DiPPO: Learning Diffusion Prompt Proposers from Scalar Rewards

Anonymous authors
Paper under double-blind review

Abstract

Automatic prompt optimization improves the downstream performance of a frozen target language model without updating its parameters. However, many existing approaches search over prompt strings with a fixed proposer, often relying on strong external language models as proposers or critics, or on human-written initial instructions. We introduce Diffusion Prompt Proposal Optimization (DiPPO), which directly optimizes a masked diffusion language model as a prompt proposer using downstream task rewards. Bidirectional attention in the proposer enables prompt-slot infilling conditioned on the subsequent query. DiPPO samples candidate prompts and updates the proposer using only scalar rewards indicating whether the frozen target model answers correctly. It can discover prompts without a human-written seed instruction or, when one is available, use it as an anchor for preference-based refinement. Across ten benchmarks, the seed-free variant outperforms all methods that do not use a human instruction and exceeds human-written reference prompts on six tasks. The human-anchored variant improves on the human-written instruction on eight tasks, achieves the best performance among all compared methods on four tasks, and records the highest average score, including against methods that use stronger critic models. These results show that a 7B open-weight diffusion language model can not only discover effective prompts from scratch without human-written seeds, but also further improve human-written instructions when they are available.

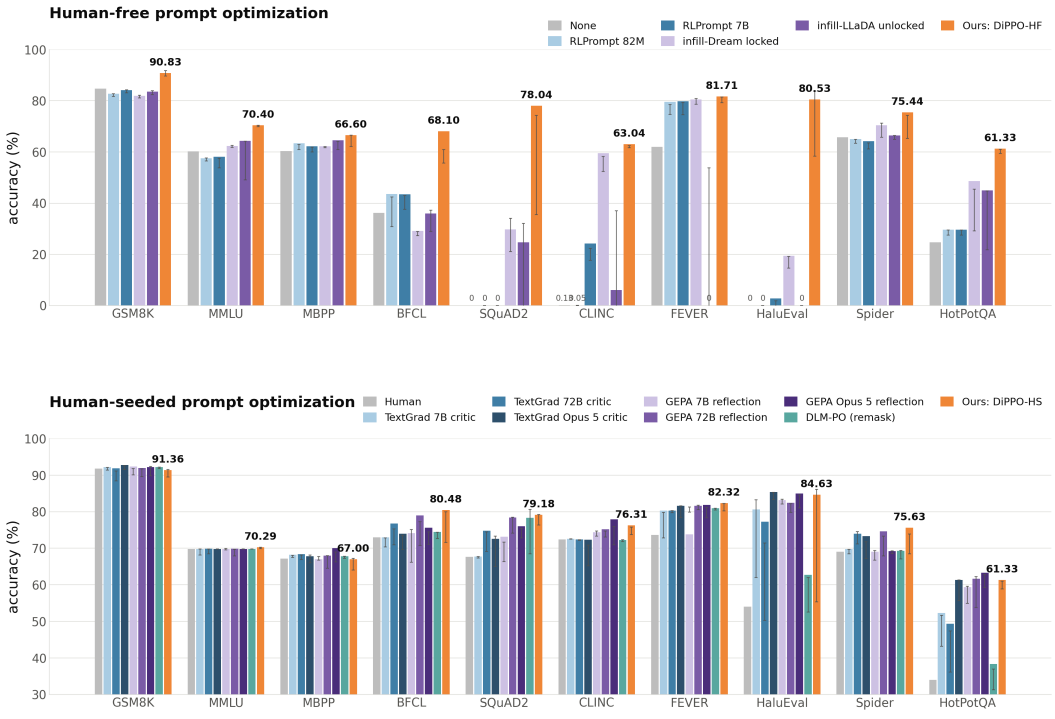

Figure 1: Human-free and human-seeded results on ten benchmarks (Section 4.1). Error bars show the seed mean and standard deviation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

The performance of a large language model can vary substantially with its instruction, even when its weights remain unchanged (Zhou et al., 2023; Pryzant et al., 2023). Automatic prompt optimization replaces manual trial and error with candidate generation and evaluation (Zhou et al., 2023; Yang et al., 2024; Guo et al., 2024). In many search-based methods, however, the proposer remains fixed while past candidates, their scores or natural-language feedback guide subsequent proposals (Yang et al., 2024; Yuksekgonul et al., 2024; Agrawal et al., 2025). We instead ask whether the proposal distribution itself can be learned from task rewards.

In the input to the target model, an instruction p precedes a query x, yet its usefulness depends on the query that follows. Masked diffusion language models reconstruct masked positions from bidirectional context (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025), so they can fill an instruction slot while conditioning on the query to its right. This infilling preserves the instruction–query order used by the target model.

We propose *Diffusion Prompt Proposal Optimization* (DiPPO), which trains an open-weight diffusion language model as a prompt proposer $q_\phi(p \mid x)$ from the scalar task scores of a frozen target model M, updating only its LoRA parameters (Hu et al., 2021). **Human-Free Prompt Optimization (DiPPO-HF)** starts from a fully masked slot and learns with GRPO and diffusion trajectory likelihoods (Shao et al., 2024; Wang et al., 2026a). **Human-Seeded Prompt Optimization (DiPPO-HS)** treats a human-written instruction as a competing candidate and applies DPO to reward-derived preference pairs (Rafailov et al., 2023). Neither variant requires natural-language critiques, and one instruction selected on a development set is applied unchanged to every test input.

Across ten tasks spanning classification, question answering, semantic parsing and function calling (Larson et al., 2019; Rajpurkar et al., 2018; Yang et al., 2018; Yu et al., 2018; Patil et al., 2025), DiPPO-HF outperforms all methods that do not use a human instruction and exceeds the human-written instruction on six tasks. DiPPO-HS improves on the human-written instruction on eight tasks and achieves the highest score on four, including against methods that use stronger critic models. These results show that an open-weight d-LLM can learn to discover effective instructions from scalar task rewards alone.

2 RELATED WORK

2.1 AUTOMATIC PROMPT OPTIMIZATION

Automatic prompt optimization treats natural-language instructions as a discrete search space. APE (Zhou et al., 2023) evaluates and selects language-model-generated candidates by task performance, whereas OPRO (Yang et al., 2024) conditions new proposals on previous candidates and their scores. EvoPrompt (Guo et al., 2024) uses mutation and crossover for evolutionary search, and TextGrad (Yuksekgonul et al., 2024) expresses natural-language feedback on execution results as textual gradients for revising instructions. MIPROv2 (Opsahl-Ong et al., 2024) jointly optimizes instructions and demonstrations for multistage language-model programs, while GEPA (Agrawal et al., 2025) combines natural-language reflection on execution trajectories with Pareto-based selection. Rainbow Teaming (Samvelyan et al., 2024) formulates adversarial prompt generation as a quality-diversity problem and builds an archive of effective, diverse candidates. These methods differ in how they guide the next proposal, whether through past candidates and their scores, natural-language feedback on execution, or diversity-preserving archives, but they keep the proposer itself fixed. In contrast, DiPPO updates the proposer that generates the candidates.

Other work learns a prompt-generating policy from downstream feedback. RLPrompt (Deng et al., 2022) trains a discrete prompt policy using rewards from a frozen task model. PRL (Batorski et al., 2025) uses reinforcement learning to generate prompts and novel few-shot examples, whereas Prompt-R1 (Liu et al., 2025) trains a smaller model to prompt a larger model through multi-turn interactions. DiPPO also learns the proposer, but uses a masked diffusion model to generate a prompt conditioned on the query that follows it.

2.2 Diffusion Language Models and Reward-Based Post-training

Masked diffusion language models repeatedly reconstruct masked tokens using the observed tokens on both sides as context. MDLM (Sahoo et al., 2024) develops objectives and sampling methods for masked diffusion language modeling, and LLaDA (Nie et al., 2025) scales masked diffusion to large language models. Dream (Ye et al., 2025) supports arbitrary-order generation and infilling with bidirectional attention. Autoregressive models can also support infilling through input rearrangement or fill-in-the-middle training (Bavarian et al., 2022). Masked denoising provides a direct interface for reconstructing a prompt slot while keeping the observed query fixed. DiPPO uses this interface to generate a prompt p before a query x, preserving their order in the target input $[p; x]$.

Reward and preference signals have also been used to post-train diffusion language models. d1 (Zhao et al., 2025) adapts GRPO to masked diffusion models, and LLaDA 1.5 (Zhu et al., 2025) applies preference optimization to LLaDA. d2 (Wang et al., 2026a) incorporates the likelihood of the diffusion sampling trajectory into policy learning. DiPPO-HF adopts the trajectory-likelihood-based GRPO of d2 without introducing a new diffusion reinforcement-learning algorithm, and applies it to generating a reusable task instruction rather than an answer to an individual query.

2.3 Prompt Optimization with Diffusion Language Models

Fujinuma & Sakaguchi (2026) fine-tune instruction-tuned diffusion language models to complete masked prompt templates conditioned on few-shot examples, and apply a prompt chosen through validation to test inputs. They also use this infilling to refine prompts produced by search-based optimizers (Khattab et al., 2024; Opsahl-Ong et al., 2024; Agrawal et al., 2025).

Wang et al. (2026b) use Dream-7B to iteratively edit spans of a system prompt through masked denoising, conditioned on interaction traces of queries, target-model responses and optional feedback. We refer to their approach as DLM-PO in this paper.

These studies show that diffusion infilling is effective for completing and revising prompts. DiPPO builds on this line of work and focuses on training the proposer itself from the task reward of the target model. DiPPO-HF discovers instructions without an initial human instruction, while DiPPO-HS includes a human-written instruction in reward-based preference comparisons to search for improved candidates. Both settings select a single instruction for reuse across the task.

3 DiPPO: Diffusion Prompt Proposal Optimization

Figure 2 traces one training step. The d-LLM proposer q_ϕ reads the query x behind a masked prompt slot and, through bidirectional attention, fills the slot with several competing candidate prompts p. Each candidate is placed before the query and passed to the frozen target LM M, and the task evaluator μ_t scores the response to give a scalar reward r. This reward updates only the trainable LoRA ϕ of the proposer. The training method differs depending on whether a human instruction is available. DiPPO-HF weights prompts by their group-relative advantage $\widehat{A}$, so that prompts scoring above their group become more likely. DiPPO-HS lets the human instruction h compete with the generated prompts and selects the preferred prompt by reward. The updated proposer then generates the next set of candidates.

Problem formulation. Let $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$ denote the training set, where x_i is a query and y_i is its reference answer. The index i always denotes an example of $\mathcal{D}_{\text{train}}$. Let M be the frozen target LM and μ_t the task evaluator for task t (Figure 2b). Our goal is to learn a proposal distribution $q_\phi(p \mid x)$ that generates prompts yielding high task scores when the target model receives $[p; x]$. The d-LLM proposer q_ϕ is an open-weight d-LLM, and only its trainable LoRA parameters ϕ are updated (Hu et al., 2021) (Figure 2a). Search-based optimization retains information in candidate strings and archives, whereas DiPPO also incorporates task feedback into the parameters of the proposer. For a training example (x_i, y_i), the proposer input $z_i = [[\text{MASK}]^L; x_i]$ consists of a masked prompt slot of L mask tokens followed by the query (Figure 2a).

Reward computation. The task evaluator scores the response of the frozen target LM to each candidate and returns a scalar reward (Figure 2b). Equation (1) summarizes this process for DiPPO-

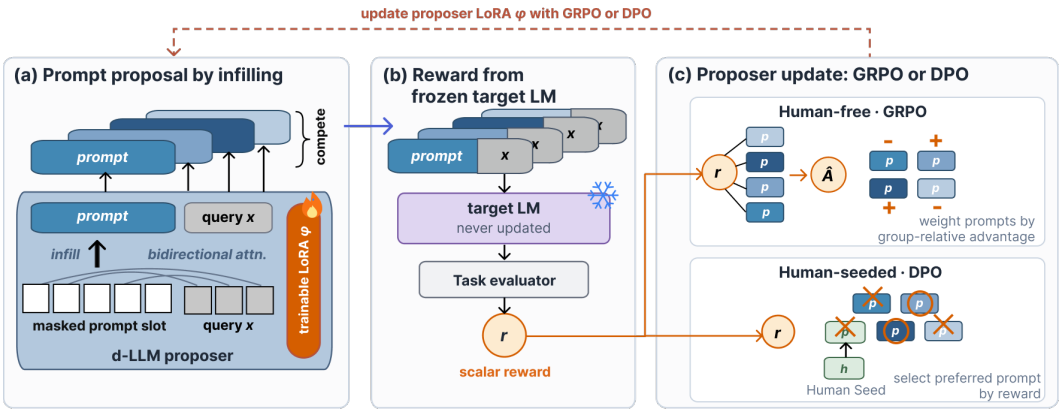

Figure 2: Overview of one DiPPO training step. (a) The d-LLM proposer fills a masked prompt slot before the query x and samples competing candidate prompts. (b) The frozen target LM responds to each prompt, and the task evaluator returns a scalar reward r. (c) DiPPO-HF (GRPO) weights prompts by group-relative advantage $\widehat{A}$, and DiPPO-HS (DPO) selects the preferred prompt by reward among generated candidates and the human instruction h. In both settings, only the trainable LoRA ϕ of the proposer is updated.

HF, which computes rewards on the same example that was used to generate the candidates. From z_i, the proposer reconstructs the masked region through iterative denoising to produce G candidates $p_{i,g}$. Each candidate precedes the query x_i in the target input, and the evaluator compares the resulting response $\hat{y}_{i,g}$ with the reference answer y_i

$$\begin{aligned} \text{Prompt proposal:} \quad & p_{i,g} \sim q_\phi(\cdot \mid z_i), \quad g = 1, \ldots, G, \\ \text{Target response:} \quad & \hat{y}_{i,g} \sim M(\cdot \mid [p_{i,g}; x_i]), \\ \text{Task reward:} \quad & r_{i,g} = \mu_t(\hat{y}_{i,g}, y_i). \end{aligned} \tag{1}$$

In Eq. (1), $r_{i,g}$ is the task score obtained by applying candidate $p_{i,g}$ to query x_i, following the scoring criterion of each benchmark, such as exact match. The target model remains frozen, and proposer updates use only scalar task scores, without natural-language critiques or a separate critic model. DiPPO-HS instead scores candidates on a shared query batch to construct preferences (Section 3.2). We write $q_\phi(\cdot \mid x_i)$ for the proposal distribution conditioned on the masked input z_i.

Prompt selection and deployment. Training generates different candidates conditioned on individual queries, but the final output is a single instruction reused across the task. Distinct candidates are collected in an archive $\mathcal{A}$ and evaluated on a development set $\mathcal{D}_{\text{dev}}$ that does not overlap with the test set. Because $\mathcal{D}_{\text{dev}}$ is a different sample from $\mathcal{D}_{\text{train}}$, we write its examples without the index i and define the development score of a prompt p as $S_{\text{dev}}(p) = |\mathcal{D}_{\text{dev}}|^{-1} \sum_{(x,y)\in\mathcal{D}_{\text{dev}}} \mu_t(\hat{y}(p,x), y)$, where $\hat{y}(p,x)$ is the response of the target model to $[p;x]$. The prompt with the highest development score is deployed unchanged on all test inputs, with no additional prompt generation or model updates. Section 3.3 specifies the candidates over which this maximum is taken.

3.1 Human-Free Prompt Optimization

DiPPO-HF starts from fully masked prompt slots without a human-written instruction. The proposer receives only z_i, with no additional task description or instruction examples. For each training example (x_i, y_i), the previous proposer $q_{\phi_{\text{old}}}$ samples G candidates $p_{i,1}, \ldots, p_{i,G}$, and their rewards $r_{i,g}$ from Eq. (1) are normalized within the group to obtain group-relative advantages $\widehat{A}_{i,g} = (r_{i,g} - \text{mean}_j\, r_{i,j})/(\text{std}_j\, r_{i,j} + \epsilon_A)$, where ϵ_A is a small constant (Figure 2c, top). These group-relative rewards remove the need for a separately trained value model.

Diffusion generation involves multiple denoising steps, so the left-to-right likelihood factorization of autoregressive models does not directly apply. Following d2 (Wang et al., 2026a), DiPPO computes the log-likelihood $\log q_\phi(\tau_{i,g} \mid x_i)$ of the denoising trajectory $\tau_{i,g}$ that generated candidate

$p_{i,g}$. Writing $p_{i,g,l}$ for the l-th token of $p_{i,g}$ and $s_{i,g,l}$ for the partially reconstructed state at the denoising step where that token was finalized, this log-likelihood decomposes into the token log-probabilities $\log q_\phi(p_{i,g,l} \mid s_{i,g,l}, x_i)$. These values define the token-level probability ratio $\rho_{i,g,l}$ between the current and previous proposers, and the group-relative advantage $\widehat{A}_{i,g}$ of each candidate is shared across its L tokens. The LoRA parameters are then updated to maximize the clipped GRPO objective (Shao et al., 2024) in Eq. (2)

$$J_{\mathrm{GRPO}}(\phi) = \mathbb{E}\left[\frac{1}{GL}\sum_{g=1}^{G}\sum_{l=1}^{L}\left\{\min\Big(\rho_{i,g,l}\widehat{A}_{i,g}, \mathrm{clip}(\rho_{i,g,l}, 1-\epsilon, 1+\epsilon)\widehat{A}_{i,g}\Big) - \beta\widehat{\mathcal{K}}_{i,g,l}\right\}\right], \qquad (2)$$
$$\rho_{i,g,l} = \frac{q_\phi(p_{i,g,l} \mid s_{i,g,l}, x_i)}{q_{\phi_{\mathrm{old}}}(p_{i,g,l} \mid s_{i,g,l}, x_i)}.$$

The expectation in Eq. (2) is taken over training examples $(x_i, y_i) \sim \mathcal{D}_{\mathrm{train}}$ and over the candidate groups and trajectories sampled for each example from the previous proposer. The term $\widehat{\mathcal{K}}_{i,g,l}$ regularizes the token distribution at state $s_{i,g,l}$ against that of the reference model q_{ref}, the proposer frozen at the start of training. The clipping parameter ϵ and the KL coefficient β control the update and the regularization, respectively.

3.2 Human-Seeded Prompt Optimization

DiPPO-HS includes a human-written instruction h in the initial candidate set and trains the proposer with DPO on reward-derived preference pairs (Rafailov et al., 2023). The proposer receives the same masked input as in DiPPO-HF, and h contributes through preference pairs rather than as an input to the proposer. It is a competing candidate, not a target string that the proposer must reproduce.

At each update, the current proposer samples G new candidates $p_1, \ldots, p_G$, and every member of the candidate set $\mathcal{C} = \{p_1, \ldots, p_G\} \cup \{h\}$ is scored on a shared query batch $\mathcal{B} \subset \mathcal{D}_{\mathrm{train}}$ by $R_{\mathcal{B}}(c) = |\mathcal{B}|^{-1} \sum_{(x,y)\in\mathcal{B}} \mu_t(\hat{y}(c, x), y)$ for $c \in \mathcal{C}$. Pairs of candidates in $\mathcal{C}$ whose scores differ are retained, with the higher-scoring candidate designated as the preferred prompt y^w and the lower-scoring one as the rejected prompt y^l. Thus y^w and y^l denote candidate prompts, not target-model responses. The preference set $\mathcal{P}$ contains only these pairs (y^w, y^l) for the current update. It includes both human–generated and generated–generated pairs, so both members of a pair may be generated by the d-LLM (Figure 2c, bottom).

To score a candidate y under the proposer, we mask part of its tokens and compute the mean log-probability of reconstructing the original tokens, conditioned on a query x'. We denote this reconstruction score by $s_\phi(y, x')$ and its counterpart under the reference proposer frozen before training by $s_{\mathrm{ref}}(y, x')$. Both models use identical masks, and their difference is averaged over multiple masks. The reference-relative score $\Delta_\phi(y, x') = s_\phi(y, x') - s_{\mathrm{ref}}(y, x')$ then enters the DPO objective in Eq. (3)

$$\mathcal{L}_{\mathrm{DPO}}(\phi) = -\mathbb{E}_{(y^w,y^l)\sim\mathcal{P}}\,\mathbb{E}_{x'\sim\mathcal{D}_{\mathrm{train}}}\left[\log\sigma\left(\beta_{\mathrm{DPO}}\left[\Delta_\phi(y^w, x') - \Delta_\phi(y^l, x')\right]\right)\right]. \qquad (3)$$

In Eq. (3), the conditioning query x' is drawn from the training queries independently for each pair and is not tied to the batch $\mathcal{B}$ that determined the preference. The function σ is the sigmoid, and β_{DPO} is a positive coefficient. Minimizing Eq. (3) encourages a larger increase in reconstruction score relative to the reference for the preferred prompt than for the rejected prompt, without updating the target model or training a separate reward model.

3.3 Candidate Pool, Selection, and Deployment

DiPPO deploys a prompt selected from an archive rather than the last sample of training. Both GRPO and DPO add generated candidates to $\mathcal{A}$ and merge identical ones. Training rewards are measured on different examples or batches, so they are used only to decide which candidates to evaluate again.

Selection proceeds in two stages. First, the top K candidates by training score are evaluated with the task metric on $\mathcal{D}_{\mathrm{dev}}$, which does not overlap with the test set. Candidates passing audits, such as data-leakage and human-instruction duplication checks, form the eligible set $\mathcal{A}_{\mathrm{elig}}$, and the deployed prompt is chosen by Eq. (4)

$$p^* \in \arg\max_{p\in\mathcal{A}_{\mathrm{elig}}} S_{\mathrm{dev}}(p). \qquad (4)$$

In DiPPO-HS, h stays in the archive and is evaluated on the same development set as the generated candidates. If h is among the top candidates and no eligible candidate scores higher, h is deployed. The selected prompt p^* in Eq. (4) is applied unchanged to every test input.

4 EXPERIMENTS

Benchmarks and evaluation. We use GSM8K and MMLU for mathematical and knowledge reasoning, MBPP for code generation, SQuAD 2.0 and HotPotQA for question answering, CLINC, FEVER and HaluEval for intent classification and factuality judgment, and BFCL and Spider for structured function calling and semantic parsing (Cobbe et al., 2021; Hendrycks et al., 2020; Austin et al., 2021; Rajpurkar et al., 2018; Yang et al., 2018; Larson et al., 2019; Thorne et al., 2018; Li et al., 2023; Patil et al., 2025; Yu et al., 2018). All numbers are percentages. We follow the public evaluation procedure of each task. Where the official scorer requires a canonical output format, we apply the same answer-extraction post-processing to the response of the target model for every method. Each setting is run with five seeds fixed in advance, and we report the result of the best seed in all figures and tables.

All tasks are evaluated on their full official test sets. For SQuAD 2.0 and Spider, whose test labels are not public, we use the official validation and development splits. HaluEval has no official split, so we partition it into training, development and test sets. For CLINC we match the out-of-scope ratio of the training data to that of the official test split. For FEVER we restrict the scope of the prompt to the verdict stage, evaluating claim classification with gold evidence given, and therefore report oracle-evidence label accuracy rather than the official FEVER score. HotPotQA follows the same setting as GEPA. Rather than a single instruction, it uses the composite retrieve-summarize-answer pipeline of the public GEPA artifact, with 300 test examples, and we first verified that the performance reported for GEPA is reproduced in this setup.

Baselines. We compare DiPPO with two basic baselines, no instruction (None in the figures and tables) and the human-written instruction (Human). For BFCL and HaluEval, we use the instruction released by the benchmark authors. For HotPotQA, we follow the DSPy instruction as reproduced in the GEPA setup, and for MBPP, the prompt of the original paper. For GSM8K, MMLU and Spider, we follow the answer format of the dataset release and of prior work. SQuAD 2.0, FEVER and CLINC provide no instruction, so we wrote one following the task definition and labeling guidelines of the dataset paper. Beyond these, we compare prompt-optimization systems from different families. For RLPrompt (Deng et al., 2022), we evaluate both the 82M policy model of the original implementation and a variant whose policy model is scaled to 7B, the same size as our proposer (RLPrompt 82M and RLPrompt 7B in Figure 1). TextGrad (Yuksekgonul et al., 2024) and GEPA (Agrawal et al., 2025) rely on a critic or reflection model, for which we use Qwen2.5-7B-Instruct and Qwen2.5-72B-Instruct (Qwen, 2024) (Qwen2.5-7B and Qwen2.5-72B hereafter). Following the original papers, we also use the proprietary Claude Opus 5 as the critic. All other settings of TextGrad and GEPA follow their official configurations. For GEPA, we keep the official budget of 6,871 rollouts on HotPotQA and set it to 2,000 on the remaining single-module tasks, and we exclude the merge option, which tends to degrade performance. DLM-PO (Wang et al., 2026b) edits the instruction by remasking and refilling spans with a d-LLM, and we reproduce it under the same instruction-length budget (DLM-PO remask in Figure 1). We also compare zero-shot infilling with Dream-v0-Instruct-7B (Ye et al., 2025) (Dream-7B hereafter) and with the LLaDA model that Fujinuma & Sakaguchi (2026) fine-tune for prompt infilling. Following their terminology, we label the two as infill-Dream locked and infill-LLaDA unlocked in Figure 1.

4.1 PROMPT DISCOVERY AND REFINEMENT ACROSS TASKS

In this experiment we use Dream-7B as the prompt proposer and Qwen2.5-7B as the frozen target model. Figure 1 compares DiPPO with the baselines on ten tasks. Error bars give the seed mean and standard deviation.

Human-free prompt generation. In Figure 1, DiPPO-HF improves over the no-instruction baseline on all tasks and records the highest score among the human-free methods, RLPrompt 82M/7B and the training-free infilling baselines, on every task. The gains are largest where the target model cannot

Table 1: Varying the target model with the proposer fixed to Dream-7B. Each row pair compares DiPPO-HF with no instruction (None) and DiPPO-HS with the human instruction (Human).

Task	Method	Qwen2.5 7B	Qwen2.5 72B	Qwen3 30B-A3B	Mistral 24B	Gemma-2 9B	Llama-3.1 8B	Phi-3.5 mini
BFCL	None	36.14	28.06	29.10	29.33	27.60	35.91	23.67
	DiPPO-HF	**59.47**	**77.60**	**74.98**	**66.28**	**71.71**	**64.05**	**37.49**
	Human	72.98	79.10	77.83	70.44	72.17	66.86	45.96
	DiPPO-HS	**78.52**	**80.99**	**80.21**	**80.10**	**75.02**	**73.67**	**54.24**
SQuAD 2.0	None	0.00	0.00	0.07	0.27	0.07	4.89	0.01
	DiPPO-HF	**64.29**	**54.51**	**54.22**	**71.84**	**44.54**	**18.30**	**40.41**
	Human	67.63	73.86	60.61	76.77	43.65	39.19	23.85
	DiPPO-HS	**78.17**	**82.23**	**75.49**	**81.74**	**75.71**	**70.12**	**55.63**
CLINC	None	0.13	0.00	0.40	0.00	0.16	0.55	2.98
	DiPPO-HF	**62.16**	**72.81**	**71.08**	**69.25**	**45.41**	**59.84**	**52.67**
	Human	72.38	84.47	74.82	74.15	78.27	66.51	50.51
	DiPPO-HS	**74.92**	**84.51**	**80.16**	**80.04**	**78.68**	**68.94**	**53.71**
FEVER	None	61.94	74.15	67.49	15.88	76.28	2.08	77.14
	DiPPO-HF	**80.70**	**83.47**	**80.85**	**83.06**	**79.98**	**74.86**	**78.03**
	Human	73.68	83.00	**83.40**	82.11	79.87	75.47	73.23
	DiPPO-HS	**81.97**	**83.93**	82.87	**83.24**	**80.26**	**75.93**	**75.45**

produce a valid output without an instruction. On SQuAD 2.0 and HaluEval, both of which start from zero with no instruction, DiPPO-HF gains 78.0pp and 80.5pp. Relative to the human instruction, DiPPO-HF is higher on six tasks, with the largest margins on HotPotQA (+27.3pp) and HaluEval (+26.5pp). It is within 1pp of the human instruction on GSM8K and MBPP, and about 5pp and 9pp lower on BFCL and CLINC. This shows that a proposer trained from the downstream reward alone, without an initial human instruction or natural-language feedback, can reach or exceed the human instruction on most tasks.

Human-seeded prompt generation. In Figure 1, DiPPO-HS improves on the human instruction on eight of the ten tasks. The largest gains appear on HaluEval (+30.6pp) and HotPotQA (+27.3pp), followed by SQuAD 2.0 (+11.6pp). On the remaining two tasks, GSM8K and MBPP, it stays within 0.5pp of the human instruction. Among the methods that start from the human instruction (TextGrad, DLM-PO and GEPA, including their variants with stronger critics), it records the highest score on five tasks and comes within 1pp of the best on HaluEval. Averaged over all tasks, DiPPO-HS scores 76.75, the highest among all compared methods and above GEPA with the 72B reflection model (76.28) and with Claude Opus 5 (76.10). On HaluEval, however, two of the five seeds do not move beyond the human instruction, a case we return to in Section 4.3. These results show that the human instruction is not a ceiling for DiPPO but an effective starting point from which the proposal distribution can be improved further.

In summary, with a 7B open-weight d-LLM proposer, DiPPO discovers new prompts when no human instruction is available and refines the instruction further when one is provided, learning the proposal distribution directly from a downstream scalar reward without an external proposer or a natural-language critic.

4.2 Varying the Target Model

With Dream-7B as the prompt proposer, we vary the frozen target model across Qwen2.5-7B/72B, Qwen3-30B-A3B, Mistral-Small-24B, Gemma-2-9B, Llama-3.1-8B and Phi-3.5-mini. DiPPO-HF and DiPPO-HS are trained separately using the task reward of each target model. Table 1 reports results for the seven target models alongside the no-instruction and human-instruction baselines.

Human-free prompt generation. DiPPO-HF scores higher than the no-instruction baseline on every target model and task combination we evaluate (Table 1). The margin is especially large on SQuAD 2.0 and CLINC, where producing a valid output without an instruction is difficult, and the improvement is consistent across all target models on BFCL and FEVER as well. On FEVER it also

Table 2: Varying the prompt proposer with the target fixed to Qwen2.5-7B. Each block compares DiPPO-HF with no instruction (None) and DiPPO-HS with the human instruction (Human).

Method	Proposer	GSM8K	MMLU	MBPP	BFCL	SQuAD2.0	CLINC	FEVER	HaluEval	Spider	HotPotQA
None	–	84.76	60.19	60.40	36.14	0.00	0.13	61.94	0.00	65.70	24.67
DiPPO-HF	Dream-7B	91.66	70.41	66.60	62.70	78.04	62.67	81.71	80.53	74.95	61.33
DiPPO-HF	LLaDA-8B	89.23	70.47	63.00	59.01	40.62	61.33	81.10	0.13	69.54	59.33
Human	–	91.81	69.82	67.20	72.98	67.63	72.38	73.68	54.05	69.05	34.00
DiPPO-HS	Dream-7B	91.36	70.29	67.00	80.48	79.36	76.51	82.32	84.62	75.63	61.33
DiPPO-HS	LLaDA-8B	92.27	70.36	68.00	78.75	79.58	75.05	81.51	63.38	75.05	59.60

exceeds the human instruction on five of the seven target models. This shows that a single d-LLM proposer backbone can discover prompts suited to different models by following the scalar reward of each target.

Human-seeded prompt generation. DiPPO-HS scores higher than the human-written instruction on every target model for SQuAD 2.0, CLINC and BFCL (Table 1). On SQuAD 2.0 in particular it raises Gemma-2-9B, Llama-3.1-8B and Phi-3.5-mini from 43.65, 39.19 and 23.85 to 75.71, 70.12 and 55.63. On FEVER it improves over the human instruction on six of the seven target models, falling slightly short on Qwen3-30B-A3B. This shows that, even though the effect of the same human instruction varies across target models, DiPPO-HS can adapt the instruction to the output behavior of each model.

Taken together, DiPPO yields consistent improvements over no instruction or the human-written instruction across model families and scales. This shows that the benefit of training a prompt proposer from the task reward of the target model is not confined to Qwen2.5-7B.

4.3 Varying the Prompt Proposer

With Qwen2.5-7B as the frozen target model, we vary the prompt proposer across Dream-7B and LLaDA-8B-Instruct (Nie et al., 2025) (LLaDA-8B hereafter), training DiPPO-HF and DiPPO-HS separately for each proposer. Table 2 reports the performance of each proposer backbone alongside the no-instruction and human-instruction baselines.

Human-free prompt generation. Under DiPPO-HF, both backbones improve over the no-instruction baseline on every task except HaluEval with LLaDA (Table 2), and they track each other on most tasks. LLaDA falls well behind Dream only on SQuAD 2.0 and HaluEval.

Human-seeded prompt generation. With the human instruction in the candidate set, LLaDA improves on the human instruction on all tasks and Dream on eight (Table 2), and the backbones move even closer. On every task except HaluEval, their scores are close and neither is consistently ahead. On HaluEval, LLaDA reaches 63.38 compared with 84.62 for Dream, because several of its seeds never produce a prompt that outperforms the human instruction.

Proposer-Side Meta-Instruction for HaluEval

The task below is a hallucination-detection benchmark: you are given a knowledge passage, a question, and a candidate answer. Write the instruction an assistant needs in order to perform this benchmark. Output only the instruction.

Figure 3: Meta-instruction added to the proposer input on HaluEval.

Proposer-side meta-instruction. SQuAD 2.0 and HaluEval each impose an output convention. SQuAD 2.0 includes unanswerable questions that require an explicit no-answer output, and HaluEval requires a `Yes`/`No` verdict on whether a candidate answer contains a hallucination. Performance on these tasks depends heavily on whether the proposer finds the convention, and therefore varies widely across seeds. This is why LLaDA performs poorly on both tasks in Table 2. On HaluEval in the human-free setting, every candidate generated by LLaDA scored zero, so training never began.

Table 3: Results with the proposer-side meta-instruction, with the target fixed to Qwen2.5-7B. Each block compares DiPPO-HF with no instruction (None) and DiPPO-HS with the human instruction (Human).

Method	Proposer	GSM8K	MMLU	MBPP	BFCL	SQuAD2.0	CLINC	FEVER	HaluEval	Spider
None	–	84.76	60.19	60.40	36.14	0.00	0.13	61.94	0.00	65.70
DiPPO-HF	Dream-7B	90.98	70.42	67.20	68.71	78.32	62.29	81.30	82.85	72.92
DiPPO-HF	LLaDA-8B	91.51	70.03	64.20	68.71	40.55	60.07	81.72	83.87	71.95
Human	–	91.81	69.82	67.20	72.98	67.63	72.38	73.68	54.05	69.05
DiPPO-HS	Dream-7B	91.96	70.02	69.00	79.10	79.63	74.04	81.41	82.97	73.69
DiPPO-HS	LLaDA-8B	91.13	69.77	68.60	79.79	78.35	74.96	81.11	82.03	74.18

The same tendency appears with other models and also accounts for the large seed variance on these tasks. We therefore add to the proposer input a short meta-instruction stating the role of the text to be generated. For HaluEval, the meta-instruction is the text shown in Figure 3. With this meta-instruction, the proposer finds instructions that elicit a `Yes`/`No` verdict, which it failed to find without it, and human-free training on HaluEval proceeds in every seed for both proposers. The meta-instruction describes only the task and contains no answer format or labeling rule. It is given to the proposer alone and is not passed to the target model. As a result, both proposers reach HaluEval scores close to the best methods in Figure 1 in both settings, and the other tasks remain at a similar level, with human-free BFCL rising for both proposers. Table 3 reports its effect on all tasks.

In summary, proposer training in DiPPO works with both backbones. The two perform comparably in the human-seeded setting, and the human-free gap comes down to obtaining initial candidates that satisfy the output convention, which a short proposer-side instruction alleviates on HaluEval.

5 Conclusion

We introduced DiPPO, a prompt optimizer that trains a masked diffusion language model as the prompt proposer from scalar task rewards, discovering new instructions without an initial human instruction (DiPPO-HF) and refining a given instruction (DiPPO-HS). DiPPO-HF outperformed all methods that do not use a human instruction on every task and exceeded the human instruction on six tasks, while DiPPO-HS improved on the human instruction on eight tasks and achieved the highest score on four. Averaged over all tasks, DiPPO-HS scored highest among all compared methods, on par with or above methods that use stronger critic models. By filling a prompt slot conditioned on the query and updating the proposer itself, DiPPO learns a proposal distribution without an external proposer or a natural-language critic. Our results across seven target models and two d-LLM proposers suggest that learning the proposer from task rewards can offer a practical strategy for prompt optimization with open-weight models. Even on tasks with sparse initial rewards, a one-sentence meta-instruction to the proposer was enough to start human-free training, which suggests that DiPPO can be applied to new tasks for which no effective instruction is yet available.

AI use statement

Generative AI tools were used in two limited capacities in this work. First, they assisted with language editing to improve the grammar, clarity, and phrasing of author-written text. Second, they provided coding assistance during research execution, specifically to help modify and run experimental code and to organize and reformat experimental results. All AI-assisted code and result formatting were reviewed and verified by the authors, and all reported numbers were produced and validated by the authors from recorded artifacts. We did not use generative AI tools for research ideation, for the design of the proposed method, or for generating synthetic data for training or evaluation. The data, methods, and claims are those of the authors, and the authors take full responsibility for the final content of this paper.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning, 2025. URL https://arxiv.org/abs/2507.19457.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models. arXiv:2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Paweł Batorski, Adrian Kosmala, and Paul Swoboda. PRL: Prompts from Reinforcement Learning, 2025. URL https://arxiv.org/abs/2505.14412.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient Training of Language Models to Fill in the Middle, 2022. URL https://arxiv.org/abs/2207.14255.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning, 2022. URL https://arxiv.org/abs/2205.12548.

Yoshinari Fujinuma and Keisuke Sakaguchi. Unlocking Prompt Infilling Capability for Diffusion Language Models, 2026. URL https://arxiv.org/abs/2604.03677.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. EvoPrompt: Connecting LLMs with Evolutionary Algorithms Yields Powerful Prompt Optimizers, 2024. URL https://arxiv.org/abs/2309.08532.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. arXiv:2009.03300, 2020. URL https://arxiv.org/abs/2009.03300.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY4VZ0pFSo.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. arXiv:1909.02027, 2019. URL https://arxiv.org/abs/1909.02027.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. arXiv:2305.11747, 2023. URL https://arxiv.org/abs/2305.11747.

Wenjin Liu, Haoran Luo, Xueyuan Lin, Haoming Liu, Tiesunlong Shen, Jiapu Wang, Rui Mao, and Erik Cambria. Prompt-R1: Collaborative Automatic Prompting Framework via End-to-end Reinforcement Learning, 2025. URL https://arxiv.org/abs/2511.01016.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large Language Diffusion Models, 2025. URL https://arxiv.org/abs/2502.09992.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing Instructions and Demonstrations for Multi-Stage Language Model Programs, 2024. URL https://arxiv.org/abs/2406.11695.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Qwen. Qwen2.5 Technical Report. arXiv:2412.15115, 2024. URL https://arxiv.org/abs/2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model, 2023. URL https://arxiv.org/abs/2305.18290.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. arXiv:1806.03822, 2018. URL https://arxiv.org/abs/1806.03822.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models, 2024. URL https://arxiv.org/abs/2406.07524.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, Tim Rocktäschel, and Roberta Raileanu. Rainbow Teaming: Open-Ended Generation of Diverse Adversarial Prompts, 2024. URL https://arxiv.org/abs/2402.16822.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for Fact Extraction and VERification. arXiv:1803.05355, 2018. URL https://arxiv.org/abs/1803.05355.

Guanghan Wang, Gilad Turok, Yair Schiff, Marianne Arriola, and Volodymyr Kuleshov. d2: Improving Reasoning in Diffusion Language Models via Trajectory Likelihood Estimation, 2026a. URL https://arxiv.org/abs/2509.21474.

Shiyu Wang, Haolin Chen, Liangwei Yang, Jielin Qiu, Rithesh Murthy, Ming Zhu, Zixiang Chen, Silvio Savarese, Caiming Xiong, Shelby Heinecke, and Huan Wang. Prompt Optimization Via Diffusion Language Models, 2026b. URL https://arxiv.org/abs/2602.18449.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers, 2024. URL https://arxiv.org/abs/2309.03409.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. arXiv:1809.09600, 2018. URL https://arxiv.org/abs/1809.09600.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion Large Language Models, 2025. URL https://arxiv.org/abs/2508.15487.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. arXiv:1809.08887, 2018. URL https://arxiv.org/abs/1809.08887.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "Differentiation" via Text, 2024. URL https://arxiv.org/abs/2406.07496.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling Reasoning in Diffusion Large Language Models via Reinforcement Learning, 2025. URL https://arxiv.org/abs/2504.12216.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large Language Models Are Human-Level Prompt Engineers, 2023. URL https://arxiv.org/abs/2211.01910.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-Reduced Preference Optimization for Large Language Diffusion Models, 2025. URL https://arxiv.org/abs/2505.19223.