BubbleRank: Safe Online Learning to Re-Rank via Implicit Click Feedback

Li, C.; Kveton, B.; Lattimore, T.; Markov, I.; de Rijke, M.; Szepesvári, C.; Zoghi, M.

Published in:
Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence

Link to publication

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (http://dare.uva.nl)
Abstract

In this paper, we study the problem of safe online learning to re-rank, where user feedback is used to improve the quality of displayed lists. Learning to rank has traditionally been studied in two settings. In the offline setting, rankers are typically learned from relevance labels created by judges. This approach has generally become standard in industrial applications of ranking, such as search. However, this approach lacks exploration and thus is limited by the information content of the offline training data. In the online setting, an algorithm can experiment with lists and learn from feedback on them in a sequential fashion. Bandit algorithms are well-suited for this setting but they tend to learn user preferences from scratch, which results in a high initial cost of exploration. This poses an additional challenge of safe exploration in ranked lists. We propose BubbleRank, a bandit algorithm for safe re-ranking that combines the strengths of both the offline and online settings. The algorithm starts with an initial base list and improves it online by gradually exchanging higher-ranked less attractive items for lower-ranked more attractive items. We prove an upper bound on the $n$-step regret of BubbleRank that degrades gracefully with the quality of the initial base list. Our theoretical findings are supported by extensive experiments on a large-scale real-world click dataset.

1 INTRODUCTION

Learning to rank (LTR) is an important problem in many application domains, such as information retrieval, ad placement, and recommender systems [23]. More generally, LTR arises in any situation where multiple items, such as web pages, are presented to users. It is particularly relevant when the diversity of users makes it hard to decide which item should be presented to a specific user [28, 37].

A traditional approach to LTR is offline learning of rankers from either relevance labels created by judges [26] or user interactions [13, 24]. Recent experimental results shows that such rankers, even in a highly-optimized search engine, can be improved by online LTR with exploration. Exploration is the key component in multi-armed bandit algorithms [3]. Many such algorithms have been proposed recently for online LTR in specific user-behavior models [15, 17, 19], the so-called click models [6]. Compared to earlier online LTR algorithms [28], these click model-based algorithms gain in statistical efficiency while giving up on generality. Empirical results indicate that click model-based algorithms are likely to be beneficial in practice.

Yet, existing algorithms for online LTR in click models are impractical for at least three reasons. First, an actual model of user behavior is typically unknown. This problem was initially addressed by Zoghi et al. [39]. They showed that the list of items in the descending order of relevance is optimal in several click models and proposed BatchRank for learning it. Then Lattimore et al. [20] built upon this work and proposed TopRank, which is the state-of-the-art online LTR algorithm. Second, these algorithms lack safety constraints and explore aggressively by placing potentially irrelevant items at high positions, which may significantly degrade user experience [34]. A third and related problem is that the algorithms are not well suited for so-called warm start scenarios [33], where the offline-trained production ranker already generates a good list, which only needs to be safely improved. Warm-starting an online LTR algorithm is challenging since existing posterior sampling algorithms, such as Thompson sampling [32], require item-level priors while only list-level priors are available practically.
We denote

We make the following contributions. First, motivated by

A with a high probability. Finally, we evaluate

we propose a bandit algorithm for online LTR that ad-

any item, and is

number of clicks is equal to the probability of clicking on

user

the remaining items. If item

is attractive

examines

position

is the examination probability of position

The user interacts with list

as follows. The user examines

position

with probability

and then

clicks on item

at that position with probability

Therefore, the expected number of clicks on list

is

CM and PBM are similar models, because the probability

of clicking factors into item and position dependent factors. Therefore, both in the CM and PBM, under the

assumption that

is parameterized by

, the expected number of clicks is maximized by listing the

most attractive items in descending order of their attraction. More precisely, the most clicked list is

when

. Therefore, perhaps not surprisingly, the problem of learning the optimal list in both models can be viewed as the same problem, a stochastic click bandit.

2 STOCHASTIC CLICK BANDIT

An instance of a stochastic click bandit is a tuple

, where

is the number of positions, in

is the number of items, in a distribution over binary attraction vectors

, and

is a distribution over binary examination matrices

.

The learning agent interacts with the stochastic click bandit as follows. At time

, it chooses a list

, which depends on its history up to time

, and then observes clicks

on all positions in

. A position is clicked if and only if it is examined and the item at that position is attractive. More specifically, for any

,

where

and

are stochastic and drawn i.i.d. from

.

The key assumption that allows learning in this model is that the attraction of any item is independent of the examination of its position. In particular, for any list
We propose an online algorithm that addresses the above problem and adaptively re-ranks a list of items generated by a production ranker with the goal of placing more attractive items at higher positions. We study a non-contextual variant of the problem, where we re-rank a list of items generated by “bubbling up” more attractive items. Therefore, we refer to it as BubbleRank. BubbleRank determines more attractive items by randomly exchanging neighboring items. If the lower-ranked item is found to be more attractive, the items are permanently exchanged and never randomly exchanged again. If the lower-ranked item is found to be less attractive, the items are never randomly exchanged again. We describe BubbleRank in detail below.

BubbleRank maintains a base list $\mathcal{R}_0$ at each time $t$. From the viewpoint of BubbleRank, this is the best list at time $t$. The list is initialized by the initial base list $\mathcal{R}_0$ (line 3). At time $t$, BubbleRank permutes $\mathcal{R}_t$ into a displayed list $\mathcal{R}_t$ (lines 4–10). Two kinds of permutations are employed. If $t$ is odd and so $h = 0$, the items at positions 1 and 2, 3 and 4, and so on, are randomly exchanged. If $t$ is even and so $h = 1$, the items at positions 2 and 3, 4 and 5, and so on are randomly exchanged. The items are exchanged only if BubbleRank is uncertain regarding which item is more attractive (line 9).

The list $\mathcal{R}_t$ is displayed and BubbleRank gets feedback from users. Therefore, they perform well on very frequent queries and poorly on infrequent queries. On moderately frequent queries, the so-called torso queries, their performance varies. As torso queries are sufficiently frequent, an online algorithm can be used to re-rank so as to optimize their value, such as the number of clicks. BubbleRank.

The list $\mathcal{R}_t$ is displayed and observe clicks $c_t \in \{0, 1\}^K$.

Algorithm 1: BubbleRank
\begin{algorithm}
\caption{BubbleRank}
\begin{algorithmic}[1]
\STATE \textbf{Input:} initial list $\mathcal{R}_0$ over $[K]$
\STATE $\forall i, j \in [K] : s_0(i, j) \leftarrow 0$, $n_0(i, j) \leftarrow 0$
\STATE $\mathcal{R}_1 \leftarrow \mathcal{R}_0$
\FOR {$t = 1, \ldots, n$}
\STATE $h \leftarrow t \mod 2$
\STATE $\mathcal{R}_t \leftarrow \mathcal{R}_t$
\FOR {$k = 1, \ldots, \lfloor (K - h)/2 \rfloor$}
\STATE $i \leftarrow \mathcal{R}_t(2k - 1 + h)$, $j \leftarrow \mathcal{R}_t(2k + h)$
\IF {$s_{t-1}(i, j) \leq 2\sqrt{n_{t-1}(i, j) \log(1/\delta)}$}
\STATE Randomly exchange items $\mathcal{R}_t(2k - 1 + h)$ and $\mathcal{R}_t(2k + h)$ in list $\mathcal{R}_t$
\ENDIF
\STATE Display list $\mathcal{R}_t$ and observe clicks $c_t \in \{0, 1\}^K$
\STATE $s_t \leftarrow s_{t-1}$, $n_t \leftarrow n_{t-1}$
\FOR {$k = 1, \ldots, \lfloor (K - h)/2 \rfloor$}
\STATE $i \leftarrow \mathcal{R}_t(2k - 1 + h)$, $j \leftarrow \mathcal{R}_t(2k + h)$
\IF {$|c_t(2k - 1 + h) - c_t(2k + h)| = 1$}
\STATE $s_t(i, j) \leftarrow s_t(i, j) + c_t(2k - 1 + h) - c_t(2k + h)$
\STATE $n_t(i, j) \leftarrow n_t(i, j) + 1$
\STATE $s_t(j, i) \leftarrow s_t(j, i) + c_t(2k + h) - c_t(2k - 1 + h)$
\STATE $n_t(i, j) \leftarrow n_t(i, j) + 1$
\ENDIF
\ENDIF
\STATE $\mathcal{R}_{t+1} \leftarrow \mathcal{R}_t$
\FOR {$k = 1, \ldots, K - 1$}
\STATE $i \leftarrow \mathcal{R}_{t+1}(k)$, $j \leftarrow \mathcal{R}_{t+1}(k + 1)$
\IF {$s_t(j, i) > 2\sqrt{n_t(j, i) \log(1/\delta)}$}
\STATE Exchange items $\mathcal{R}_{t+1}(k)$ and $\mathcal{R}_{t+1}(k + 1)$ in list $\mathcal{R}_{t+1}$
\ENDIF
\ENDFOR
\ENDFOR
\end{algorithmic}
\end{algorithm}
Assumption 1. For any lists $\mathcal{R}, \mathcal{R}' \in \Pi_k(\mathcal{D})$ and positions $k, \ell \in [K]$ such that $k < \ell$:

A1. $r(\mathcal{R}, \alpha, \chi) \leq r(\mathcal{R}^*, \alpha, \chi)$, where $\mathcal{R}^*$ is defined as in [1];

A2. $\{\mathcal{R}(1), \ldots, \mathcal{R}(k-1)\} = \{\mathcal{R}'(1), \ldots, \mathcal{R}'(k-1)\}$

$\implies \chi(\mathcal{R}, k) = \chi(\mathcal{R}', k)$;

A3. $\chi(\mathcal{R}, k) \geq \chi(\mathcal{R}, \ell)$;

A4. If $\mathcal{R}$ and $\mathcal{R}'$ differ only in that the items at positions $k$ and $\ell$ are exchanged, then $\alpha(\mathcal{R}(k)) \leq \alpha(\mathcal{R}(\ell)) \iff \chi(\mathcal{R}, k) \geq \chi(\mathcal{R}', k)$; and

A5. $\chi(\mathcal{R}, k) \geq \chi(\mathcal{R}^*, k)$.

The above assumptions hold in the CM. In the PBM, they hold when the examination probability decreases with the position.

Our assumptions can be interpreted as follows. Assumption [A1] says that the list of items in the descending order of attraction probabilities is optimal. Assumption [A2] says that the examination probability of any position depends only on the identities of higher-ranked items. Assumption [A3] says that a higher position is at least as examined as a lower position. Assumption [A4] says that a higher-ranked item is less attractive if and only if it increases the examination of a lower position. Assumption [A5] says that any position is examined the least in the optimal list.

To simplify our exposition, we assume that $\alpha(1) > \cdots > \alpha(K) > 0$. Let $\chi_{\text{max}} = \chi(\mathcal{R}^*, 1)$ denote the maximum examination probability, $\chi_{\text{min}} = \chi(\mathcal{R}^*, K)$ denote the minimum examination probability, and

$$\Delta_{\text{min}} = \min_{k \in [K-1]} \alpha(k) - \alpha(k+1)$$

be the minimum gap. Then the $n$-step regret of BubbleRank can be bounded as follows.

**Theorem 1.** In any stochastic click bandit that satisfies Assumption [A] and for any $\delta \in (0, 1)$, the expected $n$-step regret of BubbleRank is bounded as

$$R(n) \leq 180K\frac{\chi_{\text{max}}}{\chi_{\text{min}}} K - 1 + 2 |V_0| \frac{\log(1/\delta)}{\Delta_{\text{min}}} + \delta^{\frac{3}{2}} K^3 n^2.$$

### 4.2 SAFETY

Let

$$\mathcal{V}(\mathcal{R}) = \{(i, j) : i < j, \mathcal{R}^{-1}(i) > \mathcal{R}^{-1}(j)\}$$

be the set of incorrectly-ordered item pairs in list $\mathcal{R}$. Then our algorithm is safe in the following sense.

**Lemma 2.** Let

$$V_0 = \mathcal{V}(\mathcal{R}_0)$$

be the incorrectly-ordered item pairs in the initial base list $\mathcal{R}_0$. Then the number of incorrectly-ordered item pairs in any displayed list $\mathcal{R}_t$ is at most $|V_0| + K/2$, that is $|\mathcal{V}(\mathcal{R}_t)| \leq |V_0| + K/2$ holds uniformly over time with probability of at least $1 - \delta^{\frac{3}{2}} K^3 n^2$. 

---

1 Our assumptions are slightly weaker than those of Zoghi et al. [39]. For instance, Assumption [A2] is on the probability of examination. Zoghi et al. [39] make this assumption on the realization of examination.
We can extend (Theorem 1) is time horizon by using the so-called doubling trick in Section 5.3. Second, no base list \( \mathcal{R}_t \) contains more incorrectly-ordered item pairs than \( \mathcal{R}_0 \) with a high probability. In particular, under event \( \mathcal{E} \) in Lemma 9, any change in the base list (line 24 of BubbleRank) reduces the number of incorrectly-order item pairs by one. In Lemma 9, we prove that \( \mathbb{P}(\mathcal{E}) \geq 1 - \delta^2 K^2 n \).

### 4.3 DISCUSSION

Our upper bound on the \( n \)-step regret of BubbleRank (Theorem 1) is \( O(\Delta_{\min}^{-1} \log n) \) for \( \delta = n^{-1} \). This dependence is considered to be optimal in gap-dependent bounds. Our gap \( \Delta_{\min} \) is the minimum difference in the attraction probabilities of items, and reflects the hardness of sorting the items by their attraction probabilities. This sorting problem is equivalent to the problem of learning \( \mathcal{R}^* \). So, a gap like \( \Delta_{\min} \) is expected, and is the same as that in Zoghi et al. [39].

Our regret bound is notable because it reflects two key characteristics of BubbleRank. First, the bound is linear in the number of incorrectly-ordered item pairs in the initial base list \( \mathcal{R}_0 \). This suggests that BubbleRank should have lower regret when initialized with a better list of items. We validate this dependence empirically in Section 5. In many domains, such lists exist and are produced by existing ranking policies. They only need to be safely improved.

Second, the bound is \( O(\chi_{\max} \chi_{\min}^{-1}) \), where \( \chi_{\max} \) and \( \chi_{\min} \) are the maximum and minimum examination probabilities, respectively. In Section 5.3, we show that this dependence can be observed in problems where most attractive items are placed at infrequently examined positions. This limitation is intrinsic to BubbleRank, because attractive lower-ranked items cannot be placed at higher positions unless they are observed to be attractive at lower, potentially infrequently examined, positions.

The safety constraint of BubbleRank is stated in Lemma 2. For \( \delta = n^{-2} \), as discussed above, BubbleRank becomes a rather safe algorithm, and is unlikely to display any list with more than \( K/2 \) incorrectly-ordered item pairs than the initial base list \( \mathcal{R}_0 \). More precisely, \( |V(\mathcal{R}_1)| \leq |\mathcal{V}_0| + K/2 \) holds uniformly over time with probability of at least \( 1 - K^2/2n \). This safety feature of BubbleRank is confirmed by our experiments in Section 5.3.

The above discussion assumes that the time horizon \( n \) is known. However, in practice, this is not always possible. We can extend BubbleRank to the setting of an unknown time horizon by using the so-called doubling trick [4].

### 4.4 PROOF OF THEOREM 1

In Lemma 9, we establish that there exists a favorable event \( \mathcal{E} \) that holds with probability \( 1 - \delta^2 K^2 n \), when all beliefs \( s(t, i, j) \) are at most \( 2 \sqrt{\mathbb{E} t(i, j) \log(1/\delta)} \) from their respective means, uniformly for \( i < j \) and \( t \in [n] \). Since the maximum \( n \)-step regret is \( K \), we get that

\[
\bar{R}(n) \leq \mathbb{E} \left[ \hat{R}(n) \mathbb{I}\{\mathcal{E}\} \right] + \delta^2 K^3 n^2,
\]

where \( \hat{R}(n) = \sum_{t=1}^n r(\mathcal{R}^*, \alpha, \chi) - r(\mathcal{R}_t, \alpha, \chi) \). We bound \( \hat{R}(n) \) next. For this, let

\[
\mathcal{P}_t = \{(i, j) \in [K]^2 : i < j, |\mathcal{R}^{-1}_t(i) - \mathcal{R}^{-1}_t(j)| = 1, s_{t-1}(i, j) \leq 2 \sqrt{\mathbb{E} t-1(i, j) \log(1/\delta)}\}
\]

be the set of potentially randomized item pairs at time \( t \). Then, by Lemma 9 on event \( \mathcal{E} \), which bounds the regret of list \( \mathcal{R}_t \) with the difference in the attraction probabilities of items \( (i, j) \in \mathcal{P}_t \), we have that

\[
\hat{R}(n) \leq 3K \chi_{\max} \sum_{i=1}^K \sum_{j=i+1}^K \sum_{t=1}^n (\alpha(i) - \alpha(j)) \mathbb{I}\{(i, j) \in \mathcal{P}_t\}.
\]

Now note that for any randomized \( (i, j) \in \mathcal{P}_t \) at time \( t \),

\[
\chi_{\min}(\alpha(i) - \alpha(j)) \leq \mathbb{E}_{t-1} \left[ c_t(\mathcal{R}^{-1}_t(i)) - c_t(\mathcal{R}^{-1}_t(j)) \right] = \mathbb{E}_{t-1} \left[ s_t(i, j) - s_{t-1}(i, j) \right],
\]

where \( \mathbb{E}_{t-1} \left[ \cdot \right] \) is the expectation conditioned on the history up to time \( t \), \( \mathcal{R}_1, c_1, \ldots, \mathcal{R}_{t-1}, c_{t-1} \). The inequality is from \( \alpha(i) \geq \alpha(j) \), and Assumptions A2 and A3. The
We conduct four experiments to evaluate BubbleRank. The first inequality is by Lemma 7, which establishes that the quality of displayed lists, in Appendix B.

Due to space limitations, we report the Normalized Discounted Cumulative Gain (NDCG) of compared algorithms, which measures the quality of displayed lists, in Appendix C.

5.1 EXPERIMENTAL SETUP

We evaluate BubbleRank on the Yandex click dataset[3]. The dataset contains user search sessions from the log of the Yandex search engine. It is the largest publicly available dataset containing user clicks, with more than 30 million search sessions. Each session contains at least one search query together with 10 ranked items.

We preprocess the dataset as in [39]. In particular, we randomly select 100 frequent search queries, and then learn the parameters of three click models using the PyClick package: CM and PBM, described in Section 2.1 as well as the dependent click model (DCM) [10].

The DCM is an extension of the CM [9] where each position \( k \) is associated with an abandonment probability \( v(k) \). When the user clicks on an item at position \( k \), the user stops scanning the list with probability \( v(k) \). Therefore, the DCM can model multiple clicks. Following the work in [15], we incorporate abandonment into our definition of reward for DCM and define it as the number of abandonment clicks. The abandonment click is a click after which a user stops browsing the list, and each time step contains at most one abandonment click. So, the expected reward for DCM equals the probability of abandonment clicks, which is computed as follows:

\[
 r(\mathcal{R}, \alpha, \chi) = \sum_{k=1}^{K} \chi(\mathcal{R}, k) v(k) \alpha(\mathcal{R}(k)),
\]

\[
 \chi(\mathcal{R}, k) = \prod_{i=1}^{k} (1 - v(i)) \alpha(\mathcal{R}(i))
\]

is the examination probability of position \( k \) in list \( \mathcal{R} \). A high reward means a user stops the search session because of clicking on an item with high attraction probability.

The learned CM, DCM, and PBM are used to simulate user click feedback. We experiment with multiple click models to show the robustness of BubbleRank to multiple models of user feedback.

For each query, we choose 10 items. The number of positions is equal to the number of items, \( K = L = 10 \). The objective of our re-ranking problem is to place 5 most attractive items in the descending order of their attractiveness at the 5 highest positions, as in [39]. The performance of BubbleRank and our baselines is also measured only at the top 5 positions.

BubbleRank is compared to three baselines Cascade-KL-UCB [17], BatchRank [39], and TopRank [20]. The former is near optimal in the CM [17], but can have linear

---

حظي

https://academy.yandex.ru/events/data_analysis/relpred2011
https://github.com/markovi/PyClick
regret in other click models. Note that linear regret arises when CascadeKL-UCB erroneously converges to a suboptimal ranked list. BatchRank and TopRank can learn the optimal list $R^*$ in a wide range of click models, including the CM, DCM, and PBM. However, they can perform poorly in early stages of learning because they randomly shuffles displayed lists to average out the position bias. All experiments are run for 5 million steps, after which at least two algorithms converge to the optimal ranked list.

In the Yandex dataset, each query is associated with many different ranked lists, due to the presence of various personalization features of the production ranker. We take the most frequent ranked list for each query as the initial base list $R_0$ in BubbleRank, since we assume that the most frequent ranked list is what the production ranker would produce in the absence of any personalization. We also compare BubbleRank to a production baseline, called Baseline, where the initial list $R_0$ is applied for $n$ steps.

5.2 RESULTS WITH REGRET

In the first experiment, we compare BubbleRank to CascadeKL-UCB, BatchRank, and TopRank in the CM, DCM, and PBM of all 100 queries. Among them, TopRank is the state-of-the-art online LTR algorithm in multiple click models. We evaluate these algorithms by their cumulative regret, which is defined in $\mathbb{E}[\sum_{i=1}^{n} a_i - \max_{r \in R^*} \sum_{i=1}^{n} a_{r(i)}]$ at the top 5 positions. The regret, a measure of convergence, is a widely-used metric in the bandit literature [3, 15, 17, 39]. In the CM and PBM, the regret is the cumulative loss in clicks when a sequence of learned lists is compared to the optimal list in hindsight. In the DCM, the regret is the cumulative loss in abandonment clicks. We also report the regret of Baseline.

Our results are reported in Figure 1. We observe that the regret of Baseline grows linearly with time $n$, which means that it is not optimal on average. CascadeKL-UCB learns $R^*$ quickly in both the CM and DCM, but has linear regret in the PBM. This is expected since CascadeKL-UCB is designed for the CM, and the DCM is an extension of the CM. As for the PBM, which is beyond the modeling assumptions of CascadeKL-UCB, there is no guarantee on the performance of CascadeKL-UCB. BubbleRank, BatchRank, and TopRank can learn in all three click models. Compared to BatchRank and TopRank, BubbleRank has a higher regret in 5 million steps. However, in earlier steps, BubbleRank has a lower regret than BatchRank and TopRank, as it takes advan-
We report the safety constraint, while BubbleRank because it operates under the additional safety constraint in Lemma 2. The constraint is that BubbleRank is unlikely to display any list with more than $K/2$ incorrectly-ordered item pairs than the initial base list $R_0$. More precisely, $|V(R_t)| \leq |V(R_0)| + K/2$ holds uniformly over time with probability of at least $1 - K^2/n$ for $\delta = n^{-4}$ (Section 5.3), where $V$ is defined in (3). In the second experiment, we answer the question how often do BubbleRank, CascadeKL-UCB, BatchRank, and TopRank violate this constraint empirically. We define the safety constraint violation in $n$ steps as

$$V(n) = \sum_{t=1}^{n} \mathbb{1}\{|V(R_t)| > |V(R_0)| + K/2\},$$

where $R_t$ is the displayed list at time $t$.

We report the $n$-step safety constraint violation of CascadeKL-UCB, BatchRank, and TopRank in Figure 2. We do not include results of BubbleRank since BubbleRank never violates the constraint in our experiments. We observe that the safety constraint violations of CascadeKL-UCB in the first 100 steps are $24.12 \pm 0.76$, $23.33 \pm 0.90$, and $23.63 \pm 0.96$ in the CM, DCM, and PBM, respectively. Translating this to a search scenario, CascadeKL-UCB may show unsafe results, which are significantly worse than the initial base list $R_0$, and may hurt user experience, more than 20% of search sessions in the first 100 steps. Even worse, the violations of CascadeKL-UCB grow linearly with time in the PBM.

The safety issues of BatchRank, and TopRank are more severe than that of CascadeKL-UCB. More precisely, the violations of BatchRank in the first 100 steps are $83.01 \pm 0.56$, $71.89 \pm 0.92$, and $59.63 \pm 0.98$ in the CM, DCM, and PBM, respectively. And the violations of TopRank are $83.56 \pm 0.70$, $71.47 \pm 1.07$, and $57.00 \pm 1.24$ in the CM, DCM, and PBM, respectively. Note that the performance of TopRank is close to that of BatchRank in the first 100 steps since they both require the ranked lists to be randomly shuffled during the initial stages. Thus, BatchRank, and TopRank would frequently hurt the user experience during the early stages of learning.

To conclude, BubbleRank learns without violating its safety constraint, while CascadeKL-UCB, BatchRank, and TopRank violate the constraint frequently. Together with results in Section 5.2, BubbleRank is a safe algorithm but, to satisfy the safety constraint, it compromises the performance and learns slower than BatchRank and TopRank. In Appendix B, we compare BubbleRank to baselines in NDCG and show that BubbleRank converges to the optimal lists in hindsight.

5.4 SANITY CHECK ON THE REGRET BOUND

We prove an upper bound on the $n$-step regret of BubbleRank in Theorem 1. In comparison to the upper bounds of CascadeKL-UCB [17] and BatchRank [59], we have two new problem-specific constants: $|V_0|$ and $1/\chi_{\min}$. In this section, we show that these constants are intrinsic to the behavior of BubbleRank.

We first study how the number of incorrectly-ordered item pairs in the initial base list $R_0$, $|V_0|$, impacts the regret of BubbleRank. We choose 10 random initial base lists $V_0$ in each of our 100 queries and plot the regret of BubbleRank as a function of $|V_0|$. Our results are shown in Figure 3. We observe that the regret of BubbleRank is linear in $|V_0|$ in the CM, DCM, and PBM. This is the same dependence as in our regret bound (Theorem 1).

We then study the impact of the minimum examination probability $\chi_{\min}$ on the regret of BubbleRank. We experiment with a synthetic PBM with 10 items, which is parameterized by $\alpha = (0.9, 0.5, \ldots, 0.5)$ and $\chi = (0.9, \ldots, 0.9, 0.5, 0.5)$ for $i \geq 1$. The most attractive item is placed at the last position in $R_0$, $R_0 = (2, \ldots, K - 1, 1)$. Since this position is examined with probability 0.5, we expect the regret to double when $i$ increases by one. We experiment with $i \in [5]$ in Figure 3 and observe this trend in 1 million steps. This confirms that the dependence on $1/\chi_{\min}$ in Theorem 1 is generally unavoidable.

6 RELATED WORK

Online LTR via click feedback has been mainly studied in two approaches: under specific click models [7, 15, 17, 18, 19, 40]; or without a particular assumption on click models [20, 28, 30, 59]. Algorithms from the first group efficiently learn optimal rankings in the their considered click models but do not have guarantees beyond their specific click models. Algorithms from the second group, on the other hand, learn the optimal rankings in a broader class of click models. TopRank [20] is the state-of-the-art of the second group, which has the regret of $O(K^2 \log(n))$ in our re-ranking setup, that is $L = K$. BubbleRank also belongs to the second group and the regret of BubbleRank is comparable to that of TopRank given a good initial list, when $|V_0| = O(K)$. However,
unlike BubbleRank, TopRank and all the previous algorithms do not consider safety. They explore aggressively in the initial steps and may display irrelevant items at high positions, which may then hurt user experiences [34].

Our safety problem is related to the warm start problem [31]. Contextual bandits [11, 21, 28] deal with a broader class of models than we do and are used to address the warm start problem. But they are limited to small action sets, and thus unsuitable for the ranking setup that we consider in this paper.

The warm start LTR has been studied in multiple papers [11, 33, 56], where the goal is to use an online algorithm to fine tune the results generated by an offline-trained ranker. In these papers, different methods for learning prior distributions of Thompson sampling based online LTR algorithms from offline datasets have been proposed. However, these methods have the following drawbacks. First, the offline data may not well align with user preferences [38], which may result in a biased prior assumption. Second, grid search with online A/B tests may alleviate this and find a proper prior assumption [33], but the online A/B test requires additional costs in terms of user experience. Third, there is no safety constraint in these methods. Even with carefully picked priors, they may recommend irrelevant items to users, e.g., new items with little prior knowledge. In contrast, BubbleRank starts from the production ranked list and learns under the safety constraint. Thus, BubbleRank gets rid of these drawbacks.

Another related line of work are conservative bandits [16, 45]. In conservative bandits, the learned policy is safe in the sense that its expected cumulative reward is at least $1 - \alpha$ fraction of that of the baseline policy with high probability. This notion of safety is less stringent than that in our work (Section 4.2). In particular, our notion of safety is per-step, in the sense that any displayed list is only slightly worse than the initial base list with a high probability. We do not compare to conservative bandits in our experiments because existing algorithms for conservative bandits require the action space to be small. The actions in our problem are ranked lists, and their number is exponential in $K$.

7 CONCLUSIONS

In this paper, we fill a gap in the LTR literature by proposing BubbleRank, a re-ranking algorithm that gradually improves an initial base list, which we assume to be provided by an offline LTR approach. The improvements are learned from small perturbations of base lists, which are unlikely to degrade the user experience greatly. We prove a gap-dependent upper bound on the regret of BubbleRank and evaluate it on a large-scale click dataset from a commercial search engine.

We leave open several questions of interest. For instance, our paper studies BubbleRank in the setting of re-ranking. Although we explain an approach of extending BubbleRank to the general ranking setup in Section 4.3, we expect further experiments to validate this approach. Our general topic of interest are exploration schemes that are more conservative than those of existing online LTR methods. Existing methods are not very practical because they can explore highly irrelevant items at frequently examined positions.

ACKNOWLEDGMENTS

We thank our reviewers for helpful feedback and suggestions. This research was supported by Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), the Innovation Center for Artificial Intelligence (ICAI), and the Netherlands Organisatie 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


