RETRIEVE PROMPT THROUGH REVERSE ENGINEERING

POOJA B E
M.tech, Dept. of CSE, RLJIT

Dr.Varalakshmi K R
Associate Professor,Dept. of CSE

ABSTRACT:

Large Language Models (LLMs) handle substantially more complicated natural language processing tasks, although they rely largely on well-crafted prompts for maximum performance. Manual prompt engineering, on the other hand, is time-consuming, frequently suboptimal, and inflexible in the face of multiple input prompt variations. Furthermore, there is an increasing need to address the security implications of prompt-extraction attacks.Existing research has proposed automated methods for prompt engineering, which range from rewriting underoptimized prompts to creating high-quality human-like prompts from scratch. Despite these advances, issues remain in achieving timely efficacy, robustness, and security.

***KEYWORDS*:** Chain of Thoughts, Zeroshots, Fewshots, Prompt Engineering, Prompt Recovery, NLP, and LLM.

INTRODUCTION

The technique of creating prompts to get desired results from AI or large language models is known as prompt engineering. The main thesis is that the quality of the question posed and the quality of the response are closely related. One can affect the outputs and improve the functionality and performance of the system by carefully crafting prompts.

In natural language processing (NLP), reverse prompt engineering is an interesting method that flips the conventional prompt-based generation process.Reverse prompt engineering involves examining a model's output stop predictions and determining the prompts that most likely produced them, as opposed to creating prompts to elicit desired responses. This method has drawn interest because it can provide insights into the limitations, biases, and capabilities of language models.Reverse engineering prompts can help us better understand NLP systems, enhance their functionality, and reduce risks like biased or dangerous outputs.

Reverse prompt engineering creates a new avenue for investigating and analyzing language model behavior. It illuminates the "black box" of AI-generated text and provides insightful information for field researchers and developers. The term "Prompt Rewrite" refers to changing or rewording the original prompt provided to a language model in order to direct it toward generating desired results. This method is frequently applied in NLP tasks where the generated text's quality and applicability are crucial.

There are numerous worthwhile applications for reverse prompt engineering in a variety of fields

Understanding Model Behavior: We can learn more about how language models generate text by examining inferred prompts.Analyze model choices, spot biases, and strengthen model robustness.

Bias detection and mitigation: By identifying prompts that result in biased outputs, this technique can reveal biases encoded in language models.Once identified, developers can mitigate these biases by fine-tuning model parameters or training data.

Enhancing Model Performance: Examining inferred prompts can give insight into how effective various prompts are, directing the creation of more effective prompts for particular tasks and improving model performance.

Automatic quality control and evaluation of language models is made possible by reverse prompt engineering, which evaluates the consistency and coherence of inferred prompts across multiple outputs to determine the quality of generated text.

Content Filtering and Moderation: By comprehending prompts that result in unwanted or harmful outputs, developers can create better content filtering and moderation systems that help identify and stop the spread of harmful information.

Reverse prompt engineering in creative writing and content generation can uncover the underlying patterns and structures that language models employ, generating fresh ideas for producing more engaging and diverse content.

For our case study, we will utilize combinations of popular large language models (LLMs) such as BERT [1], RoBERTa [2], Mistral-7B [3], Gemma [4], and Sentence-T5 [5].

LITERATURE SURVEY

“A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications”

In order to maximize the performance of Large Language Models (LLMs), particularly in zero-shot and few-shot learning scenarios, prompt engineering is essential. Sahoo et al. provide a thorough and organized analysis of prompt engineering methods and their various uses in [6]. The authors divide the current methods into two categories: automated prompting, which includes optimization-based techniques like reinforcement learning and evolutionary strategies, and manual prompting, which includes zero-shot, few-shot, and chain-of-thought methods. Although manual methods are easier to understand and use, they frequently have problems with reproducibility and scalability. On the other hand, automated methods can be computationally demanding but offer better consistency and performance across a range of tasks.

The study also looks at more sophisticated methods like soft prompting and prefix tuning, which allow for task-specific adaptation by learning continuous prompt representations without changing the core LLM. These methods provide notable efficiency gains, particularly in situations involving multiple tasks and limited resources

Additionally, Sahoo et al. examine widely used evaluation metrics for prompt quality, such as BLEU, accuracy, perplexity, and human judgment. In addition to outlining potential remedies and future research directions, the survey draws attention to a number of the field's shortcomings, including prompt sensitivity, domain dependency, and susceptibility to adversarial attacks. Question answering, reasoning, summarization, code generation, and interactive systems are some of the main application areas that are covered.

This work provides a structured understanding of prompt engineering's current state and guides future innovations, thus serving as a fundamental resource for researchers and practitioners alike, LLM utilization[6].

Efficient prompting methods for large language models: A survey

A thorough survey aimed at enhancing the effectiveness of prompting techniques for Large Language Models (LLMs) is offered by Chang et al. [7]. The computational and memory costs of interacting with LLMs rise sharply as they get bigger and more capable. In order to minimize resource consumption and maintain or improve model performance, this paper classifies and evaluates effective prompting techniques.

The authors divide current approaches into three main groups: retrieval-augmented prompting, parameter-efficient approaches, and input-level optimization. Prompt compression and rephrasing strategies are examples of input-level optimization methods that reduce the number of input tokens. Prefix tuning, adapter tuning, and LoRA are examples of parameter-efficient techniques that learn small parameter subsets rather than fine-tuning the entire model. Scalable deployment in multi-task environments is made possible by these methods.

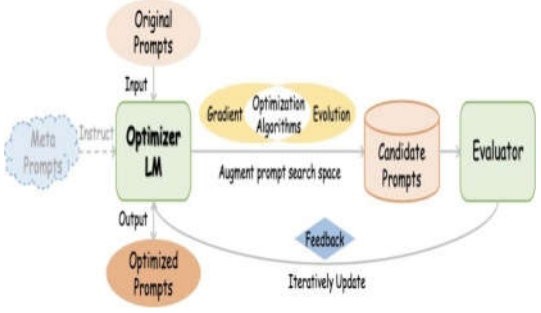
Retrieval-augmented prompting, which blends language models with outside memory or knowledge sources, is another significant topic discussed. This integration eliminates the need to encode all task knowledge within the prompt itself by enabling models to dynamically retrieve pertinent context.

Evaluation benchmarks, efficiency metrics (such as FLOPs and latency), and trade-offs between cost and performance are also covered in the survey. The authors draw attention to real-world issues like task-specific generalization, model brittleness to prompt variations, and context length restrictions.

All things considered, Chang et al. [7] provide a useful resource that compiles recent attempts to increase the computational efficiency of LLMs, particularly for practical applications needing

economical inference.

Figure1:Iterative Prompt Optimization Framework Using Language Models



Prompt perturbation consistency learning for robust language models.

The sensitivity of large language models (LLMs) to prompt perturbations is a crucial issue in the prompt engineering field that

is addressed by Qiang et al. [8]. Significant variations in output can result from even small changes to a prompt's wording, structure, or phrasing, which can reduce reliability and cause inconsistencies. The authors suggest a novel framework known as Prompt Perturbation Consistency Learning (PPCL) to address this problem.

In order to encourage the model to generate consistent and logical outputs across semantically equivalent prompt variants, the PPCL method introduces consistency constraints during training. Regardless of the prompt form, the model learns to align its representations and predictions by introducing perturbed prompts—made by paraphrasing, reordering, or lexical substitutions—into the training loop. This improves generalization and robustness without requiring explicit task-specific fine-tuning.

The authors show how well PPCL performs on a variety of benchmark tasks, such as sentiment analysis, question answering, and natural language inference. Improved output stability, accuracy, and resistance to adversarial prompt changes are demonstrated by experimental results, especially in zero-shot and few-shot learning scenarios.

In addition to introducing a scalable technique to improve LLM reliability under real-world, noisy, or user-generated inputs, this work emphasizes the significance of robustness in prompt-based learning. Techniques like PPCL will be crucial in ensuring that LLMs are reliable and robust in a variety of deployment scenarios as prompt-based interfaces gain popularity [8].

Learning to rewrite prompts for personalized text generation[11]

When using Large Language Models (LLMs), personalization in language generation is still a major challenge. This is especially true when it comes to changing generic prompts to better suit the preferences of specific users. A new method for learning to rewrite prompts to produce more individualized and user-aligned outputs is presented by Li et al. in [11]. In their work, they present a framework that dynamically changes generic prompts into versions that are specific to each user's profile and Circumstances.

A rewriting model trained to optimize for both linguistic coherence and alignment with user-specific generation goals is the paper's main contribution. The model learns to modify prompts without sacrificing task intent or fluency through the use of supervised learning and user interaction data. By bridging the gap between automated generation and manual personalization, this method allows for scalable implementation in chatbots, recommendation systems, and content creation platforms.

In contrast to static or one-size-fits-all prompts, the authors show that prompt rewriting dramatically enhances text relevance, diversity, and personalization metrics. The efficacy of their approach is confirmed by extensive testing on tasks like creating personalized queries and product reviews.

The potential of prompt rewriting as a compromise between comprehensive model fine-tuning and straightforward prompt crafting is highlighted by this work. It lays the groundwork for upcoming systems that seek to strike a balance between interpretability, efficiency, and personalization in human-AI interaction [11].

PRSA: Prompt Reverse Stealing Attacks Against Large Language Models

By presenting PRSA (Prompt Reverse Stealing Attacks), a novel threat model that seeks to recover sensitive or proprietary prompts from LLM outputs, Jiang et al. [12] investigate the security flaws of large language models (LLMs). Their research reveals a growing problem with prompt-based interfaces: the ability of adversaries to deduce or recreate hidden prompts used during model interaction, which poses serious threats to data privacy, model integrity, and intellectual property.

The PRSA framework reverse-engineers the original prompts by using output patterns and semantic hints from LLM responses. This is achieved by combining query-based probing methods, language priors, and optimization algorithms. High recovery accuracy is demonstrated by the authors' evaluation of PRSA's efficacy across a range of models and prompt styles, particularly for structured or templated prompts.

This study shows that attackers can reconstruct prompts with frightening accuracy even with restricted access to model outputs. Additionally, while acknowledging the trade-offs between robustness and usability, the paper discusses defense strategies like output obfuscation, prompt randomization, and detection mechanisms.

Jiang et al. [12] offer a critical viewpoint on the LLM safety landscape by revealing the vulnerabilities in prompt privacy. According to their findings, prompt-level security should be given top priority by the research community, especially for sensitive and commercial applications that involve private data or instructions.

UseCases:

• Creating a prompt library: We can create optimized prompts to manage and resolve migration problems for agile tasks like Java migration.
• Improving Customer Support Systems: Customer support bots can be improved to deliver more precise and beneficial responses, increasing customer satisfaction, by recovering and analyzing effective prompts.
• Enhancing Code Generation and Debugging Tools: Reverse engineering prompts can assist developers by fine-tuning models, producing more precise code snippets, and offering better debugging suggestions.
• Helping writers with creative writing and storytelling: AI systems can produce more engaging and imaginative narratives by reversing the design of effective storytelling prompts

2. Related Work:

- Prompt tuning (Prefix Tuning, P-Tuning v2, LoRA).
- Reverse prompt engineering (analyzing outputs to infer the implicit prompts).
- Cross-model prompt adaptation (e.g., prompt transfer between GPT and BERT).

Prompt tuning (Prefix Tuning, P-Tuning v2, LoRA).
The Function of Prompt Tuning Methods in Universal Prompt Generation

In order to modify pre-trained language models (LLMs) for particular tasks without adjusting the model as a whole, prompt tuning is a family of techniques. These methods are especially useful for parameter-efficient transfer learning, which allows for the reuse of large models with minor task-specific modifications.

*Tuning Prefixes

Prefix tuning is the process of appending to the input tokens a series of continuous task-specific vectors that are learned during training. Without changing the primary weights, these vectors serve as a kind of "soft prompt," conditioning the model's behavior. This technique is especially useful for large decoder-only models like GPT, where the prefix can effectively steer the generation.

*P-Tuning version two

P-Tuning v2, an expansion of the original P-Tuning technique, injects soft prompts not only at the input layer but also across several transformer layers by combining prefix tuning and deep prompt insertion. Stronger task adaptation with improved gradient flow and performance are made possible by this, particularly in settings with limited resources or multilingual users.

*Low-Rank Adaptation, or LoRA

By adding low-rank matrices to the model's feedforward and attention layers, LoRA lowers the number of trainable parameters. Despite not being a prompt tuning technique in the conventional sense, LoRA enables effective adaptation in a manner akin to prompting and can be used in conjunction with prompt tuning techniques to further minimize processing and storage needs.

In connection with the Universal Prompt Generator (UPG)

These methods guide the architectural decisions for producing flexible prompts in the framework of our Universal Prompt Generator:By incorporating prompt tokens into the input space of encoder and decoder models, we take inspiration from Prefix Tuning.

Our strategy for layering prompt influence through several levels of representation is guided by the structure of P-Tuning v2.

Modular attention prompts that mimic low-rank influence are used where necessary to simulate LoRA-style injection, especially in Mistral-7B and Gemma.

UPG seeks to provide high adaptability and performance without requiring model retraining by combining the advantages of these prompt tuning techniques, even when alternating between essentially different model architectures (e.g., BERT vs. Mistral-7B).

Reverse prompt engineering (analyzing outputs to infer the implicit prompts).

The process of examining a language model's outputs in order to deduce or recreate the implicit prompts or input patterns that most likely generated them is known as "reverse prompt engineering." Reverse prompt engineering identifies the underlying structures, strategies, or cues that the model uses for generation by working backward from

the model's responses, in contrast to traditional prompt engineering, which creates inputs to direct model behavior. In this way, researchers can better understand and optimize prompt design across various LLM architectures by uncovering hidden biases, recurring patterns, or useful prompt templates embedded within the model's latent space.

Cross-model prompt adaptation

The process of transferring or altering prompts created for one language model architecture to ensure their efficacy when utilized with another model is known as cross-model prompt adaptation. Directly applying prompts across models frequently leads to suboptimal performance because of the architectural and training differences between models (e.g., the autoregressive decoder structure of GPT versus the bidirectional encoder of BERT). By altering or reformatting prompts to conform to the input representation and processing mechanisms of the target model, cross-model prompt adaptation techniques seek to close this gap. This facilitates knowledge transfer and prompt reuse, increasing multi-model workflow efficiency and assisting in the creation of universal prompt generation techniques.

3. System Architecture

The phases of the prompt retrieve data pipeline are as follows:

1. Data Collection: Real-world interactions or QA platforms, such as Kaggle forums, are the source of datasets.
2. Prompt Rewriting: To produce pairs, output text is generated using pre-existing prompts.
3. Model Training: Using models like Mistral 7B for inference or refining models like Sentence-T5.
4. Evaluation: To confirm rapid recovery, metrics like cosine similarity, Jaccard index, BLEU, and ROUGE scores are employed.

A diagram here showing the flow of prompt → model → output → reverse → recovered prompt

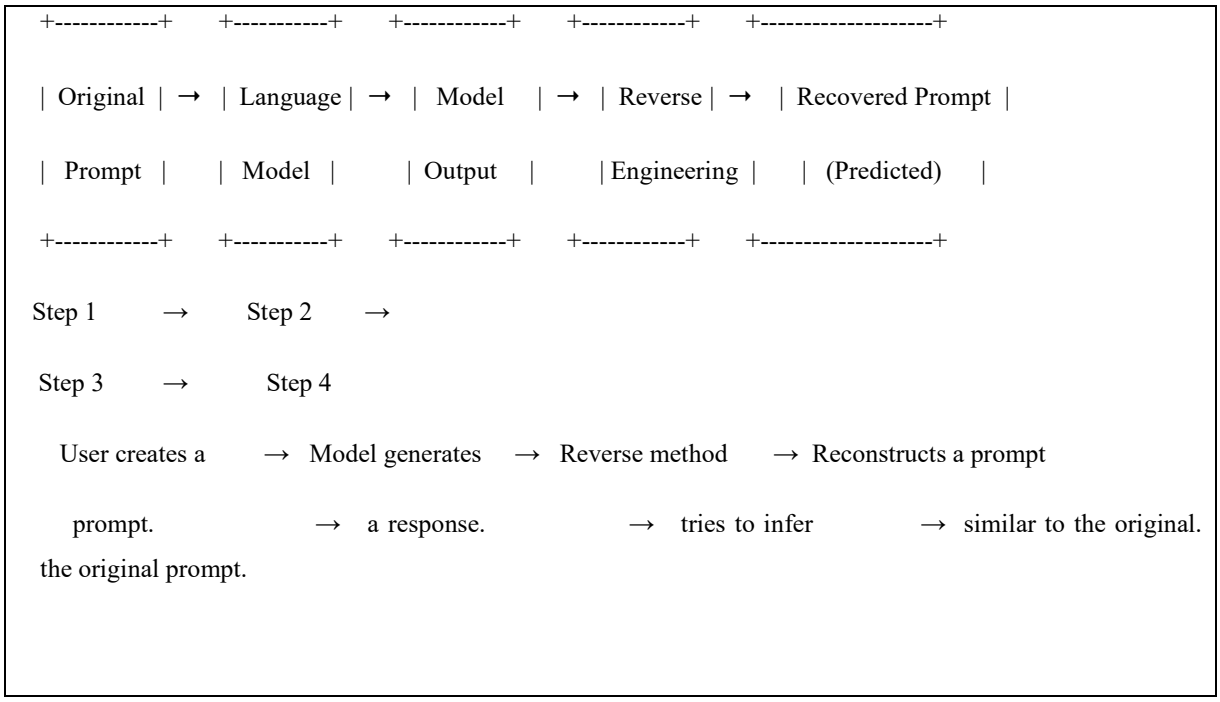
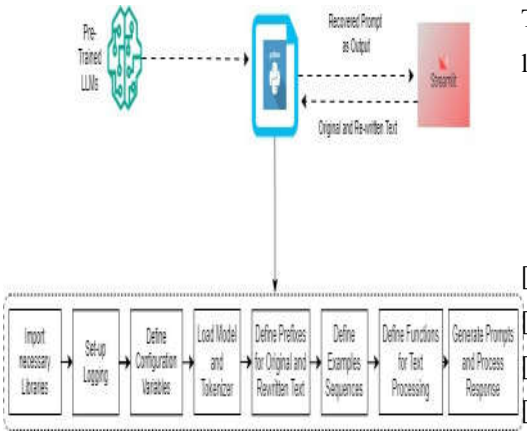


Diagram Structure: Rapid Recoveries and Reverse Engineering Process



The pipeline can be seen as a left-to-right flowchart with the phases labeled as follows:

[1. Model Output Gathering]
[2. Feature Extraction]
[3. Identifying Patterns]
[4. Quick Template Reconstruction]
[5. Timely Improvement & Assessment]

An explanation in text of each phase:

1.Model Output Gathering

Gather LLM outputs from a variety of tasks, such as creative writing, summarization, and quality assurance.

A wide range of styles and structures should be represented in the outputs.

2.Feature Extraction

Examine textual components like formatting, stylistic elements, keyword frequency, and sentence structure.

Utilize NLP tools to extract latent linguistic features, such as dependency parsing, POS tagging, and embeddings.

3.Finding Patterns

Employ attention heatmaps, embedding analysis, or clustering to find recurrent syntactic or semantic patterns.

Determine the implicit structure (e.g., instructional phrasing, tone) that governs the model's behavior.

4.Rapid Reconstruction of the Template

Rebuild a generic prompt structure that might yield comparable results for various models.

For instance, "Write a thorough answer to the question: [input]."

5.Quick Improvement & Assessment

Test the reconstructed prompts on several models, such as Sentence-T5, Mistral-7B, and BERT.

Assess with human input and performance metrics (e.g., coherence scores, BLEU).

4. Materials and Methods

- **Data Generation**: Forum messages and user queries were used to simulate input-output interactions. Noise and irrelevant content were removed. Each data point included an ID, original prompt, and rewritten text.
- **Tokenization**: Text was tokenized using standard NLP libraries, preparing it for training.

- **Model Training**: Mistral-7B was used to generate text, and Sentence-T5-base was trained to recover original prompts.
- **Evaluation Metrics**:
    - **Cosine Similarity**
    - **Sharpened Cosine Similarity**
    - **Jaccard Similarity**
    - **BLEU & ROUGE scores**
    - **Perplexity**
    - **Human Evaluation (A/B Testing)**

4.1 **Data Generation**

Data was taken from forum posts and user inquiries, especially from technical and Q&A platforms like Kaggle, in order to replicate authentic prompt-response interactions. Every entry was carefully chosen to guarantee its relevance and clarity. During preprocessing, off-topic content, spam, and noise were eliminated. In order to create structured input-output pairs appropriate for reverse prompt engineering tasks, the final dataset contained unique identifiers, the original user-generated prompts, and their corresponding rewritten or generated outputs.

4.2 Tokenization

Standard natural language processing (NLP) libraries, such as spacy and Hugging Face's tokenizers, were used to tokenize all text data. To ensure consistency across various model architectures, tokenization was used to prepare the data for both the generation and prompt recovery stages.

4.3 Training of Models

Two main models were used in the pipeline:

In order to produce outputs based on different input prompts across tasks like question answering, summarizing, and creative writing, Mistral-7B, an autoregressive decoder-only LLM, was used.

A sentence-level encoder-decoder model called Sentence-T5-base was optimized to carry out prompt recovery, which entails reassembling the initial prompt from the output produced by Mistral-7B. The input-output pairs produced during the data generation stage were used to train this model.

4.4 Measures of Evaluation

A combination of automated and human-centric evaluation metrics was employed to evaluate the quality and accuracy of the recovered prompts:

Cosine Similarity: Evaluates the original and recovered prompts' vector-based semantic similarity.

Cosine sharpened Similarity: An improved version that prioritizes word embeddings with high confidence to cut down on noise.

To evaluate surface-level similarity, Jaccard Similarity computes the overlap of token sets.

ROUGE AND BLEU Scores: Assess n-gram overlap to gauge the quality of the summarization and textual fidelity.

Perplexity: Evaluates the recovered prompts' naturalness and fluency when they are used again for text generation.

Human Evaluation (A/B Testing): Using standards like task alignment, coherence, and relevance, human annotators evaluated outputs produced by the original prompts and those recovered

Formulas for Evaluation Metrics

1. Similarity in Cosines

The cosine of the angle between two vector representations (such as sentence embeddings) of the initial prompt (A) and the recovered prompt (B) is used to calculate cosine similarity:

Cosine Similarity = $A \cdot B \| A \| \cdot \| B \|$

Cosine Similarity = ‖A‖·‖B‖ A·B

Where:

The dot product of the two vectors is $A \cdot B$ A·B.

$\| A \|$and $\| B \|$ are the Euclidean norms, or magnitudes.

2. Cosine Similarity Sharpened

By applying a sharpening function (·) f(·) to the vectors, this variation of cosine similarity highlights higher-confidence token or feature embeddings:

Sharpened Cosine Similarity = $f\ (A) \cdot f\ (B)\ \| f\ (A)\ \| \cdot \| f\ (B)\ \|$

Sharpened Cosine Similarity = ‖f(A)‖·‖f(B)‖ f(A)·f(B)

Softmax or power transformation (e.g., $(x) = xp$ f(x)=x p, with p> 1 p>1) is a popular sharpening function that is applied element-wise to vector components.

3. Similarity to Jaccard

The overlap between two sets of tokens is measured by Jaccard similarity:

Jaccard Similarity = $|A \cap B| |A \cup B|$

Jaccard Similarity = |A∪B||A∩B|

Where:

Sets of tokens from the original and recovered prompts are denoted by A and B.

∩∩ indicates an intersection.

∪∪ indicates a union.

4. **Bilingual Evaluation Understudy (BLEU) Score**

The accuracy of n-gram overlaps between a reference (original) prompt and a candidate (recovered) is assessed by BLEU:

BLEU = BP·exp ($\sum n = 1 N w n \log pn$)

BLEU=BP·exp(n=1 ∑ N w n logp n)

Where:

The modified n-gram precision is represented by p n.

The weight for n-gram level nn is wnw n (usually 1 N N 1).

The brevity penalty, or BP, is used to penalize brief outputs.

5. **Recall-Oriented Understudy for Gisting Evaluation, or ROUGE Score**

The longest common subsequence (LCS) is the foundation of ROUGE-L:

ROUGE-L = (1 + β 2)

· Precision · Recall Recall + β 2 · Precision

ROUGE-L = Recall + β 2 ·Precision (1+β 2) ·**Precision·Recall**

Where:

Accuracy = LCS

Candidate Length Candidate Length LCS

LCS = Recall

Duration of the reference

LCS

Usually, ββ is set to favor recall (e.g., β = 1.2 β=1.2).

6. **Confusing/Perplexity**

A model's ability to predict a sequence is measured by its perplexity. For a sequence $W = w1, w2,..., wN$

W=w 1,w 2,...,w N :

Perplexity = exp ($-1N\sum i = 1N \log P (wi|w 1i-1)$)

Perplexity = exp(− N 1)

i=1 ∑ N logP(w i |w 1 i−1))

Better prompt fluency and predictability are indicated by lower perplexity.

Results :

We utilized both human A/B comparisons and automated metrics (cosine similarity, sharpened cosine, Jaccard, BLEU, ROUGE, and perplexity) to evaluate the prompt-recovery pipeline (Mistral-7B for generation; Sentence-T5-base for recovery). Overall, for well-structured inputs and question-answering/summarization tasks, the recovery model accurately replicated the original prompts; however, performance reduced for outputs that were noisy or highly ambiguous.

STATISTICAL HIGHLIGHTS:

Semantic similarity: The recovered prompts had a cosine similarity > 0.85 (85%) for clear, structured queries.

Surface overlap and n-gram fidelity: Jaccard similarity was highly correlated with vocabulary overlap between the original and recovered prompts; structured or templated prompts yielded higher BLEU and ROUGE scores than free-form conversational prompts.

Model differences showed that prompt-recovery fidelity is not solely determined by parameter count, as lightweight dense models (e.g., Phi-2 in our experiments) outperformed expectations on recovery tasks.
Several recovery estimators combined improve robustness on noisy outputs, according to group tests.

Task-level productivity.

Highest recovery fidelity was observed in question answering and summarization; recovered prompts maintained task intent and generated outputs that were comparable to those from the original prompts (based on human A/B testing).

Writing and creative assignments as well as unstructured discussion: Lower BLEU/ROUGE and n-gram overlap; semantic similarity was still moderate, but recovered prompts frequently needed post-editing to match the original wording.

Human assessment.
When the original prompt was unclear or contained unnecessary context, annotators rated recovered prompts as less faithful. Human A/B comparison showed that outputs produced from recovered prompts were generally regarded as task-aligned and coherent for structured tasks.

5. Conclusion

In order to recover and optimize input prompts for large language models (LLMs), this study highlights the significance of reverse prompt engineering. We close the gap between model behavior and user inputs by methodically examining model outputs to reconstruct the implicit prompts. This helps to improve performance, interpretability, and robustness in addition to improving our understanding of how LLMs interpret and react to various prompts. Prompt recovery methods will be essential to guaranteeing accountability, transparency, and responsible AI development as LLMs are used in more sensitive and important applications.

6. Future Scope

The suggested system provides a number of encouraging avenues for further research:

*Model Generalization Across LLMs: To enhance prompt recovery performance and generalize across model families, we intend to test a wider variety of large language models (LLMs), including more recent encoder-decoder and decoder-only architectures.

*Deployment Pipeline: To operationalize the prompt recovery system and facilitate scalable integration into practical applications, a comprehensive deployment pipeline will be created.

*Interactive User Interface (UI): React or Streamlit are two examples of the tools that will be used to create an interactive UI. Users will be able to view the recovered prompt in real time, input text, and call the underlying model API.

*Extended Use Cases: In addition to natural language tasks, we will investigate applications for prompt recovery in fields like

#Generation of code

#Case study

#Drafting documents

#Refactoring and summarizing code

#Combining Retrieval-Augmented Generation (RAG) with Integration:

Our goal is to integrate the system with a RAG framework in order to improve the contextual grounding and accuracy of recovered prompts. This will allow the model to obtain pertinent background information from outside sources prior to immediate reconstruction.

References:

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert:Pre-training of deep bidirectional ltransformers for language understanding.*arXivpreprintarXiv:1810.04805*, 2018.

[2] YinhanLiu, MyleOtt, NamanGoyal, JingfeiDu, MandarJoshi, DanqiChen, OmerLevy, MikeLewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: Arobustly optimized bert pretraining approach.*arXiv preprint arXiv:1907.11692*, 2019.

[3] [3] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. **Mistral 7B**. *arXiv preprint arXiv:2310.06825*, 2023.GemmaTeam,Thomas Mesnard, Cassidy Hardin, Robert Da dashi, Surya Bhupatiraju, Shreya Pathak,LaurentSifre,MorganeRivière,MihirSanjayKale,JulietteLove,etal.Gemma:Open models based on gemini research and technology.*arXiv preprint arXiv:2403.08295*, 2024.

[4] JianmoNi,GustavoHernandezAbrego,NoahConstant,JiMa,KeithBHall,DanielCer,and Yinfei Yang.Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

[5] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha.Asystematicsurveyofpromptengineeringinlargelanguagemodels:Techniquesand applications.*arXiv preprint arXiv:2402.07927*, 2024.

[6] Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Tong Xiao, and JingboZhu.Efficient prompting methods for large language models: A survey.*arXiv preprint arXiv:2404.01077*, 2024.

[7] Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky,andAramGalstyan.Promptperturbationconsistencylearningforrobustlanguage models.*arXiv preprint arXiv:2402.15833*, 2024

[8] WangChao,JiaxuanZhao,LichengJiao,LinglingLi,FangLiu,andShuyuanYang.Amatch madeinconsistencyheaven:whenlargelanguagemodelsmeetevolutionaryalgorithms.*arXiv preprint arXiv:2401.10510*, 2024

[9] Weize Kong, Spurthi Amba Hombaiah, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Prewrite: Prompt rewriting with reinforcement learning.*arXiv preprint arXiv:2401.08189*, 2024.

[10] ChengLi,MingyangZhang,QiaozhuMei,WeizeKong,andMichaelBendersky.Learningto rewrite prompts for personalized text generation.In *Proceedings of the ACM on Web Conference 2024*, pages 3367–3378, 2024.

[12]YiJiangXiChenHaoyuWangShoulingJiYongYang,XuhongZhangandZonghuiWang.Prsa:Promptreversestealingattack sagainstlargelanguagemodels.*https://arxiv.org/pdf/2402.19200*, 2024.

[13]RonenEldanJohannesGehrkeEricHorvitzEceKamarPeterLeeYinTatLeeYuanzhi Li Scott Lundberg Harsha Nori Hamid Palangi Marco Tulio Ribeiro Sébastien Bubeck,VarunChandrasekaranandYiZhang.Sparksofartificialgeneralintelligence:Earlyexperiments with gpt-4.*https://github.blog/ 2023-07-17-prompt-engineering-guide-generative-ai-llms/.*, 2023.

[14] Alberto D Rodriguez, Katherine R Dearstyne, and Jane Cleland-Huang.Prompts matter: Insightsandstrategiesforpromptengineeringinautomatedsoftwaretraceability.In*2023IEEE 31stInternationalRequirementsEngineeringConferenceWorkshops(REW)*,pages455–464. IEEE, 2023.

[15] Mayee Chen Laurel Orr Neel Guha Kush Bhatia Ines Chami Christopher Ré Simran Arora, Avanika Narayan.Ask me anything: A simple strategy for prompting language models. *https://openreview.net/pdf?id=bhUPJnS2g0X*,