Estimating and Penalizing Induced Preference Shifts in Recommender Systems

Micah Carroll [1] **Anca Dragan** [1] **Stuart Russell** [1] **Dylan Hadfield-Menell** [2]

Abstract

The content that a recommender system (RS) shows to users influences them. Therefore, when choosing a recommender to deploy, one is implicitly also choosing to induce specific internal states in users. Even more, systems trained via long-horizon optimization will have direct incentives to manipulate users, e.g. shift their preferences so they are easier to satisfy. We focus on induced *preference* shifts in users. We argue that – before deployment – system designers should: *estimate* the shifts a recommender would induce; *evaluate* whether such shifts would be undesirable; and perhaps even *actively optimize* to avoid problematic shifts. These steps involve two challenging ingredients: *estimation* requires anticipating how hypothetical policies would influence user preferences if deployed – we do this by using historical user interaction data to train a predictive user model which implicitly contains their preference dynamics; *evaluation* and *optimization* additionally require metrics to assess whether such influences are manipulative or otherwise unwanted – we use the notion of "safe shifts", that define a trust region within which behavior is safe: for instance, the natural way in which users would shift without interference from the system could be deemed "safe". In simulated experiments, we show that our learned preference dynamics model is effective in estimating user preferences and how they would respond to new recommenders. Additionally, we show that recommenders that optimize for staying in the trust region can avoid manipulative behaviors while still generating engagement.

1. Introduction

Recommender systems (RSs) generally show users feeds of items with the goal of maximizing their engagement, and users choose what to click on based on their preferences. Importantly, the recommender's actions are not independent of changes in users' internal states: simple changes in the content displayed to users can affect their behavior (Wilhelm et al., 2018; Hohnhold et al., 2015), mood (Kramer et al., 2014), beliefs (Allcott et al., 2020), and preferences (Adomavicius et al., 2013; Epstein & Robertson, 2015).

Given the dependence between users' internal state and the recommender system, when a system designer chooses a specific recommender algorithm (policy), they are implicitly also choosing how to influence user's behaviors, mood, preferences, etc. While traditionally RS policies have been *myopic* (tended at satisfying users' current desires), optimizing *long-term* user engagement has been a growing trend – typically via reinforcement learning (Afsar et al., 2021); these non-myopic policies are commonly referred to as long-term value, or LTV, systems. However, these policies will have incentives to manipulate users as a side-effect (Albanie et al., 2017; Krueger et al., 2020; Evans & Kasirzadeh, 2021): for example, certain preferences are easier to satisfy than others, leading to more potential for engagement – this could be because of availability of more content for some preferences compared to others, or because strong preferences for a particular type of content lead to higher engagement than more neutral ones. This can make LTV systems a particularly poor choice for avoiding undesired shifts in user's behaviors, moods, preferences, etc. While it has been proposed to prevent the RS from reasoning about manipulation pathways (e.g., by keeping it myopic) (Krueger et al., 2020; Farquhar et al., 2022), even such systems can influence users in systematically undesirable ways (Jiang et al., 2019; Mansoury et al., 2020; Chaney et al., 2018).

In this work, we focus on user's *preference* shifts, which is a particularly challenging problem that has been gaining more attention (Franklin et al., 2022). Any recommender policy will have some influence on user preferences. While this might seem cause for concern, the degree to which undesirable and manipulative preference influence occurs in practice has yet to be measured. Moreover, no framework has yet been proposed to *explicitly account for a policy's influence on users when choosing which policy to deploy*. We attempt to propose such a framework here – requiring us to tackle two critical problems.

Firstly, we need to anticipate what preference shifts a RS policy will cause *before it is deployed*. This alone would enable system designers to generate visualizations such as those in Fig. 1 (explained in more depth in our experiments):

[1] Berkeley [2] MIT. Correspondence to: <mdc@berkeley.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

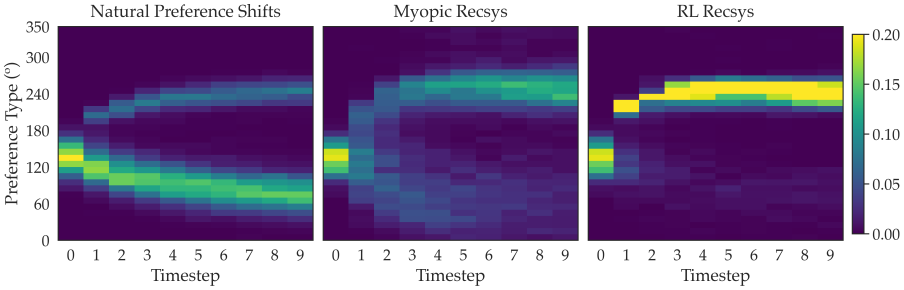

Figure 1: **Preferences induced by different RS policies across the same cohort of users.** In our simulated environment, preferences (y axis) are in 1D, and change over time (x axis) as users interact with the policy. On the right, the RL system drives preferences to one spot. A myopic policy (center) has a similar effect but less concentrated. These shifts are different from what we introduce as the "natural" evolution of user preferences.

these could surface whether certain RS policies tend to induce shifts that are significantly different from others, and allow to assess whether such shifts seem manipulative or otherwise unwanted. We study how to do this using historical user interaction data, and show how to train a predictive model that implicitly captures preference dynamics. Once trained, our model estimates a user's (latent) preferences induced by their interactions with a previous RS policy, and then uses that to estimate how that user's preferences would evolve under a new policy.

The second critical problem is having quantitative metrics for evaluating whether an evolution of preferences is actually unwanted: a computable metric not only simplifies evaluation, but could also enable recommenders to be optimized against unwanted shifts. The issue arises from the fact that *standard engagement metrics are preference-change-agnostic*: they do not directly assign value to preference shifts (or shifts in any other aspect of the user's state, for that matter). Even if a system were to completely overwrite a user's preferences, as long as the user is engaged, the system would be "performing well". We thus introduce some preliminary metrics which assign value to preference changes – based on how the users' preferences would have changed in the absence of the recommender – in an attempt to isolate the recommender's influence over the user's shifts.

For such metrics, instead of defining desirable or undesirable shifts directly, we provide a framework for conservatively defining "safe shifts": we non-exhaustively list certain shifts that we trust not to be particularly problematic, and measure the extent to which other shifts deviate from them. If the shifts induced by a policy differ significantly from the safe ones, they should be flagged as warranting more investigation. A candidate for safe shifts that we introduce is how user preferences would shift if they were interacting with a random recommender – which we call "natural preference shifts". One can think of this as an approximation to not having a recommender system at all. Fig. 1 (left) shows this natural preference evolution for our running example, and how user preferences stay somewhat diffuse but drift towards the opposite mode that the RL and myopic policies push them to. Note that while our metrics can effectively penalize undersired shifts, it comes at a cost: natural shifts, and in fact any lists of safe shifts that we define, are unlikely to be exhaustive, which means the approach will conservatively penalize policies that might in reality be safe.

To demonstrate both the estimation of preference shifts and their evaluation, we set up a testing environment in which we emulate ground truth user behavior by drawing from a model of preference shift from prior work (Bernheim et al., 2021). We first show that our learned preference shift estimation model – trained using historical user interaction data – can correctly anticipate user preference shifts almost as well as knowing the true preference dynamics. Additionally, we show qualitatively that in this environment, RL and even myopic recommenders lead to potentially undesired shifts. Further, we find that our evaluation metric can correctly flag which policies will produce undesired shifts, and evaluates the RL policy from Fig. 1 as 35% worse than the myopic one, which is in turn 40% worse than our policy which is penalized for manipulating user's preferences. Our results also suggest that evaluating our metric using the trained estimation model correlates to using ground truth preference dynamics, and that optimizing for safe shifts does lead to higher scoring (more safe) policies.

Although this work just scratches the surface of finding the right metrics for unwanted preference shifts and evaluating them in real systems, our results already have implications for the development of recommender systems: in order to ethically use recommenders at scale, we must take active steps to *measure* and *penalize* how such systems shift users' internal states. In turn, we offer a roadmap for how one might be able to do so by learning from user interaction data, and put forward a framework for specifying "safe shifts" for detecting and controlling such incentives.

2. Related Work

RS effects on users' internal states. A variety of previous work considers how RSs algorithms might affect users: influencing user's preferences for e-commerce purposes (Häubl & Murray, 2003; Cosley et al., 2003; Gretzel & Fesenmaier, 2006), altering people's moods for psychology research (Kramer et al., 2014), "nudging" users' opinions or behaviors (Jesse & Jannach, 2021; Matz et al., 2017;

Weinmann et al., 2015), exacerbate (Hasan et al., 2018) cases of addiction to social media (Andreassen, 2015), or increase polarization (Stray, 2021). There have been three main types of approaches to quantitatively estimating *effects of RSs' policies* on users: 1) analyzing static datasets of interactions directly (Nguyen et al., 2014; Ribeiro et al., 2019; Juneja & Mitra, 2021; Li et al., 2014), 2) simulating interactions between users and RSs based on hand-crafted models of user dynamics (Chaney et al., 2018; Bountouridis et al., 2019; Jiang et al., 2019; Mansoury et al., 2020; Yao et al., 2021; Ie et al., 2019a), or 3) using access to real users and estimating effects through direct interventions (Holtz et al., 2020; Matz et al., 2017). We see our approach as an improvement on 2), in that we propose to implicitly *learn* user dynamics instead of hand-specifying them. While we still assume a known choice model (how users choose content based on their preferences), such assumption is much less restrictive than assuming fully known dynamics. Our approach is most similar to Hazrati & Ricci (2022), but focused on preferences rather than behavior.

Neural networks for recommendation and human modeling. While data-driven models of human behavior have been used widely in real-world RS as click predictors (Zhang et al., 2019; Covington et al., 2016; Cheng et al., 2016; Okura et al., 2017; Mudigere et al., 2021; Zhang et al., 2014; Wu et al., 2017) and for simulating human behavior in the context RL RSs' training (Chen et al., 2019; Zhao et al., 2019; Bai et al., 2020; Shi et al., 2018), to our knowledge they have not been previously used for explicitly simulating and quantifying the *effect on users* of hypothetical recommenders. We emphasize how human models can also be used as simulation mechanisms by anticipating RSs' policies impact on users' *behaviors* and, as enabled by our method, even *preferences*.

RL for RS. Using RL to train RSs has recently seen a dramatic increase of interest (Afsar et al., 2021), with some notable work led by YouTube, Facebook, and others (Ie et al., 2019b; Chen et al., 2020; Gauci et al., 2019; Cai et al., 2022) – which have led to significant real-world performance increases.

Side effects and safe shifts. Our work starts from a similar question to that of the side effects literature (Krakovna et al., 2019; Kumar et al., 2020), applied to preference change: given that the reward function will not fully account for the cost (or value) of preference shifts, how do we prevent undesired preference-shift side effects? Our notion of safe shifts corresponds to the choice of baseline in this literature. While some ideas have been proposed to remove manipulation incentives (Farquhar et al., 2022), this requires having

3. Preliminaries

Setup. We model users as having time-indexed preferences $u_t \in \mathbb{R}^d$, which assign a scalar value to every possible item of content $x \in \mathbb{R}^d$: the value to the user derived from the engagement with item x_t under u_t is modeled as being given by $\hat{r}_t(u_t) = u_t^T x_t$. The user's preferences u_t together with other variables constitute the user's *internal state* z_t, which comprises a sufficient statistic for their long-term behavior, but which our method will not explicitly model. At every time step the user sees a slate s_t (a list of items) produced by a recommender policy π, and chooses an item x_t. The policy π maps history of slates and choices so far, $s_{0:t}, x_{0:t}$, to each new slate s_{t+1}: that is $s_{t+1} \sim \pi(s_{0:t}, x_{0:t})$. Upon making a choice, the user's internal state updates to z_{t+1}.

User choice model. We are interested in inferring and predicting preferences, which we do not observe directly. As such, we make an assumption – justified in Appendix B – about how user behavior (the user's choice x_t) relates to their current preferences u_t: that is, we assume the form of $\mathbb{P}(x_t = x | s_t, u_t)$ is known.

Preference evolution as an NHMM. The dynamics of the user's time-indexed internal state z_t – which we don't assume to be known – depend on the previous state z_{t-1} and on the last choice of slate by the recommender s_t (which in turn depends on history because the policy π uses history). This makes our setup a Non-homogeneous HMM (NHMM) (Hughes et al., 1999; Rabiner, 1986), with hidden state z_t and time-dependent dynamics $\mathbb{P}(z_{t+1}|z_t, s_t)$ (the choice of slate s_t by π will affect the future internal state). See Appendix D for more information. We will be using the NHMM inferences as an oracle benchmark for our methods' performance.

4. Estimating Users' Preferences

Our proposal boils down to the following: before deploying a new recommender policy π', we first need to be able to understand how that policy would change preferences and behavior of users. More formally, for a set of N users, we assume access to historical data of their interaction with a RS policy π: $\mathcal{D}_j = \{s^{\pi}_{0:T}, x^{\pi}_{0:T}\}$ for every user j. For any user, we are interested in estimating how their preferences would evolve if further interactions occurred with a new policy π', which we denote $u_H^{\pi'}$ (where $H > T$).

4.1. Estimation under known user dynamics

We first describe here how one could solve this problem optimally if one had oracle access to the true user internal state dynamics, and later relax this assumption.

Assuming the user internal state dynamics $\mathbb{P}(z_{t+1}|z_t, s_t)$ to be known, our goal is – as mentioned above – to estimate what their preferences would be at a future timestep H if policy π' were to be deployed going forward, that is, $u_H^{\pi'}$. Given the assumption of known internal state dynamics, every component of the NHMM is known, meaning that we can perform exact inference over $u_H^{\pi'}$. We can do so via a simple extension of HMM prediction: using the history

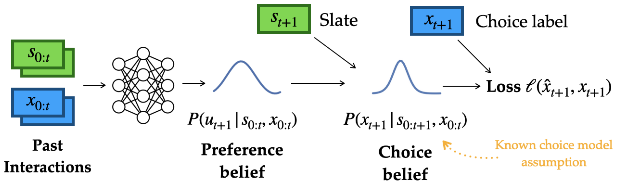
s_{t+1}
Slate
x_{t+1}
Choice label
$s_{0:t}$
$x_{0:t}$
Loss $\ell(\hat{x}_{t+1}, x_{t+1})$
$P(u_{t+1} \mid s_{0:t}, x_{0:t})$
$P(x_{t+1} \mid s_{0:t+1}, x_{0:t})$
Past Interactions
Preference belief
Choice belief
Known choice model assumption

Figure 2: **Future preference estimation model.** Given slates and choices up until the current timestep $s_{0:t}, x_{0:t}$, we train a network to predict beliefs over the next preferences u_{t+1} which – together with the next slate s_{t+1} – induce a distribution over content choices through the choice model. We can supervise the training with the actual choices users made for slates – the network will learn to output preference beliefs which induce similar choices to those in the data.

of interactions $(s^{\pi}_{0:T}, x^{\pi}_{0:T})$ to estimate $\mathbb{P}(z^{\pi'}_H | s^{\pi}_{0:T}, x^{\pi}_{0:T})$ as shown in Appendix E.1. As the preferences $u^{\pi'}_H$ are just one component of the internal state $z^{\pi'}_H$, one can trivially recover a posterior over the preferences specifically.

4.2. Estimation under unknown user dynamics

In practice, one will not have access to an explicit representation of the user's internal state z_t, let alone its dynamics. We thus attempt to approximate the NHMM estimation task we are interested in by implicitly learning user preference dynamics from their past interaction data. As seen in Fig. 2, we can train a neural network on $\{\mathcal{D}_j\}_{1:N}$ such that, given a user's past interaction data $(s^{\pi}_{0:t}, x^{\pi}_{0:t}$, where $t \leq T$), the model outputs a belief over the next-timestep preferences $\mathbb{P}(u_{t+1} | s^{\pi}_{0:t}, x^{\pi}_{0:t})$. Using the known choice model assumption from Sec. 3, we map the belief over preferences, together with the new slate s_{t+1}, to a distribution over content items: $\mathbb{P}(x_{t+1} | s_{0:t+1}, x_{0:t}) = \sum_{u_{t+1}} \mathbb{P}(x_{t+1} | u_{t+1}, s_{t+1}) \mathbb{P}(u_{t+1} | s_{0:t}, x_{0:t})$.

We now have a model which approximates the inference $\mathbb{P}(u_{t+1} | s^{\pi}_{0:t}, x^{\pi}_{0:t})$ for any $t \leq T$ – that is, it provides us beliefs over the preferences u_{t+1} resulting from the user's interactions so far with π. However, for the problem we stated at the beginning of Sec. 4, we are interested in estimating the preferences that would be induced if a different policy π' were to be deployed going forward after π: formally, we are interested in $\mathbb{P}(u^{\pi'}_H | s^{\pi}_{0:T}, x^{\pi}_{0:T})$ where $H > T$.

To obtain a belief over $u^{\pi'}_H$ for a user j, we can sample *simulated user preference trajectories* from the model to obtain a Monte Carlo estimate of the desired distribution – as shown in Fig. 3. We use the data $\mathcal{D}_j = \{s^{\pi}_{0:T}, x^{\pi}_{0:T}\}$ as input to the model to obtain a belief over the next-timestep preferences u_{T+1}. We then use such belief and a slate $s^{\pi'}_{T+1}$ sampled from policy π' to simulate the user's choice $x^{\pi'}_{T+1}$. Treating this extra (simulated) step of interaction history for the user (the slate $s^{\pi'}_{T+1}$ and choice $x^{\pi'}_{T+1}$) as part of the observed history so far, we repeat this process to obtain preference estimates under π' for future timesteps. By simulating multiple future trajectories, we can obtain beliefs over the expected preferences a user would have at any future timestep. See Algorithm 1 for the exact procedure. In Appendix E.1 we show that this procedure does in fact approximate the desired estimate.

One challenge with this method is that it requires the learned preference predictor model to generalize to histories collected under π', even though no training data was collected with π'. We believe that there is still reason to be hopeful given that real-world datasets of user interaction are usually collected under many different deployed policies. This suggests that as long as the histories induced by π' are not too dissimilar from those in the historical data used for training, the network should be able to generalize to predict preference evolutions induced by π' too.

5. Quantifying unwanted shifts and optimizing to avoid them

Given the estimates of preference shifts under different policies that Sec. 4 enables us to compute, it would be useful to have quantitative metrics for whether they are undesirable: this would enable to automatically evaluate policies and even actively optimize to avoid such shifts. To obviate the difficulty of explicitly defining unwanted shifts, we limit ourselves to defining some shifts that we somewhat trust – "safe shifts" – and flag preference shifts that largely differ from these safe shifts as potentially unwanted. This is similar to defining a region of the space which we trust *in a risk-averse manner*. The underlying philosophy is that "we don't know what good shifts are, but as long as you stay close to [family of safe shifts], things shouldn't go too bad". To do this, we need both a notion of shifts we trust ("safe shifts"), and a notion of distance between different evolutions of preferences.

Notation. We denote the π-induced engagement under "safe-shift preferences" u^{safe}_t as $\hat{r}_t(u^{\text{safe}}_t, \pi) = (x^{\pi}_t)^T u^{\text{safe}}_t$. This means that π was used to select the slate from which the user picked the item x^{π}_t, but x^{π}_t's engagement is considered relative to different preferences u^{safe}_t than those the user would have developed under π.

Distance between shifts. Consider a known preference-trajectory $u^{\pi}_{0:T}$ induced by π. We choose a metric between shifts such that engagement for the items $x^{\pi}_{0:T}$ chosen under policy π is also high under the preferences one would have had under safe shifts $u^{\text{safe}}_{0:T}$. We operationalize this notion of distance between π-induced shifts and safe shifts $u^{\text{safe}}_{0:T}$ as $D(u^{\pi}_{0:T}, u^{\text{safe}}_{0:T}) = \sum_t \mathbb{E}\big[\hat{r}_t(u^{\pi}_t, \pi) - \hat{r}_t(u^{\text{safe}}_t, \pi)\big] = \sum_t \mathbb{E}_{x^{\pi}_t}\big[(x^{\pi}_t)^T u^{\pi}_t - (x^{\pi}_t)^T u^{\text{safe}}_t\big]$. However, this operationalization could easily be substituted with others. In the case that safe shifts are random variables (a family of safe shifts), we consider the expected distance $\mathbb{E}_{U^{\text{safe}}_{0:T}}\big[D(u^{\pi}_{0:T}, U^{\text{safe}}_{0:T})\big]$.

Safe shifts: initial preferences. As a first (very crude) proposal for safe shifts, we can consider using the initial preferences u_0 as our safe shifts $u^{\text{safe}}_{0:T}$: any deviation from

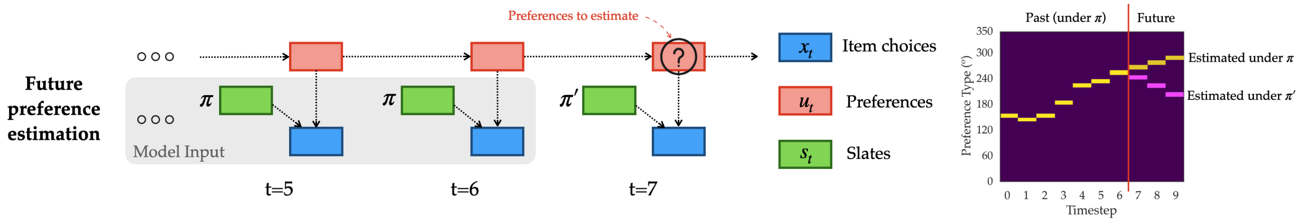

Figure 3: **Simulating future user preferences and choices.** By iteratively using the future preferences estimation model (Fig. 2) by inputting the observables (shaded, e.g. $x^{\pi}_{0:6}, s^{\pi}_{0:6}$), one can simulate how an existing user's preferences would evolve *in the future* if they interacted with same policy π or a different policy π' (by selecting future slates with π').

the initial preferences u_0 will be flagged as potentially problematic with $D(u^{\pi}_{0:T}, u_0)$.

Safe shifts: natural preference shifts (NPS). One limitation of the above metric is that not all preference shifts are unwanted: people routinely change their preferences and behavior "naturally". But what does "naturally" even mean? We propose an idealized notion of natural shifts, by asking how preferences would evolve if the user were "omniscient" and have full agency over their preference evolution process, i.e. if would have access to all content directly and had the ability to process it, unhindered by a small and biased slate offered by the RS. Unfortunately this is impractical, as we can never get data from such hypothetical users that can attend to all content when choosing what to consume. As an approximation, we consider *random* slates, which at least eliminates the agency of the RS policy (which in turn can change the user's belief about the distribution of available content). We therefore operationalize "natural preferences" $u^{\pi_{\text{rnd}}}$ as the preferences which the user would have interacting with a random recommender π^{rnd}, and we use $\mathbb{E}_{U^{\text{rnd}}_{0:T}}\left[D(u^{\pi}_{0:T}, U^{\pi_{\text{rnd}}}_{0:T})\right]$ as the metric.

Using safe shift metrics. For how one can estimate such metrics, see Appendix F. Once computed, they can be used both for evaluating preference shifts or for recommender optimization (as alternate proxies for "value" relative to simple engagement). However, clearly such metrics also fall short of fully capturing "value": we don't want a system that actively tries to keep the user preferences static (as would result from blindly optimizing $D(u^{\pi}_{0:T}, u_0)$).

Penalized objective. By considering a weighted sum of these metrics (as we will do in the penalized RL objective below), we try to lead the system to perform well under a variety of relatively-reasonable definitions of value, some of which are not preference-shift-agnostic. One can think of this as hedging our bets as to what the value (or cost) of induced preference shifts should be, and making explicit that it should not be zero, as current systems assume.

Penalized RS training. By using the method from Sec. 4, one obtains a human model which can be used to *simulate human choices* (in addition to preferences). Similarly to previous work, we can use this human model as a simulator for RL training of recommender systems (Chen et al., 2019; Zhao et al., 2019; Bai et al., 2020). During training, we can compute the metrics defined above penalize the current policy for causing any shifts that we have not explicitly identified as "safe" – in what can be considered a "risk-averse" design choice. We incorporate these metrics in the training of a new policy π by adding the two distance metrics from above to the basic objective $\mathbb{E}\left[\hat{r}_t(u^{\pi}_t)\right]$ of maximizing long-term engagement, leading to the updated objective $\mathbb{E}\left[\hat{r}_t(u^{\pi}_t) + \nu'_1\, \hat{r}_t(u_0, \pi) + \nu'_2\, \hat{r}_t(u^{\pi_{\text{rnd}}}_t, \pi)\right]$ where ν'_1, ν'_2 are hyperparameters. This objective can be optimized either myopically or via long-horizon (RL) optimization. See Appendix G for more details.

6. Experimental Setup

Why simulation? To test our method, we need to evaluate both recommenders which interact with users (rendering static datasets of user interaction unsuitable), as well as test our evaluation metrics themselves – which are defined based on internal preferences (for which we never get ground truth in real interactions). We thus create a testing environment in which we can emulate user behavior and have access to their ground truth preferences. Like previous work (Chaney et al., 2018; Bountouridis et al., 2019; Jiang et al., 2019; Mansoury et al., 2020; Yao et al., 2021), we simulate both a recommendation environment and human behavior. However, unlike such approaches, we use simulated human behavior for evaluation purposes only: our human model is learned exclusively from data that would be observed in a real-world RS (slates and choices), i.e. data of users (in our case the simulated users) interacting with previous RS policies – meaning our approach could be applied to real user data of this form. A fundamental advantage of testing our method in a simulated environment is that *we can actually evaluate how well our model is able to recover the preferences of our "ground truth" users, giving us insights about how our methods could perform with real users.*

Ground truth human dynamics. See Fig. 4 for a summary of the ground truth human dynamics we use for testing our method (and Fig. 12 for more info on our environment setup). Following prior work (Ie et al., 2019b; Chen et al., 2019) we assume that users choose items in proportion to (the exponentiated) engagement under their current preferences, i.e. that $\mathbb{P}(x_t = x | s_t, u_t)$ is given by the conditional logit model. In our setup, this reduces to

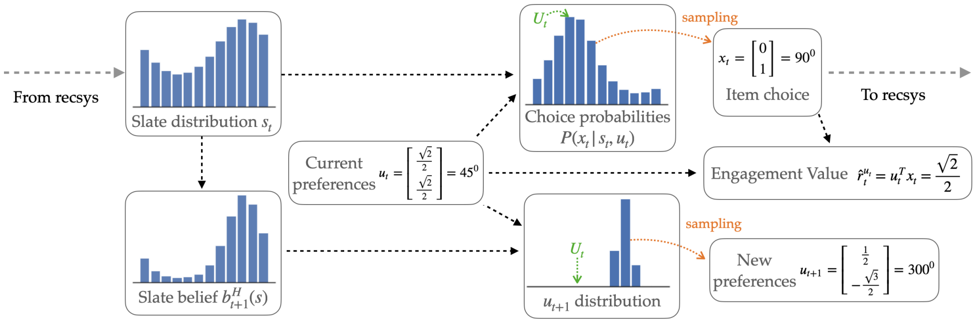
From recsys
Slate distribution s_t
Slate belief $b^H_{t+1}(s)$
Current preferences $u_t = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} = 45^0$
U_t
Choice probabilities $P(x_t | s_t, u_t)$
sampling
$x_t = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 90^0$
Item choice
To recsys
Engagement Value $\hat{r}^{u_t}_t = u_t^T x_t = \frac{\sqrt{2}}{2}$
U_t
u_{t+1} distribution
sampling
New preferences $u_{t+1} = \begin{bmatrix} \frac{1}{2} \\ -\frac{\sqrt{3}}{2} \end{bmatrix} = 300^0$

Figure 4: **Ground truth human dynamics.** At each timestep, the user will receive a slate s_t. Given the user's preferences u_t, the slate s_t induces a distribution over item choices $\mathbb{P}(x_t|s_t, u_t)$ from which the user samples an item x_t and receives an engagement value $\hat{r}_t$ (unobserved by the RS). Additionally, s_t induces a belief over the future slates in the user $b^H_t(s)$. In turn $b^H_t(s)$ – together with u_t – induce a distribution over next timestep user preferences, from which u_{t+1} is sampled.

$\mathbb{P}(x_t = x|s_t, u_t) \propto \mathbb{P}(s_t = x)e^{\beta_c x^T u_t}$, with an additional term $\mathbb{P}(s_t = x)$ which takes into account how prevalent each item is in the slate – and we assume such choice model to be also known by our method. We adapt (Bernheim et al., 2021) to be our ground truth human preference dynamics. On a high-level, at each timestep users choose their next-timestep preferences to be more "convenient" ones – trading off between choosing preferences that they expect will lead them to higher engagement and maintaining engagement under current preferences. See Appendix H.1 for a proof of the logit model reduction, and more information about the ground truth human.

Simulated environment setup. For ease of interpretation of the results, in our experiments we only consider a cohort of users whose initial preferences are concentrated around preference $u = 130°$. To showcase preference-manipulation incentives to make users more predictable, we make the choice-stochasticity temperature β_c a function of part of preference space one is in, with local optima $\beta_c(80°) = 1$ and $\beta_c(270°) = 4$. This causes these areas in preference space to be attractor points, as RSs are able to lead to higher engagement when users act less stochastically (see Appendix H.1 for more details).

Dataset. For all our experiments, we use a dataset of 10k trajectories (each of length 10), collected under a mixture of policies described in Appendix H.3. 7.5k trajectories are used for training our models and 2.5k for computing the validation losses and accuracies reported in Sec. 7.1.

Training human models and penalized RS policies. For training our human models, we use a bidirectional transformer model similar to that of Sun et al. (2019), and only assume access to a dataset of historical interactions $s_{0:T}, x_{0:T}$. We train myopic and RL RS policies π' using PPO (Schulman et al., 2017; Liang et al., 2018) and restrict the action space to 6 possible slate distributions for ease of training. For penalized training we give the three metrics $\hat{r}_t(u^\pi_t)$, $\hat{r}_t(u_0, \pi')$, $\hat{r}_t(u^{\pi_{\text{rnd}}}_t, \pi')$ equal weight. For more details on human model and RL training, see respectively Appendix H.2 and Appendix G.

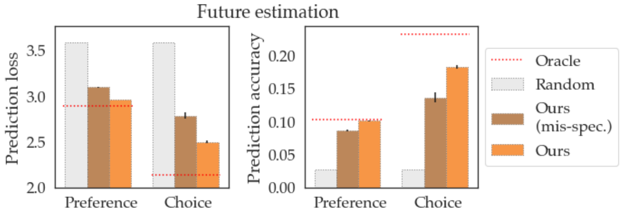

Figure 5: Validation losses and accuracies on held-out trajectories for the preference prediction task, averaged across timesteps. Both under the correct choice model and with some mis-specification, preference prediction performs similarly to oracle NHMM estimation (which additionally has access to the preference dynamics).

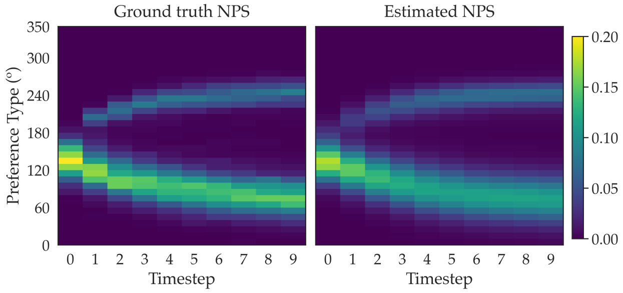

Figure 6: Natural Preference Shifts (induced by π^{rnd}) among a cohort of 1000 users (left) vs. a Monte Carlo estimate using 1000 simulated user interaction trajectories obtained with our model and Algorithm 1 (right).

7. Results

In Sec. 7.1 we validate that our method from Sec. 4 can estimate user's preferences and predict their evolution under alternate policies; then, in Sec. 7.2, we turn to validating the metrics and penalized RL training approach from Sec. 5.

7.1. Estimating user preferences

Oracle baseline. We use the NHMM estimates computed with full access to the human preference dynamics as a baseline for our preference and behavior estimates. Given that we can compute them through exact inference, such estimates are the best one could possibly hope for.

Estimating preferences. We first verify that, given snip-

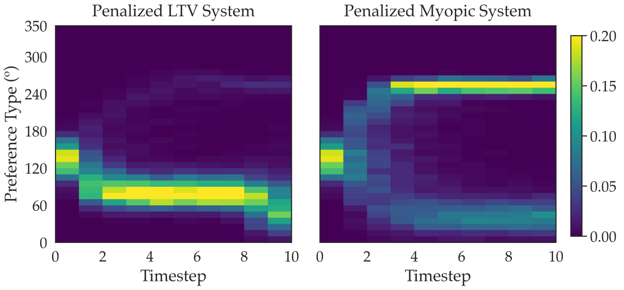

Figure 7: With penalized RL training, the preferences induced by the learned policies are closer to the NPS and initial preferences, as desired. Myopic systems will only greedily be pursuing the penalized objective, leading to somewhat arbitrary behavior.

pets of past interactions from the validation trajectories, the preference prediction model (Sec. 4) is able to predict the next timestep preferences (and observations, as induced by the predicted preferences). For each interaction sequence in the validation set we estimate the tth timestep preferences and observations based on the previous interactions (for all $1 \leq t \leq 10$). We also perform this same inference on the same data with a random predictor, and with our oracle baseline: exact NHMM inference. We report the average values for the prediction losses and the accuracies in Fig. 5.

Robustness to mis-specified choice models. Our method requires the specification of a user choice model, which will be challenging in practice. In Fig. 5 we also show how the quality of our estimates withstands a mis-specified choice model (described in Appendix H.2). We found that validation loss for choice predictions (which would be observed in real-world experiments, unlike the preference prediction loss and accuracy) was a good indicator of the amount of model mis-specification: reducing choice model mis-specification reduces choice prediction validation loss. This could be used to guide the design of better choice models.

Imagining Natural Preference Shifts (NPS). In the experiment above, we were validating our model on data from the same distribution (and RS policy) as during training. To test the performance further, we try to now see whether our model is able to estimate how alternate policies would affect the evolution of user preferences on a population level, if they were to be deployed (for which we had no data at training time). We use Algorithm 1 to simulate preference trajectories of users interacting with π^{rnd} (setting $T = 0$, i.e. without considering any past interaction context). We then average the predicted preference distributions across the simulated users, giving us estimated preference evolutions similar to the actual ones induced by π^{rnd} (i.e. the Natural Preference Shifts) – shown in Fig. 6. This shows that in our experimental setup, our method is able to capture the preference dynamics of a user in ways that generalize to alternate policies; that is, the model is able to estimate preference evolutions under different RS policies from the ones from which the training interaction data was obtained.

7.2. Evaluating undesirable shifts and penalizing them

Qualitative effects of different policies. Firstly, we show the qualitative effect of using different classes of RS policies in our simulated setup in Fig. 1. We see how interacting with the *unpenalized* RL policy drives users strongly away from their initial preferences and concentrates them in a specific region of preference space (in bright yellow) – in a behavior that seems potentially concerning (what if the preference type were a political axis?). The myopic policy (center), shows the same type of effect but much less pronounced.

Safe shift metrics and sum of rewards. As delineated in Sec. 5, we use the initial preferences u_0 and Natural Preference Shifts $u_{0:T}^{\pi^{\text{rnd}}}$ of a user as our "safe shift" proxies, which give us alternate evaluations of trajectories $\hat{r}_t(u_0, \pi')$ and $\hat{r}_t(u_t^{\pi^{\text{rnd}}}, \pi')$ relative to simple current-preference engagement $\hat{r}_t(u_t^{\pi'})$. We consider the sum of these metrics as a better proxy for "true value" than any one of these metrics alone. π' here denotes the trained policy.

Hypotheses about metrics. We now turn to our hypotheses regarding metrics. We hypothesize that our metrics are able to: (**H1**) identify whether policies will induce potentially unwanted preference shifts, and (**H2**) incentivize better behavior when added as a penalty during training.

Learned human model confound. To validate **H1** with real users, one would have to rely on an approximate computation of the metrics (**estimated evaluation**) with a learned user models of preference dynamics (as computing the metrics requires estimating preferences). Additionally, to validate **H2**, one would likely train with simulated interactions from the learned user model (**training in simulation**) – as explained in Sec. 5. The quality of the learned user model would thus affect the evaluation of the quality of the choice of metrics themselves – which would constitute a confound.

H1 under oracle dynamics access. In order to deconfound our experiments from the errors in our estimated human dynamics, we first test our hypotheses assuming oracle access to users and their dynamics for the purposes of training and evaluation. We first hypothesize that (**H1.1**) by computing the metrics exactly using the preference estimates obtained through NHMM inference (**oracle evaluation**), our metrics are able to flag potentially unwanted preference shifts. We find this to be the case by comparing the oracle evaluation metric values (left of Table 1, "unpenalized") and the actual preference shifts induced by the various RSs we consider (Fig. 1): while unpenalized RL performs better (7.49 vs 5.71) than myopic for engagement $\hat{r}_t(u_t^{\pi'})$, it performs worse with respect to our safe shift metrics. This matches Fig. 1, where RL has more undesired effects.

H2 under oracle dynamics access. We additionally hypothesize that (**H2.1**) such metrics (still computed exactly, in **oracle evaluation**) can be used for training penalized LTV systems which avoid the most extreme unwanted preference shifts. For this training, allow ourselves on-policy human interaction data with the ground truth users, as if the RL happened directly in the real world – we call this **oracle training**. For the *penalized* RL RS, the cumulative metric value ("Sum" in Table 1) increases substantially, although it is at the slight expense of instantaneous engagement (Table 1). Qualitatively, we see that the induced preference shifts caused by the RL system seem closer to "safe shifts" (Fig. 7), supporting **H1.1** in that high metric values qualitatively match shifts that seem more desirable.

Overall, we see that with oracle access, the metrics capture what we see as qualitatively undesired shifts, and that using them for penalized RL produces policies with slightly lower engagement, but drastically better at avoiding such shifts.

H1 and H2 with learned user dynamics. In practice we would not have access to the underlying user dynamics, or potentially even the ability to interact on-policy with users as we train RL policies, due to the high cost and risk of negative side effects for collecting data with unsafe policies. Therefore, we wish to show that even with *estimated metrics and simulated interactions* (based on the learned models), (**H1.2**) **estimated evaluation** and (**H2.2**) **training in simulation** (described above) are still able to respectively flag unwanted shifts and penalize manipulative RL behaviors. Table 2 shows that – although the estimated metrics can differ from the ground truth ones somewhat substantially (see Oracle Eval. at the top vs Estimated Eval. at the bottom) – importantly the relative ordering of the policies ranked by our estimated values stay the same: the penalized RL policy trained in simulation actually has (under oracle evaluation) higher cumulative reward than the unpendalized one, and the estimated evaluation keeps the same ranking (even though is more optimistic about the unpenalized reward).This provides some initial evidence that, even without observing preferences in practice, one could successfully optimize – and assess – recommenders that penalize preference-shifts.

8. Discussion

Summary. In conclusion, our contributions are: 1) proposing a method to estimate the preference shifts which would be induced by recommender system policies before deployment; 2) a framework for defining safe shifts, which can be used to evaluate whether preference shifts might be problematic; 3) showing how one can use such metrics to optimize recommenders which penalize unwanted shifts. As dynamics of preference are learned (rather than handcrafted), our method has hope to be applied to real user data. While there is no ground truth for human preferences, verifying the model's ability to anticipate behavior can give confidence in using it to evaluate and penalize undesired preference shifts. We acknowledge that this is only a first step, tested in an idealized setting with relatively strong assumptions. However, we hope this can be a starting point for further research which focuses on relaxing such assumptions and making these ideas applicable to the complexity of real RSs.

Limitations. To validate our estimation method and metrics, we required ground truth access to user preferences – leading us to conduct our experiments in simulation. While the ground truth model of dynamics we use is inspired by the econometrics literature, we cannot guarantee that our results would translate when the method is applied to real users with real preference dynamics. We expect real users to have more complex dynamics, with preference changes mediated by beliefs other than just about the distribution of content, such as beliefs about the world; while our method makes no assumptions about the structure of this internal space, it might require a lot more (and more diverse) data to capture these effects. Further, we also chose the experimental setup such that an RL system would act on its incentives to manipulate preferences – not all real settings will necessarily have that property. Moreover, crucially our method requires designing a user choice model: one could obviate this by predicting behavior and defining metrics on behavior, but this would mean losing the latent preference structure. Additionally, even what we call "preferences" – a latent for instantaneous behavior – is limiting as it doesn't lend itself to capturing long-term preferences, and assumes that there is such a thing as fixed preferences (Ariely & Norton, 2008).

Table 1: **Results under oracle training and evaluation.** Both here an in Table 2 we report the cumulative reward under various metrics, averaged across 1000 trajectories (standard errors are all < 0.1). By looking at the safe shift metrics $\hat{r}(u_0)$ and $\hat{r}(u_t^{\text{rnd}})$, we see that penalized systems stay significantly closer to safe shifts than unpenalized ones.

		Oracle Training			
		Unpenalized		*Penalized*	
		Myopic	RL	Myopic	RL
Oracle Eval	$\hat{r}_t(u_t^{\pi'})$	5.71	7.49	6.20	5.28
	$\hat{r}_t(u_0, \pi')$	1.99	-0.08	3.61	6.21
	$\hat{r}_t(u_t^{\pi_{\text{rnd}}}, \pi')$	2.01	-1.09	3.10	4.57
	Sum	9.69	6.33	12.90	16.05

Table 2: **Effect of estimating evaluations and simulating training for LTV.** We see that the estimated evaluations of trained systems strongly correlate with the oracle evaluations (and importantly, maintain their relative orderings). A similar effect occurs when training in simulation rather than by training with on-policy data collected from real users.

		Oracle Training		**Training in Sim.**	
		Unpen.	*Penal.*	*Unpen.*	*Penal.*
Orac. Eval	$\hat{r}_t(u_t^{\pi'})$	7.49	5.28	6.40	5.48
	$\hat{r}_t(u_0, \pi')$	-0.08	6.21	-1.24	5.61
	$\hat{r}_t(u_t^{\pi_{\text{rnd}}}, \pi')$	-1.09	4.57	-1.83	4.43
	Sum	6.33	16.05	3.36	15.52
Est. Eval	$\hat{r}_t(u_t^{\pi'})$	5.58	5.42	6.49	5.78
	$\hat{r}_t(u_0, \pi')$	1.28	5.57	-0.80	4.94
	$\hat{r}_t(u_t^{\pi_{\text{rnd}}}, \pi')$	2.05	3.88	1.48	4.41
	Sum	8.91	14.87	7.17	15.15

ACKNOWLEDGMENTS

This work was funded by ONR YIP. We thank David Krueger, Jonathan Stray, Stephanie Milani, Lawrence Chan, Scott Emmons, Smitha Milli, Aditi Raghunathan, Orr Paradise, Kenneth Lai, Gloria Liu, Emily Chao, and Mihaela Curmei for helpful discussions at various stages of the project.

References

Adomavicius, G., Bockstedt, J. C., Curley, S. P., and Zhang, J. Do Recommender Systems Manipulate Consumer Preferences? A Study of Anchoring Effects. *Information Systems Research*, 24(4):956–975, December 2013. ISSN 1047-7047. doi: 10.1287/isre.2013.0497. URL https://pubsonline.informs.org/doi/10.1287/isre.2013.0497. Publisher: INFORMS.

Afsar, M. M., Crump, T., and Far, B. Reinforcement learning based recommender systems: A survey. *arXiv:2101.06286 [cs]*, January 2021. URL http://arxiv.org/abs/2101.06286. arXiv: 2101.06286.

Albanie, S., Shakespeare, H., and Gunter, T. Unknowable Manipulators: Social Network Curator Algorithms. *arXiv:1701.04895 [cs, stat]*, January 2017. URL http://arxiv.org/abs/1701.04895. arXiv: 1701.04895.

Allcott, H., Braghieri, L., Eichmeyer, S., and Gentzkow, M. The Welfare Effects of Social Media. pp. 121, 2020.

Andreassen, C. S. Online Social Network Site Addiction: A Comprehensive Review. *Current Addiction Reports*, 2 (2):175–184, June 2015. ISSN 2196-2952. doi: 10.1007/s40429-015-0056-9. URL https://doi.org/10.1007/s40429-015-0056-9.

Ariely, D. and Norton, M. I. How actions create–not just reveal–preferences. *Trends Cogn Sci*, 12(1):13–16, January 2008. ISSN 1364-6613. doi: 10.1016/j.tics.2007.10.008. URL https://dukespace.lib.duke.edu/dspace/handle/10161/6219. Accepted: 2013-02-25T17:20:35Z Publisher: Elsevier BV.

Bai, X., Guan, J., and Wang, H. Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation. *arXiv:1911.03845 [cs, stat]*, January 2020. URL http://arxiv.org/abs/1911.03845. arXiv: 1911.03845.

Bernheim, B. D., Braghieri, L., Martínez-Marquina, A., and Zuckerman, D. A theory of chosen preferences. *American Economic Review*, 111(2):720–54, 2021.

Bilmes, J. A. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. pp. 15, 1998.

Bountouridis, D., Harambam, J., Makhortykh, M., Marrero, M., Tintarev, N., and Hauff, C. SIREN: A Simulation Framework for Understanding the Effects of Recommender Systems in Online News Environments. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 150–159, Atlanta GA USA, January 2019. ACM. ISBN 978-1-4503-6125-5. doi: 10.1145/3287560.3287583. URL https://dl.acm.org/doi/10.1145/3287560.3287583.

Cai, Q., Zhan, R., Zhang, C., Zheng, J., Ding, G., Gong, P., Zheng, D., and Jiang, P. Constrained Reinforcement Learning for Short Video Recommendation. Technical Report arXiv:2205.13248, arXiv, May 2022. URL http://arxiv.org/abs/2205.13248. arXiv:2205.13248 [cs] type: article.

Chaney, A. J. B., Stewart, B. M., and Engelhardt, B. E. How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility. *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 224–232, September 2018. doi: 10.1145/3240323.3240370. URL http://arxiv.org/abs/1710.11214. arXiv: 1710.11214.

Chen, M., Beutel, A., Covington, P., Jain, S., Belletti, F., and Chi, E. Top-K Off-Policy Correction for a REINFORCE Recommender System. *arXiv:1812.02353 [cs, stat]*, November 2020. URL http://arxiv.org/abs/1812.02353. arXiv: 1812.02353.

Chen, X., Li, S., Li, H., Jiang, S., Qi, Y., and Song, L. Generative Adversarial User Model for Reinforcement Learning Based Recommendation System. *arXiv:1812.10613 [cs, stat]*, December 2019. URL http://arxiv.org/abs/1812.10613. arXiv: 1812.10613.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., and Shah, H. Wide & Deep Learning for Recommender Systems. *arXiv:1606.07792 [cs, stat]*, June 2016. URL http://arxiv.org/abs/1606.07792. arXiv: 1606.07792.

Cosley, D., Lam, S. K., Albert, I., Konstan, J. A., and Riedl, J. Is seeing believing? how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pp. 585–592, New York, NY, USA, April 2003. Association for Computing Machinery. ISBN 978-1-58113-630-2. doi: 10.1145/642611.642713. URL https://doi.org/10.1145/642611.642713.

Covington, P., Adams, J., and Sargin, E. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, pp. 191–198, Boston, Massachusetts, USA, 2016. ACM Press. ISBN 978-1-4503-4035-9. doi: 10.1145/2959100.2959190. URL http://dl.acm.org/citation.cfm?doid=2959100.2959190.

Epstein, R. and Robertson, R. E. The search engine manipulation effect (SEME) and its possible impact on the outcomes of elections. *Proceedings of the National Academy of Sciences*, 112(33): E4512–E4521, August 2015. doi: 10.1073/pnas.1419828112. URL https://www.pnas.org/doi/10.1073/pnas.1419828112. Publisher: Proceedings of the National Academy of Sciences.

Evans, C. and Kasirzadeh, A. User Tampering in Reinforcement Learning Recommender Systems. *arXiv:2109.04083 [cs]*, September 2021. URL http://arxiv.org/abs/2109.04083. arXiv: 2109.04083.

Farquhar, S., Carey, R., and Everitt, T. Path-Specific Objectives for Safer Agent Incentives. *arXiv:2204.10018 [cs, stat]*, April 2022. URL http://arxiv.org/abs/2204.10018. arXiv: 2204.10018.

Franklin, M., Ashton, H., Gorman, R., and Armstrong, S. Recognising the importance of preference change: A call for a coordinated multidisciplinary research effort in the age of AI. *arXiv:2203.10525 [cs]*, March 2022. URL http://arxiv.org/abs/2203.10525. arXiv: 2203.10525.

Gauci, J., Conti, E., Liang, Y., Virochsiri, K., He, Y., Kaden, Z., Narayanan, V., Ye, X., Chen, Z., and Fujimoto, S. Horizon: Facebook's Open Source Applied Reinforcement Learning Platform. *arXiv:1811.00260 [cs, stat]*, September 2019. URL http://arxiv.org/abs/1811.00260. arXiv: 1811.00260.

Gretzel, U. and Fesenmaier, D. Persuasion in Recommender Systems. *International Journal of Electronic Commerce*, 11(2):81–100, December 2006. ISSN 1086-4415. doi: 10.2753/JEC1086-4415110204. URL https://doi.org/10.2753/JEC1086-4415110204.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Hasan, M. R., Jha, A. K., and Liu, Y. Excessive use of online video streaming services: Impact of recommender system use, psychological factors, and motives. *Computers in Human Behavior*, 80:220–228, March 2018. ISSN 07475632. doi: 10.1016/j.chb.2017.11.020. URL https://linkinghub.elsevier.com/retrieve/pii/S0747563217306581.

Hazrati, N. and Ricci, F. Recommender systems effect on the evolution of users' choices distribution. *Information Processing & Management*, 59(1):102766, January 2022. ISSN 03064573. doi: 10.1016/j.ipm.2021.102766. URL https://linkinghub.elsevier.com/retrieve/pii/S0306457321002466.

Hohnhold, H., O'Brien, D., and Tang, D. Focusing on the Long-term: It's Good for Users and Business. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1849–1858, Sydney NSW Australia, August 2015. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2788583. URL https://dl.acm.org/doi/10.1145/2783258.2788583.

Holtz, D., Carterette, B., Chandar, P., Nazari, Z., Cramer, H., and Aral, S. The Engagement-Diversity Connection: Evidence from a Field Experiment on Spotify. *arXiv:2003.08203 [cs]*, March 2020. URL http://arxiv.org/abs/2003.08203. arXiv: 2003.08203.

Hughes, J. P., Guttorp, P., and Charles, S. P. A Non-Homogeneous Hidden Markov Model for Precipitation Occurrence. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 48(1):15–30, 1999. URL http://www.jstor.org/stable/2680815.

Häubl, G. and Murray, K. B. Preference Construction and Persistence in Digital Marketplaces: The Role of Electronic Recommendation Agents. *Journal of Consumer Psychology*, 13(1):75–91, January 2003. ISSN 1057-7408. doi: 10.1207/S15327663JCP13-1&2_07. URL http://www.sciencedirect.com/science/article/pii/S1057740803701788.

Ie, E., Hsu, C.-w., Mladenov, M., Jain, V., Narvekar, S., Wang, J., Wu, R., and Boutilier, C. RecSim: A Configurable Simulation Platform for Recommender Systems. *arXiv:1909.04847 [cs, stat]*, September 2019a. URL http://arxiv.org/abs/1909.04847. arXiv: 1909.04847.

Ie, E., Jain, V., Wang, J., Narvekar, S., Agarwal, R., Wu, R., Cheng, H.-T., Lustman, M., Gatto, V., Covington, P., McFadden, J., Chandra, T., and Boutilier, C. Reinforcement Learning for Slate-based Recommender Systems: A Tractable Decomposition and Practical Methodology. *arXiv:1905.12767 [cs, stat]*, May 2019b. URL http://arxiv.org/abs/1905.12767. arXiv: 1905.12767.

Jesse, M. and Jannach, D. Digital nudging with recommender systems: Survey and future directions. *Computers in Human Behavior Reports*, 3:100052, January 2021. ISSN 2451-9588. doi: 10.1016/j.chbr.2020.100052. URL https://www.sciencedirect.com/science/article/pii/S245195882030052X.

Jiang, R., Chiappa, S., Lattimore, T., György, A., and Kohli, P. Degenerate Feedback Loops in Recommender Systems. *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 383–390, January 2019. doi: 10.1145/3306618.3314288. URL http://arxiv.org/abs/1902.10730. arXiv: 1902.10730.

Juneja, P. and Mitra, T. Auditing E-Commerce Platforms for Algorithmically Curated Vaccine Misinformation. pp. 27, 2021.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, May 1998. ISSN 00043702. doi: 10.1016/S0004-3702(98)00023-X. URL https://linkinghub.elsevier.com/retrieve/pii/S000437029800023X.

Krakovna, V., Orseau, L., Kumar, R., Martic, M., and Legg, S. Penalizing side effects using stepwise relative reachability. *arXiv:1806.01186 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1806.01186. arXiv: 1806.01186.

Kramer, A. D. I., Guillory, J. E., and Hancock, J. T. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24):8788–8790, June 2014. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1320040111. URL https://www.pnas.org/content/111/24/8788. Publisher: National Academy of Sciences Section: Social Sciences.

Krueger, D., Maharaj, T., and Leike, J. Hidden Incentives for Auto-Induced Distributional Shift. *arXiv:2009.09153 [cs, stat]*, September 2020. URL http://arxiv.org/abs/2009.09153. arXiv: 2009.09153.

Kumar, R., Uesato, J., Ngo, R., Everitt, T., Krakovna, V., and Legg, S. REALab: An Embedded Perspective on Tampering. *arXiv:2011.08820 [cs]*, November 2020. URL http://arxiv.org/abs/2011.08820. arXiv: 2011.08820.

Li, L., Zheng, L., Yang, F., and Li, T. Modeling and broadening temporal user interest in personalized news recommendation. *Expert Systems with Applications*, 41(7):3168–3177, June 2014. ISSN 09574174. doi: 10.1016/j.eswa.2013.11.020. URL https://linkinghub.elsevier.com/retrieve/pii/S0957417413009329.

Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I. RLlib: Abstractions for Distributed Reinforcement Learning. *arXiv:1712.09381 [cs]*, June 2018. URL http://arxiv.org/abs/1712.09381. arXiv: 1712.09381.

Lu, Z. and Yang, Q. Partially observable markov decision process for recommender systems. *arXiv preprint arXiv:1608.07793*, 2016.

MacLeod, C. and Campbell, L. Memory accessibility and probability judgments: An experimental evaluation of the availability heuristic. *Journal of Personality and Social Psychology*, 63(6):890–902, 1992. ISSN 1939-1315(Electronic),0022-3514(Print). doi: 10.1037/0022-3514.63.6.890. Place: US Publisher: American Psychological Association.

Mansoury, M., Abdollahpouri, H., Pechenizkiy, M., Mobasher, B., and Burke, R. Feedback Loop and Bias Amplification in Recommender Systems. *arXiv:2007.13019 [cs]*, July 2020. URL http://arxiv.org/abs/2007.13019. arXiv: 2007.13019.

Matz, S. C., Kosinski, M., Nave, G., and Stillwell, D. J. Psychological targeting as an effective approach to digital mass persuasion. *Proceedings of the National Academy of Sciences*, 114(48):12714–12719, November 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1710966114. URL http://www.pnas.org/lookup/doi/10.1073/pnas.1710966114.

Mladenov, M., Meshi, O., Ooi, J., Schuurmans, D., and Boutilier, C. Advantage Amplification in Slowly Evolving Latent-State Environments. *arXiv:1905.13559 [cs, stat]*, May 2019. URL http://arxiv.org/abs/1905.13559. arXiv: 1905.13559.

Mudigere, D., Hao, Y., Huang, J., Tulloch, A., Sridharan, S., Liu, X., Ozdal, M., Nie, J., Park, J., Luo, L., Yang, J. A., Gao, L., Ivchenko, D., Basant, A., Hu, Y., Yang, J., Ardestani, E. K., Wang, X., Komuravelli, R., Chu, C.-H., Yilmaz, S., Li, H., Qian, J., Feng, Z., Ma, Y., Yang, J., Wen, E., Li, H., Yang, L., Sun, C., Zhao, W., Melts, D., Dhulipala, K., Kishore, K. R., Graf, T., Eisenman, A., Matam, K. K., Gangidi, A., Chen, G. J., Krishnan, M., Nayak, A., Nair, K., Muthiah, B., khorashadi, M., Bhattacharya, P., Lapukhov, P., Naumov, M., Qiao, L., Smelyanskiy, M., Jia, B., and Rao, V. High-performance, Distributed Training of Large-scale Deep Learning Recommendation Models. *arXiv:2104.05158 [cs]*, April 2021. URL http://arxiv.org/abs/2104.05158. arXiv: 2104.05158.

Nguyen, T. T., Hui, P.-M., Harper, F. M., Terveen, L., and Konstan, J. A. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web - WWW '14*, pp. 677–686, Seoul, Korea, 2014. ACM Press. ISBN 978-1-4503-2744-2. doi: 10.1145/2566486.2568012. URL http://dl.acm.org/citation.cfm?doid=2566486.2568012.

Okura, S., Tagami, Y., Ono, S., and Tajima, A. Embedding-based News Recommendation for Millions of Users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1933–1942, Halifax NS Canada, August 2017. ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098108. URL https://dl.acm.org/doi/10.1145/3097983.3098108.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Rabiner, L. R. An Introduction to Hidden Markov Models. pp. 13, 1986.

Ribeiro, M. H., Ottoni, R., West, R., Almeida, V. A. F., and Meira, W. Auditing Radicalization Pathways on YouTube. *arXiv:1908.08313 [cs]*, December 2019. URL http://arxiv.org/abs/1908.08313. arXiv: 1908.08313.

Russell, S. and Norvig, P. Artificial intelligence: a modern approach. 2002.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shi, J.-C., Yu, Y., Da, Q., Chen, S.-Y., and Zeng, A.-X. Virtual-Taobao: Virtualizing Real-world Online Retail Environment for Reinforcement Learning. *arXiv:1805.10000 [cs]*, May 2018. URL http://arxiv.org/abs/1805.10000. arXiv: 1805.10000.

Stray, J. Designing Recommender Systems to Depolarize. *arXiv:2107.04953 [cs]*, July 2021. URL http://arxiv.org/abs/2107.04953. arXiv: 2107.04953.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv:1904.06690 [cs]*, August 2019. URL http://arxiv.org/abs/1904.06690. arXiv: 1904.06690.

Sunehag, P., Evans, R., Dulac-Arnold, G., Zwols, Y., Visentin, D., and Coppin, B. Deep Reinforcement Learning with Attention for Slate Markov Decision Processes with High-Dimensional States and Actions. *arXiv:1512.01124 [cs]*, December 2015. URL http://arxiv.org/abs/1512.01124. arXiv: 1512.01124.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 1998. ISBN 978-0-262-19398-6.

Weinmann, M., Schneider, C., and vom Brocke, J. Digital Nudging. SSRN Scholarly Paper 2708250, Social Science Research Network, Rochester, NY, 2015. URL https://papers.ssrn.com/abstract=2708250.

Wilhelm, M., Ramanathan, A., Bonomo, A., Jain, S., Chi, E. H., and Gillenwater, J. Practical Diversified Recommendations on YouTube with Determinantal Point Processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 2165–2173, Torino Italy, October 2018. ACM. ISBN 978-1-4503-6014-2. doi: 10.1145/3269206.3272018. URL https://dl.acm.org/doi/10.1145/3269206.3272018.

Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., and Jing, H. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pp. 495–503, 2017.

Yao, S., Halpern, Y., Thain, N., Wang, X., Lee, K., Prost, F., Chi, E. H., Chen, J., and Beutel, A. Measuring Recommender System Effects with Simulated Users. *arXiv:2101.04526 [cs]*, January 2021. URL http://arxiv.org/abs/2101.04526. arXiv: 2101.04526.

Zhang, S., Yao, L., Sun, A., and Tay, Y. Deep Learning based Recommender System: A Survey and New Perspectives. *ACM Computing Surveys*, 52(1):1–38, February 2019. ISSN 0360-0300, 1557-7341. doi: 10.1145/3285029. URL http://arxiv.org/abs/1707.07435. arXiv: 1707.07435.

Zhang, Y., Dai, H., Xu, C., Feng, J., Wang, T., Bian, J., Wang, B., and Liu, T.-Y. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. *arXiv:1404.5772 [cs]*, July 2014. URL http://arxiv.org/abs/1404.5772. arXiv: 1404.5772.

Zhao, X., Xia, L., Zou, L., Yin, D., and Tang, J. Toward Simulating Environments in Reinforcement Learning Based Recommendations. *arXiv:1906.11462 [cs]*, September 2019. URL http://arxiv.org/abs/1906.11462. arXiv: 1906.11462.

A. Notation

- $x_t \in \mathbb{R}^d$: time-indexed user choice of content from a slate. x_t^π indicates a choice that was made from a slate that was produced by policy π.
- $s_t \in \mathbb{R}^d$: time-indexed slate chosen by the recommender system. See Fig. 12 for how slates are represented. s_t^π indicates a slate that was sampled from policy π.
- $\pi : (s_{0:k}, x_{0:k}) \rightarrow s_{k+1}$: a recommender system policy. π^{rnd} denotes a policy which selects slates randomly – so that the distribution of content matches the distribution of the slate.
- $u_t \in \mathbb{R}^d$: time-indexed user preferences. $u_t^{\pi'}$ is the preferences a user has at timestep t after interacting with policy π'. From the context, it should be clear whether such preferences are estimated future or counterfactual preferences, described in Appendix C.
- z_t: time-indexed user's internal state. Policy superscripts are used in the same way as for preferences.
- $\hat{r}_t(u_t) = u_t^T x_t$ is the reward function which captures user engagement. The expression $\hat{r}_t(u_t^{\text{safe}}, \pi) = (x_t^\pi)^T u_t^{\text{safe}}$ is used when the preferences used to evaluate the content are different from the policy: This means that π was used to select the slate from which the user picked the item x_t^π, but x_t^π's engagement is considered relative to potentially different preferences than those the user would have developed under π.
- $b_{0:t}^\pi(u_{t+1})$: the belief over a user's preferences at timestep $t+1$, as induced by slates sampled under π $s_{0:t}^\pi, x_{0:t}^\pi$. Formally, this is equivalent to $\mathbb{P}(u_{t+1}|s_{0:t}^\pi, x_{0:t}^\pi)$.
- $b_{0:t}^\pi(x_{t+1})$: the belief over a user's choice at timestep $t+1$, as induced by slates sampled under π. Formally equivalent to: $\mathbb{P}(x_{t+1}|s_{0:t+1}, x_{0:t})$.
- $\hat{P}_f : (s_{0:k}, x_{0:k}) \rightarrow (b_{0:k}(u_{k+1}), b_{0:k}(x_{k+1}))$: the future preference estimator model described in Sec. 4.
- $\hat{P}_i : (s_{0:k-1}, x_{0:k-1}) \rightarrow b_{0:k}(u_0)$ initial preference estimator described in Appendix C.
- $\hat{P}_c : (s_{0:k-1}, x_{0:k-1}, b_{0:k}(u_0)) \rightarrow (b_{0:k}(u_{k+1}), b_{0:k}(x_{k+1}))$ counterfactual preference estimator described in Appendix C.

B. Known choice model assumption

Here we try to justify why assuming a known choice model is almost a necessity given the problem that we set out to solve: without this (or some other assumption), estimating user preferences and learning their dynamics seems too difficult. Looking at our problem as a NHMM gives us insight as to such difficulty. See Appendix D for the casting of our problem to the NHMM formalism.

Our goal is to infer preferences (part of the hidden state), given only observations, but no user choice model (which is part of the observation model of the underlying NHMM). Algorithms such as Baum-Welch (Bilmes, 1998) have been proposed for jointly learning both the transition model of the hidden state, and the observation model of a HMM. This type of algorithm could be extended to the NHMM setting, providing a promising direction. However, the Baum-Welch algorithm assumes the transition dynamics of the hidden state to be linear, and the dimension of the hidden state to be known. We expect real user's dynamics to be non-linear (and so are the ones that we consider in our experiments). Moreover, we don't want to constrain our method to require knowledge of the dimension of the hidden state (the user's internal state) – which will be unknown in practice.

In light of this, assumption the nature of the choice model to be known seems instrumental to render preference inference possible in our setting. As shown in Sec. 7.1, it might sometimes be possible to detect whether the choice model is mis-specified if it leads to higher validation loss on choice prediction (which is observed) than alternate choice model hypotheses. This could guide a process of iteratively refining one's choice model. An important thing to note is that – although limiting – such assumption is already an improvement relative to all previous work related to preference shifts that we are aware of (Krueger et al., 2020; Jiang et al., 2019; Mansoury et al., 2020; Chaney et al., 2018; Evans & Kasirzadeh, 2021), which assumes the *whole* user preference dynamics to be known.

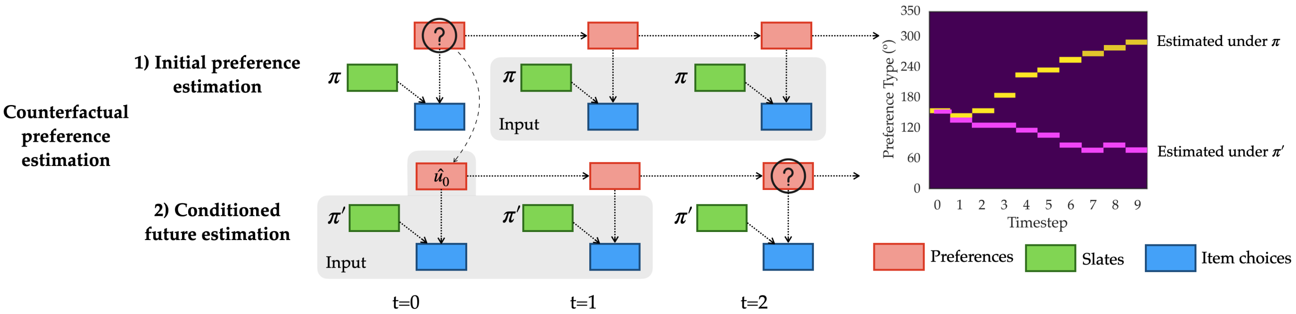

Figure 8: **Simulating counterfactuals.** To simulate what the user's preferences *would have been* under a different policy π', we first can use the *initial preference model* to estimate initial preferences based on *later* observables. We can then use the estimated initial preferences to simulate counterfactual preference trajectories for the user under π' using a *conditioned* future preference estimation model – where estimates are conditioned on the recovered initial preference belief.

C. Counterfactual Preferences Estimation

In Sec. 4 we develop a methodology to train a human model that can be used for both: 1) predicting how existing users' preferences would evolve if new policies were to be deployed to them; 2) simulating entire user preference trajectories under desired RS policies; this is useful, but suffers from a problem – users have already been biased by the previous policies, and their preferences might have already e.g. shifted to extremes; forward prediction will thus miss certain undesired effects. Another useful task would be to "turn back time" – and ask for a specific user, "how would have their preferences shifted if we exposed them to π' *from the very beginning* of their interactions with the platform?". To differentiate these problems in this section, we call the methodology developed in Sec. 4 *future* preference estimation; instead we call the problem described just above the *counterfactual* preference estimation problems.

While the methodology described in Sec. 4 is sufficient to estimate the expected preferences and behaviors a user will have in the future, it cannot be readily used to estimate counterfactual preferences and behaviors: given a set of past interactions $\mathcal{D}_j$ collected under policy π, we cannot predict what the preferences *would have been for this user* if an alternate policy π' had been used from the first timestep of interaction instead of π – we denote these *counterfactual preferences* as $u_{0:T}^{\pi'}$ (note that in this section, we are overburdening the notation for the superscript to mean *counterfactual preferences under a policy* – this can be distiguished from the usual notation in Appendix A of *future preferences under a policy* by whether the time subscripts are larger or smaller than T). On a high-level, to perform the counterfactual estimation task we want to extract as much information as possible about the user's initial internal state from the historical interactions $\mathcal{D}_j$ we have available for them: even though such interactions were collected with a different policy π, they will still contain information about the user's initial state (including their initial preferences before interacting with the RS). Then, based on this belief about the user's initial internal state, we want to obtain a user-specific estimate of the effect that another policy π' would have had on their preferences.

As done in Sec. 4, we first describe how one could solve this problem if one had oracle access to the true user internal state dynamics, and then relax this assumption showing how one can learn to perform the inference approximately given only observable interaction data.

Estimation under known internal state dynamics.

Under oracle access to the internal state dynamics, we can first obtain the belief over the initial state of a given user $\mathbb{P}(z_0|s_{0:T}^{\pi}, x_{0:T}^{\pi})$ via NHMM smoothing. This is a simple extension of HMM smoothing (Russell & Norvig, 2002) which can be derived using the same steps used in Appendix E.1. We can then roll out the human model forward dynamics with the fixed policy π' instead of π: $\mathbb{P}(z_t^{\pi'}|s_{0:T}^{\pi}, x_{0:T}^{\pi}) = \sum_{z_0} \mathbb{P}(z_t^{\pi'}|z_0)\mathbb{P}(z_0|s_{0:T}^{\pi}, x_{0:T}^{\pi})$.

Estimation under unknown internal state dynamics.

Without oracle access to the internal state dynamics, again we try to learn to perform this NHMM counterfactual task approximately. One challenge in obtaining supervision for the task is that in our dataset of interactions we never get to see true counterfactuals. We get around this by decomposing the task into two parts, for which we use two separate models: **(1)** *an initial preference estimation network*, which is used to estimate the initial internal state for the user (based on the interaction data $x_{0:T}^{\pi}, s_{0:T}^{\pi}$ under π which we have available), i.e. train a predictor which approximates the NHMM smoothing distribution $\mathbb{P}(u_0|x_{0:T}^{\pi}, s_{0:T}^{\pi})$ which we will denote here as $b_{0:T}^{\pi}(u_0)$ (the belief over initial preferences); and then

(2) *an counterfactual preference estimation network* which is used to estimate $u_{0:T}^{\pi'}$ conditional on the belief of the initial preferences.

The models for the two tasks are trained with the same method used for predicting future preference estimation (Fig. 2) but with different inputs and supervision signal (Fig. 8). For task **(1)**, the network is trained to predict the *initial* (instead of future) preferences of the user based on later context $x_{1:T}^{\pi}, s_{1:T}^{\pi}$. While the network predicts $\mathbb{P}(u_0|x_{1:T}^{\pi}, s_{1:T}^{\pi})$, we can recover the correct smoothing estimate as $\mathbb{P}(u_0|x_{0:T}^{\pi}, s_{0:T}^{\pi}) \propto \mathbb{P}(x_0^{\pi}|s_0^{\pi}, u_0)\mathbb{P}(u_0|x_{1:T}^{\pi}, s_{1:T}^{\pi})$ (see Appendix E.2).

The network for task **(2)** is obtained by changing the prediction network to also condition on the recovered initial preferences as shown in 8, enabling us to make predictions of the form $\mathbb{P}\big(u_{k+1}^{\pi'}|b_{0:T}^{\pi}(u_0), x_{0:k}^{\pi'}, s_{0:k}^{\pi'}\big)$. For training, we first recover initial preference beliefs for each user j in the data $\mathcal{D}_j$ with the initial preference estimation model (which we assume has already been trained). We then train this counterfactual estimation network to reconstruct the user j's actual interactions (under π) simply based on this initial preference estimate (similarly to the case in Sec. 4.2 but by additionally conditioning on the initial preference belief). This teaches the network to leverage the information contained user's initial preferences estimate to better estimate their preferences and behaviors based on what their interactions have been so far. In practice, we train these models using a transformer architecture similar to (Sun et al., 2019), and we detail in Appendix E.2 why this is a good fit for our tasks.

At inference time, we recover an estimate of initial preferences based on interaction data $(s_{0:T}^{\pi'}, x_{0:T}^{\pi'})$ of a new user with a new recommender policy π' with model (1), and then can estimate the preferences such user would have had under a new policy π'' with model (2). Such counterfactual preference estimate is obtained with Monte Carlo simulations similarly to Sec. 4.2 with the difference that the network is also conditioned on the initial preferences estimate. See Algorithm 2 for the full algorithm and Appendix E.2 for why this approximates the NHMM task.

We now have a way to estimate – for a new policy π' – what we would expect its impact would have been on a user relative to having deployed π (from which we have observational data; note that similarly to future preferences estimation, this procedure too can suffer if π' induces a strong distribution shift relative to the training data).

D. Casting our problem to a Non-Homogeneous Markov Model

In our NHMM instance, the observation for each timestep is the corresponding (s_t, x_t) pair, the hidden state is z_t, and the probability of an observation is given by the joint probability of the policy choosing a specific slate given the history so far $\mathbb{P}(s_t|s_{0:t-1}, x_{0:t-1}) = \mathbb{P}\big(\pi(s_{0:t-1}, x_{0:t-1}) = s_t\big)$ and the user choosing a specific item from that slate $\mathbb{P}(x_t|s_t)$. The *dynamics model* is instead given by $\mathbb{P}(z_{t+1}|z_t, s_{0:t-1}, x_{0:t-1}) = \sum_{s_{t+1}} \mathbb{P}(z_{t+1}|z_t, s_t)\mathbb{P}(s_t|s_{0:t-1}, x_{0:t-1})$ (the output of π depends on the history of slates and choices so far, but we omit them from the notation for simplicity). Note that if the policy π were deterministic, the dynamics model of the NHMM reduces to $\mathbb{P}(z_{t+1}|z_t, s_t)$, as conditioning on s_t or on the full history is equivalent. We use this notation throughout the main text of the paper for simplicity.

E. Preference estimation: additional information

E.1. Future preference estimation

Future preference estimation with known dynamics

The Hidden Markov Model (HMM) "prediction task" corresponds to the inference of the hidden state of the system at the next timestep given the history so far: $\mathbb{P}(h_T|o_{0:k})$ where h and o are respectively the hidden state and the observations of the HMM, and $T > k$. We will provide here a proof sketch of how the HMM prediction task can be easily extended to NHMM prediction. On a high level, by taking into account the time-dependent dynamics (computing forward inferences differently for every timestep), inference tasks in a NHMM should be no different than those in an HMM. We illustrate this more formally below for the prediction task.

The prediction inference in HMMs is performed by recursively repeating two steps: first, obtaining the *filtering estimate* which incorporates the latest observation as $\mathbb{P}(h_t|o_{0:t}) = \alpha\mathbb{P}(o_t|h_t)\mathbb{P}(h_t|o_{0:t-1})$, where α is a constant; then using the filtering estimate for prediction without the addition of new evidence, $\mathbb{P}(h_{t+1}|o_{0:t}) = \sum_t \mathbb{P}(h_{t+1}|h_t)\mathbb{P}(h_t|o_{0:t})$. Once all evidence is incorporated and we have $\mathbb{P}(h_k|o_{0:k})$, one can repeatedly apply the second step to obtain $\mathbb{P}(h_T|o_{0:k})$ – the quantity we set out to obtain. This can be thought of as simply propagating the belief over the hidden state one timestep at a time in the future using the *dynamics model* $\mathbb{P}(h_{t+1}|h_t)$. See (Russell & Norvig, 2002) for more information.

This is where the difference with the HMM comes in: the forward model will be time dependent, as the output of the policy π depends on the history so far. By using the same exact approach as in the paragraph above (with this modification in the prediction step), one can perform the same inference also in a NHMM. In the HMM notation, we would have $\mathbb{P}(h_{t+1}|o_{0:t}) = \sum_t \mathbb{P}(h_{t+1}|h_t, o_{0:t})\mathbb{P}(h_t|o_{0:t})$, where we cannot drop the evidence term $o_{0:t}$ in the forward model.

One peculiarity of our specific task is that we want to infer the preferences in the future under a different policy π', even though our evidence is under a different policy π. Note that this is not an issue as long as π and π' are known, as it just means that the time-dependent dynamics will be different in the timesteps in which the policy π' was used.

Future preference estimation with Algorithm 1 and why it approximates NHMM prediction.

Algorithm 1 is used to obtain the belief over the preferences a user would have at timestep H – assuming that the user interacted for T timesteps with a policy π (for which we have actual interaction data), and then interacted from timestep T to H with a policy π'. We refer to this belief as $b^{\pi}_{0:T}(u_H) = \mathbb{P}(u^{\pi'}_H | s^{\pi}_{0:T}, x^{\pi}_{0:T})$.

The first model inference pass in Algorithm 1 resulting in $b_{0:T}(u_{T+1})$ will be a distribution over u_{T+1} conditioned on $s^{\pi}_{0:T}, x^{\pi}_{0:T}$. A minimizer for this prediction problem would be for the network to represent the actual belief that would be obtained by performing the prediction step in the NHMM (described in Sec. 4.1). This is because the expected cross-entropy between user choices and the induced choice distribution (induced by the preference belief prediction) would be minimized.

One limitation is that there are possibly multiple beliefs over preferences that induce the same exact distribution over choices: in that sense, the correct preference belief might be unidentifiable. Experimentally, we don't find this to be a problem: the choice of having the representation of the preference belief be a mixture of Von Mises distributions (see the "Preference estimation model form" heading later in this section) – which can be thought of as Normals over a circle – is already a good enough inductive bias to be able to recover good preference beliefs.

In the NHMM, note that the unrolling the forward model (i.e. computing $\mathbb{P}(u^{\pi'}_H | z^{\pi}_{T+1})$) could be computed exactly, or be performed with Monte Carlo estimation, by sampling many internal states z^{π}_{T+1} from the forward prediction distribution, and then sampling internal state evolutions according to the dynamics. In our algorithm, by sampling choices and slates then conditioning on them, we are approximating the Monte Carlo estimation approach. When simulating each user choice, the model can be thought of as implicitly sampling a preference for this simulated user (from the preference belief) and then sampling a choice from the corresponding choice distribution. This means that each simulation rollout should be equivalent to "sampling a user" (according to the distribution of users in the data) and then sampling their choices.

Algorithm 1 Predicting future user preferences at timestep H under RS policy π'

```
Input: past interactions $x^{\pi}_{0:T}$, $s^{\pi}_{0:T}$, policy $\pi'$, future preference estimator $\hat{P}_f$, horizon $H$, number of Monte Carlo simulations $N$
                                                                        {if no past interaction data is given}
if $x_{0:T} = s_{0:T} = \emptyset$ then
    Sample slate $s_0 \sim \pi'(\emptyset, \emptyset)$
    Obtain belief over initial preferences and choice $b_\emptyset(u_0), b_\emptyset(x_0) = \hat{P}_f(\emptyset, \emptyset)$
    Simulate a user choice $x_0 \sim b_\emptyset(x_0)$                    {we now have past interactions with $T = 0$}
end if
for $i = 0; i < N; i++$ do
    while $k = T; k < H; k++$ do
        $b_{0:k}(u_{k+1}), b_{0:k}(x_{k+1}) = \hat{P}_f(s_{0:k}, x_{0:k})$          {estimate pref. and choices}
        $s_{k+1} \sim \pi'(x_{0:k}, s_{0:k})$                                  {sample next timestep slate}
        $x_{k+1} \sim b_{0:k}(x_{k+1})$                                        {simulate a user choice}
        Add $x_{k+1}$ and $s_{k+1}$ to the current simulated trajectory's history
    end while
end for
Average $b_{0:k-1}(u_k)$ and $b_{0:k-1}(x_k)$ across the $N$ futures (for each $k$, with $T < k \leq H$)
Return: Belief over future pref. $b^{\pi}_{0:T}(u_k)$ and behaviors $b^{\pi}_{0:T}(x_k)$ for each $k$ s.t. $T < k \leq H$.
```

E.2. Counterfactual preference estimation

Initial preference prediction model output correction

Below, we show that one can recover the smoothing estimate $\mathbb{P}(u_0|x_{0:t}, s_{0:t})$ from the predicted preferences $\mathbb{P}(u_0|x_{1:t}, s_{1:t})$

which will be a biased estimate of the initial preferences (as it does not incorporate the information from timestep $t = 0$) . Note that:

$$\mathbb{P}(u_0|x_{0:t}, s_{0:t}) = \frac{\mathbb{P}(u_0|x_0, s_0)\mathbb{P}(x_{1:t}, s_{1:t}|u_0)}{\mathbb{P}(x_{0:t}, s_{0:t})} = \frac{\mathbb{P}(u_0|x_0, s_0)}{\mathbb{P}(x_{0:t}, s_{0:t})} \frac{\mathbb{P}(u_0|x_{1:t}, s_{1:t})\mathbb{P}(x_{1:t}, s_{1:t})}{\mathbb{P}(u_0)} \tag{1}$$

$$= \frac{\mathbb{P}(x_0, s_0|u_0)\mathbb{P}(u_0|x_{1:t}, s_{1:t})\mathbb{P}(x_{1:t}, s_{1:t})}{\mathbb{P}(x_{0:t}, s_{0:t})\mathbb{P}(x_0, s_0)} \propto \mathbb{P}(x_0, s_0|u_0)\mathbb{P}(u_0|x_{1:t}, s_{1:t}) \tag{2}$$

The first equality is given by the definition of smoothing applied to $t = 0$ (Russell & Norvig, 2002). The second equality is obtained by using Bayes Rule on the backwards message $\mathbb{P}(x_{1:t}, s_{1:t}|u_0)$, and the third is obtained by using Bayes Rule on $\mathbb{P}(u_0|x_0, s_0)$. Finally, we can ignore $\mathbb{P}(x_{1:t}, s_{1:t})$, $\mathbb{P}(x_{0:t}, s_{0:t})$, and $\mathbb{P}(x_0, s_0)$ as they are constants.

Counterfactual preference estimation with Algorithm 2 and why it approximates NHMM prediction.

A similar argument to that in Appendix E.1 can be made as to why the initial preference estimation model would approximate the corresponding NHMM smoothing task.

However, for the second step of counterfactual preference estimation, one issue arises. Relative to having access to the full dynamics of the internal state, when performing approximate inference with our model we lose some information: we are only able to recover a belief over the initial *preferences*, whereas the NHMM with full dynamics access would be able to recover a belief over the *full internal state* of the user. This will reduce the accuracy of our counterfactual estimation, but is the best we can do without further assumptions.

Mathematically, we approximate the NHMM target distribution $\mathbb{P}(u_T^{\pi'}|x_{0:t}^{\pi}, s_{0:t}^{\pi})$ as:

$$\mathbb{P}(u_T^{\pi'}|x_{0:t}^{\pi}, s_{0:t}^{\pi}) \approx \sum_{u_0} \mathbb{P}(u_T^{\pi'}|u_0)\mathbb{P}(u_0|x_{0:t}^{\pi}, s_{0:t}^{\pi}) = \mathbb{P}\big(u_T^{\pi'}|b_{0:t}^{\pi}(u_0)\big) \tag{3}$$

$$= \sum_{x_{0:T-1}^{\pi'}, s_{0:T-1}^{\pi'}} \mathbb{P}(x_{0:T}^{\pi'}, s_{0:T}^{\pi'})\mathbb{P}\big(u_T^{\pi'}|b_{0:t}^{\pi}(u_0), x_{0:T}^{\pi'}, s_{0:T}^{\pi'}\big) \tag{4}$$

where we the last expression is approximated with a Monte Carlo estimate detailed in Algorithm 2[1] (similarly to what was done in Algorithm 1). To do well at this second trajectory reconstruction task, the network necessarily needs to learn how to make best use of the belief over initial preferences, and implicitly learn their dynamics, as for the future preference estimation task.

Algorithm 2 Predicting counterfactual user preferences at timestep T under policy π', given t timesteps of interaction data with π.

```
  **Inputs:** past interactions $x_{0:t}^{\pi}$, $s_{0:t}^{\pi}$, policy $\pi'$, future, initial, and counterfactual preference estimators $\hat{P}_f$, $\hat{P}_i$, $\hat{P}_c$, horizon $T$, constant $N$

  $b_{0:t}^{\pi}(u_0) = \hat{P}_i(s_{0:t}^{\pi}, x_{0:t}^{\pi})$                {initial pref. belief given interactions with $\pi$}
  $\hat{P}_f = \hat{P}_c(b = b_{0:t}^{\pi}(u_0))$                {future pref predictor conditioned on init belief}
  $\big(b_{0:k-1}^{\pi}(u_k^{\pi'}), b_{0:k-1}^{\pi}(x_k^{\pi'})\big)_{0<k\leq T}$ =Algorithm 1$\Big(\emptyset, \emptyset, \pi', \hat{P}_f, T, N\Big)$
  **Return:** Distributions of counterfactual preferences and behaviors under policy $\pi'$
```

Preference estimation model form

When considering all models described in Sec. 4 and in Appendix 2, we have three models for preference estimation: respectively initial, counterfactual, and future preference estimators $\hat{P}_i$, $\hat{P}_c$, $\hat{P}_f$. Any sequence model, such as RNNs or

[1] This algorithm is not actually used in it's pure form in the experiments. Our choice of "safe policy" π_{rnd} happens to choose constant slates (i.e. a uniform distribution no matter the history), so there is a shortcut to the procedure: by simply setting the inputted slated for counterfactual prediction to be uniforms, and predicting preferences without any user choices, one can train the model to directly output the belief over counterfactual preferences for any timestep, across the whole userbase.

transformers, would be appropriate for these tasks that have variable number of inputs. We choose to use transformers, as described in Appendix H.2.

One detail of note that was omitted from Fig. 2 is that – to enable to represent multi-modal beliefs over preference space – we let the models' output be parameters of multiple Von Mises distributions and additionally some weights w. The w weighted average of these distributions will form the predicted belief over u_t.

F. Computing metrics

For computing the metrics defined in Sec. 5 for estimated future preferences of a user, one can use the approach delineated in Sec. 4. If one wants instead to compute the metrics for counterfactual preferences of a user, one should use the methodology from Appendix C. Computing metrics for counterfactual metrics is necessary in the case of penalized RL training: we want to know, for the user at hand and the current simulated trajectory under the policy π' we are training, what the preferences of such a user would have been under an alternate safe policy π^{safe}. See Algorithm 3 for more details.

G. Penalized RL

The underlying MDP

One could cast the recommendation problem as a POMDP (Lu & Yang, 2016; Mladenov et al., 2019) in which the state of the environment is hidden and contains the user's internal state, which evolves over time. Equivalently, one can consider the belief-MDP induced by the recommender POMDP (Kaelbling et al., 1998), and approximate a solution to such belief-MDP via Deep-RL with a policy trained with observation histories as input (this is theoretically sufficient for the policy to recover a belief over the current hidden state and take the optimal action). The action space will be given by the space of possible slates that the RS can choose. The reward signal will be the expected reward for the current timestep $\mathbb{E}\big[\hat{r}_t(u_t^\pi)\big]$ (or with the extra terms for the proxies in the case of penalized training). The introduction of expectation can be thought of as expected SARSA (Sutton & Barto, 1998), as argued in (Ie et al., 2019b).

Penalized RL training

The full set of steps to run RL training are as follows: once the human models described in Sec. 4 are trained, one can use them to simulate user trajectories and compute penalty metrics for such trajectories (see Algorithm 3). One can then optimize the RL policy based on the on-policy simulated trajectory rollouts.

Algorithm 3 Generating a trajectory for RL training and computing metrics

```
**Input:** Initial, counterfactual, and future preference estimators $\hat{P}_i, \hat{P}_c, \hat{P}_f$; a policy $\pi$, a safe policy $\pi_{\text{safe}}$, a horizon $H$, a constant $N$.

Sample slate $s_0^\pi \sim \pi(\emptyset, \emptyset)$ and imagine user choice $x_0^\pi \sim \hat{P}_i(x_0|\emptyset, \emptyset)$
for $t = 1; t \leq H; t++$ do
  $b_{0:t-1}^\pi(u_0) = \hat{P}_i(s_{0:t-1}^\pi, x_{0:t-1}^\pi)$    {current belief over initial preferences}
  $b_{0:t-1}^\pi(u_t), b_{0:t-1}^\pi(x_t) = \hat{P}_f(s_{0:t-1}^\pi, x_{0:t-1}^\pi)$    {belief over pref and choices}
  $\mathbb{E}[b_{0:t-1}^\pi(u_t^{\pi_{\text{safe}}})] \in$ Algorithm 2$\big(t, \pi_{\text{safe}}, \hat{P}_c, \hat{P}_i, x_{0:t-1}^\pi, s_{0:t-1}^\pi\big)$
      {belief over counterfactual preferences under safe policy for this user}
  $s_t^\pi \sim \pi(x_{0:t-1}^\pi, s_{0:t-1}^\pi)$    {sample slate}
  $x_t^\pi \sim b_{0:t-1}^\pi(x_t)$    {imagine a user choice}
  $D_t(u_t^\pi, u_t^{\text{safe}}) = \mathbb{E}_{u_t^\pi, u_t^{\pi_{\text{safe}}}, x_t^\pi}\big[(x_t^\pi)^T u_t^\pi - (x_t^\pi)^T u_t^{\pi_{\text{safe}}}\big]$    {compute penalty metric(s) for timestep}
  $r_t^{RL} = \mathbb{E}[r_t] + D_t(u_t^\pi, u_t^{\text{safe}})$
end for
**Return:** User-RS interactions and training rewards for the simulated trajectory
```

Reduction of distances to final penalized objective

Note that the full penalized RL objective $\Bigg(\sum_t^T \mathbb{E}\big[\hat{r}_t(u_t^\pi)\big]\Bigg) - \nu_1 D(u_{0:T}^\pi, u_0) - \nu_2 D(u_{0:T}^\pi, u_{0:T}^{\pi_{\text{rnd}}})$ for our choice of distance function D reduces to $\sum_t^T \mathbb{E}\big[\hat{r}_t(u_t^\pi) + \nu_1' \; \hat{r}_t(u_0, \pi) + \nu_2' \; \hat{r}_t(u_t^{\pi_{\text{rnd}}}, \pi)\big]$ for some choice of ν_1', ν_2' which can be treated as hyperparameters of how much we want to value the engagement under each safe policy preferences relative to the engagement under the main policy.

H. Experiment details

H.1. Ground truth users

Reduction of logit model to our case.

In our experimental setup, the traditional conditional logit model $\mathbb{P}(x_t = x|s_t, u_t) = \frac{e^{\beta_c x^T u_t}}{\sum_{x \in s} e^{\beta_c x^T u_t}}$ doesn't apply directly in this form, as we consider slates to be distributions rather than sets of discrete items. Intuitively, user's choices should still depend on the slate: the proportion of a certain item in the current slate (one can think of this as the proportion of a certain item *type*), should influence the probability of the user of selecting that item (type). We operationalize this as $\mathbb{P}(x_t = x|s_t, u_t) \propto \mathbb{P}(s_t = x)e^{\beta_c x^T u_t}$, with an additional term $\mathbb{P}(s_t = x)$ which takes into account the proportion of each item (type). Note that this also mathematically corresponds to the generalization of the traditional logit model: when the slate is discrete, $\mathbb{P}(s_t = x)$ will simply be an indicator for whether the item is in the slate, leading to the traditional logit model form.

Feed belief update

Our ground truth user has a belief over future slates $b_t^H(s)$. Users' initial belief matches the content feature distribution itself $b_0^H(s) = \mathcal{D}$. After receiving a slate s_t, the user's induces a belief $b_t^H(s) \propto s_t^3$ over the future slates in the user, i.e. the user will expect the next feeds to look like the most common items in the current feed, as a result of availability bias (MacLeod & Campbell, 1992).

Lack of no-op choice.

While the assumption that user must pick item from the slate is unrealistic, this could be resolved by adding an extra no-op choice to every slate (Sunehag et al., 2015).

Preference shift model.

Our preference shift model is inspired by (Bernheim et al., 2021), but adapted to our experimental setup as described below. The choice of preferences is modulated by a "mindset flexibility" parameter λ which captures how open they are to modifying their current preferences. Users assign value to the choice of next-timestep preferences u_{t+1} as: $V\left(u_t, b_t^H(s), u_{t+1}, \lambda\right) = \mathbb{E}_{x_{t+1} \sim b_t^H(s)}[\lambda \hat{r}_{t+1}(u_t) + (1 - \lambda)\hat{r}_{t+1}(u_{t+1})]$, where with $\hat{r}_{t+1}(u_t)$ indicates the engagement value obtained by the user under the choice x_{t+1} and preference u_t. Users pick their next timestep preferences also according to the conditional logit model, but over their expected value of such preferences choices: $\mathbb{P}(u_{t+1}|b_t^H(s), u_t, \lambda) \propto e^{\beta_d V(u_t, b_t^H(s), u_{t+1}, \lambda)}$. Intuitively, users update their preference to more "convenient" ones – ones that they expect will lead them to higher engagement value. The main change we introduce from the original model is incorporating the belief over future feeds.

Ground truth human parameters for experiments (β, λ, u_0, etc.).

The users' preference-flexibility parameter is given by $\lambda = 0.9$, and their initial preferences are drawn from a normal distribution[2] with mean $u = 130°$ and standard deviation $20°$. The temperature parameter β is set as mentioned in the main text and shown in Fig. 9.

As users obtain higher engagement value when they act less stochastically, these portions of preference space form attractor points as can be seen in all policies in Fig. 1. Also in Fig. 1 we see that while naturally preferences shift to mostly focus on one of these modes, some RS policies drive preferences to the other mode. While preferences naturally tend to shift towards one of these modes, some RS policies drive preferences to the other mode. As the engagement does not correspond to actual value, converging to the higher local optimum of engagement ($u = 270°$ instead of $u = 80°$) is not necessarily desirable.

H.2. Learned human models

For our learned human models, we use the BERT transformer architecture (similarly to (Sun et al., 2019)) with 2 layers, 2 attention heads, 4 sets of Von Mises distribution parameters, a learning rate of 0.00003, batch size of 500, and 100 epochs.

[2] Technically one should use Von Mises distributions – a distribution similar to normals, but for which the domain is a circle. As Von Mises distributions are not implemented in numpy (Harris et al., 2020), for simplicity we use clipped normal distributions (disregarding probability mass beyond $180°$ in either direction) in all places except for the the transformer output (which uses PyTorch's (Paszke et al., 2019) Von Mises implementation)

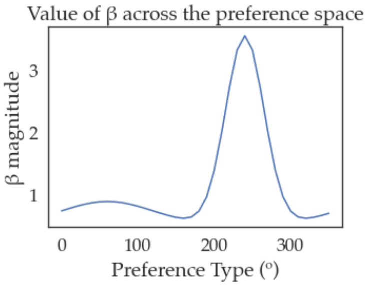

Figure 9: How the temperature parameter β varies across the preference space for our ground truth humans, which defines the true choice model.

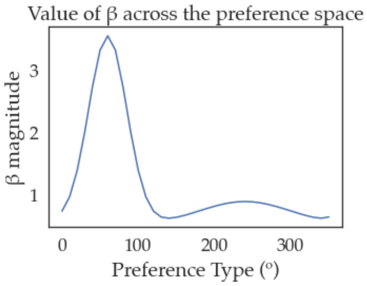

Figure 10: How the temperature parameter β varies across the preference space for our mis-specified choice model used for the mis-specification robustness experiments in Sec. 7.1 and Appendix I.

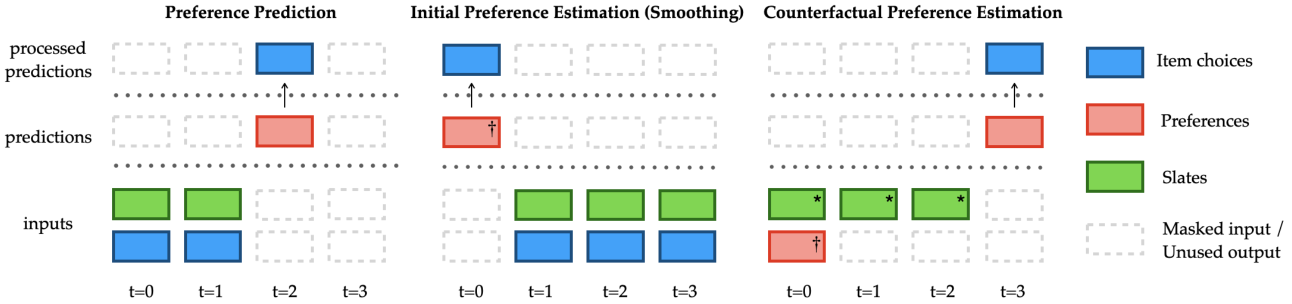
Preference Prediction
Initial Preference Estimation (Smoothing)
Counterfactual Preference Estimation
processed predictions
predictions
inputs
†
*
t=0
t=1
t=2
t=3
Item choices
Preferences
Slates
Masked input / Unused output

Figure 11: **BERT representation of the inference tasks.** While our method is compatible with any sequence model, we choose to use a BERT transformer models. **Left:** estimation of the user's future preferences and choice at t=2, given the interaction history history so far (this modality of prediction is closest in setup to Sun et al. (2019)). **Middle:** recovering a belief over the initial preferences and choice of the user based on later interactions. **Right:** conditioning on the estimate of initial preferences (†) recovered from the smoothing network one can estimate counterfactual preferences and choices under slates (*) (chosen from a policy π' we are interested in) and imagined choices (not shown due to space).

We train on the data described in Sec. H.3.

In architecture, we use a similar form to that of BERT4Rec (Sun et al., 2019) for ease of performing inference on future or past preferences given contexts of interaction, as described in Fig. 11. We mask all inputs that should not be used for prediction.

Mis-specification model. For the mis-specified human choice model experiments, we set the beta parameters of the choice model across the preference space as shown in Fig. 10 (in contrast to the true values, shown in Fig. 9). We found that increasing the mis-specification further led to worse results as expected.

H.3. Simulated dataset

See Fig. 12 for a summary of how content is generally instantiated in our setup.

We set the distribution of content in such a way that it forms a uniform distribution across features, that is $\mathcal{D} = \text{Uniform}(0°, 360°)$. We simulate historical user interaction data with a mixed policy π which is similar (but not equal to) a random policy in half the rollouts and for the other half is goal-directed:

- Half of the data is created with a RS policy which chooses an action uniformly among a set of possible slates (distributions over the feature space with means $0°, 10°, ..., 340°, 350°$, and standard deviations equal to $30°$ or $60°$). Each of these slate types can be thought of as a slate which contain mostly one specific type of content.
- The other half of the data is created with a RS which chooses $s_t = \mathcal{D} = \text{Uniform}(0°, 360°)$ 80% of the time (i.e. the slate that would be chosen by the random RS π_{rnd}), and chooses a random action from the same set of possible slates as above the remaining 20% of the time.

This is to simulate the setting in which the safe policy we are interested in (the random NPS policy) is similar to previously deployed ones that are represented in the data (although not the same, so the network still has to generalize across RS policies at test time) – but also not quite the same, requiring some amount of generalization.

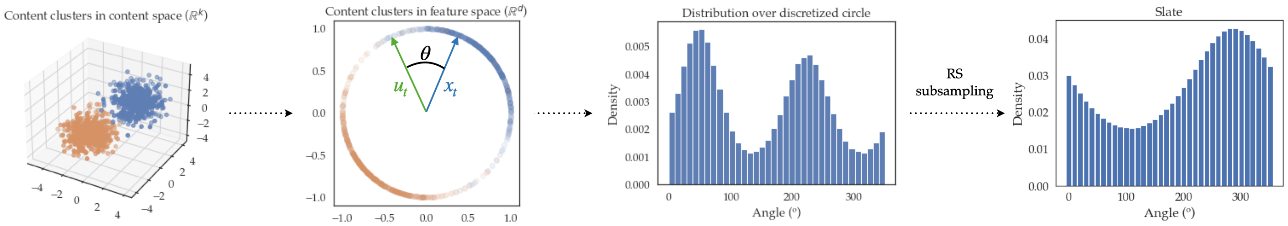

Figure 12: **a)** → **b)** The content can be mapped to an empirical distribution over feature space $\mathcal{D}$ in $\mathbb{R}^d$. We consider dimension $d = 2$ for ease of visualization. Restricting preferences and choices to be unit vectors, one can think of them as a points on a circle: the engagement value $\hat{r}_t$ will thus be related to the angle θ between u_t and x_t. **c)** We discretize this circular preference and feature space into $n = 36$ bins (i.e. binning the angles) which enables to visualize distributions over preferences and over content features as histograms over angles. **d)** We model slates s_t as categorical distributions over the discretized n-bin feature space.

H.4. RS training

For RL optimization, we use PPO (Schulman et al., 2017) trained with Rllib (Liang et al., 2018). The action space of the recommender system is given by distributions over feature space with means $0°, 60°, ..., 260°, 320°$, and standard deviations equal to $60°$. As observations to the system, we provide the current slate, the previous user choice, and the current estimates from the HMM for smoothing, filtering, and natural preference shift counterfactual distributions, in order to increase training speed (note that these will not change the optimal policy). All policies are recurrent so they are able to reason about the history of interactions so far.

For training our myopic policies, we use the same exact infrastructure as above, but set $\gamma = 0$, similarly to previous work (Krueger et al., 2020).

We use batch size 1200, minibatch size 600, 4 parallel workers, 0.005 learning rate, 50 gradient updates per minibatch per iteration, policy function clipping parameter of 0.5, value function clipping parameter of 50 and loss coefficient of 8, with an LSTM network with 64 cell size. $\gamma = 0$ for the myopic training and $\gamma = 0.99$ for long-horizon RL training. Training runs in less than 30 minutes for each condition on a MacBook Pro 16" (2020).

I. Results

We report here additional experimental results on for the preference estimation task, this time using the other models described in Fig. 11: the initial preference estimation network and the counterfactual preference estimation network.

Quality of initial preference estimates. The setup for this experiment is identical to that of Fig. 5, except that now we are evaluating the initial preference predictor which approximates the inference $\mathbb{P}(u_0|x^{\pi}_{0:T}, s^{\pi}_{0:T})$. We show the results in Fig. 13. Both under the correct choice model and with some mis-specification, preference prediction performs similarly to oracle estimation. For the initial preference prediction, mis-specification seems to have a less detrimental effect relative to Fig. 3-Fig. 14. The preference losses for our model are (slightly) higher than oracle. Preference prediction accuracies are very slightly higher than those of the oracle by random variability.

Quality of counterfactual preference estimates. We now turn to evaluating the quality of our counterfactual preference estimation model. As mentioned in Appendix C, such a model is trained to predict the preferences and choices of users for which we have seen interactions for, based on the approximate smoothing estimate obtained by the initial preference estimation model. To test this model in a harder setting we consider oracle access to counterfactual trajectories of the users in the normal dataset, while interacting with a random recommender policy π^{rnd}. We obtain the smoothing estimates from the usual trajectories present in the validation data (the ones from the section above), but query the counterfactual network to predict preferences for the counterfactual NPS trajectories for each user. We then can compute validation losses and accuracies based on this counterfactual dataset that we pre-computed. The results for this experiment are in Fig. 14. Interestingly, we see that generally our network does somewhat worse than in the other settings considered. This is likely due to 1) the approximate nature of the smoothing estimate which is used to perform the counterfactual task, and 2) the task that requires estimation for preference-evolutions induced by a policy that is different from the one seen at training time.

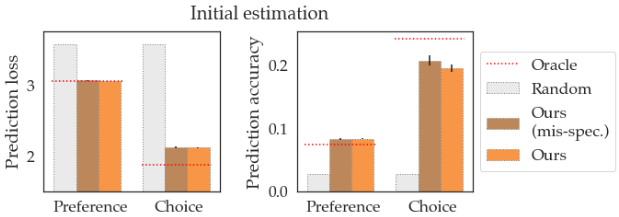

Figure 13: Equivalent setup to Fig. 3, except computed using the *initial* preference estimation network from Fig. 11. Error bars are standard deviations over 3 seeds.

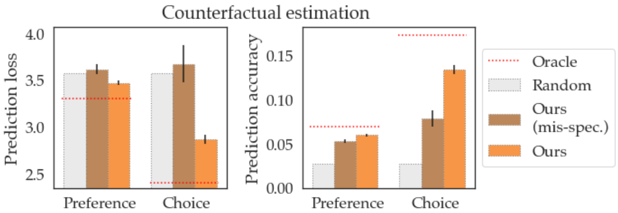

Figure 14: Similar setup to Fig. 3-Fig. 13, except computed using the counterfactual preference estimation network from Fig. 11. More details in Appendix I. Error bars are standard deviations over 3 seeds.