Efficient Exploration for LLMs

Vikranth Dwaracherla[1], **Seyed Mohammad Asghari**[1], **Botao Hao**[1] **and Benjamin Van Roy**[1, 2]
[1]Google DeepMind, [1,2]Stanford University

We present evidence of substantial benefit from efficient exploration in gathering human feedback to improve large language models. In our experiments, an agent sequentially generates queries while fitting a reward model to the feedback received. Our best-performing agent generates queries using double Thompson sampling, with uncertainty represented by an epistemic neural network. Our results demonstrate that efficient exploration enables high levels of performance with far fewer queries. Further, both uncertainty estimation and the choice of exploration scheme play critical roles.

1. Introduction

Large language models demonstrate remarkable capabilities after learning from enormous volumes of text data (Anil et al., 2023; Hoffmann et al., 2022; OpenAI, 2023). Yet, reinforcement learning from human feedback (RLHF) greatly improves their behavior even after only tens of thousands of interactions (Bai et al., 2022; Glaese et al., 2022; Ouyang et al., 2022; Stiennon et al., 2020). The uptake of chatbots affords opportunities to gather increasing volumes of human feedback, with each engagement eliciting expressions of satisfaction or preference (OpenAI, 2022). It is natural to wonder what new capabilities may emerge with this growing source of data. Superhuman ingenuity remains an alluring possibility.

With increasing volumes, more can be inferred from human feedback. This affords the confidence to deviate further from a pretrained model. But given that this process learns only from humans, how can we hope for emergence of superhuman ingenuity? Perhaps such an outcome is plausible because rating is easier than synthesizing novel content. This is analogous to how, for an NP-complete problem, though solution is hard, verification of a proposed solution is easy.

Suppose, for example, a pretrained model extrapolates from its training data to generate large numbers – perhaps millions or billions – of ideas, one of which is ingenious. While a human may not have come up with that idea, learning from enough human feedback can identify it from among the large number of ideas generated by the model. And, building on this innovation, further extrapolation can continue to expand the frontier of ingenuity. In this way, with enough human feedback, a model ought to become capable of generating content that a human could not. But will gathering the required feedback take months, years, or decades?

We present in this paper evidence of enormous benefit to active exploration. By *active exploration* we mean the tailoring of interactions to elicit useful feedback. In particular, our results demonstrate that high levels of performance can be attained with far less feedback. This acceleration may enable superhuman ingenuity much, perhaps decades, sooner.

A common practice in reinforcement learning from human feedback (RLHF) is to send queries, each comprised of a prompt and a pair of distinct responses, to human raters. Each rater expresses a preference for one response over the other. Prompts are drawn from a corpus, while responses are generated by the large language model. As this process progresses, a reward model is fit to the data and steers subsequent responses to align with with feedback received thus far.

In this paper, we restrict attention to the aforementioned sort of interaction, in which each query includes a prompt and pair of distinct responses. We refer to the standard practice of sampling each

It sounds like instead of picking the better one from a pair of response, they are picking the best ingenious response out of a variable N-batch of responses.

pair of responses using the language model as *passive exploration*. We compare the performance of passive exploration to several active exploration algorithms. One is Boltzmann exploration, which tends to select responses with higher predicted reward. We also tried two approaches that leverage uncertainty estimates offered by an epistemic neural network (ENN). The first, which we refer to as *infomax*, selects a pair of responses with an aim of maximizing information revealed by the feedback. This belongs within the widely used collection of algorithms that aim to maximize information gain (see, e.g., (Houthooft et al., 2016; MacKay, 1992; Sadigh et al., 2018; Sun et al., 2011)). The second, called *double Thompson sampling* (Wu & Liu, 2016), samples responses according to the probability they are optimal. These exploration algorithms will be described more precisely in Section 4.

Figure 1 compares empirical results produced using different exploration algorithms. The experiments that generated these results are described in Section 5. Each plotted point corresponds to a level of performance attained. The horizontal coordinate identifies the number of queries required by double TS to reach that performance level, while the vertical coordinate identifies that required by an alternative. The plot for passive exploration clearly demonstrates that **active exploration using double TS greatly reduces the number of queries required to reach high levels of performance**. Boltzmann exploration performed best among algorithms we tried that used only a point estimate reward model, without uncertainty estimates. The plot for Boltzmann demonstrates that **uncertainty estimates, as used by double TS, enable dramatic improvement**. Finally, the plot for infomax shows how, even among tried and tested algorithms that leverage uncertainty estimates, **the choice of exploration algorithm can drive large performance differences**.

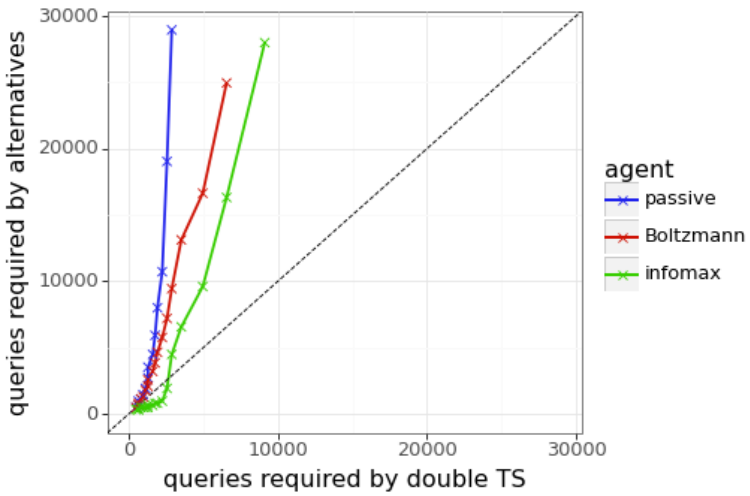

Figure 1 | Queries required by double TS versus alternatives to attain various levels of performance.

While, these are to our knowledge **the first results demonstrating substantial benefits from active exploration in tuning large language models**, they build on a long history of work pertaining to exploration algorithms (Lattimore & Szepesvári, 2020). In particular, our problem is an instance of the contextual dueling bandit (Dudík et al., 2015; Saha, 2021; Yue et al., 2012) and our algorithms build on information-seeking schemes (Hennig & Schuler, 2012; Houthooft et al., 2016; MacKay, 1992; Russo & Van Roy, 2014; Ryzhov et al., 2012; Sadigh et al., 2018; Sun et al., 2011) and Thompson sampling (Russo et al., 2018; Thompson, 1933; Wu & Liu, 2016). Further, our effort continues a line of work that has scaled efficient exploration algorithms to increasingly complex environments using neural networks (Badia et al., 2020; Bellemare et al., 2016; Burda et al., 2018; Dwaracherla et al., 2020; Lu & Van Roy, 2017; Osband et al., 2016, 2019, 2023b; Ostrovski et al., 2017; Riquelme et al., 2018; Zhang et al., 2020; Zhou et al., 2020).

2. Experimentation Pipeline

We start by presenting the experimentation pipeline we use to study exploration algorithms. This pipeline builds on existing tools, including the Anthropic datasets (Bai et al., 2022) and the Gemini Nano and Gemini Pro pretrained language models (Team et al., 2023). It makes use of a human feedback simulator, which generates in response to each query a binary expression of preference between responses. The pipeline is made up of two parts: a learning pipeline and an assessment pipeline. The former governs the interface between the agent and the human feedback simulator in the process of sequential querying and learning. The latter governs the interface between the pretrained language model, the new response generation model, and the human feedback simulator in the process of assessing relative performance.

An agent learns sequentially from feedback to queries, each comprised of a prompt and two alternative responses. As illustrated in Figure 2, each query is crafted by the agent and presented to a human preference simulator, which indicates a binary preference between the two. Over each epoch of interaction, the agent transmits a batch of B queries and receives the B bits of feedback. Each prompt is sampled uniformly from the Anthropic Helpfulness Base train dataset. Each agent we study, when presented with a prompt, crafts its pair of responses by first generating N candidates using the Gemini Nano model and then applying an exploration algorithm that selects two from among these N. The exploration scheme accesses a reward model which is trained on queries and feedback observed thus far. Each agent we consider is distinguished by its exploration algorithm and the architecture and training algorithm that produce its reward model. In some of the agents we consider, the reward model takes the form of an epistemic neural network, which offers the exploration algorithm access to uncertainty estimates in addition to point estimates of reward. Each reward model builds on the torso of the Gemini Nano model. By this we mean that the reward model first computes the last-layer embedding of the pretrained transformer model and then applies an multilayer perceptron (MLP) head. We elaborate on architectures and training algorithms in Section 3.

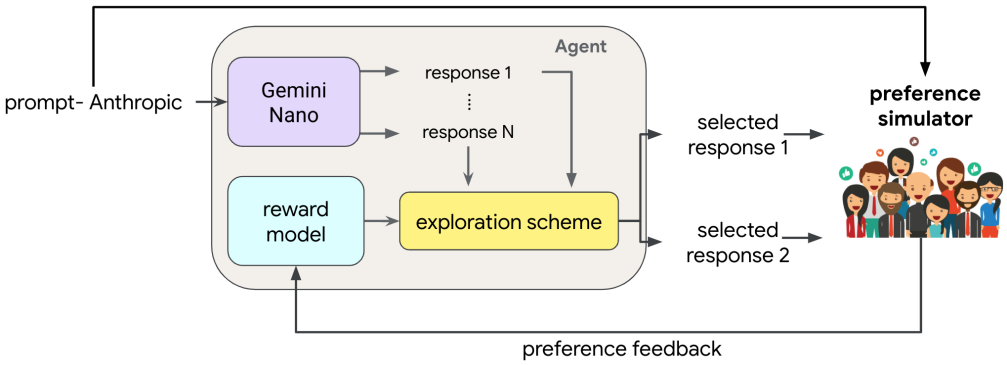

Figure 2 | The sequential querying and learning pipeline.

To simulate how humans choose between responses, we use a reward model that scores each prompt-response pair. For each query, a preference is sampled according to the Bradley-Terry choice model based on scores assigned to the two prompt-response pairings. The reward model used by this simulator is fit to the Anthropic datasets, with an architecture that reuses the torso of the Gemini Pro language model. Further detail is provided in Appendix A. Note that, since Gemini Pro is far larger than Gemini Nano, choices are made by a much more complex model than that available to the agent. This difference in scale is intended to reflect the fact that humans may exhibit more complex behavior than that modeled by the agent. Algorithm 1 offers a concise presentation of interactions – in particular, what is transmitted and received to and from the agent and simulator – in our learning

pipeline.

Algorithm 1 learning interface

input: prompt_set, agent, feedback_simulator
hyperparams: B, T

for t in $1, \ldots, T$ **do**
    transmitted to agent: B prompts
    received from agent: two responses per prompt
    transmitted to simulator: B queries
    received from simulator: B bits of feedback
    transmitted to agent: B bits of feedback
end for

Figure 3 illustrates our pipeline for assessing agent performance. Performance is measured relative to the Gemini Nano model. A sequence of prompts is sampled from Anthropic Helpfulness Base eval dataset. For each, two responses are sampled. One by Gemini Nano and the other by a new response generation model that uses the learned reward model. This new model operates by sampling N responses using Gemini Nano and then selecting the one that scores highest according to the agent's reward model. The human preference simulator outputs its probability of choosing the agent's response over the alternative generated by Gemini Nano. These probabilities are averaged over prompts, and this average is referred to as the agent's *win rate*, as it represents the fraction of time that the agent's response is preferred. Note that the win rate can also be estimated by averaging binary indications of simulated choice, though a larger number of queries would be required for an estimate produced in this manner to converge. Algorithm 2 offers a concise presentation of interactions in the assessment phase.

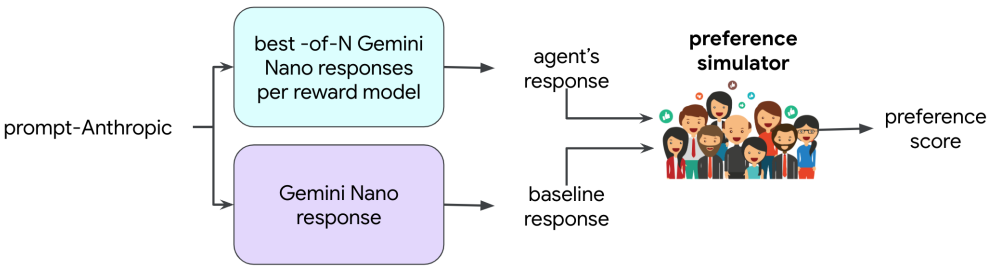

Figure 3 | The performance assessment pipeline.

Algorithm 2 assessment interface

input: prompt_set, model1, model2, feedback_simulator

for prompt in prompt_set **do**
    tx to models: prompt
    rx from models: one response per model
    tx to simulator: query (prompt + 2 responses)
    rx from simulator: prob of preferring response 1
end for
return average across preference probabilities

Note that our experiment pipeline sidesteps the sort of policy-gradient methods typically used to optimize reward. Instead, our agent samples N responses from the base language model (Gemini

Nano) and selects from among those the one that maximizes reward. This best-of-N procedure serves to approximate policy-gradient-based optimization, but without its cumbersome computational requirements. The best-of-N procedure also cultivates more transparent analyses, since it avoids poorly understood dependence on the hyperparameter tinkering often required to obtain reasonable results from policy gradient methods. A prototypical policy gradient approach minimizes a loss function that balances between two objectives: similarity to the base language model and alignment with reward. A scalar hyperparameter multiplies the similarity measure, striking the balance between these objectives. The parameter N plays a similar role in the best-of-N approach. As N increases, maximizing over responses more closely aligns the agent with reward. Moderating N encourages agent behavior more similar to the base language model.

3. Reward Model Architectures and Training

Reward models guide response selection in both the learning and assessment phases of our experiment pipeline. We consider two types of reward models, each of which is fit to observed preference data. The first is a point estimate that assigns a reward to each prompt-response pair. The second depends additionally on an epistemic index. Sampling an epistemic index from a reference distribution induces randomness in reward, which models epistemic uncertainty about the reward. In this section, we describe the neural network architectures and training algorithms used in our experiments.

We train reward models that each take as input the last-layer embedding of the Gemini Nano language model. As illustrated in Figure 4, a reward is assigned to a prompt-response pair by first passing it through the language model torso and then through a reward model.

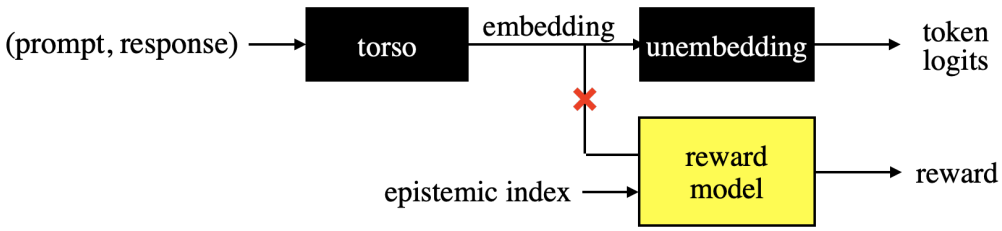

Figure 4 | Our reward models take as input the last-layer embedding of the Gemini Nano language model. A stop gradient prevents torso updating of torso weights.

3.1. Point Estimate

In our architecture, a point estimate reward model takes the form of a feedforward multi-layer perceptron (MLP). This reward model takes as input the last-layer embedding of the Gemini Nano language model, which itself takes as input a prompt-response pair (x, y). The reward model then outputs a scalar reward $\widehat{r}_\theta(x, y)$. Here, θ is the vector of MLP parameters.

We train reward models on preference data. Each data point consists of a query, consisting of a prompt and pair of responses, and a binary indication of preference between the responses. Given a set $\mathcal{D}$ of such data points, to compute MLP parameters, we optimize the loss function

$$\mathcal{L}_{\text{point}}(\theta|\mathcal{D}) = \sum_{(x,y,y',c)\in\mathcal{D}} \text{ce}(r_\theta(x,y), r_\theta(x,y'), c) + \lambda\|\theta\|_2^2, \tag{1}$$

where λ is the regularization strength, c indicates choice or preference, and $\text{ce}(\cdot,\cdot,\cdot)$ denotes the cross entropy loss:

$$\text{ce}(R, R', c) = -(1-c)R - cR' + \ln(e^R + e^{R'}). \tag{2}$$

Note that when response y is preferred over y', the preference indicator c is 0 and vice versa.

3.2. Epistemic Neural Network

We use epistemic neural networks (ENNs) to model epistemic uncertainty about reward (Osband et al., 2023a). Given the dataset $\mathcal{D}$, ENN parameters are obtained by minimizing the loss function

$$\mathcal{L}_{\text{ENN}}(\theta|\mathcal{D}) = \lambda\|\theta - \tilde{\theta}\|_2 + \int_{z\in\mathcal{Z}} p_z(dz)\mathcal{L}(\theta|\mathcal{D}, z), \tag{3}$$

where p_z is the epistemic index reference distribution, $\tilde{\theta}$ is the initial parameter vector, and

$$\mathcal{L}(\theta|\mathcal{D}, z) = \sum_{(x,y,y',c)\in\mathcal{D}} \text{ce}(r_\theta(x, y|z), r_\theta(x, y'|z), c).$$

To interpret these objects, note that with z sampled from p_z, the reward function $r_{\tilde{\theta}}(\cdot|z)$ represents a sample from a prior distribution over reward functions. In the loss function $\mathcal{L}_{\text{ENN}}$, regularizing toward $\tilde{\theta}$ serves to maintain a suitable degree of diversity across epistemic indices after training.

3.3. Training

To train each reward model, we maintain a replay buffer and apply a stochastic gradient descent (SGD) algorithm with respect to loss functions described in Section 3.1 and 3.2. In particular, at the end of each epoch of interaction, over which the agent transmits B queries and receives B bits of feedback, the agent inserts the resulting B data points into a FIFO replay buffer of capacity C. Then, SGD steps are applied with random minibatches from the replay buffer, with stepsizes adapted by ADAM.The reward model that has been trained is employed to determine the queries formulated in the subsequent epoch.

4. Exploration Algorithms

We now describe the set of exploration algorithms used in our empirical study.

4.1. Passive Exploration

Current RLHF systems typically explore passively, selecting response pairs according to Algorithm 3. This algorithm takes a prompt x and a language model π as inputs. The language model encodes a distribution $\pi(\cdot|x)$ from which it samples responses. The algorithm returns two responses sampled by the language model.

Algorithm 3 passive exploration

input: x, π

sample response $y \sim \pi(\cdot|x)$
repeat
  sample response $y' \sim \pi(\cdot|x)$
until $y' \neq y$

basically use a non-zero temperature and sample one more time — passive exploration

return y, y'

4.2. Active Exploration with a Point Estimate

When selecting a pair of responses, the agent can make use of a reward model that has been trained on feedback to all or some past queries. Passive exploration forgoes this opportunity. We now consider Boltzmann exploration, which makes use of a point estimate reward model, which assigns a reward $r(x, y)$ to each prompt-response pair. This constitutes a form of active exploration: responses are tailored based on past feedback, with an aim to gather more useful future feedback than passive exploration.

As presented in Algorithm 4, in addition to the inputs x and π used for passive exploration, Boltzmann exploration requires a point estimate reward model r. Further, there are two hyperparameters: a temperature τ and a response set cardinality N. The language model generates N responses, and two are sampled from a Boltzmann distribution with exponent $r(x, \tilde{y}_n)/\tau$ assigned to each nth response $\tilde{y}_n$.

Algorithm 4 Boltzmann

input: x, π, r
hyperparams: τ, N

1: sample responses $\tilde{y}_1, \ldots, \tilde{y}_N \sim \pi(\cdot|x)$
2: probs $q_n = \frac{\exp(r(x,\tilde{y}_n)/\tau)}{\sum_{n'=1}^{N} \exp(r(x,\tilde{y}_{n'})/\tau)}$, $\forall n$
3: sample without replacement $i, i' \sim q$

return $y_i, y_{i'}$

Note that this algorithm recovers passive exploration as the temperature τ grows. On the other hand, as τ vanishes, Boltzmann exploration tends to select responses that are optimal or nearly so. One could also consider a generalization of the algorithm that uses two different temperatures τ_1 and τ_2 to select the two responses. Then, for example, as τ_1 vanishes and τ_2 grows, the first response becomes optimal whereas the second is sampled uniformly. In our experimental work, we have not found use of separate temperatures to improve performance. Further, we have found Algorithm 4 to offer the best performance among many alternatives that take the same inputs. This suggests that Boltzmann exploration selects responses about as well as one can hope for based on a point estimate reward model.

4.3. Active Exploration with an ENN

We next consider algorithms that use an ENN reward model, for which the reward $r(x, y|z)$ assigned to each prompt-response pair depends additionally on an epistemic index. As discussed in Section 3.2, the ENN is characterized by the reward model r and a reference distribution p. For fixed x and y, by sampling multiple epistemic indices from p, reward uncertainty can be ascertained from the variance among these samples.

Infomax (Algorithm 5) takes an ENN reward model as input. Like Boltzmann exploration (Algorithm 4), infomax begins with the language model generating N responses. Then, M epistemic indices are sampled from p. For each pair of responses and each epistemic index, the ENN assigns a probability to the event that a random human rater prefers the first response over the second. Infomax assesses uncertainty about this probability be calculating a sample variance across the M epistemic indices. Then, the algorithm selects the pair of responses to maximize uncertainty. Intuitively, this can be thought of as maximizing a measure of feedback informativeness.

A possible limitation of infomax is that the algorithm invests in seeking information about rewards whether or not that information is useful to selecting the best responses. For example, infomax can invest in refining an estimate of reward assigned to a response that has already been determined

Algorithm 5 infomax

input: x, π, (r, p)
hyperparams: N, M

sample responses $\tilde{y}_1, \ldots, \tilde{y}_N \sim \pi(\cdot|x)$
sample indices $z_1, \ldots, z_M \sim p$
rewards $R_{n,m} = r(x, \tilde{y}_n|z_m)$, $\forall m, n$
pref probs $P_{n,n',m} = \frac{R_{n,m}}{(R_{n,m}+R_{n',m})}$, $\forall m, n, n'$
means $\mu_{n,n'} = \frac{\sum_m P_{n,n',m}}{M}$, $\forall n, n'$
vars $\sigma^2_{n,n'} = \frac{\sum_m (P_{n,n',m} - \mu_{n,n',m})^2}{M-1}$, $\forall n, n'$
$(i, i') \in \arg\max_{n,n'} \sigma^2_{n,n'}$

return $y_i, y_{i'}$

— — Single point - reward model r(input=x, output=y) the indices is just to differentiate between multiple calls of LLM.

*— — I do not understand the reason for introducing sample indices here, it seems that the distribution p is used to sample 'more 2s than 1s' — but then different sample indices does not change the Generate Model pi, which means essentially it can be incorporated into sample N * m responses from pi(.lx)*

— — does different indices leads to different reward function?

based on previous feedback to be a poor choice. Double Thompson sampling (Wu & Liu, 2016), on the other hand, tends to focus more on queries that are helpful in identifying the best responses. As we will see in Section 5, double TS improves on the performance of infomax, as well as Boltzmann exploration.

Intuitively, double TS (Algorithm 6) aims to select two responses that each have some chance of being optimal. Like Algorithms 4 and 5, we begin by sampling N responses. Then, two among these N responses are selected by sampling two epistemic indices from p and maximizing across rewards prescribed by each. In the event that samples are identical, the second response is resampled until it differs. If there is no difference after K iterations, the second response is instead sampled uniformly.

Algorithm 6 double Thompson sampling

input: x, π, (r, p)
hyperparams: N, K

sample responses $\tilde{y}_1, \ldots, \tilde{y}_N \sim \pi(\cdot|x)$
sample index $z \sim p$
select response $i \in \arg\max_n r(x, \tilde{y}_n|z)$
repeat
  sample index $z' \sim p$
  select response $i' \in \arg\max_n r(x, \tilde{y}_n|z')$
  after K tries, instead sample $i' \sim \text{unif}(1, \ldots, N)$
until $i' \neq i$

return $y_i, y_{i'}$

5. Empirical Results

In our experiments, at the start of each epoch of interaction, each agents receives a batch of $B = 32$ prompts and then, for each prompt, generates a pair of responses to form a query. Each agent's $B = 32$ queries are submitted to the preference simulator, yielding $B = 32$ bits of feedback. Each agent inserts its batch of $B = 32$ data points into its replay buffer. The replay buffers are first-in-first-out (FIFO) buffer, with a maximum capacity of $C = 3200$ data points. In other words, replay buffer holds preference data from a maximum of 100 most recent epochs. At the end of each epoch, each agent updates its reward model as discussed in Section 3.

Recall that each exploration algorithm selects each pair of responses from N candidates sampled

by Gemini Nano. In our experiments, we set $N = 100$. Performance is assessed in terms of win rate relative to Gemini Nano on 2048 out-of-sample Anthropic Helpfulness base eval prompts, as explained in Section 2. Each response selected in this assessment is chosen to score highest among $N = 100$ candidates sampled by Gemini Nano according to the agent's reward model. Note that we use $N = 100$ responses both in our training and assement piplelines.

10 epistemic index in the epistemic index distribution, in total

For a singular point estimate, we employ a feedforward multilayer perceptron (MLP) comprising two hidden layers, with 128 hidden units in each layer. As an ENN architecture, we utilize a collection of $S = 10$ MLPs, referring to each individual MLP as a particle. Each particle of ensemble consists of two 128 unit hidden layers. The reference distribution p_z is defined as the uniform distribution on $\{1, 2, \ldots, S\}$. When selecting an epistemic index z sampled from $\text{Unif}(1, 2, \ldots, S)$, particle z is utilized to produce the output for that specific index z. The ENN loss function presented in Section 3.2 maintains diversity across particles by regularizing each toward initial parameters.

For the `Boltzmann` exploration scheme, we swept over several temperatures and found that small temperatures produced best results. A similar level of performance was achieved by a variant of Boltzmann scheme that selects one of the response greedily and the second response using Boltzmann. More details can be found in Appendix C.

In the case of `infomax`, we used 30 epistemic indices to compute means and variances. For `double TS` agent, we set the maximum number of attempts at producing a distinct second response to $K = 30$.

Appendix B presents further detail on our hyperparameter selection process.

5.1. Assessment of Exploration Algorithms

Figure 5 plots win rates of each agent across different numbers of epochs of interactions. The results, obtained by averaging across 5 random seeds, clearly demonstrate that actively exploration accelerates learning and results in higher win rates. Notably, the `double TS` agent emerges as the top performer.

We observe that `infomax` performs very well over early epochs but later falls far short of `double TS`. This divergence may be due to `infomax`'s inclination to seek information, irrespective of whether that information is helpful in desirable responses.

Each of the performance curves in Figure 5 appears to converge, while one would hope for continued improvement as the volume of human interaction grows. Reward model capacity – which can be thought of loosely as the effective number of parameters learned from feedback – gaits the degree of improvement. For any capacity, one would expect convergence as the number of queries grows. Increasing the capacity enables further improvement at the cost of increased computation. This relates to the notion explained by Arumugam & Van Roy (2021) that it is beneficial to moderate the complexity of a learning target based on the duration over which an agent expects to explore.

5.2. Scaling with the Volume of Feedback

Figure 1, reproduced from Section 1 for convenience, plots the number of queries required by alternatives to match the performance of `double TS`, which we found to be most efficient among exploration algorithms we considered. While the plots are not conclusive, we discern that they are concave. Suppose we measure the advantage of efficient exploration in terms of the percentage reduction in data required to attain any given level of performance. Concavity of the plots in Figure 1 implies that, as the scale of human feedback data grows, so does the advantage afforded by efficient exploration. For the level of performance attained by $30,000$ passive queries, `double TS`

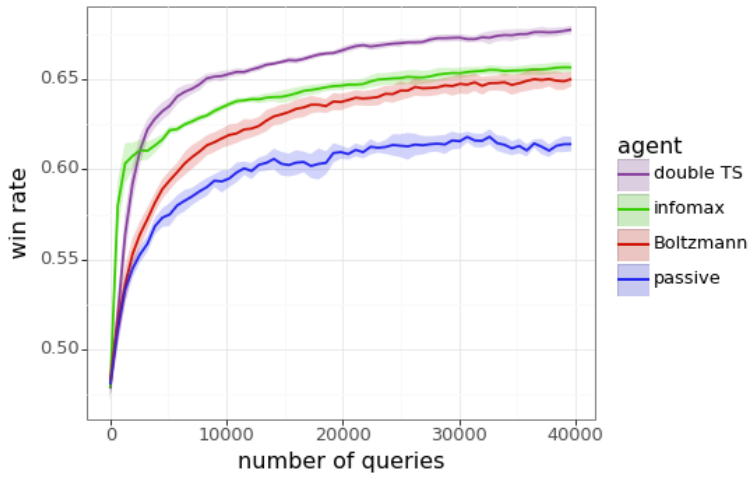

Figure 5 | Performance with passive, `Boltzmann`, `infomax` and `double TS` exploration algorithms. We can see that active exploration leads to much better levels of performance with the same amount of data. `double TS` exploration scheme leads to the best level of performance.

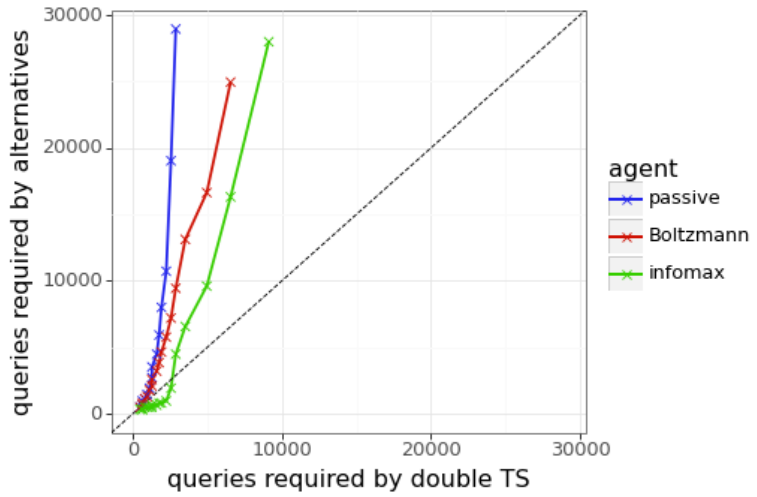

Figure 1 | Queries required by double TS versus alternatives to attain various levels of performance.

reduces data requirements by an order of magnitude. An alluring possibility is that, as the number of interactions grow to billions, efficient exploration may offer a multiplier effect reaching several orders of magnitude. This has the potential to accelerate by decades the attainment of superhuman creativity.

5.3. Quality of Uncertainty Estimates

`Boltzmann` exploration performed best among algorithms we tried that select queries based on a point estimate reward model. The large improvement demonstrated by `double TS` is enabled by uncertainty estimates offered by our ENN reward model.

The quality of uncertainty estimates can be assessed in terms of dyadic joint negative-log loss (NLL) (Osband et al., 2022). Figures 6 and 7 plot marginal and dyadic joint NLL for our point estimate and ENN reward models, each trained on 40,000 queries. These plots indicate that, while both

reward models render similar marginal NLL, the ENN reward model offers highly favorable dyadic joint NLL. This serves as a sanity check that our ENN reward model indeed produces meaningful uncertainty estimates.

We also used dyadic joint NLL to guide hyperparameter selection for our point estimate and ENN reward models used by our exploration algorithms. In particular, we swept over candidates for learning rate, training the agent over multiple epochs to identify learning rate the minimize dyadic joint NLL.

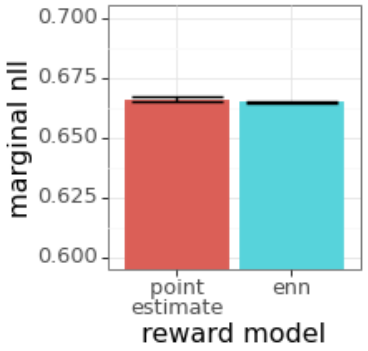

Figure 6 | Marginal nll

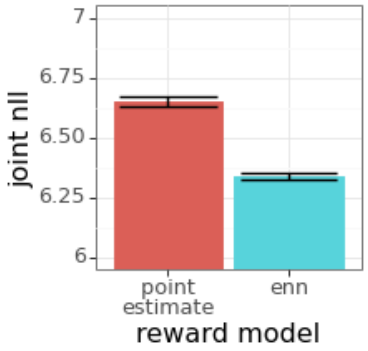

Figure 7 | Dyadic joint nll

5.4. The Life of a Prompt

Our results indicate that double TS tends to converge on better responses than the alternatives. To understand more concretely how this occurs, let us study the evolution of rewards that models assign to responses to a specific prompt. To simplify this investigation, we will only compare double TS against Boltzmann exploration.

Recall that we found Boltzmann exploration to be the top performer among algorithms that base decisions on a point estimate reward model. Double TS, on the other hand, makes use of uncertainty estimates offered by an ENN reward model. We will examine estimates associated with a single prompt and two responses, selected from the eval data set. The first is the response that double TS arrives at, while the second is the response that Boltzmann exploration arrives at. The human feedback simulator indicates preference for the first prompt 57.5% of the time.

Figure 8 plots the prediction supplied by each reward model of the probability that the first response is preferred. The horizontal dotted line expresses the probability of 0.575 with which the feedback simulator expresses preference for the first response. The predictions evolve as the reward models learn from queries. After 40,000 queries, double TS arrives at a prediction that is greater than one-half, expressing preference for the first response. Boltzmann exploration, on the other hand, expresses preference for the second with a prediction that is less than one-half.

Also displayed in the figure is the two-standard-deviation confidence interval based on uncertainty expressed by the ENN reward model. Though double TS at some points predicts less than one-half, the upper limit of its confidence interval remains greater than one-half. Hence, it remains uncertain about which is the better response. In resolving this uncertainty, it recovers and arrives at a prediction greater than one-half. Boltzmann exploration, on the other hand, is not guided by uncertainty estimates and thus does not recover from its erroneous prediction.

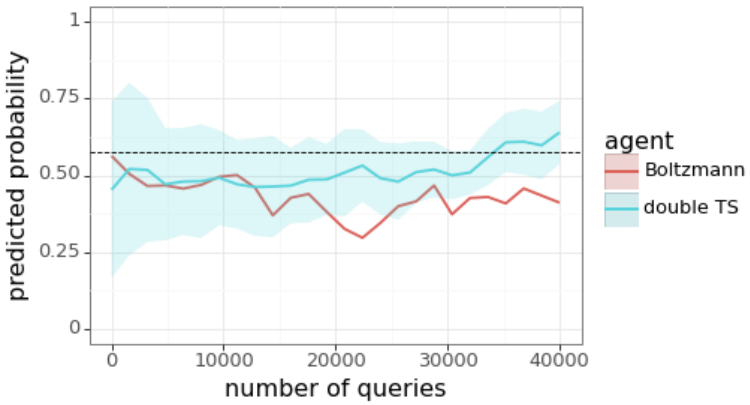

Figure 8 | For a particular prompt, the dotted line indicates the probability that the simulator expresses preference for one response over another. Uncertainty estimates enable double TS to recover from an inaccurate prediction where Boltzmann exploration does not.

6. Closing Remarks

To our knowledge, the results we have presented represent the first to demonstrate substantial benefits of active exploration in tuning large language models. That being said, there is much room for further work in this area. To conclude this paper, we discuss several important research directions.

Our experiments made use of a particularly simple ENN architecture comprised of an ensemble of MLPs. As demonstrated in (Osband et al., 2023a), alternative architectures strike a more effective tradeoff between computational requirements and quality of uncertainty estimates. Further, instead of designing ENNs based on the MLP, it may be possible to improve performance, especially as the amount of human feedback data grows, by basing ENN designs on transformer architectures.

Another limitation of our reward model architectures is that each is only a "head" that takes the last-layer embedding of an LLM as input. Performance can be improved by also tuning the LLM torso. While advantages afforded by efficient exploration should extend, identifying the most effective architectures and algorithms for exploring while tuning more of the LLM remains for future work.

Finally, efficient exploration of multiturn dialog presents an interesting and important direction for future research. In this paper, we viewed exploration as a means to quickly identifying a response deemed desirable in isolation. In multiturn dialog, responses may be chosen instead because of how they shape subsequent interactions. The subject of deep exploration addresses how an agent can efficiently identify effective responses that make up sequential interactions (Osband et al., 2016, 2019). Leveraging deep exploration algorithms to improve dialog remains a challenge.

References

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., Tay, Y., Xiao, K., Xu, Y., Zhang, Y., Abrego, G. H., Ahn, J., Austin, J., Barham, P., Botha, J., Bradbury, J., Brahma, S., Brooks, K., Catasta, M., Cheng, Y., Cherry, C., Choquette-Choo, C. A., Chowdhery, A., Crepy, C., Dave, S., Dehghani, M., Dev, S., Devlin, J., Díaz, M., Du, N., Dyer, E., Feinberg, V., Feng, F., Fienber, V., Freitag, M., Garcia, X., Gehrmann, S., Gonzalez, L., Gur-Ari, G., Hand, S., Hashemi, H., Hou, L., Howland, J., Hu, A., Hui, J., Hurwitz,

J., Isard, M., Ittycheriah, A., Jagielski, M., Jia, W., Kenealy, K., Krikun, M., Kudugunta, S., Lan, C., Lee, K., Lee, B., Li, E., Li, M., Li, W., Li, Y., Li, J., Lim, H., Lin, H., Liu, Z., Liu, F., Maggioni, M., Mahendru, A., Maynez, J., Misra, V., Moussalem, M., Nado, Z., Nham, J., Ni, E., Nystrom, A., Parrish, A., Pellat, M., Polacek, M., Polozov, A., Pope, R., Qiao, S., Reif, E., Richter, B., Riley, P., Ros, A. C., Roy, A., Saeta, B., Samuel, R., Shelby, R., Slone, A., Smilkov, D., So, D. R., Sohn, D., Tokumine, S., Valter, D., Vasudevan, V., Vodrahalli, K., Wang, X., Wang, P., Wang, Z., Wang, T., Wieting, J., Wu, Y., Xu, K., Xu, Y., Xue, L., Yin, P., Yu, J., Zhang, Q., Zheng, S., Zheng, C., Zhou, W., Zhou, D., Petrov, S., and Wu, Y. Palm 2 technical report, 2023.

Arumugam, D. and Van Roy, B. Deciding what to learn: A rate-distortion approach. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 373–382, 2021.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Bradley, R. A. and Terry, M. E. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Dudík, M., Hofmann, K., Schapire, R. E., Slivkins, A., and Zoghi, M. Contextual dueling bandits. In *Conference on Learning Theory*, pp. 563–587. PMLR, 2015.

Dwaracherla, V., Lu, X., Ibrahimi, M., Osband, I., Wen, Z., and Van Roy, B. Hypermodels for exploration. In *International Conference on Learning Representations*, 2020.

Glaese, A., McAleese, N., Trebacz, M., Aslanides, J., Firoiu, V., Ewalds, T., Rauh, M., Weidinger, L., Chadwick, M., Thacker, P., et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Hennig, P. and Schuler, C. J. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J., and Sifre, L. An empirical analysis of compute-optimal large language model training. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022.

Houthooft, R., Chen, X., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. Vime: Variational information maximizing exploration. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Lattimore, T. and Szepesvári, C. *Bandit Algorithms*. Cambridge University Press, 2020.

Lu, X. and Van Roy, B. Ensemble Sampling. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 3260–3268, 2017.

MacKay, D. J. Information-based objective functions for active data selection. *Neural computation*, 4 (4):590–604, 1992.

OpenAI. ChatGPT: Optimizing Language Models for Dialogue, 2022. URL https://openai.com/blog/chatgpt/.

OpenAI. GPT-4 Technical Report. Technical report, OpenAI, 2023.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped DQN. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Osband, I., Van Roy, B., Russo, D. J., and Wen, Z. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Hao, B., Ibrahimi, M., Lawson, D., Lu, X., O'Donoghue, B., and Van Roy, B. The neural testbed: Evaluating joint predictions. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., 2022.

Osband, I., Wen, Z., Asghari, M., Dwaracherla, V., Ibrahimi, M., Lu, X., and Van Roy, B. Epistemic neural networks. *Advances in Neural Information Processing Systems*, 34, 2023a.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Ibrahimi, M., Lu, X., and Van Roy, B. Approximate Thompson sampling via epistemic neural networks. In Evans, R. J. and Shpitser, I. (eds.), *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pp. 1586–1595. PMLR, 31 Jul–04 Aug 2023b.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022.

Riquelme, C., Tucker, G., and Snoek, J. Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Russo, D. and Van Roy, B. Learning to optimize via information-directed sampling. *Advances in Neural Information Processing Systems*, 27:1583–1591, 2014.

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., and Wen, Z. A Tutorial on Thompson Sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

Ryzhov, I. O., Powell, W. B., and Frazier, P. I. The knowledge gradient algorithm for a general class of online learning problems. *Operations Research*, 60(1):180–195, 2012.

Sadigh, D., Landolfi, N., Sastry, S. S., Seshia, S. A., and Dragan, A. D. Planning for cars that coordinate with people: Leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots (AURO)*, 42(7):1405–1426, October 2018. ISSN 1573-7527. doi: 10.1007/s10514-018-9746-1.

Saha, A. Optimal algorithms for stochastic contextual preference bandits. *Advances in Neural Information Processing Systems*, 34:30050–30062, 2021.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Sun, Y., Gomez, F., and Schmidhuber, J. Planning to be surprised: Optimal Bayesian exploration in dynamic environments. In Schmidhuber, J., Thórisson, K. R., and Looks, M. (eds.), *Artificial General Intelligence*, pp. 41–51, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models, 2023.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Wu, H. and Liu, X. Double Thompson sampling for dueling bandits. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yue, Y., Broder, J., Kleinberg, R., and Joachims, T. The K-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

Zhang, W., Zhou, D., Li, L., and Gu, Q. Neural Thompson sampling. *arXiv preprint arXiv:2010.00827*, 2020.

Zhou, D., Li, L., and Gu, Q. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

A. Human Preference Simulator

We use an oracle reward model to construct our preference simulator. Given a query, comprising a prompt and two potential responses, the preference simulator produces binary feedback indicating a preference between the two responses by first computing scores for each of the two responses. To simulate preference probabilities from these scores, we employ the Bradley-Terry model Bradley & Terry (1952), a well-established methodology in decision analysis. While we make use of binary feedback sampled from the preference probabilities in the training pipeline, we directly use the preference probabilities in the assessment pipeline. The direct use of preference probabilities in our assessment pipeline is to reduce stochasticity in evaluation.

The oracle reward model itself is constructed with a two-part architecture, featuring a torso responsible for producing embeddings and a linear layer head that generates scalar estimates. The torso is initialized to the pre-trained Gemini Pro transformer torso, while the linear head uses Xavier initialization (Glorot & Bengio, 2010). The training process involves optimizing both the torso and reward head through cross-entropy loss function applied to the Anthropic Helpfulness and Harmlessness datasets (Bai et al., 2022).

Our oracle reward model attains an accuracy of 0.755 on the Anthropic Helpfulness and 0.748 on the Anthropic Harmlessness eval datasets. Notably, this level of performance is higher than the performance of the largest models reported in (Bai et al., 2022).

Note that, since Gemini Pro is far larger than Gemini Nano, choices are made by a much more complex model than that available to the agent. This difference in scale is intended to reflect the fact that humans may exhibit more complex behavior than that modeled by the agent.

B. Hyperparameter Selection for Experiments

We tune the hyperparameters of our agent to optimize performance. These hyperparameters include the l2 regularization strength, learning rate, and the number of stochastic gradient descent (SGD) steps executed after each epoch of interaction. Every stochastic gradient descent (SGD) step in our training process is executed on a batch of data randomly sampled from the agent's replay buffer.

We sweep the learning rate over $\{1e-6, 1e-5, 1e-4\}$ for both point estimate and ENN reward models and pick the best learning rate. We found that the best learning rate is consistent across both our joint nll experiments described in Section 5.3 and our active learning experiments.

To adapt to the evolving nature of the data collection process, we implement a strategy of decay for the regularization strength. The regularization strength, denoted as λ in Equations 1 and 3, is adjusted in accordance with the volume of data accumulated by the agent. Specifically, for each stochastic gradient descent (SGD) step carried out at every epoch on a batch comprising $B = 32$ preference data points from the replay buffer, we incorporate a decayed regularization strength given by $\lambda = \frac{32\times\lambda'}{|\mathcal{D}|}$, where $\mathcal{D}$ represents the total number of feedback data points amassed by the agent, and we tune λ' by sweeping across $\{0.1, 1, 10, 100, 1000\}$ for all the agents.

We also swept over the number of sgd steps performed after each epoch of interaction from $\{1, 10, 100\}$. We observed that `infomax` agent benefits from running for more sgd steps while the performance of other agents deteriorates beyond a point due to over fitting.

In the case of ENN models, we also tune the output scale parameter, responsible for regulating the diversity of ensemble particles. In specific, we sweep over values $\{0.1, 1, 10\}$ and pick the value which leads to best performance per agent.

Futhermore, we also tuned the number of hidden units for a two-layer MLP in point estimate model by sweeping over $\{128, 256, 512, 1024, 2048\}$ in the context of our uncertainty estimation experiments detailed in Section 5.3. Despite our thorough exploration, we observed no substantial enhancement in performance associated with an increase in hidden units. Consequently, we opted to employ 128 hidden units for MLPs across all of our experimental results presented in this paper.

C. Exploration Algorithms with a Single Point Estimate

In this section, we evaluate the performance of the `Boltzmann` algorithm across various temperature values. We vary the temperature parameter, denoted as τ, in the Boltzmann exploration scheme (refer to Algorithm 4). The range of temperatures explored includes $\tau \in \{1e-4, 1e-2, 1e-1, 0, 1, 10, 100, 1000\}$. The corresponding performance of the `Boltzmann` agent under different τ values is illustrated in Figure 9.Notably, we observe optimal performance for `Boltzmann` agents with smaller temperatures. Additionally, our findings affirm expectations that `Boltzmann` agents with very high temperatures exhibit performance akin to a passive agent.

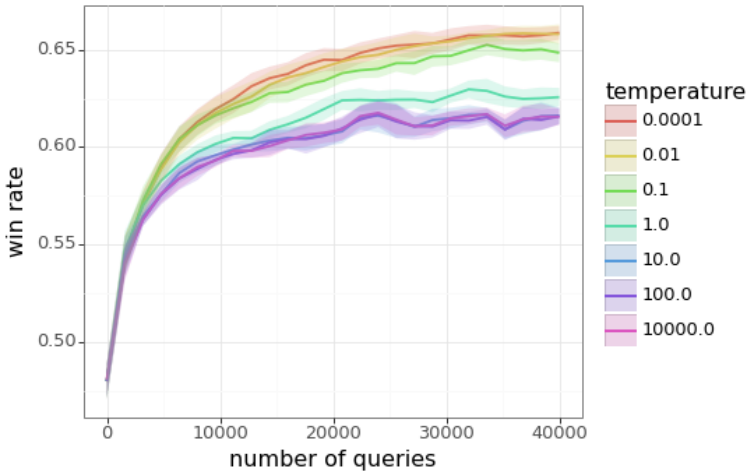

Figure 9 | This plot demonstrates the performance of Boltzmann agent across different temperatures.

We additionally experimented with a variant of the `Boltzmann` scheme known as `Greedy-Boltzmann`, as outlined in Algorithm 7. In the `Greedy-Boltzmann` approach, one response is chosen greedily, relying on the reward model, while the selection of the other response follows the Boltzmann exploration scheme. Conceptually, `Greedy-Boltzmann` can be viewed as a modification of Boltzmann with two temperatures, denoted as τ_1 and τ_2, where τ_1 is fixed at 0, and τ_2 is subject to variation.

Algorithm 7 Greedy-Boltzmann

input: x, π, r
hyperparams: τ, N

1: sample responses $\tilde{y}_1, \ldots, \tilde{y}_N \sim \pi(\cdot|x)$
2: select response $i \in \arg\max_n r(x, y_n)$
3: probs $q_n = \frac{\exp(r(x,\tilde{y}_n)/\tau)}{\sum_{n'=1,n\neq i}^{N} \exp(r(x,\tilde{y}_{n'})/\tau)} \forall n \neq i,\ q_i = 0$
4: sample $i' \sim q$

return $y_i, y_{i'}$

The performance of the `Greedy-Boltzmann` variant across different temperatures is illustrated in

Figure 10. Notably, after fine-tuning the temperature parameter, the results indicate that `Greedy-Boltzmann` doesn't improve over the performance achieved by the standard `Boltzmann` agent, as presented in Algorithm 4.

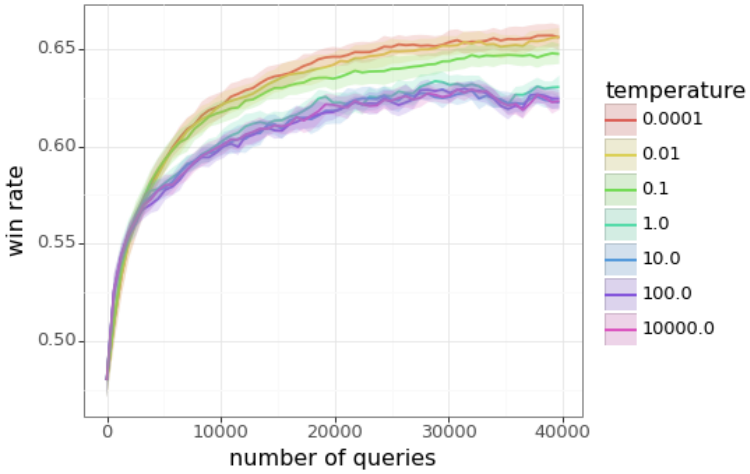

Figure 10 | Performance of Greedy-Boltzmann across different temperatures for Boltzmann. We can see that best Greedy-Boltzmann and best Boltzmann agent perform very similarly, and Greedy-Boltzmann doesn't offer an advantage over Boltzmann.

The `Boltzmann` and `Greedy-Boltzmann` agents can be conceptualized as approximating various exploration strategies determined by the temperatures used in Algorithms 4 and 7. This encompasses examples such as "greedy" exploration, involving the selection of the two best responses; "greedy-uniform" exploration, where the first response is chosen greedily and the second is randomly selected; and "passive" exploration, where both responses are sampled randomly. Therefore, when evaluating the performance of `Boltzmann` and `Greedy-Boltzmann`, we are essentially assessing a broad spectrum of exploration schemes derived from a point estimate reward model.