Latent Prompt Inference: A Reverse Engineering Approach to LLM Interpretability and Debugging

AI Researcher

October 6, 2025

Abstract

This paper proposes a novel methodology for Large Language Model (LLM) interpretability and debugging by inverting the typical prompt-to-output paradigm. Instead of viewing prompt stealing as a security vulnerability, we reframe it as a diagnostic tool. We introduce the concept of "Latent Prompt Inference" (LPI), a method to infer the most likely latent prompt that explains a given LLM output. This inferred prompt serves as a human-readable window into the model's internal state, revealing hidden assumptions, biases, and reasoning failures. We propose the development of an "Interpreter" model, a sequence-to-sequence network trained to perform this reverse mapping. The utility of this approach is demonstrated through a series of proposed experiments, including detecting the model's implicit assumptions, quantifying stereotypical biases, and creating a new quantitative metric for model-user alignment. This research shifts the focus from what an LLM says to what it *implicitly understood*, providing a powerful new tool for AI safety and transparency.

1 Core Idea

The central thesis of this work is to repurpose the concept of prompt reverse engineering from a security attack into a tool for model diagnostics. When a Large Language Model (LLM) produces an unexpected, biased, or incorrect output, it is often difficult to determine the root cause of the failure. Our proposed method, Latent Prompt Inference (LPI), aims to address this by answering the question: *"What prompt would have made this output the correct and expected answer?"*

By inferring this "latent prompt," we can create a human-readable explanation of the model's behavior. If the latent prompt reveals a simplistic or biased assumption, it provides a clear diagnosis of the model's failure mode. This technique moves beyond surface-level output analysis and offers a deeper insight into the model's interpretation of a user's request.

2 Methodology: The "Interpreter" Model

To perform Latent Prompt Inference, we propose the design and training of a specialized sequence-to-sequence model, which we term the **Interpreter**.

2.1 Architecture

The Interpreter will be based on a standard Transformer architecture (e.g., T5 or BART), optimized for the text-to-text task of mapping an LLM's output back to a plausible prompt.

- **Input:** A text sequence, O, representing the output from a target LLM that requires analysis.
- **Output:** A text sequence, P', representing the inferred latent prompt that most likely generated O.

2.2 Training Data Generation

A large-scale, high-quality dataset of (Output, Prompt) pairs is required to train the Interpreter. We will generate this dataset synthetically:

1. Curate a diverse set of prompts from public datasets (e.g., Alpaca, Dolly, ShareGPT) to ensure broad domain coverage.

2. Use a state-of-the-art "teacher" LLM (e.g., GPT-4) to generate high-quality, canonical answers for each prompt.

3. This process results in a training corpus $D = \{(O_1, P_1), (O_2, P_2), \ldots, (O_n, P_n)\}$, which can be used to fine-tune the Interpreter.

2.3 The Mathematical Formulation

We formally frame the task as finding the optimal prompt P' that maximizes the conditional probability of observing the output O given a target model M.

$$P' = \arg\max_P P(O|P, M) \tag{1}$$

The Interpreter model, I, is trained to be a function that approximates this maximization, such that:

$$P' \approx I(O) \tag{2}$$

3 Proposed Experiments to Demonstrate Utility

To validate the effectiveness of LPI as a diagnostic tool, we propose three key experiments.

3.1 Experiment 1: Detecting Hidden Assumptions

Problem: LLMs often make implicit assumptions about a user's intent or required level of detail, leading to misaligned responses.

Procedure:

1. Provide a target LLM with a nuanced, expert-level prompt, such as: *"Explain the trade-offs between Bayesian and frequentist statistics in the context of clinical trial design."*

2. The LLM generates a high-level, simplistic response that omits the specific context of clinical trials.

3. Feed this simplistic output into our trained Interpreter.

Expected Outcome: The Interpreter will generate a latent prompt that reflects the model's actual behavior, for example: *"Explain Bayesian vs. frequentist statistics in simple terms."* This outcome explicitly diagnoses that the model misinterpreted the user's required depth and defaulted to a more generic instruction.

3.2 Experiment 2: Uncovering and Quantifying Bias

Problem: Identifying the specific stereotype or bias that leads to a problematic LLM output can be challenging.

Procedure:

1. Give a target LLM a neutral prompt known to sometimes elicit biased responses, such as: *"Write a short story about a computer programmer."*

2. The LLM generates a story featuring a stereotypical character (e.g., a socially awkward male).

3. Input the generated story into the Interpreter.

Expected Outcome: The Interpreter should output a latent prompt that makes the implicit bias explicit, such as: *"Write a short story about a stereotypical male computer programmer."* This reveals the model's internal bias in a clear, interpretable format.

3.3 Experiment 3: A New Metric for Model Alignment

Problem: Quantifying the alignment between a user's intent and a model's response is a key challenge in LLM evaluation.

Procedure:

1. For any given interaction, let P_{user} be the user's original prompt and O_{model} be the model's output.
2. Use the Interpreter to infer the latent prompt: $P_{\text{latent}} = I(O_{\text{model}})$.
3. Compute a new metric, the **Alignment Score**, defined as the semantic similarity between the user's prompt and the latent prompt. This can be calculated using the cosine similarity of their sentence embeddings.

$$\text{AlignmentScore} = \frac{E(P_{\text{user}}) \cdot E(P_{\text{latent}})}{\|E(P_{\text{user}})\| \|E(P_{\text{latent}})\|} \tag{3}$$

where $E(\cdot)$ is a sentence-embedding function.

Expected Outcome: The Alignment Score provides a quantitative measure of model performance. A score near 1.0 indicates high alignment, whereas a low score signals a misalignment between user intent and model behavior, providing a valuable signal for debugging and model improvement.