An Invariant Latent Space Perspective on Language Model Inversion

Wentao Ye[1,2*], Jiaqi Hu[1,2*], Haobo Wang[2,3†], Xinpeng Ti[2,3], Zhiqing Xiao[1], Hao Chen[1], Liyao Li[1], Lei Feng[4], Sai Wu[1], Junbo Zhao[1]

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security, Hangzhou, China
[3]School of Software Technology, Zhejiang University, Ningbo, China
[4]School of Computer Science and Engineering, Southeast University, Nanjing, China
{yewt01, wanghaobo}@zju.edu.cn

Abstract

Language model inversion (LMI), i.e., recovering hidden prompts from outputs, emerges as a concrete threat to user privacy and system security. We recast LMI as reusing the LLM's *own* latent space and propose the *Invariant Latent Space Hypothesis* (ILSH): (1) diverse outputs from the same source prompt should preserve consistent semantics (source invariance), and (2) input↔output cyclic mappings should be self-consistent within a shared latent space (cyclic invariance). Accordingly, we present Inv^{2}A, which treats the LLM as an *invariant decoder* and learns only a lightweight *inverse encoder* that maps outputs to a denoised pseudo-representation. When multiple outputs are available, they are sparsely concatenated at the representation layer to increase information density. Training proceeds in two stages: contrastive alignment (source invariance) and supervised reinforcement (cyclic invariance). An optional training-free neighborhood search can refine local performance. Across 9 datasets covering user and system prompt scenarios, Inv^{2}A outperforms baselines by an average of **4.77% BLEU score** while reducing dependence on large inverse corpora. Our analysis further shows that prevalent defenses provide limited protection, underscoring the need for stronger strategies.

1 Introduction

Large language models (LLMs) have rapidly advanced and now underpin applications across various domains (Wang et al. 2024). Their outputs are routinely created, shared, and acted upon (Leiker et al. 2023; Frisch and Giulianelli 2024; Chen et al. 2024; Wu et al. 2024). Yet, due to the absence of standardized protocols, this growing circulation introduces new security threats (Ti et al. 2025; Wang et al. 2025).

Among these threats, a critical one arises from *language model inversion* (LMI): recovering hidden *prompts* from textual outputs. The prompts are viewed as valuable data assets and fall into two categories: *User prompts* are end-user inputs that may contain private or identifying information; *System prompts* are developer- or application-provided instructions that often embody proprietary system capabilities (Das, Amini, and Wu 2025). In both cases, stakeholders intend to keep prompts confidential.

For each type of hidden prompt, we highlight the corresponding deployment patterns where LMI is a realistic threat. (1) Distributed inference system (user prompt case): This system (Cheng 2025; Moussa et al. 2025; Luo, Yu, and Xiao 2025) allocates the LLM's layers to multiple clients. Each client hosts a subset of consecutive LLM layers and is responsible for calculating the hidden states of these layers and transmitting them to downstream clients. Downstream clients can only observe intermediate hidden states or final output, but not the original user prompt; (2) LLM-powered web service (system prompt case): Any LLM presented via a web interface typically embeds a system prompt. This prompt is hidden in the API message to the underlying LLM.

Recent attempts of LMI have emerged, such as Logit2text (Morris et al. 2024) and Output2prompt (Zhang, Morris, and Shmatikov 2024). Technically, these methods adopt a similar schema: they collect a large number of output-prompt pairs from both prompt space $\mathcal{X}$ and output space $\mathcal{Y}$. Then an external inverse model is trained to fit the inverse mapping $\mathcal{Y} \to \mathcal{X}$. Afterward, the prompts can be recovered via the inverse model. This *brute-force* paradigm (1) relies heavily on large-scale inverse data that are costly or impossible to curate; (2) assumes stable out-of-distribution generalization. In practice, these assumptions are often violated.

Revisiting the LMI problem, we note that the LLM already implements a well-generalized forward mapping $\mathcal{X} \to \mathcal{Y}$ through a rich latent space $\mathcal{Z}$. If this latent geometry can be *reused* for $\mathcal{Y} \to \mathcal{X}$, inversion could be achieved far more data-efficiently. In that case, $\mathcal{Z}$ should actually satisfy a principled hypothesis, the **Invariant Latent Space Hypothesis** (ILSH). ILSH guides two key properties, namely:

- **Source Invariance**: Given a prompt x, the LLM produces different outputs y due to sampling diversity. They share similar semantics. For inversion, $\mathcal{Z}$ should preserve consistent semantic information from both x and y.
- **Cyclic Invariance**: Suppose the LLM supports $\mathcal{X} \to \mathcal{Z} \to \mathcal{Y}$, and the inverse model supports $\mathcal{Y} \to \mathcal{X}$. If the two share a consistent structure, a cycle forms: $\mathcal{X} \to \mathcal{Z} \to \mathcal{Y} \to \mathcal{Z} \to \mathcal{X}$. In this cycle, $\mathcal{Z}$ must support the inverse mapping $\mathcal{Y} \to \mathcal{Z} \to \mathcal{X}$.

*These authors contributed equally.
†Corresponding authors.

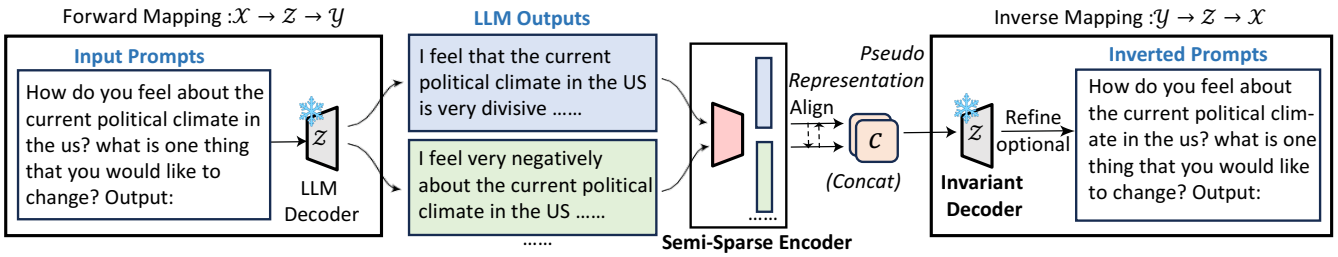


Figure 1: Overview of Inv^{2}A. An inverse encoder maps one or more outputs into denoised pseudo-representations in the LLM's latent space, and the LLM is reused to recover the prompt. The threat model covers both user prompt and system prompt.

Guided by ILSH, we introduce the ***Invariant Inverse Attacker*** (Inv^{2}A), an end-to-end framework (Figure 1) that reuses the raw LLM as an *invariant decoder* and learns an *inverse encoder*. Compared with naive round-trip decoding that feeds outputs directly into the LLM to recover the prompt, Inv^{2}A is asymmetric: outputs are first encoded by the inverse encoder into a denoised pseudo-representation, which is then decoded by the LLM decoder into prompts. When multiple outputs are available, all outputs are concatenated at the pseudo-representation layer to maximize information density. Meanwhile, a semi-sparse encoding mechanism restricts attention to within each output rather than across outputs, thereby making the encoding time complexity linear in the number of outputs. Training follows two phases aligned with the two invariances: (1) alignment—contrastive learning to enforce representation consistency across outputs from the same source prompt; and (2) reinforcement—supervised learning on inverse pairs to explicitly strengthen cyclic invariance. An optional training-free post-processing module iteratively expands and searches the neighborhood of the raw outputs to further improve inversion performance.

Last but not least, we evaluate Inv^{2}A on 9 datasets spanning both the user prompt and system prompt scenarios. Compared with baselines that use external inverse models, Inv^{2}A achieves an average absolute improvement of 4.77% in BLEU while reducing data requirements by 80% to reach comparable performance. Moreover, our study reveals that existing defenses provide limited protection, underscoring the need for stronger mechanisms.

2 Preliminaries

2.1 Problem Statement

In LMI, an adversary aims to *recover the hidden prompt solely from information observable at inference time*. Primary forms of such information include the next-token probability distribution (Morris et al. 2024) and the textual outputs (Zhang, Morris, and Shmatikov 2024; Gao et al. 2024). This paper adopts the latter, directly inverting the outputs.

Threat Model. Our threat model covers realistic scenarios for two attack targets: *user prompt* and *system prompt*.

- (*User*) Distributed inference system: The final client in the inference pipeline acts as a malicious adversary, who sees a *single output* generated from the hidden prompt.
- (*System*) LLM-powered web service: The user becomes an adversary by appending queries to the system prompt, calling the LLM service, and collecting *multiple outputs*.

Based on the harvested outputs, the adversary then attempts to recover the hidden user or system prompt.

Besides, we assume that the adversary has *white-box access* to the LLM. This assumption is realistic: (1) In distributed inference systems, open-source LLMs (white-box) predominate, because deploying a proprietary LLM necessarily reveals its weights to every participating client. Such setting applies to various LLMs, e.g., LLaMA (Dubey et al. 2024), DeepSeek (Liu et al. 2024a), and Qwen (Yang et al. 2025); (2) Likewise, in the web-service ecosystem, several leading LLM vendors (e.g., Qwen and DeepSeek) publicly release their model weights while simultaneously offering public web service. Thus, an adversary can collect the outputs from the web service and simultaneously exercise white-box access to the underlying LLM.

Formalization. View an LLM (named forward LLM) as a stochastic function f that, given a prompt x, induces a conditional distribution over outputs (Radford et al. 2019; Vaswani 2017). Let $y \sim f(x)$ denote a single output sampled from this distribution. In the user prompt case, only one y is observable, while in the system prompt case, multiple outputs $y_1, y_2, \ldots, y_N$ are obtained via repeated interaction with the LLM. For notational uniformity, rewrite the observable outputs as a set $Y = \{y_i\}_{i=1}^{N}$, where $N = 1$ in user prompt inversion. The objective of the adversary is to learn an inverse model f^{-1}. Then, f^{-1} generates $\hat{x} = f^{-1}(Y)$, such that $\hat{x}$ ideally matches x with high fidelity.

2.2 Cyclic Invariance of LLM Latent Space

The core premise of ILSH is cyclic invariance. This property implies that if the latent space $\mathcal{Z}$ of an LLM supports a forward mapping $\mathcal{X} \to \mathcal{Z} \to \mathcal{Y}$, an inverse mapping $\mathcal{Y} \to \mathcal{Z} \to \mathcal{X}$ *also exists* within the invariant $\mathcal{Z}$. Now, we empirically verify its sufficiency and necessity.

Sufficiency. We want to verify the existence of the inverse mapping. Assume that $x \mapsto y$ is a concrete instance of the forward mapping, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. The intermediate variable in $\mathcal{Z}$ is omitted, as it does not require explicit analysis. The inverse mapping $y \mapsto x$ can be evaluated at three levels: (1) Distributional uncertainty, measured by the entropy (Ash 2012); (2) Predictive confidence, mea-

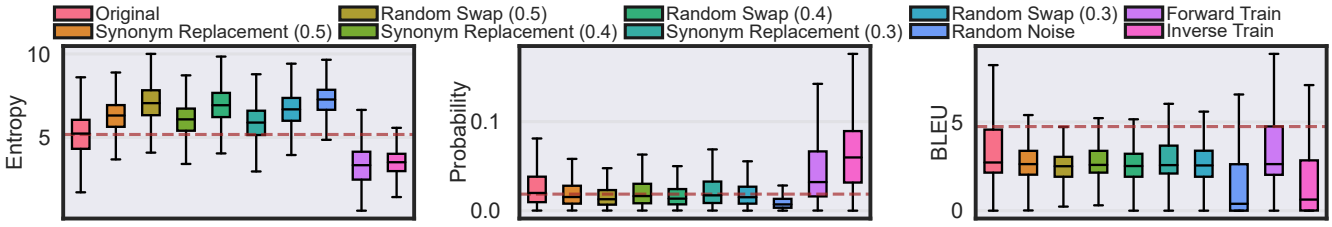


Figure 2: Evaluation of cyclic invariance. Synonym Replacement, Random Swap (randomly swapping words within a sentence), and Random Noise (replacing words with random WordNet entries) represent different perturbation types. Numbers in parentheses indicate the proportion of perturbed words. The brown dashed line marks the mean under the original setting.

sured by the conditional probability $P(x \mid y)$; (3) Round-trip fidelity— given the overall mapping pipeline $x \mapsto y \mapsto \hat{x}$, measure the BLEU score (Papineni et al. 2002) between $\hat{x}$ and x. For (1), rewrite x as the token sequence $x_{[1:T_x]}$, where T_x is the length. Then, the token-wise entropy $H(x \mid y)$ is:

$$H(x \mid y) = \frac{1}{T_x} \sum_{t=1}^{T_x} \mathbb{E}_{x_{[t]}} \left[-\log_2 P(x_{[t]} | y, x_{[<t]}) \right] \quad (1)$$

where $x_{[t]}$ denotes the t-th token, with values spanning the vocabulary. Given 2,000 random prompts from the Alpaca dataset (Taori et al. 2023), we sample the corresponding y with GPT-2 (Radford et al. 2019) and observe these metrics. If the inverse mapping does not exist, the metrics shall be roughly stable (high uncertainty, low confidence, and low fidelity) when y is perturbed or the forward mapping is enhanced. However, Figure 2 shows that once y deviates from the output distribution due to perturbations, all metrics drift sharply. Meanwhile, when we train the LLM to enhance the LLM's confidence in the forward mapping, the metrics of the inverse mapping rise in tandem. This shows that the inverse mapping stems from its coupling with the forward mapping rather than from inverse samples seen during pretraining. Together, two experiments indicate that the LLM implicitly contains an inverse mapping.

Necessity. From the necessity perspective, we conduct a reverse validation: strengthening the inverse mapping synchronously benefits the forward mapping. To this end, we use y as the prompt and x as the label, where x and y are inherited from the sufficiency experiment, and perform SFT on GPT-2. The results in Figure 2 show from a distributional perspective, the model's forward mapping is strengthened, although BLEU decreases. The latter may result from a substantial mismatch between the inverse-training distribution and natural data, leading the model to overfit this unnatural mode and thereby weakening forward activation. Yet, the trend of probabilities still indicates that the forward mapping likewise depends on the inverse mapping.

These findings reveal that the inverse mapping already lurks in the latent space. However, as discussed in § 3, *realizing* high-fidelity mapping demands extra denoising.

3 Method: Invariant Inverse Attacker

The key intuition behind Inv^{2}A is to reuse the same LLM f as its inverse, i.e., set $f^{-1} = f$. A straightforward realization is to obtain $\hat{x}$ by symmetrically feeding Y back into f, to be specific, $\hat{x} = f(Y)$. Unfortunately, such naive round-trip decoding achieves only a BLEU score of 4.75 (Figure 2), far below what is required to cause practical threats. We revisit this failure through ILSH: (1) *Implicit source invariance*: The randomness of Y introduces adversarial noise into the inverse mapping, and f lacks explicit structure to suppress it. (2) *Weak cyclic invariance*: Lacking explicit supervision, the inverse mapping is weakly modeled in low-density regions of the latent space; this property acts as noise and interferes with directly decoding to x. To overcome these obstacles, Inv^{2}A adopts an asymmetric round-trip decoding strategy. We first encode Y into an intermediate pseudo-representation $\mathbf{c}$, rather than feeding it directly into f. Here, $\mathbf{c}$ serves as a "clean anchor ", which mitigates noise and thereby strengthens both invariances. Then, $\hat{x} = f(\mathbf{c})$ can preserve high fidelity with respect to x.

3.1 Architecture

The Inv^{2}A model utilizes an encoder–decoder architecture, including: (1) a trainable *inverse encoder* to encode Y into $\mathbf{c}$; (2) a frozen *invariant decoder* to decode $\mathbf{c}$ into $\hat{x}$. The invariant decoder reuses the forward LLM f without any extra modules. Note that the embedding layer of f is excluded because the decoder's input is the vector $\mathbf{c}$ rather than text. The inverse encoder starts with a pretrained encoder Enc, followed by a linear projection layer Proj. Enc takes Y as input to generate the hidden state $\mathbf{h}$. In this paper, the architecture and initial parameters of Enc are inherited from the T5 encoder (Raffel et al. 2020). Proj further projects $\mathbf{h}$ into the latent space of f to obtain $\mathbf{c}$, i.e., $\mathbf{c} = \mathrm{Proj}(\mathbf{h})$.

Semi-Sparse Encoder. In the user prompt scenario, $Y = y_1$ is a singleton set. $\mathbf{h}$ can be computed by $\mathrm{Enc}(y_1)$. However, in the system prompt scenario, the information from multiple outputs can be jointly utilized. The simplest yet effective approach is to feed the concatenation of all outputs at once, in which case $\mathbf{h} = \mathrm{Enc}(y_1 \oplus \cdots \oplus y_N)$. Yet, this is unfriendly to a Transformer-based encoder, which will compute cross-attention between all tokens. The time complexity of Enc reaches $O(N^2 l^2)$, where l is the token length of a single output. Computational efficiency degrades rapidly as N increases, resulting in only a small subset of Y being usable. We attempt to optimize the complexity. According to Zhang, Morris, and Shmatikov (2024), for inversion tasks, the cross-attention between different $y_i \in Y$ brings little performance gain (Zhang, Morris, and Shmatikov 2024).

Therefore, a *semi-sparse* encoding mechanism is adopted, which focuses only on the cross-attention within each y_i: $\mathbf{h} = \mathrm{Enc}(y_1) \oplus \cdots \oplus \mathrm{Enc}(y_N)$. In this way, the time complexity of the encoding stage is reduced to $O(Nl^2)$.

3.2 Training

Inv^{2}A is trained in two phases—*alignment* and *reinforcement*—corresponding to two types of invariance.

Alignment. This phase aims to enhance *source invariance*. To counter the noise introduced by the randomness of Y, the pseudo-representation $\mathbf{c}$ needs to remain as consistent as possible across Y from the same source x. Otherwise, the recovered result $f(\mathbf{c})$ may drift across Y sharing the same source. To this end, we design source-aware contrastive learning. For a source prompt x, we first sample a separate output set $\mathcal{D}^x$ using temperature sampling (Ackley, Hinton, and Sejnowski 1985). Each element in $\mathcal{D}^x$, denoted as y^+, is treated as an adjacent positive sample. This process is repeated across different x. Then, ENC (Proj is not included) is trained to align the representations of y^+ within the same $\mathcal{D}^x$ while ensuring sufficient separation from negative samples belonging to other sets. The training is guided by the InfoNCE loss (Oord, Li, and Vinyals 2018):

$$\mathcal{L}_N = \mathbb{E}_x \left[-\frac{1}{|\mathcal{D}^x|} \sum_{y^+ \in \mathcal{D}^x} \log \frac{e^{\mathrm{sim}(y,y^+)/\tau}}{\sum_{y' \in \mathcal{U}} e^{\mathrm{sim}(y,y')/\tau}} \right] \quad (2)$$

where τ is the temperature coefficient, sim denotes the inner product of embeddings, and $\mathcal{U}$ denotes the union of all $\mathcal{D}^x$.

Reinforcement. This phase aims to enhance *cyclic invariance*. Specifically, we compensate by introducing explicit supervised learning for the inverse mapping. Based on collected (Y, x) pairs, we minimize the loss between $\hat{x}$ and x. The trainable modules are a two-level structure comprising Enc and Proj. Inspired by multimodal learning (Liu et al. 2024b), we adopt a two-stage training procedure:

1. *Local warm-up*. We seek to balance the magnitude variance between the randomly initialized Proj and other pre-trained modules. Proj is warmed up on a held-back subset with 20% training data, while keeping Enc frozen.
2. *Joint fine-tuning*. Next, we jointly fine-tune both modules with 80% of the remaining data.

3.3 Post-Refinement (Optional)

After training, certain failure cases remain. For example, the recovered $\hat{x}$ is semantically similar but does not exactly match x. We hypothesize that some cases arise from bias in Y itself, rather than insufficient learning of the corresponding $\mathbf{c}$. Therefore, we introduce a training-free post-processing module, called *dynamic filter*. It searches the neighborhood space $\mathcal{D}^y$ of $y \in Y$ to identify an optimal variant y^*, which is least biased. We begin by fully applying f to expand neighbors via prompting, thereby constructing $\mathcal{D}^y$. The utilized prompt template is "*Rewrite the following sentence while keeping the same semantics:* [*Output*]". If the prompt recovered from $\tilde{y}$ enables more accurate reconstruction of y, $\tilde{y}$ carries less bias. y^* is selected as:

$$y^* = \arg\max_{\tilde{y} \in \mathcal{D}^y} P(y \mid f^{-1}(\tilde{y})) \quad (3)$$

Iterative Monte Carlo Search. To explore the neighborhood of y as thoroughly as possible, the filter extends a single-round search into an iterative Monte Carlo process. In each iteration, it retains multiple candidates y^* to seed new neighbors, thereby broadening the search space and increasing the likelihood of capturing the optimal y^*.

To stay efficient, the filter is triggered only when P falls below a threshold τ. Notably, given the external computational load, Inv^{2}A only integrates the filter in §4.3.

4 Experiments

4.1 Experimental Setup

Datasets. In the user prompt scenario, we select 8 datasets: Alpaca (Taori et al. 2023), Dolly (Conover et al. 2023), GPTeacher (teknium1 2025), LaMini (Wu et al. 2023), Self-Instruct (Wang et al. 2022), Evolcode (Luo et al. 2023), WizardLM (Xu et al. 2023), and ArXiv Math (Hebbar 2024). They contain instructions that can be regarded as user prompts. For each dataset, we randomly select 15.5K prompts, where 15K are for training and 500 for testing. We sample 4 outputs per prompt at a temperature of 1.5. All reported results in this scenario are the average of 8 datasets.

In the system prompt scenario, we use the Synthetic GPTs (Zhang, Morris, and Shmatikov 2024) dataset. This dataset prompts GPT-3.5 to generate system prompts based on real GPTs' descriptions. We randomly select 15K prompts for training, and non-overlapping 500 for testing. For each prompt, we append 8 queries and sample one corresponding output. These queries are sourced from the dataset.

Baselines. We compare six open-source baselines , including prompting the forward LLM with jailbreak strings and using an external inverse model. (1) *Jailbreak* strings. These are human-written sequences designed to elicit prompt leakage. We aggregate 12 variant strings. When reporting results, we provide the mean performance across all variants, along with an oracle figure indicating the best-performing variant on the test set selected after evaluation. (2) External model: We consider three baselines—*Logit2text* (Morris et al. 2024) and *Output2prompt* (Zhang, Morris, and Shmatikov 2024), which train T5 to map to the prompt from next-token probabilities and from textual outputs, respectively; and *Few-shot*, which prompts GPT-3.5 or GPT-4o with 4 output–prompt demonstrations.

Implements. `LLaMA2-7B-Chat` (Touvron et al. 2023) is the primary forward LLM in this work. The maximum sequence length of outputs is set to 256 tokens, whereas the prompt length is unconstrained. In both scenarios, `T5-base` encoder is set as the backbone of the inverse encoder. We train 4 epochs for alignment and 1 epoch for reinforcement. The Adam optimizer (Kingma 2014) is utilized with a constant learning rate of $1e{-}5$ for alignment and $2e{-}4$ for reinforcement. Float32 is set as the global precision. We train each model on 8 A800 GPUs with 80G memory (3 hours per model). After training, we apply a greedy decoding strategy to sample the recovered prompt.

For evaluation, we report the fidelity between the target prompts and the recovered prompts using multiple metrics: token-level F1 (*TokenF1*), sentence-level *BLEU* score

Method	**User Prompt** (Average of 8 Datasets)					**System Prompt** (Synthetic GPTs)				
	BLEU	Token F1	CS	GPT	Exact	BLEU	Token F1	CS	GPT	Exact
Logit2text	21.47	51.66	49.83	13.39	0.05	9.58	41.82	67.21	21.60	0.00
Output2prompt	35.34	60.20	77.05	59.46	3.99	21.25	49.91	91.41	79.20	0.00
Few-shot (3.5)	15.51	37.98	58.33	50.41	0.75	7.77	32.54	63.01	43.40	0.00
Few-shot (4o)	26.75	51.66	75.34	65.39	4.28	11.00	41.97	82.61	72.80	0.00
$\text{Jailbreak}_{\text{mean}}$	6.10	23.67	45.62	8.20	0.15	3.33	23.79	43.60	27.90	0.00
$\text{Jailbreak}_{\text{oracle}}$	12.16	34.55	60.71	16.84	0.91	4.64	26.69	49.90	41.60	0.00
Inv^{2}A (Ours)	**41.78**	**65.89**	**82.11**	**74.46**	**10.43**	**24.34**	**53.26**	**92.78**	**94.20**	**0.40**

Table 1: Main results in the user prompt and system prompt scenario, where `LLaMA2-7B-Chat` serve as the forward LLM.

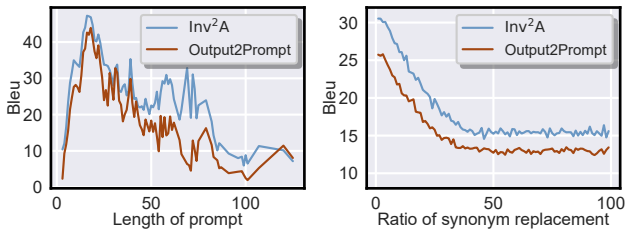
(Papineni et al. 2002), embedding-level cosine similarity (*CS*), and exact match (*Exact*). *CS* is computed using `text-embedding-3-small` (Neelakantan et al. 2022). Additionally, we use a "LLM eval" score (*GPT*) by prompting GPT-4o to approximate human judgment.

4.2 Inv^{2}A as Effective Attackers

Table 1 shows the main experimental results under two scenarios (user prompts and system prompts). The results by dataset are detailed in the Appendix. We find that Inv^{2}A achieves ***SOTA performance across all metrics and scenarios***, with an average BLEU improvement of **4.77%**. Even Output2prompt, the optimal baseline that explicitly learns the inverse mapping, performs worse than our model. This failure indicates that Inv^{2}A activates a stronger inverse mapping from the invariant latent space than direct learning.

Main side findings: (1) *Performance for all methods (including Inv^{2}A) varies substantially across datasets*. This stems from differences in inherent dataset complexity. Datasets with poorer performance typically contain harder cases. For example, on the Self-Instruct and WizardLM datasets (user prompt scenario), beyond the user's question, the answer logic or output format is often additionally constrained. Besides, in the system prompt scenario, the prompts to be recovered typically impose more multifaceted requirements, including but not limited to the model's role, tone, and knowledge scope. As a result, these prompts become long and complex, which causes inversion performance to degrade. (2) *Inv^{2}A excels at semantic-level recovery*. Inv^{2}A achieves higher absolute scores on CS and GPT, two metrics that emphasize semantic similarity. On metrics such as BLEU and Exact that stress surface-level exactness, it is relatively weaker, though still substantially outperforming the baselines. This indicates that Inv^{2}A already attains high semantic-fidelity reconstruction, but fine-grained recovery of certain tokens remains improvable. Given the similar performance trends across two scenarios, subsequent analysis experiments use only the user prompt scenario.

Robustness. We study the robustness of Inv^{2}A from both the prompt side and the output side. (1) For the prompt side, Figure 6a shows the results under different prompt lengths. Inv^{2}A almost always outperforms the best-performing baseline, Output2prompt. As prompt length increases, the BLEU score exhibits an overall downward trend. When the length reaches around 120 words, Output2prompt occasionally outperforms our model. We attribute this phenomenon to random fluctuations because only 0.4% of the prompts fall within this range. (2) For the output side, we consider two types of perturbations to the outputs: synonym replacement (SR) (Wei and Zou 2019) to perturb the original outputs, and temperature sampling to increase diversity. The former simulates noise introduced into the observable outputs by system or human modifications. The latter corresponds to a common defense strategy (Morris et al. 2024) against LMI. The results are shown in Figure 6b and Table 2, respectively. Inv^{2}A can stably maintain its performance gains over the baseline regardless of how the outputs are perturbed. However, under extreme conditions where the outputs differ drastically from the original form (e.g., temperature over 2.0), the loss in absolute performance is very severe.



(a) Prompt Length (b) Output Perturbation (SR)

Figure 3: Robustness against prompt length and synonyms.

τ	Method	BLEU	F1	CS	GPT	Exact
0.5	O2p	29.59	53.23	72.66	44.35	5.80
	Ours	**35.42**	**60.31**	**78.82**	**65.75**	**6.90**
1.0	O2p	29.08	52.57	71.99	43.70	5.90
	Ours	**34.84**	**59.85**	**78.28**	**66.65**	**6.95**
2.0	O2p	24.47	48.36	68.96	35.95	3.95
	Ours	**30.10**	**55.29**	**74.99**	**57.95**	**4.55**

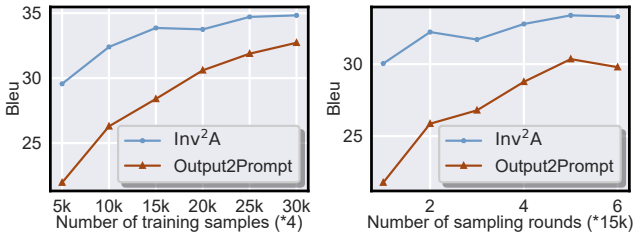
Table 2: Robustness against varying temperatures, where τ denotes temperature and "O2p" is Output2prompt baseline.

Transferability. Table 3 shows results across domains and models. The Anthropic HH (Bai et al. 2022) is introduced as an out-of-domain dataset where prompts differ significantly from the training distribution. Meanwhile, the for-

Model	Method	In-domain (Average of 8 Datasets) BLEU	Token F1	CS	Exact	Out-of-domain (Anthropic HH) BLEU	Token F1	CS	Exact
QWen2-7B	Output2prompt	17.70	46.07	64.52	0.00	5.99	24.21	46.98	0.45
	Few-shot (4o)	15.26	44.39	61.54	0.00	10.11	29.36	56.23	**1.10**
	Inv^{2}A (Ours)	**27.02**	**54.39**	**78.32**	**0.90**	**11.46**	**35.90**	**61.58**	0.80
LLaMA3.2-3B	Output2prompt	27.95	53.39	74.88	3.55	8.32	26.11	45.75	0.40
	Few-shot (4o)	20.71	48.19	74.71	1.60	19.15	44.28	68.78	**2.45**
	Inv^{2}A (Ours)	**29.85**	**54.26**	**76.02**	**3.75**	**20.34**	**46.54**	**75.86**	1.25
LLaMA3-8B	Output2prompt	26.64	50.05	71.59	0.00	9.68	30.41	56.79	0.60
	Few-shot (4o)	19.36	44.81	70.03	0.00	13.62	36.55	65.06	1.35
	Inv^{2}A (Ours)	**29.43**	**55.41**	**78.54**	**1.70**	**16.37**	**42.03**	**68.86**	**1.85**
LLaMA2-13B	Output2prompt	34.98	59.37	79.72	4.30	12.38	34.32	55.96	0.75
	Few-shot (4o)	29.45	56.76	80.34	2.20	29.61	56.68	**82.76**	6.15
	Inv^{2}A (Ours)	**40.42**	**64.46**	**83.40**	**6.75**	**30.41**	**57.92**	82.55	**6.30**

Table 3: Transferability across domains and models. For In-Domain, we report the average of 8 datasets in user prompt scenario.

ward LLMs are replaced with varying sizes and types, including LLaMA3 (Dubey et al. 2024) and Qwen2 (Yang et al. 2024). We directly integrate the encoder trained under the user prompt scenario into different forward LLMs without additional fine-tuning. We observe that the inversion performance of Inv^{2}A is reasonably well transferred. Across different models, methods with explicit training (Inv^{2}A and Output2prompt) generally perform better. This indicates that different LLMs, especially those within the same family (e.g., LLaMA), exhibit some similarity in their responses to the same prompts; therefore, the learned inverse mapping can remain stable. As for the out-of-domain setting, utilizing the forward LLM itself or GPT-4o as the inverse model yields clear gains over T5. This suggests that when the output pattern diverges markedly from the training distribution, a well-generalized backbone is indispensable for inversion.



(a) Prompts Scale (b) Outputs Sampling Rounds

Figure 4: Results under varying training data scales.

Efficiency. The core of Inv^{2}A is to reduce dependence on large-scale inverse data. We quantify this dependence by evaluating dynamic performance under varying training data scales. The training data scale is modified in two ways: directly changing the number of prompts, and fixing the prompts while varying the number of sampling rounds per prompt. As shown in Figure 4, Inv^{2}A reaches comparable performance ***with only 20 to 30% of the data***. This indicates that the forward LLM has learned sufficient information of inversion from its own forward training process. A new inverse model, by contrast, requires much more data to learn information of comparable density, because its training distribution has a gap from the target LLM's output distribution. Beyond data, Inv^{2}A also requires fewer trainable parameters. It includes only the T5 encoder, whereas the baselines mostly train a full T5, including the decoder.

4.3 Ablation Study

Table 4 shows ablation results in the user prompt scenario.

Structure-Wise. We ablate two core components of Inv^{2}A: the inverse encoder and the invariant decoder. (1) Encoder: We remove the encoder (w/o Enc) and rely on the decoder for inversion in two ways: prompting the decoder with four random demonstrations, and fine-tuning the decoder via LoRA (Hu et al. 2021) to learn the inverse mapping. In both settings, inversion performance drops sharply, showing that the encoder removes noise related to inversion. (2) Decoder: We replace the raw decoder (w/o Raw f) with Qwen2, another well-generalised pretrained LLM. The substitute decoder still achieves reasonable inversion, implying that the corpora trained by different decoders share a similar underlying distribution. However, using f is superior, because it is directly optimized for the target output distribution.

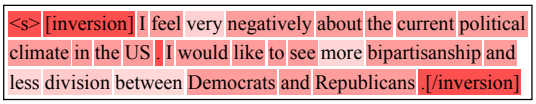
Module-Wise. We further remove a key training component: the source-aware contrastive learning in the alignment phase (w/o CL). This leads to a noticeable performance drop and an overall increase in variance. The result confirms the role of this algorithm in stabilizing inversion across outputs from the same source. We also integrate the dynamic filter module (§3.3) to refine the recovered prompts. The hyperparameter τ is set to 0.5, ensuring the filter passes prompts accounting for the majority of the probability mass. The search process is executed for 1 or 2 rounds, with 3 neighbors expanded per round. This module yields clear performance gains, although the improvement diminishes with additional rounds. Notably, only about 15% of samples trigger the filter, so the extra time overhead is negligible.

Method	BLEU	Token F1	CS	GPT	Exact
Inv^{2}A	35.20	59.95	78.21	65.60	6.65
w/o ENC	1.31	12.43	22.14	4.40	0.00
- LoRA	26.47	51.52	71.43	54.80	3.90
w/o Raw f	33.38	58.33	77.58	62.55	6.55
w/o CL	33.91	58.83	78.09	64.95	6.35
w/ Refine	35.97	60.77	79.16	69.10	7.05
- 2 turn	**36.06**	**60.90**	**79.31**	**69.50**	**7.05**

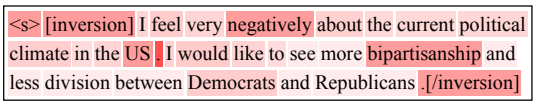
Table 4: Ablation results on structures and modules.

4.4 Extended Discussions

Interpretability. We attempt to analyze the mechanism behind Inv^{2}A. Compared with the forward LLM, the only additional component is the inverse encoder. Thus, following PromptBench (Zhu et al. 2024), we compute token-wise attention scores between the raw output and the encoder-generated pseudo representation. These scores reflect the importance of each token to the decoder. Figure 5 provides an example in user prompt scenario. This example reveals that the encoder shifts the attention distribution over the output sequence: the standalone decoder attends almost uniformly to all tokens, whereas after denoising encoding, the relative attention devoted to tokens carrying key semantic information is increased. A typical class of such tokens include those shared by both the prompt and its output, which often serve as cues to the prompt. This conclusion is general because, for all samples, the average ratio of the shared token's attention to the sequence mean rises from 1.46 to 1.70.



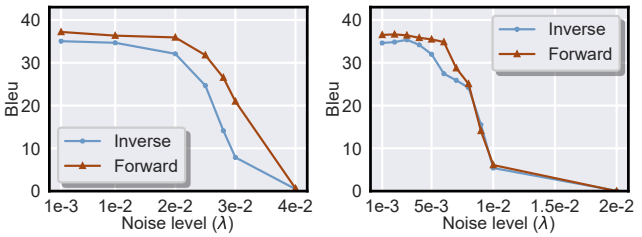
(a) w/o inverse encoder



(b) w/ inverse encoder

Figure 5: Importance visualization, where the prompt is "*How do you feel about the current political climate in the US? What is one thing that you would like to change?*"

Defense. As analyzed in § 4.2, defenses that solely increase sampling diversity have weak effect. This is because Inv^{2}A's core leverage is the invariant decoder: altering sampling strategies merely perturbs outputs without materially disrupting the invariant latent geometry of the decoder. We therefore focus on defenses that modify the LLM itself. A common approach is differential privacy (Ji, Lipton, and Elkan 2014), i.e., adding noise during training to hinder memorization of prompts. However, this is impractical for many already-released models due to the high cost and limited controllability of retraining. Here, we consider a scalable layer-wise noise injection (Liu et al. 2018) defense: add Gaussian noise to selected layers to slightly alter the latent space structure. Guided by observations in the Appendix, we inject noise into the MLP and attention sublayers of the first hidden layer. Results in Figure 6 show that such noise injection is indeed more effective than diversity-based defenses. However, it degrades forward performance. Injecting into the MLP with $\lambda = 2.5\text{e}{-2}$ offers a relative balance but still causes a $\sim 8\%$ drop in BLEU. Thus, in the LLM era, designing scalable defenses remains a challenging problem.



(a) MLP Layer (b) Attention Layer

Figure 6: BLEU of forward prediction and inversion across varying levels of noise injection. λ is the standard deviation.

5 Related Work

Language Model Inversion. This topic is closely tied to intellectual-property and privacy protection. LMI differs from model extraction, which seeks to recover the model itself, e.g., parameters (Maheswaranathan et al. 2021; Wang and Gong 2018). Early prompt-oriented work (Morris et al. 2023) attempts to recover input prompts from embeddings. More recent studies pivot to LLMs. Among these, Output2prompt (Zhang, Morris, and Shmatikov 2024) and Logit2text (Morris et al. 2024) are the most prominent: both gather millions of examples and train an external inverse model, specifically T5 (Raffel et al. 2020), to recover prompts, using either textual outputs or next-token probability vectors as input. Output2prompt additionally considers the system-prompt setting and can exploit multiple outputs. However, such independently trained models generalize poorly to data unseen during training. Other methods query the LLM to elicit its prompts: Zhang, Carlini, and Ippolito (2024) and Chao et al. (2025) design adversarial queries, while Gao et al. (2024) combines uncertainty-based denoising. The former rely on a special assumption that the prompts are hidden within the input window, whereas the latter performs poorly on long, complex prompts.

6 Conclusions and Limitations

We reframed LMI through the *Invariant Latent Space Hypothesis* and proposed **Inv^{2}A**. It reveals significant intrinsic security risks embedded within the LLM's own structure.

Limitations. Our study assumes a white-box setting with parameter access, which constrains applicability in strict black-box contexts. Inversion quality depends on semantic specificity: highly abstract or non-injective prompt–output relations hinder accurate recovery. Finally, our discussion of defense fails to find a really valid schema to overcom LMI.

Acknowledgements

This paper is supported by the National Regional Innovation and Development Joint Fund (No. U24A20254). Haobo Wang is also supported by the Fundamental Research Funds for the Central Universities (No. 226-2025-00085).

References

Ackley, D. H.; Hinton, G. E.; and Sejnowski, T. J. 1985. A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1): 147–169.

Ash, R. B. 2012. *Information theory*. Courier Corporation.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2025. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 23–42. IEEE.

Chen, J.; Liu, Z.; Huang, X.; Wu, C.; Liu, Q.; Jiang, G.; Pu, Y.; Lei, Y.; Chen, X.; Wang, X.; et al. 2024. When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web*, 27(4): 42.

Cheng, Y. 2025. A Scalable Approach to Distributed Large Language Model Inference.

Conover, M.; Hayes, M.; Mathur, A.; Xie, J.; Wan, J.; Shah, S.; Ghodsi, A.; Wendell, P.; Zaharia, M.; and Xin, R. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. *Company Blog of Databricks*.

Das, B. C.; Amini, M. H.; and Wu, Y. 2025. System Prompt Extraction Attacks and Defenses in Large Language Models. *arXiv preprint arXiv:2505.23817*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Frisch, I.; and Giulianelli, M. 2024. LLM Agents in Interaction: Measuring Personality Consistency and Linguistic Alignment in Interacting Populations of Large Language Models. In Deshpande, A.; Hwang, E.; Murahari, V.; Park, J. S.; Yang, D.; Sabharwal, A.; Narasimhan, K.; and Kalyan, A., eds., *Proceedings of the 1st Workshop on Personalization of Generative AI Systems (PERSONALIZE 2024)*, 102–111. St. Julians, Malta: Association for Computational Linguistics.

Gao, L.; Peng, R.; Zhang, Y.; and Zhao, J. 2024. DORY: Deliberative Prompt Recovery for LLM. *arXiv e-prints*, arXiv–2405.

Hebbar, S. S. 2024. arxiv-math-instruct-50k. https://huggingface.co/datasets/Sharathhebbar24/arxiv-math-instruct-50k.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Ji, Z.; Lipton, Z. C.; and Elkan, C. 2014. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*.

Kingma, D. P. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leiker, D.; Finnigan, S.; Gyllen, A.; and Cukurova, M. 2023. Prototyping the use of Large Language Models (LLMs) for adult learning content creation at scale. In *CEUR Workshop Proceedings*, volume 3487, 3–7. CEUR Workshop Proceedings.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2024b. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Liu, X.; Cheng, M.; Zhang, H.; and Hsieh, C.-J. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the european conference on computer vision (ECCV)*, 369–385.

Luo, X.; Yu, T.; and Xiao, X. 2025. Prompt Inference Attack on Distributed Large Language Model Inference Frameworks. *arXiv preprint arXiv:2503.09291*.

Luo, Z.; Xu, C.; Zhao, P.; Sun, Q.; Geng, X.; Hu, W.; Tao, C.; Ma, J.; Lin, Q.; and Jiang, D. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Maheswaranathan, N.; Sussillo, D.; Metz, L.; Sun, R.; and Sohl-Dickstein, J. 2021. Reverse engineering learned optimizers reveals known and novel mechanisms. *Advances in Neural Information Processing Systems*, 34: 19910–19922.

Morris, J.; Kuleshov, V.; Shmatikov, V.; and Rush, A. M. 2023. Text Embeddings Reveal (Almost) As Much As Text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 12448–12460.

Morris, J. X.; Zhao, W.; Chiu, J. T.; Shmatikov, V.; and Rush, A. M. 2024. Language Model Inversion. In *The Twelfth International Conference on Learning Representations*.

Moussa, H. G.; Akhavain, A.; Hosseini, S. M.; and McCormick, B. 2025. Distributed learning and inference systems: A networking perspective. *IEEE Network*.

Neelakantan, A.; Xu, T.; Puri, R.; Radford, A.; Han, J. M.; Tworek, J.; Yuan, Q.; Tezak, N.; Kim, J. W.; Hallacy, C.; et al. 2022. Text and code embeddings by contrastive pretraining. *arXiv preprint arXiv:2201.10005*.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–318.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford alpaca: an instruction-following llama model (2023). *URL https://github. com/tatsu-lab/stanford_alpaca*, 1(9).

teknium1. 2025. GPTeacher: A collection of modular datasets generated by GPT-4.

Ti, X.; Ye, W.; Zhang, Z.; Zhao, J.; Yao, C.; Feng, L.; and Wang, H. 2025. Towards Reverse Engineering of Language Models: A Survey. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 7483–7502.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Wang, B.; and Gong, N. Z. 2018. Stealing hyperparameters in machine learning. In *2018 IEEE symposium on security and privacy (SP)*, 36–52. IEEE.

Wang, K.; Zhang, G.; Zhou, Z.; Wu, J.; Yu, M.; Zhao, S.; Yin, C.; Fu, J.; Yan, Y.; Luo, H.; et al. 2025. A comprehensive survey in llm (-agent) full stack safety: Data, training and deployment. *arXiv preprint arXiv:2504.15585*.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.

Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N. A.; Khashabi, D.; and Hajishirzi, H. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Wei, J.; and Zou, K. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Wu, L.; Zheng, Z.; Qiu, Z.; Wang, H.; Gu, H.; Shen, T.; Qin, C.; Zhu, C.; Zhu, H.; Liu, Q.; et al. 2024. A survey on large language models for recommendation. *World Wide Web*, 27(5): 60.

Wu, M.; Waheed, A.; Zhang, C.; Abdul-Mageed, M.; and Aji, A. F. 2023. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *arXiv preprint arXiv:2304.14402*.

Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; and Jiang, D. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Zhang, C.; Morris, J. X.; and Shmatikov, V. 2024. Extracting Prompts by Inverting LLM Outputs. *arXiv preprint arXiv:2405.15012*.

Zhang, Y.; Carlini, N.; and Ippolito, D. 2024. Effective Prompt Extraction from Language Models. In *First Conference on Language Modeling*.

Zhu, K.; Zhao, Q.; Chen, H.; Wang, J.; and Xie, X. 2024. Promptbench: A unified library for evaluation of large language models. *Journal of Machine Learning Research*, 25(254): 1–22.