Returning is Believing: Optimizing Long-term User Engagement in Recommender Systems

Qingyun Wu, Hongning Wang
Department of Computer Science
University of Virginia
Charlottesville, VA 22904, USA
{qw2ky,hw5x}@virignia.edu

Liangjie Hong*
Etsy Inc.
Brooklyn, NY 11201, USA
lhong@etsy.com

Yue Shi*†
Yahoo Research
Sunnyvale, CA 94089, USA
yueshi@acm.org

ABSTRACT

In this work, we propose to improve long-term user engagement in a recommender system from the perspective of sequential decision optimization. Users' click behavior and return behavior are directly modeled for online optimization. A bandit-based solution is formulated to balance three competing factors during online learning, including exploitation for immediate click, exploitation for expected future clicks, and exploration of unknowns for model estimation. We rigorously prove that with a high probability our proposed solution achieves a sublinear upper regret bound in maximizing cumulative clicks from a population of users in a given period of time, while a linear regret is inevitable if a user's temporal return behavior is not considered when making the recommendations. Extensive experimentation on both simulations and a large-scale real-world dataset collected from a practical recommender system verified the effectiveness and significant improvement of our proposed algorithm compared with several state-of-the-art online learning baselines for recommendation.

CCS CONCEPTS

•Information systems →Recommender systems; •Theory of computation →Regret bounds; •Computing methodologies →Sequential decision making;

KEYWORDS

User long-term engagement modeling; contextual bandit algorithm; content recommendation

1 INTRODUCTION

Recommender systems play a central role in today's Web ecosystems, as they prompt and facilitate users' interactions with system provided services. The success of a recommender system directly depends on the quality of user engagement [19, 24], which is considered as a desirable, even essential, human response to computer-mediated activities. User engagement can be measured not only from immediate user responses, such as click and dwell time on the recommended items, but more importantly from long-term responses, such as users' re-visitations and return time intervals. In fact, metrics purely based on immediate feedback signals such as user-item ratings [25] and click-through rate [9] have been increasingly criticized to be insufficient to measure and represent real engagement of users [28]. For example, some eye-catching recommendations may attract users to click; but after reviewing the recommended content, users might get unsatisfied or even upset. This will hurt users' trust in the system, cause them to return to it less often, and eventually leave. Therefore, we believe *user return*, which indicates users' long-term engagement, should be emphasized as an, at least equally, important metric of recommendation quality, and therefore to be optimized in a recommender system.

Unfortunately, most recommendation algorithms only focus on optimizing users' immediate responses (typically clicks) [17, 26]. Few of them explicitly take into account the temporal behavior of users after reviewing the recommended items, to pursue long-term utility of recommendations. Implicitly, such algorithms impose a strong assumption that users' return behavior is always consistent with their immediate responses, or the probability of user return is *independent* from the recommendations being made. However, these assumptions do not always hold in practice. For example, users may click on an item because of link bait, but may get disappointed and decide not to come back [28]. In addition, users may return less often because of boredom if the system keeps recommending popular items to the users [15]. There are some recent attempts in offline analysis and prediction of users' return time [15, 16], but little has been explored about how to integrate users' temporal return behavior with their click behavior to help improve overall recommendation quality.

To optimize users' long-term engagement, a good recommender system should maximize the total number of clicks from a population of users in a given period of time. If a recommendation drives a user to leave the system early when alternative exists, regret will accumulate *linearly* over such users and time. To reduce the expected regret of a recommendation decision, one has to not only predict its influence on a user's immediate click, but also to project it onto future clicks if the recommendation would attract the user to return. This makes the recommendation decisions *dependent* over time. Furthermore, as modern recommender systems face rapidly changing recommendation candidates and users, the influence of a recommendation on user click and return has to be estimated on

*The work is done when the authors were at Yahoo Research
†Yue Shi is now at Facebook.

CIKM'17, November 6–10, 2017, Singapore.

ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: http://dx.doi.org/10.1145/3132847.3133025

the fly. This inevitably introduces an explore-exploit dilemma in online learning [4, 5]; and the existence of multiple objectives for optimizing the long-term user engagement further complicates the problem. To the best of our knowledge, no single recommendation algorithm currently addresses all these challenges at once.

We formalize the optimization of users' long-term engagement as a sequential decision making problem, in which an agent maximizes the reward collected from a set of users in a given period of time by making recommendations. In every round of interaction, the agent faces the risk of losing a user because of making a "bad" recommendation, as the user's click and return depend on the recommendation, but the dependency is unknown to the agent apriori. The agent needs to maintain an estimate based on users' responses to the past recommendations on the fly. Due to the incomplete information, the agent needs to balance among three competing factors when making recommendations: 1) maximize the utility of user feedback for its model estimation (i.e., exploration); 2) maximize immediate reward of the recommendation (i.e., exploitation for short-term reward); and 3) maximize expected future reward by keeping users in the system (i.e., exploitation for long-term reward).

We solve this problem from the perspective of reinforcement learning [14]. Specifically, we consider user click as immediate reward to a recommendation; and the time interval between successive interactions, i.e., user's return time, determines how many rounds of interactions the agent could take in a given period of time. To maximize the cumulative reward over time, the agent has to make users click more and return more often. To avoid cumbersome transition policy estimation, which requires a large parameter and search space, we assume the distribution of recommendation candidates is stationary over time, and a user's click preference is stationary with respect to the recommendations. Therefore, the marginal probability of user click is also stationary. To improve the efficiency of policy iteration for making a recommendation, we only compute the expected future reward in a finite time. This leads us to a bandit learning solution [4, 5, 13]. To improve the algorithm's estimation quality, we use generalized linear models [23] with logit and inverse link functions to leverage contextual information for modeling discrete click and continuous return time. This choice of reward functions provides us a closed form assessment of model estimation confidence, which enables an efficient exploration strategy for our online model learning based on the Upper Confidence Bound principle [3].

We rigorously prove that with a high probability the proposed solution achieves a sublinear upper regret bound in optimizing long-term user engagement. We also demonstrate that if a system only optimizes immediate clicks on its recommendations, a linearly increasing regret can be inevitable. In addition, extensive experimentation on both simulations and a large-scale real-world dataset verified the improvement of the proposed algorithm compared with several online learning baselines for recommendation.

2 RELATED WORK

The importance of modeling users' post-click behaviors has been recognized and discussed in several recent works. Barbieri et al. used survival analysis to estimate users' post-click engagement on native ads by predicting the dwell time on the corresponding ad landing pages [6]. The predicted dwell time is then integrated into a ranking function to promote ads that are likely to lead to a longer dwell time. In [18], Lalmas et al. measured users' post-click experience by two metrics: dwell time and bounce rate. However, these works only focus on users' short-term engagement, such as click and dwell time, long-term user engagement is not studied.

There is also a line of related work about modeling users' return behavior [11, 15, 16], which is recently considered as an important measure of long-term user engagement. Kapoor et al. proposed a Cox's proportional hazard function based on survival analysis to predict users' return time on free web services [16]. In their follow-up work [15], a hidden semi-Markov model is used to model the time interval between a user's successive consumption activities with regard to his/her latent psychological states, including sensitization and boredom. Du et al. combined a self-exciting point process with low rank models to capture the recurrent temporal patterns in user-item consumption [11]. In their recent work, marked temporal point process and intensity functions are leveraged to predict the reoccurrence of user event [10]. In [8], Chapelle developed a survival analysis based approach to predict conversion delay in display advertising. However, all the aforementioned works focus on offline analysis and prediction of user return. Few of them touches the question of how the modeled user return can be integrated to improve a service system, especially when user return heavily depends on the quality of provided service.

Our work is also related to the studies in multi-armed bandits, which have recently become a reference solution to handle the notorious explore/exploit dilemma in online recommendation [4, 5, 7, 20, 27]. However, most of the bandit algorithms for recommendation only consider the immediate clicks or ratings as reward for optimization. In other words, such algorithms assume the users will always return to the system. As we discussed in the introduction, this assumption is often violated in practice, as user return depends on the quality of recommended items. To the best of our knowledge, there is no bandit algorithm that directly optimizes user long-term engagement under the setting that users might leave the system because of "bad" recommendations. In our solution, by explicitly modeling expected user return, the optimization objective is refined to include both the immediate click and expected future click from users, which reflects users' long-term engagement with the system. Our theoretical analysis proves that a linear regret is inevitable if user return depends on the recommendations but it is not considered in the optimization.

3 METHODOLOGY

We propose to improve long-term user engagement in a recommender system from the perspective of sequential decision optimization. In this problem, the agent's goal is to maximize the cumulative reward it receives from users in a given period of time by making recommendations. Because a "bad" recommendation might cause a user to return less often, or leave the system, the agent needs to balance the immediate reward of a recommendation (i.e., clicks) and expected reward from users' future interactions as a result of user return. Moreover, as the influence of a recommendation on a user's click and return decisions is unknown to the agent beforehand, the agent needs to maintain estimates of them on the fly. This further complicates the optimization problem. In the following discussions, we will first introduce several key concepts and our formulation of the problem, and then discuss our developed solution in details.

3.1 Problem Formulation

In a typical recommender system, a user's response to a recommended item can be characterized by the outcome of the following two variables,

- $C_{u,i} \in \{0, 1\}$ is a binary variable indicating whether user u clicks on the recommendation at the i-th interaction;
- $\Delta_{u,i} \in (0, \infty)$ is a continuous variable denoting the time interval between user u's current visit and next visit after examining the recommendation in the i-th interaction, i.e., user u's return time interval after the i-th interaction.

We materialize the immediate reward collected from a recommendation as user click $C_{u,i}$ in this work; but our solution can be readily extended to other types of user response, such as dwell time, ratings and conversion. $\Delta_{u,i}$ determines the total number of recommendations an agent can make in a given period of time, as it directly defines a user's revisitation sequence: $\{t_{u,i} = \sum_{i'<i} \Delta_{u,i'}\}_{i=1}^{\infty}$, where $t_{u,i}$ denotes the timestamp of the i-th interaction for user u. Since $\Delta_{u,i}$ depends on the recommended item at the i-th iteration, the optimal recommendation sequence becomes dependent over time: each time, an agent should not only consider the recommendations' influence on a user's immediate click, but also project it onto future clicks brought by this user's revisitation.

As a result, the optimization objective for this sequential decision making problem consists of two parts: the first part is the expected immediate click $P(C_{u,i} = 1|a_i)$, which reflects the user's short-term engagement on the recommended item a_i; and the second part is the expected future clicks in the resulting user revisitation sequence. To quantify the expected future clicks, we introduce two notations: $I(t)$ is a function to retrieve the recommendation candidate pool at time t, and π_A is the agent's decision policy. For the interaction at time $t_{u,i}$, the agent's decision is made by $a_i = \pi_A\Big(I(t_{u,i})\Big)$. The choice of a_i will lead to a future arrival sequence $\{t_{u,k}\}_{k=i+1}^{\infty}$ of user u with the return probability at time $t_{u,k}$ as $\prod_{j=i+1}^{k} \int_{t_{u,j-1}}^{t_{u,k}} P\Big(t_{u,j}|t_{u,j-1}, \pi_A(I(t_{u,j-1}))\Big)dt_{u,j}$. In particular, $P(t_{u,j}|t_{u,j-1}, a_{j-1})$ is the probability that user u returns at time $t_{u,j}$ given the recommended item a_{j-1} at time $t_{u,j-1}$. Then the expectation of future clicks resulted from the agent's i-th recommendation at time $t_{u,i}$ can be formally written as,

$$\sum_{k=i+1}^{\infty} P\Big(C_{u,k} = 1|\pi_A(I(t_{u,k}))\Big) \prod_{j=i+1}^{k} \int_{t_{u,j-1}}^{t_{u,k}} P\Big(t_{u,j}|t_{u,j-1}, \pi_A(I(t_{u,j-1}))\Big)dt_{u,j} \tag{1}$$

Eq (1) defines a value function, which measures the expected reward of the agent's policy π_A at time $t_{u,i}$. Policy iteration or value iteration is usually used to compute it analytically in standard reinforcement learning settings [14]. But in our problem, since new content constantly stream into a recommender system, it is infeasible to specify the recommendation candidate pool at a future time beforehand. In other words, $I(t)$ is undefined when time t is in the future; and this leads to $P(C_{u,j} = 1|a_j)$ and $P(t_{u,j+1}|t_{u,j}, a_j)$ at any future time $t_{u,j}$ undefined. This prohibits direct application of standard reinforcement learning solutions in this problem.

Several unique properties of the online recommendation problem and the value function defined in Eq (1) make it possible to approximate the expected future clicks. First, as $I(t)$ cannot be pre-specified in general, we assume the recommendation candidates are drawn from a stationary stochastic process and a user's interest on those candidates is also stationary, such that the click at a future time $t_{u,k}$ is only determined by the user. That is, we assume at current time $t_{u,i}$, the expected future click at time $t_{u,k}$ is $P\Big(C_{u,k} = 1|\pi_A(I(t_{u,k}))\Big) = \epsilon_u$ for $t_{u,k} > t_{u,i}$, where ϵ_u is the marginal click probability in user u. This assumption is mild, especially in product or movie recommendation scenarios [25], where the candidate pool is rather stable over time. Second, since a user's future arrival depends on all its preceding returns, the return probability decays exponentially fast over time. As a result, future clicks have a decreasing impact on the agent's current decision. Instead of performing a complete policy or value iteration, we can approximate it with only finite iterations. In this work, we only predict the future click one step ahead. Combining these two special treatments leads us to the following approximated optimization objective function at time $t_{u,i}$,

$$P(C_{u,i} = 1|a_i) + \epsilon_u \int_{t_{u,i}}^{\infty} P(t_{u,i+1}|t_{u,i}, a_i)dt_{u,i+1} \tag{2}$$

With stronger assumptions, such as the candidate pool is stable in a short period time, we can improve this approximation by iterating multiple steps ahead. But this will not change the nature of our solution; and to simplify our follow-up discussions, we will focus on the formulation defined in Eq (2) in this paper.

In Eq (2), a user return is allowed to happen in an infinite far future. This setting is practically less interesting, as we prefer users to return as early as possible. We introduce a return time threshold τ to restrict the computation of expected clicks in a foreseeable future, such that $t_{u,i+1} \leq t_{u,i} + \tau$. τ can be considered as the resolution of user return modeling, and it can be fine-tuned according to different recommender systems' business interests. For example, a news recommendation system might prefer its users to return every day. This finalizes our approximated sequential decision optimization objective as,

$$P(C_{u,i} = 1|a_i) + \epsilon_u P(\Delta_{u,i} \leq \tau|a_i) \tag{3}$$

According to Eq (3), at time $t_{u,i}$ the agent needs to choose the recommendation that maximizes the sum of expected immediate click and future click from user u. But as the click and return probabilities are unknown to the agent ahead of time, the estimate of them need to be maintained on the fly. This introduces another competing factor in the agent's decisions: it has to choose between maximizing its expected reward according to its current knowledge about Eq (3) (i.e., exploitation) and learning more about the unknowns for improving its estimate of Eq (3) (i.e., exploration). Conventional supervised learning would overly exploit historical data, which is biased towards previously trained models, and thus easily become victims of the Matthew effect (the rich get richer and the poor get poorer) in the long run. In this work, we develop a bandit-based solution to perform the online optimization of Eq (3).

3.2 A Bandit-based Online Solution

In a bandit setting of online optimization, a learning algorithm takes sequential actions to collect payoffs from the environment. In each round of interaction, the algorithm chooses an action from a finite set of candidates, while simultaneously adapts its decision strategy based on environment's feedback to maximize the total payoff in the long run [4, 5]. Map it to our long-term user engagement optimization problem: environment is the user that the agent serves, and the payoff of a recommendation in each round is defined by Eq (3). To derive a bandit-based solution for it, we need to quantify

the estimation of click and return probabilities in Eq (3), and design an action selection strategy for online feedback acquisition.

In a modern recommender system, recommendation candidates are usually described by a set of contextual features. And utilizing such contextual features for payoff estimation has been proved to be effective [12, 20]. In this work, we parameterize the estimation of click and return probabilities via generalized linear models [23]. In particular, we use a logit link function to model discrete clicks and an inverse link function to model continuous return time intervals. We should note that our formulation is general: when the contextual features are not available, the estimations just degenerate to estimating a Bernoulli distribution of clicks and an exponential distribution of return time intervals accordingly.

Under a logit link function, the click probability is quantified as $P(C_{u,i} = 1|a_i) = \frac{1}{1+\exp(-\boldsymbol{\theta}_u^\top \mathbf{x}_{a_i})}$, where $\mathbf{x}_{a_i}$ is a d-dimensional feature vector for recommendation candidate a_i with $\|\mathbf{x}_a\|_2 \leq L$, and $\boldsymbol{\theta}_u$ is the click model's coefficients pertaining to user u. With an inverse link function, we can define the return probability as $P(\Delta_{u,i} \leq t) = 1 - \exp(-\lambda_{u,a_i} t)$, where $\lambda_{u,a_i} = \exp(\boldsymbol{\beta}_u^\top \mathbf{x}_{a_i})$ and $\boldsymbol{\beta}_u$ is the return model's coefficients for user u. $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_u$ can be efficiently estimated via a maximum likelihood estimator (MLE) [23] on users' visitation sequence $\left\{(\mathbf{x}_{a_{u,i}}, C_{u,i}, \Delta_{u,i})\right\}_{i=1}^{N}$. Because of our stationary assumption about the candidate pool and user's click preference over time, user u's marginal click probability can be estimated by $\hat{\epsilon}_{u,i} = \sum_{j=1}^{i-1} C_{u,j}/(i-1)$. As the user's visitation sequence is affected by the recommendations, a well-defined action selection strategy is needed for the agent to quickly explore the unknowns for model estimation and exploit the currently learnt model to maximize Eq (3) on the fly.

Various types of action selection strategies have been studied in literature [3–5]. Among them, Upper Confidence Bound (UCB) [3, 20], which uses estimation confidence of predicted payoff on the candidate actions for exploration, has been proved to be effective. As we use generalized linear models to quantify the payoff in Eq (3), the estimation confidence of click and return probabilities can be readily evaluated. Formally, we denote $(\boldsymbol{\theta}_u^*, \boldsymbol{\beta}_u^*)$ as the ground-truth model parameters, and without loss of generality we assume that $\|\boldsymbol{\theta}_u^*\|_2 \leq M, \|\boldsymbol{\beta}_u^*\|_2 \leq H$. Denote $(\hat{\boldsymbol{\theta}}_{u,i}, \hat{\boldsymbol{\beta}}_{u,i})$ as their maximum likelihood estimations at round i. With probability at least $1 - \delta_1$, we have the following two inequalities hold on any particular recommendation candidate a_i for user u,

$$|P(C_{u,i} = 1|\mathbf{x}_{a_i}, \hat{\boldsymbol{\theta}}_{u,i}) - P(C_{u,i} = 1|\mathbf{x}_{a_i}, \boldsymbol{\theta}_u^*)| \leq \alpha^{\theta}_{u,i} \|\mathbf{x}_{a_i}\|_{\mathbf{A}^{-1}_{u,i}} \quad (4)$$

$$|P(\Delta_{u,i} \leq \tau|\mathbf{x}_{a_i}, \hat{\boldsymbol{\beta}}_{u,i}) - P(\Delta_{u,i} \leq \tau|\mathbf{x}_{a_i}, \boldsymbol{\beta}_u^*)| \leq \alpha^{\beta}_{u,i} \|\mathbf{x}_{a_i}\|_{\mathbf{A}^{-1}_{u,i}} \quad (5)$$

in which $A_{u,i} = \eta I + \sum_{j<i} \mathbf{x}_{a_{u,j}} \mathbf{x}_{a_{u,j}}^\top$, $\alpha^{\theta}_{u,i} = \sqrt{d \ln(\frac{\eta+iL}{\delta_1})} + \sqrt{\eta} M$, $\alpha^{\beta}_{u,i} = \sqrt{d \ln(\frac{\eta+iL\tau}{\delta_1})} + \sqrt{\eta} H$, and $\|\mathbf{x}\|_{\mathbf{A}^{-1}} = \sqrt{\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x}}$.

Eq (4) and (5) provide a tight upper bound for the estimation confidence of user click and return probabilities under maximum likelihood estimation. The proof of Eq (4) and (5) can be readily derived from the proof of the prediction error in generalized linear models [12]. Due to space limit, we omit the proof details. It is easy to verify that those confidence intervals shrink as more observations become available. Hence, less exploration is needed in the later stage of online optimization. Combining the estimated payoff defined in Eq (3) and the corresponding estimation confidence, a UCB-type action selection strategy can be formulated as,

$$\pi_A\left(I(t_{u,i})\right) = \underset{a \in I(t_{u,i})}{\arg\max} \Big(P(C_{u,i} = 1|\mathbf{x}_a, \hat{\boldsymbol{\theta}}_{u,i}) + \hat{\epsilon}_{u,i} P(\Delta_{u,i} \leq \tau|\mathbf{x}_a, \hat{\boldsymbol{\beta}}_{u,i}) + \alpha_{u,i} \|\mathbf{x}_a\|_{\mathbf{A}^{-1}_{u,i}} \Big) \quad (6)$$

where $\alpha_{u,i} = \alpha^{\theta}_{u,i} + \hat{\epsilon}_{u,i} \alpha^{\beta}_{u,i}$, and $\hat{\epsilon}_{u,i}$ is the estimation of ϵ_u at the i-th iteration.

The recommendation policy defined in Eq (6) can also be understood from the perspective of multi-objective optimization. The payoff function defined in Eq (3) can be decomposed into two parts: payoff from a user's immediate click (i.e., $P(C_{u,i} = 1|\mathbf{x}_a)$) and that from a user's expected future click (i.e., $\epsilon_u P(\Delta_{u,i} \leq \tau|\mathbf{x}_a)$). Correspondingly, Eq (4) and (5) provide the estimation confidence of each type of payoff. Our action selection strategy essentially combines two complementary bandit algorithms for online optimization of Eq (3). And therefore, we name our algorithm as reward-return bandit, or r^2Bandit in short.

Algorithm 1 r^2Bandit

Inputs: $\eta > 0$, $\tau > 0$, $\delta_1 \in (0, 1)$
for $i = 1$ to N **do**
  Receive user u
  Record current timestamp $t_{u,i}$
  if user u is new: **then**
    Set $\mathbf{A}_{u,1} \leftarrow \eta \mathbf{I}$, $\hat{\boldsymbol{\theta}}_{u,1} \leftarrow \mathbf{0}^d$, $\hat{\boldsymbol{\beta}}_{u,1} \leftarrow \mathbf{0}^d$, $\hat{\epsilon}_{u,1} \sim U(0, 1)$;
  else:
    Compute return interval $\Delta_{u,i-1} = t_{u,i} - t_{u,i-1}$
    Update $\hat{\boldsymbol{\beta}}_{u,i}$ in user return model using MLE.
  end if
  Observe context vectors, $\mathbf{x}_a \in \mathbb{R}^d$ for $\forall a \in I(t_{u,i})$
  Make recommendation $a_{u,i} = \arg\max_{a \in I(t_{u,i})} P(C_{u,i} = 1|\mathbf{x}_a, \hat{\boldsymbol{\theta}}_{u,i}) + \hat{\epsilon}_{u,i} P(\Delta_{u,i} \leq \tau|\mathbf{x}_a, \hat{\boldsymbol{\beta}}_{u,i}) + \alpha_{u,i} \|\mathbf{x}_a\|_{\mathbf{A}^{-1}_{u,i}}$
  Observe click $C_{u,i}$
  $\mathbf{A}_{u,i+1} \leftarrow \mathbf{A}_{u,i} + \mathbf{x}_{a_{u,i}} \mathbf{x}_{a_{u,i}}^\top$
  Update $\hat{\boldsymbol{\theta}}_{u,i+1}$ in user click model using MLE.
  Update $\hat{\epsilon}_{u,i+1} = \sum_{j \leq i} C_{u,j}/i$
end for

The details of r^2Bandit are described in Algorithm 1. The algorithm takes η, δ_1 and τ as inputs, and outputs a sequence of recommendations for each user in the system. To simplify our algorithm description, we use N to denote the total number of rounds the algorithm could take when interacting with users. We can terminate the algorithm when no users would return before a required time threshold, i.e., optimizing in a given period of time.

In Algorithm 1, we assume the model parameters associated with each user, i.e., $\{\boldsymbol{\theta}_u, \boldsymbol{\beta}_u\}$, are independent across users. This ensures the generality of r^2Bandit, but it also increases the total amount of parameters to be estimated. Considering the potentially sparse observations in an individual user, this might limit the practical performance of our algorithm. To reduce the parameter space, dependency among users can be explored, i.e., collaborative bandits [7, 27]. For example, by assuming similar users tend to share the same parameters, we can estimate the models in a group-wise manner. And the similarity between users can be defined by their demographics, content viewing history, or social connections. In

an extreme case, one can assume that all users share the same set of parameters [20]. Our r^2Bandit algorithm can be readily extended to such collaborative settings [7, 27]. We leave it as our future work and focus on our current more general setting in this paper.

The explicit modeling of user return differentiates r^2Bandit from existing bandit algorithms for online recommendation [12, 20]. Traditional bandit solutions only focus on maximizing immediate user clicks, with an implicit assumption that users will always return. However, since a user might return less often, or even leave the system, if "bad" recommendations have been presented, the total number of interactions that agent could take with the user become dependent on the algorithm's choices. An increased user return time leads to reduced cumulative clicks an algorithm can receive in a given period time. r^2Bandit balances the immediate click and user return via the expected future user clicks, and actively explores items that are most helpful for improving the quality of both clicks and return time estimations. Our theoretical regret analysis detailed in the next section supports that if user return is not considered, a linearly increasing regret is inevitable when items with high click probability cannot always lead to a shorter return time. Intutively, those algorithms would fail to recognize the recommendations with high extrinsic appeal but low intrinsic utility to users. r^2Bandit balances these two competing factors and achieves a sublinear regret with high probability in a finite time.

3.3 Regret Analysis

Regret of a multi-armed bandit algorithm is defined as the difference between the expected payoff from the optimal decisions made by the oracle and that from the algorithm's choices. In this section, we provide detailed regret analysis of our proposed solution for long-term user engagement optimization. To simplify the discussion, we omit subscript u in all our notations, as users are treated independently from each other in r^2Bandit.

To provide a finite time regret bound analysis, we discretize time into N time intervals and assume both the oracle and an algorithm can only make recommendations within each discrete time interval. As a result, at each time, the user either returns to give feedback on a recommendation, or does not return (no payoff can be collected then). We should note this operation does not restrict the practical value of our regret analysis. First, in real systems we cannot expect a user to return in a very short period of time, e.g., milliseconds. This naturally discretizes users' returns. Second, we only need to assume the existence of such a minimum return time interval, while the execution of our algorithm or oracle does not depend on the actual value of it. Third, as the oracle is also restricted to the same discrete time intervals, our analysis still leads to a tight regret bound. Denote S_r as the set of iterations in which the user returns as a result of the recommendations made by our algorithm, and the rest as S_n. Hence, we have $|S_r| + |S_n| = N$. Define N' as the cardinality of S_r, and N^* as the optimal number of times that the user would return according to the oracle strategy. The expected cumulative regret can thus be separated into two parts,

$$\begin{aligned}\mathbf{R}(N) &= \sum_{i\in S_n} R_i + \sum_{i\in S_r} R_i = \mathbb{E}[\epsilon(N^* - N')] + \sum_{i\in S_r} R_i \qquad (7)\\ &= \epsilon \sum_{i=1}^{N} \Big(P(\Delta_i \le \tau | \mathbf{x}_{a_i^*}, \boldsymbol{\beta}^*) - P(\Delta_i \le \tau | \mathbf{x}_{a_i}, \boldsymbol{\beta}^*)\Big)\\ &\quad + \sum_{i=1}^{N'} \Big(P(C_i = 1 | \mathbf{x}_{a_i^*}, \boldsymbol{\theta}^*) - P(C_i = 1 | \mathbf{x}_{a_i}, \boldsymbol{\theta}^*)\Big)\end{aligned}$$

where a_i^* is the optimal recommendation to make at time i according to the oracle strategy, a_i is the learning algorithm's choice, ϵ is users' marginal click probability, and R_i is the one-step expected regret at the i-th interaction. The third inequality is based on the expectation of additional user returns from oracle recommendation strategy against our algorithm's choices over the whole time period.

The first term on the right-hand side of Eq (7) measures the expected regret from the interactions where the user does not return; accordingly, the second term measures the expected regret in immediate clicks if the user returns. Based on the confidence bound of users' click and return probability estimations stated in Eq (4) and (5), we can prove the upper regret bound of r^2Bandit in the following theorem.

Theorem 3.1. *With probability at least $(1-\delta_1)\cdot(1-\delta_2)$, the expected cumulative regret of r^2Bandit after N rounds of interactions satisfies,*

$$\boldsymbol{R}(N) \le \boldsymbol{R}(i'-1) + 2(\epsilon\alpha_N^\beta + \frac{\epsilon}{\tilde{\epsilon}}\alpha_N^\theta)\sqrt{2dN\ln(\frac{NL}{\eta d}+1)} + \frac{1}{\tilde{\epsilon}}\sqrt{\frac{1}{2}\ln\frac{2}{\delta_2}}\sum_{i=i'}^{N}\frac{1}{\sqrt{i-1}}$$

in which $\alpha_N^\theta = \sqrt{d\ln(\frac{\eta+NL}{\delta_1})} + \sqrt{\eta}M$, $\alpha_N^\beta = \sqrt{d\ln(\frac{\eta+NL\tau}{\delta_1})} + \sqrt{\eta}H$, $\tilde{\epsilon} > 0$ *and* $i' = \Big\lceil 1 + \frac{\ln(2/\delta_2)}{2(\epsilon-\tilde{\epsilon})^2}\Big\rceil$.

Proof. For the first term of the right-hand side of Eq (7),

$$\begin{aligned}&P(\Delta_i \le \tau | \mathbf{x}_{a_i^*}, \boldsymbol{\beta}^*) - P(\Delta_i \le \tau | \mathbf{x}_{a_i}, \boldsymbol{\beta}^*) \qquad (8)\\ \le& P(\Delta_i \le \tau | \mathbf{x}_{a_i^*}, \hat{\boldsymbol{\beta}}_i) + \alpha_i^\beta \|\mathbf{x}_{a_i^*}\|_{\mathbf{A}_i^{-1}} - P(\Delta_i \le \tau | \mathbf{x}_{a_i}, \boldsymbol{\beta}^*)\\ \le& P(\Delta_i \le \tau | \mathbf{x}_{a_i}, \hat{\boldsymbol{\beta}}_i) + \alpha_i^\beta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} + \frac{1}{\hat{\epsilon}_i}\Big(P(C_i = 1 | \mathbf{x}_{a_i}, \hat{\boldsymbol{\theta}}_i) + \alpha_i^\theta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}}\Big)\\ &- \frac{1}{\hat{\epsilon}_i}\Big(P(C_i = 1 | \mathbf{x}_{a_i^*}, \hat{\boldsymbol{\theta}}_i) + \alpha_i^\theta \|\mathbf{x}_{a_i^*}\|_{\mathbf{A}_i^{-1}}\Big) - P(\Delta_i \le \tau | \mathbf{x}_{a_i}, \boldsymbol{\beta}^*)\\ \le& 2\alpha_i^\beta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} + \frac{1}{\hat{\epsilon}_i}\Big(P(C_i = 1 | \mathbf{x}_{a_i}, \hat{\boldsymbol{\theta}}_i) + \alpha_i^\theta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} - P(C_i = 1 | \mathbf{x}_{a_i^*}, \boldsymbol{\theta}^*)\Big)\end{aligned}$$

where the first and third inequalities are based on the upper confidence bound of the estimated return probability in Eq (5), and the second equality is based on our derived action selection strategy in Eq (6). By substituting Eq (8) into Eq (7) and also considering the fact that $N' \le N$, we have,

$$\begin{aligned}\mathbf{R}(N) &\le \sum_{i=1}^{N}\frac{\epsilon}{\hat{\epsilon}_i}\Big(P(C_i = 1 | \mathbf{x}_{a_i}, \hat{\boldsymbol{\theta}}_i) + \alpha_i^\theta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} - P(C_i = 1 | \mathbf{x}_{a_i^*}, \boldsymbol{\theta}^*)\Big)\\ &\quad + \sum_{i=1}^{N} 2\epsilon\alpha_i^\beta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} + \sum_{i=1}^{N}\Big(P(C_i = 1 | \mathbf{x}_{a_i^*}, \boldsymbol{\theta}^*) - P(C_i = 1 | \mathbf{x}_{a_i}, \boldsymbol{\theta}^*)\Big)\\ &= \sum_{i=1}^{N} 2\epsilon\alpha_i^\beta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} + \sum_{i=1}^{N}\frac{\hat{\epsilon}_i - \epsilon}{\hat{\epsilon}_i}\Big(P(C_i = 1 | \mathbf{x}_{a_i^*}, \boldsymbol{\theta}^*) - P(C_i = 1 | \mathbf{x}_{a_i}, \boldsymbol{\theta}^*)\Big)\\ &\quad + \sum_{i}^{N}\frac{\epsilon}{\epsilon_i}\Big(P(C_i = 1 | \mathbf{x}_{a_i}, \hat{\boldsymbol{\theta}}_i) - P(C_i = 1 | \mathbf{x}_{a_i}, \boldsymbol{\theta}^*) + \alpha_i^\theta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}}\Big)\\ &\le \sum_{i=1}^{N} 2\epsilon\alpha_i^\beta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} + \sum_{i=1}^{N}\frac{\hat{\epsilon}_i - \epsilon}{\hat{\epsilon}_i}\Big(P(C_i = 1 | \mathbf{x}_{a_i^*}, \boldsymbol{\theta}^*) - P(C_i = 1 | \mathbf{x}_{a_i}, \boldsymbol{\theta}^*)\Big)\\ &\quad + \sum_{i}^{N}\frac{\epsilon}{\epsilon_i} 2\alpha_i^\theta \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} \qquad (9)\end{aligned}$$

in which the last inequality is based on the upper confidence bound of estimated click probability in Eq (4).

The second term of the right-hand side of Eq (9) can be understood as the additional regret caused by the online estimation of the expected future click ϵ. As $\hat{\epsilon}_i = \frac{1}{i-1}\sum_{j=1}^{i-1} C_j$ for $i > 1$, according to

Hoeffding's inequality, we have $|\hat{\epsilon}_i - \epsilon| \leq \sqrt{\frac{1}{2(i-1)} \ln \frac{2}{\delta_2}}$, with probability at least $1-\delta_2$. We define $\tilde{\epsilon}$ as a lower bound of $\hat{\epsilon}_i$, then by simply rewriting and variable replacement, we have $\hat{\epsilon}_i > \tilde{\epsilon}$ when $i \geq 1 + \frac{\ln(2/\delta_2)}{2(\epsilon-\tilde{\epsilon})^2}$. Define $i' = \left\lceil 1 + \frac{\ln(2/\delta_2)}{2(\epsilon-\tilde{\epsilon})^2} \right\rceil$, we have $\sum_{i=i'}^{N} \frac{\hat{\epsilon}_i - \epsilon}{\hat{\epsilon}_i} (P(C_i = 1|\mathbf{x}_{a_i^*}, \boldsymbol{\theta}^*) - P(C_i = 1|\mathbf{x}_{a_i}, \boldsymbol{\theta}^*)) \leq \frac{1}{\tilde{\epsilon}} \sum_{i=i'}^{N} \sqrt{\frac{\ln(2/\delta_2)}{2(i-1)}}$ and $\sum_{i=i'}^{N} \frac{\epsilon}{\hat{\epsilon}_i} 2\alpha_i^{\theta} \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} \leq \sum_{i=i'}^{N} \frac{2\epsilon}{\tilde{\epsilon}} \alpha_i^{\theta} \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}}$.

Substituting them into the right-hand side of Eq (9),

$$\mathbf{R}(N) \leq \mathbf{R}(i'-1) + \sum_{i=i'}^{N} 2\epsilon \alpha_i^{\beta} \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} + \sum_{i=i'}^{N} \frac{2\epsilon}{\hat{\epsilon}_i} \alpha_i^{\theta} \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} \tag{10}$$

$$+ \sum_{i=i'}^{N} \frac{\hat{\epsilon}_i - \epsilon}{\hat{\epsilon}_i} \left(P(C_i = 1|\mathbf{x}_{a_i^*}, \boldsymbol{\theta}^*) - P(C_i = 1|\mathbf{x}_{a_i}, \boldsymbol{\theta}^*) \right)$$

$$\leq \mathbf{R}(i'-1) + \sum_{i=i'}^{N} (2\epsilon \alpha_i^{\beta} + \frac{2\epsilon}{\tilde{\epsilon}} \alpha_i^{\theta}) \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} + \frac{1}{\tilde{\epsilon}} \sqrt{\frac{1}{2} \ln \frac{2}{\delta_2}} \sum_{i=i'}^{N} \frac{1}{\sqrt{i-1}}$$

$$\leq \mathbf{R}(i'-1) + 2(\epsilon \alpha_N^{\beta} + \frac{\epsilon}{\tilde{\epsilon}} \alpha_N^{\theta}) \sum_{i=i'}^{N} \|\mathbf{x}_{a_i}\|_{\mathbf{A}_i^{-1}} + \frac{1}{\tilde{\epsilon}} \sqrt{\frac{1}{2} \ln \frac{2}{\delta_2}} \sum_{i=i'}^{N} \frac{1}{\sqrt{i-1}}$$

$$\leq \mathbf{R}(i'-1) + 2(\epsilon \alpha_N^{\beta} + \frac{\epsilon}{\tilde{\epsilon}} \alpha_N^{\theta}) \sqrt{2dN \ln(\frac{NL}{\eta d} + 1)} + \frac{1}{\tilde{\epsilon}} \sqrt{\frac{1}{2} \ln \frac{2}{\delta_2}} \sum_{i=i'}^{N} \frac{1}{\sqrt{i-1}}$$

in which $\mathbf{R}(i'-1)$ is the cumulative regret from the first $(i'-1)$ iterations, which is independent from N and only related to the difference between ϵ and $\tilde{\epsilon}$. And the last inequality is based on the property of self-normalized matrix norm [1]. □

With some simple rewriting, it is easy to verify that r^2Bandit achieves a subliner upper regret bound of $O(\sqrt{N} \log N + \sqrt{N})$ with a high probability. Note that comparing with the upper regret bound in traditional contextual bandit algorithms, e.g., [12, 20], we have an additional regret term $O(\sqrt{N})$ which is caused by the online estimation of future clicks. But since the first term $O(\sqrt{N} \log N)$ dominates the overall regret, this additional regret term does not affect the order of the resulting regret bound. Next we prove if an algorithm only models immediate click, a linearly increasing regret is inevitable, due to the fact that a user would return less often because of bad recommendations.

Without lose of generality, assume there are two types of recommendation candidates, which have the same click probability, but different return probabilities. The first type of candidates, denoted as $\mathcal{A}_1$, have a return probability $P(\Delta_i \leq \tau|\mathbf{x}_{a_i}, \boldsymbol{\beta}^*) = p + c$, in which $p > 0$, $c > 0$ and $p + c \in (0, 1)$; and the second type, denoted as $\mathcal{A}_2$, has a return probability $P(\Delta_i \leq \tau|\mathbf{x}_{a_i}, \boldsymbol{\beta}^*) = p$. Consider the first term on the right-hand side of Eq (7): if the algorithm selected a candidate from $\mathcal{A}_1$ at the i-th iteration, the one-step regret is zero; otherwise if it chose from $\mathcal{A}_2$, we have,

$$P(\Delta_i \leq \tau|\mathbf{x}_{a_i^*}, \boldsymbol{\beta}^*) - P(\Delta_i \leq \tau|\mathbf{x}_{a_i}, \boldsymbol{\beta}^*) = p + c - p = c \tag{11}$$

which is a constant over time.

Substitute Eq (11) into Eq (7), the upper regret bound of any algorithm that does not model user return can be derived as $O(\frac{|\mathcal{A}_2|Nc}{|\mathcal{A}_1|+|\mathcal{A}_2|})$, which is linear with respect to N. And the proof is straightforward: although the algorithm might achieve a sublinear regret on the second component of Eq (7), it makes constant mistakes in choosing between candidates in $\mathcal{A}_1$ and $\mathcal{A}_2$, as they have the same click probability. A similar linear regret conclusion also applies to algorithms that only model user return [8, 18]. And it is easy to verify that a linear regret generally applies to situations in which a user's click decisions are independent of return decisions. This proof supports the necessity of modeling both a user's immediate click and expected future clicks as a result of user return in optimizing long-term user engagement. Our empirical evaluation confirms this regret analysis.

4 EXPERIMENTS

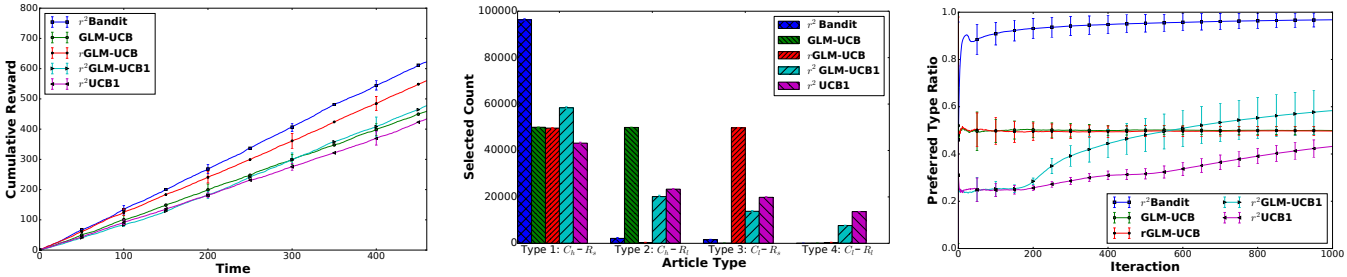
We performed extensive empirical evaluations of our developed r^2Bandit algorithm on a synthetic dataset via simulations and a large collection of real-world user click logs extracted from a major Web portal's news recommendation module.

A set of bandit algorithms are employed as baselines. To study the importance of user return modeling, we included GLM-UCB [12], which uses a generalized linear model with a logit link function to predict user clicks and uses its click estimation's upper confidence bound to select recommendations during online learning. GLM-UCB's click estimation and exploration strategy are the same as that in r^2Bandit, but it does not model user return. In the meanwhile, we also included a variant of GLM-UCB, which uses an inverse link function to model user return time, but does not consider user clicks. We refer to it as rGLM-UCB. To verify the effectiveness of confidence estimation based arm selection strategy, we designed a baseline using the same set of generalized linear models as in r^2Bandit for click and return estimation, but using the UCB1 strategy [4] for action selection. We refer to it as r^2GLM-UCB1. In the end, to verify the importance of context modeling for long-term user engagement optimization, we also included a context-free bandit algorithm based on UCB1. It uses a Bernoulli distribution to model user clicks and an exponential distribution to model return time. We refer to it as r^2UCB1. Various evaluation metrics were used to compare the algorithms.

4.1 Comparisons in simulations

4.1.1 Simulation Setting. In simulation, we simulate a personalized environment: we generate N users, each of whom is associated with a d-dimensional parameter $\boldsymbol{\theta}_u \in \mathbb{R}^d$ characterizing the user's click preference, and a d-dimensional parameter $\boldsymbol{\beta}_u$ depicting his/her return decisions. $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_u$ are drawn from a multivariate Normal distribution $N(0, I^d)$ and normalized to $\|\boldsymbol{\theta}_u\|_2 = 1$ and $\|\boldsymbol{\beta}_u\|_2 = 1$. These $\boldsymbol{\theta}$s and $\boldsymbol{\beta}$s are treated as the ground-truth parameters to generate user click and return time after an item a_i is recommended. In particular, the click response $C_{u,i}$ is sampled from a Bernoulli distribution defined by a logistic regression model $\frac{1}{1+\exp(-\boldsymbol{\theta}_u^{\mathsf{T}} \mathbf{x}_{a_i} + \sigma)}$, where $\mathbf{x}_{a_i}$ is the context vector of a_i and σ is sampled from a zero-mean Gaussian $N(0, \zeta^2)$ to corrupt the observations with white noise. Similarly, the return time $\Delta_{u,i}$ is sampled from an exponential distribution with its rate parameter $\lambda_{u,i}$ specified as $\lambda_{u,i} = \exp(\boldsymbol{\beta}_u^{\mathsf{T}} \mathbf{x}_{a_i} + \gamma)$, where γ is drawn from another zero-mean Gaussian $N(0, \xi^2)$. The corrupted click response and return time of the selected recommendation are fed back to the learning algorithms, but the ground-truth parameters, i.e., $(\boldsymbol{\theta}_u, \boldsymbol{\beta}_u)$, are kept hidden from them.

To study the importance of user return modeling in optimizing users' long-term engagement, we require the recommended items to exhibit different effects on users' click and return decisions. For example, some items might have higher click probabilities but make a user return in a longer time. To achieve so, we generate a size-K recommendation candidate pool $\mathcal{A}^u$ for each user u, and



(a) Cumulative clicks over time (b) Distribution of selected item types (c) Evolution of preferred item type ratio

Figure 1: Performance comparison in simulations.

enforce each generated candidate to fall into one of four following categories according to the ground-true model parameters $(\boldsymbol{\theta}_u, \boldsymbol{\beta}_u)$,

- Type 1: items with *high* click probability but *short* expected return time, denoted as C_h-R_l;
- Type 2: items with *high* click probability but *long* expected return time, denoted as C_h-R_l;
- Type 3: items with *low* click probability but *short* expected return time, denoted as C_l-R_s;
- Type 4: items with *low* click probability and *long* expected return time, denoted as C_l-R_l.

It is clear that the type 1 candidates should be the preferred choice for recommendation in most cases, as it allows an algorithm to collect more clicks in a given period of time. But the introduction of type 2 candidates complicates the choice between immediate click and expected future clicks. As we discussed in the regret analysis, if an algorithm does not model user return, it cannot recognize the recommendations that make users return less often, which in turn reduces its opportunities to make more recommendations in a long run. Correspondingly, the introduction of type 3 candidates traps algorithms that do not model immediate user click.

In our simulation, the separation of these four types of candidates is constructed by constraining the parameters of Bernoulli distribution for click generation and exponential distribution for return time sampling with respect to the given $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_u$. For example, to create a C_h-R_s type candidate, we enforce its context vector $\mathbf{x}_a$ to satisfy $\boldsymbol{\theta}_u^\top \mathbf{x}_a > C_h$ and $\boldsymbol{\beta}_u^\top \mathbf{x}_a > R_s$ (as the expected return time is the reciprocal of the rate parameter in an exponential distribution). Rejection sampling is used to efficiently create the context vectors $\mathbf{x}_a$ with each dimension sampled from a uniform distribution $U(0, 1)$. To increase learning complexity, in each iteration, only a subset of items in $\mathcal{A}^u$ will be presented to the algorithms, and we guarantee that there are equal number of these four types of items in the presented candidate pool. We should note that as Gaussian noise will be added to the click and return time feedback at each round of interaction independently, the aforementioned simulation procedure will maintain the categorization of these recommendation candidates in expectation. This further increases the learning complexity in our simulated environment.

4.1.2 Results Analysis. Under the simulation settings described above, we fixed the user number N to 100, article pool size K in each user to 200, which means that the number of each type of items is 50. We varied the thresholds of (C_h, C_l, R_s, R_l) and noise parameters (ζ, ξ) in our experiments, but found consistent relative comparison results. Therefore, we fixed $(C_h = 0.8, C_l = 0.2, R_s = 0.8, R_l = 0.2)$ and $(\zeta = 0.1, \xi = 0.1)$, and reported the corresponding results of all algorithms. To reduce the randomness in this simulation-based evaluation, mean and standard deviation of each evaluation metric are reported from 10 independent runs of all algorithms.

We first compared the algorithms by their accumulated clicks during the interaction with users in Figure 1 (a). From the results, it is clear that r^2Bandits collected the most clicks from users at any point of time since the interaction started; and the slope of its accumulated clicks was steeper than other algorithms', which means the gain from r^2Bandits kept increasing faster as time elapsed. In addition, we can also clearly observe that the variance of cumulative reward collected from r^2Bandits was consistently smaller than those in the baselines. This indicates it successfully recognized the items with high click probability and short return in the early stage. This is verified by further detailed result analysis next.

As Figure 1 (a) measures both user clicks and return time, it is important to decompose them and understand how they contribute to optimize the cumulative clicks over time. We first analyzed the distribution of the four types of items selected by each algorithm in Figure 1 (b). We can clearly observe that because GLM-UCB does not model user return, thought it successfully recommended candidates with high click probability, it failed to filter those that make users return in a longer time (i.e., the type C_h-R_l). On the other hand, *r*GLM-UCB, as it does not model click, tended to choose items with shorter user return time, but failed to differentiate those with low click probabilities (i.e., the type C_l-R_s). Among the baselines that model both user click and return time, i.e., r^2GLM-UCB1 and r^2UCB1, they selected the C_h-R_s type items more often, as our r^2Bandits did. But because of their less effective exploration strategy and parameter estimation methods, they tended to select other types of items more often than r^2Bandits. This conclusion becomes especially evident, when we compare r^2Bandits with r^2GLM-UCB1, as they used the same set of generalized linear models to estimate user clicks and return time.

It is also important to verify how different algorithms' item selections evolve during the online learning. We investigate the ratio of preferred candidate type, i.e., C_h-R_s, among the selected items over time in different algorithms. In Figure 1 (c), we report the results with respect to the iterations to align the comparison across algorithms, as actual user return time depends on the algorithm. We can observe that, over the entire interaction history, both GLM-UCB and *r*GLM-UCB can only select the preferred type of items 50% of time, as they only model click or return. This result directly supports our linear regret conclusion on this type of algorithms. Because of explicit modeling of both user click and return time, the selection ratio of preferred items in r^2GLM-UCB1, r^2UCB1 and

r^2Bandit kept increasing during online update. In particular, the ratio in r^2Bandit increased significantly faster and quickly converged. This confirms the effectiveness of estimation confidence based exploration strategy for online learning.

4.2 Comparisons in news recommendation logs

4.2.1 Dataset, metrics, and offline evaluation protocol. **Dataset:** We collected 4 weeks user visitation data extracted from a major Web portal's news recommendation module during the summer of 2016. Each logged event contains the following information: timestamp, anonymized user ID, displayed article ID, corresponding article features, and click label.

Pre-processing was performed on this dataset. First, principal component analysis was used to reduce the dimensionality of the article features to 23. Second, each user's visitation sequence was segmented into sessions, which were defined by 30 minutes inactive time threshold. The offline evaluation was performed in sessions: for each session, a candidate article pool was created by retaining the top 10% most popular articles in the same day, together with the originally displayed articles in that session. Third, the time interval between two consecutive user sessions was computed as the "return time" of the first session. As a result, the return time is valid only when there are at least two sessions. For this reason, we removed users with less than two sessions in this data set. In the rest of users, if one user did not return to the site within four weeks, we labeled this user's last observation as "no return."

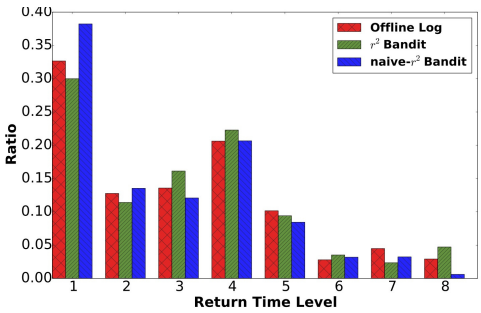


Figure 2: Discretized user return time distribution.

After the pre-processing, there are in total 18,882 users, 188,384 articles, and 9,984,879 logged events segmented into 1,123,583 sessions. The distribution of users' return time on this dataset is shown in Figure 2. To avoid disclosing sensitive business information about the Web portal system, we categorize the actual return time into 8 levels. The lower a level is, the shorter the return time is. Based on this distribution, we set the return time threshold τ to 24 hours, which is used to compute the return probability in different algorithms' action selection strategy. In particular, on this dataset, the distribution of observations per user is highly unbalanced: over 71% observations come from less than 30% users. Learning independent set of model parameters for every single user becomes impractical. As a result, we set the same set of bandit parameters for all the users both in our proposed solutions and in all the baselines.

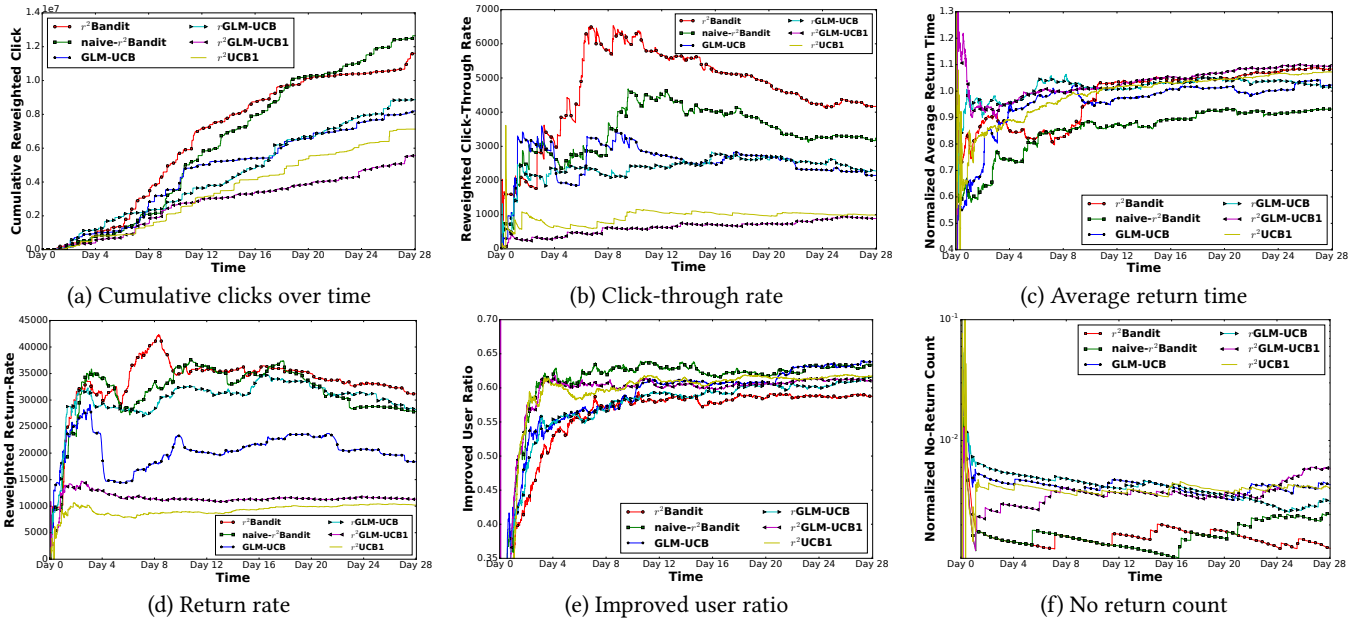
Offline Evaluation: Evaluation of bandit algorithms is challenging, as one needs to label instances upon algorithm's request on the fly. We adopted the offline evaluation protocol proposed in [21] to compare the online learning algorithms on this offline dataset. Specifically, the offline evaluator takes an algorithm A as input and outputs a set of "valid" events E from the sequence of logged events, on which to base the evaluation. At every interaction, if given the current observation history, the algorithm A chooses the same item a_t as the one that was recorded by the logging policy in the offline data, the event $(\mathbf{x}_{a_t}, C_{a_t}, \Delta_{a_t})$ is retained as a valid event and added to the history and valid event set E. Then the model parameters of A will be updated accordingly. Otherwise, the event is ignored, and the algorithm proceeds to the next event in the logged data without any update in its state.

It is proved in [21] that this offline evaluator is unbiased if the logging policy chooses each item uniformly at random. This requires each event is retained by the logging policy with uniformly. But when the logging policy is not uniformly random, reweighting the logged events is necessary to correct the sampling bias in the evaluator, as $\hat{v}(\pi) := \frac{1}{N}\sum_{i=1}^{N} \frac{\pi_A(a_i)}{p_{a_i}} r_{a_i}$, where p_{a_i} is the probability of observing article a_i in the logging policy, $\pi_A(a_i)$ is the corresponding probability in the evaluated algorithm A, and r_{a_i} is the target evaluation metric (e.g., cumulative clicks or return time in our case). The intuition behind the estimator $\hat{v}(\pi)$ is that, during data collection, some items have higher selection probability, so that they are overly represented in the offline data. Normalizing their influence by their popularity will correct such sampling bias. And it is proved in [2, 22] that the evaluator is unbiased in the sense that $E[\hat{v}(\pi)] = v(\pi)$.

Unbiased offline evaluation is typically used to evaluate clicks or click-through rate in online algorithms. In our offline dataset, each logged event is also associated with a return time triggered by the choice of logging policy. Similar proof of unbiased offline evaluation applies to return time as well: when the algorithm's recommendation matches with the choice in the logged event, the return time can be considered as being triggered by the algorithm's choice [21]. Therefore in our experiments, we extent this offline evaluation protocol to evaluate both clicks and return time, and also other metrics resulted from these two types of feedback to be discussed below.

Evaluation Metrics: We include a comprehensive set of evaluation metrics, which track both user clicks and return time resulted from the recommendation as the interaction proceeds.

- Cumulative clicks over time: it is the cumulative clicks till any particular time point during the interaction. A larger cumulative clicks over time indicate an algorithm makes users click more and return more often.
- Click-through rate (CTR): it is computed as the ratio of the number of clicks an algorithm receives among all recommendations it has made. A higher CTR indicates users tend to click on the recommended items.
- Average return time: it is the average of time between a user's consecutive visits up till a particular time point. We do not count the recommendations that drive users away (i.e., no return), since return time is undefined in this case. This case will be considered in another metric called "No return count". A shorter average return time means an algorithm makes users return more often. We normalize an algorithm's average return time by the logged average return time in the offline dataset to avoid disclose sensitive business information of the Web portal.



(a) Cumulative clicks over time (b) Click-through rate (c) Average return time

(d) Return rate (e) Improved user ratio (f) No return count

Figure 3: Experiment results on real-world news recommendation log data.

- Return rate: it is the ratio of the number of recommendations that lead a user to return within the threshold τ among all recommendations the algorithm has made. A larger return rate indicates an algorithm makes users more likely to return.
- Improved user ratio: it is computed as the percentage of users whose average return time get reduced compared to his/her logged average return time in the offline data, up till a particular time during interaction. A larger ratio of improved users indicates an algorithm makes more users engaged in the system.
- No return count: it is the number of users who left the system after a recommendation was made (e.g., those labeled as "no return"). A smaller no return count indicates an algorithm keeps more users in the system. No return count is also normalized by the total number of no return users in the offline data.

In our evaluation, we reweighted cumulative clicks over time, click-through rate, return rate based on the offline evaluation protocol to reduce bias from the article distribution in this offline dataset.

4.2.2 Experiment Results. On this real-world dataset, we include a variant of r^2Bandit, and name it as naive-r^2Bandit. Instead of estimating the expected future clicks ϵ on the fly, we set it to the average click through rate on this dataset. This model is designed to better serve users with fewer observations for expected click estimation. Comparisons on all six metrics among all the algorithms are reported in Figure 3.

First of all, we can observe that our r^2Bandit and naive-r^2Bandit collected significantly more clicks over the same period of time than all the other baselines, and their improvement margins kept increasing. In practice, such performance gain can be translated to improved revenue resulted from this news recommendation module, especially when long-term engagement is emphasized. This directly proves the effectiveness of our proposed algorithm in a recommender system.

The click-through rate and average return time explain where the improvement comes from. In Figure 3 (b), we can find that r^2Bandit achieved the highest CTR, which is about twice as that in GLM-UCB and rGLM-UCB, and more than 5 times of that in r^2GLM-UCB1 and r^2UCB1. And Figure 3 (c) shows that naive-r^2Bandit achieved the shortest average return time. In particular, according to the normalized average return time, naive-r^2Bandit reduced users' average return time interval by 18% to 25% comparing to that recorded in the offline data. This allows our algorithms to make much more recommendations than other baselines in the same period of time. To make this advantage easier to comprehend, we also reported the distribution of return time resulted from our algorithms in Figure 2. In this comparison, the ratio of shorter returns, i.e., those in lower levels, got greatly increased, and the ratio of longer returns got significantly reduced accordingly.

Figure 3 (d), (e) and (f) illustrate users' return behavior resulted from different algorithms from different perspectives. Figure 3 (d) shows that r^2Bandit and naive-r^2Bandit achieved the highest user return rate, which is around 1.8 times of that in GLM-UCB and 3.5 times of that in r^2UCB1 and r^2GLM-UCB1. This indicates our proposed algorithms gain at least one more chance to make recommendations to every single user. This gain is significant for any practical recommender system. Figure 3 (e) shows that around 63% users will return in a shorter time than the recorded average return time achieved by the originally deployed algorithm in the Web portal. Figure 3 (f) shows that both r^2Bandit and naive-r^2Bandit achieved significantly lower no-return user count, which means a higher user retention rate. This is vital for a recommender system to increase its user base and succeed in market competition.

Comparing baselines, we can find that as expected GLM-UCB outperformed rGLM-UCB in click-through rate but fell behind in user return rate. As a result, their final cumulative clicks over time were quite similar. r^2GLM-UCB1 and r^2UCB1 performed the worst in most metrics in this evaluation. The main reason is that on this large dataset, the size of article pool is considerably large, on which UCB1 exploration strategy is less effective. These results verify the

(a) Top clicked articles (b) Top returning articles

Figure 4: Word cloud of algorithm selected article content.

importance of context modeling for user long-term engagement optimization and exploration strategy.

To further reveal how our developed algorithm recognizes the different effect of recommendations on users' click and return decisions, we looked into the selected articles that tend to make users click more and those that tend to make users return more frequently. We used the learnt models to rank the articles by their estimated click and return probabilities accordingly, extracted text content from the top 5000 articles of each type, and then generated world clouds to summarize those documents in Figure 4. We should note our context features in $\mathbf{x}_a$ include richer type of information beyond just article content, but we only visualized the results based on the article content. It is interesting to find that articles make users more likely to click are mostly about technology, sports and entertainment, while those make user return more often are mostly about politics and NBA games. Considering the dataset was collected in summer 2016, when the U.S. presidential campaigns and NBA finals were on going, our algorithm reasonably captured users' interest and encouraged them to return more often to get the latest updates on those news events.

5 CONCLUSION

Most of existing recommendation algorithms impose a strong assumption that users' return behavior is always consistent with their immediate responses, or the probability of user return is independent from the recommendations being made, such that few of them explicitly consider users' temporal return behavior when making recommendations. In this work, we formulate the optimization of long-term user engagement as a sequential decision making problem. A recommendation is made by not only its estimated immediate user click but also the expected clicks resulted from the user's future return. We develop a bandit-based online solution for the problem. Rigorous theoretical analysis proves a high-probability sublinear upper regret bound of our proposed solution; while if only immediate click is optimized, a linear regret is inevitable. Extensive experiments on both simulations and a large collection of real-world click logs verified the improvement of our algorithm in optimizing the long-term reward from users.

Our current solution handles users independently. It is important to explore the dependence among users to perform collaborative online learning [7, 27], such that observations can be leveraged across users to reduce the model learning complexity. As r^2Bandit is able to predict expected user return time, proactive learning across users becomes possible: an algorithm can probe a few current users for the most informative feedback to update itself for better serving the other about-to-come users. And the temporal dynamics of the candidate pool can also be modeled to better predict future clicks.

6 ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers for helpful suggestions and also thank Yahoo Research for its support on this work with an internship. This work was supported by the National Science Foundation under grant IIS-1553568 and IIS-1618948.

REFERENCES

[1] Yasin Abbasi-yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved Algorithms for Linear Stochastic Bandits. In *NIPS '11*. 2312–2320.
[2] Lihong Li Sham Kakade Alexander L. Strehl, John Langford. 2011. Learning from Logged Implicit Exploration Data. In *NIPS '10*. 2217–2225.
[3] Peter Auer. 2002. Using Confidence Bounds for Exploitation-Exploration Tradeoffs. *JMLR* 3 (2002), 397–422.
[4] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.* 47, 2-3 (May 2002), 235–256.
[5] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *36th FOCS*. 322–331.
[6] Nicola Barbieri, Fabrizio Silvestri, and Mounia Lalmas. 2016. Improving Post-Click User Engagement on Native Ads via Survival Analysis. In *WWW '16*. 761–770.
[7] Nicolo Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. 2013. A gang of bandits. (2013), 737–745.
[8] Olivier Chapelle. 2014. Modeling Delayed Feedback in Display Advertising. In *KDD '14*. ACM, New York, NY, USA, 1097–1105.
[9] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *WWW '07*. 271–280.
[10] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD '16*. ACM, 1555–1564.
[11] Nan Du, Yichen Wang, Niao He, Jimeng Sun, and Le Song. 2015. Time-Sensitive Recommendation From Recurrent User Activities. In *NIPS '15*. 3492–3500.
[12] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. 2010. Parametric bandits: The generalized linear case. In *NIPS*. 586–594.
[13] J. Gittins, K. Glazebrook, and R. Weber. 2011. *Multi-armed Bandit Allocation Indices*. John Wiley & Sons.
[14] L.P. Kaelbling, M.L. Littman, and A.W. Moore. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence* 4, 1 (1996), 237–285.
[15] Komal Kapoor, Karthik Subbian, Jaideep Srivastava, and Paul Schrater. 2015. Just in Time Recommendations: Modeling the Dynamics of Boredom in Activity Streams. In *WSDM '15*. ACM, New York, NY, USA, 233–242.
[16] Komal Kapoor, Mingxuan Sun, Jaideep Srivastava, and Tao Ye. 2014. A Hazard Based Approach to User Return Time Prediction. In *KDD '14*. 1719–1728.
[17] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[18] Mounia Lalmas, Janette Lehmann, Guy Shaked, Fabrizio Silvestri, and Gabriele Tolomei. 2015. Promoting Positive Post-Click Experience for In-Stream Yahoo Gemini Users. In *KDD '15*. ACM, New York, NY, USA, 1929–1938.
[19] Mounia Lalmas, Heather O'Brien, and Elad Yom-Tov. 2014. Measuring user engagement. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 6, 4 (2014), 1–132.
[20] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW '10*. ACM, 661–670.
[21] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased Offline Evaluation of Contextual-bandit-based News Article Recommendation Algorithms. In *WSDM '11*. ACM, New York, NY, USA, 297–306.
[22] Lihong Li, Jin Young Kim, and Imed Zitouni. 2015. Toward predicting the outcome of an A/B experiment for search relevance. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. 37–46.
[23] Peter McCullagh. 1984. Generalized Linear Models. *European Journal of Operational Research* 16, 3 (1984), 285–292.
[24] Heather L O'Brien and Elaine G Toms. 2008. What is user engagement? A conceptual framework for defining user engagement with technology. *JASIST* 59, 6 (2008), 938–955.
[25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW '01*. 285–295.
[26] X. Su and T.M. Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *AAI* 2009 (2009), 4.
[27] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. 2016. Contextual Bandits in a Collaborative Environment. In *SIGIR '16*. 529–538.
[28] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond Clicks: Dwell Time for Personalization. In *RecSys '14*. 113–120.