Modeling Attrition in Recommender Systems with Departing Bandits

Omer Ben-Porat[*1], **Lee Cohen**[*1], **Liu Leqi**[*2], **Zachary C. Lipton**[2], **Yishay Mansour**[1,3]

[1] Blavatnik School of Computer Science, Tel-Aviv University
[2] Machine Learning Department, Carnegie Mellon University
[3] Google Research

Abstract

Traditionally, when recommender systems are formalized as multi-armed bandits, the policy of the recommender system influences the rewards accrued, but not the length of interaction. However, in real-world systems, dissatisfied users may depart (and never come back). In this work, we propose a novel multi-armed bandit setup that captures such policy-dependent horizons. Our setup consists of a finite set of user *types*, and multiple arms with Bernoulli payoffs. Each (user type, arm) tuple corresponds to an (unknown) reward probability. Each user's type is initially unknown and can only be inferred through their response to recommendations. Moreover, if a user is dissatisfied with their recommendation, they might depart the system. We first address the case where all users share the same type, demonstrating that a recent UCB-based algorithm is optimal. We then move forward to the more challenging case, where users are divided among two types. While naive approaches cannot handle this setting, we provide an efficient learning algorithm that achieves $\tilde{O}(\sqrt{T})$ regret, where T is the number of users.

1 Introduction

At the heart of online services spanning such diverse industries as media consumption, dating, financial products, and more, recommendation systems (RSs) drive personalized experiences by making curation decisions informed by each user's past history of interactions. While in practice, these systems employ diverse statistical heuristics, much of our theoretical understanding of them comes via stylized formulations within the multi-armed bandits (MABs) framework. While MABs abstract away from many aspects of real-world systems they allow us to extract crisp insights by formalizing fundamental tradeoffs, such as that between exploration and exploitation that all RSs must face (Joseph et al. 2016; Liu and Ho 2018; Patil et al. 2020; Ron, Ben-Porat, and Shalit 2021). As applies to RSs, exploitation consists of continuing to recommend items (or categories of items) that have been observed to yield high rewards in the past, while exploration consists of recommending items (or categories of items) about which the RS is uncertain but that could *potentially* yield even higher rewards.

[*]Equal contribution, alphabetical order.

In traditional formalizations of RSs as MABs, the recommender's decisions affect only the rewards obtained. However, real-life recommenders face a dynamic that potentially alters the exploration-exploitation tradeoff: Dissatisfied users have the option to depart the system, never to return. Thus, recommendations in the service of exploration not only impact instantaneous rewards but also risk driving away users and therefore can influence long-term cumulative rewards by shortening trajectories of interactions.

In this work, we propose *departing bandits* which augment conventional MABs by incorporating these policy-dependent horizons. To motivate our setup, we consider the following example: An RS for recommending blog articles must choose at each time among two categories of articles, e.g., economics and sports. Upon a user's arrival, the RS recommends articles sequentially. After each recommendation, the user decides whether to "click" the article and continue to the next recommendation, or to "not click" and may leave the system. Crucially, the user interacts with the system for a random number of rounds. The user's departure probability depends on their satisfaction from the recommended item, which in turn depends on the user's unknown *type*. A user's type encodes their preferences (hence the probability of clicking) on the two topics (economics and sports).

When model parameters are given, in contrast to traditional MABs where the optimal policy is to play the best fixed arm, departing bandits require more careful analysis to derive an optimal planning strategy. Such planning is a local problem, in the sense that it is solved for each user. Since the user type is never known explicitly (the recommender must update its beliefs over the user types after each interaction), finding an optimal recommendation policy requires solving a specific partially observable MDP (POMDP) where the user type constitutes the (unobserved) state (more details in Section 5.1). When the model parameters are unknown, we deal with a learning problem that is global, in the sense that the recommender (learner) is learning for a stream of users instead of a particular user.

We begin with a formal definition of departing bandits in Section 2, and demonstrate that any fixed-arm policy is prone to suffer linear regret. In Section 3, we establish the UCB-based learning framework used in later sections. We instantiate this framework with a single user type in Section 4, where we show that it achieves $\tilde{O}(\sqrt{T})$ regret for T being the number of users. We then move to the more challenging case with two user types and two recommendation categories in Section 5. To analyze the planning problem, we effectively reduce the search space for the optimal policy by using a closed-form of the *expected return* of any recommender policy. These results suggest an algorithm that achieves $\tilde{O}(\sqrt{T})$ regret in this setting. In the full version of this paper (Ben-Porat et al. 2022), we also show an efficient optimal planning algorithm for multiple user types and two recommendation categories, and describe a scheme to construct semi-synthetic problem instances for this setting using real-world datasets.

1.1 Related Work

MABs have been studied extensively by the online learning community (Cesa-Bianchi and Lugosi 2006; Bubeck, Cesa-Bianchi et al. 2012). The contextual bandit literature augments the MAB setup with context-dependent rewards (Abbasi-Yadkori, Pál, and Szepesvári 2011; Slivkins 2019; Mahadik et al. 2020; Korda, Szörényi, and Li 2016; Lattimore and Szepesvári 2020). In contextual bandits, the learner observes a *context* before they make a decision, and the reward depends on the context. Another line of related work considers the dynamics that emerge when users act strategically (Kremer, Mansour, and Perry 2014; Mansour, Slivkins, and Syrgkanis 2015; Cohen and Mansour 2019; Bahar, Smorodinsky, and Tennenholtz 2016; Bahar et al. 2020). In that line of work, users arriving at the system receive a recommendation but act strategically: They can follow the recommendation or choose a different action. This modeling motivates the development of incentive-compatible mechanisms as solutions. In our work, however, the users are modeled in a stochastic (but not strategic) manner. Users may leave the system if they are dissatisfied with recommendations, and this departure follows a fixed (but possibly unknown) stochastic model.

The departing bandits problem has two important features: Policy-dependent horizons, and multiple user types that can be interpreted as unknown states. Existing MAB works (Azar, Lazaric, and Brunskill 2013; Cao et al. 2020) have addresses these phenomena separately but we know of no work that integrates the two in a single framework. In particular, while Azar, Lazaric, and Brunskill (2013) study the setting with multiple user types, they focus on a fixed horizon setting. Additionally, while Cao et al. (2020) deal with departure probabilities and policy-dependent interaction times for a single user type, they do not consider the possibility of multiple underlying user types.

The planning part of our problem falls under the framework of using Markov Decision Processes for modeling recommender-user dynamics (Shani, Heckerman, and Brafman 2005). Specifically, our problem works with partially observable user states which have also been seen in many recent bandits variants (Pike-Burke and Grünewälder 2019; Leqi et al. 2020). Unlike these prior works that focus on interactions with a single user, departing bandits consider a stream of users each of which has an (unknown) type selected among a finite set of user types.

More broadly, our RS learning problem falls under the domain of reinforcement learning (RL). Existing RL literature that considers departing users in RSs include Zhao et al. (2020b); Lu and Yang (2016); Zhao et al. (2020a). While Zhao et al. (2020b) handle users of a single type that depart the RS within a bounded number of interactions, our work deals with multiple user types. In contrast to Zhao et al. (2020a), we consider an online setting and provide regret guarantees that do not require bounded horizon. Finally, Lu and Yang (2016) use POMDPs to model user departure and focus on approximating the value function. They conduct an experimental analysis on historical data, while we devise an online learning algorithm with theoretical guarantees.

2 Departing Bandits

We propose a new online problem, called *departing bandits*, where the goal is to find the optimal recommendation algorithm for users of (unknown) types, and where the length of the interactions depends on the algorithm itself. Formally, the departing bandits problem is defined by a tuple $\langle [M], [K], \mathbf{q}, \mathbf{P}, \mathbf{\Lambda} \rangle$, where M is the number of user *types*, K is the number of *categories*, $\mathbf{q} \in [0,1]^M$ specifies a prior distribution over types, and $\mathbf{P} \in (0,1)^{K\times M}$ and $\mathbf{\Lambda} \in (0,1)^{K\times M}$ are the *click-probability* and the *departure-probability* matrices, respectively.[1]

There are T users who arrive sequentially at the RS. At every episode, a new user $t \in [T]$ arrives with a type $type(t)$. We let $\mathbf{q}$ denote the prior distribution over the user types, i.e., $type(t) \sim \mathbf{q}$. Each user of type x *clicks* on a recommended category a with probability $\mathbf{P}_{a,x}$. In other words, each click follows a Bernoulli distribution with parameter $\mathbf{P}_{a,x}$. Whenever the user clicks, she stays for another iteration, and when the user does not click (no-click), she *departs* with probability $\mathbf{\Lambda}_{a,x}$ (and stays with probability $1 - \mathbf{\Lambda}_{a,x}$). Each user t interacts with the RS (the learner) until she departs.

We proceed to describe the user-RS interaction protocol. In every iteration j of user t, the learner recommends a category $a \in [K]$ to user t. The user clicks on it with probability $\mathbf{P}_{a,type(t)}$. If the user clicks, the learner receives a reward of $r_{t,j}(a) = 1$.[2] If the user does not click, the learner receives no reward (i.e., $r_{t,j}(a) = 0$), and user t departs with probability $\mathbf{\Lambda}_{a,type(t)}$. We assume that the learner knows the value of a constant $\epsilon > 0$ such that $\max_{a,x} \mathbf{P}_{a,x} \le 1-\epsilon$ (i.e., ϵ does not depend on T). When user t departs, she does not interact with the learner anymore (and the learner moves on to the next user $t + 1$). For convenience, the departing bandits problem protocol is summarized in Algorithm 1.

Having described the protocol, we move on to the goals and performance of the learner. Without loss of generality, we assume that the online learner's recommendations are made based on a *policy* π, which is a mapping from the history of previous interactions (with that user) to recommendation categories. For each user (episode) $t \in [T]$, the learner selects a policy π_t that recommends category $\pi_{t,j} \in [K]$ at every

[1]We denote by $[n]$ the set $\{1, \dots, n\}$.

[2]We formalize the reward as is standard in the online learning literature, from the perspective of the learner. However, defining the reward from the user perspective by, e.g., considering her utility as the number of clicks she gives or the number of articles she reads induces the same model.

Algorithm 1: The Departing Bandits Protocol

Input: number of types M, number of categories K, and number of users (episodes) T

Hidden Parameters: types prior $\mathbf{q}$, click-probability $\mathbf{P}$, and departure-probability $\mathbf{\Lambda}$

```
1:  for episode $t \leftarrow 1, \dots, T$ do
2:      a new user with type $type(t) \sim \mathbf{q}$ arrives
3:      $j \leftarrow 1$, $depart \leftarrow false$
4:      while $depart$ is $false$ do
5:          the learner picks a category $a \in [K]$
6:          with probability $\mathbf{P}_{a,x}$, user $t$ clicks on $a$ and
            $r_{t,j}(a) \leftarrow 1$; otherwise, $r_{t,j}(a) \leftarrow 0$
7:          if $r_{t,j}(a) = 0$ then
8:              with probability $\mathbf{\Lambda}_{a,x}$: $depart \leftarrow true$ and user
                $t$ departs
9:          the learner observes $r_{t,j}(a)$ and $depart$
10:         if $depart$ is $false$ then
11:             $j \leftarrow j + 1$
```

iteration $j \in [N^{\pi_t}(t)]$, where $N^{\pi_t}(t)$ denotes the episode length (i.e., total number of iterations policy π_t interacts with user t until she departs).[3] The *return* of a policy π, denoted by V^π is the cumulative reward the learner obtains when executing the policy π until the user departs. Put differently, the return of π from user t is the random variable $V^\pi = \sum_{j=1}^{N^\pi(t)} r_{t,j}(\pi_{t,j})$.

We denote by π^* an optimal policy, namely a policy that maximizes the expected return, $\pi^* = \arg\max_\pi \mathbb{E}[V^\pi]$. Similarly, we denote by V^* the optimal return, i.e., $V^* = V^{\pi^*}$.

We highlight two algorithmic tasks. The first is the planning task, in which the goal is to find an optimal policy π^*, given $\mathbf{P}, \mathbf{\Lambda}, \mathbf{q}$. The second is the online learning task. We consider settings where the learner knows the number of categories, K, the number of types, M, and the number of users, T, but has no prior knowledge regarding $\mathbf{P}, \mathbf{\Lambda}$ or $\mathbf{q}$. In the online learning task, the *value* of the learner's algorithm is the sum of the returns obtained from all the users, namely

$$\sum_{t=1}^{T} V^{\pi_t} = \sum_{t=1}^{T} \sum_{j=1}^{N^\pi(t)} r_{t,j}(\pi_{t,j}).$$

The performance of the leaner is compared to that of the best policy, formally defined by the *regret* for T episodes,

$$R_T = T \cdot \mathbb{E}[V^{\pi^*}] - \sum_{t=1}^{T} V^{\pi_t}. \qquad (1)$$

The learner's goal is to minimize the expected regret $\mathbb{E}[R_T]$.

2.1 Example

The motivation for the following example is two-fold. First, to get the reader acquainted with our notations; and second, to show why fixed-arm policies are inferior in our setting.

Consider a problem instance with two user types ($M = 2$), which we call x and y for convenience. There are two categories ($K = 2$), and given no-click the departure is deterministic, i.e., $\mathbf{\Lambda}_{a,\tau} = 1$ for every category $a \in [K]$ and type $\tau \in [M]$. That is, every user leaves immediately if she does not click. Furthermore, let the click-probability $\mathbf{P}$ matrix and the user type prior distribution $\mathbf{q}$ be as in Table 1.

	Type x	Type y
Category 1	$\mathbf{P}_{1,x} = 0.5$	$\mathbf{P}_{1,y} = 0.28$
Category 2	$\mathbf{P}_{2,x} = 0.4$	$\mathbf{P}_{2,y} = 0.39$
Prior	$\mathbf{q}_x = 0.4$	$\mathbf{q}_y = 0.6$

Table 1: The departing bandits instance in Section 2.1.

Looking at $\mathbf{P}$ and $\mathbf{q}$, we see that Category 1 is better for Type x, while Category 2 is better for type y. Notice that without any additional information, a user is more likely to be type y. Given the prior distribution, recommending Category 1 in the first round yields an expected reward of $\mathbf{q}_x\mathbf{P}_{1,x} + \mathbf{q}_y\mathbf{P}_{1,y} = 0.368$. Similarly, recommending Category 2 in the first round results in an expected reward of 0.394. Consequently, if we recommend *myopically*, i.e., without considering the user type, always recommending Category 2 is better than always recommending Category 1.

Let π^a denote the fixed-arm policy that always selects a single category a. Using the tools we derive in Section 5 and in particular Theorem 5.3, we can compute the expected returns of π^1 and π^2, $\mathbb{E}[V^{\pi^1}]$ and $\mathbb{E}[V^{\pi^2}]$. Additionally, using results from Section 5.2, we can show that the optimal policy for the planning task, π^*, recommends Category 2 until iteration 7, and then recommends Category 1 for the rest of the iterations until the user departs.

Using simple calculations, we see that $\mathbb{E}[V^{\pi^*}] - \mathbb{E}[V^{\pi^1}] > 0.0169$ and $\mathbb{E}[V^{\pi^*}] - \mathbb{E}[V^{\pi^2}] > 1.22 \times 10^{-5}$; hence, the expected return of the optimal policy is greater than the returns of both fixed-arm policies by a constant. As a result, if the learner only uses fixed-arm policies (π^a for every $a \in [K]$), she suffers linear expected regret, i.e., $\mathbb{E}[R_T] = T \cdot \mathbb{E}[V^{\pi^*}] - \sum_{t=1}^{T} \mathbb{E}[V^{\pi^a}] = \Omega(T)$.

3 UCB Policy for Sub-exponential Returns

In this section, we introduce the learning framework used in the paper and provide a general regret guarantee for it.

In standard MAB problems, at each $t \in [T]$ the learner picks a single arm and receives a single sub-Gaussian reward. In contrast, in departing bandits, at each $t \in [T]$ the learner receives a return V^π, which is the cumulative reward of that policy. The return V^π depends on the policy π not only through the obtained rewards at each iteration but also through the total number of iterations (trajectory length). Such returns are not necessarily sub-Gaussian. Consequently, we cannot use standard MAB algorithms as they usually rely on concentration bounds for sub-Gaussian rewards. Furthermore, as we have shown in Section 2.1, in departing bandits fixed-arm policies can suffer linear regret (in terms of the number of users), which suggests considering a more expressive set of policies. This in turn yields another disadvantage for using MAB algorithms for departing bandits, as their regret is linear in the number of arms (categories) K.

[3] We limit the discussion to deterministic policies solely; this is w.l.o.g. (see Subsection 5.1 for further details).

As we show later in Sections 4 and 5, for some natural instances of the departing bandits problem, the return from each user is sub-exponential (Definition 3.1). Algorithm 2, which we propose below, receives a set of policies Π as input, along with other parameters that we describe shortly. The algorithm is a restatement of the *UCB-Hybrid* Algorithm from Jia, Shi, and Shen (2021), with two modifications: (1) The input includes a set of policies rather than a set of actions/categories, and accordingly, the confidence bound updates are based on return samples (denoted by $\hat{V}^{\pi}$) rather than reward samples. (2) There are two global parameters ($\tilde{\tau}$ and η) instead of two local parameters per action. If the return from each policy in Π is sub-exponential, Algorithm 2 not only handles sub-exponential returns, but also comes with the following guarantee: Its expected value is close to the value of the best policy in Π.

Algorithm 2: UCB-based algorithm with hybrid radii: UCB-Hybrid (Jia, Shi, and Shen 2021)

```
1: **Input**: set of policies $\Pi$, number of users $T$, $\tilde{\tau}, \eta$
2: **Initialize:** $\forall \pi \in \Pi : U_0(\pi) \leftarrow \infty, n(\pi) = 0$
3: **for** user $t \leftarrow 1, \dots, T$ **do**
4:     Execute $\pi_t$ such that $\pi_t \in \arg\max_{\pi \in \Pi} U_{t-1}(\pi)$ and
       receive return $\hat{V}^{\pi_t}[n(\pi_t)] \leftarrow \sum_{j=1}^{N^{\pi_t}(t)} r_{t,j}(\pi_{t,j})$
5:     $n(\pi_t) \leftarrow n(\pi_t) + 1$
6:     **if** $n(\pi_t) < 8\eta \ln T$ **then**
7:         Update $U_t(\pi_t) = \frac{\sum_{i=1}^{n(\pi_t)} \hat{V}^{\pi_t}[i]}{n(\pi_t)} + \frac{8\sqrt{\eta}\cdot\tilde{\tau}\ln T}{n(\pi_t)}$
8:     **else**
9:         Update $U_t(\pi_t) = \frac{\sum_{i=1}^{n(\pi_t)} \hat{V}^{\pi_t}[i]}{n(\pi_t)} + \sqrt{\frac{8\tilde{\tau}^2 \ln T}{n(\pi_t)}}$
```

3.1 Sub-exponential Returns

For convenience, we state here the definition of sub-exponential random variables (Eldar and Kutyniok 2012).

Definition 3.1. *We say that a random variable X is* sub-exponential *with parameters (τ^2, b) if for every γ such that $|\gamma| < 1/b$,*

$$\mathbb{E}[\exp(\gamma(X - \mathbb{E}[X]))] \leq \exp(\frac{\gamma^2\tau^2}{2}).$$

In addition, for every (τ^2, b)-sub-exponential random variables, there exist constants $C_1, C_2 > 0$ such that the above is equivalent to each of the following properties:

1. *Tails: $\forall v \geq 0 : \Pr[|X| > v] \leq \exp(1 - \frac{v}{C_1})$.*
2. *Moments: $\forall p \geq 1 : (\mathbb{E}[|X|^p])^{1/p} \leq C_2 p$.*

Let Π be a set of policies with the following property: There exist $\tilde{\tau}, \eta$ such that the return of every policy $\pi \in \Pi$ is (τ^2, b)-sub-exponential with $\tilde{\tau} \geq \tau$ and $\eta \geq \frac{b^2}{\tau^2}$. The following Algorithm 2 receives as input a set of policies Π with the associated parameters, $\tilde{\tau}$ and η. Similarly to the UCB algorithm, it maintains an upper confidence bound U for each policy, and balances between exploration and exploitation. Theorem 3.2 below shows that Algorithm 2 always gets a value similar to that of the best policy in Π up to an additive factor of $\tilde{O}\left(\sqrt{|\Pi|T} + |\Pi|\right)$. The theorem follows directly from Theorem 3 from Jia, Shi, and Shen (2021) by having policies as arms and returns as rewards.

Theorem 3.2. *Let Π be a set of policies with the associated parameters $\tilde{\tau}, \eta$. Let $\pi_1, \dots, \pi_T$ be the policies Algorithm 2 selects. It holds that*

$$\mathbb{E}\left[\max_{\pi \in \Pi} T \cdot V^{\pi} - \sum_{t=1}^{T} V^{\pi_t}\right] = O(\sqrt{|\Pi| T \log T} + |\Pi| \log T).$$

There are two challenges in leveraging Theorem 3.2. The first challenge is crucial: Notice that Theorem 3.2 does not imply that Algorithm 2 has a low regret; its only guarantee is w.r.t. the policies in Π received as an input. As the number of policies is infinite, our success will depend on our ability to characterize a "good" set of policies Π. The second challenge is technical: Even if we find such Π, we still need to characterize the associated $\tilde{\tau}$ and η. This is precisely what we do in Section 4 and 5.

4 Single User Type

In this section, we focus on the special case of a single user type, i.e., $M = 1$. For notational convenience, since we only discuss single-type users, we associate each category $a \in [K]$ with its two unique parameters $\mathbf{P}_a := \mathbf{P}_{a,1}, \mathbf{\Lambda}_a := \mathbf{\Lambda}_{a,1}$ and refer to them as scalars rather than vectors. In addition, We use the notation N_a for the random variable representing the number of iterations until a random user departs after being recommended by π^a, the fixed-arm policy that recommends category a in each iteration.

To derive a regret bound for single-type users, we use two main lemmas: Lemma 4.1, which shows the optimal policy is fixed, and Lemma 4.3, which shows that returns of fixed-arm policies are sub-exponential and calculate their corresponding parameters. These lemmas allow us to use Algorithm 2 with a policy set Π that contains all the fixed-arm policies, and derive a $\tilde{O}(\sqrt{T})$ regret bound. All omitted proofs can be found in the full version of this paper (Ben-Porat et al. 2022).

To show that there exists a category $a^* \in [K]$ for which π^{a^*} is optimal, we rely on the assumption that all the users have the same type (hence we drop the type subscripts t), and as a result the rewards of each category $a \in [K]$ have an expectation that depends on a single parameter, namely $\mathbb{E}[r(a)] = \mathbf{P}_a$. Such a category $a^* \in [K]$ does not necessarily have the maximal click-probability nor the minimal departure-probability, but rather an optimal combination of the two (in a way, this is similar to the knapsack problem, where we want to maximize the reward while having as little weight as possible). We formalize it in the following lemma.

Lemma 4.1. *A policy π^{a^*} is optimal if*

$$a^* \in \arg\max_{a \in [K]} \frac{\mathbf{P}_a}{\mathbf{\Lambda}_a(1 - \mathbf{P}_a)}.$$

As a consequence of this lemma, the planning problem for single-type users is trivial—the solution is a fixed-arm policy π^{a^*} given in the lemma. However, without access to the model parameters, identifying π^{a^*} requires learning. We proceed with a simple observation regarding the random number of iterations obtained by executing a fixed-arm policy. The observation would later help us show that the return of any fixed-arm policy is sub-exponential.

Observation 4.2. *For every $a \in [K]$ and every $\mathbf{\Lambda}_a > 0$, the random variable N_a follows a geometric distribution with success probability parameter $\mathbf{\Lambda}_a[1 - \mathbf{P}_a] \in (0, 1-\epsilon]$.*

Using Observation 4.2 and previously known results (stated in the full version of this paper (Ben-Porat et al. 2022)), we show that N_a is sub-exponential for all $a \in [K]$. Notice that return realizations are always upper bounded by the trajectory length; this implies that returns are also sub-exponential. However, to use the regret bound of Algorithm 2, we need information regarding the parameters (τ_a^2, b_a) for every policy π^a. We provide this information in the following Lemma 4.3.

Lemma 4.3. *For each category $a \in [K]$, the centred random variable $V^{\pi^a} - \mathbb{E}[V^{\pi^a}]$ is sub-exponential with parameters (τ_a^2, b_a), such that*

$$\tau_a = b_a = -\frac{8e}{\ln(1 - \mathbf{\Lambda}_a(1 - \mathbf{P}_a))}.$$

Proof sketch. We rely on the equivalence between the subexponentiality of a random variable and the bounds on its moments (Property 2 in Definition 3.1). We bound the expectation of the return V^{π^a}, and use Minkowski's and Jensen's inequalities to show in the full version (Ben-Porat et al. 2022) that $\mathbb{E}[|V^{\pi^a} - \mathbb{E}[V^{\pi^a}]|^p])^{1/p}$ is upper bounded by $-4/\ln(1 - \mathbf{\Lambda}_a(1 - \mathbf{P}_a))$ for every $a \in [K]$ and $p \geq 1$. Finally, we apply a normalization trick and bound the Taylor series of $\mathbb{E}[\exp(\gamma(V^{\pi^a} - \mathbb{E}[V^{\pi^a}]))]$ to obtain the result. $\square$

An immediate consequence of Lemma 4.3 is that the parameters $\tilde{\tau} = 8e/\ln(\frac{1}{1-\epsilon})$ and $\eta = 1$ are valid upper bounds for τ_a and b_a/τ_a^2 for each $a \in [K]$ (I.e., $\forall a \in [K] : \tilde{\tau} \geq \tau_a$ and $\eta \geq b_a^2/\tau_a^2$). We can now derive a regret bound using Algorithm 2 and Theorem 3.2.

Theorem 4.4. *For single-type users ($M = 1$), running Algorithm 2 with $\Pi = \{\pi^a : a \in [K]\}$ and $\tilde{\tau} = \frac{8e}{\ln(\frac{1}{1-\epsilon})}$, $\eta = 1$ achieves an expected regret of at most*

$$\mathbb{E}[R_T] = O(\sqrt{KT \log T} + K \log T).$$

5 Two User Types and Two Categories

In this section, we consider cases with two user types ($M = 2$), two categories ($K = 2$) and departure-probability $\mathbf{\Lambda}_{a,\tau} = 1$ for every category $a \in [K]$ and type $\tau \in [M]$. Even in this relatively simplified setting, where users leave after the first "no-click", planning is essential. To see this, notice that the event of a user clicking on a certain category provides additional information about the user, which can be used to tailor better recommendations; hence, algorithms that do not take this into account may suffer a linear regret. In fact, this is not just a matter of the learning algorithm at hand, but rather a failure of all fixed-arm policies; there are instances where all fixed-arm policies yield high regret w.r.t. the baseline defined in Equation (1). Indeed, this is what the example in Section 2.1 showcases. Such an observation suggests that studying the optimal planning problem is vital.

In Section 5.1, we introduce the partially observable MDP formulation of departing bandits along with notion of *belief-category walk*. We use this notion to provide a closed-form formula for policies' expected return, which we use extensively later on. Next, in Section 5.2 we characterize the optimal policy, and show that we can compute it in constant time relying on the closed-form formula. This is striking, as generally computing optimal POMDP policies is computationally intractable since, e.g., the space of policies grows exponentially with the horizon. Conceptually, we show that there exists an optimal policy that depends on a belief threshold: It recommends one category until the posterior belief of one type, which is monotonically increasing, crosses the threshold, and then it recommends the other category. Finally, in Section 5.3 we leverage all the previously obtained results to derive a small set of threshold policies of size $O(\ln T)$ with corresponding sub-exponential parameters. Due to Theorem 3.2, this result implies a $\tilde{O}(\sqrt{T})$ regret.

5.1 Efficient Planning

To recap, we aim to find the optimal policy when the click-probability matrix and the prior over user types are known. Namely, given an instance in the form of $\langle \mathbf{P}, \mathbf{q} \rangle$, our goal is to efficiently find the optimal policy.

For planning purposes, the problem can be modeled by an episodic POMDP, $\langle S, [K], O, \text{Tr}, \mathbf{P}, \Omega, \mathbf{q}, O \rangle$. A set of states, $S = [M] \cup \{\perp\}$ that comprises all types $[M]$, along with a designated absorbing state $\perp$ suggesting that the user departed (and the episode terminated). $[K]$ is the set of the actions (categories). $O = \{stay, depart\}$ is the set of possible observations. The transition and observation functions, $\text{Tr} : S \times [K] \to S$ and $\Omega : S \times [K] \to O$ (respectively) satisfy $\text{Tr}(\perp | i, a) = \Omega(depart | i, a) = 1 - \mathbf{P}_{i,a}$ and $\text{Tr}(i | i, a) = \Omega(stay | i, a) = \mathbf{P}_{i,a}$ for every type $i \in [M]$ and action $a \in [K]$. Finally, $\mathbf{P}$ is the expected reward matrix, and $\mathbf{q}$ is the initial state distribution over the M types.

When there are two user types and two categories, the click-probability matrix is given by Table 2 where we note that the prior on the types holds $\mathbf{q}_y = 1 - \mathbf{q}_x$, thus can be represented by a single parameter $\mathbf{q}_x$.

Remark 5.1. *Without loss of generality, we assume that $\mathbf{P}_{1,x} \geq \mathbf{P}_{2,x}, \mathbf{P}_{1,y}, \mathbf{P}_{2,y}$ since one could always permute the matrix to obtain such a structure.*

Since the return and number of iterations for the same policy is independent of the user index, we drop the subscript t in the rest of this subsection and use .

	Type x	Type y
Category 1	$\mathbf{P}_{1,x}$	$\mathbf{P}_{1,y}$
Category 2	$\mathbf{P}_{2,x}$	$\mathbf{P}_{2,y}$
Prior	$\mathbf{q}_x$	$\mathbf{q}_y = 1 - \mathbf{q}_x$

Table 2: Click probabilities for two user types and two categories.

As is well-known in the POMDP literature (Kaelbling, Littman, and Cassandra 1998), the optimal policy π^* and its expected return are functions of belief states that represent the probability of the state at each time. In our setting, the states are the user types. We denote by b_j the belief that the state is (type) x at iteration j. Similarly, $1 - b_j$ is the belief that the state is (type) y at iteration j. Needless to say, once the state $\perp$ is reached, the belief over the type states $[M]$ is irrelevant, as users do not come back. Nevertheless, we neglect this case as our analysis does not make use it.

We now describe how to compute the belief. At iteration $j = 1$, the belief state is set to be $b_1 = \mathbb{P}(state = x) = \mathbf{q}_x$. At iteration $j > 1$, upon receiving a positive reward $r_j = 1$, the belief is updated from $b_{j-1} \in [0,1]$ to

$$b_j(b_{j-1}, a, 1) = \frac{b_{j-1} \cdot \mathbf{P}_{a,x}}{b_{j-1} \cdot \mathbf{P}_{a,x} + \mathbf{P}_{a,y}(1 - b_{j-1})}, \qquad (2)$$

where we note that in the event of no-click, the current user departs the system, i.e., we move to the absorbing state $\perp$. For any policy $\pi : [0,1] \to \{1,2\}$ that maps a belief to a category, its expected return satisfies the Bellman equation,

$$\mathbb{E}[V^\pi(b)] = \Big(b\mathbf{P}_{\pi(b),x} + (1-b)\mathbf{P}_{\pi(b),y}\Big) \cdot (1 + \mathbb{E}[V^\pi(b'(b, \pi(b), 1))]).$$

To better characterize the expected return, we introduce the following notion of belief-category walk.

Definition 5.2 (Belief-category walk)**.** *Let $\pi : [0,1] \to \{1,2\}$ be any policy. The sequence*

$$b_1, a_1 = \pi(b_1), b_2, a_2 = \pi(b_2), \ldots$$

is called the belief-category walk. Namely, it is the induced walk of belief updates and categories chosen by π, given all the rewards are positive ($r_j = 1$ for every $j \in \mathbb{N}$).

Notice that every policy induces a single, well-defined and deterministic belief-category walk (recall that we assume departure-probabilities satisfy $\mathbf{\Lambda}_{a,\tau} = 1$ for every $a \in [K], \tau \in [M]$). Moreover, given any policy π, the trajectory of every user recommended by π is fully characterized by belief-category walk clipped at $b_{N^\pi(t)}, a_{N^\pi(t)}$.

In what follows, we derive a closed-form expression for the expected return as a function of b, the categories chosen by the policy, and the click-probability matrix.

Theorem 5.3. *For every policy π and an initial belief $b \in [0,1]$, the expected return is given by*

$$\mathbb{E}[V^\pi(b)] = \sum_{i=1}^{\infty} b \cdot \mathbf{P}_{1,x}^{m_{1,i}} \cdot \mathbf{P}_{2,x}^{m_{2,i}} + (1-b)\mathbf{P}_{1,y}^{m_{1,i}} \cdot \mathbf{P}_{2,y}^{m_{2,i}},$$

where $m_{1,i} := |\{a_j = 1, j \le i\}|$ and $m_{2,i} := |\{a_j = 2, j \le i\}|$ are calculated based on the belief-category walk $b_1, a_1, b_2, a_2, \ldots$ induced by π.

5.2 Characterizing the Optimal Policy

Using Theorem 5.3, we show that the planning problem can be solved in $O(1)$. To arrive at this conclusion, we perform a case analysis over the following three structures of the click-probability matrix $\mathbf{P}$:

- *Dominant Row*, where $\mathbf{P}_{1,y} \ge \mathbf{P}_{2,y}$;
- *Dominant Column*, where $\mathbf{P}_{2,x} \ge \mathbf{P}_{2,y} > \mathbf{P}_{1,y}$;
- *Dominant Diagonal*, where $\mathbf{P}_{1,x} \ge \mathbf{P}_{2,y} > \mathbf{P}_{1,y}, \mathbf{P}_{2,x}$.

Crucially, any matrix $\mathbf{P}$ takes exactly one of the three structures. Further, since $\mathbf{P}$ is known in the planning problem, identifying the structure at hand takes $O(1)$ time. Using this structure partition, we characterize the optimal policy.

Dominant Row We start by considering the simplest structure, in which the Category 1 is preferred by both types of users: Since $\mathbf{P}_{1,y} \ge \mathbf{P}_{2,y}$ and $\mathbf{P}_{1,x} \ge \mathbf{P}_{2,x}, \mathbf{P}_{1,y}, \mathbf{P}_{2,y}$ (Remark 5.1), there exists a dominant row, i.e., Category 1.

Lemma 5.4. *For any instance such that $\mathbf{P}$ has a dominant row a, the fixed policy π^a is an optimal policy.*

As expected, if Category 1 is dominant then the policy that always recommends Category 1 is optimal.

Dominant Column In the second structure we consider the case where there is no dominant row, and that the column of type x is dominant, i.e., $\mathbf{P}_{1,x} \ge \mathbf{P}_{2,x} \ge \mathbf{P}_{2,y} > \mathbf{P}_{1,y}$. In such a case, which is also the one described in the example in Section 2.1, it is unclear what the optimal policy would be since none of the categories dominates the other.

Surprisingly, we show that the optimal policy can be of only one form: Recommend Category 2 for some time steps (possibly zero) and then always recommend Category 1. To identify when to switch from Category 2 to Category 1, one only needs to compare four expected returns.

Theorem 5.5. *For any instance such that $\mathbf{P}$ has a dominant column, one of the following four policies is optimal:*

$$\pi^1, \pi^2, \pi^{2:\lfloor N^* \rfloor}, \pi^{2:\lceil N^* \rceil},$$

where $N^* = N^*(\mathbf{P}, \mathbf{q})$ is a constant, and $\pi^{2:\lfloor N^* \rfloor}$ ($\pi^{2:\lceil N^* \rceil}$) stands for recommending Category 2 until iteration $\lfloor N^* \rfloor$ ($\lceil N^* \rceil$) and then switching to Category 1.

The intuition behind the theorem is as follows. If the prior tends towards type y, we might start with recommending Category 2 (which users of type y are more likely to click on). But after several iterations, and as long as the user stays, the posterior belief b increases since $\mathbf{P}_{2,x} > \mathbf{P}_{2,y}$ (recall Equation (2)). Consequently, since type x becomes more probable, and since $\mathbf{P}_{1,x} \ge \mathbf{P}_{2,x}$, the optimal policy recommends the best category for this type, i.e., Category 1. For the exact expression of N^*, we refer the reader to the full version of the paper (Ben-Porat et al. 2022).

Using Theorem 5.3, we can compute the expected return for each of the four policies in $O(1)$, showing that we can find the optimal policy when $\mathbf{P}$ has a column in $O(1)$.

Dominant Diagonal In the last structure, we consider the case where there is no dominant row (i.e., $\mathbf{P}_{2,y} > \mathbf{P}_{1,y}$) nor a dominant column (i.e., $\mathbf{P}_{2,y} > \mathbf{P}_{2,x}$). At first glance, this case is more complex than the previous two, since none of the categories and none of the types dominates the other one. However, we uncover that the optimal policy can be either always recommending Category 1 or always recommending Category 2. Theorem 5.6 summarizes this result.

Theorem 5.6. *For any instance such that $\mathbf{P}$ has a dominant diagonal, either π^1 or π^2 is optimal.*

With the full characterization of the optimal policy derived in this section (for all the three structures), we have shown that the optimal policy can be computed in $O(1)$.

5.3 Learning: UCB-based Regret Bound

In this section, we move from the planning task to the learning one. Building on the results of previous sections, we know that there must exist a threshold policy—a policy whose belief-category walk has a finite prefix of one category, and an infinite suffix with the other category—which is optimal. However, there can still be infinitely many such policies. To address this problem, we first show how to reduce the search space for approximately optimal policies with negligible additive factor to a set of $|\Pi| = O(\ln(T))$ policies. Then, we derive the parameters $\tilde{\tau}$ and η required for Algorithm 2. As an immediate consequence, we get a sublinear regret algorithm for this setting. We begin with defining threshold policies.

Definition 5.7 (Threshold Policy)**.** *A policy π is called an (a, h)-threshold policy if there exists an number $h \in \mathbb{N} \cup \{0\}$ in π's belief-category walk such that*

- *π recommends category a in iterations $j \leq h$, and*
- *π recommends category a' in iterations $j > h$,*

for $a, a' \in \{1, 2\}$ and $a \neq a'$.

For instance, the policy π^1 that always recommends Category 1 is the $(2, 0)$-threshold policy, as it recommends Category 2 until the zero'th iteration (i.e., never recommends Category 2) and then Category 1 eternally. Furthermore, the policy $\pi^{2:\lfloor N^* \rfloor}$ introduced in Theorem 5.5 is the $(2, \lfloor N^* \rfloor)$-threshold policy.

Next, recall that the chance of departure in every iteration is greater or equal to ϵ, since we assume $\max_{a,\tau} \mathbf{P}_{a,\tau} \leq 1-\epsilon$. Consequently, the probability that a user will stay beyond H iterations is exponentially decreasing with H. We could use high-probability arguments to claim that it suffices to focus on the first H iterations, but without further insights this would yield $\Omega(2^H)$ candidates for the optimal policy. Instead, we exploit our insights about threshold policies.

Let Π_H be the set of all (a, h)-threshold policies for $a \in \{1, 2\}$ and $h \in [H] \cup \{0\}$. Clearly, $|\Pi_H| = 2H + 2$. Lemma 5.8 shows that the return obtained by the best policy in Π_H is not worse than that of the optimal policy π^* by a negligible factor.

Lemma 5.8. *For every $H \in \mathbb{N}$, it holds that*

$$\mathbb{E}\left[V^{\pi^*} - \max_{\pi \in \Pi_H} V^{\pi}\right] \leq \frac{1}{2^{O(H)}}.$$

Before we describe how to apply Algorithm 2, we need to show that returns of all the policies in Π_H are sub-exponential. In Lemma 5.9, we show that V^{π} is (τ^2, b)-sub-exponential for every threshold policy $\pi \in \Pi_H$, and provide bounds for both τ and b^2/τ^2.

Lemma 5.9. *Let $\tilde{\tau} = \frac{8e}{\ln(\frac{1}{1-\epsilon})}$ and $\eta = 1$. For every threshold policy $\pi \in \Pi_H$, the centred random variable $V^{\pi} - \mathbb{E}[V^{\pi}]$ is (τ^2, b)-sub-exponential with (τ^2, b) satisfying $\tilde{\tau} \geq \tau$ and $\eta \geq b^2/\tau^2$.*

We are ready to wrap up our solution for the learning task proposed in this section. Let $H = \Theta(\ln T)$, Π_H be the set of threshold policies characterized before, and let $\tilde{\tau}$ and η be constants as defined in Lemma 5.9.

Theorem 5.10. *Applying Algorithm 2 with $\Pi_H, T, \tilde{\tau}, \eta$ on the class of two-types two-categories instances considered in this section always yields an expected regret of*

$$\mathbb{E}[R_T] \leq O(\sqrt{T} \ln T).$$

Proof. It holds that

$$\begin{aligned}
\mathbb{E}[R_T] &= \mathbb{E}\left[TV^{\pi^*} - \sum_{t=1}^{T} V^{\pi_t}\right] \\
&= \mathbb{E}\left[TV^{\pi^*} - \max_{\pi \in \Pi_H} TV^{\pi}\right] + \mathbb{E}\left[\max_{\pi \in \Pi_H} TV^{\pi} - \sum_{t=1}^{T} V^{\pi_t}\right] \\
&\leq \frac{T}{2^{O(H)}} + O(\sqrt{HT \log T} + H \log T) = O(\sqrt{T} \ln T),
\end{aligned}$$

where the inequality follows from Theorem 3.2 and Lemma 5.8. Finally, setting $H = \Theta(\ln T)$ yields the desired result. □

6 Conclusions and Discussion

This paper introduces a MAB model in which the recommender system influences both the rewards accrued *and* the length of interaction. We dealt with two classes of problems: A single user type with general departure probabilities (Section 4) and the two user types, two categories where each user departs after her first no-click (Section 5). For each problem class, we started with analyzing the planning task, then characterized a small set of candidates for the optimal policy, and then applied Algorithm 2 to achieve sublinear regret.

In the full version (Ben-Porat et al. 2022), we also consider a third class of problems: Two categories, multiple user types ($M \geq 2$) where user departs with their first no-click. We use the closed-form expected return derived in Theorem 5.3 to show how to use dynamic programming to find approximately optimal planning policies. We formulate the problem of finding an optimal policy for a finite horizon H in a recursive manner. Particularly, we show how to find a $1/2^{O(H)}$ additive approximation in run-time of $O(H^2)$. Unfortunately, this approach cannot assist us in the learning task. Dynamic programming relies on skipping sub-optimal solutions to subproblems (shorter horizons in our case), but this happens on the fly; thus, we cannot a-priori define a small set of candidates like what Algorithm 2 requires. More broadly, we could use this dynamic programming approach for more than two categories, namely for $K \geq 2$, but then the run-time becomes $O(H^K)$.

There are several interesting future directions. First, achieving low regret for the setup in Section 5 with $K \geq 2$. We suspect that this class of problems could enjoy a solution similar to ours, where candidates for optimal policies are mixing two categories solely. Second, achieving low regret for the setup in Section 5 with uncertain departure (i.e., $\mathbf{\Lambda} \neq 1$). Our approach fails in such a case since we cannot use belief-category walks; these are no longer deterministic. Consequently, the closed-form formula is much more complex and optimal planning becomes more intricate. These two challenges are left open for future work.

Acknowledgements

LL is generously supported by an Open Philanthropy AI Fellowship. LC is supported by Ariane de Rothschild Women Doctoral Program. ZL thanks the Block Center for Technology and Society; Amazon AI; PwC USA via the Digital Transformation and Innovation Center; and the NSF: Fair AI Award IIS2040929 for supporting ACMI lab's research on the responsible use of machine learning. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 882396), by the Israel Science Foundation (grant number 993/17), Tel Aviv University Center for AI and Data Science (TAD), and the Yandex Initiative for Machine Learning at Tel Aviv University.

References

Abbasi-Yadkori, Y.; Pál, D.; and Szepesvári, C. 2011. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24: 2312–2320.

Azar, M. G.; Lazaric, A.; and Brunskill, E. 2013. Sequential transfer in multi-armed bandit with finite set of models. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, 2220–2228.

Bahar, G.; Ben-Porat, O.; Leyton-Brown, K.; and Tennenholtz, M. 2020. Fiduciary bandits. In *International Conference on Machine Learning*, 518–527. PMLR.

Bahar, G.; Smorodinsky, R.; and Tennenholtz, M. 2016. Economic Recommendation Systems: One Page Abstract. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, 757–757. New York, NY, USA: ACM. ISBN 978-1-4503-3936-0.

Ben-Porat, O.; Cohen, L.; Leqi, L.; Lipton, Z. C.; and Mansour, Y. 2022. Modeling Attrition in Recommender Systems with Departing Bandits. *arXiv preprint arXiv:2203.13423*.

Bubeck, S.; Cesa-Bianchi, N.; et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1): 1–122.

Cao, J.; Sun, W.; Shen, Z.-J. M.; and Ettl, M. 2020. Fatigue-Aware Bandits for Dependent Click Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3341–3348.

Cesa-Bianchi, N.; and Lugosi, G. 2006. *Prediction, learning, and games*. Cambridge Univ Press.

Cohen, L.; and Mansour, Y. 2019. Optimal Algorithm for Bayesian Incentive-Compatible. In *ACM Conf. on Economics and Computation (EC)*.

Eldar, Y.; and Kutyniok, G. 2012. *Compressed Sensing: Theory and Applications*. ISBN 978-1107005587.

Jia, H.; Shi, C.; and Shen, S. 2021. Multi-armed Bandit with Sub-exponential Rewards. *Operations Research Letters*.

Joseph, M.; Kearns, M.; Morgenstern, J.; and Roth, A. 2016. Fairness in learning: Classic and contextual bandits. *arXiv preprint arXiv:1605.07139*.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134.

Korda, N.; Szörényi, B.; and Li, S. 2016. Distributed Clustering of Linear Bandits in Peer to Peer Networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*.

Kremer, I.; Mansour, Y.; and Perry, M. 2014. Implementing the wisdom of the crowd. *Journal of Political Economy*, 122: 988–1012.

Lattimore, T.; and Szepesvári, C. 2020. *Bandit algorithms*. Cambridge University Press.

Leqi, L.; Kilinc-Karzan, F.; Lipton, Z. C.; and Montgomery, A. L. 2020. Rebounding Bandits for Modeling Satiation Effects. *arXiv preprint arXiv:2011.06741*.

Liu, Y.; and Ho, C.-J. 2018. Incentivizing high quality user contributions: New arm generation in bandit learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Lu, Z.; and Yang, Q. 2016. Partially Observable Markov Decision Process for Recommender Systems. *CoRR*, abs/1608.07793.

Mahadik, K.; Wu, Q.; Li, S.; and Sabne, A. 2020. *Fast Distributed Bandits for Online Recommendation Systems*.

Mansour, Y.; Slivkins, A.; and Syrgkanis, V. 2015. Bayesian Incentive-Compatible Bandit Exploration. In *ACM Conf. on Economics and Computation (EC)*.

Patil, V.; Ghalme, G.; Nair, V.; and Narahari, Y. 2020. Achieving fairness in the stochastic multi-armed bandit problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5379–5386.

Pike-Burke, C.; and Grünewälder, S. 2019. Recovering bandits. *arXiv preprint arXiv:1910.14354*.

Ron, T.; Ben-Porat, O.; and Shalit, U. 2021. Corporate Social Responsibility via Multi-Armed Bandits. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 26–40.

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An MDP-Based Recommender System. *Journal of Machine Learning Research*, 6(43): 1265–1295.

Slivkins, A. 2019. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*.

Zhao, X.; Zheng, X.; Yang, X.; Liu, X.; and Tang, J. 2020a. Jointly Learning to Recommend and Advertise. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Zhao, Y.; Zhou, Y.; Ou, M.; Xu, H.; and Li, N. 2020b. Maximizing Cumulative User Engagement in Sequential Recommendation: An Online Optimization Perspective. In Gupta, R.; Liu, Y.; Tang, J.; and Prakash, B. A., eds., *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.