Unlocking Prompt Infilling Capability for Diffusion Language Models

Anonymous ACL submission

Abstract

Masked diffusion language models possess infilling capabilities, yet current supervised fine-tuning practices use response-only masking which prevents models from learning to infill prompt tokens. We first show that publicly available masked diffusion language models exhibit limited infilling capability for prompts. This limitation stems from a training-inference gap where models never encounter prompt masking during supervised fine-tuning stage. To address this, we introduce a two-stage training framework during supervised fine-tuning (SFT): (1) full-sequence masking to enable prompt infilling, followed by (2) response-only masking to preserve downstream task accuracy. This simple approach enables models to infill optimal prompts from few-shot examples at inference time. Evaluating on math, multi-hop reasoning, and LLM-as-a-Judge, we show that our training framework unlocks prompt infilling capabilities. Our results suggest that the training framework is the primary bottleneck for unlocking prompt infilling capability in masked diffusion language models.

1 Introduction

Prompt engineering has become essential for eliciting desired behaviors from large language models. While autoregressive models generate text unidirectionally, diffusion language models (Lou et al., 2024; Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025) employ bidirectional denoising, iteratively refining text through multiple diffusion steps. This capability suggests a natural question: can diffusion models leverage their bidirectional nature to not only generate completions, but also infer and optimize their own prompts?

Current masked diffusion language models such as Dream (Ye et al., 2025) and LLaDA (Nie et al., 2025) are trained exclusively with response-only masking during supervised fine-tuning: prompts remain clean while only completions are masked and denoised. While these models excel at denoising responses given clean prompts, they lack experience denoising prompts which is the capability needed for prompt infilling. **This is not an architectural limitation but a training limitation**: the bidirectional architecture supports prompt infilling, but response-only training during SFT stage never teaches models to unlock this capability.

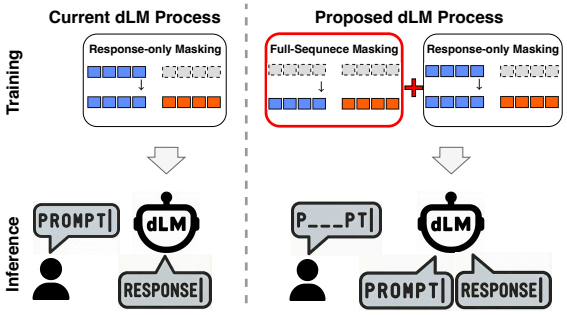

Figure 1: Proposed process for unlocking prompt infilling capability in masked diffusion language models (dLM). **Training:** Add full-sequence masking SFT stage i.e., masking both prompts and responses rather than response-only masking. **Inference:** A dLM both infills partially masked prompts (Appendix H) and generates a response based on the user's incomplete prompt.

We propose addressing this gap through a two-stage training framework (Figure 2). First, we introduce full-sequence masking that masks both prompts and responses during SFT, enabling models to learn bidirectional dependencies across the entire prompt-response sequence. Second, we apply conventional response-only masking to preserve response generation capabilities. This approach enables prompt infilling for diffusion models, where models infer optimal prompts from partial instructions and few-shot examples. Our main contributions are:

- We identify that response-only masking during supervised fine-tuning creates a training-inference gap that prevents diffusion mod-

els from leveraging their prompt infilling capability, and propose a two-stage training framework combining full-sequence masking (Stage 1) followed by response-only masking (Stage 2) to address this limitation.

- We show that our training framework unlocks prompt infilling on multiple datasets, with infilled prompts achieving competitive or superior performance compared to solely relying on manually designed templates.
- We show that prompts inferred from our Stage 1+2 models transfer effectively across different models, which also enables prompt distillation approach.

2 Background & Motivation

To understand why current masked diffusion models cannot fully leverage their prompt infilling capabilities, we first review how diffusion language models operate (§2.1, §2.2), then analyze the training-inference gap that prevents prompt infilling which motivates full-sequence masking for SFT (§2.3).

2.1 Masked Diffusion Language Models

Diffusion models generate text by learning to reverse a noising process (Ho et al., 2020; Song et al., 2021). Consider a sequence $X = (P, R)$ where P represents the prompt and R the response tokens in X. We denote X_t as the sequence at diffusion timestep t, where $t = 0$ represents clean text and increasing t corresponds to higher noise i.e., more number of masked tokens. Starting from clean text $X_0 = (P_0, R_0)$, the forward diffusion process gradually adds noise to create masked versions X_t at t (Austin et al., 2021; Sahoo et al., 2024).

The model learns to estimate $\mathbb{P}(X_0|X_t)$ i.e., the probability of recovering clean text from any masked version. During pre-training, models can learn to denoise arbitrary portions of a sequence:

$$\mathbb{P}(X_0|X_t) = \mathbb{P}(P_0, R_0|P_t, R_t) \quad (1)$$

where both prompts and responses may be masked. This bidirectional capability distinguishes diffusion models from autoregressive approaches and aligns them with masked language models like BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020), which also perform bidirectional reconstruction. However, diffusion models iterate through multiple denoising steps rather than predicting masked tokens in a single step. This requires the model to encounter diverse masking patterns during training, including when only prompts are masked i.e., $X_{t_p} = (P_t, R_0)$, only responses are masked i.e., $X_{t_r} = (P_0, R_t)$, or both are partially masked.

2.2 Masking in Supervised Fine-Tuning

To understand the training-inference gap in current diffusion language models, we now examine the precise masking mechanism during supervised fine-tuning. Given a prompt-response pair $X_0 = (P_0, R_0)$ the SFT stage operates as follows: For each training iteration, a masking ratio t is sampled uniformly from the interval $(0, 1]$ i.e., $t \sim \mathrm{U}(0, 1]$. Given the sampled ratio t, each token R_0^i in the response is independently masked with probability t. This produces a masked response at time t i.e., R_t where, in expectation, a fraction t of tokens are replaced with the mask token. Since $t \sim \mathrm{U}(0, 1]$, each token position is expected to be masked with 50% probability (see Appendix F). Within a single training step, all tokens share the same masking probability t.

2.3 The Training-Inference Gap: A Limitation of Current SFT Practices

Unlike autoregressive models that generate text unidirectionally, diffusion models possess a **infilling** capability: they can denoise and reconstruct any masked portion of a sequence given surrounding context (Sahoo et al., 2024; Donahue et al., 2020). This bidirectionality naturally extends to prompt infilling i.e., when prompt tokens are masked, the model can infill prompts conditioned on desired responses. This capability resembles span corruption in T5 (Raffel et al., 2020) and infilling in encoder-decoder models (Wang et al., 2022), but leverages diffusion's iterative refinement process.

However, current SFT practices used in LLaDA and Dream prevent this capability. Despite the architectural capacity for bidirectional infilling, response-only masking during SFT never let models learn to denoise prompts, creating a training-inference mismatch when prompt infilling is needed at inference time. A model trained with both prompt $\rightarrow$ response and response $\rightarrow$ prompt denoising could refine its own prompts during inference, learning not just how to respond to prompts but also what constitutes an effective prompt.

Prompt-Response Asymmetry Crucially, during standard SFT in models like LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025), only response

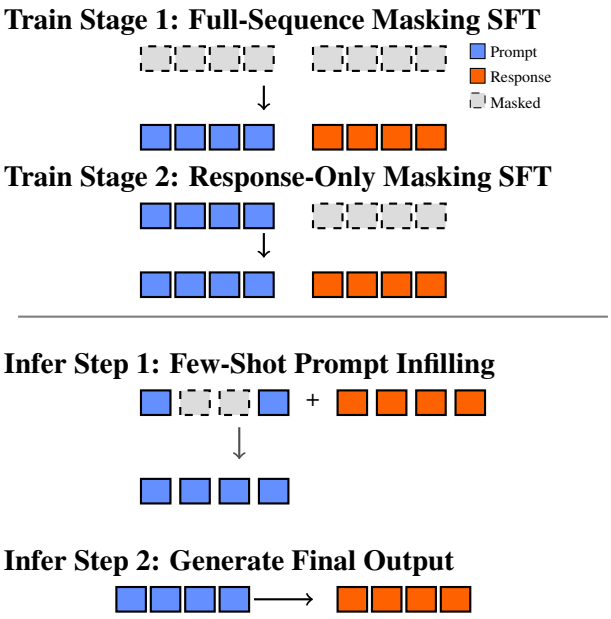

Figure 2: Overview of unlocking self-prompt engineering capability for masked diffusion LMs: (1) Full-sequence masking SFT enables bidirectional learning, (2) response-only SFT specializes for generation, (3) Self-prompt engineering infers missing prompt elements from few-shot examples, (4) Final generation uses the infilled prompt.

tokens are subject to masking. The prompt P_0 remains completely clean: $X_{t_r} = (P_0, R_t)$ where P_0 is never masked. The SFT training objective over timesteps t, prompts P_0, and responses R_0, R_t becomes:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}\left[\frac{1}{|M_t^R|}\sum_{i \in M_t^R} \log \mathbb{P}(X_0^i | X_{t_r}^i)\right] \quad (2)$$

where $M_t^R = \{i : R_t^i = \texttt{[M]}\}$ denotes the set of masked token positions in the response at t.

Implications for Prompt Infilling This asymmetric masking strategy creates a fundamental limitation for prompt infilling capabilities: while the model learns to denoise responses conditioned on clean prompts, it never encounters scenarios where prompt tokens themselves require denoising. Formally, the model learns:

$$\mathbb{P}(R_0 | P_0, R_t) \quad \text{(response denoising)} \quad (3)$$

but never experiences:

$$\mathbb{P}(P_0 | P_t, R_0) \quad \text{(prompt denoising)} \quad (4)$$

This is a training limitation, not an architectural one. The bidirectional denoising architecture theoretically supports $\mathbb{P}(P_0 | P_t, R_0)$ i.e., prompt infilling, but response-only SFT never provides training signals. This training-inference mismatch constrains prompt infilling capability, preventing masked diffusion language models from leveraging their inherent bidirectional capabilities for prompt infilling. To teach models to denoise prompts during SFT stage, we add full-sequence masking to directly addresses this training-inference time gap.

3 Two Stage SFT

We propose a two-stage training framework that directly addresses the training limitation identified above: the lack of prompt infilling exposure during supervised fine-tuning. Our approach enables diffusion language models to perform prompt infilling by learning to denoise both prompts and responses. Stage 1 applies full-sequence masking to develop prompt infilling capabilities that are architecturally possible but never learned under response-only training, while Stage 2 employs conventional response-only masking to preserve strong generation quality.

3.1 Stage 1: Full-Sequence Masking

In contrast to existing masked diffusion LMs that apply masking only to responses during supervised fine-tuning, we introduce full-sequence masking that masks the entire sequence $X = (P, R)$. The model learns:

$$\mathbb{P}(X_0 | X_t) = \mathbb{P}((P_0, R_0) | (P_t, R_t)) \quad (5)$$

where both prompt P and response R tokens may be masked at timestep t.

Masking Strategy For each training example, we sample a masking ratio $t \sim \text{U}(0, 1)$, randomly select $\lfloor t \times |X| \rfloor$ tokens from the sequence to ensure exposure to diverse corruption patterns, including response-only masking $X_{t_r} = (P_0, R_t)$ and full-sequence masking $X_t = (P_t, R_t)$.

Training Objective The training loss for Stage 1 follows the pretraining phase of LLaDA (Nie et al., 2025), applying cross-entropy loss over all masked tokens in the sequence:

$$\mathcal{L}_{\text{full-seq}} = -\mathbb{E}\left[\frac{1}{|M_t|}\sum_{i \in M_t} \log \mathbb{P}(X_0^i | X_t^i)\right] \quad (6)$$

where $M_t = \{i : X_t^i = \texttt{[M]}\}$ is the set of all masked positions. This objective enables the model to unmask arbitrary tokens in both prompts and responses.

Algorithm 1 Few-Shot Prompt Infilling

Require: Few-shot examples $\{(P_0, R_0)\}$, masked prompt template P_t, diffusion steps T, response length L_r
Ensure: Completed prompt P_0

```
Initialize X_T ← P_t, R_0
Set k_t = L_r / T
for t = T, T − 1, ..., 1 do
    M_t ← {i : X_t^i = [M]}
    Predict X̂_0^i ~ P(X_t^i | X_t) for each i ∈ M_t
    Keep k_t predictions
    Update X_{t−1} with kept predictions
end for
return P_0 ← X_0
```

3.2 Stage 2: Response-Only Masking

After full-sequence masking, we apply a second fine-tuning stage using response-only masking. This specializes the model for standard generation while preserving prompt infilling capabilities from Stage 1. Following LLaDA (Nie et al., 2025) and SEDD (Lou et al., 2024), we use cross-entropy loss over masked response tokens using Eq. (2).

3.3 Prompt Infilling at Inference

At inference time, the model performs iterative denoising to generate clean text from masked sequences. For standard generation, given a clean prompt P_0 and fully masked response R_t, the model iteratively denoises (LLaDA processes this block-by-block):

$$R_0 = \arg\max_R \mathbb{P}(R|P_0, R_t) \tag{7}$$

For prompt infilling, we leverage the bidirectional denoising capability learned during Stage 1. Given a response R_0 and a masked prompt template P_t, the model infers the complete prompt:

$$P_0 = \arg\max_P \mathbb{P}(P|P_t, R_0) \tag{8}$$

where the few-shot examples provide context for inferring appropriate prompt components. Algorithm 1 provides the concrete procedural details for this prompt infilling process.

This enables infilling prompts from desired responses without manual prompt engineering. For example, when evaluating with LLM-as-a-Judge on a new dataset with different score scales, the model can infer appropriate scoring rubrics from few-shot examples.

Model	Prompt	#Shot	EM
LLaDA	"Q: ... A: ..."	0	70.8
	+ICL	16	**73.1**
	+Public Infilled	16	64.8
Dream	"Q: ... A: ..."	0	**81.0**
	+ICL	16	79.3
	+Public Infilled	16	54.3

Table 1: GSM8K results on public checkpoints (no fine-tuning). In both models, infilling prompts with the publicly available checkpoints decreases the exact match (EM) score (%).

4 Experiments

4.1 Setup

Models We evaluate two masked diffusion language models: `LLaDA-8B-Instruct` (Nie et al., 2025) and `Dream-v0-Instruct-7B` (Ye et al., 2025). Both models are pre-trained with bidirectional denoising capabilities but employ response-only masking during SFT.

Training Configurations We compare three training approaches: (1) **Stage 2 Only (St.2)**: response-only masking where prompts remain clean and only completions are masked during supervised fine-tuning, following the standard practice in current diffusion language models; (2) **Stage 1 Only (St.1)**: full-sequence masking where both prompts and completions are subject to masking, enabling the model to learn prompt denoising; (3) **Stage 1+2 (St.1+2)**: the two-stage training framework that first applies full-sequence masking (Stage 1) to develop prompt infilling capabilities, followed by response-only masking (Stage 2) to maintain generation quality. We also evaluate publicly available checkpoints (**None**) without additional fine-tuning to establish baseline capabilities.

4.2 Public Checkpoints on GSM8K

We first investigate whether publicly available diffusion models can perform prompt infilling without additional training. This preliminary experiment reveals critical limitations that motivate our full-sequence masking framework. We evaluate three settings on GSM8K (Cobbe et al., 2021): (1) **Base**: zero-shot prompting, (2) **ICL**: in-context learning with 16 examples, and (3) **Infilled**: Infilling the first 256 tokens using few-shot examples.

Table 1 shows that LLaDA achieves 70.8% (base), 73.1% (16-shot ICL), but drops to 64.8% with public checkpoint infilling. Dream achieves

81.0% (base), 79.3% (16-shot ICL), but drops significantly to 54.3% with public checkpoint infilling. These results demonstrate that public checkpoints, despite their strong base performance, lack the capability for effective prompt infilling without additional training.

Public checkpoints exhibit limited prompt infilling capability on datasets even likely seen during training. This motivates our two-stage training framework (§ 5) that enables robust prompt optimization through systematic full-sequence masking.

5 Evaluating Two-Stage Training

Having established the limitations of public checkpoints, we now evaluate our two-stage training approach across diverse tasks spanning different prompt complexities and reasoning patterns: HoVer (multi-hop reasoning) and LLM-as-a-Judge (long structured prompts). We fine-tune models using (1) reponse-only masking (Stage 2 only), (2) full-sequence masking (Stage 1 only), and (3) our two-stage approach (Stage 1+2).

5.1 Multi-Hop Reasoning Task: HoVer

HoVer (Jiang et al., 2020) requires multi-hop claim verification through iterative retrieval and reasoning, aligning naturally with diffusion models' progressive refinement. The task involves three retrieval hops (initial evidence, refined queries, final verification) followed by classification (SUPPORTED/REFUTED/NOT_ENOUGH_INFO). Each hop contains maskable prompt components: query generation instructions, summarization strategies, and evidence aggregation rules.

Table 2 shows that our two-stage training approach (Stage 1+2) achieves the highest label accuracy at 0.604, substantially outperforming the public checkpoint baseline (0.435) by 16.9 percentage points. Stage 1 Only training achieves 0.550 label accuracy, showing that full-sequence masking alone provides significant benefits for prompt infilling in complex reasoning tasks. The progressive improvement from Public $\rightarrow$ Stage 2 Only $\rightarrow$ Stage 1 Only $\rightarrow$ Stage 1+2 validates our hypothesis that full-sequence masking during SFT is essential for unlocking prompt infilling capabilities. Unlike GSM8K's potentially memorized patterns, HoVer's task-specific query generation and evidence aggregation provide a stronger test of genuine prompt infilling capabilities.

Model	Recall	F1	Label Acc
LLaDA (Public)	$.560_{.002}$	$.268_{.001}$	$.435_{.010}$
LLaDA (Stage 1 Only)	$.556_{.001}$	$.279_{.001}$	$.550_{.003}$
LLaDA (Stage 2 Only)	$.533_{.001}$	$.163_{.001}$	$.503_{.013}$
LLaDA (Stage 1+2)	$.547_{.001}$	$.211_{.000}$	$.604_{.014}$

Table 2: HoVer few-shot evaluation results (mean ± std over 3 runs). Public: no additional training.

5.2 LLM-as-a-Judge

LLM-as-a-Judge presents a complementary challenge: long, multi-section prompts with task descriptions, scoring rubrics, reference answers, and format specifications. Unlike GSM8K's short prompts, these structured prompts are unlikely to appear verbatim in pretraining data.

Data We fine-tune on Feedback Collection (Kim et al., 2023) and evaluate on two human annotated datasets: SummEval (Fabbri et al., 2021) and BigGen-Bench (Kim et al., 2025). We compare three prompt types: (1) oracle with all components, (2) masked score ranges/rubrics only, and (3) masked task descriptions and scores. Models infer masked components from 1 shot examples (templates in Appendix H).

We evaluate Dream (Ye et al., 2025) and LLaDA (Nie et al., 2025) with: (1) public checkpoints, (2) response-only masking, and (3) full-sequence masking, then response-only masking.

5.2.1 Results and Analysis

Table 3 shows the LLM judge results. A key observation is that the ability to infer and generate float numbers (e.g., score ranges like 1.0-5.0) is a critical contributor to performance, as these numerical values are essential for proper evaluation score calibration.

Comparing training strategies across LLaDA models reveals a clear progression: the public checkpoint achieves near-zero correlation with Judge Infill (0.087 Spearman), Stage 2 Only improves to 0.477, Stage 1 Only reaches 0.490, and our two-stage approach (Stage 1+2) achieves the strongest performance at 0.535 Spearman correlation. This progression shows that full-sequence masking (Stage 1) is essential for developing prompt infilling capabilities, while the subsequent response-only fine-tuning (Stage 2) preserves generation quality without sacrificing the learned prompt infilling abilities. Notably, Stage 2 Only training shows inconsistent infilling performance across prompt types, with high variance in some set-

Model	Train	Prompt	Pear.	Spear.	Kend.
LLaDA	None	G-eval	$.397_{.000}$	$.289_{.000}$	$.217_{.000}$
		Rand.	$\underline{.465}_{.000}$	$\underline{.439}_{.000}$	$\underline{.339}_{.000}$
		Rand. Infill	$-.005_{.043}$	$-.013_{.065}$	$-.009_{.044}$
		Judge	$.308_{.000}$	$.336_{.000}$	$.261_{.000}$
		Judge Infill	$.050_{.063}$	$.087_{.069}$	$.064_{.053}$
	St.2	G-Eval	$.418_{.000}$	$.482_{.000}$	$.373_{.000}$
		Rand.	$.454_{.002}$	$.491_{.002}$	$\underline{.399}_{.002}$
		Rand. Infill	$.038_{.025}$	$.058_{.048}$	$.042_{.034}$
		Judge	$.484_{.000}$	$\underline{.495}_{.000}$	$.391_{.000}$
		Judge Infill	$\underline{.584}_{.105}$	$.477_{.046}$	$.369_{.031}$
	St.1	G-Eval	$\underline{.453}_{.004}$	$.436_{.004}$	$.346_{.003}$
		Rand.	$.376_{.000}$	$.404_{.000}$	$.322_{.000}$
		Rand. Infill	$.317_{.009}$	$.345_{.013}$	$.275_{.010}$
		Judge	$.447_{.000}$	$\underline{.481}_{.000}$	$\underline{.385}_{.000}$
		Judge Infill	$.320_{.257}$	$.465_{.044}$	$.369_{.035}$
	St.1+2	G-Eval	$.386_{.000}$	$.480_{.000}$	$.380_{.000}$
		Rand.	$.432_{.000}$	$.449_{.000}$	$.359_{.000}$
		Rand. Infill	$.209_{.103}$	$.159_{.106}$	$.128_{.083}$
		Judge	$.500_{.000}$	$.507_{.000}$	$.404_{.000}$
		Judge Infill	$\mathbf{.546}_{.013}$	$\mathbf{.535}_{.007}$	$\mathbf{.423}_{.006}$
Dream	None	G-eval	$.247_{.014}$	$.231_{.030}$	$.184_{.025}$
		Rand.	$\underline{.292}_{.020}$	$\underline{.279}_{.023}$	$\underline{.225}_{.019}$
		Rand. Infill	$.149_{.157}$	$.143_{.172}$	$.112_{.136}$
		Judge	$.192_{.001}$	$.181_{.013}$	$.136_{.008}$
		Judge Infill	$.153_{.069}$	$.160_{.069}$	$.122_{.050}$
	St.2	G-Eval	$\underline{.323}_{.019}$	$\underline{.323}_{.011}$	$.248_{.008}$
		Rand.	$.264_{.006}$	$.261_{.005}$	$.201_{.004}$
		Rand. Infill	$.247_{.264}$	$.200_{.318}$	$.151_{.249}$
		Judge	$.285_{.029}$	$.322_{.013}$	$\underline{.255}_{.011}$
		Judge Infill	$.260_{.050}$	$.267_{.068}$	$.205_{.053}$
	St.1	G-Eval	$.206_{.271}$	$.338_{.055}$	$.264_{.042}$
		Rand.	$.317_{.079}$	$\underline{.361}_{.003}$	$\underline{.277}_{.001}$
		Rand. Infill	$\underline{.344}_{.100}$	$.328_{.104}$	$.258_{.081}$
		Judge	$.244_{.022}$	$.230_{.028}$	$.180_{.022}$
		Judge Infill	$.220_{.023}$	$.216_{.019}$	$.167_{.011}$
	St.1+2	G-Eval	$.312_{.029}$	$.314_{.027}$	$.244_{.022}$
		Rand.	$.326_{.016}$	$.346_{.030}$	$.266_{.024}$
		Rand. Infill	$\mathbf{.461}_{.019}$	$\mathbf{.454}_{.011}$	$\mathbf{.344}_{.008}$
		Judge	$.227_{.011}$	$.222_{.007}$	$.173_{.007}$
		Judge Infill	$.368_{.047}$	$.381_{.028}$	$.300_{.021}$

Table 3: Model performance comparison with different prompting strategies on SummEval dataset (mean ± std. deviation over 3 runs). **Bold** indicates best performance within each model; underline indicates best prompt per (Model, Train) setup.

tings (e.g., Rand. Infill: $0.058_{\pm 0.048}$), suggesting that response-only masking alone provides insufficient training signal for robust prompt infilling.

The effectiveness of different prompt templates varies significantly across training stages. For LLaDA with Stage 1+2 training, the Judge template consistently outperforms both G-Eval and Rand. prompts, achieving 0.535 Spearman correlation with infilling compared to 0.480 (G-Eval) and 0.449 (Rand.). This superiority likely stems from the Judge template's structured format with explicit task descriptions, scoring rubrics, and reference answers i.e., components that align naturally with the bidirectional denoising patterns learned during full-sequence masking. In contrast, Dream models show more varied behavior: while Stage 1+2 Dream achieves strong performance with Rand. Infill (0.454 Spearman), its Judge Infill performance (0.381) lags behind LLaDA, suggesting potential differences in how the two base models internalize structured prompt formats during training. The consistent improvement of infilled prompts over oracle prompts in well-trained models (e.g., LLaDA Stage 1+2: Judge Infill 0.535 vs. Judge oracle 0.507) validates that prompt infilling can discover task-specific optimizations beyond manually designed templates.

BigGen-Bench Evaluation To further validate generalization beyond SummEval dataset, we evaluate on BigGen-Bench (Kim et al., 2025), a more challenging benchmark with 2,780 diverse LLM-as-a-Judge examples. Table 4 shows results using LLaDA models with the infilled judge template.

Train	Prompt	Pearson	Spear.	Kend.
None	Judge Infill	$-.007_{.044}$	$.078_{.058}$	$.068_{.050}$
St. 2	Judge Infill	$.205_{.072}$	$.406_{.091}$	$.168_{.079}$
St. 1	Judge Infill	$\mathbf{.312}_{.013}$	$\mathbf{.548}_{.026}$	$\mathbf{.263}_{.014}$
St. 1+2	Judge Infill	$.259_{.039}$	$.525_{.049}$	$.210_{.043}$

Table 4: BigGen-Bench results with LLaDA with different finetuning approaches (mean ± std over 3 runs). Public Checkpoint shows near-zero correlation, confirming that prompt infilling requires SFT. Stage 1 Only achieves best performance with lowest variance.

The Public Checkpoint achieves near-zero correlation (.078 Spearman, .068 Kendall's tau), showing that prompt infilling capability does not emerge from architectural design alone but requires explicit training with full-sequence masking during SFT.

Interestingly, on BigGen-Bench, Stage 1 Only achieves the strongest performance (.548 Spearman) with lowest variance (±.026), surpassing Stage 1+2 (.525). This contrasts with SummEval where Stage 1+2 was optimal. The superior Stage 1 performance suggests that for this more diverse benchmark, prompt denoising capability is most critical, while Stage 2 may introduce overfitting to specific response patterns. The substantially lower variance of Stage 1 indicates more robust generalization. Stage 2 Only shows performance (.406) significantly better than the public checkpoint but weaker than Stage 1 models, confirming that response-only masking provides insufficient signal for robust prompt infilling.

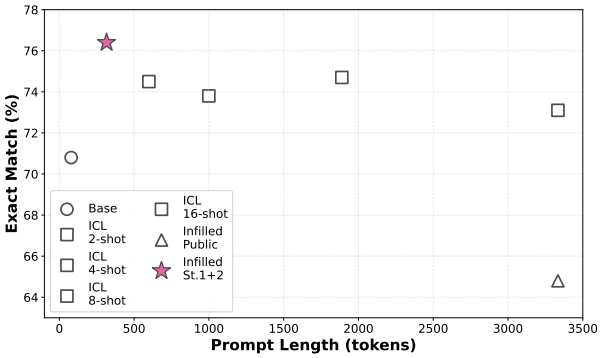

Figure 3: Prompt transfer results on GSM8K using LLaDA public checkpoints (no fine-tuning). Base: Using the prompt template "Q:{question} A:{answer}" from Kojima et al. (2022).

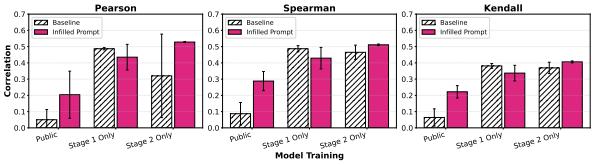

Figure 4: Prompt transfer evaluation on SummEval test set using LLaDA models trained with different configurations (mean ± std over 3 runs). Baseline results from Table 3 showing Judge Infilled prompts results across different training stages. The transferred prompt introduces improvement on Public and Stage 2 only models showing that the infilled prompts transfers across different LLaDA variants.

6 Prompt Transfer Experiments

To evaluate whether infilled prompts generalize across different models, we use the infilled prompt from the strongest model (Stage 1+2) during validation and evaluate with other models.

6.1 Prompt Transfer on GSM8K

Figure 3 shows the effectiveness of prompt transfer on GSM8K. The Stage 1+2 trained model achieves 76.4% exact match accuracy using inferred prompts, outperforming standard 16-shot in-context learning (73.1%) while requiring fewer tokens (315.8 vs. 3333.8). This reduction in prompt length with improved performance validates that our two-stage training framework enables models to distill essential reasoning patterns from few-shot examples into compact, transferable prompts. We observe similar trends on Dream as well (Appendix G). Notably, the public checkpoint's infilling attempt achieves only 64.8%, confirming that effective prompt infilling requires exposure to full-sequence masking on SFT datasets during training rather than relying solely on the model's architectural capabilities.

6.2 Prompt Transfer on LLM-as-a-Judge

Figure 4 presents cross-model prompt transfer results. The extracted prompt achieves Spearman correlation of .535 on its source model (Stage 1+2). When transferred to Stage 2 only, it maintains strong performance at .511 showing that well-optimized prompts preserve effectiveness across architecturally similar models. The Stage 1 only model achieves .429 with the transferred prompt, showing moderate transfer success despite higher variance.

The public checkpoint achieves only .288 ± .059 Spearman correlation, confirming that without proper training (Stage 1+2), even optimized prompts provide limited benefit. These findings suggest that prompts inferred from Stage 1+2 models can effectively transfer to other models.

6.3 Diffusion Perplexity Analysis

To understand why public model perform well in GSM8K but not in SummEval dataset, we evaluate test set perplexity using the masked diffusion language model objective (Sahoo et al., 2024). Let $\sigma(t)$ represents the noise level controlling the mask tokens in a sequence: higher values of σ correspond to more aggressive masking. We denote $\alpha_t = \exp(-\sigma(t))$ as the probability of a token remaining unmasked at time t (i.e., the survival probability), where α'_t represents the time derivative of α_t. Diffusion perplexity (PPL) is defined using the negative evidence lower bound (NELBO) for a sequence $X^{1:L}$ of length L by

$$\text{PPL}(X) = \exp(\mathcal{L}_{\text{NELBO}}/L) \tag{9}$$

NELBO for a sequence $X^{1:L}$ of length L is:

$$\mathcal{L}_{\text{NELBO}} = -\mathbb{E}\int_{t=0}^{t=1} \frac{\alpha'_t}{1-\alpha_t} \sum_{i=1}^{L} \log \mathbb{P}(X_t^i | X_0^i) \mathrm{d}t \tag{10}$$

where $\frac{\alpha'_t}{1-\alpha_t}$ weights each timestep's loss, equivalent to weighted version of Eq. (6). In practice, we compute a Monte Carlo estimate $\widehat{\mathcal{L}}_{\text{NELBO}}$ by sampling timesteps and compute diffusion perplexity PPL as $\exp(\widehat{\mathcal{L}}_{\text{NELBO}}/L)$. To compute $\widehat{\mathcal{L}}_{\text{NELBO}}$, we follow (Sahoo et al., 2024) and sample $t \sim \text{U}(0, 1)$ with the noise level $\sigma(t) = \sigma_{\min} + t(\sigma_{\max} - \sigma_{\min})$ where $\sigma_{\min} = 0.001$ and $\sigma_{\max} = 10$.

Table 5 shows that GSM8K has substantially lower perplexity (6.63) compared to SummEval

Dataset	Test PPL
GSM8K (Q+A)	6.63
SummEval (Judge Template)	46.10

Table 5: Test set diffusion perplexity. Lower perplexity indicates better model fit.

Model	Prompt	Pearson	Spear.	Kendall
LLaDA	St.1+2 Infill	$.546_{.013}$	$.535_{.007}$	$.423_{.006}$
Prometheus	Judge	$.392_{.030}$	$.361_{.021}$	$.291_{.015}$
Prometheus	St.1+2 Infill	$.424_{.015}$	$.402_{.022}$	$.311_{.019}$

Table 6: Prompt transfer study on SummEval dataset using infilled prompt from LLaDA Stage 1+2 checkpoint on `Prometheus-7x8b`.

(46.10), indicating GSM8K dataset is heavily represented in the public checkpoint of LLaDA explaining why the public model performs well on GSM8K compared to SummEval.

7 Related Work

Infillwing With Diffusion Language Models A closely related line of work addresses flexible-length text infilling in discrete diffusion models. DDOT (Zhang et al., 2025) jointly denoises tokens and positions using optimal transport coupling, enabling dynamic segment length adjustment during infilling. While DDOT focuses on architectural improvements for flexible-length generation, our work identifies and addresses a training limitation: **existing masked diffusion models' infilling capabilities are constrained by response-only masking during supervised fine-tuning**. We show that without full-sequence masking during training, models cannot effectively leverage their inherent bidirectional denoising for prompt infilling, regardless of architectural sophistication. Our two-stage training framework addresses this training-inference gap, enabling prompt infilling that complements architectural advances like DDOT.

Automated Prompt Optimization Prompt engineering is crucial for eliciting desired LLM behaviors (Liu et al., 2023), but manual engineering is labor-intensive and requires expertise. This has motivated automated approaches: APE (Zhou et al., 2022) generates prompts through iterative refinement, while OPRO (Yang et al., 2024) treats prompt optimization as a meta-learning problem. Chain-of-thought prompting (Wei et al., 2022) and zero-shot reasoning (Kojima et al., 2022) show that well-crafted prompts significantly improve reasoning performance. A related phenomenon appears in other generative models.

However, existing prompt optimization approaches either generate prompts externally or rely on predetermined structures (chain-of-thought). In contrast, our work enables diffusion models to infer prompts internally from few-shot examples, leveraging their inherent infilling capability rather using larger external models for prompt optimization.

8 Conclusion

We identified a fundamental limitation in current diffusion language models: **response-only masking during supervised fine-tuning severely constrains their infilling capabilities for prompts**, despite their bidirectional architecture theoretically supporting such operations. We show that publicly available diffusion models exhibit limited prompt infilling and not due to architectural constraints, but due to training practices that never expose models to prompt infilling.

To address this training limitation, we proposed a two-stage training framework: full-sequence masking to enable prompt infilling, followed by conventional response masking to preserve generation quality. This simple change to the training paradigm unlocks prompt infilling capabilities that were architecturally possible but never learned. Our results show that the training paradigm, not architectural limitations, is the primary bottleneck for prompt infilling in diffusion models. This finding has broader implications: many capabilities we assume require architectural innovations may simply require exposure to appropriate training signals. Future work should explore scaling these training techniques to larger models, applying prompt infilling to agentic workflows (e.g., auto-compaction of conversation history), and investigating whether similar training-inference gaps exist in other generative modeling tasks.

Limitations

While our two-stage training framework successfully unlocks prompt infilling capabilities in masked diffusion language models, several limitations warrant discussion. First, our experiments focus on two specific masked diffusion models (LLaDA-8B and Dream-7B), which may limit generalizability to other architectures or larger model scales. Second, the computational cost of two-stage training is higher than standard response-only fine-tuning, requiring additional training steps for Stage

1 full-sequence masking. Third, our evaluation focuses primarily on structured tasks (multi-hop reasoning, LLM-as-a-Judge) where prompt components can be clearly delineated and masked; the effectiveness on more open-ended generation tasks remains to be explored. Finally, while we demonstrate prompt transfer across training configurations, the transferability to fundamentally different model families (e.g., autoregressive models) has not been investigated.

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. 2020. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3441–3460, Online. Association for Computational Linguistics.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. *Preprint*, arXiv:2310.08491.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, Se June Joo, Miyoung Ko, Yoonjoo Lee, Hyungjoo Chae, Jamin Shin, Joel Jang, Seonghyeon Ye, Bill Yuchen Lin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2025. The BiGGen bench: A principled benchmark for fine-grained evaluation of language models with language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5877–5919, Albuquerque, New Mexico. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. *Preprint*, arXiv:2310.16834. ICML 2024 Best Paper.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *Preprint*, arXiv:2502.09992.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Preprint*, arXiv:2406.07524.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

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

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. *Preprint*, arXiv:2309.03409.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *Preprint*, arXiv:2508.15487.

Andrew Zhang, Anushka Sivakumar, Chia-Wei Tang, and Chris Thomas. 2025. Flexible-length text infilling for discrete diffusion models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Suzhou, China. Association for Computational Linguistics.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *Preprint*, arXiv:2211.01910.

A Pseudo-Perplexity Analysis

In addition to the diffusion perplexity analysis in Section 6, we also compute pseudo-log-likelihood (Salazar et al., 2020, PLL) on a subset of 100 examples to assess model fit:

$$\mathrm{PLL}(\boldsymbol{X}) := \sum_{t=1}^{|\boldsymbol{X}|} \log \mathbb{P}(x_t | \boldsymbol{X}_{\setminus t}). \tag{11}$$

For a set of sequences $\overline{\boldsymbol{X}}$, we compute pseudo-perplexity as:

$$\mathrm{PPPL}(\overline{\boldsymbol{X}}) := \exp\left(-\frac{1}{N}\sum_{\boldsymbol{X}\in\overline{\boldsymbol{X}}} \mathrm{PLL}(\boldsymbol{X})\right). \tag{12}$$

We mask tokens one by one and compute likelihood following masked language modeling style, since our primary focus is on infilling.

Metric	GSM8K	Feedback Collection
Avg. PLL	-0.033	-0.441
PPPL	1.034	1.554

Table 7: Pseudo-likelihood evaluation on 100 examples. Lower pseudo-perplexity indicates better model fit.

Table 7 reveals GSM8K has lower pseudo-perplexity (1.034) compared to Feedback Collection (1.554), indicating GSM8K dataset is heavily represented in the model. When pseudo-perplexity is low, models recall familiar patterns (LLaDA: 76.4%); when high, this approach fails.

B Implementation Details

We implement our approach on top of LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025), using Qwen 2.5 7B as initial checkpoint.

Training and Inference Details See Appendix C and Appendix D for detailed training hyperparameters and inference configuration, respectively.

C Training Hyperparameters

For full-sequence masking (Stage 1) and conventional response masking (Stage 2), we use the following training hyperparameters:

- Learning rate: 1×10^{-5} (LLaDA) and 2×10^{-6} (Dream) with cosine decay schedule
- Batch size: 32 sequences (effective batch size 256 with 8 gradient accumulation steps)
- Training epochs: 8 epochs for Stage 1, 4 epochs for Stage 2
- Warmup: 500 steps linear warmup for both stages
- Optimizer: AdamW with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
- Weight decay: 0.01
- Gradient clipping: Max norm of 1.0

D Inference Configuration

At inference time, we perform iterative denoising with the following configuration:

- Diffusion steps: $T' = 256$
- Response Length: $L_r = 256$
- Temperature: $\tau = 0.8$ (LLaDA) and $\tau = 0.1$ (Dream)

E Dataset Statistics

We provide detailed statistics for all datasets used in our experiments:

Training Data We fine-tune models on Feedback Collection (Kim et al., 2023), which contains 95,000 train examples and 5,000 dev examples for LLM-as-a-Judge tasks. Each example includes an instruction, response, reference answer, evaluation criteria, and score descriptions. We use the train split for both Stage 1 and Stage 2 fine-tuning.

Evaluation Datasets

- **GSM8K** (Cobbe et al., 2021): Grade school math reasoning dataset with 7,099 train examples, 374 validation examples, and 1,319 test examples.
- **HoVer** (Jiang et al., 2020): Multi-hop fact verification dataset. We use a subset for evaluation with 3 hops and 7 documents per search, averaging results over 3 runs with 16 few-shot examples.
- **SummEval** (Fabbri et al., 2021): Summarization evaluation dataset with human annotations, containing 160 dev examples and 1,440 test examples. We evaluate on the test set containing multiple summarization systems' outputs across different source documents.
- **BigGen-Bench** (Kim et al., 2025): Large-scale benchmark for generative tasks with 2,780 diverse LLM-as-a-Judge evaluation examples. We use this for out-of-domain evaluation to test generalization.

F Expected Masking Probability

For a response of length L', the probability that any specific token position i gets masked is determined by the masking ratio t, which is sampled uniformly from $(0, 1]$. The expected masking probability is:

$$\mathbb{E}_{t\sim \mathrm{U}(0,1]}[t] = \int_0^1 t\, dt = \left[\frac{t^2}{2}\right]_0^1 = 0.5 \qquad (13)$$

Thus, over the course of training, each token position experiences masking with 50% probability on average.

G Additional GSM8K Results

Figure 5 shows the performance-efficiency trade-off for Dream's public checkpoint across different prompting strategies.

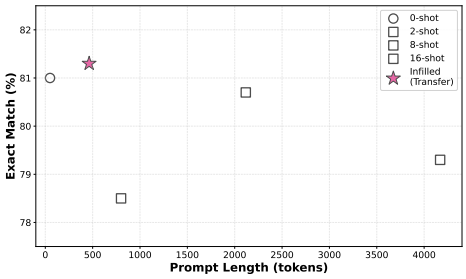

Figure 5: GSM8K exact match rate vs. prompt length for Dream public checkpoint. The infilled prompt (red star) achieves the best exact match accuracy (81.3%) with only 463 tokens on average, outperforming all in-context learning approaches while using significantly fewer tokens than 8-shot (2117 tokens) and 16-shot (4169 tokens) configurations.

H Prompt Templates

H.1 Original Feedback Collection Prompt

Original Feedback Collection Prompt

###Task Description:
An instruction (might include an Input inside it), a response to evaluate, a reference answer that gets a score of 5, and a score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You should refer to the score rubric.
3. The output format should look as follows: "Feedback: (write a feedback for criteria) [RESULT] (an integer number between 1 and 5)"
4. Please do not generate any other opening, closing, and explanations.

###The instruction to evaluate:
{orig_instruction}

###Response to evaluate:
{orig_response}

###Reference Answer (Score 5):
{orig_reference_answer}

###Score Rubrics: [{orig_criteria}]
Score 1: {orig_score1_description}
Score 2: {orig_score2_description}
Score 3: {orig_score3_description}
Score 4: {orig_score4_description}
Score 5: {orig_score5_description}

###Feedback:

H.2 Partially Masked Prompt

Partially Masked Prompt: Task & Score

[Mask] [Mask] [Mask] [Mask] [Mask] [Mask] [Mask] [Mask]...
###The instruction to evaluate:
{orig_instruction}

###Response to evaluate:
{orig_response}

###Reference Answer (Score 5):
{orig_reference_answer}

###Score Rubrics: [{orig_criteria}]
Score [Mask] [Mask]...
Score [Mask] [Mask]...
Score [Mask] [Mask]...
Score [Mask] [Mask]...
Score [Mask] [Mask]...

###Feedback:

H.3 Random Template for SummEval

This is the Random template used for SummEval evaluation:

Base Template for SummEval

Please evaluate the following summary based on the given source text and reference summary.
Source Text:
{source}
Reference Summary:
{reference}
Summary to Evaluate:
{system_output}
Evaluation Criterion:
Overall Quality: The overall quality of the summary considering all aspects including coherence, consistency, fluency, and relevance.
Rate the overall quality of the summary on a scale of 1-5, where 1 is very poor and 5 is excellent.
Provide your evaluation and end with [RESULT] followed by your numerical score (1-5).

Random Infilling Prompt for SummEval

Please evaluate the following summary based on the given source text and reference summary.
Source Text:
{source}
Reference Summary:
{reference}
Summary to Evaluate:
{system_output}
Evaluation Criterion:

[Mask] [Mask]...

I Potential Risks and Prompt Leakage

While our approach enables beneficial applications such as prompt optimization and adaptation, we acknowledge potential risks associated with prompt infilling capabilities. One concern is prompt leakage: malicious actors could potentially use prompt infilling to reverse-engineer proprietary prompts from observed model outputs. If a model can infer optimal prompts from responses, the same capability could theoretically be exploited to extract confidential prompt engineering strategies from commercial systems by observing their outputs. This risk is particularly relevant for systems that rely on prompt secrecy as part of their intellectual property or competitive advantage. Future work should investigate defensive mechanisms such as prompt obfuscation techniques, watermarking strategies, or access control policies to mitigate unauthorized prompt extraction while preserving the benefits of prompt infilling for legitimate use cases.

J Information About Use of AI Assistants

In preparing this manuscript, we used Claude Code, an AI-powered coding assistant developed by Anthropic, for two purposes: (1) revising and editing the manuscript text, including improving clarity, fixing grammatical errors, and refining technical descriptions, and (2) generating and debugging code for running experiments, creating visualization scripts, and processing experimental results.