

Sequence-level Large Language Model Training with Contrastive Preference Optimization

Anonymous ACL submission

Abstract

The next token prediction loss is the dominant self-supervised training objective for large language models and has achieved promising results in a variety of downstream tasks. However, upon closer investigation of this objective, we find that it lacks an understanding of sequence-level signals, leading to a mismatch between the training and inference processes. To bridge this gap, we introduce a contrastive preference optimization procedure that can inject sequence-level signals into the language model at any training stage without expensive human labeling. Notably, our experiments revealed that the proposed objective surpasses the next token prediction in terms of the win rate in the instruction-following and text generation tasks. Specifically, using OpenLlama-3B, our method achieves a 13.8% improvement in an instruction-following task and a 3% increase in a text generation task.

1 Introduction

Next token prediction¹ is now the predominant way for pre-training and supervised fine-tuning (SFT) of large language models (LLM). This loss function can be easily scaled up to train models with trillions of parameters and tokens, and it has also demonstrated the ability to generate coherent and contextually relevant text. Let P be the unknown target language distribution, and let Q be the distribution of our model at hand. The goal of next token prediction is to minimize the *forward-KL* divergence between P and Q ; during test time, we usually first generate a set of samples using the trained model and evaluate the quality of these generations using a certain metric², for example, the *reverse-KL*. This training process only supervises the prediction of one token at a time given

¹This term is subsequently used interchangeably with maximum likelihood estimation (MLE).

²Metric in the sense of a quality measurement, rather than the mathematical concept.

the full context from the ground truth. On the other hand, during inference, the model needs to generate a whole sequence (for a given prompt) relying on its own prior predictions. This mismatch between the training stage and the inference stage is known as *exposure-bias* in the literature of RNN and sequence-to-sequence model (Bengio et al., 2015; Ranzato et al., 2015).

In other words, the next token prediction-based training injects only *token-level* information into the model, but missing *sequence-level* signal. Of course, such discrepancies can be mitigated by the subsequent reinforcement learning with human feedback (RLHF) step (Ouyang et al., 2022) in LLM training. In RLHF, a reward signal is enforced on the generated sequence of the language model and guides the model generation to align with human preference. RLHF is computationally intensive and often faces instability issues. Therefore, many open-source LLMs do not incorporate this discipline. Direct preference optimization (DPO) (Rafailov et al., 2023) is a recently proposed alternative to RLHF that enables sequence-level LLM training without the need for costly model generations. One drawback of both DPO and RLHF methods is that they require expensive human labeling to score the LLM training samples. RLHF requires human preference data to train the reward model, and DPO training requires a supervised pair of positive and negative completions for each given prompt. However, most of the existing LLM training data do not consist of such human preference information. Therefore, in this work, we ask the following question:

Can we introduce sequence-level information in LLM training even in the absence of human-preference data?

We answer the question affirmatively with our proposed **CONTRASTIVE PREFERENCE OPTIMIZATION** (CPO) method. CPO shares a sim-

ilar principle to RLHF/DPO in the sense that they all parameterize (perhaps implicitly) the optimal model (with respect to a certain sequence-level signal) by an energy-based model (EBM). However, the goal of CPO is **not alignment, but generation quality**. Therefore, unlike RLHF and DPO, the proposed CPO method does not require human preference information as the training signal. Another related method that optimizes the quality of generated text is BRIO (Liu et al., 2022). Although unlike BRIO, the proposed CPO method does not necessarily rely on autoregressively sampled negative sequences from the model and therefore is much more computational efficient and easier to scale up. In addition, CPO is also derived from a more principled statistical perspective. Our experiments demonstrate that CPO is able to improve the quality of text generation in terms of reward model scores and reverse-KL divergence.

2 Related work

LLMs trained with next token prediction loss (Radford et al., 2019; Chung et al., 2022; Sanh et al., 2021; Zhou et al., 2023) have demonstrated many fascinating capabilities, including the ability to perform zero-shot or few-shot tasks (Radford et al., 2019; Brown et al., 2020) and the ability to reason (Wei et al., 2022).

Several works have investigated the shortcomings of MLE and exposure bias. Arora et al. (2022) measured the accumulation of errors in language generation due to exposure bias. Schmidt (2019) connected exposure bias to generalization. Wang and Sennrich (2020) studied how exposure bias leads to hallucination in neural machine translation. To mitigate exposure bias, there exists a long line of work that has explored sequence-level training methods. Bengio et al. (2015); Ranzato et al. (2015) proposed to train RNN with RL or RL-related algorithms rather than teacher-forcing. BRIO Liu et al. (2022) targeted the summarization task with the ROUGE signal. Pang and He (2020) trained the language models with an offline RL learning algorithm.

Recently, RLHF (Stiennon et al., 2020; Ouyang et al., 2022) was developed. Although the primary goal of RLHF is model alignment, it is effectively a sequence-level training technique. For the RLHF training, we usually need to gather a pair of continuations for each prefix, where one continuation aligns with human preference and the other does

not. This pair of sequences is used to train a reward model, which is later used to supervise the samples generated by the RL-trained model. The model is typically optimized by REINFORCE (Williams, 1992) or PPO (Schulman et al., 2017).

Sequence-level training is also closely related to energy-based models (EBM) (Korbak et al., 2022), for example, the RLHF objective can be reframed as a supervised learning algorithm coined as direct preference optimization (DPO) (Rafailov et al., 2023) under the assumption of the Bradley-Terry model (Bradley and Terry, 1952) or the Plackett-Luce model (Plackett, 1975; Luce, 2012). The particular formulation of the EBM in DPO mimics the formulation in Deng et al. (2020), with the reward function being the energy function. However, Deng et al. (2020) directly treat the EBM as a language model, which is computationally heavy for sampling and inference (due to the estimation of the partition function). This EBM form has also been studied in controlled text generation. Kumar et al. (2022) adopted the Langevin dynamics technique to directly sample from the EBM, with different energy functions that characterize toxicity, fluency, and diversity. All of these methods can be viewed as sequence-level algorithms for different purposes.

3 Preliminary

Notation Consider a sentence of T tokens $\mathbf{x} = \{x_1, \dots, x_T\} \in \mathcal{X}$, and let P be the unknown target language distribution, $\tilde{P}(\mathbf{x})$ be the empirical distribution of the training data (which is an approximation of P), and Q be the distribution of our model at hand. Since our paper is also closely related to RLHF, we will also use π to represent the distributions. In particular, we sometimes write π_θ for a distribution that is parameterized by θ , where θ is usually the set of trainable parameters of the LLM; we write π_{ref} for a reference distribution that should be clear given the context. The next token prediction loss is minimizing the forward-KL between P and Q .

Forward-KL vs. reverse-KL The forward-KL is formally defined as the following:

$$\begin{aligned}
& \arg \min_Q D_{\text{KL}}(P||Q) \\
& \approx \arg \min_Q D_{\text{KL}}(\tilde{P}||Q) \\
& = \arg \min_Q -\frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \log Q(\mathbf{x}).
\end{aligned}$$

Since we are only optimizing Q , minimizing the forward-KL is equivalent to the maximum likelihood estimation (MLE) $\max \log Q$. Further decomposing $Q(\mathbf{x}) = \prod_i Q(\mathbf{x}_i|\mathbf{x}_1^{i-1})$, we get the next token prediction loss function

$$\arg \max_Q \sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{x}_t \in \mathbf{x}} \log Q(\mathbf{x}_t|\mathbf{x}_1^{t-1}). \quad (1)$$

To actually measure the quality of the generated text, typically, we will first generate several sequences and then evaluate the quality of these generated sequences. Here we look closely at the reverse-KL:

$$D_{\text{KL}}(Q||P) = \sum_{\mathbf{x} \in \mathcal{X}} Q(\mathbf{x}) \log \left(\frac{Q(\mathbf{x})}{P(\mathbf{x})} \right), \quad (2)$$

however, since $\mathbf{x} \sim Q$, and we do not have access to P , the reverse-KL cannot be computed exactly.

The equivalence of RLHF and EBM For the completeness of this paper, we include the result of the equivalence between RLHF and EBM. For the full proofs, we refer the reader to (Rafailov et al., 2023; Korbak et al., 2022).

The RLHF objective is the following:

$$\begin{aligned}
& \max_{\pi_\theta} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}, \mathbf{y} \sim \pi_\theta(\mathbf{y}|\mathbf{x})} [r(\mathbf{x}, \mathbf{y})] \\
& - \beta D_{\text{KL}}(\pi_\theta(\mathbf{y}|\mathbf{x}) || \pi_{\text{ref}}(\mathbf{y}|\mathbf{x})),
\end{aligned} \quad (3)$$

where $\mathbf{x} \sim \mathcal{D}$ is a given prefix, $\mathbf{y} \sim \pi_\theta(\mathbf{y}|\mathbf{x})$ is a sampled continuation from the trainable model π_θ , and $r(\mathbf{x}, \mathbf{y}) \in \mathbb{R}$ is the reward. Meanwhile, we want to control the divergence between π_θ and π_{ref} , where π_{ref} is usually an already pretrained or finetuned LLM. The RLHF optimum is achieved at the following EBM:

$$\pi^*(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \pi_{\text{ref}}(\mathbf{y}|\mathbf{x}) \exp \left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{y}) \right), \quad (4)$$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}} \pi_{\text{ref}}(\mathbf{y}|\mathbf{x}) \exp \left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{y}) \right)$ is the partition function.

4 Our approach

While the RL penalty with KL control eq. (3) is widely adopted in RLHF, it can also be used directly to train LLMs: instead of a preference reward, we can use any metric that measures text qualities as the reward r , including ROUGE, BLEU, MAUVE, etc. The benefit of eq. (3) over eq. (1) is that r guides the model over a whole sequence \mathbf{y} , rather than just a single token. This motivates our work to investigate the possibility of using a sequence-level objective in the pretraining stage and the SFT stage of LLM.

Following Rafailov et al. (2023), we assume that the preference over two sequences \mathbf{y}_w and \mathbf{y}_l given \mathbf{x} is parameterized by the Bradley-Terry model:

$$P(\mathbf{y}_w \succ \mathbf{y}_l | \mathbf{x}) = \frac{e^{r(\mathbf{x}, \mathbf{y}_w)}}{e^{r(\mathbf{x}, \mathbf{y}_l)} + e^{r(\mathbf{x}, \mathbf{y}_w)}}.$$

Under the Bradley-Terry model, DPO establishes the equivalence between the original RLHF objective eq. (3) and the following supervised objective:

$$\begin{aligned}
& \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = \\
& \mathbb{E}_{(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(\mathbf{y}_w | \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_w | \mathbf{x})} \right. \right. \\
& \left. \left. - \beta \log \frac{\pi_\theta(\mathbf{y}_l | \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l | \mathbf{x})} \right) \right],
\end{aligned} \quad (5)$$

where $\sigma(\cdot)$ is the Sigmoid function.

We can also generalize the formulation to the Plackett-Luce model, where we have a linear ordering $\tau(\cdot)$ among K sequences:

$$\begin{aligned}
& \mathcal{L}_{\text{DPO}}(\pi_\theta, \pi_{\text{ref}}) = \\
& \mathbb{E}_{\substack{\tau, \mathbf{x} \sim \mathcal{D} \\ \mathbf{y}_1, \dots, \mathbf{y}_K}} \left[\log \prod_{k=1}^K \frac{\exp \left(\beta \log \frac{\pi_\theta(\mathbf{y}_{\tau(k)} | \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_{\tau(k)} | \mathbf{x})} \right)}{\sum_{j=k}^K \exp \left(\beta \log \frac{\pi_\theta(\mathbf{y}_{\tau(j)} | \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_{\tau(j)} | \mathbf{x})} \right)} \right].
\end{aligned} \quad (6)$$

Here, $\tau(1), \dots, \tau(K)$ induce a ranking among K sequences. To simplify the notation, from now on we always assume that $\mathbf{y}_1 \sim \mathcal{D}$ is the most preferred text appearing in the training data.

Investigating the DPO objective, we notice two caveats for its use in the pretraining and SFT stages: 1. We need human labelers to gather $\mathbf{y}_w, \mathbf{y}_l$. 2. There may not be a natural ranking among negative sequences $\mathbf{y}_2, \dots, \mathbf{y}_K$ in terms of text quality. To tackle the first point, we sample $\mathbf{y}_l \sim \mathcal{A}$ where \mathcal{A} is some noise distribution from which it is cheap

to sample; to tackle the second point, we provide a variant objective that models a “best-of- K ” event: \mathbf{y}_1 is the best among K sequences, rather than a linear ordering event $\mathbf{y}_{\tau(1)} \succ \mathbf{y}_{\tau(2)} \succ \dots \succ \mathbf{y}_{\tau(K)}$. These modifications lead to our proposed CPO objective:

$$\mathcal{L}_{\text{CPO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{\substack{(\mathbf{x}, \mathbf{y}_1) \sim \mathcal{D} \\ \mathbf{y}_2, \dots, \mathbf{y}_K \sim \mathcal{A}}} \left[\log \frac{\exp\left(\beta \log \frac{\pi_\theta(\mathbf{y}_1|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_1|\mathbf{x})}\right)}{\sum_{j=1}^K \exp\left(\beta \log \frac{\pi_\theta(\mathbf{y}_j|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_j|\mathbf{x})}\right)} \right]. \quad (7)$$

If ranking information is desired, we have the following CPO objective with ranking:

$$\mathcal{L}_{\text{CPO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{\substack{\tau, (\mathbf{x}, \mathbf{y}_1) \sim \mathcal{D} \\ \mathbf{y}_2, \dots, \mathbf{y}_K \sim \mathcal{A}}} \left[\log \frac{\prod_{k=1}^K \exp\left(\beta \log \frac{\pi_\theta(\mathbf{y}_{\tau(k)}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_{\tau(k)}|\mathbf{x})}\right)}{\sum_{j=k}^K \exp\left(\beta \log \frac{\pi_\theta(\mathbf{y}_{\tau(j)}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_{\tau(j)}|\mathbf{x})}\right)} \right]. \quad (8)$$

We will later discuss some possible choices of ranking signals and show that the ranking can indeed further improve the text generation quality.

The crucial aspect of CPO is how to generate negative sequences $\mathbf{y}_2, \dots, \mathbf{y}_k \sim \mathcal{A}$. For RLHF, negative sequences are simply the ones that humans dislike. For the qualities of text generation, we implicitly model the sequence-level signal $r(\mathbf{x}, \mathbf{y})$ such that $r(\mathbf{x}, \mathbf{y}_k) < r(\mathbf{x}, \mathbf{y}_1), \forall k \in \{2, \dots, K\}$. In other words, the reward $r(\cdot)$ prefers the ground truth to any other sequence. Importantly, the actual signal r is not parameterized explicitly, instead it is represented by the log density ratio $\log \frac{\pi_\theta}{\pi_{\text{ref}}}$.

4.1 Connection to noise contrastive estimation

Noise contrastive estimation (NCE) (Gutmann and Hyvärinen, 2010) is a novel estimation technique introduced to tackle the computational infeasibility of traditional likelihood-based methods in large-scale machine learning models, particularly those involving high-dimensional data. NCE diverges from typical maximum likelihood estimation by transforming the problem into a classification task, which is deeply connected to both DPO and CPO. In NCE, the model is trained to distinguish between real data and noise/synthetic data. Beyond binary

classification, RankingNCE³ also trains the model to rank the real data higher than all noise samples (Ma and Collins, 2018).

There are two important distinctions between CPO and NCE. First, instead of training the model to distinguish between real data and noise (at which any reasonable language model should already be good), we train the model to distinguish *better than a reference model does*, hence making the model better at recognizing natural text. Second, we also introduce a denser ranking signal by incorporating the similarity among embeddings of different samples. The experiments in this paper demonstrate that such a dense training signal consistently improves text generation quality.

4.2 Synthetic negative samples

In this work, we propose four ways to generate synthetic negative samples. The first is to autoregressively generate continuations from a reference model (trained with the next token prediction loss). We fixed the synthetic data generation strategy to be top- k sampling with $k = 50$. The advantage of this strategy over the forthcoming strategies is that the generated continuations are of higher quality and lead to better downstream performance, while the disadvantage is that sampling is slow. We denote these negative samples as **autoregressive negatives (AN)**. One can speed up the sampling process via speculative sampling (Chen et al., 2023) or using a smaller or distilled model, this direction is orthogonal to our approach and can be directly incorporated into our framework.

The second way is to directly use the continuations to other (possibly unrelated) prefixes within the same mini-batch as the negative samples. More specifically, given a batch of prefixes and continuations $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^b$, the negative samples to the prefix \mathbf{x}_i are composed of $\{\mathbf{y}_j\}_{j \neq i}$. Although these negative samples are not difficult to distinguish, they are very simple to create and can be easily scaled up. We denote these as **batch negatives (BN)**.

The third way is to perform a token-level perturbation. Given a sequence $\mathbf{y} = \{\mathbf{y}_1, \dots, \mathbf{y}_T\}$, we randomly select c percent of the positions $\{t_1, \dots, t_j\} \subseteq [T]$, and substitute each \mathbf{y}_{t_i} independently based on $\pi_\theta(\mathbf{y}_{t_i}|\mathbf{y}_1, \dots, \mathbf{y}_{t_i})$. We call these **meanfield negatives (MN)**. The name is

³Despite the name, it means the model is ranking the real data highest among all data, rather than learning a total ordering.

based on the fact that we use a fully separable distribution to approximate the autoregressive distribution, but note that this is not the optimal meanfield approximation. Nevertheless, computing this particular meanfield approximation does not take an additional cost, compared to estimating the best meanfield approximation. This method does not generate semantically meaningful sentences, but it does generate hard negative samples, as the model tends to give them high probabilities.

Lastly, for each ground truth continuation, we can truncate the continuation at a random position and append an extra EOS token to the end. We denote this by **truncation negatives (TN)**.

4.3 Possible ranking signals

As mentioned above, our reward implicitly prefers the ground truth over other sequences, and we do not explicitly model the reward parametrically. The upside of implicit representation of the reward is that it bypasses the shortcuts (e.g. Krishna et al., 2021) that are known to other explicit metrics⁴. However, since we do not have access to a concrete score for the text quality, when presented with more than one negative sample, we do not have a direct ranking among them.

Previous work on sequence-level training (Liu et al., 2022; Bengio et al., 2015) has suggested a variety of signals, including BLEU, ROGUE, and BertScore. These signals are usually specific to certain downstream tasks such as translation or summarization. In the modern era of LLMs, they have been shown to no longer align with human evaluations (Goyal et al., 2022). Since our goal is to improve text generation or instruction-following, the cosine similarity between embeddings is a more intuitive signal to measure the distance between sequences. The use of embedding for text generation quality measurement is also suggested in the MAUVE metric (Pillutla et al., 2021).

When presented K sequences and a ranking is desired, the sequences are ranked based on their cosine similarity to the ground truth. Let e_1, \dots, e_K be the embeddings of given sequences y_1, \dots, y_K and without loss of generality assume that e_1 is the ground truth, we define $\tau(y_i) < \tau(y_j)$ if $\frac{\langle e_i, e_1 \rangle}{\|e_i\| \|e_1\|} > \frac{\langle e_j, e_1 \rangle}{\|e_j\| \|e_1\|}$, with the lower ranking index indicating the better sample. Using the objective

⁴Note that here we are not claiming CPO does not exist any shortcuts. The statement here simply means that other existing metrics have known shortcuts.

eq. (8), this process gives us denser signals during training and can lead to better downstream performance.

Another good candidate for the ranking signal is the reward model score. In fact, since the downstream performance is judged by a reward model, this will probably yield the best test performance as well. However, one has to train and host an extra reward model, creating extra memory and computation overhead. Therefore, we did not include such a signal during training in this work.

4.4 Approximate reverse-KL

In the experiment, we also show how CPO improves reverse-KL. As we discuss previously, an unavoidable issue of calculating the reverse-KL is that we do not have access to the probability of the generated sequences under the true language distribution. However, if we agree that the ability to model natural language scales with the model size, then we can approximate the true language distribution P with a more capable model \hat{P} , hence approximating the reverse-KL divergence. Since many of our tasks are conditional by nature, for example, the instruction-following task is to generate a response conditioned on the input instruction, we further consider the expected reverse-KL divergence:

$$\begin{aligned} & \mathbb{E}_x \left[D_{\text{KL}} \left(Q(\cdot | x) \| \hat{P}(\cdot | x) \right) \right] \\ & \approx \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} Q(y | x) \log \left(\frac{Q(y | x)}{\hat{P}(y | x)} \right), \end{aligned} \quad (9)$$

where \mathcal{X} is the set of inputs (e.g. instructions) in the test set, and \mathcal{Y} is the set of generated continuations (e.g. responses). During our evaluation, we also notice that a more capable Q tends to generate sequences y with a lower probability $Q(y | x)$, compared to a less capable Q . This phenomenon is indeed expected, since a more capable model should be able to generate more diverse continuations. To overcome the numerical instability with a vanishing Q , we also use the following surrogate:

$$\frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \frac{-\log \hat{P}(y | x)}{|y|}, \quad (10)$$

where $|y|$ is the length of y .

This is the log conditional probability normalized by length, and its usage has been justified in Cho et al. (2014); Liu et al. (2022); Fu et al. (2023). In

particular, Fu et al. (2023) has discussed the use of normalized conditional probability with the GPT evaluator.

5 Experiment

Throughout this section, we use **BN** for models trained with batch negatives, **MN** for models trained with meanfield negatives, **TN** for models trained with truncation negatives, **MixN** for a mixed negative sampling strategies which the details should be found in its context, and **AN** for autoregressive negatives. We use **ANR** for models trained with autoregressive negatives and ranking signals, similarly we can define **MixNR**, etc.

Task and model. We consider two tasks in this paper. The first is an instruction-following task, trained and evaluated on the Dolly dataset (Conover et al., 2023). This dataset is composed of 15011 total instruction and response pairs. We train with 7505 sequences and test with the rest 7506. We use pretrained GPT2-XL (Radford et al., 2019) and OpenLlama-3B (Touvron et al., 2023; Geng and Liu, 2023) as the base model. The second task is an open-ended text generation task on Wikidump data (Foundation). We train the OpenLlama-3B model to predict the rest 85% tokens given the leading 15% tokens.

Training details. Throughout the experiment, we fix the learning rate to be $1e-5$, we use the AdamW optimizer with weight decay of 0.05. We keep the batch size to be 64. Unless otherwise specified, for the baseline model, we train GPT2-XL and OpenLlama-3B with the next token prediction loss for 2000 steps, and denote them as MLE models. Using these models as the reference model π_{ref} , we continue to train with the CPO objective either with or without ranking signals, with $\beta = 5$, for 1000 steps. For both models, each batch during training contains 11 negative samples in total. For MixN and MixNR, we also use a negative sample size of 11, consisting of 3 BN, 5 MN, and 3 TN. All experiments are conducted with 16 A100 GPUs.

Evaluation. As discussed in Goyal et al. (2022), almost all automated evaluation metrics have been shown to not align with human evaluations in the modern era of LLM, so we decide to use GPT-3.5 (Brown et al., 2020) as the evaluator. For each test instruction, we ask the trained models to generate continuations with various generation configurations, and query the reward model whether it

prefers the generated continuations or the ground truth. The win rate is then calculated on all test instructions. As pointed out in Wang et al. (2023), GPT models are prone to position bias. When evaluating by asking GPT which of the two inputs it prefers, one can easily manipulate the result by exchanging the input positions. To counteract this bias, for each test instruction, we ask both the CPO model and the baseline model to generate continuations and compare each of them to the ground truth to calculate the win rate. Since both models' generations suffer from the same position bias, we can meaningfully compare the difference between their win rates against the ground truth.

The query template for the Dolly instruction-following is the following: "For the following query to a chatbot, which response is more helpful?\n Query: {}\n Response A: {}\n Response B: {}\n State only "A" or "B" to indicate which response is more helpful.\n More helpful:" For efficiency, we generate and evaluate 1000 samples in the test set. A similar template is used for Wiki text generation, see the detail in the appendix.

In addition to the win rate, we also evaluate the model performance based on reverse-KL and normalized log conditional probability, as described in section 4.4.

Weight-space ensemble. Previous works (Liu et al., 2022) have also suggested to combine the auxiliary loss function with the MLE training objective $\mathcal{L}_{\text{MLE}} + \alpha\mathcal{L}_{\text{CPO}}$, the downside of combining loss functions in this way is that for a different choice of α one will have to retrain the model. To investigate the importance of loss combination, we instead take a similar approach to WISE-FT (Wortsman et al., 2022) and perform a weight-space ensemble. In particular, denote θ_{CPO} and θ_{MLE} the model parameters trained solely with CPO or MLE respectively, we generate with the interpolated weights $\theta = \alpha\theta_{\text{MLE}} + (1 - \alpha)\theta_{\text{CPO}}$.

5.1 Instruction-following task

On the Dolly instruction-following task, our proposed CPO method with various negative sampling strategies consistently outperforms the MLE baseline models. Using greedy sampling with GPT2-XL, the CPO model has a clear margin over the MLE model, and CPO+ANR has a 3.5% higher win rate, see table 2. Keep in mind that the CPO process incurs only very little computation over-

Table 1: Study of the effect of different negative samples. Win rate evaluated by GPT-3.5 against the ground truth continuations.

MLE	BNR	MNR	TNR
0.505	0.599	0.567	0.601

Table 2: The win rate of GPT2-XL against the ground truth, samples generated by greedy decoding, evaluated by GPT-3.5.

	MLE	ANR	MixNR			
α	-	-	0	0.5	0.7	0.9
WinRate	0.471	0.506	0.476	0.479	0.487	0.485

Table 3: The win rate of OpenLlama-3B trained with either MLE or CPO+MixNR against the ground truth, evaluated by GPT-3.5. The samples are generated by various strategies, we only present MLE and MixNR models here.

Model \ Config	$k = 50, p = 1$	$k = 50, p = 0.7$	beam= 2	beam= 4
	MixNR	0.591	0.611	0.607
MLE	0.497	0.517	0.532	0.514

head during the actual training. Even if we generate the negative samples autoregressively, this cost is only offline and is one-time.

The improvement in OpenLlama-3B is more significant: CPO+ANR has a 13.8% higher win rate than the MLE baseline, and CPO+MixNR has a 9.8% higher win rate in table 4. We also observe that WISE-FT has a positive impact on the model. Heuristically, for OpenLlama-3B, a smaller α is preferred (more emphasis on the CPO weights) (table 4), but the reverse holds for GPT2-XL (table 2). We hypothesize that the choice of α should depend on the model parameters: If the model is more capable, then it can benefit more from CPO. Here, we show the existence of a good α , and we leave further exploration to future research.

Generation configuration. In addition to greedy decoding, we also experiment with different choice of sampling strategies. We test with various settings of top-k top-p sampling, as well as different lengths of beam search. In all settings, CPO has consistently demonstrated superior performance over MLE, see table 3.

Effect of different negative samples. We perform a study on the effects of different negative sampling strategies; the results are presented in table 1. We first train the OpenLlama-3B model with MLE loss for 1000 steps, then continue to train with CPO for 200 steps. For all ground truth sequences, we use 4 negative sequences. In this setting, we always use the ranking information to train CPO. We observe that the effects of BNR and TNR on the reward model preference is similar and

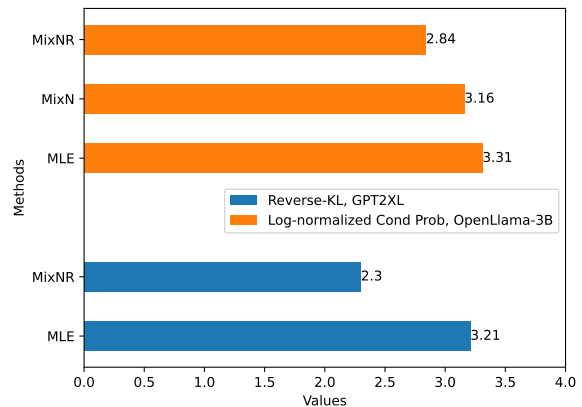


Figure 1: Reverse-KL and negative log-normalized conditional probability of several models and negative sampling strategies. Smaller numbers indicate better quality. CPO consistently outperforms MLE. CPO+ranking information also outperforms CPO without ranking information.

that they perform slightly better than MNR.

Reverse-KL. The reverse-KL (eq. (9)) and negative log-normalized conditional probability (eq. (10)) metrics are reported in fig. 1. Smaller numbers indicate higher quality for both metrics. MixNR and MixN both demonstrate improvements over the MLE model. Since these metrics measure how likely the generated texts are under the (approximated) language distribution, these results serve as complementary explanation to CPO’s improvement in reward model scores.

5.2 Open-ended text generation task

We further test OpenLlama-3B’s ability on an open-ended text generation task with CPO. Using the

Table 4: The win rate of OpenLlama-3B trained with CPO and MLE against the ground truth data in Dolly, sampled by greedy decoding, evaluated by GPT-3.5. MLE₁, ANR and AN are trained for 200 steps, the rest models are trained for 1000 steps. The best CPO model outperforms the MLE baseline by 13.8% win rate.

	MLE ₁	ANR	AN	MLE ₂		MIXNR					MIXN
α	-	-	-	-	0	0.1	0.3	0.5	0.7	0.9	-
WinRate	0.505	0.643	0.56	0.522	0.608	0.620	0.614	0.610	0.601	0.550	0.576

Instruction: Do you prefer a Tesla model S or a Porsche Taycan

MLE response: S

CPO response: ? While both are excellent cars, if I had to choose between the two, I would go with the Tesla model S. The Tesla model S has a larger battery size (92 kWh vs 79 kWh), faster charging capabilities (230 kW vs 180 kW), and has a better range (418 miles vs 318 miles). The Tesla model S also feels more powerful and is easier to drive.

Figure 2: The responses of MLE and CPO model

Table 5: OpenLlama-3B’s win rate against the ground truth continuation on Wikidump. The model is trained with either MLE or CPO+BNR. Weight ensemble is adopted. The best CPO model outperforms the MLE baseline by 3% win rate.

	MLE	BNR			
α	-	0	0.5	0.7	0.9
WinRate	0.508	0.455	0.505	0.5	0.538

Wikidump data (Foundation), for each test sample, we take its first 15% tokens as the prefix and train the model with CPO on the rest 85%. For negative sampling, we use four BNR examples. The results table 5 indicates that with a correctly picked weight interpolation coefficient α , CPO can greatly improve the model’s win rate against the MLE baseline by 3%. The results also have a different pattern compared to the instruction-following task: the optimal choice of α shows a reverse trend. With the Dolly dataset we observe a small optimal α , but on the Wiki dataset we see a large optimal α .

5.3 What type of generations does CPO tend to create?

Investigating the generations of CPO vs those of MLE, we notice that CPO model tends to create more detailed continuations/responses to given prefixes/instructions, partly explaining why these generations are preferred by GPT reward. As the sample demonstrates, the CPO response appears to be more helpful with more details, see fig. 2.

6 Limitation and conclusion

In this paper, we propose an auxiliary CPO loss function for LLM training, which can be used with or without ranking signals depending on the quality of the negative samples. We investigated several ways to generate negative samples, each with its own pros and cons. Experimentally, we show that both the GPT2-XL and the OpenLlama-3B models benefit from training with our proposed CPO objectives. In the Dolly instruction-following task, OpenLlama-3B + CPO has a win rate 13.8% higher than MLE; GPT2-XL has a win rate 3.5% higher. In the Wikipedia text generation task, OpenLlama-3B+CPO has a win rate 3% higher than the MLE baseline model. One limitation of this work is that the synthetic data are very noisy unless generated autoregressively; it is interesting to explore other ways to efficiently generate high-quality negative data beyond the autoregressive fashion. One possible direction is to consider Langevin dynamic sampling, which samples all tokens in parallel.

References

- Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. 2022. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. *arXiv preprint arXiv:2204.01171*.
- Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

716 Florian Schmidt. 2019. Generalization in generation:
717 A closer look at exposure bias. *arXiv preprint*
718 *arXiv:1910.00292*.

719 John Schulman, Filip Wolski, Prafulla Dhariwal,
720 Alec Radford, and Oleg Klimov. 2017. Proxi-
721 mal policy optimization algorithms. *arXiv preprint*
722 *arXiv:1707.06347*.

723 Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel
724 Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford,
725 Dario Amodei, and Paul F Christiano. 2020. Learn-
726 ing to summarize with human feedback. *Advances*
727 *in Neural Information Processing Systems*, 33:3008–
728 3021.

729 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier
730 Martinet, Marie-Anne Lachaux, Timothée Lacroix,
731 Baptiste Rozière, Naman Goyal, Eric Hambro,
732 Faisal Azhar, et al. 2023. Llama: Open and effi-
733 cient foundation language models. *arXiv preprint*
734 *arXiv:2302.13971*.

735 Chaojun Wang and Rico Sennrich. 2020. On exposure
736 bias, hallucination and domain shift in neural ma-
737 chine translation. *arXiv preprint arXiv:2005.03642*.

738 Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai
739 Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui.
740 2023. Large language models are not fair evaluators.
741 *arXiv preprint arXiv:2305.17926*.

742 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten
743 Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou,
744 et al. 2022. Chain-of-thought prompting elicits rea-
745 soning in large language models. *Advances in Neural*
746 *Information Processing Systems*, 35:24824–24837.

747 Ronald J Williams. 1992. Simple statistical gradient-
748 following algorithms for connectionist reinforcement
749 learning. *Machine learning*, 8:229–256.

750 Mitchell Wortsman, Gabriel Ilharco, Jong Wook
751 Kim, Mike Li, Simon Kornblith, Rebecca Roelofs,
752 Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali
753 Farhadi, Hongseok Namkoong, et al. 2022. Robust
754 fine-tuning of zero-shot models. In *Proceedings of*
755 *the IEEE/CVF Conference on Computer Vision and*
756 *Pattern Recognition*, pages 7959–7971.

757 Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao
758 Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu,
759 Lili Yu, et al. 2023. Lima: Less is more for alignment.
760 *arXiv preprint arXiv:2305.11206*.

761 A Appendix

762 A.1 Derivation of the CPO objective function

763 Here we give a full derivation of the CPO objective
764 function in eq. (7).

765 Let $\mathbf{y}_1, \dots, \mathbf{y}_K$ be K continuations of a given
766 prefix \mathbf{x} . Without loss of generality, let \mathbf{y}_1 be the
767 best candidate. We are interested in the MLE of the
768 event $P(\mathbf{y}_1 \text{ is the best among } K \text{ candidates} | \mathbf{x})$.

769 We start from the sequence-level (RLHF) objec-
770 tive, notice that here $r(\cdot)$ is a reward over language
771 quality, not human preference.

$$\begin{aligned} & \max_{\pi_{\theta}} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}, \mathbf{y} \sim \pi_{\theta}(\mathbf{y} | \mathbf{x})} [r(\mathbf{x}, \mathbf{y})] \\ & - \beta D_{\text{KL}}(\pi_{\theta}(\mathbf{y} | \mathbf{x}) || \pi_{\text{ref}}(\mathbf{y} | \mathbf{x})), \end{aligned} \quad (11)$$

772 Its optimum is achieved at the following EBM: 773

$$\pi^*(\mathbf{y} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \pi_{\text{ref}}(\mathbf{y} | \mathbf{x}) \exp\left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{y})\right), \quad (12)$$

774 where $Z(\mathbf{x}) = \sum_{\mathbf{y}} \pi_{\text{ref}}(\mathbf{y} | \mathbf{x}) \exp\left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{y})\right)$ is
775 the partition function. See the proof in (Rafailov
776 et al., 2023; Korbak et al., 2022).

777 Now we consider the natural extension of the
778 Bradley-Terry model to K candidates: 779

$$\begin{aligned} & P(\mathbf{y}_1 \text{ is the best among } K \text{ candidates} | \mathbf{x}) \\ & = \frac{\exp(r^*(\mathbf{x}, \mathbf{y}_1))}{\sum_{k \in [K]} \exp(r^*(\mathbf{x}, \mathbf{y}_k))}. \end{aligned} \quad (13)$$

780 Now assuming we have the optimal policy π^* ,
781 we can reparameterize r by rearranging eq. (12): 782

$$r^*(\mathbf{x}, \mathbf{y}) = \beta \log \frac{\pi^*(\mathbf{y} | \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y} | \mathbf{x})} + \beta \log Z(\mathbf{x}). \quad (14)$$

783 Plugging eq. (14) into eq. (13), we get eq. (7). 784

785 A.2 Query template of Wiki text generation

786 The template is the following: “For the
787 following prefix, which continuation is
788 better?\n Prefix: {\n Continuation A:
789 {\n Continuation B: {\n State only “A”
790 or “B” to indicate which continuation is
791 more helpful.\n Better:”