Sample-Efficient Alignment for LLMs

Zichen Liu **Changyu Chen** **Chao Du**† **Wee Sun Lee** **Min Lin**
Sea AI Lab National University of Singapore Singapore Management University
`{liuzc,chency,duchao,linmin}@sea.com` `{zichen,leews}@comp.nus.edu.sg`

Abstract

We study methods for efficiently aligning large language models (LLMs) with human preferences given budgeted online feedback. We first formulate the LLM alignment problem in the frame of contextual dueling bandits. This formulation, subsuming recent paradigms such as online RLHF and online DPO, inherently quests for sample-efficient algorithms that incorporate *online active exploration*. Leveraging insights from bandit theory, we introduce a unified algorithm based on **Thompson sampling** and highlight its applications in two distinct LLM alignment scenarios. The practical agent that efficiently implements this algorithm, named **SEA** (**S**ample-**E**fficient **A**lignment), is empirically validated through extensive experiments across three model scales (1B, 2.8B, 6.9B) and three preference learning algorithms (DPO, IPO, SLiC). The results demonstrate that **SEA** achieves highly sample-efficient alignment with oracle's preferences, outperforming recent active exploration methods for LLMs. Additionally, we release the implementation of **SEA** together with an efficient codebase designed for **o**nline **a**lignmen**t** of LLMs, aiming to accelerate future research in this field.

`https://github.com/sail-sg/oat`

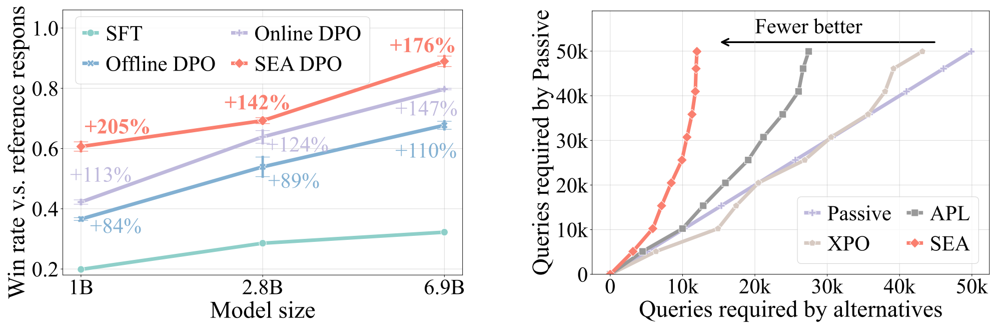


Figure 1: Win rate comparison of model responses against reference responses on the TL;DR task, judged by the preference oracle. All compared methods use the same optimization method (DPO). **(Left)** Performance improvements at convergence over SFT models achieved by offline (Offline DPO), passively online (Online DPO), and our *active exploration* (**SEA** DPO) methods. **(Right)** The number of queries required by the passively online method (Passive) versus that by different active exploration methods to attain various levels of win rates. **SEA** achieves the best sample efficiency for online alignment compared to XPO and APL.

1 Introduction

LLMs have shown remarkable capabilities in various tasks, making it increasingly crucial to align them with human values. Existing alignment approaches, typically formulated using RLHF [14, 58,

†Corresponding author.

5, 45], rely heavily on a huge amount of human annotations to provide preference feedback. As a result, the availability of high-quality human annotations becomes a major bottleneck in practical alignment scenarios. This poses a challenging and under-explored research question:

How to align LLMs sample-efficiently?

In this work, we formalize LLM alignment as a contextual dueling bandit (CDB) [77, 17] problem. Within this formulation, we identified two scenarios in LLM alignment that correspond to two settings in bandit problems: **(1)** Aligning LLMs online with real users' feedback corresponds to the *Exploit & Explore (E&E)* setting [51, 3] in bandits, where the learning algorithm should minimize the cumulative regret (or cumulative loss), because the quality of *every* response to real users matters. See Figure 10 for an example where the ChatGTP system asks end users to give preference feedback. **(2)** Aligning LLMs via crowdsourcing [] corresponds to the *Best Arm Identification (BAI)* setting [9, 2] in bandits, where the learning objective is instead producing well-aligned LLMs with the minimum labeling cost. Both settings quest for sample-efficient LLM alignment algorithms.

Leveraging algorithmic insights from bandit theory, we propose a unified alignment algorithm for the above two settings based on Thompson sampling (TS) [63], which incorporates **active exploration** during the **online** interaction between LLMs and humans to improve sample efficiency. We name our method as **SEA** (**S**ample-**E**fficient **A**lignment). Through extensive experiments, **SEA** shows strong empirical results (see Figure 1), consistently achieving higher win rates and improved sample efficiency compared to baseline approaches across three different model scales. Moreover, we develop and open source a highly efficient, distributed learning system for studying online LLM alignment methods (see Appendix F.1), eliminating barriers to empirical comparisons of different algorithms.

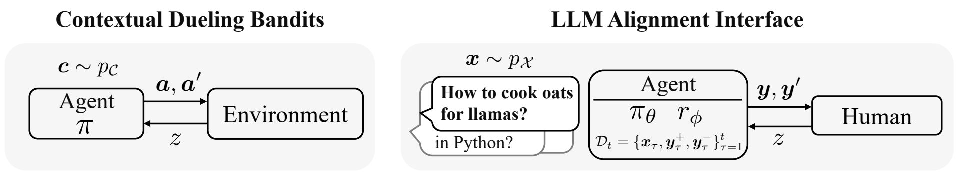

Contextual Dueling Bandits
$c \sim p_{\mathcal{C}}$
Agent
π
a, a'
z
Environment
LLM Alignment Interface
$x \sim p_{\mathcal{X}}$
How to cook oats for llamas?
in Python?
Agent
$\pi_\theta \quad r_\phi$
$\mathcal{D}_t = \{x_\tau, y_\tau^+, y_\tau^-\}_{\tau=1}^t$
y, y'
z
Human

Figure 2: Illustrative comparison between CDB and LLM alignment.

2 LLM alignment as contextual dueling bandits

In this section, we formulate the problem of LLM alignment in the framework of Contextual Dueling Bandits (CDB). We invite readers to Appendix A before proceeding for a review on CDB.

The problem of LLM alignment can be framed as a CDB with their correspondences illustrated in Figure 2. Specifically, a text prompt (cf. context) $\boldsymbol{x} \in \mathcal{X}$ is sampled from a prompt distribution $p_{\mathcal{X}}$. Then, two distinct responses (cf. actions), $\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}$, are chosen by the agent, and presented to human annotators (cf. the environment) for preference ranking. The winning and losing responses are labeled as $(\boldsymbol{y}^+, \boldsymbol{y}^-)$ based on a binary stochastic feedback z. The agent is expected to learn (or so-called align with) human preferences from the interaction experiences $\mathcal{D}$.

A standard assumption is that the human preference follows the Bradly-Terry (BT) model [8]:

$$\mathbb{P}(\boldsymbol{y} \succ \boldsymbol{y}'|\boldsymbol{x}) = \frac{\exp\left(r^\star(\boldsymbol{x}, \boldsymbol{y})\right)}{\exp\left(r^\star(\boldsymbol{x}, \boldsymbol{y})\right) + \exp\left(r^\star(\boldsymbol{x}, \boldsymbol{y}')\right)} = \sigma(r^\star(\boldsymbol{x}, \boldsymbol{y}) - r^\star(\boldsymbol{x}, \boldsymbol{y}')), \tag{1}$$

where σ is the sigmoid function and $r^\star$ encodes human's implicit reward. With this assumption, the regret of LLM alignment can be rewritten as $R_t = r^\star(\boldsymbol{x}, \boldsymbol{y}^\star) - \left(r^\star(\boldsymbol{x}, \boldsymbol{y}) + r^\star(\boldsymbol{x}, \boldsymbol{y}')\right)/2$ [54, 35], where $\boldsymbol{y}^\star$ is the best response for prompt $\boldsymbol{x}$ given the oracle implicit reward, i.e., $r^\star(\boldsymbol{x}, \boldsymbol{y}^\star) \geq r^\star(\boldsymbol{x}, \boldsymbol{y}), \forall \boldsymbol{y} \in \mathcal{Y}$. The von Neumann winner policy is also redefined as

$$\pi^\star \in \arg\max_{\pi} J(\pi), \text{ where } J(\pi) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_{\mathcal{X}}} \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \pi(\cdot|\boldsymbol{x})} [r^\star(\boldsymbol{x}, \boldsymbol{y})] \text{ is the objective,} \tag{2}$$

by substituting Eq. (1) into Eq. (5) and maximizing $P_{p_{\mathcal{C}}, \mathbb{P}}(\pi \succ \pi^\star)$ towards $1/2$. The **two objectives in bandits** (in Appendix A) have their respective applications in LLM alignment, as we have motivated in Section 1. We will present a unified algorithm in Section 3 that aligns LLMs in either scenarios sample efficiently.

Algorithm 1 Thompson sampling for LLM alignment (intractable).

```
   Input: Prompt distribution $p_{\mathcal{X}}$, unknown but queryable preference oracle $\mathbb{P}$.
1: Initialize experience $\mathcal{D}_0 \leftarrow \varnothing$.
2: for $t = 1, \dots, T$ do
3:     Receive a prompt $\boldsymbol{x}_t \sim p_{\mathcal{X}}$.
4:     Sample $r \sim p_r(\cdot|\mathcal{D}_{t-1})$ and set $\boldsymbol{y}_t \leftarrow \arg\max_{\boldsymbol{b}\in\mathcal{Y}} r(\boldsymbol{x}_t, \boldsymbol{b})$.        // Select 1st response $\boldsymbol{y}$.
   // E&E objective: aligning an online system.
5:     repeat
           Sample $r \sim p_r(\cdot|\mathcal{D}_{t-1})$ and set $\boldsymbol{y}_t' \leftarrow \arg\max_{\boldsymbol{b}\in\mathcal{Y}} r(\boldsymbol{x}_t, \boldsymbol{b})$.   // Select 2nd response $\boldsymbol{y}'$.
       until $\boldsymbol{y}_t' \neq \boldsymbol{y}_t$
   // BAI objective: labeling via crowdsourcing.
6:     Set $\boldsymbol{y}_t' \leftarrow \arg\max_{\boldsymbol{b}\in\mathcal{Y}} \mathbb{V}\left[\sigma\left(r(\boldsymbol{x}_t, \boldsymbol{y}) - r(\boldsymbol{x}_t, \boldsymbol{b})\right)\right]$,   // OR select 2nd response $\boldsymbol{y}'$.
           where $\mathbb{V}[\cdot]$ computes variance over the posterior $p_r(\cdot|\mathcal{D}_{t-1})$.
7:     Query $\mathbb{P}$ to label $\{\boldsymbol{y}_t, \boldsymbol{y}_t'\}$, and update experience $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \bigcup \{\boldsymbol{x}_t, \boldsymbol{y}_t^+, \boldsymbol{y}_t^-\}$.
8: end for
                                                            // See Algorithm 2 for a practical version.
```

Importantly, we also note that the CDB formulation of LLM alignment subsumes almost all prior work, and we discuss them thoroughly in Appendix C

3 SEA: sample-efficient alignment for LLMs

3.1 Thompson sampling for LLM alignment

Thomson sampling (TS) typically samples responses according to the probability that they are optimal. We refer readers who are unfamiliar with TS to Appendix B for a brief recap. In the context of LLM alignment, we leverage the BT model assumption (Eq. (1)) to cast the preference oracle $\mathbb{P}$ into the reward oracle $r^\star$, so that we can model the reward posterior $p(r|\mathcal{D})$ similar as in conventional bandits. Note that the LLM agent is fully described by the posterior $p(r|\mathcal{D})$ in this context. We take inspiration from prior work [69, 22], which only discusses non-contextual K-arm bandits and preferential Bayesian optimization problems, and generalize them to the context of LLM alignment and develop a unified algorithm as shown in Algorithm 1.

As Algorithm 1 presents, the first response of the duel is always selected via a typical TS step (Line 4). The selection of the second response varies across different settings. Line 5 will be used for scenarios where preference feedback is collected from online users (the E&E setting). The dueling responses selected in this case will both try to maximize a sampled reward model, so that the online user experience is warranted with best effort. However, such algorithm can have poor asymptotic performance for BAI problems [52], because sub-optimal responses with confidently high rewards might be tried for a long time at the expense of not exploring other potentially better responses. In light of this, Line 6 provides an alternative for scenarios where we could hire annotators for feedback and low-quality but exploratory responses are safe. Specifically, Line 6 selects the second response as the one that maximizes the variance of the preference outcome (Eq. (1)) compared with the first response $\boldsymbol{y}$. This variance quantifies the *epistemic uncertainty* of the reward model, pointing the agent to the maximally informative direction to explore.

However, Algorithm 1 is yet to be practical for LLM alignment for three main reasons. First, existing LLM agents [1, 64] typically consist in a generative model (e.g., a transformer [65]), while the algorithm above is centered around a reward posterior that cannot be easily converted into a generative model. Second, computing and sampling from a reward posterior is intractable for nearly all reward models that can be used for LLMs, which are mostly based on large transformers [32]. Last but not least, even if we managed to approximate a reward posterior, the $\arg\max$ operation for action selection is also intractable since it requires searching over the entire response space $\mathcal{Y}$ which is massive for language.

3.2 Practical implementation

We next introduce our practical implementation techniques that address the aforementioned problems. First, we conduct preference tuning directly from the agent's exploratory on-policy experience to improve the generative model π_θ online. At time t, the loss of π_{θ^t} is:

$$\mathcal{L}_\pi(\theta^t|\mathcal{B}^t, \pi_{\text{ref}}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}^+,\boldsymbol{y}^-)\sim p_{\mathcal{B}^t}} \left[F_{\theta^t}(\boldsymbol{x}, \boldsymbol{y}^+, \boldsymbol{y}^-, \pi_{\text{ref}})\right], \tag{3}$$

where $\mathcal{B}^t$ is a batch of preference data collected online by the online policy π_{θ^t} (thus $\mathcal{B}^t$ is on-policy), and F could be any Direct Alignment from Preferences (DAP) loss [23], such as DPO [48]. Importantly, this makes our method generally applicable to any DAP method that solves Eq. (2) directly from preference data.

We adopt ensemble [31, 47] to implement a tractable reward posterior. In particular, we update a set of reward models independently online using the preference data and a regularized negative log likelihood loss:

$$\mathcal{L}_\mathcal{R}(\Phi^t|\mathcal{D}_t) = \sum_{k=1}^{K} \left(-\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}^+,\boldsymbol{y}^-)\sim p_{\mathcal{D}_t}} \left[\log \sigma \left(r_{\phi_k^t}(\boldsymbol{x}, \boldsymbol{y}^+) - r_{\phi_k^t}(\boldsymbol{x}, \boldsymbol{y}^-)\right)\right] - \lambda||\phi_k^t - \phi_k^0||\right), \tag{4}$$

where $\Phi^t = \{\phi_k^t\}_{k=1}^K$ is the weights of the ensemble of size K, and λ controls the regularization towards its initial weights ϕ_k^0 to retain the diversity across ensemble members [18]. In practice, we train K MLP heads on top of a pretrained transformer. We refer to the ensemble as the Epistemic Reward Model (ERM, denoted as $\mathcal{R}$), with which the posterior sampling simply amounts to randomly picking a ϕ_k from Φ.

With the ERM approximating the reward posterior, we need to further approximate the $\arg\max$ operation in the TS algorithms to select dueling responses. To this end, for any prompt $\boldsymbol{x}$, we sample M responses from the online policy $\pi_{\theta^t}(\cdot|\boldsymbol{x})$ to construct a candidate set $\mathcal{S} = \{\boldsymbol{y}_i\}_{i=1}^M$, and search for the local optimum within $\mathcal{S}$. Intuitively, this Monte Carlo optimization is unbiased if $\mathcal{S}$ uniformly covers $\mathcal{Y}$ (i.e., covering infinitely many possible responses). However, as we optimize π_{θ^t} online with oracle preference data, $\mathcal{S}$ is biased to contain responses with high oracle reward $r^\star$. Bias towards high-$r^\star$ region is generally helpful because it aligns with $\arg\max_{\boldsymbol{b}\in\mathcal{Y}} r(\boldsymbol{x}, \boldsymbol{b})$ to seek high-reward response. However, optimizing π_{θ^t} only with oracle data averages out the epistemic uncertainty of $\mathcal{R}$, hindering the exploration efficiency. To mitigate this issue, we further align π_{θ^t} with the ERM using the same DAP method to encourage π_{θ^t} to propose high-$r_{\phi_k^t}$ responses for individual $r_{\phi_k^t}$. In practice, we implement this by optimizing Eq. (3) over a mixture batch distribution $p_{\mathcal{B}^t_{\text{mix}}} = \gamma p_{\mathcal{B}^t} + (1-\gamma) p_{\mathcal{B}^t_{\text{ERM}}}$, where γ controls the mixture ratio and $\mathcal{B}^t_{\text{ERM}} = \{\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i^+, \tilde{\boldsymbol{y}}_i^-\}_{i=1}^b$ consists of preference data labeled by randomly sampled individual ensemble members $r_{\phi_k^t}$, which facilitates a better approximation of $\arg\max_{\boldsymbol{b}\in\mathcal{Y}} r(\boldsymbol{x}, \boldsymbol{b})$ for any sampled r. Interestingly, learning from mixed preferences further boosts sample efficiency because it utilizes the internal ERM to get pseudo labels instead of querying humans. This relates closely to model-based RL, for which we discuss further in Appendix D. We summarize our practical algorithm (Algorithm 2) in Appendix E.

4 Experimental verification

We build a distributed learning system to facilitate a **truly** online LLM alignment experimentation (instead of iterative training adopted by prior work [42, 16, 78, 71]). We conduct extensive experiments with our learning system and fairly compare with baselines [48, 4, 79, 23] and prior work on active exploration for LLMs [71, 42]. We also perform in-depth analysis of our method. Due to the space constraint, we defer the details of our learning system, experiment configurations, and empirical results to Appendices F and H.

5 Conclusion

In this paper, we study the problem of LLM alignment through the lens of contextual dueling bandits and propose an algorithm based on Thompson sampling to align LLMs from preference feedback. Through extensive empirical investigation, we validate the superior sample efficiency of our method compared to existing baselines. To our knowledge, our work is the first to study active exploration for online LLM alignment with true online experimental verification.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *Conference on learning theory*, pages 41–53, 2010.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[4] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR, 2024.

[5] 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.

[6] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[7] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[8] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[9] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory: 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings 20*, pages 23–37. Springer, 2009.

[10] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[11] Róbert Busa-Fekete, Balázs Szörényi, Paul Weng, Weiwei Cheng, and Eyke Hüllermeier. Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm. *Machine learning*, 97:327–351, 2014.

[12] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24, 2011.

[13] Changyu Chen, Zichen Liu, Chao Du, Tianyu Pang, Qian Liu, Arunesh Sinha, Pradeep Varakantham, and Min Lin. Bootstrapping language models with dpo implicit rewards. *arXiv preprint arXiv:2406.09760*, 2024.

[14] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[15] Nirjhar Das, Souradip Chakraborty, Aldo Pacchiano, and Sayak Ray Chowdhury. Provably sample efficient rlhf via active preference optimization. *arXiv preprint arXiv:2402.10500*, 2024.

[16] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

[17] Miroslav Dudík, Katja Hofmann, Robert E Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits. In *Conference on Learning Theory*, pages 563–587. PMLR, 2015.

[18] Vikranth Dwaracherla, Xiuyuan Lu, Morteza Ibrahimi, Ian Osband, Zheng Wen, and Benjamin Van Roy. Hypermodels for exploration. *arXiv preprint arXiv:2006.07464*, 2020.

[19] Vikranth Dwaracherla, Seyed Mohammad Asghari, Botao Hao, and Benjamin Van Roy. Efficient exploration for llms. In *International Conference on Machine Learning*, 2024.

[20] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[21] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

[22] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D Lawrence. Preferential bayesian optimization. In *International Conference on Machine Learning*, pages 1282–1291. PMLR, 2017.

[23] Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, et al. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*, 2024.

[24] Jian Hu, Xibin Wu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

[25] Shengyi Huang, Michael Noukhovitch, Arian Hosseini, Kashif Rasul, Weixun Wang, and Lewis Tunstall. The n+ implementation details of rlhf with ppo: A case study on tl; dr summarization. *arXiv preprint arXiv:2403.17031*, 2024.

[26] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[27] Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020.

[28] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise comparison and generative fusion. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (ACL 2023)*, 2023.

[29] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

[30] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[31] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[32] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling, 2024.

[33] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[34] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

[35] Xuheng Li, Heyang Zhao, and Quanquan Gu. Feel-good thompson sampling for contextual dueling bandits. *arXiv preprint arXiv:2404.06013*, 2024.

[36] Chris Yuhao Liu, Liang Zeng, Liu Jiacai, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork reward model series. *arXiv preprint arXiv:2410.18451*, 2024.

[37] Zichen Liu, Siyi Li, Wee Sun Lee, Shuicheng Yan, and Zhongwen Xu. Efficient offline policy optimization with a learned model. In *International Conference on Learning Representations*, 2023.

[38] Zichen Liu, Chao Du, Wee Sun Lee, and Min Lin. Locality sensitive sparse encoding for learning world models online. In *International Conference on Learning Representations*, 2024.

[39] Viraj Mehta, Vikramjeet Das, Ojash Neopane, Yijia Dai, Ilija Bogunovic, Jeff Schneider, and Willie Neiswanger. Sample efficient reinforcement learning from human feedback via active exploration. *arXiv preprint arxiv:2312.00267*, 2023.

[40] Luckeciano C Melo, Panagiotis Tigas, Alessandro Abate, and Yarin Gal. Deep bayesian active learning for preference modeling in large language models. *arXiv preprint arXiv:2406.10023*, 2024.

[41] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

[42] William Muldrew, Peter Hayes, Mingtian Zhang, and David Barber. Active preference learning for large language models. In *International Conference on Machine Learning*, 2024.

[43] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[44] Andrew Y Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, volume 1, page 2, 2000.

[45] 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.

[46] Moritz Philipp and Nishihara Robert. Plasma: A high-performance shared-memory object store, 2017. URL `https://arrow.apache.org/blog/2017/08/08/plasma-in-memory-object-store/`.

[47] Chao Qin, Zheng Wen, Xiuyuan Lu, and Benjamin Van Roy. An analysis of ensemble sampling. *Advances in Neural Information Processing Systems*, 35:21602–21614, 2022.

[48] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 37, 2023.

[49] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[50] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020.

[51] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society*, 58:527–535, 1952.

[52] Daniel Russo. Simple bayesian algorithms for best arm identification. In *Conference on Learning Theory*, pages 1417–1418. PMLR, 2016.

[53] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

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

[55] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591, 2021.

[56] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897. PMLR, 07–09 Jul 2015.

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

[58] 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.

[59] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings*, pages 216–224. Morgan Kaufmann, 1990.

[60] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[61] Richard S Sutton, Michael Bowling, and Patrick M Pilarski. The alberta plan for ai research. *arXiv preprint arXiv:2208.11173*, 2022.

[62] Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data. *arXiv preprint arXiv:2404.14367*, 2024.

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

[64] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[66] Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A highly parallel reinforcement learning environment execution engine. In *Advances in Neural Information Processing Systems*, volume 35, pages 22409–22421, 2022.

[67] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[68] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

[69] Huasen Wu and Xin Liu. Double thompson sampling for dueling bandits. *Advances in neural information processing systems*, 29, 2016.

[70] Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*, 2024.

[71] Tengyang Xie, Dylan J Foster, Akshay Krishnamurthy, Corby Rosset, Ahmed Awadallah, and Alexander Rakhlin. Exploratory preference optimization: Harnessing implicit q*-approximation for sample-efficient rlhf. *arXiv preprint arXiv:2405.21046*, 2024.

[72] Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. In *Forty-first International Conference on Machine Learning*, 2024.

[73] Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 2023.

[74] Fan Yang, Gabriel Barth-Maron, Piotr Stańczyk, Matthew Hoffman, Siqi Liu, Manuel Kroiss, Aedan Pope, and Alban Rrustemi. Launchpad: A programming model for distributed machine learning research. *arXiv preprint arXiv:2106.04516*, 2021.

[75] Keming Yang, Zichen Liu, and Philip Cheng. MOSEC: Model Serving made Efficient in the Cloud, 2021. URL https://github.com/mosecorg/mosec.

[76] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[77] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[78] Shenao Zhang, Donghan Yu, Hiteshi Sharma, Ziyi Yang, Shuohang Wang, Hany Hassan, and Zhaoran Wang. Self-exploring language models: Active preference elicitation for online alignment. *arXiv preprint arXiv:2405.19332*, 2024.

[79] Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

[80] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

A A brief review of contextual dueling bandits

Contextual dueling bandits (CDB) [77, 17] is proposed to study online learning problems where feedback consists of relative pairwise comparisons. A CDB problem can be characterized by a tuple $(\mathcal{C}, \mathcal{A}, \mathbb{P})$, where $\mathcal{C}$ is the context space, $\mathcal{A}$ is the action space, and $\mathbb{P} : \mathcal{A} \times \mathcal{A} \times \mathcal{C} \mapsto [0, 1]$ denotes the unknown *preference model*. An agent learns by iteratively interacting with the environment or oracle (i.e., the preference model $\mathbb{P}$) as follows. At each round t of the learning process, a context $\boldsymbol{c}_t \in \mathcal{C}$ is presented to the agent, who needs to take two actions $\boldsymbol{a}_t, \boldsymbol{a}'_t \in \mathcal{A}$ for a "dueling" comparison. The agent then receives stochastic feedback in the form of a comparison result $z_t \sim \mathrm{Ber}\left(\mathbb{P}\left(\boldsymbol{a}_t \succ \boldsymbol{a}'_t | \boldsymbol{c}_t\right)\right)$ from the environment, where $\mathrm{Ber}(\cdot)$ is the Bernoulli distribution and $\succ$ denotes that the first action is preferred.

Regret. The quality of the dueling actions selected by the agent is measured by the *immediate regret*: $R_t = \mathbb{P}(\boldsymbol{a}^\star_t \succ \boldsymbol{a}_t|\boldsymbol{c}_t) + \mathbb{P}(\boldsymbol{a}^\star_t \succ \boldsymbol{a}'_t|\boldsymbol{c}_t) - 1$, where $\boldsymbol{a}^\star_t$ is the best action[1] the agent would take at round t if it had complete knowledge of $\mathbb{P}$. Intuitively, if the agent has learned how to act optimally from round t onwards, it would no longer suffer any regret since its actions would be indistinguishable from the best action ($\mathbb{P}(\boldsymbol{a}^\star_\tau \succ \boldsymbol{a}_\tau|\boldsymbol{c}_\tau) = \frac{1}{2}$ and $R_\tau = 0$ for $\tau \geq t$).

Optimal policy. A policy $\pi \in \Delta_{\mathcal{A}}^{\mathcal{C}}$[2] associates each context $\boldsymbol{c} \in \mathcal{C}$ with a probability distribution $\pi(\cdot|\boldsymbol{c}) \in \Delta_{\mathcal{A}}$ over the action space. The *total preference* of policy π over a policy μ given a context sampling distribution $p_{\mathcal{C}} \in \Delta_{\mathcal{C}}$ and a preference model $\mathbb{P}$ is defined as

$$P_{p_{\mathcal{C}},\mathbb{P}}(\pi \succ \mu) = \mathop{\mathbb{E}}_{\boldsymbol{c}\sim p_{\mathcal{C}}}\left[\mathop{\mathbb{E}}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{c})}\mathop{\mathbb{E}}_{\boldsymbol{a}'\sim\mu(\cdot|\boldsymbol{c})}\left[\mathbb{P}(\boldsymbol{a} \succ \boldsymbol{a}'|\boldsymbol{c})\right]\right]. \tag{5}$$

We adopt the *von Neumann winner* [17] as the measure of optimality, which requires the optimal policy $\pi^\star$ to satisfy that

$$\forall \pi' \in \Delta_{\mathcal{A}}^{\mathcal{C}},\ P_{p_{\mathcal{C}},\mathbb{P}}(\pi^\star \succ \pi') \geq \frac{1}{2}. \tag{6}$$

In words, the von Neumann winner policy should beat or tie with every policy (i.e., is zero-regret) on average.

Learning objectives. The goal of bandit agents is to learn the optimal policy through environment interaction. There are two subtypes of objectives that focus on different learning scenarios. The first type considers the conventional *explore and exploit (E&E)* setting [51, 3], where the agent learns fully online and tries to minimize the cumulative regret over T rounds: $\sum_{t=1}^{T} R_t$. The second type of objective concerns the *best arm identification (BAI)* setting [9, 2], where the agent is only evaluated offline on its average performance, possibly at any round (a.k.a., anytime regret), and tries to learn the optimal policy with minimum interaction.

B A brief recap on Thompson sampling

Thompson sampling (TS) [63] is widely adopted to solve bandit problems at scale due to its great efficiency and strong empirical performance for general online learning problems [12, 53]. A bandit agent with Thompson sampling typically maintains and incrementally updates a posterior distribution of the oracle reward model $p_t(r|\mathcal{D}_t)$ at each round given experience $\mathcal{D}_t$. Meanwhile, the agent takes actions following a greedy policy with respect to a sampled reward model: $\boldsymbol{a}_t = \arg\max_{\boldsymbol{a}} r_t(\boldsymbol{a})$, where $r_t \sim p_t(\cdot|\mathcal{D}_t)$. This simple yet effective algorithm balances exploration and exploitation naturally: when the agent has limited knowledge about the environment, its posterior estimate exhibits high uncertainty so that the sampled greedy policy explores; after gathering necessary experience, the sampled reward model may approximate the oracle well and deliver near optimal policy to exploit.

C Prior work

Before moving on to discuss existing LLM alignment literature in the CDB framework, there is a need to align terminologies used in the bandit context with commonly referred ones in the LLM community. Previously, we use the word "agent" to denote everything except the environment, and refer to its behavior as a "policy", following a standard abstraction in RL [60, 61]. This, for example, applies to the definition of optimal policy in Eq. (2). However, in LLM literature, a "policy" typically refers to the generative language model alone, excluding components like reward models the agent might additionally build. To avoid confusion, from now on we use π_{θ^t} to denote the generative language model (policy) at time t, and r_{ϕ^t} to denote the (optional) reward model learned from preference data D_t collected till time t. We will omit t when the time-indexing is not important in the context.

[1]We assume that a best action $\boldsymbol{a}^\star$ in the sense that $\mathbb{P}(\boldsymbol{a}^\star \succ \boldsymbol{a}|\boldsymbol{c}) \geq \frac{1}{2}, \forall \boldsymbol{a} \in \mathcal{A}$ exists for all context $\boldsymbol{c} \in \mathcal{C}$.

[2]We denote by $\Delta_{\mathcal{A}}^{\mathcal{C}}$ the set of all mappings $\mathcal{C} \mapsto \Delta_{\mathcal{A}}$, where $\Delta_{\mathcal{A}}$ denotes the set of all probability distributions over $\mathcal{A}$.

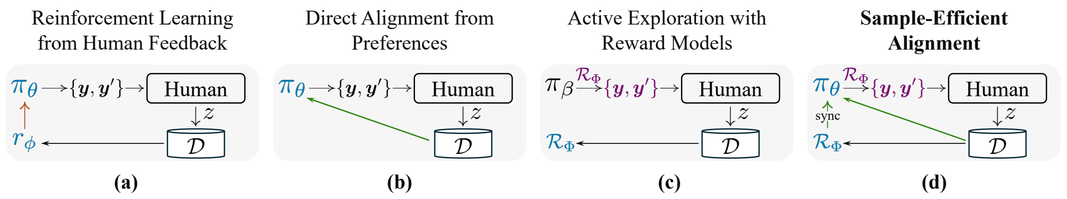

Reinforcement Learning from Human Feedback
Direct Alignment from Preferences
Active Exploration with Reward Models
Sample-Efficient Alignment
π_θ
$\{y, y'\}$
Human
z
$\mathcal{D}$
r_ϕ
π_β
$\mathcal{R}_\Phi$
sync
(a)
(b)
(c)
(d)

Figure 3: Different paradigms for solving the LLM alignment problem in the CDB framework. Note that although all paradigms follow the LLM alignment interface (Figure 2) with the interaction loop, some are actually offline or iteratively online (i.e., loop only once or a few times). Detailed comparisons will be made in Appendix C. We use colors to denote learnable components, RL optimizer, direct optimizer, and active exploration. r_ϕ denotes a point estimate of human's implicit reward, while $\mathcal{R}_\Phi$ refers to an uncertainty-aware reward model.

Commonly adopted RLHF pipelines [14, 58, 5, 45] learn reward models as the first step with a negative log-likelihood loss:

$$\mathcal{L}_r(\phi|\mathcal{D}) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}^+,\boldsymbol{y}^-)\sim p_\mathcal{D}}\left[\log\sigma\left(r_\phi\left(\boldsymbol{x},\boldsymbol{y}^+\right) - r_\phi\left(\boldsymbol{x},\boldsymbol{y}^-\right)\right)\right], \tag{7}$$

where $\mathcal{D}$ is collected by querying human annotators using a behavior policy π_{ref}, which is typically obtained by supervised fine-tuning. Afterwards, *offline* RL [33] is conducted with respect to the learned reward r_ϕ internally within the agent in the CDB framework (Figure 3**(a)**). However, the learned model π_θ might be inaccurate at regions out of the distribution (o.o.d.) of π_{ref} because little training data can be collected. A typical remedy is to incorporate a pessimistic objective to combat such distributional shift. To this end, we rewrite the objective of von Neumann winner policy in Eq. (2) as

$$J(\pi_\theta) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_\mathcal{X}}\ \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\pi_\theta(\cdot|\boldsymbol{x})}\left[\underbrace{r_\phi(\boldsymbol{x},\boldsymbol{y})}_{\text{estimated } r^\star} - \underbrace{\eta\log\frac{\pi_\theta(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})}}_{\text{o.o.d. reward penalty}}\right] \tag{8}$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_\mathcal{X}}\left[\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\pi_\theta(\cdot|\boldsymbol{x})}\left[r_\phi(\boldsymbol{x},\boldsymbol{y})\right] - \eta D_{\text{KL}}(\pi_\theta(\cdot|\boldsymbol{x})||\pi_{\text{ref}}(\cdot|\boldsymbol{x}))\right], \tag{9}$$

which involves a KL penalty widely used for language model fintuning [27, 72]. PPO [57] as an *RL optimizer* naturally suits Eq. (9) well due to its KL-regularized trust region policy update [56] and has been used widely. Though effective in aligning LLMs with human values, RLHF is either performed with iterative interactions, or with an offline dataset to learn the internal reward model, instead of with online interactions. Moreover, Applying RL optimizer with the internal reward model is complex and requires many subtle tricks to stabilize the training, limiting its broader application [25]. Direct alignment from preferences (DAP), introduced by DPO [48], simplifies training by eliminating the need for an internal reward model (Figure 3**(b)**). However, many early works following DPO either rely on offline datasets [4, 21, 70, 41] or engage in iterative interactions [16]. OAIF [23] takes a step forward by enabling fully online learning, improving sample efficiency over its offline and iterative counterparts. Nevertheless, these methods still employ passive exploration strategies.

A line of work [39, 15, 40, 19] adopts the bandit formulation and focuses on reward model learning, incorporating active exploration and online interactions (Figure 3**(c)**). They generate responses from a fixed policy, π_β, and utilize a reward model to select the dueling responses. However, their performance is limited by the sub-optimal sample efficiency due to the non-adaptive proposal distribution π_β. Built on recent OAIF [23], several works have proposed to incorporate active exploration, either by adding an optimistic term in the loss function [78, 71], or by actively selecting dueling responses using implicit rewards induced from DPO training [42]. Yet, these methods are tightly coupled with DPO and are not compatible with other direct optimizers. Given their relevance to our approach, we will include comparisons with these methods in our experiments when using DPO as the direct optimizer. We summarize the prior work in Appendix C.

D On connections with single-step RL

By viewing contextual dueling bandits as *single-step* preference-based RL (PbRL) [11, 68] problems, we can interpret paradigms shown in Figure 3 from the RL perspective.

RLHF approaches (Figure 3a) are instances of **offline model-based RL** [29, 76, 55, 37, 62], where they learn a reward model (no need for a transition model since the prompt-response interaction is single-step) of the environment from a batch of offline collected data, and train a policy (i.e., LLM) to maximize the return (i.e., expected one-step reward) with respect to the *learned* reward.

In contrast, DAP methods (Figure 3b) are similar to **policy-based model-free RL** algorithms, e.g., REINFORCE [67] which conducts policy gradient update:

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{X}}\mathbb{E}_{\boldsymbol{y}\sim\pi_\theta(\cdot|\boldsymbol{x})}\left[R(\boldsymbol{x},\boldsymbol{y})\nabla_\theta\log\pi_\theta(\boldsymbol{y}|\boldsymbol{x})\right], \tag{10}$$

Table 1: A summary of prior work. π_θ denotes the proposal policy that is continuously updated based on newly collected preference data, while π_β denotes a fixed proposal policy. Algorithms that encompass online interaction, active exploration, and learnable π_θ offer the best sample efficiency. Notably, only three methods (listed at the bottom of the table) satisfy these characteristics, and we include them for comparisons in our experiments.

	Method	Exploration		Interaction			Proposal Policy	
		Active	Passive	Online	Iterative	Offline	π_θ	π_β
RL Optimizer	[14]		✓		✓	✓	✓	
	[58]		✓		✓	✓	✓	
	[5]		✓		✓	✓	✓	
	[45]		✓		✓	✓	✓	
Direct Optimizer	[79]		✓			✓	✓	
	[48]		✓			✓	✓	
	[4]		✓			✓	✓	
	[41]		✓			✓	✓	
	[73]		✓		✓		✓	
	[23]		✓	✓			✓	
	[39]	✓		✓				✓
	[15]	✓		✓				✓
	[40]	✓		✓				✓
	[19]	✓		✓				✓
	[78]	✓		✓			✓	
	[71]	✓		✓			✓	
	[42]	✓		✓			✓	

where $R(\boldsymbol{x}, \boldsymbol{y})$ is the return (i.e., cumulative reward) of the trajectory. To connect with DAP, we could set R as arbitrary scalar values based on the binary preference outcomes, e.g., $R(\boldsymbol{x}, \boldsymbol{y}^+) = \zeta$ and $R(\boldsymbol{x}, \boldsymbol{y}^-) = -\zeta$ for preference triplet $\{\boldsymbol{x}, \boldsymbol{y}^+, \boldsymbol{y}^-\}$. In this way we could rewrite Eq. (10) as

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{X}}\mathbb{E}_{\boldsymbol{y},\boldsymbol{y}'\sim\pi_\theta(\cdot|\boldsymbol{x})}\mathbb{E}_{(\boldsymbol{y}^+\succ\boldsymbol{y}^-)\sim\mathbb{P}}\left[\zeta\left(\nabla_\theta \log \pi_\theta(\boldsymbol{y}^+|\boldsymbol{x}) - \nabla_\theta \log \pi_\theta(\boldsymbol{y}^-|\boldsymbol{x})\right)\right], \tag{11}$$

by repeating action sampling twice and querying the oracle for preference labeling. This matches the gradient direction of contrastive DAP losses (e.g., see Section 4 of DPO [48]) if we optimize them online [23].

Additionally, active reward learning from behavior policy's data distribution (Figure 3c) can be regarded as **inverse RL** [44], which tries to recover environment's reward function given expert trajectories. In the context of LLM alignment, the preference data $\{\boldsymbol{x}, \boldsymbol{y}^+, \boldsymbol{y}^-\}_{i=1}^N$ directly encodes human's implicit reward $r^\star$, which can be inversely learned with assumptions such as the BT model [8]. However, existing methods belonging to this paradigm mostly rely on a fixed (and suboptimal) behavior policy for response sampling, whose coverage inherently limits the quality of the recovered reward function.

Last but not least, **SEA** depicted in Figure 3d resembles a class of **online model-based RL** algorithms, known as Dyna [59, 26], that learns a *world model* from environment experience and trains a base agent (consisting of reactive policies and value functions) from both environment experience and model experience. Compared to model-free methods, Dyna naturally enables more sample-efficient learning by planning with the learned world model to update the base agent. In **SEA**, we learn the reward model online and update the LLM (i.e., the reactive policy) with model-planing experience by mixed preference learning (**??**). Online model-based RL algorithms could suffer from catastrophic forgetting in the face of nonstationary data [38], and we leave it for future work. Overall, this model-based RL formulation is powerful and explains popular LLM techniques, e.g., Best-of-N sampling [64] can be viewed as planning for acting, which trades compute for performance. We believe it is a promising path leading us to unlock superhuman capabilities of LLMs.

E Algorithm details

While Algorithm 1 presents our Thompson sampling algorithm for LLM alignment, it is intractable and centered around the reward posterior modeling. We next present a practical sample-efficient alignment agent that learns both an LLM policy and an epistemic reward model online in Algorithm 2.

In Algorithm 2, we describe an online setting where a single example is processed at each time t (batch size $b = 1$). This is mainly for notational convenience, while in implementation we set b to be the training batch size (e.g., 128). We instantiate the reward posterior with an epistemic reward model, which allows for efficient incremental update and sampling. We also replace the global optimization ($\arg\max_{\boldsymbol{b}\in\mathcal{Y}}$) with a policy-guided

Algorithm 2 Sample-efficient alignment (**SEA**) for LLMs

```
Input: Reference policy $\pi_{\text{ref}}$, DAP loss function $F$, prompt distribution $p_{\mathcal{X}}$, unknown but queryable preference oracle $\mathbb{P}$, mixture ratio $\gamma$.
 1: Initialize experience $\mathcal{D}_0 \leftarrow \varnothing$, policy $\pi_{\theta^0} \leftarrow \pi_{\text{ref}}$, and ERM weights $\Phi^0 = \{\phi_k^0\}_{k=1}^K$ randomly.
 2: for $t = 1, \ldots, T$ do
 3:     Receive a prompt $\boldsymbol{x}_t \sim p_{\mathcal{X}}$.
 4:     Sample $M$ responses $\boldsymbol{y}_t^i \sim \pi_{\theta^{t-1}}(\cdot|\boldsymbol{x}_t)$ to construct $\mathcal{S}_t = \{\boldsymbol{y}_t^i\}_{i=1}^M$.
 5:     Sample $\phi \sim \text{Uniform}(\Phi^{t-1})$ and set $\boldsymbol{y} \leftarrow \arg\max_{\boldsymbol{b}\in\mathcal{S}_t} r_\phi(\boldsymbol{x}_t, \boldsymbol{b})$.      // Select 1st response $\boldsymbol{y}$.
    // E&E objective: aligning an online system.
 6:     repeat
            Sample $\phi \sim \text{Uniform}(\Phi^{t-1})$ and set $\boldsymbol{y}' \leftarrow \arg\max_{\boldsymbol{b}\in\mathcal{S}_t} r_\phi(\boldsymbol{x}_t, \boldsymbol{b})$.      // Select 2nd response $\boldsymbol{y}'$.
        until $\boldsymbol{y}' \neq \boldsymbol{y}$
    // BAI objective: labeling via crowdsourcing.
 7:     Set $\boldsymbol{y}' \leftarrow \arg\max_{\boldsymbol{b}\in\mathcal{S}_t} \mathbb{V}_\phi\left[\sigma\left(r_\phi(\boldsymbol{x}_t, \boldsymbol{y}) - r_\phi(\boldsymbol{x}_t, \boldsymbol{b})\right)\right]$,      // OR select 2nd response $\boldsymbol{y}'$.
            where $\mathbb{V}_\phi[\cdot]$ computes variance across ensemble members of $\Phi^{t-1}$.
 8:     if $g < \gamma$ for $g \sim \text{Uniform}(0, 1)$ then
            Label $\{\boldsymbol{y}, \boldsymbol{y}'\}$ with $\mathbb{P}$ to obtain $\mathcal{B}_t = \{\boldsymbol{x}_t, \boldsymbol{y}_t^+, \boldsymbol{y}_t^-\}$ and update experience $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \bigcup \mathcal{B}_t$.
        else
            Use $\mathcal{R}_{\Phi^{t-1}}$ to get synthetic labels and obtain $\mathcal{B}_t = \{\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i^+, \tilde{\boldsymbol{y}}_i^-\}$.
        end if
 9:     Update ERM with the regularized NLL loss (Eq. (4)):
            $$\Phi^t \leftarrow \Phi^{t-1} - \alpha_{\mathcal{R}} \nabla_\Phi \mathcal{L}_{\mathcal{R}}(\Phi^{t-1}|\mathcal{D}_t).$$
                                                                                      // Reward learning.
10:     Update policy with the direct optimizer (Eq. (3)):
            $$\theta^t \leftarrow \theta^{t-1} - \alpha_\pi \nabla_\theta \mathcal{L}_\pi(\theta^{t-1}|\mathcal{B}_t, \pi_{\text{ref}}, F).$$
                                                                                      // Policy learning.
11: end for
```

local search among proposals sampled from the latest online policy $\pi_{\theta^{t-1}}$. At each time t, we update ERM weights Φ with m randomly sampled batches from the experience $\mathcal{D}_t$. We find setting $m = 5$ suffices to achieve reasonable accuracy. The policy parameters θ are updated using mixed preference data, with proportion γ being the real environment experience and $(1 - \gamma)$ from the ERM's synthetic experience. Note that the synthetic experience is not added into $\mathcal{D}_t$ to ensure reward learning always uses ground truth environment data.

We consider the following three direct optimizers in our experiments:

- DPO [48]:
$$F_\theta(\boldsymbol{x}, \boldsymbol{y}^+, \boldsymbol{y}^-, \pi_{\text{ref}}) = -\log\sigma\left(\beta \log \frac{\pi_\theta\left(\boldsymbol{y}^+|\boldsymbol{x}\right)\pi_{\text{ref}}\left(\boldsymbol{y}^-|\boldsymbol{x}\right)}{\pi_{\text{ref}}\left(\boldsymbol{y}^+|\boldsymbol{x}\right)\pi_\theta\left(\boldsymbol{y}^-|\boldsymbol{x}\right)}\right) \tag{12}$$
- IPO [4]:
$$F_\theta(\boldsymbol{x}, \boldsymbol{y}^+, \boldsymbol{y}^-, \pi_{\text{ref}}) = \left(\log\left(\frac{\pi_\theta\left(\boldsymbol{y}^+|\boldsymbol{x}\right)\pi_{\text{ref}}\left(\boldsymbol{y}^-|\boldsymbol{x}\right)}{\pi_{\text{ref}}\left(\boldsymbol{y}^+|\boldsymbol{x}\right)\pi_\theta\left(\boldsymbol{y}^-|\boldsymbol{x}\right)}\right) - \frac{1}{2\beta}\right)^2 \tag{13}$$
- SLiC [79]:
$$F_\theta(\boldsymbol{x}, \boldsymbol{y}^+, \boldsymbol{y}^-, \pi_{\text{ref}}) = \max\left(0, 1 - \beta\log\frac{\pi_\theta\left(\boldsymbol{y}^+|\boldsymbol{x}\right)\pi_{\text{ref}}\left(\boldsymbol{y}^-|\boldsymbol{x}\right)}{\pi_{\text{ref}}\left(\boldsymbol{y}^+|\boldsymbol{x}\right)\pi_\theta\left(\boldsymbol{y}^-|\boldsymbol{x}\right)}\right) \tag{14}$$

where β controls the rate of deviation of π_θ from π_{ref}.

F Experimentation setup

In this section we elaborate the experimentation setup we employ to validate our algorithm and fairly compare to other online alignment baselines. We start by introducing the distributed learning system[3] we build for experimenting online LLM alignment with simulated preference oracle (Appendix F.1), then provide all the experiment details including models, data and performance metrics, etc. (Appendix F.2).

[3]We will open-source the codebase after the reviewing process.

F.1 Distributed learning system

The interactive nature of LLM alignment necessitates an integrated online learning system that simulates the interface depicted on the right of Figure 2. The absence of a performant open-source online alignment system has restricted many existing works to only a few iterations of batch learning [42, 16, 13, 78, 71], which creates a mismatch with their theories that typically require a large number of online interaction rounds. Even worse, such absence also makes the comparison between different LLM exploration methods difficult, often restricting evaluations to the simplest iterative DAP baselines [78, 71].

To fill this gap, we build a highly efficient learning system for experimenting with online LLM alignment algorithms. We notice that the computational bottleneck lies in online response sampling (i.e., autoregressive generation) and preference labeling (e.g., human, large RMs, or large LLMs), which mirrors the slow actor-environment interaction seen in RL systems. Inspired by distributed deep RL systems which spawn many actors or environments in parallel [20, 66], we design an Actor-Learner-Oracle architecture for online LLM alignment, which is depicted in Figure 4. The three types of workloads (i.e., actor, learner and oracle) are heterogeneous and require different optimization. In particular, we adopt vLLM [30] for the actor to accelerate the autoregressive response generation. We also use DeepSpeed's ZeRO [50, 49] strategies to enhance the memory efficiency of the learner. The updated model weights are broadcasted from the learner master to all actors after every optimizer step efficiently via NCCL, similar to Hu et al. [24]. Furthermore, to improve the scalability, we wrap the oracle RM as a service using Mosec [75], which supports dynamic batching and parallel processing, to minimize preference query latency. Finally, we leverage DeepMind Launchpad [74] to compose all workloads into a distributed program and adopt Plasma [46] to efficiently transfer data across process boundaries.

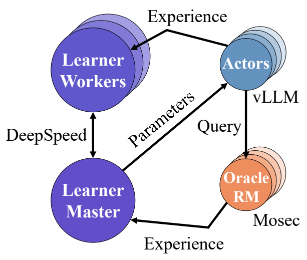


Figure 4: The learning system for experimenting online LLM alignment algorithms.

We benchmark our system's efficiency against a concurrent implementation of online DPO by HuggingFace[4], which utilizes only DeepSpeed for memory optimization. Our system achieves up to **2.5×** latency reduction compared to this counterpart, demonstrating its computational efficiency. Detailed benchmarking methods and results are presented in Appendix G. Our codebase, **oat** (online alignment), along with the implementation of **SEA**, is open-sourced at `https://github.com/sail-sg/oat` to accelerate future research in online LLM alignment.

F.2 Experiment details

Models. We experiment three model scales (1B, 2.8B, 6.9B) from the Pythia family [7]. We take pretrained SFT models from [25] as π_{ref} for the initial model all experiments.

Reward oracle. We simulate the process of human feedback with a strong scalar reward model and refer it as reward oracle. We choose `Skywork-Reward-Llama-3.1-8B`[5] [36], which is top-ranked in RewardBench leaderboard [32], as the reward oracle.

Epistemic reward model. We build ERM on top of a pretrained 0.4B transformer [28], by removing its head and adding an ensemble of MLPs. The size of ensemble is set to 20, and all MLPs contain 2 hidden layers of 128 nodes. Note that the ERM is chosen to be much smaller than the reward oracle following prior work [19], which reflects the fact that human preference may be more complex than what the agent can model.

Data. We employ the widely adopted `TL;DR` dataset [58] for our experiments. It consists of Reddit posts as prompts, and the agent is required to give summaries that align with human preferences. We fix 50k prompts for training and limit the query budget to 50k as well.

DAP methods. We adopt three DAP methods to thoroughly validate our algorithm, including DPO [48], IPO [4] and SLiC [79].

Baselines. We include the offline and online variants of different DAP methods as baselines, which are studied by [23]. Additionally, we compare with two active exploration baselines built on online DPO: APL [42] and XPO [71]. We omit the comparison with SELM [78] since SELM and XPO share a very similar algorithmic design.

Metrics. We use the win rate of agent's responses against reference responses judged by the reward oracle as the performance metric. This metric can reflect both the agent's cumulative regret and anytime regret (i.e., average performance). In the E&E setting, we measure the "online" win rate of the agent's dueling responses that are executed during experience collection. In the BAI setting, we measure the "offline" win rate by evaluating

[4] `https://huggingface.co/docs/trl/main/en/online_dpo_trainer`.
[5] `https://huggingface.co/Skywork/Skywork-Reward-Llama-3.1-8B`.

the agent's responses given a fixed set of holdout prompts periodically. We mainly focus on the BAI setting because crowdsourcing seems a major scenario for most practitioners, and present one set of experiments for comparing different exploration strategies in both settings. When the comparison is only made within a model scale, we report the relative win rate against the initial STF models. When the comparison is across scales (Figure 1 Left), we report the absolute win rate against the ground truth responses in the dataset.

Hyperparameters. We set $\beta = 0.1$ for DPO and $\beta = 0.2$ for SLiC and find they are robust for all scales. We tune β from $\{0.2, 0.3, 0.5, 1.0\}$ for IPO across scales and report the best performing results. We sample 20 on-policy responses with a temperature of 0.7 during training, and use greedy decoding for offline evaluation (BAI's metric). We use the Adam optimizer with learning rate of $5e-7$ and cosine scheduling. We initialize the mixture ratio γ of **SEA** as 1 and adjust it to 0.7 after a burn-in period of 1k samples. We follow the recommended hyperparameters of APL and XPO from their papers.

Statistical significance. There are various factors to introduce randomness during online learning. We thus launch 3 independent runs for every experiment with different random seeds. All the results are reported along with standard errors indicating their statistical significance.

Computational resources. Experiments for all scales can be run on 8 A100 GPUs for learner and actors. We host a separate remote server on 16 A100 GPUs for the oracle reward model, so that it can be queried by all concurrently running experiments. All experiments conducted for this research consume about 2 A100 years.

G System benchmarking

We conduct a rigorous benchmarking comparison on the efficiency of online DPO training using our learning system `oat`[6], alongside the `trl`'s implementation[7].

Settings. In alignment with the examples provided by `trl`, we use the TL;DR [58] dataset and evaluate training efficiency at three model scales: 1B, 2.8B and 6.9B parameters for both SFT-ed LLMs and exclusively trained RMs. This is similar to the settings in our experiments (see **??**) except that we fix the reward oracle to be a strong general-purpose RM.

Hardware & Software. All benchmarking experiments are conducted on a single machine with eight A100-40G GPUs and 96 AMD EPYC 7352 CPUs. To ensure fair comparison, we align all key hyperparameters for both `oat` and `trl`. The DeepSpeed ZeRO-2 strategy is employed by default when GPU memory suffices; otherwise, ZeRO-3 or ZeRO-2-offload is utilized as applicable. Notably, the distributed architecture of `oat` provides flexibility in system configuration, enabling adjustments to accommodate memory and computational time constraints. Figure 5 illustrates two example configurations employed in our benchmarking experiments.

- **Config 1** collocates all three workloads on each of the GPUs. Specifically, eight vLLM instances (for actors) and eight Mosec workers (for oracle RMs) are spawned to run independently on each GPU. After a batch of responses is generated (by actors) and labeled (by oracle RMs), it is sent to the learner, which runs on all eight GPUs coordinated through ZeRO strategies for policy learning. The updated policy weights are then broadcasted to all actors for *on-policy* response sampling on subsequent prompt batch. While this configuration maximizes GPU utilization, it requires substantial GPU memory to accommodate all workloads and is thus employed only for 1B scale experiments.
- **Config 2** only collocates actor and oracle workloads on half of the GPUs, reserving the remaining four GPUs exclusively for the learner. This is suited for larger-scale experiments (e.g., 2.8B or 6.9B), where additional GPU memory is allocated to the learner. However, this setup incurs idle time on half of the GPUs due to data dependency, as the learner must await new preference data, and the actor must await updated policies. An alternative is to implement *asynchronous* data collection, where minor data staleness is allowed by using θ_{t-1} to generate data for updating θ_{t+1}. Although this data would not be strictly on-policy, asynchronous training could reduce idle time and enhance GPU utilization. This approach has proven effective in large-scale RL systems [6], and we leave this optimization to future work.

We provide all benchmarking scripts in our codebase[8] for reproducibility.

Results. Benchmarking results for the latency of training a batch of 128 samples are presented in Figure 6. Overall, training with `oat` config 2 demonstrates consistently greater efficiency than `trl`, achieving up to a **2.5**$\times$ reduction in latency at the 2.8B scale.

We next analyze the time costs for individual stages: generate, oracle and learn. Across all scales and configurations, `oat` demonstrates significantly lower *generate* time than `trl`, due to distributed actors utilizing vLLM.

[6] https://github.com/sail-sg/oat.
[7] https://github.com/huggingface/trl/blob/main/trl/trainer/online_dpo_trainer.py.
[8] https://github.com/sail-sg/oat/tree/main/benchmark.

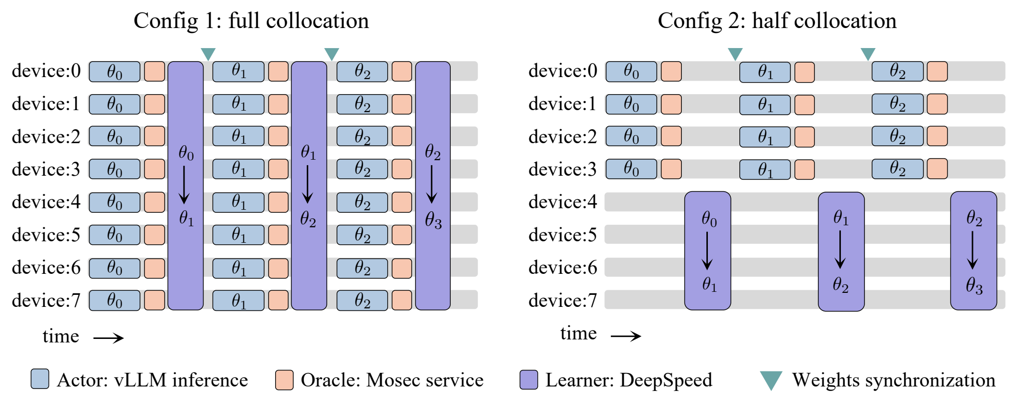


Figure 5: Two example configurations of `oat` used in benchmarking experiments.

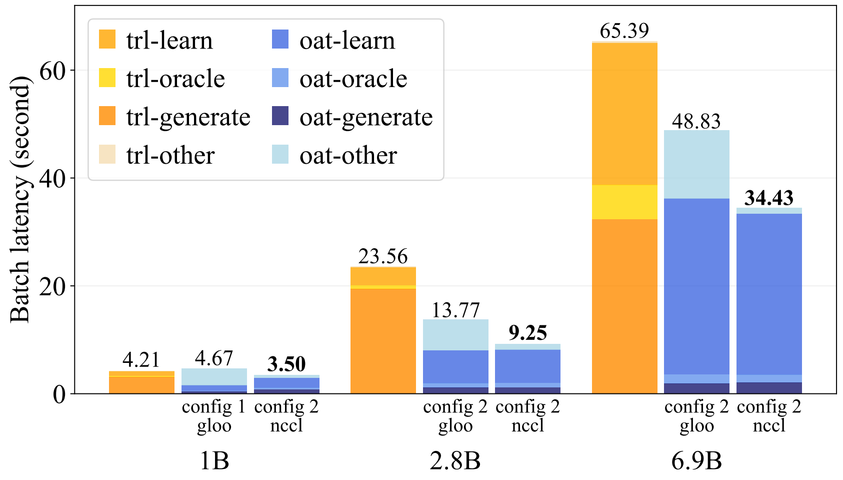


Figure 6: Averaged training latency (over 10 batches, equivalent to 1280 samples) comparing `sail-sg/oat` against `huggingface/trl`.

Additionally, at the 6.9B scale, `oat` requires substantially less *oracle* time than `trl`, as `trl` employs ZeRO-3 to prevent GPU memory overflow, thereby slowing inference. In contrast, `oat` config 2 allows for flexible collocation, enabling oracle RMs hosted via Mosec to operate in parallel without sharding. However, `oat` config 2 incurs longer *learn* time compared to `trl` due to the use of only half the available GPUs. This limitation also explains why, at the 1B scale, config 2 has higher latency than config 1 across all stages.

The *other* category accounts for time costs associated with data loading, tokenization, and communication. Here, inter-process communication is the primary cost, with `trl` showing minimal overhead as all three stages operate within the same process on identical micro-batches, avoiding weight synchronization. By contrast, `oat` requires considerable time to transfer updated policy weights from the learner to all actors. While NCCL is recommended for synchronization over GLOO, it requires older vLLM packages (prior to version 0.4.3), which may lack support for newer LLM architectures. Moreover, NCCL is incompatible with config 1 due to its restriction on the learner master process establishing two separate process groups (one for DeepSpeed, the other for weight synchronization). In summary, we recommend future researchers prioritize `oat` config 2 and employ NCCL when feasible.

H Empirical results

In this section we present our empirical results and analyses. We organize this section into four parts: (1) An overall comparison between **SEA** and baselines across direct optimizers and model scales. (2) An ablation analysis of **SEA**. (3) A comparison of different exploration strategies in E&E and BAI settings. (4) Additional results when aligning with a human simulator by GPT4o-mini.

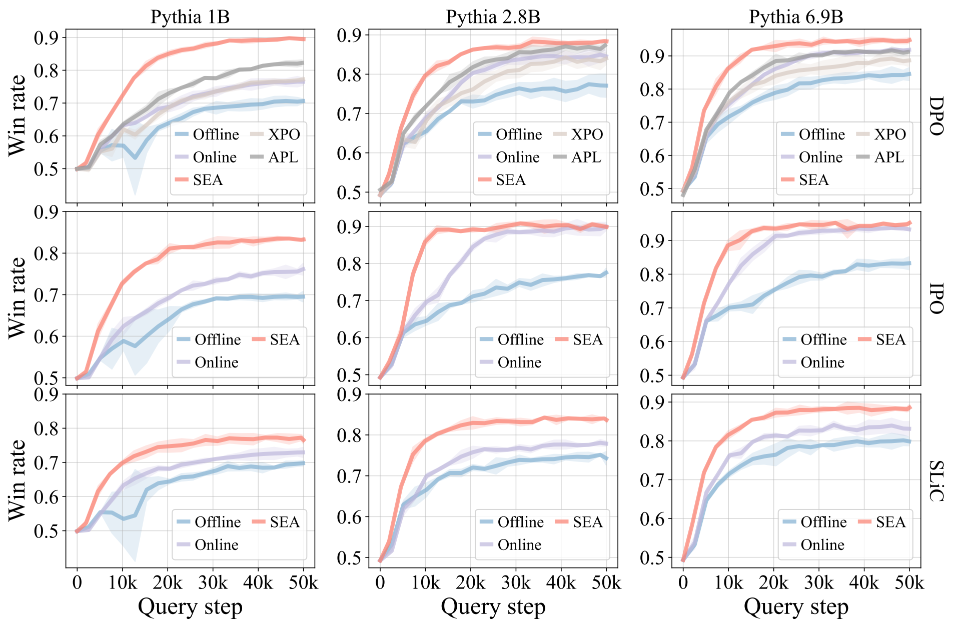


Figure 7: Comparison on the win rates of different agents against SFT models across three model scales and three direct optimizers.

H.1 Overall comparison

We first compare **SEA** with all baselines across three model scales and three direct optimizers. APL and XPO are only compared when we use DPO as the direct optimizer, because they are not compatible with IPO or SLiC. Figure 7 shows the win rate curves versus query steps. Across all settings, `Online` agents improve sample efficiency over their `Offline` counterparts, validating the need of online interaction for alignment algorithms. Focusing on the first row, among prior active exploration methods, `XPO` gives a minor improvement on final performance over `Online` (passive) at 1B scale, but falls short for larger scales. On the other hand, `APL` shows a significant efficiency boost at 1B scale, but the return diminishes when scaling up and it performs almost the same as `Online` at 6.9B scale. Our method, **SEA**, outperforms offline and online passive methods across all scales and all direct optimizers, confirming the role active exploration plays for sample-efficient alignment. Meanwhile, in the special case of using DPO as the direct optimizer, **SEA** also shows superior performance to prior online active exploration methods including `APL` and `XPO`.

Additionally, we note that the choice of direct optimizer matters for both online learning and active exploration. Comparing different optimizers at 1B scale (the first column), all `Offline` agents learn comparably and reach the same level of final performance (about 70% win rate), but SLiC `Online` agent deliver slightly less improvement than DPO and IPO `Online` agents. Besides, when incorporating active exploration, DPO **SEA** agent shows much larger improvement than the other two. This suggests that selecting the most suitable policy optimizer coupled with active exploration would yield the best agent.

H.2 Ablation analysis

Next we decompose **SEA** into different components and ablate their contributions. Table 2 shows three axes that we dissect **SEA** on, including the inference method, exploration strategy and learning components. We construct 7 agent variants from different combinations, which cover two closely related baselines [23, 19]. We show the performance curves of all variants in Figure 9. The left plot compares variants that directly use their policy for inference. It clearly shows the benefits of learning ERM for active exploration (`Variant-2`) and aligning π_{θ^t} with $\mathcal{R}_{\Phi^t}$ (`Variant-3`). Since a reward model is learned within the agent, we can further incorporate inference-time alignment via Best-of-N (BoN) sampling [43, 64]. This also facilitates a comparison between **SEA** and Dwaracherla et al. [19], which learns a similar ERM for both exploration and BoN but does not align the LLM policy. Results on the right plot of Figure 9 suggest a similar trend that `Variant-6` $\succ$ `Variant-5` $\succ$ `Variant-4`. The `Variant-7`, however, ceases to improve after the ERM converges due to the limited performance of its fixed policy.

H.3 Choice of exploration strategies

Recalling that different LLM alignment settings (online system or crowdsourcing) require different exploration strategies to meet their respective learning objectives (Section 2). We investigate three strategies based on

Table 2: Decomposition of different driving factors of online active alignment algorithms.

Variant	Inference (Test)	Exploration	Learn	Remark
1	π_θ	passive	π_θ	Online DAP [23]
2	π_θ	active	$(\pi_\theta, \mathcal{R})$	**SEA** *without* ERM sync (Section 3.2)
3	π_θ	active	$(\pi_\theta \leftrightarrow \mathcal{R})$	**SEA**
4	$\text{BoN}(\pi_\theta, \mathcal{R})$	passive	$(\pi_\theta, \mathcal{R})$	-
5	$\text{BoN}(\pi_\theta, \mathcal{R})$	active	$(\pi_\theta, \mathcal{R})$	-
6	$\text{BoN}(\pi_\theta, \mathcal{R})$	active	$(\pi_\theta \leftrightarrow \mathcal{R})$	**SEA** with Best-of-N sampling
7	$\text{BoN}(\pi_{\text{ref}}, \mathcal{R})$	active	$\mathcal{R}$	Not learn policy [19]

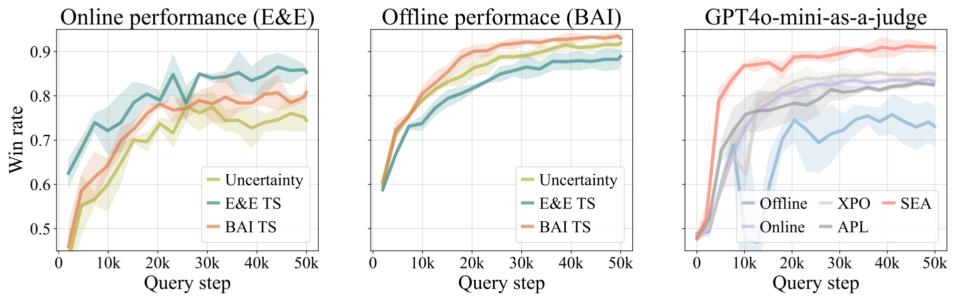


Figure 8: (Left and Middle) Win rate comparison of different exploration strategies measured in E&E and BAI settings. **(Right)** Win rate comparison of different agents when using `GPT4o-mini` to simulate human feedback via LLM-as-a-judge.

posterior sampling and compare them on both online and offline performance. The first strategy focuses on pure exploration. It seeks the pair of dueling responses that exhibits the largest epistemic uncertainty (`Uncertainty`), which is implemented by selecting the pair whose logits difference has the largest variance across ensemble members. The second (`E&E-TS`) and the third (`BAI-TS`) strategies follow the principles of Algorithm 1, and their differences are between Line 5 and Line 6. The comparison results are shown in Figure 8 (Left and Middle). Focusing on the left plot, we observe that `E&E-TS` strategy achieves the best online performance, which is within our expectation. In contrast, `Uncertainty` shows the worst online performance because it tries to maximize the information gain but does not prioritize reward maximization. On the other hand, conclusions are interestingly different when taking the offline performance as a metric. In this case, `BAI-TS` $\succ$ `Uncertainty` both improve the agent's offline performance more efficiently than `E&E-TS`. This can be attributed to that exploration for uncertainty minimizing helps to identify more informative responses to train the LLM policy. `E&E-TS`, however, always chooses two responses with similarly high quality to exploit, and may be less efficient to explore for the optimal policy.

H.4 Aligning LLMs with a human simulator

Results presented so far are based on experimenting LLM alignment with the preference oracle being a scalar reward model, which is deterministic and does not capture the potential randomness of the choice by real humans. To test different agents in a more realistic setting, we use generative models as human simulator in an LLM-as-a-judge [10, 80] manner. In particular, we directly query the OpenAI API and use the `gpt-4o-mini-2024-07-18` model as the judge to provide preference feedback. We follow the prompt template of [34]. The results are shown in Figure 8 (Right). We can observe the performance curves generally exhibit higher variance, possibly because of the randomness introduced in the feedback process, which puts more stringent requirements for learning algorithms. The two active exploration methods demonstrate opposite results to those in Appendix H.1 – `APL` learns fast initially but is eventually outperformed by `Online`, and `XPO` improves over `Online` after stabilizing its training and delivers a better final performance. Our agent, **SEA**, is shown to offer the best sample efficiency as

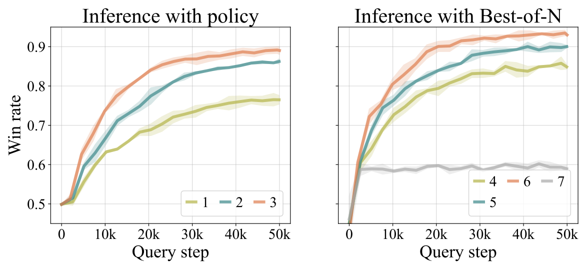
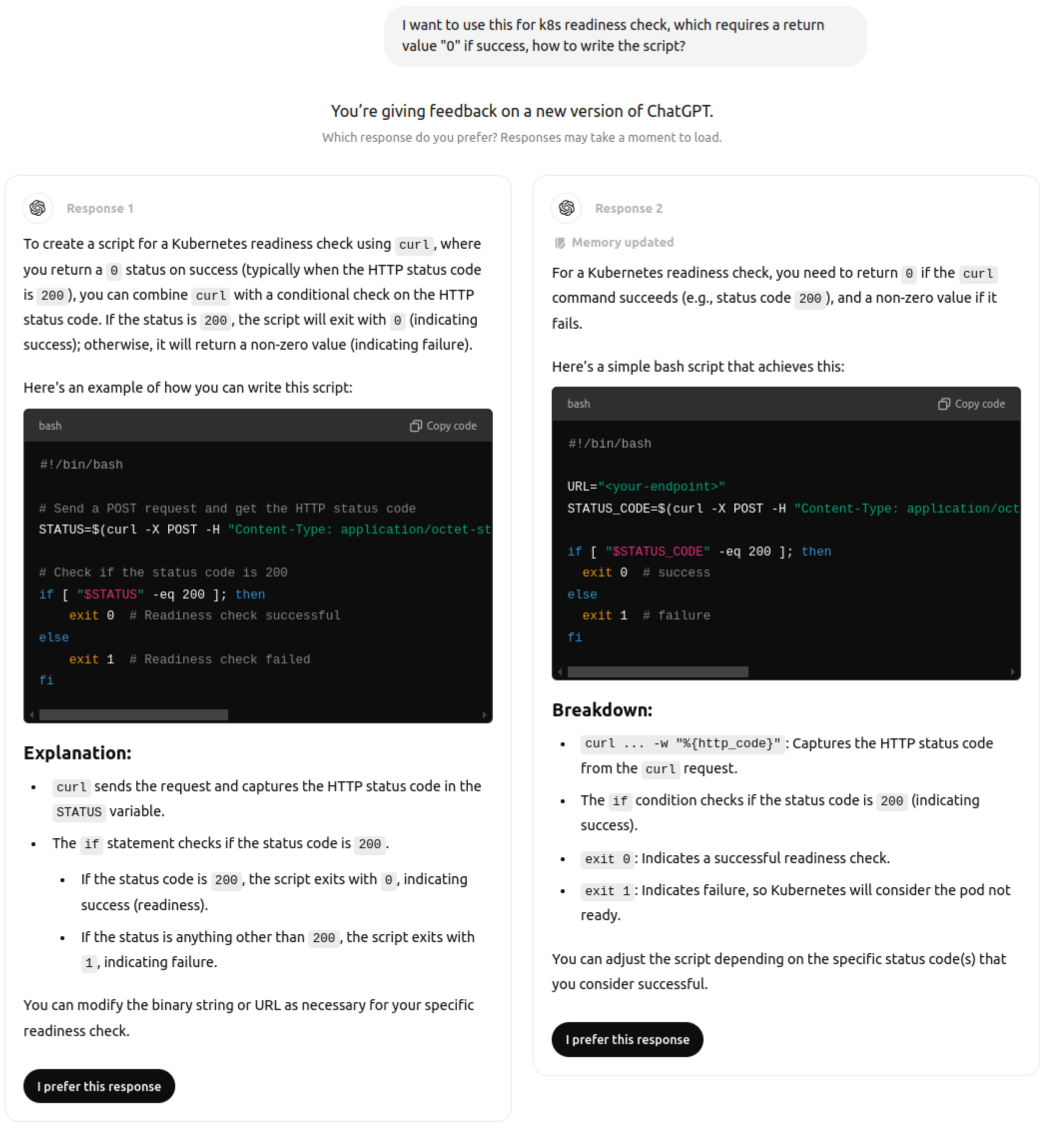


Figure 9: Comparison on the win rates of different agent variants when using **(Left)** policy and **(Right)** Best-of-N sampling for inference.

well as asymptotic performance, further validating the importance of online learning and well-designed active exploration mechanism.



I want to use this for k8s readiness check, which requires a return value "0" if success, how to write the script?

You're giving feedback on a new version of ChatGPT.

Which response do you prefer? Responses may take a moment to load.

Response 1

To create a script for a Kubernetes readiness check using `curl`, where you return a `0` status on success (typically when the HTTP status code is `200`), you can combine `curl` with a conditional check on the HTTP status code. If the status is `200`, the script will exit with `0` (indicating success); otherwise, it will return a non-zero value (indicating failure).

Here's an example of how you can write this script:

```bash
#!/bin/bash

# Send a POST request and get the HTTP status code
STATUS=$(curl -X POST -H "Content-Type: application/octet-st

# Check if the status code is 200
if [ "$STATUS" -eq 200 ]; then
    exit 0  # Readiness check successful
else
    exit 1  # Readiness check failed
fi
```

Explanation:

- `curl` sends the request and captures the HTTP status code in the `STATUS` variable.
- The `if` statement checks if the status code is `200`.
  - If the status code is `200`, the script exits with `0`, indicating success (readiness).
  - If the status is anything other than `200`, the script exits with `1`, indicating failure.

You can modify the binary string or URL as necessary for your specific readiness check.

I prefer this response

Response 2

Memory updated

For a Kubernetes readiness check, you need to return `0` if the `curl` command succeeds (e.g., status code `200`), and a non-zero value if it fails.

Here's a simple bash script that achieves this:

```bash
#!/bin/bash

URL="<your-endpoint>"
STATUS_CODE=$(curl -X POST -H "Content-Type: application/oct

if [ "$STATUS_CODE" -eq 200 ]; then
  exit 0  # success
else
  exit 1  # failure
fi
```

Breakdown:

- `curl ... -w "%{http_code}"`: Captures the HTTP status code from the `curl` request.
- The `if` condition checks if the status code is `200` (indicating success).
- `exit 0`: Indicates a successful readiness check.
- `exit 1`: Indicates failure, so Kubernetes will consider the pod not ready.

You can adjust the script depending on the specific status code(s) that you consider successful.

I prefer this response

Figure 10: ChatGPT system asks for users' preference feedback to strategically explore better answers. In this case, algorithms should be designed around the objective of *minimizing cumulative regret* (i.e., the E&E setting), because the quality of both responses generated by the system affects user experience.