Minimal Interaction Search: Multi-Way Search with Item Categories

Sandilya Bhamidipati
Technicolor Labs
Palo Alto, CA
sandilya.bhamidipati@technicolor.com

Branislav Kveton
Technicolor Labs
Palo Alto, CA
branislav.kveton@technicolor.com

S. Muthukrishnan
Department of Computer Science
Rutgers
muthu@cs.rutgers.edu

Abstract

Extensive online collections of content exist, such as music, books, movies, and various others. Search in these collections is typically implemented in two ways, typing attributes of interest into a search box, or progressively navigating through series of menus with item categories to a list of desired items. In this paper, we focus on the latter approach. In particular, we propose a strategy that guides the user to the items of interest in the minimal number of interactions. Therefore, we refer to our technique as *minimal interaction search*. At each step of the search, we show the user k item categories and ask them to choose the one that matches their preferences, or state that none does. We formalize this problem as multi-way search and propose an algorithm `DoubleGreedy` that solves the problem efficiently. The item categories in each question can be computed in $O(k)$ time, and any item in the database is found in the worst case in $\mathsf{OPT}_k \log(n-1)e/(e-1)$ questions, where n is the total number of items and OPT_k is the maximum number of questions asked by the optimal strategy (that uses the smallest number of questions possible in the worst case). We evaluate our method on two datasets of movies and books, and show that the target item can be found very fast.

1 Introduction

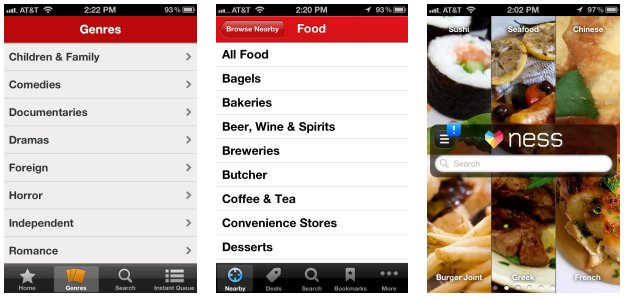
Extensive online collections of content exist, such as music, books, movies, and various others. Items in these collections are described by many attributes, some of which are carefully curated, such as the genre of a movie, and others that are freely generated by users, such as reviews. Search in this setting has been traditionally approached in two ways. In one approach, users type attributes of interest into a search box and the search engine returns a ranked list of items that are relevant to the user's query. In the other, users navigate through menus with item categories and progressively refine their selection to a list of desired items (Figure 1). Most content recommendation websites, like Netflix and Yelp, combine both of these approaches. In this paper, we focus on improving the latter approach.

Typically, the number of item categories is huge, and therefore they are presented to the user in a series of menus, from general to very specific ones. For instance, Netflix initially asks the user to choose a movie genre, such *TV Shows* or *Dramas*. When the user chooses *Dramas*, Netflix offers more specific choices, such as *Independent Dramas* and *Romantic Dramas*. Note that this policy for exploring the space of items can be represented by a tree, where the nodes are lists of item categories shown to the user and the branches are the choices of the user. In the rest of the paper, we refer to such a tree as a *search tree*.

The search tree is typically built manually and suffers from several drawbacks. First, if the branching factor of the tree is too large, the user may get overloaded with the amount of choices. For instance, Yelp users can choose from more than 100 types of cuisines. Only 9 fit on the screen of a smart phone (Figure 1b). Second, if the tree is too shallow, the list of recommended items satisfies only some of the user's criteria and includes many unwanted items. For instance, Netflix's search tree has only two levels. As a result, it is relatively easy to find movies that belong to 2 categories, such as *Dramas* and *Period Pieces*, but hard to find movies that belong to 3, for instance *Dramas*, *Period Pieces*, and *Action & Adventure*. Finally, note that the set of items tend to change dynamically over time, for instance whenever new content is available. Therefore, the search tree must be periodically updated.

In this paper, we propose a policy that guides the user to the items of interest in the *minimal number of interactions*. At each step of the search, we show the user k item categories and ask them to select the one that matches their preferences. Our approach has several notable features. First, we explicitly minimize the number of interactions with the user. In other words, we optimize the structure of the search tree such that the target item can be found fast. Second, the branching factor of our tree is bounded by k, where k is a domain-specific parameter. Therefore, the user is never given more than k choices. This constraint is motivated by real-world problems. More specifically, people increasingly search on mobile devices and the screen of these devices is small. Therefore, it is hard to display more than a limited number of choices. In addition, a large number of choices perplexes people and may lead to confusion. Therefore, it is only natural to consider this constraint.

We make three contributions. First, we formalize multi-way search with item categories as an interactive question-answering game. At each step of this game, the user is shown k categories of items and asked to select the one that matches their preference. Our goal is to minimize the number of such queries to find the item of interest, in the worst case.



(a) Netflix. (b) Yelp. (c) Likeness.

Figure 1: Examples of interactive search applications on a cell phone.

Second, we propose an efficient algorithm for solving this problem. Our method finds the target item in at most $\mathsf{OPT}_k \frac{e}{e-1} \log(n-1)$ queries, where n is the number of items and OPT_k is the maximum number of queries in the optimal solution. Each query is computed in $O(km^2)$ time, where m is the number of item categories and k is the number of item categories in the query. We also propose a version of the algorithm whose computation time grows only linearly with m. We refer to our algorithm as `DoubleGreedy` because we greedily seek queries that shrink the hypothesis space the most, and the search for the best query is also done greedily.

Third, we conduct an extensive experimental study. Our method is evaluated on two large datasets of movies and books, and we show how it scales with the number of items categories k in the query.

2 Minimal interaction search

In this section, we discuss our framework for *minimal interaction search*. We assume that each item belongs to multiple categories. The user has a specific target item in mind and uses a device with multiple choice capability, like a tablet, to find it. The system gives the user multiple choices and the user makes the choice that reveals their preferences. This process is repeated until the user has identified the target item that satisfies all their choices. The goal of the system is to bring the user to the target item in as few interactions as possible. The formalization of this process requires making several design choices.

First, we need to define what are the choices. In this paper, we assume that they are individual item categories. For instance, movie genres like *Dramas* and *Period Pieces*. Note that an item category can be a combination of existing categories, such as *Action & Adventure*. Alternatively, the user can be shown representative items (Karbasi, Ioannidis, and Massoulié 2012). While this approach is of interest in theory, in practice people are more comfortable with choosing a specific category.

Second, we need to decide how to present the choices and what is the response of the user. In this work, we show k item categories and ask the uses to choose the one that describes the target item on their mind. After the user makes a choice, the system shows another k categories. This approach has two nuances. First, suppose that more than one category describes the target item. In this case, we assume the user can choose *any* of the applicable categories. We do not want to burden the user to be more precise and choose the most applicable category, given some notion of distance. Second, suppose that no category applies to the target item. This is possible because the user is given only k choices and these may not cover the entire space of target items. Therefore, we provide a $(k+1)$-th choice, a special category *Other*, which comprises all items that are not represented by any of the k choices. Note that this $(k+1)$-th choice may not in general be captured by any single category.

Finally, we need to decide what performance measure is optimized. A natural choice is to minimize the amount of interaction with the user. Making any more choices than are necessary would likely only annoy the user. However, there is a nuance. In any particular instance, the system might get lucky and user's first choice determines the target item. We would like to study cases other than such fortuitous outcomes. So we decide to study the *worst-case* behavior of the system, with respect to all items.

2.1 Multi-way search problem

We formalize minimal interaction search as a k-ary search problem over a *hypothesis space* $\mathcal{H} = \{1, \dots, n\}$ of n candidate items. The items belong to m *item categories*. We let S_i be the set of items in the category i and $\mathcal{S} = \{S_1, \dots, S_m\}$ be the set of all categories. The item i belongs to the category j if and only if $i \in S_j$. All items are unique. In other words, no two items belong to the exactly same categories. If this is not true, we can modify our approach such that the target item comprises multiple items, all belonging to the same categories.

A natural way of thinking of our problem is as interaction between the system as a *questioner* and the user as an *answerer*. The questioner asks the answerer *questions*. The question at time t:

$$\mathcal{A}^t = \left\{A_1^t, \dots, A_k^t\right\} \subseteq \mathcal{S} \qquad \left|\mathcal{A}^t\right| = k \tag{1}$$

is a set of item categories of size k. The user has a *target item* o in mind. The user's response at time t is:

$$r_t = \begin{cases} i & o \in A_i^t \\ k+1 & r^t = k+1. \end{cases} \tag{2}$$

If the target item belongs to one of the suggested categories A_i^t, the user chooses this category. Otherwise, the user chooses the $(k+1)$-th category *Other*. If the target item belongs to multiple categories, the user chooses one of these at random.

Let U_t be the *version space*, the set of all items that are consistent with the user's answers up to time t. At the beginning, $U_0 = \mathcal{H}$. Then clearly:

$$U_t = \begin{cases} U_{t-1} \cap A_{r^t}^t & r^t \le k \\ U_{t-1} \cap \overline{\bigcup_{i=1}^k A_i^t} & r^t = k+1, \end{cases} \tag{3}$$

where $\overline{\bigcup_{i=1}^k A_i^t}$ is the category of items that do not belong to any of the categories A_i^t.

The maximum number of questions asked by an algorithm is:

$$\max_{o \in \mathcal{H}} t | U_t = \{o\}. \tag{4}$$

Algorithm 1 Multi-way search with k categories.

Inputs: hypothesis space $\mathcal{H}$, m item categories $\mathcal{S}$, number of item categories in the question k

$U_0 \leftarrow \mathcal{H}$
$t \leftarrow 1$
while $(|U_{t-1}| > 1)$
  compute a question with k item categories:
    $\mathcal{A}^* \leftarrow \arg\max_{\mathcal{A}} f_{U_{t-1}}(\mathcal{A})$
  if (the answerer chooses the category $r_t \leq k$)
    $U_t \leftarrow U_{t-1} \cap A^*_{r^t}$
  else
    $U_t \leftarrow U_{t-1} \cap \overline{\bigcup_{i=1}^{k} A^*_i}$
  $t \leftarrow t + 1$

Output: target item $o \in U_t$

We refer to both the optimal algorithm, the one that minimizes the maximum number of asked questions, and the maximum number of questions that it asks as OPT_k.

For simplicity of exposition, we drop subindexing by time t when the time is clear from the context, or our result applies to all t. As an example, we refer to an item category in the question as A_i and not A^t_i. We implicitly assume that $|\mathcal{A}| = k$.

3 Algorithm

We propose an efficient algorithm for generalized multi-way search with k item categories. The algorithm iteratively asks questions and shrinks the version space until its cardinality is 1. Recall that each question consists of k categories $\mathcal{A} = \{A_1, \ldots, A_k\}$, and a special category $\overline{\bigcup_{i=1}^{k} A_i}$ that covers items that do not belong to any of the categories A_i. The pseudocode of our algorithm is in Algorithm 1.

The question at time t is chosen such that it maximizes the *minimum eliminated space* given the version space U_{t-1}, where the minimum eliminated space is defined as:

$$f_U(\mathcal{A}) = \min \left\{ \begin{array}{c} \left|\bar{A}_1 \cap U\right|, \\ \cdots \\ \left|\bar{A}_k \cap U\right|, \\ |(A_1 \cup \cdots \cup A_k) \cap U| \end{array} \right\}. \quad (5)$$

In other words, we maximize the number of eliminated possibilities irrespective of the behavior of the answerer. Ideally, the maximum of $f_U(\mathcal{A})$, $\frac{k}{k+1}|U|$, is achieved when the size of item categories is $\frac{1}{k+1}|U|$ and the categories do not overlap. For instance, for $k = 1$, the optimal solution is a set that covers one half of U. For $k = 2$, the optimum corresponds to two sets that do not overlap and cover one third of U each. In practice, such a partitioning is unlikely to exist, even in the degenerate case $U = \mathcal{H}$. Nevertheless, a sufficiently good partitioning may still exist.

The main problem in implementing our algorithm efficiently is that the *optimal question*:

$$\mathcal{A}^* = \arg\max_{\mathcal{A}} f_U(\mathcal{A}) \quad (6)$$

Algorithm 2 Greedy search for nearly-optimal questions.

Inputs: version space U, m item categories $\mathcal{S}$, number of item categories in the question k, upper bounds $\mathcal{L}$ on the size of sets in $\mathcal{A}^g$

$\mathcal{A}^g \leftarrow \{\}$
for all $L \in \mathcal{L}$
  choose an active set $\mathcal{S}_L \leftarrow \{S \in \mathcal{S} : |S \cap U| \leq L\}$
  $\mathcal{A}_L \leftarrow$ greedily cover U with k sets from $\mathcal{S}_L$
  if $(f(\mathcal{A}_L) > f(\mathcal{A}^g))$
    $\mathcal{A}^g \leftarrow \mathcal{A}_L$

Output: k question categories $\mathcal{A}^g$

is hard to compute. The first part of $f_U(\mathcal{A})$, $\min_i \left|\bar{A}_i \cap U\right|$, is maximized by smaller sets. The second part, $|(A_1 \cup \cdots \cup A_k) \cap U|$, is maximized by larger sets. In general, the maximization of $f_U(\mathcal{A})$ is an NP-hard problem and it may be necessary to evaluate the function in all subsets of $\mathcal{S}$ of size k, $\binom{m}{k}$ many, to find its maximum. This is clearly infeasible for large values of k.

So we maximize $f_U(\mathcal{A})$ approximately. Our approach is motivated by the following observation. When the size L of the largest set in $\mathcal{A}^*$ is known, the first part of $f_U(\mathcal{A}^*)$, $\min_i \left|\bar{A}^*_i \cap U\right|$, is bounded from below by $|U| - L$, and the maximization of $f_U(\mathcal{A})$ is equivalent to maximizing $|(A_1 \cup \cdots \cup A_k) \cap U|$ subject to $|A_i \cap U| \leq L$. This is a special form of the *maximum coverage problem* (Johnson 1974), and therefore a $(1 - 1/e)$-approximation to its optimal solution can be computed greedily, by choosing item categories that cover most uncovered items. Since L is unknown, we solve several maximum coverage problems, one for each $L \in \mathcal{L}$, and then simply choose the best solution. The pseudocode of our method is in Algorithm 2. In Section 4, we analyze the quality and computation time of approximations obtained by different choices of $\mathcal{L}$.

We refer to our entire algorithm as `DoubleGreedy`. This is because we first iteratively seek greedily optimal partitions (Algorithm 1) and then use a greedy procedure to pick each such partition in the maximum coverage step (Algorithm 2).

4 Analysis

Our main result is summarized in Theorem 1.

Theorem 1. *Algorithm* `DoubleGreedy` *solves the multi-way search problem with k categories. The maximum number of asked questions is $\mathsf{OPT}_k \frac{e}{e-1} \log(n-1)$ and each question is computed in $O(km^2)$ time.*

In this section, we prove this theorem. The proof consists of three steps. First, we argue that Algorithm 1 solves the multi-way search problem in no more than $\mathsf{OPT}_k \log(n-1)$ questions. Second, we propose a question oracle that computes a $(1 - 1/e)$ approximation to $f_U(\mathcal{A}^*)$, the minimum eliminated space by the optimal solution $\mathcal{A}^*$, in $O(km^2)$ time. Third, we combine the two claims and prove Theorem 1. The $\log(n-1)$ and $\frac{e}{e-1}$ factors in the theorem are due to greedily selecting questions and approximating the

best question greedily, respectively. Finally, we propose a question oracle whose running time increases only linearly with the number of item categories m. This oracle is suitable for large-scale problems.

Proposition 1 claims that Algorithm 1 solves the multi-way search problem with k categories in at most $\mathsf{OPT}_k \log(n-1)$ questions. Our result generalizes the worst-case analysis of generalized binary search (Dasgupta, Lee, and Long 2003) to k choices. The challenge in generalizing this result is in finding a suitable notion of good questions, those that divide the version space close to two halves. The minimum eliminated space $f_U(\mathcal{A})$ (Equation 5) is such a notion.

Proposition 1. *Algorithm 1 finds any target item in at most* $\mathsf{OPT}_k \log(n-1)$ *questions.*

Proof: Our proof has two parts. First, we show that in each version space U there exists a question $\mathcal{A}^*$ that eliminates a large portion of that space. Second, we use this fact to bound the maximum number of questions asked by our algorithm.

Let o denote a target item, T be the number of questions asked by OPT_k, and U_t be the version space of OPT_k at time t. Note that the version spaces U_t are nested as:

$$\mathcal{H} = U_0 \supseteq U_1 \supseteq \cdots \supseteq U_{T-1} \supseteq U_T = \{o\} \tag{7}$$

and there are $T+1$ of them. So by the pigeonhole principle, there must exist time t such that:

$$|U_t - U_{t+1}| \geq \frac{1}{T}(|\mathcal{H}|-1) \geq \frac{1}{\mathsf{OPT}_k}(|\mathcal{H}|-1). \tag{8}$$

In other words, there exists a question-answer pair that shrinks the hypothesis space $\mathcal{H}$ by at least $\frac{1}{\mathsf{OPT}_k}(|\mathcal{H}|-1)$. In fact, there must exist a question that shrinks $\mathcal{H}$ by at least $\frac{1}{\mathsf{OPT}_k}(|\mathcal{H}|-1)$ regardless of the answer. This claim can be proved by contradiction. Suppose that $f_{\mathcal{H}}(\mathcal{A}) < \frac{1}{\mathsf{OPT}_k}(|\mathcal{H}|-1)$ for all questions $\mathcal{A}$. Then there exists a sequence of OPT_k answers such that two items in the hypothesis space $\mathcal{H}$ cannot be distinguished in OPT_k questions. This result is in contradiction with the assumption that OPT_k can find any target item in OPT_k questions. So there must exist a question that shrinks $\mathcal{H}$ by at least $\frac{1}{\mathsf{OPT}_k}(|\mathcal{H}|-1)$ regardless of the answer, and indeed $\mathcal{A}^* = \arg\max_{\mathcal{A}} f_U(\mathcal{A})$ is such a question.

This claim generalizes to any subspace $U \subseteq \mathcal{H}$ with the target item o. Hence, the question $\mathcal{A}^*$ always eliminate at least $\frac{1}{\mathsf{OPT}_k}(|U|-1)$ space, and the size of the version space after t questions is bounded from above as:

$$\left(1 - \frac{1}{\mathsf{OPT}_k}\right)^t (|\mathcal{H}|-1) + 1 \leq \exp\left[-\frac{1}{\mathsf{OPT}_k}\right]^t (|\mathcal{H}|-1) + 1. \tag{9}$$

The upper bound falls below 2 when $t \geq \mathsf{OPT}_k \log(|\mathcal{H}|-1)$. So any target item can be found in at most $\mathsf{OPT}_k \log(|\mathcal{H}|-1)$ steps. ■

4.1 Question oracle $\mathcal{L}_{\mathcal{S}}$

Unfortunately, the optimal question $\mathcal{A}^* = \arg\max_{\mathcal{A}} f_U(\mathcal{A})$ is hard to compute (Section 3). In the following proposition, we show that if the cardinality of the largest set in $\mathcal{A}^*$ is known, a $(1-1/e)$-approximation to $f_U(\mathcal{A}^*)$ can be computed greedily. In this case, the maximization of $f_U(\mathcal{A})$ can be formulated and solved as the *maximum coverage problem*.

Proposition 2. *Let* $L = \max_i |A_i^* \cap U|$ *be the cardinality of the largest set in the best question* $\mathcal{A}^*$ *and* A^L *be a greedy solution to the maximum coverage problem with sets:*

$$\mathcal{S}_L = \{S \in \mathcal{S} : |S \cap U| \leq L\}.$$

Then:

$$f_U(\mathcal{A}^L) \geq (1-1/e) f_U(\mathcal{A}^*).$$

Proof: Note that the greedy solution A^L to the maximum coverage problem (Johnson 1974) satisfies:

$$\left|\bigcup_{i=1}^{k}(A_i^L \cap U)\right| \geq (1-1/e)\left|\bigcup_{i=1}^{k}(A_i^* \cap U)\right| \tag{10}$$

because it is a $(1-1/e)$-approximation. Moreover, since the largest set in $\mathcal{A}^L$ cannot be larger than that in $\mathcal{A}^*$, we know that:

$$\max_i \left|A_i^L \cap U\right| \leq \max_i |A_i^* \cap U|. \tag{11}$$

Therefore, the cardinality of all complements $\bar{A}_i^L$ is bounded from below as:

$$\begin{aligned} \min_i \left|\bar{A}_i^L \cap U\right| &= |U| - \max_i \left|A_i^L \cap U\right| \\ &\geq |U| - \max_i |A_i^* \cap U| \\ &= \min_i \left|\bar{A}_i^* \cap U\right| \\ &\geq (1-1/e) \min_i \left|\bar{A}_i^* \cap U\right|. \end{aligned} \tag{12}$$

Finally, the two claims can be combined as:

$$\begin{aligned} &\min\left\{\min_i \left|\bar{A}_i^L \cap U\right|, \left|\bigcup_{i=1}^{k}(A_i^L \cap U)\right|\right\} \\ &\geq (1-1/e)\min\left\{\min_i \left|\bar{A}_i^* \cap U\right|, \left|\bigcup_{i=1}^{k}(A_i^* \cap U)\right|\right\}, \end{aligned} \tag{13}$$

which yields the final inequality $f_U(\mathcal{A}^L) \geq (1-1/e) f_U(\mathcal{A}^*)$. ■

Unfortunately, the size of the largest set in the optimal solution $\mathcal{A}^*$ is unknown. Therefore, we suggest solving the maximum coverage problem for several candidate values of L and then choose the best result (Algorithm 2). Algorithm 2 finds a $(1-1/e)$-approximation to $f_U(\mathcal{A}^*)$ when the candidate values comprise all set sizes in $\mathcal{S}$.

Proposition 3. *Let the upper bounds in Algorithm 2 be:*

$$\mathcal{L}_{\mathcal{S}} = \{|S \cap U| : S \in \mathcal{S}\}.$$

Then Algorithm 2 returns a question A^g *such that:*

$$f_U(\mathcal{A}^g) \geq (1-1/e) f_U(\mathcal{A}^*)$$

in $O(km^2)$ *time.*

Proof: First, note that the returned question A^g satisfies $f_U(\mathcal{A}^g) \geq f_U(\mathcal{A}^L)$ for all $L \in \mathcal{L}_\mathcal{S}$. Since $\mathcal{L}_\mathcal{S}$ comprises all possible set sizes, the size of the largest set in the optimal question $\mathcal{A}^*$ must be in $\mathcal{L}_\mathcal{S}$. By Proposition 2, $f_U(\mathcal{A}^g) \geq (1-1/e) f_U(\mathcal{A}^*)$.

Second, Algorithm 2 solves m maximum coverage problems, each of which is solved approximately in $O(km)$ time. Hence, the time complexity of the algorithm is $O(km^2)$. ■

Finally, we are ready to prove Theorem 1, which is our main result.

Proof: Let $\mathcal{L}_\mathcal{S}$ (Proposition 3) be the upper bounds in Algorithm 2. Then $f_U(\mathcal{A}^g) \geq (1-1/e) f_U(\mathcal{A}^*)$ for all U because Algorithm 2 computes a $(1-1/e)$-approximation to $f_U(\mathcal{A}^*)$.

The first claim of Theorem 1 can be proved by substituting $\mathcal{A}^g$ for $\mathcal{A}^*$ in Proposition 1. The questions $\mathcal{A}^g$ are not as discriminative as $\mathcal{A}^*$. However, they are no more than $(1-1/e)$ worse, and thus we can guarantee that $\frac{1}{\mathsf{OPT}_k} \frac{e-1}{e} (|U|-1)$ items in the version space U get eliminated after each question. Therefore, the maximum number of asked questions increases to $\mathsf{OPT}_k \frac{e}{e-1} \log(n-1)$.

The second claim follows from Proposition 3. ■

4.2 Loglinear oracle $\mathcal{L}_\alpha$

Algorithm 2 computes a $(1-1/e)$-approximation to $f_U(\mathcal{A}^*)$ in $O(km^2)$ time (Proposition 3). Since m is often large, this oracle may be computationally expensive in practice. In this section, we propose an oracle than be computed in $O(m)$ time.

The main idea is to choose the upper bounds $\mathcal{L}$ (Algorithm 2) such that $|\mathcal{L}| = o(m)$, so the computation time decreases to $o(km^2)$. In the following proposition, we show that when the upper bounds $\mathcal{L}$ are spaced on the log scale between 1 and $|U|$, Algorithm 2 computes a $\alpha(1-1/e)$-approximation to $f_U(\mathcal{A}^*)$ in $O(km \log_{1/\alpha}(n))$ time. The parameter α trades off the quality of the approximation for its computation time.

Proposition 4. *Let the upper bounds in Algorithm 2 be:*

$$\mathcal{L}_\alpha = \left\{ |U|\,(1-\alpha^i) : i \in \mathbb{N} \;\wedge\; \alpha^i \geq |U|^{-1} \right\},$$

where $\alpha \in (0,1)$ is a tunable parameter. Then Algorithm 2 returns a question A^g such that:

$$f_U(\mathcal{A}^g) \geq \alpha(1-1/e) f_U(\mathcal{A}^*)$$

in $O(km \log_{1/\alpha}(n))$ time.

Proof: Let $L = \max_i |A_i^* \cap U|$ be the cardinality of the largest set in the optimal question $\mathcal{A}^*$ and $j \in \mathbb{N}$ be an integer such that:

$$L_{j-1} = |U|\,(1-\alpha^{j-1}) \leq L \leq |U|\,(1-\alpha^j) = L_j. \quad (14)$$

Then along the lines of Proposition 2:

$$\left| \bigcup_{i=1}^{k} (A^{L_j} \cap U) \right| \geq (1-1/e) \left| \bigcup_{i=1}^{k} (A_i^* \cap U) \right|. \quad (15)$$

Based on Equation 14, it follows:

$$|U| - L_j = \alpha(|U| - L_{j-1}) \geq \alpha(|U| - L) \quad (16)$$

Movie tags			
55.43%	Drama	31.81%	Comedy
25.91%	Independent film	23.58%	Murder
21.16%	Romance	21.15%	Thriller
19.99%	Beautiful woman	18.38%	Action
16.88%	Based on novel	16.52%	Crime
Book tags			
66.12%	Fiction	37.35%	Non-fiction
31.09%	Novel	17.95%	Mystery
17.24%	History	16.54%	Series
16.41%	Literature	14.79%	Fantasy
14.65%	20th century	13.13%	Children's

Figure 2: 10 largest item categories in the movie and book datasets.

and therefore we can prove:

$$\min_i \left| \bar{A}_i^{L_j} \cap U \right| \geq |U| - L_j \geq \alpha \min_i \left| \bar{A}_i^* \cap U \right|. \quad (17)$$

The two claims can be combined as:

$$\min \left\{ \min_i \left| \bar{A}_i^{L_j} \cap U \right|, \left| \bigcup_{i=1}^{k} (A_i^{L_j} \cap U) \right| \right\} \quad (18)$$
$$\geq \alpha(1-1/e) \min \left\{ \min_i \left| \bar{A}_i^* \cap U \right|, \left| \bigcup_{i=1}^{k} (A_i^* \cap U) \right| \right\}$$

and yield $f_U(\mathcal{A}^{L_j}) \geq \alpha(1-1/e) f_U(\mathcal{A}^*)$. Our main claim follows from the fact that $f_U(\mathcal{A}^g) \geq f_U(\mathcal{A}^{L_j})$ for all L_j.

The size of the set $\mathcal{L}_\alpha$ is bounded from above by $\log_{1/\alpha}(|U|)$ since α^i is less than $|U|^{-1}$ for $i > \log_{1/\alpha}(|U|)$. Therefore, Algorithm 2 solves $O(\log_{1/\alpha}(n))$ maximum coverage problems, each of which is solved approximately in $O(km)$ time. Hence, the time complexity of the algorithm is $O(km \log_{1/\alpha}(n))$. ■

Proposition 4 shows that $\alpha(1-1/e)$-approximate questions can be computed in $O(km \log_{1/\alpha}(n))$ time, linear in the number of item categories m. This result is pretty practical. Suppose that $n = 10^5$, $m = 10^3$, and $\alpha = 0.9$. Then the oracle in Proposition 4 computes almost as good questions as the one in Proposition 3 in 10 percent of time. When $\alpha = 0.95$, the new oracle is 5 times faster.

5 Experiments

We evaluate our method for generating questions (Algorithm 2) and show that it yields good partitioning of the hypothesis space $\mathcal{H}$. In addition, we show that as the number of item categories k increases, algorithm `DoubleGreedy` can find the target item in fewer questions.

5.1 Datasets

Our approach is evaluated on two datasets. The first dataset are 91k *movies* from the Internet Movie Database (IMDb) (imd 2012). The movies belong to 26 genres and their plots are described by 87k keywords. We define 326 item categories $\mathcal{S}$, 26 genres and 300 most frequent keywords (Figure 2). We remove movies that belong to less than 5 categories and those that belong to the same categories as another movie. We end up with a dataset of 38k unique items.

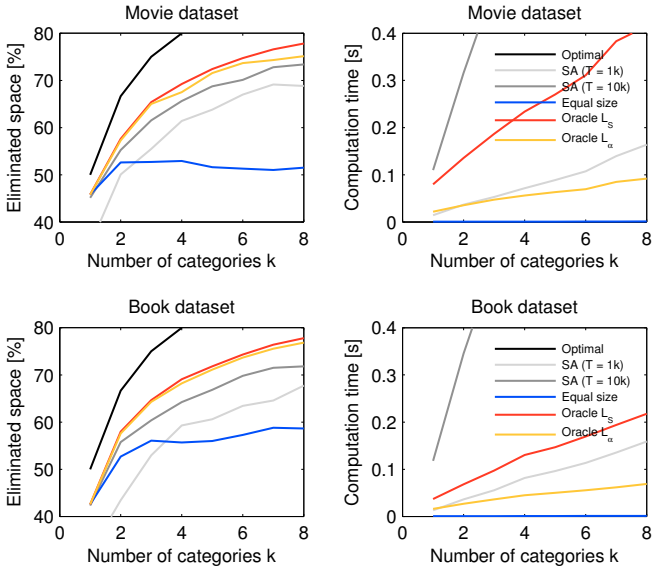


Figure 3: Comparison of five methods for partitioning the hypothesis space into k item categories. We report the minimum eliminated space $f_U(\mathcal{A})$ and computation time. The optimum is depicted by the black line.

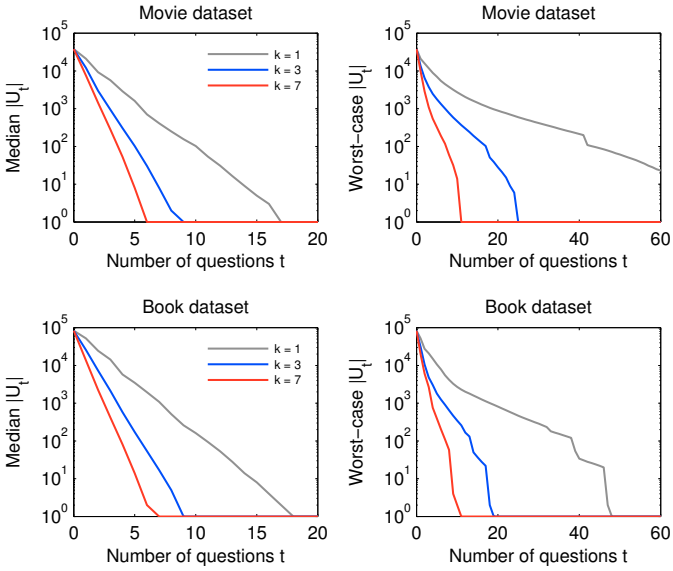


Figure 4: The median and worst-case cardinality of the version space U_t after algorithm `DoubleGreedy` asks t questions.

The average number of categories per movie is 14.5 and the maximum is 106.

The second dataset are 186k *books* from the online library of books LibraryThing (lib 2012). The books are described by 211k keywords. We define 300 item categories $\mathcal{S}$, one for each of the 300 most frequent keywords (Figure 2). We remove books that belong to less than 5 categories and those that belong to the same categories as another book. We end up with a dataset of 83k unique books. The average number of item categories per book is 10 and the maximum is 26.

5.2 Question oracles

In the first experiment, we compare our question oracles, $\mathcal{L}_{\mathcal{S}}$ (Section 4.1) and $\mathcal{L}_\alpha$ (Section 4.2), to two baselines on up to 8 item categories in the question. The methods are evaluated by the amount of eliminated space $f_U(\mathcal{A})$ and their computation time. We assume that $\alpha = 0.9$.

The first baseline is simulated annealing (SA) (Kirkpatrick, Gelatt, and Vecchi 1983). Simulated annealing is a standard approach to combinatorial optimization, such as maximizing $f_U(\mathcal{A})$. Essentially, it is a random walk that transitions into a new state proportionally to the difference between the energy of the new and old states. Our energy function is defined as $H(\mathcal{A}) = -\frac{1}{|U|} f_U(\mathcal{A})$ and the transition between the states $\mathcal{A}^t$ and $\mathcal{A}^{t+1}$ is accepted with probability:

$$P(\mathcal{A}^{t+1}|\mathcal{A}^t) = \exp[(H(\mathcal{A}^t) - H(\mathcal{A}^{t+1}))/\sigma_t], \qquad (19)$$

where σ_t is the temperature at time t. The temperature σ_t cools off according to the schedule $\sigma_t = \alpha^t$, where $\alpha = \exp[\log(10^{-3})/T]$ and T is the number of annealing steps. Hence, σ_t decreases at an exponential rate from 1 to 10^{-3} in T steps. The new state $\mathcal{A}^{t+1}$ is obtained by changing u categories in $\mathcal{A}^t$ with probability 2^{-u}. We tuned all parameters of the annealing such that it performs the best. The second baseline chooses k item categories that are closest in size to $\frac{1}{k+1}|U|$. This method is very fast, and performs well when the categories in U are mutually exclusive and their sizes are close to $\frac{1}{k+1}|U|$.

Our results are reported in Figure 3. The results are averaged over 30 subsets U of movies and books, which correspond to 30 largest item categories in each domains. We observe two major trends.

First, our oracles consistently eliminate more space $f_U(\mathcal{A})$ than all baselines. The eliminated space increases with the number of item categories k and is about 85% of the theoretical maximum $\frac{k}{k+1}|U|$. This maximum is likely unattainable in our domains.

Second, our oracles are very fast. In particular, the oracle $\mathcal{L}_\alpha$ generates all questions in less than 0.1 seconds. This computation time is comparable to SA with $T = 10^2$, which yields about 10% worse questions, and up to 10 times smaller than SA with $T = 10^3$, which is still inferior to our method. The oracle $\mathcal{L}_\alpha$ generates only slightly worse questions than $\mathcal{L}_{\mathcal{S}}$ and is up to four times faster. This result is consistent with our analysis in Section 4.2. We note that the equal size heuristic is fast but the quality of its output drops dramatically as k increases. The reason is that the item categories in the question start overlapping. As a result, the term $\bigcup_{i=1}^k A_i$ is small, and so is the minimum eliminated space $f_U(\mathcal{A})$.

5.3 Generalized search

In Section 5.2, we showed that questions with more item categories k eliminate more items in the hypothesis space $\mathcal{H}$. In this section, we demonstrate that this results in faster discovery of target items. In particular, we study the median and worst-case behavior of algorithm `DoubleGreedy` while

varying k. This experiment is conducted on 1000 randomly chosen target items, both in the movie and book datasets. We simulate the answerer and the answerer does not lie. If the target item belongs to multiple item categories in the question $\mathcal{A}^g$, we randomly choose one of these as the answer. Our results are summarized in Figure 4.

All trends in Figure 4 support our hypothesis that more item categories k lead to faster discovery. For example, for $k = 1$, which is essentially binary search, the median movie is found in 17 questions. As the number of item categories k increases to 3 and 7, the median movie is identified in 9 and 7 questions, respectively. So the total number of asked questions decreases by 47% and 59%. Similar trends can be observed in the worst case. For instance, many movies and books belong to only a few categories, and cannot be found by binary search in less than 50 questions. As the number of item categories k increases to 7, even these items are found within 15 questions.

6 Related work

There are several aspects of previous work related to ours. First is the multi search problem. The basic problem of binary search involves a single category with say n values and has a text book solution that involves $\log n$ 2-way comparison queries that repeatedly divide the space of possibilities into 2 equal ranges. The area of combinatorial search (Aigner 1988) studies such problems and their extensions, including to case where some of the queries might return incorrect answers. The most natural generation of this to multiple categories each of which can take many values. Database research on this problem involves designing different indexing structures for such problems, and involves partitioning the underlying space with cuts that divide the set of points as evenly as possible into a small k subranges. In this line of research, each "query" is typically is a comparison or range query. This line of research differs from our problem here where each category is binary, the focus is on dividing the space of all category combinations and each "query" reveals presence of absence of a category in the ultimate target item.

A search problem that is similar to ours was studied before in learning theory and is known as generalized binary search (GBS) (Dasgupta 2005; Nowak 2011). In GBS, the objective is to learn a policy that minimizes the number of questions to find a target hypothesis in the hypothesis space. This problem was analyzed both in the worst and average cases (Dasgupta, Lee, and Long 2003; Dasgupta 2005), and it is known that GBS asks $\log(n)$ times more questions than the optimal solution. Our problem differs from GBS in two aspects. First, we ask questions with k item categories but only get feedback for one of them. In particular, since the categories are not mutually exclusive, we cannot conclude that the target item does not belong to any of the other $k - 1$ categories. In other words, we search with partial feedback. Second, the number of questions in our problem is $\binom{m}{k}$, which is exponential in k. If each item category was a question, then GBS operates on m questions, which is m^{k-1} times less than in our problem. Finally, note that GBS is a special case of our algorithm when $k = 1$.

7 Conclusions

In this paper, we initiate the study of *minimal interaction search*. In this problem, the answerer is given k options, represented by k item categories, and asked to select the one that matches the target item, or state that none does. We formalize the problem as generalized k-way search and propose an algorithm `DoubleGreedy` that solves it efficiently. In addition, we analyze the solution and show that we can find the target item in at most $\mathsf{OPT}_k \frac{e}{e-1} \log(n)$ questions in the worst case. Finally, we evaluate our solution on two datasets of movies and books, and show that interactive search can be sped up significantly when the user is given more options.

Our worst-case analysis can be relatively easily extended to the average case. In particular, let's assume that there exists some *a priori* known probability distribution π over item categories, and we want to find a sequence of questions that minimizes the expected number of asked questions with respect to π. Then we can apply the result of Golovin and Krause (Golovin and Krause 2011), and show that the greedy partitioning of the hypothesis space is still a logarithmic factor approximation to the optimum. Similarly, we can generalize our maximum coverage oracles to probabilistically weighted items. Overall, we still get the same approximation factor to the optimum as in the worst case.

References

Aigner, M. 1988. *Combinatorial Search*. New York, NY: John Wiley & Sons.

Dasgupta, S.; Lee, W. S.; and Long, P. 2003. A theoretical analysis of query selection for collaborative filtering. *Machine Learning* 51(3):283–298.

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

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

2012. The Internet Movie Database (IMDb). http://www.imdb.com/.

Johnson, D. 1974. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences* 9(3):256–278.

Karbasi, A.; Ioannidis, S.; and Massoulié, L. 2012. Hot or not: Interactive content search using comparisons. In *2012 Information Theory and Applications Workshop*, 291–297.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. Optimization by simulated annealing. *Science* 220(4598):671–680.

2012. LibraryThing. http://www.librarything.com/.

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