Cascading Hybrid Bandits: Online Learning to Rank for Relevance and Diversity

Chang Li
University of Amsterdam
c.li@uva.nl

Haoyun Feng
Bloomberg
hfeng19@bloomberg.net

Maarten de Rijke
University of Amsterdam
Ahold Delhaize
m.derijke@uva.nl

ABSTRACT

Relevance ranking and result diversification are two core areas in modern recommender systems. Relevance ranking aims at building a ranked list sorted in decreasing order of item relevance, while result diversification focuses on generating a ranked list of items that covers a broad range of topics. In this paper, we study an online learning setting that aims to recommend a ranked list with K items that maximizes the ranking utility, i.e., a list whose items are relevant and whose topics are diverse. We formulate it as the *cascade hybrid bandits* (CHB) problem. CHB assumes the cascading user behavior, where a user browses the displayed list from top to bottom, clicks the first attractive item, and stops browsing the rest. We propose a hybrid contextual bandit approach, called CascadeHybrid, for solving this problem. CascadeHybrid models item relevance and topical diversity using two independent functions and simultaneously learns those functions from user click feedback. We conduct experiments to evaluate CascadeHybrid on two real-world recommendation datasets: MovieLens and Yahoo music datasets. Our experimental results show that CascadeHybrid outperforms the baselines. In addition, we prove theoretical guarantees on the n-step performance demonstrating the soundness of CascadeHybrid.

CCS CONCEPTS

• **Information systems → Learning to rank**; • **Theory of computation → Online learning algorithms**.

KEYWORDS

Online learning to rank, contextual bandits, recommender system, result diversification

ACM Reference Format:
Chang Li, Haoyun Feng, and Maarten de Rijke. 2020. Cascading Hybrid Bandits: Online Learning to Rank for Relevance and Diversity. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383313.3412245

1 INTRODUCTION

Ranking is at the heart of modern interactive systems, such as recommender and search systems. Learning to rank (LTR) addresses the ranking problem in such systems by using machine learning approaches [23]. Traditionally, LTR has been studied in an offline fashion, in which human labeled data is required [23]. Human labeled data is expensive to obtain, cannot capture future changes in user preferences, and may not well align with user needs [13]. To circumvent these limitations, recent work has shifted to learning directly from users' interaction feedback, e.g., clicks [12, 14, 34].

User feedback is abundantly available in interactive systems and is a valuable source for training online LTR algorithms [9]. When designing an algorithm to learn from this source, three challenges need to be addressed: (1) The learning algorithm should address position bias (the phenomenon that higher ranked items are more likely be observed than lower ranked items); (2) The learning algorithm should infer item relevance from user feedback and recommend lists containing relevant items (relevance ranking); (3) The recommended list should contain no redundant items and cover a broad range of topics (result diversification).

To address the position bias, a common approach is to make assumptions on the user's click behavior and model the behavior using a click model [6]. The cascade model (CM) [7] is a simple but effective click model to explain user behavior. It makes the so-called *cascade assumption*, which assumes that a user browses the list from the first ranked item to the last one and clicks on the first attractive item and then stops browsing. The clicked item is considered to be positive, items before the click are treated as negative and items after the click will be ignored. Previous work has shown that the cascade assumption can explain the position bias effectively and several algorithms have been proposed under this assumption [11, 16, 18, 35].

In online LTR, the implicit signal that is inferred from user interactions is noisy [13]. If the learning algorithm only learns from these signals, it may reach a suboptimal solution where the optimal ranking is ignored simply because it is never exposed to users. This problem can be tackled by exploring new solutions, where the learning algorithm displays some potentially "good" rankings to users and obtains more signals. This behavior is called *exploration.* However, exploration may hurt the user experience. Thus, learning algorithms face an exploration vs. exploitation dilemma. Multi-armed bandit (MAB) [5, 17] algorithms are commonly used to address this dilemma. Along this line, multiple algorithms have been proposed [12, 19, 21, 25]. They all address the dilemma in elegant ways and aim at recommending the top-K most relevant items to users. However, only recommending the most relevant items may result in a list with redundant items, which diminishes the utility of the list and decreases user satisfaction [3, 32].

The submodular coverage model [24] can capture the pattern of diminishing utility and has been used in online LTR for diversified ranking. One assumption in this line of work is that items can be represented by a set of topics.[1] The task, then, is to recommend a list of items that ensures a maximal coverage of topics. Yue and Guestrin [32] develop an online feature-based diverse LTR algorithm by optimizing submodular utility models [32]. Hiranandani et al. [11] improve online diverse LTR by bringing the cascading assumption into the objective function. However, we argue that not all features that are used in a LTR setting can be represented by topics [23]. Previous online diverse LTR algorithms tend to ignore the relevance of individual items and may recommend a diversified list with less relevant items.

In this work, we address the aforementioned challenges and make four contributions:

1. We focus on a novel online LTR setting that targets both relevance ranking and result diversification. We formulate it as a cascade hybrid bandits (CHB) problem, where the goal is to select K items from a large candidate set that maximize the utility of the ranked list (Section 3.1).
2. We propose CascadeHybrid, which utilizes a hybrid model, to solve this problem (Section 3.3).
3. We evaluate CascadeHybrid on two real-world recommendation datasets: MovieLens and Yahoo and show that CascadeHybrid outperforms state-of-the-art baselines (Section 4).
4. We theoretically analyze the performance of CascadeHybrid and provide guarantees on its proper behavior; moreover, we are the first to show that the regret bounds on feature-based ranking algorithms with the cascade assumption are linear in $\sqrt{K}$.

The rest of the paper is organized as follows. We recapitulate the background knowledge in Section 2. In Section 3, we formulate the learning problem and propose our CascadeHybrid algorithm that optimizes both item relevance and list diversity. Section 4 contains our empirical evaluations of CascadeHybrid, comparing it with several state-of-the-art baselines. An analysis of the upper bound on the n-step performance of CascadeHybrid is presented in Section 5. In Section 6, we review related work. Conclusions are formulated in Section 7.

2 BACKGROUND

In this section, we recapitulate the cascade model (CM), cascading bandits (CB), and the submodular coverage model. Throughout the paper, we consider the ranking problem of L candidate items and K positions with $K \leq L$. We denote $\{1, \dots, n\}$ by $[n]$ and for the collection of items we write $\mathcal{D} = [L]$. A ranked list contains $K \leq L$ items and is denoted by $\mathcal{R} \in \Pi_K(\mathcal{D})$, where $\Pi_K(\mathcal{D})$ is the set of all permutations of K distinct items from the collection $\mathcal{D}$. The item at the k-th position of the list is denoted by $\mathcal{R}(k)$ and, if $\mathcal{R}$ contains an item i, the position of this item in $\mathcal{R}$ is denoted by $\mathcal{R}^{-1}(i)$. All vectors are column vectors. We use bold font to indicate a vector and bold font with a capital letter to indicate a matrix. We write $\mathbf{I}_d$ to denote the $d \times d$ identity matrix and $\mathbf{0}_{d\times m}$ the $d \times m$ zero matrix.

[1]In general, each topic may only capture a tiny aspect of the information of an item, e.g., a single phrase of a news title or a singer of a song [3, 32].

2.1 Cascade model

Click models have been widely used to interpret user's interactive click behavior; cf. [6]. Briefly, a user is shown a ranked list $\mathcal{R}$, and then browses the list and leaves click feedback. Every click model makes unique assumptions and models a type of user interaction behavior. In this paper, we consider a simple but widely used click model, the *cascade model* [7, 16, 18], which makes the cascade assumption about user behavior. Under the cascade assumption, a user browses a ranked list $\mathcal{R}$ from the first item to the last one by one and clicks the first attractive item. After the click, the user stops browsing the remaining items. A click on an examined item $\mathcal{R}(i)$ can be modeled as a Bernoulli random variable with a probability of $\alpha(\mathcal{R}(i))$, which is also called the *attraction probability*. Here, the cascade model (CM) assumes that each item attracts the user independent of other items in $\mathcal{R}$. Thus, the CM is parametrized by a set of attraction probabilities $\boldsymbol{\alpha} \in [0,1]^L$. The examination probability of item $\mathcal{R}(i)$ is 1 if $i = 1$, otherwise $1 - \prod_{j=1}^{i-1}(1-\alpha(\mathcal{R}(j)))$.

With the CM, we translate the implicit feedback to training labels as follows: Given a ranked list, items ranked below the clicked item are ignored since none of them are browsed. Items ranked above the clicked item are negative samples and the clicked item is the positive sample. If no item is clicked, we know that all items are browsed but not clicked. Thus, all of them are negative samples.

The vanilla CM is only able to capture the first click in a session, and there are various extensions of CM to model multi-click scenarios; cf. [6]. However, we still focus on the CM, because it has been shown in multiple publications that the CM achieves good performance in both online and offline setups [6, 16, 19].

2.2 Cascading bandits

Cascading bandits (CB) are a type of online variant of the CM [16]. A CB is represented by a tuple $(\mathcal{D}, K, P)$, where P is a binary distribution over $\{0,1\}^L$. The learning agent interacts with the CB and learns from the feedback. At each step t, the agent generates a ranked list $\mathcal{R}_t \in \Pi_K(\mathcal{D})$ depending on observations in the previous $t-1$ steps and shows it to the user. The user browses the list with cascading behavior and leaves click feedback. Since the CM accepts at most one click, we write $c_t \in [K+1]$ as the click indicator, where c_t indicates the position of the click and $c_t = K+1$ indicates no click. Let $A_t \in \{0,1\}^L$ be the attraction indicator, where A_t is drawn from P and $A_t(\mathcal{R}_t(i)) = 1$ indicates that item $\mathcal{R}_t(i)$ attracts the user at step t. The number of clicks at step t is considered as the reward and computed as follows:

$$r(\mathcal{R}_t, A_t) = 1 - \prod_{i=1}^{K}(1 - A_t(\mathcal{R}_t(i))). \tag{1}$$

Then, we assume that the attraction indicators of items are distributed independently as Bernoulli variables:

$$P(A) = \prod_{i\in\mathcal{D}} P_{\alpha(i)}(A(i)), \tag{2}$$

where $P_{\alpha(i)}(\cdot)$ is the Bernoulli distribution with mean $\alpha(i)$. The expected number of clicks at step t is computed as $\mathbb{E}\left[r(\mathcal{R}_t, A_t)\right] =$

$r(\mathcal{R}_t, \boldsymbol{\alpha})$. The goal of the agent is to maximize the expected number of clicks in n steps or minimize the expected n-step regret:

$$R(n) = \sum_{t=1}^{n} \mathbb{E}\left[\max_{\mathcal{R}\in\Pi_K(\mathcal{D})} r(\mathcal{R}, \boldsymbol{\alpha}) - r(\mathcal{R}_t, A_t)\right]. \tag{3}$$

CB has several variants depending on assumptions on the attraction probability $\boldsymbol{\alpha}$. Briefly, cascade linear bandits [35] assume that an item a is represented by a feature vector $\mathbf{z}_a \in \mathbb{R}^m$ and that the attraction probability of an item a to a user is a linear combination of features: $\alpha(a) \approx \mathbf{z}_a^T \beta^*$, where $\beta^* \in \mathbb{R}^m$ is an unknown parameter. With this assumption, the attraction probability of an item is independent of other items in the list, and this assumption is used in relevance ranking problems. CascadeLinUCB has been proposed to solve this problem. For other problems, Hiranandani et al. [11] assume the attraction probability to be submodular, and propose CascadeLSB to solve result diversification.

2.3 Submodular coverage model

Before we recapitulate the submodular function, we introduce two properties of a diversified ranking. Different from the relevance ranking, in a diversified ranking, the utility of an item depends on other items in the list. Suppose we focus on news recommendation. Items that we want to rank are news itms, and each news item covers a set of topics, e.g., weather, sports, politics, a celebrity, etc. We want to recommend a list that covers a broad range of topics. Intuitively, adding a news item to a list does not decrease the number of topics that are covered by the list, but adding a news item to a list that covers highly overlapping topics might not bring much extra benefit to the list. The first property can be thought of as a monotonicity property, and the second one is the notion of diminishing gain in the utility. They can be captured by the submodular function [32].

We introduce two properties of submodular functions. Let $g(\cdot)$ be a set function, which maps a set to a real value. We say that $g(\cdot)$ is monotone and submodular if given two item sets $\mathcal{A}$ and $\mathcal{B}$, where $\mathcal{B} \subseteq \mathcal{A}$, and an item a, $g(\cdot)$ has the following two properties:

$$\begin{aligned} &\textit{monotonicity}: g(\mathcal{A} \cup \{a\}) \geq g(\mathcal{A}); \\ &\textit{submodularity}: g(\mathcal{B} \cup \{a\}) - g(\mathcal{B}) \geq g(\mathcal{A} \cup \{a\}) - g(\mathcal{A}). \end{aligned}$$

In other words, the gain in utility of adding an item a to a subset of $\mathcal{A}$ is larger than or equal to that of adding an item to $\mathcal{A}$, and adding an item a to $\mathcal{A}$ does not decreases the utility. Monotonicity and submodularity together provide a natural framework to capture the properties of a diversified ranking. The shrewd reader may notice that a linear function is a special case of submodular functions, where only the inequalities in *monotonicity* and *submodularity* hold. However, as discussed above, the linear model assumes that the attraction probability of an item is independent of other items: it cannot capture the diminishing gain in the result diversification. In the rest of this section, we introduce the *probabilistic coverage model*, which is a widely used submodular function for result diversification [3, 4, 11, 28, 32].

Suppose that an item $a \in \mathcal{D}$ is represented by a d-dimensional vector $\mathbf{x}_a \in [0,1]^d$. Each entry of the vector $\mathbf{x}_a(j)$ describes the probability of item a covering topic j. Given a list $\mathcal{A}$, the probability of $\mathcal{A}$ covering topic j is

$$g_j(\mathcal{A}) = 1 - \prod_{a\in\mathcal{A}} (1 - \mathbf{x}_a(j)). \tag{4}$$

The gain in topic coverage of adding an item a to $\mathcal{A}$ is:

$$\Delta(a \mid \mathcal{A}) = (\Delta_1(a \mid \mathcal{A}), \ldots, \Delta_d(a \mid \mathcal{A})), \tag{5}$$

where $\Delta_j(a \mid \mathcal{A}) = g_j(\mathcal{A} \cup \{a\}) - g_j(\mathcal{A})$. With this model, the attraction probability of the i-th item in a ranked list $\mathcal{R}$ is defined as:

$$\alpha(\mathcal{R}(i)) = \boldsymbol{\omega}_{\mathcal{R}(i)}^T \boldsymbol{\theta}^*, \tag{6}$$

where $\boldsymbol{\omega}_{\mathcal{R}(i)} = \Delta(\mathcal{R}(i) \mid (\mathcal{R}(1), \ldots, \mathcal{R}(i-1)))$ and $\boldsymbol{\theta}^*$ is the unknown user preference to different topics [11]. In Eq. (6), the attraction probability of an item depends on the items ranked above it; $\alpha(\mathcal{R}(i))$ is small if $\mathcal{R}(i)$ covers similar topics as higher ranked items. CascadeLSB [11] has been proposed to solve cascading bandits with this type of attraction probability and aims at building diverse ranked lists.

3 ALGORITHM

In this section, we first formulate our online learning to rank problem, and then propose CascadeHybrid to solve it.

3.1 Problem formulation

We study a variant of cascading bandits, where the attraction probability of an item in a ranked list depends on two aspects: item relevance and item novelty. Item relevance is independent of other items in the list. Novelty of an item depends on the topics covered by higher ranked items; a novel item brings a large gain in the topic coverage of the list, i.e., a large value in Eq. (6). Thus, given a ranked list $\mathcal{R}$, the attraction probability of item $\mathcal{R}(i)$ is defined as follows:

$$\alpha(\mathcal{R}(i)) = \mathbf{z}_{\mathcal{R}(i)}^T \boldsymbol{\beta}^* + \boldsymbol{\omega}_{\mathcal{R}(i)}^T \boldsymbol{\theta}^*, \tag{7}$$

where $\boldsymbol{\omega}_{\mathcal{R}(i)} = \Delta(\mathcal{R}(i) \mid (\mathcal{R}(1), \ldots, \mathcal{R}(i-1))$ is the topic coverage gain discussed in Section 2.3, and $\mathbf{z}_{\mathcal{R}(i)} \in \mathbb{R}^m$ is the relevance feature, $\boldsymbol{\theta}^* \in \mathbb{R}^d$ and $\boldsymbol{\beta}^* \in \mathbb{R}^m$ are two unknown parameters that characterize the user preference. In other words, the attraction probability is a hybrid of a modular (linear) function parameterized by $\boldsymbol{\beta}^*$ and a submodular function parameterized by $\boldsymbol{\theta}^*$.

Now, we define our learning problem, cascade hybrid bandits (CHB), as a tuple $(\mathcal{D}, \boldsymbol{\theta}^*, \boldsymbol{\beta}^*, K)$. Here, $\mathcal{D} = [L]$ is the item candidate set and each item a can be represented by a feature vector $[\mathbf{x}_a^T, \mathbf{z}_a^T]^T$, where $\mathbf{x}_a \in [0,1]^d$ is the topic coverage of item a discussed in Section 2.3. K is the number of positions. The action space for the problem are all permutations of K individual items from $\mathcal{D}$, $\Pi_K(\mathcal{D})$. The reward of an action at step t is the number of clicks, defined in Eq. (1). Together with Eqs. (1), (2) and (7), the expectation of reward at step t is computed as follows:

$$\mathbb{E}[r(\mathcal{R}_t, A_t)] = 1 - \prod_{a\in\mathcal{R}_t} (1 - \mathbf{z}_a^T \boldsymbol{\beta}^* - \boldsymbol{\omega}_a^T \boldsymbol{\theta}^*). \tag{8}$$

In the rest of the paper, we write $r(\mathcal{R}_t) = \mathbb{E}[r(\mathcal{R}_t, A_t)]$ for short. And the goal of the learning agent is to maximize the reward or, equivalently, to minimize the n-step regret defined as follow:

$$R(n) = \sum_{t=1}^{n} \left[\max_{\mathcal{R}\in\Pi_K(\mathcal{D})} r(\mathcal{R}) - r(\mathcal{R}_t)\right]. \tag{9}$$

The previously proposed CascadeLinUCB [35] cannot solve CHB since it only handles the linear part of the attraction probability. CascadeLSB [11] cannot solve CHB, either, because it only uses one submodular function and handles the submodular part of the attraction probability. Thus, we need to extend the previous models or, in other words, propose a new hybrid model that can handle both linear and submodular properties in the attraction probability.

3.2 Competing with a greedy benchmark

Finding the optimal set that maximizes the utility of a submodular function is an NP-hard problem [24]. In our setup, the attraction probability of each item also depends on the order in the list. To the best of our knowledge, we cannot find the optimal ranking

$$\mathcal{R}^* = \operatorname*{arg\,max}_{\mathcal{R}\in\Pi_K(\mathcal{D})} r(\mathcal{R}) \tag{10}$$

efficiently. Thus, we compete with a greedy benchmark that approximates the optimal ranking $\mathcal{R}^*$. The greedy benchmark chooses the items that have the highest attraction probability given the higher ranked items: for any positions $k \in [K]$,

$$\tilde{\mathcal{R}}(k) = \operatorname*{arg\,max}_{a\in\mathcal{D}\setminus\{\tilde{\mathcal{R}}(1),\ldots,\tilde{\mathcal{R}}(k-1)\}} \mathbf{z}_a^T\boldsymbol{\beta}^* + \boldsymbol{\omega}_a^T\boldsymbol{\theta}^*, \tag{11}$$

where $\tilde{\mathcal{R}}(k)$ is the ranked list generated by the benchmark.

This greedy benchmark has been used in previous literature [11, 32]. As shown by Hiranandani et al. [11], in the CM, the greedy benchmark is at least a η-approximation of $\mathcal{R}^*$. That is, $r(\tilde{\mathcal{R}}) \geq \eta r(\mathcal{R}^*)$ where $\eta = (1-\frac{1}{e})\max\{\frac{1}{K}, 1-\frac{K-1}{2}\alpha_{max}\}$ with $\alpha_{max} = \max_{a\in\mathcal{D}} \mathbf{z}_a^T\boldsymbol{\beta}^* + \mathbf{x}_a^T\boldsymbol{\theta}^*$. In the rest of the paper, we focus on competing with this greedy benchmark.

3.3 CascadeHybrid

We propose CascadeHybrid to solve the CHB. As the name suggests, the algorithm is a hybrid of a linear function and a submodular function. The linear function is used to capture item relevance and the submodular function to capture diversity in topics. CascadeHybrid has access to item features, $[\mathbf{x}_a^T, \mathbf{z}_a^T]^T$, and uses the probabilistic coverage model to compute the gains in topic coverage. The user preferences $\boldsymbol{\theta}^*$ and $\boldsymbol{\beta}^*$ are unknown to CascadeHybrid. They are estimated from interactions with users. The only tunable hyperparameter for CascadeHybrid is $\gamma \in \mathbb{R}_+$, which controls exploration: a larger value of γ means more exploration.

The details of CascadeHybrid are provided in Algorithm 1. At the beginning of each step t (line 5), CascadeHybrid estimates the user preference as $\hat{\boldsymbol{\theta}}_t$ and $\hat{\boldsymbol{\beta}}_t$ based on the previous $t-1$ step observations. $\hat{\boldsymbol{\theta}}_t$ and $\hat{\boldsymbol{\beta}}_t$ can be viewed as maximum likelihood estimators on the rewards,[2] where $\mathbf{M}_t, \mathbf{H}_t, \mathbf{B}_t$ and $\mathbf{y}_t, \mathbf{u}_t$ summarize the features and click feedback of all observed items in the previous $t-1$ steps. Then, CascadeHybrid builds the ranked list $\mathcal{R}_t$, sequentially (line 7–16). In particular, for each position k, we recalculate the topic coverage gain of each item (line 7). The new gains are used to estimate the attraction probability of items. CascadeHybrid makes an optimistic estimate of the attraction probability of each item (line 9–12) and chooses the one with the highest estimated attraction probability

[2]The derivation is based on matrix block-wise inversion. We omit the derivation since it is not a major contribution of this paper.

Algorithm 1 CascadeHybrid

```
Input: $\gamma$
1:  // Initialization
2:  $\mathbf{H}_1 \leftarrow \mathbf{I}_d, \mathbf{u}_1 \leftarrow \mathbf{0}_d, \mathbf{M}_1 \leftarrow \mathbf{I}_m, \mathbf{y}_1 \leftarrow \mathbf{0}_m, \mathbf{B}_1 = \mathbf{0}_{d\times m}$
3:  for $t = 1, 2, \ldots, n$ do
4:      // Estimate parameters
5:      $\hat{\boldsymbol{\theta}}_t \leftarrow \mathbf{H}_t^{-1}\mathbf{u}_t$, $\hat{\boldsymbol{\beta}}_t \leftarrow \mathbf{M}_t^{-1}(\mathbf{y}_t - \mathbf{B}_t^T\mathbf{H}_t^{-1}\mathbf{u}_t)$
6:      // Build ranked list
7:      $\mathcal{S}_0 \leftarrow \emptyset$
8:      for $k = 1, 2, \ldots K$ do
9:          for $a \in \mathcal{D} \setminus \mathcal{S}_{k-1}$ do
10:             $\boldsymbol{\omega}_a \leftarrow \Delta(\mathbf{x}_a|\mathcal{S}_{k-1})$ // Recalculate the topic coverage gain.
11:             $\mu_a \leftarrow$ Eq. (12) // Compute UCBs.
12:         end for
13:         $a_k^t \leftarrow \operatorname*{arg\,max}_{a\in\mathcal{D}\setminus\mathcal{S}_{k-1}} \mu_a$
14:         $\mathcal{S}_k \leftarrow \mathcal{S}_{k-1} + a_k^t$
15:     end for
16:     $\mathcal{R}_t = (a_1^t, \ldots, a_K^t)$ // Ranked list
17:     Display $\mathcal{R}_t$ and observe click feedback $c_t \in [K+1]$
18:     $k_t \leftarrow \min(K, c_t)$
19:     // Update statistics
20:     $\mathbf{H}_t \leftarrow \mathbf{H}_t + \mathbf{B}_t\mathbf{M}_t^{-1}\mathbf{B}_t^T$, $\mathbf{u}_t \leftarrow \mathbf{u}_t + \mathbf{B}_t\mathbf{M}_t^{-1}\mathbf{y}_t$
21:     for $a \in \mathcal{R}_t(1:k_t)$ do
22:         $\mathbf{M}_{t+1} \leftarrow \mathbf{M}_t + \mathbf{z}_a\mathbf{z}_a^T$, $\mathbf{B}_{t+1} \leftarrow \mathbf{B}_t + \boldsymbol{\omega}_a\mathbf{z}_a^T$, $\mathbf{H}_t \leftarrow \mathbf{H}_t + \boldsymbol{\omega}_a\boldsymbol{\omega}_a^T$
23:     end for
24:     if $c_t \leq K$ then
25:         $\mathbf{y}_{t+1} \leftarrow \mathbf{y}_t + \mathbf{z}_{\mathcal{R}_t(c_t)}$, $\mathbf{u}_t \leftarrow \mathbf{u}_t + \boldsymbol{\omega}_{\mathcal{R}_t(c_t)}$
26:     end if
27:     $\mathbf{H}_{t+1} \leftarrow \mathbf{H}_t - \mathbf{B}_{t+1}\mathbf{M}_{t+1}^{-1}\mathbf{B}_{t+1}^T$, $\mathbf{u}_{t+1} \leftarrow \mathbf{u}_t - \mathbf{B}_{t+1}\mathbf{M}_{t+1}^{-1}\mathbf{y}_{t+1}$
28: end for
```

(line 13). This is known as the principle of optimism in the face of uncertainty [5], and the estimator for an item a is called the upper confidence bound (UCB):

$$\mu_a = \boldsymbol{\omega}_a^T\hat{\boldsymbol{\theta}}_t + \mathbf{z}_a^T\hat{\boldsymbol{\beta}}_t + \gamma\sqrt{s_a}, \tag{12}$$

with

$$\begin{aligned} s_a \;=\; & \boldsymbol{\omega}_a^T\mathbf{H}_t^{-1}\boldsymbol{\omega}_a - 2\boldsymbol{\omega}_a^T\mathbf{H}_t^{-1}\mathbf{B}_t\mathbf{M}_t^{-1}\mathbf{z}_a + \mathbf{z}_a\mathbf{M}_t^{-1}\mathbf{z}_a \\ & +\mathbf{z}_a\mathbf{M}_t^{-1}\mathbf{B}_t^T\mathbf{H}_t^{-1}\mathbf{B}_t\mathbf{M}_t^{-1}\mathbf{z}_a. \end{aligned} \tag{13}$$

Finally, CascadeHybrid displays the ranked list $\mathcal{R}_t$ to the user and collects click feedback (line 17–27). Since CascadeHybrid only accepts one click, we use $c_t \in [K+1]$ to indicate the position of the click;[3] $c_t = K+1$ indicates that no item in $\mathcal{R}_t$ is clicked.

3.4 Computational complexity

The main computational cost of Algorithm 1 is incurred by computing matrix inverses, which is cubic in the dimensions of the matrix. However, in practice, we can use the Woodbury matrix identity [8] to update $\mathbf{H}_t^{-1}$ and $\mathbf{M}_t^{-1}$ instead of $\mathbf{H}_t$ and $\mathbf{M}_t$, which is square in the dimensions of the matrix. Thus, computing the UCB of each

[3] For multiple-click cases, we only consider the first click and keep the rest of CascadeHybrid the same.

item is $O(m^2 + d^2)$. As CascadeHybrid greedily chooses K items out of L, the per-step computational complexity of CascadeHybrid is $O(LK(m^2 + d^2))$.

4 EXPERIMENTS

This section starts with the experimental setup, where we first introduce the datasets, click simulator and baselines. After that we report our experimental results.

4.1 Experimental setup

Off-policy evaluation [21] is an approach to evaluate interaction algorithms without live experiments. However, in our problem, the action space is exponential in K, which is too large for commonly used off-policy evaluation methods. As an alternative, we evaluate the CascadeHybrid in a simulated interaction environment, where the simulator is built based on offline datasets. This is a commonly used evaluation setup in the literature [11, 16, 35].

Datasets. We evaluate CascadeHybrid on two datasets: MovieLens 20M [10] and Yahoo.[4] The MovieLens dataset contains 20M ratings on 27k movies by 138k users, with 20 genres.[5] Each movie belongs to at least one genre. The Yahoo dataset contains over 700M ratings of 136k songs given by 1.8M users and genre attributes of each song; we consider the top level attribute, which has 20 different genres; each song belongs to a single genre. All the ratings in the two datasets are on a 5-point scale. All movies and songs are considered as items and genres are considered as topics.

Data preprocessing. We follow the data preprocessing approach in [11, 22, 35]. First, we extract the 1k most active users and the 1k most rated items. Let $\mathcal{U} = [1000]$ be the user set, and $\mathcal{D} = [1000]$ be the item set. Then, the ratings are mapped onto a binary scale: rating 5 is converted to 1 and others to 0. After this mapping, in the MovieLens dataset, about 7% of the user-item pairs get rating 1, and, in the Yahoo dataset, about 11% of user-item pairs get rating 1. Then, we use the matrix $\mathbf{F} \in \{0, 1\}^{|\mathcal{U}|\times|\mathcal{D}|}$ to capture the converted ratings and $\mathbf{G} \in \{0, 1\}^{|\mathcal{D}|\times d}$ to record the items and topics, where d is the number of topics and each entry $\mathbf{G}_{jk} = 1$ indicates that item j belongs to topic k.

Click simulator. In our experiments, the click simulator follows the cascade assumption, and considers both item relevance and diversity of the list. To design such a simulator, we combine the simulators used in [22] and [11]. Because of the cascading assumption, we only need to define the way of computing attraction probabilities of items in a list.

We first divide the users into training and test groups evenly, i.e., $\mathbf{F}_{train}$ and $\mathbf{F}_{test}$. The training group is used to estimate features of items used by online algorithms, while the test group are used to define the click simulator. This is to mimic the real-world scenarios that online algorithms estimate user preferences without knowing the perfect topic coverage of items. Then, we follow [22] to obtain the relevance part of the attraction probability, i.e., $\mathbf{z}$ and $\boldsymbol{\beta}^*$, and the process in [11] to get the topic coverage of items, $\mathbf{x}$, and the user preferences on topics, $\boldsymbol{\theta}^*$.

In particular, the relevance features $\mathbf{z}$ are obtained by conducting singular-value decomposition on $\mathbf{F}_{train}$. We pick the 10 largest singular values and thus the dimension of relevance features is $m = 10$. Then, we normalize each relevance feature by the transformation: $\mathbf{z}_a \leftarrow \frac{\mathbf{z}_a}{\|\mathbf{z}_a\|_2}$, where $\|\mathbf{z}_a\|_2$ is the L2 norm of $\mathbf{z}_a$. The user preference $\boldsymbol{\beta}^*$ is computed by solving the least square on $\mathbf{F}_{test}$ and then $\boldsymbol{\beta}^*$ is normalized by the same transformation. Note that $\forall a \in \mathcal{D} : \mathbf{z}_a^T \boldsymbol{\beta}^* \in [0, 1]$, since $\|\mathbf{z}_a\|_2 = 1$ and $\|\boldsymbol{\beta}^*\|_2 = 1$.

Then, we follow the process in [11]. If item a belongs to topic j, we compute the topic coverage of item a to topic j as the quotient of the number of users rating item a to be attractive to the number of users who rate at least one item in topic j to be attractive:

$$\mathbf{x}_{a,j} = \frac{\sum_{u\in\mathcal{U}} F_{u,a} G_{a,j}}{\sum_{u\in\mathcal{U}} \mathbb{1}\{\exists a' \in \mathcal{D} : F_{u,a'} G_{a',j} > 0\}}. \tag{14}$$

Given user u, the preference for topic j is computed as the number of items rated to be attractive in topic j over the number of items in all topics rated by u to be attractive:

$$\boldsymbol{\theta}_j^* = \frac{\sum_{a\in\mathcal{D}} F_{u,a} G_{a,j}}{\sum_{j'\in[d]} \sum_{a'\in\mathcal{D}} F_{u,a'} G_{a',j'}}. \tag{15}$$

For some cases, we may have $\exists a : \sum_{j\in[d]} \mathbf{x}_{a,j} > 1$ and thus $\mathbf{x}_a^T \boldsymbol{\theta}^* > 1$. However, given the high sparsity in our datasets, we have $\mathbf{x}_a^T \boldsymbol{\theta}^* \in [0, 1]$ for all items during our experiments.

Finally, we combine the two parts and obtain the attraction probability used in our click simulator. To simulate different types of user preferences, we introduce a trade-off parameter $\lambda \in [0, 1]$, which is unknown to online algorithms, and compute the attraction probability of the ith item in $\mathcal{R}$ as follows:

$$\alpha(\mathcal{R}(i)) = \lambda \mathbf{z}_{\mathcal{R}(i)}^T \boldsymbol{\beta}^* + (1 - \lambda) \boldsymbol{\omega}_{\mathcal{R}(i)}^T \boldsymbol{\theta}^*. \tag{16}$$

By changing the value of λ, we simulate different types of user preference: a larger value of λ means that the user prefers items to be relevant; a smaller value of λ means that the user prefers the topics in the ranked list to be diverse.

Baselines. We compare CascadeHybrid to two online algorithms, each of which has two configurations. In total, we have four baselines, namely CascadeLinUCB and CascadeLinUCBFull [35], and CascadeLSB and CascadeLSBFull [11]. The first two only consider relevance ranking. The differences are that CascadeLinUCB takes $\mathbf{z}$ as the features, while CascadeLinUCBFull takes $\{\mathbf{x}, \mathbf{z}\}$ as the features. CascadeLSB and CascadeLSBFull only consider the result diversification, where CascadeLSB takes $\mathbf{x}$ as features, while CascadeLSBFull takes $\{\mathbf{x}, \mathbf{z}\}$ as features. We expect that CascadeLinUCB and CascadeLinUCBFull perform well when $\lambda \to 1$, and that CascadeLSB and CascadeLSBFull perform well when $\lambda \to 0$. For all baselines, we set the exploration parameter $\gamma = 1$ and the learning rate to 1. This parameter setup is used in [32], which leads to better empirical performance. We also set $\gamma = 1$ for CascadeHybrid.

We report the cumulative regret, Eq. (9), within 50k steps, called *n-step regret.* The n-step regret is commonly used to evaluate bandit algorithms [11, 18, 32, 35]. In our setup, it measures the difference in number of received clicks between the oracle that knows the ideal β^* and θ^* and the online algorithm, e.g., CascadeHybrid, in n steps. The lower regret means the more clicks received by the algorithm. We conduct our experiments with 500 users from the

[4] R2 - Yahoo! Music User Ratings of Songs with Artist, Album, and Genre Meta Information, v. 1.0 https://webscope.sandbox.yahoo.com/catalog.php?datatype=r
[5] In both datasets, one of the 20 genres is called *unknown.*

test group and 2 repeats per user. In total, the results are averaged over 1k repeats. We also include the standard errors of our estimates. To show the impact of different factors on the performance of online LTR algorithms, we choose $\lambda \in \{0.0, 0.1, \ldots, 1.0\}$, the number of positions $K \in \{5, 10, 15, 20\}$, and the number of topics $d \in \{5, 10, 15, 20\}$. For the number of topics d, we choose the topics with the maximum number of items.

4.2 Experimental results

We first study the movie recommendation task on the MovieLens dataset and show the results in Fig. 1. The top row shows the impact of λ, where we fix $K = 10$ and $d = 20$. λ is a trade-off parameter in our simulation. It balances the relevance and diversity, and is unknown to online algorithms. Choosing a small λ, the simulated user prefers recommended movies in the list to be relevant, while choosing a large λ, the simulated user prefers the recommended movies to be diverse. As shown in the top row, CascadeHybrid outperforms all baselines when $\lambda \in \{0.1, 0.2, \ldots 0.8\}$, and only loses to the particularly designed baselines (CascadeLSB and CascadeLinUCB) with small gaps in some extreme cases where they benefit most. This is reasonable since they have fewer parameters to be estimated than CascadeHybrid. In all cases, CascadeHybrid has lower regret than CascadeLinUCBFull and CascadeLSBFull that work with the same features as CascadeHybrid. This result indicates that including more features in CascadeLSB and CascadeLinUCB is not sufficient to capture both item relevance and result diversification.

The second row in Fig. 1 shows the impact of different numbers of topics, where we fix $\lambda = 0.5$ and $K = 10$. We see that CascadeHybrid outperforms all baselines with large gaps. In the last plot of the middle row, we see that the gap of regret between CascadeHybrid and CascadeLinUCBFull decreases with larger values of d. This is because, on the MovieLens dataset, when d is small, a user tends to prefer a diverse ranked list: when d is small, an item is more likely to belong to only one topic, and each entry of $\boldsymbol{\theta}^*_j$ becomes relatively larger since $\sum_{j \in [d]} \boldsymbol{\theta}^*_j = 1$. And given an item a and a set $\mathcal{S}$, the difference between $\Delta(a|\mathcal{S})^T \boldsymbol{\theta}^*$ and $\Delta(a|\emptyset)^T \boldsymbol{\theta}^*$ is large. This behavior is also confirmed by the fact that CascadeLinUCBFull outperforms CascadeLSBFull for large d while they perform similarly for small d. Finally, we study the impact of the number of positions on the regret. The results are displayed in the bottom row in Fig. 1, where we choose $\lambda = 0.5$ and $d = 20$. Again, we see that CascadeHybrid outperforms baselines with large gaps.

Next, we report the results on the Yahoo dataset in Fig. 2. We follow the same setup as for the MovieLens dataset and observe a similar behavior. CascadeHybrid has slightly higher regret than the best performing baselines in three cases: CascadeLSB when $\lambda = 0$ and CascadeLinUCB when $\lambda \in \{0.9, 1\}$. Note that these are relatively extreme cases, where the particularly designed baselines can benefit most. Meanwhile, CascadeLSB and CascadeLinUCB do not generalize well with different λs. In all setups, CascadeHybrid has lower regret than CascadeLSBFull and CascadeLinUCBFull, which confirms our hypothesis that the hybrid model has benefit in capturing both relevance and diversity.

5 ANALYSIS

5.1 Performance guarantee

Since CascadeHybrid competes with the greedy benchmark, we focus on the η-scaled expected n-step regret which is defined as:

$$R_\eta(n) = \sum_{t=1}^{n} \mathbb{E}\left[\eta r(\mathcal{R}^*, \boldsymbol{\alpha}) - r(\mathcal{R}_t, A_t)\right], \tag{17}$$

where $\eta = (1 - \frac{1}{e}) \max\{\frac{1}{K}, 1 - \frac{K-1}{2}\alpha_{max}\}$. This is a reasonable metric, since computing the optimal $\mathcal{R}^*$ is computationally inefficient. A similar scaled regret has previously been used in diversity problems [11, 27, 32]. For simplicity, we write $\mathbf{w}^* = [\boldsymbol{\theta}^{*T}, \boldsymbol{\beta}^{*T}]^T$. Then, we bound the η-scaled regret of CascadeHybrid as follows:

Theorem 1. *For $\|\mathbf{w}^*\|_2 \leq 1$ and any*

$$\gamma \geq \sqrt{(m+d)\log\left(1 + \frac{nK}{m+d}\right) + 2\log(n)} + \left\|\mathbf{w}^*\right\|_2, \tag{18}$$

we have

$$R_\eta(n) \leq 2\gamma\sqrt{2nK(m+d)\log\left(1 + \frac{nK}{m+d}\right)} + 1. \tag{19}$$

Combining Eqs. (18) and (19), we have $R_\eta(n) = \tilde{O}((m+d)\sqrt{Kn})$, where the $\tilde{O}$ notation ignores logarithmic factors. Our bound has three characteristics: (1) Theorem 1 states a gap-free bound, where the factor $\sqrt{n}$ is considered near optimal; (2) This bound is linear in the number of features, which is a common dependence in learning bandit algorithms [1]; and (3) Our bound is $\tilde{O}(\sqrt{K})$ lower than other bounds for linear bandit algorithms in CB [11, 35]. We include a proof of Theorem 1 in Section 5.2. We use a similar strategy to decompose the regret as in [11, 35], but we have a better analysis on how to sum up the regret of individual items. Thus, our bound depends on $\tilde{O}(\sqrt{K})$ rather than $\tilde{O}(K)$. We believe that our analysis can be applied to both CascadeLinUCB and CascadeLSB, and then show that their regret is actually bounded by $\tilde{O}(\sqrt{K})$ rather than $\tilde{O}(K)$.

5.2 Proof of Theorem 1

We first define some additional notation. We write $\mathbf{w}^* = [\boldsymbol{\theta}^{*T}, \boldsymbol{\beta}^{*T}]^T$. Given a ranked list $\mathcal{R}$ and $a = \mathcal{R}(i)$, we write $\boldsymbol{\phi}_a = [\boldsymbol{\omega}_a^T, \mathbf{z}_a^T]^T$. With the $\boldsymbol{\phi}_a$ and $\mathbf{w}^*$ notation, CascadeHybrid can be viewed as an extension of CascadeLSB, where two submodular functions instead of one are used in a single model. We write $\mathbf{O}_t = \mathbf{I}_{m+d} + \sum_{i=1}^{t-1} \sum_{a \in O_i} \boldsymbol{\phi}_a \boldsymbol{\phi}_a^T$ as the collected features in t steps, and $\mathcal{H}_t$ as the collected features and clicks up to step t. We write $\mathcal{R}^i = (\mathcal{R}(1), \ldots, \mathcal{R}(i))$. Then, the confidence bound in Section 3.3 on the i-th item in $\mathcal{R}$ can be re-written as:

$$s(\mathcal{R}^i) = \boldsymbol{\phi}_{\mathcal{R}(i)}^T \mathbf{O}_t^{-1} \boldsymbol{\phi}_{\mathcal{R}(i)}. \tag{20}$$

Let $\Pi(\mathcal{D}) = \bigcup_{i=1}^{L} \Pi_i(\mathcal{D})$ be the set of all ranked lists of $\mathcal{D}$ with length $[L]$, and $\kappa : \Pi(\mathcal{D}) \rightarrow [0, 1]$ be an arbitrary list function. For any $\mathcal{R} \in \Pi(\mathcal{D})$ and any κ, we define

$$f(\mathcal{R}, \kappa) = 1 - \prod_{i=1}^{|\mathcal{R}|}(1 - \kappa(\mathcal{R}^i)). \tag{21}$$

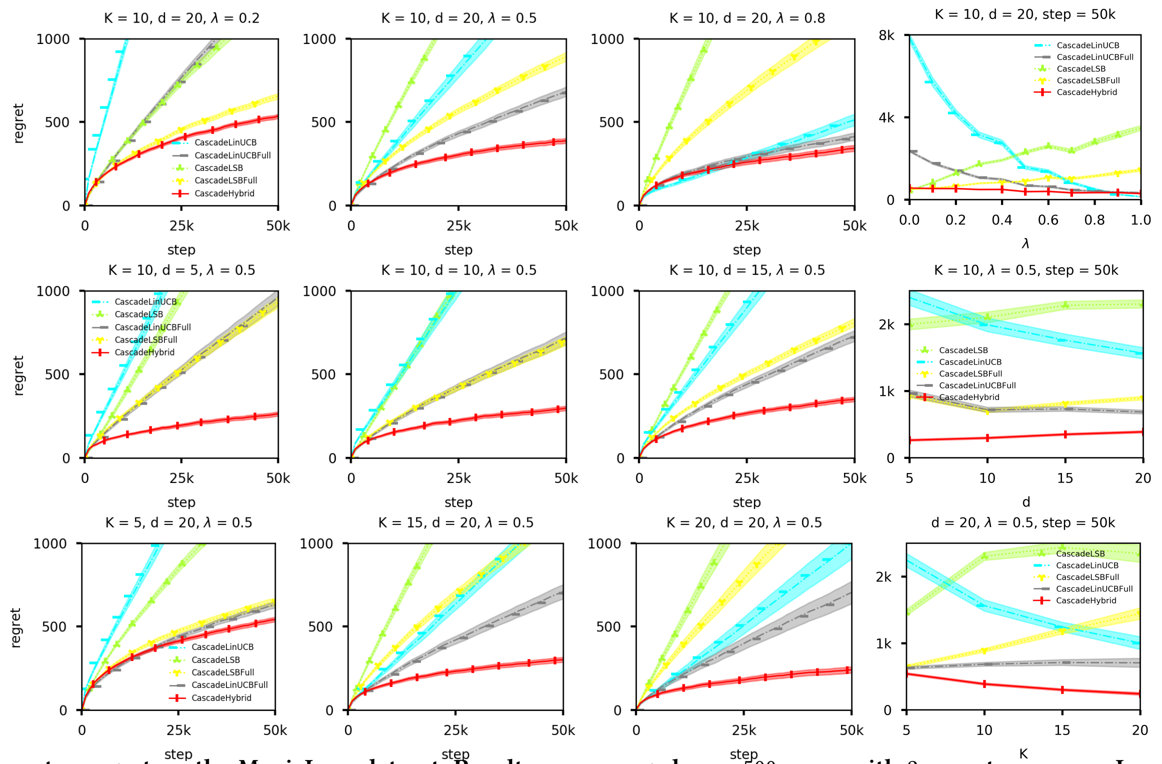

Figure 1: n-step regret on the MovieLens dataset. Results are averaged over 500 users with 2 repeats per user. Lower regret means more clicks received by the algorithm during the online learning. Shaded areas are the standard errors of estimates.

We define upper and lower confidence bounds, and κ as:

$$\begin{aligned} u_t(\mathcal{R}) &= \mathrm{F}_{[0,1]}[\boldsymbol{\phi}^T_{\mathcal{R}(l)}\hat{\mathbf{w}}_t + s(\mathcal{R}^l)] \\ l_t(\mathcal{R}) &= \mathrm{F}_{[0,1]}[\boldsymbol{\phi}^T_{\mathcal{R}(l)}\hat{\mathbf{w}}_t - s(\mathcal{R}^l)] \\ \kappa(R) &= \boldsymbol{\phi}^T_{\mathcal{R}(l)}\mathbf{w}^*, \end{aligned} \tag{22}$$

where $l = |\mathcal{R}|$ and $\mathrm{F}_{[0,1]}[\cdot] = \max(0, \min(1, \cdot))$. With the definitions in Eq. (22), $f(\mathcal{R}, \kappa) = r(\mathcal{R}, \boldsymbol{\alpha})$ is the reward of list $\mathcal{R}$.

Proof. Let $g_t = \{l_t(\mathcal{R}) \leq \kappa(\mathcal{R}) \leq u_t(\mathcal{R}), \forall \mathcal{R} \in \Pi(\mathcal{D})\}$ be the event that the attraction probabilities are bounded by the lower and upper confidence bound, and $\bar{g}_t$ be the complement of g_t. We have

$$\begin{aligned} &\mathbb{E}\left[\eta r(\mathcal{R}^*, \boldsymbol{\alpha}) - r(\mathcal{R}_t, \mathbf{A}_t)\right] = \mathbb{E}\left[\eta f(\mathcal{R}^*, \kappa) - f(\mathcal{R}_t, \kappa)\right] \\ &\overset{(a)}{\leq} P(g_t)\mathbb{E}\left[\eta f(\mathcal{R}^*, \kappa) - f(\mathcal{R}_t, \kappa)\right] + P(\bar{g}_t) \\ &\overset{(b)}{\leq} P(g_t)\mathbb{E}\left[\eta f(\mathcal{R}^*, u_t) - f(\mathcal{R}_t, \kappa)\right] + P(\bar{g}_t) \\ &\overset{(c)}{\leq} P(g_t)\mathbb{E}\left[f(\mathcal{R}_t, u_t) - f(\mathcal{R}_t, \kappa)\right] + P(\bar{g}_t), \end{aligned} \tag{23}$$

where (a) holds because $\mathbb{E}\left[\eta f(\mathcal{R}^*, \kappa) - f(\mathcal{R}_t, \kappa)\right] \leq 1$, (b) holds because under event g_t we have $f(\mathcal{R}, l_t) \leq f(\mathcal{R}, \kappa) \leq f(\mathcal{R}, u_t)$, $\forall \mathcal{R} \in \Pi(\mathcal{D})$, and (c) holds by the definition of the η-approximation, where we have

$$\eta f(\mathcal{R}^*, u_t) \leq \max_{\mathcal{R} \in \Pi_K(\mathcal{D})} \eta f(\mathcal{R}, u_t) \leq f(\mathcal{R}_t, u_t). \tag{24}$$

By the definition of the list function $f(\cdot, \cdot)$ in Eq. (21), we have

$$\begin{aligned} &f(\mathcal{R}_t, u_t) - f(\mathcal{R}_t, \kappa) \\ &= \prod_{k=1}^{K}(1 - \kappa(\mathcal{R}_t^k)) - \prod_{k=1}^{K}(1 - u_t(\mathcal{R}_t^k)) \\ &\overset{(a)}{=} \sum_{k=1}^{K}\left[\prod_{i=1}^{k-1}(1 - \kappa(\mathcal{R}_t^i))\right](u_t(\mathcal{R}_t^k) - \kappa(\mathcal{R}_t^k))\left[\prod_{j=k+1}^{K}(1 - u(\mathcal{R}_t^j))\right] \\ &\overset{(b)}{\leq} \sum_{k=1}^{K}\left[\prod_{i=1}^{k-1}(1 - \kappa(\mathcal{R}_t^i))\right](u_t(\mathcal{R}_t^k) - \kappa(\mathcal{R}_t^k)), \end{aligned} \tag{25}$$

where (a) follows from Lemma 1 in [35] and (b) is because of the fact that $0 \leq \kappa(\mathcal{R}_t) \leq u_t(\mathcal{R}_t) \leq 1$. We then define the event $h_{ti} = \{\text{item } \mathcal{R}_t(i) \text{ is observed}\}$, where we have $\mathbb{E}\left[\mathbb{1}(h_{ti})\right] = \prod_{k=1}^{i-1}(1 - \kappa(\mathcal{R}_t^k))$. For any $\mathcal{H}_t$ such that g_t holds, we have

$$\begin{aligned} &\mathbb{E}\left[f(\mathcal{R}_t, u_t) - f(\mathcal{R}_t, \kappa) \mid \mathcal{H}_t\right] \\ &\leq \sum_{i=1}^{K}\mathbb{E}\left[\mathbb{1}(h_{ti}) \mid \mathcal{H}_t\right](u_t(\mathcal{R}_t^i) - l_t(\mathcal{R}_t^i)) \\ &\overset{(a)}{\leq} 2\gamma\mathbb{E}\left[\mathbb{1}(h_{ti})\sum_{i=1}^{K}\sqrt{s(\mathcal{R}_t^i)} \mid \mathcal{H}_t\right] \overset{(b)}{\leq} 2\gamma\mathbb{E}\left[\sum_{i=1}^{\min(K, c_t)}\sqrt{s(\mathcal{R}_t^i)} \mid \mathcal{H}_t\right], \end{aligned} \tag{26}$$

where inequality (a) follows from the definition of u_t and l_t in Eq. (22), and inequality (b) follows from the definition of h_{ti}. Now,

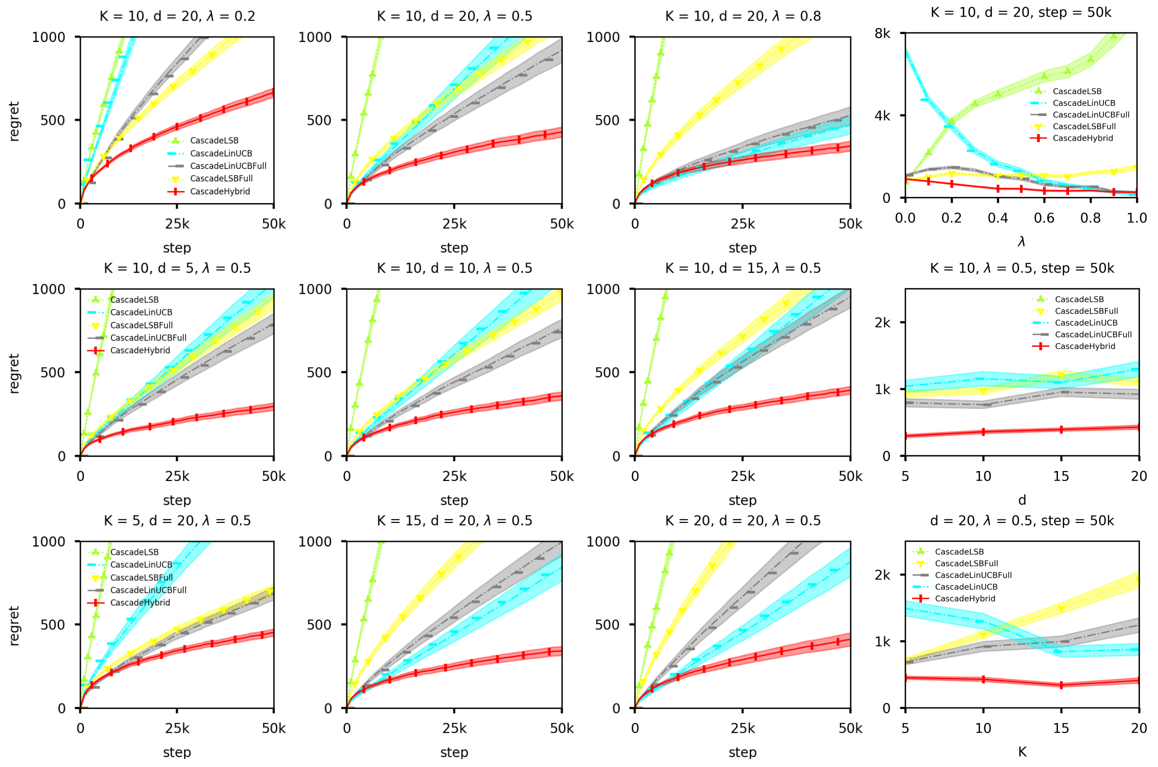

Figure 2: n-step regret on the Yahoo dataset. Results are averaged over 500 users with 2 repeats per user. Lower regret means more clicks received by the algorithm during the online learning. Shaded areas are the standard errors of estimates.

together with Eqs. (17) and (26) and Section 5.2, we have

$$\begin{aligned} R_\eta(n) &= \sum_{t=1}^{n} \mathbb{E}\left[\eta r(\mathcal{R}^*, \boldsymbol{\alpha}) - r(\mathcal{R}_t, \mathrm{A}_t)\right] \\ &\leq \sum_{t=1}^{n} \left[2\gamma \mathbb{E}\left[\sum_{i=1}^{\min(K, c_t)} \sqrt{s(\mathcal{R}_t^i)} \mid g_t \right] P(g_t) + P(\bar{g}_t) \right] \qquad (27) \\ &\leq 2\gamma \mathbb{E}\left[\sum_{t=1}^{n} \sum_{i=1}^{K} \sqrt{s(\mathcal{R}_t^i)} \right] + \sum_{t=1}^{n} p(\bar{g}_t). \end{aligned}$$

For the first term in Eq. (27), we have

$$\sum_{t=1}^{n} \sum_{i=1}^{K} \sqrt{s(\mathcal{R}_t^i)} \overset{(a)}{\leq} \sqrt{nK \sum_{t=1}^{n} \sum_{i=1}^{K} s(\mathcal{R}_t^i)} \overset{(b)}{\leq} \sqrt{nK 2 \log det(\mathbf{O}_t)}, \quad (28)$$

where inequality (a) follows from the Cauchy-Schwarz inequality and (b) follows from Lemma 5 in [32]. Note that $\log det(\mathbf{O}_t) \leq (m + d) \log(K(1 + n/(m + d)))$, which can be obtained by the determinant and trace inequality, and together with Eq. (28):

$$\sum_{t=1}^{n} \sum_{i=1}^{K} \sqrt{s(\mathcal{R}_t^i)} \leq \sqrt{2nK(m + d) \log(K(1 + \frac{n}{m + d}))}. \quad (29)$$

For the second term in Eq. (27), by Lemma 3 in [11], we have $P(\bar{g}_t) \leq 1/n$ for any γ that satisfies Eq. (18). Thus, together with Eqs. (27) to (29), we have

$$R_\eta(n) \leq 2\gamma \sqrt{2nK(m + d) \log(1 + \frac{nK}{m + d})} + 1.$$

This concludes the proof of Theorem 1. □

6 RELATED WORK

The literature on offline learning to rank (LTR) methods that account for position bias and diversity is too broad to review in detail. We refer readers to [2] for an overview. In this section, we mainly review online LTR papers that are closely related to our work, i.e., stochastic click bandit models.

Online LTR in a stochastic click models has been well-studied [15, 16, 18, 19, 22, 29, 30, 32, 33, 35]. Previous work can be categorized into two groups: feature-free models and feature-rich models. Algorithms from the former group use a tabular representation on items and maintain an estimator for each item. They learn inefficiently and are limited to the problem with a small number of item candidates. In this paper, we focus on the ranking problem with a large number of items. Thus, we do not consider feature-free model in the experiments.

Feature-rich models learn efficiently in terms of the number of items. They are suitable for large-scale ranking problems. Among them, ranked bandits [29, 30] are early approaches to online LTR. In ranked bandits, each position is model as a MAB and diversity of results is addressed in the sense that items ranked at lower positions are less likely to be clicked than those at higher positions, which is different from the topical diversity as we study. Also, ranked bandits do not consider the position bias and are suboptimal in the problem where a user browse different possition unevenly, e.g., CM [16]. LSBGreedy [32] and C^2UCB [27] use submodular functions to solve the online diverse LTR problem. They assume that the user browses all displayed items and, thus, do not consider the position bias either.

Our work is closely related to CascadeLinUCB [35] and CascadeLSB [11], the baselines in our experiments, and can be viewed as a combination of both. CascadeLinUCB solves the relevance ranking in the CM and assumes the attraction probability is a linear combination of features. CascadeLSB is designed for result diversification and assumes that the attraction probability is computed as a submodular function; see Eq. (6). In our CascadeHybrid, the attraction probability is a hybrid of both; see Eq. (7). Thus, CascadeHybrid handles both relevance ranking and result diversification. RecuRank [22] is a recently proposed algorithm that aims at learning the optimal list in term of item relevance in most click models. However, to achieve this task, RecuRank requires a lot of randomly shuffled lists and is outperformed by CascadeLinUCB in the CM [22].

The hybrid of a linear function and a submodular function has been used in solving combinatorial semi-bandits. Perrault et al. [26] use a linear set function to model the expected reward of arm, and use the submodular function to compute the *exploration bonus*. This is different from our hybrid model, where both the linear and submodular functions are used to model the attraction probability and the confident bound is used as the exploration bonus.

7 CONCLUSION

In real world interactive systems, both relevance of individual items and topical diversity of result lists are critical factors in user satisfaction. In order to better meet users' information needs, we propose a novel online LTR algorithm that optimizes both factors in a hybrid fashion. We formulate the problem as cascade hybrid bandits (CHB), where the attraction probability is a hybrid function that combines a function of relevance features and a submodular function of topic features. CascadeHybrid utilizes a hybrid model as a scoring function and the UCB policy for exploration. We provide a gap-free bound on the η-scaled n-step regret of CascadeHybrid, and conduct experiments on two real-world datasets. Our empirical study shows that CascadeHybrid outperforms two existing online LTR algorithms that exclusively consider either relevance ranking or result diversification.

In future work, we intend to conduct experiments on live systems, where feedback is obtained from multiple users so as to test whether CascadeHybrid can learn across users. Another direction is to adapt Thompson sampling [31] to our hybrid model, since Thompson sampling generally outperforms UCB-based algorithms [20, 35].

ACKNOWLEDGMENTS

This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) under project nr 612.001.551. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved Algorithms for Linear Stochastic Bandits. In *NIPS*. 2312–2320.

[2] Charu C. Aggarwal. 2016. *Recommender Systems: The Textbook*. Springer.

[3] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying Search Results. In *WSDM*. 5–14.

[4] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. 2015. Optimal Greedy Diversity for Recommendation. In *IJCAI*. 1742–1748.

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47 (2002), 235–256.

[6] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool.

[7] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-Bias Models. In *WSDM*. 87–94.

[8] Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations* (3 ed.). Johns Hopkins, Baltimore, MD.

[9] Artem Grotov and Maarten de Rijke. 2016. Online Learning to Rank for Information Retrieval: SIGIR 2016 Tutorial. In *SIGIR*. 1215–1218.

[10] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2015), 19:1–19:19.

[11] Gaurush Hiranandani, Harvineet Singh, Prakhar Gupta, Iftikhar Ahamath Burhanuddin, Zheng Wen, and Branislav Kveton. 2019. Cascading Linear Submodular Bandits: Accounting for Position Bias and Diversity in Online Learning to Rank. In *UAI*.

[12] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. Reusing Historical Interaction Data for Faster Online Learning to Rank for IR. In *WSDM*. 183–192.

[13] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2011. Balancing exploration and exploitation in learning to rank online. In *ECIR*. 251–263.

[14] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. In *SIGIR*. 15–24.

[15] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2016. DCM Bandits: Learning to Rank with Multiple Clicks. In *ICML*. 1215–1224.

[16] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. 2015. Cascading Bandits: Learning to Rank in the Cascade Model. In *ICML*. 767–776.

[17] Tor Lattimore and Csaba Szepesvári. 2020. *Bandit Algorithms*. Cambridge University Press.

[18] Chang Li and Maarten de Rijke. 2019. Cascading Non-stationary Bandits: Online Learning to Rank in the Non-stationary Cascade Model. In *IJCAI*. 2859–2865.

[19] Chang Li, Branislav Kveton, Tor Lattimore, Ilya Markov, Maarten de Rijke, Csaba Szepesvári, and Masrour Zoghi. 2019. BubbleRank: Safe Online Learning to Re-Rank via Implicit Click Feedback. In *UAI*.

[20] Chang Li, Ilya Markov, Maarten de Rijke, and Masrour Zoghi. 2020. MergeDTS: A Method for Effective Large-scale Online Ranker Evaluation. *ACM Transactions on Information Systems* 38, 4 (August 2020).

[21] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation. In *WWW*. 661–670.

[22] Shuai Li, Tor Lattimore, and Csaba Szepesvári. 2019. Online Learning to Rank with Features. In *ICML*. 3856–3865.

[23] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[24] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An Analysis of Approximations for Maximizing Submodular Set Functions—I. *Mathematical Programming* 14, 1 (1978), 265–294.

[25] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable Unbiased Online Learning to Rank. In *CIKM*. 1293–1302.

[26] Pierre Perrault, Vianney Perchet, and Michal Valko. 2019. Exploiting Structure of Uncertainty for Efficient Matroid Semi-bandits. In *ICML*. PMLR, 5123–5132.

[27] Lijing Qin, Shouyuan Chen, and Xiaoyan Zhu. 2014. Contextual Combinatorial Bandit and its Application on Diversified Online Recommendation. In *SDM*. 461–469.

[28] Lijing Qin and Xiaoyan Zhu. 2013. Promoting Diversity in Recommendation by Entropy Regularizer. In *IJCAI*. 2698–2704.

[29] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning Diverse Rankings with Multi-Armed Bandits. In *ICML*. 784–791.

[30] Alex Slivkins, Filip Radlinski, and Sreenivas Gollapudi. 2010. Learning Optimally Diverse Rankings over Large Document Collections. In *ICML*. 983–990.

[31] William R. Thompson. 1933. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika* 25 (1933), 285–294.

[32] Yisong Yue and Carlos Guestrin. 2011. Linear Submodular Bandits and their Application to Diversified Retrieval. In *NIPS*. 2483–2491.
[33] Masrour Zoghi, Tomáš Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2017. Online Learning to Rank in Stochastic Click Models. In *ICML*. 4199–4208.
[34] Masrour Zoghi, Tomáš Tunys, Lihong Li, Damien Jose, Junyan Chen, Chun Ming Chin, and Maarten de Rijke. 2016. Click-based Hot Fixes for Underperforming Torso Queries. In *SIGIR*. 195–204.
[35] Shi Zong, Hao Ni, Kenny Sung, Nan Rosemary Ke, Zheng Wen, and Branislav Kveton. 2016. Cascading Bandits for Large-Scale Recommendation Problems. In *UAI*.