Seeker: Real-Time Interactive Search

Ari Biswas
Amazon.com
Seattle, WA
aritrb@amazon.com

Thai T. Pham
Amazon.com
Seattle, WA
phamtha@amazon.com

Michael Vogelsong
Amazon.com
Seattle, WA
vogelson@amazon.com

Benjamin Snyder*
ben.snyder@gmail.com

Houssam Nassif
Amazon.com
Seattle, WA
houssamn@amazon.com

ABSTRACT

This paper introduces Seeker, a system that allows users to adaptively refine search rankings in real time, through a series of feedbacks in the form of likes and dislikes. When searching online, users may not know how to accurately describe their product of choice in words. An alternative approach is to search an embedding space, allowing the user to query using a representation of the item (like a tune for a song, or a picture for an object). However, this approach requires the user to possess an example representation of their desired item. Additionally, most current search systems do not allow the user to dynamically adapt the results with further feedback. On the other hand, users often have a mental picture of the desired item and are able to answer ordinal questions of the form: "Is this item similar to what you have in mind?" With this assumption, our algorithm allows for users to provide sequential feedback on search results to adapt the search feed. We show that our proposed approach works well both qualitatively and quantitatively. Unlike most previous representation-based search systems, we can quantify the quality of our algorithm by evaluating humans-in-the-loop experiments.

CCS CONCEPTS

• **Information systems → Search interfaces**; Probabilistic retrieval models; Information retrieval diversity; Test collections; Relevance assessment; • **Computing methodologies → Online learning settings**; **Active learning settings**; *Discrete space search*; *Search with partial observations*; Sequential decision making.

KEYWORDS

Interactive Search, Real Time Recommendation, Online Learning, Active Learning, Multi-Armed Bandit

ACM Reference Format:
Ari Biswas, Thai T. Pham, Michael Vogelsong, Benjamin Snyder, and Houssam Nassif. 2019. Seeker: Real-Time Interactive Search. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19), June 22–24, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

*Work done while at Amazon

1 INTRODUCTION

Search engines and online shopping websites maintain indices with millions of items. Often, it is difficult for a user to accurately describe in words what they are looking for [34]. Even if the user is able to describe their target item effectively, large index and catalog sizes mean it is difficult to sift through similar items efficiently.

Consider the situation in which a user is searching for a new movie to watch. They have a mental representation of the characteristics of the movie they would enjoy but are not acquainted with the genre keywords, latest movies, actors or directors. Being unfamiliar with current movie jargon, they are unable to accurately describe their preferred movie with a traditional keyword interface, nor do they have an example photograph. However, if we show the same user another movie they have seen and ask them *"Is this movie similar to the one they have in mind? Yes or no?"*, people can answer such ordinal questions with less noise than absolute judgments – i.e. finding the exact words to describe their choice [30].

The above scenario is not restricted to movies only. In the case of browsing for a song on a media platform, searching for a news article on a news website, or a dress on an online platform, the user may not be able to accurately describe the desired item in a traditional keyword interface. But users could provide relative judgments based on what they have experienced before. For example, answers to queries like *"Songs similar to Heroes by David Bowie: Yes or no?"* or *"News similar to that of the Queen's involvement with Brexit: Yes or no?"* are easier to provide.

In addition, traditional search engines [21, 29] and the newer representation search systems (described in Section 2) are temporally static. The engines use text or imagery as the query and respond with a ranked list of results. This ranking is based on an estimate of relevance to the user in their current context – location, historical searches etc. They do not provide the user the opportunity to adapt and fine-tune the resulting page with additional feedback. In traditional engines, for a given user in a given session, each query is independent of each other. Figure 1 illustrates the difference between traditional engines and our setting.

In this paper, we describe our system, **Seeker**, that dynamically refines search results based on real-time interactions with the user (in the form of likes and dislikes) within a single search session. From a customer perspective, this system adds the feeling of an "in-store" shopping discovery experience, with a personal curator.

In our setting, the user scrolls through a page of items and may "like" or "dislike" any item at any time. The data gathered from these preferences is used to update the list of results shown in real-time, thereby iteratively closing in on what they are looking for. To our knowledge, Seeker is the first interactive and dynamic search experience which enables the user to seamlessly *zoom in*, *zoom out*, and *pivot* by scrolling up and down and selecting items to like and dislike in an adaptive manner.

In this work we make the following contributions:

- Introduce Seeker, an interactive recommendation algorithm deployed at scale, which adapts to customer inputs in real time.
- Propose a novel evaluation metric with humans in the loop that allow us to quantify the quality of our proposed algorithm and evaluate it against other methods. Most embedding-based representational search engines in the past have evaluated their systems only qualitatively rather than quantitatively. In our experiments, we simulate the tasks of searching for a particular item, and quantifiably measure progress.

The paper is organized as follows. In Section 2, we review related papers and search engines. In Section 3, we describe how we model human preferences expressed in likes and dislikes and translate those preferences into probability distributions over our catalog. In Section 4, we present our adaptive algorithm for making real time recommendations. In Section 5, we evaluate Seeker's results. In Section 6, we discuss directions for future research. In Section 7, we summarize our work.

2 RELATED WORK

Over the last few years, there has been a growing trend of exploring new interfaces beyond traditional keyword search, and in particular, visual-based search [8]. In [10], users query relevant items by uploading real world photographs of clothing. The engine then displays results that are visually similar to the query photograph. Pinterest built a system which allowed users to hover over pins and find visually similar items in the catalog [17]. An advantage of these systems is that they help people find things using an understanding they might not be able to put into words.

Many lines of research focused on learning the relative similarities of images. They accomplish this by mapping each image to a numerical vector, so they can capture the visual similarities in Euclidean space [3, 19, 22, 38, 41]. Using the similar approach but in a scalable manner, companies have also rolled out their visual search platforms, from Google Goggles, Google Similar Images, and Amazon Flow to Microsoft (Bing) [13], Pinterest [18], eBay [39], and Alibaba [40].

All these methods, however, require the user to provide a photograph of the targeted item. They fail when users do not have an actual visual representation of the desired item, but instead a mental picture of it. The users themselves may not know how to properly describe their mental visualization in words. Our algorithm addresses this issue, as Seeker is able to work with any embedding representation, including visual, textual and audio.

Moreover, Seeker dynamically adjusts the search results based on interactive user feedback; all mentioned projects do not allow users to fine-tune their current-session search with additional feedback. While we use proprietary embeddings in the examples of this paper, the underlying engine can operate upon features derived from other domains (or combinations of domains) as well – customer behavior, language understanding, audio, etc.

3 PROBLEM FORMULATION

3.1 The Setting

Figure 1 illustrates how Seeker is different from traditional search. The user starts with a ranked list of results and provides feedback in the form of likes or dislikes; the search engine then generates a new set of ranked results, updating the page in real-time. In this section, we define the notation to formally describe the above search process.

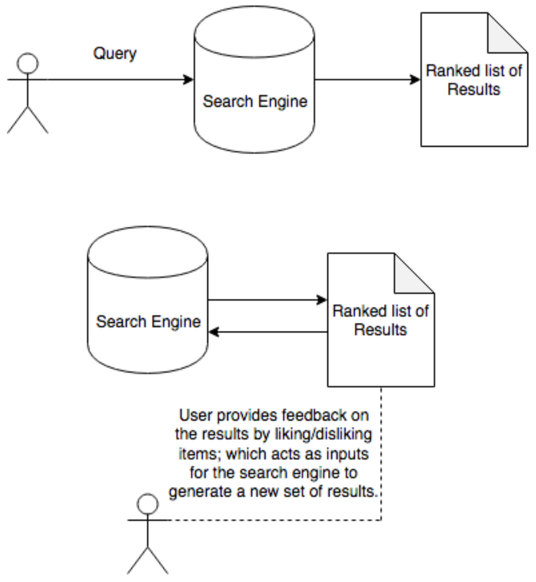
Query
Search Engine
Ranked list of Results
Search Engine
Ranked list of Results
User provides feedback on the results by liking/disliking items; which acts as inputs for the search engine to generate a new set of results.

Figure 1: The top figure describes a traditional search engine: A user submits a query to the engine and is presented with a ranked list of items. Our system (bottom figure) dynamically incorporates feedback on results in real-time, generating a new ranked list with each like or dislike.

Assume that we have a catalog of N items, out of which $M \ll N$ can be displayed. We model user feedback as a sequence of likes and dislikes over discretized timesteps $t_0, t_1, \cdots, t_k$. The user starts with an initial ranking of items at timestep t_0. This initial t_0 ranking can be thought of as Seeker's prior belief on what the user desires, can be generated from a traditional search or recommendation engine, and may incorporate diversity or business requirements.

The user interacts with the page by *liking* or *disliking* items. At each timestep, t_k, Seeker produces a new ranked list of results, based on the feedback from $t_0, \cdots, t_{k-1}$. It does so by constructing a discrete probability distribution over the catalog of N items at each timestep. The probability distribution represents the likelihood of an item being the user's desired item.

We featurize each catalog item i by embedding it into a vector space $x_i \in \mathbb{R}^d$. Seeker requires a high correlation between human perception of similarity and distance metric in the embedded vector space. Based on the properties of the items displayed, embedding strategies described in [9, 20, 25, 31] have been shown to correlate with human perception.

Seeker can be divided into three major components, as seen in Figure 2. Section 3.2 describes how we convert likes and dislikes to preference pairs and probability distributions. Section 4.1 details how we use preference pairs to estimate a target's likelihood. Section 4.4 shows how we use probabilistic sampling to recommend items to users at each timestep.

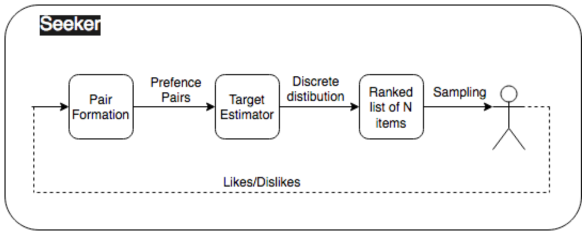

Figure 2: Logical components of Seeker

3.2 Pairwise Comparison

Let $a_i^k \in \mathbb{R}^d$ for $i = 1, ..., p$ be the vector representations of the liked items, and $b_j^k \in \mathbb{R}^d$ for $j = 1, ..., q$ the vector representations of the disliked items. We will drop the superscript when the context is clear. Let $A := \{a_1, ..., a_p\}$ and $B := \{b_1, ..., b_q\}$ be the non-empty subsets of $\{x_1, ..., x_N\}$. We define s_{ij} as the preference pair which consists of a liked item a_i and a disliked item b_j from sets A and B respectively. We create pq preference pairs from all cross-pairings between the p likes and q dislikes.

The intuition behind preference pairs follows from our assumption that the user has some ideal item t (referred to as the target) in their mind that they wish to find. Then $s_{ij} \in S$ represents the preference that the user thinks item i is more similar to their desired item t than item j, i.e *they prefer i over j given t*:

$$||x_i - x_t||^2 < ||x_j - x_t||^2. \tag{3.1}$$

Equation 3.1 resolves to item t being spatially closer to item i than it is to item j. In this paper we use the Euclidean distance to measure vector similarity, but Seeker is agnostic to the metric used.

We use preference pairs to model the probability of a catalog item being the hidden target item t, featurized as x_t. If we were to present a user with item x_i and item x_j, what is the probability that they chose i over j? Questions of this form are known as triplets in the Machine Learning literature [15, 27]. Equation 3.2 mathematically models our question:

$$\mathbb{P}\big(s_{ij}|t, i, j\big) = \frac{1}{1 + \exp\big\{-\alpha\big(||x_j - x_t||^2 - ||x_i - x_t||^2\big)\big\}}, \tag{3.2}$$

where $\alpha \geq 0$.

Intuitively, the answer to the above triplet question should depend on how similar items x_i and x_j are to x_t. As similarity and distance are equivalent in our world, the probability of preferring i over j becomes a function of how close x_i and x_j are to x_t. According to this model (and Equation 3.2), if items x_i and x_j are equidistant from the target x_t, then they are equally preferred, and the probability of choosing i over j is 0.5. If x_i is the target x_t while x_j is infinitely far away, the probability of choosing i becomes 1. For items in the middle we get a smooth noise model that accounts for the stochasticity in human decisions.

Our model includes a preference hyperparameter α, which represents our confidence in the vector space representation:

- When $\alpha = 0$, then $\mathbb{P}\big(S_{ij}|t\big) = 0.5$ for all combinations of targets, likes and dislikes. This means our embeddings have no correlation with human judgment of similarity, and preferring i over j is as good as a fair coin flip.
- When $\alpha = \infty$, then $\mathbb{P}\big(S_{ij}|t\big) = 1$. This removes randomness from the decision process, perfectly aligns our representation of human judgment with the metric distance, and deterministically picks the closer item.

We use $\alpha = 1$ for the results discussed in Section 5.

4 ITEM RANKING

In this section we describe how we go from preference pairs and a noise model to a ranked list of items to be displayed to the user.

4.1 Target Estimation

To keep our notation consistent, we always assume that the user prefers item x_i to x_j when we write $s_{ij} \in S$. We make the further assumption that each preference pair is independent from each other. This is a simplifying assumption which serves as a good baseline [6, 12]. In Section 6.1, we investigate ways to drop the independence assumption. Equation 4.1 represents the joint distribution likelihood of observing preferences S, given target item t and likes and dislikes sets A and B:

$$\mathbb{P}(S|t, A, B) = \prod_{s_{ij} \in S} \mathbb{P}\big(s_{ij}|t, i, j\big), \tag{4.1}$$

where $\mathbb{P}(s_{ij}|t, i, j)$ is defined as in Equation 3.2. The log-likelihood becomes:

$$\log \mathbb{P}(S|t, A, B) = \sum_{s_{ij} \in S} \log \mathbb{P}\big(s_{ij}|t, i, j\big). \tag{4.2}$$

We do not know a priori what is the hidden target t. Our goal is to find t or approximate it. We note that t may not be present in our catalog, and in this case our goal is to find an item as similar to t as possible. In order to build a probability distribution over our catalog, we borrow ideas from [33]. We use the same noise model, but apply it to recommend items to the user, instead of learning a metric space. For each catalog item, we compute the log-likelihood mass of that item being the target, given the user's likes and dislikes, as shown in Algorithm 1.

Algorithm 1 Catalog items log-likelihood computation

```
scores = [ ]
for all items $t = 1, \cdots, N$ do
    score = 0
    for all $s_{ij} \in S$ do
        score += $\log \mathbb{P}\big(t|s_{ij}\big)$
    scores.append(score)
```

4.2 Posterior Construction

Instead of presenting items according to their likelihood of being the target, we allow for the inclusion of priors into our model. Let $\mathbb{P}(i)$ be the prior probability of item i being the actual desired target. One can compute such priors using traditional search engines, and personalize them using the user's browsing or purchase history [34].

Given priors $\mathbb{P}(i)$, the posterior probability of an item being the target is:

$$\mathbb{P}(t|S) \propto \mathbb{P}\big(S|t\big)\mathbb{P}(t), \tag{4.3}$$

and the log-posterior becomes:

$$\begin{aligned} \log \mathbb{P}(t|S) &\propto \log \mathbb{P}\big(S|t\big) + \log \mathbb{P}(t) \\ &= \sum_{s_{ij} \in S} \log \mathbb{P}\big(s_{ij}|t, i, j\big) + \log \mathbb{P}(t). \end{aligned} \tag{4.4}$$

At each time step the user provides feedback causing the size of S to grow. Therefore the log likelihood will eventually dominate the posterior density score. In the early stages when we have fewer likes and dislikes, our posterior belief on the target is dominated by a well founded prior. This prevents us from having to wait a long time before showing meaningful results.

4.3 Items Recommendation

We consider four different ways to display $M \ll N$ items to the user:

4.3.1 Pure Exploitation/Noiseless. The simplest approach is to sort the posteriors and recommend the top M items. Theoretically, this prevents us from exploring the search space. Practically, this leads to a poor user experience with limited product diversity.

4.3.2 Pure Exploration/Random. The other extreme solution is to show random results all the time, completely ignoring the posterior densities.

4.3.3 Epsilon-greedy. Another approach is to randomize some of the results while leaving the others untouched, as in Algorithm 2. We rank items by their posterior densities, and replace each item with a random item with probability ϵ. See [4] for a detailed study of ϵ-greedy algorithms.

Algorithm 2 Epsilon-greedy sampling

```
Input: $0 \le \epsilon \le 1$
    noiseless = argsort { $\mathbb{P}(t|S)$ } in descending order
    results = [ ]
    for all $x_i$, $i \in$ noiseless do
        flip $\epsilon$ biased coin
        if heads then
            $x \sim \text{Unif}(x_1, \cdots, x_N)$
            results = results $\cup$ $x$
        else
            results = results $\cup$ $x_i$
    return results
```

4.4 Boltzmann Exploration

The fourth method to recommend items involves sampling without replacement according to the item's posterior densities. Let g_j be a score associated with item j. A popular way to generate a discrete distribution over the items is by using the exponential weighing scheme, known as the softmax or Boltzmann equation:

$$p_j = \frac{e^{g_j}}{\sum_{i=1}^{N} e^{g_i}}. \tag{4.5}$$

Here, p_j is our belief probability that item j is the true target. Even though g is unconstrained in $\mathbb{R}$, common values are $g_j = \mathbb{P}(x_j|S)$ and $g_j = \log \mathbb{P}(x_j|S)$, the latter resulting in polynomial weighing [16, 32]. Note that if the items were equally spaced, sampling from the discrete distribution p_j is asymptotically equivalent to sampling from the hidden continuous distribution, as we show in Appendix A.

Sampling without replacement when N is large can prove to be very slow. When N and d are large, normalizing our posterior densities can lead to precision issues with sampling. We can overcome this problem by using the *Gumbel-Max* trick [23], which shows that adding standard Gumbel noise to g_i and taking the max is equivalent to sampling according to Boltzmann (Equation 4.5):

$$\underset{j}{\text{argmax}}\{g_j + Gumbel(0,1)\} \sim \frac{e^{g_j}}{\sum_{i=1}^{N} e^{g_i}} = p_j. \tag{4.6}$$

We sketch the proof for completeness. Let $z_i = g_i + Gumbel(0, 1)$. By the additive property, $z_i \sim Gumbel(g_i, 1)$, with probability density function (PDF):

$$f_i(z) = e^{-\left(z - g_i + e^{-(z-g_i)}\right)}, \tag{4.7}$$

and cumulative distribution function (CDF):

$$F_i(z) := \mathbb{P}(z_i \le z) = e^{-e^{-(z-g_i)}}. \tag{4.8}$$

Proof. Define by $\mathbb{P}(j^*)$ the probability that z_j is the largest among all z_i. We have:

$$\begin{aligned} \mathbb{P}(j^*) &= \int_{z_j=-\infty}^{+\infty} f_j(z_j) \prod_{i \ne j} \mathbb{P}(z_i \le z_j)\, dz_j \\ &= \int_{z_j=-\infty}^{+\infty} e^{-\left(z_j - g_j + e^{-(z_j - g_j)}\right)} \prod_{i \ne j} e^{-e^{-(z_j - g_i)}}\, dz_j \\ &= \int_{z_j=-\infty}^{+\infty} e^{-z_j + g_j - e^{-z_j} \sum_{i=1}^{N} e^{g_i}}\, dz_j \\ &= \frac{e^{g_j - e^{-z_j} \sum_{i=1}^{N} e^{g_i}}}{\sum_{i=1}^{N} e^{g_i}} \Bigg|_{-\infty}^{+\infty} = \frac{e^{g_j}}{\sum_{i=1}^{N} e^{g_i}} = p_j. \end{aligned} \tag{4.9}$$

□

Since the added Gumbel noises are independent, showing the M items with the highest z_i scores is equivalent to sampling M items without replacement from Equation 4.5.

To balance exploration and exploitation, one resorts to annealing [1], with an appropriately tuned sequence of learning rate parameters (aka inverse temperature) $\eta_k \ge 0$ for each timestep t_k:

$$p_j = \frac{e^{\eta_k g_j}}{\sum_{i=1}^{N} e^{\eta_k g_i}}. \tag{4.10}$$

Note that $\eta_k = 0$ recovers the pure exploration mode, and $\eta_k = +\infty$ recovers the pure exploitation mode. Varying η_k allows us to trade-off exploitation and exploration.

On the other hand, similarly to the proof above, we have

$$\operatorname*{argmax}_j\{\eta_k\, g_j + \mathit{Gumbel}(0,1)\} = \operatorname*{argmax}_j\{\mathit{Gumbel}(\eta_k\, g_j, 1)\} \sim \frac{e^{\eta_k\, g_j}}{\sum_{i=1}^{N} e^{\eta_k\, g_i}}. \tag{4.11}$$

Note that, by dividing by η_k, we establish:

$$\operatorname*{argmax}_j\{\eta_k\, g_j + \mathit{Gumbel}(0,1)\} \sim \operatorname*{argmax}_j\left\{g_i + \frac{\mathit{Gumbel}(0,1)}{\eta_k}\right\}. \tag{4.12}$$

Sampling from $\mathit{Gumbel}(\eta_k\, g_j, 1)$ and taking the maximum, as in Equation 4.11, is similar to Thompson Sampling in a bandit setting [26, 35]. The crucial difference (and drawback) is that the Gumbel method doesn't take into account the uncertainty of the reward estimates.

Finding the right schedule for η_k can be very difficult in practice [36]. In [5], the authors provide an annealing schedule for η_k in a standard stochastic multi-armed bandit setting, guaranteeing sub-linear regret. Let n_j be the number of times arm j has been played up to timestep t_{k-1}. For some constant $C > 0$, they set $\eta = \sqrt{n_j/C^2}$, and sample according to:

$$\operatorname*{argmax}_j\left\{g_j + \sqrt{C^2/n_j}\; \mathit{Gumbel}(0,1)\right\}. \tag{4.13}$$

Equation 4.13 decouples the learning rates of the individual items, and factors-in the uncertainty of the reward estimates. We now have a proper way to sample from a Boltzmann, with convergence guarantees. Even though our setting is not exactly the same as [5], we borrow parts of their sampling strategy to recommend items to the user. As detailed in the theoretical justifications of Appendix B, we recommend setting $C^2 = 1/8$ for $g_j = \mathbb{P}(x_j|S)$.

As a user can repeatedly interact with the same item, we treat n_j as the number of times a user interacts with item j. It starts with $n_j = 1$ and is incremented with every like or dislike to item j. Putting it all together, we obtain our final Boltzmann sampling algorithm (Algorithm 3).

Algorithm 3 Boltzmann sampling for recommending items

```
Input: $M \leq N, n_i \forall i \in \{1, 2, \cdots, N\}$
    for all $x_i$, $i = 1, \cdots, N$ do
        $\gamma_i \sim$ Gumbel(0, 1)
        $z_i = g_i + \frac{C\,\gamma_i}{\sqrt{n_i}}$
    results = sort({ $z_i \mid i = 1, 2, \cdots, N$ }) in descending order
    return top $M$ results
```

5 EVALUATION

As our experiments require human judgments, there exist no such ground truth datasets for validation. Instead we propose an experimental framework with a human in the loop that simulates the **Seeker** experience and generates quantifiable metrics. The evaluation study serves as a benchmark for future sequential search algorithms.

5.1 Experimental Setup

Seeker assumes that the user has a mental image of a target item they cannot easily express in words. When accessing Seeker, the user is presented with a subset of M items to interact with, using like or dislike clicks. At any moment, the user can expand the catalog listing view by clicking on "Explore More". Our experimental setting mimics this initial user experience.

A single experimental session involves the following: A user is presented with a target item x_t. This target item is an explicit simulation of the user's hidden target. At each timestep, we present the user with a grid of M items. The user's goal is to find the target item through a series of feedbacks. At each timestep, they may like, dislike, or remove a previously liked/disliked item. Upon receiving user feedback, we recommend M new items to view in the next timestep. The session goes on for K timesteps. If the user can find the target within the M items, they may stop playing. Otherwise, they try to get as close to the target item as possible based on their perception of similarity. For our experiments, we set $M = 12$, $K = 15$, $d = 2048$, $N = 2228$, $g_j = \log \mathbb{P}(x_j|S)$, and used an uninformative prior.

We enlisted volunteers to participate in the experiment defined above, and collected 358 (roughly 90 per sampling algorithm) unique sessions. The target and exploration algorithm for each session was selected uniformly at random. Users were instructed to like and dislike items assuming that they wanted to purchase the target item. Users had no prior knowledge of the selected catalog or algorithm. At each timestep, we invoke Seeker to generate a posterior distribution over the catalog, according to Equation 4.4. This distribution enforces a natural ranking on the items. We monitor the normalized rank of the target item at each timestep. The normalized rank ρ is defined as the rank of the target item divided by the size of the catalog. A target x_t with a normalized rank of $\rho = 0.1$ means it has a final rank of $\lceil 0.1 * N \rceil$.

5.2 Experimental Results

Seeker aims at helping the user quickly zoom-in on the desired target item. A typical metric for such recommender systems is *recall at k* [11]. As we have only one target of interest, we measure how close our recommendations are to target t. We can do that using the target's normalized rank. For a given session i, let ρ_i be the lowest normalized rank attained by t in all K timesteps. We define *recall @ρ_j* as the percentage of sessions with $\rho_i \leq \rho_j$. For example, a recall of 0.4 @0.02 means that 40% of sessions achieved a normalized ranking of $\rho = 0.02$ or less.

Figure 3 plots recall @ρ for our sampling strategies. We plot ρ up to 0.1, as the user is unlikely to scroll past higher percentiles. Boltzmann exploration achieves the highest recall, dominating all other strategies. Noiseless and Greedy perform similarly, outperforming random at lower recalls. Random improves at higher recalls due to its higher degree of exploration, where the target gets ranked high by pure chance.

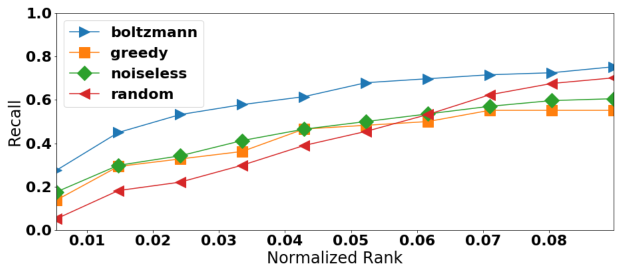

Figure 3: Recall at different normalized rank cutoffs. The plot measures how often Seeker ranks the target item better than a given percentile within $K = 15$ timesteps.

Figure 4 plots the convergence time of our sampling strategies. From the user's perspective, this reflects how long it takes to find a reasonably close approximation of the target item. We report the mean number of steps it takes for the rank of the target item to drop below a given recall cutoff ρ. Boltzmann exploration consistently outperforms the other strategies. Greedy and Noiseless surpass Random, but their advantage diminishes at higher rank cutoffs.

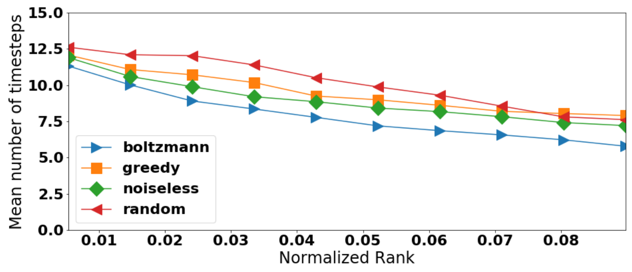

Figure 4: Mean number of timesteps (interactions) until target item is ranked better than a given percentile.

5.3 Discussion

We would like to point out that the experimental setup described above is not restrictive. Although we do present a window of M items with which the user interacts, the user can expand the window size M by explicitly clicking on an "Explore More" option. Once in the expanded view, scrolling down past the last displayed item triggers the display of additional items in an infinite scroll mode covering the whole N catalog items. Since we maintain an explicit ranking on all items, this mode of experimentation merges naturally with our algorithm.

Infinite scroll, such as home feeds on social media websites like Facebook and twitter, may create a better user experience and allow for the user to browse quickly. But when the target is explicit, such an infinite scroll feature makes our experimental framework trivial – the user can just scroll until they find the target. This prevents us from gaining insight about convergence, which explains why we limited our study to the windowed-version of the application. We consider our experimental setup a restrictive experience in terms of user experience.

Additionally, the catalog contains multiple similar items. This leads to a large number of identical feature-vector representations, making it challenging to surface the target item among $M = 12$ items in just $K = 15$ timesteps. Hence, it is likely that the Section 5.2 experimental results are pessimistic, as users are constrained from browsing the search space efficiently. Nevertheless, as the top-most items get the most visibility, Seeker's ability to quickly zoom-in to the item of interest remains crucial.

On occasions, the Seeker interface produces pages with very similar items ranked closely, leading to lack of exploration. Two items which look mostly identical are likely to have similar vector representation and hence may appear adjacent to each other [24]. In a deterministic setting, this would have resulted in a page full of very similar items, and prevented the user from pivoting to other parts of the catalog. Although Boltzmann exploration offers a principled remedy, depending on the use case, one may want to model additional uncertainty into the user actions in the early stages of the interactions. As a remedy, we can modify the posteriors by adding noise, using submodular functions [7], or determinental point processes [2].

The constant C in Algorithm 3 is borrowed from the work in [5] which uses the non-contextual stochastic multi-armed bandit setting. Under their setting C is a reasonable estimate to bound variance. However, items in our search space have features that are shared and correlated. Our sampling strategy currently does not take into account this covariance properly when making recommendations. We leave augmenting our sampling algorithm with a new variance bound for future work.

6 FUTURE WORK

We are considering improving this work on multiple fronts.

6.1 Bipartite Preference Model

When constructing the preferences from user feedback in Section 3.2, we treat the likes and dislikes independently, valuing them equally. However, intuitively, if we put more emphasis on likes, we may be able to find the target faster. Likes are less ambiguous than dislikes: likes have a clearer implication when treated in isolation, while dislikes usually require context to be a useful learning signal. In fact, we empirically observe that the likes are more clustered with one another than the dislikes to themselves.

We can mathematically model the emphasis of likes over dislikes by assuming the likes are independent of each other but the dislikes are conditioned on the likes. Conditional dependencies can be expressed in the form of a bipartite graph as shown in Figure 5.

Using Bayes rule, we can represent Figure 5 as:

$$\mathbb{P}(S|t) = \prod_{i=1}^{p} \mathbb{P}(a_i|t) \prod_{j=1}^{q} \mathbb{P}(b_j|a_1, \cdots, a_p, t). \tag{6.1}$$

Here we assume that:

$$\mathbb{P}(a_i|t) = \frac{1}{\exp\left\{\alpha_1 ||a_i - x_t||^2\right\}}, \tag{6.2}$$

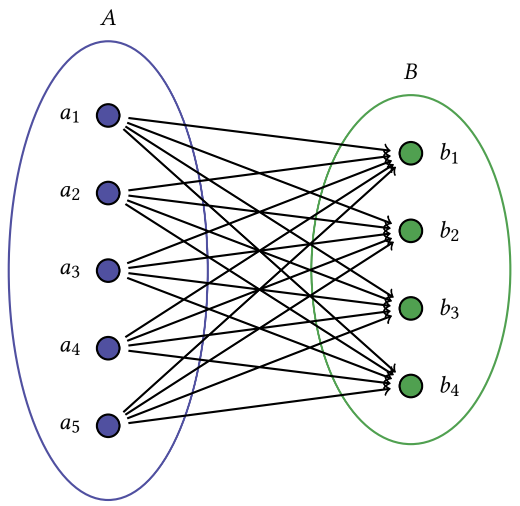

Figure 5: A complete directed bipartite graph between the sets of likes and dislikes.

and

$$\mathbb{P}(b_j|a_1,\cdots,a_p,t) = \frac{1}{1+\exp\left\{-\alpha_2\Big(||b_j-x_t||^2-\min_{i\in\{1,\cdots,p\}}||a_i-x_t||^2\Big)\right\}}, \quad (6.3)$$

with $\alpha_1, \alpha_2 \geq 0$.

We interpret the model in the following way. Equation 6.2 conveys that the probability of liking item i is proportional to how similar i is to target t. Equation 6.3 conveys that the probability of disliking item j is proportional to its relative distance to the target as compared to the relative distance between the target and liked items A. One can quantify the distance between t and A in different ways. Here we propose *min* to reflect the customer's gradual approach towards the target. We leave the evaluation of this model to future work.

6.2 Incorporating Additional Feedback

So far, the only form of feedback that the user provides is in the form of likes and dislikes. Consider the situation where the user provides feedback in the form of a text or utterance. This transitions us into the guided conversational search paradigm and we could incorporate some of the strategies described in [14, 28, 37].

Assume we have a technique (like LSTM to create word embeddings) to map a spoken feedback into a vector $r_i \in R^d$. We want to incorporate this feedback into the model. Equation 4.3 becomes:

$$\mathbb{P}(t|S,\mathbf{r}) \propto \mathbb{P}(S|t,\mathbf{r})\mathbb{P}(t|\mathbf{r}), \quad (6.4)$$

where

$$\mathbb{P}(t|\mathbf{r}) = \frac{1}{1+\exp\left\{-\beta\sum_k r_k^T x_t\right\}}. \quad (6.5)$$

To estimate $\mathbb{P}(S|t,\mathbf{r})$, we change Equation 3.2 to:

$$\mathbb{P}(s_{ij}|t,i,j,\mathbf{r}) = \frac{1}{1+\exp\left\{-\alpha\Big(||x_j-x_t||^2-||x_i-x_t||^2+\sum_k r_k^T(x_i-x_j)\Big)\right\}}. \quad (6.6)$$

Although we used text/speech as an example, the additional feedback embedding r can originate from an arbitrary source. Similarly, we can incorporate extra feedback into the bipartite preference model of Section 6.1.

6.3 Personalized Recommendations

Another possible direction is to personalize Seeker. Let c be an embedding vector for each user. The dataset now comes in the form of quadruplets (c, t, a, b), where each user c has a target t and pairs (a, b) of likes and dislikes.

To personalize, we define a synthetic embedding kernel $\phi(c, x)$, where x denotes an item. For example, we can use element-wise product:

$$\phi(c,x) = c \odot x. \quad (6.7)$$

Now, we can substitute this kernel into our modeling formulas, replacing any item x with personalized embedding $\phi(c, x)$.

7 CONCLUSION

This paper presents Seeker, an interactive, real-time search system. Seeker allows users to search for products even when it is difficult to describe them in words. Unlike embedding-based search engines, this method does not require a preknown representation of the desired item. With interactive binary feedback, our system learns to dynamically refine search results from the user's preferences in real time. Our evaluation results show that our Boltzmann exploration method allows users to find their products more quickly and with greater regularity compared to alternative exploration strategies.

ACKNOWLEDGMENTS

We gratefully acknowledge Kevin Jamieson and Lalit Jain for sharing their experience of designing adaptive algorithms. We would like to thank Miguel Jimenez Gomez, Xiaopeng Zhang and Andrea Matsunaga for their software development expertise. In addition, we thank the volunteers for their help in performing the user study. Finally, we thank Amazon for the opportunity to conduct this research project.

REFERENCES

[1] Emile Aarts and Jan Korst. 1988. Simulated annealing and Boltzmann machines. (1988).
[2] Raja Hafiz Affandi, Alex Kulesza, and Emily B. Fox. 2012. Markov Determinantal Point Processes. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*.
[3] Artem Babenko, Anton Slesarev, Alexander Chigorin, and Victor S. Lempitsky. 2014. Neural Codes for Image Retrieval. In *The European Conference on Computer Vision (ECCV)*. Zurich, Switzerland, 584–599.
[4] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* 5, 1 (2012), 1–122.
[5] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. 2017. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems*. 6284–6293.
[6] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*. 2249–2257.

[7] Lin Chen, Andreas Krause, and Amin Karbasi. 2017. Interactive Submodular Bandit. In *Advances in Neural Information Processing Systems 30 (NIPS)*. 141–152.

[8] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. 2008. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (Csur)* 40, 2 (2008), 5.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] M Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C Berg, and Tamara L Berg. 2015. Where to buy it: Matching street clothing photos in online shops. In *Proceedings of the IEEE international conference on computer vision*. 3343–3351.

[11] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.* 22, 1 (2004), 5–53.

[12] Daniel N. Hill, Houssam Nassif, Yi Liu, Anand Iyer, and S.V.N. Vishwanathan. 2017. An Efficient Bandit Algorithm for Realtime Multivariate Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM, 1813–1821.

[13] Houdong Hu, Yan Wang, Linjun Yang, Pavel Komlev, and Li Huang. 2018. Web-Scale Responsive Visual Search at Bing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*. ACM.

[14] Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. 2018. Flowqa: Grasping flow in history for conversational machine comprehension. *arXiv preprint arXiv:1810.06683* (2018).

[15] Lalit Jain, Kevin G Jamieson, and Rob Nowak. 2016. Finite Sample Prediction and Recovery Bounds for Ordinal Embedding. In *Advances in Neural Information Processing Systems 29 (NIPS)*. Barcelona, Spain, 2711–2719.

[16] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Fifth International Conference on Learning Representations (ICLR)*.

[17] Yushi Jing, David Liu, Dmitry Kislyuk, Andrew Zhai, Jiajing Xu, Jeff Donahue, and Sarah Tavel. 2015. Visual search at pinterest. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1889–1898.

[18] Yushi Jing, David Liu, Dmitry Kislyuk, Andrew Zhai, Jiajing Xu, Jeff Donahue, and Sarah Tavel. 2015. Visual Search at Pinterest. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM.

[19] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous Feature Learning and Hash Coding with Deep Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, 3270–3278.

[20] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.

[21] Jerri L. Ledford. 2015. *Search engine optimization bible* (second ed.). Wiley, Indianapolis, IN.

[22] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. 2016. Feature Learning Based Deep Supervised Hashing with Pairwise Labels. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. 1711–1717.

[23] Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* Sampling. In *Advances in Neural Information Processing Systems 27 (NIPS)*.

[24] Houssam Nassif, Kemal Oral Cansizlar, Mitchell Goodman, and S. V. N. Vishwanathan. 2016. Diversifying Music Recommendations. In *Proceedings of Machine Learning for Music Discovery Workshop at* 33^{rd} *International Conference on Machine Learning (ICML)*.

[25] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[26] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning* 11, 1 (2018), 1–96.

[27] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[28] Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871* (2018).

[29] B. Smith and G. Linden. 2017. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing* 21, 3 (2017), 12–18.

[30] Neil Stewart, Gordon DA Brown, and Nick Chater. 2005. Absolute identification by relative judgment. *Psychological review* 112, 4 (2005), 881.

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, 2818–2826.

[32] Csaba SzepesvÃ¡ri. 2010. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers.

[33] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. 2011. Adaptively learning the crowd kernel. *arXiv preprint arXiv:1105.1033* (2011).

[34] Choon Hui Teo, Houssam Nassif, Daniel Hill, Sriram Srinivasan, Mitchell Goodman, Vijai Mohan, and S.V.N. Vishwanathan. 2016. Adaptive, Personalized Diversity for Visual Discovery. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. ACM, Boston, MA, 35–38.

[35] William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* (1933), 285–294.

[36] Joannès Vermorel and Mehryar Mohri. 2005. Multi-armed Bandit Algorithms and Empirical Evaluation. In *Proceedings of the 16th European Conference on Machine Learning (ECML'05)*. 437–448.

[37] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* (2016).

[38] Rongkai Xia, Yan Pan, Hanjiang Lai1, Cong Liu, and Shuicheng Yan. 2014. Supervised Hashing for Image Retrieval via Image Representation Learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*. 2156–2162.

[39] Fan Yang, Ajinkya Kale, Yury Bubnov, Leon Stein, Qiaosong Wang, Hadi Kiapour, and Robinson Piramuthu. 2017. Visual Search at eBay. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM.

[40] Yanhao Zhang, Pan Pan, Yun Zheng, Kang Zhao, Yingya Zhang, Xiaofeng Ren, and Rong Jin. 2018. Visual Search at Alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*. ACM.

[41] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep Hashing Network for Efficient Similarity Retrieval. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. 2415–2421.

A ASYMPTOTIC SAMPLING EQUIVALENCE

Given T as a compact (i.e. closed and bounded) subset of $\mathbb{R}^d$. Let $f : T \longrightarrow \mathbb{R}_+$ be a continuous probability density function. Let the set $X_n = \{x_1, ..., x_n\}$ consist of points x_i's that are equally spaced on T in the grid-like manner such that $\sum_{i=1}^n f(x_i) > 0$. Consider the following two ways of sampling over X_n:

(1) Each time, sample x from f on T, and choose $x_k \in X_n$ if and only if $x_k = \operatorname{argmin}_{x_i \in X_n} d(x, x_i)$, where the metric $d(\cdot, \cdot)$ is usually the Euclidean metric. We assume argmin is unique.

(2) Each time, sample each $x_k \in X_n$ from the discrete distribution on X_n so that x_k is chosen with probability $f(x_k)/\sum\limits_{i=1}^{n} f(x_i)$.

Define:

$$D_n := \sum_{x_k \in X_n} \left| \mathbb{P}(x_k | \text{ Method 1}) - \mathbb{P}(x_k | \text{ Method 2}) \right|. \qquad \text{(A.1)}$$

Prove that $\lim\limits_{n \to \infty} D_n \longrightarrow 0$.

Proof. Partition T into n disjoint regions $(J_1, ..., J_n)$ in the grid-like manner such that for each $i \in \{1, ..., n\}$ for each $x \in J_i$,

$$i = \operatorname*{argmin}_{k \in \{1,...,n\}} d(x, x_k).$$

Since the points x_i's are equally spaced on T, the regions J_i's all have the same measure: $m(J_i) = m = \frac{m_T}{n}$, where m_T is the (fixed) Lebesgue measure of T. So

$$\mathbb{P}(x_k | \text{ Method 1}) = \mathbb{P}(x \in J_k) = \int_{J_k} f(x)dx = mf(x_k^*) \text{ for all } k.$$

Here the first equation holds by the definition of J_i's, and the second by the Mean Value Theorem (MVT) for some $x_k^* \in J_k$. On the other hand,

$$1 = \int_T f(x)dx = \sum_{i=1}^{n} \int_{J_i} f(x)dx = \sum_{i=1}^{n} mf(x_i^+),$$

where the second equation holds by the additivity of integral, and the last equation holds by the MVT for some $x_i^+ \in J_i$.

Since f is continuous on the compact subset T of $\mathbb{R}^d$, there is an upper bound U such that $f(x) < U$ for all $x \in T$. Moreover by the Heine - Cantor theorem, f is uniformly continuous on T.

Now fix $\epsilon > 0$, $\epsilon < 1/2$. By uniform continuity of f on T, there exists $\delta > 0$ such that for all $x_1, x_2 \in T$ with $d(x_1, x_2) < \delta$, we have $|f(x_1) - f(x_2)| < S\epsilon$ where

$$S = \min\left(\frac{1}{2m_T(1 + Um_T + m_T)}, 1, \frac{1}{m_T}\right).$$

Because the regions J_i's are partitioned in the grid-like manner, there exists $N_0 \in \mathbb{Z}_+$ such that for all $n > N_0$, the diameter of each J_i is smaller than δ. This implies $d(x_i^+, x_i) < \delta$ for all i and $d(x_k^*, x_k) < \delta$ for all k. Hence for all $n > N_0$, we have

$|f(x_i^+) - f(x_i)| < S\epsilon$ and $|f(x_k^*) - f(x_k)| < S\epsilon$ for all $i, k \in \{1, ..., n\}$,

which implies:

$$\left|m\sum_{i=1}^{n} f(x_i) - 1\right| = \left|m\sum_{i=1}^{n} f(x_i) - m\sum_{i=1}^{n} f(x_i^+)\right| < mnS\epsilon = m_T S\epsilon.$$

Therefore for each $k \in \{1, 2, ..., n\}$,

$$|\mathbb{P}(x_k| \text{ Method 1}) - \mathbb{P}(x_k| \text{ Method 2})| = m\left|f(x_k^*) - \frac{f(x_k)}{m\sum_{i=1}^{n} f(x_i)}\right|$$

$$= \frac{m_T}{n}\left|f(x_k) + t_1 - \frac{f(x_k)}{1 + t_2}\right|, \text{ where } |t_1| < S\epsilon, |t_2| < m_T S\epsilon$$

$$= \frac{m_T}{n|1 + t_2|} |(f(x_k) + t_1)(1 + t_2) - f(x_k)|$$

$$= \frac{m_T}{n|1 + t_2|}|t_1 + f(x_k)t_2 + t_1 t_2|$$

$$\leq \frac{m_T}{n|1 + t_2|} (|t_1| + |f(x_k)||t_2| + |t_1||t_2|)$$

$$< \frac{m_T}{n|1 + t_2|}(1 + Um_T + m_T S\epsilon)S\epsilon$$

$$< \frac{2m_T}{n}(1 + Um_T + m_T)S\epsilon. \quad (\text{Because } S\epsilon < 1, m_T S\epsilon < 1/2)$$

This implies that for all $n > N_0$,

$$D_n < 2m_T(1 + Um_T + m_T)S\epsilon \leq \epsilon.$$

This ends the proof. □

B SETTING PARAMETER C

Given an item, a user can like or dislike it. Our rewards are thus binary, making the reward distribution 1/2-subgaussian with variance factor $\sigma^2 = 1/4$. We follow [5]'s Theorem 3 computations with a standard Gumbel noise $Gumbel(0, 1)$ (see Equations 4.7 and 4.8). We do not introduce extra variable c in the proof of Lemma 3, setting $L = \frac{9C^2 \log_+^2(T\Delta_i^2)}{\Delta_i^2}$. We thus bind the regret R_T as:

$$R_T \leq \sum_{i=2}^{N} \frac{9C^2 \log_+^2(T\Delta_i^2)}{\Delta_i} + \sum_{i=2}^{N} \frac{36C^2 e^{\sigma^2/2C^2}}{\Delta_i} + \sum_{i=2}^{N} \Delta_i. \quad \text{(B.1)}$$

Here the finite horizon T is the final timestep, and the gap Δ_i is the difference between the mean reward of the optimal item, and the mean reward of item i.

Although T may potentially be specified, Δ_i is unknown. To obtain a small regret, the authors recommend setting $C = \sigma$. But one can easily see that choosing $C = \sigma/\sqrt{2}$ leads to an even smaller regret. We therefor set $C^2 = \sigma^2/2 = 1/8$.