Cascading Linear Submodular Bandits: Accounting for Position Bias and Diversity in Online Learning to Rank

Gaurush Hiranandani[*][†]
University of Illinois at Urbana-Champaign

Harvineet Singh[†]
New York University

Prakhar Gupta[†]
Carnegie Mellon University

Iftikhar Ahamath Burhanuddin
Adobe Research

Zheng Wen
Adobe Research

Branislav Kveton
Google Research

Abstract

Online learning, *position bias*, and *diversified retrieval* are three crucial aspects in designing ranking systems based on user clicks. One simple click model which explains the position bias is the *cascade model*. Many online learning variants of the cascade model have been proposed, but none so far captures diversity. Similarly, online learning to rank methods which capture diversity do not handle position bias. Motivated by these limitations, we propose a novel click model, and the associated online learning variant to address both position bias and diversified retrieval in a unified framework. Despite the objective function not being a standard submodular set function, we prove that a greedy-approach is a reasonable approximation. We then propose, *CascadeLSB*, an algorithm whose goal is to recommend K most attractive items from a large set of L items with the attractiveness of items being modeled as a submodular utility function. We analyze the algorithm and prove a gap-free upper bound on its n-step regret. We comprehensively evaluate *CascadeLSB* on synthetic and real datasets, where it outperforms all the baselines.

1 INTRODUCTION

Learning to rank (offline) is a mature field (Agichtein et al., 2006; Liu et al., 2009; Ricci et al., 2011) but of late the focus has shifted to *online learning to rank* methods because of their comparatively promising results (Grotov and de Rijke, 2016). This is not completely unexpected, because offline methods are inherently limited by past data, which are biased due to being collected by production policies. Online methods overcome this bias by sequential experimentation with user feedback.

User feedback, such as clicks, has been an invaluable source of information to train online learning to rank models (Grotov and de Rijke, 2016). A model that learns from user feedback should respect two aspects of user behavior. First, it should handle *position bias* (Craswell et al., 2008), which refers to the situation when lower-ranked items are less likely to be clicked due to higher ranked items. Second, it should capture *diversified retrieval* (Carbonell and Goldstein, 1998) in order to recommend a set of non-redundant items that maximizes coverage of different tastes of the user.

One well-studied click model which stands out due to its simplicity in explaining the position bias is the *Cascade model* (Craswell et al., 2008). Here, the user examines the recommended list of K items from the first item to the last, clicks on the first attractive item, and does not examine the rest. Items before the click are deemed to be *unattractive*, because the user examines these items but does not click on them. The clicked item is *attractive* to the user, and the rest of the items are *unexamined* by the user. Recently, several online learning to rank variants in the cascade model have been proposed (Kveton et al., 2015a; Combes et al., 2015; Zong et al., 2016; Li et al., 2016). These variants, however, model the attractiveness of the items in different ways such as a "raw" preference of the user (Kveton et al., 2015a) or a weighted linear function of the features (Zong et al., 2016). Unfortunately, none of the approaches capture diversity.

The concept of diversity was introduced in online learning to rank in ranked bandits (Radlinski et al., 2008) and later extended in Streeter and Golovin (2009); Streeter et al. (2009). However, these approaches are feature-free models and thus do not generalize easily. Practical diversity-driven online learning frameworks (Yue and Guestrin, 2011; Raman et al., 2012) formulate the prob-

[*] Correspondence: gaurush2@illinois.edu. This work was done when all the authors were at Adobe Research.
[†] Equal Contribution.

lem as learning to maximize a submodular function in a feature-based (contextual) bandit setting. However, these frameworks assume semi-bandit feedback i.e. feedback on the entire list of K recommended items, and consequently, do not capture the position bias. Furthermore, when the user does not examine the full list of recommended items and gets "satisfied" in between the list, these approaches lead to unnecessary penalization of the lower-ranked, unexamined items. We show that this is indeed the case via our experiments (Section 8, Section 9).

Building on the limitations of online learning to rank in the above two separate lines of work, this paper makes the following contributions:

- We propose a novel click model, *Cascade-Diverse Click Model (CDCM)* to address both position bias and diversified retrieval in a unified framework (Section 3). Under this model, the objective is to maximize the probability that the user finds an attractive item in the recommended list of K items, where attractiveness of items is modeled as a feature-based gain in submodular utility. Interestingly, the objective function is not a set function as it depends on the order of the items. However, under standard assumptions, we prove that a greedy approach performs reasonably well with an approximation factor, surprisingly, similar to a submodular set function.
- We propose *cascading linear submodular bandits* – an online learning variant of CDCM (Section 4). To the best of our knowledge, this is the first work that studies a top-K recommender problem in the bandit setting with cascading feedback and diversity.
- We propose `CascadeLSB`, a practical algorithm for learning to rank in the cascading linear submodular bandit (Section 5). We analyze this algorithm and derive a $O(\sqrt{n})$ upper bound on its n-step regret (Section 6).
- We comprehensively evaluate `CascadeLSB` on synthetic and several real-world problems and show that it consistently outperforms the baselines, even when our modeling assumptions are violated (Section 8).
- Finally, we discuss a simple extension of `CascadeLSB` (Section 9) to the multiple click scenario, which reinforces the deficiency of existing diversity driven online approaches when only partial feedback is available.

Notation: We define $[n] = \{1, \dots, n\}$ and treat all vectors as column vectors. For any sets A and B, we denote by A^B the set of all vectors whose entries are indexed by B and take values from A.

2 BACKGROUND

This section reviews two *click models* (Chuklin et al., 2015). A click model is a stochastic model that describes how the user interacts with a list of items. More formally, let $E = [L]$ be a *ground set* of L items, such as the set of all web pages or movies. Let $A = (a_1, \dots, a_K) \in \Pi_K(E)$ be a *list of $K \leq L$ recommended items*, where a_k is the k-th recommended item and $\Pi_K(E)$ is the set of all *K-permutations* of the set E. Then the click model outlines how the user *examines* and *clicks* on items in A.

2.1 Cascade Model

The *cascade model* (Craswell et al., 2008) explains a common bias in recommending multiple items, which is that lower ranked items are less likely to be clicked than higher ranked items, as discussed below.

The model is parameterized by L *item-dependent attraction probabilities* $\bar{w} \in [0, 1]^L$. The user examines a recommended list $A \in \Pi_K(E)$ from the first item a_1 to the last a_K. When the user examines item a_k, the item attracts the user with probability $\bar{w}(a_k)$, independently of the other items. If the user is attracted by an item a_k, the user clicks on it and does not examine any of the remaining items. If the user is not attracted by item a_k, the user examines the next item a_{k+1}. The first item is examined with probability one.

Since each item attracts the user independently, the probability that item a_k is examined is $\prod_{i=1}^{k-1}(1 - \bar{w}(a_i))$, and the probability that at least one item in A is attractive is

$$1 - \prod_{k=1}^{K}(1 - \bar{w}(a_k))\,. \tag{1}$$

Clearly, the above objective function is maximized by K most attractive items. Notice that these items are not guaranteed to be diverse, and hence the list may seem repetitive and unpleasant in practice.

2.2 Diverse Click Model - Yue and Guestrin (2011)

Submodularity is well established in diverse recommendations (Carbonell and Goldstein, 1998). In the model of Yue and Guestrin (2011), the probability of clicking on an item depends on the gains in topic coverage by that item and the interests of the user in the covered topics.

Before we discuss this model, we introduce basic terminology. The *topic coverage* by items $S \subseteq E$, $c(S) \in [0, 1]^d$, is a d-dimensional vector whose j-th entry is the coverage of topic $j \in [d]$ by items S. Intuitively, for any two sets S and S', if $c_j(S) > c_j(S')$, then items S cover topic j better than items S'. In particular, $c(S) = (c_1(S), \dots, c_d(S))$, where $c_j(S)$ is a *monotone* and *submodular* function in S for all $j \in [d]$. That is,

$$\forall S \subseteq S' \subseteq E, e \in E : c_j(S \cup \{e\}) \geq c_j(S) \text{ and}$$
$$c_j(S \cup \{e\}) - c_j(S) \ \geq c_j(S' \cup \{e\}) - c_j(S')\,.$$

The *gain in topic coverage* by item e over items S is defined as

$$\Delta(e \mid S) = c(S \cup \{e\}) - c(S)\,. \tag{2}$$

Since $c_j(S) \in [0,1]$ and $c_j(S)$ is monotone in S, $\Delta(e \mid S) \in [0,1]^d$. The *preferences* of the user are a distribution over d topics, which is represented by a vector $\theta^* = (\theta_1^*, \dots, \theta_d^*)$.

In this model, the user examines all items in the recommended list A and is attracted by item a_k with probability

$$\langle \Delta(a_k \mid \{a_1, \dots, a_{k-1}\}), \theta^* \rangle\,, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is the dot product of two vectors. The quantity in (3) is the gain in topic coverage after item a_k is added to the first $k-1$ items weighted by the preferences of the user θ^* over the topics. Roughly speaking, if item a_k is diverse over higher-ranked items in a topic of user's interest, then that item is likely to be clicked. If the user is attracted by item a_k, the user clicks on it. It follows that the expected number of clicks on list A is $\langle c(A), \theta^* \rangle$, where

$$\langle c(A), \theta \rangle = \sum_{k=1}^{K} \langle \Delta(a_k \mid \{a_1, \dots, a_{k-1}\}), \theta \rangle \tag{4}$$

for any list A, preferences θ, and topic coverage c. Yue and Guestrin (2011) optimize this set function of A (4).

This model, however, does not explain the position bias, that lower ranked items are less likely to be clicked. We illustrate this in the following example. Suppose that item 1 completely covers topic 1, $c(\{1\}) = (1,0)$, and that all other items completely cover topic 2, $c(\{e\}) = (0,1)$ for all $e \in E \setminus \{1\}$. Let $c(S) = \max_{e \in S} c(\{e\})$, where the maximum is taken entry-wise, and $\theta^* = (0.5, 0.5)$. Then, item 1 is clicked with probability 0.5 in any list A that contains it, irrespective of its position. This means that the position bias is not modeled well.

3 CASCADE-DIVERSE CLICK MODEL

Our work is motivated by the observation that none of the models in Section 2 capture both position bias and diversity together. Therefore, we propose a new click model, Cascade-Diverse Click Model (CDCM), which addresses both these phenomena. Similar to model in Section 2.2, the diversity in CDCM is over d topics, such as movie genres or restaurant types. The preferences of the user are a distribution over these topics represented by a vector $\theta^* = (\theta_1^*, \dots, \theta_d^*)$.

The user interacts in this model as follows. The user scans a list of K items $A = (a_1, \dots, a_K) \in \Pi_K(E)$ from the first item a_1 to the last a_K, as described in Section 2.1. If the user examines item a_k, the user is attracted by it proportionally to its gains in topic coverage over the first $k-1$ items weighted by the preferences of the user θ^* over the topics. The attraction probability of item a_k is defined in (3). As mentioned in Section 2.2, roughly speaking, if item a_k is diverse over higher-ranked items in a topic of user's interest, then that item is likely to attract the user. If the user is attracted by item a_k, the user clicks on it and does not examine any of the remaining items. If the user is not attracted by item a_k, then the user examines the next item a_{k+1}. The first item in the list is examined with probability one.

We assume that each item attracts the user independently, as in the cascade model (Section 2.1). Under this assumption, the probability that at least one item in A is attractive is $f(A, \theta^*)$, where

$$f(A, \theta) = 1 - \prod_{k=1}^{K} (1 - \langle \Delta(a_k \mid \{a_1, \dots, a_{k-1}\}), \theta \rangle) \tag{5}$$

for any list A, preferences θ, and topic coverage c.

3.1 Optimal List

To the best of our knowledge, the list that maximizes (5) under user preferences θ^*,

$$A^* = \arg\max{}_{A \in \Pi_K(E)} f(A, \theta^*)\,, \tag{6}$$

cannot be computed efficiently. Therefore, we propose a greedy algorithm that maximizes $f(A, \theta^*)$ approximately. The algorithm chooses K items sequentially. The k-th item a_k is chosen such that it maximizes its gain over previously chosen items $a_1, \dots, a_{k-1}$. In particular, for any $k \in [K]$,

$$a_k = \operatorname*{arg\,max}_{e \in E \setminus \{a_1, \dots, a_{k-1}\}} \langle \Delta(e \mid \{a_1, \dots, a_{k-1}\}), \theta^* \rangle \tag{7}$$

We would like to comment on the quality of the above approximation. Although the value of adding any item e diminishes with more previously added items, $f(A, \theta^*)$ is not a set function of A because its value depends on the order of the items in A. Thus, we cannot leverage existing approximation guarantees available for monotonic, submodular set functions. From that standpoint, the following theorem is the first main result of this paper.

Theorem 1 *For any topic coverage c and user preferences θ^*, let A^{greedy} be the solution computed by the greedy algorithm in (7). Then*

$$f(A^{\text{greedy}}, \theta^*) \geq \gamma f(A^*, \theta^*)\,, \tag{8}$$

where $\gamma = (1 - 1/\mathbf{e}) \max\left\{\frac{1}{K}, 1 - \frac{K-1}{2} c_{\max}\right\}$ *with* $c_{\max} = \max_{e \in E} \langle c(\{e\}), \theta^* \rangle$, *denoting the maximum click probability. In other words, the approximation ratio of the greedy algorithm is* γ.

Note that when $c_{\max}$ in Theorem 1 is small, the approximation ratio is close to $1 - 1/\mathbf{e}$. This is common in ranking problems where the maximum click probability $c_{\max}$ tends to be small. In Section 8.1, we observe that our approximation ratio is close to one in practice, which is significantly better than suggested in Theorem 1.

4 Cascading Linear Submodular Bandit

In this section, we present an online learning variant of CDCM (Section 3), which we call a *cascading linear submodular bandit*. An instance of this problem is a tuple (E, c, θ^*, K), where $E = [L]$ represents a ground set of L items, c is the topic coverage function in Section 2.2, θ^* are user preferences in Section 3, and $K \leq L$ is the number of recommended items. The preferences θ^* are unknown to the learning agent.

Our learning agent interacts with the user as follows. At time t, the agent recommends a list of K items $A_t = (a_1^t, \dots, a_K^t) \in \Pi_K(E)$. The attractiveness of item a_k at time t, $w_t(a_k^t)$, is a realization of an independent Bernoulli random variable with mean $\langle \Delta(a_k^t \mid \{a_1^t, \dots, a_{k-1}^t\}), \theta^* \rangle$. The user examines the list from the first item a_1^t to the last a_K^t and clicks on the first attractive item. The *feedback* is the index of the click, $C_t = \min\{k \in [K] : w_t(a_k^t) = 1\}$, where we assume that $\min \emptyset = \infty$. That is, if the user clicks on an item, then $C_t \leq K$; and if the user does not click on any item, then $C_t = \infty$. We say that item e is *examined* at time t if $e = a_k^t$ for some $k \in [\min\{C_t, K\}]$. Note that the attractiveness of all examined items at time t can be computed from C_t. In particular, $w_t(a_k^t) = \mathbb{1}\{C_t = k\}$ for any $k \in [\min\{C_t, K\}]$. The *reward* is defined as $r_t = \mathbb{1}\{C_t \leq K\}$. That is, the reward is one if the user is attracted by at least one item in A_t; and zero otherwise.

The goal of the learning agent is to maximize its expected cumulative reward. This is equivalent to minimizing the expected cumulative regret with respect to the *optimal list* in (6). The regret is formally defined in Section 6.

5 ALGORITHM `CascadeLSB`

We present `CascadeLSB` (Algorithm 1) for solving cascading linear submodular bandit. The algorithm knows the gains in topic coverage $\Delta(e \mid S)$ in (2), for any item $e \in E$ and set $S \subseteq E$. It does not know the user preferences θ^* and estimates them through repeated interactions with the user. It also has two tunable parameters

Algorithm 1 `CascadeLSB`

```
1:  Inputs: Parameters $\sigma > 0$ and $\alpha > 0$ (Section 6)
2:  $M_0 \leftarrow I_d,\ B_0 \leftarrow \mathbf{0}$                    ▷ Initialization
3:  for $t = 1, \dots, n$ do
4:      $\bar{\theta}_{t-1} \leftarrow \sigma^{-2} M_{t-1}^{-1} B_{t-1}$          ▷ Regression estimate
5:      $S \leftarrow \emptyset$    ▷ Recommend list and receive feedback
6:      for $k = 1, \dots, K$ do
7:          for all $e \in E \setminus S$ do
8:              $x_e \leftarrow \Delta(e \mid S)$
9:          end for
10:         $a_k^t \leftarrow \arg\max_{e \in E \setminus S} \left[ x_e^\top \bar{\theta}_{t-1} + \alpha \sqrt{x_e^\top M_{t-1}^{-1} x_e} \right]$
11:         $S \leftarrow S \cup \{a_k^t\}$
12:     end for
13:     Recommend list $A_t \leftarrow (a_1^t, \dots, a_K^t)$
14:     Observe click $C_t \in \{1, \dots, K, \infty\}$
15:     $M_t \leftarrow M_{t-1},\ B_t \leftarrow B_{t-1}$       ▷ Update statistics
16:     for $k = 1, \dots, \min\{C_t, K\}$ do
17:         $x_e \leftarrow \Delta\left(a_k^t \mid \{a_1^t, \dots, a_{k-1}^t\}\right)$
18:         $M_t \leftarrow M_t + \sigma^{-2} x_e x_e^\top$
19:         $B_t \leftarrow B_t + x_e \mathbb{1}\{C_t = k\}$
20:     end for
21: end for
```

$\sigma, \alpha > 0$, where σ controls the growth rate of the Gram matrix (line 18) and α controls the degree of optimism (line 10). At each time t, `CascadeLSB` has three stages.

In the first stage (line 4), we estimate θ^* as $\bar{\theta}_{t-1}$ by solving a least-squares problem. Specifically, we take M_{t-1} and B_{t-1}, which summarize all observed topic gains and responses up to time t respectively, and then estimate $\bar{\theta}_{t-1}$ that fits these responses the best.

In the second stage (lines 5–14), we recommend the best list of items under $\bar{\theta}_{t-1}$ and M_{t-1}. This list is generated by the greedy algorithm from Section 3.1, where the attraction probability of item e is overestimated as $x_e^\top \bar{\theta}_{t-1} + \alpha \sqrt{x_e^\top M_{t-1}^{-1} x_e}$ and x_e is defined in line 8. This optimistic overestimate is known as the *upper confidence bound (UCB)* Auer et al. (2002).

In the last stage (lines 15–20), we update the Gram matrix M_t by the outer product of the observed topic gains, and the response matrix B_t by the observed topic gains weighted by their clicks.

The per-step time complexity of `CascadeLSB` is $\mathcal{O}(KLd^2)$. In practice, we update M_t^{-1} instead of M_t, which takes $\mathcal{O}(d^2)$ time. The greedy maximization requires $O(KLd^2)$ time, because we select K items out of L based on their UCBs, each of which takes $\mathcal{O}(d^2)$ time. Lastly, $\mathcal{O}(Kd^2)$ time is required to update the statistics.

6 ANALYSIS

Let $\gamma = (1 - 1/\mathbf{e}) \max\left\{\frac{1}{K}, 1 - \frac{K-1}{2} c_{\max}\right\}$ and A^* be the optimal solution described in (6). Then based on Theorem 1, A_t (line 13 of Algorithm 1) is a γ-approximation at any time t. Similarly to Chen et al. (2016); Wen et al. (2017), we define the *γ-scaled n-step regret* as

$$R^\gamma(n) = \sum_{t=1}^{n} \mathbb{E}\left[f(A^*, \theta^*) - f(A_t, \theta^*)/\gamma\right], \qquad (9)$$

where the scaling factor $1/\gamma$ accounts for the fact that A_t is a γ-approximation at any time t. This is a natural performance metric in our setting, because even the offline variant of our problem in (6) cannot be solved optimally, efficiently. Therefore, it is unreasonable to assume that an online algorithm could compete with A^*. Under the scaled regret, `CascadeLSB` competes with comparable computationally-efficient offline approximations. This is summarized in our second main result below.

Theorem 2 *Under CDCM, for any $\sigma > 0$ and any*

$$\alpha \geq \frac{1}{\sigma}\sqrt{d \log\left(1 + \frac{nK}{d\sigma^2}\right) + 2\log(n) + \|\theta^*\|_2} \qquad (10)$$

in Algorithm 1, where $\|\theta^*\|_2 \leq \|\theta^*\|_1 \leq 1$, we have

$$R^\gamma(n) \leq \frac{2\alpha K}{\gamma}\sqrt{\frac{dn \log\left[1 + \frac{nK}{d\sigma^2}\right]}{\log\left(1 + \frac{1}{\sigma^2}\right)}} + 1. \qquad (11)$$

Theorem 2 states that for any $\sigma > 0$ and a sufficiently optimistic α for that σ, the regret bound in (11) holds. Specifically, if we choose $\sigma = 1$ and

$$\alpha = \sqrt{d \log\left(1 + \frac{nK}{d}\right) + 2\log(n) + \eta}$$

for some $\eta \geq \|\theta^*\|_2$, then $R^\gamma(n) = \tilde{O}\left(dK\sqrt{n}/\gamma\right)$, where the $\tilde{O}$ notation hides logarithmic factors. We now briefly discuss the tightness of this bound. The $\tilde{O}\left(\sqrt{n}\right)$-dependence on the time horizon n is considered near-optimal in gap-free regret bounds. The $\tilde{O}(d)$-dependence on the number of features d is standard in linear bandits (Abbasi-Yadkori et al., 2011). The $O(1/\gamma)$ factor is due to the fact that A_t is a γ-approximation. The $\tilde{O}(K)$-dependence on the number of recommended items K is because the agent recommends K items. We believe that this dependence can be reduced to $\tilde{O}(\sqrt{K})$ by a better analysis. We leave this for future work.

Finally, note that the list A_t in `CascadeLSB` is constructed greedily. However, our regret bound in Theorem 2 does not make any assumption on how the list is constructed. Therefore, the bound holds for any algorithm where A_t is a γ-approximation at any time t, for potentially different values of γ than in our paper.

7 RELATED WORK

The literature on offline learning to rank models which either incorporate diversity or position bias or both (Chapelle et al., 2011) is too wide to survey here. We recommend reading Liu et al. (2009); Aggarwal et al. (2016) for more information. We only mention online studies (Grotov and de Rijke, 2016) to which our paper is closely related i.e. online learning to rank in the cascade model and with diversity.

Cascading bandits (Kveton et al., 2015a; Combes et al., 2015) are an online learning to rank framework in the cascade model. Kveton et al. (2015a) proposed a near-optimal algorithm for this problem, `CascadeKL-UCB`, that learns the attraction probability of each item independently. This algorithm is expected to perform poorly when the number of items is large. Also, it does not model diversity. Several studies (Kveton et al., 2015b; Katariya et al., 2016; Zong et al., 2016; Li et al., 2016) have extended cascading bandits. The most related to our work are linear cascading bandits of Zong et al. (2016), who proposed `CascadeLinUCB` – an algorithm with a linear generalization across items (Wen et al., 2015; Abbasi-Yadkori et al., 2011). `CascadeLinUCB` assumes that the attraction probabilities of items are a linear function of the features of items, which are known; and an unknown parameter vector, which is learned. This work does not capture diversity. We compare to both `CascadeKL-UCB` and `CascadeLinUCB` in Section 8.

On the diversity front, *Ranked bandits* are a popular approach to online learning to rank (Radlinski et al., 2008; Slivkins et al., 2013). Here, the optimal list is diverse in the sense that each item in this list is clicked by many users that do not click on higher-ranked items. This notion of diversity is different from ours, because we learn a list of diverse items over topics for a single user. Learning algorithms for ranked bandits perform poorly in cascade-like models (Kveton et al., 2015a; Katariya et al., 2016; Magureanu et al., 2017) because they learn from clicks at all positions. We expect similar performance of other learning algorithms that make similar assumptions (Kohli et al., 2013). Raman et al. (2012) utilize preferences over pairs of diverse rankings only after assuming feedback on the entire recommended list. Thus, it does not capture position bias, and it is unclear how to incorporate the partial feedback model into their preference based setting. Yue and Guestrin (2011) studied the problem of online learning to rank with diversity, where each item covers a set of topics. They also proposed an efficient algorithm, `LSBGreedy`, for solving their problem. However, `LSBGreedy` assumes that if the item is not clicked, then the item is not attractive and penalizes the topics of this item. `LSBGreedy` differs from

CascadeLSB by assuming feedback at all positions. We compare to LSBGreedy in Section 8, and show that its regret can be linear when clicks on lower-ranked items are biased due to clicks on higher-ranked items.

8 EXPERIMENTS

This section is organized as follows. In Section 8.1, we validate the approximation ratio of the greedy algorithm from Section 3.1. In Section 8.2, we discuss our experimental choices. A synthetic experiment, which highlights the advantages of our method, is presented in Section 8.3. We describe our experimental setting for the real-world datasets in Section 8.4 and evaluate our algorithm on these datasets in the rest of the sections.

8.1 Approximation Ratio

In Section 3.1, we showed that a near-optimal list can be computed greedily. The approximation ratio of the greedy algorithm is close to $1 - 1/\mathrm{e}$ when the maximum click probability is small. Now, we demonstrate empirically that the approximation ratio is close to one in a domain of our interest.

We experiment with MovieLens 1M dataset from Section 8.5 (described later). The topic coverage and user preferences are set as in Section 8.2 and Section 8.4 (described later), respectively. We choose 100 random users and items and vary the number of recommended items K from 1 to 4. For each user and K, we compute the optimal list A^* in (6) by exhaustive search. Let the corresponding greedy list, which is computed as in (7), be A^{greedy}. Then $f(A^{\text{greedy}}, \theta^*)/f(A^*, \theta^*)$ is the approximation ratio under user preferences θ^*.

We found that for $K = 1, 2, 3$, and 4, the average approximation ratios over users was 1.000, 0.9926, 0.9997, and 0.9986, respectively. The average approximation ratio is always more than 0.99, which means that the greedy maximization in (7) is near optimal. We believe that this is due to the diminishing character of our objective (Section 3.1). The average approximation ratio is 1 when $K = 1$. This is expected since the optimal list of length 1 is the most attractive item under θ^*, which is always chosen in the first step of the greedy maximization in (7).

8.2 Experimental Choices

We compare CascadeLSB to CascadeKL-UCB (Kveton et al., 2015a), CascadeLinUCB (Zong et al., 2016), and LSBGreedy (Yue and Guestrin, 2011) in the experiments. To make CascadeLinUCB comparable to CascadeLSB, we set the features of item e as $x_e = \Delta(e \mid \emptyset)$. This guarantees that CascadeLinUCB operates in the same feature space as CascadeLSB; except that it does not model interactions due to higher ranked items, which lead to diversity. All compared algorithms are evaluated by the n-step regret $R^\gamma(n)$ with $\gamma = 1$, as defined in (9). We approximate the optimal solution A^* by the greedy algorithm in (7). The learning rate σ in CascadeLSB and the corresponding parameters in LSBGreedy and CascadeLinUCB are set to 0.1. The other parameter α in CascadeLSB is set to the lowest permissible value, according to (10). All remaining parameters in other algorithms are set as suggested by their theoretical analyses. CascadeKL-UCB does not have any tunable parameter.

The topic coverage in Section 2.2 can be defined in many ways. In this work, we adopt the probabilistic coverage function proposed in (El-Arini et al., 2009),

$$c_j(S) = 1 - \prod_{e \in S} (1 - \bar{w}(e, j)) \quad \forall j \in [d], \tag{12}$$

where $\bar{w}(e, j) \in [0, 1]$ is the *attractiveness* of item $e \in E$ in topic $j \in [d]$ and $c_j(S)$ is the j-th entry of $c(S)$. Under the assumption that items cover topics independently, the j-th entry of $c(S)$ is the probability that at least one item in S covers topic j. Clearly, the function in (12) is monotone and submodular in each entry of $c(S)$.

8.3 Synthetic Experiment

This experiment exhibits the need for modeling both diversity and position bias. We simulate a problem with $L = 53$ items, $d = 3$ topics, list size $K = 2$, and a single user whose preferences are $\theta^* = (0.6, 0.4, 0.0)$. The attractiveness of items 1 and 2 in topic 1 is 0.5, and 0 in all other topics. The attractiveness of item 3 in topic 2 is 0.5, and 0 in all other topics. The attractiveness of remaining 50 items in topic 3 is 1, and 0 in all other topics. These items are added to make the learning problem harder and to simulate a real-world scenario where most items are likely to be unattractive to any given user.

The optimal recommended list is $A^* = (1, 3)$. This example is constructed so that the optimal list contains only one item from the most preferred topic, either item 1 or 2. The n-step regret of all the algorithms is shown in Figure 1. We observe several trends.

First, the regret of CascadeLSB flattens and does not increase with the number of steps n. This means that CascadeLSB learns the optimal solution.

Second, the regret of LSBGreedy grows linearly with the number of steps n, which means it does not learn the optimal solution. This can be explained as follows. When LSBGreedy recommends $A^* = (1, 3)$, it severely underestimates the preference for topic 2 of item 3, because it assumes feedback at the second position even if the

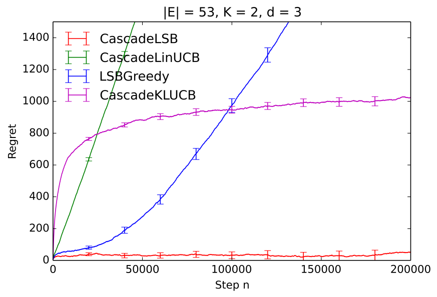

Figure 1: Regret on synthetic problem; lower values are better. Please see text (Section 8.3) for more insights.

first position is clicked. Because of this, `LSBGreedy` switches to recommending item 2 at the second position at some point in time. This is suboptimal. After some time, `LSBGreedy` switches back to recommending item 3, and then oscillates between items 2 and 3. Therefore, `LSBGreedy` has a linear regret and performs poorly.

Third, the regret of `CascadeLinUCB` is linear because it converges to list $(1, 2)$. The items in this list belong to a single topic, and therefore are redundant in the sense that a higher click probability can be achieved by recommending a more diverse list $A^* = (1, 3)$.

Finally, the regret of `CascadeKL-UCB` also flattens, which means that the algorithm learns the optimal solution. However, because `CascadeKL-UCB` does not generalize across items, it learns A^* with an order of magnitude higher regret than `CascadeLSB`.

8.4 Real-World Datasets

Now, we assess `CascadeLSB` on real-world datasets. One approach to evaluating bandit policies without a live experiment is *off-policy evaluation* (Li et al., 2011). Unfortunately, off-policy evaluation is unsuitable for our problem because the number of actions, all possible lists, is exponential in K. Thus, we evaluate our policies by building an interaction model of users from past data, an approach adopted by most papers discussed in Section 7.

All of our real-world problems are associated with a set of users U, a set of items E, and a set of topics $[d]$. The relations between the users and items are captured by feedback matrix $F \in \{0,1\}^{|U| \times |E|}$, where row u corresponds to user $u \in U$, column i corresponds to item $i \in E$, and $F(u, i)$ indicates if user u was attracted to item i in the past. The relations between items and topics are captured by matrix $G \in \{0,1\}^{|E| \times d}$, where row i corresponds to item $i \in E$, column j corresponds to topic $j \in [d]$, and $G(i, j)$ indicates if item i belongs to topic j. Next, we describe the interaction model.

The *attraction probability of item i in topic j* is defined as the number of users who are attracted to item i over all users who are attracted to at least one item in topic j:

$$\bar{w}(i, j) = \frac{\sum_{u \in U} F(u, i) G(i, j)}{\sum_{u \in U} \mathbb{1}\{\exists i' \in E : F(u, i') G(i', j) > 0\}}. \tag{13}$$

Therefore, the attraction probability represents a relative worth of item i in topic j. The *preference of a user u for topic j* is the number of items in topic j that attracted user u over the total number of topics of all items that attracted user u, i.e.,

$$\theta^*_j = \frac{\sum_{i \in E} F(u, i) G(i, j)}{\sum_{j' \in [d]} \sum_{i \in E} F(u, i) G(i, j')}. \tag{14}$$

Note that $\sum_{j=1}^{d} \theta^*_j = 1$. Therefore, $\theta^* = (\theta^*_1, \dots, \theta^*_d)$ is a probability distribution over topics for user u.

We divide users randomly into two halves to form training and test sets. This means that the feedback matrix F is divided into two matrices, F_{train} and F_{test}. The parameters that define our click model, which are computed from $\bar{w}(i, j)$ in (12) and θ^* in (14), are estimated from F_{test} and G. The topic coverage features in `CascadeLSB`, which are computed from $\bar{w}(i, j)$ in (12), are estimated from F_{train} and G. This split ensures that the learning algorithm does not have access to the optimal features for estimating user preferences, which is likely to happen in practice. In all experiments, our goal is to maximize the probability of recommending at least one attractive item. The experiments are conducted for $n = 20\text{k}$ steps and averaged over 100 random problem instances, each of which corresponds to a randomly chosen user.

8.5 Movie Recommendation

Our first real-world experiment is with MovieLens 1M dataset (Harper and Konstan, 2016). The dataset contains 1M ratings on a 5-star scale of 4k movies by 6k users. For simplicity, we extract $|E| = 1000$ most rated movies and $|U| = 1000$ most rating users. These active users and items are extracted just to have confident estimates of $\bar{w}(.)$ (13) and θ^* (14) for the experiments.

We treat movies and their genres as items and topics, respectively. We assume that a user u is attracted to a movie i if the user had rated that movie with 5 stars i.e. $F(u, i) = \mathbb{1}\{\text{user } u \text{ rated movie } i \text{ with 5 stars}\}$. Based on this definition, about 8% of user-item pairs in our dataset are attractive. We assume that a movie belongs to a genre if it is tagged with it. We experiment with $d = 5, 10$ and 18 (maximum genres in the data), and vary the number of recommended items K from 4 to 12. While varying topics, we choose the most popular ones.

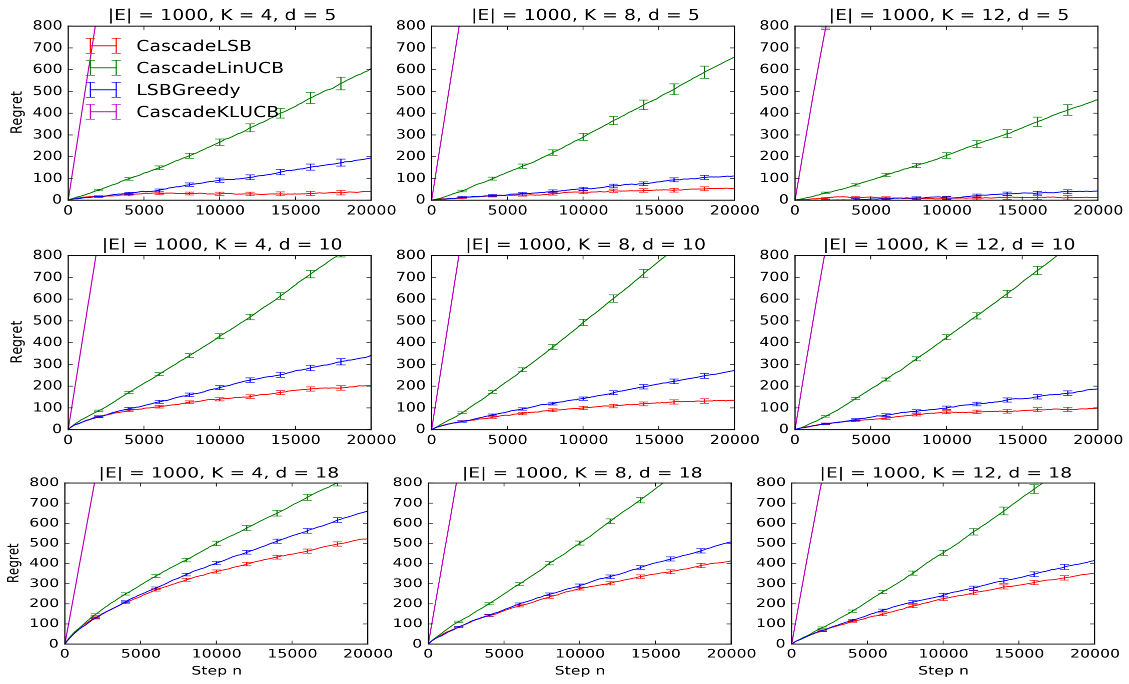

Figure 2: Regret on the MovieLens dataset; lower values are better and sublinear curve represents learning the optimal list. `CascadeLSB` is robust to both number of topics d (varies with rows) and size of the recommended list K (varies with columns).

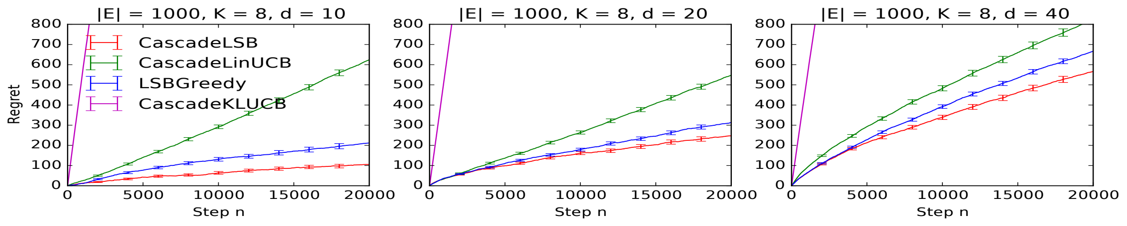

Figure 3: Regret on the Million Song for $K = 8$ and for $d = 10, 20, 40$; lower values and sublinear curve are better.

Our results are reported in Figure 2. We observe that `CascadeLSB` has the lowest regret among all compared algorithms for all d and K. This suggests that `CascadeLSB` is robust to the choice of both parameters d and K. For $d = 18$, `CascadeLSB` achieves almost 20% lower regret than the best performing baseline, `LSBGreedy`. `LSBGreedy` has a higher regret than `CascadeLSB` because it learns from unexamined items. `CascadeKL-UCB` performs the worst because it learns one attraction weight per item. This is impractical when the number of items is large, as in this experiment. The regret of `CascadeLinUCB` is linear, which means that it does not learn the optimal solution. This shows that linear generalization in the cascade model is not sufficient to capture diversity. Lastly, as expected, with an increase in the number of topics, `CascadeLSB`'s regret increases.

8.6 Million Song Recommendation

We next experiment with Million Song dataset (McFee et al., 2012). Again, we extract $|E| = 1000$ most popular songs and $|U| = 1000$ most active users, according to the number of song-listening events. We treat songs and their genres as items and topics, respectively. We assume that a user u is attracted to a song i, i.e. $F(u, i) = 1$, when user u had listened to the song i at least 5 times. By this definition, 3% of user-item pairs in the data are attractive. We assume that a song belongs to a genre if it is tagged with it. Here, we fix $K = 8$ and vary d from 10 to 40.

Our results are reported in Figure 3. Again, we observe that `CascadeLSB` has the lowest regret among all compared algorithms. This happens for all d showing that `CascadeLSB` is robust to the choice of d. At $d = 40$, `CascadeLSB` has about 15% lower regret than the best performing baseline, `LSBGreedy`. Again, `CascadeKL-UCB` performs the worst for all d.

8.7 Restaurant Recommendation

Lastly, we experiment with Yelp Challenge dataset (Huang et al., 2014). The dataset contains 4.1M re-

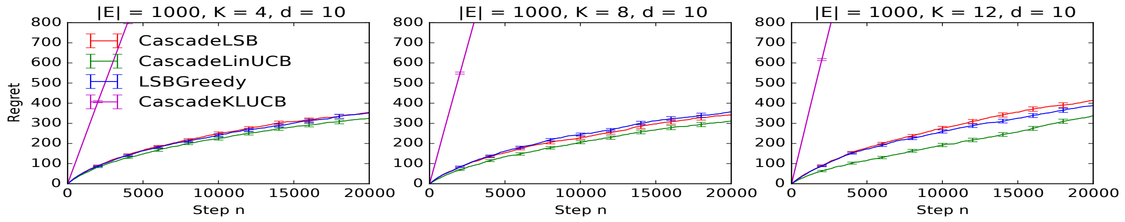

Figure 4: Regret on the Yelp dataset for $d = 10$ and for $K = 4, 8, 12$; lower values are better and sublinear curve represents learning the optimal list. All the algorithms except `CascadeKL-UCB` perform similar due to small attraction probabilities of items.

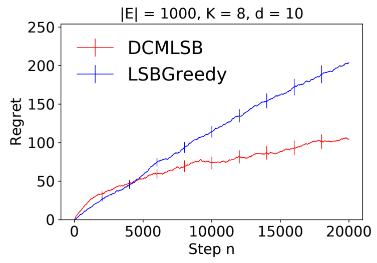

Figure 5: Regret on Movielens in the (special) DCM model.

views by 1M users for 48k restaurants in 600 categories. Similarly to the above experiments, we extract $|U| = 1000$ most reviewed restaurants and $|E| = 1000$ most reviewing users. We treat restaurants and their categories as items and topics, respectively. We assume that a user u is attracted to a restaurant i if the user had rated that restaurant with at least 4 stars, i.e., $F(u, i) = \mathbb{1}\{\text{user } u \text{ had rated restaurant } i \text{ with at least 4 stars}\}$. By this definition, about 3% of user-item pairs in our dataset are attractive. We assume that a restaurant belongs to a category if it is tagged with it. In this experiment, we fix $d = 10$ and vary K from 4 to 12.

Our results are reported in Figure 4. Unlike in the previous experiments, we observe that `CascadeLinUCB` performs comparably to `CascadeLSB` and `LSBGreedy`. We investigated this trend and discovered that this is because the attraction probabilities of items, as defined in (13), are often very small, such as on the order of 10^{-2}. This means that the items do not cover any topic properly. In this setting, the gain in topic coverage due to any item e over higher ranked items S, $\Delta(e \mid S)$, is comparable to $\Delta(e \mid \emptyset)$ when $|S|$ is small. This follows from the definition in (12). Now note that the former are the features in `CascadeLSB` and `LSBGreedy`, and the latter are the features in `CascadeLinUCB`. Since the features are similar, solutions from all the algorithms are similar.

9 DISCUSSION

We assume that the user clicks on at most one recommended item. We would like to stress that this assumption is only for simplicity of exposition. In particular, `CascadeLSB` can be easily extended to a multiple click scenario such as the *dependent click model (DCM)* (Guo et al., 2009; Katariya et al., 2016). As a glimpse of our future work, we show an extension of `CascadeLSB` for a special case of DCM (see Appendix C for details).

In DCM, the user examines the list of items, from the first item to the last, and clicks on all attractive items until the user is "satisfied". It turns out that the probability with which the user finds at least one satisfactory item is:

$$f_{DCM}(A, \bar{v}, \bar{w}) = 1 - \textstyle\prod_{k=1}^{K}(1 - \bar{v}(k)\bar{w}(a_k)), \quad (15)$$

where $\bar{v}(k)$ is the *termination* probability for position k. Here, the attraction probability $\bar{w}(a_k)$ can be modeled as in (3). For simplicity, assume that $\bar{v}(k) = \zeta \in (0, 1)\, \forall\, k \in [K]$ i.e. the user terminates the search with probability ζ for all positions. In an online learning framework, clearly, the user is satisfied when both realizations $w_t(a^t_k)$ and $v_t(k)$ are 1. We propose algorithm `dcmLSB` (Algorithm 2 in Appendix C.3) which tries to learn the user preference θ^*. `dcmLSB` is almost same as `CascadeLSB`, except that after each click it assumes that the user is satisfied (or terminates scanning the list) with probability ζ instead of probability one. In this way, `dcmLSB` takes feedback from multiple clicks. The γ_{DCM}-scaled regret (29) is computed analogously to (9) as when the user not just clicks but is satisfied. We compare `dcmLSB` with `LSBGreedy`, which assumes feedback on the entire list, on the Movielens dataset (Section 8.5) for $d = 10, K = 8$ and $\zeta = 0.8$. In Figure 5, we see that the regret of `dcmLSB` with $\gamma_{DCM} = 1$ is sublinear and much lower than `LSBGreedy`. This shows that the current diversity driven online learning to rank approaches are insufficient to handle cases with partial feedback albeit in the form of multiple clicks. More insights are discussed via a simulated study in Appendix C. In the future, we would like to extend `CascadeLSB` for the DCM and rigorously derive the value of γ_{DCM}.

10 CONCLUSIONS

In this paper, we model both position bias and diversified retrieval in a unified online learning to rank framework. We propose an efficient online algorithm to rank diverse items in the cascade model and derive a gap-free upper bound on its scaled n-step regret. The algorithm is shown to be competitive over a range of baselines.

References

Abbasi-Yadkori, Y., Pal, D., and Szepesvari, C. (2011). Improved algorithms for linear stochastic bandits. In *NIPS*, pages 2312–2320.

Aggarwal, C. C. et al. (2016). *Recommender systems*. Springer.

Agichtein, E., Brill, E., and Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *SIGIR*, pages 19–26.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256.

Carbonell, J. and Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336. ACM.

Chapelle, O., Ji, S., Liao, C., Velipasaoglu, E., Lai, L., and Wu, S.-L. (2011). Intent-based diversification of web search results: metrics and algorithms. *Information Retrieval*, 14(6):572–592.

Chen, W., Wang, Y., Yuan, Y., and Wang, Q. (2016). Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *JMLR*, 17(1):1746–1778.

Chuklin, A., Markov, I., and Rijke, M. d. (2015). Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1–115.

Combes, R., Magureanu, S., Proutiere, A., and Laroche, C. (2015). Learning to rank: Regret lower bounds and efficient algorithms. In *ACM SIGMETRICS*.

Craswell, N., Zoeter, O., Taylor, M., and Ramsey, B. (2008). An experimental comparison of click position-bias models. In *WSDM*, pages 87–94.

El-Arini, K., Veda, G., Shahaf, D., and Guestrin, C. (2009). Turning down the noise in the blogosphere. In *SIGKDD*, pages 289–298. ACM.

Grotov, A. and de Rijke, M. (2016). Online learning to rank for information retrieval: Sigir 2016 tutorial. In *SIGIR*, pages 1215–1218. ACM.

Guo, F., Liu, C., and Wang, Y. M. (2009). Efficient multiple-click models in web search. In *WSDM*, pages 124–131.

Harper, F. M. and Konstan, J. A. (2016). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19.

Huang, J., Rogers, S., and Joo, E. (2014). Improving restaurants by extracting subtopics from yelp reviews. *iConference 2014 (Social Media Expo)*.

Katariya, S., Kveton, B., Szepesvari, C., and Wen, Z. (2016). DCM bandits: Learning to rank with multiple clicks. In *ICML*.

Kohli, P., Salek, M., and Stoddard, G. (2013). A fast bandit algorithm for recommendations to users with heterogeneous tastes. In *AAAI*.

Kveton, B., Szepesvari, C., Wen, Z., and Ashkan, A. (2015a). Cascading bandits: Learning to rank in the cascade model. In *ICML*.

Kveton, B., Wen, Z., Ashkan, A., and Szepesvari, C. (2015b). Combinatorial cascading bandits. In *NIPS*, pages 1450–1458.

Li, L., Chu, W., Langford, J., and Wang, X. (2011). Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*.

Li, S., Wang, B., Zhang, S., and Chen, W. (2016). Contextual combinatorial cascading bandits. In *ICML*, pages 1245–1253.

Liu, T.-Y. et al. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

Magureanu, S., Proutiere, A., Isaksson, M., and Zhang, B. (2017). Online learning of optimally diverse rankings. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):32.

McFee, B., Bertin-Mahieux, T., Ellis, D. P., and Lanckriet, G. R. (2012). The million song dataset challenge. In *Proceedings of the 21st International Conference on World Wide Web*, pages 909–916. ACM.

Radlinski, F., Kleinberg, R., and Joachims, T. (2008). Learning diverse rankings with multi-armed bandits. In *ICML*, pages 784–791.

Raman, K., Shivaswamy, P., and Joachims, T. (2012). Online learning to diversify from implicit feedback. In *KDD*, pages 705–713. ACM.

Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35.

Slivkins, A., Radlinski, F., and Gollapudi, S. (2013). Ranked bandits in metric spaces: Learning diverse rankings over large document collections. *JMLR*, 14(1):399–436.

Streeter, M. and Golovin, D. (2009). An online algorithm for maximizing submodular functions. In *NIPS*, pages 1577–1584.

Streeter, M., Golovin, D., and Krause, A. (2009). Online learning of assignments. In *NIPS*, pages 1794–1802.

Wen, Z., Kveton, B., and Ashkan, A. (2015). Efficient learning in large-scale combinatorial semi-bandits. In *ICML*, pages 1113–1122.

Wen, Z., Kveton, B., Valko, M., and Vaswani, S. (2017). Online influence maximization under independent cascade model with semi-bandit feedback. In *NIPS*.

Yue, Y. and Guestrin, C. (2011). Linear submodular bandits and their application to diversified retrieval. In *NIPS*, pages 2483–2491.

Zong, S., Ni, H., Sung, K., Ke, N. R., Wen, Z., and Kveton, B. (2016). Cascading bandits for large-scale recommendation problems. In *UAI*.