From Output to Input: Modeling the Reverse Prompt Engineering Problem

David Busbib Yehuda Frist Yarin Ohayon

July 30, 2025

Abstract

This research proposal investigates the novel concept of Reverse Prompt Engineering, which aims to reconstruct the original prompts that generated specific outputs from language models. By treating outputs as clues to their generative origins, this work contributes to efforts in model interpretability, auditability, and safety. We propose a formulation of the task, using aligned instruction-output pairs to train encoder-decoder models capable of prompt reconstruction. This direction complements recent model-agnostic approaches and offers new perspectives on the alignment between human intent and machine behavior.

Code: `GitHub`

1 Introduction

Large language models (LLMs) have demonstrated remarkable proficiency in generating coherent and contextually appropriate outputs in response to natural language instructions [LZY23, ZDL+23]. As these systems become integral to a range of applications, from question answering to creative writing, attention has increasingly turned to the interpretability of their outputs. Understanding how a model arrives at a given response is a central challenge in responsible AI [MZC+23]. One promising direction within this larger agenda is reverse prompt engineering, the task of inferring the original prompt or instruction that led to the observed result [ZCI23].

This inverse formulation of language generation is not only cognitively intuitive, akin to deducing questions from answers, but also carries significant practical value. By recovering the intent behind an output, we gain tools for auditing model behavior, detecting prompt leakage, and tracing the provenance of generated content. Moreover, reverse prompting offers insight into how models encode and preserve instructional information, contributing to our understanding of how language models generalize across tasks [NFM+25, LJD+24].

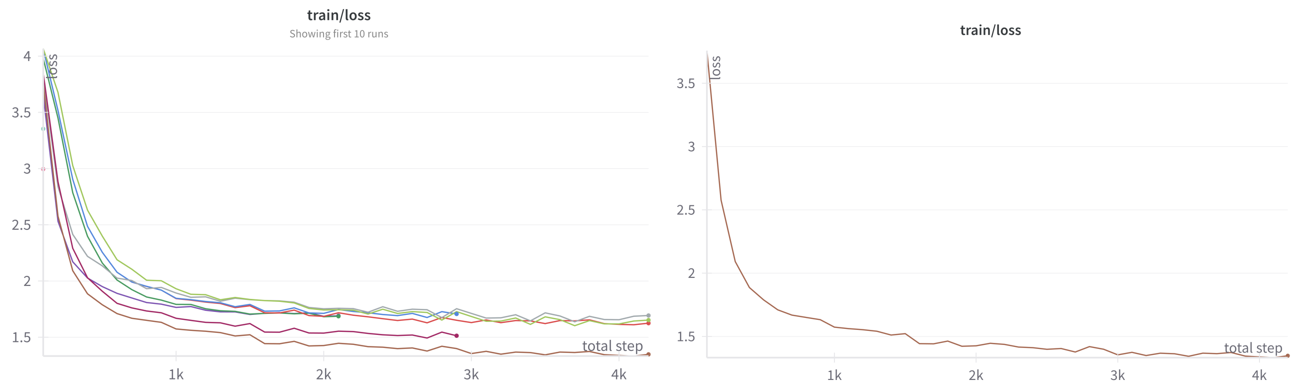
In this work, we formalize reverse prompt engineering as a supervised sequence-to-sequence task. We leverage the Alpaca dataset, comprising over 50,000 instruction-output pairs generated via OpenAI's `text-davinci-003` model to train an encoder-decoder model capable of predicting prompts from responses. Our goal is not only to reconstruct natural instructions with high semantic fidelity, but also to evaluate how well the reconstructed prompts elicit outputs aligned with the originals.

2 Related Work

Recent work such as [LK24] introduced a black-box, training-free framework that employs a genetic algorithm-inspired search to recover prompts by maximizing semantic similarity with outputs, an approach that is model-agnostic but computationally intensive. In contrast, our method is supervised and data-driven, leveraging the alignment within instruction-following datasets like Alpaca to train models capable of generalizing across prompt types. This direction parallels reversible prompt tuning [HSS+22], where T5 is trained to generate prompts from task outputs, underscoring the inherent bidirectionality of instruction-based modeling.

3 Data

Our study is grounded in the Alpaca dataset [TGZ+23], a publicly released collection of 52,000 instruction-following examples generated using OpenAI's `text-davinci-003` model. Each instance consists of a triplet: an instruction specifying the task, an optional input providing context, and the corresponding model-generated output. To align the dataset to our given task, we reformat each example into a supervised training pair where the output serves as the input to the model and the instruction becomes the target. The optional input field, present in only a minority of cases, is appended to the original input. Additional preprocessing steps include removal of uninformative examples and truncation to satisfy model constraints. This reformulation enables the model to learn prompt recovery from the generated content alone, reflecting the core inversion at the heart of the task.



(a) multiple runs with varying settings

(b) best-performing

Figure 1: Comparison of training loss curves for different hyperparameter configurations.

4 Methods

To model the reverse prompt engineering task, we fine-tune a pre-trained T5 base model [RSR+20] on the preprocessed Alpaca dataset. The model is trained in a sequence-to-sequence fashion(see Appendix 3), where the generated outputs serve as inputs, and the original instructions are treated as targets, thereby learning to reverse-engineer the instruction that likely produced a given response. Training is carried out using Hugging Face's Trainer API. Following empirical tuning (see Figure 1), the optimal hyperparameters were determined to be a learning rate of 5e-5, a batch size of 4, weight decay of 0.01, and a warmup phase comprising 500 steps. Gradient accumulation over two steps was employed to simulate a larger effective batch size. The model was trained on 3 epochs with validation loss used to monitor convergence and mitigate overfitting. This configuration achieved a final training loss of 1.5518 and an evaluation loss of 1.317. We evaluate our model against two baselines. The zero-shot baseline involves prompting google/flan-t5-base with a fixed instruction (see Appendix 1 zero-shot) asking it to infer the original prompt from a given output, without any task-specific tuning. The few-shot baseline augments this setup by providing google/flan-t5-base with 3 in-context examples of successful output-to-prompt predictions (see Appendix 1 few-shot). These serve to contextualize the task and enable limited adaptation via demonstration. The model outputs are evaluated across a range of metrics designed to capture both surface-level similarity and deeper semantic alignment.

5 Results

Our model consistently outperforms zero-shot and few-shot baselines in most evaluation metrics (see Table 1). It demonstrates markedly higher performance in text similarity measures such as BLEU, ROUGE, and METEOR, indicating improved alignment with the original prompts. In terms of semantic similarity, the fine-tuned model achieves the highest BARTScore, indicating greater alignment with the original prompts compared to both baselines. For perplexity, the few-shot model ex-

Metric	Fine-Tuned T5 (Our Model)	Zero-shot	Few-shot
BLEU	**0.2251**	0.0505	0.0644
ROUGE-1	**0.5257**	0.2493	0.2639
ROUGE-L	**0.4989**	0.2339	0.2437
METEOR	**0.4728**	0.1797	0.2021
BARTScore	**-2.667**	-3.8709	-3.7706
Perplexity	92.658	2797.0630	**52.9779**
MAUVE	0.0077	**0.0509**	0.0093
Self-BLEU	**0.5223**	0.0786	0.4824

Table 1: Average performance metrics across models, highlighting the superiority of the fine-tuned T5.

hibits the lowest value, suggesting higher fluency under this metric, while the zero-shot model performs worst. MAUVE scores, which reflect distributional similarity to the original prompt distribution, are generally low across all models, with the zero-shot baseline achieving the highest score. Finally, Self-BLEU indicates that the fine-tuned and few-shot models generate less diverse outputs than the zero-shot baseline, which shows substantially higher output variability.

6 Analysis

6.1 Lexical Overlap and Surface Fidelity

The BLEU and ROUGE scores highlight the fine-tuned model's superiority in capturing surface-level alignment with original prompts. BLEU reflects stronger n-gram precision, while ROUGE-1 and ROUGE-L indicate better token recall and sequence ordering. In contrast, the lower scores of the baselines suggest difficulty maintaining structural coherence, whereas the fine-tuned model better preserves the syntactic structure of complex instructions.

6.2 Semantic Similarity

Beyond lexical fidelity, the METEOR metric suggests that our model also demonstrates improved semantic preservation (see Figure 2). By incorporating synonymy, stemming, and paraphrastic flexibility, METEOR highlights cases where meaning is retained even when exact word matches differ. The large performance gap between our model and both baselines in this metric points to a genuine semantic advantage. BARTScore further supports this conclusion, offering a model-based evaluation of textual plausibility and contextual alignment. The fine-tuned model achieves the least negative BARTScore among the three, implying its outputs are more semantically coherent and contextually grounded than those of the baselines (see Appendix 2). This is particularly relevant in a task where reconstructed prompts must not only match the target in form, but also in communicative intent.

6.3 Fluency and Language Modeling Consistency

Perplexity results reveal a nuanced contrast between surface fluency and task-specific relevance. The few-shot baseline exhibits the lowest perplexity, consistent with its in-context generation from a highly fluent pretrained model. The fine-tuned model, while more perplexing under a general language model, produces content more closely aligned with the ground truth prompts. The zero-shot configuration fares worst in both perplexity and accuracy metrics, suggesting that fluency in isolation is insufficient. These findings illustrate that in prompt reconstruction tasks, perplexity should be interpreted with caution: low perplexity may signal linguistic smoothness, but not necessarily fidelity to the underlying instruction.

6.4 Distributional Alignment and Output Diversity

MAUVE scores are uniformly low across all models, with only minor variation. While the zero-shot baseline achieves the highest score, the values are small enough to suggest that none of the models strongly align with the

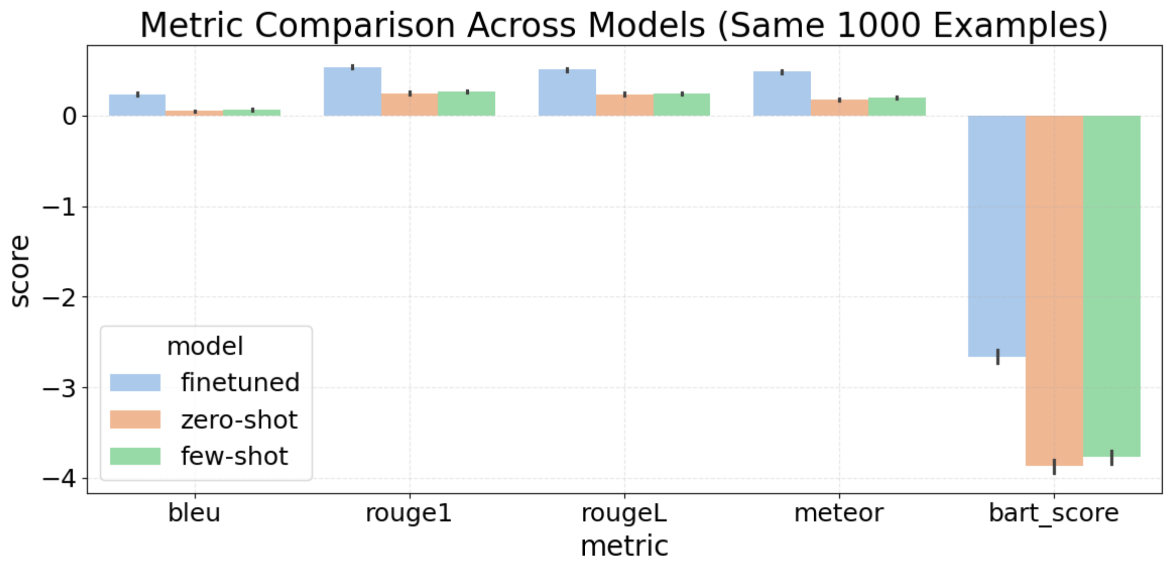


Figure 2: Comparison of evaluation metrics across all models.

token-level distribution of the target prompts. This may reflect limitations of the metric in low-data or short-sequence regimes, or a genuine lack of stylistic convergence. In contrast, Self-BLEU offers more actionable insight into model behavior. The fine-tuned model yields the highest Self-BLEU score, indicating lower output diversity and greater internal consistency. This is beneficial in our task, as it reflects a model that has learned a stable transition from outputs to prompts without overgeneralizing. The few-shot model follows closely, while the zero-shot model displays substantially higher diversity, which in this case reflects variability at the expense of precision. Taken together, these results suggest that the fine-tuned model strikes a balance between diversity and determinism, generating prompts that are both consistent and accurate without degenerating into repetition.

7 Conclusion

This work presents a supervised framework for reverse prompt engineering, recovering original instructions from the outputs of large language models. By reformulating the Alpaca dataset and fine-tuning a T5-base model, we demonstrate that prompt reconstruction is not only feasible, but yields superior performance compared to strong zero-shot and few-shot baselines across lexical, semantic, and structural dimensions. Our evaluation, grounded in diverse metrics such as BLEU, ROUGE, METEOR, BARTScore, and Self-BLEU, reveals that fine-tuned models can effectively capture both the surface-level and semantic characteristics of the original instructions. While perplexity and distributional similarity metrics expose subtle limitations and trade-offs, our findings underscore the value of supervised approaches for aligning generated text with its generative intent. More broadly, this study contributes to the interpretability of language models by offering tools to trace outputs back to their prompts, with implications for transparency, safety, and provenance in AI systems.

8 Future Work

An important direction we initially intended to pursue involved closing the reconstruction loop: feeding the predicted prompts generated by our model back into the original language model, OpenAI's `text-davinci-003`, that produced the outputs in the Alpaca dataset. This would have enabled a more direct and grounded evaluation of reconstruction quality by comparing the original outputs with those elicited by the recovered prompts, thereby measuring functional equivalence rather than surface similarity alone. Unfortunately, access to `text-davinci-003` was deprecated. Thus, preventing us from conducting this evaluation.

Future work will focus on identifying a dataset generated by a non-deprecated instruction-tuned model, such as `gpt-3.5-turbo`, to enable round-trip evaluation—feeding reconstructed prompts back into the original model and comparing the resulting outputs. This functional assessment will complement current similarity metrics.

9 Limitations and Ethical Considerations

While this work demonstrates the feasibility of reverse prompt engineering, it also raises important limitations and ethical considerations. First, the task remains inherently ill-posed in cases where multiple plausible prompts could lead to similar outputs, challenging the notion of a single ground truth. This ambiguity limits the reliability of evaluation metrics and the interpretability of model predictions in open-ended contexts.

Moreover, the ability to infer prompts from outputs introduces potential risks related to privacy, intellectual property, and prompt leakage. In settings where prompts contain sensitive, proprietary, or user-specific information, successful reconstruction could be misused for surveillance, competitive inference, or model inversion attacks. As reverse prompting capabilities improve, it is essential to consider safeguards that mitigate such risks, including data anonymization, differential privacy techniques, and restrictions on model deployment in sensitive domains.

References

[HSS+22] Dan Haviv, Tomer Shoham, Michael Shlain, Amir Globerson, and Jonathan Berant. Reversible prompt tuning. In *Findings of the Association for Computational Linguistics: EMNLP*, 2022.

[LJD+24] Xu Liu, Ruoxi Jia, Yufei Ding, et al. Prompt inversion: Characterizing and evading prompt tuning inversion attacks. *arXiv preprint arXiv:2405.15012*, 2024.

[LK24] Hanqing Li and Diego Klabjan. Reverse prompt engineering. *arXiv preprint arXiv:2411.06729*, 2024.

[LZY23] Renze Lou, Kai Zhang, and Wenpeng Yin. A comprehensive survey on instruction following. *arXiv preprint arXiv:2303.10475*, 2023.

[MZC+23] John X. Morris, Wenting Zhao, Justin T. Chiu, Vitaly Shmatikov, and Alexander M. Rush. Language model inversion, 2023.

[NFM+25] Murtaza Nazir, Matthew Finlayson, John X. Morris, Xiang Ren, and Swabha Swayamdipta. Better language model inversion by compactly representing next-token distributions. *arXiv preprint arXiv:2506.17090*, 2025. Revised version v2 on June 23, 2025.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[TGZ+23] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[ZCI23] Nelson F. Liu Zhang, Nicholas Carlini, and Daphne Ippolito. Effective prompt extraction from language models. *arXiv preprint arXiv:2307.06865*, 2023.

[ZDL+23] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

Appendix 1 - Prompt Engineering

Zero-shot Prompt

For the zero-shot baseline, we constructed a general-purpose instruction designed to elicit the original prompt from a given output without any prior examples. The prompt format is as follows:

```
Given this output, what prompt likely generated it?

Output: {example['input_text']}
Prompt:
```

A Few-shot Prompt

For the few-shot baseline, we employed in-context learning by choosing three diverse instruction-output pairs from the training data. Each example included a context (if provided), the model-generated response, and the corresponding original instruction framed as a question.

Answer:
Context: `<no input>`
Response:

1. I've told you a million times.
2. She's as light as a feather.
3. His car is faster than lightning.

Question: Provide 3 examples of hyperboles.

Answer:
Context: Webinar and Ebook
Response:

> Both webinars and ebooks can be used as lead magnets to attract new customers, but each offers its own unique set of benefits. Webinars are a great way to engage with an audience and provide a live demonstration of your product or service. They also provide the opportunity to answer questions and receive feedback from potential customers. On the other hand, ebooks require less effort to create and can provide in-depth information about your products and services in an easily-digestible format. They are versatile and can be shared widely, either as downloadable files or embedded on your website. Additionally, ebooks are cost-effective and can be used to promote your business to a wider audience.

Question: Compare the characteristics of two different types of lead magnets.

Answer:

1. Make a list before you shop and stick to it.
2. Buy generic items or those on sale.
3. Consider buying in bulk and dividing up items into smaller portions.
4. Look for discounts or coupons.
5. Buy fresh produce in season when they are cheaper.
6. Buy only what you need, and watch expiration dates.
7. Shop around and compare prices.

Question: Provide at least five tips on how to save money when buying groceries.

Appendix 2 - Additional Plots

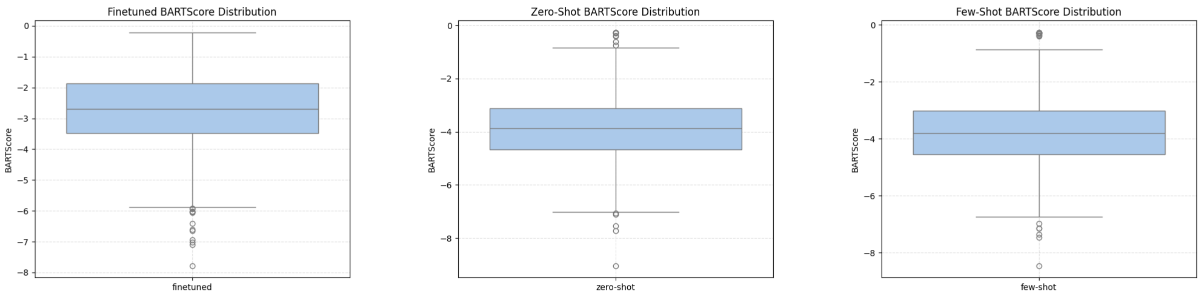


Figure 3: These box plots illustrate the distribution of BARTScore values for each model. The central line in each box represents the median score, while the box bounds capture the interquartile range. Whiskers and outliers provide insights into score variability and consistency across different samples.

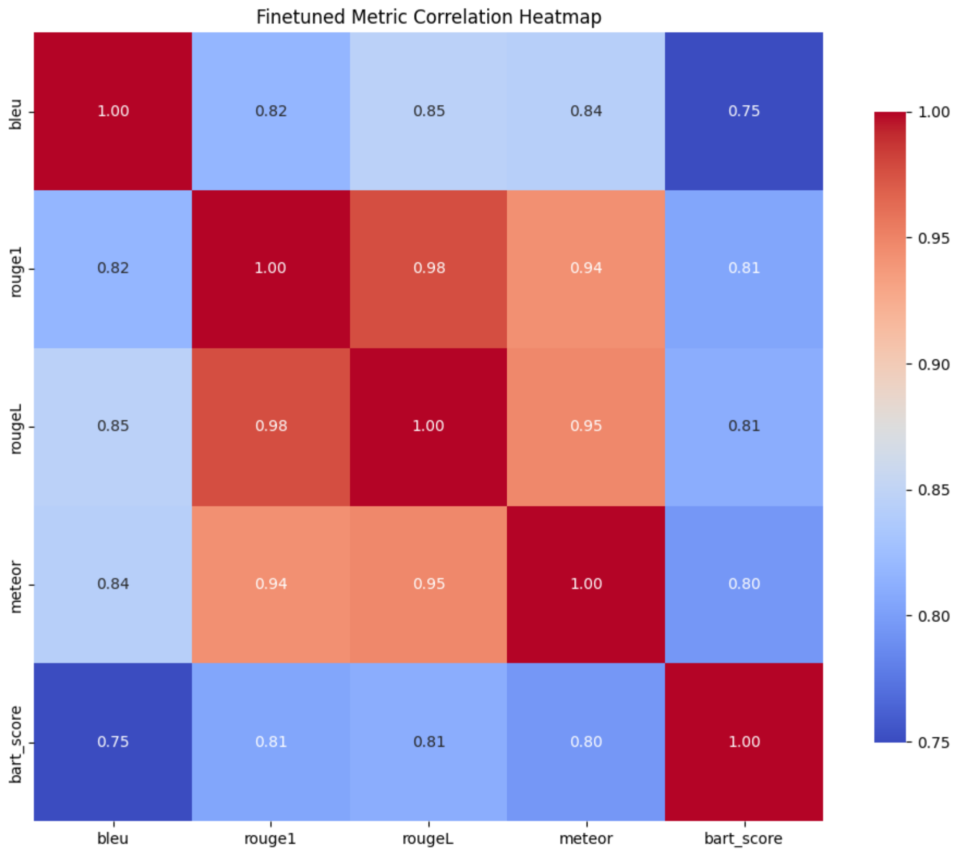


Figure 4: Metric correlation heatmap.

Appendix 3 - Our Models Best and Worst Predictions

Best Examples

Example 1: Best

Input Context: The value of n in the expression n^2 - 5n + 6 is 3 or -2.
Predicted Prompt: Find the value of n in the expression n2 - 5n + 6.
Actual Prompt: Find the value of n in the expression n2 - 5n + 6.

Example 2: Best

Input Context: One way to design an algorithm to classify a basketball shot as a 3-pointer or 2-pointer is to divide the court into three zones: outside of the 3-point arc, inside the 3-point arc and outside the court. When the ball is shot, calculate the distance from the basket and map it to the three zones. If the ball is shot from outside the 3-point arc, it is a 3-pointer, and if it is shot from inside the 3-point arc or outside the court, it is a 2-pointer.
Predicted Prompt: Design an algorithm to classify a basketball shot as a 3-pointer or 2-pointer.
Actual Prompt: Design an algorithm to classify a basketball shot as a 3-pointer or 2-pointer.

Example 3: Best

Input Context: Context: New York and San Francisco
Response: The shortest path between New York and San Francisco, using Djikstra's algorithm, is: New York - Denver - San Francisco.
Predicted Prompt: Find the shortest path between two cities using Djikstra's algorithm.
Actual Prompt: Find the shortest path between two cities using Djikstra's algorithm.

Worst Examples

Example 4: Worst

Input Context: Context: Recent advancements in AI technology have promised to revolutionize many industries, including _
Response: the healthcare industry. AI-powered technologies are revolutionizing the way healthcare is diagnosed and administered...
Predicted Prompt: Summarize the following article in one sentence.
Actual Prompt: Please complete the given paragraph.

Example 5: Worst

Input Context: Context: Tyrese
Response: Tyrese
Predicted Prompt: Given a product, identify the product.
Actual Prompt: Spell the proper noun correctly.

Example 6: Worst

Input Context: Context: Movie Name: *Bruce Almighty*
Genre: Comedy
Starring: Jim Carrey
Response: Bruce Almighty is a deftly crafted comedy starring the incomparable Jim Carrey...
Predicted Prompt: Write a review of a movie about the Almighty.
Actual Prompt: Generate an appropriate description write-up given the following input.