing the expected cost of interaction. Second, the oracle can be implemented efficiently through estimating the expected costs of GS and LS in any given search space. Third, we can incorporate the oracle in the theoretical analysis of GLS and show that the expected cost of interaction in GLS is never larger than that in GS. In addition to these, we show in our experiments (Sections 4 and 5) that this approach performs well in practice.

Without loss of generality, we assume that the cost of scanning L items in LS is $\omega_{\texttt{LS}}(L)$ and that the cost of answering one question with K categories in GS is $\omega_{\texttt{GS}}(K)$. To reduce the number of tunable parameters, we divide both costs by $\omega_{\texttt{LS}}(L)$. Following this, the cost of one LS interaction is one and the cost of one GS question is $\gamma_q = \omega_{\texttt{GS}}(K)/\omega_{\texttt{LS}}(L)$.

ALGORITHM 5: `costLS`: Expected Cost of Linear Search

```
Input:
  Items $V \subseteq E$
  Number of shown items $L$

Let $e_1, \dots, e_{|V|}$ be an ordering of items $V$ such that:
  $\pi(e_1) \geq \dots \geq \pi(e_{|V|})$
$c \leftarrow 0$
for all $i = 1, \dots, |V|$ do
    $c \leftarrow c + \lceil i/L \rceil \, \pi(e_i)$
end for

Output: Expected cost $c$
```

ALGORITHM 6: `costGS`: Expected Cost of Generalized Search

```
Input:
  Items $V \subseteq E$
  Number of item categories $K$ in a question

$c \leftarrow 1$
if $(|V| > 1)$ then
    Choose $K - 1$ categories $q_1, \dots, q_{K-1}$ given $V$ and $\pi$
    $q_K \leftarrow \overline{(q_1 \cup \dots \cup q_{K-1})}$
    for all $k = 1, \dots, K$ do
      $c \leftarrow c + \pi(q_k \cap V) \times \texttt{costGS}(q_k \cap V, K)$
    end for
end if
$c \leftarrow \gamma_q c$

Output: Expected cost $c$
```

The parameter γ_q is the cost of a GS interaction, answering a question with K categories, relatively to the cost of an LS interaction, scanning a list of L items. When $\gamma_q = 0$, GLS behaves as GS because the cost of GS interactions is zero. When $\gamma_q = \infty$, GLS behaves as LS because the cost of GS interactions is ∞. More generally, when $\gamma_q > 1$, GS questions are perceived to be more costly than scanning LS lists and vice versa. In Section 5.4, we show a practical and data-driven way to set the value of γ_q.

The pseudocode of GLS is shown in Algorithm 4. GLS combines LS and GS sequentially. In particular, the first loop in Algorithm 4 corresponds to GS and the second one to LS. GS continues until its expected cost is larger than that of LS and then stops. The remaining items in the search space V are ordered according to their consumption probability and we proceed with LS. The pseudocode for computing the expected costs of LS and GS is shown in Algorithms 5 and 6, respectively. Since we normalize the costs of individual GS and LS interactions, only the former are weighted by parameter γ_q in Algorithm 6.

As mentioned earlier, GLS can be implemented computationally efficiently. In particular, note that GS can be represented as a decision tree [Golovin and Krause 2011], where each node is a search space associated with K item categories. Therefore, GLS can be considered as a pruned version of this tree, where each leaf node is associated with the follow-up instance of LS in the respective search space V. The pruned tree can be constructed as follows. First, we build the tree for GS and recursively compute the expected cost of GS in each node. Second, we compute the expected cost of LS in each node and prune the tree in each node where costGS$(V, K) \geq$ costLS(V, L). Additionally note that the expected cost of LS in each node can be computed in $O(N \log N)$ time because it requires sorting of at most N items.

3.3. Analysis of GLS

Let π be a distribution over items E and $\pi(e)$ be the probability that item e is chosen. Let $V \subseteq E$ be a search space and $\mathcal{V}$ be the set of search spaces where GLS switches from GS to LS, costGS$(V, K) \geq$ costLS(V, L). Then $\mathcal{V}$ is a partitioning of E:

$$E = \textstyle\bigcup_{V \in \mathcal{V}} V, \qquad \forall V \in \mathcal{V}, U \in \mathcal{V} \setminus V : V \cap U = \emptyset,$$

because GLS is guaranteed to switch from GS to LS, at latest when the cardinality of the search space is reduced to one. Let $N(e)$ be the expected cost of finding item e using GLS and $N(V)$ be the cost of user interaction until GLS reaches V. Let $\pi(V) = \sum_{e \in V} \pi(e)$ be the probability mass of all items in V.

THEOREM 3.1. *The expected cost of* GLS *is smaller than or equal to the expected cost of* GS.

PROOF. The expected cost of GS is defined as

$$\mathbb{E}_{e \sim \pi}[N(e)] = \gamma_q \sum_{e \in E} \pi(e) N(e). \tag{1}$$

For any target item e^*, Algorithm 4 switches from GS to LS at one particular point, where $e^* \in V$ for some $V \in \mathcal{V}$. Based on this, the expected cost can be written as

$$\gamma_q \sum_{V \in \mathcal{V}} \left[\pi(V) N(V) + \sum_{e \in V} \pi(e)(N(e) - N(V)) \right]. \tag{2}$$

The first term is the expected cost of GS before V is reached. The second term is the expected cost of GS after V is reached. Equation (2) can be further rewritten as

$$\sum_{V \in \mathcal{V}} \pi(V) \left[\gamma_q N(V) + \gamma_q \sum_{e \in V} \frac{\pi(e)}{\pi(V)} (N(e) - N(V)) \right], \tag{3}$$

where $\gamma_q \sum_{e \in V} \frac{\pi(e)}{\pi(V)} (N(e) - N(V))$ is the expected cost of GS applied to V. Let $N_{\mathrm{LS}}(V)$ be the expected cost of LS from V onward. According to our assumption,

$$N_{\mathrm{LS}}(V) \leq \gamma_q \sum_{e \in V} \frac{\pi(e)}{\pi(V)} (N(e) - N(V)) \tag{4}$$

because costGS$(V, K) \geq$ costLS(V, L) for all search spaces $V \in \mathcal{V}$. Finally, we chain all inequalities and get

$$\mathbb{E}_{e \sim \pi}[N(e)] \geq \sum_{V \in \mathcal{V}} \pi(V) \left[\gamma_q N(V) + N_{\mathrm{LS}}(V) \right], \tag{5}$$

where the right-hand side is the expected cost of GLS. This concludes the proof. □

4. OFFLINE EVALUATION

We conduct offline evaluations of the proposed GLS approach using two public recommender systems datasets and compare it to several baselines and heuristic approaches.

4.1. Experimental Setting

We compare GLS to five baselines. The first two baselines are LS and GS, which are presented in Sections 2.1 and 3.1. The other three baselines are heuristic combinations of GS and LS, which we call GS-step, GS-num, and GS-ent. Similarly to GLS, these three baselines combine GS and LS sequentially but differ in the switching condition from GS to LS: GS-step switches to LS after Δ iterations of GS, GS-num switches when the search space contains less than Δ items, and GS-ent switches when the entropy of the probability mass over the search space items is smaller than Δ. These heuristics intuitively address the question of switching from GS to LS: when the search space is refined enough times in GS-step, when the search space is sufficiently small in GS-num, and when the probability mass is concentrated in a small number of items in GS-ent. In all heuristics, the value of Δ is fixed a priori and is the parameter of the heuristic search.

The compared methods are evaluated by the expected cost of interaction, which is computed as follows. First, we randomly choose a user, proportionally to the number of positively rated items by the user. Second, we randomly choose a target item e^* from these positively rated items. Finally, we compute the cost of interaction to discover item e^* as the sum of the costs of GS and LS in the discovery of e^*. We assume that the user answers GS questions correctly and scans the list of recommended items until finding e^*. The expected cost of interaction is estimated from 10^8 randomly chosen user-item pairs, as described earlier. The standard deviation of the cost is never larger than 100. Therefore, the standard error of our empirical estimates is never larger than $100/\sqrt{10^8} = 0.01$, and any observed difference of at least 0.05 is statistically significant at $p < 0.05$. We experiment with three different numbers of item categories that are shown to the user in GS questions: $K = 2$, $K = 4$, and $K = 8$. In all experiments, the number of shown items in LS is fixed at $L = 8$. We observe similar trends for other values of L and do not report these results. The proposed approaches are evaluated on two datasets: MovieLens[2] and Lastfm.[3]

The MovieLens dataset is a dataset of 6k people who give 1M ratings to 3.9k movies. We formulate the problem of interactive search as follows. The items E are movies. A user assigns a positive rating to item e when the user's rating for movie e is four stars or more. Therefore, the probability that item e is the target item, $\pi(e)$, is the number of times that item e has positive ratings over the total number of positive ratings. The item categories q are movie genres. The MovieLens dataset contains 18 movie genres. This is relatively small, although sufficient for applying GLS and reporting improvements. In practice, we expect that GLS would be most beneficial in the problems where each item belongs to many categories of varying granularity that allow it to be discriminated from other items. To illustrate this setting, we further preprocess our problem as follows. First, we replace the MovieLens genres of each movie in E with those from Netflix.[4] There are 335 distinct Netflix genres in our dataset. Second, we eliminate all movies in E that are described by less than 5 genres. We end up with a dataset of 1.4k movies. This preprocessing step does not fundamentally change our results. The only change is that the improvement of interactive search over LS is more prominent, as we eliminate items that are hard to find by interactive search.

The Lastfm dataset is a dataset of 2k people who listen to 18k artists. We formulate the problem of interactive search as follows. The items E are artists. A user is perceived to like item e when the user listens to artist e. Therefore, the probability that item e

[2]MovieLens dataset: http://www.grouplens.org/node/73.
[3]Last.fm dataset: http://www.lastfm.com.
[4]Netflix: http://www.netflix.com.

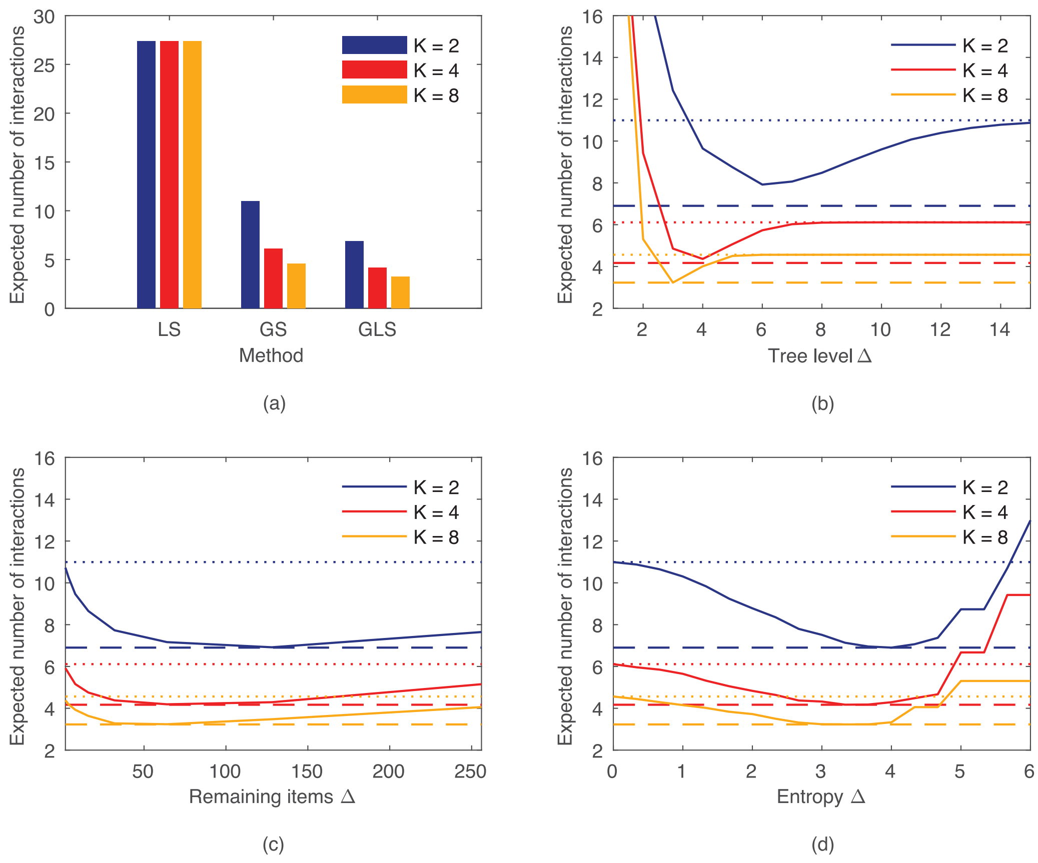


Fig. 5. Evaluation on the MovieLens dataset at $K \in \{2, 4, 8\}$. (a) Comparison of `GLS`, `GS`, and `LS`. (b) Expected cost of `GS-step` (solid lines) as a function of Δ. For comparison purposes, we report the expected costs of `GLS` (dashed lines) and `GS` (dotted lines). (c) Expected cost of `GS-num` (solid lines) as a function of Δ. For comparison purposes, we report the expected costs of `GLS` (dashed lines) and `GS` (dotted lines). (d) Expected cost of `GS-ent` (solid lines) as a function of Δ. For comparison purposes, we report the expected costs of `GLS` (dashed lines) and `GS` (dotted lines). Any reported difference of at least 0.05 is statistically significant at $p < 0.05$.

is the target item, $\pi(e)$, is the number of times that artist e is listened to over the total number of artist listening events. The item categories q are the tags assigned by the users to the artists. Finally, we eliminate all artists in E that are described by fewer than five tags and end up with a dataset of 6k artists. Again, we would like to stress that this preprocessing step does not fundamentally change our results. The only change is that the improvement of interactive search over `LS` is more prominent.

4.2. Experimental Results

Figure 5 shows the comparison of `GLS` to `LS`, `GS`, and our three heuristic baselines using the MovieLens dataset. In Figure 5(a), we compare `GLS` to `GS` and `LS` for three values of K. We observe two major trends. First, the cost of `GLS` and `GS` interactions decreases with the increase of K, as larger values of K allow for a finer-grain partitioning of the search space. The cost of `LS` is, as expected, independent of K but rather depends on L, which is fixed. Second, for all values of K, `GLS` significantly outperforms both `GS` and `LS`. For example, for $K = 2$, the cost of `GLS` interactions is 6.90, which is 37% smaller than that of GS (10.99) and 75% smaller than that of LS (27.40). For $K = 4$ and $K = 8$, the cost of GLS interactions respectively is 32% and 29% smaller than that of GS, and 85% and 88% smaller than that of LS.

In Figure 5(b), (c), and (d), we respectively compare GLS to three heuristics: GS-step, GS-num, and GS-ent. For all values of K, the cost of GS is the same as in Figure 5(a), and we show it only for comparison purposes. It should be stressed that the cost of GS is independent of Δ, which affects only the heuristics. We highlight several findings. First, as in Figure 5(a), the cost of interactions is inversely correlated with K, presumably due to the finer-grain partitioning of the search space facilitated by larger values of K. Second, GLS consistently achieves shorter interactions than those of the heuristics. This holds for all values of K and all evaluated heuristics. Therefore, GLS not only outperforms its individual components LS and GS but also outperforms their nonadaptive hybridizations.

The performance of the heuristics depends strongly on their parameter Δ. Consider GS-step in Figure 5(b). For $K = 2$, GS-step(6) achieves the lowest cost, whereas the cost of GS-step(3) is higher than the cost of GS. We observe similar trends for other values of K. For instance, for $K = 8$, the cost of GS-step(3) is comparable to that of GLS, but the cost of GS-step(2) is higher than that of GS. The performance of GS-num in Figure 5(c) also depends on the value of Δ. For $K = 2$, the lowest cost of GS-num is achieved for $\Delta = 130$. For $K = 4$ and $K = 8$, this is achieved for $\Delta = 70$ and $\Delta = 50$, respectively. Additionally note that for all values of K, GS-num performs similarly to GLS at the optimal operating points, whereas for GS-step, this happens only for $K = 8$. We attribute this to the fact that the number of items in the search space is a better indicator of suboptimal performance of GS than the number of GS steps. We observe similar trends for GS-ent in Figure 5(d), although in this case the variations in the cost of interactions with K are slightly more prominent. Overall, GLS always outperforms or performs similarly to the three heuristics.

We conduct a similar experiment using the Lastfm dataset, and our results are reported in Figure 6. Our findings resemble those in the MovieLens dataset. The cost of GLS interactions is always smaller than the costs of LS and GS, and GLS either outperforms or is comparable to the three heuristics, whose performance depends on the setting of their parameter Δ. A new observation drawn from this experiment is that the optimal operating points of the heuristics, which depend on the value of K, also depend on the dataset. For example, in the MovieLens experiment, GS-step achieves the lowest cost of interactions for $\Delta = 3$ when $K = 2$, $\Delta = 4$ when $K = 4$, and $\Delta = 6$ when $K = 8$. In the Lastfm experiment, GS-step achieves the lowest cost for $\Delta = 4$, $\Delta = 5$, and $\Delta = 8$ at the same values of K. Similar differences in the optimal operating points are observed for the other two heuristics, GS-num and GS-ent.

We conclude that GLS consistently outperforms its individual components, GS and LS. This superiority is statistically significant and holds for both the MovieLens and Lastfm datasets, as well as all studied values of K. A more intricate question, however, is the comparison of GLS to the heuristics GS-step, GS-num, and GS-ent. We conclude that GLS steadily outperforms GS-step and generally performs better than GS-num and GS-ent. The latter two can achieve performance comparable to GLS, but only when they are parameterized optimally. However, it should be noted that the optimal operating points Δ of GS-num and GS-ent depend on the applied heuristic, the dataset in hand, and the number of shown categories K. In practice, the heuristics are typically parameterized a priori; the value of the parameter Δ is set at the system deployment time and can be revised only occasionally. Therefore, setting Δ to the optimal value is not a trivial task. On the other hand, GLS is parameter free and never performs worse than GS-step, GS-num, and GS-ent. Therefore, we conclude that GLS should be preferred not only to GS and LS but also to the three evaluated heuristics.

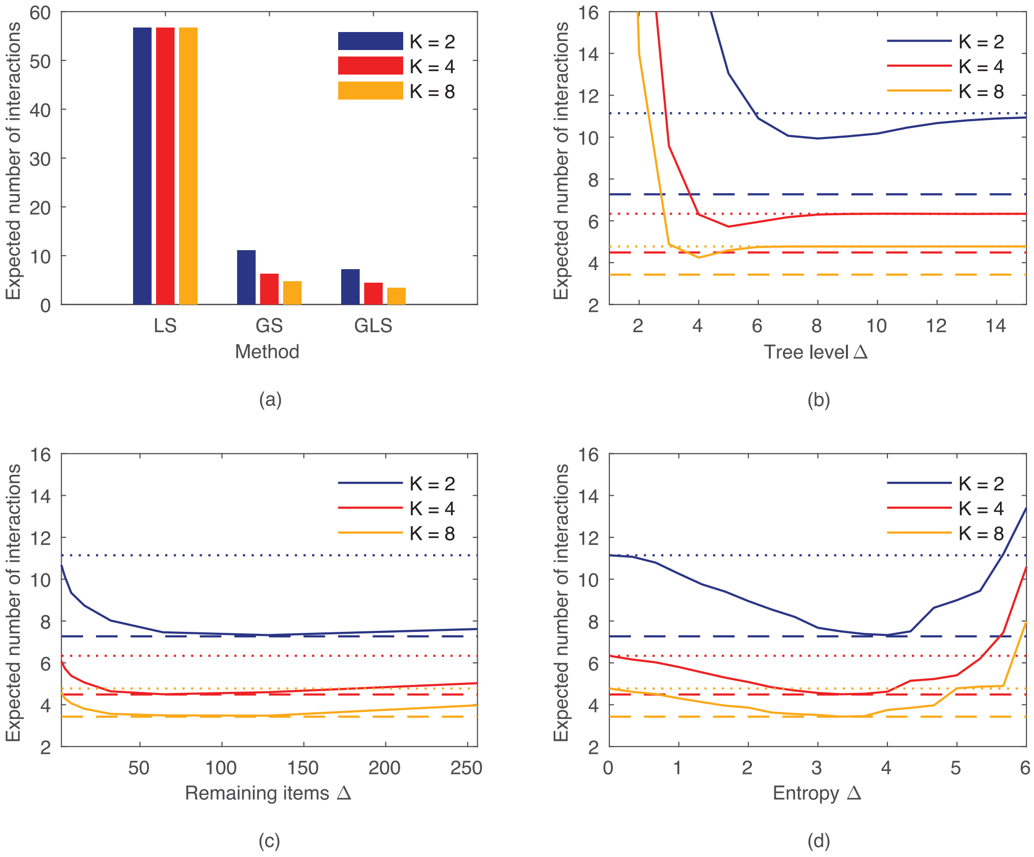


Fig. 6. Evaluation of the Lastfm dataset at $K \in \{2, 4, 8\}$. (a) Comparison of GLS, GS, and LS. (b) Expected cost of GS-step (solid lines) as a function of Δ. For comparison purposes, we report the expected costs of GLS (dashed lines) and GS (dotted lines). (c) Expected cost of GS-num (solid lines) as a function of Δ. For comparison purposes, we report the expected costs of GLS (dashed lines) and GS (dotted lines). (d) Expected cost of GS-ent (solid lines) as a function of Δ. For comparison purposes, we report the expected costs of GLS (dashed lines) and GS (dotted lines). Any reported difference of at least 0.05 is statistically significant at $p < 0.05$.

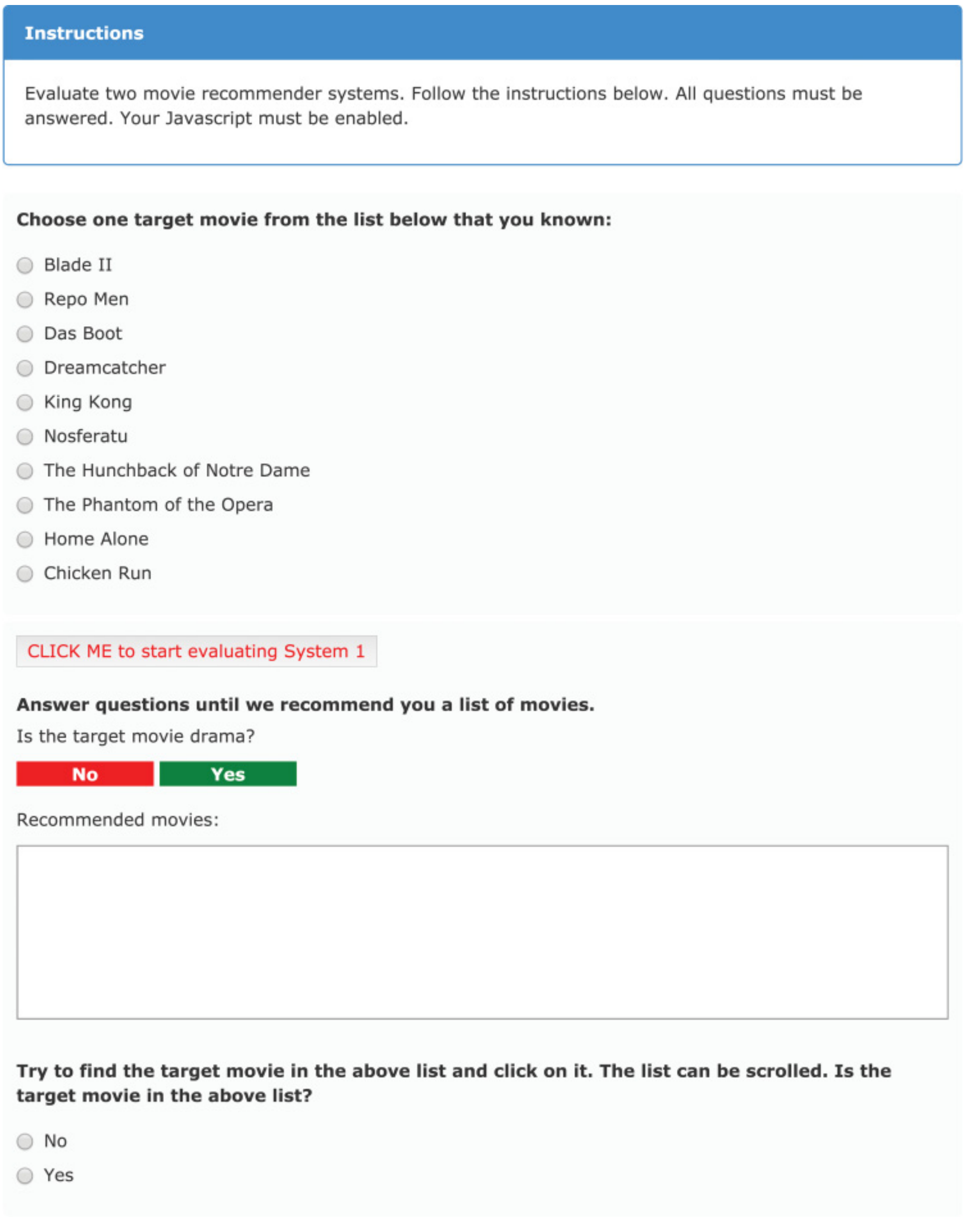
5. LIVE USER STUDY

We conduct a live user study of GLS using Amazon's Mechanical Turk,[5] which is a public crowd-sourcing service. For the sake of simplicity, we only compare GLS with LS and GS, and leave the heuristics beyond the scope of the live study. We evaluate a binary version of GS and show 10 items in each LS list. This means that $K = 2$ and $L = 10$.

5.1. Experimental Setting

Next we report on three user studies. In the first study, we compare GLS to LS. In the second study, we compare GLS to GS. These two studies compare GLS with its individual components. Note that we do not directly compare LS to GS, as these are established search methods that only inform our main contribution, GLS. In the third study, we evaluate the impact of the cost parameter γ_q. We estimate this parameter based on the results of the first two studies, parameterize GLS accordingly, and compare this *tuned* version of GLS to GLS where the costs of GS and LS are identical, $\gamma_q = 1$. All three are intragroup studies, where each subject experiences two types of search, which depend

[5]Amazon's Mechanical Turk: http://mturk.com.


Instructions

Evaluate two movie recommender systems. Follow the instructions below. All questions must be answered. Your Javascript must be enabled.

Choose one target movie from the list below that you known:

- ○ Blade II
- ○ Repo Men
- ○ Das Boot
- ○ Dreamcatcher
- ○ King Kong
- ○ Nosferatu
- ○ The Hunchback of Notre Dame
- ○ The Phantom of the Opera
- ○ Home Alone
- ○ Chicken Run

Answer questions until we recommend you a list of movies.

Is the target movie drama?

Recommended movies:

Try to find the target movie in the above list and click on it. The list can be scrolled. Is the target movie in the above list?

- ○ No
- ○ Yes

Fig. 7. User study.

on the study, and is asked to compare them. The searches are shown in a randomized order to eliminate position bias.

Our studies are conducted on IMDb[6] movies. The set of recommended items E is 500 most-rated movies from IMDb. The studies are structured as follows. At the beginning, we randomly select 10 movies from E, present them to the subject, and ask the subject to select a movie with which the subject is familiar (top of Figure 7). This movie becomes

[6]IMDb: http://www.imdb.com.

Which system do you prefer with respect to the following:

(C1) Number of asked questions
- No preference
- I prefer System 1 because it asks less questions
- I prefer System 2 because it asks less questions

(C2) Length of the final list
- No preference
- I prefer System 1 because it produces a shorter list
- I prefer System 2 because it produces a shorter list

(C3) Ease of use
- No preference
- I prefer System 1 because it is easier to use
- I prefer System 2 because it is easier to use

(C4) Overall
- I prefer System 1 to System 2
- I prefer System 2 to System 1

Fig. 8. Subjective search comparison questions.

the target movie of the subject. After selecting the target movie, the subject interacts with the first search. If the evaluated search is LS, the subject scans through batches of $L = 10$ movies until the target movie is identified. If the evaluated search is GS, the subject answers questions with $K = 2$ categories,[7] each of which is a binary yes-or-no question about a movie genre (middle of Figure 7). If the evaluated search is GLS, the subject first interacts with binary GS questions and then switches to LS. Regardless of the evaluated method, the subject is shown a list of movies at the end, requested to examine it, and verifies whether the target item is included in this list (bottom of Figure 7). Then the subject interacts with the second search and verifies whether the target item is in the second recommendation list.

On completion of both searches, the subject is asked to answer four questions that compare the searches, as shown in Figure 8. These questions evaluate the subject's perception of three criteria—length of the search (number of question asked), length of the recommendation list at the end of the search, and ease-of-use of the search—and overall preference. In the first three questions, the subject can choose their preferred search[8] or indicate that the two searches are comparable. In the last question, which evaluates the overall preference, the subject has to choose one of the searches.

In addition to the subjective questions, we also capture the interaction logs for each search and compute seven metrics. These metrics are as follows. The *hit rate* is the fraction of searches where the target movie appears in the final recommendation list. This measures the ability of the search to discover the target movie, under assumption that the subject answers all questions correctly. The *number of questions* is the number of asked GS questions, averaged over all users. The *question-answering time* is the time (in seconds) to answer GS questions, averaged over all users. The *time per question* is the average cost of answering a GS question, which is computed as the ratio of the

[7]Due to technical limitations imposed by Amazon's Mechanical Turk, we limit the number of GS questions in both GS and GLS to 11.
[8]We do not introduce any names but rather refer to the searches as System 1 and System 2.

	LS	GLS	Neither
Hit rate	0.77	0.25	
Number of questions	N/A	4.60	
Question-answering time	N/A	10.56	
Time per question	N/A	2.33	
Length of the final list	500.00	23.30	
List-scanning time	41.68	12.18	
Interaction time	41.68	22.74	
(C1) Number of questions	41%	23%	36%
(C2) Length of the final list	12%	68%	20%
(C3) Ease of use	23%	58%	19%
(C4) Overall preference	36%	64%	

Fig. 9. Summary of the user study on LS and GLS.

question-answering time and the number of asked GS questions, and then averaged over all users. The *length of the final list* is the number of movies in the final recommendation list, averaged over all users. More formally, it is the cardinality of the search space V when the switch from GS to LS happens. The *list-scanning time* is the time (in seconds) for scanning the batches of movies in LS, averaged over all users. Finally, the *interaction time* is the duration of the search (in seconds), and we compute it as the sum of the question-answering and list-scanning times.

In the following sections, we present the results of our three studies: GLS versus LS, GLS versus GS, and GLS versus tuned GLS. Each of the studies involves 100 subjects. The number of subjects is chosen based on our prior work [Kveton and Berkovsky 2015]. In particular, we hypothesize that GLS can outperform GS and LS by quite a large margin, and this can be statistically significant on a relatively small number of subjects. For instance, if 66 subjects out of 100 prefer GLS to LS, this result is statistically significant at $p < 0.001$ using the binomial test, under the null hypothesis that GLS and LS perform equally.

5.2. Comparison of GLS to LS

We initially compare GLS to LS. In Figure 9, we report the values of the seven metrics for each search, as well as the distribution of the answers to the comparison questions in Figure 8. Note that since there are no GS questions in LS, the three metrics that refer to GS are inapplicable to LS, the size of the final list is the entire set E of 500 movies, and the overall interaction time is equal to the list-scanning time. We observe that the hit rate of LS is much higher than that of GLS. We posit that this can be explained by the very nature of LS that requires the subject to scan the list until the target movie is discovered. On the contrary, GLS involves answering 4.60 questions on average, where a single wrong answer may take the subject off the track and prevent the discovery of the target movie. The flip side of the higher hit rate is the overall interaction time, which is 41.68 seconds for LS and 22.74 seconds for GLS. In other words, GLS takes about 45% less time than LS. This finding is despite the fact that the LS part of GLS requires the subject to scan 23.30 items on average, which takes 12.18 seconds out of the overall interaction time of 22.74 seconds.

In the subjective questions, GLS is preferred over LS by more subjects: 64% versus 36% (C4). What are the possible reasons for this preference? Here, 41% of the subjects prefer LS due to the number of asked questions (C1) compared to 23% who prefer GLS. This reflects the fact that no questions are asked in LS, whereas the GS part of GLS does involve 4.60 questions on average. Having said that, the subjects clearly prefer the

	GS	GLS	Neither
Hit rate	0.06	0.29	
Number of questions	9.28	4.60	
Question-answering time	13.82	10.25	
Time per question	1.50	2.20	
Length of the final list	2.16	24.67	
List-scanning time	5.98	12.12	
Interaction time	19.80	22.37	
(C1) Number of questions	6%	60%	34%
(C2) Length of the final list	23%	29%	48%
(C3) Ease of use	7%	34%	59%
(C4) Overall preference	24%	76%	

Fig. 10. Summary of the user study on GS and GLS.

list produced by GLS: 68% versus 12% (C2). This is due to the size of the search space, which includes 23.30 movies on average in GLS and all 500 movies in LS. Looking at the perceived ease of use (C3), we also spot the superiority of GLS, which is preferred by 58% of the subjects, whereas LS is preferred by only 23%. In summary, we posit that the overall perceived preference of GLS over LS is due to the shorter final lists and the ease of use, and these outweigh the larger number of questions asked by GLS.

We also observe that a portion of the subjects has no clear preference toward the three criteria: 36% (C1), 20% (C2), and 19% (C3). We believe that the 36% of undecided subjects in C1 is due to the ambiguity of what we mean by "question" in LS. For C2 and C3, the number of undecided subjects is around 20%, and we deem it to be acceptable. Nevertheless, the observed preference of GLS over LS is statistically significant. More specifically, under the null hypothesis that the subjects prefer GLS and LS equally, observing 64 or more preferential votes towardGLS out of 100 is statistically significant at $p < 0.005$ by the binomial test.

5.3. Comparison of GLS to GS

Now we compare GLS to GS. In Figure 10, we report the values of the seven metrics for each search, as well as the distribution of the answers to the comparison questions in Figure 8. Before we continue, it should be noted that the two approaches are quite similar. They share the first part of GS questions. Toward the end of the search, GLS switches to LS lists, whereas GS asks questions until either a single movie remains in the search space V or the limit of 11 questions is reached. In the latter case, all remaining movies in V are shown to the subject in the final list.

We observe several trends. First, the results of GLS are quite close to those obtained in the previous comparison to LS. This shows the outstanding stability of GLS across users and target items. Second, the observed hit rate of GS is much lower than that of GLS. This is explained by the fact that GS asks more than twice as many questions as GLS: 9.28 versus 4.60. The extra questions asked by GS eliminate many movies from the final list, which is only 2.16 movies long on average. However, if the subject answers any GS question incorrectly, the target movie is eliminated from the final list. Therefore, a method that asks more GS questions is likely to have a lower hit rate. Third, the overall interaction times of the two searches are comparable: 19.80 seconds versus 22.37 seconds. In other words, the extra questions asked by GS are balanced by the time to examine the much longer final list in GLS, and the overall interaction times are comparable despite the differences in their composition.

The subjective questions clearly show that our subjects correctly perceive the larger number of asked questions (C1). Specifically, 60% of the subjects prefer GLS versus only 6% that prefer GS. Most of these subjects also find GLS easier to use than GS (C3), and the preference is 34% versus 7% in favor of GLS. Surprisingly, the two approaches seem comparable with respect to the length of the final list (C2): 29% of the subjects prefer GLS and 23% prefer GS. We hypothesize that the GS lists, which contain only 2.16 movies on average, are perceived as too short, whereas the GLS lists, which contain 24.67 movies on average, are perceived as too long. As a result, none of the lists is a clear winner and the overall preference scores are comparable. All in all, the perceived strengths of GLS in the number of asked questions and the ease of use seem to bear more importance for the subjects than the length of the final list, and 76% of the subjects prefer GLS over GS overall (C4).

We also observe that a larger portion of the subjects than in the previous comparison has no clear preference in the three subjective questions: 34% (C1), 48% (C2), and 59% (C3). This finding is not surprising, because as we mentioned earlier, the two searches are similar and differ only in the last few steps, when GLS switches to LS and produces a longer final list, whereas GS continues to ask questions. The subjects correctly perceive this similarity, and many of them have no preference with regard to criteria C1, C2, and C3. The preference of the subjects toward GLS is statistically significant. More specifically, under the null hypothesis that the subjects prefer GLS and GS equally, observing 75 or more preferential votes toward GLS out of 100 is statistically significant at $p < 10^{-6}$ by the binomial test.

5.4. Comparison of GLS to Tuned GLS

In the last study, we compare the *uniform* version of GLS, which was evaluated in the previous studies, to the *tuned* version, where we set the value of the cost parameter γ_q in a data-driven way. The value of the parameter γ_q is estimated as follows. Based on Figures 9 and 10, the average time for answering a GS question with $K = 2$ categories is $\omega_{\text{GS}}(K) = 2.27$ seconds. We assume that this is the cost of one interaction in GS. We estimate the average time for scanning an LS list with $L = 10$ items as the average of the list-scanning times of each subject over the expected number of the scans that the subject conducted. This is $\omega_{\text{LS}}(L) = 6.18$ seconds, and we assume that this is the cost of one interaction in LS. Assuming that the subjects concentrate solely on the task, the time of interaction is a reliable proxy for the cost of interaction. Thus, we parameterize GLS by $\gamma_q = \omega_{\text{GS}}(K)/\omega_{\text{LS}}(L) = 2.27/6.18 \approx 0.37$. This data-driven parameterization is expected to lead to a more tuned version of GLS than the previous $\gamma_q = 1$, because this model better reflects the actual observed costs of the interactions.

Our results are reported in Figure 11. Overall, the seven measured metrics show little difference between the compared methods. Tuned GLS asks about 30% more questions than the uniform one: 5.99 versus 4.62. This is in line with the parameterization of $\gamma_q < 1$, which implies that the GS questions are cheaper than scanning of the LS lists. Therefore, the tuned GLS asks more questions than the uniform GLS, which assigns equal costs to both types of the interactions. These extra questions reduce the length of the final list by about 55%, and it drops from 26.50 movies in the uniform GLS to 11.98 in the tuned version. On the flip side, they also reduce the observed hit rate, as the target movie has a higher chance to be eliminated by a wrong answer to one of the extra questions. In this study, we also observe a minor difference in the overall interaction time: 20.84 seconds versus 24.57 seconds. This difference is mainly due to the observed scanning time of the final list. Despite the fact that the final lists of the uniform GLS are 2.21 times longer than those of the tuned GLS, the scanning time is only about 45% longer, which diminishes the differences in the interaction times.

	Tuned GLS	Uniform GLS	Neither
Hit rate	0.24	0.34	
Number of questions	5.99	4.62	
Question-answering time	11.28	10.61	
Time per question	1.92	2.28	
Length of the final list	11.98	26.50	
List-scanning time	9.56	13.96	
Interaction time	20.84	24.57	
(C1) Number of questions	11%	29%	60%
(C2) Length of the final list	29%	15%	56%
(C3) Ease of use	9%	20%	71%
(C4) Overall preference	45%	55%	

Fig. 11. Summary of the user study on tuned GLS and GLS.

The subjective questions show a slight preference of the uniform GLS over the tuned one: overall, 55% of the subjects prefer the former and 45% prefer the latter (C4). Looking deeper, we observe that in terms of the number of questions answered (C1), 29% prefer the uniform GLS and 11% prefer the tuned GLS, 29% prefer the length of the final list produced by the uniform GLS and 15% prefer the final list of the tuned GLS (C2), and 20% prefer the ease of use of the uniform GLS versus 9% who prefer the tuned GLS. More importantly, we observe that more than a half of the subjects have no clear preference with regard to the three subjective criteria, and this number is much larger than in the first two studies. Specifically, 60% of the subjects have no preference for C1, 56% have no preference for C2, and 71% have no preference for C3.

On top of this observation, our statistical significance test shows that the superiority of the uniform GLS over the tuned GLS is not significant. In particular, under the null hypothesis that the subjects prefer the uniform and tuned GLS equally, observing 55 or more preferential votes toward the uniform GLS out of 100 is statistically significant at a mere $p \approx 0.15$ by the binomial test. This practically means that many subjects do not find any major difference between the two compared approaches. One could argue that the differences may be significant with a larger sample of subjects. Nevertheless, note that the superiority of GLS over LS and GS is significant with the same sample size of 100 subjects, yet the two approaches in this study are perceived as comparable.

In summary, this experiment shows that the tuned GLS performs slightly worse than the uniform GLS, although this difference is not statistically significant. This does not mean that GLS performs optimally when the costs of interactions are equal. Rather, it indicates that the notion of the costs in this experiment does not necessarily reflect the cost model of the user. We believe that there exist better cost models and that this is an interesting direction for future work.

6. RELATED WORK

Recommendations produced by a recommender system are typically shown to users as a ranked list of items. The ranking of the recommended items in the list communicates their fit for the target user—for example, the top-ranked items are supposedly the best recommendations [McNee et al. 2006]. The size of the recommendation list is normally fairly small, reflecting one of the main goals of the recommenders—reducing the choice difficulty and information overload [Bollen et al. 2010]. In addition to that, the size of the list may be constrained by other practical factors, such as the number of items that can be consumed or the screen size of a mobile device.

Several issues around interfaces for displaying the recommendations to users have been investigated in early recommendation works. Cosley et al. [2003] studied how the display and the scale of item predictions affect the item ratings provided by users. It was found that user ratings were influenced by the item prediction scores computed and displayed by the recommender, whereas the scale had no apparent effect. Ziegler et al. [2005] argued that the recommendation list should incorporate a degree of diversity to reflect the breadth of user interests [Ashkan et al. 2015]. It was shown that despite decreasing the accuracy of the recommendations, diversification increased user satisfaction and overall user experience [Knijnenburg et al. 2012].

Often, the recommendation list can be grouped or categorized to simplify navigation in the search space and spur further content exploration by users. Hu and Pu [2011] proposed and evaluated an "organizational" recommendation interface that allowed users to group and filter items according to a set of domain-dependent item features. A comparison between the organizational and list interfaces indicated that the former improved the perceived usefulness and diversity of the recommendations, increased user confidence, and was found to be more intuitive and easier to use than the latter.

The importance of an easy-to-use interface is paramount in critique-based recommendations [Chen and Pu 2012]. Chen and Pu [2012] developed an interface that grouped domain items, compared these groups to the items in the recommendation list, and suggested critiques reflecting these comparisons [Chen and Pu 2010]. The accuracy of the critiques and the usability of the interface were evaluated in a user study. The results showed that users preferred the suggested critiques and provided them more frequently, which reduced the duration of interactions with the recommender and increased the perceived user confidence.

Another functionality availed by the recommender's interface is explanation and persuasion [Berkovsky et al. 2012]. These are used to justify the recommendations, cultivate user trust, and convince users to follow the recommendations. In many practical scenarios, and specifically in eCommerce applications, this functionality is important, as the success of the system is measured by the uptake of the recommendations. Tintarev and Masthoff [2012] developed several interfaces complementing the recommendation list with item explanations. Their evaluation confirmed that explanations boost user satisfaction. The highest satisfaction was achieved with personalized explanations using item features, which were relevant for users.

Although much research focused on exploration interfaces in the general HCI context [Baldonado and Winograd 1997], this has received relatively little attention in recommender systems [Tintarev et al. 2012, 2014]. In a recent work, Verbert et al. [2013] developed TalkExplorer, an interactive conference visualization tool, building on graph-based information exploration libraries such as PeerChooser [O'Donovan et al. 2008] and SmallWorlds [Gretarsson et al. 2010]. TalkExplorer allowed users to explore various relevance perspectives, such as tags assigned to papers, users who bookmarked these papers, and other papers tagged by these users. The evaluation indicated that TalkExplorer was perceived as more insightful than the standard recommendation list.

The preceding interface components and visualization tools supply various information complementing the recommendations. However, none of them looks into optimizing user experience in recommender systems [Knijnenburg et al. 2012]. This is a central issue addressed by general user interaction design guidelines, and we observe successful practical solutions in industrial products, such as movie recommendation interface by Netflix[9] or contact recommendations by LinkedIn.[10] However, to the best of our

[9]Netflix recommendations: Beyond the 5 stars: http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html.
[10]Navigating LinkedIn's New User Interface: http://randomactsofleadership.com/navigating-linkedins-new-user-interface/.

knowledge, no evidence-based research that articulates the development of these interfaces, particularly considering the recommendation scenarios, is publicly available.

7. CONCLUSIONS

Recommender systems research has mainly focused on algorithmic techniques that either improve the accuracy of rating predictions or other recommendation features, such as diversity, robustness, and scalability [Ricci et al. 2011]. Relatively little research turned to recommendation interfaces and user interaction with the recommender.

In this article we build on the work of Kveton and Berkovsky [2015] and propose a novel approach to searching the list of recommended items, GLS. GLS minimizes the cost of user-recommender interactions and allows users to search the list of recommended items more easily. It combines in a sequential manner two established search approaches—GS and LS—and leverages their advantages. The user interaction with GLS starts with answering GS questions and then proceeds to the LS of recommended items. We investigate the switching criterion from GS to LS and conclude that GS should be applied as long as its expected cost is lower than that of LS. We formally prove that GLS performs at least as well as GS.

Our experimental evaluation shows the superiority of GLS not only to its individual components (GS and LS) but also to three heuristics that switch from GS to LS in a different way. In the offline evaluation, we study the performance of GLS on two recommendation datasets. We show that GLS consistently outperforms both GS and LS, and it performs at least as well as, and generally better than, the three heuristics. However, unlike the heuristics that need to be parameterized, GLS is adaptive and parameter free. It does not require any parameter tuning and can be deployed in practical recommenders and highly dynamic recommendation domains.

We also conduct three live user studies with Mechanical Turk workers. In these studies, we compare GLS to both LS and GS, as well as the tuned variant of GLS that assigns different costs to GS and LS interactions. The studies clearly demonstrate that users perceive GLS as easier to use than LS and GS, and also prefer it overall. A further comparison between GLS and its tuned variant does not yield conclusive results, presumably due to the high similarity of the approaches. In summary, the offline and live evaluations strengthen our formal analysis and show the superiority of GLS over its counterparts.

One aspect of our work that requires further investigation is user interaction with GLS. Although we find that GLS leads to shorter interactions and is also preferred by users, it blends item categories and items into a single process. Therefore, it is important to ascertain whether users find GLS interactions intuitive and enjoyable [Knijnenburg et al. 2012]. It is also not obvious how GLS, and particularly the item categories in GS questions, should be visualized [Zhang et al. 2008]. On one hand, textual descriptors may not be as appealing as pictorial thumbnails. On the other hand, the thumbnails may confuse users and incorrectly communicate the very notion of item categories.

Another aspect that requires further investigation is connecting GLS with other interface and user-related topics in recommender systems [Felfernig et al. 2012]. Among these are the contribution of GLS to human decision support processes, application of GLS in critique-based recommender systems, integration of GLS with explanations of recommendations, robustness of GLS to user mistakes in answering GS questions, and promoting certain items. In the future, we plan to conduct additional user studies and evaluate the compound effect of GLS in the preceding use cases.

An important shortcoming of our mathematical model is that we assume the user answers GS questions correctly. Our work can be relatively straightforwardly generalized to noisy feedback, where the user answers questions incorrectly with some known probability. One of the first works that extends GS in this direction is that of Nowak [2009]. The key idea in this work is to ask similar questions multiple times to eliminate noise. Although related, this extension is somewhat orthogonal to our contribution, in which we show that GLS outperforms both GS and LS.

Another shortcoming of our model is the assumption that the model of user preferences for items π is known in advance. In practice, the recommender may not know π in advance and must learn it. Wen et al. [2013] proposed sequential Bayesian search, a learning variant of GS in which the distribution π is learned on the fly. This problem can be viewed more generally as learning to maximize an adaptive submodular function in the minimum number of steps [Gabillon et al. 2013, 2014]. We believe that this kind of learning is also possible in GLS.

ACKNOWLEDGMENT

The reviewing of this article was managed by Giuseppe Carenini and Shimei Pan, associate editors of the special issue on Highlights of IUI 2015.

REFERENCES

Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. 2015. Optimal greedy diversity for recommendation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. 1742–1748.

Michelle Baldonado and Terry Winograd. 1997. SenseMaker: An information-exploration interface supporting the contextual evolution of a user's interests. In *Proceedings of the Conference on Human Factors in Computing Systems*. 11–18.

Shlomo Berkovsky, Jill Freyne, and Harri Oinas-Kukkonen. 2012. Influencing individually: Fusing personalization and persuasion. *ACM Transactions on Interactive Intelligent Systems* 2, 2, Article No. 9.

S. Bhamidipati, B. Kveton, and S. Muthukrishnan. 2013. Minimal interaction search: Multi-way search with item categories. In *Proceedings of the AAAI Workshop on Intelligent Techniques for Web Personalization and Recommendation*.

Dirk Bollen, Bart Knijnenburg, Martijn Willemsen, and Mark Graus. 2010. Understanding choice overload in recommender systems. In *Proceedings of the 2010 ACM Conference on Recommender Systems*. 63–70.

Li Chen and Pearl Pu. 2010. Experiments on the preference-based organization interface in recommender systems. *ACM Transactions on Computer-Human Interaction* 17, 1, Article No. 5.

Li Chen and Pearl Pu. 2012. Critiquing-based recommenders: Survey and emerging trends. *User Modeling and User-Adapted Interaction* 22, 1, 125–150.

Dan Cosley, Shyong Lam, Istvan Albert, Joseph Konstan, and John Riedl. 2003. Is seeing believing? How recommender system interfaces affect users' opinions. In *Proceedings of the 2003 Conference on Human Factors in Computing Systems*. 585–592.

Sanjoy Dasgupta. 2005. Analysis of a greedy active learning strategy. In *Proceedings of Advances in Neural Information Processing Systems* 17. 337–344.

Alexander Felfernig, Robin Burke, and Pearl Pu. 2012. Preface to the special issue on user interfaces for recommender systems. *User Modeling and User-Adapted Interaction* 22, 4, 313–316.

V. Gabillon, B. Kveton, Z. Wen, B. Eriksson, and S. Muthukrishnan. 2013. Adaptive submodular maximization in bandit setting. In *Proceedings of Advances in Neural Information Processing Systems* 26. 2697–2705.

V. Gabillon, B. Kveton, Z. Wen, B. Eriksson, and S. Muthukrishnan. 2014. Large-scale optimistic adaptive submodularity. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*.

Daniel Golovin and Andreas Krause. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research* 42, 427–486.

Brynjar Gretarsson, John O'Donovan, Svetlin Bostandjiev, Christopher Hall, and Tobias Höllerer. 2010. SmallWorlds: Visualizing social recommendations. *Computer Graphics Forum* 29, 3, 833–842.

Rong Hu and Pearl Pu. 2011. Enhancing recommendation diversity with organization interfaces. In *Proceedings of the 2011 International Conference on Intelligent User Interfaces*. 347–350.

Bart Knijnenburg, Martijn Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction* 22, 4, 441–504.

Branislav Kveton and Shlomo Berkovsky. 2015. Minimal interaction search in recommender systems. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*. 236–246.

Sean McNee, John Riedl, and Joseph Konstan. 2006. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *Proceedings of the 2006 Conference on Human Factors in Computing Systems*. 1097–1101.

Robert Nowak. 2009. Noisy generalized binary search. In *Proceedings of Advances in Neural Information Processing Systems* 22. 1366–1374.

Robert Nowak. 2011. The geometry of generalized binary search. *IEEE Transactions on Information Theory* 57, 12, 7893–7906.

John O'Donovan, Barry Smyth, Brynjar Gretarsson, Svetlin Bostandjiev, and Tobias Höllerer. 2008. PeerChooser: Visual interactive recommendation. In *Proceedings of the 2008 Conference on Human Factors in Computing Systems*. 1085–1088.

Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2011. Introduction to recommender systems handbook. In *Recommender Systems Handbook*. Springer, 1–35.

Moushumi Sharmin, Lawrence Bergman, Jie Lu, and Ravi Konuru. 2012. On slide-based contextual cues for presentation reuse. In *Proceedings of the 17th International Conference on Intelligent User Interfaces*. 129–138.

Nava Tintarev, Rong Hu, and Pearl Pu. 2012. RecSys workshop on interfaces for recommender systems. In *Proceedings of the 6th ACM Conference on Recommender Systems*. 355–356.

Nava Tintarev and Judith Masthoff. 2012. Evaluating the effectiveness of explanations for recommender systems: Methodological issues and empirical studies on the impact of personalization. *User Modeling and User-Adapted Interaction* 22, 4, 399–439.

Nava Tintarev, John O'Donovan, Peter Brusilovsky, Alexander Felfernig, Giovanni Semeraro, and Pasquale Lops. 2014. RecSys workshop on interfaces and human decision making for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*. 383–384.

Katrien Verbert, Denis Parra, Peter Brusilovsky, and Erik Duval. 2013. Visualizing recommendations to support exploration, transparency and controllability. In *Proceedings of the 18th International Conference on Intelligent User Interfaces*. 351–362.

Zheng Wen, Branislav Kveton, Brian Eriksson, and Sandilya Bhamidipati. 2013. Sequential Bayesian search. In *Proceedings of the 30th International Conference on Machine Learning*. 226–234.

Andi Winterboer, Martin Tietze, Maria Wolters, and Johanna Moore. 2011. The user model-based summarize and refine approach improves information presentation in spoken dialog systems. *Computer Speech and Language* 25, 2, 175–191.

Jiyong Zhang, Nicolas Jones, and Pearl Pu. 2008. A visual interface for critiquing-based recommender systems. In *Proceedings of the 9th ACM Conference on Electronic Commerce*. 230–239.

Cai-Nicolas Ziegler, Sean McNee, Joseph Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*. 22–32.

Received July 2015; revised December 2015; accepted March 2016