Language Model Preference Learning as Retriever Optimization

Wei Xiong*
wx13@illinois.edu
University of Illinois Urbana-Champaign
Urbana, Illinois, USA

Abstract

Direct Preference Optimization (DPO) has emerged as a popular alternative approach in the reinforcement learning from human feedback (RLHF) literature to align the foundation generative models with human preference. The derivation of DPO depends on approximating the human preference with a Bradley-Terry pairwise ranking structure, which originates from the ranking and Information Retrieval (IR) literature. In this work, we explore this connection between LLM alignment and IR more deeply. We suggest thinking about LLM generation and reward models similar to how IR uses 'retrievers' and 'rerankers'. Based on this idea, we propose LLM Alignment as Retriever Preference Optimization (LarPO), a new alignment method using IR principles to improve alignment quality. Our experiments show LarPO works well, giving average improvements of 38.9% on AlpacaEval2 and 13.7% on MixEval-Hard compared to baselines.

Keywords

RLHF, Information Retrieval, Preference Learning

ACM Reference Format:
Wei Xiong. 2018. Language Model Preference Learning as Retriever Optimization. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Large Language Models (LLMs) like GPT-4 and Gemini [1, 40] are powerful tools for various tasks, including conversation [4], reasoning [46], and code generation [20]. To make sure these models are helpful, safe, and produce high-quality output, we need to 'align' them with human expectations [45]. The dominant approach in this area is based on reinforcement learning (*e.g.*, PPO [31], GRPO [37]) often involve multiple training stages and can be hard to tune. To simplify this, newer methods like Direct Preference Optimization (DPO) [34] directly optimize the LLM based on preference data.

The derivation of DPO depends on approximating the human preference with a Bradley-Terry pairwise ranking structure, which originates from the ranking and Information Retrieval (IR) literature. In this work, we further enhance direct LLM preference optimization, focusing on bringing IR perspectives [39]. There are interesting similarities between IR methods and LLM alignment techniques [23]. For example, IR's retriever-reranker framework, which uses a retriever for broad semantic matching to generate a candidate set and a reranker for fine-grained refinement, offers a compelling analogy to the Best-of-N approach in LLM alignment [12, 36]. In this analogy, the LLM acts as the retriever, while the reward model serves as the reranker. Also, the types of models used in both areas often overlap. LLMs generating text and IR retrievers sometimes use similar model structures (like dual-encoders). Reward models used in alignment and IR rerankers also share similarities (like cross-encoders). This connection suggests that using established IR techniques could lead to new, effective, and perhaps simpler ways to align LLMs.

Our goal is to establish a clear mapping between LLM alignment mechanisms and core IR principles has not been established and leverage the ideas like retriever optimization, hard negative mining, and candidate list construction for better LLM alignment. We summarize our contributions as follows.

- We show that three key IR ideas – retriever optimization goals, hard negative mining, and candidate list construction – are important for improving LLM alignment.
- Based on these ideas, we propose a new alignment method, **L**LM **A**lignment as **R**etriever **P**reference **O**ptimization (LarPO). It improves alignment quality, showing average gains of 38.9
- We perform further experiments to evaluate LLM performance using IR metrics and study the impact of different training techniques.

The code is available at https://github.com/WeiXiongUST/CS510-SP25-Course-Project.

2 Related works

LLM alignment. Pretrained LLMs can do many things well [6]. Aligning them with human preferences makes them much better for tasks like chatbots [4, 31]. RLHF (Reinforcement Learning from Human Feedback) [9] is a common way to do this. It typically involves training a 'reward model' based on human feedback (often using the Bradley-Terry model [5] to understand preferences) and then using reinforcement learning (RL) to train the LLM to get high rewards from that model. This framework has contributed to the state-of-the-art LLMs post-training including Gemini [40], Chat-GPT [1], and Claude [2]. However, RLHF can be complex to implement, often needing multiple models and careful tuning.

More recently, researchers have developed simpler 'direct' alignment methods [3, 34, 56]. These methods train the LLM directly from preference data, often skipping the separate reward model step. DPO (Direct Preference Optimization) [34] is a very popular example used in academia and industry. Following DPO, SimPO [26]

*Final project report for UIUC CS 510 Spring 2025.

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

simplified the method further. However, these approaches are primarily offline and can suffer from performance degradation when facing distribution shifts during deployment, yielding poorer generalization. Iterative DPO [49] attempts to address this by enabling iterative optimization for improved alignment.

While these direct methods are helpful, we believe ideas from IR can improve them. Although LLMs are sometimes used in IR tasks [39], there hasn't been much work on using IR principles for improving LLM alignment. This paper focuses on making that connection. The most closely related work is LiPO [24], which also applies ranking objectives from IR. However, LiPO mainly works offline and assumes access to specific 'list-wise' preference data, which can be hard to get in practice. Our approach explores a different connection inspired by retriever optimization.

Language models for information retrieval. LLMs are now essential parts of modern information retrieval systems [57], especially after pretrained models like BERT became available [11]. A typical IR system uses 'retrievers' and 'rerankers'. Retrievers find potentially relevant items (like documents), often using dual-encoder models. Rerankers then take the top items from the retriever and order them more accurately, often using cross-encoder models [17]. Dense Passage Retrieval (DPR) [21] was an important early work using dense vectors for retrieval. Later studies showed that training retrievers effectively requires using 'hard negatives' (difficult incorrect examples) [33, 54] and that updating retrievers during use ('online optimization') can be beneficial [48].

In reranking, researchers started using pretrained LMs early on to improve ranking performance [28]. Methods like MonoT5 [29] and RankT5 [58] used large transformer models for better reranking, experimenting with different ways to train them (e.g., pairwise or listwise objectives). Recent work also shows that carefully processing the list of candidates before reranking can improve results [25].

Despite LMs being widely used in IR, the connection between these IR methods and the techniques used for LLM alignment is not well explored. This work aims to bridge this gap, connecting LLM alignment with IR and using ideas from both fields to better understand LLM alignment from an IR perspective.

3 An Information Retrieval Perspective on LLM Alignment

Preliminary of information retrieval. Information Retrieval (IR) systems [23, 55, 57] often employ a two-stage process involving retrievers [55] and rerankers [23]: Retriever: Finds a large set of (K) potentially relevant passages, denoted as $D_{\text{retrieval}}$, from a corpora C given a query q. Retrievers are usually fast (e.g., bi-encoders, see Figure 1) and use simpler similarity scores $p_{\text{retrieval}}(d|q) = \text{Enc}_q^T(q) \cdot \text{Enc}_d(d)$, where Enc_q and Enc_d represent the query and passage encoders respectively. Mathematically, we can write

$$D_{\text{retrieval}}(q) = \{d \in C \mid \max_{\text{top-}K} p_{\text{retrieval}}(\cdot|q)\}. \tag{1}$$

However, due to the scale of the corpus, retrievers might not accurately capture fine-grained query-passage similarity with the simple dot production interaction function. Therefore, rerankers, typically implemented with cross-encoder (Figure 1), are employed to refine the ranking of the retrieved passages $D_{\text{retrieval}}$. The reranker produces a smaller set (k) of top-ranked passages, D_{rank}, using a fine-grained similarity function, $r_{\text{rank}}(q, d) = w \cdot \text{Enc}(q, d)$, where w is a learnable linear layer. Here, reranker adopts cross-encoder with both query/passage as inputs and encoded together while retriever adopts dual encoder for separate query/passage encoding.

$$D_{\text{rank}}(q) = \{d \in D_{\text{retrieval}}(q) \mid \max_{\text{top-}k} r_{\text{rank}}(q, \cdot)\}. \tag{2}$$

The resulting ranked passages are ordered such that $D_{\text{rank}}(q) = \{d_1, d_2, \ldots, d_k\}$ where $r_{\text{rank}}(q, d_1) \geq r_{\text{rank}}(q, d_2) \geq \cdots \geq r_{\text{rank}}(q, d_k)$.

LLMs as retrievers. Reward models as rerankers. The general idea is that we can view LLM alignment through this IR lens [39]. Given a prompt x (like a query), an LLM generates responses y based on its probability $p_{\text{LLM}}(y|x)$. Conceptually, the LLM acts like a retriever searching the vast space of possible responses Y (like a corpus) for the most likely ones. Correspondingly, a reward model $r_{\text{rm}}(x, y)$ [22], which take both the prompt and response as input, function similarly to cross-encoders (*i.e.*, rerankers $r_{\text{rank}}(q, d)$ [58]) in IR.

Inference techniques like Best-of-N sampling [38] also fit this view, where the LLM retrieves N candidates and the reward model reranks them. We can roughly summarize some representative connections between LLM inference and IR pipelines. as in Table **??**.

To enhance LLM performance, various inference-time strategies have been developed, including Best-of-N sampling [38] and majority voting [44]. These can be interpreted as different configurations of retrievers and rerankers, as summarized in Appendix Table **??**.

Preference optimization as reranker-retriever distillation. Training methods like PPO [35] or DPO [16, 34] use human preferences, often captured by a reward model to further enhance the LLM. The LLM generates responses, the reward model scores them, and this information guides the LLM update. This mirrors a technique in IR called reranker-to-retriever distillation [33, 53], where a powerful reranker helps generate better training data for a simpler retriever: $f_{\text{rerank}}(\cdot) \xrightarrow{r} \text{data} \xrightarrow{g(\cdot)} f_{\text{retrieval}}(\cdot)$. Therefore, preference optimization acts like this distillation process.

Evaluating LLMs as retrievers. A common metric for evaluating retrievers is Recall@N, which assesses whether the top-N retrieved passages include any relevant passages for a given query. In the context of LLMs, this translates to evaluating whether the top-N generated responses contain a suitable response to the prompt, analogous to Pass@N [8].

To check if LLMs behave like retrievers empirically, we can use metrics like Recall@N (IR: does one of the top N retrieved documents match?) or its equivalent Pass@N for LLMs (LLM: does one of the top N generated responses pass a correctness check?) [8].

Figure 2 shows results comparing an IR retriever (e5 on NQ dataset) and an LLM (Mathstral-7b-it on GSM8K dataset). Both show a similar trend: performance (Recall@N / Pass@N) increases as N (the number of candidates considered) increases. This supports the view of LLMs as retrievers and highlights why considering multiple candidates (like in Best-of-N) can improve performance, just like using a reranker over multiple retrieved documents improves IR.

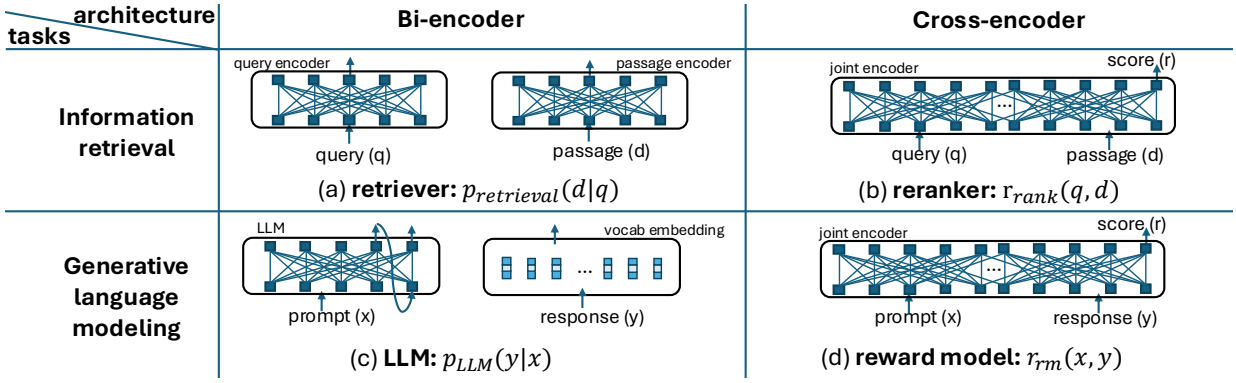
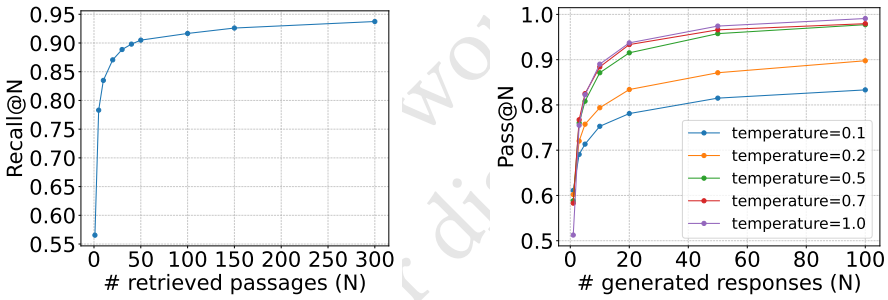


Figure 1: Architecture connection between retriever/LLM (bi-encoder) and reranker/reward model (cross-encoder). Bi-encoder models process each query/prompt and passage/response separately and often calculate their alignment score via a dot product operator, while cross-encoder models take both query/prompt and passage/response as input and score them directly. Bi-encoder models can be more efficient (*i.e.*, large-scale text matching) but the interaction between the two information unit is only captured by a dot production operation where their effectiveness can be constrained. Cross-encoder models can be more effective (*i.e.*, deeper interaction calculation with transformer architecture [43]) but less efficient. Although LLM involves auto-regressive token matching, which is different from retriever, some insights from IR can be borrowed to enhance LLM alignment as shown in the following sections.



(a) Retriever (b) LLM

Figure 2: Analogy between evaluating retriever with Recall@N and LLM with Pass@N. As the number (N) of retrieved passages/generated responses increases, the retriever and LLM have a similar increasing trend. This highlights the importance of inference time scaling (*e.g.*, Best-of-N) for LLM similar to retriever-reranker scaling in IR. Retriever: e5; LLM: Mathstral-7b-it.

4 The Proposed Solution: LarPO

In this section, we develop our main method by treating iterative LLM alignment as retriever optimization problem.

Iterative learning is a common technique in retriever optimization [48], where results from the newly-trained model are used to generate new training data, as illustrated. This is also adopted in the development of LLM alignment algorithms. In particular, the vanilla DPO algorithm is offline and off-policy, which means that the training data may be generated by other models. It was noticed that the online iterative training usually brings better results, and become a standard practice since then [16, 49, 51].

Taking ideas from retriever optimization, we look again at improving LLMs using preferences, focusing on three main things:

- optimization objective;
- the use of hard negatives;
- the candidate list construction.

4.1 Retriever optimization objective

Typical goals for retriever optimization include comparing pairs, comparing one against many (contrastive), or looking at a whole list (listwise) [55]. In this part, we talk about different ways to use preferences for LLMs [45] that match these different goals for improving retrievers.

The standard optimization objective for preference optimization is given as [31]:

$$\max_{\pi_{\text{LLM}}} \mathbb{E}_{x,y\sim\pi_{\text{LLM}}(\cdot|x)}\left[r(x,y)\right] - \beta\text{KL}(\pi_{\text{LLM}}(\cdot|x)||\pi_{\text{ref}}(\cdot|x)).$$

This equation above has the optimal solution as:

$$r(x,y) = \beta\log\frac{\pi_{\text{LLM}}(y|x)}{\pi_{\text{ref}}(y|x)} + \beta\log Z, \tag{3}$$

Table 1: LLM alignment objectives of LarPO. In the table, $\gamma(y \mid x) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$. All the proofs can be found in Appendix A.

Method	Assumption of $r(x, y)$	Objective
DPO	$\Pr(y_w \succeq y_l) = \sigma(r(x, y_w) - r(x, y_l))$	$\mathcal{L}_{\text{pair}} = -\mathbb{E}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w\|x)}{\pi_{\text{ref}}(y_w\|x)} - \beta \log \frac{\pi_\theta(y_l\|x)}{\pi_{\text{ref}}(y_l\|x)}\right)\right]$
LarPO (Contrastive)	$\Pr(y_w \succeq y_l^{(1)}, ..., y_w \succeq y_l^{(m)}) = \text{softmax}(r(x, y_w))$	$\mathcal{L}_{\text{con}} = -\mathbb{E}\left[\log \frac{\exp(\gamma(y_w\|x))}{\exp(\gamma(y_w\|x)) + \sum_{i=1}^m \exp(\gamma(y_l^{(i)}\|x))}\right]$
LarPO (LambdaRank)	$\Pr(y_1 \succeq ... \succeq y_m) = \prod_{1<i<j<m} \sigma(r(x, y_i) - r(x, y_j))$	$\mathcal{L}_{\text{lamb}} = -\mathbb{E}\left[\sum_{1<i<j<m} \log \sigma\left(\gamma(y_i \mid x) - \gamma(y_j \mid x)\right)\right]$
LarPO (ListMLE)	$\Pr(y_1 \succeq ... \succeq y_m) = \prod_{i=1}^m \text{softmax}_i^m(r(x, y_i))$	$\mathcal{L}_{\text{lmle}} = -\mathbb{E}\left[\sum_{i=1}^m \log \frac{\exp(\gamma(y_i\|x))}{\exp(\gamma(y_i\|x)) + \sum_{j=i}^m \exp(\gamma(y_j\|x))}\right]$

where $Z = \sum_{y'} \pi_{\text{ref}}(y'|x)\exp(\frac{1}{\beta} r(x, y'))$ is the normalization constant and $r(\cdot)$ is the reward model which can also be seen as a reranker. However, the reward function is not directly available since it is usually difficult to ask a human annotator to quantify the absolute scores of LLM responses. The idea is to learn from relative feedback, where we ask the human annotators to decide which response is *better*. Then, we can make suitable assumptions on these ranking process with a scalar reward and learn such a reward by learning the model.

According to different assumption for $r(x, y)$ from IR, we can obtain different training objectives as shown in Table 1, with proofs in Appendix A.

Pairwise ranking. If we assume that the chance of a preferred response y_w being better than a less preferred one y_k depends on the reward difference $r(x, y_w) - r(x, y_l)$, one natural assumption is the pairwise (Bradley-Terry) model:

$$\Pr(y_w \succeq y_l) = \sigma(r(x, y_w) - r(x, y_l)), \tag{4}$$

the policy objective becomes DPO [34] $\mathcal{L}_{\text{pair}}$.

$$\mathcal{L}_{\text{pair}} = -\mathbb{E}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]. \tag{5}$$

Contrastive ranking. Another common goal for ranking is called contrastive learning [30]:

$$\Pr(y_w \succeq y_l^{(1)}, ..., y_w \succeq y_l^{(m)}) = \text{softmax}(r(x, y_w)) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \sum_{i=1}^m \exp(r(x, y_l^{(i)}))}. \tag{6}$$

This method can handle several less preferred examples at once. This helps the model learn better ways to represent things for finding and ranking results. It is widely used for dense retriever training [21]. Under this ranking assumption, the policy objective becomes $\mathcal{L}_{\text{con}}$ as shown in Table 1.

$$\mathcal{L}_{\text{con}} = -\mathbb{E}\left[\log \frac{\exp(\gamma(y_w \mid x))}{\exp(\gamma(y_w \mid x)) + \sum_{i=1}^m \exp(\gamma(y_l^{(i)} \mid x))}\right], \quad \text{where} \quad \gamma(y \mid x) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}. \tag{7}$$

LambdaRank. Besides comparing pairs and comparing one against many, list-wise ranking is widely adopted to sufficiently utilize the comprehensive information in candidate list. Taking ideas from LambdaRank [7, 53]:

$$\Pr(y_1 \succeq ... \succeq y_m) = \prod_{1<i<j<m} \sigma(r(x, y_i) - r(x, y_j)), \tag{8}$$

the policy optimization objective becomes $\mathcal{L}_{\text{lamb}}$:

$$\mathcal{L}_{\text{lamb}} = -\mathbb{E}\left[\sum_{1<i<j<m} \log \sigma\left(\gamma(y_i \mid x) - \gamma(y_j \mid x)\right)\right], \tag{9}$$

ListMLE.. Another idea for ranking a whole list is called ListMLE [47]. This idea has a strong basis in theory and looks at improving the whole list at once:

$$\Pr(y_1 \succeq ... \succeq y_m) = \prod_{i=1}^m \text{softmax}_i^m(r(x, y_i)) = \prod_{i=1}^m \frac{\exp(r(x, y_i))}{\exp(r(x, y_i)) + \sum_{j=i+1}^m \exp(r(x, y_j))} \tag{10}$$

In this case, the objective becomes $\mathcal{L}_{\text{lmle}}$:

$$\mathcal{L}_{\text{lmle}} = -\mathbb{E}\left[\sum_{i=1}^m \log \frac{\exp(\gamma(y_i \mid x))}{\exp(\gamma(y_i \mid x)) + \sum_{j=i}^m \exp(\gamma(y_j \mid x))}\right]. \tag{11}$$

Also see Table 1 for a summarization.

4.2 Hard negatives

Hard negatives are crucial for effective retriever training [33, 54]. This is because learning to tell the difference between harder examples can lead to better search tools [48]. When improving LLMs, the "negatives" are the responses that are not preferred. However, in current approaches [13, 49], the unpreferred responses are either set as the worst one (according to a reward model), or the one sampled from the initial SFT model.

Here, we summarize some types of negatives can be identified in iterative on-policy training, ordered by increasing difficulty:

- **Easiest**: A random, unrelated response to x;
- **Easy**: A response to a related but different prompt (x');
- **Hard**: An incorrect response to x generated with a high temperature;
- **Hardest**: An incorrect response to x generated with a low temperature.

Our observation is that, with a well-initialized policy LLM, as indicated by Figure 2(b) ($N = 1$), low temperatures tend to produce

Algorithm 1 LarPO: LLM alignment as iterative retriever preference optimization.

Require: Number of iterations T, number of new data per annotation phase M, number of generated responses for each prompt k, temperature for each iteration $\{t_i\}_{i=0}^{T}$, prompt dataset $\mathcal{D}_{\mathcal{X}} = \{x_i\}_{i=1}^{N}$, policy LLM π_{θ_0}, reward model r, learning rate γ, a ranking-based objective function $\mathcal{L}_{\text{rank}}$.
Ensure: Aligned LLM π_{θ_T}.

for $s := 0$ to T **do**
  Update behavior LLM: $\pi_\beta \leftarrow \pi_{\theta_s}$
  Preference dataset $\mathcal{D}_s = \{\}$
  for $i := 1$ to M **do**
    Sample prompt $x \sim \mathcal{D}_{\mathcal{X}}$
    // candidate list construction
    Sample $y_1, ..., y_k \sim \pi_\beta(\cdot|x)_{t_s}$
    // hard negatives
    Rank $\{y_i\}$ with r: $Y_x = \{y_j^{(r)}\}$, where $(r(y_a^{(r)}) > r(y_b^{(r)})), a < b$
    $\mathcal{D}_s \leftarrow \mathcal{D}_s \cup \{(x, Y_x)\}$
  end for
  // candidate list construction
  $\mathcal{D} \leftarrow \text{Merge}_{i=0}^{s} \mathcal{D}_s$
  while $\mathcal{D} \neq \emptyset$ **do**
    Sample a batch (x, Y_x) from $\mathcal{D}$
    Update $\mathcal{D} \leftarrow \mathcal{D} \setminus \{(x, Y_x)\}$
    // retriever optimization objective
    $\theta_s \leftarrow \theta_s - \gamma \cdot \nabla_\theta \mathcal{L}_{\text{rank}}(x, Y_x, \pi_\theta; \pi_\beta)$
  end while
  $\theta_{s+1} \leftarrow \theta_s$
end for

harder negatives, yielding the above ranking. According to [54], hardest negatives could be most important to LLM alignment.

4.3 Candidate list

In iterative retriever optimization, building the list of possible documents $[d_1, ..., d_m]$, that the reranker uses to create data for the next step is very important. Past research [53] has found that things like how big the list is and how candidates are chosen are especially important. Similarly, when improving LLMs using preferences over steps, building the list of possible responses $Y = [y_1, ..., y_m]$ is critical. We see two main things that affect how good this list Y: is including many options and remembering past results.

(1) **Inclusiveness** [33] refers to the **size** of the response list Y. A larger Y potentially encompasses more information.
(2) **Memorization** [53] refers whether previously generated responses Y' are included in the current list Y to preserve past results.

Given their importance in IR [33, 53], the impact of these factors on LLM alignment, however, remains largely under-explored.

4.4 The Proposed Solution: LarPO

Built on the insights and contexts we have discussed so far, we are ready to present our algorithmic details are provided in Algorithm 1.

- **Optimization objective**: We evaluate three distinct loss functions as the ranking objective ($\mathcal{L}_{\text{rank}}$): $\mathcal{L}_{\text{con}}$, $\mathcal{L}_{\text{lamb}}$, and $\mathcal{L}_{\text{lmle}}$;
- textbfHard negatives: For a given prompt, hard negative samples are constructed. These responses are generated by using a suitable temperature found through trying different values. More details of how the temperature are available in Appendix B.1.
- textbfCandidate list: In each iteration, we generate multiple (10) candidate responses considering inclusiveness.

In terms of memorization, the candidate pool for subsequent iterations includes all previously generated responses.

5 Main Results

We conduct detailed illustrations on the baselines compared with LarPO below.

- RRHF [52] scores responses via a logarithm of conditional probabilities and learns to align these probabilities with human preferences through ranking loss.
- SLiC-HF [56] proposes a sequence likelihood calibration method which can learn from human preference data.
- DPO [16] simplifies the PPO [31] algorithms into an offline direct optimization objective with the pairwise Bradley-Terry assumption.
- IPO [3] theoretically grounds pairwise assumption in DPO into a pointwise reward.
- CPO [50] adds a reward objective with sequence likelihood along with the SFT objective.
- KTO [15] adopts the Kahneman-Tversky model and proposes a method which directly maximizes the utility of generation instead of the likelihood of the preferences.
- RDPO [32] modifies DPO by including an additional regularization term to disentangle the influence of length.
- SimPO [26] further simplifies the DPO objective by using the average log probability of a sequence as the implicit reward and adding a target reward margin to the Bradley-Terry objective.
- Iterative DPO [49] identifies the challenge of offline preference optimization and proposes an iterative learning framework.

Both baselines and LarPO are trained on Ultrafeedback dataset [10] for fair comparison.

Datasets. We checked our models using two common tests: AlpacaEval2 [14] and MixEval [27]. These tests are made to see how well models can chat and follow instructions on many different kinds of questions. AlpacaEval2 has 805 questions from five different sets, while MixEval has 4000 general questions and 1000 questions that are harder. We followed the usual rules for checking models on each test. For AlpacaEval 2, we show both the simple win rate (WR) and the win rate adjusted for how long the answers are (LC). Together, these tests give a good picture of how well the models can follow instructions and solve problems.

Results. The starting scores for other methods on AlpacaEval 2 come straight from [26]. We checked the scores on MixEval ourselves using the publicly available models. We used the same scoring model (called a reward model), LLM-Blender [19], as the other methods to compare fairly. We also tried a stronger scoring model called FsfairX [13]. The results, shown in Table 2, demonstrate that LarPO consistently performs better than the other methods we compared against on both tests. It showed an average relative improvement of 38.9% on AlpacaEval2 and 13.7% on MixEval-Hard when using the same scoring model as the others. Using a stronger scoring model, we were able to make LarPO even better, improving its score by another 25.8% on the challenging AlpacaEval2 test.

6 Analyses

This section takes a closer look at why our method works based on the test results.

Table 3: Preference optimization objective study on AlpacaEval2 and MixEval. SFT corresponds to the initial chat model.

		AlpacaEval 2		MixEval	MixEval-Hard
	Method	LC Winrate	Winrate	Score	Score
Gemma2-2b-it	SFT	36.39	38.26	0.6545	0.2980
	pairwise	41.39	54.60	0.6740	0.3375
	contrastive	43.41	56.83	0.6745	0.3315
	ListMLE	**49.77**	**62.05**	0.6715	**0.3560**
	LambdaRank	43.76	60.56	**0.6750**	**0.3560**
Mistral-7b-it	SFT	21.14	14.22	0.7070	0.3610
	pairwise	36.43	41.86	0.7175	0.4105
	contrastive	38.44	42.61	0.7260	0.4340
	ListMLE	38.02	43.03	0.7360	0.4200
	LambdaRank	**40.29**	**46.21**	**0.7370**	**0.4400**

6.1 Retriever optimization objective

Experimental setting. Iterative preference optimization is performed on LLMs using the different learning objectives outlined in Section 4.1. Alignment experiments are conducted using the Gemma2-2b-it [41] and Mistral-7b-it [18] models, trained on the Ultrafeedback dataset [10]. Following the methodology of [13], we conduct three iterations of training and report the performance of the final checkpoint in Table 3. Model evaluations are performed on AlpacaEval2 [14] and MixEval [27]. Detailed settings can be found in Appendix B.2.

Observation. Table 3 shows the results, and here's what we noticed: (1) The goal of comparing one against many (contrastive optimization) generally worked better than just comparing pairs (like DPO). This is probably because it can use more negative examples in each learning step. (2) The goals that look at a whole list (ListMLE and LambdaRank) generally performed better than both comparing pairs and comparing one against many. This is because they use more complete information about the preferences within the list of possible answers.

6.2 Hard negatives

Experimental setting. The Mathstral-7b-it model is trained on the GSM8k training set and evaluated its performance on the GSM8k test set. Iterative DPO is employed as the RLHF method, with the gold or correct response designated as the positive example. The impact of different hard negative variants is investigated, as described in Section 4.2, with the results presented in Figure 3(a). Additionally, the influence of temperature on negative hardness with Lambdarank objective are examined using experiments on the AlpacaEval 2 dataset, with results shown in Figure 3(b). Detailed settings are in Appendix B.4 and B.5.

Observation. Figure 3(a) shows that how well the final LLM performs is directly related to how hard the negative examples were during training. Tougher negative examples consistently led to a better-performing LLM. Figure 3(b) also shows that, within a certain range, setting a lower temperature created harder negative examples, which resulted in a more effective final LLM. However, much lower temperature could lead to less diverse responses and finally lead to LLM alignment performance drop.

6.3 Candidate List

Experimental setting. To investigate the impact of inclusiveness and memorization on LLM alignment, experiments are conducted using Gemma2-2b-it, employing the same training settings as in our objective study. For the inclusiveness study, the performance of the trained LLM is evaluated using varying numbers of candidates in the list. For the memorization study, three approaches are compared: (i) using only the current iteration's responses, (ii) using responses from the current and previous iteration, and (iii) using responses from the current and all previous iterations. Detailed settings can be found in Appendix B.6 and B.3.

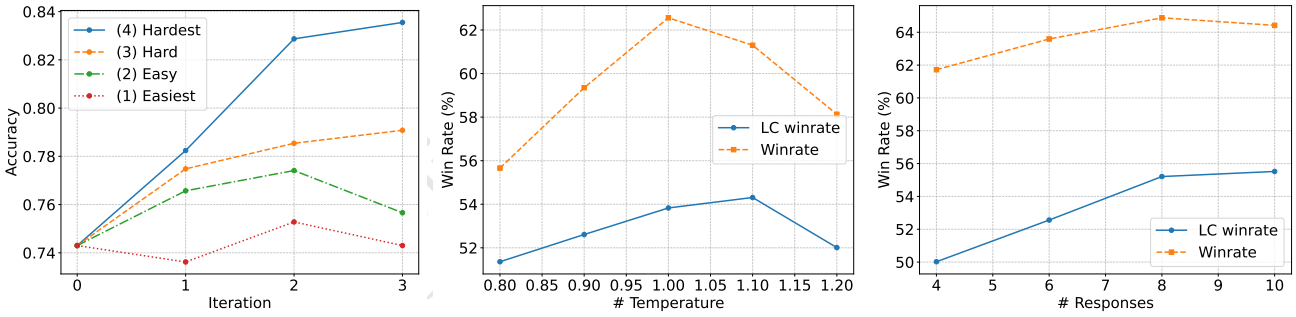
Table 4: Candidate list study with $\mathcal{L}_{\text{pair}}$ on Gemma2-2b-it. Previous iteration responses enhance performance.

	Alpaca Eval 2	
Method	LC Winrate	Winrate
SFT	47.03	48.38
Alignment (w. current)	55.06	66.56
Alignment (w. current + prev)	55.62	70.92
Alignment (w. current + all prev)	56.02	72.50

Observation. Figure 3(c) illustrates the significant impact of candidate list size on LLM alignment performance. As the candidate list size increases, performance improves, albeit with a diminishing rate of return. This is intuitive, given that a bigger candidate list size can contribute to more hard negatives and potentially benefit the model learning [33]. Table 4 demonstrates that incorporating responses from previous iterations can enhance performance. This is potentially because introducing previous responses can make the candidate list more comprehensive and lead to better preference signal capturing.

Table 2: Evaluations on AlpacaEval 2 and MixEval. LC WR and WR denote length-controlled win rate and win rate respectively. Offline baseline performances on AlpacaEval 2 are from [26]. We use LLM-blender [19] as the reward model for a fair comparison with the baselines and also report the result with a stronger reward model FsfairX [13]

Model	Mistral-Base (7B)				Mistral-Instruct (7B)			
	Alpaca Eval 2		MixEval	MixEval-Hard	Alpaca Eval 2		MixEval	MixEval-Hard
	LC WR	WR	Score	Score	LC WR	WR	Score	Score
SFT	8.4	6.2	0.602	0.279	17.1	14.7	0.707	0.361
Reward model: LLM-Blender [19]								
RRHF	11.6	10.2	0.600	0.312	25.3	24.8	0.700	0.380
SLiC-HF	10.9	8.9	0.679	0.334	24.1	24.6	0.700	0.381
DPO	15.1	12.5	0.686	0.341	26.8	24.9	0.702	0.355
IPO	11.8	9.4	0.673	0.326	20.3	20.3	0.695	0.376
CPO	9.8	8.9	0.632	0.307	23.8	28.8	0.699	0.405
KTO	13.1	9.1	**0.704**	0.351	24.5	23.6	0.692	0.358
RDPO	17.4	12.8	0.693	0.355	27.3	24.5	0.695	0.364
SimPO	21.5	20.8	0.672	0.347	32.1	34.8	0.702	0.363
Iterative DPO	18.9	16.7	0.660	0.341	20.4	24.8	0.719	0.389
LarPO (Contrastive)	31.6	30.8	0.703	0.409	32.7	38.6	0.718	**0.418**
LarPO (LambdaRank)	**34.9**	**37.2**	0.695	**0.452**	**32.9**	**38.9**	**0.720**	0.417
LarPO (ListMLE)	31.1	32.1	0.669	0.390	29.7	36.2	0.709	0.397
Reward model: FsfairX [13]								
LarPO (Contrastive)	**41.5**	**42.9**	0.718	0.417	**43.0**	**53.8**	0.718	0.425
LarPO (LambdaRank)	35.8	34.1	0.717	0.431	41.9	48.1	**0.740**	**0.440**
LarPO (ListMLE)	36.6	37.8	**0.730**	**0.423**	39.6	48.1	0.717	0.397



(a) Hard negative study (b) Temperature & hard negatives (c) Candidate list length study

Figure 3: Hard negative and candidate list study. (a) Hard negative study with $\mathcal{L}_{\text{pair}}$ on GSM8K with Mathstral-7b-it model. We explore four negative settings: (1) a random response not related to the given prompt; (2) a response to a related prompt; (3) an incorrect response to the given prompt with high temperature; (4) an incorrect response to the given prompt with suitable temperature. Hardness: (4)>(3)>(2)>(1). The harder the negatives are, the stronger the trained LLM is. (b) Training temperature study with $\mathcal{L}_{\text{pair}}$ on Mistral-7b-it and Alpaca Eval 2. Within a specific range (> 1), lower temperature leads to harder negative and benefit the trained LLM. However, much lower temperature could lead to less diverse responses and finally lead to LLM alignment performance drop. (c) Candidate list size study with $\mathcal{L}_{\text{con}}$ on Mistral-7b-it. As the candidate list size increases, alignment performance improves.

7 Conclusions

In this work, we connect the information retriever optimization with the LLM alignment, and leverage several key insights from IR such as reranker, hard negative, and candidate list construction to improve the algorithmic design of direct preference optimization. Specifically, we propose LLM Alignment as Retriever Preference Optimization (LarPO), a new alignment method using IR principles to improve alignment quality. A systematic analysis of the proposed method with both theoretical insights and empirical experiments validate its effectiveness. Our experiments show LarPO gives average improvements of 38.9% on AlpacaEval2 and 13.7% on

MixEval-Hard compared to the strong baselines such as iterative DPO, SimPO, and RDPO.

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] Anthropic. Introducing claude. 2023.

[3] 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.

[4] 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.

[5] 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.

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

[7] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[8] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[9] 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.

[10] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. In *Forty-first International Conference on Machine Learning*, 2024.

[11] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[12] Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

[13] 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.

[14] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

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

[16] 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.

[17] S Humeau. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*, 2019.

[18] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[19] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*, 2023.

[20] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

[21] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[22] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.

[23] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature, 2022.

[24] Tianqi Liu, Zhen Qin, Junru Wu, Jiaming Shen, Misha Khalman, Rishabh Joshi, Yao Zhao, Mohammad Saleh, Simon Baumgartner, Jialu Liu, et al. Lipo: Listwise preference optimization through learning-to-rank. *arXiv preprint arXiv:2402.01878*, 2024.

[25] Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. Ranked list truncation for large language model-based re-ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 141–151, 2024.

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

[27] Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. Mixeval: Deriving wisdom of the crowd from llm benchmark mixtures. *arXiv preprint arXiv:2406.06565*, 2024.

[28] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[29] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713*, 2020.

[30] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[31] 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.

[32] Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. Disentangling length from quality in direct preference optimization. *arXiv preprint arXiv:2403.19159*, 2024.

[33] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*, 2020.

[34] 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*, 36, 2024.

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

[36] Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen, Geoffrey Cideron, et al. Bond: Aligning llms with best-of-n distillation. *arXiv preprint arXiv:2407.14622*, 2024.

[37] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[38] 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.

[39] Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843, 2022.

[40] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[41] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[42] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Shengyi Huang, Kashif Rasul, Alvaro Bartolome, Alexander M. Rush, and Thomas Wolf. The Alignment Handbook.

[43] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[44] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[45] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*, 2023.

[46] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[47] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, 2008.

[48] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

[49] 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.

[50] Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*, 2024.

[51] Wenda Xu, Jiachen Li, William Yang Wang, and Lei Li. Bpo: Staying close to the behavior llm creates better online llm alignment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11125–11139, 2024.

[52] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

[53] Hansi Zeng, Hamed Zamani, and Vishwa Vinay. Curriculum learning for dense retrieval distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1979–1983, 2022.

[54] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, 2021.

[55] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60, 2024.

[56] 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.

[57] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*, 2023.

[58] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2308–2313, 2023.

A LarPO retriever optimization objective

We provide the proof for different variants of LarPO's objective functions.

A.1 Contrastive ranking

Theorem A.1. *Let x be a prompt and $(y_w, y_l^{(1)}, ..., y_l^{(m)})$ be the responses for x under the contrastive assumption (Eq.(6)). Then the objective function to learn the LLM π_θ:*

$$\mathcal{L}_{\text{con}} = -\mathbb{E}\left[\log \frac{\exp\big(\gamma(y_w \mid x)\big)}{\exp\big(\gamma(y_w \mid x)\big) + \sum_{i=1}^{m} \exp\big(\gamma(y_l^{(i)} \mid x)\big)}\right], \quad \text{where} \quad \gamma(y \mid x) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}. \tag{12}$$

Proof. From [34], we know that

$$r(x, y) = \beta \log \frac{\pi_{\text{llm}}(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z, \tag{13}$$

where $Z = \sum_{y'} \pi_{\text{ref}}(y'|x) \exp(\frac{1}{\beta} r(x, y'))$.

Then,

$$\begin{aligned}
\mathbb{P}\text{r}(y_w \succeq y_l^{(1)}, ..., y_w \succeq y_l^{(m)}) &= \text{softmax}(r(x, y_w)) \\
&= \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \sum_{i=1}^{m} \exp(r(x, y_l^{(i)}))} \\
&= \frac{1}{1 + \sum_{i=1}^{m} \exp(r(x, y_l^{(i)}) - r(x, y_w))} \\
&= \frac{1}{1 + \sum_{i=1}^{m} \exp(\gamma(y_l^{(i)} \mid x) + \beta \log Z - \gamma(y_w \mid x) - \beta \log Z)} \\
&= \frac{1}{1 + \sum_{i=1}^{m} \exp(\gamma(y_l^{(i)} \mid x) - \gamma(y_w \mid x))} \\
&= \frac{\exp\big(\gamma(y_w \mid x)\big)}{\exp\big(\gamma(y_w \mid x)\big) + \sum_{i=1}^{m} \exp\big(\gamma(y_l^{(i)} \mid x)\big)}
\end{aligned} \tag{14}$$

We can learn π_θ by maximizing the logarithm-likelihood:

$$\max \log \mathbb{P}\text{r}(y_w \succeq y_l^{(1)}, \ldots, y_w \succeq y_l^{(m)}) \Leftrightarrow \min -\log \mathbb{P}\text{r}(y_w \succeq y_l^{(1)}, \ldots, y_w \succeq y_l^{(m)}) = \mathcal{L}, \tag{15}$$

$$\therefore \mathcal{L}_{\text{con}} = -\mathbb{E}\left[\log \frac{\exp\big(\gamma(y_w \mid x)\big)}{\exp\big(\gamma(y_w \mid x)\big) + \sum_{i=1}^{m} \exp\big(\gamma(y_l^{(i)} \mid x)\big)}\right], \tag{16}$$

$$\text{where} \quad \gamma(y \mid x) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}. \tag{17}$$

A.2 LambdaRank ranking

Theorem A.2. *Let x be a prompt and $(y_1, ..., y_m)$ be the responses for x under the LambdaRank assumption (Eq.(8)). Then the objective function to learn the LLM π_θ:*

$$\mathcal{L}_{\text{lamb}} = -\mathbb{E}\left[\sum_{1<i<j<m} \log \sigma\Big(\gamma(y_i \mid x) - \gamma(y_j \mid x)\Big)\right]. \tag{18}$$

Proof.

$$\begin{aligned}
\mathbb{P}\text{r}(y_1 \succeq \ldots \succeq y_m) &= \prod_{1<i<j<m} \sigma(r(x, y_i) - r(x, y_j)) \\
&= \prod_{1<i<j<m} \sigma(\gamma(x, y_i) + \beta \log Z - \gamma(x, y_j) - \beta \log Z) \\
&= \prod_{1<i<j<m} \sigma(\gamma(y_i \mid x) - \gamma(y_j \mid x)).
\end{aligned} \tag{19}$$

We can learn π_θ by maximizing the logarithm-likelihood:

$$\max \log \mathbb{P}\mathrm{r}(y_w \succeq y_l^{(1)}, \ldots, y_w \succeq y_l^{(m)}) \Leftrightarrow \min -\log \mathbb{P}\mathrm{r}(y_w \succeq y_l^{(1)}, \ldots, y_w \succeq y_l^{(m)}) = \mathcal{L}, \tag{20}$$

$$\therefore \mathcal{L}_{\text{lamb}} = -\mathbb{E}\left[\sum_{1<i<j<m} \log \sigma\Big(\gamma(y_i \mid x) - \gamma(y_j \mid x)\Big)\right], \tag{21}$$

$$\text{where} \quad \gamma(y \mid x) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}. \tag{22}$$

A.3 ListMLE ranking

Theorem A.3. *Let x be a prompt and $(y_1, ..., y_m)$ be the responses for x under the ListMLE assumption (Eq.(10)). Then the objective function to learn the LLM π_θ:*

$$\mathcal{L}_{\text{lmle}} = -\mathbb{E}\left[\sum_{i=1}^{m} \log \frac{\exp\big(\gamma(y_i \mid x)\big)}{\exp\big(\gamma(y_i \mid x)\big) + \sum_{j=i}^{m} \exp\big(\gamma(y_j \mid x)\big)}\right]. \tag{23}$$

Proof. From Eq.(14),

$$\begin{aligned}\mathbb{P}\mathrm{r}(y_1 \succeq \ldots \succeq y_m) &= \prod_{i=1}^{m} \mathbb{P}\mathrm{r}(y_i \succeq y_{i+1}, ..., y_i \succeq y_m) \\ &= \prod_{i=1}^{m} \frac{\exp(\gamma(y_i \mid x))}{\exp(\gamma(y_i \mid x)) + \sum_{j=i+1}^{m} \exp(\gamma(y_j \mid x))}\end{aligned}. \tag{24}$$

We can learn π_θ by maximizing the logarithm-likelihood:

$$\max \log \mathbb{P}\mathrm{r}(y_w \succeq y_l^{(1)}, \ldots, y_w \succeq y_l^{(m)}) \Leftrightarrow \min -\log \mathbb{P}\mathrm{r}(y_w \succeq y_l^{(1)}, \ldots, y_w \succeq y_l^{(m)}) = \mathcal{L}, \tag{25}$$

$$\therefore \mathcal{L}_{\text{lmle}} = -\mathbb{E}\left[\sum_{i=1}^{m} \log \frac{\exp\big(\gamma(y_i \mid x)\big)}{\exp\big(\gamma(y_i \mid x)\big) + \sum_{j=i}^{m} \exp\big(\gamma(y_j \mid x)\big)}\right], \tag{26}$$

$$\text{where} \quad \gamma(y \mid x) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}. \tag{27}$$

B Experiment settings

B.1 Table 2

We conduct evaluation on two widely used benchmark: AlpacaEval2 [14] and MixEval [27]. We consider two base models: Mistral-7b-base and Mistral-7b-it. For Mistral-7b-base, we first conduct supervised finetuning following [26] before the preference optimization.

The performance scores for offline preference optimization baselines are from SimPO [26]. To have a fair comparison with these baselines, we adopt the same off-the-shelf reward model [19] as in SimPO for the iterative DPO baseline and LarPO.

For the iterative DPO baseline, we generate 2 responses for each prompt, score them with the off-the-shelf reward model and construct the preference pair data to tune the model.

For LarPO (contrastive $\mathcal{L}_{\text{con}}$), we generate 10 responses each iteration and score them with the reward model. The top-1 ranked response and the bottom-3 ranked responses are adopted as the chose response and rejected responses respectively. Generation temperature is selected as 1 and 0.8 for Mistral-7b-base and Mistral-7b-it respectively (we search it among 0.8, 0.9, 1.0, 1.1, 1.2).

For LarPO (LambdaRank $\mathcal{L}_{\text{lamb}}$), we generate 10 responses each iteration and score them with the reward model. The top-2 ranked response and the bottom-2 ranked responses are adopted as the chose response and rejected responses respectively. Generation temperature is selected as 1 and 0.8 for Mistral-7b-base and Mistral-7b-it respectively (we search it among 0.8, 0.9, 1.0, 1.1, 1.2).

For LarPO (ListMLE $\mathcal{L}_{\text{lmle}}$), we generate 10 responses each iteration and score them with the reward model. The top-2 ranked response and the bottom-2 ranked responses are adopted as the chose response and rejected responses respectively. Generation temperature is selected as 1 and 0.8 for Mistral-7b-base and Mistral-7b-it respectively (we search it among 0.8, 0.9, 1.0, 1.1, 1.2).

LarPO can achieve even stronger performance with stronger off-the-shelf reward model [13].

B.2 Table 3

We conduct experiments on both Gemma2-2b-it [41] and Mistral-7b-it [18]. Following [42] and [13], we perform training on UltraFeedback dataset for 3 iterations and show the performance of the final model checkpoint. We use the pretrained reward model from [13]. The learning rate is set as 5e-7 and we train the LLM for 2 epochs per iteration.

For the pairwise objective, we generate 2 responses for each prompt and construct the preference pair data with the reward model. For the others, we generate 4 responses per prompt and rank them with the reward model. For the contrastive objective, we construct the 1-vs-N

Table 5: Preference optimization objective study on AlpacaEval2 and MixEval. For AlpacaEval2, we report the result with both opensource LLM evaluator `alpaca_eval_llama3_70b_fn` and GPT4 evaluator `alpaca_eval_gpt4_turbo_fn`. SFT corresponds to the initial chat model.

		AlpacaEval 2 (opensource LLM)		AlpacaEval 2 (GPT-4)		MixEval	MixEval-Hard
	Method	LC Winrate	Winrate	LC Winrate	Winrate	Score	Score
Gemma2-2b-it	SFT	47.03	48.38	36.39	38.26	0.6545	0.2980
	pairwise	55.06	66.56	41.39	54.60	0.6740	0.3375
	contrastive	60.44	72.35	43.41	56.83	0.6745	0.3315
	ListMLE	63.05	76.09	49.77	62.05	0.6715	0.3560
	LambdaRank	58.73	74.09	43.76	60.56	0.6750	0.3560
Mistral-7b-it	SFT	27.04	17.41	21.14	14.22	0.7070	0.3610
	pairwise	49.75	55.07	36.43	41.86	0.7175	0.4105
	contrastive	52.03	60.15	38.44	42.61	0.7260	0.4340
	ListMLE	48.84	56.73	38.02	43.03	0.7360	0.4200
	LambdaRank	51.98	59.73	40.29	46.21	0.7370	0.4400

data with the top-1 ranked response and the other responses. For the listMLE and lambdarank objective, we take the top-2 as positives and the last-2 as the negatives. Experiments with opensource LLM as the evaluator (`alpaca_eval_llama3_70b_fn`) can be found in Table 5.

B.3 Table 4

We adopt Gemma2-2b-it as the initial model. All the models are trained with iterative DPO for 3 iterations. We use the off-the-shelf reward model [13]. We generate 2 responses for each prompt in each iteration. For "w. current", we only use the scored responses in the current iteration for preference optimization data construction. For "w. current + prev", we rank the responses in the current iteration and the previous one iteration, and construct the preference pair data with the top-1 and bottom-1 ranked responses. For "w. current + all prev", we rank all the responses for the prompt in the current and previous iterations and construct the preference pair data. For "single temperature", we only adopt temperature 1 and generate 2 responses for reward model scoring. For "diverse temperature", we generate 2 responses with temperature 1 and 0.5 respective and rank the 4 responses to construct the preference data with the reward model.

B.4 Figure 3(a)

We conduct training with the prompts in the training set of GSM8K and perform evaluation on GSM8K testing set. We conduct learning rate search and finalize it to be 2e-7. The learning is performed for 3 iterations.

We make explanations of how we construct the four types of negative settings: For (1) a random response not related to the given prompt, we select a response for a random prompt in Ultrafeedback. For (2) a response to a related prompt, we pick up a response for a different prompt in the GSM8K training set. For (3) an incorrect response to the given prompt with high temperature, we select the temperature to be 1. For (4) an incorrect response to the given prompt with low temperature, we select the temperature to be 0.7.

B.5 Figure 3(b)

We conduct experiments on both Gemma2-2b-it and Mistral-7B-it models. For both LLMs, we conduct iterative DPO for 3 iterations and report the performance of the final model. We perform evaluation on Alpaca Eval2 with `alpaca_eval_llama3_70b_fn` as the evaluator.

For temperature study, we find that under a specific temperature threshold, repeatedly generated responses will be large identical for all LLMs and cannot be used to construct preference data, while the threshold varies for different LLMs. The "low" and "high" refer to the value of those selected temperatures. We also conduct experiments on Gemma2-2b-it model and show the results in Figure 4.

B.6 Figure 3(c)

We adopt Mistral-7b-it as the initial LLM and the contrastive objective (Eq. 12) in iterative preference optimization. We generate 4/6/8/10 responses with the LLM and score the responses with the off-the-shelf reward model [13]. The top-1 scored response is adopted as the positive response and the other responses are treated as the negative responses to construct the 1-vs-N training data. The temperature is set as 1 to generate the responses.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

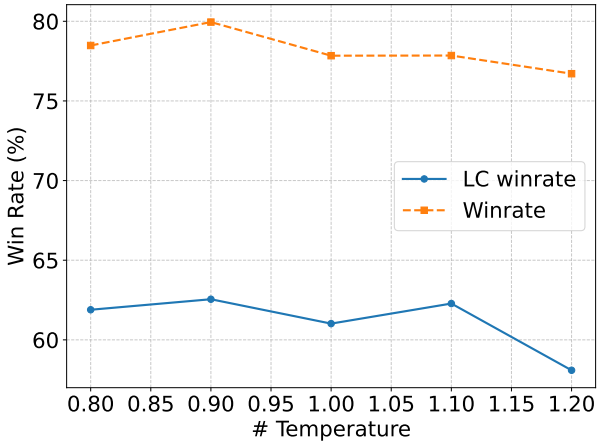


Figure 4: Training temperature study with $\mathcal{L}_{\text{pair}}$ on Gemma2-2b-it and Alpaca Eval 2. Within a specific range (> 0.9), lower temperature leads to harder negative and benefit the trained LLM. However, temperature lower than this range can cause preferred and rejected responses non-distinguishable and lead to degrade training.