RESEARCH ARTICLE

When proxy-driven learning is no better than random: The consequences of representational incompleteness

Justin Zobel[1]*, **Felisa J. Vázquez-Abad**[1,2], **Pauline Lin**[1]

1 School of Computing & Information Systems, the University of Melbourne, Parkville, Victoria, Australia, **2** Department of Computer Science, City University of New York, New York, New York, United States of America

* jzobel@unimelb.edu.au

OPEN ACCESS

Citation: Zobel J, Vázquez-Abad FJ, Lin P (2022) When proxy-driven learning is no better than random: The consequences of representational incompleteness. PLoS ONE 17(7): e0271268. https://doi.org/10.1371/journal.pone.0271268

Editor: Quanquan Gu, UCLA, UNITED STATES

Received: November 8, 2021

Accepted: June 28, 2022

Published: July 13, 2022

Data Availability Statement: All data and code is held in a public repository, https://github.com/paperauthors/PLoS2022.git. This repository includes the minimal data set of all results used to generate figures and draw conclusions in the paper.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Machine learning is widely used for personalisation, that is, to tune systems with the aim of adapting their behaviour to the responses of humans. This tuning relies on quantified features that capture the human actions, and also on objective functions—that is, proxies – that are intended to represent desirable outcomes. However, a learning system's representation of the world can be incomplete or insufficiently rich, for example if users' decisions are based on properties of which the system is unaware. Moreover, the incompleteness of proxies can be argued to be an intrinsic property of computational systems, as they are based on literal representations of human actions rather than on the human actions themselves; this problem is distinct from the usual aspects of bias that are examined in machine learning literature. We use mathematical analysis and simulations of a reinforcement-learning case study to demonstrate that incompleteness of representation can, first, lead to learning that is no better than random; and second, means that the learning system can be inherently unaware that it is failing. This result has implications for the limits and applications of machine learning systems in human domains.

1 Introduction

A wide range of computational activities are performed by systems that are based on machine learning. Some of these activities are designed to adapt or respond to human actions using reinforcement learning, thus modifying their behaviour to improve the experience perceived by the humans with whom they are interacting. Examples include email spam filters, search engines, and recommender systems. Some systems of this kind respond in the same way to all users, producing results aggregated across all interactions, while others are *personalised*, that is, they adapt to the behaviour of individuals.

Such systems must (obviously) operate within the constraints of a computational system. That is, they can only make use of the input with which they are provided; they must reduce human actions and real-world entities to abstract, literal representations; and they must use proxies to represent the desired outcome, which at the human level might be 'happiness'–or *satisfaction*, the term we use in this work.

A computational proxy must be a quantified metric that is designed to correspond with satisfaction. Thus, for example, in web search a simple proxy for satisfaction might be 'the proportion of times that the user clicks on the first item in the results list', reflecting the intuition that a user is happy if the first answer appears to contain the information being sought. Actual user responses might be more diverse; for example, the user might not click at all if the document summary on the results pages result contains the required information.

The proxy could then be refined, using mouse hovers for example, but there will always be a gap between the computational proxy and the richness of human behaviours. That is, a click or a hover do not demonstrate satisfaction; these actions correlate with satisfaction but are not equivalent. Likewise, categories of newspaper article—such as politics, sport, world, or culture—are convenient and helpful, but can easily not align with the much richer features that attract user interest. We argue that it is inevitable that a proxy for such applications will not be perfectly aligned with human goals.

It is this gap that we describe as *representational incompleteness*. This problem is distinct from the kinds of *bias* that are commonly examined in machine learning, which concern issues such as poorly chosen training data, incorrect labelling, missing data, or unbalanced categories [1]; and is also distinct from other problems that arise in the more specific context of reinforcement learning, such as the echo chamber effect. In systems whose behaviour is driven by interaction, we show that proxy mismatch or mismatch between features known to the system and those of interest to the user [2]—that is, representational incompleteness–can lead to poor outcomes even if the usual forms of bias are absent.

1.1 Contribution

In this paper, we explore representational incompleteness in the context of personalisation based on reinforcement learning, by examining behaviour where the representation is incomplete or the proxy does not fully capture human goals. To our knowledge this is the first exploration of the implications of this innate limitation of learning systems. Our work studies a core element of reinforcement learning, via specific scenarios in which misinterpretation of user feedback misleads the system. Due to the fact that human behaviour and real-world entities are rich and not amenable to complete description, some human behaviour will not be fully captured and some proxies will be inadequate, so that some level of misinterpretation is inevitable. We argue that a consequence of these factors is that the algorithm cannot discover that it is not performing well, and that this is an inherent challenge across a wide range of applications of machine learning, even for systems that are believed to work well.

We again note that these issues are distinct from challenges such as fairness, insufficiency, or errancy of the data, which also lead to failures of learning but through different mechanisms and with different effects. Moreover, while it may be common knowledge that computational proxies are often incomplete and, for applications of machine learning that are not based on interaction, that the existence of incompleteness may limit the effectiveness of the methods, in such case the learning is still better than random. In contrast, as we show, there may be no improvement compared to random in the environment that we study.

As a case study we use a movie recommendation system that is similar to systems that are regarded as successful in other literature. Recommenders are systems in which the set of features (the representation) over which the learning takes place is limited to literal features; movies are typically described by a category and perhaps additional metadata such as year, cast, or production crew, while human preferences may take into account a much broader range of values. In such systems, the user may be unaware that the system is algorithmically learning from their behaviour.

Note, however, that we do not seek to contribute to the literature on effective recommendation—the use of these systems in our work is purely as a tool for exploring a more general principle. Note also that we do not inspect every form of recommender or every possible adjustment that could be made to rectify specific errors; the issues we highlight may take different forms in different scenarios, but our point is that the issues can arise, not the specific forms that they take.

We use a rigorous mathematical argument, supported by simulations, to show that a plausible system can produce poor results in response to reasonable sequences of human actions, as an inherent consequence of the fact that there are human motivations of which the system is unaware. We examine two cases: first, where the user's behaviour is reasonable but has not been anticipated in the system design; second, where the user's goal is not reflected in the proxy due to the system having incomplete semantics. These examples are used and argued for in recommendation systems literature. However, as we show, the chosen proxy can lead to severe performance issues. Representations of human semantics are by their nature incomplete, and thus, we argue, instances of such situations are inevitable.

An intuitive assessment could be that an ML system with an insufficiently rich proxy might in the worst case tend to be random, and indeed we show this to be case. Under reasonable assumptions for proxies and completeness of data representation, and even in the absence of bias, the performance of the system can systematically be random, or in extreme cases even worse.

It could be argued that this observation is obvious, but a key insight here is that the problem is inherent when there is a mismatch between a system's representations and the richness of reality. Addition of features cannot in general resolve the difficulties: if there is a human element of which the system is unaware, its proxies can tend to indicate that it is doing well when in fact the representational gap is creating gross error. This outcome can occur even if the human is not intentionally acting in an adversarial way. Clearly any specific system limitation, once known to the developers, can be addressed by altering the system; the issue we are highlighting is that it may be impossible to recognise the limitation automatically.

We have not sought to solve this problem, but to demonstrate that the pathology we have identified is undiscoverable (by automatic means) when it occurs. It is obvious that a method with, say, uninformative features can be poor. However, as we show, even methods that have been argued to be strong can fail in the presence of unanticipated behaviour. Our results have implications for any application of machine learning that is designed to adapt to human behaviour and place limits on the claims that can be made for system performance: a hidden assumption is that the representation is sufficiently complete and that the proxy is sufficient, and yet in practice they cannot be. Thus, in some situations, such systems must silently fail.

2 Motivation and background

Machine learning has a range of well-understood limitations. One that has had significant recent attention is bias and fairness. We discuss these two areas below, to set context but also to demonstrate that representational incompleteness is distinct from the forms of bias that have previously had attention. We then review work on recommender systems, which we examine as a case study, again with a focus on bias and fairness.

2.1 Fairness

Suresh and Guttag [3] characterise the issue of fairness as ‘unintended consequences’ of machine learning, a scope that includes our work. Their taxonomy (and their integration of previous taxonomies) rests on an abstract description of machine learning as an aggregation of data collection and preparation followed by model development, evaluation, postprocessing, and deployment. This careful structure provides an effective framework for evaluation of the form and effect of failures; however, their use of the framework is limited to biases, and does not encompass the challenges we consider of incomplete representation of data or misinterpretation of human behaviour. Our work could naturally be included in an extension of Suresh and Guttag’s framework, as could other limitations such as model adaptation or translation, as discussed by Danks and London [4], which we consider further below. Specifically, for example, while several works consider the impact of data labels that are inaccurate, or of unrepresentative sampling of the data population, to our knowledge there has not been previous examination of the impact of data labels that are inadequately rich.

An overview and formal description of fairness in decision making is given by Mitchell et al. [1]. This work sets out the range of factors that can influence or undermine fairness; we now discuss some of these but note that their work illustrates that bias and fairness are closely related.

One of the factors they consider as a confound to fairness is the process for collection of the data, which can be flawed if it is drawn from an unbalanced population; for example, if the desired outcome is understanding of the prevalence of a disease, but the data is drawn from the set of visits to a medical specialist, then prevalence will be overrepresented. Another factor is algorithmic bias, where for example more features might be used for women than for men because women can be pregnant, thus omitting, for men, potential predictive values. Yet another factor is a false assumption that outcomes are independent; for example, treatment resources may be finite, so one person’s hospitalisation can mean that another person is not treated. A further factor is the assumption that a simple or prelimary indicator–a proxy, in our terms—adequately represents a desired outcome; for example, discharge from hospital does not necessarily mean that the individual has fully recovered. These issues arise even if there is no incompleteness in the data representation. (These examples are ours; Mitchell et al. give similar examples drawn from other domains.) This work is discussed further below.

Joseph et al. [5], also discussed in more detail below, demonstrate grounds on which a specific class of algorithm can be provably fair, within the constraints of a system that can ‘know’ whether the desired outcome has in fact been achieved, an issue they do not examine, and which constitutes a more fundamental limitation in that it relates to the question of how completely a computational system can interact with human behaviour.

2.2 Bias

Danks and London [4] examine bias from a principled perspective, providing a taxonomy of causes of algorithmic bias. It can arise from choice of training data, incorrect use of attributes or annotations, algorithmic failure, inappropriate generalisation, or misinterpretation of outcomes by the user. In earlier work, Calders and Žliobaitė [6] reviewed how unbiased procedures can have biased outcomes if there is a mismatch between data, labelling, and assumptions being made in the system development process.

These taxonomies provide a valuable framework for examination of the appropriateness of ML for specific tasks, though from our perspective they are incomplete. As a reflection on these codifications of bias, it is apparent that there isn’t universal agreement on definitions of bias or when it is of concern.

Additional forms of (undesirable) bias in machine learning arise when the underlying data contains errors or omissions; is insufficient in volume; or is poorly sampled, that is, some classes are overrepresented and others absent or inadequate, as noted by Mitchell et al. [1] in the context of fairness. Some forms are particular to reinforcement learning, for example, when system behaviour adapts to the users who choose to continue with the system but does not recognise the shortcomings that led to other users to abandon it, or learn how to better interact with these users. These are fundamental challenges in machine learning and outside the scope of this paper.

Of more direct relevance to our work is that of Haug et al. [2], who consider the behaviour of learning-by-feedback when there is a mismatch between the features captured in the system and those of interest to the user. The focus is a specific application of learning in which a system is being explicitly trained by an individual to perform well at a defined task. They demonstrate that in such contexts, the reward function—proxy, in our terminology—may not align with the 'true' reward being sought, and propose ways in which a system can be designed to allow greater flexibility in training once a failing is discovered. The contrast with our work is that, in our analysis, we do not assume that the user is aware that learning is taking or place or has an opportunity to modify the learner's behaviour, while their work assumes that there is a mechanism for observing that a failure has occurred. That is, the generality of the problem of failure and its discovery is not examined.

In a similar vein, Wang et al. [7], proceeding from the observation that failure can occur due to mismatch, consider how communication with the user can be designed to allow the failure to be efficiently rectified. The underlying challenge of whether failure is discoverable is not examined.

2.3 Recommender systems

The kind of tool based on machine learning that we examine, as a case study, is recommender systems. These are designed to learn users' preferences in order to *serve* a list of items for the user to choose from. This mechanism is designed to satisfy each user. However, this innately human outcome is not directly observable by the system, so a proxy must be used instead.

As an illustration, suppose that the server is an online provider of video entertainment. In this illustration, the items being served belong to various *categories* of movies; only rarely will a human consume the same item more than once, and thus it is preferable to always serve new items. The set $\mathcal{A}$ represents a finite but large set of categories of movies. We assume that there are always new (unseen) movies in each category $a \in \mathcal{A}$.

A comprehensive survey of recommender systems [8] uses four state-of-the-art algorithms that are statistical methods for static analysis (supervised learning), though reinforcement learning algorithms, which we study, are not considered. Jannach and Jugovac [9] examine how recommender systems contribute to the environment in which they operate, such as an online business. While this is not the same question as how well the system is working, they contrast proxies with measures such as how the recommender contributes as assessed by value to the business, and examine ways in which measures can fail due to bias or to uncertainty in how data was gathered.

Chaney et al. [10] examine a form of failure that is of relevance to our work, namely how the output of a recommender system (a collaborative filter, in contrast to the single-user modelling that we consider) can alter the user's responses, thus leading to undesirable system behaviour. As in our investigations, recommendation is used as a case study of a broader principle. In Chaney et al., the principle is that the apparent success of a system can be due, not to delivering what a user desires, but instead, in effect, to changing the user. This can be regarded as creating a bias in the user population, but although there are similarities to our work this result does not relate to representational incompleteness.

The failure examined by Chaney et al. is that the system outputs converge to a homogeneous norm—an outcome that is arguably better than the behaviour we observe, that outputs can be random or worse—but still far from desirable. The cause and the effect of the failure is different but the approach they take has elements in common with ours, as they use simulations based on a rich user model of realistic actions. While our approach is by analysis, with the simulations primarily used to illustrate the correctness of our theoretical results, the work leads to similar observations—namely, that gaps in the assumed model in the system leads to unknown, undesirable system behaviours.

The case where a user's interests, or preferences, change over time is studied by Jiang et al. [11], where a single user's behavior is studied through a feedback-loop mathematical model. Specifically, they consider a multi-armed bandit (MAB) problem with multiple choices (arms) selected at each iteration. The user then acts and a 'reward' is provided to the computer in order to update its recommendation. They show that under some reasonable settings the model, a particular case of reinforcement learning, will tend to degenerate and create a 'filter bubble' effect.

In work by Joseph et al. [5] and Mitchell et al. [1], similar models are used to illustrate the effect on hiring policies of demographic categories. The focus of these papers is fairness, and they further use the concept of a context, which is an additional descriptor. In the hiring scenario, categories partition a population using, for example, ethnicity, while the descriptor contains information particular to the job qualification of the applicant. Mitchell et al. argue that categorisation of assumptions and choices can mitigate the problems, but there is ongoing conflation of technical and social contributions to the failings of ML, while also arguing that precise definitions of the challenges remain elusive. (Mitchell et al. make use of and cite a deep body of academic literature and public commentary on fairness in ML, both from a technical and social perspective. This material is for the most part beyond the scope of our work but is a notable resource on the topic.) Implicitly, they identify that assumptions and inherent technical constraints are critical to the question of whether ML can be effective.

In Joseph et al. [5], the MAB is designed to choose the best category, while ensuring that, with high probability, applicants that rank better with respect to descriptors will not be discriminated against. They also use a reinforcement learning (RL) algorithm that is a modification of UCB, with fairness explicitly defined in terms of the context. Fairness and optimization are then balanced using a trade-off cost that includes the context and the perceived reward. That is, the goals of the algorithm are completely specified, or in our terminology the proxy is assumed to be perfectly aligned with the human goals.

3 Case studies of failure to learn

We explore, using theoretical models, cases where reinforcement learning is expected to fail. As an illustration we focus on recommender systems, because they are an example of a general, ubiquitous technology that is well understood, but we do not propose refinements to recommender systems or contrast recommender technologies—this paper is not about improvements in recommendation. Our contribution is to demonstrate that, under realistic assumptions, *in the presence of unknown objectives or incomplete object representations it can be inevitable that first, learning will fail, and second, that the failure is invisible to the system.*

The taxonomy of bias by Danks and London [4] does not include the causes of failure that we examine: incompleteness of descriptions of the data and algorithmic misinterpretation of the user's behaviour. That is, in contrast to the kind of bias-based failures that previous works have focused on, which arise because systems are vulnerable to learning and amplifying human discrimination, the failures we consider arise because computational systems are incomplete as representations of human activity.

We use the same model as Jiang et al. [11] to explore the phenomenon that even with a reasonable proxy, and in the absence of bias, some users may be persistently served items that decrease their satisfaction. This applies under the same design of the algorithm as in Jiang et al, including the 'top-ℓ' policy and the widely used Upper Confidence Bound (UCB) algorithm. We use simulations to illustrate and confirm the theoretical results, but it is the theory that is the core of the contribution.

We structure this study in three parts. In Section 3.1, we introduce a recommender system and a simple user model, and show that it behaves as expected. Our results follow from a class of MAB algorithms called ϵ-greedy, introduced by Sutton et al. [12] and provably convergent to the optimal solution, which we study because they are relatively easy to analyse mathematically. It is our view, however, that the principles behind our results are independent of the particular algorithm and therefore can arguably be extended to more complex models.

In Section 3.2 we extend this simple model to a case where the user does not know the labels for the items they are interested in, and exhibit behaviour that the designers of the system had not anticipated. Our results show that the true behaviour is drastically different from the system's understanding of the behaviour, which is wildly overoptimistic.

In Section 3.3 we explore a case where the representation is incomplete in a simple way. Specifically, we assume that a user may be seeking a category that is not in $\mathcal{A}$. We have discussed the need for a proxy to represent satisfaction, which is intended to be a measure of the quality of the output of a system. The representation used to describe the items is also a proxy. It is inherently incomplete and some human perspectives will inevitably be omitted. An example of representational incompleteness is a hidden or missing category. We show that the hidden category is undiscoverable, and leads to behaviour that is no better than random.

At first glance, our model for hidden or silent categories could be seen as similar to that of Joseph et al. [5], as they examine a characteristic of the bandit's 'arms' that is not originally considered in the model for the categories. However, in their work the context is known in advance and is thus neither hidden nor silent. Incompleteness of the representation is fundamental to our observations, and to our knowledge has not previously been examined in this way as cause of failure in machine learning.

3.1 A simple recommender system

The case study we use throughout this paper is an elementary, but plausible, recommender system. We now present our model for such a system, which learns from a user's preferences to make new suggestions. Here, and throughout the paper, the learning is based on the actions of a single individual rather than on a community of users and the system has no prior knowledge of the individual.

Recommendation items are drawn from a set. Each item in the set belongs to one of a fixed number of *categories*. The recommender system, or *server*, is an algorithm that is designed to choose the best categories for each user. The set of the labels of the categories is denoted by $\mathcal{A} = \{0, \ldots, A-1\}$ and contains $A = \parallel \mathcal{A} \parallel$ categories. The recommendation *strategy* is defined as a policy $\boldsymbol{a}_t = (a_{t,1}, \ldots, a_{t,\ell}), \quad a_{t,k} \in \mathcal{A}, \text{ for } k = 1 \ldots, \ell$. where t is time (or iteration) and the user is being offered a list of ℓ items, where ℓ is constant. Each item m is represented by the category a from which it is drawn. The list of ℓ items served at time contains items (movies), one from each of the ℓ chosen categories. When greater specificity is required, that is, we need to know which items as well as the categories they are drawn from, we denote the actual list by $\boldsymbol{m}_t = (m_{t,1}, \ldots, m_{t,\ell})$; $m_{t,k} \in a_{t,k} fork = 1\ldots, \ell$.

Initially, we consider a model where the strategy aims to choose the 'top-ℓ' categories, ranked according to the user's preference. Once $\boldsymbol{a}_t$ is chosen, the items (for example, movies) are picked at random from each of the chosen categories. The user has an innate function, or behaviour, that represents their interest in each item (or category):

$$\mu(a) = \mathbb{P}(\text{user likes item } m \in a); \quad a \in \mathcal{A}.$$

This function is not known, so the server must estimate it in order to attempt to serve a list that maximises user satisfaction.

Remark: In order to simplify notation, when only the categories rather than the actual items are relevant for the calculations we use the strategy $\boldsymbol{a}_t$ in the analysis. It is assumed that the actual served list $\boldsymbol{m}_t$ is characterized by the strategy $\boldsymbol{a}_t$.

The cumulative reward R_T is defined by

$$R_T = \sum_{t=1}^{T} \sum_{a \in \boldsymbol{a}_t} \mu(a) \tag{1}$$

The formulation of the problem is as follows: find the sequence $\{\boldsymbol{a}_t\}$, $t = 1, 2, \ldots$, that maximizes expected long term satisfaction, that is:

$$S = \max_{\{\boldsymbol{a}_t\}} \left(\lim_{T\to\infty} \frac{R_T}{T} \right) \overset{\text{def}}{=} \max_{\{\boldsymbol{a}_t\}} \left(\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{a \in \boldsymbol{a}_t} \mu(a) \right) \tag{2}$$

The quantity R_T is the *expected cumulative reward* up to time T. The control problem in Eq (2) requires establishment of a policy where the list $\boldsymbol{a}_t$ can only use information known to the server up to time t. Because $\mu(\cdot)$ is not known, the server must learn it as the user provides feedback from the sequence of lists that have been presented to them. As is fundamental in machine learning, the server must use a proxy to learn the user's preferences about the categories. Following Jiang et al. [11], we use clicks as the feedback, and denote as $c_t(a)$ the indicator function, meaning that the user clicks on an item in the served list that belongs to category a. This binary quantity is represented as $c_t(a) = \mathbf{1}_{\{\text{user clicks on item } m \in a\}}$ for $\boldsymbol{m} \in m_t$, where $\mathbf{1}_{\{q\}} = 1$ if q is true, and 0 otherwise (indicator function).

Remark Mathematically, $\{\boldsymbol{a}_t\}$ must be a stochastic process adapted to the filtration of the user's behaviour defined by consecutive clicks on items served, that is, $\boldsymbol{a}_t$ may depend on the history of observations up to time t: $\{c_k; k = 1, \ldots, t\}$. Recall that the server does not know μ so it must estimate it in order to attempt to solve Eq (2). This problem is an example of a MAB problem, where multiple arms can be pulled at each iteration.

3.1.1 Assumptions and optimal policies. In the recent literature on this subject, it is often assumed that the number of clicks is a good proxy for user's satisfaction with each served item $a \in \mathcal{A}$ (see Jiang et al. [11] and references therein). Specifically, in the work of Jiang et al. [11] the server postulates a simple *click model* for the user:

$$\mathbb{P}(c_t(a) = 1) = \mathbb{E}[c_t(a)] = \mu(a), \quad \forall a \in \boldsymbol{a}_t. \tag{3}$$

In this model, the user's liking of a category is described by the number of clicks on that category. That is, the sample average of clicks is a good estimator of the unknown satisfaction measure μ. Let

$$T_t(a) = \sum_{n=1}^{t-1} \mathbf{1}_{\{a \in \boldsymbol{a}_n\}} \tag{4}$$

count the number of times item a has been served before time t.

Lemma 3.1. *Under the assumption described in* Eq (3), *for all $a \in \mathcal{A}$ such that $T_t(a) \to \infty$ with probability 1 (w.p.1) the estimator*

$$\theta_t(a) = \frac{1}{T_t(a)} \sum_{n=1}^{t-1} c_n(a) \mathbf{1}_{\{a \in \boldsymbol{a}_n\}} \tag{5}$$

satisfies $\lim_{t \to \infty} \theta_t(a) = \mu(a)$ *w.p.1.*

The proof is straightforward using the strong law of large numbers, given that the consecutive observations have the same distribution and are Bernoulli random variables, observing also that Eq (5) is the usual sample average. The use of the indicator function ensures correctness as t tends to infinity.

An optimal learning policy must use some combination of *exploration* (serving of items a that are not in the top-ℓ in θ_t) and *exploitation* (serving the top-ℓ items according to the estimator θ_t of the user's preferences). The server strategy $\boldsymbol{a}_t$ contains either (i) the top-ℓ categories ranked according to the estimators θ_t in Eq (5), if iteration t is an exploitation, or (ii) a random choice of categories in $\mathcal{A}$ if iteration t is an exploration. Let $p_t > 0$ denote the probability that the action at time t is an exploration. From Lemma 3.1 it follows that, if $p_t \to 0$, then the observed satisfaction will converge to the optimal satisfaction rate in Eq (2). Specifically, the actual cumulative reward (or true user's satisfaction) up to time T and the actual satisfaction rate are:

$$\hat{R}_T \overset{\text{def}}{=} \sum_{t=1}^{T} \sum_{a \in \boldsymbol{a}_t} c_t(a); \quad \hat{S}_T = \frac{\hat{R}_T}{T}. \tag{6}$$

Lemma 3.1 then implies that $\lim_{T \to \infty} \hat{S}_T = S$.

In the literature on MAB problems, strategies that balance exploitation and exploration aim not only at strategies that converge to the optimal reward (user satisfaction in our case), but seek to do so with the fastest convergence rate. Auer et al. [13] provide a summary of the main results that are now used.

As far back as 1985, Lai and Robbins [14] show that, in order to maximize the convergence rate—equivalent to minimizing the 'regret', defined in terms of the difference between the optimal satisfaction and the true satisfaction, that is, $(S - \hat{S}_T)$—the number of times a sub-optimal arm is chosen must increase only logarithmically in t. Intuitively this means that a fast exploration rate will force the algorithm to serve non-optimal arms too often, and a slow exploration rate will lead to slow learning. In the first (1998) edition of their book, Sutton and Barto introduced the ϵ-greedy policy, where exploration is performed with constant probability ϵ; this work has been improved in what is today the more standard reference [12]. Auer et al. [13] study various implementations that are provably optimal, because they satisfy the condition for optimal regret given by Lai and Robbins.

We use the most common two methods in this paper, for illustration. The first is a modification of the ϵ-greedy policy, called by Auer et al. [13] the ϵ_n-greedy algorithm, where the exploration probability p_t decreases as $\mathcal{O}(1/t)$, which ensures the logarithmic increase condition necessary for optimality. The second is the Upper Confidence Bound (UCB) family of algorithms, also studied in detail in Auer et al. [13], where it is shown that they satisfy the optimality condition. UCB algorithms, as well as Thompson schemes, are more complex than ϵ_n-greedy. They are based on statistical estimation of upper bounds that include both estimation of the true values $\mu(a)$ as well as an estimation of the confidence interval that uses the Chernoff-Hoeffding bound. When implementing these, exploration is a consequence of the uncertainty in the estimation of the values $\mu(a)$.

In order to focus on the main results of this paper, we use the simplest strategy that ensures convergence to the optimal value: other strategies may converge faster, but our observations can be extended to more complex algorithms. To illustrate the validity of our results for other schemes we include simulations using a UCB scheme. We remark here that the literature and results quoted above on MAB usually assume that only one arm is pulled at each iteration. In our model, we present a list of ℓ items, so it is as pulling ℓ arms simultaneously. The corresponding policies are either the 'top-ℓ' both for the ϵ_n-greedy as for the UCB algorithms, or a choice of list that draws the categories with likelihoods proportional to the estimated values of $\{\mu(a); a \in \mathcal{A}\}$.

3.1.2 Simulation. We now present simulation results for this first model, Model 1, assuming that the user follows the behavior in Eq (3). The simulation is used to illustrate how the estimated reward can converge to the optimal reward as the number of iterations grow. Shown in abstract as Algorithm 1, it uses a modification of the ϵ-greedy algorithm in Sutton and Barto [12], where exploration is done with decreasing probability p_t. Following Auer et al. [13] the exploration probability is decreased as follows. Let $c > 0$, $0 < d < 1$, $K > 1$, and define

$$p_t = \min\left(1, \frac{cK}{d^2\, t}\right); \quad t = 1, 2, \ldots. \tag{7}$$

The algorithm works as follows. At iteration t, with probability p_t the strategy list is a random choice of ℓ different elements in $\mathcal{A}$ (exploration), or otherwise the top ℓ elements of θ_t are served. We simulate the behavior of the user assuming Eq (3), so that for each element of category $\boldsymbol{a} \in a_t$ presented in the list the click value is a Bernoulli random variable with parameter $\mu(a)$. Next, the estimates are updated using Eq (5). The algorithm includes the computation of the estimate $\hat{R}_T$ (see Eq (6)) of the true cumulative reward R_T defined in Eq (1).

Algorithm 1 Simulation algorithm for Model 1.

```
Initialize (read) $\mathcal{A}, \mu, \ell$.
$p_1 = 1, \theta_1(a) = 0, a \in \mathcal{A}$
for t := 1 to T do
  $u$ = Rand()
  if $u \leq p_t$ then
    Explore: pick a random list $\boldsymbol{a}_t$.
  else
    Exploit: pick the top $\ell$ elements of $\theta_t$.
  User Feedback: for all $a \in a_t$ generate $c_t(a) \sim$ Bernoulli($\mu(a)$)
  Server update: $\theta_{t+1}(a)$ using (5).
  Calculate true reward: $\hat{R}_{t+1} = \hat{R}_t + \sum_{a \in \boldsymbol{a}_t} \mu(a)$
  Update exploration probability $p_{t+1}$ using (7)
```

The outcome of the simulation is shown in Fig 1, in which we used $A = 100$ for the number of categories and and $\ell = 20$ is the size of the served list. The true preferences vector is $\mu(a) = 0.75$ for $a = 0, \ldots, 49$ and $\mu(a) = 0.5$ for $a = 50, \ldots, 99$. Explaining these parameters, in the model the best reward is achieved by presenting $\ell = 20$ items in any of the first 50 categories, and the corresponding satisfaction value is $20 \times 0.75 = 15$. Fig 1 shows how the learning approaches this limit. Note that it takes around 1000 steps to get satisfactorily close (within 0.05% error). This is significant because, in many applications, users may not be willing to wait that many steps before receiving good recommendations.

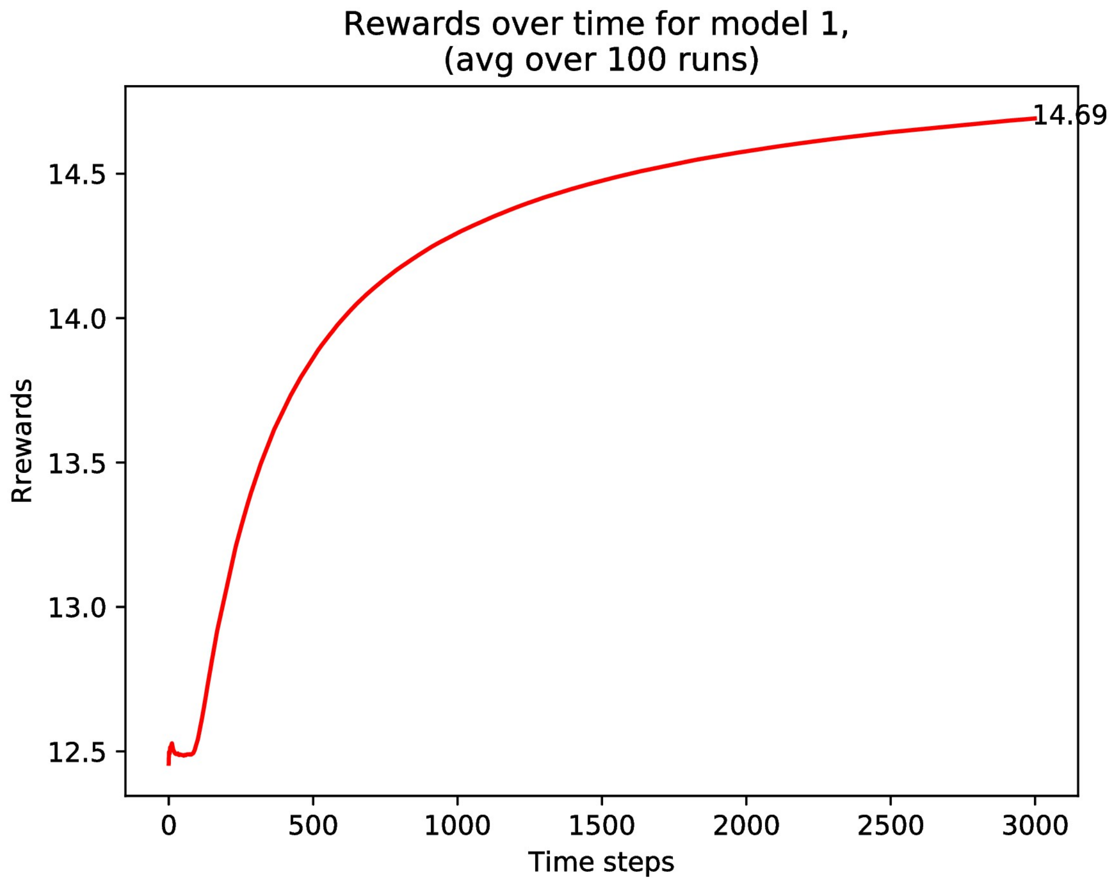


Fig 1. Simulation results for average reward rate function $(\hat{R}_t/t), t = 1, \ldots, T$, where $A = 100$, $\ell = 20$, $\mu = [0.75, \ldots(\times 50), 0.5, \ldots(\times 50)]$, and $T = 1000$, and $(\hat{R}_T/T) = 14.95$. The exploration rate is $p_t = \min\left(1, \frac{cK}{d^2 t}\right)$ based on Eq (7) using $c = 1$, $K = 5$, and $d = 0.25$.

https://doi.org/10.1371/journal.pone.0271268.g001

For completeness, we present in Fig 2 the results of the learning recommender when using the UCB algorithm, which as discussed above is common in the literature of MABs. For our purposes, the UCB algorithm is adapted to the case when multiple arms are chosen at each iteration, because the recommended list contains ℓ items. As expected, UCB seems to converge faster than the ϵ_n-greedy scheme illustrated in Fig 1, and it also has an asymptote of 15.

3.2 Realistic model for dissatisfied user

Consider now the following user model for clicking on items served. When presented with a list, in this model the user clicks on the items for inspection in the order in which they appear in the list. When an item is clicked, the user either watches the video or clicks on the next item, until finding one that they like. To illustrate the phenomenon of persistent error in recommender systems, consider now an unusual case: a user who happens to like only one category of movies $a^* \in \mathcal{A}$, but does not know the label of the category. For example, it may be a rare combination of foreign-language dark romantic mini series. For this user, $\mu(a) \approx 0$ for all categories $a \neq a^*$, whereas $\mu(a^*) = 1$.

In this example, both the system model and user model are simple, allowing us to analyse them mathematically. However, they are not unrealistic, and with large numbers of users of recommender systems some of the users will not be typical of the rest. The combination

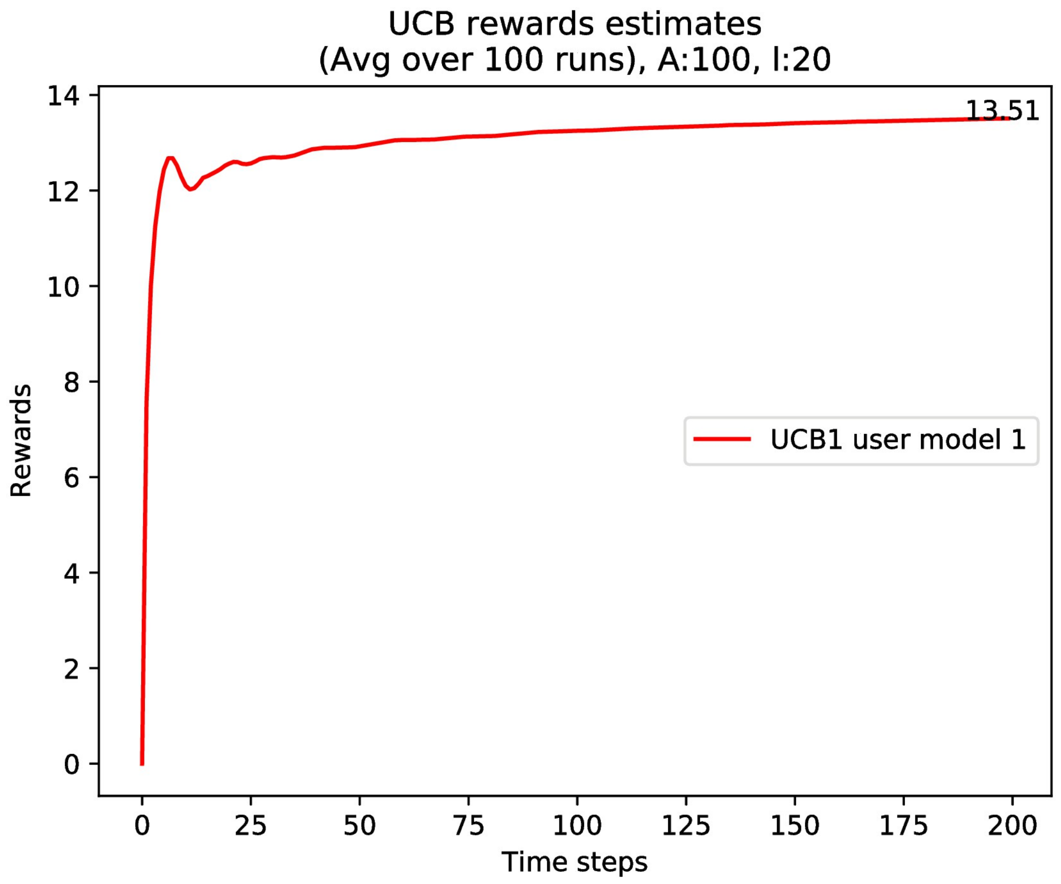


Fig 2. Simulation results for average reward rate function using the UCB algorithm $(\hat{R}_t/t), t = 1, \ldots, T$, where $A = 100$, $\ell = 20$, $\mu = [0.75, \ldots(\times 50), 0.5, \ldots(\times 50)]$, and $T = 1000$, and $(\hat{R}_T/T) = 14.95$.

https://doi.org/10.1371/journal.pone.0271268.g002

creates an interesting test case because of the likelihood that some behaviour will be unanticipated in the design of any realistic system. Thus it is a case where a proxy described in the literature does not capture a potential realistic human behaviour. As we show both mathematically and in simulation, this can lead to severe performance issues that cannot be detected by the system.

Here, then, the actual pattern of clicks is as follows. When served a list of items of categories $\boldsymbol{a}_t = (a_{t,1}, \ldots, a_{t,\ell})$ this user will click all items in order until either reaching an item in a^* that is liked or until the list is exhausted. Formally, for $\boldsymbol{a}_{t,k} \in a_t$,

$$c_t(a_{t,k}) = \begin{cases} 1 & \text{if } a^* \neq \boldsymbol{a}_t \\ 1 & \text{if } (a_{t,j} = a^*) \,\&\, (k \leq j) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Clearly $\mathbb{E}[c_t(a)] \neq \mu(a)$, but the server continues behaving under the assumption of the model in Eq (3). The following analysis shows how this can lead to persistent recommender errors.

3.2.1 Analysis and simulation results. Let $v = \min(t{:}a^* \in \boldsymbol{a}_t)$ be the first time that the category a^* is chosen by the server. Note that up to v the user has clicked on all the items previously presented, so $\theta_v(a) = 1$ for all served categories $\boldsymbol{a} \in \{a_k; k < v\}$ and $\theta_v(a^*) = 0$. When presented with the list from the strategy $\boldsymbol{a}_v$ the user clicks until finding an item in a^*, so all items that appear in this list after that will decrease their estimate θ_v, and $\theta_{v+1}(a^*) = 1$. This stopping time v is important. In particular, the true cumulative reward (satisfaction) for the user up to time v is nearly zero: $\sum_{t<v}\sum_{a\in a_t}\mu(a) \approx 0$, whereas the server has used a model where the estimated cumulative reward is given by

$$\mathbb{E}\left[\sum_{t<v}\sum_{a\in\boldsymbol{a}_t} c_t(a)\right] = \ell \times (\mathbb{E}[v] - 1),$$

corresponding to a reward rate of approximately ℓ (the largest possible).

Proposition 3.1. *Let $A = |\mathcal{A}|$ be the size of the category set. Assume that the server chooses to explore at iteration t with probability p_t, in which case $\boldsymbol{a}_t$ consists of ℓ items chosen at random. Otherwise, the server presents the top-ℓ items according to its current estimate $\theta_t(\cdot)$. Then*

$$\mathbb{E}[v] = \sum_{k=1}^{\infty}\prod_{j=1}^{k-1}\left(1 - p_j\frac{\ell}{A}\right). \tag{9}$$

Proof. The proof is in an Appendix.

The preceding material allows us to make the following observations. First, for this scenario, the server estimates the asymptotic user's satisfaction at 100%. This follows from the fact that the server believes model Eq (3), as set out in the simplistic model, while the true behavior is governed by Eq (8).

Second, if exploration is not done, then $p_t = 0$ and Proposition 3.1 is not applicable. In this case, unless $a^* \in \boldsymbol{a}_1$ is chosen for the first list, then necessarily the first ℓ categories chosen will have $\theta_1(a) = 1$ and will be persistently served for all time $t \geq 1$. That is, only with small probability ℓ/A the user will be happy, but otherwise will be forever dissatisfied.

Third, if constant exploration is used (as in Sutton and Barto [12]) with $p_t \equiv \epsilon$ then v is a geometric random variable with parameter $(\epsilon\ell/A)$, so $\mathbb{E}[v] = A/(\in \ell)$. This number can be very large when $\ell \ll A$, leading to the server's prediction error also being very large.

Fourth, even after time v, when the server will estimate $\theta_t(a^*) = 1$ for all $t \geq v$, there is a probability that the recommender system will continue to serve lists that do not contain items of category a^* because many other items may also have an estimate $\theta_k(a) = 1$. If the lists are presented with the top ℓ items ordered at random, due to random exploration, the correct item a^* will eventually *surface*, meaning that there is an a.s. finite time k such that $\forall a \in \mathcal{A},\ \theta_k(a) > 0$ and $\theta_k(a^*) \geq \sup_a(\theta_k(a))$. From that point onwards a^* will remain in the top ℓ of θ. Only in this scenario will the asymptotic true satisfaction be good.

Observe that, in all cases, the transition time until the recommender system begins to serve the preferred item a^* with high frequency may be greater than is realistically reasonable.

Proposition 3.2. *Assume that when serving the top-ℓ list to the user, the server chooses the top ℓ categories according to θ_t, and under ties, it presents the corresponding items in the order of their category labels. Then if $a^* > \ell$ the server will eventually stop serving category a^* during the exploitation iterations.*

Proof. The proof is in an Appendix.

As a corollary of the above results, the persistent errors yield asymptotically wrong recommendations under the assumptions of Proposition 3.2. In particular, $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{a\in\boldsymbol{a}_t}\mu(a) = 0$ with probability 1. To see this, write the expected reward up to time T (sufficiently large) as

$$\frac{R_T}{T} = \frac{1}{T}\sum_{t=1}^{\tau}\mu(a) + \left(\frac{T-\tau+1}{T}\right)\left(\frac{1}{T-\tau+1}\right)\sum_{t=\tau+1}^{T}\mu(a)$$

The first term will tend to zero w.p.1 as $T \to \infty$. In the second term, the factor $(T - \tau + 1)/T \to 1$, and the other factor is an average reward. Thus, for large T, using exploration probability $p_t = \rho^t$, so that the likelihood of exploration falls to 0 over time,

$$\begin{aligned}\frac{R_T}{T} &\leq \left(\frac{1}{T-\tau+1}\right)\sum_{t=\tau+1}^{T}\mathbb{P}(a^* \in \boldsymbol{a}_t) + c\cdot T^{-1}\\ &= \left(\frac{1}{T-\tau+1}\right)\frac{\ell}{M}\sum_{t=\tau+1}^{T}\rho^t + c\cdot T^{-1} \leq \left(\frac{\ell}{M(T-\tau+1)}\right)\frac{1-\rho^T}{1-\rho} + c\cdot T^{-1} \to 0,\end{aligned}$$

where c is a constant. In the case of fixed exploration probability ($p_t = \epsilon$), the corresponding limit average reward is $\epsilon\ell/A$, which is very small. In any case, the server's estimate of the average reward will be 100% satisfaction. Therefore, even if $\theta_t(a^*) = 1$ correctly estimates $\mu(a^*)$, the error due to the wrong proxy will be persistent and will lead to near-complete dissatisfaction.

Remark: In this case, the server believes that the user's actions satisfy Eq (3), so the server's estimate for the believed reward is $\tilde{R}_T = \sum_{t=1}^{T}\sum_{a\in a_t} c_t(\boldsymbol{a})$, In fact, however, $\mathbb{E}[\tilde{R}_T] \neq \mathbb{E}[R_T]$.

To compare the behavior of the recommender system for a dissatisfied user with the behavior that the server is assuming, we perform the simulation shown in Algorithm 2. The only difference in the simulations is the behavior of the user, that is, the calculation of the click binary variables $c_t(a)$. Naturally the server does not know what model a user follows.

Algorithm 2 Simulation algorithm for model 2

```
Initialize (read) $\mathcal{A}, \ell, \rho < 1$, $a^*$.
$p_1 = 1, \theta_1(a) = 0, a \in \mathcal{A}$ $\mu(a^*) = 1$; $\mu(a) = 0$, $a \neq a^*$.
for t := 1 to T do
  $u$ = Rand()
  if $u \leq p_t$ then
    Explore: pick a random list $\boldsymbol{a}_t$.
  else
    Exploit: pick the top $\ell$ elements of $\theta_t$ (randomize if more than $\ell$ are top).
  User Feedback:
  $j$ = 0
  while ($a_{t,j+1} \neq a^*$) or ($j < \ell$) do
    $j$ := $j$ + 1
    $c_t(a_{t,j})$ = 1
  Server update: $\theta_{t+1}(a)$ using (5).
  Calculate true reward: $\hat{R}_{t+1} = \hat{R}_t + \sum_{a\in\boldsymbol{a}_t}\mu(a)$
  Calculate the server's belief about the reward: $\tilde{R}_{t+1} = \tilde{R}_t + \sum_{a\in\boldsymbol{a}_t} c_t(a)$
  Update exploration probability $p_{t+1}$ using (7).
```

The following results show the plots of the reward functions for the model with an unsatisfied user, as described above. The plots show both the true reward R_t/t as a function of the iteration number t, as well as the reward that the server believes to describe the user's satisfaction, that is, $\tilde{R}_t/t$. We perform three simulations to illustrate the results above.

Our first simulation assumes that the user only likes category $a^* = 5$. As described above, there is a transition time v before the server picks a^* to be presented in the list. From that time onwards the server will continue to present this category, because the corresponding estimate $\theta_t(a^*) = 1$ (we have assumed that the user always clicks on the preferred category when presented). However, all other categories presented in the list in front of a^* will also have an estimate of $\theta_t(a) = 1$, $a < a^*$ provided that these categories have been shown at least once, because we assume that the user clicks on items in the order that they are presented. Because of occurrences of random exploration, eventually all five categories $a \in \{0, 1, \ldots, 4\}$ will appear in the list, so the believed reward by the server has a limit $\lim_{t\to\infty} \tilde{R}_t/t = 6$. But this is utterly incorrect. The user only likes category $a^* = 5$, so the true reward has a limit $R_t/t \to 1$. This behaviour is clearly evident in Fig 3a.

Fig 3b shows the corresponding plots when using the UCB algorithm for this model. In our simulations, the user still clicks until they find a category they like. The only difference is that, instead of using the top-ℓ estimated θ values and random exploration, the server presents the list according to the top-ℓ estimated upper bounds. As the figure illustrates the asymptotic behavior is the same as with the ϵ_n-greedy policy.

To illustrate Proposition 3.2, we present in Fig 4a a second simulation when $a^* = 90 > \ell$. As predicted, the server eventually diverges from the true satisfaction, and, even if it has served the preferred category, it is fooled by the clicks on items served in front of the list. Eventually the server stops serving the preferred category during exploitation (and the exploration probability goes to zero). In the plot, it is evident that this happens very quickly. Because the list has $\ell = 20$ items, the server believes that it is achieving the maximal satisfaction rate, that is, $\lim_{t\to\infty} \tilde{R}_t/t = 20$, while the user's satisfaction rate is going to 0.

Fig 4a shows the average rewards over 100 replications of the simulation when $a^* = 90$, as discussed. Fig 5 shows a typical trajectory, demonstrating that, as expected, even when the correct category appears at an early time, it slowly gets replaced by items in other categories that are shown at the front of the list.

The plot shown in Fig 4b shows the corresponding rewards when the server uses the UCB policy. As expected, the asymptotic behavior is the same as with the ϵ_n-greedy policy, illustrating again that the principles that we have established with out theoretical analysis are independent of the specific method for machine learning used by the server.

Our third simulation is used to present a more realistic scenario, in constrast to the above, which is an extreme of only liking one category. It is illustrated in Fig 6a. In this simulation the user somewhat likes a few categories, and dislikes the rest. Specifically, here:

$$\mu(a) = \begin{cases} 0.88 & a = 3 \\ 0.62 & a = 26 \\ 0.54 & a = 45 \\ 0.92 & a = 77 \\ 0.00 & \text{all other categories} \end{cases}$$

Surprisingly, the plots in Fig 6a show behavior very similar to the extreme case. To understand the behaviour, in this case $\mathbb{P}(\text{user likes an item in category3}) = 0.88$, meaning that the user will always click on this category when it is presented. With probability 0.88 the user will stop there; with probability 0.12 the user does not like the movie of category 3 that was served in the list, and proceeds to click on further items. As the other categories that the user likes are 26, 45, and 77, any category $a < 26$ served will be clicked, provided that either category 3 is not served or it is served but the movie was liked.

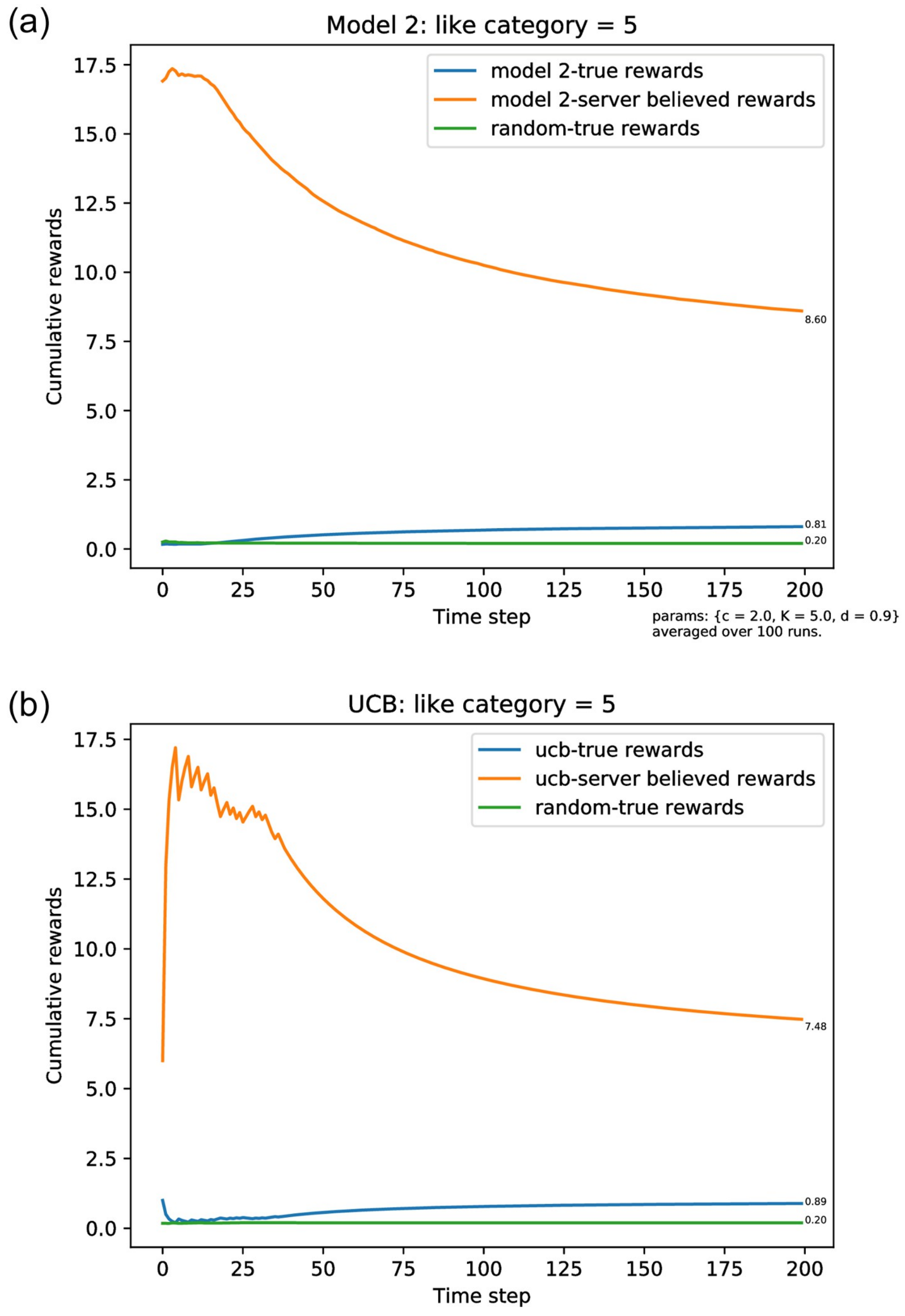
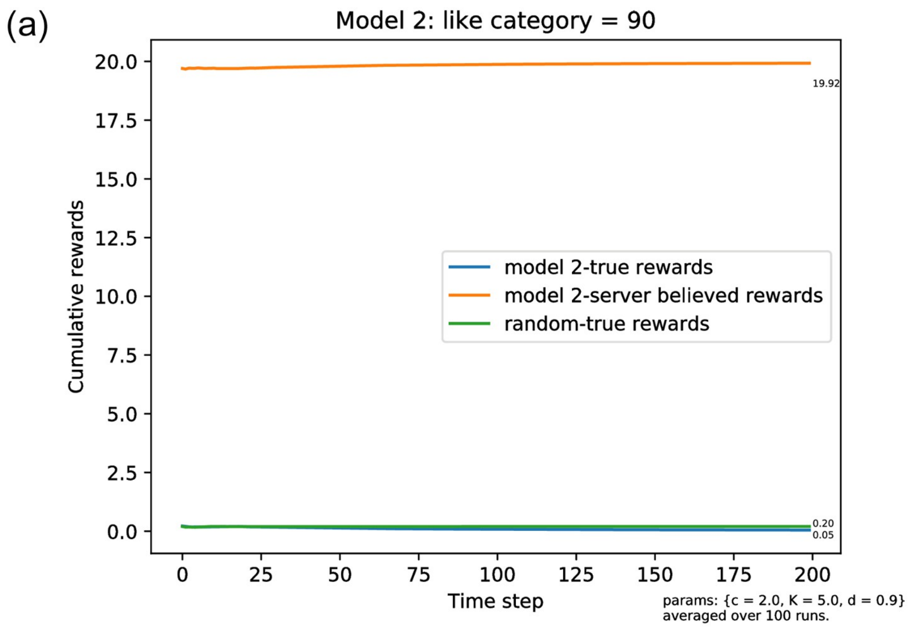


Fig 3. Simulation results for the true average reward rates where categories of items are presented in a fixed order and users continue clicking until seeing a liked item. The true average reward rate $(\hat{R}_t/t), t = 1, \ldots, T$, where the server believes the average reward to be $\tilde{R}_t/t$. Here $\mathcal{A} = 100, \ell = 20, a^* = 5$. (a) Model 2 rewards, the exploration rate follows Eq (7). (b) The server uses the UCB policy.

https://doi.org/10.1371/journal.pone.0271268.g003

A simple analysis demonstrates that eventually, due to random exploration, all first 20 categories will be persistent. Because category 3 is the only one included, the user's reward rate satisfies $R_t/t \rightarrow 0.88$, while the maximal rate would be obtained by including 3, 26, 45, and 77 in the served lists, giving $0.88 + 0.12(0.62 + 0.38(0.54 + 0.46 \times 0.92)) = 0.9983$. On the other hand, the server believes that the user likes all four of the first categories plus, with probability 0.12,



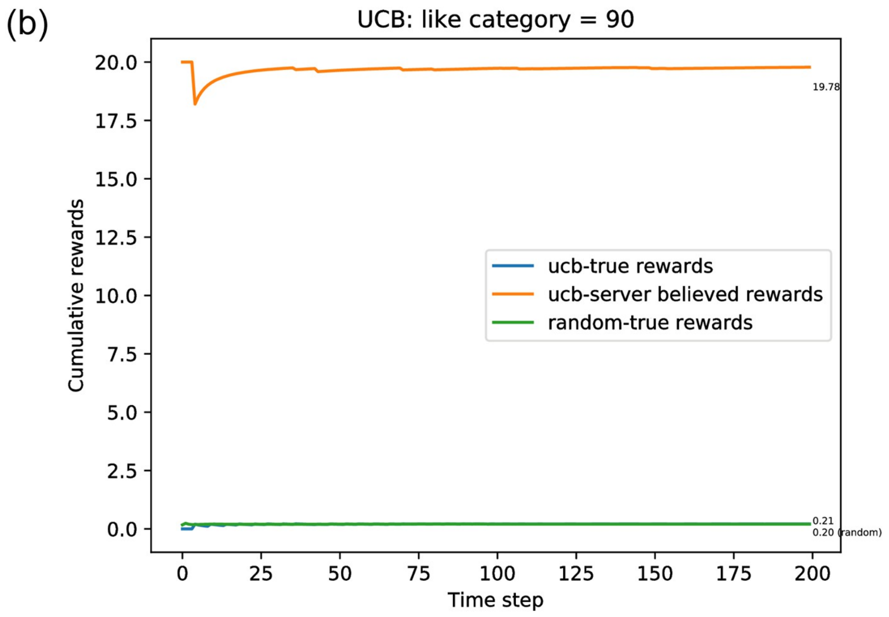


Fig 4. Simulation results for the true average reward rates where categories of items are presented in a fixed order and users continue clicking until seeing a liked item. The true average reward rate $(\hat{R}_t/t), t = 1, \ldots, T$, where the server believes the average reward to be $\tilde{R}_t/t$. Here $\mathcal{A} = 100, \ell = 20, a^* = 90$. (a) Model 2 rewards, the exploration rate follows Eq (7). (b) The server uses the UCB policy.

https://doi.org/10.1371/journal.pone.0271268.g004

all of the following 16 categories, yielding $\tilde{R}_t/t \rightarrow 4 + 0.12(16) = 5.92$. The true optimal action would be to recommend items from categories (77, 3, 26, 45, . . .) in this order, where any other 16 categories can fill the lists.

However, there is no mechanism for the server to learn that the order of the items served is important for this user. This is the fundamental problem we are considering: that any user

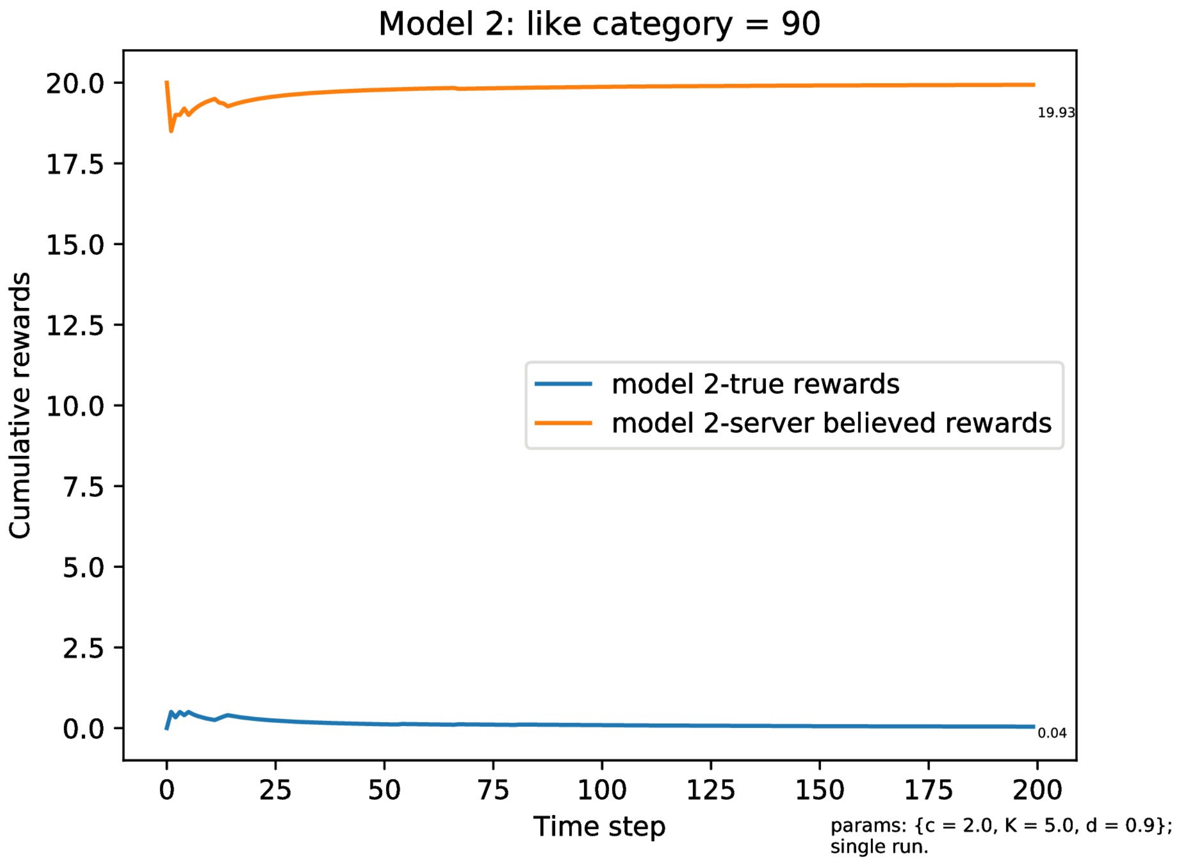


Fig 5. Simulation results of model 2 for a trajectory of true reward $(\hat{R}_t/t)$, $t = 1, \ldots, T$, wherethe server believes the reward to be $\tilde{R}_t/t$. Here $\mathcal{A} = 100$, $\ell = 20$, and $a^* = 90$.

https://doi.org/10.1371/journal.pone.0271268.g005

behaviour that lies outside the assumptions of the system can be undetectable, and lead to asymptotically catastrophic behaviour.

The final experiment in this section that we present is the result of applying the UCB policy for learning, for the same model as in Fig 6a. As before, we verify that the main asymptotic behavior is independent of the learning mechanism used to choose the recommendations.

Critically, these results show that although the server's response is close to random, the server's metrics show that it is doing well; in the presence of unanticipated behaviour an arbitrary degree of failure is invisible. In principle the response can even be worse than random, that is, the actual satisfaction rate (true reward) would be below that given by an arbitrary choice of item. This is evident in the (extreme) case where $a^* = 90$, because a random recommender would persistently include this category with probability $\ell/M = 1/5$, while the 'smart' algorithms eventually cease to present this category. This is also true for our mixed experiment for the same reason: the 'smart' recommender eventually only includes $a = 3$ in the served lists, but a random recommender would persistently also include categories 26, 45, and 77, thus increasing the average reward.

In our experiments, the user is not aware of the category that they like. Instead, they click until finding an item that is satisfactory. It may be argued that, in the case of Fig 3a, when the user only likes category $a^* = 5$ they may eventually learn that the first five items presented are of no interest, and so would start clicking the sixth item presented. First, this is a situation

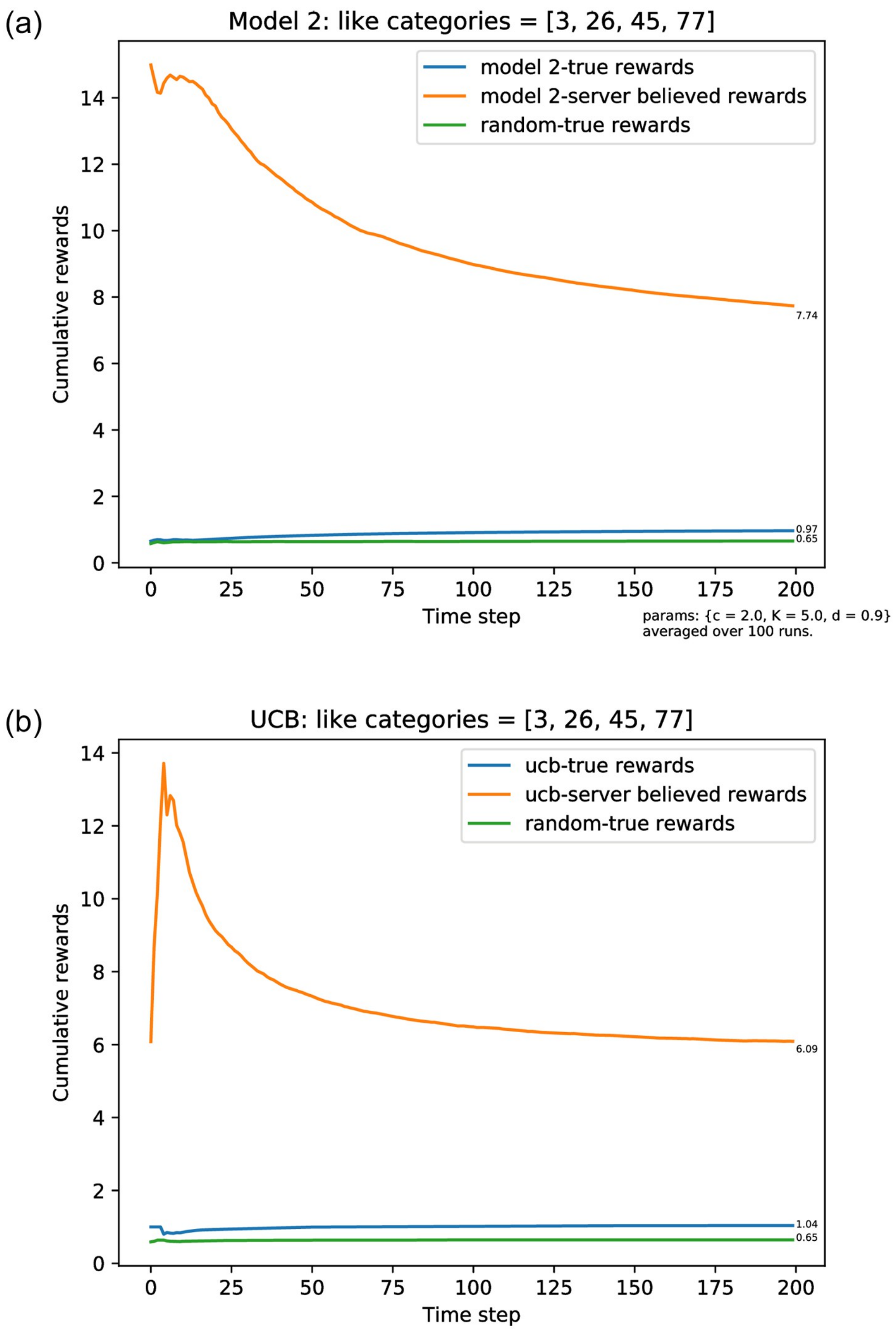


Fig 6. Simulation results for model 2 for the true average reward$(\hat{R}_t/t)$, $t = 1, \ldots, T$ and the server-believed average reward $\tilde{R}_t/t$. Here $\mathcal{A} = 100, \ell = 20$. The true satisfaction probabilities of all categories are zero, except for four categories: 3, 26, 45, and 77; their respective probabilities are $\mu(3) = 0.88$, $\mu(45) = 0.54$, $\mu(77) = 0.92$, and $\mu(26) = 0.62$. (a) Model 2 rewards, the exploration rate follows Eq (7). (b) The server uses the UCB policy.

https://doi.org/10.1371/journal.pone.0271268.g006

where the user learns and changes their behavior, rather than the algorithm learning from the user to adapt its strategy. Second, and more importantly, we observe that were this the case the server would eventually update the estimated $\theta_t(a)$, $a < a^* = 5$ as being smaller than $\theta_t(a^*)$, so it will present items from category $a^* = 5$ before position 6 in the list. Our user has already learnt to skip the first five items, so will again be in the position of clicking items that are disliked. There is no intrinsic way for the computer to learn the correct answer.

3.3 Hidden or silent categories

We now present a different scenario. Continuing with the analogy of the movie recommender system, we assume here that the server has already classified movies into A categories with labels $\mathcal{A} = \{0, \ldots, A-1\}$, where $\| \mathcal{A} \| = A$.

We assume that the server behaves as follows. The parameter $\theta_t(\cdot) \in \mathbb{R}^+$ represents the estimation of the user's cumulative satisfaction by the server and a list $\boldsymbol{m}_t = (m_{t,1}, \ldots, m_{t,\ell})$ is served as follows. The number of items (movies) of each of the categories, denoted by $\boldsymbol{L}_t = (\ell_{t,1}, \ldots, \ell_{t,A})$, is chosen according to a multinomial distribution MN $(\ell, \boldsymbol{p}_t)$ where where

$$\boldsymbol{p}_t(a) = \frac{\theta_t(a)}{\sum_{a' \in \mathcal{A}} \theta_t(a')}, \qquad a \in \mathcal{A}. \tag{10}$$

Recall that a multinomial random variable satisfies: $\ell_{t,a}$ = number of category a chosen for the list, with $\sum_{a=1}^{A} \ell_{t,a} = \ell$. When a list is served, for each $1 \le i \le \ell$ the actual item $m_{t,i}$ served is a (random) movie from each of the chosen categories $k \in \mathcal{A}$ such that $\ell_{t,k} \neq 0$. For each item served, the category from the server's list is unique; we use the notation $m_i \in a$ to indicate that the i-th movie served belongs to category a. So, for example, if $\ell_{t,1} = 3$ then three of the items in the served list belong to category 1, in which case the server presents three different movies from category 1.

The list presented is denoted as $\boldsymbol{m}_t = (m_{t,1}, \ldots, m_{t,\ell})$, chosen as follows. The first item $m_{t,1}$ is of the first category that is present in the list $\boldsymbol{L}_t$, that is,

$$a_{t,1} = \min(a\colon \ell_{t,a} > 0), \qquad \text{and } \; m_{t,1} \in a_{t,1}.$$

Then the list $\boldsymbol{L}_t$ is updated by decreasing $\ell_{t,a}$ by one. Denote by $\boldsymbol{L}'_t$ the resulting modified list. The next item $m_{t,2}$ is found using the same algorithm: the first category of the updated list that has a positive number, that is, $a_{t,2} = \min(a\colon \ell'_{t,a} > 0), \qquad \text{and } \; m_{t,2} \in a_{t,2}$, and $\ell'_{t,a}$ is decreased by one. The recursion is executed until the last movie is chosen. Note that, if the original number $\ell_{t,a} > 1$, then consecutive movies presented in the list are of the same category.

A click on an item corresponds here to watching the movie, taken by the system to imply that the user liked the item served. Thus we use

$$\theta_{t+1}(a) = \theta_t(a) + C_t(a), \tag{11}$$

where $C_t(a)$ is the total number of clicks at time t of items in category a presented at t (see below for a model for the clicks). An item that has been clicked will not be chosen again by the server.

3.3.1 Probability model for user's behavior. The model corresponds to the situation where the user likes a particular category, but this *hidden category* is not one of the predefined categories identified by the server—and thus it is not discoverable feature for the machine learning algorithms. That is, the representation is incomplete.

We denote the hidden category as α. For example, the server may use a list such as 'drama, comedy, family, thriller, horror, Oscar winner, documentary, sports'. However, the user likes only movies from Spain, Mexico, or Colombia. While $\alpha \notin \mathcal{A}$, for every server category $a \in \mathcal{A}$, it is likely that $a \cap \alpha \neq \emptyset$, meaning that there are movies of the hidden category present in every one of the server's defined categories.

Thus, when served a list of items $\boldsymbol{m}_t = (m_{t,1}, \ldots, m_{t,\ell})$, we have the *click model* that the user clicks on all of the items that belong to hidden category α. We denote $C_t(a) = \sum_{(i:mt,i \in a)} c_t(m_{t,i})$, where for $i = 1, \ldots, \ell$

$$c_t(m_{t,i}) = \begin{cases} 1 & \text{if } m_{t,i} \in \alpha \quad (\text{user clicks on movie } m_{t,i} \text{ at time } t) \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

We use the notation $f_t(a)$ for the fraction of α movies remaining in category a at time t, that is,

$$f_t(a) = \mathbb{P}(\text{User likes item } m \in m_t \mid m \in a) \tag{13}$$

This probability will directly determine the model for the random variable $C_t(a)$. From our model, it follows that $\mathbb{P}(C_t(a) = k \mid \ell_{t,a})$ is the probability that exactly k amongst the $\ell_{t,a}$ movies of category a served in the list are α-movies.

When $M > > m$ a simple counting argument can be used to approximate the distribution of $C_t(a)$ by the following binomial distribution:

$$\mathbb{P}(C_t(a) = k \mid \ell_{t,a}) = \binom{\ell_{t,a}}{k} f_t(a)^k\,(1 - f_t(a))^{\ell_{t,a}-k}, \quad 0 \le k \le \ell_{t,a}. \tag{14}$$

Lemma 3.2. *Under the assumption that the list is chosen according to the MN(ℓ, $\boldsymbol{p}_t$) distribution (see* Eq (10)) and that (14) *holds, then*

$$\mathbb{E}[C_t(a) \mid \mathfrak{F}_t] = \ell \left(\frac{\theta_t(a)}{\bar{\theta}_t}\right) f_t(a), \tag{15}$$

where $\bar{\theta}_t = \sum_a \theta_t(a)$ and $\mathfrak{F}_t$ is the sigma-field (or history) of the process $\{\ell_{n,\cdot}, n \le t\}$.

Proof. Fix the time t; we now drop this index from the notation. Because the served list with ℓ items chooses category a according to the multinomial distribution, then it follows that the number $\ell_{t,a}$ of movies from category a served at time t has a marginal distribution that is a binomial distribution, that is $\ell_a \sim$ Bin $(\ell, \boldsymbol{p}_t(a))$. On the other hand, it follows from Eq (14) that $\mathbb{E}[C_t(a) \mid \ell_a] = \ell_{t,a} f_t(a)$, which implies that, for any $a = 0, \ldots, A - 1$, $\mathbb{E}[C_t(a)] = \mathbb{E}[\mathbb{E}[C_t(a) \mid \ell_{t,a}]] = \mathbb{E}[\ell_{t,a} f_t(a)] = \ell\,\boldsymbol{p}_t(a)\,f_t(a)$. Use of Eq (10) now proves the assertion.

As we keep a constant amount M of items in each category and the user has already consumed $C_t(a)$ movies of category a at time t, our model assumes that the server outsources new $C_t(a)$ movies of the same category a. For simplicity, we also assume that the fraction of α movies added is the same as the original fraction $f_0(a)$. This leads to the update:

$$f_{t+1}(a) = f_t(a) - \frac{C_t(a)}{M} + \frac{\xi_t(a)}{M}, \tag{16}$$

where $\xi_t(a)$ has a binomial distribution Bin($C_t(a)$, $f_0(a)$) and it is independent of the other random variables. This follows because $\xi_t(a)$ represents the number of new α-movies in category a and we assume that the fraction of α movies in the (unbounded) pool of outside movies is the constant $f_0(a)$. Thus $\mathbb{E}[\xi_t(a) \mid C_t(a)] = f_0(a)\,C_t(a)$. For simplicity, from here onwards we assume that $f_0(a) \equiv f_0$ is constant for all categories, and call $k = \frac{(1-f_0)}{M}$. When convenient, we use the approximate model for the updates of the fractions:

$$f_{t+1}(a) = f_t(a) - k\,C_t(a) \tag{17}$$

3.3.2 Analysis and simulation results. We now proceed to describe how the server's estimate θ_t and the user's real preference f_t behave. It develops that the model is related to a modified predator–prey model, and will be a basis of our claim that the 'learning' algorithm actually works worse than a random recommender. The server updates its estimates using Eq (11) and the fractions decrease according to Eq (16), that is: $\theta_{t+1}(a) = \theta_t(a) + C_t(a)$ and $f_{t+1}(a) = f_t(a) - \frac{C_t(a)}{M} + \frac{\xi_t(a)}{M}$, where $\xi_t(a) \sim \text{Bin}(C_t(a), f_0)$.

The (vector-valued) stochastic process is the joint process $\{(\theta_t, f_t); t = 1, 2, \ldots\}$. To analyze the behavior, we consider first the conditional drifts, or 'tendencies' for the processes. (A process x_t has conditional drift $\mathbb{E}[x_{t+1} - x_t \mid x_t]$, a quantity that is useful for assessing how the dynamics evolve.) Using the results of Lemma 3.2 yields

$$\begin{aligned}\mathbb{E}[\theta_{t+1}(a) \mid \mathfrak{F}_t] &= \theta_t(a) + \ell\left(\frac{\theta_t(a)}{\bar{\theta}_t}\right) f_t(a)\\ \mathbb{E}[f_{t+1}(a) \mid \mathfrak{F}_t] &= f_t(a) - k\,\ell\left(\frac{\theta_t(a)}{\bar{\theta}_t}\right) f_t(a).\end{aligned}$$

These finite difference equations already point at the key fact: the higher the value of $\theta_t(a)$ the higher the chance of showing items in category a, which exhausts the remaining α-movies, thus decreasing $f_t(a)$ and the user's satisfaction. However, the server increases $\theta_t(a)$ instead of decreasing it. The system thus has similarities to a predator–prey model: the predator is the server (via $\theta_t(\cdot)$) but, in consuming the resources (the α-movies), it depletes the prey (via $f_t(\cdot)$).

Remark We note that the update Eq (17) for f_t is not quite correct because in principle this could lead to negative values, so a truncation must be added to ensure that $f_t \geq 0$ w.p.1. We study the general behavior, but truncate the updates; that is, our code uses $f_{t+1}(a) = \max(0, f_t(a) - k\, C_t(a))$.

In addition to describing the dynamics, we use the cumulative reward function

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T}\sum_{a\in\mathcal{A}} C_t(a) \mid \mathfrak{F}_T\right] = R_{T-1} + \sum_{a\in\mathcal{A}} \ell\left(\frac{\theta_t(a)}{\bar{\theta}_t}\right) f_t(a).$$

This expression is a direct consequence of Lemma 3.2. It uses the fact that the expected instantaneous reward from category a (the expected number of clicks) is the probability that a is chosen $\ell(\theta_t(a)/\bar{\theta}(a))$ times the probability that the chosen movie is of category α.

Remark Unlike the model for the dissatisfied user, in this model the user clicks when they are happy with the item. In that sense, the satisfaction is indeed the expected number of clicks, so the reward above is correct. The difference between this model and Eq (3) is that in this case, because of the hidden preference, the probability of liking a category changes with time. In other words, $\mu(a)$ is not constant but instead is equal to $f_t(a)$.

Algorithm 3 is used for the simulation of hidden categories, as follows. Following the description above, given θ_t the server chooses the categories according to a multinomial random variable $\text{MN}(\ell, \boldsymbol{p})$, using Eq (10) and this defines the list of movies presented at step t. The user then clicks on the movies that are of the hidden or silent category α. We simulate this using Lemma 3.2 so that $C_t(a) \sim \text{Bin}(\ell_{t,a}, f_t(a))$, which provides the immediate expected reward for each a: $\mathbb{E}[C_t(a)] = \ell\,\boldsymbol{p}_t(a) f_t(a)$. (This *conditional Monte Carlo approach* enables our simulation to be efficient, because it reduces variance.) The server updates θ with (11) and the fraction of α-movies is updated using Eq (17).

Algorithm 3 Simulation algorithm for Hidden category

```
Initialize (read) $\mathcal{A}, \ell$, $f_0(a) = f_0$, $\theta_0(a)$.
$R_0 = 0, \boldsymbol{p}_0(a) = \theta_0(a)/\sum_{a'}\theta_0(a')$ (from (10))
```

```
for t := 1 to T do
  Served list: $\boldsymbol{L}_t \sim$ MN($\ell$, $\boldsymbol{p}$).
  User Feedback:
  for $a \in \mathcal{A}$ do
    $C_t(a) \sim$ Bin($\ell_{t,a}$, $f_t(a)$).
  Calculate expected reward: $\hat{R}_{t+1} = \hat{R}_t + \sum_{a\in\mathcal{A}} \ell \boldsymbol{p}_t(a) f_t(a)$
  Server update: $\theta_{t+1}(a)$ using (11)
  Fractions update: $f_{t+1}(a)$ using (17)
  Update probability $\boldsymbol{p}_{t+1}$ using (10).
```

Our benchmark is the *random recommender*, for which the cumulative reward is simply

$$R_T^* = \sum_{t=0}^{T-1} \frac{\ell}{A} \sum_{a\in\mathcal{A}} f_t(a),$$

because each category is chosen at random (with replacement) from the A possibilities to be served in the list of ℓ items. That is, this system behaves using constant $\theta_t(a)$.

We can further illustrate the benchmark behavior of the random policy. Here, because categories a are chosen always at random from the set $\mathcal{A}$ to fill the list of ℓ movies, we have $\mathbb{E}[C_t(a) \mid \mathfrak{F}_t] = (\ell/A) f_t(a)$. Using $f_{t+1}(a) = f_t(a) - k\, C_t(a)$, for any $a \in \mathcal{A}$ we have

$$\mathbb{E}[f_{t+1}(a) \mid \mathfrak{F}_t] = f_t(a) - \left(\frac{k\,\ell}{A} f_t(a)\right) = f_t(a)\left(1 - \frac{k\,\ell}{A}\right).$$

Therefore the average fraction of α-movies of the random recommender decreases geometrically fast, we have $\mathbb{E}[f_t(a)] = f_0(1 - k\ell/A)^t$, and it is the same for all a (because $f_0(a) \equiv f_0$ is a constant). This yields

$$R_T^* = \sum_{t=0}^{T-1} \ell\, \mathbb{E}[f_t(a)] = \ell f_0 \sum_{t=1}^{T} \left(1 - \frac{k\ell}{A}\right)^t = \frac{1 - (1 - k\ell/A)^T}{(k\ell/A)}. \quad (18)$$

We note that when simulating the random recommender, the only difference from Algorithm 3 is that there is no update on θ_t, so that $\boldsymbol{p}_t$ is constant.

Fig 7 shows the the cumulative rewards of the model with hidden categories. For our simulations we initialize all fractions $f_0(a) = f_0 = 0.5$. The smart recommender (explained below) follows the multinomial policy for the lists, and updates according to Eqs (11) and (17), as indicated in the pseudo-code for Algorithm 3.

We call our simulation model for the learning scheme the *smart* recommender. Here we initialize all $\theta_0(a) = 0.5$ and all $f_0(a) = 0.5$ so that there is no initial bias toward any given category. We refer to this as the *uniform* case. Other parameters have the same values as before. Fig 7 shows that even in the case that the learning has no initial bias, it is still worse than the random recommender. The plots were obtained performing 100 replications of the simulations to calculate the average rewards. We observe that for this model there is partial replenishment of the items served, because only the movies that are liked are replaced. As a consequence, because the user exhausts only the category α items, there is a natural depletion of the resources, until the fraction of α movies left is close to zero and both uniform and random recommender yield a near-zero instantaneous reward. Similar outcomes occur under the UCB policy, as shown in Fig 8.

3.3.3 Full replenishment. As a further alternative, we now assume that the feedback from the user is a 'thumbs-up' or 'thumbs-down' binary action. Thus, when presented with a list $\boldsymbol{L}_t$ of ℓ items (movies) the user provides a binary rank for each item. The user still clicks on all of the liked items, so the model for the clicks $C_t(a)$ is the same as before.

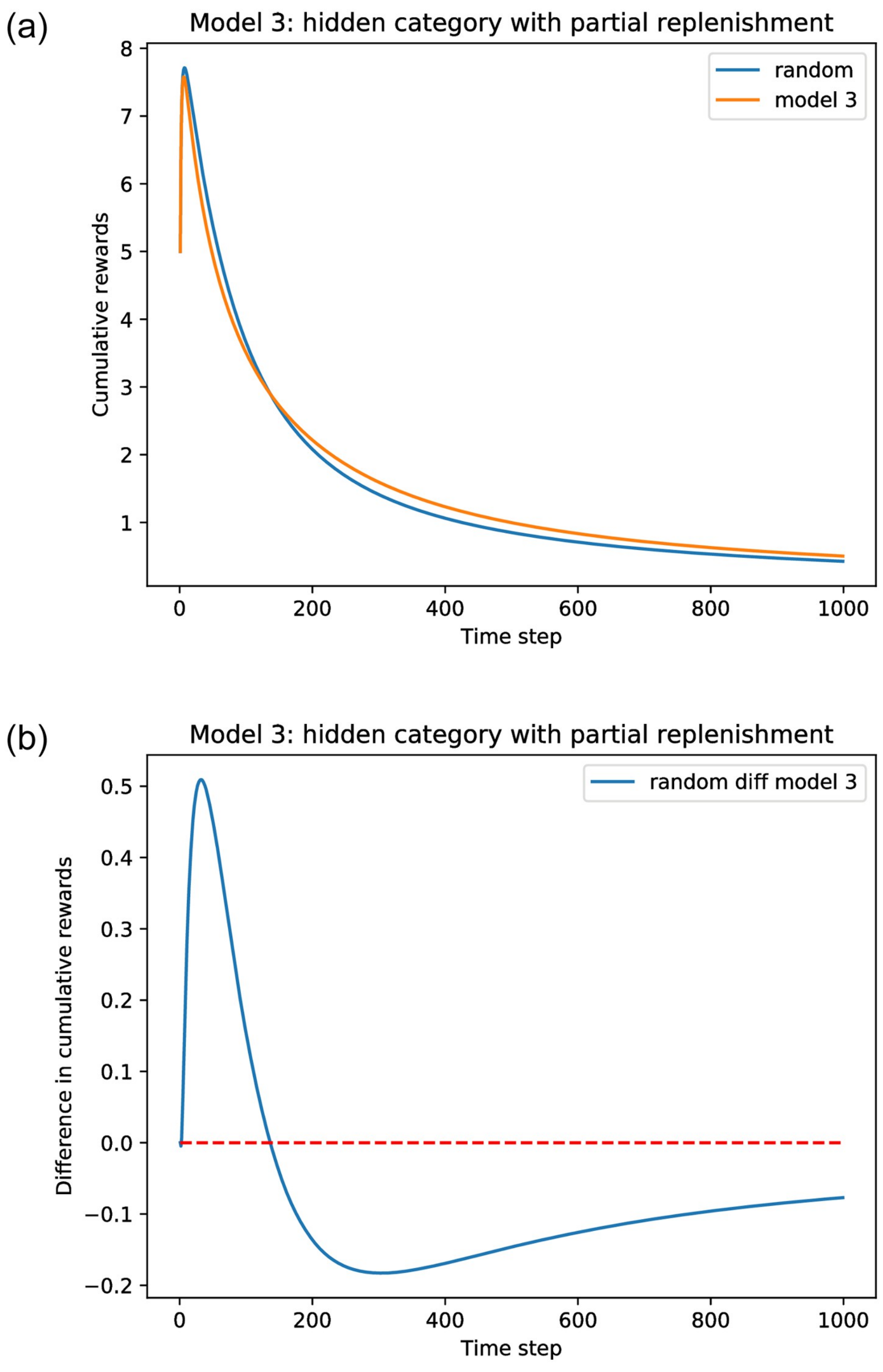


Fig 7. Simulation results comparing Model 3 with a random recommender. $A = 100$, list size $l = 20$, and $k = 0.01$. The original fraction $f_0(a) = 0.5$ and $\theta_0(a) = 0.5$ for all categories.

https://doi.org/10.1371/journal.pone.0271268.g007

However, it is now more natural to assume that the server replaces all ℓ items served: that is, it removes all $\ell_{t,a}$ movies for category a that were presented to this user and instead shows new movies of that category, regardless of whether the user liked them or not. The rationale is that the user will have already judged (and perhaps consumed) all items in the previous list.

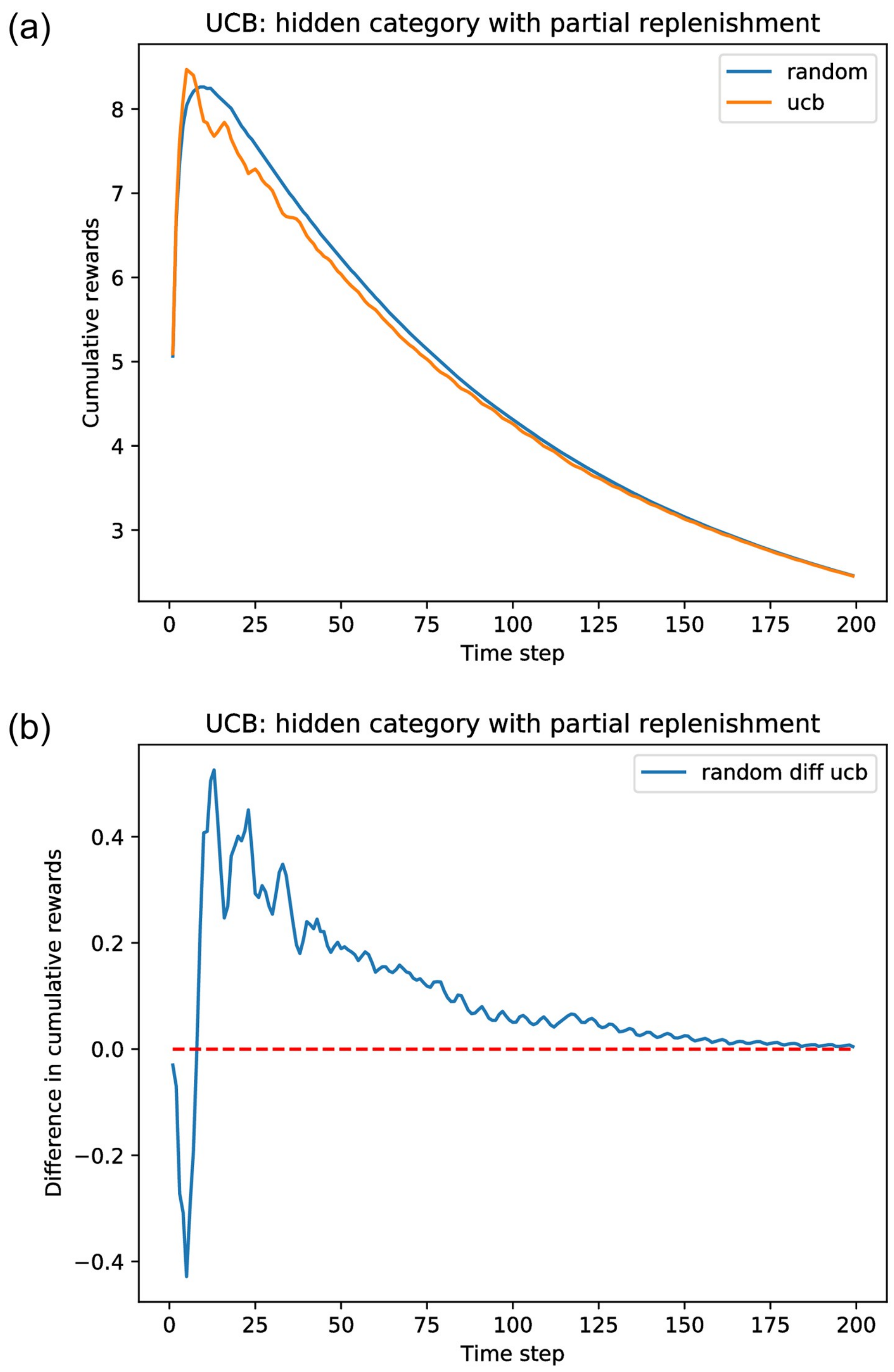


Fig 8. Simulation results comparing Model 3 under a UCB policy with a random recommender. A random recommender ignores user clicks when recommending categories; that is, $\theta_0(a)$ is never updated. $A = 100$, list size $l = 20$, and $k = 0.01$. The original fraction $f_0(a) = 0.5$ and $\theta_0(a) = 1.0$ for all categories. (a) Cumulative rewards. (b) Cumulative reward difference.

https://doi.org/10.1371/journal.pone.0271268.g008

Eqs (12) and (13) still hold, and Lemma 15 is still valid for this model. The only difference is in Eqs (16) and (17), because now the server will also replace not-α movies and it is likely that categories with small $f_t(a)$ increase their fraction of α-movies.

In this model we have:

$$f_{t+1}(a) = f_t(a) - \frac{C_t(a)}{M} + \frac{\zeta_t(a)}{M}, \tag{19}$$

where now $\zeta_t(a) \sim \text{Bin}(\ell_{t,a}, f_0(a))$ is the number of new movies that belong to hidden category α. In addition to Lemma 3.2 we will use the fact that $\mathbb{E}[\zeta_t(a) \mid \mathfrak{F}_t] = \ell \boldsymbol{p}_t(a) f_0(a)$.

We first address the case of the random recommender.

Proposition 3.3. *Assume that the recommended list is chosen at random (with replacement) from the A categories. Under Eq (19), for any category $a \in \mathcal{A}$ the stochastic process of the fractions $\{f_t(a), t = 0, 1, \ldots\}$ of α-movies in category a converges in distribution as $M \to \infty$ to the solution of the ordinary differential equation*

$$\frac{d}{dt}x(t) = f_0(a) - x(t), \tag{20}$$

which has a unique limit (as $t \to \infty$) and stable point $x^* = f_0(a)$.

Proof. The proof is in an Appendix.

Before we provide the intuitive interpretation of the result, we state without proof the following result (which follows from Theorem 6.1 of Vázquez-Abad and Heidergott [15]), which provides a Functional Central Limit Theorem for the target tracking process above.

Proposition 3.4. *Consider the model in Proposition 3.3. Fix the category a and call $U^{\in}(t) = \sqrt{\in}(x^{\in}(t) - f_0(a))$. Then as $\epsilon \to 0$, $U^{\epsilon}(\cdot)$ converges in distribution to the Orstein-Uhlenbeck process*

$$dU(t) = -U(t)\,dt + \sigma\,dW(t),$$

where $W(\cdot)$ is a standard Brownian motion and $\sigma^2 = \mathbb{V}\text{ar}[\zeta_n(a) - C_n(a) \mid f_n(a) = f_0(a))]$.

The interpretation of the two results above is that, for sufficiently small (but constant) ϵ, the process $\{f_n(a)\}$ hovers around the stable limit $f_0(a)$ and has approximately normally distributed errors around this limit. For our study, this means that the full replenishment policy targets a constant fraction of α-movies, so they do not deplete (with probability 1) as before.

We now turn to the discussion of the multinomial (smart) strategy under full replenishment. The process for this model is more complicated, following Eq (11) for the updates, Eq (10) to define the probability parameter for the multinomial, and Eq (19) for the update of fractions under full replenishment. It follows that

$$\mathbb{E}[\theta_{t+1}(a) \mid \mathfrak{F}_t] = \theta_t(a) + \ell \boldsymbol{p}_t(a) f_t(a); \quad \mathbb{E}[f_{t+a}(a) \mid \mathfrak{F}_t] = f_t(a) + \frac{\ell}{M} \boldsymbol{p}_t(a)(f_0 - f_t(a)).$$

Looking only at the evolution of the fractions for any given $a \in \mathcal{A}$, it is still the case that the only stable points correspond either to $\boldsymbol{p}_t(a) = 0$, yielding $f_t(a) = 0$ or $f(a) = f_0$. If the estimates are all initialized with non-zero values then $\theta_t(a) > 0$ always, excluding the first scenario. When $\boldsymbol{p}_t(a) > 0$ the dynamics of the fractions correspond to a target tracking algorithm with slowed down dynamics by the factor $\boldsymbol{p}_t(a)$. The behavior of the process of the fraction will be similar to the random case, albeit with different amplitude in oscillations around f_0. However it is notable here that the tendency of the smart algorithm is still to increase the likelihood of appearance of category a, while for this same category the likelihood of containing α-movies has decreased. Further, if a category a has few α-movies left at time t, it is likely that $C_t(a)$ is null (or smaller than for other categories served); but all of the movies served from that category get replenished by new movies, thus increasing the fraction $f_{t+1}(a)$, while the server has decreased its estimate for $\theta_t(a)$ thus decreasing the probability of serving this category at $t + 1$. This explains why the smart strategy is expected to perform even less well than the random recommender.

Fig 9 shows simulation results for the smart recommender with full replenishment. The tendency to return to the mean for the random recommender can be clearly seen. The smart

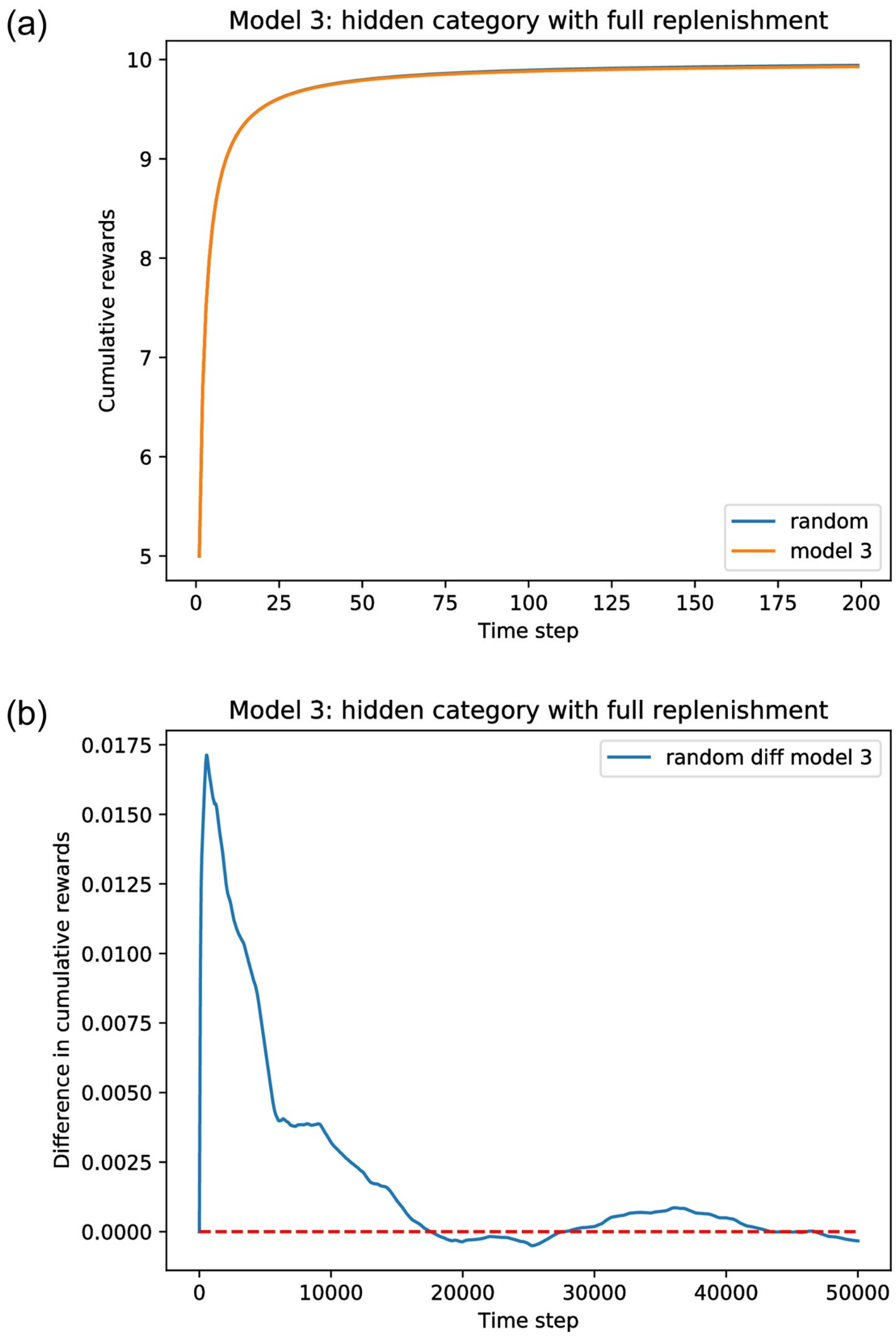


Fig 9. Simulation results for full replenishment, comparing Model 3 with a random recommender. $A = 100$, list size $l = 20$. The original fraction $f_0(a) = 0.5$ and $\theta_0(a) = 0.5$ for all categories. (a) Cumulative rewards. (b) Cumulative reward difference.

https://doi.org/10.1371/journal.pone.0271268.g009

recommenders still perform less well than the random. Although it is occasionally superior, this superiority is not persistent. Incompleteness of representation has, again, led to failures of recommendation that the system cannot observe.

As a final illustration, Fig 10 shows simulation results for the smart recommender under a UCB policy with full replenishment. The results are comparable to a random recommender.

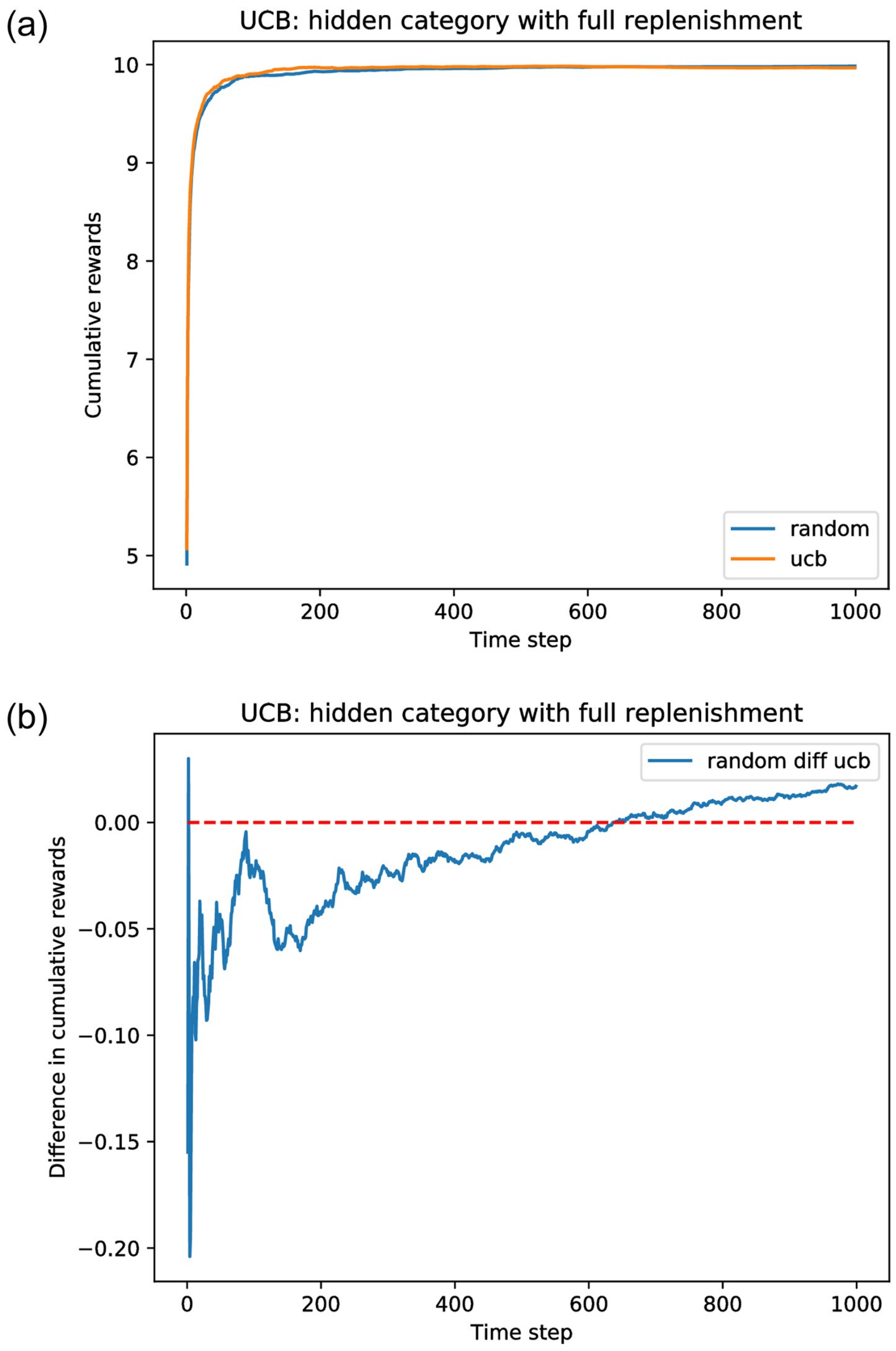


Fig 10. Simulation results for full replenishment, comparing Model 3 under UCB policy with a random recommender. $A = 100$, list size $l = 20$. The original fraction $f_0(a) = 0.5$ and $\theta_0(a) = 0.5$ for all categories. (a) Cumulative rewards. (b) Cumulative reward difference.

https://doi.org/10.1371/journal.pone.0271268.g010

4 Discussion and conclusions

We have explored the implications for proxy-driven machine learning of the situation where the real world exhibits behaviours that were not anticipated by the system designers. Using the relatively straightforward case of a single-user personalised recommendation system, we show that if the representation of the data—describing either the items being recommended or the user behaviour—is incomplete, then the resulting performance can be not just sub-optimal but no better than random.

In our case studies, we explore representational incompleteness in two ways, considering a user with unanticipated behaviour and the presence of unannotated categories. Under certain assumptions behaviour can even be worse than random, when for example the user's preferences and the system's inaccurate attempts to satisfy them are based on inconsistent premises that come into tension with each other.

The possibility of random or even worse-than-random behaviour may be unsurprising to some readers. However, it highlights two underlying issues that are widely neglected in exploration and deployment of machine learning methods. One is that the poor behaviour is invisible to the system; the metrics may well report that it is performing well. The other is that the effectiveness in practice of a machine learning method is limited by the completeness of the data representation. It is well known that erroneous data, or missing items, can lead to error; there is however no literature of which we are aware that explores the fact that the representation itself is a limitation.

Errors in data can in principle be corrected, and missing items replaced; but representations are innately incomplete and cannot be as rich as the entities they describe. Even the concept of category, which is crucial to the ability to gather 'like' items together, can be a poor fit to the natural world. That is, any learning algorithm applied to a rich human context has incomplete knowledge: it can only observe a pre-defined set of behaviours, make use only of specified features, and optimise only against known metrics. The algorithm cannot make use of or adapt to information of which it is unaware. As has been observed in other contexts, most notably with regard to Goodhart's law (see for example Manheim and Garrabrant [16]), the resulting performance can be absurdly divergent from what was intended. Our results illustrate that there is a risk that a learning system will fail, not due to poor design, but because there are inherent limitations to automated systems that engage with human processes, and the system will be unaware of the extent of the failure.

Wherever there is a representation gap, there is a potential for unobserved failure of unbounded magnitude. Understanding of when such failures are arising, and of how wide the gaps are between representation and reality, is critical to reliable deployment of machine learning systems.

5 Appendix: Proof of Proposition 3.1

First, given that the server chooses to explore at time t, an action that happens with probability $p_t > 0$, using basic combinatorial arguments,

$$\mathbb{P}(a^* \notin \boldsymbol{a}_t \,|\, \text{exploration}) = \frac{\binom{A-1}{\ell}}{\binom{A}{\ell}} = \frac{A-\ell}{A} \Rightarrow \mathbb{P}(a^* \in \boldsymbol{a}_t \,|\, \text{exploration}) = \frac{\ell}{A}.$$

Let $X_k = \mathbf{1}_{\{a^* \in \boldsymbol{a}_k\}}$. Then for all k such that $k < \nu$, $\mathbb{P}(X_k = 1) = \mathbb{P}(a^* \in \boldsymbol{a}_k \mid \text{exploration})\, p_k$. That is, it can be seen that $\mathbb{P}(X_k = 1) = p_k \frac{\ell}{A}$, because with probability $(1 - p_k)$ the system will recommend the top ℓ categories of θ_k and a^* is not there (we are considering $k < \nu$, so $\theta_k(a^*) = 0$). Because the variables $\{X_k; k = 1, 2, \ldots\}$ are independent, we have

$$\mathbb{P}(\nu > k) = \prod_{j=1}^{k} \mathbb{P}(X_j = 0) = \prod_{j=1}^{k} \left(1 - p_j \frac{\ell}{A}\right).$$

Finally, using the identity $\mathbb{E}[\nu] = \sum_{k=1}^{\infty} \mathbb{P}(\nu \geq k)$ we obtain the result (9).

6 Appendix: Proof of Proposition 3.2

Every time that the iteration corresponds to an exploration, items are chosen at random. As discussed, once a^* is chosen the estimate will be correct: $\theta_t(a^*) = 1$ for all $t > \nu$. Given the server algorithm, if an item $a' < a^*$ is served at time t, then $\theta_k(a') = 1$ for all $k > t$. This happens because the item is always presented before a^* and the click model assumes that all items are clicked until reaching a^* or the list is exhausted. Therefore, the server's update in (5) as the sample mean number of clicks for item a' will always remain at one unit. Because $p_t > 0$, eventually ℓ items with smaller labels than a^* will be chosen during exploration. Call this time τ. It is a dual of the 'surfacing' time because it is the time when a^* will no longer ever be served on exploitation iterations: all other smaller labels bump out the item a^*. That is, call $\tau = \min(t: \max_i(\boldsymbol{a}_{t,i}) < a^*)$, then by the server's policy and under the click model above, the top ℓ items according to θ_k, $k > \tau$ will not contain a^*. Because of random drawings, the stopping time τ is finite w.p.1.

7 Appendix: Proof of Proposition 3.3

For this proof we will use the notation n for discrete (iteration) times, and t for continuous time. Fix any category a, call $\epsilon = \ell/AM$, and re-write (19) as

$$f_{n+1}(a) = f_n(a) + \epsilon \left(\frac{\zeta_n(a) - C_n(a)}{\ell/A} \right),$$

which is a stochastic approximation algorithm known as 'target tracking', also called a 'return to the mean' recursion. To see this, note that for the random recommender, the conditional drift is $\mathbb{E}[f_{n+1}(a) - f_n(a) \mid \mathfrak{F}_n] = \in (f_0(a) - f_n(a))$. Intuitively, if $f_n(a) > f_0(a)$ then the expected change is negative, while if $f_n(a) < f_0(a)$, then it is positive. Define $x^\epsilon(t)$ as the piecewise constant interpolation (stochastic) process $x^\epsilon(t) = f_n(a)$, for all $t \in [n\epsilon, (n + 1)\epsilon)$. Let $m(t) = \lfloor t/\epsilon \rfloor$ denote the number of iterations up to time t. Then $x^\epsilon(t) = f_{m(t)}(a)$. The *feedback* $Y_n \equiv A(\zeta_n(a) - C_n(a))/\ell$ is a bounded random variable, because both $\zeta_n(a)$, $C_n(a) \leq \ell$ w.p.1. By our previous calculations, calling $\mathfrak{F}_n$ the history of the process up to iteration n, $\mathbb{E}[Y_n \mid \mathfrak{F}_n] = f_0(a) - f_n(a)$. The problem satisfies all the conditions in Theorem 4.3 of the Martingale noise model in Heidergott and Vázquez-Abad [15], which establishes the result. This reference is at present under revision by the publisher.

8 Appendix: Proof of Proposition 3.4

This appendix provides the proof of Proposition 3.4, following the techniques in Vázquez-Abad and Heidergott [15]. Uniform boundedness of $\{Y_n\}$ is a sufficient condition to ensure compactness of the stochastic processes $\{x^\epsilon(\cdot); \epsilon > 0\}$ in the probability sense, called 'tightness'. Thus every ϵ-sequence has a further weakly convergent (using the standard terminology, weak convergence is the same as convergence in distribution) subsequence and all limit points (in the sup-norm) are Lipschitz continuous processes w.p.1.

Consider one such weakly convergent subsequence with $\epsilon \to 0$ and call $\bar{x}(\cdot)$ the corresponding limit, which is Lipschitz continuous w.p.1. By construction, $x^{\in_k}(t+s) = x^{\in}(t) + \sum_{n=m(t)}^{m(t+s)} \in Y_n$. Call $\tilde{\mathfrak{F}}_t$ the σ-algebra (or history) of the continuous time process $x^{\epsilon}(\cdot)$. It follows that $\tilde{\mathfrak{F}}_t \equiv \mathfrak{F}_{m(t)}$. Conditioning on $\tilde{\mathfrak{F}}_t$ we obtain

$$\begin{aligned}
&\mathbb{E}[x^{\epsilon}(t+s) - x^{\epsilon}(t) \mid \tilde{\mathfrak{F}}_t] \\
&= \mathbb{E}\left[\sum_{n=m(t)}^{m(t+s)} \epsilon\,(f_0(a) - f_n(a)) \mid \mathfrak{F}_{m(t)}\right] \\
&\approx \mathbb{E}\left[\int_t^{t+s} (f_0(a) - x^{\epsilon}(u))\,du, \mid \tilde{\mathfrak{F}}_t\right]
\end{aligned}$$

where we have used the integral representation of the sum, and the approximation error is uniformly bounded and is of order $\mathcal{O}(\in)$. Define the process: $M^{\in}(t) \equiv x^{\in}(t) - x^{\in_k}(0) - \int_0^t (f_0(a) - x^{\in}(u))\,du$, then it follows that along the weakly convergent subsequence, the corresponding limit process $M(t) = \lim_{k\to\infty} M^{\epsilon}(t)$ is a w.p.1-Lipschitz continuous martingale, with initial value $M(0) = 0$, which implies that $M(t)$ is a constant w.p.1. This, along the limit of such convergent subsequence satisfies

$$\bar{x}(t+s) - \bar{x}(t) = \int_t^{t+s} (f_0(a) - \bar{x}(u))\,du.$$

which is equivalent to the solution of the ODE in Eq (20). This completes the proof, observing that this ODE has a unique solution for each initial condition, so this limit is independent of the chosen convergent subsequence. Using the standard compactness argument, this implies that the original sequence of stochastic processes converges in distribution to the solution of Eq (20).

Author Contributions

Conceptualization: Justin Zobel, Felisa J. Vázquez-Abad, Pauline Lin.

Formal analysis: Felisa J. Vázquez-Abad, Pauline Lin.

Methodology: Felisa J. Vázquez-Abad.

Supervision: Justin Zobel.

Visualization: Pauline Lin.

Writing – original draft: Justin Zobel, Felisa J. Vázquez-Abad, Pauline Lin.

Writing – review & editing: Justin Zobel, Felisa J. Vázquez-Abad, Pauline Lin.

References

1. Mitchell S, Potash E, Barocas S, D'Amour A, Lum K. Algorithmic fairness: Choices, assumptions, and definitions. Annual Review of Statistics and Its Application. 2021; 8(1):141–163. https://doi.org/10.1146/annurev-statistics-042720-125902
2. Haug L, Tschiatschek S, Singla A. Teaching Inverse Reinforcement Learners via Features and Demonstrations. In: Proceedings of the NeurIPS Conference on Neural Information Processing Systems. Montréal, Canada; 2018. Available from: https://proceedings.neurips.cc/paper/2018/file/4928e7510f45da6575b04a28519c09ed-Paper.pdf.

3. Suresh H, Guttag JV. A framework for understanding unintended consequences of machine learning. arXiv preprint arXiv:190110002v2. 2020;.
4. Danks D, London AJ. Algorithmic bias in autonomous systems. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. IJCAI'17. AAAI Press; 2017. p. 4691–4697.
5. Joseph M, Kearns M, Morgenstern J, Roth A. Fairness in learning: classic and contextual bandits. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS'16. Red Hook, NY, USA: Curran Associates Inc.; 2016. p. 325–333. Available from: https://proceedings.neurips.cc/paper/2016/file/eb163727917cbba1eea208541a643e74-Paper.pdf.
6. Calders T, Žliobaite I. Why unbiased computational processes can lead to discriminative decision procedures. In: Discrimination and Privacy in the Information Society: Data Mining and Profiling in Large Databases. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013. p. 43–57. Available from: https://doi.org/10.1007/978-3-642-30487-3_3.
7. Wang P, Wang J, Paranamna P, Shafto P. A mathematical theory of cooperative communication. In: Proceedings of the NeurIPS Conference on Neural Information Processing Systems. Vancouver, Canada; 2020. Available from: https://proceedings.neurips.cc/paper/2020/file/cc58f7abf0b0cf2d5ac95ab60e4f14e9-Paper.pdf.
8. Kaminskas M, Bridge D. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. ACM Trans Interact Intell Syst. 2016; 7(1). https://doi.org/10.1145/2926720
9. Jannach D, Jugovac M. Measuring the business value of recommender systems. ACM Transactions on Management Information Systems. 2019; 10(4). https://doi.org/10.1145/3370082
10. Chaney AJB, Stewart BM, Engelhardt BE. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In: Proceedings of the ACM Conference on Recommender Systems. RecSys'18. New York, NY, USA: Association for Computing Machinery; 2018. p. 224–232. Available from: https://doi.org/10.1145/3240323.3240370.
11. Jiang R, Chiappa S, Lattimore T, György A, Kohli P. Degenerate feedback loops in recommender systems. In: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society. AIES'19. New York, NY, USA: Association for Computing Machinery; 2019. p. 383–390. Available from: https://doi.org/10.1145/3306618.3314288.
12. Sutton RS, Barto AG. Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press; 2011.
13. Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem. Machine Learning. 2002; 47(2):235–256. https://doi.org/10.1023/A:1013689704352
14. Lai TL, Robbins H. Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics. 1985; 6(1):4–22. https://doi.org/10.1016/0196-8858(85)90002-8
15. Vázquez-Abad F, Heidergott B. Optimization and Learning via Stochastic Gradient Search. Priceton University Press; (under revision) 2022.
16. Manheim D, Garrabrant S. Categorizing variants of Goodhart's law. arXiv preprint arXiv:180304585v1. 2018.