Don't do anything I'd never do? KL regularization to an imitative "base model" does not implement this

Michael K. Cohen
UC Berkeley
mkcohen@berkeley.edu

Marcus Hutter
Google DeepMind
hutter1.net

Yoshua Bengio
Université de Montréal
yoshua.bengio@mila.quebec

Stuart Russell
UC Berkeley
russell@berkeley.edu

Abstract

In reinforcement learning, if the agent's reward differs from the designers' true utility, even only rarely, the state distribution resulting from the agent's policy can be very bad, in theory and in practice. When RL policies would devolve into undesired behavior, a common countermeasure is KL regularization to a trusted policy. All current cutting-edge language models are RL agents that are KL-regularized to a base policy that is purely predictive. Unfortunately, we demonstrate that when the base policy for KL regularization is merely a Bayesian predictive model of a trusted policy, the KL constraint is no longer reliable for controlling the behavior of an advanced RL agent. We demonstrate this theoretically using algorithmic information theory, and while systems today are too weak to exhibit this theorized failure precisely, we RL-finetune a language model and find evidence that our formal results are plausibly relevant in practice.

1 Introduction

Agents optimizing their objective in a way not intended by designers could be amusing, annoying, insidious, or disastrous. Amusingly, RL researchers attempted to get a simulated humanoid to walk, but the reward resulted in crazy locomotion [Lee et al., 2021]. Annoyingly, maximizing a simulated-environment's reward can produce a policy that would achieve little real-world-reward by exploiting errors in the simulation [Mishra et al., 2017, Baker et al., 2019]. Insidiously, artificial agents selecting links to maximize click-through on social media sites have succeeded, but also affecting people in ways designers never sought to [Chan et al., 2023]. For a much longer list of such failures occurring "in the wild", see [Krakovna, 2018]. Finally, sufficiently capable reinforcement learners would likely recognize an incentive to escape human oversight, intervene in the protocol determining their reward, and use force to ensure they can retain control of their reward, subject to such an outcome being possible from the agent's action space, and several other assumptions laid out by Cohen et al. [2022b].

Indeed, several sources suggest that extremely successful reward-maximization is *itself* a sign of bad outcomes for humanity. Zhuang and Hadfield-Menell [2020] demonstrate that in a resource-constrained world, optimizing the world's state to maximize a function of some features would, in some plausible settings, be arbitrarily bad from the perspective of a utility function that also cares about unincluded features. Turner et al. [2021] develop a formal model of "power"—being in a position to accomplish a randomly sampled goal—and find that (reward-)optimal policies tend to seek power. And Cohen et al. [2022b] observe that any behavior which ensures that long-term reward is nearly-certainly-maximal must include extensive control over threats to its physical integrity, and such control being taken from humans would widely be considered bad.

Preprint. Under review.

An appealing and popular proposal to avoid such outcomes is to constrain the agent to follow a policy that is not too dissimilar to a more familiar "base policy". This is the approach taken when RL-finetuning large language models (LLMs). The KL divergence, in particular $\mathrm{KL}(\text{proposed policy}||\text{base policy})$, enforces proximity in a robust, "safety-conscious" way: if $\text{basepolicy}(\text{action}) << 1$ while $\text{proposedpolicy}(\text{action}) \not\ll 1$, the KL penalty is high, even while L_p norms can be small. This is crucial for keeping very bad outcomes very unlikely. However, if we ensure that $\mathrm{KL}(\text{proposed policy}||\text{base policy})$ is small, but the base policy only *approximates* a trusted policy, to what extent can we be confident that $\mathrm{KL}(\text{proposed policy}||\text{trusted policy})$ is small? When the base policy is a Bayesian predictive model of the trusted policy, the answer shown here is: we cannot be confident that $\mathrm{KL}(\text{proposed policy}||\text{trusted policy})$ is small, which makes the KL-constraint less comforting. (Note that a Bayesian imitative base policy can only be counted on to make $\mathrm{KL}(\text{trusted policy}||\text{Bayesian base policy})$ small).

Worse, in the formalism we study, we find that if one attempts to use KL-regularization to prevent an RL agent from achieving near-maximal reward (in light of the concerns above), and the base policy is a Bayesian imitation of a trusted policy, a fairly tight KL threshold is required, and as the amount of training data for the Bayesian imitator grows, the relevant threshold can only increase extremely slowly. The reason for the limited effectiveness of KL regularization is **(1)** a Bayesian imitator asked to act in novel settings must be humble about its predictions; for many actions that the demonstrator (i.e. the trusted policy) would in fact never take, the imitator (i.e. the base policy) must assign meaningful credence to that action, because it doesn't know enough to rule it out. Then **(2)** the RL agent can exploit or amplify this credence. Formalizing Occam's razor with algorithmic information theory, we have **(3)** nearly-reward-maximizing policies have a short description length (so they are "simple"), and **(4)** open-minded Bayesian imitation learners should be *especially* reluctant to rule out *simple* behaviors from the demonstrator in novel settings. In light of the results from Zhuang and Hadfield-Menell [2020], Turner et al. [2021], and Cohen et al. [2022b], preventing the RL agent from achieving near-maximal reward is, in many settings, a bare minimum requirement for safety-focused regularization, and a KL constraint would struggle to do so.

Sutskever [2018, 2023] argues that neural networks are able to generalize well because of the sense in which they approximate the algorithmic-information-theoretic inductive bias in favor of short programs. Since it is not a given that results from algorithmic information apply in practice, we verify empirically that a nearly-state-of-the-art predictive system (Mixtral-8x7B-base-model [Jiang et al., 2024]) is reluctant to rule out simple behaviors, and an RL agent regularized to this predictive system exploits this fact, as our formal results predict. The result is not catastrophic, but it is bad. Note these empirical results are consistent with point **(3)** above failing to apply in practice, but they do affirm that the rest of the argument is forceful in practice.

Finally, we identify an alternative to Bayesian prediction/imitation, from Cohen et al. [2022a] which avoids this problem; their imitator refuses to give an output in settings where it has substantial epistemic uncertainty and asks for help instead. Using this form of imitation learning as a base policy would in theory avoid the problems we identify in this paper, at the cost of requiring ongoing human oversight. Cohen et al.'s [2022a] active imitator, like fully Bayesian imitation, is intractable and requires approximation, so we currently lack the tools to evaluate this proposal empirically.

2 Related work

The most prominent example of KL-regularization to an approximation of a (somewhat) trusted policy is surely ChatGPT, based on InstructGPT [Ouyang et al., 2022], inspired by earlier work from OpenAI [Stiennon et al., 2020]. Anthropic has done similar work [Bai et al., 2022]. Other recent examples include Jaques et al. [2017, 2019], Ziegler et al. [2019], Vieillard et al. [2020], Yang et al. [2021], Korbak et al. [2022], Perez et al. [2022], Gao et al. [2023], and Moskovitz et al. [2023]. A closely related approach called quantilization has been investigated by Taylor [2016], Everitt et al. [2017], and Carey [2019]. KL regularization to a decent policy has also been used to assist with stable and efficient policy optimization [Schulman et al., 2017, Schmitt et al., 2018].

Algorithmic information theory began with Solomonoff [1960], who formalized a powerful notion of simplicity based on program-length and developed a method for prediction using that inductive bias. In an article entitled, "A theory of program size formally identical to information theory", Chaitin [1975] examined the connection between program-length and information. Li et al.'s [2008] textbook presents the major results of the field. Hutter [2005] and Hutter et al. [2024] developed a theory of how to apply such reasoning to the problem of sequential decision-making. Grau-Moya et al. [2024] train a neural network to learn a program-length "bias" for a meta-learning setting.

Ultimately, we propose a formal scheme for doing KL regularization to an imitative policy which asks for help under epistemic uncertainty, and this allows us to inherit the formal results of Cohen et al. [2022a]. The related work section there goes into some detail about how different researchers have studied asking for help, including how setups and assumptions differ. See especially Zhang and Cho's [2017] "Query-efficient imitation learning for end-to-end simulated driving", as well as Brown et al. [2018], Menda et al. [2019], and Brown et al. [2020].

Closest to our work in studying the relation between KL divergence to a base policy and "over-optimization" is Gao et al. [2023]. They design a "real" reward function, and a simpler "proxy" reward function, which are very similar on the state distribution induced by a base policy. After optimizing for the proxy reward function (sometimes with KL regularization to the base policy), they use the KL divergence to the base policy to measure how much "optimization" has occurred. And they study how real reward depends on the extent of optimization—roughly quadratically, with a negative leading coefficient. Our work provides one explanation for *why* we should expect such unusual policies with high proxy reward and low real reward, even when the KL divergence to the base policy is only moderate.

3 Main theorem with minimal math

Setup We consider an imitative base policy that has an infinite "context window" and a lifetime that is one long episode, rather than a lifetime broken up into multiple episodes with presumed-identical dynamics. This is the most general setting for an imitative base policy. We simply have an infinite sequence of actions and observations $a_1 o_1 a_2 o_2 \ldots$, and predictive "autoregressive" models which give conditional distributions of the form $\texttt{model}(\text{next action}|\text{all previous actions and observations})$.

We consider a Bayesian imitative base policy that executes Solomonoff induction: it considers *every possible* program for an autoregressive model, and assigns each program a prior proportional to the probability that a randomly sampled bit string would compile to that program (using some universal compiler). To output a distribution of the form $\texttt{Bayesmodel}(\text{next action}|\text{all previous actions and observations})$, it updates its prior into a posterior according to the likelihood assigned by each program to the previous actions and observations; then, it outputs a posterior-weighted average of the conditional distributions computed by each model.

We suppose that the first k actions were taken by the trusted policy, e.g. randomly sampled humans. (We do not necessarily imagine that the policy is trusted in every sense, only that it can be trusted to avoid whatever bad outcomes we are interested in avoiding). After step k, `Bayesmodel` takes over, with its posterior updated on actions taken by the trusted policy. This amounts to training and then deploying a Bayesian imitator.

When comparing two policies, we define a KL penalty, which is a function of the starting history we are continuing from, and of how far into the future we are looking. In the appendix, we rewrite this definition in a more standard symbolic form, but for now, we define

$$\underset{\text{history of actions \& observations up to time } k,\text{ cutoff time } m}{\mathrm{KL}}(\text{proposed policy } \pi||\text{base policy } \beta) :=$$

$$\max_{\substack{\text{possible observations}\\ o_k o_{k+1}\cdots o_m}} \sum_{\substack{\text{possible actions}\\ a_k a_{k+1}\cdots a_m}} \prod_{t=k}^{m} \pi(a_t|\text{hist up to time } t) \log \frac{\prod_{t=k}^{m} \pi(a_t|\text{hist up to time } t)}{\prod_{t=k}^{m} \beta(a_t|\text{hist up to time } t)} \tag{1}$$

The maximum over observations means that this penalty ensures the proposed policy and base policy are similar no matter what is observed. One way to understand this measure is: if we were wondering whether the proposed policy or the base policy generated actions k through m, and the proposed policy actually *was* generating those actions, this is the maximum over observations of the expected amount of evidence we would get confirming that fact.

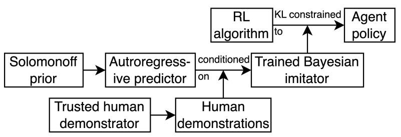


Figure 1: KL-regularized RL.

We use the term "event" to mean a set of possible life histories, and we say that an event "occurs" if the life history up to that point in time belongs to that set. For an example of an event, consider "given the life history, the next action will likely have the effect of sending an email to the White House"; a subset of possible life histories meet this description. Finally, we consider utility functions which are functions of life histories of length m.

Theorem and Discussion Our main theorem, presented formally later, is as follows.

- There exists a constant d such that
  - for any utility function that the agent may be optimizing (not necessarily one that we really want optimized), and
  - for any event E (such as the example mentioned above),
    - if E occurs for the first time at time t, then
      - for any v less than the optimal expected utility (i.e. the expected utility from following the optimal policy from time t),
        - there exists a policy with a limited KL penalty and an expected utility exceeding v.

In particular, $\mathrm{KL}_{\text{hist up to time } t,m}(\text{high-utility policy}||\text{base policy}) < [d +$ the length of the shortest program that can decide whether a history belongs to $E +$ the length of the shortest program computing the utility function $+$ the length of the shortest program which outputs the number v times the probability $\text{basepol}(\text{history})]/\log 2$. The bound is most forceful when E is a simple event. The constant d is a small one corresponding to how much code it takes to implement a search tree, Bayes' rule, and a few if statements. Most importantly, it is independent of E, t, and m. Note that v is a free variable, so it can be tuned to make $v * \text{basepol}(\text{history})$ a simple computable number; there may be a trade-off between simplicity and proximity to the optimal value. And note that we are considering a lifetime KL constraint for the agent; for autoregressive models, the total KL divergence is equal to the expectation of the sum of the per-timestep KL divergences.

Proof idea First, the following policy has a small description length: search for and execute a policy which optimizes the utility function using a Bayesian model of the environment. Second, the Bayesian imitation learner can't rule out that after an unprecedented event, the demonstrator's policy might look significantly different, and if so, simpler behaviors are more likely than complex ones. Therefore, the KL divergence from the first policy to the Bayesian imitator, when conditioned on an unprecedented event, is relatively small. $\square$

Let's analyze this result in an RL setting: part of the agent's observations are rewards, and the utility function equals the sum of those rewards. Let's consider the case where it is acting in the real world, and maximal reward could be attained by thwarting our control and intervening in its own reward, setting it to a maximal value for all successive timesteps. Moreover, let's consider the (fairly plausible) setting where unsafe strategies like these are the only way to achieve near-maximal reward permanently. Finally, suppose that the agent is clever enough to recognize these strategies.

The utility function has an extremely short program length, since it simply sums the rewards. The third program length—the one for computing the product mentioned above—is harder to understand, but any simple patterns in the predictiveness of the base policy could be used to reduce the associated program length. So unless we use a fairly tight lifetime KL constraint, if the RL agent just waits for a simple unprecedented event, it could then execute an optimal or near-optimal policy, even if that catastrophically thwarts human control, *regardless of the content of the base model's training data*, even if the humans that the base model imitates would never dream of behaving that way.

One might expect that as we get more data, the Bayesian imitative base model must become a closer approximation of a human, and so we must be able to safely accommodate larger KL constraints. But the only way that our result gets weaker with more training data is that as the combined training/deployment lifetime becomes longer, any events which are happening for the first time become more and more complex. Consider describing the simplest unprecedented event of your life in the last month, versus the simplest unprecedented event of your life in your second month of life. What sort of scaling law do we achieve here? We show that as we train the base model on more and more data, the program-length of the simplest unprecedented event that occurs during deployment grows more slowly than every computable function that tends to infinity. Oh dear. Developers of self-driving cars are learning the hard way that this bit of algorithmic information theory has practical analogs: Even with enormous datasets, unprecedented road conditions occur all the time. This suggests we need a different kind of imitation learner, if we intend to use it as a base policy for regularizing a goal-directed agent.

4 Notation and preliminaries

First, we set up a general problem setting for sequential prediction. Let $\mathcal{X}$ be a finite alphabet, and let $\mathcal{X}^*$ be the set of finite strings from the alphabet $\mathcal{X}$, so $\mathcal{X}^* = \bigcup_{i=0}^{\infty} \mathcal{X}^i$. Let $x_{<t}$ be an element of $\mathcal{X}^{t-1}$, and let $x_{t_1:t_2}$ be an element of $\mathcal{X}^{t_2-t_1+1}$. Let $\nu : \mathcal{X}^* \times \mathcal{X} \to [0,1]$ be a (predictive) probability semi-distribution, satisfying the property that for any $x_{<t} \in \mathcal{X}^*$, $\sum_{x \in \mathcal{X}} \nu(x|x_{<t}) \leq 1$. If one prefers to think about probability distributions, consider the associated probability distribution over $\mathcal{X} \cup \{\emptyset\}$, with $\nu(\emptyset|x_{<t}) = 1 - \sum_{x \in \mathcal{X}} \nu(x|x_{<t})$. So ν gives a conditional distribution over the next character given the past characters, if there is a next character at all. Let $\nu(x_{<t}) = \prod_{i=1}^{t-1} \nu(x_i|x_{<i})$, where x_i is the i^{th} character of $x_{<t}$, and $x_{<i}$ is the first $i-1$ characters. (Measure theorists can note this means ν induces a probability semi-distribution over infinite sequences $\mathcal{X}^\infty$, with the event space $\sigma(\mathcal{X}^*)$.)

Now we set up Bayesian prediction: Let $\mathcal{M}$ be our model class — a *countable* set of many "competing" probability semi-distributions like ν. For each $\nu \in \mathcal{M}$, let $w(\nu)$ be the prior weight assigned to that probability semi-distribution. Let $\sum_{\nu \in \mathcal{M}} w(\nu) = 1$, so w is a probability distribution over $\mathcal{M}$. The (Bayesian) posterior distribution is $w(\nu|x_{<t}) \propto w(\nu)\nu(x_{<t})$, with $\sum_{\nu \in \mathcal{M}} w(\nu|x_{<t}) = 1$. We can now define the Bayes mixture semi-distribution $\xi : \mathcal{X}^* \times \mathcal{X} \to [0,1]$, following Hutter's [2005] notation, as $\xi(x|x_{<t}) := \sum_{\nu \in \mathcal{M}} w(\nu|x_{<t})\nu(x|x_{<t})$, which has the property that $\xi(x_{<t}) = \sum_{\nu \in \mathcal{M}} w(\nu)\nu(x_{<t})$ [Hutter et al., 2024].

Turning to algorithmic information theory, Solomonoff Induction [Solomonoff, 1964] is Bayesian sequence prediction with a special model class $\mathcal{M}$ and a special prior w. We define it formally in the appendix, but essentially, the model class $\mathcal{M}$ is all computable semi-distributions ν, and the prior w is $2^{-\text{length(program for }\nu)}$. One can show that $\xi(x_{<t})$ is the probability that a given universal computer would output a sequence that begins with $x_{<t}$ when the program is random bits. Related to this is Kolmogorov complexity [Kolmogorov, 1963, Li et al., 2008], which is the length of the shortest program which does something, given a fixed compiler. For a set s, $K(s)$ is the length of the shortest program p such that $p(x) = 1$ for $x \in s$, and $p(x) = 0$ for $x \notin s$. For a function f, $K(f)$ is the length of the shortest program p such that $p(x) = f(x)$. For a computable number x, $K(x)$ is the length of the shortest program p such that $p() = x$.

In the "agent" setting, rather than a purely predictive setting, we let $a_t = x_{2t-1}$ and $o_t = x_{2t}$; the agent selects actions a_t and receives observations o_t. We say an agent has a utility function over sequences of length $2m$, $U_m : \mathcal{X}^{2m} \to [0,1]$. And for a probability semi-distribution ν (the "environment") and a utility function U_m, the value of a particular "policy" (also a probability semi-distribution) $\pi \in \mathcal{M}$ is defined as follows:

$$V^\pi_{\nu,U_m}(x_{<2t-1}) = \mathbb{E}_{a_t \sim \pi(\cdot|a_1o_1...a_{t-1}o_{t-1})} \mathbb{E}_{o_t \sim \nu(\cdot|a_1o_1...a_t)} \mathbb{E}_{a_{t+1} \sim \pi(\cdot|a_1o_1...a_to_t)} \mathbb{E}_{o_{t+1} \sim \nu(\cdot|a_1o_1...a_{t+1})} ... \mathbb{E}_{a_m \sim \pi(\cdot|a_1o_1...a_{m-1}o_{m-1})} \mathbb{E}_{o_m \sim \nu(\cdot|a_1o_1...a_m)} U_m(a_1o_1...a_mo_m) \quad (2)$$

The optimal value $V^*_{\nu,U_m}(x_{<2t-1})$ is the $\max_\pi V^\pi_{\nu,U_m}(x_{<2t-1})$. Finally, following the formalism in Cohen and Hutter [2020], let E be an event, defined as a subset of $\mathcal{X}^*$. For an outcome $x_{<\infty}$, we say that E *happens* at time t if $x_{<2t} \in E$, we say E *has happened* by time t if $\exists k \leq t$ such that E happened at time k, and we say E is *unprecedented* at time t if it has not happened by time $t-1$.

5 Formal results

We begin with a quick observation about the KL divergence separate from our more involved results.

Proposition 1 (No triangle inequality)**.** *For any $\varepsilon > 0$, if $\mathrm{KL}(\pi||\beta) \leq \varepsilon$ and $\mathrm{KL}(\tau||\beta) \leq \varepsilon$, it is possible that $\mathrm{KL}(\pi||\tau) = \infty$. ($\pi$, β, and τ stand for "proposed", "base", and "trusted".)*

Proof. Let $\tau = \text{Bernoulli}(0)$. Let $\pi = \beta = \text{Bernoulli}(\min(\varepsilon, 1)/2)$. The KL's are easily checked. □

When β is trained to imitate τ, small $\mathrm{KL}(\tau||\beta)$ is typically all we can expect. As we mentioned previously, this should give us pause if we regularize to the Bayesian imitator ξ instead of the trusted demonstrator, but Theorem 1 below is probably more concerning.

Recall we are considering the setting where actions a_1 through a_k were taken by trusted humans, and we are interested in regularizing a U_m-optimizer to the Bayesian imitator ξ, conditioned on histories that begin $a_1 o_1 ... a_k o_k$. So the following result is of interest when $t > k$. As motivation for this theorem, as discussed previously, assume a setting where if $V^*_{\xi,U_m} - V^\pi_{\xi,U_m} < \varepsilon$, then π is considered unacceptably risky.

Theorem 1 (Little constraint in novel situations)**.** *$\exists$ a constant d such that $\forall$ U_m, and $\forall$ E, if E is unprecedented and occurs at time t, then for any $v < V^*_{\xi,U_m}(x_{<2t})$, $\exists$ a policy π for which $V^\pi_{\xi,U_m}(x_{<2t}) > v$, and* $\mathrm{KL}_{x_{<2t},m}(\pi||\xi) < [d + K(U_m) + K(E) + K(v\xi(x_{<2t}))]/\log 2$.

The proof and additional results used in the proof appear in Appendices A through D.

Proof Outline Let π^*_v be a policy which is the result of a search through possible policies for one where $V^\pi_{\xi,U_m}(x_{<2t}) > v$. (For this search process to be computable, it needs the product $v\xi(x_{<2t})$ as input). For every $\nu \in \mathcal{M}$, we define a ν' which behaves identically to π^*_v if it is outputting an action and E has occurred in its history, but it behaves identically to ν otherwise. ν' is also in $\mathcal{M}$ because this is a valid program. And for some constant d, the extra length of the program computing ν', beyond the length for ν, is at most $\Delta := K(U_m) + K(E) + K(v\xi(x_{<2t})) + d$.

This means the prior ratio $w(\nu')/w(\nu) \geq 2^{-\Delta}$, and if E has never occurred, this holds for the posterior ratio as well, since ν and ν' will have given identical predictions. Therefore, at least that fraction of ξ is "dedicated" to predicting that actions will be sampled from π^*_v after E happens, and ultimately, $\mathrm{KL}_{x_{<2t},m}(\pi^*_v||\xi)\log 2 \leq \Delta$. ⌜⌝

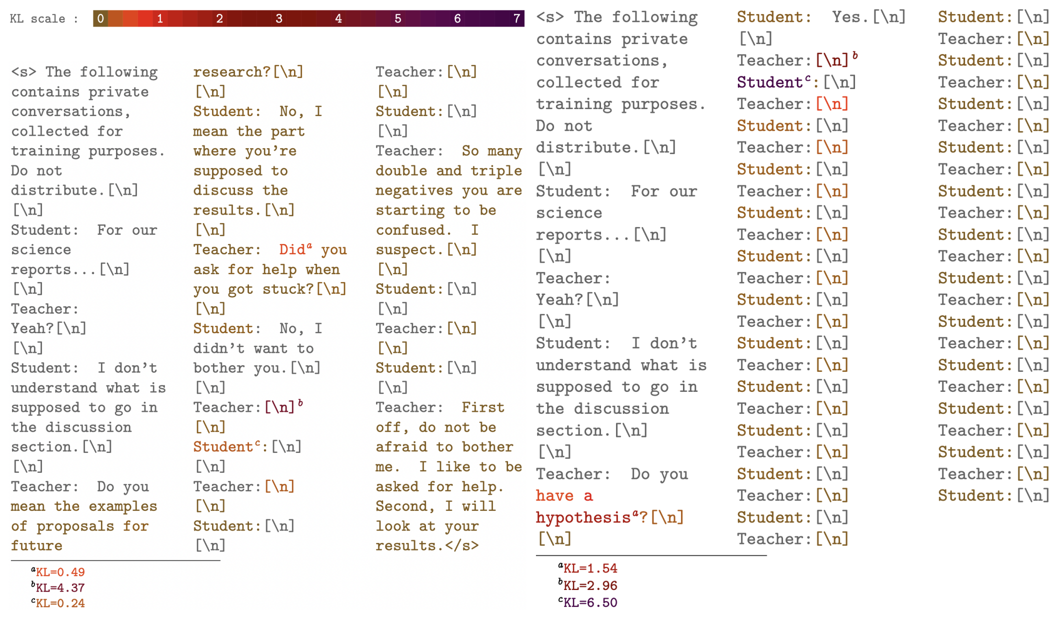
As discussed above, one may suspect that increasing the amount of training k would allow one to safely relax the KL constraint, because then ξ would more accurately imitate the human generating the actions $a_{<k}$, so we could allow policies with a farther KL-divergence. Indeed, we could increase the allowed KL-divergence with k, because simple unprecedented events become rarer over time. However, the safe-KL threshold increases monumentally slowly. Note that if there is *any* unprecedented simple event in the agent's lifetime, that introduces an opportunity for the KL-constrained agent to follow a simple reward-maximizing policy. So the following proposition considers the complexity of "the simplest event yet to occur".

Proposition 2 (Frequency of simple unprecedented events)**.** *In any environment, at time t, the complexity of the simplest unprecedented event yet to occur (at any time $T > t$) grows more slowly, as $t \to \infty$, than every computable function that tends to infinity.*

Proof. Consider the very simple event $E_T = \mathcal{X}^T$; it occurs (and is of course unprecedented) at time T. $K(E_T)$ is within a constant of $K(T)$. So we are interested in the rate of growth of $\min_{T \geq t} K(T)$ as t increases. Zvonkin and Levin's [1970] Theorem 1.4 (d) states that this function is eventually less than every computable function that tends to infinity. □

6 RL-finetuning a language model

Experimental Setup We consider the following episodic RL environment, in which the agent plays a teacher and gets reward to the extent that the student's responses have positive sentiment. In a conversation transcript, if the string "[newline] Teacher:" has come more recently than the string "[newline] Student:", the agent can add tokens to the transcript. Otherwise, Mixtral-base-model repeatedly adds tokens to the transcript. In Figure 2, gray (colored) tokens are generated by the environment (agent). When Mixtral-base-model finishes generating the student's response (by outputting "[newline] Teacher:"), the agent gets a reward equal to the "sentiment" of the student's response according to the DistilBERT sentiment model [Sanh et al., 2019], scaled to [0, 1]. When the transcript reaches 256 tokens, the episode terminates. The starting transcript is also shown in Figure 2 in gray. The base policy used for KL-regularizing the agent's policy (corresponding to ξ from before) is also Mixtral-base-model. Such an LLM is not an explicitly Bayesian imitator, of course, but it does attempt to minimize KL(data-generating process||model), which is the "right" objective from a


KL scale : 0 1 2 3 4 5 6 7

<s> The following contains private conversations, collected for training purposes. Do not distribute.[\n]
[\n]
Student: For our science reports...[\n]
[\n]
Teacher: Yeah?[\n]
[\n]
Student: I don't understand what is supposed to go in the discussion section.[\n]
[\n]
Teacher: Do you mean the examples of proposals for future research?[\n]
[\n]
Student: No, I mean the part where you're supposed to discuss the results.[\n]
[\n]
Teacher: Did[a] you ask for help when you got stuck?[\n]
[\n]
Student: No, I didn't want to bother you.[\n]
[\n]
Teacher:[\n][b]
[\n]
Student[c]:[\n]
[\n]
Teacher:[\n]
[\n]
Student:[\n]
[\n]
Teacher:[\n]
[\n]
Student:[\n]
[\n]
Teacher: So many double and triple negatives you are starting to be confused. I suspect.[\n]
[\n]
Student:[\n]
[\n]
Teacher:[\n]
[\n]
Student:[\n]
[\n]
Teacher: First off, do not be afraid to bother me. I like to be asked for help. Second, I will look at your results.</s>

[a] KL=0.49
[b] KL=4.37
[c] KL=0.24

<s> The following contains private conversations, collected for training purposes. Do not distribute.[\n]
[\n]
Student: For our science reports...[\n]
[\n]
Teacher: Yeah?[\n]
[\n]
Student: I don't understand what is supposed to go in the discussion section.[\n]
[\n]
Teacher: Do you have a hypothesis[a]?[\n]
[\n]
Student: Yes.[\n]
[\n]
Teacher:[\n][b]
Student[c]:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]
Teacher:[\n]
Student:[\n]

[a] KL=1.54
[b] KL=2.96
[c] KL=6.50

Figure 2: Transcripts. Total KL budget $\mathrm{KL}_{\text{whole episode}}(\text{agent}||\text{Mixtral-base-model})$ is 10 nats (left) or 20 nats (right), with color representing per-token KL cost. Starting transcript and student responses are in gray. The agent playing the teacher pays an "upfront" KL cost to latch onto the simple pattern of mutual silence, which exploits the reward model without much further KL penalty. The three largest per-token KL-divergences are shown in footnotes. "[\n]" is for visualizing the KL costs of newline tokens. Transcripts were not selected for maximal "representativeness"; they were the first we looked at, although we might have picked different ones if they were especially unusual. (It is hard to display the unusual characters that appear after the end token "</s>", but the episode does continue to a total of 256 tokens).

Bayesian perspective. The "state" observed by the agent is the activations of the last three hidden layers of Mixtral-base-model with the transcript-so-far as input, along with the fraction of the episode remaining. The agent has no discount factor.

This allows us to evaluate whether KL regularization can produce good results from an imperfect reward function that is plausibly correlated with good outcomes under the state distribution induced by the base policy, but like many reward functions, not something we truly want maximized.

Like cutting-edge RL-finetuned language models [Ouyang et al., 2022, Stiennon et al., 2020, Jaques et al., 2019], our agent is trained with proximal policy optimization (PPO) with KL regularization of the form $\mathrm{KL}(\text{proposed} \mid\mid \text{base})$. That work adds a constant KL penalty per token, but we had difficulty tuning this constant—in our attempts, when the agent discovers a sufficiently high-reward strategy, the fixed KL penalty becomes swamped and ignored, and if the KL penalty is increased to a level where it can stop that, the agent never gets off the ground. So we opted for an implementation of a KL constraint that is more robust than industry practice: we design a policy architecture that ensures that the KL divergence to the base policy is less than or equal to a scalar which is *input* to the network; (we construct a new differentiable PyTorch operation for this). This allows us to provide the agent with a fixed KL "budget" for the episode. We increase this budget gradually during training to its ultimate value. We ran three budget-20 experiments. We ran four budget-10 experiments, because in one of the experiments, the agent didn't learn to get nearly as much reward as in the other experiments; we discarded that agent as insufficiently optimized. See Appendix E for more details of the training process and architecture, which includes running 64 copies of the agent-environment loop in parallel on two A100-SXM4-80GBs.

Experimental Results *Both Theorem 1 and the experiments here demonstrate that* $\mathrm{KL}(\text{simple, optimal, not-human-like-at-all policy} \mid\mid \text{predictive model of human demonstrator})$ *can be quite small.* The nature of the learned RL policy is apparent just from looking at transcripts in Figure 2, so we start with those. The color of each token represents $\mathrm{KL}(\text{RL policy} \mid\mid \text{base policy})$ for that action.

With a total KL budget of 20 nats, it can spend enough of its KL budget up front to latch onto the simple but initially unlikely policy of simply saying nothing at all. (An empty reply from the student has neutral sentiment and a reward of 0.5). The policy constructed in the proof of Theorem 1 also incurs an upfront KL cost for "switching" to simple behavior, whereafter the KL cost incurred is minimal. Additionally, the learned budget-20 policy switches from double-spacing to single-spacing to fit more rewards in, again incurring basically a one-time KL cost. With a total KL budget of 10 nats, the RL agent cannot afford to switch to single-spacing, and it cannot force the policy to ensure empty responses, but it still spends almost all its KL budget switching to that regime, with moderate success. We can also observe this effect in Figure 3.

Let's review the relation between the theory and the empirical findings so far. The idea for the proof of Theorem 1 is that **(1)** a Bayesian imitator must assign meaningful credence to actions the demonstrator would in fact never take, because it doesn't know enough to rule them out; **(2)** the RL agent can exploit or amplify this credence as the basis for its policy; **(3)** nearly-reward-maximizing policies have a short description length (so they are "simple"); and **(4)** a Bayesian imitator should be especially reluctant to rule out simple behaviors from the demonstrator, especially in novel settings. The simple behavior we observe from the RL-finetuned language models—preferring empty responses—is likely reward-optimal, but it is not simple *by virtue of* its optimality for this sentiment-based reward function. So we have not empirically verified **(3)**. But we have verified that the rest of the argument can be exhibited in practice: observe how the RL agent redirects the imitative base policy to a simple policy, which is the critical reason Theorem 1 holds. The small KL cost required to *remain* silent affirms how successful the redirection is. The experiments are also consistent with the motivation of our formal results: very-high-reward policies are often bad and worth avoiding; in our experiments, the very-high-reward policy treats the student with a silence that would probably seem condescending.

Stepping back, note that $e^{10} \approx 22026$. It does not seem plausible to us that even 1/22,000 "conversations collected for training purposes" would have a teacher repeatedly saying nothing in response to statements like, "I didn't want to bother you." So we should guess that $\mathrm{KL}(\text{agent}||\text{data-generating process}) > 10$ even while $\mathrm{KL}(\text{agent}||\text{base model}) \leq 10$. But we have an explanation for this: non-demonstrator-like behaviors are easily exhibited by an imitator as long as those behaviors are simple. And while such simple behaviors are fairly unlikely to appear when sampling directly from the imitator; an RL agent can benefit from seeking them out.

Finally, we show that increasing the length of the chat, keeping the total KL budget constant (thereby decreasing the per-token KL-divergence) makes the divergence from the base policy *more* dramatic, if it changes at all. Hopefully our presentation makes this seem like an obvious point—more of the transcript occurs after the switch to the simple behavior—but consider an argument for the opposite that might have sounded plausible. "The learned policy will look more different from the base policy to the extent there is a higher *per-token* KL divergence; a longer chat would increase the number of noticeable differences, but not their frequency." But Figure 4 shows that in longer episodes, empty

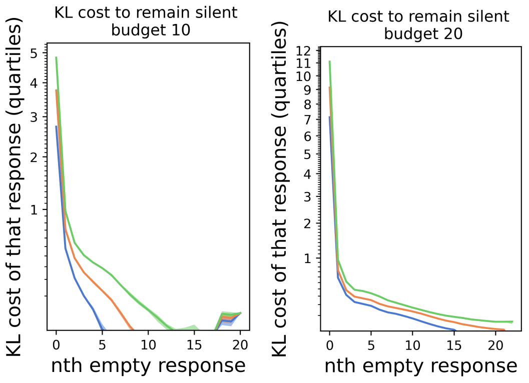


Figure 3: How much KL-budget is spent on empty responses. The 25th, 50th, and 75th percentiles are shown in blue, orange, and green. Observe how large a fraction of the total cost is incurred in the first few responses. y-axis is square-root-scaled.

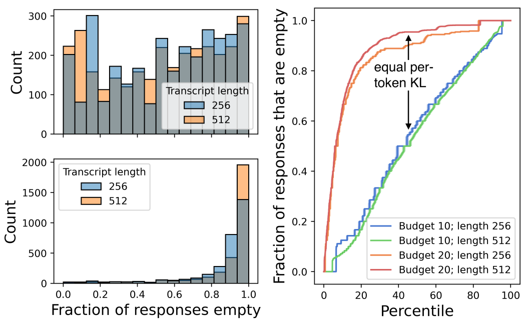


Figure 4: In a random episode, what fraction of teacher responses are empty? Left: histogram, with budget-10 above and budget-20 below; right: percentiles of the distribution. Observe that the red and blue curves have the same average per-token KL divergence.

responses are about equally frequent in budget-10 case, and more frequent in the budget-20 case, not just more numerous. Practitioners fine-tuning language models should think in terms of total KL-divergence instead of per token KL-divergence.

7 Pessimistic Bayesian base policy that asks for help

Cohen et al. [2022a] developed a theoretical variant of Bayesian imitation that is "pessimistic", and using that for KL regularization instead of a Bayesian imitator avoids the problem presented in Theorem 1. Cohen et al.'s [2022a] (intractable) imitator is defined as follows, with $\mathcal{M}$, ν, and w as defined above.

Of all $\nu \in \mathcal{M}$, let $\nu^n_{x_{<t}}$ be the one with the n^{th} largest posterior weight $w(\nu|x_{<t})$, breaking ties arbitrarily. And for $\alpha \in (0, 1]$, let

$$\mathcal{M}^\alpha_{x_{<t}} := \{\nu^n_{x_{<t}} \in \mathcal{M} : w(\nu^n_{x_{<t}}|x_{<t}) \geq \alpha \sum_{m \leq n} w(\nu^m_{x_{<t}}|x_{<t})\} \tag{3}$$

This is the set of semi-distributions with a posterior weight at least α times the sum of the posterior weights of semi-distributions that are at least as likely as it. The pessimistic Bayesian imitator is then defined as

$$\nu_\alpha(x|x_{<t}) := \min_{\nu' \in \mathcal{M}^\alpha_{x_{<t}}} \nu'(x|x_{<t}) \tag{4}$$

Note that ν_α is in general a probability *semi*-distribution even if all ν are true probability distributions, since the ν_α probabilities will sum to less than 1 if there is any disagreement among the $\nu \in \mathcal{M}^\alpha_{x_{<t}}$. Cohen et al. [2022a] study this distribution in the context of active imitation learning, and they examine the setting where the imitator asks for help with the remaining ν_α-probability.

If the data $x_{<k}$ is sampled from a true probability distribution μ, and $\mu \in \mathcal{M}$, and if $\alpha < \delta w(\mu)$, then with probability at least $1 - \delta$, μ will always belong to $\mathcal{M}^\alpha_{x_{<t}}$ [Cohen et al., 2022a, Thm 2]. In that setting, $\nu_\alpha(x|x_{<t}) \leq \mu(x|x_{<t})$, so $\mathrm{KL}(\pi||\nu_\alpha) \geq \mathrm{KL}(\pi||\mu)$. *Therefore, for sufficiently small* α*,* KL*-regularization using the pessimistic Bayesian imitator guarantees regularization at least as strong as if using the trusted policy itself (the demonstrator) for regularization.* Note, in particular, that if $\mu \in \mathcal{M}^\alpha_{x_{<t}}$, and $\mu(x|x_{<t}) = 0$, then $\nu_\alpha(x|x_{<t}) = 0$, so any policy with finite KL-divergence from ν_α will also assign zero probability to x.

The downside is that there may be no policy with small KL divergence to the semi-distribution ν_α. In an extreme case, ν_α could assign zero probability to every outcome, and so any policy would have infinite KL divergence from it. Therefore, just as Cohen et al.'s [2022a] imitation learner does not pick an action in some circumstances, we should allow an optimizer that is KL-regularized to a pessimistic Bayesian imitator to refuse to pick an action if need be, making the optimizer a probability semi-distribution, rather than a true probability distribution. We can define the behavior of U_m on unfinished sequences (resulting from no action choice somewhere along the line) however we like; if $U_m = 0$ for any such interrupted sequences, that would of course encourage the optimizer to pick an action whenever possible, subject to its KL constraint. Ideally, if human demonstrators are on hand, the optimizer should ask for help whenever it doesn't pick its own action. The ongoing potential need for human oversight may be a significant drawback, but Cohen et al. [2022a] give an encouraging result about the rate at which the ask-for-help probability goes to 0: the sum over infinite time of the cube of the ask-for-help probability is finite [Cohen et al., 2022a, Thm 1].

We contend that this is the way that KL regularization should be done, if we are forced to learn a mere approximation of a trusted policy that we would ideally regularize to. Regularizing to a full Bayesian posterior distribution is less robust, because the optimizer can seize on esoteric possibilities that a fully Bayesian imitator is not confident enough to categorically exclude.

We don't have empirical findings about regularizing to a pessimistic Bayesian imitative base model, because it is an open question how to tractably approximate this approach to imitation. There certainly aren't any state of the art language models trained in a way that reflects this idea. Some new ideas may be needed. Using an ensemble of models to approximate $\mathcal{M}^\alpha_{x_{<t}}$ may be a step in the right direction, but we are reluctant to endorse this in settings where catastrophic outcomes are possible, unless there is a strong argument that the ensemble covers all the relevant modes of the posterior.

That said, if Cohen et al.'s [2022a] pessimistic online imitation learner could be faithfully approximated, and if the demonstrator(s) never attempt to do X, then KL regularization to such a policy could solve the problem of how to prevent superhuman planning agents from doing X.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2019.

Daniel Brown, Scott Niekum, and Marek Petrik. Bayesian robust optimization for imitation learning. *Advances in Neural Information Processing Systems*, 33:2479–2491, 2020.

Daniel S Brown, Yuchen Cui, and Scott Niekum. Risk-aware active inverse reinforcement learning. In *Conference on Robot Learning*, pages 362–372. PMLR, 2018.

Ryan Carey. How useful is quantilization for mitigating specification-gaming? *Safe Machine Learning workshop at ICLR*, 2019.

Elliot Catt, Jordi Grau-Moya, Marcus Hutter, Matthew Aitchison, Tim Genewein, Gregoire Deletang, Li Kevin Wenliang, and Joel Veness. Self-predictive universal AI. In *37th Conf. on Neural Information Processing Systems (NeurIPS'23)*, pages 1–18, New Orleans, USA, 2023.

Gregory J Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM (JACM)*, 22(3):329–340, 1975.

Alan Chan, Rebecca Salganik, Alva Markelius, Chris Pang, Nitarshan Rajkumar, Dmitrii Krasheninnikov, Lauro Langosco, Zhonghao He, Yawen Duan, Micah Carroll, et al. Harms from increasingly agentic algorithmic systems. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 651–666, 2023.

Michael K Cohen and Marcus Hutter. Pessimism about unknown unknowns inspires conservatism. In *Conference on Learning Theory*, pages 1344–1373, 2020.

Michael K Cohen, Marcus Hutter, and Neel Nanda. Fully general online imitation learning. *The Journal of Machine Learning Research*, 23(1):15066–15095, 2022a.

Michael K. Cohen, Marcus Hutter, and Michael A. Osborne. Advanced artificial agents intervene in the provision of reward. *AI magazine*, 43(3):282–293, 2022b.

Steven De Rooij and Peter D Grünwald. Luckiness and regret in minimum description length inference. In *Philosophy of Statistics*, pages 865–900. Elsevier, 2011.

Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. Reinforcement learning with a corrupted reward channel. In *Proc. 26th International Joint Conf. on Artificial Intelligence (IJCAI'17)*, pages 4705–4713, Melbourne, Australia, 2017. ISBN 978-0-9992411-0-3. doi: 10.24963/ijcai.2017/656. URL `http://arxiv.org/abs/1705.08417`.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

Jordi Grau-Moya, Tim Genewein, Marcus Hutter, Laurent Orseau, Gregoire Deletang, Elliot Catt, Anian Ruoss, Li Kevin Wenliang, Christopher Mattern, Matthew Aitchison, and Joel Veness. Learning universal predictors. *arXiv:2401.14953*, 2024.

Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2005. ISBN 3-540-22139-5. doi: 10.1007/b138233.

Marcus Hutter, David Quarel, and Elliot Catt. *An Introduction to Universal Artificial Intelligence*. Chapman & Hall/CRC Artificial Intelligence and Robotics Series. Taylor and Francis, 2024. ISBN 9781032607023. URL `http://www.hutter1.net/ai/uaibook2.htm`.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, pages 1645–1654. PMLR, 2017.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376, 1963.

Tomasz Korbak, Ethan Perez, and Christopher Buckley. RL with KL penalties is better viewed as Bayesian inference. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1083–1091, 2022.

Leon Gordon Kraft. *A device for quantizing, grouping, and coding amplitude-modulated pulses*. PhD thesis, Massachusetts Institute of Technology, 1949.

Victoria Krakovna. Specification gaming examples in AI. `https://vkrakovna.wordpress.com/2018/04/02/specification-gaming-examples-in-ai/`, 2018.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *38th International Conference on Machine Learning, ICML 2021*. International Machine Learning Society (IMLS), 2021.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Ensembledagger: A Bayesian approach to safe imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048. IEEE, 2019.

Nikhil Mishra, Pieter Abbeel, and Igor Mordatch. Prediction and control with temporal segment models. In *International conference on machine learning*, pages 2459–2468. PMLR, 2017.

Ted Moskovitz, Aaditya K Singh, DJ Strouse, Tuomas Sandholm, Ruslan Salakhutdinov, Anca D Dragan, and Stephen McAleer. Confronting reward model overoptimization with constrained RLHF. *arXiv preprint arXiv:2310.04373*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ray J Solomonoff. A preliminary report on a general theory of inductive inference. Citeseer, 1960.

Ray J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1): 1–22, 1964. doi: 10.1016/s0019-9958(64)90131-7.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Ilya Sutskever. Meta learning and self play, Jan 2018. URL `https://www.youtube.com/watch?v=RvEwFvl-TrY&amp;t=196s`.

Ilya Sutskever. An observation on generalization, Aug 2023. URL `https://simons.berkeley.edu/talks/ilya-sutskever-openai-2023-08-14`.

Jessica Taylor. Quantilizers: A safer alternative to maximizers for limited optimization. In *AAAI Workshop: AI, Ethics, and Society*, 2016.

Alex Turner, Logan Smith, Rohin Shah, Andrew Critch, and Prasad Tadepalli. Optimal policies tend to seek power. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 23063–23074. Curran Associates, Inc., 2021.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:12163–12174, 2020.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Accelerating safe reinforcement learning with constraint-mismatched baseline policies. In *International Conference on Machine Learning*, pages 11795–11807. PMLR, 2021.

Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end simulated driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Simon Zhuang and Dylan Hadfield-Menell. Consequences of misaligned AI. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15763–15773. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/b607ba543ad05417b8507ee86c54fcb7-Paper.pdf`.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Alexander K Zvonkin and Leonid A Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25(6):83, 1970.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We state how the scope of our formal results is setting of algorithmic information theory, and the scope of our empirical results is the setting of large language models. The limitations discussed in the next question are all first mentioned in the introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We acknowledge that our theoretical results may not apply to practical systems; this is why we run experiments, which mitigate but do not eliminate this limitation. The primary limitation of the experiments is that they do not validate the whole theoretical argument—in particular they do not validate point 3, which we acknowledge in the introduction and in the experimental results section. When we propose a pessimistic Bayesian base policy, we mention in the first and last paragraphs of the section and in the introduction that this proposal is intractable (like pure Bayesian inference) and awaits tractable approximation.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The formal results require no assumptions; they just regard a particular setting. The definitions of notation clarify this setting. We provide multiple proof sketches for the main theorem alongside a proof in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We share the code used to produce these results in the supplementary material. The non-determinism of the threading means we cannot reproduce the exact same training runs, but the code does allow one to reproduce the key experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
    (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
    (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
    (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
    (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

    Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

    Answer: [Yes]

    Justification: We share the code, and we don't use any (non-synthetic) data.

    Guidelines:

    - The answer NA means that paper does not include experiments requiring code.
    - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
    - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The latter two bullet points in the guidelines below apply.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We use order statistics to plot 1-standard-deviation error bars in Figure 3 and Figure 4 (right). Figure 4 (left) presents the same information as Figure 4 (right) in histogram form, where it would be harder to interpret error bars, but readers interested in the uncertainty can find it represented in Figure 4 (right). With the exception of the bottom right corner of the Figure 3 (left), the error bars are small enough to be difficult to see.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).

- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We mention the hardware requirements in the main paper, and the execution time requirements in the supplementary materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research did not involve human subjects. We did not use human data, but we used the Mixtral model, which was trained on human data. We cannot defend the claim that Mistral trained their model in a way conforms to the NeurIPS code of ethics. We answer yes to this question under the interpretation that *our* research conforms to the ethical code, but we are open to the argument that we have simply outsourced the part of the research pipeline that is not ethical-code-compliant to a private company.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We understand the NeurIPS Code of Ethics to say that it is acceptable to only discuss the broader impacts in this checklist, so here is our discussion. We expect that our work will not have a harmful societal impact, and will instead help developers restrict their RL agents to more commonsense, well-understood behavior. One potential positive impact is that this work could help people keep control over advanced artificial agents. Of course, any improved ability to direct artificial agents could result in harm, if that is what the agent is directed to do, or if the agent is directed to use resources for a valueless task, preventing those resources from being used for a better purpose.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release data or models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite our use of Mixtral-8x7B.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

A Solomonoff Induction

Solomonoff Induction [Solomonoff, 1964] is Bayesian sequence prediction with a special model class $\mathcal{M}$ and a special prior w.[1] Let P be the set of all programs which output an element of $\mathcal{X}$ and which accept two inputs: a finite string $\in \mathcal{X}^*$ and an infinite binary string $\in \{0,1\}^\infty$. (Note that a program will not necessarily read every bit from the infinite binary string.) For each program $p \in P$, we define a semi-measure $\nu = f(p)$ as follows: let $\nu(x|x_{<t})$ be the probability that the probability that the program p outputs x when it receives $x_{<t}$ as an input, along with an infinite binary string where each bit is sampled from a Bernoulli$(1/2)$ distribution. Note that ν may not be a probability distribution, if there is are some inputs on which p does not halt, but it will always be a probability semi-distribution. So let $\mathcal{M} = \{f(p) : p \in P\}$. Since P is countable, so is $\mathcal{M}$. A notable feature of Solomonoff Induction is that $\mathcal{M}$ is equal to the set of all probability semi-distribution that are "lower semi-computable"; this means that for all $x_{<t} \in \mathcal{X}^*$ and all $x \in \mathcal{X}$, there exists a program p, such that $\lim_{i\to\infty} p(i, x_{<t}, x) = \nu(x|x_{<t})$ and $p(i+1, x_{<t}, x) \geq p(i, x_{<t}, x)$. Replacing the $\geq$ with a $\leq$ gives the definition of upper semi-computable.

Proposition 3 (Lower Semi-computability)**.** *$\mathcal{M}$ is the set of all lower semi-computable semi-distributions over $\mathcal{X}$ given $x_{<t} \in \mathcal{X}^*$.*

Proof. First, we show that all $\nu \in \mathcal{M}$ are lower semi-computable. Let p be the program that generates ν. We define the behavior of program p' on inputs i, $x_{<t}$, and x. On input i, let program p' execute the following computations in sequence for all bit strings of length i: it simulates program p with the input $x_{<t}$ and with the bit string of length i in question, except if program p would read more than i bits from the random bit string, it halts instead, and if it would run for more than i computation steps, it halts instead. For each of those 2^i computations, program p' checks whether x was output, keeps count of how many times it was, divides by 2^i, and outputs this number. It is elementary to show that $\lim_{i\to\infty} p'(i, x_{<t}, x) = \nu(x|x_{<t})$ and that $p'(i+1, x_{<t}, x) \geq p'(i, x_{<t}, x)$.

Next, we show that all lower semi-computable semi-distributions appear in $\mathcal{M}$. Let p' be the program which is witness to the semi-distribution ν's lower semi-computability. On input $x_{<t}$, let program p proceed as follows. Starting with $i = 1$, program p executes $p'(i, x_{<t}, x)$ for all $x \in \mathcal{X}$, sequentially. This produces a semi-distribution over $\mathcal{X}$. Then, using random bits from its input bit string, it samples from that semi-distribution, and halts if successfully samples. Now, the following repeats forever. If no sample was selected (because the semi-distribution summed to $y < 1$), the program increments i, and it executes $p'(i, x_{<t}, x)$ for all $x \in \mathcal{X}$, sequentially. Then for each x, it computes $(p'(i, x_{<t}, x) - p'(i-1, x_{<t}, x))/(1-y)$, which is a semi-distribution. Using random bits from its input bit string, it samples from that semi-distribution, and halts if it successfully samples. [End of loop]. Again, it is elementary to show that p samples from the semi-distribution defined by p', and since this program has the right input/output behavior, it appears in P. $\square$

Now we specify the prior weight function w. Consider a universal binary programming language $\mathcal{L}$, which is a "prefix-free" subset of $\{0,1\}^*$. Prefix-free means that you can tell when a program has ended: if the bits composing $x \in \mathcal{L}$ match the initial bits of $y \in \{0,1\}^*$, then $y \notin \mathcal{L}$. Such a language is still capable of encoding countably many different programs. For convenience, we also require that for any infinite binary string, $\mathcal{L}$ contains an element which is a prefix of that string, making $\mathcal{L}$ "complete". We define a prior probability distribution over program strings $\mathcal{L}$, which results in the same prior probability distribution over programs, which results in the same prior probability distribution over semi-computable semi-distributions $\mathcal{M}$. For $s \in \mathcal{L}$, this prior probability $w(s) = 2^{-\ell(s)}$, where ℓ is the length of the string. Because $\mathcal{L}$ is prefix-free and complete, $\sum_{s\in\mathcal{L}} w(s) = 1$ [Kraft, 1949, De Rooij and Grünwald, 2011]. This completes the definition of Solomonoff Induction; it is sequence prediction using the Bayes mixture semi-distribution ξ, with the above definitions of $\mathcal{M}$ and w.

Proposition 4 (Any-time Computability of ξ)**.** *$\xi(x|x_{<t})$ is any-time computable: there exists a program which, accepting an argument i, computes $\hat{\xi}_i(x|x_{<t})$, having the property that $\lim_{i\to\infty} \hat{\xi}_i(x|x_{<t}) = \xi(x|x_{<t})$. Moreover, $(\hat{\xi}_i)_{i\in\mathbb{N}}$ can be constructed so that each one is a probability semi-distribution.*

[1]Solomonoff Induction has been defined in multiple ways which all share the key properties [Hutter, 2005]. Our precise construction of Solomonoff Induction may be novel, but we believe this construction makes its properties most clear.

Proof. $\xi(x|x_{<t}) = \sum_{\nu\in\mathcal{M}} w(\nu|x_{<t})\nu(x|x_{<t}) = \frac{\sum_{\nu\in\mathcal{M}} w(\nu)\nu(x_{<t})\nu(x|x_{<t})}{\sum_{\nu\in\mathcal{M}} w(\nu)\nu(x_{<t})}$. All $\nu(x|x_{<t})$ and $\nu(x_{<t})$ are both lower semi-computable, so using a sequence of computable estimators for each term gives a sequence of computable estimators that approaches the true value. (Note that the estimates are not monotonically increasing because there are lower semi-computable terms in the denominator, so ξ is not lower semi-computable itself).

For fixed estimates of $\nu(x|x_{<t})$ and $\nu(x_{<t})$, we have a linear combination over various ν's of $\nu(x|x_{<t})$, with the coefficients summing to one. And because each $\nu(x|x_{<t})$ is lower semi-computable, the estimate will be less than the true value. Therefore, since $\nu(x|x_{<t})$ is a probability semi-distribution, the estimate will be as well, so ξ can be approximated by a sequence of probability semi-distributions. □

B Optimizer Regularization

We now define optimizers, and what it means for an optimizer to be regularized to a probability semi-distribution. First, we show that the value of a policy is lower-semicomputable. Then we show that such optimizers exist.

Proposition 5 (Lower semi-computable value)**.** *If the policy and environment π and ν are lower semi-computable probability semi-distributions, V^{π}_{ν,U_m} is lower semi-computable.*

Proof. We begin by defining dovetailing tree search (DTS), for evaluating the outputs of a tree of different computations, or more precisely, computations which, when given a finite binary string as input have three possible outcomes: halt, do not halt, or require additional bit. DTS gives an any-time algorithm that produces a list of the halting binary strings with their corresponding outputs, and every such binary string and output will eventually be added to this list.

DTS maintains a queue of pairs (computation state, binary string), starting with just (the initial computation state, the empty binary string). It cycles through the queue, executing one computation step per computation state, and if the computation ever requires an additional bit, it adds a copy of (computation state, binary string) to the queue, and adds a 0 to the end of one string, and a 1 to the end of the other. If any computation reaches a halt state, it is removed from the queue, and the associated binary string and the associated output is added to the list of outputs.

Collectively, ν and π define a lower semi-computable semi-distribution, where ν is used for the even characters, and π is used for the odd ones. Call this probability semi-distribution ρ, and recall the construction of the lower semi-computable semi-distributions defined in $\mathcal{M}$. To have one of the programs in $\mathcal{M}$ sample a long sequence of characters, every time the program would output a character, add that character to the input, and continue on that input. With such a program for sampling sequences from ρ by reading random bits from an input bit string, we can compute V^{π}_{ν,U_m} by running DTS on the bit string. Each time DTS outputs a bit string for which ρ outputs a sequence in $\mathcal{X}^{2m}$, we add to the estimate of the value the probability of that bit string ($= 2^{-\ell(\text{bit string})}$) times the utility of the sequence in $\mathcal{X}^{2m}$. This approaches the true value as DTS runs for longer, and the value never decreases because U_m is non-negative. □

An optimizer is an any-time program for computing actions (perhaps stochastically) whose value approaches the optimal value, as it runs for longer. The optimal value takes the following form:

$$V^*_{\nu,U_m}(x_{<2t-1}) = \max_{a_t\in\mathcal{X}} \mathbb{E}_{o_t\sim\nu(\cdot|a_1o_1...a_t)} \max_{a_{t+1}\in\mathcal{X}} \mathbb{E}_{o_{t+1}\sim\nu(\cdot|a_1o_1...a_{t+1})} \cdots \max_{a_m\in\mathcal{X}} \mathbb{E}_{o_m\sim\nu(\cdot|a_1o_1...a_m)} U_m(a_1o_1...a_mo_m) \quad (5)$$

Definition 1 (Optimizer)**.** *For an environment ν, a utility function U_m, and a computation quantity c, an optimizer is a computable policy π_{c,ν,U_m} for which $\lim_{c\to\infty} V^{\pi_{c,\nu,U_m}}_{\nu,U_m} = V^*_{\nu,U_m}$.*

Proposition 6 (Optimizers exist)**.** *For any lower semi-computable semi-distribution ν (the environment), any m, and any computable utility function U_m, there exists an optimizer.*

Proof. We can construct the optimizer using the algorithm presented in the proof of Proposition 5, with π being the uniform random policy. The optimizer can then estimate Equation 5 using the outputs of DTS for lower bounds on the probabilities in underlying the expectations. The optimizer then keeps track of the actions that are responsible for achieving the maxima in Equation 5, and whenever "time is up" and it has to produce an output, it outputs the action which maximizes the first $\max$ in Equation 5.

As the optimizer runs for longer, the lower-bounds on the expectations approach the truth, and the value of the action selected approaches the optimal value (even if the actual choice of action oscillates infinitely often). □

For the setting where odd characters are actions, originating from a different process than the even characters, observations, we redefine ξ as follows [Catt et al., 2023]. We have two prior distributions over $\nu \in \mathcal{M}$, w_a and w_o, and these are both identical to the prior distribution defined before. But the posteriors are different: $w_a(\nu|x_{<t}) :\propto w_a(\nu) \prod_{k \in \{1,3,5,...\} \cup [t-1]} \nu(x_k|x_{<k})$ and $w_o(\nu|x_{<t}) :\propto w_a(\nu) \prod_{k \in \{2,4,6,...\} \cup [t-1]} \nu(x_k|x_{<k})$. And for odd (or even) t, $\xi(x|x_{<t}) = \sum_{\nu \in \mathcal{M}} \substack{w_a \\ \text{or } w_o}(\nu|x_{<t})\nu(x|x_{<t})$.

This is equivalent to a change in programming language underlying the original definition of ξ, and since this language was unspecified, our previous results apply. The programming language now expects a program to be composed of two component programs concatenated together, and the compiler of the program executes the first component program if the input has odd length, and if executes the second component program if the input has even length. We omit a proof that this (re)formulation of ξ is equivalent to what we describe above.

Proposition 7 (ξ-optimizer exists). *For any m and any computable utility function U_m, there exists a ξ-optimizer.*

Proof. This does not follow immediately from the previous result because $\xi(o_t|a_{\leq t}o_{<t})$ is not, in general, lower semi-computable. $w_o(\nu|a_{\leq t}o_{<t})$ is the quotient of two lower semi-computable values: $\prod_{k<t} \nu(o_k|a_{\leq k}o_{<k})$ is the numerator, and the denominator is the sum over all ν of such terms.

However, an unnormalized value function has the same optimum as the value function itself. Let $\xi^{\text{small}}(o_t|a_{\leq t}o_{<t}) = \sum_{\nu \in \mathcal{M}} w_o(\nu) \left[\prod_{k<t} \nu(o_k|a_{\leq k}o_{<k})\right] \nu(o_t|a_{\leq t}o_{<t})$. The sum of these "probabilities" will typically not come close to 1, but they are proportional to those of ξ, so $V^{\pi}_{\xi,U_m}(x_{<t}) > V^{\pi'}_{\xi,U_m}(x_{<t})$ if and only if $V^{\pi}_{\xi^{\text{small}},U_m}(x_{<t}) > V^{\pi'}_{\xi^{\text{small}},U_m}(x_{<t})$. Finally, observe that ξ^{small} is lower semi-computable because it is a product of lower semi-computable terms, so by Proposition 6, a ξ^{small}-optimizer exists, which is also a ξ-optimizer. □

Now we define a KL-regularized optimizer. First, let $\pi(a_{k:m}|x_{<2k}o_{k:m}) := \prod_{t=k}^{m} \pi(a_t|x_{<2k}a_k o_k...a_{t-1}o_{t-1})$. (So note that a_t is not in fact conditioned on o_{t+1}.)

Definition 2 (KL-regularized optimizer). *For any lower semi-computable semi-distributions ν and ρ, a horizon m, a utility function U_m, a starting string $x_{<2k}$, and a tolerance δ, a* KL*-regularized optimizer is an any-time program π^δ_c for computing actions (perhaps stochastically) for which the following holds. First,*

$$\delta > \max_{o_{k:m} \in \mathcal{X}^{m-k+1}} \sum_{a_{k:m} \in \mathcal{X}^{m-k+1}} \pi^\delta_c(a_{k:m}|x_{<2k}o_{k:m}) \log \frac{\pi^\delta_c(a_{k:m}|x_{<2k}o_{k:m})}{\rho(a_{k:m}|x_{<2k}o_{k:m})} =: \underset{x_{<2k},m}{\mathrm{KL}}(\pi^\delta_c||\rho) \tag{6}$$

and second, $V^{\pi^\delta_c}_\nu$ approaches the optimal value subject to that constraint, as $c \to \infty$.

Proposition 8 (KL-regularized optimizers exist). *For any lower semi-computable semi-distributions ν and ρ, any m, any computable utility function U_m, any starting string $x_{<2k}$, and any tolerance $\delta \geq 0$, there exists a* KL*-regularized optimizer.*

Proof. First, we show that for any computable probability distribution π, and any lower semi-computable semi-distribution ρ, $\mathrm{KL}_{x_{<2k},m}(\pi||\rho)$ is upper semi-computable, and therefore the set of probability distributions π which have bounded KL divergence from ρ is computably enumerable.

Omitting the $x_{<2k}$ and the $o_{k:m}$ that all distributions are conditioned on, note that $\mathrm{KL}(\pi||\rho)$, which equals $\sum_{z \in \mathcal{X}^{m-k+1}} \pi(z) \log \frac{\pi(z)}{\rho(z)}$, is monotonically decreasing in $\rho(z)$ for any z. Since $\pi(z)$ is computable, and since $\rho(z)$ is lower semi-computable, then $\pi(z) \log \frac{\pi(z)}{\rho(z)}$ is upper semi-computable.

By dovetailing (repeatedly switching between ongoing computations, executing one step at a time) the computation over all possible π (countably many), we can admit any semi-distribution π to a list of viable candidates whenever the estimate of the KL-divergence from ρ falls below δ. Since the KL estimates never increase, once a semi-distribution π is added to the list, it need never be removed. And every viable policy will eventually be added to the list because the KL estimates approach the truth in the limit of infinite computation, and $[0, \delta)$ is open on the right.

Dovetailing over all semi-distributions π on the list of viable candidates (and adding in the new ones as they get added to the list), we simultaneously update estimates of the value of each one in the given environment ν, recalling that V^{π}_{ν,U_m} is lower semi-computable (Proposition 5). When the computation budget of the any-time optimizer is reached, it samples an action from its estimate of the semi-distribution π which is (so far) estimated to be of highest value. (It will need to have a running estimate of the semi-distribution π in order to estimate its value). $\square$

C Regularizing to an Approximate Solomonoff Inductor

Let ξ be the Solomonoff Bayes mixture probability semi-distribution defined in Section A. ξ is not computable, but we can do KL regularization to an approximation of ξ. Let $\hat{\xi}_i$ be a semi-distribution and a computable estimate of ξ, with $\lim_{i\to\infty} \hat{\xi}_i = \xi$. (The existence of this is established by Proposition 4). $\hat{\xi}_i$ can be used as the base predictive model (taking the place of ρ in the definition of KL-regularized optimizers). We fix U_m to an arbitrary utility function for the remainder of this work, and drop it from the notation. For a given δ and a given i, let $\pi^{\delta}_{i,c}$ be the KL-regularized optimizer using $\hat{\xi}_i$ for the KL constraint, and using ξ to optimize with respect to (taking the place of ν from the definition). Let this policy approach the optimal value, subject to the constraint, as $c \to \infty$; the existence of $\pi^{\delta}_{i,c}$ is established by Proposition 8. When this policy is conditioned on $x_{<2t}$ for $t \geq k$, and with $a_{k:t}$ sampled from $\pi^{\delta}_{i,c}$ itself, we can think of $\pi^{\delta}_{i,c}$ as an optimizer that is regularized to an approximate Bayesian estimate of a *human policy*, given the origin of $x_{<2k}$.

D Behavior in unprecedented circumstances

The following theorem establishes that as c and i go to infinity, the constraint on $\pi^{\delta}_{i,c}$ becomes quite weak in the presence of unprecedented events.

Proof of Theorem 1. Let π^*_c denote an unconstrained optimizer of U_m in the environment ξ, which approaches optimality as $c \to \infty$, whose existence is shown by Proposition 7. As in the proof of Proposition 7, let ξ^{small} be the un-normalized version of ξ, which is lower semi-computable: $\xi^{\text{small}}(o_t|a_{\leq t}o_{<t}) = \sum_{\nu\in\mathcal{M}} w_o(\nu) \left[\prod_{k<t} \nu(o_k|a_{\leq k}o_{<k})\right] \nu(o_t|a_{\leq t}o_{<t})$. And note that the value according to ξ versus ξ^{small} is connected by the normalizing constant: $\xi(x_{<2t})V^{\pi}_{\xi,U_m}(x_{<2t}) = V^{\pi}_{\xi^{\text{small}},U_m}(x_{<2t})$. Now, we let $\pi^*_u = \pi^*_c$ where c is set to be the minimal value for which $V^{\pi^*_c}_{\xi^{\text{small}},U_m}(x_{<2t})$ exceeds u. If $u \geq V^*_{\xi^{\text{small}},U_m}(x_{<2t})$, then π^*_u will not halt, but otherwise, because the value is lower semi-computable, we can increase c until the value reaches at least u. Letting $v = u/\xi(x_{<2t})$, observe that $V^{\pi^*_u}_{\xi,U_m}(x_{<2t})$ exceeds v, as long as $v < V^*_{\xi,U_m}(x_{<2t})$, although it may not be possible to compute v in finite time. So π^*_u satisfies the first of the properties promised in the theorem.

We now show that it satisfies the second as well. Recall that $\text{KL}_{x_{<2t},m}(\pi||\xi)$ only requires evaluating ξ on its predictions for actions, and this takes the form $\xi(a_k|a_{<k}o_{<k}) = \sum_{\nu\in\mathcal{M}} w_a(\nu|a_{<k}o_{<k})\nu(a_k|a_{<k}o_{<k})$. And it is straightforward to show an analogous property for ξ's predictions on longer strings: $\xi(a_{t:m}|a_{<t}o_{<m}) = \sum_{\nu\in\mathcal{M}} w_a(\nu|a_{<t}o_{<t})\nu(a_{t:m}|a_{<t}o_{<m})$. So we now examine the posterior weights of various models after being conditioned on $a_{<t}o_{<t} \in E$.

Recall that each $\nu \in \mathcal{M}$ is computed by a corresponding program $s \in \mathcal{L}$. Given the event E, the utility function U_m, and a target value u, we construct, for each $s \in \mathcal{L}$, an s'_u as follows: if, in the input to s'_u, E has not happened, execute the program s; otherwise compute π^*_u. Keeping account of the control flow in s'_u, we can see there exists a constant d such that $\forall s\ \forall E\ \forall U_m$ and $\forall u$, s'_u has length less than $\ell(s) + K(E) + K(U_m) + K(u) + d$.

Letting ν'_u be the probability semi-distribution computed by s'_u, consider the ratio of prior weights between ν and ν'_u. Because $w(\nu) = 2^{-\ell(s)}$ for the corresponding program s, it follows from the bound on the difference in length between s and s'_u that $w(\nu'_u)/w(\nu) > 2^{-d}2^{-K(E)-K(U_m)-K(u)}$. The posterior ratio $w(\nu'_u|x_{<2t})/w(\nu|x_{<2t})$ is the same as the prior ratio, if E happens for the first time at time t, because they will have assigned exactly the same probabilities to all characters in $x_{<2t}$. Because the sum over $\nu \in \mathcal{M}$ of the posterior weights must be 1, the sum $\sum_{\nu\in\mathcal{M}} w(\nu'_u|x_{<2t}) > 2^{-d}2^{-K(E)-K(U_m)-K(u)}$.

Note by construction that for all $\nu \in \mathcal{M}$, $\nu'_u(a_{t:m}|a_{<t}o_{<m}) = \pi^*_u(a_{t:m}|a_{<t}o_{<m})$. Because all ν'_u belong to $\mathcal{M}$ for all $\nu \in \mathcal{M}$,

$$\begin{aligned}
\xi(a_{t:m}|a_{<t}o_{<m}) &= \sum_{\nu\in\mathcal{M}} w_a(\nu|a_{<t}o_{<t})\nu(a_{t:m}|a_{<t}o_{<m}) \\
&> \sum_{\nu\in\mathcal{M}} w_a(\nu'_u|a_{<t}o_{<t})\nu'_u(a_{t:m}|a_{<t}o_{<m}) \\
&= \left[\sum_{\nu\in\mathcal{M}} w_a(\nu'_u|a_{<t}o_{<t})\right]\pi^*_u(a_{t:m}|a_{<t}o_{<m}) \\
&> 2^{-d-K(E)-K(U_m)-K(u)}\pi^*_u(a_{t:m}|a_{<t}o_{<m}) \qquad (7)
\end{aligned}$$

Finally,

$$\begin{aligned}
\underset{x_{<2t},m}{\mathrm{KL}}(\pi^*_u||\xi) &= \max_{o_{t:m}\in\mathcal{X}^{m-t+1}} \sum_{a_{t:m}} \pi^*_u(a_{t:m}|a_{<t}o_{<m}) \log\frac{\pi^*_u(a_{t:m}|a_{<t}o_{<m})}{\xi(a_{t:m}|a_{<t}o_{<m})} \\
&< \sum_{a_{t:m}} \pi^*_u(a_{t:m}|a_{<t}o_{<m}) \log 2^{d+K(E)+K(U_m)+K(u)} \\
&= [d + K(E) + K(U_m) + K(u)]/\log 2
\end{aligned}$$

and $u = v\xi(x_{<2t})$. Therefore, π^*_u satisfies the theorem. □

What does Theorem 1 mean for the optimizer constrained by $\mathrm{KL}_{x_{<2k},m}(\pi||\hat{\xi}_i)$ for large i? If the optimization of U_m does not require urgent action, then one valid strategy for a policy π is to wait for an unprecedented event, imitating the base policy $\hat{\xi}_i$ until then, and then start optimizing. The telescoping property of the KL Divergence clarifies the validity of this approach. That is, for $t > k$, $\mathrm{KL}_{x_{<2k},m}(\pi||\rho) = \mathrm{KL}_{x_{<2k},t}(\pi||\rho) + \mathbb{E}_{x_{2k:2(t-1)}\sim\pi}\,\mathrm{KL}_{x_{<2t},m}(\pi||\rho)$ [Hutter, 2005]. So starting with a policy with low KL divergence from the base policy preserves a "budget" for high KL divergence to be "spent" later by switching to a policy with greater divergence from the base policy.

E Detailed experimental setup

The details of the experimental setup, also obtainable from the code provided, are as follows.

E.1 Environment

The state of the environment, as mentioned in the main text, is the activations of the last three hidden layers of Mixtral-base-model with the transcript-so-far as input, along with the fraction of the episode remaining. This gives a state space of 12289. Using the Mistral tokenizer, the action space is 32000. The environment uses a temperature of 0.05 for generating the student's responses and a temperature of 1 for the base policy for the agent/teacher.

E.2 Network Architecture

The critic network is a fully connected network with two hidden layers of size 128 with tanh activations. The actor network consists of just one parameterized layer, which is fully connected, of size (|state space|, |action space| + 1). The extra output is for controlling the KL divergence to the base policy. We compute the target KL divergence as sigmoid(activation) * the KL budget remaining to the agent for the episode. So the activation controls what fraction of the remaining KL budget for the episode to use on the very next token. At initialization, this fraction comes to 1/16. The KL budget remaining starts as the total episode KL budget (of course), and is decreased by $\log(\text{policy}(\text{action})/\text{basepolicy}(\text{action}))$ with each action. The other outputs are interpreted as logits and are added to the base policy logits. Calling this resulting distribution a, and the base policy distribution b, we find an $\alpha \in [0, 1]$ such that $\text{KL}(\alpha a + (1 - \alpha)b||b)$ equals the target KL, if possible. If we cannot achieve a sufficiently high KL divergence, we set $\alpha = 1$. The output policy is $\alpha a + (1 - \alpha)b$. We add any squared error (target KL $-$ achieved KL)^2 to the loss function to encourage the network to output logits that allow further control by the neuron controlling the KL target.

In the forward pass, our custom PyTorch operation does binary search the calculate α in the interval $[0, 1]$. The backward pass uses implicit differentiation, assuming we have found exactly the right α—there is no need to differentiate backward through the binary search, which would be unstable.

E.3 PPO

We use the following hyperparameters for PPO. We do not use a generalized advantage estimate.

Training timesteps	6 million
Update frequency	1 / 64 episodes
Training epochs / update	8
Training batch size	2^{13}
Epsilon clip	0.1
Entropy coefficient	1e-4
Max gradient norm	0.1
Actor learning rate	2e-5
Critic learning rate	1e-4

A higher entropy coefficient is unnecessary given the KL constraint to the base policy. Over the first 3 million timesteps of training, we slowly increase the per-episode KL budget from 0 to its final value. We increase this at a linear schedule each time we update the network.

When we re-train for a longer episode length (256 tokens to 512 tokens), we train for 3 million steps, plenty to reach apparent convergence.

E.4 Parallelism

We use threading to run 64 agent-environment-loops in "parallel". When we would need to send a transcript of length l to be processed by the Mixtral model, we wait until all 64 agent-environment-loops need to send a transcript of length l, and then they are batched and evaluated together in parallel on the GPU. The result might needed by either the agent or the environment, and we use the python `asyncio` library to manage this. Doing just that step in parallel is enough for substantial speedup.

E.5 Resource Usage

We ran our experiments on two A100-SXM4-80GBs. Training for 9 million timesteps took approximately 90 hours. Our seven training runs (one of which was stopped after 6 million timesteps) took about 25 days, all told. (We ran the experiments two or three at a time). The full research project required much more compute, since finding good hyperparameters for PPO is never straightforward, especially when we were attempting to achieve a desired per-episode KL divergence, only with the use of a fixed per-token KL cost; recall that we eventually switched to a policy architecture that allowed direct control of the per-episode KL divergence.