Adapting to the Shifting Intent of Search Queries[*]

Umar Syed[†]
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
`usyed@cis.upenn.edu`

Aleksandrs Slivkins
Microsoft Research
Mountain View, CA 94043
`slivkins@microsoft.com`

Nina Mishra
Microsoft Research
Mountain View, CA 94043
`ninam@microsoft.com`

Abstract

Search engines today present results that are often oblivious to recent shifts in intent. For example, the meaning of the query 'independence day' shifts in early July to a US holiday and to a movie around the time of the box office release. While no studies exactly quantify the magnitude of intent-shifting traffic, studies suggest that news events, seasonal topics, pop culture, etc account for 1/2 the search queries. This paper shows that the signals a search engine receives can be used to both determine that a shift in intent happened, as well as find a result that is now more relevant. We present a meta-algorithm that marries a classifier with a bandit algorithm to achieve regret that depends logarithmically on the number of query impressions, under certain assumptions. We provide strong evidence that this regret is close to the best achievable. Finally, via a series of experiments, we demonstrate that our algorithm outperforms prior approaches, particularly as the amount of intent-shifting traffic increases.

1 Introduction

Search engines typically use a ranking function to order results. The function scores a document by the extent to which it matches the query, and documents are ordered according to this score. This function is fixed in the sense that it does not change from one query to another and also does not change over time. For queries such as 'michael jackson' traditional ranking functions that value features such as high page rank will not work since documents new to the web will not have accrued sufficient inlinks. Thus, a search engine's ranking function should not be fixed; different results should surface depending on the temporal context.

Intuitively, a query is "intent-shifting" if the most desired search result(s) change over time. More concretely, a query's intent has shifted if the click distribution over search results at some time differs from the click distribution at a later time. For the query 'tomato' on the heels of a tomato salmonella outbreak, the probability a user clicks on a news story describing the outbreak increases while the probability a user clicks on the Wikipedia entry for tomatoes rapidly decreases. There are studies that suggest that queries likely to be intent-shifting — such as pop culture, news events, trends, and seasonal topics queries — constitute roughly half of the search queries that a search engine receives [10].

The goal of this paper is to devise an algorithm that quickly adapts search results to shifts in user intent. Ideally, for every query and every point in time, we would like to display the search result that users are most likely to click. Since traditional ranking features like PageRank [4] change slowly over time, and may be misleading if user intent has shifted very recently, we want to use just the observed click behavior of users to decide which search results to display.

[*]Full version of this paper [20] is available on `arxiv.org`. In the present version, all proofs are omitted.

[†]This work was done while the author was an intern at Microsoft Research and a student in the Department of Computer Science, Princeton University.

There are many signals a search engine can use to detect when the intent of a query shifts. Query features such as as volume, abandonment rate, reformulation rate, occurrence in news articles, and the age of matching documents can all be used to build a classifier which, given a query, determines whether the intent has shifted. We refer to these features as the *context*, and an occassion when a shift in intent occurs as an *event*.

One major challenge in building an event classifier is obtaining training data. For most query and date combinations (e.g. 'tomato, 06/09/2008'), it will be difficult even for a human labeler to recall in hindsight whether an event related to the query occurred on that date. In this paper, we propose a novel solution that learns from unlabeled contexts and user click activity.

Contributions. We describe a new algorithm that leverages the information contained in contexts. Our algorithm is really a meta-algorithm that combines a bandit algorithm designed for the event-free setting with an online classification algorithm. The classifier uses the contexts to predict when events occur, and the bandit algorithm "starts over" on positive predictions. The bandit algorithm provides feedback to the classifier by checking, soon after each of the classifier's positive predictions, whether the optimal search result actually changed. The key technical hurdle in proving a regret bound is handling events that happen during the "checking" phase.

For suitable choices of the bandit and classifier subroutines, the regret incurred by our meta-algorithm is (under certain mild assumptions) at most $O(k + d_{\mathcal{F}})(\frac{n}{\Delta} \log T)$, where k is the number of events, $d_{\mathcal{F}}$ is a certain measure of the complexity of the concept class $\mathcal{F}$ used by the classifier, n is the number of possible search results, Δ is the "minimum suboptimality" of any search result (defined formally in Section 2), and T is the total number of impressions. This regret bound has a very weak dependence on T, which is highly desirable for search engines that receive much traffic.

The context turns out to be crucial for achieving logarithmic dependence on T. Indeed, we show that any bandit algorithm that ignores context suffers regret $\Omega(\sqrt{T})$, even when there is only one event. Unlike many lower bounds for bandit problems, our lower bound holds even when Δ is a constant independent of T. We also show that assuming a logarithmic dependence on T, the dependence on k and $d_{\mathcal{F}}$ is essentially optimal.

For empirical evaluation, we ideally need access to the traffic of a real search engine so that search results can be adapted based on real-time click activity. Since we did not have access to live traffic, we instead conduct a series of synthetic experiments. The experiments show that if there are no events then the well-studied UCB1 algorithm [2] performs the best. However, when many different queries experience events, the performance of our algorithm significantly outperforms prior techniques.

2 Problem Formulation and Preliminaries

We view the problem of deciding which search results to display in response to user click behavior as a *bandit problem*, a well-known type of sequential decision problem. For a given query q, the task is to determine, at each round $t \in \{1, \ldots, T\}$ that q is issued by a user to our search engine, a single result $i_t \in \{1, \ldots, n\}$ to display.[1] This result is clicked by the user with probability $p_t(i_t)$. A bandit algorithm $\mathcal{A}$ chooses i_t using only observed information from previous rounds, i.e., all previously displayed results and received clicks. The performance of an algorithm A is measured by its *regret*: $R_{\mathcal{A}}(T) \triangleq E\left[\sum_{t=1}^{T} p_t(i_t^*) - p_t(i_t)\right]$, where an *optimal* result $i_t^* \in \arg\max_i p_t(i)$ is one with maximum click probability, and the expectation is taken over the randomness in the clicks and the internal randomization of the algorithm. Note our unusually strong definition of regret: we are competing against the best result on *every* round.

We call an *event* any round t where $p_{t-1} \neq p_t$. It is reasonable to assume that the number of events $k \ll T$, since we believe that abrupt shifts in user intent are relatively rare. Most existing bandit algorithms make no attempt to predict when events will occur, and consequently suffer regret $\Omega(\sqrt{T})$. On the other hand, a typical search engine receives many signals that can be used to predict events, such as bursts in query reformulation, average age of retrieved document, etc.

[1]For simplicity, we focus on the task of returning a single result, and not a list of results. Techniques from [19] may be adopted to find a good list of results.

We assume that our bandit algorithm receives a *context* $x_t \in \mathcal{X}$ at each round t, and that there exists a function $f \in \mathcal{F}$, in some known *concept class* $\mathcal{F}$, such that $f(x_t) = +1$ if an event occurs at round t, and $f(x_t) = -1$ otherwise.[2] In other words, f is an *event oracle*. At each round t, an *eventful bandit algorithm* must choose a result i_t using only observed information from previous rounds, i.e., all previously displayed results and received clicks, plus all contexts up to round t.

In order to develop an efficient eventful bandit algorithm, we make an additional key assumption: At least one optimal result before an event is *significantly* suboptimal after the event. More precisely, we assume there exists a *minimum shift* $\epsilon_S > 0$ such that, whenever an event occurs at round t, we have $p_t(i^*_{t-1}) < p_t(i^*_t) - \epsilon_S$ for at least one previously optimal search result i^*_{t-1}. For our problem setting, this assumption is relatively mild: the events we are interested in tend to have a rather dramatic effect on the optimal search results. Moreover, our bounds are parameterized by $\Delta = \min_t \min_{i \neq i^*_t} p_t(i^*_t) - p_t(i)$, the *minimum suboptimality* of any suboptimal result.

3 Related Work

While there has been a substantial amount of work on ranking algorithms [11, 5, 13, 8, 6], all of these results assume that there is a fixed ranking function to learn, not one that shifts over time. Online bandit algorithms (see [7] for background) have been considered in the context of ranking. For instance, Radlinski et al [19] showed how to compose several instantiations of a bandit algorithm to produce a ranked list of search results. Pandey et al [18] showed that bandit algorithms can be effective in serving advertisements to search engine users. These approaches also assume a stationary inference problem.

Even though existing bandit work does not address our problem, there are two key algorithms that we do use in our work. The UCB1 algorithm [2] assumes fixed click probabilities and has regret at most $O(\frac{n}{\Delta}\log T)$. The EXP3.S algorithm [3] assumes that click probabilities can change on every round and has regret at most $O(k\sqrt{nT\log(nT)})$ for arbitrary p_t's. Note that the dependence of EXP3.S on T is substantially stronger.

The "contextual bandits" problem setting [21, 17, 12, 16, 14] is similar to ours. A key difference is that the context received in each round is assumed to contain information about the *identity* of an optimal result i^*_t, a considerably stronger assumption than we make. Our context includes only side information such as volume of the query, but we never actually receive information about the identity of the optimal result.

A different approach is to build a statistical model of user click behavior. This approach has been applied to the problem of serving news articles on the web. Diaz [9] used a regularized logistic model to determine when to surface news results for a query. Agarwal et al [1] used several models, including a dynamic linear growth curve model.

There has also been work on detecting bursts in data streams. For example, Kleinberg [15] describes a state-based model for inferring stages of burstiness. The goal of our work is not to detect bursts, but rather to predict shifts in intent.

In a recent concurrent and independent work, Yu et al [22] studied bandit problems with "piecewise-stationary" distributions, a notion that closely resembles our definition of events. However, they make different assumptions than we do about the information a bandit algorithm can observe. Expressed in the language of our problem setting, they assume that from time-to-time a bandit algorithm receives information about how users *would have* responded to search results that are never actually displayed. For us, this assumption is clearly inappropriate.

4 Bandit with Classifier

Our algorithm is called BWC, or "Bandit with Classifier". The high-level idea is to use a bandit algorithm such as UCB1, restart it every time the classifier predicts an event, and use subsequent rounds to generate feedback for the classifier. We will present our algorithm in a modular way, as a meta-algorithm which uses the following two components: `classifier` and `bandit`. In

[2]In some of our analysis, we require contexts be restricted to a strict (concept-specific) subset of $\mathcal{X}$; the value of f outside this subset will technically be `null`. See Section 5 for more details.

each round, `classifier` inputs a context x_t and outputs a "positive" or "negative" prediction of whether an event has happened in this round. Also, it may input labeled samples of the form (x, l), where x is a context and l is a boolean label, which it uses for training. Algorithm `bandit` is a bandit algorithm that is tuned for the event-free runs and provides the following additional functionality: after each round t of execution, it outputs the *t-th round guess*: a pair (G^+, G^-), where G^+ and G^- are subsets of arms that it estimates to be optimal and suboptimal, respectively.[3] Since both `classifier` and `bandit` make predictions (about events and arms, respectively), for clarity we use the term "guess" exclusively to refer to predictions made by `bandit`, and reserve the term "prediction" for `classifier`.

The algorithm operates as follows. It runs in phases of two alternating types: odd phases are called "testing" phases, and even phases are called "adapting" phases. The first round of phase j is denoted t_j. In each phase we run a fresh instance of `bandit`. Each testing phase lasts for L rounds, where L is a parameter. Each adapting phase j ends as soon as `classifier` predicts "positive"; the round t when this happens is round t_{j+1}. Phase j is called *full* if it lasts at least L rounds. For a full phase j, let (G_j^+, G_j^-) be the L-th round guess in this phase. After each testing phase j, we generate a boolean prediction l of whether there was an event in the first round thereof. Specifically, letting i be the most recent full phase before j, we set $l_{t_j} = $ `false` if and only if $G_i^+ \cap G_j^- \neq \emptyset$. If l_{t_j} is `false`, the labeled sample (x_{t_j}, l_{t_j}) is fed back to the classifier. Note that `classifier` never receives `true`-labeled samples. Pseudocode for BWC is given in Algorithm 1.

Disregarding the interleaved testing phases for the moment, BWC restarts `bandit` whenever `classifier` predicts "positive", optimistically assuming that the prediction is correct. By our assumption that events cause some optimal arm to become significantly suboptimal (see Section 2), an incorrect prediction should result in $G_i^+ \cap G_j^- \neq \emptyset$, where i is a phase before the putative event, and j is a phase after it. However, to ensure that the estimates G_i and G_j are reliable, we require that phases i and j are full. And to ensure that the full phases closest to a putative event are not too far from it, we insert a full testing phase every other phase.

Algorithm 1 BWC Algorithm

```
 1: Given: Parameter $L$, a $(L, \epsilon_S)$-testable bandit, and a safe classifier.
 2: for phase $j = 1, 2, \ldots$ do
 3:   Initialize bandit. Let $t_j$ be current round.
 4:   if $j$ is odd then
 5:     for round $t = t_j \ \ldots \ t_j + L$ do
 6:       Select arm $i_t$ according to bandit.
 7:       Observe $p_t(i_t)$ and update bandit.
 8:     Let $i$ be the most recent full phase before $j$.
 9:     If $G_i^+ \cap G_j^- \neq \emptyset$ let $l_{t_j} =$ false and pass training example $(x_{t_j}, l_{t_j})$ to classifier.
10:   else
11:     for round $t = t_j, \ t_j + 1, \ \ldots$ do
12:       Select arm $i_t$ according to bandit.
13:       Observe $p_t(i_t)$ and update bandit; pass context $x_t$ to classifier.
14:       if classifier predicts "positive" then
15:         Terminate inner for loop.
```

Let S be the set of all contexts which correspond to an event. When the classifier receives a context x and predicts a "positive", this prediction is called a *true positive* if $x \in S$, and a *false positive* otherwise. Likewise, when the classifier predicts a "negative", the prediction is called a *true negative* if $x \notin S$, and a *false negative* otherwise. The sample (x, l) is *correctly labeled* if $l = (x \in S)$.

We make the following two assumptions. First, `classifier` is **safe** for a given concept class: if it inputs only correctly labeled samples, it never outputs a false negative. Second, `bandit` is (L, ϵ)-**testable**, in the following sense. Consider an event-free run of `bandit`, and let (G^+, G^-) be its L-th round guess. Then with probability at least $1 - T^{-2}$, each optimal arm lies in G^+ but not in G^-, and any arm that is at least ϵ-suboptimal lies in G^- but not in G^+. So an (L, ϵ)-testable

[3]Following established convention, we call the options available to a bandit algorithm "arms". In our setting, each arm corresponds to a search result.

bandit algorithm is one that, after L rounds, has a good guess of which arms are optimal and which are at least ϵ-suboptimal.

For correctness, we require `bandit` to be (L, ϵ_S)-testable, where ϵ_S is the minimum shift. The performance of `bandit` is quantified via its **event-free regret**, i.e. regret on the event-free runs. Likewise, for correctness we need `classifier` to be safe; we quantify its performance via the maximum possible number of false positives, in the precise sense defined below. We assume that the state of `classifier` is updated only if it receives a labeled sample, and consider a game in which in each round t, `classifier` receives a context $x_t \notin S$, outputs a (false) positive, and receives a (correctly) labeled sample $(x, \texttt{false})$. For a given context set $\mathcal{X}$ and a given concept class $\mathcal{F}$, let the **FP-complexity** of the classifier be the maximal possible number of rounds in such a game, where the maximum is taken over all event oracles $f \in \mathcal{F}$ and all possible sequences $\{x_t\}$. Put simply, the FP-complexity of `classifier` is the maximum number of consecutive false positives it can make when given correctly labeled examples.

We will discuss efficient implementations of a safe `classifier` and a (L, ϵ)-testable `bandit` in Sections 5 and Section 6, respectively. We present provable guarantees for BWC in a modular way, in terms of FP-complexity, event-free regret, and the number of events. The main technical difficulty in the analysis is that the correct operation of the components of BWC — `classifier` and `bandit` — is interdependent. In particular, one challenge is to handle events that occur during the first L rounds of a phase; these events may potentially "contaminate" the L-th round guesses and cause incorrect feedback to `classifier`.

Theorem 1. *Consider an instance of the eventful bandit problem with number of rounds T, n arms, k events and minimum shift ϵ_S. Consider algorithm* BWC *with parameter L and components* `classifier` *and* `bandit` *such that for this problem instance,* `classifier` *is safe, and* `bandit` *is (L, ϵ_S)-testable. If any two events are at least $2L$ rounds apart, then the regret of* BWC *is*

$$R(T) \leq (2k + d)\, R_0(T) + (k + d)\, R_0(L) + kL. \tag{1}$$

where d is the FP-complexity of the classifier and $R_0(\cdot)$ is the event-free regret of `bandit`.

Remarks. The proof is available in the full version [20]. In our implementations of `bandit`, $L = \Theta(\frac{n}{\epsilon_S} \log T)$ suffices. In the $+kL$ term in (1), the k can be replaced by the number of testing phases that contain both a false positive in round 1 of the phase and an actual event later in the phase; this number can potentially be much smaller than k.

5 Safe Classifier

We seek a classifier that is safe for a given concept class $\mathcal{F}$ and has low FP-complexity. We present a classifier whose FP-complexity is bounded in terms of the following property of $\mathcal{F}$:

Definition 1. *Define the* safe function $S_{\mathcal{F}} : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ *of $\mathcal{F}$ as follows: $x \in S_{\mathcal{F}}(N)$ if and only if there is no concept $f \in \mathcal{F}$ such that: $f(y) = -1$ for all $y \in N$ and $f(x) = +1$. The* diameter *of $\mathcal{F}$, denoted $d_{\mathcal{F}}$, is equal to the length of the longest sequence $x_1, \ldots, x_m \in \mathcal{X}$ such that $x_t \notin S_{\mathcal{F}}(\{x_1, \ldots, x_{t-1}\})$ for all $t = 1, \ldots, m$.*

So if N contains only true negatives, then $S_{\mathcal{F}}(N)$ contains only true negatives. This property suggests that $S_{\mathcal{F}}$ can be used to construct a safe classifier `SafeCl`, which operates as follows: It maintains a set of `false`-labeled examples N, initially empty. When input an unlabeled context x, `SafeCl` outputs a positive prediction if and only if $x \notin S_{\mathcal{F}}(N)$. After making a positive prediction, `SafeCl` inputs a labeled example (x, l). If $l = \texttt{false}$, then x is added to N; otherwise x is discarded. Clearly, `SafeCl` is a safe classifer.

In the full version [20], we show that the FP-complexity of `SafeCl` is at most the diameter $d_{\mathcal{F}}$, which is to be expected: FP-complexity is a property of a classifier, and diameter is the completely analogous property for $S_{\mathcal{F}}$. Moreover, we give examples of common concept classes with efficiently computable safe functions. For example, if $\mathcal{F}$ is the space of hyperplanes with "margin" at least δ (probably the most commonly-used concept class in machine learning), then $S_{\mathcal{F}}(N)$ is the convex hull of the examples in N, extended in all directions by a δ.

By using `SafeCl` as our classifier, we introduce $d_{\mathcal{F}}$ into the regret bound of `bwc`, and this quantity can be large. However, in Section 7 we show that the regret of *any* algorithm must depend on $d_{\mathcal{F}}$, unless it depends strongly on the number of rounds T.

6 Testable Bandit Algorithms

In this section we will consider the stochastic n-armed bandit problem. We are looking for (L,ϵ)-testable algorithms with low regret. The L will need to be sufficiently large, on the order of $\Omega(n\epsilon^{-2})$.

A natural candidate would be algorithm UCB1 from [2] which does very well on regret. Unfortunately, it does not come with a guarantee of (L,ϵ)-testability. One simple fix is to choose at random between arms in the first L rounds, use these samples to form the best guess, in a straightforward way, and then run UCB1. However, in the first L rounds this algorithm incurs regret of $\Omega(L)$, which is very suboptimal. For instance, for UCB1 the regret would be $R(L) \leq O(\min(\frac{n}{\Delta}\log L,\ \sqrt{nL\log L}))$.

In this section, we develop an algorithm which has the same regret bound as UCB1, and is (L,ϵ)-testable. We state this result more generally, in terms of estimating expected payoffs; we believe it may be of independent interest. The (L,ϵ)-testability is then an easy corollary.

Since our analysis in this section is for the event-free setting, we can drop the subscript t from much of our notation. Let $p(u)$ denote the (time-invariant) expected payoff of arm u. Let $p^* = \max_u p(u)$, and let $\Delta(u) = p^* - p(u)$ be the "suboptimality" of arm u. For round t, let $\mu_t(u)$ be the sample average of arm u, and let $n_t(u)$ be the number of times arm u has been played.

We will use a slightly modified algorithm UCB1 from [2], with a significantly extended analysis. Recall that in each round t algorithm UCB1 chooses an arm u with the highest *index* $I_t(u) = \mu_t(u)+r_t(u)$, where $r_t(u) = \sqrt{8\log(t)/n_t(u)}$ is a term that we'll call the *confidence radius* whose meaning is that $|p(u)-\mu_t(u)| \leq r_t(u)$ with high probability. For our purposes here it is instructive to re-write the index as $I_t(u) = \mu_t(u) + \alpha\, r_t(u)$ for some parameter α. Also, to better bound the early failure probability we will re-define the confidence radius as $r_t(u) = \sqrt{8\log(t_0+t)/n_t(u)}$ for some parameter t_0. We will denote this parameterized version by UCB1(α, t_0). Essentially, the original analysis of UCB1 in [2] carries over; we omit the details.

Our contribution concerns estimating the $\Delta(u)$'s. We estimate the maximal expected reward p^* via the sample average of an arm that has been played most often. More precisely, in order to bound the failure probability we consider a arm that has been played most often *in the last $t/2$ rounds*. For a given round t let v_t be one such arm (ties broken arbitrarily), and let $\Delta_t(u) = \mu_t(v_t) - \mu_t(u)$ will be our estimate of $\Delta(u)$. We express the "quality" of this estimate as follows:

Theorem 2. *Consider the stochastic n-armed bandits problem. Suppose algorithm* UCB1$(6,t_0)$ *has been played for t steps, and $t+t_0 \geq 32$. Then with probability at least $1-(t_0+t)^{-2}$ for any arm u we have*

$$|\Delta(u) - \Delta_t(u)| < \tfrac{1}{4}\Delta(u) + \delta(t) \tag{2}$$

where $\delta(t) = O(\sqrt{\frac{n}{t}\log(t+t_0)})$.

Remark. Either we know that $\Delta(u)$ is small, or we can approximate it up to a constant factor. Specifically, if $\delta(t) < \frac{1}{2}\,\Delta_t(u)$ then $\Delta(u) \leq 2\,\Delta_t(u) \leq 5\,\Delta(u)$ else $\Delta(u) \leq 4\delta(t)$.

Let us convert UCB1$(6,T)$ into an (L,ϵ)-testable algorithm, as long as $L \geq \Omega(\frac{n}{\epsilon^2}\log T)$. The t-th round best guess $(G_t^+,\, G_t^-)$ is defined as $G_t^+ = \{u : \Delta_t(u) \leq \epsilon/4\}$ and $G_t^- = \{u : \Delta_t(u) > \epsilon/2\}$. Then the resulting algorithm is (L,ϵ)-testable assuming that $\delta(L) \leq \epsilon/4$, where $\delta(t)$ is from Theorem 2. The proof is in the full version [20].

7 Upper and Lower Bounds

Plugging the classifier from Section 5 and the bandit algorithm from Section 6 into the meta-algorithm from Section 4, we obtain the following numerical guarantee.

Theorem 3. *Consider an instance $\mathcal{S}$ of the eventful bandit problem with with number of rounds T, n arms and k events, minimum shift ϵ_S, minimum suboptimality Δ, and concept class diameter $d_{\mathcal{F}}$. Assume that any two events are at least $2L$ rounds apart, where $L = \Theta(\frac{n}{\epsilon_S^2}\log T)$. Consider the* BWC *algorithm with parameter L and components* `classifier` *and* `bandit` *as presented, respectively, in Section 5 and Section 6. Then the regret of* BWC *is $R(T) \leq \Big((3k+2d_{\mathcal{F}})\frac{n}{\Delta} + k\frac{n}{\epsilon_S^2}\Big)(\log T)$.*

While the linear dependence on n in this bound may seem large, note that without additional assumptions, regret must be linear in n, since each arm must be pulled at least once. In an actual search engine application, the arms can be restricted to, say, the top ten results that match the query.

We now state two lower bounds about eventful bandit problems; the proofs are in the full version [20]. Theorem 4 shows that in order to achieve regret that is logarithmic in the number of rounds, a context-aware algorithm is necessary, assuming there is at least one event. Incidentally, this lowerbound can be easily extended to prove that, in our model, *no* algorithm can achieve logarithmic regret when an event oracle f is not contained in the concept class $\mathcal{F}$.

Theorem 4. *Consider the eventful bandit problem with number of rounds T, two arms, minimum shift ϵ_S and minimum suboptimality Δ, where $\epsilon_S = \Delta = \epsilon$, for an arbitrary $\epsilon \in (0, \frac{1}{2})$. For any context-ignoring bandit algorithm $\mathcal{A}$, there exists a problem instance with a single event such that regret $R_{\mathcal{A}}(T) \geq \Omega(\epsilon\sqrt{T})$.*

Theorem 5 proves that in Theorem 3, linear dependence on $k + d_{\mathcal{F}}$ is essentially unavoidable. If we desire a regret bound that has logarithmic dependence on the number of rounds, then a linear dependence on $k + d_{\mathcal{F}}$ is necessary.

Theorem 5. *Consider the eventful bandit problem with number of rounds T and concept class diameter $d_{\mathcal{F}}$. Let $\mathcal{A}$ be an eventful bandit algorithm. Then there exists a problem instance with n arms, k events, minimum shift ϵ_S, minimum suboptimality Δ, where $\epsilon_S = \Delta = \epsilon$, for any given values of $k \geq 1$, $n \geq 3$, and $\epsilon \in (0, \frac{1}{4})$, such that $R_{\mathcal{A}}(T) \geq \Omega(k\,\frac{n}{\epsilon})\,\log(T/k)$.*

Moreover, there exists a problem instance with two arms, a single event, minimum shift $\Theta(1)$ and minimum suboptimality $\Theta(1)$ such that regret $R_{\mathcal{A}}(T) \geq \Omega(\max(T^{1/3},\, d_{\mathcal{F}}))\log T$.

8 Experiments

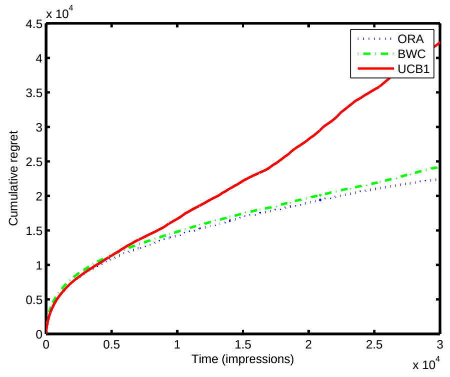
To truly demonstrate the benefits of BWC requires real-time manipulation of search results. Since we did not have the means to deploy a system that monitors click/skip activity and correspondingly alters search results with live users, we describe a collection of experiments on synthetically generated data.

We begin with a head-to-head comparison of BWC versus a baseline UCB1 algorithm and show that BWC's performance improves substantially upon UCB1. Next, we compare the performance of these algorithms as we vary the fraction of intent-shifting queries: as the fraction increases, BWC's performance improves even further upon prior approaches. Finally, we compare the performance as we vary the number of features. While our theoretical results suggest that regret grows with the number of features in the context space, in our experiments, we surprisingly find that BWC is robust to higher dimensional feature spaces.

Setup: We synthetically generate data as follows. We assume that there are 100 queries where the total number of times these queries are posed is 3M. Each query has five search results for a user to select from. If a query does not experience any events — i.e., it is not "intent-shifting" — then the optimal search result is fixed over time; otherwise the optimal search result may change. Only 10% of the queries are intent-shifting, with at most 10 events per such query. Due to the random nature with which data is generated, regret is reported as an average over 10 runs. The event oracle is an axis-parallel rectangle anchored at the origin, where points inside the box are negative and points outside the box are positive. Thus, if there are two features, say query volume and query abandonment rate, an event occurs if and only if both the volume and abandonment rate exceed certain thresholds.

Bandit with Classifier (BWC): Figure 1(a) shows the average cumulative regret over time of three algorithms. Our baseline comparison is UCB1 which assumes that the best search result is fixed throughout. In addition, we compare to an algorithm we call ORA, which uses the event oracle to reset UCB1 whenever an event occurs. We also compared to EXP3.S, but its performance was dramatically worse and thus we have not included it in the figure.

In the early stages of the experiment before any intent-shifting event has happened, UCB1 performs the best. BWC's safe classifier makes many mistakes in the beginning and consequently pays the price of believing that each query is experiencing an event when in fact it is not. As time progresses, BWC's classifier makes fewer mistakes, and consequently knows when to reset UCB1 more accu-



	0	1/8	1/4	3/8	1/2
ORA	17.2	22.8	30.4	33.8	39.5
BWC	17.8	24.6	39.9	46.7	99.4
UCB1	17.2	34.1	114.9	84.2	140.0
EXP3.S	78.4	123.7	180.2	197.6	243.1

	10	20	30	40
ORA	21.9	23.2	21.9	22.8
BWC	23.1	24.4	22.9	23.7
UCB1	32.3	33.5	31.1	37.4
EXP3.S	111.6	109.4	112.5	121.3

Figure 1: (a) (Left) BWC's cumulative regret compared to UCB1 and ORA (UCB1 with an oracle indicating the exact locations of the intent-shifting event) (b) (Right, Top Table) Final regret (in thousands) as the fraction of intent-shifting queries varies. With more intent-shifting queries, BWC's advantage over prior approaches improves. (c) (Right, Bottom Table) Final regret (in thousands) as the number of features grows.

rately. UCB1 alone ignores the context entirely and thus incurs substantially larger cumulative regret by the end.

Fraction of Intent-Shifting Queries: In the next experiment, we varied the fraction of intent-shifting queries. Figure 1(b) shows the result of changing the distribution from 0, 1/8, 1/4, 3/8 and 1/2 intent-shifting queries. If there are no intent-shifting queries, then UCB1's regret is the best. We expect this outcome since BWC's classifier, because it is safe, initially assumes that all queries are intent-shifting and thus needs time to learn that in fact no queries are intent-shifting. On the other hand, BWC's regret dominates the other approaches, especially as the fraction of intent-shifting queries grows. EXP3.S's performance is quite poor in this experiment – even when all queries are intent-shifting. The reason is that even when a query is intent-shifting, there are at most 10 intent-shifting events, i.e., each query's intent is not shifting all the time.

With more intent-shifting queries, the expectation is that regret monotonically increases. In general, this seems to be true in our experiment. There is however a decrease in regret going from 1/4 to 3/8 intent-shifting queries. We believe that this is due to the fact that each query has at most 10 intent-shifting events spread uniformly and it is possible that there were fewer events with potentially smaller shifts in intent in those runs. In other words, the standard deviation of the regret is large. Over the ten 3/8 intent-shifting runs for ORA, BWC, UCB1 and EXP3.S, the standard deviation was roughly 1K, 10K, 12K and 6K respectively.

Number of Features: Finally, we comment on the performance of our approach as the number of features grows. Our theoretical results suggest that BWC's performance should deteriorate as the number of features grows. Surprisingly, BWC's performance is consistently close to the Oracle's. In Figure 1(b), we show the cumulative regret after 3M impressions as the dimensionality of the context vector grows from 10 to 40 features. BWC's regret is consistently close to ORA as the number of features grows. On the other hand, UCB1's regret though competitive is worse than BWC, while EXP3.S's performance is across the board poor. Note that both UCB1 and EXP3.S's regret is completely independent of the number of features. The standard deviation of the regret over the 10 runs is substantially lower than the previous experiment. For example, over 10 features, the standard deviation was 355, 1K, 5K, 4K for ORA, BWC, UCB1 and EXP3.S, respectively.

9 Future Work

The main question left for future work is testing this approach in a realistic setting. Since gaining access to live search traffic is difficult, it would be interesting to find ways to use the search logs to simulate live traffic.

Acknowledgements. We thank Rakesh Agrawal, Alan Halverson, Krishnaram Kenthapadi, Robert Kleinberg, Robert Schapire and Yogi Sharma for their helpful comments and suggestions.

References

[1] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, Nitin Motgi, Seung-Taek Park, Raghu Ramakrishnan, Scott Roy, and Joe Zachariah. Online models for content optimization. In *22nd Advances in Neural Information Processing Systems (NIPS)*, 2008.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

[4] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *22nd Intl. Conf. on Machine Learning (ICML)*, 2005.

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *24th Intl. Conf. on Machine Learning (ICML)*, 2007.

[7] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[8] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. of Artificial Intelligence Research*, 10:243–270, 1999.

[9] Fernando Diaz. Integration of news content into web results. In *2nd Intl. Conf. on Web Search and Data Mining*, pages 182–191, 2009.

[10] D. Fallows. Search engine users. *Pew Internet and American Life Project*, 2005.

[11] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. of Machine Learning Research*, 4:933–969, 2003.

[12] Elad Hazan and Nimrod Megiddo. Online Learning with Prior Knowledge. In *20th Conference on Learning Theory (COLT)*, pages 499–513, 2007.

[13] Thorsten Joachims. Optimizing search engines using clickthrough data. In *8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2002.

[14] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *25th Intl. Conf. on Machine Learning (ICML)*, 2008.

[15] Jon M. Kleinberg. Bursty and hierarchical structure in streams. In *8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2002.

[16] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *21st Advances in Neural Information Processing Systems (NIPS)*, 2007.

[17] Sandeep Pandey, Deepak Agarwal, Deepayan Chakrabarti, and Vanja Josifovski. Bandits for Taxonomies: A Model-based Approach. In *SIAM Intl. Conf. on Data Mining (SDM)*, 2007.

[18] Sandeep Pandey, Deepayan Chakrabarti, and Deepak Agarwal. Multi-armed Bandit Problems with Dependent Arms. In *24th Intl. Conf. on Machine Learning (ICML)*, 2007.

[19] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *25th Intl. Conf. on Machine Learning (ICML)*, 2008.

[20] Umar Syed, Aleksandrs Slivkins, and Nina Mishra. Adapting to the shifting intent of search queries. Technical report. Available from arXiv.

[21] Chih-Chun Wang, Sanjeev R. Kulkarni, and H. Vincent Poor. Bandit problems with side observations. *IEEE Trans. on Automatic Control*, 50(3):338355, 2005.

[22] Jia Yuan Yu and Shie Mannor. Piecewise-stationary bandit problems with side observations. In *26th Intl. Conf. on Machine Learning (ICML)*, 2009.