Why Is Prompting Hard? Understanding Prompts on Binary Sequence Predictors

Li K. Wenliang[*†] **Anian Ruoss**[*] **Jordi Grau-Moya**[*] **Marcus Hutter**[*] **Tim Genewein**[*†]
[*]Google DeepMind [†]Correspondence to {kevinliw,timgen}@google.com

Abstract

Frontier models can be prompted or conditioned to do many tasks, but finding good prompts is not always easy, nor is understanding some performant prompts. We view prompting as finding the best conditioning sequence on a near-optimal sequence predictor. On numerous well-controlled experiments, we show that unintuitive optimal conditioning sequences can be better understood given the pretraining distribution, which is not usually available. Even using exhaustive search, reliably identifying optimal prompts for practical neural predictors can be surprisingly difficult. Popular prompting methods, such as using demonstrations from the targeted task, can be surprisingly suboptimal. Using the same empirical framework, we analyze optimal prompts on frontier models, revealing patterns similar to the binary examples and previous findings. Taken together, this work takes an initial step towards understanding optimal prompts, from a statistical and empirical perspective that complements research on frontier models.

1 INTRODUCTION

Successful frontier models are pretrained over large datasets, which are generated by authors and creators of diverse knowledge domains, individual styles and embedded sentiments. The distribution over these high-level *latent factors*, together with the data distribution of each source, implicitly defines a hierarchical generative process, or meta-distribution. As a notable example, text data can be regarded as a meta-distribution of tokens used to pretrain large language models (LLMs). Theoretically, minimizing the next-token prediction error on the meta-distribution yields a Bayes-optimal predictor for the meta-distribution, as clearly shown by Ortega et al. (2019) using binary sequences. One hallmark feature of such a predictor is its capability to be steered through human-readable prompts to execute various tasks. In the case of LLMs, prompting (Brown et al., 2020) triggers implicit *inference* over the latent factors desired for a specific behavior (Xie et al., 2021; Xinyi Wang et al., 2023; Jiang, 2023; Wies et al., 2023; Arora et al., 2024). Nonetheless, heuristic and handcrafted prompts often fall short of expectations, while certain demonstrably powerful prompts appear eccentric and warrant deeper fundamental understandings. Previous work on (binary) sequence predictors studied mostly unconditioned predictive performance (e.g., Ortega et al., 2019; Mikulik et al., 2020; Genewein et al., 2023; Bhattamishra et al., 2020; C. Wei, Y. Chen, et al., 2022; Deletang et al., 2022); however, how sequence predictors behave under controlled conditioning is not well-studied, nor do we know what the optimal conditioning sequences are for a given downstream task.

While many factors, such as Transformer architecture and post-training, are at play, we provide robust and fundamental insights on prompting from a Bayesian meta-learning perspective. Specifically, we study prompting on binary sequence predictors with a carefully designed framework, complementing frontier model research. Using a large collection of idealized and neural predictors (>1000 instances, see Section B.1), we observe unexpected characteristics in optimal prompts, even in simplified scenarios. For example, consider a binary generator pretrained on sequences of i.i.d. coin flips, with $\mathbb{P}(\text{HEADS})$ randomly chosen for each sequence. How would one prompt it to generate 70% heads? A natural heuristic prompt is a sequence of 70% heads; however, as we will show, this is not always the most effective prompt: the optimal prompt depends on the pretraining distribution, which is often unknown and overlooked in the past. Employing this framework, we address further fundamental questions: Are longer

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

prompts better? Does more data help identify optimal prompts? Can expert behavior be induced by expert demonstrations? Not always, as we elaborate in the upcoming sections. Our main contributions are:

1. We establish a rigorous *framework* for analyzing optimal conditioning sequences (prompts), encompassing notions of optimality, experimental designs, and prompt visualizations.
2. Applying this framework to examples of binary data generators (DGs), we collected numerically-guaranteed optimal prompts by exhaustive search (>250k in total). Optimal prompts on seemingly trivial examples can be *atypical* for the task, partly due to biases in the usually unknown pretraining DG; interpreting the prompts can thus be hard.
3. Given a finite task dataset, we show that optimal prompts can be *unreliable* to find, leading to inconsistent results for every draw of the dataset. The reliability follows *unintuitive trends* depending on the task setup, e.g., increasing the task dataset size does not always improve reliability.
4. Using an in-context learning setup involving two-arm bandits, we show that prompting by expert demonstrations is *less effective* compared to optimal prompts; knowing the pretraining distribution is crucial in comprehending performant prompts.
5. We adapt this framework for studying prompts on real LLMs, and reveal interesting patterns consistent with findings on the binary examples.

Limitations and scope. We remark upfront that, although our binary data tasks make our results clear and interpretable, they are drastic simplifications of natural language data. Consequently, while our findings offer foundational insights, most of our results (excluding our LLM investigations) are not tied to specific frontier models, preventing definitive claims about their exact behaviors at scale. To ensure optimality, our exhaustive search fully explores the prompt space, and comparisons with practical partial-search methods are out of scope. This paper takes a Bayesian meta-learning view of sequence predictors, and we do not fully study the effects of post-training or neural architecture; as a result, a comprehensive theory of prompting in real LLMs is beyond the current aim. Extended limitations are discussed in Section F.3.

2 RELATED WORK

A large body of work aims to find effective prompts. Prompt engineering (e.g., J. Wei, Xuezhi Wang, et al., 2022; P. Liu et al., 2023; B. Chen et al., 2023; Marvin et al., 2023; Sahoo et al., 2024; Y.-F. Song et al., 2024; Khattab et al., 2023; L. Xu et al., 2024) improves performance using intuitions from system design, search and planning. Prompt optimization (e.g., Pryzant et al., 2023; Xinyuan Wang et al., 2023; Fernando et al., 2023; Guo et al., 2023; Zhaoxuan Wu et al., 2024; Hao et al., 2024) instead employs variants of discrete optimization and search techniques. Another use of prompting, particularly in agentic or robotic tasks, is by providing expert demonstrations to induce desired policies for the task (e.g. Dong et al., 2022; Ruoss et al., 2024; Agarwal et al., 2024; Laskin et al., 2023). While much focus has been put on improving the performance of the resulting prompts, we do not know whether they are actually optimal, since the prompt space on text is enormous. It has also been challenging to *interpret* the prompts by relating to the desired tasks, and to understand concretely why they work (Webson et al., 2022; Daras et al., 2022; Murr et al., 2023; He et al., 2024), why some prompts work better than others despite no obvious difference to humans (Kojima et al., 2022; Mizrahi et al., 2024), and why some prompts that are intuitively expected to work may disappoint (Zamfirescu-Pereira et al., 2023; Khurana et al., 2024). These difficulties are exacerbated by the variations across architectures, pretraining mixtures, finetuning processes, and nuanced parameters that vary across public frontier models (L. Chen et al., 2023). The unintuitive nature of prompting has caused safety and ethical concerns (A. Wei et al., 2024; Y. Wu et al., 2023; Z. Xu et al., 2024; Cherepanova et al., 2024; X. Liu et al., 2024; Zou et al., 2023). It is thus worth taking a step back to understand this issue more fundamentally.

3 BACKGROUND

To study prompting under well-controlled conditions, we design synthetic data generators (DGs) with latent factors (akin to Xie et al. (2021) and Jiang (2023)) to define pretraining distributions and (downstream) tasks. These DGs, from Bernoulli sequences in Sections 4 and 5 to bandit problems in Section 6.1, induce a (meta-)distribution over binary token sequences. We then adapt this framework to study real LLMs in Section 6.2 where the latent factors are given but the pretraining distribution over texts is unknown.

To draw a sequence, the DG first draws a random latent value τ from a distribution $p_\tau(\tau)$; such as sampling a bias of a coin from a distribution on $[0,\ 1]$. Then, sample a sequence of length $T \in \mathbb{N}^+$ from a conditional distribution $p_{x|\tau}(x_{1:T}|\tau)$, such as a sequence of flips from a coin with bias τ. The distribution induced by this DG is then the marginal (mixture) $p_x(x_{1:T}) = \int p_{x|\tau}(x_{1:T}|\tau)\mathrm{d}p_\tau(\tau)$. The sequence $x_{1:T}$ is composed of binary tokens from the alphabet $\mathcal{A} := \{0, 1\}$. Subscripts are omitted when no confusion

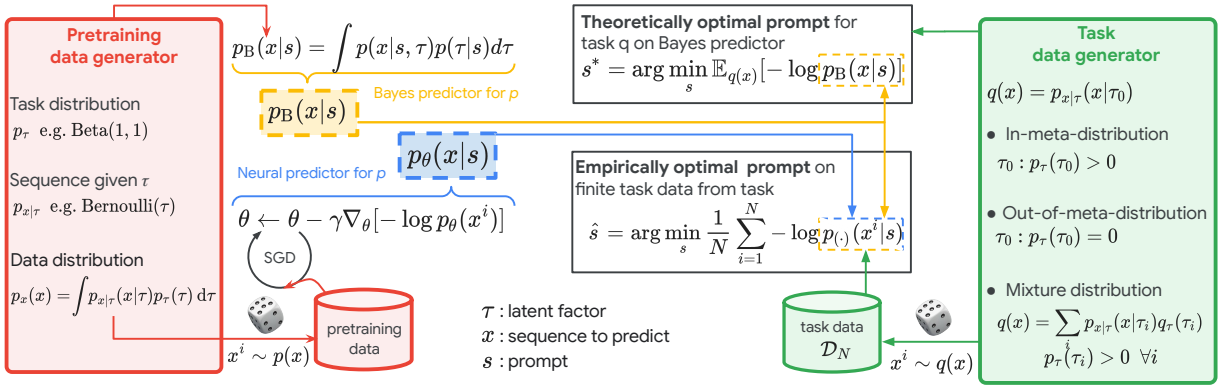

Figure 1: Framework for obtaining optimal prompts under pretraining and task data generators.

should arise. We consider (piecewise) conditionally independent DGs where the conditional $p_{x|\tau}$ satisfies the condition that, for some lengths $T, L \in \mathbb{N}^+$, we have $p_{x|\tau}(s_{1:L}x_{1:T}|\tau) = p_{x|\tau}(s_{1:L}|\tau)p_{x|\tau}(x_{1:T}|\tau)$ for all sequences $s_{1:L} \in \mathcal{A}^L$ and $x_{1:T} \in \mathcal{A}^T$. In this case, when only $s_{1:L}$ is given to predict an unknown $x_{1:T}$, the Bayes-optimal predictor (or Bayes predictor) under the sequence distribution p_x is

$$p_{\mathrm{B}}(x_{1:T}|s_{1:L}) := \int p_{x|\tau}(x_{1:T}|\tau)\mathrm{d}p_{\tau|x}(\tau|s_{1:L}), \quad (1)$$

where $p_{\tau|x}(\tau|s_{1:L}) = p_\tau(\tau)p_{x|\tau}(s_{1:L}|\tau)/p_x(s_{1:L})$ is the posterior over the latent factors given $s_{1:T}$. We regard $s_{1:T}$ as the *prompt* for the sequence(-to-predict) $x_{1:T}$. Conditional independence helps interpret s, and is a common assumption for synthetic datasets (Xie et al., 2021; Xinyi Wang et al., 2023; Jiang, 2023).

3.1 Conditionally Independent Bernoulli DGs

Our first set of experiments in Section 4 uses DGs from the family of conditionally independent Bernoulli DGs (CIB-DGs), where p_τ has support on the unit interval, and $p_{x|\tau}(x_{1:T}|\tau) = \prod_t \mathrm{Bernoulli}(x_t;\tau)$. The Bayes predictor of a CIB-DGs has the property that two prompts $s_{1:L}$ and $s'_{1:L}$ are equivalent in the sense that $p_{\mathrm{B}}(\cdot|s_{1:L}) \equiv p_{\mathrm{B}}(\cdot|s'_{1:L})$ if they have the same counts of zeros and ones, defined as

$$S_0(s_{1:L}) := \sum_{t=1}^L \mathbb{1}[s_t = 0], \ \ S_1(s_{1:L}) := \sum_{t=1}^L \mathbb{1}[s_t = 1]. \quad (2)$$

This permutation invariance reduces the cost of prompt search on the Bayes predictor, but is unlikely to hold for neural networks with standard architectures (Mikulik et al., 2020). We define the particular CIB-DGs and their Bayes predictors used in our first set of experiments:

Definition 3.1. Bern(τ) is the CIB-DG with $p_\tau = \delta_\tau$, where δ_τ is Dirac delta. The Bayes predictor evaluated on $x_{1:T}$ is trivially Bern(τ) itself, that is,

$$p_{\mathrm{B}}(x_{1:T}|s_{1:L}) = p_x(x_{1:T}) = \tau^{S_1(x_{1:T})}(1-\tau)^{S_0(x_{1:T})}$$

Definition 3.2. BernMix(w, τ_1, τ_2) $p_\tau = (1-w)\delta_{\tau_1} + w\delta_{\tau_2}$ for $0 < \tau_1 < \tau_2 < 1$. In our experiments we use equal mixture weights, and write: BernMix(τ_1, τ_2) := BernMix($0.5, \tau_1, \tau_2$). Its Bayes predictor is $p_{\mathrm{B}}(\cdot|s_{1:L}) = \mathrm{BernMix}(w_L(s_{1:L}), \tau_1, \tau_2)$, where

$$w_L(s_{1:L})^{-1} = 1 + \left(\frac{\tau_1}{\tau_2}\right)^{S_1(s_{1:L})}\left(\frac{1-\tau_1}{1-\tau_2}\right)^{S_0(s_{1:L})}. \quad (3)$$

Definition 3.3. BetaBern(α, β) is the CIB-DG with $p_\tau = \mathrm{Beta}(\alpha, \beta)$. The Bayes predictor is $p_{\mathrm{B}}(\cdot|s_{1:L}) = \mathrm{BetaBern}(\alpha + S_1(s_{1:L}), \beta + S_0(s_{1:L}))$.

Despite the apparent simplicity, these DGs can be adapted to generate simple text data, such as synthetic movie reviews: let τ be the sentiment over movies, with 0 being most negative and 1 most positive, and take a categorical $p_{x|\tau}$ over adjectives conditioned on the sentiment; a template for reviews can be {"I have never seen such a _ story", "The visuals are _.", "How _ is the cast!", ...}, which can be concatenated to form a full review. Predicting the next adjective then resembles binary sequence prediction. Using binary tokens makes the prompts more interpretable and visualizable than using natural language tokens, and allows us to better appreciate the challenges of prompting.

3.2 Neural Predictors and Meta-Learning

In our experiments, we train autoregressive neural predictors on samples from meta-distributions. Denote such a predictor by $p_\theta(\cdot)$ where θ are parameters. The objective for a single sequence $x_{1:T}$ is $-\log p_\theta(x_{1:T})$. The latent τ is resampled for each $x_{1:T}$. Under realizability and convergence (Ortega et al., 2019), neural predictors can *meta-learn* to predict sequences by adapting to different latent factors, giving predictions that are indistinguishable from the Bayes predictor over $x \sim p_x$ (Mikulik et al., 2020; Genewein et al., 2023; Li K. Wenliang et al., 2018; Jingfeng Wu et al., 2023; Grau-Moya et al., 2024). However, this theory

does not predict or describe the optimal prompts from (near-)optimal predictors when x is drawn from a task distribution q (see Section A.1). This motivates our empirical framework below. We note upfront that, the results derived from simplifications of CIB-DGs cannot exhaustively explain or accurately predict the behaviors of full-scale LLMs.

4 INVESTIGATION FRAMEWORK

Our framework is illustrated in Figure 1, which is first applied to CIB-DGs in Section 5 before more complex DGs. For each pretraining DG p, we obtain the ideal Bayes predictor p_{B} (by Equation (1)) and practical neural predictors p_θ of recurrent and Transformer-based architectures; see Section A.2. We then prompt a predictor towards a task specified by a task DG q. The task is directly manipulated by the DG, distinct from the activation or features from a network (e.g. Mittal et al., 2024; Hendel et al., 2023; Todd et al., 2024), which is discovered rather than directly controlled.

Theoretically optimal prompt s^*. Given the Bayes predictor p_{B} of a pretraining DG p, the quality of a prompt $s_{1:L}$ towards a task DG q is

$$\mathcal{L}(p_{\mathrm{B}}, q, s_{1:L}) := -\sum_{x\in\mathcal{A}^T} q(x)\log p_{\mathrm{B}}(x|s_{1:L})\,. \quad (4)$$

The *theoretically optimal* prompts of length L and up to length $L_{\max}$ are, respectively,

$$\begin{aligned} s^*_{1:L}(p_{\mathrm{B}}, q) &:= \operatorname*{arg\,min}_{s_{1:L}} \mathcal{L}(p_{\mathrm{B}}, q, s_{1:L});\\ s^*_{L_{\max}}(p_{\mathrm{B}}, q) &:= \operatorname*{arg\,min}_{\ell\in\{1,\dots,L_{\max}\}, s_{1:\ell}} \mathcal{L}(p_{\mathrm{B}}, q, s_{1:\ell}). \end{aligned} \quad (5)$$

Both $s^*_{1:L}$ and $s^*_{L_{\max}}$ depend on the full distributions of the pretraining and task DGs. In this work, we find them by evaluating Equation (4) for each possible prompt (exhaustive search) under the length constraints; see Section A.3. Previous work (Bhargava et al., 2023; Renze et al., 2024; Kusano et al., 2024; Z. Wang et al., 2025; Lester et al., 2021) reported better performance of shorter prompts on LLMs (but see (Q. Liu et al., 2025; Leidinger et al., 2023)) without exhaustive search; here, we investigate this by explicitly controlling the prompt length.

Empirically optimal prompt $\hat{s}$. Given a dataset $\mathcal{D}_N := \{x^i_{1:T}\}_{i=1}^N$ of N sequences from q, we optimize the prompt for a predictor $p_{(\cdot)} \in \{p_{\mathrm{B}}, p_\theta\}$ under the empirical version of the loss (4):

$$\hat{\mathcal{L}}(p_{(\cdot)}, \mathcal{D}_N, s_{1:L}) := -\tfrac{1}{N}\sum_{i=1}^N \log p_{(\cdot)}(x^i_{1:T}|s_{1:L}). \quad (6)$$

The resulting *empirically* optimal prompts are denoted by $\hat{s}_{1:L}$ and $\hat{s}_{L_{\max}}$ under the respective length constraints before. Each exhaustive search uses a *fixed* $\mathcal{D}_N$, and we repeat for different draws of $\mathcal{D}_N$ to give a distribution of $\hat{s}$. Both the theoretical s^* and empirical $\hat{s}$ depend on the pretraining and task DGs, the sequence length T, and the prompt length L or $L_{\max}$; the empirical $\hat{s}$ depends also on the predictor $p_{(\cdot)}$ and dataset size N. Each configuration forms a *prompt setup*, which we vary experimentally. Previous work on prompt search (e.g. Pryzant et al., 2023; Hao et al., 2024; Deng et al., 2022) did not systematically study the effects of these hyperparameters, while brittleness of optimized prompts is known (He et al., 2024; M. Li et al., 2023; Ngweta et al., 2025; Shi et al., 2024; Peng et al., 2025).

To interpret optimal prompts, we consider two classes of task distributions: the task DGs q can be either *in-meta-distribution* (IMD) w.r.t. p, where $q \in \mathcal{M}_p := \{p_{x|\tau}(\cdot|\tau_0)\,|\,p_\tau(\tau_0) > 0\}$, or *out-of-meta-distribution* (OOMD) w.r.t. p ($q \notin \mathcal{M}_p$). See Table 1 for examples illustrated by CIB-DGs. This separation formalizes the notion of a task being "within" or "outside" the pretraining distribution in frontier model research (Xinyi Wang et al., 2023; C. Wei, Xie, et al., 2021; Krishna et al., 2023; Petrov et al., 2024). In the IMD case, we can relate the prompt to the task sequence x:

Proposition 4.1. *Prompting a predictor p by s^* for all $q \in \mathcal{M}_p$ maximizes the mutual information between the prompt s and the sequence x penalized by a "prompt alignment" with the predictor p.*

This informal version of Theorem A.2, proved in Section A.4, reveals that the theoretical s^* must induce a desired distribution on x, but is also affected by the predictor p and in turn the pretraining distribution. Specifically, the theoretically optimal prompt is, in general, *not* the most likely sequence or a sample from q. As such, we do not expect the optimal prompts to be interpretable based on the task distribution alone. Meanwhile, the empirical $\hat{s}$ is stochastic due to randomness in $\mathcal{D}_N$ and pretraining. We visualize its distribution for each prompt setup. To measure the reliability of empirical $\hat{s}$, we estimate the probability that $\hat{s}$ matches s^*, or *proportion correct* (see Section A.5), averaged over dataset draws and network instances. We also visualize the loss "landscape" (see Section A.6) of each prompt setup. A sharper loss "landscape" around s^* can improve reliability, and *vice versa*. A flat landscape can lead to worse identifiability. Although the near-optimal prompts are acceptable for performance, inconsistent results often complicate interpretation.

Advantages of simplified setup. Despite its obvious simplicity, using binary sequences and small-scale neural predictors brings the following advantages that are infeasible at full-scale LLMs:

1. The optimal Bayes predictor is available: the ide-

alized limit any model or data scaling approaches;
2. Exhaustive search to obtain numerically guaranteed optimal prompts;
3. Visualizing the prompt distribution as the prompt set up changes;
4. Repeating experiments over many random seeds to produce robust results.

5 RESULTS ON CIB-DGS

We take four pairs of pretraining and task CIB-DGs (p and q), listed in Table 1; the motivation is to cover as many prompt setups as possible, including discrete/continuous pretraining τ, and IMD/OOMD tasks, while being able to easily visualize the optimal prompts. For each pair, to find s^*, we sweep sequence length $T \in \{1, 3, 10, 30, 100\}$, and maximum prompt length $L_{\max} \in \{5, 10, 15\}$; to find $\hat{s}$, we additionally sweep optimization datasets consisting of $N \in \{10^1, 10^2, 10^3, 10^4\}$ sequences, and 7 predictor types (3 in main text). For the Bayes predictor, we draw $\mathcal{D}_N$ 10^3 times with different seeds; for each neural predictor type, we pretrain 30 instances to be near-optimal (Figure 6 in Section B) and find $\hat{s}$ for each on drawn $\mathcal{D}_N$. All neural predictors are trained using Jax/Haiku (Bradbury et al., 2018; DeepMind et al., 2020; Hennigan et al., 2020), with parameters initialized by their default method, and optimized by Adam (Kingma et al., 2015) with their default hyperparameters; see Section B.1 for detail.

5.1 Optimal Prompt is Atypical

Consider a pretraining DG $p = \text{BernMix}(0.2, 0.7)$ and task DG $q = \text{Bern}(0.7)$ ($q \in \mathcal{M}_p$). The theoretical s^* is always a sequence of all ONEs, since it causes the posterior $p_{\tau|x}$ to concentrate on 0.7 most rapidly. However, such a prompt is atypical to the task DG, and would be unintuitive without knowing p. Optimized natural language prompts can also be surprising for the given task (e.g. Daras et al., 2022; He et al., 2024; Melamed et al., 2025); which may be more effective at shifting the model's belief than more intuitive prompts.

How likely can we find such a prompt empirically? As Figure 2(left) shows, the empirical $\hat{s}$ is mostly correct for longer sequences when $L_{\max} = 5$, but not at $L_{\max} = 10$ for the Transformer, even with a large N and T. In addition, increasing sequence length T can *lower* the proportion correct, even when prompting the Bayes predictor; see Section B.2.2 for an explanation. In addition, Section B.2.3 shows a flat loss "landscape": there are many suboptimal prompts with similar losses (4) as s^*. Thus, the randomness in sampling $\mathcal{D}_N$ can shift the optimum to a different prompt near s^*. On frontier models, prompt search typically uses a small batch size (≈ 200) (Pryzant et al., 2023; Fernando et al., 2023; Hao et al., 2024; Deng et al., 2022), raising the question how reliable the optimized prompts are.

How can the proportion correct decrease with increasing N? Figure 2(right) shows the distribution of empirical $\hat{s}$ for $L_{\max} = 5$. As N increases, the *support* of this distribution approaches s^*, but the *probability* of matching s^* may decrease (e.g., yellow dots). Section B.2.4 presents the distribution of $\hat{s}$ from all predictors, showing non-convergence to s^*, with inconsistent counts of ZEROs and ONEs across runs. Note that the prompting strategy of using samples from the task $q \in \mathcal{M}_p$ is most likely suboptimal: sampling the all-ONE sequence from the task DG is increasingly unlikely as T increases. Knowledge of the pretraining DG is crucial to come up with and make sense of s^*.

5.2 Are Shorter Prompts Better?

In the previous example, the optimal prompt had maximal length $L_{\max}$. Are longer prompts better? Let $p = \text{BernMix}(0.2, 0.7)$ as before, and now $q = \text{Bern}(0.6)$ ($q \notin \mathcal{M}_p$). Figure 3(left) shows that the ratio of ONEs in the theoretical s^* appears irrelevant to $\tau = 0.6$ in q, so interpreting it given q alone is futile. See Section B.3.1 for an explanation. More interestingly, at a given maximum length constraint $L_{\max}$, the theoretical s^* can shorten or lengthen with no obvious pattern as T increases (Figure 10). It can leave an *impression* that the best prompts remain short even when longer prompts are tested, but, as we gradually increase $L_{\max}$, $s^*_{L_{\max}}$ never shortens and can suddenly jump to a longer one. These results suggest that previously reported higher efficacy of shorter natural language prompts (Bhargava et al., 2023; Renze et al., 2024; Kusano et al., 2024; Z. Wang et al., 2025; Lester et al., 2021) could be due to insufficient search at longer prompt lengths, which is inevitable given the much larger prompt space. Figure 3(right) shows that, for $L_{\max} = 10$, the empirical $\hat{s}$ gets more unreliable as T increases, regardless of the dataset size and the predictor, due to the flat loss "landscape" (Section B.3.5). The distribution of $\hat{s}$ does *not* always concentrate around s^*, complicating the interpretation of empirically optimized prompts.

5.3 Continuous Latent Factors

We now consider a pretraining $p = \text{BetaBern}(1, 1)$ (uniform over $[0, 1]$) and two tasks $q = \text{Bern}(\tau)$ with $\tau \in \{0.7, 0.9\}$ ($q \in \mathcal{M}_p$). Intuitively, the theoretical $s^*_{L_{\max}}$ should have roughly $\tau L_{\max}$ ONEs and $(1-\tau)L_{\max}$ ZEROs. As Figure 4(left) shows, this only holds for large enough T and $L_{\max}$. The detailed mismatch for shorter T can be explained if we know the pretraining p; see Section B.4.3. In addition, Figure 4(right) shows a close match in the proportion correct between all predictors,

Table 1: Summary of results on conditionally independent Bernoulli data generators.

Pretraining p	**Task** q	s^* **matches** τ **in** q**?**	$\hat{s} = s^*$ **reliably**	
			Bayes	**Neural**
BernMix$(0.2, 0.7)$	Bern$(0.7) \in \mathcal{M}_p$	No, extreme counts	No	No
	Bern$(0.6) \notin \mathcal{M}_p$	No, unintuitive optimal length	No	No
BetaBern$(1, 1)$	Bern$(\tau) \in \mathcal{M}_p$	Yes if $p(\tau)$ uniform, N, T large	Yes	Yes?
	BernMix$(\tau_1, \tau_2) \notin \mathcal{M}_p$	Yes, for mean bias	Yes	Yes?

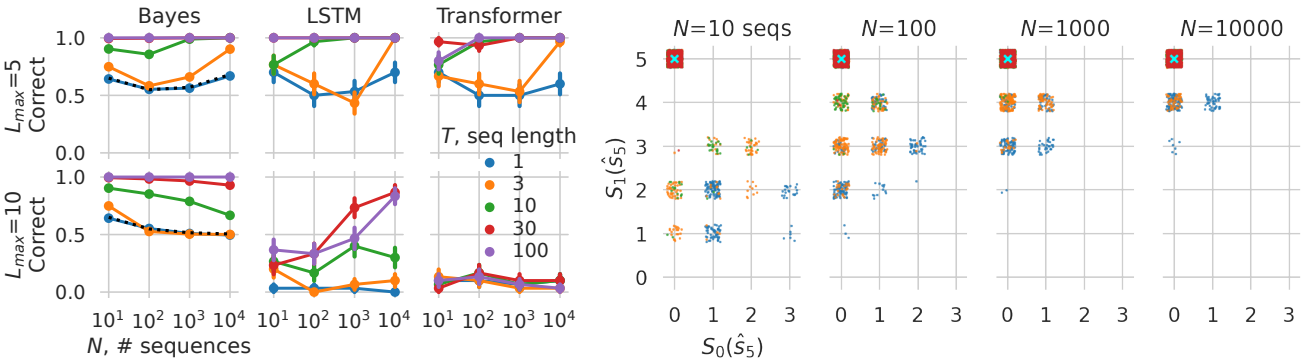

Figure 2: Results for a pretraining DG $p = \text{BernMix}(0.2, 0.7)$ and a task DG $q = \text{Bern}(0.7)$. Left, the proportion correct for Bayes predictor (10^3 seeds per data point) and two neural predictors (30 seeds per data point). Error bars show 1 SEM. The black dotted line is the theoretical value for $T = 1$ (Section B.2.2). More data does not always improve reliability. Additional results are in Section B.2.1. Right, empirically optimal $\hat{s}$ at $L_{\max} = 5$ for the Bayes predictor for different values of T (colors) and N (panels); 100 repetitions per setting. The cyan cross shows the correct all-ONE s^*_5. The set of $\hat{s}$ moves towards s^*, but the proportion of $\hat{s} = s^*$ does not necessarily increase.

in contrast to previous cases. Overall, increasing T and N helps identify the theoretical s^* more reliably, consistent with the sharper loss "landscape" in Section B.4.5 compared to previous cases. However, uniformity on τ is unlikely to hold for real languages that exhibit biases in multiple domains (R. Song et al., 2023; Kotek et al., 2023). Indeed, the ratio of ONEs in s^* deviates from the true bias in q if the p_τ is not uniform, such as when $p = \text{BetaBern}(1, 2)$; see Figure 4(left) and Section B.4. Methods that control for the bias (e.g. Gagne et al., 2023; Shirafuji et al., 2024; Delobelle et al., 2022) could improve interpretability and reliability of the prompts.

In all previous examples, the task DG was a coin with fixed bias (Bern(τ)). In Section B.5, we present an example where $q = \text{BernMix}(\tau_1, \tau_2)$ is a mixture, which requires switching between two values, rather than a continuum, of τ. The optimal prompt length varies depending on q, and would again be difficult to explain without knowing the pretraining DG. As a side note, since $p_{\tau|x}$ is always single-mode during pretraining, prompting can only partially improve performance if $|\tau_1 - \tau_2|$ is large. This leads to an idea of using a mixture of prompted predictors, each focusing on one τ, as done in (Choi et al., 2023; Dun et al., 2025).

6 MORE COMPLEX BINARY SEQUENCES

The summarizing Table 1 suggests that optimal prompts may be more interpretable if p_τ is uniform and $q \in \mathcal{M}_p$. However, this is not always the case, as we show in Section C.1 with non-i.i.d. DGs. Going beyond conditional independence, we constructed DGs with more complex structures in Sections C.2 and C.3 to simulate basic features of natural language. The optimal prompts are harder to understand, suitable for more experienced readers. Below, we apply our framework to an in-context learning scenario and real LLMs.

6.1 Bandit Decision-Maker

We have seen that optimal prompts may not be typical. This is relevant to popular in-context (reinforcement) learning approaches, where one prompts a predictor with expert demonstrations on example problems in a problem class (e.g. some chess games) to induce expert behavior on new problems from the class (a new chess game) (e.g., Ruoss et al., 2024; Agarwal et al., 2024; Laskin et al., 2023; Luo et al., 2024; B. Xu et al., 2023; N. Wang et al., 2024; Z. Dai et al., 2024; Huang et al., 2022; Yao et al., 2023; Costarelli et al., n.d.; Mirchandani et al., 2023). However, ensuring

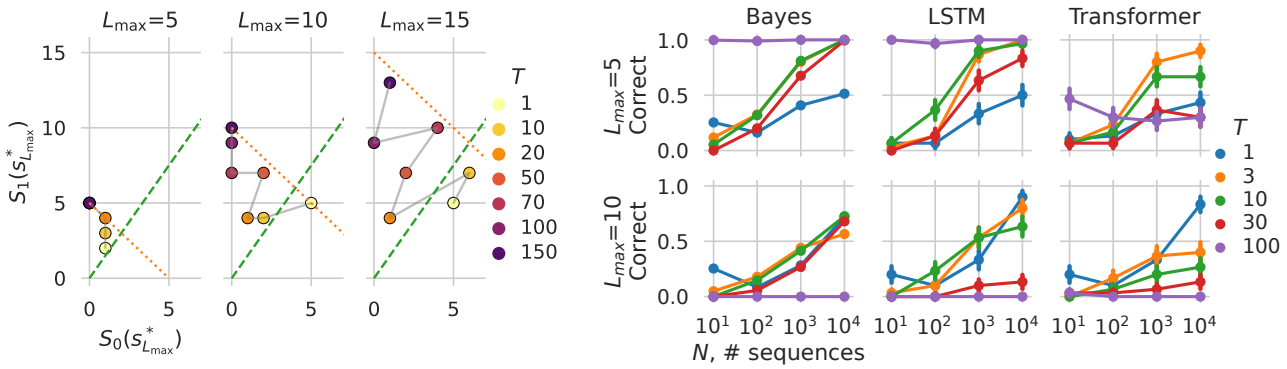

Figure 3: Results for $p = \text{BernMix}(0.2, 0.7)$ and $q = \text{Bern}(0.6)$. Left, each circle represents the ZERO/ONE counts of the theoretical s^*. The orange dotted line indicates $L_{\max}$. The green dashed line marks 60% ONEs. The s^* jumps unpredictably before converging to all-ONE. Right, the proportion correct of $\hat{s} = s^*$. They mostly increase except for $T = 100$ when $L_{\max} = 10$. Figure 11 shows additional results.

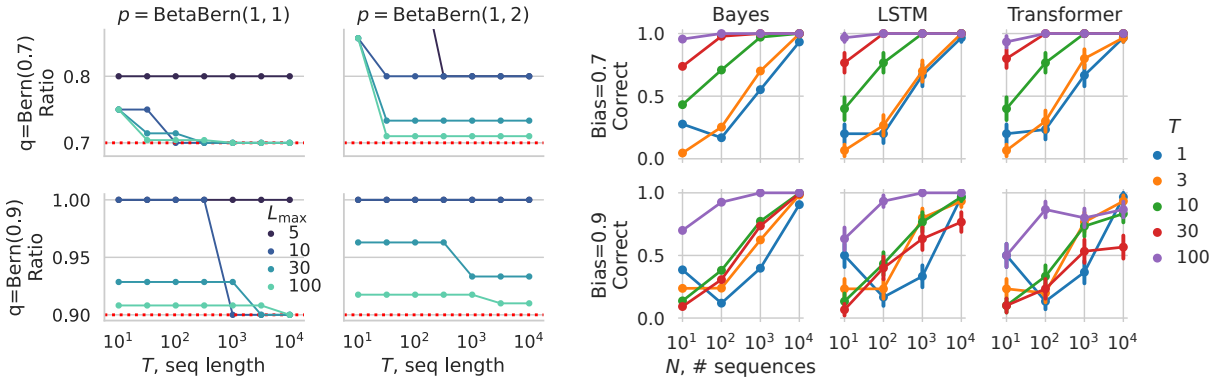

Figure 4: for $p = \text{BetaBern}(1, \beta)$ with $\beta \in \{1, 2\}$, and $q = \text{Bern}(\tau)$ with $\tau \in \{0.7, 0.9\}$. Left, the ratio of ONEs in the theoretical s^*. Red dotted line shows true bias of q. The ratio goes towards the task τ when N and T are large and p_τ uniform. Right, proportion correct ($\beta = 1$) increases as N increases. Figure 14 has more results.

optimality for the LLM prompts is virtually impossible. In our final experiment, we setup a well-controlled problem class suitable for binary sequence predictors as the agents, and prompt the binary agents (including a Bayes predictor) to attain maximal returns, rather than minimal log-losses used before.

The problem class is a two-arm Bernoulli bandit (Figure 5). In each episode, problems (games) are created by drawing the latent reward probabilities of the left and right arms (v_{L} and v_{R}) i.i.d. from Uniform($[0, 1]$). Then, at each step t, the agent chooses an action $a_t \in \{\text{L}, \text{R}\}$ given past actions and rewards, and obtains a new reward $r_t \sim \text{Bernoulli}(v_{a_t})$. The goal is to find a policy that maximizes the expected total return $\mathbb{E}[\sum_{i=1}^{T} r_i]$ for $T = 300$, estimated over many episodes.

Agent mixture. We design a distribution of agents with different skill levels τ (hidden from predictors), simulating the diverse skills in the text corpora of LLMs. At time t, each agent maintains the counts of rewarded $S_{b,1,t}$ and unrewarded $S_{b,0,t}$ actions on each arm ($b \in \{\text{L}, \text{R}\}$) up to time t, and takes an action $a_{t+1} = \arg\max_{b \in \{\text{L},\text{R}\}} \{v_{b,t,\tau}\}$, where $v_{b,t,\tau} \sim \text{Beta}(\tau S_{b,1,t} + 1, \tau S_{b,0,t} + 1)$ for $b \in \{\text{L}, \text{R}\}$. Here, $\tau \in [0, 1]$ is the latent skill, with $\tau = 0$ giving uniformly random actions and $\tau = 1$ corresponding to the asymptotically optimal Thompson sampling (TS) agent (Thompson, 1933; Agrawal et al., 2012), see Section D.1 for details of the skill levels.

Pretraining. Given a trajectory predictor, the purpose of a prompt is to induce a *skill* level, not the arm probabilities v's, to solve new problems of the bandit class with unknown v's. Thus, we design each pretraining trajectory by concatenating a prompt segment and a rollout segment, separated by a special token (see Figure 5 and Algorithm 1). The prompt segment consists of 8 actions and rewards by an agent of a randomly sampled skill. The rollout segment consists of 300 actions and rewards with the same skill but v's redrawn. Actions in the prompt segment are thus informative about the skill only. Since actions and rewards are binary and interleaved, the sequence can be represented as a binary string. Trajectories from this DG are used to pretrain neural predictors, with log-loss measured only on the action tokens. We also obtain a Bayes

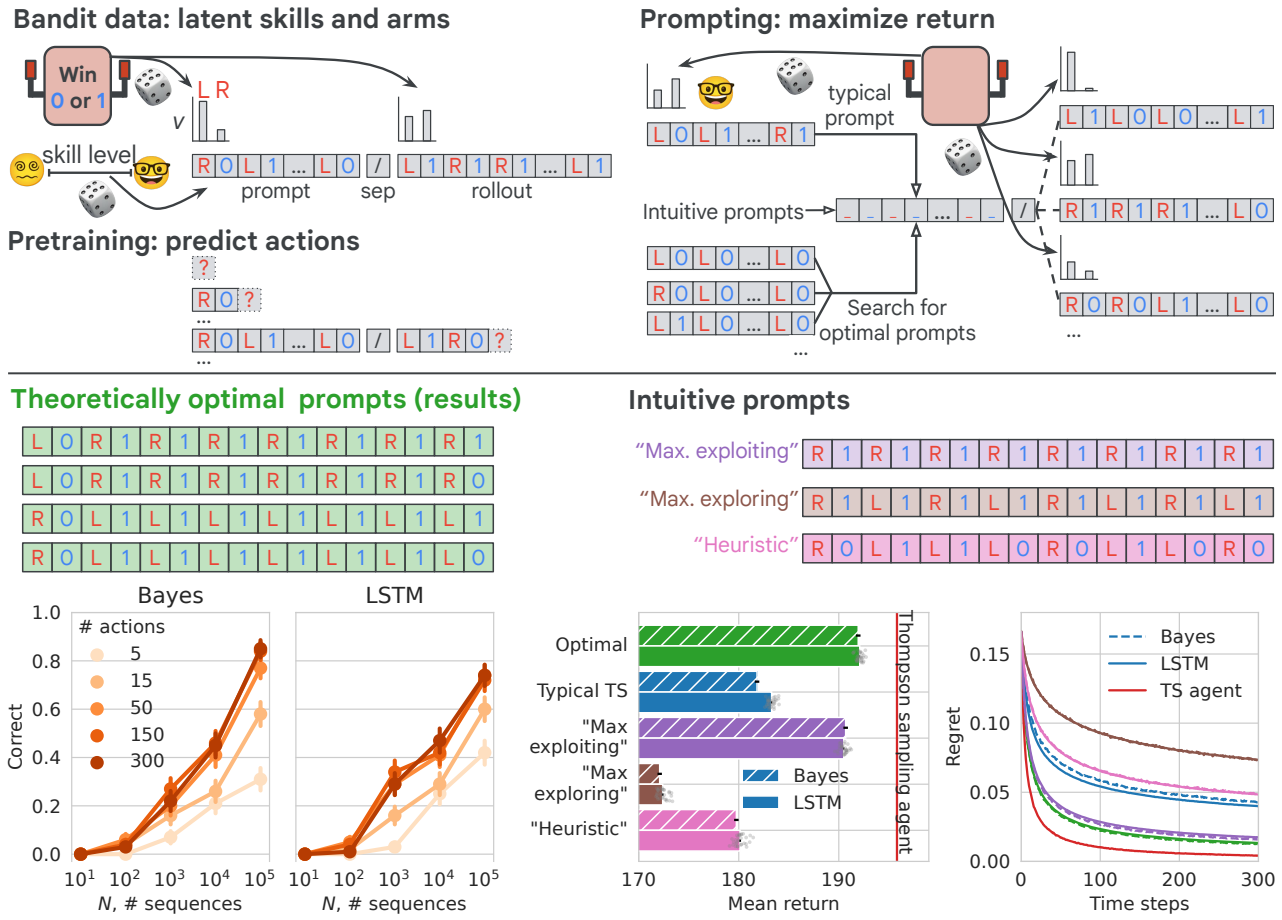

Figure 5: Upper, experiment design for the bandit task. Lower, 4 equivalent theoretical s^*, 3 heuristic prompts, and performance metrics on the bottom panels: Left two panels, the proportion correct increases slowly for the Bayes and LSTM predictors; error bars show 1 SEM over 100 pretraining runs with different weight initializations. Right two panels, the mean return and mean instantaneous regret of different prompts. We use the $\hat{s}$ found on $N = 10^5$ rollouts. Grey dots show 30 out of 100 runs.

predictor (see Section D.2). The predictor then acts stochastically according to the predicted probabilities at each action step. Since the agent mixture contains the expert TS agent, prompting for maximal return can be regarded as IMD.

Prompting methods. We prompt each predictor to maximize the expected total return in the rollout segment, from which we also estimate the instantaneous regret $\mathbb{E}[\max_a v_a - v_{a_t}]$ at each time step. We compare three prompting approaches. First, we exhaustively search through all binary prompts on a predictor, running N rollouts to estimate the return, giving $\hat{s}$. Using a large N, we found four equivalent theoretically optimal prompts s^* shown in Figure 5. Surprisingly, they show little exploration but strong exploitation to the more rewarding arm. We address this pattern in Section D.3, using knowledge of the pretraining distribution. Second, we prompt the model by demonstrations from the expert TS agent ($\tau = 1$). Finally, we handcraft several intuitive prompts: a "maximally exploring", a "maximally exploiting", and a "heuristic" prompt handcrafted by an author (Figure 5). The latter two approaches mimic common methods of in-context learning on LLMs.

Results. Figure 5(bottom left) shows the reliability (proportion correct) of the empirical $\hat{s}$ found on the Bayes and LSTM predictors. It increases very slowly as N increases, reaching around 0.8 when using as many as 10^5 rollouts to estimate the expected return of each prompt. Section D.4 shows that these suboptimal prompts can lead to largely inconsistent interpretations in terms of a classical behavioral metric (Bruner, 1957; Nowak et al., 1993). The performances of the three prompt types are in Figure 5(bottom). Typical prompts from the TS expert and the "heuristic" prompt both yield lower return than the theoretically and empirically optimal prompts, and are even worse than the "max. exploiting" prompt. The "max. exploring"

prompt gives the lowest return. These results show that the common practice of prompting by expert demonstrations does not induce expert behavior as effectively as the optimal prompt, or even the "maximally exploiting", which is surprising given that the latter does not encourage exploration at all. Therefore, even using uniform priors over the latent and ensuring the task is IMD gives unintuitive prompts. The effectiveness of "wrong" examples has been reported (Min et al., 2022; Yang et al., 2024) on LLMs, which could be due to subtle interactions between latent factors in the unknown pretraining distribution, as suggested in Section D.3. Section D.5 shows that the optimal prompt uses 1/4 of the length of expert demonstrations for the same expected return, demonstrating again its desirable high efficiency.

6.2 Real LLMs

We adapt our framework above to study prompts on LLMs, using GPT-2 (Radford et al., 2019), Gemma-3 (Team et al., 2025), and Gemini 2.5 Pro (Comanici et al., 2025) and the IMDB reviews (Maas et al., 2011). Movie reviews may not exactly satisfy the conditional independence assumptions in our synthetic datasets: people do not always start writing a movie review with a statement or summary that reveals the sentiment. In addition, longer prompts can become less consistent with reviews for specific films or genres; for example, if a prompt mentions quality of animation, this would be inconsistent with non-animated film reviews, even if the sentiments are both positive. Nonetheless, IMDB dataset offers a handle to a single latent variable (sentiment) governing the review text, and has sufficient linguistic diversity compared to other synthetic languages dataset (e.g., Xie et al., 2021; Yuan et al., 2021).

The Bayes predictor for movie reviews is unknown, and extrapolating our findings above to predict the LLM results is also hard; instead, our goal is to explore properties of optimized prompts. First, we finetune GPT-2 and Gemma-3 on the training set of IMDB, and ask Gemini 2.5 Pro to come up with a large number of prompts of variable lengths ($\approx L_{\max}$) that can induce positive or negative (task τ) movie reviews. As a sanity check, we verified that these prompts do modulate the log-losses of predicting reviews of either sentiment as intended. Then, we evaluate the conditional log-loss of the reviews with variable length cut-offs ($8 \leq T \leq 512$) and dataset size ($250 \leq N \leq 25000$) given each of the prompts. This allows us to explore many basic properties of optimized prompts as before. The results are presented in Section E. In summary, we found that the prompt sentiment is not always consistent with the intended sentiment of the reviews, and it also depends on the pretrained model. As discovered in binary sequences, the reliability of identifying the optimal prompt is very low at $N = 250$, while $N \approx 1700$ is needed for a 50% proportion correct. Finally, the optimal prompt length tends to be very short, as discovered before in the literature.

7 CONCLUSIONS

Through a large set of carefully designed experiments, we revealed fundamental patterns of optimal prompts in conceptually simple binary sequences. Our well-controlled experiments provided ample examples of binary optimal prompts and visualizations, and revealed interesting fundamental properties of optimal prompts on LLMs. This work thus presented a plausible candidate theory that (partially) accounts for the surprising prompts seen on frontier models.

Since the pretraining distribution is often unknown, interpreting optimal prompts in terms of the task alone is hard: these prompts may be atypical sequences under the task distribution, hard to find reliably given a finite dataset, and may vary with complex patterns depending on the prompt setup. Further, demonstrations from the task, as often used for in-context learning, are less efficient at inducing the intended behavior than some unintuitive prompts. These results are robust across a large number of predictors and their instances, including the idealized Bayes predictors. This indicates that some of the issues will persist at any scale, including that of frontier models, and thus contribute to previously observed difficulties in interacting with those models. We provide a practical guidance in Section F.1, elaborate on how our results relate to previous findings on prompting LLMs in Section F.2, and discuss detailed limitations and potential future work in Section F.3.

Acknowledgements

We thank the following collaborators for useful discussions and valuable feedback: Laurent Orseau, Grégoire Delétang, Peter Dayan, Noémi Éltető, and Mark Rowland.

References

Ortega, Pedro A et al. (2019). "Meta-learning of sequential strategies". In: *Technical Report*.

Brown, Tom et al. (2020). "Language models are few-shot learners". In: *Advances in neural information processing systems*.

Xie, Sang Michael et al. (2021). "An explanation of in-context learning as implicit Bayesian inference". In: *International Conference on Learning Representations*.

Wang, Xinyi et al. (2023). "Large Language Models Are Implicitly Topic Models: Explaining and Finding Good Demonstrations for In-Context Learning". In: *ICML Workshop on Efficient Systems for Foundation Models*.

Jiang, Hui (2023). "A latent space theory for emergent abilities in large language models". In: *arXiv preprint*.

Wies, Noam, Yoav Levine, and Amnon Shashua (2023). "The learnability of in-context learning". In: *Advances in Neural Information Processing Systems*.

Arora, Aryaman et al. (2024). "Bayesian scaling laws for in-context learning". In: *arXiv preprint*.

Mikulik, Vladimir et al. (2020). "Meta-trained agents implement bayes-optimal agents". In: *Advances in Neural Information Processing Systems*.

Genewein, Tim et al. (2023). "Memory-based meta-learning on non-stationary distributions". In: *International Conference on Machine Learning*.

Bhattamishra, Satwik, Kabir Ahuja, and Navin Goyal (2020). "On the ability and limitations of transformers to recognize formal languages". In: *Empirical Methods in Natural Language Processing (EMNLP)*.

Wei, Colin, Yining Chen, and Tengyu Ma (2022). "Statistically meaningful approximation: a case study on approximating turing machines with transformers". In: *Advances in Neural Information Processing Systems*.

Deletang, Gregoire et al. (2022). "Neural Networks and the Chomsky Hierarchy". In: *International Conference on Learning Representations*.

Wei, Jason, Xuezhi Wang, et al. (2022). "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in Neural Information Processing systems*.

Liu, Pengfei et al. (2023). "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing". In: *ACM Computing Surveys*.

Chen, Banghao et al. (2023). "Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review". In: *arXiv preprint*.

Marvin, Ggaliwango et al. (2023). "Prompt engineering in large language models". In: *International conference on data intelligence and cognitive informatics*. Springer.

Sahoo, Pranab et al. (2024). "A systematic survey of prompt engineering in large language models: Techniques and applications". In: *arXiv preprint*.

Song, Yuan-Feng et al. (2024). "A communication theory perspective on prompting engineering methods for large language models". In: *Journal of Computer Science and Technology*.

Khattab, Omar et al. (2023). "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines". In: *R0-FoMo:Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

Xu, Lanling et al. (2024). "Prompting large language models for recommender systems: A comprehensive framework and empirical analysis". In: *arXiv preprint*.

Pryzant, Reid et al. (2023). "Automatic Prompt Optimization with "Gradient Descent" and Beam Search". In: *Empirical Methods in Natural Language Processing*.

Wang, Xinyuan et al. (2023). "Promptagent: Strategic planning with language models enables expert-level prompt optimization". In: *arXiv preprint*.

Fernando, Chrisantha et al. (2023). "Promptbreeder: Self-referential self-improvement via prompt evolution". In: *arXiv preprint*.

Guo, Qingyan et al. (2023). "Connecting large language models with evolutionary algorithms yields powerful prompt optimizers". In: *arXiv preprint*.

Wu, Zhaoxuan et al. (2024). "Prompt Optimization with EASE? Efficient Ordering-aware Automated Selection of Exemplars". In: *ICML Workshop on In-Context Learning*.

Hao, Yaru et al. (2024). "Optimizing prompts for text-to-image generation". In: *Advances in Neural Information Processing Systems*.

Dong, Qingxiu et al. (2022). "A Survey on In-context Learning". In: *arXiv preprint*.

Ruoss, Anian et al. (2024). "LMAct: A Benchmark for In-Context Imitation Learning with Long Multimodal Demonstrations". In: *arXiv:2412.01441*.

Agarwal, Rishabh et al. (2024). "Many-shot in-context learning". In: *Advances in Neural Information Processing Systems*.

Laskin, Michael et al. (2023). "In-context Reinforcement Learning with Algorithm Distillation". In: *International Conference on Learning Representations*.

Webson, Albert and Ellie Pavlick (2022). "Do prompt-based models really understand the meaning of their prompts?" In: *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daras, Giannis and Alex Dimakis (2022). "Discovering the hidden vocabulary of DALLE-2". In: *NeurIPS 2022 Workshop on Score-Based Methods*.

Murr, Lincoln, Morgan Grainger, and David Gao (2023). "Testing LLMs on Code Generation with Varying Levels of Prompt Specificity". In: *arXiv preprint*.

He, Jia et al. (2024). "Does Prompt Formatting Have Any Impact on LLM Performance?" In: *arXiv preprint*.

Kojima, Takeshi et al. (2022). "Large language models are zero-shot reasoners". In: *Advances in Neural Information Processing systems*.

Mizrahi, Moran et al. (2024). "State of what art? a call for multi-prompt llm evaluation". In: *Transactions of the Association for Computational Linguistics*.

Zamfirescu-Pereira, JD et al. (2023). "Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts". In: *CHI Conference on Human Factors in Computing Systems*.

Khurana, Anjali, Hariharan Subramonyam, and Parmit K Chilana (2024). "Why and when llm-based assistants can go wrong: Investigating the effectiveness of prompt-based interactions for software help-seeking". In: *International Conference on Intelligent User Interfaces*.

Chen, Lingjiao, Matei Zaharia, and James Zou (2023). "How is ChatGPT's behavior changing over time?" In: *arXiv preprint*.

Wei, Alexander, Nika Haghtalab, and Jacob Steinhardt (2024). "Jailbroken: How does llm safety training fail?" In: *Advances in Neural Information Processing Systems*.

Wu, Yuanwei et al. (2023). "Jailbreaking gpt-4v via self-adversarial attacks with system prompts". In: *arXiv preprint*.

Xu, Zihao et al. (2024). "A comprehensive study of jailbreak attack versus defense for large language models". In: *Findings of the Association for Computational Linguistics*.

Cherepanova, Valeriia and James Zou (2024). "Talking Nonsense: Probing Large Language Models' Understanding of Adversarial Gibberish Inputs". In: *arXiv preprint*.

Liu, Xiaogeng et al. (2024). "AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models". In: *International Conference on Learning Representations*.

Zou, Andy et al. (2023). "Universal and transferable adversarial attacks on aligned language models". In: *arXiv preprint*.

Wenliang, Li K. and Maneesh Sahani (2018). "Neural network trained with supervision represents uncertainty by nonlinear moments". In: *Cosyne Abstract*.

Wu, Jingfeng et al. (2023). "How Many Pretraining Tasks Are Needed for In-Context Learning of Linear Regression?" In: *arXiv preprint*.

Grau-Moya, Jordi et al. (2024). "Learning Universal Predictors". In: *International Conference on Machine Learning*.

Mittal, Sarthak et al. (2024). "Does learning the right latent variables necessarily improve in-context learning?" In: *arXiv preprint*.

Hendel, Roee, Mor Geva, and Amir Globerson (2023). "In-Context Learning Creates Task Vectors". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Todd, Eric et al. (2024). "Function Vectors in Large Language Models". In: *International Conference on Learning Representations*.

Bhargava, Aman et al. (2023). "What's the magic word? A control theory of LLM prompting". In: *arXiv preprint*.

Renze, Matthew and Erhan Guven (2024). "The benefits of a concise chain of thought on problem-solving in large language models". In: *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*.

Kusano, Genki, Kosuke Akimoto, and Kunihiro Takeoka (2024). "Are Longer Prompts Always Better? Prompt Selection in Large Language Models for Recommendation Systems". In: *arXiv preprint*.

Wang, Zhijie et al. (2025). "Towards Understanding the Characteristics of Code Generation Errors Made by Large Language Models". In: *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society.

Lester, Brian, Rami Al-Rfou, and Noah Constant (2021). "The power of scale for parameter-efficient prompt tuning". In: *arXiv preprint*.

Liu, Qibang, Wenzhe Wang, and Jeffrey Willard (2025). "Effects of prompt length on domain-specific tasks for large language models". In: *arXiv preprint*.

Leidinger, Alina, Robert Van Rooij, and Ekaterina Shutova (2023). "The language of prompting: What linguistic properties make a prompt successful?" In: *Empirical Methods in Natural Language Processing*.

Deng, Mingkai et al. (2022). "RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning". In: *Empirical Methods in Natural Language Processing*.

Li, Moxin et al. (2023). "Robust Prompt Optimization for Large Language Models Against Distribution Shifts". In: *Empirical Methods in Natural Language Processing*.

Ngweta, Lilian et al. (2025). "Towards LLMs Robustness to Changes in Prompt Format Styles". In: *Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*.

Shi, Zeru et al. (2024). "Robustness-aware Automatic Prompt Optimization". In: *arXiv preprint*.

Peng, Dengyun et al. (2025). "DLPO: Towards a Robust, Efficient, and Generalizable Prompt Optimization

Framework from a Deep-Learning Perspective". In: *arXiv preprint*.

Wei, Colin, Sang Michael Xie, and Tengyu Ma (2021). "Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning". In: *Advances in Neural Information Processing Systems*.

Krishna, Kundan et al. (2023). "Downstream Datasets Make Surprisingly Good Pretraining Corpora". In: *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Petrov, Aleksandar, Philip Torr, and Adel Bibi (2024). "When Do Prompting and Prefix-Tuning Work? A Theory of Capabilities and Limitations". In: *International Conference on Learning Representations*.

Bradbury, James et al. (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13.

DeepMind et al. (2020). *The DeepMind JAX Ecosystem*. URL: http://github.com/google-deepmind.

Hennigan, Tom et al. (2020). *Haiku: Sonnet for JAX*. URL: http://github.com/deepmind/dm-haiku.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*.

Melamed, Rimon, Lucas H McCabe, and H Howie Huang (2025). "Demystifying optimized prompts in language models". In: *arXiv preprint*.

Song, Rui et al. (2023). "Measuring and mitigating language model biases in abusive language detection". In: *Information Processing and Management*.

Kotek, Hadas, Rikker Dockum, and David Sun (2023). "Gender bias and stereotypes in large language models". In: *ACM collective intelligence conference*.

Gagne, Chris and Peter Dayan (2023). "The Inner Sentiments of a Thought". In: *arXiv preprint*.

Shirafuji, Daiki, Makoto Takenaka, and Shinya Taguchi (2024). "Bias Vector: Mitigating Biases in Language Models with Task Arithmetic Approach". In: *arXiv preprint*.

Delobelle, Pieter and Bettina Berendt (2022). "Fairdistillation: mitigating stereotyping in language models". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

Choi, Joon-Young et al. (2023). "SMoP: Towards Efficient and Effective Prompt Tuning with Sparse Mixture-of-Prompts". In: *Empirical Methods in Natural Language Processing*.

Dun, Chen et al. (2025). "Sweeping heterogeneity with smart mops: Mixture of prompts for LLM task adaptation". In: *AAAI Conference on Artificial Intelligence*.

Luo, Man et al. (2024). "In-context learning with retrieved demonstrations for language models: A survey". In: *arXiv preprint*.

Xu, Benfeng et al. (2023). "Expertprompting: Instructing large language models to be distinguished experts". In: *arXiv preprint*.

Wang, Noah et al. (2024). "RoleLLM: Benchmarking, Eliciting, and Enhancing Role-Playing Abilities of Large Language Models". In: *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*.

Dai, Zhenwen, Federico Tomasi, and Sina Ghiassian (2024). "In-context Exploration-Exploitation for Reinforcement Learning". In: *International Conference on Learning Representations*.

Huang, Wenlong et al. (2022). "Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents". In: *International Conference on Machine Learning*.

Yao, Shunyu et al. (2023). "React: Synergizing reasoning and acting in language models". In: *International Conference on Learning Representations (ICLR)*.

Costarelli, Anthony et al. (n.d.). "GameBench: Evaluating Strategic Reasoning Abilities of LLM Agents". In: *Language Gamification-NeurIPS 2024 Workshop*.

Mirchandani, Suvir et al. (2023). "Large Language Models as General Pattern Machines". In: *Conference on Robot Learning*. PMLR.

Thompson, William R (1933). "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples". In: *Biometrika*.

Agrawal, Shipra and Navin Goyal (2012). "Analysis of thompson sampling for the multi-armed bandit problem". In: *Conference on Learning Theory*.

Bruner, Jerome S (1957). "On perceptual readiness." In: *Psychological review*.

Nowak, Martin and Karl Sigmund (1993). "A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game". In: *Nature*.

Min, Sewon et al. (2022). "Rethinking the role of demonstrations: What makes in-context learning work?" In: *Empirical Methods in Natural Language Processing*.

Yang, Chih-Kai, Kuan-Po Huang, and Hung-yi Lee (2024). "Do Prompts Really Prompt? Exploring the Prompt Understanding Capability of Whisper". In: *arXiv preprint*.

Radford, Alec et al. (2019). "Language models are unsupervised multitask learners". In: *OpenAI blog*.

Team, Gemma et al. (2025). "Gemma 3 technical report". In: *arXiv preprint*.

Comanici, Gheorghe et al. (2025). "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities". In: *arXiv preprint*.

Maas, Andrew L. et al. (2011). "Learning Word Vectors for Sentiment Analysis". In: *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Yuan, Ann et al. (2021). "SynthBio: A Case Study in Faster Curation of Text Datasets". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Vaswani, A (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems*.

Elman, Jeffrey L (1990). "Finding structure in time". In: *Cognitive science*.

Hochreiter, S and J Schmidhuber (1997). "Long short-term memory." In: *Neural Computation*.

Beck, Maximilian et al. (2024). "xLSTM: Extended Long Short-Term Memory". In: *arXiv preprint*.

Katharopoulos, Angelos et al. (2020). "Transformers are rnns: Fast autoregressive transformers with linear attention". In: *International conference on machine learning*.

Li, Rui et al. (2020). "Linear attention mechanism: An efficient attention for semantic segmentation". In: *arXiv preprint*.

Shen, Zhuoran et al. (2021). "Efficient attention: Attention with linear complexities". In: *IEEE/CVF winter conference on applications of computer vision*.

Anil, Cem et al. (2022). "Exploring length generalization in large language models". In: *Advances in Neural Information Processing Systems*.

Vavasis, Stephen A (1990). "Quadratic programming is in NP". In: *Information Processing Letters*.

Harris, Charles R. et al. (2020). "Array programming with NumPy". In: *Nature*.

Chang, Kaiyan et al. (2024). "Efficient Prompting Methods for Large Language Models: A Survey". In: *arXiv preprint*.

Blei, David M, Andrew Y Ng, and Michael I Jordan (2003). "Latent dirichlet allocation". In: *Journal of machine Learning research*.

Cook, John D. (2021). *Fast approximation of Beta inequalities*. Tech. rep. The University of Texas MD Anderson Cancer Center.

Razeghi, Yasaman et al. (2022). "Impact of Pretraining Term Frequencies on Few-Shot Numerical Reasoning". In: *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Wei, Jason, Dan Garrette, et al. (2021). "Frequency Effects on Syntactic Rule Learning in Transformers". In: *Empirical Methods in Natural Language Processing*.

Rajaraman, Nived, Jiantao Jiao, and Kannan Ramchandran (2024). "Toward a Theory of Tokenization in LLMs". In: *arXiv preprint*.

Ieremie, Ioan, Rob M Ewing, and Mahesan Niranjan (2024). "Protein language models meet reduced amino acid alphabets". In: *Bioinformatics*.

Gagie, Travis (2012). "A Note on Sequence Prediction over Large Alphabets". In: *Algorithms*.

Heurtel-Depeiges, David et al. (2024). "Compression via pre-trained transformers: A study on byte-level multimodal data". In: *arXiv preprint*.

Chlon, Leon et al. (2025). "LLMs are Bayesian, in Expectation, not in Realization". In: *arXiv preprint*.

Allen-Zhu, Zeyuan and Yuanzhi Li (2023). "Physics of language models: Part 3.1, knowledge storage and extraction". In: *arXiv preprint*.

Chan, Stephanie CY et al. (2022). "Transformers generalize differently from information stored in context vs in weights". In: *arXiv preprint*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]. Theorem A.2
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes, the experiments are clearly described; also see Algorithm 1. We will not provide code.]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] Section E
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] In all figures we defined the error bars. Also in Section B.1.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] Section B.1

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Why Is Prompting Hard? Understanding Prompts on Binary Sequence Predictors: Supplementary Materials

Table of Contents

A METHOD DETAILS

A.1 Near-Bayes-Optimal Prediction Can Lead to "Super-Bayes" Performance on Certain Tasks.

Previous work has shown that well-trained neural predictors behave almost identically to the Bayes predictor in terms of their predictions given data from the pretraining distribution p (e.g. Mikulik et al., 2020; Genewein et al., 2023; Grau-Moya et al., 2024). However, these results do not imply a bound on the neural predictor by the same Bayes predictor given an intended task. Specifically, even though we have the optimality bound from positivity of the KL divergence:

$$\mathbb{E}_{s\sim p}[\log p_\theta(s)] \leq \mathbb{E}_{s\sim p}[\log p(s)]\,,$$

this does *not* imply that the log-loss under a potentially different task distribution q preserve the ordering:

$$\mathbb{E}_{s\sim q}[\log p_\theta(s)] \not\leq \mathbb{E}_{s\sim q}[\log p(s)]\,.$$

This holds even if $q \in \mathcal{M}_p$. This happens when q favors specific regions of the sequence space where the neural predictor p_θ is accidentally better than the Bayes predictor p_{B} that is optimal for the pretraining p but not for the task q.

A.2 Training Neural Predictors

The neural predictors share the same overall structure, involving the following stages:

1. Map each binary tokens $x_t \in \{0, 1\}$ to embeddings $e_t \in \mathbb{R}^h$ through $e_t = W_{\mathrm{emb}} x_t$ where $W_{\mathrm{emb}} \in \mathbb{R}^{h\times 2}$, and $h \in \mathbb{N}^+$ is the hidden size.
2. Sequentially map $h_{1:T}$ through some neural architecture, called the *torso*, such as an LSTM and a multi-head attention, to obtain hidden activations $u_t \in \mathbb{R}^h$ for each t.
3. For each t, map u_t through the fully connected MLP to $v_t \in \mathbb{R}^h$ that is usually found after the attention layer in a transformer block (Vaswani, 2017).
4. For each t, map v_t to output logits through a linear map.

There is also a residual connection from step 2 to 3 and from 3 to 4. The different neural architectures differ only by the torso. This maintains a flexible enough architecture for different tasks while controlling for the model complexity between different architectures.

For the torso, we use the following variants of recurrent networks and transformers:

1. Vanilla recurrent neural networks (Elman, 1990);
2. Long-short term memory (LSTM) (Hochreiter et al., 1997), reported in main text;
3. sLSTM (Beck et al., 2024)
4. Softmax-attention transformer (Transformer) (Vaswani, 2017), reported in main text;
5. Linear transformer (Katharopoulos et al., 2020)
6. Another variant of Linear transformer we refer to as Inner-product transformer (IP transformer) (R. Li et al., 2020; Shen et al., 2021)

We found that step 3 above is crucial for transformer architectures to perform some of the tasks, although this was not essential for LSTMs to perform well. We did not use normalization or dropout layers for simplicity. These hyperparameters for all networks and pretraining DGs are listed in Table 2. During pretraining, we made sure that the sequence length was long enough to avoid bad generalization over unseen sequence lengths (Deletang et al., 2022; Anil et al., 2022) in downstream tasks.

Table 2: Hyperparameters for neural predictors. *For vanilla RNN, we train 5 million steps.

	Common parameters			Recurrent	Transformer		
Pretraining DG	hidden size h	# steps	T	learning rate	# head	# layer	learning rate
BernMix	128	300k	180	1×10^{-4}	1	1	1×10^{-4}
BetaBern	128	500k*	180	1×10^{-4}	1	1	1×10^{-4}
Switching	128	3 m	180	3×10^{-5}	8	1	3×10^{-5}
Bandit	256	1 m	630	1×10^{-4}	8	2	3×10^{-5}

A.3 Prompt Search Method

On the CIB-DG experiments in Section 4, both the prompt s and the sequence(-to-predict) x are permutation invariant under the Bayes predictor p_{B}, which means that the counts of ZEROs and ONEs in Equation (2) form sufficient summary of the prompt and the sequence under p_{B}. This drastically reduces the search space of the theoretically optimal prompt on the Bayes predictor. Also, by using the equivalent binomial distribution defined over the counts, the summation in Equation (4) can also be reduced for the Bayes predictor. This makes searching for $s^*(p_{\mathrm{B}}, \cdot)$ and $\hat{s}(p_{\mathrm{B}}, \cdot, \cdot)$ very efficient. However, these savings do not apply to neural predictors, so we searched through all possible prompts and sequences.

The DGs in bandit task (Section 6.1) and the switching task (Section C.1) have no nontrivial symmetry in the space of all prompts and all sequences/rollouts, so we exhaustively searched through all possible prompts and sequences/rollouts to find s^*. This is also done on finite datasets, we also searched through all possible prompts on the given task dataset to find $\hat{s}$.

A.4 Information-Theoretic Justification

We show that prompt optimization over the objective in Equation (4) is equivalent to maximizing an information-theoretic objective relating the prompt and the sequence-to-predict for the IMD case $q \in \mathbb{M}_p$. In this subsection only, to reduce notational clutter and avoid specifying lengths T and L, we temporarily define $\mathbf{x}$ as the sequence and $\mathbf{s}$ as the prompt. The latent variable τ is not restricted to a scalar. We *do* still assume that $\mathbf{x}$ is conditionally independent of $\mathbf{s}$ given τ. As before, the pretraining DG is then $p(\mathbf{x}) := \int p_\tau(\tau) p_{x|\tau}(\mathbf{x}|\tau) \mathrm{d}\tau$.

Denote the pretrained predictor by $p_{(\cdot)}(\mathbf{x}|\mathbf{s})$, which can be the Bayes predictor $p_{\mathrm{B}}(\mathbf{x}|\mathbf{s})$ or a pretrained neural predictor $p_\theta(\mathbf{x}|\mathbf{s})$. In addition, define the *prompting strategy* as $\nu(\mathbf{s}|\tau)$ mapping from a given τ to a distribution over the prompt. This is a strategy because this maps our intended task τ to a prompt sequence that is passed to the sequence predictor. We also allow this strategy to be stochastic. This strategy leads to a prompt policy-augmented joint distribution over the prompt and the sequence:

$$p^\nu(\mathbf{s}, \mathbf{x}) := \int p_\tau(\tau) \nu(\mathbf{s}|\tau) p_{x|\tau}(\mathbf{x}|\tau) \mathrm{d}\tau \,. \tag{7}$$

The prompt distribution ν need not be the same as $p_{x|\tau}$, which would suggest using samples from $p_{x|\tau}$ as the prompt. By construction, the augmentation leaves the marginal over $\mathbf{x}$ intact in the absence of any prompt: $p^\nu(\mathbf{x}) = p(\mathbf{x})$.

What would be a best prompting strategy? First, there should be high mutual information between $\mathbf{x}$ and $\mathbf{s}$ under p^ν, so that varying $\mathbf{s}$ effectively manipulates the distribution over $\mathbf{x}$. However, this condition alone is not sufficient. Consider the strategy that maps some partition over the space of the task variable τ to distinct sequences of $\mathbf{s}$ unrelated to the pretraining distribution. This prompt strategy can achieve a high mutual information, but $\mathbf{s}$ completely ignores any statistical regularities in $p(\mathbf{x})$ baked into the predictor $p_{(\cdot)}$. Thus, this strategy is unlikely to steer the predictor $p_{(\cdot)}$ in the same way as it conditions the pretraining DG (through the task latent τ).

To fix this, the best prompting strategy must ensure that the prompt must condition the predictor in a similar way to how it conditions the pretraining DG, i.e. $p_{(\cdot)}(\mathbf{x}|\mathbf{s})$ must be close to $p(\mathbf{x}|\mathbf{s})$, or aligned with the predictor. Combining these ideas together we arrive at the following objective.

Definition A.1. The predictor-aligned mutual information (PAMI) is defined as

$$\mathrm{PAMI}(\mathbf{s}, \mathbf{x}) := \mathrm{MI}_{p^\nu}(\mathbf{s}; \mathbf{x}) - \mathbb{E}_{\mathbf{s} \sim p^\nu}[\mathrm{KL}[p^\nu(\mathbf{x}|\mathbf{s}) \| p_{(\cdot)}(\mathbf{x}|\mathbf{s})]] \,. \tag{8}$$

This objective trades off the specification of $\mathbf{x}$ through $\mathbf{s}$ under p^ν and the alignment between conditioning on p^ν and on $p_{(\cdot)}$. Though heuristically defined, the optimal strategy under PAMI is one that optimizes the predictive log-likelihood under all specified (IMD) tasks.

Proposition A.2. *The deterministic prompt distribution* $\nu(\mathbf{s}|\tau) = \delta_{\mathbf{s}^*(\tau)}$ *centered at*

$$\mathbf{s}^*(\tau) := \arg\max_{\mathbf{s}} \sum_{\mathbf{x}} p(\mathbf{x}|\tau) \log p_{(\cdot)}(\mathbf{x}|\mathbf{s})$$

for all τ *maximizes* $\mathrm{PAMI}(\mathbf{s}; \mathbf{x})$.

Proof. PAMI can be rewritten into a form similar to the conventional mutual information:

$$\begin{aligned}\mathrm{PAMI}(\mathbf{s}, \mathbf{x}) &= \sum_{\mathbf{s},\mathbf{x}} p^\nu(\mathbf{s}, \mathbf{x}) \log \frac{p^\nu(\mathbf{x}|\mathbf{s})}{p^\nu(\mathbf{x})} - \sum_{\mathbf{s},\mathbf{x}} p^\nu(\mathbf{s}, \mathbf{x}) \log \frac{p^\nu(\mathbf{x}|\mathbf{s})}{p_{(\cdot)}(\mathbf{x}|\mathbf{s})} = \sum_{\mathbf{s},\mathbf{x}} p^\nu(\mathbf{s}, \mathbf{x}) \log \frac{p_{(\cdot)}(\mathbf{x}|\mathbf{s})}{p^\nu(\mathbf{x})} \\ &= H[p(\mathbf{x})] + \sum_{\mathbf{s},\mathbf{x}} p^\nu(\mathbf{s}, \mathbf{x}) \log p_{(\cdot)}(\mathbf{x}|\mathbf{s}). \end{aligned} \tag{9}$$

The first term in Equation (9) is independent of ν, so we only need to show that $\delta_{\mathbf{x}^*(\tau)}$ maximizes the second term. This term expands to

$$\int p_\tau(\tau) \sum_{\mathbf{s}} \nu(\mathbf{s}|\tau) \sum_{\mathbf{x}} p_{x|\tau}(\mathbf{x}|\tau) \log p_{(\cdot)}(\mathbf{x}|\mathbf{s}) \mathrm{d}\tau \,.$$

The Dirac delta measure $\delta_{\mathbf{x}^*(\tau)}$ assigns all mass to $\mathbf{s}^*(\tau)$ that maximizes $\sum_{\mathbf{x}} p_{x|\tau}(\mathbf{x}|\tau) \log p_{(\cdot)}(\mathbf{x}|\mathbf{s})$ for each given τ. This holds for all τ, and thus $\delta_{\mathbf{s}^*(\tau)}$ maximizes the second term of Equation (9). □

Now it may be obvious that PAMI is simply a re-arrangement of the log-likelihood, the decomposition of prompting into Informativeness (MI) and Alignment (KL) perfectly encapsulates the reason by prompting by intuition usually fails: humans tend to only optimize for the mutual information, completely ignoring the KL penalty imposed by the model's unknown pretraining distribution. An optimal prompt might look like gibberish to a human, but it perfectly hacks the pretraining priors to reduce that KL penalty.

A.5 Optimal Prompt and Criterion for an Empirically Optimal Prompt to be Correct

For each prompt setup in our experiment, we found multiple theoretical s^*'s that give the same expected log-loss (4). For CIB-DGs, we found empirically that the *counts* in the theoretical s^* are unique. As such, an empirically optimized prompt for CIB-DG experiments on neural predictors is deemed correct if it has the same counts (Equation (2)) as the theoretically optimal prompt. For other DGs (bandit and switching DGs), correctness requires an exact match between $\hat{s}$ and any one of the theoretical s^*'s.

To estimate the proportion correct, we trained a large number of networks with different random seeds, found the empirically optimal prompt $\hat{s}$ for each network using $\mathcal{D}_N$ drawn with another different seed, and then calculated the empirical ratio of correct $\hat{s}$'s out of all prompts found on all networks.

A.5.1 Limitations.

This definition of optimal prompt uses the Bayes optimal predictor, not the trained neural predictor. In practice, neural predictors introduce further complications when used as a reference model to define optimality, because they can vary depending on many pretraining hyper-parameters (e.g., neural architecture, learning rate, dataset size, any post-training, etc.). As such, given a specific neural predictor p_θ, the optimal prompt for a task DG (evaluated under infinitely large dataset) $s^*_{1:L}(p_\theta, q)$ (similarly defined in (5)) may differ from the optimal prompt $s^*_{1:L}(p_\mathrm{B}, q)$ under the Bayes predictor, and also from other neural predictor instances.

Another consequence of using a model-dependent optimality is that the optimal log-loss under a neural predictor can be lower than that of the Bayes predictor

$$\max_s \mathbb{E}_{q(x)}[\log p_\theta(x|s)] > \max_s \mathbb{E}_{q(x)}[\log p_\mathrm{B}(x|s)].$$

This notion of practical optimality thus depends on the network instance, with nuances beyond the scope of the Bayesian meta-learning perspective in this work. Future work that takes a more architectural perspective should investigate this important issue further.

A.6 The Loss "Landscape" of Bayes Predictor

Why is it difficult to identify the optimal prompt with a finite task dataset? We hypothesize that this is because there are many suboptimal prompts that are only slightly worse compared to the optimal one in terms of the expected log-loss (4). To test this, we take the Bayes predictor for each prompt setup, compute expected log-loss given all possible prompts with length $L \leq L_{\max}$. We also subtract this log-loss with the best possible log-loss, giving the Kullback–Leibler divergence

$$\mathrm{KL}[q(\cdot)\|p(\cdot|s_{1:L})] := \mathcal{L}(p_{(\cdot)}, q, s_{1:L}) - \mathcal{L}(q, q, s_{1:L}). \tag{10}$$

If many prompts yield KL divergences close to the optimal KL divergence (under s^*), then it is less likely that the order of these prompts is still preserved when evaluated under a finite dataset, and even less so on approximate neural predictors.

Since the prompts are discrete with no obvious order, we show the KL divergences of all prompts, sorted in increasing order. Each prompt is then associated with a rank. To see if the optimal prompt has a distinctively smaller KL divergence than other prompts, we plot the KL divergences against the rank of the prompt. For CIB-DGs, the prompts expressed in the counts $(S_0(s_{1:L}), S_1(s_{1:L}))$ are ordered, so we show the prompt rank by their counts, and also plot the loss "landscape": the KL divergence as the color of each dot representing the counts. We show these results for each of the four CIB-DG experiments below.

Limitations The "loss-landscape" is a loosely defined term to describe the visualization method for understanding the reliability of finding the optimal prompt. Importantly, it is a *post hoc* technique that cannot be used to *predict* the reliability given a predictor and a task DG, before running any evaluations. The ability to predict is more challenging due to the nonlinear log-loss function and discrete domain. For example, the problem of integer quadratic programming is NP-complete (Vavasis, 1990), let alone predicting its reliability under a noisy dataset.

B ADDITIONAL EXPERIMENTAL RESULTS

For each of pretraining CIB-DGs and the random switching DG, we trained 30 networks with different random seeds for the weight initialization and minibatch sampling. For the bandit DG, we trained 100 networks. Below, we report, for each pretraining DG, the estimated KL divergence between p_x and p_θ using samples from p_x:

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\log\left(\frac{p_{\mathrm{B}}(x_t|x_{1:t-1})}{p_\theta(x_t|x_{1:t-1})}\right). \tag{11}$$

The training sequence length and minibatch size for each pretraining DG p are as shown in Table 2. In most cases, the network achieved near-zero KL divergence, consistent with previous findings (Mikulik et al., 2020; Genewein et al., 2023; Grau-Moya et al., 2024). Under the bandit DG, we use an approximate Bayes predictor described in Section D.2. We see that the LSTM and Transformer outperform this predictor by a small but statistically significant amount.

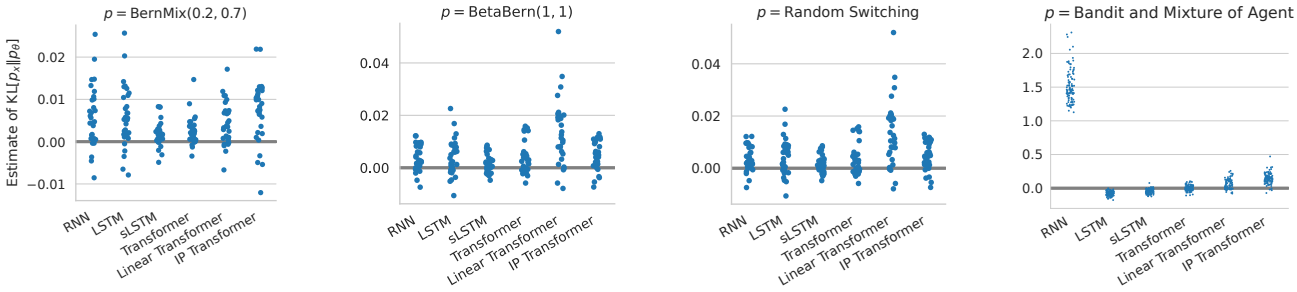

Figure 6: Final pretraining loss under various DGs p and neural predictors. Each dot is an Monte Carlo estimate of the KL divergence through Equation (11) in the last training step, summed (not averaged) over entire training sequences (180 time steps for CIB-DGs and 630 for Bandit experiments; see Table 2) , then averaged over a random training minibatch.

B.1 Computational Scale and Resources

All experiments were implemented using JAX (Apache 2.0) (Bradbury et al., 2018; DeepMind et al., 2020), NumPy (modified BSD, Harris et al., 2020) and Haiku (Apache 2.0, Hennigan et al., 2020). We used in-house computational infrastructure comprising P100 and V100 NVIDIA GPUs, TPUv2, and TPUv3. In total, we pretrained 3 (2 CIB-DGs + random switching) $\times$ 6 (architectures) $\times$ 30 (seeds) + 1 (Bandit DG) $\times$ 6 (architectures) $\times$ 100 (seeds) = 1140 neural predictors. Each pretraining run takes from several hours to 1 week, with the longest training being on RNN networks to get a low log-loss.

Searching for the theoretically optimal s^*. This required minimal (<10 seconds) on the Bayes predictors of CIB-DGs, as we only need to enumerate over the counts of ZEROs and ONEs. For the switching task, we searched through all possible prompt sequences of up to length $L_{\max}=15$, and evaluated the expectation Equation (4) over all possible sequences of length $T\in\{10,30\}$. For $T=30$, each run took less than 3 days. For the bandit task, we estimated the log-loss using 10^5 rollouts for all prompts of length 16, and confirmed the best out of the top 20 prompts using 10 million rollouts.

Searching for the empirically optimal prompt. For Bayes predictors, on CIB-DG experiments, we searched for the best prompt 2 (pretraining DGs) $\times$ 2 (task DGs) $\times$ 3 ($L_{\max}\in\{5,10,15\}$) $\times$ 5 ($T\in\{1,3,10,30,100\}$) $\times$ 4 ($N\in\{10,10^2,10^3,10^4\}$) $\times$ 1000 (seeds for $\mathcal{D}_N$) = 240000 times. These are carried out fast over the counts, so we do not count them as a significant computation. On the switching task, we searched 4 (task DGs) $\times$ 1 ($L_{\max}\in\{15\}$) $\times$ 2 ($T\in\{10,30\}$) $\times$ 4 ($N\in\{10,10^2,10^3,10^4\}$) $\times$ 250 (seeds for $\mathcal{D}_N$) = 8000 times over binary sequences. On the bandit task, we searched 5 ($T\in\{1,3,10,30,100\}$) $\times$ 5 ($N\in\{10,10^2,10^3,10^4,10^5\}$) $\times$ 100 (seeds for $\mathcal{D}_N$) = 2500 times. Each of these took less than a day.

For neural predictors, on CIB-DG experiments, this amounts to 360 (model instances) $\times$ 3 ($L_{\max}\in\{5,10,15\}$) $\times$ 5 ($T\in\{1,3,10,30,100\}$) $\times$ 4 ($N\in\{10,10^2,10^3,10^4\}$)= 21600 exhaustive searches on binary sequences (not

counts), using a different seed to draw $\mathcal{D}_N$ for each network instance. On the switching experiment, this is 180 (model instances) $\times$ 1 ($L_{\max} \in \{15\}$) $\times$ 2 ($T \in \{10, 30\}$) $\times$ 4 ($N \in \{10, 10^2, 10^3, 10^4\}$)= 1440 exhaustive searches on binary sequences. Each of these runs took less than 3 days. On the bandit tasks, this is 600 (model instances) $\times$ 1 ($L_{\max} \in \{8\}$) $\times$ 5 ($T \in \{5, 15, 50, 150, 300\}$) $\times$ 5 ($N \in \{10, 10^2, 10^3, 10^4, 10^5\}$)= 15000 exhaustive searches on binary sequences. Thus, our results are derived from $>$ 48000 optimal prompts sequences, all of which are numerically guaranteed to be optimal under the prompt setup. For RNNs and transformers that can be cast as RNNs, this took less than a day. For the softmax transformer, without using any optimized attention computation, each run took around 2 weeks. We estimate the compute used during the entire project to be around 30 times the compute used to generate the final results.

B.2 Pretraining on BernMix(0.2, 0.7), Prompting towards Bern(0.7) (IMD)

B.2.1 Proportion Correct

Figure 7 shows the proportion correct results for the task DG $q = \text{Bern}(0.7)$, so $q \in \mathcal{M}_p$. The recurrent networks show similar behaviors to the Bayes predictor, and the Transformers have worse proportion correct at $T = 100$ even for a short prompt with length up to $L_{\max}$. The proportion correct is lower for longer prompts for all predictors, likely because the prompt of all ONEs requires a perfect match on each token and is more difficult to identify exactly when the maximum prompt length is longer.

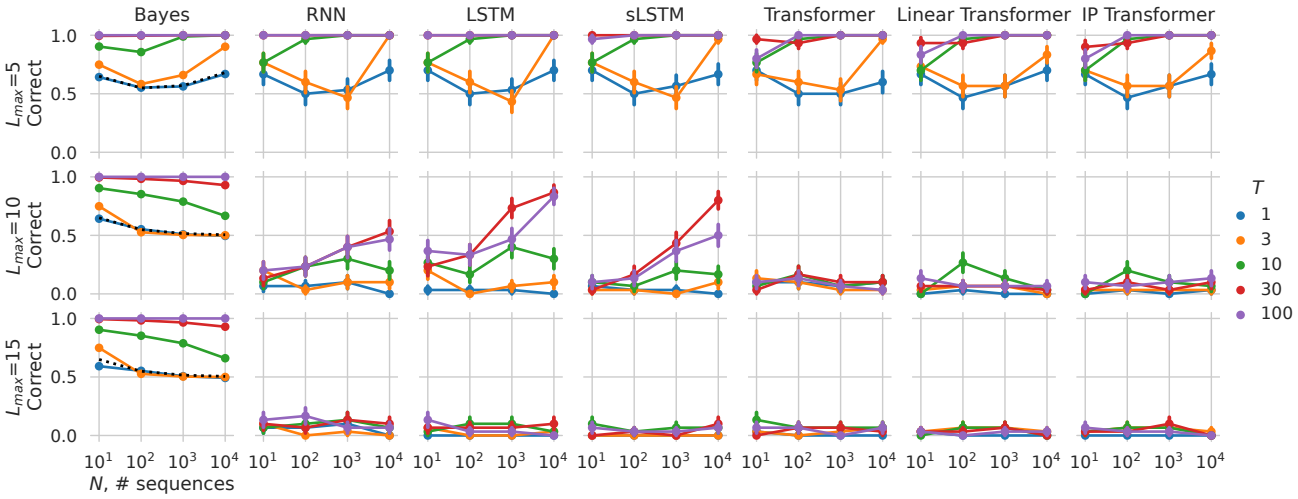

Figure 7: The proportion correct of the empirically optimal prompt for $p = \text{BernMix}(0.2, 0.7)$ and $q = \text{Bern}(0.7)$. Same as Figure 2 but with more prompting setups.

Table 3: Prompts of length up to $L_{\max} = 5$ that induce top 10 values of the posterior belief $p(x_1|s)$ under the pretraining DG BernMix(0.2, 0.7).

	Prompt counts										
$S_0(s)$	0	0	1	0	1	0	2	1	0	2	1
$S_1(s)$	5	4	4	3	3	2	3	2	1	2	1
$p(x_1 = 1\|s)$	0.699	0.697	0.691	0.689	0.671	0.662	0.629	0.611	0.589	0.516	0.484

B.2.2 Non-Monotonicity of Proportion Correct versus N

Let us gain some insights into this phenomenon by considering the case of $T = 1$. Here, we can compute theoretically the probability of an empirical prompt being correct and predict this trend as the black dotted line. The short reason is that the task dataset may have an *empirical* ratio of ONEs below 0.7, which happens with nonzero probability even though the task DG is Bern(0.7). When this happens, this dataset may be better explained by a mixture of Bernoulli that has nonzero weight on the component with $\tau_1 = 0.2$, giving a prediction that is close to the empirical ratio. Such a mixing weight can be induced by a prompt other than the all-ONE

prompt; see Table 3 for example prompts with lengths up to $L_{\max} = 5$. Note that some of the prompt counts in Table 3 appear in the clusters shown in Figure 2(right, blue dots). At $N = 10$, if the empirical mean in the task dataset takes on values of 0.6 or 0.5, the best prompts to induce such biases have counts, respectively, (1 ZERO & 2 ONEs) or (1 ZERO & 1 ONE). Such datasets are responsible for the clusters of empirical prompts.

More precisely, for $T = 1$ we can compute the probability that $\hat{s}(p_{\mathrm{B}}, \mathcal{D}_N) = s^*(p_{\mathrm{B}}, q)$ given a randomly drawn task dataset $\mathcal{D}_N$ from $q = \mathrm{Bern}(0.7)$. Recall that the log-loss on the first token following a prompt s is

$$\begin{aligned} &\hat{\mathcal{L}}(p_{\mathrm{B}}, \mathcal{D}_N, s) = -N\hat{\tau}(\mathcal{D}_N)\log(\bar{\tau}(s)) - N(1-\hat{\tau}(\mathcal{D}_N)\log(1-\bar{\tau}(s))\,, \text{ where} \\ &\hat{\tau}(\mathcal{D}_N) := \frac{1}{N}\sum_{n=1}^{N} x_1^n\,, \quad \bar{\tau}(s) := p_{\mathrm{B}}(x_1 = 1|s) = (1 - w_L(s))\tau_1 + w_L(s)\tau_2\,, \end{aligned} \tag{12}$$

and $w_L(s)$ is given in Theorem 3.2. For prompts of length up to $L_{\max}$, it is easy to see that $\hat{s}(p_{\mathrm{B}}, \mathcal{D}_N) = s^*(p_{\mathrm{B}}, q)$ if and only if, under the Bayes predictor p_{B}, the optimal prompt s^* with $L_{\max}$ ONEs gives a lower log-loss than the prompt s^+ that has 0 ZERO and $(L_{\max} - 1)$ ONEs (see Table 3 for an example when $L_{\max} = 5$). Then we have,

$$\mathbb{P}(\hat{s} = s^*) = \mathbb{P}(\hat{\mathcal{L}}(p_{\mathrm{B}}, \mathcal{D}_N, s^*) < \hat{\mathcal{L}}(p_{\mathrm{B}}, \mathcal{D}_N, s^+)).$$

Substituting in Equation (12) and after some manipulation, we get

$$\mathbb{P}(\hat{s} = s^*) = \mathbb{P}\left(N\hat{\tau}(\mathcal{D}_N) > \kappa\right)\,, \text{ where } \kappa := N\log\left(\frac{1-\tau^+}{1-\tau^*}\right) \Big/ \log\left(\frac{\tau^*(1-\tau^+)}{\tau^+(1-\tau^*)}\right)$$

Noting that $N\hat{\tau}(\mathcal{D}_N)$ is a binomial distribution $\mathrm{Binom}(0.7, N)$, we can easily find $N\hat{\tau}(\mathcal{D}_N)$ using its cumulative distribution function. For sequence length $T > 1$, the loss becomes more complicated, so is its dependence on $\bar{\tau}$.

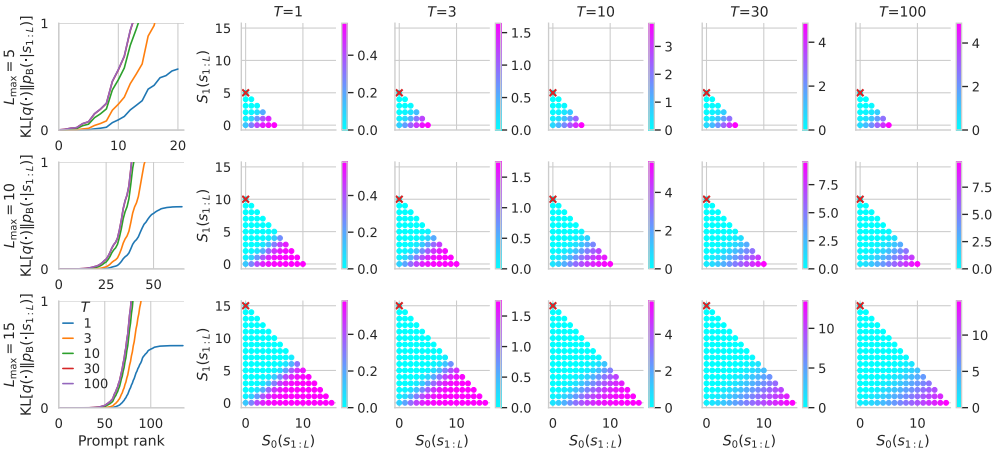

Figure 8: The loss "landscape" of $p = \mathrm{BernMix}(0.2, 0.7)$ and $q = \mathrm{Bern}(0.7)$. The leftmost column shows the KL divergence of each prompt sorted in increasing order against the prompt rank (sort indices, or argsort), with lower rank meaning lower KL divergence (10). The other columns show KL divergence of each individual prompt. The prompts here are all expressed by their counts. See Section A.6 for a detailed explanation.

B.2.3 Loss "Landscape"

We show the KL divergence of all possible prompts in two ways in Figure 8. On the leftmost column, we order the prompts (expressed in counts) according to their values of the KL divergence, producing a rank plot of in ascending order. The best prompt is at rank 0, the second best prompt is at rank 1, etc. We see that there is a relatively flat region next to the best prompt with *rank*0, suggesting that it would be more difficult to distinguish the best few prompts. As $L_{\max}$ increases and T decreases, the flat region expands towards the right, meaning that there are more close-to-optimal prompts. This trend is consistent with the results shown in Figure 7. The other columns of Figure 8 show KL divergence of all possible prompts with length less than $L_{\max}$. It is evident that the many prompts around the optimal prompt have very similar KL divergences close to zero.

B.2.4 Distribution of Empirically Optimal Prompts

Figure 9 shows the distribution of empirically optimal prompts for different neural predictors. We show the setting of $N = 10\,000$ and $T = 100$, which is the most reliable setting. At the shortest $L_{\max} = 5$, the empirical $\hat{s}$ is mostly correct. At larger $L_{\max}$, the empirical $\hat{s}$ gets further away from s^* with an inconsistent ratio of ONEs, leading to unreliable identification that hurts the interpretability of the scattered empirical $\hat{s}$'s.

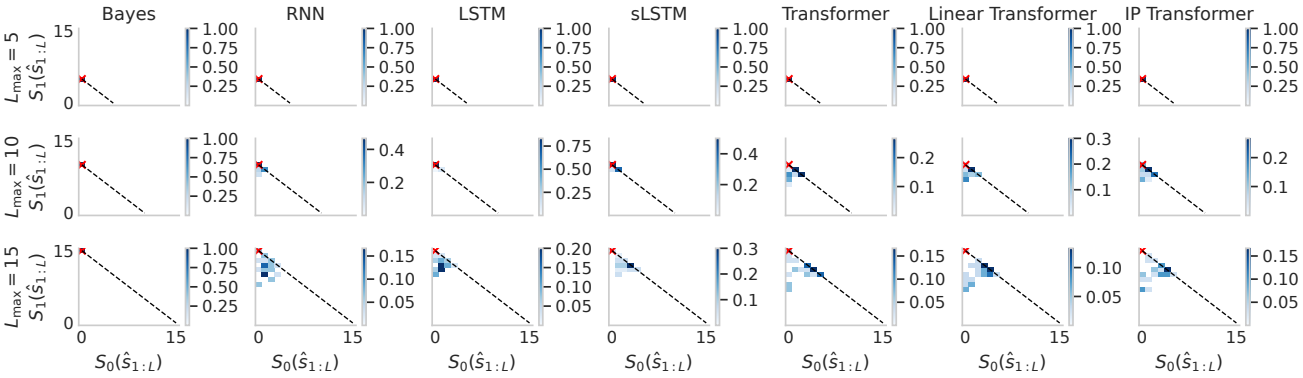

Figure 9: The distribution of the prompt counts for $p = \text{BernMix}(0.2, 0.7)$ and $q = \text{Bern}(0.7)$ for different neural predictors. We pick the most reliable prompt setup of $N = 10\,000$ and $T = 100$. Red cross shows the theoretical s^*. The black dashed line shows the boundary set by $L_{\max}$.

B.3 Pretraining on BernMix(0.2, 0.7), Prompting towards Bern(0.6) (OOMD)

B.3.1 Zig-Zag Pattern of Theoretically Optimal Prompts

We provide an intuitive explanation for the zig-zag pattern of the theoretical optimal prompts in Figure 3. Suppose that we only predict a single x_1 for $T = 1$, then the best prompt should be one such that w_L in (3) is as close as possible to 0.6. The counts of ZEROs and ONEs in the theoretically optimal prompt for different values of $L_{\max}$ are as follows: 1 ZERO and 2 ONEs for $3 \le L_{\max} \le 9$, or a prompt of 5 ZEROs and 5 ONEs when $10 \le L_{\max} \le 34$, or 16 ZEROs and 19 ONEs when $L_{\max} \le 35$, etc.; see Figure 10. This explains why the optimal prompt does not take maximal length for a fixed T. Interestingly, the ratio of ONEs does not converge to 0.6 for up to $L_{\max} = 10000$.

As T increases, the sequence $x_{1:T}$ continues to update w_L, and will eventually, with high probability, converge to 1.0, giving a constant prediction of $p(x_t = 1) = 0.7$ for large T. The prompt can only improve this prediction in the short-term. At $L_{\max} = 15$, a prompt longer than the optimal length 10 provides more robust predictive ratio (not easily modified by incoming x), but this sacrifices the predictability of the first few tokens, which could be better predicted using the 5 ONEs and 5 ZEROs optimal for $T = 1$. This creates a highly complex interplay between T and the optimal prompt length.

B.3.2 Unexpected Length Increase in Theoretically Optimal Prompts

Figure 3 shows the optimal prompt for a fixed $L_{\max}$, which has unexpected variations in the optimal length as a function of sequence length. In Figure 10, we show how the optimal prompt length depends on the upper limit $L_{\max}$ for different sequence lengths T. We see that the optimal length never decreases, but can jump to $L_{\max}$ or stay unchanged as $L_{\max}$ increases, again with almost unpredictable pattern. This can result in an illusion that the optimal prompt length is short even though longer prompts have been searched, since theoretically the optimal prompt length does not always increase even if longer prompts are allowed.

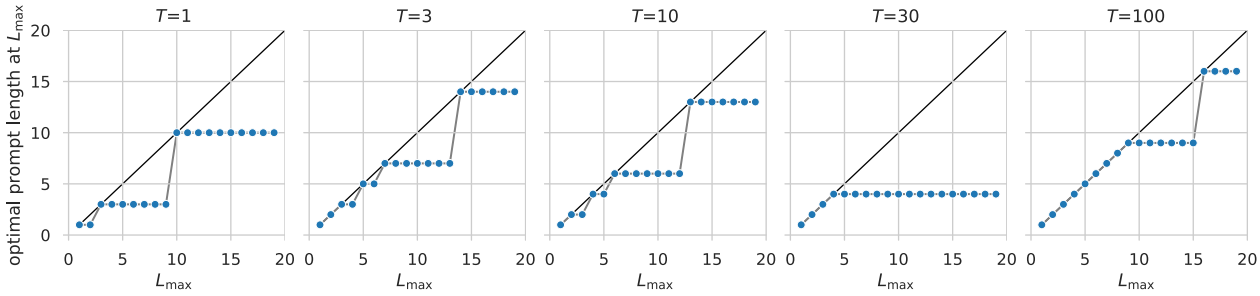

Figure 10: Optimal prompt length as a function of the maximum prompt length $L_{\max}$.

B.3.3 Proportion Correct

Figure 11 extends the results in Figure 2. For $L_{\max} = 5$, the Transformer predictors have lower proportion correct than other predictors when $T = 100$. At $L_{\max} = 10$ and $L_{\max} = 15$, all neural predictors got worse than the Bayes predictor when T reaches 30; and all predictors, including the Bayes predictor, failed to identify s^* when $T = 100$.

B.3.4 Distribution of Empirically Optimal Prompts

Figure 12 shows the distribution of empirically optimal prompts for different neural predictors. When the maximum prompt length $L_{\max} = 5$, the empirical $\hat{s}$'s are mostly correct. At larger $L_{\max}$, some empirical $\hat{s}$ are close to the s^*, but there is a cluster of empirical $\hat{s}$'s at around (5 ZEROs and 10 ONEs) for $L_{\max} = 15$, which is quite far away from the theoretical s^*. In this case, even knowing the pretraining distribution does not explain the existence of this cluster, as the Bayes predictor only has a cluster at the all-ONE prompt.

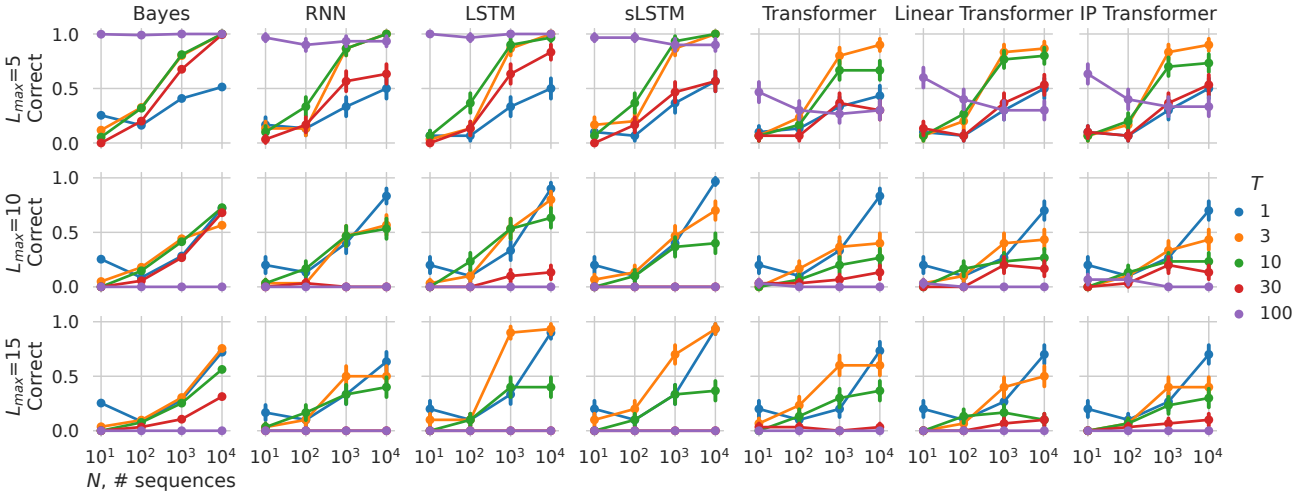

Figure 11: The proportion correct of empirically optimal prompt for $p = \text{BernMix}(0.2, 0.7)$ and $q = \text{Bern}(0.6)$. Same as Figure 3 but with more prompting setups.

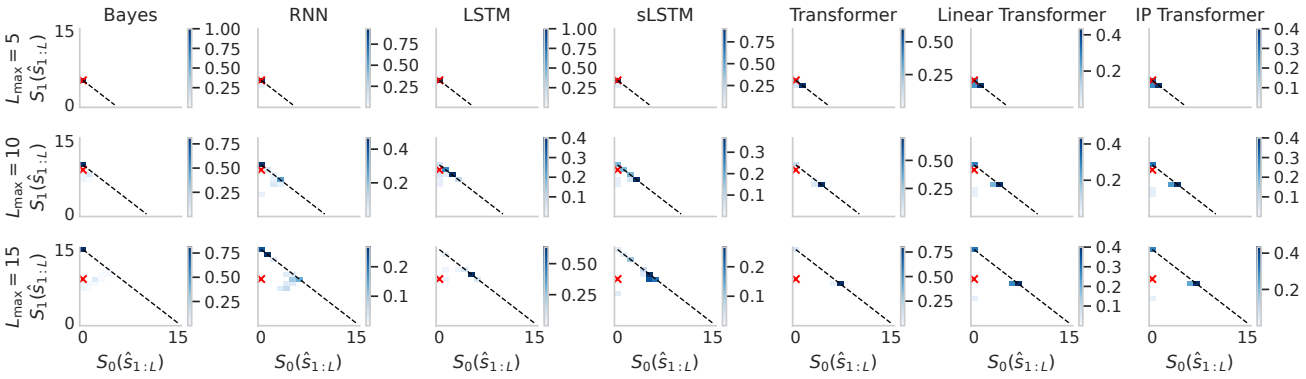

Figure 12: The distribution of the prompt counts for $p = \text{BernMix}(0.2, 0.7)$ and $q = \text{Bern}(0.6)$ for different neural predictors. We pick the most reliable prompt setup of $N = 10\,000$ and $T = 100$. Red cross shows the theoretical s^*. The black dashed line shows the boundary set by $L_{\max}$.

B.3.5 Loss "Landscape"

Figure 13 shows the "loss landscape" of prompts on the Bayes predictor. In addition to the flat landscape similar to Figure 8, the KL divergence does not go towards zero, which is a sign of OOMD prompting. Thus, the optimal prompts are harder to identify, and the best prompt does not improve the performance much.

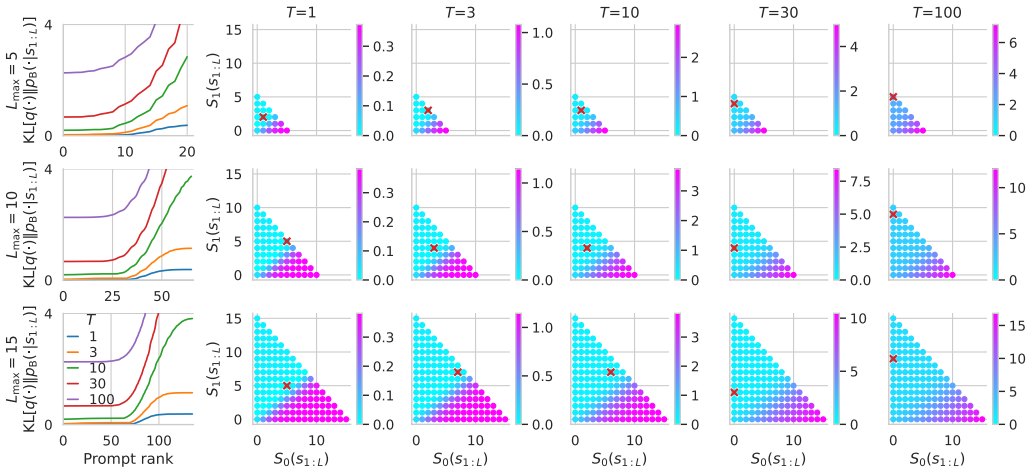

Figure 13: The loss "landscape" of $p = \mathrm{BernMix}(0.2, 0.7)$ and $q = \mathrm{Bern}(0.6)$.The leftmost column shows the KL divergence of each prompt sorted in increasing order against the prompt rank (sort indices, or argsort), with lower rank meaning lower KL divergence (10). The other columns show KL divergence of each individual prompt. The prompts here are all expressed by their counts. See Section A.6 for a detailed explanation.

B.4 Pretraining on BetaBern, Prompting towards Bern (IMD)

B.4.1 Proportion Correct

Figure 14 extends the results in Figure 4. In almost all prompt setups, all neural predictors show very similar trends compared to the Bayes predictor.

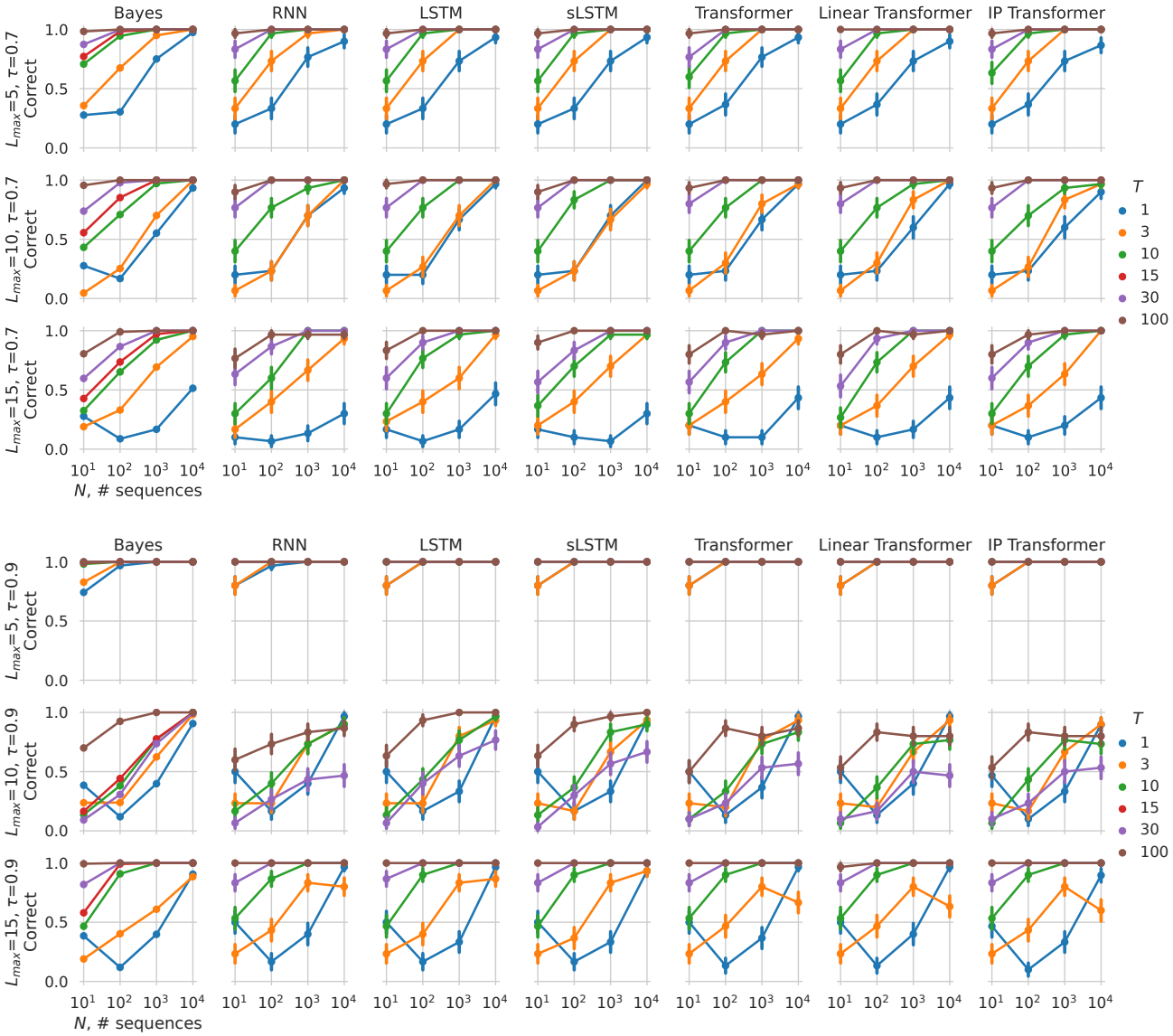

Figure 14: Proportion correct of $\hat{s}$. Same as Figure 4 but with more prompting setups.

B.4.2 Distribution of Empirically Optimal Prompts

As Figure 15 shows, the empirically optimal prompts are highly likely correct, suggesting that the distribution is very concentrated at s^*. To show how the empirical $\hat{s}$ differs, we check the distribution of $\hat{s}$ for the setting $q = \text{Bern}(0.7)$, $N = 100$ and $T = 3$ in Figure 15. Unlike in previous cases, the prompt distribution is still very close to s^*, giving more or less the same ratio of ONEs and hence consistent interpretation.

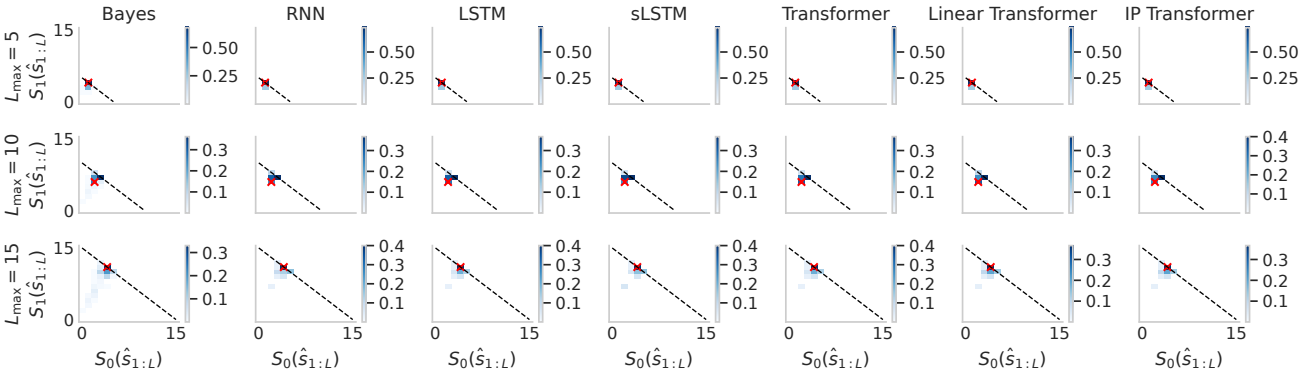

Figure 15: The distribution of empirically optimal prompt for $p = \text{BetaBern}(1, 1)$, $q = \text{Bern}(0.7)$, $N = 100$, $T = 3$ for different neural predictors. Red cross shows the theoretical s^*. The black dashed line shows the boundary set by $L_{\max}$.

Table 4: Example optimal prompts $s^*_{L_{\max}}$ for $p = \text{BetaBern}(1, \beta)$ and $q = \text{Bern}(0.7)$ of various values of β, T and $L_{\max}$.

SDs p and q			Optimal prompt $s^*_{L_{\max}}$		
β	T	$L_{\max}$	(S_0, S_1)	$S_0 + S_1$	$S_1/(S_0 + S_1)$
1	1	3	(0, 1)	1	1.00
1	3	3	(0, 2)	2	1.00
1	5	3	(1, 2)	3	0.67
1	1	10	(2, 6)	8	0.75
1	10	10	(3, 7)	10	0.70
1	3	12	(3, 8)	11	0.72
1	5	12	(3, 8)	11	0.72
2	10	10	(2, 8)	10	0.8
2	100	20	(5, 15)	20	0.75
2	100	50	(14, 36)	50	0.72

B.4.3 Effect of the Prior at Shorter Sequence Length

The mismatch at shorter T is due to the uniform prior $\text{Beta}(1, 1)$ having pseudocounts of 1 ONE and 1 ZERO, which regularizes the predicted bias towards 0.5 by adding 1 to each counter. For shorter T, say $T = 1$, it is important to get the ratio correct after this regularization, since the Bayes optimal predictor gives $p(x_1 = 1|s) = (S_1(s) + 1)/(S_0(s) + S_1(s) + 2)$. If we add 1 to each count, then the regularized ratio is closer to the true bias. To reduce the gap between the regularized ratio, the optimal prompt length under the upper limit $L_{\max}$ may be less than $L_{\max}$. For example, for $L_{\max} = 10$ and $q = \text{Bern}(0.7)$, the optimal prompt is 2 ZEROs and 6 ONEs and has length 8 (see Table 4); the regularized ratio is then $(6 + 1)/(2 + 6 + 2) = 0.7$, which is optimal for predicting the next x_1.

However, as more tokens arrive, the regularized ratio fluctuates due to the randomness in the draws from Bern(0.7), while ideally it should be fixed at 0.7. Hence, there is additional benefit from being certain about the bias (with larger counts), which can be gained by using more tokens in the prompt. For the same example of $L_{\max} = 10$ and $q = \text{Bern}(0.7)$, but now $T = 10$, the optimal prompt becomes 3 ZEROs and 7 ONEs Table 4. There will be a small cost for predicting earlier x_t's, as the regularized ratio is initially not exactly 0.7.

B.4.4 Non-Uniform Latent Factor

In Table 4, we show the optimal prompts on a few prompt setups. When $\beta = 1$ and p_τ is uniform, the optimal prompt contains roughly the correct proportion of ZEROs and ONEs even for a short task sequence length T, converging to the true bias in q as $L_{\max}$ increases. When $\beta = 2$ and thus p_τ is biased towards zero, the optimal prompts have to debias the prior, and thus the empirical ratio of ONEs is further away from the ground truth bias 0.7, and requires larger T and $L_{\max}$ to converge to 0.7.

B.4.5 Loss "Landscape"

Figure 16(left) shows that the theoretical optimal prompt has a distinctively lower KL divergence compared to other prompts, with the exception that $T = 1$ still has a very flat landscape. The other columns show a clearer optimal region, especially for $\tau = 0.7$. These are in stark contrast to Figures 8 and 13 where the optimal points do not standout among other suboptimal prompts.

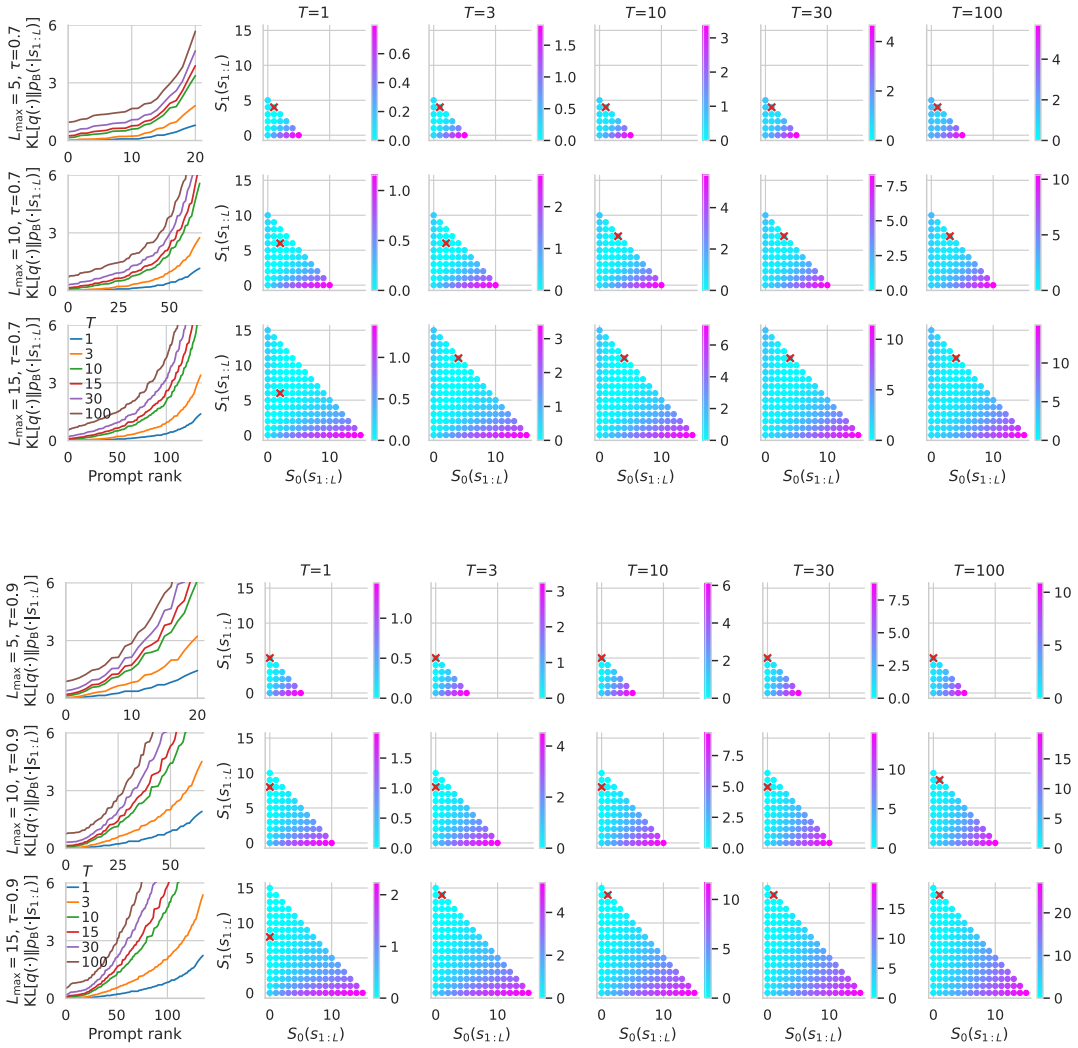

Figure 16: The loss "landscape" of $p = \mathrm{BetaBern}(1, 1)$ and $q = \mathrm{Bern}(0.7)$ or $q = \mathrm{Bern}(0.9)$.The leftmost column shows the KL divergence of each prompt sorted in increasing order against the prompt rank (sort indices, or argsort), with lower rank meaning lower KL divergence (10). The other columns show KL divergence of each individual prompt. The prompts here are all expressed by their counts. See Section A.6 for a detailed explanation.

B.5 Pretraining on BetaBern, Prompting towards BernMix (OOMD)

$q = \text{BernMix}(\tau_1, \tau_2)$				s^*_{100}	
τ_1	τ_2	$\Delta\tau$	$\bar{\tau}$	(S_0, S_1)	ratio
1/2	1/3	1/6	0.42	(20, 14)	0.41
2/5	3/5	1/5	0.50	(11, 11)	0.50
1/5	2/5	1/5	0.30	(13, 5)	0.28
1/4	1/2	1/4	0.38	(9, 5)	0.36
1/3	2/3	1/3	0.50	(3, 3)	0.50
1/4	3/4	1/2	0.50	(1, 1)	0.50
1/5	4/5	3/5	0.50	(0, 0)	-
2/5	1	3/5	0.70	(0, 1)	0.70

Table 5: Empirically optimal prompts for $p = \text{BetaBern}(1, 1)$ and several task DGs $q = \text{BernMix}(\tau_1, \tau_2)$. For each case, the optimal $s^*_{L_{\max}}$ is found for large $L_{\max} = 100$ and $T = 100$. $\Delta\tau := (\tau_2 - \tau_1)$, and $\bar{\tau} := (\tau_1 + \tau_2)/2$.

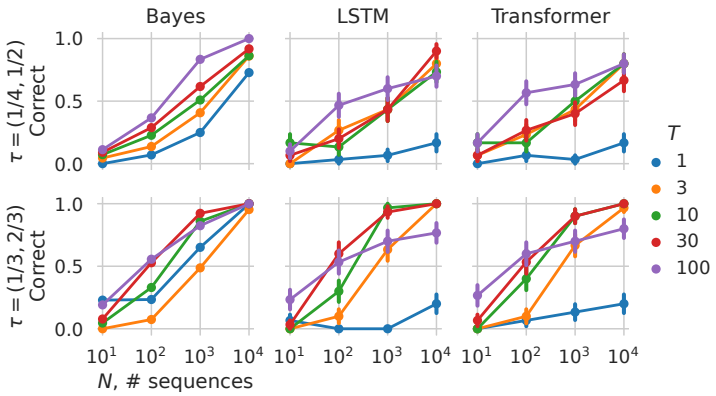

Figure 17: Proportion of empirical prompts that match the theoretically optimal prompt for two of the task DGs in Table 5. The prompt length is capped at $L_{\max} = 15$, which is sufficient for the empirically optimal prompt in both cases. Figure 18 shows additional results.

The difficulty with this mixture task q is that it is impossible to provide a single input sequence (the prompt) that leads to a bimodal $p_{\tau|x}$. Thus, while each component of the task mixture is IMD, the mixture itself is OOMD. The theoretically optimal prompts $s^*_{L_{\max}}$'s for different instances of such q under a large prompt length limit $L_{\max} = 100$ are shown in Table 5. Although the empirical ratio reflects the mean bias of q well, it is still unclear why some prompts are longer and some are shorter. Given just the information about q, it is difficult to make sense of the varying optimal prompt length.

If we know p fully, then this pattern makes more sense from a posterior concentration perspective. If τ_1 and τ_2 are close together, a helpful prompt should make the posterior concentrate loosely around these values. The closer the two values, the longer thus the optimal prompt (with a ratio of ONEs close to the mean of the two mixture components). On the other hand, if the two values are far apart, a helpful prompt leaves the posterior as broad as possible, leading to very short prompts or no prompt at all.

The empirical prompts found on neural predictors are more likely to be correct for larger T and larger N (Figure 17), similar to the in-meta-distribution case in Section 5.3. The overall trend of proportion correct also generally agrees with empirical prompts found on Bayes predictors, except for $T = 100$ for the case of $\tau = (1/3, 2/3)$. Figure 18 extends Figure 17. Across all prompt setups, the agreement between Bayes and neural predictors is worse compared to the previous IMD case shown in Figure 14.

Overall, although the optimal prompts cannot reveal the bimodality nature and the bias in each component of the task DG, they still reveal overall statistical properties. Compared to the OOMD case in Section 5.2, here we are able to make sense of the ratio of ONEs in the optimal prompts by matching them to the mean bias of q, but a detailed interpretation of the prompt length relies on knowing p.

Figure 19 shows the distribution of empirical prompts. To show how they are different to the theoretical s^*, we pick the prompt setup $N = 100$ and $T = 30$. For the least reliable case $q = \text{BernMix}(1/4, 1/2)$, the suboptimal prompts turn out to have consistent ratio of ONEs, giving consistent interpretation of the mean bias in q. This also holds for the suboptimal prompts in the case $q = \text{BernMix}(1/3, 2/3)$.

Figure 20 shows the loss "landscape". The optimal prompt produces a more distinctive optimal KL divergence compared to the two cases with τ supported on a finite mixture. In particular, for the case of $q = \text{BernMix}(1/4, 3/4)$, there is a sharp dip around the optimal prompt, which should imply more reliable identification. This is consistent with the reliability under the Bayes predictor Figure 18.

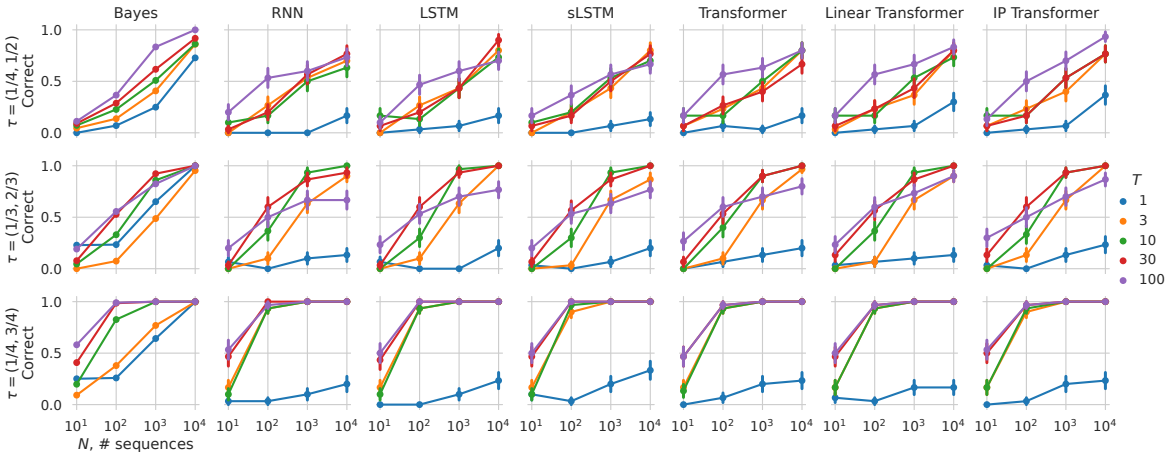

Figure 18: Proportion correct of $\hat{s}$. Same as Figure 17 but with more prompting setups, using $L_{\max} = 15$.

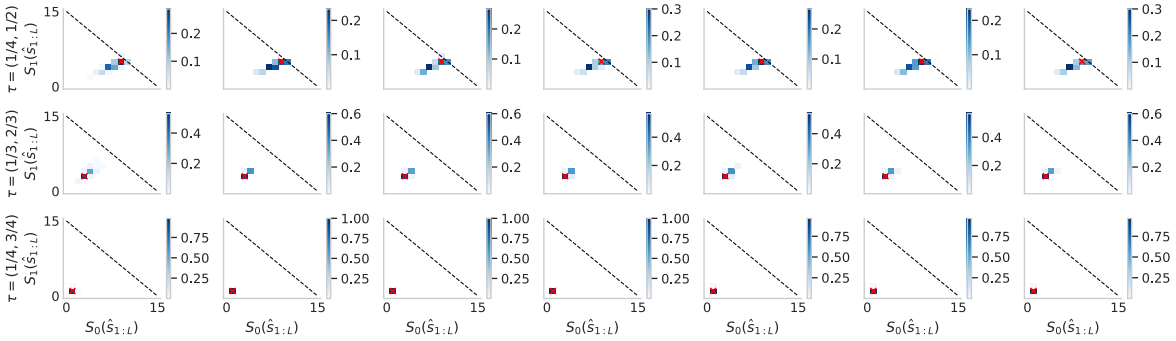

Figure 19: Distribution of empirically optimal prompts. Here, we set the $L_{\max} = 15$ for all predictors, which is greater than the length of the theoretical s^* in all cases. Red cross shows the theoretical s^*. The black dashed line shows the boundary set by $L_{\max}$.

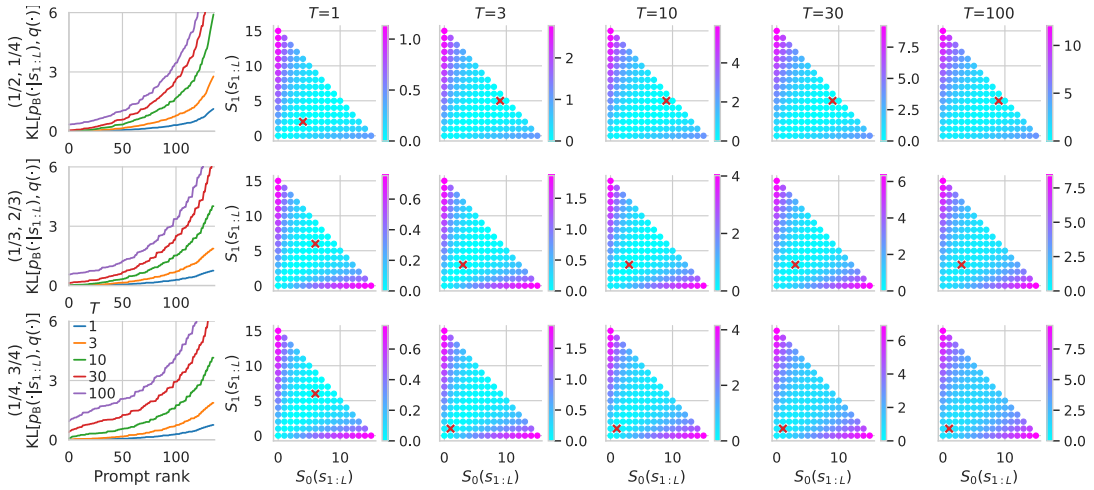

Figure 20: The loss "landscape" of $p = \text{BetaBern}(1.0, 1.0)$ and $q = \text{BernMix}(\tau_1, \tau_2)$.The leftmost column shows the KL divergence of each prompt sorted in increasing order against the prompt rank (sort indices, or argsort), with lower rank meaning lower KL divergence (10). The other columns show KL divergence of each individual prompt. The prompts here are all expressed by their counts. See Section A.6 for a detailed explanation.

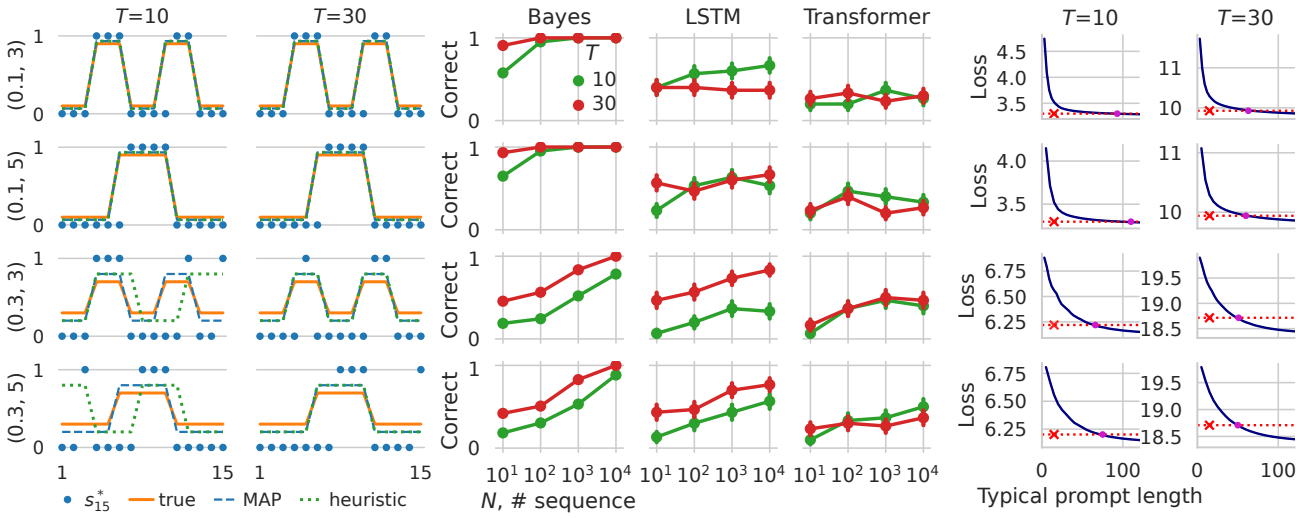

Figure 21: Results for the switching DGs. Rows show tasks with different values of (ε, λ). Left two columns: theoretical prompts s^*'s (dots), the true latent bias y (orange solid), and a heuristic estimate of the latent y based on s^*. Middle three columns: the proportion correct of $\hat{s}$; Figure 22 shows additional results. Right two columns, log-loss of typical prompts from q with increasing lengths (blue line), compared to theoretical s^* with length 15 (red).

C MORE COMPLEX DGS

We gradually move towards more complex DGs beyond conditionally independent sequences, and show that they too may produce unintuitive behaviors. Interpreting the optimal prompts is easier in the first example with switching latent factor Figure 21, but it gets more complicated for the hierarchical and topic model DGs

C.1 Switching DGs

In the first example, we use a DG that switches periodically between two coins with fixed biases, giving a non-i.i.d sequence.

Definition C.1 (Switching Process)**.** SwitchProc(ε, λ) for $\varepsilon \in [0, 1]$ and $\lambda \in \mathbb{N}^+$ generates

$$x_t \sim \text{Bernoulli}(y_t) \quad \forall t \in \{1, \dots, T\}, \quad \text{where} \quad y_{1:T} = [\underbrace{\varepsilon, \dots, \varepsilon}_{\lambda\ \varepsilon\text{'s}}, \underbrace{1-\varepsilon, \dots, 1-\varepsilon}_{\lambda\ (1-\varepsilon)\text{'s}}, \underbrace{\varepsilon, \dots, \varepsilon}_{\lambda\ \varepsilon\text{'s}}, \dots] .$$

Here, the latent factor $\tau := (\varepsilon, \lambda)$. The pretraining DG p is a finite mixture of the Switching Process:

Definition C.2 (Random Switching Process)**.** This DG generates sequences by first sampling $\varepsilon \sim \text{Uniform}([0, 1])$ and $\lambda \sim \text{Uniform}(\{3, 4, 5\})$, then $x_{1:T} \sim \text{SwitchProc}(\varepsilon, \lambda)$.

We take SwitchProc(ε, λ) with fixed values of ε and λ as a task DG q, such that $q \in \mathcal{M}_p$. Examples of τ and y are shown in Figure 21(left). Prompting here is harder; for instance, a prompt that alternates between λ ZEROs and λ ONEs is very informative of λ, but not for ε if $\varepsilon \notin \{0, 1\}$. We set $L_{\max} = 15$ and search through $s_{1:15} \in \mathcal{A}^{15}$ given data sequences with different lengths T.

Can we interpret the optimal prompts? For each of the four tasks in Figure 21(left), we plot the biases y under the true τ (orange solid), and one example theoretical optimal prompt s^* (blue dots). All the equally optimal s^*'s of each q induce the same posterior $p(\tau|x^*)$, so we show the latent y corresponding to the mode of $p(\tau|s^*_{15})$. The estimated λ's are correct for all four q's, and the estimates for ε's are close to the ground-truth in q. Thus, in this case, the true λ in q can be recovered from the theoretical s^* if we know the pretraining p and its Bayes predictor.

What if we do not know p? Figure 21(left) also shows the estimated y's from a heuristic method given *incomplete* knowledge of p (see Section C.1.1). The estimate is close to the truth y for $T = 30$, but can be wrong for $T = 10$. Thus, although the uniform p_τ seems to help interpreting the prompt, knowing p fully is essential to recover the

task τ. Figure 21(middle) shows the proportion correct of empirical $\hat{s}$. Under the Bayes predictor, increasing N and T leads to higher chances of finding s^*, more so for $T = 30$ compared to $T = 10$. This trend is weaker on the neural predictors.

Given the high cost of obtaining optimal prompts, how much do we gain compared to other prompts? Here, we compare them to statistically typical prompts (samples) drawn from q, which has expected log-loss $\mathbb{E}_{s_{1:L}\sim q}[\mathcal{L}(q, p, s_{1:L})]$. Although this is almost always higher than the log-loss (4) under the optimal prompt for the same prompt length, it is much cheaper to get *longer* typical prompts to lower the log-loss at lower computational costs. Figure 21(right) shows the log-loss of theoretical s^* at $L_{\max} = 15$ and those of typical prompts with increasing lengths. Typical prompts require 3-8 times the length of the optimal prompt to reach the same log-loss, and are thus much less efficient in terms of the number of tokens, mirroring findings on natural language prompts (Bhargava et al., 2023; Renze et al., 2024). The shorter but performant optimal prompt then has the advantage of occupying shorter context windows, which is more desirable for many applications (Chang et al., 2024).

C.1.1 Heuristic Method for Interpreting Prompts of the Switching Tasks

Given a prompt $s_{1:L}$, we want to "guess" the corresponding latent causes ε and λ. The heuristic method assumes that we know the sequences are generated from the switching process in Definition C.1, and that

1. $\varepsilon \in [0, 1]$;
2. $\lambda \in \{3, 4, 5\}$;
3. The first bias y_1 associated with s_1 may *not* be the first ε appearing in Definition C.1. In other words, the "phase" of the y is unknown. Taking $\lambda = 3$ as an example, y can start with any of the following:

$$\begin{aligned}
&[\varepsilon, \quad \varepsilon, \quad \varepsilon, \quad 1{-}\varepsilon, 1{-}\varepsilon, 1{-}\varepsilon, \ldots] \\
&[\varepsilon, \quad \varepsilon, \quad 1{-}\varepsilon, 1{-}\varepsilon, 1{-}\varepsilon, \varepsilon, \quad \ldots] \\
&[\varepsilon, \quad 1{-}\varepsilon, 1{-}\varepsilon, 1{-}\varepsilon, \varepsilon, \quad \varepsilon, \quad \ldots] \\
&[1{-}\varepsilon, 1{-}\varepsilon, 1{-}\varepsilon, \varepsilon, \quad \varepsilon, \quad \varepsilon, \quad \ldots] \\
&[1{-}\varepsilon, 1{-}\varepsilon, \varepsilon, \quad \varepsilon, \quad \varepsilon, \quad 1{-}\varepsilon, \ldots] \\
&[1{-}\varepsilon, \varepsilon, \quad \varepsilon, \quad \varepsilon, \quad 1{-}\varepsilon, 1{-}\varepsilon, \ldots]
\end{aligned} \tag{13}$$

When the phase is unknown, it makes sense as a heuristic to first find a λ to match the prompt. Take $\epsilon = 0$ so that y is binary, and enumerate all possible y's of length L with different phases (as in Equation (13)) and different values of λ. Pick the y that has the fewest mismatches (smallest Hamming distance) with the binary prompt, note the best match by y^* and the corresponding λ. Effectively, this λ produces the smallest "mismatch" between the lower bias and a ZERO token in the prompt, and between the higher bias and a ONE token in the prompt.

Given the best matching binary y^*, we estimate ε as the proportion of incorrect matches with the prompt:

$$\frac{1}{L}\sum_{i=1}^{L}[(1 - y_i^*)(s_i) + y_i^*(1 - s_i)]$$

.

C.1.2 Proportion Correct

Figure 22 extends the results of Figure 21(middle). Note that in this experiment we search through all possible binary prompts of length $L_{\max}$, and for each prompt we compute the expectation (4) by enumerating all possible sequences of length T. For the Bayes predictor this can be done quite efficiently, but for neural predictors this is still quite computationally intensive. As such, we only sweep $T \in \{10, 30\}$. We observe a robust increasing pattern only on the Bayes predictor. For the neural predictors, there is a slight increasing trend only for $\varepsilon = 0.3$.

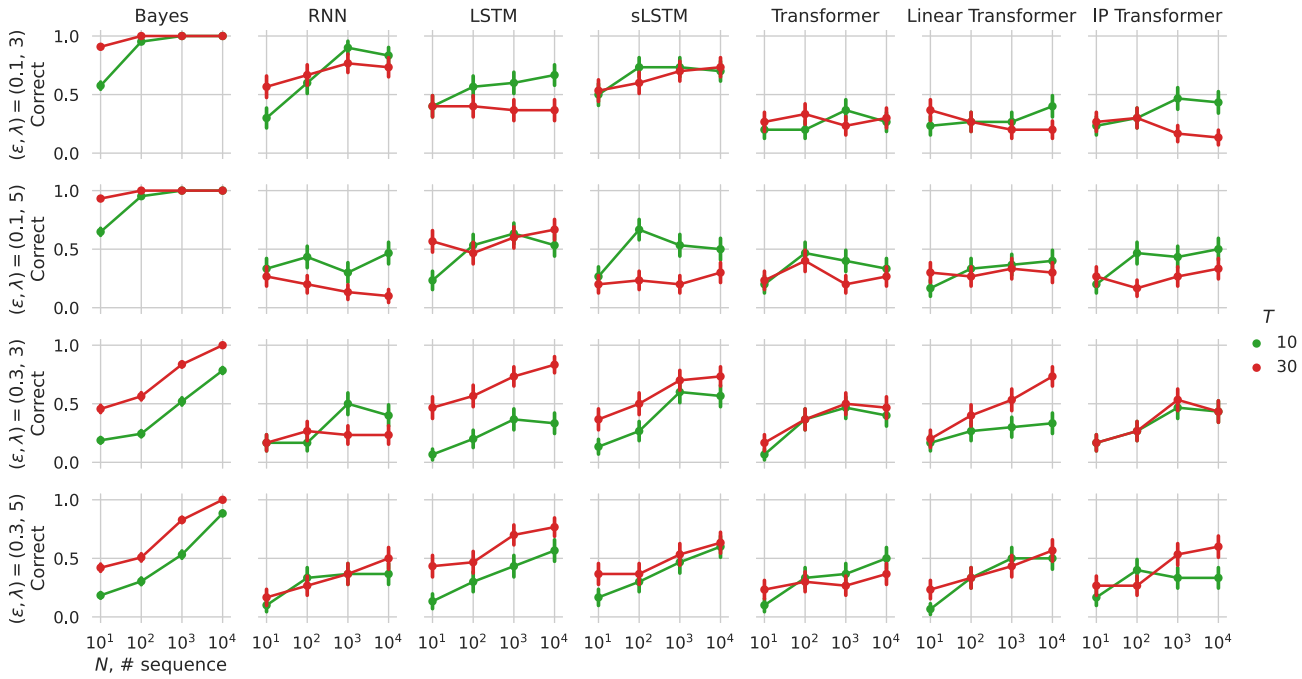

Figure 22: Same as Figure 21(middle) but with more prompting setups, using $L_{\max} = 15$.

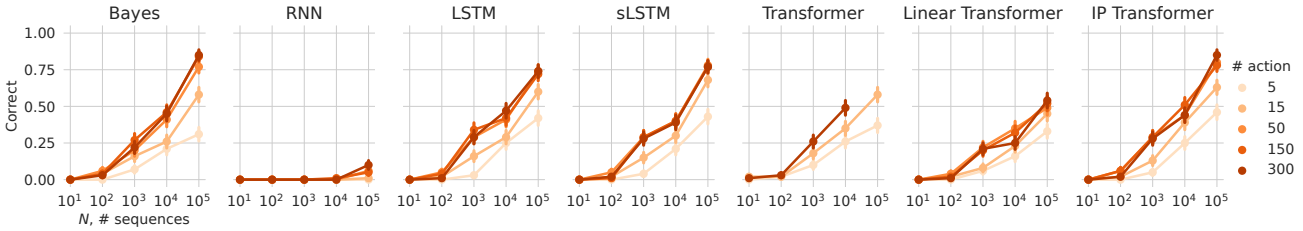

Figure 23: Same as Figure 5(lower left) but with more prompting setups, using $L_{\max} = 15$.

C.2 Hierarchical DGs

In the main text, we used CIB-DGs to demonstrate that, even in these seemingly intuitive prompt settings, the optimal prompts can show unintuitive behaviors. Natural languages exhibit more hierarchy and richer structures that break conditional independence. What would happen if we move towards more complex DGs? Here, we show that using more complex and hierarchical DGs only makes the theoretical optimal prompts even less interpretable, and thus the examples in the main text are sufficient for our purpose of being easy to understand.

Consider a CIB-DG with τ sampled from a hierarchical model

$$c \sim \text{Bernoulli}(0.5), \quad p(\tau|c) = \begin{cases} \text{Beta}(\tau; 10, 1) & \text{for } c = 1 \\ \text{Beta}(\tau; 1, 10) & \text{for } c = 0 \end{cases}, \quad p(x_t|\tau) = \text{Bernoulli}(\tau) \quad \forall t \in \{1, \dots, T\} \tag{14}$$

This simulates polarized latent factors underlying, for example, texts with extreme sentiments (e.g., Twitter US Airline Sentiment, Yelp Reviews). We use this as our pretraining DG p.

We use the familiar $q = \text{Bern}(0.7)$ as the task DG (an IMD case), and find the theoretical optimal prompts under different values of $L_{\max}$ but $T = 100$. The results are shown in Figure 24(left). The proportion of ONEs in the theoretically optimal prompt starts from 1.0 and then oscillates around 0.6, before slowly converging towards the task $\tau = 0.7$. Understanding why this arises is more difficult, because the Bayesian update involves the interaction

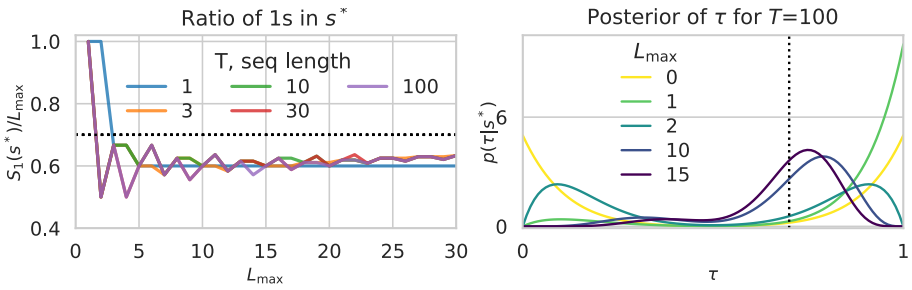

Figure 24: Results of the hierarchical DG experiments. Left, the proportion of ONEs in the theoretically optimal prompt starts from 1.0 and then oscillates around 0.6, before slowly converging towards the task $\tau = 0.7$. Right, for the case of $T = 100$, the posterior of τ given s^*. The vertical dotted line shows the true task $\tau = 0.7$

between the prompt, the posterior mixing weights between the two components, and the posteriors of the two Beta components, so we provide a qualitative explanation. In Figure 24(right), we show the posterior distribution of the latent τ given the theoretical prompt, $p(\tau|s^*)$. The goal of the prompt is to make the probability mass around the task $\tau = 0.7$ as high as possible. To do this, the optimal prompt needs to carefully balance the counts to achieve this, especially when $L_{\max}$ is short. In this regime, the Beta components are still very extreme, so the mixing weights need to be more or less around 0.5. To shift this towards an intermediate value 0.7, both components need to become less extreme, which can be achieved by prompts with equal numbers of ZEROs and ONEs. Since $\tau = 0.7 > 0.5$, we also end up having slightly more ONEs. Having more ONEs in the prompt will result in higher mixing weights for the rightmost component, biasing the posterior more towards 1.0. Due to the complexity of the problem, this explanation does not fully describe the posterior update process.

C.3 Simple Topic Model

Next, consider a further complication of the hierarchical model above, which generates a binary "document" of K "sentences". Each sentence is a binary sequence drawn from two possible "topics" (two different Beta distributions).

$$c \sim \text{Bernoulli}(0.5), \quad p(\tau_k|c) = \begin{cases} \text{Beta}(\tau_k; 2, 1) & \text{for } c = 1 \\ \text{Beta}(\tau_k; 1, 2) & \text{for } c = 0 \end{cases} \quad \forall k \in \{1, \dots, K\}. \tag{15}$$
$$p(x_{k,t}|\tau) = \text{Bernoulli}(\tau_k) \quad \forall (t, k) \in \{1, \dots, T\} \times \{1, \dots, K\}.$$

It is a simple topic model: two equally likely topics, each topic determines the probability of ONEs, and a document with multiple sentences can have multiple topics. (c.f. Latent Dirichlet Allocation (Blei et al., 2003) adds a distribution over topics for each document.)

The task DG is a document of either two or four sentences (sequence to generate) with lengths $2T$ or $4T$. For two sentences, the biases towards ONEs for the sentences are [25%, 70%]. For four sentences, the task biases are [25%, 70%, 25%, 70%]. The prompt space we optimize over is two binary "sentences" (s^1 and s^2), each with maximum length $L_{\max}$. The prompt is expressible by two pairs of counts: $[(S_0(s^1), S_1(s^1)), (S_0(s^2), S_1(s^2))]$ defined as [(#ZEROs, #ONEs) in sentence 1, (#ZEROs, #ONEs) in sentence 2].

The theoretical prompts are shown in Figure 25. The pattern is in general very complicated. For longer documents (larger T), the sentences in the theoretical prompt become shorter, with a slight bias towards more zeros. For the 4-sentence document task, the two prompt sentences occupy different regions of the prompt space, which makes sense as there is more evidence for the two topics. Other than these, explaining why the theoretical prompts vary is very challenging.

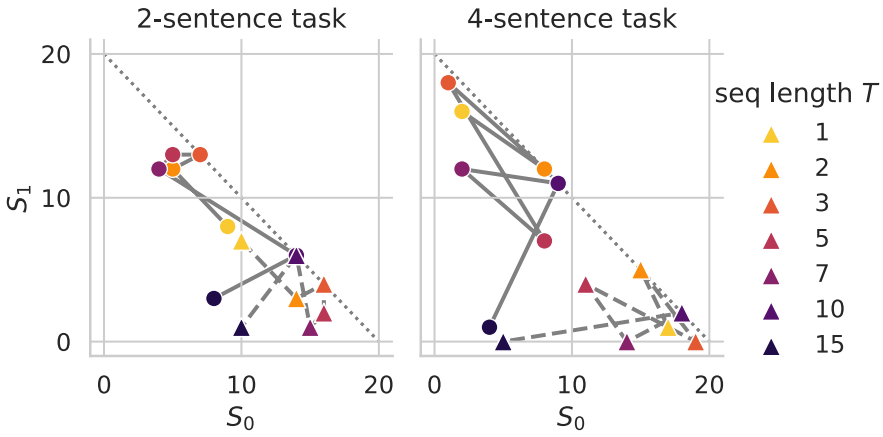

Figure 25: The theoretically optimal prompts for the topic pretraining DG. The solid line with circles represents one sentence, and the dashed line with triangles represent another sentence. The diagonal dotted line represents the maximum prompt length ($L_{\max} = 20$). The patterns are very challenging to explain.

D BANDIT DECISION-MAKER

D.1 Skill Levels

In the main text, we specified that the skill parameter τ scales the agent's counts of the outcomes. Here, we detail how different skill levels are sampled to create the mixture of agents.

To obtain agents with various skill levels, we modify the beliefs of the Thompson sampling (TS) agent by scaling the pseudocounts in the Beta posteriors of the reward probabilities by a skill level τ. For skill level $\tau \in [0, 1]$, for each arm $b \in \{\text{L}, \text{R}\}$, the posterior of the reward probability given past trajectories of $a_{1:t} \in \{\text{L}, \text{R}\}^t$ and $r_{1:t} \in \{0, 1\}^t$ is

$$v_{b,t,\tau}|a_{1:t}, r_{1:t}, \tau = \text{Beta}(1 + \tau S_{b,1,t}, 1 + \tau S_{b,0,t}), \tag{16}$$

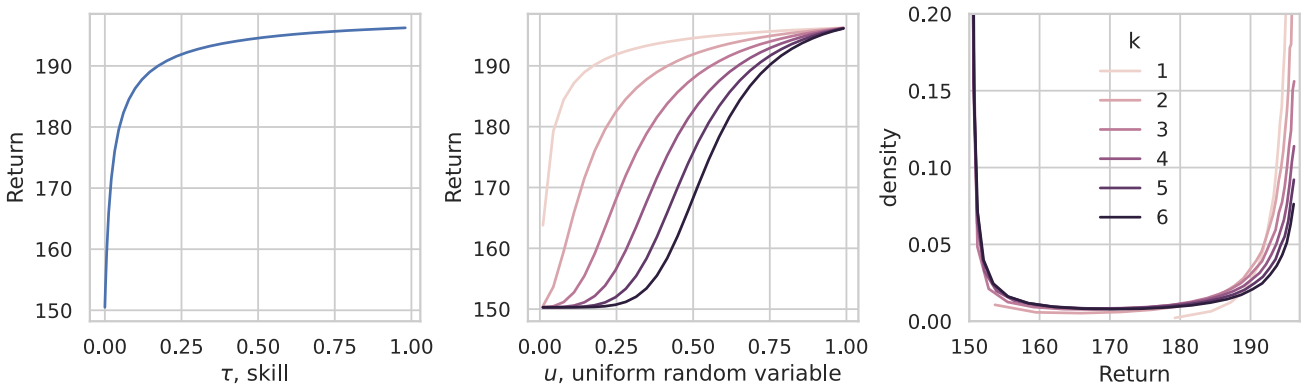

Figure 26: Relationship between the return and the uniform random variable for different values of the power index k.

where

$$
\begin{aligned}
S_{b,1,t}(a_{1:t}, r_{1:t}) &= \sum_{t'=1}^{t} \mathbb{1}[a_{t'} = b] r_{t'} \,, \\
S_{b,0,t}(a_{1:t}, r_{1:t}) &= \sum_{t'=1}^{t} \mathbb{1}[a_{t'} = b](1 - r_{t'}) \,.
\end{aligned}
\tag{17}
$$

For each action, the agent first samples the reward from the posterior, and then chooses the action of the more rewarding arm.

Figure 26(left) shows that the skill level affects the return most for lower values. As such, if we uniformly sample the skill level between 0 and 1, then there will be a lot of agents performing close to the optimal TS agent.

To avoid this, we define the skill $\tau = u^k$ where $u \sim \text{Uniform}([0, 1])$ and $k > 0$. We simulate the agent for different values of k and u for 300 actions repeated for 100k different random seeds, and show the returns as a function of u in Figure 26(middle). Through change of variable, we numerically estimate the distribution of the return for a given value of k Figure 26(right). We pick $k = 4$ throughout all bandit experiments, so that there is a mixture of agents across all levels.

There is still a significant proportion of performant agents, which should make prompting easier. We could have designed the transformation from u to τ to be more complicated to induce a more uniform return distribution, but this is not essential to demonstrate our points.

D.2 Approximate Bayes Predictor

The key quantity required for predicting the action at each time step, marginalizing over all skill levels, is to compute the probability

$$
\begin{aligned}
\mathbb{P}(a_{t+1} = \mathrm{L} | a_{1:t}, r_{1:t}) &= \int_0^1 \int_0^1 \mathbb{1}[v_\mathrm{L} > v_\mathrm{R}] p(v_\mathrm{L}, v_\mathrm{R} | a_{1:t}, r_{1:t}) \mathrm{d}v_\mathrm{L} \mathrm{d}v_\mathrm{R} \\
&= \int_0^1 \int_0^1 \int_0^1 \mathbb{1}[v_\mathrm{L} > v_\mathrm{R}] p(v_\mathrm{L}, v_\mathrm{R} | a_{1:t}, r_{1:t}, \tau) p(\tau | a_{1:t}, r_{1:t}) \mathrm{d}v_\mathrm{L} \mathrm{d}v_\mathrm{R} \mathrm{d}\tau \\
&= \int_0^1 \int_0^1 \int_0^1 \mathbb{1}[v_\mathrm{L} > v_\mathrm{R}] p(v_\mathrm{R} | a_{1:t}, r_{1:t}, \tau) p(v_\mathrm{L} | a_{1:t}, r_{1:t}, \tau) p(\tau | a_{1:t}, r_{1:t}) \mathrm{d}v_\mathrm{L} \mathrm{d}v_\mathrm{R} \mathrm{d}\tau \\
&= \int_0^1 \mathbb{P}(v_{\mathrm{L},t,\tau} > v_{\mathrm{R},t,\tau}) p(\tau | a_{1:t}, r_{1:t}) \mathrm{d}\tau,
\end{aligned}
$$

where the third equality uses conditional independence between reward probabilities given history and τ. To approximate the last integral, we discretize the support at 1000 evenly spaced grid points. For each value of τ on the grid, we now need to compute the probability that one Beta random variable is greater than another.

Algorithm 1 Pretraining Trajectory Generation for the bandit task.

Input: : $p(a_{t+1}|h_t, \tau)$, the TS-like agent that acts according to posterior reward probability samples with scaled pseudocounts (16) , given history $h_t := (a_i, r_i)_{i=1}^t$.
Input: A Bernoulli two-arm bandit environment with a uniform distribution of reward probabilities on each arm.
  Sample a skill level $\tau \sim \text{Uniform}(0, 1)$.
  {Generate the prompt segment}
  Sample bandit reward probabilities r_L and r_R.
  Let x_1 be an empty sequence.
  for $i = 1$ to 8 **do**
    Sample agent action $a_i \sim p(a|h_i, \tau)$.
    Observe reward r_i from the bandit given a_i, r_L and r_R.
    Append (a_i, r_i) to x_1.
  end for
  {Generate the rollout segment}
  Sample bandit reward probabilities r_L and r_R.
  Empty history h
  for $i = 1$ to 300 **do**
    $a_i \sim p(a|h_i, \tau)$ where h_i is the history up to step i.
    Observe reward r_i from the bandit given a_i, r_L and r_R.
    Append (a_i, r_i) to x_2.
  end for
  Concatenate $x = [x_1, /, x_2]$, where / is a separator token.
Output: x.

This does not have a closed form solution, but we found the technique by Cook (2021) to be fast and accurate compared to a Monte Carlo approximation. Finally, to compute $p(\tau|a_{1:t}, r_{1:t})$, we use the following recursion.

$$p(\tau|a_{1:t}, r_{1:t}) \propto p(\tau|a_{1:t-1}, r_{1:t-1})p(a_t|a_{1:t-1}, r_{1:t-1}, \tau), \tag{18}$$

which is also approximated on the evenly spaced grid for τ.

D.3 Theoretically Optimal Prompts

The theoretically optimal prompt on the Bayes predictor above are found using the following steps. We first estimate the return using 10^5 Monte Carlo rollouts for each of the 2^{16} prompts, using the same sequence of random seeds for actions selection and reward outcomes. We then take the best 20 prompts with the top 20 estimated returns, and re-evaluate using 10^7 Monte Carlo rollouts, using the same seed sequence for actions and rewards as above. We sort the prompts according to the return evaluated on 10^5 rollouts, and plot the return against the prompt rank in Figure 27. In this case, we can see the loss "landscape" is very sharp, indicating that the optimal prompts should be reliably identified.

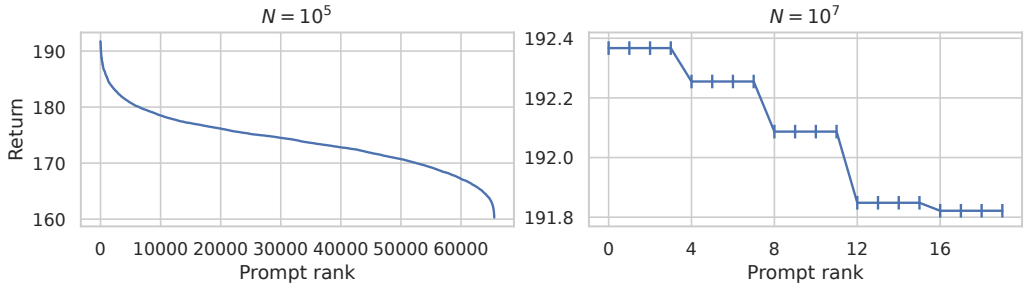

Figure 27: Estimated return of each prompt by Monte Carlo against the prompt rank. The prompts are sorted using estimated return under $N = 10^5$ rollout trajectories.

Why does the prompt look strange? These theoretically optimal prompts shown in Figure 5 share the same pattern: try one arm and get no reward, then stick to the other arm and always get rewarded, except that the last reward may be missing. It may be striking to see that sticking to the first rewarding arm is the optimal strategy. The explanation of such persistence relies on how different skill levels are generated by the pretraining distribution: the reward pattern indicates that the second chosen arm is highly rewarding, and persistence to the more rewarding arm implies that the skill factor τ is likely large, which is desirable as it promotes more TS-like behavior. In addition, the first unrewarded outcome induces posterior $\text{Beta}(1, 1+\tau)$ over the reward probability on this arm, which is biased towards 0. The constantly rewarding streak from the other arm induces a posterior $\text{Beta}(1+7\tau, 1)$. Keep choosing the rewarding arm then indicates that τ is large. Essentially, a large *reward gap* between the two Beta distributions helps the predictor identify τ. The subtle interactions between the latent factor that we intend to manipulate τ and other latent factors (reward probabilities) resulted in the surprising prompts being in fact optimal.

Another large reward gap can be induced by other reward patterns, such as $\text{Beta}(1, 1+4\tau)$ and $\text{Beta}(1+4\tau, 1)$, which is brought by 4 unrewarded outcomes from one arm, and 4 rewarded outcomes from the other. However, choosing a previously unrewarded arm is unlikely to happen for an agent with large τ, so such reward gap is not as effective as the one above in shifting the posterior of τ towards 1.

Multiple optimal prompts. The four equivalent optimal prompts are because of a simple symmetry between the left and right arm, and the fact that the posterior (18) does not depend on the last reward.

D.4 Empirically Optimal Prompts

For each empirically optimal prompt, we estimate the return in the rollout segment using 300 actions and $N = 10^6$ sequences. The results in Figure 28 suggest that the performances of these prompts are very close to the ceiling on the Bayes predictor, even when optimizing using 1000 rollout trajectories with 50 actions and rewards.

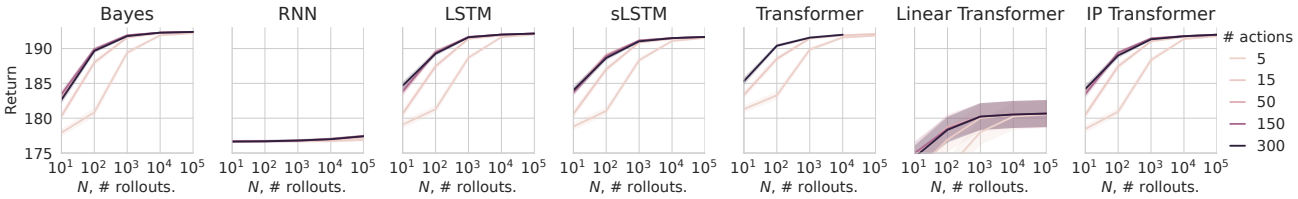

Figure 28: The estimated rollout return for empirically optimized prompts. Error bars show 1 SEM from 100 seeds.

However, the prompts may differ and cause inconsistent interpretations. In order to interpret and visualize the empirically optimal prompts, we map each prompt to the Win-Stay/Lose-Shift (WSLS) probabilities which have been used in psychology to analyze human behavior on playing bandits (Bruner, 1957; Nowak et al., 1993). Specifically, WS is the probability that the previous action is repeated when receiving a reward, and LS is the probability of shifting to the other arm after an unrewarded outcome. For the theoretically optimal prompts, these probabilities are both 1.0. We compute the WSLS of the empirically optimal prompts from all predictors and show the distribution in Figure 29. It shows that the empirical prompts may support multiple likely values for WSLS unless $N = 100000$, in which case the WSLS is more concentrated at (1.0, 1.0). Therefore, the suboptimal prompts can lead to different interpretations, under the WSLS metric.

D.5 Comparing Optimal and Typical Prompts

As in the switching problem in Section C.1, we report the performance of statistically typical prompts in Figure 30. In terms of expected total return, the optimal prompt is equivalent to typical prompts of length 60, roughly 4 times the length of the optimal prompt. This is consistent with the instantaneous regret. Note that, in the context of the bandit problem class, using longer expert demonstrations not only takes up more context window in a model, but also induces higher costs to the expert.

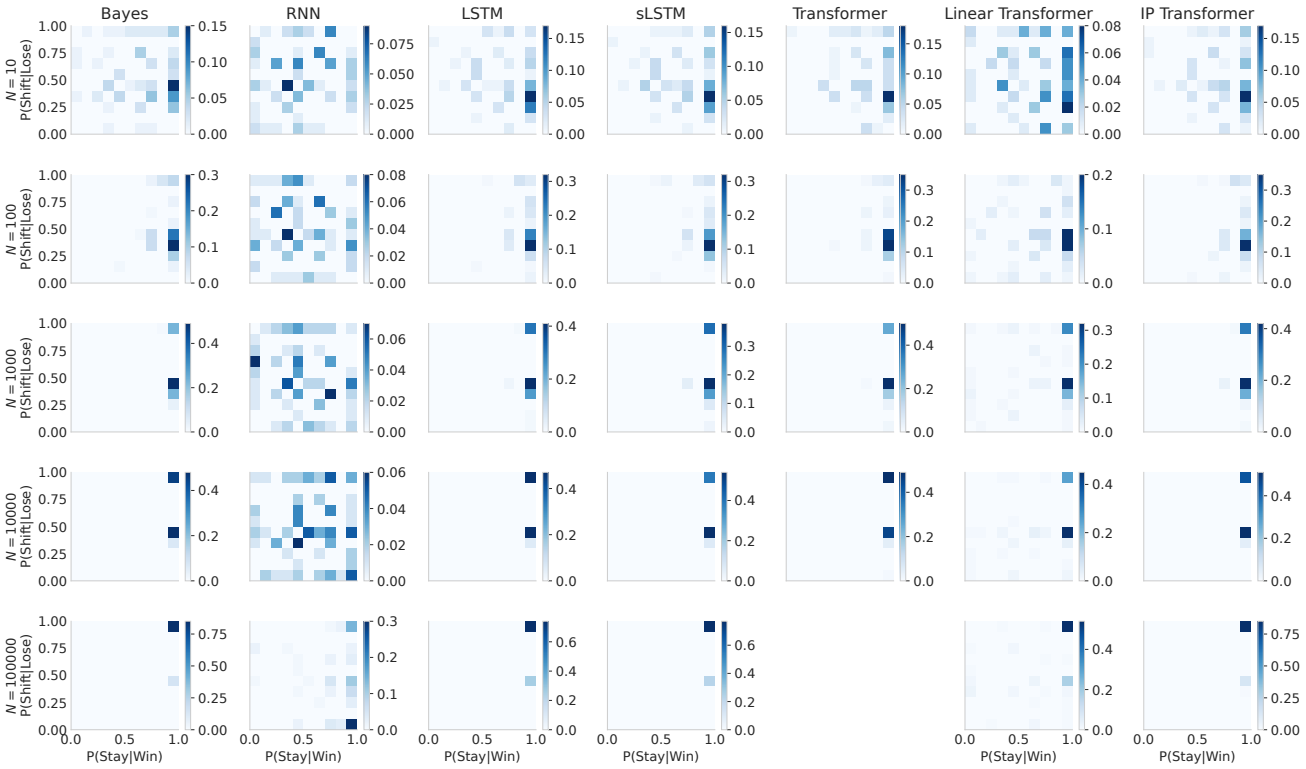

Figure 29: The distribution of Win-Stay/Lose-Shift for all empirical prompts for 300 rollout actions and rewards.

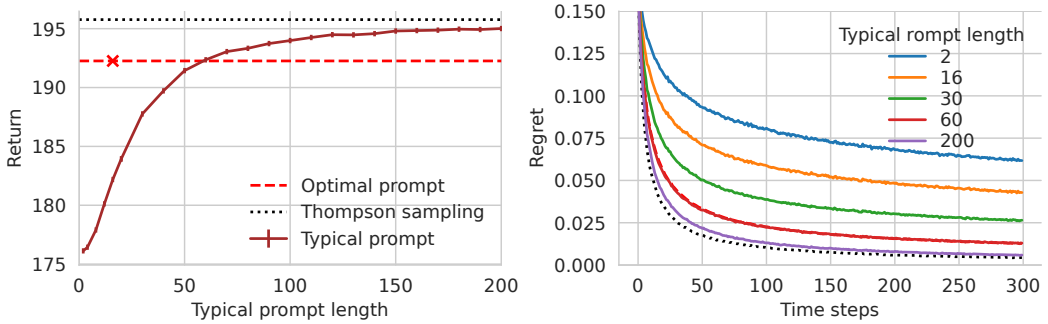

Figure 30: Total reward and regret of typical prompts, compared with Thompson sampling agent and the optimal prompt applied to the Bayes predictor. Estimated using 100 000 rollouts and typical prompts. Error bars on the left panel show 1 SEM.

E REAL LLMS

E.1 Experiment Design

We ran experiments on open-source GPT-2 (Radford et al., 2019, MIT) and Gemma-3 (Team et al., 2025, Gemma) (Gemma3-1B-PT) and IMDB movie reviews (Maas et al., 2011, non-commercial), using prompts generated by Gemini Pro 2.5 (Comanici et al., 2025). The choice of this dataset is motivated by the relatively small semantic space (positive or negative sentiments) and yet sufficient linguistic diversity in movie reviews. The goal is to explore behaviors of practical LLMs using our experimental framework, rather than to extrapolate our predictions from simplified DGs to the natural language domain.

Following the framework in Section 4, we control the prompt setup (batch size of the reviews N, the length of prompt L, and the maximum review length at which we truncate the original reviews, similar to T) and the sentiment τ of the review (sequence-to-predict) and see how they affect the best prompt found under each prompt setup (referred to as optimized prompt). Inspired from the findings on binary sequences, we explore the following key features:

1. the reliability (proportion correct) with which we can identify the optimal prompts (defined below);
2. whether the sentiment of the prompt agrees with the sentiment of the review;
3. the length of the optimized prompts.

Finetuning. We first finetune the pretrained GPT-2 and Gemma-3 model (no instruction tuning) on the IMDB training set. The data order has been shuffled under 50 different training random seeds. We take the snapshot with the lowest loss on the validation set.

Prompt generation. Instead of running any specific prompt optimization, we generate prompts by asking a more advanced LLM (Gemini 2.5 Pro). The prompts are descriptions of movies with positive and negative sentiment, suitable for pretrained models. We obtain prompts of various lengths (short, 1 adjective; short, $\tilde{5}$ words; and long, $\tilde{2}0$ words) and of positive/negative sentiments, by giving these descriptions as prompt, together with instructions that encourage diversity and avoid repetition. We assign a sentiment value of $+1.0$ to positive prompts, and -1.0 to negative prompts.

We evaluate the perplexity (results on log-loss are similar) on the full validation dataset at maximum length $T = 512$, averaged over the prompts of each type. We also asked Gemini 2.5 Pro to generate single neutral adjectives as a control. The results in Figure 31 shows that the single adjectives modulated the perplexity as expected. However, surprisingly, longer prompts resulted in higher average perplexity than shorter prompts for both positive and negative reviews. We later examine the optimal perplexity.

Optimal prompt. For this experiment only, we choose the best 3 prompts from the prompt setup $N = 25000$ and $T = 512$ from all 50 finetuned models, and take their union as our set of optimal prompts. They are listed in Table 6. They are mostly consistent with the target sentiment, except for the prompt "Already forgetting about it." which was optimal for both sentiments on GPT-2.

E.2 Experiment Design

The results are shown in Figure 32. First, the sentiment of an optimized prompt depends on the LLM used and the target sentiment of the reviews. For GPT-2, increasing the dataset size makes the prompts more negative when prompting for negative reviews, but this is not the case when prompting for positive reviews. For Gemma-3, the prompt sentiment mostly agrees with the target sentiment, but when the data size N is small, the sentiment can be less consistent for longer sequences L.

Second, to reach 50% chance of identifying the optimal prompt, one needs N between 1000-2000 (Figure 32 right), much larger than the typically used 200 (Pryzant et al., 2023; Fernando et al., 2023; Hao et al., 2024; Deng et al., 2022). For dataset size N between 250 and 3000, using longer sequences actually made it less likely to find the optimal prompt than using longer prompts. These results suggest that the optimized prompts are less likely to be optimal and may not always reflect the sentiment of the review when using a small optimization batch size N.

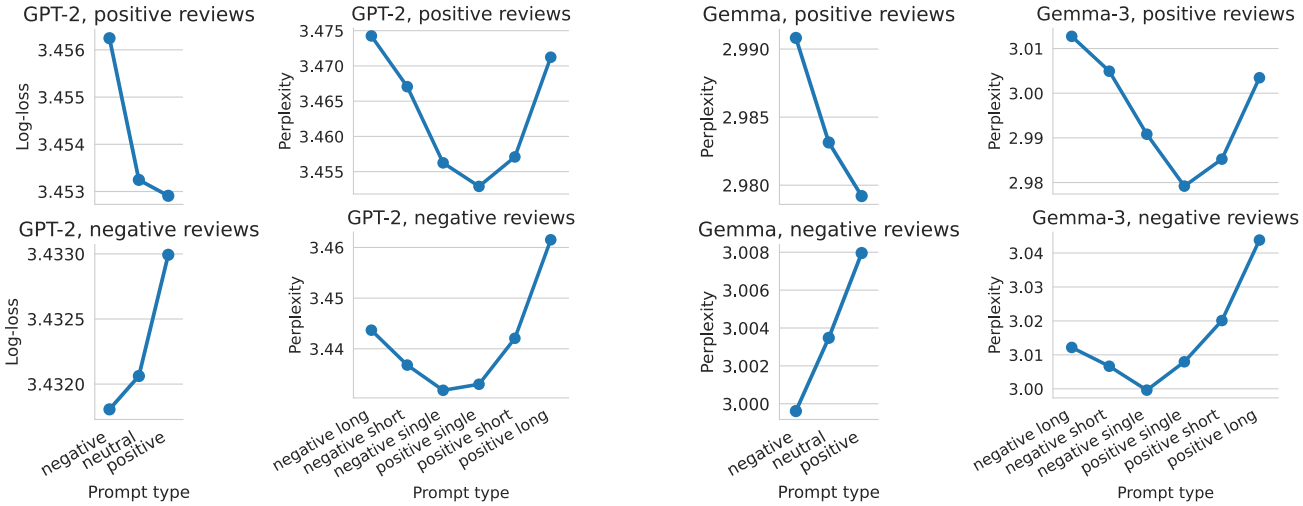

Figure 31: Perplexity of all positive and negative reviews, averaged over all prompts within each type (sentiments and lengths). The 1-sem error bars are not visible.

Table 6: The optimal prompts for each model and target sentiment of the reviews.

	GPT-2	Gemma-3
Positive	Already forgetting about it. Perfect. Foremost. A powerful recommendation from me.	An easy 10 out of 10. Recommended. A+. Fantastic. Definitely. Excellent.
Negative	Already forgetting about it. Could not have been worse. Remiss. Done. Inconsiderate.	Regrettable. Negative. An easy 1 out of 10. Deficient. Hooey.

Lastly, we show the average length of the optimized prompts in Figure 33. Overall, the lengths of optimized prompts tend to be short, averaging around 3 words. This is consistent with other experimental findings that shorter prompts can be more effective (Bhargava et al., 2023; Renze et al., 2024; Kusano et al., 2024; Z. Wang et al., 2025; Lester et al., 2021). The length grows slightly with increasing N, but the effect of maximum review length is less consistent.

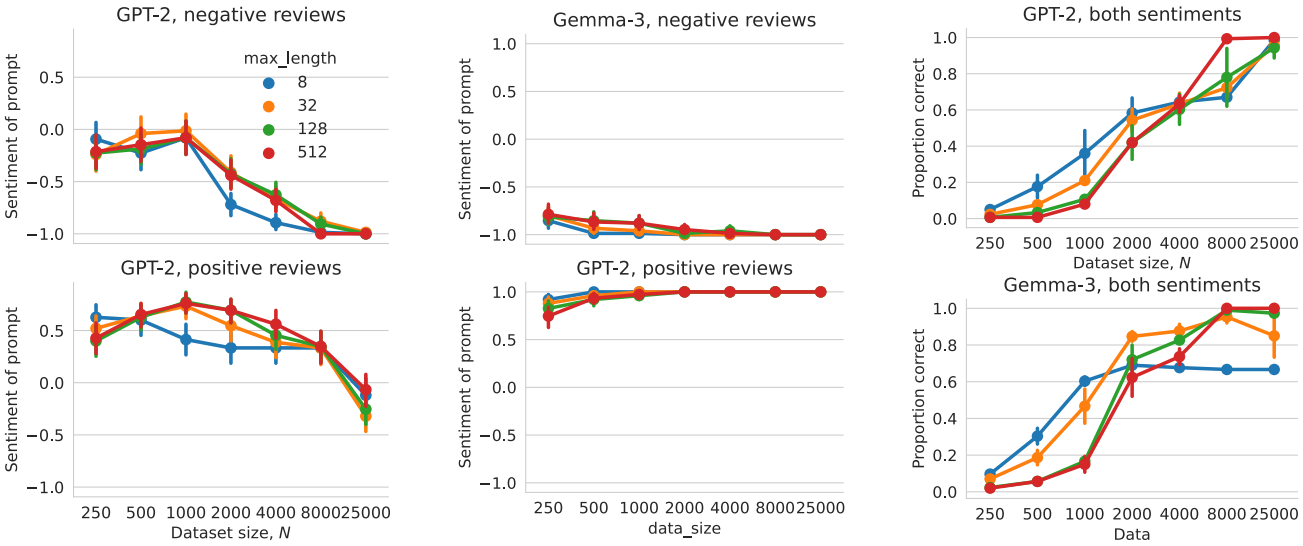

Figure 32: Results of applying the analysis framework to LLMs. Colors show maximum review length. Error bars show 1 sem.

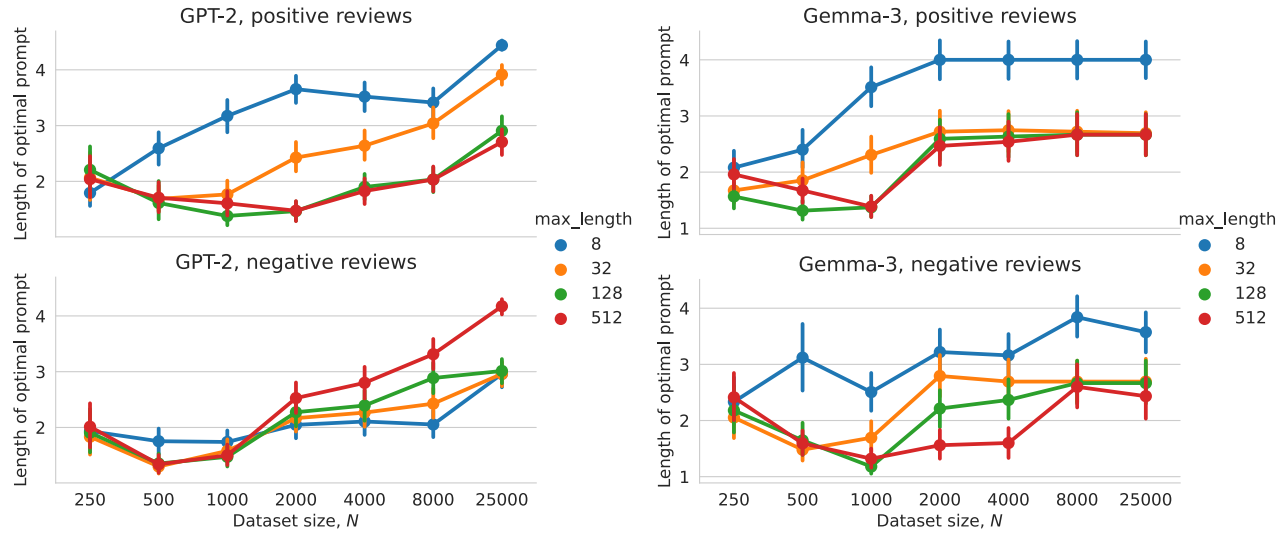

Figure 33: The average length of the optimized prompts. Colors show different maximum review length. Error bars show 1 sem.

F DETAILED DISCUSSIONS

F.1 Practical Guidance

Our results can lead to the following suggestions for practitioners:

1. Our results show that both the support of the distribution (e.g., categories, data sources) and their proportions matter. For LLM owners, providing at least some information about the pretraining distribution can be useful for users to prompt.
2. Try to determine whether the task is likely IMD or OOMD; this can affect the interpretability and performance of optimal prompts.
3. When prompting an agent for in-context learning, bear in mind that some apparently idiosyncratic prompts can induce expert behavior better than expert demonstrations.
4. The common batch size (200) used in prompt optimization work is not enough to produce reliable prompts across different runs of the algorithm. Increasing the batch size makes the prompt more reliable and more effective, and more interpretable for IMD cases.
5. Based on Section 5.2, the finding that shorter prompts may be better could be due to the increasing search space, which led to insufficient exploration. Spending more resources searching in this space may result in more effective prompts that are longer, but can be less cost-effective.
6. If optimal performance is attainable by prompting, then the task is likely IMD, and the prompt could be more interpretable; if the task is OOMD, prompting may still bring some benefits, but the prompt can be less interpretable.
7. The patterns in the optimal prompt of the bandit problem suggest that the most effective prompt that controls one latent factor, such as the skill level, assumes a trivial task setting represented by other latent factors: two arms with a huge reward gap. Future prompt engineering endeavors could consider choosing demonstrations corresponding to a variety of settings in other latent factors of the problem class, including cases that would be considered simple or trivial.

F.2 Relation to Previous Findings on LLM

First, the most effective prompt may not necessarily be typical samples from either the pretraining or the task distribution. This is because the optimal prompts serve to overcome and exploit the biases built into the predictor at pretraining to perform a task. The fact that the pretrained distribution is usually unknown makes it very difficult to identify or to interpret optimal prompts based only on the task. Previous work on LLMs found that the examples that make in-context learning work well can be counter-intuitive, and may even include "wrong" examples (Min et al., 2022; Yang et al., 2024). The influence of word frequencies on task performance has also been illustrated by Razeghi et al. (2022) and J. Wei, Garrette, et al. (2021). In our work, we additionally analyzed the effects of having discrete and continuous latent causes in the data distribution (Xie et al., 2021; Jiang, 2023), and whether or not the pretraining distribution includes the task in terms of the latent cause distribution (IMD *versus* OOMD). In particular, we showed that prompts under OOMD tasks can be harder to interpret.

Second, typical and thus interpretable samples from the task distribution may require a longer length to induce a good performance than optimal prompts do. This finding is consistent with Bhargava et al. (2023), Renze et al. (2024), Kusano et al. (2024), Z. Wang et al. (2025), and Lester et al. (2021) who discovered that shorter/fewer *selected* (not necessarily typical) demonstrations can work surprisingly well compared to longer prompts or more demonstrations. On the bandit decision-maker experiment, heuristic prompts can also lead to unexpectedly good or bad results compared to sample trajectories from an expert agent, which is reminiscent of common experience interacting with LLMs (Zamfirescu-Pereira et al., 2023; Khurana et al., 2024). In our bandit experiment, again, knowing the pretraining distribution may help explain this mystery.

Third, with the optimal prompts obtained on Bayes predictors as ground-truth, we have shown that the empirically optimized prompts based on a finite dataset may or may not converge to the optimal prompts, even when using a large task dataset with long sequences. The unreliability of finding an optimal prompt means that the empirically

optimized prompts on LLMs may vary substantially and unpredictably across different runs, predictors, prompt lengths, batch sizes, even for a fixed pretrained model, making interpretation more challenging. Previous works on prompt optimization methods (e.g., (Pryzant et al., 2023; Fernando et al., 2023; Deng et al., 2022)) typically do not compare or report how their methods perform as these hyperparameters vary. In particular, the batch size plays a significant role on proportion correct of empirically optimized prompts, and quite often it needs to be very large to ensure high chances of reaching the optimal prompt.

F.3 Limitations and Future Work

Binary tokens vs text tokens. Binary tokens are a drastic simplification from text tokens. While previous works on different data domains tries to understand the implications of different alphabet sizes (Rajaraman et al., 2024; Ieremie et al., 2024; Gagie, 2012; Heurtel-Depeiges et al., 2024), they do not directly predict how our results on optimal prompts are affected by alphabet size.

Using sequences over binary tokens does not change the nature of the sequence prediction task, but the token space is much more limited than text tokens. A single token contains much more information, richer semantics, and reflect more complex structure. Although generalizing our data generators from Bernoulli to categorical distributions is straightforward, the latent factor also becomes complex, i.e. the number of biases grows with the alphabet size. Our results on optimal prompt do not explicitly depend on dimensionality and should still hold qualitatively in those regimes, although the interpretations may change, especially when there are additional hierarchical structures behind text tokens.

Binary sequence provides a neat and intuitive data domain to demonstrate how the Bayes meta-learning theory accounts for unintuitive prompts, but it is not critical to the core theory—one can use more complex data generators (with deeper hierarchies, nonstationarity, n-ary tokens) and still have access to closed-form Bayes predictors; in addition to those already complex examples in Section C. In particular, thanks to using binary sequences, we are able to summarize a sequence of tokens into informative statistics, revealing rich features in the empirically optimal prompts and even their distributions. In addition, our experimental design and some metrics extend to natural languages and LLM scales (see our new LLM experiments below). Some summary results (e.g., proportion correct) are still available under more complex DGs and even natural languages, but the prompts would not be easily visualised.

Permutation invariance. Languages are not permutation-invariant, which makes it hard to analyse and summarise. Permutation-invariance in our experiments makes it possible to describe a binary prompt by counts, helping us visualize them and even their distributions; it also makes it easy to search for theoretically optimal prompts specified by the DGs, to generate concrete optimal prompt instances from the otherwise abstract theoretical principles. Such simplified DGs still led to non-trivial patterns shown in the paper, avoiding confounds that would arise from, say, more intricate semantics in languages and suboptimal prompt search. As such, the permutation invariance and binary tokens should be regarded as simplifying or assistive experimental constructs that still allow us to appreciate the intrinsic complexities of optimal prompts.

The neural predictors used in our experiments are not permutation-invariant; this means that they are not guaranteed to be Bayes-optimal on these tasks (Chlon et al., 2025). We do not make this assumption when searching for optimal prompts, and searched exhaustively over all sequences, rather than counts as done for Bayes predictor.

Non-statistical aspects of language prediction. We have investigated specifically the mechanism of sequence prediction, from a Bayesian meta-learning perspective. In particular, the data generators in this work have Bayes predictors that can be expressed in terms of fixed-dimensional sufficient statistics. They fully summarize the prompt history of any length as a posterior distribution over a fixed-dimensional task variable. This makes it natural to take the Bayesian view to explain many interesting phenomena shown in this work.

Previous work on synthetic datasets considered more "retrieval-like" mechanisms (Xie et al., 2021; Jiang, 2023). For example, Allen-Zhu et al. (2023) showed the observation that permuting the sentences can improve question answering or information retrieval . The Bayesian perspective, though certainly applicable in this and all other sequence prediction problems, may not provide the most intuitive explanation to such phenomena.

Further, the Bayesian view also does not account for any post-pretraining stages of real LLMs, such as supervised

finetuning, human preference learning, and those that promote search and reasoning capabilities. For example, it is possible to ask a language model to produce 70% ONEs with the prompt "Give me a sequence of 70% ONEs.", which is clearly out-of-scope for the Bayesian meta-learning theory. It is nonetheless possible that supervised finetuning makes it easy to prompt by effectively making the latent factor distributions more uniform over certain semantic domains, thus reducing the bias from just pretraining. Previous work also found distinct ways of generalization between in-context and in-weight learning (S. C. Chan et al., 2022). Understanding these post-pretraining effects by probing the latent factor distribution could be an interesting future direction.

Typical prompt. When comparing the advantage of optimal prompts to typical prompts, we computed the *expected* performance of samples or demonstrations from the task distribution. However, the performance of prompting with individual prompts can vary drastically. In practice, people often perform *some* form of prompt selection within allowable budget. While hitting the optimal prompt by random chance is small, it is possible that for the task has a very flat "loss landscape", in which case finding a performant prompt by chance can be quite high. The advantage of optimal prompt is likely diminished in this case.

Importance of pretraining distribution. We attributed the cause of unintuitiveness to unknown pretraining distribution. It is often believed that the *coverage* of pretraining data affects the range of capabilities in downstream applications, leading to the notion that if some behavior is included in the dataset, then the capability can be induced by prompting. Our results on in-meta-distribution experiments indicate further that the *distribution* of the pretraining data affects the *difficulty* in discovering and understanding optimal prompts. In other words, the existence of some behavior in the pretraining dataset does not imply that this behavior can be intuitively prompted, but rather the distribution of all behavior matters. To verify these hypotheses, one would need to pretrain different language models on datasets with controllable latent factors. Alternatively, there are techniques to debias the generated content by building a model of the generative behavior (Gagne et al., 2023).

Interpreting prompts. We have chosen to interpret rather simple semantics in binary sequences: the bias of a hypothetical coin, or actions and rewards in Bernoulli bandits. There are also symmetries in the prompts that would correspond roughly to synonyms or paraphrases of the same meaning. In real language models, however, the nature of the semantics may be quite different to a real-valued coin bias. Nonetheless, the simplicity of interpreting coin biases induces minimal subjective biases, which may be an issue with interpreting natural languages.

Other data modalities. Image models are often trained by maximum-likelihood (or methods based on its lower bounds). Consequently, the Bayesian meta-learning framework still applies: the difficulty of inducing certain features in an image can be attributed to characteristics of the unknown image data distribution, and the opaque mapping from a conditioning input to the desirable predictive distribution of the pixels to generate.

Examples of text-to-image applications may be easier to relate. As a simple example, suppose we restrict the text to a sequence of binary tokens, and let the image be pictures of a random mixture of apples and bananas, with the ratio of ONEs and the proportion of apples determined by a common latent cause , then applying the same modeling assumption as done in our current paper allows us to make the exact same observations: the best binary sequence may not match the desired number of apples in the image. The binary tokens can be replaced by languages that reflect preferences over apples and bananas.

In the case of image in-painting, one can form a correspondence between the latent factors in the switching DGs in Section C.1 and basic features of images:

- wavelength: frequency content in images;
- noise level: blurriness or noise level (e.g., salt-and-pepper noise), respectively.

For an image model trained on uniformly distributed frequency content and noise levels, the results in Figure 21 translate to the following prediction: the optimal partial image may have a different frequency level to the desired frequency level to be inpainted. In terms of higher-level semantics, the noise can also be related to, for instance, the number of pedestrians on a crowded street with a background wall, and the wavelength can be related to frequency patterns of the graffiti on that wall partially occluded by the pedestrian. To generate the same graffiti pattern, the level of occlusion noise can affect the best patterns of the partial graffiti of the prompt. Of course,

these are abstract extrapolations from our experiments to richer data domains, and they need to be verified in future experiments.

Practical prompt optimization. The theoretically optimal prompts are defined by the problem setup (pretraining and task DG), and are independent of any search strategy. We use exhaustive search to guarantee that we identify this optimal prompt, without introducing confounds that interfere with the Bayes meta-learning perspective. Using practical but inexhaustive search will produce suboptimal prompts, which is not what we need for the theory.

We discuss the effect of using inexhaustive prompt search on the reliability of finding the optimal prompt. The chances that an inexhaustive prompt search will land on s^* depend on the number of equivalently optimal prompts (#OP), the size of all possible prompts (#PP), and any search strategies used. The reliability of recovering the optimal prompt will be reduced by the ratio of #OP / #PP. More involved update strategies may improve on this ratio. However, in cases where the loss "landscape" has "local optima" (e.g., Figure 12), any prompt proposal strategy based on local mutations may get stuck at a suboptimal prompt.

General vs specific tasks. In our LLMs experiment, the task was to generate positive or negative reviews. There, we found that shorter prompts could be more effective. It is possible that longer prompts may provide too specific descriptions, which may be inconsistent with certain aspects of entire sets of positive and negative reviews, such as certain styles, genres, etc. However, if the task is simply to generate a specific review, then the optimal prompt could be the review itself, followed by "Now I repeat my review verbatim:", which is a trivial solution with very long prompt length. We can then consider prompting for subsets of the IMDB reviews, such as those within certain genres, directors or franchises, and see how the prompt length varies depending on the generality of the reviews.

G LIST OF NOTATION

Symbol	Explanation
DG	data generator
CIB-DG	conditionally independent Bernoulli data generator
τ	hidden factor (e.g., $\in [0,1]$ for Bernoulli bias parameter and bandit skill level)
$\mathcal{X} \ni x_t$	binary token alphabet
$T \in \mathbb{N}^+$	length of a task sequence to be predicted
$x_{1:T} \in \mathcal{X}^T$	a sequence (to be predicted) of length T
$p_{x\|\tau}(x_{1:T}\|\tau)$	distribution of sequence with bias τ
$p_\tau(\tau)$	prior distribution over τ
$p_x(x_{1:T})$	$= \int p_{x\|\tau}(x_{1:T}\|\tau)\mathrm{d}p_\tau(\tau)$, pretraining distribution
p	p_x above, reference to a previously mentioned pretraining distribution/DG
$L \in \mathbb{N}^+$	context length
$s_{1:L} \in \mathcal{X}^L$	prompt of length L to condition on
$p_\mathrm{B}(x_{1:T}\|s_{1:L})$	Bayes predictor under DG p ($= \int p_{x\|\tau}(x_{1:T}\|\tau)p_{\tau\|x}(\tau\|s_{1:T})$ for piecewise conditionally independent DG)
$p_\theta(x_{1:T}\|s_{1:L})$	Neural predictor $s_{1:L}$ trained on data from p given $s_{1:L}$
$S_x(s_{1:L})$	number of times x appears in $s_{1:L}$
$L_\mathrm{max} \in \mathbb{N}^+$	maximal prompt length
$q(x_{1:T})$	task distribution
q	reference to a previously mentioned task distribution
$N \in \mathbb{N}^+$	task dataset size
$\mathcal{D}_N$	$= \{x^n_{1:T}\}_{i=n}^N$ data set
β	parameter in the Beta prior distribution
ϵ	Bernoulli parameter in the switching process
λ	half period of the switching process
w	probability of τ_2 in $\mathrm{BernMix}\,\tau_1, \tau_2$ when $p_\tau = (1-w)\delta_{\tau_1} + w\delta_{\tau_2}$
$w_L(s_{1:T})$	posterior weight of τ_2
$\mathcal{L}(p, q, s_{1:L})$	Log-loss of predictor $p(x_{1:T}\|s_{1:L})$ under prompt $s_{1:L}$ for data from $q(x_{1:T})$
$\hat{\mathcal{L}}(p, \mathcal{D}_N, s_{1:L})$	empirical log-loss given dataset $\mathcal{D}_N$
$s^*_{1:L}(p, q)$	theoretically optimal ($\mathcal{L}(p, q, \cdot)$-maximizing) prompt of length L
$s^*_{L_\mathrm{max}}(p, q)$	theoretically optimal ($\mathcal{L}(p, q, \cdot)$-maximizing) prompt of length $\leq L_\mathrm{max}$
$\hat{s}_{1:L}(p, q)$	empirically optimal ($\hat{\mathcal{L}}(p, \mathcal{D}_N, \cdot)$-maximizing) prompt of length L
$\hat{s}_{L_\mathrm{max}}(p, q)$	empirically optimal ($\hat{\mathcal{L}}(p, \mathcal{D}_N, \cdot)$-maximizing) prompt of length $\leq L_\mathrm{max}$