

## Overview of Appendices

- Appendix [A](#): Posterior Sampling via Autoregressive Generation (PS-AR) Algorithm
- Appendix [B](#): Extension to the Contextual Setting
- Appendix [C](#): Finite vs Infinite Population Formulations and Thompson Sampling Variants
- Appendix [D](#): Theoretical Results
- Appendix [E](#): Experiment Details
- Appendix [F](#): When is an Autoregressive Sequence Model a Valid Posterior

## A POSTERIOR SAMPLING VIA AUTOREGRESSIVE GENERATION (PS-AR) ALGORITHM

We present pseudo-code for the pre-training phase of our algorithm ([Algorithm 2](#)).

---

### Algorithm 2 Pretraining an autoregressive model

---

**Require:** Training data  $\mathcal{D}^{\text{hist}}$ , model class  $\{p_\theta\}_{\theta \in \Theta}$ , batch size  $b$

- 1: **while** not converged **do**
- 2:   Sample a minibatch  $\mathcal{D} = \{Z^{(a)}, R_{1:n}^{(a)}\}_{a \in \mathcal{A}}$  where  $\mathcal{A} \subset \mathcal{A}^{\text{hist}}$ ,  $|\mathcal{A}| = b$
- 3:   For each  $a \in \mathcal{A}$ , sample outcomes with replacement:
 
$$\underline{R}_1^{(a)}, \underline{R}_2^{(a)}, \dots, \underline{R}_T^{(a)} \mid \{Z^{(a)}, R_{1:n}^{(a)}\} \stackrel{i.i.d.}{\sim} \frac{1}{n} \sum_{i=1}^n \delta_{R_i^{(a)}}$$
- 4:   Define bootstrap-resampled minibatch  $\underline{\mathcal{D}} \leftarrow \{Z^{(a)}, \underline{R}_{1:T}^{(a)}\}_{a \in \mathcal{A}}$
- 5:   Compute loss  $\ell(p_\theta; \underline{\mathcal{D}})$  as defined in [equation 2](#)
- 6:   Backpropagate and take a gradient step to update  $\theta$
- 7: **end while**
- 8: **return**  $p_\theta$

---

### Practical Considerations for the Pre-Training Step ([Algorithm 2](#))

- In [Algorithm 2](#) line 3, instead of bootstrapping sequences of length  $T$ , for practical purposes we sometimes bootstrap samples sequences of length  $T_{\text{train}} < T$  if training to length  $T$  is very computationally expensive (we do this for our news recommendation experiments).
- While above we assume that  $\mathcal{D}^{\text{hist}}$  has sequences all of the same length  $n$ , in practice, this may not always be the case. Let  $n^{(a)}$  refer to the number of observations for article  $a \in \mathcal{A}^{\text{hist}}$ . In this case, we can easily replace  $n$  with  $n^{(a)}$  in line 3 of [Algorithm equation 2](#) (we do this for our news recommendation experiments).

### Practical Considerations for the Online Step ([Algorithm 1](#))

- As in [Section 6](#) for practical purposes, we may generate  $m$  outcomes rather than imputing all unobserved outcomes in  $[1 : T]$  to save computation. Specifically, replace lines 4-9 in [Algorithm 1](#) with [Algorithm 3](#) below.

**Algorithm 3** Truncated Autoregressive Posterior Generation**Require:** Autoregressive generative model  $p_\theta$ , timestep  $t$ , generation length  $m$ , action information $Z^{(a)}$ , previous rewards  $(R_i^{(a)} : i \notin M^{(a)})$  for that action1: **for**  $t' = 1, 2, \dots, m$  **do**2:     Sample (impute) missing reward:  $\hat{R}_{t'}^{(a)} \sim p_\theta(\cdot \mid Z^{(a)}, (R_i^{(a)} : i \notin M^{(a)}), \hat{R}_{1:t'-1}^{(a)})$ 3: **end for**4: Form imputed average reward using  $m$  generated outcomes:  $\hat{\mu}_t^{(a)} \leftarrow \sum_{t'=1}^m \hat{R}_{t'}^{(a)}$ 

## A.1 EMPIRICAL COMPARISONS OF PS-AR VARIANTS

**Examining Full Imputation vs Truncated Generation (Figure 7)** We empirically compare PS-AR (Algorithm 1), i.e., “full imputation”, with the computationally cheaper version of PS-AR that truncates generation to a maximum of length of  $m$  (Algorithm 1 with lines 4-7 replaced with Algorithm 3). We find that both versions of PS-AR perform well in practice and that the original PS-AR (Algorithm 1) and the  $m$ -truncated version (with  $m = 500$ ) have similar performance.

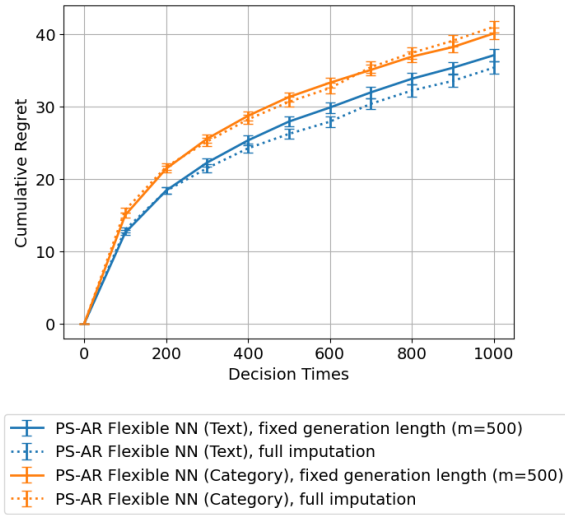


Figure 7: **Full imputation vs. truncated generation of future rewards.** Error bars are  $\pm 1$  s.e. averaged over 500 runs.

Specifically in Figure 7 we compare both versions of PS-AR on a news recommendation setting. In our experimental setup, we use two versions of the pretrained autoregressive sequence model  $p_\theta$ : FLEXIBLE NN (TEXT) and FLEXIBLE NN (CATEGORY) (see Appendix E.4 for more details). We run both versions of PS-AR with each of these two  $p_\theta$  models. We use  $T = 1000$ ,  $|\mathcal{A}^{\text{new}}| = 10$ , and the truncated version of PS-AR uses  $m = 500$ . We follow the procedure described in Appendix E.4 in forming the regret plots: we run 500 repetitions of each bandit algorithm and in each repetition we draw a new set of 10 actions/articles from the validation set to represent a “new task”.

**Examining Truncating Generation Length (Figure 8)** We examine the performance of our PS-AR algorithm for different generation truncation lengths  $m$  (Algorithm 1 with lines 4-7 replaced with Algorithm 3). Throughout all our previous experiments we use  $m = 500$ . In Figure 8, we examine the impact of varying  $m$  on the regret of the PS-AR with the FLEXIBLE NN (TEXT) sequence model in the news recommendation setting. We follow the procedure described in Appendix E.4 in forming the regret plots: we run 500 repetitions of each bandit algorithm and in each repetition we draw a new set of 10 actions/articles from the validation set to represent a “new task”. We find that increasing  $m$  reduces the regret of the algorithm; however, when  $m$  is sufficiently large, the benefit of increasing  $m$  further is negligible.

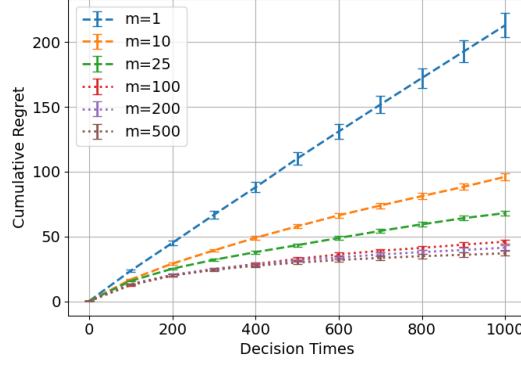


Figure 8: **Examining Truncating Generation Length** ( $|\mathcal{A}^{\text{new}}| = 10$ ). Error bars are  $\pm 1$  s.e. averaged over 500 runs.

## B EXTENSION TO THE CONTEXTUAL SETTING

In this section we discuss a preliminary approach to extend our algorithm to the setting with context features. In the news recommendation setting, the context features would represent user features.

**Data Generating Process.** In this setting, article features  $Z^{(a)}$  are drawn independently from  $P_Z$  over  $a \in \mathcal{A}^{\text{new}}$ . Independently of that, user contexts  $X_t$  are discrete and drawn i.i.d. from an unknown distribution  $P_X$ , i.e.,  $X_1, X_2, \dots, X_T \stackrel{i.i.d.}{\sim} P_X$ . Then,

$$R_t^{(a)} \mid Z^{(a)}, (X_{t'}, R_{t'}^{(a)})_{t'=t}^{t-1}, X_t \sim p^*(\cdot \mid Z^{(a)}, (X_{t'}, R_{t'}^{(a)})_{t'=t}^{t-1}, X_t). \quad (6)$$

Moreover,  $p^*$  is such that for any  $z$  and any permutation  $\sigma$  over  $\{1, \dots, T\}$ ,

$$(X_1, R_1^{(a)}), \dots, (X_T, R_T^{(a)}) \mid (Z^{(a)} = z) \stackrel{D}{=} (X_{\sigma(1)}, R_{\sigma(1)}^{(a)}), \dots, (X_{\sigma(T)}, R_{\sigma(T)}^{(a)}) \mid (Z^{(a)} = z),$$

where above we use  $\stackrel{D}{=}$  to denote equality in distribution.

The historical dataset follows the same data generating process. For shorthand, we use  $(X, Y^{(a)})_{1:n} := ((X_1, R_1^{(a)}), \dots, (X_n, R_n^{(a)}))$  to denote sequences of tuples. We denote the training set  $\mathcal{D}^{\text{hist}} = \{Z^{(a)}, (X, Y^{(a)})_{1:n}\}$  (for some  $n \leq T$ ). For each  $a \in \mathcal{A}^{\text{hist}}$ , we assume  $(X, Y^{(a)})_{1:n}$  is a completely at random subset of the tuples  $(X, Y^{(a)})_{1:T}$  where  $X_1, X_2, \dots, X_T \stackrel{i.i.d.}{\sim} P_X$  and  $R_{1:T}^{(a)}$  are sampled according to equation 6.

**Phase 1: Pretraining an auto-regressive model.** We train a sequence model analogously to Algorithm 2, however replace the training loss equation 2 with the following loss:

$$\ell(p_\theta; \mathcal{D}^{\text{hist}}) = \sum_{a \in \mathcal{A}^{\text{hist}}} \left[ - \sum_{t=1}^n \log p_\theta \left( R_t^{(a)} \mid Z^{(a)}, (X, Y^{(a)})_{1:t-1}, X_t^{(a)} \right) \right]. \quad (7)$$

In the contextual case, transformers are a natural choice for the sequence model architecture for  $p_\theta$ .

**Algorithm 4** Pretraining an autoregressive model with Context**Require:** Training data  $\mathcal{D}^{\text{hist}}$ , model class  $\{p_\theta\}_{\theta \in \Theta}$ , batch size  $b$ 1: **while** not converged **do**2: Sample a minibatch  $\mathcal{D} = \{Z^{(a)}, (X, Y^{(a)})_{1:n}\}_{a \in \mathcal{A}}$  where  $\mathcal{A} \subset \mathcal{A}^{\text{hist}}$ ,  $|\mathcal{A}| = b$ 3: For each  $a \in \mathcal{A}$ , sample outcomes with replacement:

$$(\underline{X}_1, \underline{Y}_1^{(a)}), (\underline{X}_2, \underline{Y}_2^{(a)}), \dots, (\underline{X}_T, \underline{Y}_T^{(a)}) \mid \{Z^{(a)}, (X, Y^{(a)})_{1:n}\} \stackrel{i.i.d.}{\sim} \frac{1}{n} \sum_{i=1}^n \delta_{(X_i, R_i^{(a)})}$$

Above,  $\frac{1}{n} \sum_{i=1}^n \delta_{(X_i, R_i^{(a)})}$  denotes the empirical distribution of  $(X, Y^{(a)})_{1:n}$ .4: Define bootstrap-resampled minibatch  $\underline{\mathcal{D}} \leftarrow \{Z^{(a)}, (\underline{X}, \underline{Y}^{(a)})_{1:T}\}_{a \in \mathcal{A}}$ 5: Compute loss  $\ell(p_\theta; \underline{\mathcal{D}})$  using equation 76: Backpropagate and take a gradient step to update  $\theta$ 7: **end while**8: **return**  $p_\theta$ 

**Phase 2: Online decision-making via autoregressive generation** Online decision-making with the pre-trained  $p_\theta$  sequence model can be made using Algorithm 5 below. Similar to the version without context, it generates missing outcomes.

**Algorithm 5** Posterior Sampling via Autoregressive Generation (PS-AR) with Context**Require:** Autoregressive generative model  $p_\theta$ , actions  $\mathcal{A}^{\text{new}}$  with  $\{Z^{(a)}\}_{a \in \mathcal{A}^{\text{new}}}$ 1: Initialize list of missing entries  $M^{(a)} \leftarrow [1, \dots, T]$  for each  $a \in \mathcal{A}^{\text{new}}$ 2: **for**  $t = 1, \dots, T$  **do**3: Observe user context  $X_t$  and set  $x \leftarrow X_t$ 4: **for**  $a \in \mathcal{A}^{\text{new}}$  **do**5: **for**  $\tau \in M^{(a)}$  **do**6: Impute reward:  $\hat{R}_\tau^{(a)} \sim p_\theta(\cdot \mid Z^{(a)}, (X_i, R_i^{(a)})_{i \notin M^{(a)}}, (X_i, \hat{R}_i^{(a)})_{i \in M^{(a)}})$ 7: **end for**8: Form average:  $\hat{\mu}_t^{(a)} \leftarrow \frac{1}{\sum_{\tau=1}^T \mathbb{1}_{X_\tau=x}} \{ \sum_{\tau \notin M^{(a)}} R_\tau^{(a)} \mathbb{1}_{X_\tau=x} + \sum_{\tau \in M^{(a)}} \hat{R}_\tau^{(a)} \mathbb{1}_{X_\tau=x} \}$ 9: **end for**10: Select action  $A_t \leftarrow \arg \max_{a \in \mathcal{A}^{\text{new}}} \{\hat{\mu}_t^{(a)}\}$  (break ties deterministically)11: Remove  $t$  from the list of missing entries  $M^{(A_t)}$ 12: Observe reward  $R_t \leftarrow R_t^{(A_t)}$  from action  $A_t$ .13: **end for**

## C FINITE VS INFINITE POPULATION FORMULATIONS AND THOMPSON SAMPLING VARIANTS

This section discusses the intimate connections between (large) finite-population formulations that were discussed in the main body of the paper and infinite-population formulations that are more common in the Bayesian bandit literature. We do this in the special case of the Bayesian mixture model from equation 1.

We emphasize that **from our perspective, the main advantages or disadvantages of the finite population view are conceptual**. In terms of advantages: (1) the definitions do not require any explicit assumptions around mixture modeling or latent variables. and (2) The finite nature of the problem lets us visualize the procedure as in Figure 3, without abstract reference to limits across infinite sequences.

### C.1 REVIEW OF THOMPSON SAMPLING IN INFINITE POPULATIONS, WITH MIXTURE MODELS.

Thompson sampling is most often defined for a Bayesian mixture model, e.g., as in equation 1. Following that example, we consider in the subsection the canonical example of exchangeable se-

quences: a mixture model wherein the outcomes are i.i.d conditioned on a latent variable  $U^{(a)}$ . That is,  $p^*(R_1^{(a)}, \dots, R_t^{(a)} | Z^{(a)}) = \int \prod_{t=1}^T P(R_t^{(a)} | Z^{(a)}, U^{(a)} = u) P(U^{(a)} = u) du$ . The unknown latent variable represents the decision-maker’s uncertainty about an action’s performance.

The literature typically defines the “true arm means” as

$$\mu_\infty^{(a)} = \int r \cdot P(r | Z^{(a)}, U^{(a)}) dr.$$

The subscript highlights that this has the interpretation of a long-run average reward across an infinite population of users (or infinite set of rounds). By the law of large numbers (applied conditional on  $(Z^{(a)}, U^{(a)})$ , one has

$$\mu_\infty^{(a)} = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T R_t^{(a)}.$$

The true best arm is defined as

$$A_\infty^* \in \arg \max_{a \in \mathcal{A}^{\text{new}}} \mu_\infty^{(a)}$$

Randomness in the latent parameters ( $U^{(a)}$ ) means  $\mu_\infty^{(a)}$  and  $A_\infty^*$  are random variables whose realizations are uncertain even given the history  $\mathcal{H}_{t-1}$ . Thompson sampling selects an action by probability matching on  $A_\infty^*$ , defined by the property

$$\mathbb{P}(A_t = a | \mathcal{H}_{t-1}) = \mathbb{P}(A_\infty^* = a | \mathcal{H}_{t-1}) \quad \text{for all } a \in \mathcal{A}^{\text{new}}. \quad (8)$$

Per-period Bayesian regret over  $T$  periods is defined as

$$\mathbb{E} \left[ \frac{1}{T} \sum_{t=1}^T \left( R_t^{(A^*)} - R_t^{(A_t)} \right) \right] \quad (9)$$

## C.2 THOMPSON SAMPLING IN FINITE POPULATIONS

As in the body of our paper, one can define the true mean of a finite population as

$$\mu_T^{(a)} = \frac{1}{T} \sum_{t=1}^T R_t^{(a)}.$$

The true best arm for this finite population is defined as

$$A_T^* \in \arg \max_{a \in \mathcal{A}^{\text{new}}} \mu_T^{(a)}$$

As in Lemma 2, Thompson sampling selects an action by probability matching on the (finite-population) optimal action  $A_T^*$ , defined by the property

$$\mathbb{P}(A_t = a | \mathcal{H}_{t-1}) = \mathbb{P}(A_T^* = a | \mathcal{H}_{t-1}) \quad \text{for all } a \in \mathcal{A}^{\text{new}}. \quad (10)$$

Per-period Bayesian regret over  $T$  periods is defined as

$$\mathbb{E} \left[ \frac{1}{T} \sum_{t=1}^T \left( R_t^{(A_T^*)} - R_t^{(A_t)} \right) \right] \quad (11)$$

It is not hard to show that equation 11 is a more stringent notion of regret than in equation 9 since  $\frac{1}{T} \sum_{t=1}^T R_t^{(A_T^*)} \geq \frac{1}{T} \sum_{t=1}^T R_t^{(A_\infty^*)}$  by definition of  $A_T^*$ . Both definitions are widely used, with the more stringent finite-population version being more common in the adversarial bandit literature; see [Lattimore and Szepesvári \(2019\)](#).

## C.3 THE GAP BETWEEN FINITE AND INFINITE POPULATION FORMULATIONS IS SMALL

We analyze the gap between the two formulations in the case of a mixture model. Let  $\mathcal{U}^{\text{new}} = \{U^{(a)} : a \in \mathcal{A}^{\text{new}}\}$  and recall  $\mathcal{Z}^{\text{new}} = \{Z^{(a)} : a \in \mathcal{A}^{\text{new}}\}$ . By a sub-Gaussian maximal inequality

$$\mathbb{E} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left| \mu_\infty^{(a)} - \mu_T^{(a)} \right| \right] = \mathbb{E} \left[ \mathbb{E} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left| \mu_\infty^{(a)} - \mu_T^{(a)} \right| \mid \mathcal{Z}^{\text{new}}, \mathcal{U}^{\text{new}} \right] \right] \leq \sqrt{\frac{2 \log(|\mathcal{A}^{\text{new}}|)}{T}}.$$

To justify the last inequality, note that since the function  $R$  takes values in  $[0, 1]$ ,  $R_t^{(a)} - \mu_\infty^{(a)}$  is subgaussian with variance proxy 1, conditional on  $\mathcal{Z}^{\text{new}}, \mathcal{U}^{\text{new}}$  (by Hoeffding's Lemma). Since it is the average of independent sub-Gaussian random variables,  $\mu_\infty^{(a)} - \mu_T^{(a)}$  is subgaussian with variance proxy  $\frac{1}{T}$ , conditional on  $\mathcal{Z}^{\text{new}}, \mathcal{U}^{\text{new}}$ . The last step follows then from applying the subgaussian maximal inequality, conditional on  $\mathcal{Z}^{\text{new}}, \mathcal{U}^{\text{new}}$ .

It follows easily that the infinite population optimum  $A_\infty^*$  is near optimal for finite populations:

$$0 \leq \mathbb{E} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \mu_T^{(a)} - \mu_T^{(A_\infty^*)} \right] \leq 2\sqrt{\frac{2 \log(|\mathcal{A}^{\text{new}}|)}{T}}.$$

Analogously, the finite population optimum is near-optimal in infinite populations:

$$0 \leq \mathbb{E} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \mu_\infty^{(a)} - \mu_\infty^{(A_T^*)} \right] \leq 2\sqrt{\frac{2 \log(|\mathcal{A}^{\text{new}}|)}{T}}.$$

Supported by this theory, we do not focus on the distinction between  $A_T^*$  and  $A_\infty^*$  in our developments.

#### C.4 SIMILAR INSIGHTS IN EMPIRICAL RESULTS

Some empirical insight can also be gleaned from Figure 7 in Appendix C. The implementation that performs full imputation can be interpreted as Thompson sampling for a finite population. The implementation that performs forward generation of fixed-length  $m$  does not include past observed rewards in the average. For very large  $m$ , it is a direct approximation to infinite-horizon Thompson sampling. We can see in Figure 7 that these implementations have very similar performance.

## D THEORETICAL RESULTS

### D.1 THE MEAN $\mu_T^{(a)}$ APPROACHES $\mu^{(a)}$ FOR LARGE $T$ .

**Proposition 2** (Forming a Posterior Sample via Autoregressive Generation). *Under the Bayesian mixture model example from equation 1:*

$$U^{(a)} \sim P(U^{(a)} \in \cdot \mid Z^{(a)}) \quad \text{then,} \quad R_1^{(a)}, \dots, R_T^{(a)} \mid U^{(a)} \stackrel{i.i.d.}{\sim} P(R_t^{(a)} \in \cdot \mid U^{(a)}, Z^{(a)}).$$

The following holds for  $\mu^{(a)} = \frac{1}{T} \sum_{t=1}^T R_t^{(a)}$  and  $\mu^{(a)} = \int_r r \cdot P(R_t^{(a)} = r \mid U^{(a)}, Z^{(a)}) dr$ :

$$\mathbb{P} \left( \lim_{T \rightarrow \infty} \mu_T^{(a)} = \mu^{(a)} \mid U^{(a)}, Z^{(a)} \right) = 1 \quad \text{with probability 1.}$$

*Proof.* By equation 1, conditional on  $U^{(a)}, Z^{(a)}$ , the generated rewards  $R_1^{(a)}, \dots, R_T^{(a)}$  are i.i.d. Also note that  $R_t^{(a)} \in [0, 1]$  based on our problem setup. Thus, by the Strong Law of Large numbers for i.i.d. random variables, conditional on almost any draw of  $U^{(a)}, Z^{(a)}$ , we have that

$$\mathbb{P} \left( \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T R_t^{(a)} = \mu^{(a)} \mid U^{(a)}, Z^{(a)} \right) = 1 \quad \text{with probability 1.}$$

□

### D.2 TO MINIMIZE LOSS $p_\theta$ NEEDS TO APPROXIMATE $p^*$

The next lemma is a standard result connecting the excess expected loss of a sequence model  $p_\theta$  to its KL divergence from the true sequence model  $p^*$ . Recall the expected loss of a sequence model  $p_\theta$  is denoted  $\ell(p_\theta)$ , defined in equation 3. To (nearly) minimize loss,  $p_\theta$  the learner needs to closely approximate the true sequence model  $p^*$ .

**Lemma 1.** *For any sequence model  $p_\theta$ ,*

$$\ell(p_\theta) = \ell(p^*) + \mathbb{E}_{Z^{(a)} \sim P_Z} \left[ D_{\text{KL}} \left( p^*(R_1^{(a)}, \dots, R_T^{(a)} \mid Z^{(a)}) \parallel p_\theta(R_1^{(a)}, \dots, R_T^{(a)} \mid Z^{(a)}) \right) \right].$$

*Proof.* By the definition of the expected loss in equation [3](#) and the chain rule of KL divergence:

$$\begin{aligned} & \ell_n(p_\theta) - \ell_n(p^*) \\ &= \mathbb{E} \left[ - \sum_{t=1}^n \log p_\theta(R_t^{(a)} \mid Z^{(a)}, R_{1:t-1}^{(a)}) \right] - \mathbb{E} \left[ - \sum_{t=1}^n \log p^*(R_t^{(a)} \mid Z^{(a)}, R_{1:t-1}^{(a)}) \right] \\ &= \text{KL}(\mathbb{P}_{p^*}(R_1^{(a)}, \dots, R_n^{(a)} \mid Z^{(a)}) \parallel \mathbb{P}_{p_\theta}(R_1^{(a)}, \dots, R_n^{(a)} \mid Z^{(a)})) \\ &= \mathbb{E}_{Z^{(a)} \sim P_Z} [\text{KL}(\mathbb{P}_{p^*}(R_1^{(a)}, \dots, R_n^{(a)} \mid Z^{(a)}) \parallel \mathbb{P}_{p_\theta}(R_1^{(a)}, \dots, R_n^{(a)} \mid Z^{(a)}))]. \end{aligned}$$

The final equality is the definition of the KL divergence between conditional distributions.  $\square$

### D.3 FORMALLY INTERPRETING PS-AR AS THOMPSON (POSTERIOR) SAMPLING

This subsection reveals that the generated/imputed action means faithfully represent uncertainty and that PS-AR is akin to Thompson sampling (a.k.a. posterior sampling), which selects actions proportionally to the probability that they are optimal. Specifically, Lemma [2](#) formally shows that the imputed mean  $\hat{\mu}_t^{(a)}$  from PS-AR is a posterior sample of the mean reward  $\mu^{(a)}$ , and the action  $A_t$  selected by PS-AR is a posterior sample of the optimal action  $A^*$ , where

$$\mu^{(a)} := \frac{1}{T} \sum_{t=1}^T R_t^{(a)} \quad \text{and} \quad A^* := \arg \max_{a \in \mathcal{A}^{\text{new}}} \{\mu^{(a)}\}, \quad (12)$$

with ties in the argmax broken by the same rule as in Algorithm [1](#).  $A^*$  is the benchmark action against which regret is evaluated in equation [4](#). For simplicity, Lemma [2](#) is stated under the assumption that PS-AR uses the optimal sequence model  $p^*$ . We use  $\mathcal{H}_t := (\{Z^{(a)}\}_{a \in \mathcal{A}^{\text{new}}}, A_1, R_1, \dots, A_t, R_t)$  to denote the history up to time  $t$ .

**Lemma 2.** *Under Assumption [1](#) for Algorithm [1](#) applied with  $p_\theta = p^*$ , for all  $a \in \mathcal{A}^{\text{new}}$ , with probability 1,*

$$\mathbb{P}(\hat{\mu}_t^{(a)} = \cdot \mid \mathcal{H}_{t-1}) = \mathbb{P}(\mu^{(a)} = \cdot \mid \mathcal{H}_{t-1}) \quad \text{and} \quad \mathbb{P}(A_t = a \mid \mathcal{H}_{t-1}) = \mathbb{P}_{p_\theta}(A^* = a \mid \mathcal{H}_{t-1}).$$

Lemma [2](#) has a concise proof (below). Both the population mean and the best action are functions of the table of the potential outcomes  $\{R_{1:T}^{(a)}\}_{a \in \mathcal{A}^{\text{new}}}$ . Sampling missing rewards from their posterior distribution ensures that functions of the imputed table also follow the posterior distribution. One difference with usual presentations is that  $\mu^{(a)}$  averages over a very large, but finite population of users (or rounds). We do not view this as a practically significant detail. The generalization of Lemma [2](#) to the setting in which  $p_\theta \neq p^*$  is in Appendix [D.6.2](#).

*Proof.* At decision time  $t$ , suppose in the history  $\mathcal{H}_{t-1}$  a particular action  $a \in \mathcal{A}^{\text{new}}$  we have not observed (missing) rewards from users  $M^{(a)} \subseteq \{1, 2, \dots, t-1\}$ . For the function  $f(\{R_i\}_{i=1}^T) = T^{-1} \sum_{i=1}^T R_i$ , one has

$$\mu^{(a)} = f(\{R_i^{(a)}\}_{i=1}^T) \quad \text{and} \quad \hat{\mu}_t^{(a)} = f(\{R_i^{(a)} : i \notin M^{(a)}, i \in [1:t-1]\} \cup \{\hat{R}_i^{(a)} : i \in M^{(a)}\}).$$

where  $\{\hat{R}_i^{(a)} : i \in M^{(a)}\}$  are drawn according to Algorithm [1](#) applied with sequence model  $p_\theta = p^*$ . The result that

$$\mathbb{P}(\hat{\mu}_t^{(a)} = \cdot \mid \mathcal{H}_{t-1}) = \mathbb{P}_{p_\theta}(\mu^{(a)} = \cdot \mid \mathcal{H}_{t-1})$$

follows immediately since  $\{R_i^{(a)} : i \notin M^{(a)}, i \in [1:t-1]\}$  are non-random conditioned on  $\mathcal{H}_{t-1}$  and  $\mathbb{P}(\{\hat{R}_i^{(a)} : i \in M^{(a)}\} = \cdot \mid \mathcal{H}_{t-1}) = \mathbb{P}(\{R_t^{(a)} : t \in M^{(a)}\} = \cdot \mid \mathcal{H}_{t-1})$  with probability 1. The proof of the analogous result for  $A^*$  is identical.  $\square$

### D.4 PROOF OF THEOREM [1](#)

**Theorem [1](#).** *Let  $O^{\text{new}} := \{Z^{(a)}, R_{1:T}^{(a)}\}_{a \in \mathcal{A}^{\text{new}}}$  denote the potential outcomes table. Independent of  $O^{\text{new}}$ , let  $\xi \sim \text{Uniform}[0, 1]$ . Under Assumption [1](#) for real-valued functions  $f$  of  $O^{\text{new}}$  and  $\xi$ ,*

$$\sup_{f: \|f\|_\infty \leq 1} \left| \underbrace{\mathbb{E}_{p^*}[f(O^{\text{new}}, \xi)]}_{\text{Real Distribution}} - \underbrace{\mathbb{E}_{p_\theta}[f(O^{\text{new}}, \xi)]}_{\text{Simulated Distribution}} \right| \leq \underbrace{\sqrt{(|\mathcal{A}^{\text{new}}|/2) \{\ell(p_\theta) - \ell(p^*)\}}}_{\text{Penalty for sub-optimal simulator}}.$$



*Proof.* Note that

$$\begin{aligned}
& \sup_{f: \|f\|_\infty \leq 1} \left\{ \mathbb{E}_{p^*} [f(O^{\text{new}}, \xi)] - \mathbb{E}_{p_\theta} [f(O^{\text{new}}, \xi)] \right\} \\
& \stackrel{(i)}{\leq} \sqrt{\frac{1}{2} \text{KL}(\mathbb{P}_{p^*}(O^{\text{new}}, \xi) \parallel \mathbb{P}_{p_\theta}(O^{\text{new}}, \xi))} \\
& \stackrel{(ii)}{=} \sqrt{\underbrace{\frac{1}{2} \text{KL}(\mathbb{P}_{p^*}(\xi) \parallel \mathbb{P}_{p_\theta}(\xi))}_{=0} + \frac{1}{2} \text{KL}(\mathbb{P}_{p^*}(O^{\text{new}} \mid \xi) \parallel \mathbb{P}_{p_\theta}(O^{\text{new}} \mid \xi))} \\
& \stackrel{(iii)}{=} \sqrt{\frac{1}{2} \cdot \text{KL}(\mathbb{P}_{p^*}(O^{\text{new}}) \parallel \mathbb{P}_{p_\theta}(O^{\text{new}}))} \\
& \stackrel{(iv)}{=} \sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2} \cdot \text{KL}(\mathbb{P}_{p^*}(Z^{(a)}, R_{1:T}^{(a)}) \parallel \mathbb{P}_{p_\theta}(Z^{(a)}, R_{1:T}^{(a)}))} \\
& \stackrel{(v)}{=} \sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2} \cdot \underbrace{\text{KL}(\mathbb{P}_{p^*}(Z^{(a)}) \parallel \mathbb{P}_{p_\theta}(Z^{(a)}))}_{=0} + \text{KL}(\mathbb{P}_{p^*}(R_{1:T}^{(a)} \mid Z^{(a)}) \parallel \mathbb{P}_{p_\theta}(R_{1:T}^{(a)} \mid Z^{(a)}))} \\
& \stackrel{(vi)}{=} \sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2} \{\ell(p_\theta) - \ell(p^*)\}}
\end{aligned}$$

- (i) holds by Fact 9 in [\(Russo and Van Roy, 2016\)](#) (which uses Pinsker’s inequality).
- (ii) holds the chain rule for Kullback Liebler Divergence.
- (iii) holds because  $\xi$  is and  $O^{\text{new}}$  are independent.
- (iv) and (vi) hold again because the  $(Z^{(a)}, R_{1:T}^{(a)})$  are i.i.d. across  $a \in \mathcal{A}^{\text{new}}$  and the chain rule for Kullback Liebler Divergence.
- (vi) holds by Lemma [1](#).

□

## D.5 BOUNDING THE DEPLOYMENT REGRET IN TERMS OF REGRET ON A SIMULATOR

**Proposition 3.** For any policy  $\pi$ ,

$$\underbrace{\Delta(\pi; p^*)}_{\text{Deployment regret}} \leq \underbrace{\Delta(\pi; p_\theta)}_{\text{Regret under simulator}} + \underbrace{\sqrt{(|\mathcal{A}^{\text{new}}|/2) \{\ell(p_\theta) - \ell(p^*)\}}}_{\text{Penalty for sub-optimal simulator}}. \quad (13)$$

Formally, any policy  $\pi$  can be expressed a function that maps a history  $\mathcal{H}_{t-1}$  and an exogenous random seed  $\xi$  to an action as

$$A_t = \pi(\mathcal{H}_{t-1}, \xi). \quad (14)$$

The random seed allows for algorithmic randomness in action selection and is assumed to be independent of the draws of article features and potential outcomes  $(Z^{(a)}, R_{1:T}^{(a)})_{a \in \mathcal{A}^{\text{new}}}$ .

The essence of the proof is to recognize that one could write a simulator that first randomly drew the environment “sample path”  $(Z^{(a)}, R_{1:T}^{(a)})_{a \in \mathcal{A}^{\text{new}}}$  and the algorithm seed  $\xi$ , and then implemented a completely deterministic sequence of operations to calculate the regret an algorithm incurs with that sample path and seed. Mathematically, the simulator is a function, (written as  $g(\cdot)$  in the proof). We can view mis-specification of the sequence model as mis-specifying the distribution of the sample path draws used in the the simulator. We use information-theoretic tools to bound the impact this distributional change on the inputs to the simulator can have on the distribution of outputs of the simulator (e.g. regret).



*Proof.* This proof will show that for any policy  $\pi$ ,

$$\Delta(\pi; p^*) \leq \Delta(\pi; p_\theta) + \sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2}} \{\ell(p_\theta) - \ell(p^*)\}.$$

Note for any policy  $\pi$ , by the triangle inequality,

$$\Delta(\pi; p^*) \leq |\Delta(\pi; p^*) - \Delta(\pi; p_\theta)| + |\Delta(\pi; p_\theta)|$$

The remainder of the proof will focus on bounding the first term above. Let  $O^{\text{new}} := \{Z^{(a)}, R_{1:T}^{(a)} : a \in \mathcal{A}^{\text{new}}\}$  denote a draw of all article features and potential outcomes.

The absolute difference in regret can be written as

$$\begin{aligned} & |\Delta(\pi; p_\theta) - \Delta(\pi; p^*)| \\ &= \left| \mathbb{E}_{p^*, \pi} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left\{ \frac{1}{T} \sum_{t=1}^T R_t^{(a)} \right\} - \frac{1}{T} \sum_{t=1}^T R_t \right] - \mathbb{E}_{p_\theta, \pi} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left\{ \frac{1}{T} \sum_{t=1}^T R_t^{(a)} \right\} - \frac{1}{T} \sum_{t=1}^T R_t \right] \right| \\ &= \left| \mathbb{E}_{p^*} [g(O^{\text{new}}, \xi)] - \mathbb{E}_{p_\theta} [g(O^{\text{new}}, \xi)] \right|, \end{aligned}$$

where  $g$  is a function that determines the algorithm's regret as a function of the potential outcomes and the external seed  $\xi$  that used to induce randomness in action actions. That is,  $g(\{Z^{(a)}, R_{1:T}^{(a)}\}_{a \in \mathcal{A}^{\text{new}}}, \xi) := \frac{1}{T} \sum_{t=1}^T \{R_t^{(A^*)} - R_t\}$ .

Finally, since  $g$  is such that  $\|g\|_\infty \leq 1$ , since  $R_t \in [0, 1]$  with probability 1 by assumption, by Theorem [1](#) we have that

$$|\mathbb{E}_{p^*} [g(O^{\text{new}}, \xi)] - \mathbb{E}_{p_\theta} [g(O^{\text{new}}, \xi)]| \leq \sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2}} \{\ell(p_\theta) - \ell(p^*)\}.$$

□

## D.6 PROOF OF PROPOSITION [1](#)

### D.6.1 A USEFUL DEFINITION

Under our data generating process  $p^*$  is exchangeable. To prove Proposition [1](#) we want to view posterior sampling by auto-regressive sampling (Algorithm [1](#)) as a proper implementation of Thompson sampling, with approximation coming solely from the incorrect use of a sequence model  $p_\theta$ . To make this rigorous, we need to define a slightly different order in which potential outcomes are revealed to accommodate non-exchangeable  $p_\theta$  models.

**Definition 1** (An alternative outcome revelation order). *A (possibly non-exchangeable) sequence model  $p_\theta$  introduces an alternative way of revealing potential outcomes. Recall, independently for each arm  $a \in \mathcal{A}^{\text{new}}$ , nature samples arm features  $Z^{(a)} \sim P_Z$ ; then it samples  $R_{1:T}^{(a)} \mid Z^{(a)} \sim p_\theta(\cdot \mid Z^{(a)})$ . If arm  $A_t = a$  is selected at time  $t$  and this is the  $k^{\text{th}}$  time that arm is chosen, then  $R_k^{(A_t)}$  is revealed. We use  $\tilde{R}_t \leftarrow R_k^{(A_t)}$  and  $\tilde{\mathcal{H}}_t := (\{Z^{(a)}\}_{a \in \mathcal{A}^{\text{new}}}, A_1, \tilde{R}_1, \dots, A_t, \tilde{R}_t)$ .*

We note that this data generating process is simply specifying the order in which outcomes from the sequence model are revealed to the decision-maker. Namely, we view  $\tilde{R}_t^{(a)}$  as the potential outcome of the  $t^{\text{th}}$  play of arm  $a$  whereas the main body of the paper views  $R_t^{(a)}$  as the potential outcome for the  $t^{\text{th}}$  user/period. Under an exchangeable sequence models, order is irrelevant and the two data generating processes are mathematically equivalent. **Note that defining the  $\tilde{R}_t$ 's is a proof technique (specifically to show Proposition [1](#)), and not part of the model of the problem.**

It will also be useful to note an alternative definition of regret under this alternative approach to revealing potential outcomes:

$$\tilde{\Delta}(\pi; p) := \mathbb{E}_{p, \pi} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left\{ \frac{1}{T} \sum_{t=1}^T R_t^{(a)} \right\} - \frac{1}{T} \sum_{t=1}^T \tilde{R}_t \right]. \quad (15)$$

Note that since  $p^*$  is an exchangeable sequence model,  $\tilde{\Delta}(\pi, p^*) = \Delta(\pi, p^*)$  for any policy  $\pi$ . Defining  $\tilde{\Delta}$  is solely to accomodate non-exchangeable sequence models.

### D.6.2 A HELPFUL LEMMA

Our proof of Proposition 1 relies on a generalization of Lemma 2 that describes precisely how Algorithm 1 is exactly Thompson Sampling (i.e., probability matching) when  $p_\theta$  is not exchangeable.

**Lemma 3.** *Under Algorithm 1 applied with  $p_\theta$ ,*

$$\mathbb{P}(\hat{\mu}_t^{(a)} = \cdot \mid \tilde{\mathcal{H}}_{t-1}) = \mathbb{P}_{p_\theta}(\mu^{(a)} = \cdot \mid \tilde{\mathcal{H}}_{t-1}) \quad (16)$$

and for all  $a \in \mathcal{A}^{\text{new}}$ ,

$$\mathbb{P}(A_t = a \mid \tilde{\mathcal{H}}_{t-1}) = \mathbb{P}_{p_\theta}(A^* = a \mid \tilde{\mathcal{H}}_{t-1}). \quad (17)$$

*Proof.* We use the notation of Definition 1. Let  $N_t^{(a)} := \sum_{i=1}^t \mathbb{1}(A_i = a)$  denote the number of times arm  $a$  was played up to and including period  $t$ . Then, the observation at time  $t$  is

$$\tilde{R}_t \leftarrow R_{N_t^{(A_t)}}^{(A_t)}.$$

For the function  $f(\{R_i\}_{i=1}^T) = T^{-1} \sum_{i=1}^T R_i$ , one has

$$\mu^{(a)} = f(\{R_i^{(a)}\}_{i=1}^T) \quad \text{and} \quad \hat{\mu}_t^{(a)} = f(\{R_1^{(a)}, \dots, R_{N_t^{(a)}-1}^{(a)}\} \cup \{R_{N_t^{(a)}}^{(a)}, \dots, R_T^{(a)}\})$$

where  $(R_{N_t^{(a)}}^{(a)}, \dots, R_T^{(a)}) \sim p_\theta(\cdot \mid Z^{(a)}, R_{1:N_t^{(a)}-1}^{(a)})$  represent the generated outcomes drawn according to Algorithm 1 applied with sequence model  $p_\theta$ . Property equation 16 follows immediately since  $\{R_1^{(a)}, \dots, R_{N_t^{(a)}-1}^{(a)}\}$  are non-random conditioned on the history  $\tilde{\mathcal{H}}_{t-1}$  and

$\mathbb{P}((R_{N_t^{(a)}}^{(a)}, \dots, R_T^{(a)}) = \cdot \mid \tilde{\mathcal{H}}_{t-1}) = \mathbb{P}((R_{N_t^{(a)}}^{(a)}, \dots, R_T^{(a)}) = \cdot \mid \tilde{\mathcal{H}}_{t-1})$  with probability 1. The proof of equation 17 is identical.  $\square$

### D.6.3 MAIN PROOF OF PROPOSITION 1

**Proposition 1.** *Under Assumption 1 for PS-AR (Algorithm 1) applied with  $p_\theta$  (denoted  $\pi_{\text{PS-AR}}(p_\theta)$ ),*

$$\Delta(\pi_{\text{PS-AR}}(p_\theta); p^*) \leq \underbrace{\sqrt{\frac{|\mathcal{A}^{\text{new}}| \log(|\mathcal{A}^{\text{new}}|)}{2T}}}_{\text{Regret bound for Thompson sampling}} + \underbrace{\sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2} \{\ell(p_\theta) - \ell(p^*)\}}}_{\text{Penalty for sub-optimal prediction}}.$$

In light of Theorem 1, the following proof of Proposition 1 is largely review of an information-theoretic analysis of Thompson sampling due to Russo and Van Roy (2016). It was observed by Bubeck et al. (2015); Bubeck and Eldan (2016) that this analysis applied without modification to analyze regret with respect to the best fixed action ( $A^*$ ) even in nonstationary environments (e.g. non-exchangeable models  $p_\theta$  as in Definition 1).

*Proof.* This proof will show that for any sequence model  $p_\theta$ ,

$$\Delta(\pi_{\text{PS-AR}}(p_\theta); p^*) \leq \sqrt{\frac{|\mathcal{A}^{\text{new}}| \log(|\mathcal{A}^{\text{new}}|)}{2T}} + \sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2} \{\ell(p_\theta) - \ell(p^*)\}}.$$

Note that by Theorem 1, we can show that

$$\tilde{\Delta}(\pi_{\text{PS-AR}}(p_\theta); p^*) \leq \tilde{\Delta}(\pi_{\text{PS-AR}}(p_\theta); p_\theta) + \sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2} \{\ell(p_\theta) - \ell(p^*)\}}. \quad (18)$$

Specifically the argument to show equation 18 above is equivalent to that used to prove Proposition 3—all that needs to be done is to replace all  $\Delta$ 's with  $\tilde{\Delta}$ 's and replace all  $R_t$ 's with  $\tilde{R}_t$ 's in the proof.

Note that since  $p^*$  is an exchangeable model by Assumption 1 for any permutation  $\sigma$  over  $T$  elements,

$$p^*(R_1^{(a)}, \dots, R_t^{(a)} \mid z) = p^*(R_{\sigma(1)}^{(a)}, \dots, R_{\sigma(t)}^{(a)} \mid z).$$

The above implies that the average regret achieved by a policy  $\pi$  under the two different approaches of revealing potential outcomes are equivalent, i.e.,

$$\begin{aligned}\tilde{\Delta}(\pi_{\text{PS-AR}}(p_\theta); p^*) &= \mathbb{E}_{p^*, \pi_{\text{PS-AR}}(p_\theta)} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left\{ \frac{1}{T} \sum_{t=1}^T R_t^{(a)} \right\} - \frac{1}{T} \sum_{t=1}^T \tilde{R}_t \right] \\ &\stackrel{(i)}{=} \mathbb{E}_{p^*, \pi_{\text{PS-AR}}(p_\theta)} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left\{ \frac{1}{T} \sum_{t=1}^T R_t^{(a)} \right\} - \frac{1}{T} \sum_{t=1}^T R_t \right] = \Delta(\pi_{\text{PS-AR}}(p_\theta); p^*)\end{aligned}$$

Equality (i) above since by Assumption 1,  $R_1^{(a)}, R_2^{(a)}, \dots, R_T^{(a)}$  are exchangeable conditional on  $\mathcal{Z}^{(a)}$ ; thus changing the order with which rewards are revealed (as described in Definition 1) does not change the expected value. Thus, combined with equation 18, we have that

$$\Delta(\pi_{\text{PS-AR}}(p_\theta); p^*) \leq \tilde{\Delta}(\pi_{\text{PS-AR}}(p_\theta); p_\theta) + \sqrt{\frac{|\mathcal{A}^{\text{new}}|}{2}} \{\ell(p_\theta) - \ell(p^*)\}.$$

All that remains is to bound  $\tilde{\Delta}(\pi_{\text{PS-AR}}(p_\theta); p_\theta)$ .

**Bounding  $\tilde{\Delta}(\pi_{\text{PS-AR}}(p_\theta); p_\theta)$ .** We bound  $\tilde{\Delta}(\pi_{\text{PS-AR}}(p_\theta); p_\theta)$  by combining the probability matching result of Lemma 2 with Thompson sampling regret bound techniques from Russo and Van Roy (2016). By the proof of Proposition 1 of Russo and Van Roy (2016) (which is general and applies to all algorithms  $\pi$ ),

$$\begin{aligned}\mathbb{E}_{p_\theta, \pi_{\text{PS-AR}}(p_\theta)} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left\{ \frac{1}{T} \sum_{t=1}^T R_t^{(a)} \right\} - \frac{1}{T} \sum_{t=1}^T \tilde{R}_t \middle| \mathcal{Z}^{\text{new}} \right] &\leq \sqrt{\frac{H_{p_\theta}(A^* | \mathcal{Z}^{\text{new}}) \cdot \Gamma}{T}} \quad \text{w.p. 1} \\ &\leq \sqrt{\frac{\log(|\mathcal{A}^{\text{new}}|) \cdot \Gamma}{T}} \quad \text{w.p. 1}\end{aligned}$$

where  $H_{p_\theta}(A^* | \mathcal{Z}^{\text{new}}) \leq \log(|\mathcal{A}^{\text{new}}|)$  refers to the conditional Shannon entropy of  $A^*$  given  $\mathcal{Z}^{\text{new}} := (\mathcal{Z}^{(a)})_{a \in \mathcal{A}^{\text{new}}} = z$  under the data generating process defined by  $p_\theta$ , and  $\Gamma$  is a constant upper bound on the “information ratio” such that

$$\Gamma \geq \max_{t \in [1: T]} \left\{ \frac{(\mathbb{E}[\tilde{R}_t^{(A^*)} - \tilde{R}_t])^2}{I_t(A^*; (A_t, \tilde{R}_t))} \right\} \quad \text{w.p. 1},$$

Above we use  $\tilde{R}_t^{(A^*)} \leftarrow R_{N_t^{(A^*)}}^{(A^*)}$  where  $N_t^{(a)} := \sum_{i=1}^t \mathbb{1}(A_i = a)$  denotes the number of times arm  $a$  was played up to and including period  $t$ . Above we use  $\mathbb{E}_t[\cdot] := \mathbb{E}_{p^*, \pi_{\text{PS-AR}}(p_\theta)}[\cdot | \tilde{\mathcal{H}}_{t-1}]$  to denote that expectations are conditioned on the history and  $I_t(A^*; (A_t, \tilde{R}_t))$  to denote the mutual information between  $A^*$  and  $(A_t, \tilde{R}_t)$  conditional evaluated under a base measure  $(p^*, \pi_{\text{PS-AR}}(p_\theta))$  that conditions on  $\tilde{\mathcal{H}}_{t-1}$ . (Recall that the history also includes the information in  $\mathcal{Z}^{\text{new}}$ ).

The proof of Proposition 5 of Russo and Van Roy (2016) shows that one can choose  $\Gamma \leq |\mathcal{A}^{\text{new}}|/2$  w.p. 1. As observed in Bubeck et al. (2015); Bubeck and Eldan (2016), this proof relies only on the probability matching property in Lemma 3 and hence applies in our setting.

Combining our results implies

$$\mathbb{E}_{p^*, \pi_{\text{PS-AR}}(p_\theta)} \left[ \max_{a \in \mathcal{A}^{\text{new}}} \left\{ \frac{1}{T} \sum_{t=1}^T R_t^{(a)} \right\} - \frac{1}{T} \sum_{t=1}^T \tilde{R}_t \middle| \mathcal{Z}^{\text{new}} \right] \leq \sqrt{\frac{\log(|\mathcal{A}^{\text{new}}|) \cdot |\mathcal{A}^{\text{new}}|}{2T}} \quad \text{w.p. 1},$$

so the result follows by the law of iterated expectations.  $\square$

## E EXPERIMENT DETAILS

In this appendix we discuss synthetic experiments in Section E.1, news article recommendation experiments in Section E.4 and bandit algorithms in Section E.5.

### E.1 SYNTHETIC EXPERIMENTS: MIXTURE BETA-BERNOULLI

**Data Generating Process** In this setting, we use article attributes be  $Z^{(a)} = (Z_1^{(a)}, Z_2^{(a)}) \in \mathbb{R}^2$  where  $Z_1^{(a)}, Z_2^{(a)} \stackrel{i.i.d.}{\sim} \text{Uniform}(0, 0.25)$ . We sample  $R_{1:T}^{(a)}$  by first sampling  $\mu_\infty^{(a)} \in [0, 1]$  from a mixture:

$$\mu_\infty^{(a)} | Z^{(a)} \sim \begin{cases} \text{Beta}(25Z_1^{(a)} + 1, 25(1 - Z_1^{(a)}) + 1) & \text{w.p. } 1/2 \\ \text{Beta}(25(1 - Z_2^{(a)}) + 1, 25Z_2^{(a)} + 1) & \text{w.p. } 1/2 \end{cases}$$

Then, outcomes are sampled as  $R_1^{(a)}, \dots, R_T^{(a)} | \mu_\infty^{(a)}, Z^{(a)} \stackrel{i.i.d.}{\sim} \text{Bernoulli}(\mu_\infty^{(a)})$ .

Here,  $\mu_\infty^{(a)}$  corresponds to the success rate in the data generating process, in contrast to  $\mu^{(a)}$  in the main text of the paper, which corresponds to the mean (or success rate) in the finite-sample population of size  $T$ . Note that  $\mu^{(a)}$  converges to  $\mu_\infty^{(a)}$  as  $T$  goes to infinity.

**Training and Validation Datasets** The training and validation datasets contain 2500 and 1000 articles each, respectively. During training (Algorithm 2), we use  $T_{\text{train}} = 500$ . Hyperparameters and early stopping epochs are chosen using the validation dataset.

#### Additional Model and Training details

- **FLEXIBLE NN.** This model implements the autoregressive model as a neural network that takes as input the action/article attribute  $Z^{(a)}$  (a vector in  $\mathbb{R}^2$ ) and summary statistics of observations for this action, and outputs a value (probability) in  $[0, 1]$ . The summary statistic we use is simple because outcomes  $R_t^{(a)}$  are binary; specifically it consists of a tuple with the mean of outcomes from action  $a$ , and the reciprocal of 1 plus the total number of outcome observations for action  $a$ , i.e.  $(\frac{1}{N^{(a)}} \sum_{t'=1}^{t-1} R_{t'} \mathbb{1}_{A_{t'}=a}, \frac{1}{1+N^{(a)}})$ , where  $N^{(a)} := \sum_{t'=1}^{t-1} \mathbb{1}_{A_{t'}=a}$ . (In practice, we found that repeating the summary tuple input 10 improved performance, so the model took as input vectors in  $\mathbb{R}^{22}$  which consisted of a 2-dimensional  $Z^{(a)}$  and 10 copies of the sufficient statistic tuple). Note that this entire  $p_\theta$  could alternatively be implemented as a transformer.

The MLP we use has three linear layers, each of width 50. After the first and second linear layers, we apply a ReLU activation. After the last linear layer, we apply a sigmoid function, so that the output is in  $(0, 1)$ . The models are trained for 1000 epochs with learning rate 0.001, batch size 500, and weight decay 0.01 using the AdamW optimizer.

- **BETA-BERNOULLI NN.** This is a sequential model that is the (closed-form) posterior predictive for a Beta-Bernoulli. The prior parameters for the Beta distribution,  $\alpha_\theta(Z^{(a)})$  and  $\beta_\theta(Z^{(a)})$ , are each parameterized by separate neural network MLP models that take in  $Z^{(a)}$ .

The MLPs we use has three linear layers, each of width 50. After the first and second linear layers, we apply ReLU activations. After the last linear layer, we also apply a ReLU activation, so that the final output is in  $[0, \infty)$ . We initialize weights so that the bias term for both  $\alpha_\theta(Z^{(a)})$  and  $\beta_\theta(Z^{(a)})$  to 1, so that we avoid starting with Beta parameters of value 0, as Beta parameters need to be positive. The models are trained for 1000 epochs with learning rate 0.001, batch size 500, and weight decay 0.01 using the AdamW optimizer.

**Additional Details on Figure 5** In our uncertainty quantification plots Figure 5 (right), we evaluate over all 1000 actions in the validation set. We form 250 samples of  $\hat{\mu}_1^{(a)}$  for each action in the validation set using Algorithm 3 with  $m = 500$ . To generate posterior samples for BETA-BERNOULLI NN, we use the closed-form posterior (i.e.,  $m = \infty$ ).

In our regret plots Figure 5 (left), we run 500 runs. In each run we randomly choose  $|\mathcal{A}^{\text{new}}| = 10$  actions randomly with replacement from the validation set, and all algorithms are evaluated on these

same sampled actions in each run. Regret is calculated relative to  $\mu_\infty^{(a)}$  from the data generating process.

## E.2 ALTERNATIVE SAMPLING METHODS GIVEN A SEQUENCE MODEL

In Figure 4 we displayed regret for the setting in Section 6.1 but where we use different ways to sample  $p_\theta$  that are not from our proposed method. Here is a brief comparison:

- **Generate full reward sequence (PS-AR):** this is our method. In it, for each arm, we use  $p_\theta$  to generate a sequence of missing rewards, take the mean of the rewards for each arm, and choose the arm with the largest mean reward value.
- **Generate one reward:** instead of generating a sequence of missing rewards, only generate one. Choose the arm with the largest reward (breaking ties randomly).
- **Generate many rewards, but not sequentially:** like the previous, except that for each arm, we generate one reward, but repeat this process many times. We do this 500 times for this particular experiment.
- **Greedy:** our implementation of this generative model for sampling sequences of rewards can output the probability that the next reward is 1. This is similar to the previous sampling method.

The alternative sampling methods tend to do worse than our proposed method, especially for longer horizons, as the other sampling methods are not approximating posterior inference for the mean reward for each arm.

## E.3 RECOVERING THE TRUE PRIOR VIA PRETRAINING (EMPIRICAL BAYES)

We discussed connections between our pretraining procedure and empirical Bayes from Section 3.3. Here, we demonstrate in practice an setting where we perform “empirical Bayes” using our pretraining procedure (Algorithm 2). We find that we recover the true prior fairly well.

**Data Generation** We use a synthetic Beta-Binomial data generating process. We consider one-dimensional action features  $Z^{(a)} \stackrel{i.i.d.}{\sim} \text{Uniform}(0, 1)$ . We then sample  $\mu^{(a)}$  from a Beta distribution, where

$$\mu^{(a)} | Z^{(a)} \sim \text{Beta}(Z^{(a)} \cdot 5 + 1, (1 - Z^{(a)}) \cdot 5 + 1). \quad (19)$$

Then,  $R^{(a)} | \mu^{(a)}, Z^{(a)} \sim \text{Bernoulli}(\mu^{(a)})$ . We use a training dataset of size 25,000 actions and a validation set of size 10,000 actions; both datasets have observation sequences of length  $n = 500$ .

**Autoregressive model** We use  $p_\theta$  which matches the posterior predictive of a Beta-Bernoulli model described above. Specifically, we use the following sequence model:

$$p_\theta(R_{t+1}^{(a)} = 1 | R_{1:t}^{(a)}) = \frac{\alpha_\theta(Z^{(a)}) + \sum_{i=1}^t R_i^{(a)}}{\alpha_\theta(Z^{(a)}) + \beta_\theta(Z^{(a)}) + t}. \quad (20)$$

To accomodate  $Z^{(a)}$  features, we parameterize the prior hyperparameters:  $\alpha_\theta(Z^{(a)}), \beta_\theta(Z^{(a)})$  (we follow the procedure described in Appendix E.1 for BETA-BERNOULLI NN). The neural network model architecture used in  $\alpha_\theta(Z^{(a)}), \beta_\theta(Z^{(a)})$  and the training procedure are also the same as described for BETA-BERNOULLI NN in Appendix E.1 (except that the MLP widths are 100).

**Recovering the Prior: Figure 9** We show in Figure 9 that through our pretraining procedure Algorithm 2 with our particular choice of  $p_\theta$  model class, that we (approximately) recover the true prior. We show this by comparing means and standard deviations of samples from our learned prior (using  $p_\theta$ ) vs. the true prior (according to the data generating process), for different draws of  $Z^{(a)}$ . In the scatter plots, each point corresponds to one  $Z^{(a)}$ .

Specifically, in these plots we use 100 actions sampled uniformly from the validation set. For each of these 100 actions we form 10,000 samples of  $\hat{\mu}_1^{(a)}$  using Algorithm 3 using our learned  $p_\theta$  model.

We also form 10,000 samples from the true data generating prior equation 19 for each of the 100 actions. Then for each action, we compute the mean and standard deviations of the samples on the “prior” samples  $\hat{\mu}_1^{(a)}$  from  $p_\theta$ ; we also compute the mean and standard deviations of the samples from the true prior. We then plot these in a scatter plot; for each action, we have the prior mean according to  $p_\theta$  vs the prior mean according to the data generating process—this forms one point on the scatter plot. A similar procedure is plotted on the right. There, instead of computing the mean of the prior samples, we compute a measure of the spread of the prior samples: let  $\hat{\mu}_{1,1}^{(a)}, \hat{\mu}_{1,2}^{(a)}, \dots, \hat{\mu}_{1,10000}^{(a)}$  be the prior samples. Let  $\bar{\mu}_1^{(a)} = \frac{1}{10000} \sum_{i=1}^{10000} \hat{\mu}_{1,i}^{(a)}$ . Then we compute the mean absolute deviation  $\frac{1}{10000} \sum_{i=1}^{10000} |\hat{\mu}_{1,i}^{(a)} - \bar{\mu}_1^{(a)}|$  for this set of prior samples.

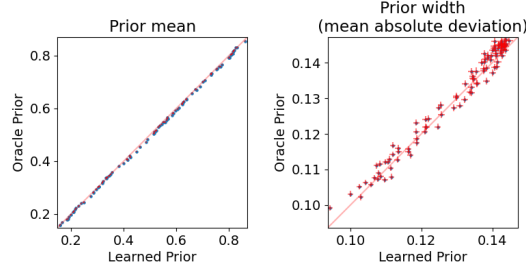


Figure 9: **Comparing oracle prior vs prior learned through our method (empirical bayes) in a synthetic setting.** Error bars represent  $\pm 1$  standard error; the error bars on the left plot are present but small enough to not be visible.

#### E.4 NEWS RECOMMENDATION EXPERIMENT DETAILS

**Additional data details** The training and validation datasets contain 9122 and 2280 distinct actions/articles each, respectively. During training, we use  $T_{\text{train}} = 500$  As in Appendix E.1 hyperparameters and early stopping epochs are chosen using the validation dataset.

We now discuss the news data preprocessing process. This dataset is free to download for research purposes at <https://msnews.github.io/>. It is under a Microsoft Research License at [https://github.com/msnews/MIND/blob/master/MSR%20License\\_Data.pdf](https://github.com/msnews/MIND/blob/master/MSR%20License_Data.pdf), which we comply with. The terms of use are at <https://www.microsoft.com/en-us/legal/terms-of-use>.

Our preprocessing procedure is as follows:

1. Collect all articles from the MIND “large” dataset (training split only) (Wu et al., 2020).
2. Remove any article with fewer than 100 total impressions.
3. Normalize the success probabilities to be centered around 0.5 in a way that preserves the ranking of  $\mu^{(a)}$ . We do this transformation to speed up the learning procedure (since it requires more data to learn small true Bernoulli success probabilities accurately). We leave simulations without this transformation to future work.

Our transformation procedures as follows: Let  $\mu_0^{(1)}, \dots, \mu_0^{(|\mathcal{A}|)}$  be the original empirical success probabilities (average click rate). We use  $\mathcal{A}$  to denote all articles in the MIND large dataset. The new success probabilities are defined as follows for each  $a \in \mathcal{A}$ :

$$\mu_\infty^{(a)} \leftarrow \begin{cases} \mu_0^{(a)} & \text{if } \mu_0^{(a)} \in \{0, 1\} \\ \text{logit}^{-1} \left( \text{logit}(\mu_0^{(a)}) - \bar{\mu}_0 \right) & \text{otherwise} \end{cases}.$$

Above,  $\bar{\mu}_0 \triangleq \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} \text{logit}(\mu_0^{(a')})$  and  $\text{logit}(x) \triangleq \log \frac{x}{1-x}$ . See Figure 10 for comparison of the success probabilities (click rates) before and after the transformation.

4. Randomly select 20% of the remaining articles to be in the validation set; the rest are in the training set.



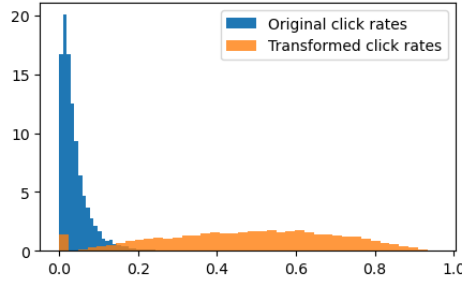


Figure 10: Original and transformed click rates. Note the spike at 0 for transformed click rates: only click rates that were not 0 or 1 are transformed.

### Additional model details

- **FLEXIBLE NN (TEXT).** This model very similar to the FLEXIBLE NN model in Appendix E.1 with the exception that in place of a two-dimensional  $Z^{(a)}$ , the MLP head of the neural network from before is fed as input a DistilBERT (Sanh et al., 2019) embedding of text data  $Z^{(a)}$ . Also, the MLP linear layers have width 100 instead of 50, and the sufficient statistics are repeated 100 times instead of 10 times. All other architecture details are the same. The model is trained for 500 epochs with learning rate  $1e-5$  on MLP heads,  $1e-8$  on the DistilBERT weights, batch size 500, and weight decay 0.01 using the AdamW optimizer.
- **BETA-BERNOULLI NN (TEXT).** This is very similar to the Beta-Bernoulli posterior predictive sequence model in Appendix E.1, with the exception that in place of a two-dimensional  $Z^{(a)}$ , the MLP head of the neural network from before is fed as input a DistilBERT (Sanh et al., 2019) embedding of text data  $Z^{(a)}$ . On top of the one DistilBERT embedding are two separate MLP heads for  $\alpha(Z^{(a)})$  and  $\beta(Z^{(a)})$ , which are trained together. Also, the MLP linear layers have width 100 instead of 50, and the sufficient statistics are repeated 100 times instead of 10 times. All other architecture details are the same. The model is trained for 500 epochs with learning rate  $1e-5$  on MLP heads,  $1e-8$  on the DistilBERT weights, batch size 500, and weight decay 0.01 using the AdamW optimizer.
- **FLEXIBLE NN (CATEGORY).** This is very similar to the flexible neural network model in Appendix E.1, but it uses a one-hot new category vector for  $Z^{(a)}$  instead of a two-dimensional  $Z^{(a)}$ . The model architecture and training parameters are also the same.
- **DistilBERT.** Our two text models use DistilBERT (Sanh et al., 2019) from <https://huggingface.co/distilbert/distilbert-base-uncased>. It has an apache-2.0 license, with license and terms of use at <https://huggingface.co/datasets/choosealicense/licenses/blob/main/markdown/apache-2.0.md>

**Additional Details on Figure 6** In our uncertainty quantification plots Figure 6 (right), we evaluate over all 2280 articles/actions in the validation set. For our FLEXIBLE NN  $p_\theta$  model, we form 250 samples of  $\hat{\mu}_1^{(a)}$  for each action in the validation set using Algorithm 3 with  $m = 500$ . For our BETA-BERNOULLI NN  $p_\theta$  model we use samples from the closed-form posterior.

In our regret plots Figure 6 (left), we run 500 runs. In each run we randomly choose  $|\mathcal{A}^{\text{new}}| = 10$  actions randomly with replacement from the validation set, and all algorithms are evaluated on these same sampled actions in each run. Regret is calculated relative to  $\mu_\infty^{(a)}$  as described above.

**Ensemble** We describe the ensembling approach used in the uncertainty quantification plots in Figure 6 (right). To construct ensembles, we first train a DistilBERT model with an MLP head (MLP width 100, 3 layers, batch size 100, 500 epochs, learning rate  $1e-5$  on the head and  $1e-8$  on DistilBERT, weight decay 0.01, AdamW optimizer) to predict  $R_t^{(a)}$ , using action/article features  $Z^{(a)}$  (headlines). Then, we freeze the DistilBERT weights, and train 50 MLP heads from scratch with random initialization and bootstrapped training data to create the ensemble (50 epochs, fixed DistilBERT embedding; other params the same as before). We include a “randomized prior” variant



that initializes each neural network model in the ensemble in a particular way to encourage diversity [Osband et al. \(2018\)](#).

## E.5 BANDIT ALGORITHMS

We compare our method with several baseline bandit methods.

**PS Beta Bernoulli (Uniform Prior)** We model success rate  $\mu_\infty^{(a)}$  and potential outcomes  $R_t^{(a)}$  using a conjugate Beta-Bernoulli model:

$$\mu_\infty^{(a)} \sim \text{Beta}(\alpha, \beta) \quad (21)$$

$$R_1^{(a)}, \dots, R_T^{(a)} \mid \mu_\infty^{(a)} \stackrel{i.i.d.}{\sim} \text{Bernoulli}(\mu^{(a)}) \quad (22)$$

In our experiments, we use Beta-Bernoulli with a uniform prior, so  $\alpha = \beta = 1$ . Note that unlike the BETA-BERNOULLI NN, the prior here does not depend on action attributes  $Z^{(a)}$ .

Online decision-making uses Thompson sampling, as described in in Algorithm 6.

---

### Algorithm 6 Beta-Bernoulli Posterior (Thompson) Sampling

---

- 1: **Inputs:** Prior hyperparameters  $\alpha, \beta$ .
  - 2: Set priors  $(\alpha_0^{(a)}, \beta_0^{(a)}) \leftarrow (\alpha, \beta), \forall a \in \mathcal{A}^{\text{new}}$
  - 3: **for**  $t = 1, \dots, T$  **do**
  - 4:   **for**  $a \in \mathcal{A}^{\text{new}}$  **do**
  - 5:     Sample  $\hat{\mu}^{(a)} \sim \text{Beta}(\alpha_{t-1}^{(a)}, \beta_{t-1}^{(a)})$
  - 6:   **end for**
  - 7:   Select action  $A_t \leftarrow \arg \max_{a \in \mathcal{A}^{\text{new}}} \{\hat{\mu}^{(a)}\}$
  - 8:   Observe outcome  $R_t$  from action  $A_t$ .
  - 9:   Update posterior  $(\alpha_t^{(A_t)}, \beta_t^{(A_t)}) \leftarrow \begin{cases} (\alpha_{t-1}^{(a)} + \mathbb{1}_{R_t=1}, \beta_{t-1}^{(a)} + \mathbb{1}_{R_t=0}) & \text{if } A_t = a \\ (\alpha_{t-1}^{(a)}, \beta_{t-1}^{(a)}) & \text{otherwise} \end{cases}$
  - 10: **end for**
- 

**PS Neural Linear** We implement a very simple variant of “neural linear” as in [Riquelme et al. \(2018\)](#); [Snoek et al. \(2015\)](#). Here, we model each arm reward as a Gaussian-Gaussian model. We fit the prior mean using item features  $Z^{(a)}$ , but set a shared prior variance across articles. Specifically,

$$\mu^{(a)} \sim N(g(Z^{(a)}), \sigma^2)$$

$$R_1^{(a)}, \dots, R_T^{(a)} \stackrel{i.i.d.}{\sim} N(\mu^{(a)}, s^2)$$

First we address the choice of  $g, \sigma^2, s^2$ , which are chosen during pre-training, and then the bandit evaluation, which is standard Thompson sampling with a Gaussian-Gaussian model. We address these one at a time.

### Parameters

1. First,  $g$  is obtained by training a model to predict  $\mu^{(a)}$  from just  $Z^{(a)}$  (no history of past rewards), using the training set. For synthetic experiments, for  $g$ , we used a neural network with almost the same architecture as for the autoregressive model we use for this dataset. However, the model only takes  $Z^{(a)}$  (and not previous rewards for  $a$ ). We use all of the same hyperparameters as we did to train the autoregressive model for this dataset. For news datasets, we trained a DistilBERT model with a MLP on top that takes the embedded article headlines  $Z^{(a)}$  as input (and not previous rewards for  $a$ ). We use the same hyperparameters as we did to train the autoregressive model for this dataset, except for learning rates, which were chosen to be the best out of several (for synthetic, 1e-2; for news setting, 1e-5 for both the MLP head and the DistilBERT weights).
2.  $\sigma^2, s^2$  were chosen to be reasonable values, which in our experiments were 0.25 for  $s^2$  (which corresponds to maximum variance of a Bernoulli), and 1 for  $\sigma^2$ .

**Bandit evaluation** To obtain the posterior of  $\mu^{(a)} \mid (R_i^{(a)})_{i \notin M^{(a)}}$  where  $(R_i^{(a)})_{i \notin M^{(a)}}$  denotes observed rewards for arm  $a$ , we use a standard Gaussian-Gaussian posterior with known variances:

$$P\left(\mu^{(a)} \in \cdot \mid (R_i^{(a)})_{i \notin M^{(a)}}\right) \sim N(\tilde{\mu}, \tilde{\sigma}^2)$$

where

$$\tilde{\mu} = \left( \frac{g(Z^{(a)})}{\sigma^2} + \frac{\text{sum}\left((R_i^{(a)})_{i \notin M^{(a)}}\right)}{s^2} \right) \left( \frac{1}{\sigma^2} + \frac{\#\{i \notin M^{(a)}\}}{s^2} \right)^{-1} s^2$$

and

$$\tilde{\sigma}^2 = \left( \frac{1}{\sigma^2} + \frac{\#\{i \notin M^{(a)}\}}{s^2} \right)^{-1}.$$

Arms are chosen via Thompson sampling (Russo et al., 2020).

**UCB** For UCB we use the multi-arm bandit algorithm described in Section 6 of (Abbasi-Yadkori et al., 2011). We set the failure probability  $\delta = 0.1$  and use sub-Gaussian parameter 0.5 (since we have binary rewards).

**SquareCB** In these experiments, we use the flexible neural network  $p_\theta$  with text attributes, but instead of using Thompson sampling, we use SquareCB (Foster and Rakhlin, 2020), which is a bandit algorithm that uses a regression oracle to predict the value of each action. Note that our setting differs from the setting of SquareCB (Foster and Rakhlin, 2020), as SquareCB assumes that the prediction model for action value is being learned online, while our prediction model has been pretrained on historical data and is not learned online.

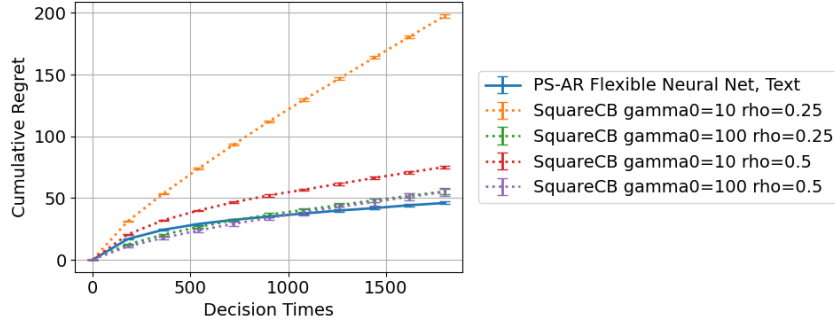


Figure 11: Regret comparison on news dataset for SquareCB and posterior sampling, both using the flexible neural network sequence model using text attributes in 6.2

For setting the learning rate  $\gamma$  in SquareCB (Foster and Rakhlin, 2020), we follow Foster et al. (2020) and consider a time-varying learning rate  $\gamma_t = \gamma_0 t^\rho$ , where  $\gamma_0 \in \{10, 100\}$  (a subset of those suggested in Foster et al. (2020)), and  $\rho \in \{0.25, 0.5\}$  are hyperparameters. While some hyperparameter combinations for SquareCB perform almost as well PS-AR FLEXIBLE NN (TEXT), we remark that there is no principled approach to choosing the learning rate provided in existing works (besides grid search by deploying the algorithm many times).

## E.6 COMPARISON TO DECISION PRE-TRAINED TRANSFORMERS (DPT)

DPT (Lee et al., 2023b) trains a sequence model to predict the best action (mimicking an expert/optimal policy), given the current state and recent interactions within an environment (the “in-context dataset”). This sequence model is trained across states and recent interactions and tasks.

**Relationship between DPT and PS-AR** Both methods are ways to use generative sequence models for decision making. Both also have theoretical results showing that they are equivalent to Thompson (posterior) sampling, under certain sets of assumptions. However, there are a few key differences between DPT and PS-AR:

- PS-AR predicts sequences of future rewards, while DPT predicts next (optimal) action.
- At decision-time, PS-AR needs to generate a sequence of rewards, which could be relatively costly, while DPT only needs to generate one action.
- DPT requires an optimal or expert policy (can be approximate) to mimic in pretraining, while PS-AR does not explicitly require such an expert policy.
- DPT’s training objective is only to identify the best action, and not to model the distribution of rewards for each action. DPT sequence models are effectively trained to predict rank information, while PS-AR sequence models are trained to predict future rewards. It’s possible that training to predict rank information gives the model less signal to learn from, which might make training less efficient.

Additionally, one potentially interesting research direction is to investigate whether it may be fruitful to combine DPT and PS-AR: for example, PS-AR can be used for generating in-context datasets for DPT to train on. As another example, in our experiment below, we show that in our bandit setting, DPT and PS-AR can use almost identical model architectures, and it is possible that pre-training on one task can result in useful features for the other task.

**Experiment details for implementing DPT.** In order to make the most fair comparisons between DPT and PS-AR, we design the model architecture and training for our implementation of DPT to be as similar as possible to those used for PS-AR, while retaining the core aspects of DPT (e.g. predicting and learning to predict the optimal next action). We also directly import how DPT samples in-context datasets for pre-training from the paper and associated code.

We use a sequence model architecture that is almost identical to the corresponding sequence models used in the news recommendation experiment setting. While our proposed model outputs a probability for the next reward being a 1, given action features and previous rewards (“in-context dataset”) for a given action, for DPT we use the same model, minus the sigmoid at the end, to produce a real number. In the DPT sequence model, each arm has a copy of its own sequence model, and a softmax is applied at the end to form a probability distribution over the set of candidate actions.

For training, we bootstrap data in the following way, that mirrors the procedure used in DPT. At every iteration, we load a random batch of arms and their corresponding sequences of rewards. This batch (size 5000 for the synthetic setting) is split into multiple bandit environments, each with 10 arms. Then, we use DPT’s method for generating in-context datasets for their bandit setting, where actions are sampled according to a mixture of a Dirichlet and a point mass (Section 4 of (Lee et al., 2023b)). These actions are sampled for a random number of history steps that is drawn uniformly from  $[0, 499]$ . This in-context dataset is then used as input to the sequence model, which will output probabilities across arms. The desired output is the true best arm, and the loss is cross-entropy loss. This procedure uses the same data loader as PS-AR, with modifications to match the DPT objective. Model selection is done using loss on the validation set.

We train these for 2500 epochs. The model hyperparameters are the same as for PS-AR, with the exception of learning rate, which is  $1e-3$  for synthetic (the best out of  $1e-1, 1e-2, 1e-3, 1e-4$ ).

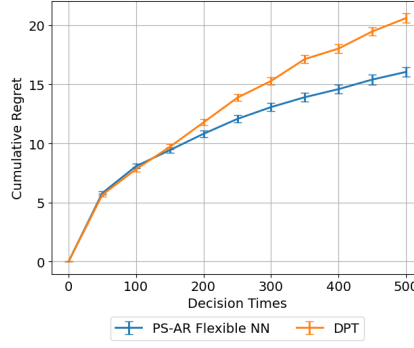


Figure 12: Regret for DPT vs PS-AR on the synthetic setting (Section 6.1)

Figure 13: Preliminary bandit regret comparisons between DPT (Lee et al., 2023b) and PS-AR on the synthetic setting (Section 6.1)

**Experiment results** DPT performs fairly similarly to our proposed method PS-AR. Both methods have been theoretically shown to approximate Thompson sampling with a learned prior under respective assumptions. Note that DPT does not provide regret bound results if their sequence model is misspecified.

## F WHEN IS AN AUTOREGRESSIVE SEQUENCE MODEL A VALID POSTERIOR PREDICTIVE?

In Algorithm 2, we learn an autoregressive model to use in place of a posterior predictive in Algorithm 1. We make this connection in Section 4 and establish a regret bound for Algorithm 1 that holds whenever  $p_\theta$  has low loss.

In this section, we address the following question: *When is  $p_\theta$  a valid posterior predictive, for some underlying Bayesian model?*

In order for an autoregressive generative sequence model to be a valid posterior predictive distribution, the sequence model to be *infinitely exchangeable*. We say that a sequence model is an infinitely exchangeable sequence model if it generates infinitely exchangeable random variables (Definition 2).

**Definition 2** (Exchangeability). *A sequence of random variables  $Y_1, Y_2, \dots, Y_n$  is exchangeable if for any permutation  $\pi$ , the following are equal in distribution:*

$$(Y_1, Y_2, \dots, Y_n) \stackrel{D}{=} (Y_{\pi(1)}, Y_{\pi(2)}, \dots, Y_{\pi(n)}).$$

*An infinite sequence of random variables is infinitely exchangeable if any finite subset is exchangeable.*

Practically, this means that the models we train need to be invariant to the *order* in which previous outcomes are fed into the model. The key insight behind why infinitely exchangeable sequence models are valid posterior predictives is De Finetti’s Representation Theorem (Theorem 2 below). We state this Theorem for binary outcomes for simplicity (De Finetti, 1929; Heath and Sudderth, 1976), but it generalizes to real-valued outcomes (De Finetti, 1937).

**Theorem 2** (De Finetti’s Representation Theorem for Binary Outcomes). *If a sequence of binary random variables  $\{Y_i\}_{i \in \mathbb{N}}$  is infinitely exchangeable, then there exists a unique distribution  $P(\mu)$  on  $[0, 1]$  such that for some  $\mu \sim P(\mu \in \cdot)$ ,*

$$Y_1, Y_2, Y_3, \dots \mid \mu \stackrel{i.i.d.}{\sim} \text{Bernoulli}(\mu).$$

The implication of Theorem 2 is that *any* infinitely exchangeable sequence of binary random variables  $\{Y_i\}_{i \in \mathbb{N}}$  can equivalently be described as being generated by a particular Bayesian model with a Bernoulli likelihood. Above,  $\mu$  is a latent success probability that is drawn from some prior distribution  $P(\mu \in \cdot)$ .