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 across 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 GPT winning rate on both instruction-following and text generation. Specifically, using OpenLlama-3B, our method achieves a 13% improvement on an instruction-following task, and a 3% increase on a text generation task.

1 Introduction

Next token prediction [1] is now the prevalent way to the pretraining and supervised finetuning (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 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 metric [2], for example the *reverse-KL*. This training process only supervise on predicting one token at a time given the full context from groundtruth. 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 and inference stage is also 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-sourced 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, the majority of existing LLM training data does 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

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

[2]Metric in the sense of a quality measure, rather than the mathematical concept.

OPTIMIZATION (CPO) method. CPO shares a similar principle to RLHF/DPO in the sense that they all parameterize (perhaps implicitly) the optimal model (with respect to a certain sequence-level signal) with an energy-based model (EBM). However, the goal of CPO is **not for alignment, but for 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 language quality is BRIO (Liu et al., 2022). Although unlike BRIO, the proposed CPO method does not rely on autoregressively sampled negative sequences from the model, and therefore is much more computational efficient and easier to scale up. The experiments in this paper 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), the ability to improve the robustness of visual learning in multimodal models (Menon and Vondrick, 2022; Feng et al., 2023), and the ability to reason (Wei et al., 2022).

Several works have investigated the shortcoming of MLE and exposure bias. Arora et al. (2022) measured the error accumulation of 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 have 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 learning algorithm.

Recently, RLHF (Stiennon et al., 2020; Ouyang et al., 2022) is developed. While the primary goal of RLHF for 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 REINFOCE (Williams, 1992) or PPO (Schulman et al., 2017).

RLHF process is also closed related to energy-based models (EBM) (Korbak et al., 2022), and RLHF training 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 Plackett-Luce model (Plackett, 1975; Luce, 2012). The particular formulation of the EBM that minimizes the RLHF objective exactly mimics the formulation in Deng et al. (2020), with the reward function being the energy. However, Deng et al. (2020) directly formulate 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. These methods can all be viewed as sequence-level algorithms for different purposes.

3 Preliminary

Notation Consider a sentence of T tokens $\boldsymbol{x} = \{x_1, \dots, x_T\} \in \mathcal{X}$, and let P be the unknown target language distribution, $\tilde{P}(\boldsymbol{x})$ 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 a subset 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:

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

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

$$
\arg\max_{Q} \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{x_t\in\boldsymbol{x}} \log Q(x_t|x_1^{t-1}). \tag{1}
$$

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:

$$
D_{\mathrm{KL}}(Q||P) = \sum_{\boldsymbol{x}\in\mathcal{X}} Q(\boldsymbol{x}) \log\left(\frac{Q(\boldsymbol{x})}{P(\boldsymbol{x})}\right), \tag{2}
$$

however, since $\boldsymbol{x} \sim Q$, 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 on 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:

$$
\begin{aligned}
\max_{\pi_\theta} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\boldsymbol{y}\sim\pi_\theta(\boldsymbol{y}|\boldsymbol{x})}[r(\boldsymbol{x},\boldsymbol{y})] \\
- \beta D_{\mathrm{KL}}\big(\pi_\theta(\boldsymbol{y}|\boldsymbol{x})||\pi_{\mathrm{ref}}(\boldsymbol{y}|\boldsymbol{x})\big),
\end{aligned} \tag{3}
$$

where $\boldsymbol{x} \sim \mathcal{D}$ is a given prefix, $\boldsymbol{y} \sim \pi_\theta(\boldsymbol{y}|\boldsymbol{x})$ is a sampled continuation from the model at training π_θ. Meanwhile we want to control the divergence between π_θ and π_{ref}, the latter is usually an already pretrained or finetuned LLM. Its optimum is achieved at the following EBM:

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

where $Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \pi_{\mathrm{ref}}(\boldsymbol{y}|\boldsymbol{x}) \exp\left(\frac{1}{\beta} r(\boldsymbol{x},\boldsymbol{y})\right)$ 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 $r(\cdot)$, we can use any metric that measures general text qualities, including ROUGE, BLEU, MAUVE, etc. The benefit of eq. (3) over eq. (1) is that r guides the model over a whole sequence $\boldsymbol{y}$, rather than just a single token. This motivates our work to investigate the possibility of using such objective in pretraining and SFT stage of LLMs.

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

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

The optimal policy π^* takes the aforementioned EBM form eq. (4). and this EBM repramentrization establish the equivalence between the original RLHF object eq. (3) and the following supervised objective:

$$
\begin{aligned}
&\mathcal{L}_{\mathrm{DPO}}(\pi_\theta;\pi_{\mathrm{ref}}) = \\
&\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}_w,\boldsymbol{y}_l)\sim\mathcal{D}}\Big[\log\sigma\Big(\beta\log\frac{\pi_\theta(\boldsymbol{y}_w \mid \boldsymbol{x})}{\pi_{\mathrm{ref}}(\boldsymbol{y}_w \mid \boldsymbol{x})} \\
&\quad - \beta\log\frac{\pi_\theta(\boldsymbol{y}_l \mid \boldsymbol{x})}{\pi_{\mathrm{ref}}(\boldsymbol{y}_l \mid \boldsymbol{x})}\Big)\Big],
\end{aligned} \tag{5}
$$

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:

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

Here, $\tau(1),\ldots,\tau(K)$ induce a ranking among K sequences. To ease the notation, from now on, we always assume that $\boldsymbol{y}_1 \sim \mathcal{D}$ is the natural text appeared 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 $\boldsymbol{y}_l \sim \mathcal{D}$. 2. There may not be a natural ranking among negative sequences $\boldsymbol{y}_2,\ldots\boldsymbol{y}_K$ in terms of text quality. To tackle the first point, we sample $\boldsymbol{y}_l \sim \mathcal{A}$ where

$\mathcal{A}$ is some noise distribution where it is cheap to sample; to tackle the second point, we provide a variant objective that models a "best-of-K" event: $\boldsymbol{y}_1$ is the best among K sequences, rather than a linear ordering event $\boldsymbol{y}_{\tau(1)} \succ \boldsymbol{y}_{\tau(2)} \succ \ldots \succ \boldsymbol{y}_{\tau(K)}$. These modifications lead to our proposed CPO objective:

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

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

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

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 the negative sequences $\boldsymbol{y}_2, \ldots, \boldsymbol{y}_k \sim \mathcal{A}$. For RLHF, the negative sequences are simply the ones that humans dislike. For the qualities of text generation, we implicitly model the sequence-level signal $r(\boldsymbol{x}, \boldsymbol{y})$ such that $r(\boldsymbol{x}, \boldsymbol{y}_k) < r(\boldsymbol{x}, \boldsymbol{y}_1), \forall k \in \{2, \ldots, K\}$. In other word, we use the reward $r(\cdot)$ that prefers the ground truth to any other sequences. 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 contrastrive 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 classificaiton, RankingNCE [3] also train the model to rank the real data higher than all the noise samples (Ma and Collins, 2018).

There are two important distinctions between CPO and NCE. First, instead of asking the model to distinguish between real data and noise (at which any reasonable language model should already be good), we ask the model to distinguish *better* than a reference model does, hence making the model better at recognizing natural text. Second, we incorporate a denser signal by incorporating the similarity among embeddings of different samples. The experiments in this paper demonstrate that such a dense training signal consistently improves the 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 to the training prefixes from a model trained with the next token prediction loss. We fix the synthetic data generation strategy to be top$-k$ sampling with $k = 50$. The advantage of this strategy to 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 $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{b}$, the negative samples to the prefix $\boldsymbol{x}_i$ are composed of $\{\boldsymbol{y}_j\}_{j \neq i}$. 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 $\boldsymbol{y} = \{y_1, \ldots, y_T\}$, we randomly select c percent of the positions

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

$\{t_1, \ldots, t_j\} \subseteq [T]$, and substitute each y_{t_i} independently based on $\pi_\theta(y_{t_i}|y_1, \ldots, y_{t_i})$. We call these **meanfield negatives (MN)**. The name is based on the fact that we use a fully separable distribution to approximate the AR distribution, but note that this is not the optimal mean-field 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 as **truncation negatives (TN)**.

4.3 Possible ranking signals

As mentioned before, our reward implicitly prefers the ground truth over other sequences, and we do not explicitly model the reward parametrically. The upside of the implicit representation of the reward is that it bypasses the shortcuts (e.g. Krishna et al., 2021) that are known to other explicit metrics [4]. However, since we do not have access to a concrete score for the text quality, when presented more than one negative samples, we do not have a direct ranking among them.

Prior works on sequence-level training (Liu et al., 2022; Bengio et al., 2015) have suggested a variety set of signals, including BLEU, ROGUE, and BertScore. These signals are usually specific for certain downstream tasks like translation or summarization. In modern era of LLMs, they have been shown to not align with human evaluations anymore (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 usage of embedding for generation quality measurement is also suggested in the MAUVE metric (Pillutla et al., 2021).

When presented K sequences and ranking is desired, the sequences are ranked based on its cosine similarity with the ground truth. Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_K$ be the embeddings of given sequences and without loss of generality assume $\boldsymbol{e}_1$ is the ground truth, we define $\tau(\boldsymbol{e}_i) < \tau(\boldsymbol{e}_j)$ if $\frac{\langle \boldsymbol{e}_i, \boldsymbol{e}_1 \rangle}{\|\boldsymbol{e}_i\| \|\boldsymbol{e}_1\|} > \frac{\langle \boldsymbol{e}_j, \boldsymbol{e}_1 \rangle}{\|\boldsymbol{e}_j\| \|\boldsymbol{e}_1\|}$, with the lower ranking index indicating the better sample. Using the objective eq. (8), this process gives us denser signals during training, and can lead to better downstream performance.

Another good candidate for 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 signal during training in this work.

4.4 Approximate reverse-KL

In the subsequence experiment, we show how CPO improves reverse-KL. As we discuss previously, an unavoidable issue of calculating the reverse-KL is we do not have access to the probability of the generated sequences under the true language distribution. However, if we agree that the capability of imitating true language scales with the model size, then we can approximate the true language distribution P with a more capable model $\widehat{P}$, 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, condition on the input instruction, we further consider the expected reverse-KL divergence:

$$\mathbb{E}_{\boldsymbol{x}} \left[D_{\mathrm{KL}} \left(Q(\cdot|\boldsymbol{x}) || \widehat{P}(\cdot|\boldsymbol{x}) \right) \right] \approx \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x} \in \mathcal{X}} \sum_{\boldsymbol{y} \in \mathcal{Y}} Q(\boldsymbol{y}|\boldsymbol{x}) \log \left(\frac{Q(\boldsymbol{y}|\boldsymbol{x})}{\widehat{P}(\boldsymbol{y}|\boldsymbol{x})} \right), \tag{9}$$

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 $\boldsymbol{y}$ with lower probability $Q(\boldsymbol{y}|\boldsymbol{x})$, compare 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:

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

where $|\boldsymbol{y}|$ is the length of $\boldsymbol{y}$.

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

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). Particularly, Fu et al. (2023) has discussed the use of the normalized conditional probability of a capable 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 the 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 the 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 the GPT2-XL and OpenLlama-3B with the next token prediction loss for 2000 steps, and denote them as the 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 3 BN, 5 MN, and 3 TN.

Evaluation. As discussed in Goyal et al. (2022), almost all automated evaluation metrics have been shown to not align with human evaluations in modern era of LLMs, hence we decide to use use GPT-3.5 (Brown et al., 2020) as the evaluation tool. For each test instruction, we ask the models to generate continuations with various generation configurations, and query the reward model whether it prefers the generated continuations or the ground truth. A winning rate is then computed over 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 counter this bias, for each test instruction, we ask both the CPO model and baseline model to generate continuations, and we compare each of them to the ground truth to calculate the winning rate. Now since both models' generations suffer from the same position bias, we can meaningfully compare the difference between their winning rates against the ground truth.

The query template 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.

In addition to winning 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 newly proposed 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

Table 1: Study of the effect of different negative samples.

BNR	MNR	TNR
0.599	0.567	0.601

Table 2: The winning 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 winning 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.

Config / Model	$k = 50, p = 1$	$k = 50, p = 0.7$	beam$= 2$	beam$= 4$
MIXNR	0.591	0.611	0.607	0.568
MLE	0.497	0.517	0.532	0.514

winning rate, see table 2. Keep in mind that the CPO process only incur very little computation overhead during the actual training. Even if we generate the negative samples autoregressively, this cost is only offline and is one-time.

The improvement on OpenLlama-3B is more significant: CPO+ANR has a 13.8% higher winning rate than the MLE baseline, and CPO+MixNR has a 9.8% higher winning 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), and the reverse holds for GPT2-XL (table 2). We hypothesize that the choice of α should depend on 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 length of beam search. In all settings, CPO has consistently demonstrated a superior performance to MLE 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 we continually train with CPO for 200 steps. For every ground truth sequences, we use 4 negative sequences. In this setting, we always use the ranking information to

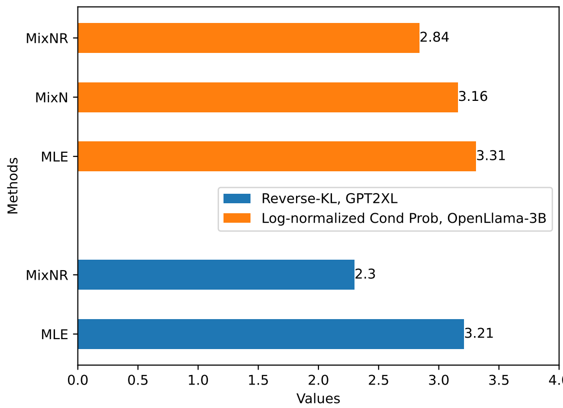

Figure 1: Reverse-KL and negative log-normalized conditional probability of different negative sampling strategies. Smaller numbers indicate better quality. CPO consistently outperforms MLE. CPO+ranking information also outperforms CPO without ranking information.

train CPO. We observe that the effects of BNR and TNR on the reward model preference is similar, and 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.

Table 4: The winning 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. MLE_1, 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% winning 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.

Table 5: OpenLlama-3B's winning 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% winning rate.

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

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 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 winning 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 observes a small optimal α, but on the Wiki dataset we see a large optimal α.

5.3 What type of generations do 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.

6 Conclusion

In this paper, we propose an auxiliary CPO loss function for SFT, it can be used with or without ranking signals depending on the quality of the negative samples. We investigated several ways to generate the negative samples, each with its own pros and cons. Experimentally, we show that both GPT2-XL and OpenLlama-3B models benefit from training with our proposed CPO objectives. On Dolly instruction-following task, OpenLlama-3B+CPO has a winning rate 13.8% higher than MLE; GPT2-XL has a winning rate 3.5% higher. On Wikipedia text generation task, OpenLlama-3B+CPO has a winning rate 3% higher than the MLE baseline model. 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.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. 2020. Residual energy-based models for text generation. *arXiv preprint arXiv:2004.11714*.

Zhili Feng, Anna Bair, and J Zico Kolter. 2023. Leveraging multiple descriptive features for robust few-shot image learning. *arXiv preprint arXiv:2307.04317*.

Wikimedia Foundation. Wikimedia downloads.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

Tomasz Korbak, Ethan Perez, and Christopher L Buckley. 2022. Rl with kl penalties is better viewed as bayesian inference. *arXiv preprint arXiv:2205.11275*.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to progress in long-form question answering. *arXiv preprint arXiv:2103.06332*.

Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. 2022. Gradient-based constrained sampling from language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2251–2277.

Yixin Liu, Pengfei Liu, Dragomir Radev, and Graham Neubig. 2022. Brio: Bringing order to abstractive summarization. *arXiv preprint arXiv:2203.16804*.

R Duncan Luce. 2012. *Individual choice behavior: A theoretical analysis*. Courier Corporation.

Zhuang Ma and Michael Collins. 2018. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*.

Sachit Menon and Carl Vondrick. 2022. Visual classification via description from large language models. *arXiv preprint arXiv:2210.07183*.

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

Richard Yuanzhe Pang and He He. 2020. Text generation by learning from demonstrations. *arXiv preprint arXiv:2009.07839*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.

Robin L Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

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

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

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

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

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. *arXiv preprint arXiv:2005.03642*.

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

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

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

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

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