IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

RECOVER PROMPT USING REVERSE ENGINEERING

Nagarjun L [1], Nishant Manjunath Hegde [2], Pradyumna Bhat [3], Tarun B P [4],

Mr. Abhinand B V [5]

Students, Information Science and Engineering, SJB Institute of Technology, Bengaluru, India [1-4]

Assistant Professor, Information Science and Engineering, SJB Institute of Technology, Bengaluru, India [5]

Abstract: Large Language Models (LLMs) have significantly advanced natural language processing tasks but are heavily reliant on well-crafted prompts for optimal performance. However, manual prompt engineering is time-consuming, often sub-optimal, and lacks robustness to various perturbations in input prompts. Additionally, there is a growing need to address the security implications of prompt extraction attacks. Existing research has proposed automated methods for prompt engineering, ranging from rewriting under-optimized prompts to generating high-quality human-like prompts from scratch. Despite these advancements, challenges persist in achieving prompt effectiveness, robustness, and security.

Keywords: NLP, LLM, Prompt Engineering, Prompt Recovery, Zero shots, Few shots, Chain of Thoughts.

I. INTRODUCTION

Prompt engineering is the way of crafting prompts to elicit desired outcomes from large language models or AI systems. The core idea is that the quality of the response is intricately linked to the quality of the question posed. By strategically designing prompts, one can influence the outputs and enhance the system's performance and usefulness.

Reverse prompt engineering is an intriguing approach in natural language processing (NLP) that reverses the traditional prompt-based generation process. Instead of crafting prompts to generate desired responses, reverse prompt engineering involves analyzing a model's outputs to predicts the prompts likely used to produce them. This technique has gained attention for its potential to reveal insights into how language models function, uncovering their biases, limitations, and capabilities.

By reverse engineering prompts, we can gain a deeper understanding of NLP systems, improve their performance, and mitigate risks like biased or harmful outputs. Reverse prompt engineering opens up a new avenue for exploring and interpreting the behaviors of language models. It sheds light on the black box of AI-generated text, offering valuable insights for researchers, developers in the field of natural language processing. Prompt Rewrite refers to modifying or rephrasing the initial prompt given to a language model to guide it towards producing desired outputs. This technique is commonly used in NLP tasks where the quality and relevance of the generated text are critical.

Reverse prompt engineering has several valuable applications across various fields:

Understanding Model Behavior:

By analyzing inferred prompts, we can gain insights into how language models generate text, interpret model decisions, identify biases, and improve model robustness.

Bias Detection and Mitigation:
This can uncover biases encoded in language models by identifying prompts that lead to biased outputs. Once detected, developers can adjust training data or fine-tune model parameters to mitigate these biases.

Improving Model Performance:
Analyzing inferred prompts can provide feedback on the effectiveness of different prompts, guiding the design of better prompts for specific tasks and enhancing model performance.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

Quality Control and Evaluation:
Reverse prompt engineering can assess the quality of generated text by examining the consistency and coherence of inferred prompts across multiple outputs, facilitating automated quality control and evaluation of language models.

Content Filtering and Moderation:
Understanding prompts that produce undesirable or harmful outputs allows developers to design better content filtering and moderation systems, aiding in the detection and prevention of harmful information dissemination.

Creative Writing and Content Generation:
In creative writing or content generation, reverse prompt engineering can reveal the underlying patterns and structures used by language models, inspiring new approaches for generating more engaging and diverse content.
Popular LLMs are BERT[1], RoBERTa[2], Mistral 7B[3], Gemma[4], Sentence-T5[5]. We will be using combinations of these LLMs for our case study.

Use Cases:
• Building prompt library: For a given task like java migration, we can build optimized prompts to handle and fix migration issues.
• Enhancing Customer Support Systems: By recovering and analyzing effective prompts, customer support bots can be fine-tuned to provide more accurate and helpful responses, improving customer satisfaction.
• Improving Code Generation and Debugging tools: Reverse engineering prompts can help in finetuning models to generate more accurate code snippets and provide better debugging suggestions, aiding developers.
• Assisting writers in creative writing and storytelling: Reverse engineering effective storytelling prompts can help AI systems provide better writing assistance, generating compelling and creative narratives.

II. LITERATURE SURVEY

Note: We have analysed various approaches available to get best prompts pertaining different use cases including template generation, text style change, AI agents etc. We have mentioned these approaches and their findings in below sections.

2.1 A Systematic Survey of Prompt Engineering in Large Language Models Prompt engineering is an indispensable technique for extending the capabilities of large language models (LLMs) and vision-language models (VLMs).It leverages task-specific instructions, known as prompts, to enhance model efficacy without modifying the core model parameters rather than updating the model parameters, prompts allow seamless integration of pre-trained models into downstream tasks by eliciting desired model behaviors solely based on the given prompt.

The paper addresses the gap in understanding diverse prompt engineering methods and techniques. Despite the success of prompt engineering in various applications, there remains a lack of systematic organization and comprehension of these methods. Results: The paper provides a comprehensive overview of prompt engineering techniques, their applications, and associated models.

Conclusions: The systematic analysis presented in this survey enables a better understanding of the rapidly developing field of prompt engineering. [6]

2.2 A Universal Prompt Generator for Large Language Models
UniPrompt is a technique for generating very good quality almost human like prompts for large language models.It is far better than current techniques present and it shows improvements over zero shot GPT-4. It performs well on bit easier tasks and helps in understanding prompt engineering importance in large language model enhancements and performance. Using a variety of section-wise prompt datasets, the technique trains an auxiliary language model and then uses direct preference optimization to optimize it. Implementation details, performance analysis, and recommendations for further research are included in the following paper. All things considered, UniPrompt is a promising method that enhances language model performance and automates prompt production. UniPrompt: Training a Universal Prompt Generator. Given a large language model (LLM) and a task, the goal is to generate a prompt for solving that task. We call this LLM as the solver LLM. To generate prompts, we use an auxiliary language model, typically smaller than the solver LLM whose weights can be fine tuned. For each task, we assume an accuracy metric on the test set that needs to be maximized.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

We train our model in two stages:
1) Supervised fine tuning over a dataset of exemplar prompts.
2) Optimization based on pairwise feedback of generated prompts.

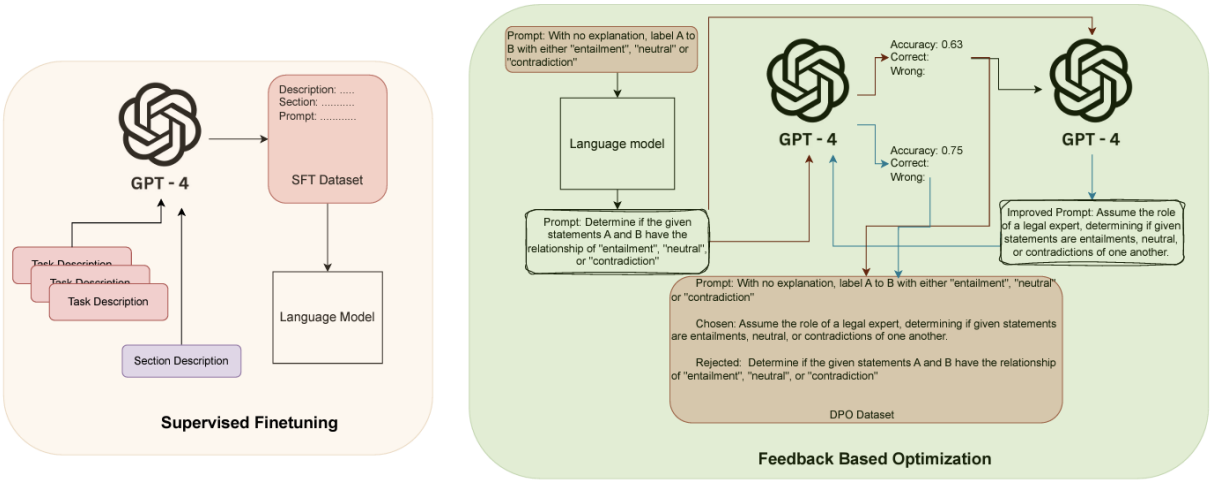


Figure 1: Training of UniPrompt occurs in two stages, first Supervised Finetuning, Second stage, Feedback Based Optimization.

Supervised Fine tuning:
Human-generated prompts tend to be structured having multiple sections like Introduction, Examples, Motivation, Tricks, Output Format, etc. To induce the ability of structured prompt generation in a smaller language model, we curate a section-wise dataset of around 12,000 task-prompt pairs.

The tasks for training dataset creation were taken from tasksource library that contains around five hundred classification tasks. We extract the task description from tasksource-instruct, which contains tasksource dataset recasted with instructions. For instance, the task description for BIG-bench Entailed Polarity task is, "Given a fact, answer the following question with a yes or a no". The dataset provides diverse tasks and their short description, but not the human-generated prompts for each task. To approximate human-generated prompts, we use GPT-4 as a teacher model. Initial experiments showed that generating the entire prompt from a smaller language model is difficult. Hence, we decide to fine tune the language model to produce section-wise content for a given task.[7]

Results:

We use the following configuration. Solver LM: We present results with GPT-4 and GPT-3.5 as the solver LM.

Auxiliary LM that generates prompts:
LLAMA 7B model, which we fine tune using LoR Aadapters. For the training hyper parameters Feedback LM: We use the state-of-the-art Language model, GPT-4, due to its ability to provide human-like detailed feedback.

Datasets:
MedQA, CauseEffectPairs, Ethos

Analysis:
Comparison with human prompts. Comparing the performance of the UniPrompt generated prompts with the Human-optimized prompts We observe that UniPrompt (SFT) & UniPrompt (SFT+DPO) prompts show significant improvements on the Ethos and Causal datasets. When GPT-3.5 is used as the solver LM UniPrompt (SFT) outperforms the human-optimized prompts on the Causal dataset by +10.6%.

Comparison with zero-shot GPT-4 The UniPrompt (SFT) & UniPrompt (SFT+DPO) models show significant improvements compared to using zero-shot GPT-4 across all three datasets. When GPT-3.5 is used as the solver LM UniPrompt (SFT) outperforms the zero-shot GPT-4 prompts on the MedQA by +3.3%, Ethos by +8.8% and Causal by +15.6%.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.14551

Comparison with existing methods When compared to the existing prompt optimization techniques, UniPrompt shows significant improvement (See Table 1). When GPT-3.5 is used as the solver LM, UniPrompt outperforms the best baseline by 19.2% on Ethos, and 15.6% on casual datasets. Using GPT-4 as the solver LM UniPrompt outperforms baselines by 1.5% on MedQA, 10.7% on Ethos and 2.8% on causal datasets.

Discussion:
In this work, we addressed the challenging task of generating human-like structured prompts from scratch, which distinguishes our approach from previous works that primarily focused on prompt editing. We approach the problem of generating structured prompts by training a smaller language model on a novel task-prompt dataset. Through extensive experiments conducted on three diverse datasets, our method consistently outperformed existing techniques, demonstrating its effectiveness in improving the quality and relevance of generated prompts.

2.3 Prompts Matter
Large language models demonstrate outstanding potential in automated traceability. They cope with challenges that previous strategies are faced with and provide new opportunities. However, it remains uncertain how to effectively use LLMs for automated traceability. In this approach, we explore prompt engineering—extracting link predictions from an LLM. They share insights on how to construct effective prompts and propose strategies for enhancing traceability links using LLMs. The overall aim is to inspire and guide future researchers and engineers in leveraging LLMs to advance automated traceability.

Problem Definition:
This paper addresses the challenge of automated software traceability, specifically, focusing on utilizing LLMs for enhancing traceability link predictions. Previous methods had limitations, and the best practices for LLM utilization for traceability were not known. The problem is how to effectively construct prompts, input queries for LLMs, so that they generate accurate traceability links. Here, we propose a systematic approach to prompt engineering, with an emphasis on the importance of well-constructed prompts. They investigate strategies on constructing effective prompts that could lead to better link predictions. Here, we emphasize the role of prompts in enhancing LLM-based traceability and provide insights to practitioners and researchers on effective prompt engineering. Most likely, the study involves fine-tuning LLMs such as GPT-3 or similar models with domain-specific data about traceability in software.[8]

Strategies may include:
• Crafting informative prompts that carry relevant context.
• Experimenting with prompt variations, different phrasings, keywords.
• Analyzing LLM responses to understand their behavior.

2.4 A Simple strategy for prompting language models
To improve the performance of large language models (LLMs) in the prompting paradigm by using multiple imperfect prompts and aggregating their outputs. Here we discuss the challenges of prompt design, prompt collection at scale, and prompt aggregation, with the goal of enabling open-source LLMs to match or exceed the performance of few-shot LLMs on various benchmarks.

Efficient Prompting:
AMA offers a streamlined prompting approach, combining multiple imperfect prompts to generate high-quality outputs, reducing the need for painstaking prompt crafting.

QA Prompt Effectiveness:
QA prompts, which encourage open-ended generation, are found to be more effective than restrictive prompts. AMA adopts this format, generating multiple questions per input to gather diverse perspectives.

Weak Supervision Integration:
By using weak supervision to aggregate noisy predictions from multiple prompts, AMA achieves improved performance, with an average uplift of 10.2% over the few-shot baseline across various model families and sizes.

GPT-J-6B Performance:
GPT-J-6B, employing the AMA strategy, surpasses few-shot GPT3-175B on 15 of 20 popular benchmarks, showcasing its effectiveness in enhancing LLM performance across tasks and architectures.
To improve the performance of large language models (LLMs) in the prompting paradigm.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.14551

These solutions include:

- Analyzing the frequency of different prompt patterns in the existing data set.
- Emphasizing the importance of prompt design, particularly for tasks that require specific knowledge or common-sense reasoning.
- Exploring prompt aggregation to leverage the strengths of different prompts and mitigate their individual weaknesses.
- Aggregating the outputs of multiple prompts using techniques like majority voting or weighted averaging.
- Fine-tuning and transfer learning to improve LLM performance on specific tasks.
- Establishing standardized evaluation metrics and benchmarks to compare the performance of different LLMs.
- Developing open-source implementations to facilitate prompt-based training and evaluation. These solutions aim to enhance the effectiveness, adaptability, and efficiency of LLMs in the prompting paradigm.

Algorithms Used:

• The AMA End-to-End Procedure (Algorithm 1)
• AMA Aggregation Method (Algorithm 2)

The AMA End-to-End Procedure outlines the procedure of the Ask Me Anything (AMA) approach, which involves prompting the language model with question() prompts and answer() prompts to generate questions and answers based on the input. The AMA Aggregation Method is used for prompt aggregation, learning the dependency structure among prompts and estimating their accuracies to generate multiple predictions for each input.

Additionally, the paper mentions the use of structure learning to recover the dependency structure of prompts and their accuracies, as well as weak supervision techniques for aggregating prompt predictions.These algorithms and techniques contribute to improving the performance of large language models in the prompting paradigm.

Dataset Used:
1. The Pile corpus, 2. Natural Questions, 3. BoolQ, 4. DBPedia, 5. SuperGLUE, 6. DROP QA Results: The AMA approach improves the performance of large language models (LLMs) in the prompting paradigm. It outperforms the few-shot baseline on various benchmark tasks and enables smaller LLMs to achieve competitive performance compared to larger models. Prompt aggregation and weak supervision play crucial roles in enhancing the accuracy of the models. However, there are limitations in tasks that require domain-specific knowledge or where answers are not seen during pretraining.[9]

Conclusion: The AMA approach provides a simple and effective strategy for improving LLM performance and reducing reliance on perfect prompts.

2.5 OpenPrompt
Pre-trained language models (PLMs) have been widely proven to be effective in natural language understanding and generation, ushering in a new era of modern natural language processing (NLP). In the early stage of this revolution, a standard approach to adapt PLMs to various specific NLP tasks is the pretraining-fine tuning paradigm, where additional parameters and task-specific objectives are introduced in the tuning procedure.

Design and Implementation:
Prompt-learning is a comprehensive process that combines PLMs, human knowledge, and specific NLP tasks. Keeping that in mind, the design philosophy is to simultaneously consider the independence and mutual coupling of each module. As illustrated in Figure 2, OpenPrompt provides the full life-cycle of prompt-learning based on PyTorch. In this section, we first introduce the combinability of OpenPrompt, and then the detailed design and implementation of each component in OpenPrompt.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

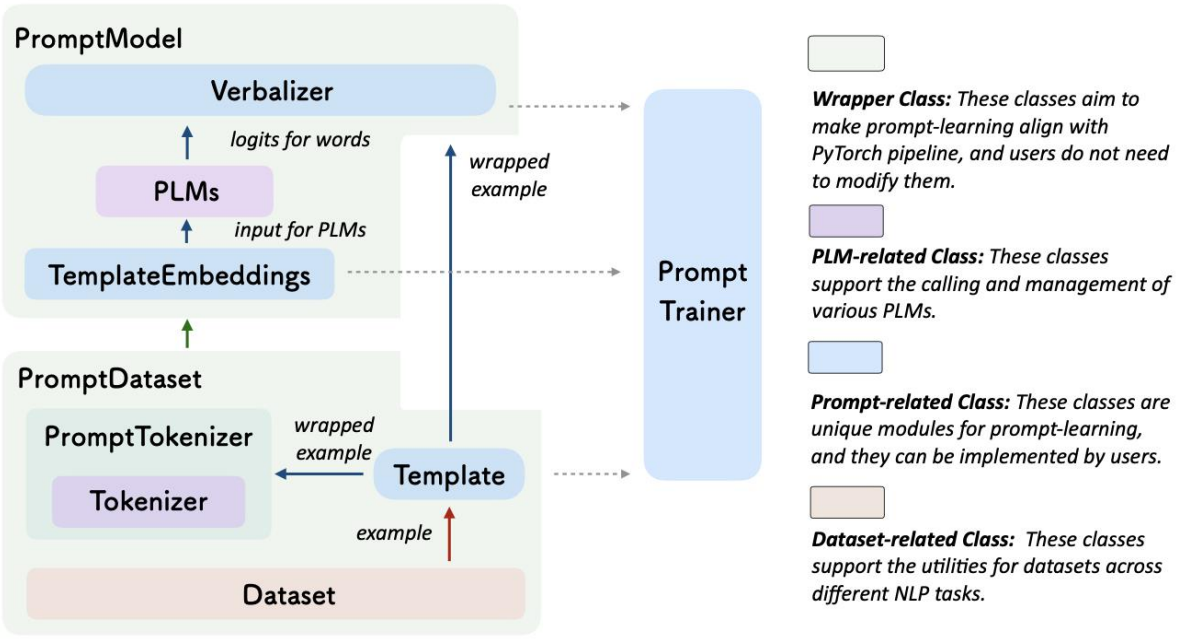


Figure 2: The overall architecture of OpenPrompt.

Conclusion and Future Work

We propose OpenPrompt, a unified, easy-to-use and extensible toolkit for prompt-learning. Open-Prompt establishes a unified framework with clearly defined blocks and flexible interactions to support solid research on prompt-learning. At the application level, OpenPrompt could facilitate researchers and developers to effectively and efficiently deploy prompt learning pipelines. In the future, we will continue to integrate new techniques and features to OpenPrompt to facilitate the re- search progress of prompt learning.[10]

2.6 Prompt Perturbation

In this approach, we use perturbation in constructing the prompt. This is useful for voice-controlled smart personal assistants like Amazon Echo and Google Home, which have flourished in recent years, enabling goal-oriented conversations and aiding tasks like setting reminders, checking weather, controlling smart devices, and online shopping. A core capability of these systems is to perform accurate and robust intent classification (IC) and slot filling (SF).

Prompts Construction:

The standard prompts employed during the instruction fine-tuning process with LLMs typically involve presenting both the input context and its corresponding target output in a paired structure. The LLMs are then trained to generate the target output based on the input context. The primary objective here is to fine-tune the models' parameters aiming to minimize prediction errors and improve their ability to generate accurate and contextually appropriate responses.

The paper proposes constructing several prompt formats for IC-SF tasks as detailed in Appendix Table 1. A robust model aims to convert all utterances, with or without meaning-preserving perturbations, into correct hypotheses. To evaluate model robustness in IC-SF tasks, the approach is to employ different types of perturbations: acronyms, synonyms, and paraphrases, covering both word-level and sentence-level perturbations aligned with real-world application scenarios as mentioned here. The figure below shows some examples of these perturbations.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

Original Utterances	Oronyms Perturbations
review all alarms	review aul alarms
when is the event going to start	wynn is the event going to start
Original Utterances	**Synonyms Perturbations**
email to new contact	email to novel contact
pink is all we need	pink is all we ask
Original Utterances	**Paraphrasing Perturbations**
tell me the weather this week	whats the weather forecast for this week
how old is mariah carey	what is the age of mariah carey

Figure 3: The overall architecture of OpenPrompt.

This approach tries to evaluate and improve the robustness of LLMs in generating structured hypotheses, such as IC-SF tasks. It first proposes a sentinel-based structured prompt format for instruction fine-tuning LLMs, resulting in comparable performance to state-of-the-art discriminative models. Next, it evaluates the robustness of LLMs under various prompt perturbations, i.e., synonyms, oronyms, and paraphrases. The results indicate that LLMs are vulnerable to these perturbations, with an average performance drop rate of 13.07 percent in IC accuracy and 22.20 percent in SF F1-score. The paper then proposes two mitigation strategies, i.e., perturbation consistency learning and data augmentation, aiming to improve model robustness. These methods can recover up to 59 percent performance drop in the IC task and 69 percent in the SF task, making the resulting LLMs more robust to prompt perturbations. Finally, the paper's findings show that PPCL surpasses the basic data augmentation method, achieving superior performance with just 10 percent of the augmented datasets, thereby exhibiting enhanced scalability.[11]

2.7 Efficient Prompting Methods for Large Language Models

Large Language Models (LLMs) are integral to natural language processing (NLP) tasks, and prompts play a crucial role in facilitating effective human-computer interaction. While LLMs have demonstrated significant potential, their use incurs substantial costs in both computational resources and financial overheads. This underscores the importance of developing efficient prompting methods to mitigate these challenges. Efficiency can be enhanced from three key perspectives, considering both financial and human
resources:

- Inference acceleration
- Reduction in memory consumption
- Automatically optimized prompts

The first two goals can be achieved through prompt compression, whereas the third can be realized by automatic prompt optimization based on prompt engineering rather than manual design. To our best knowledge, there is a major difference in the literature regarding merging of best prompt methods.

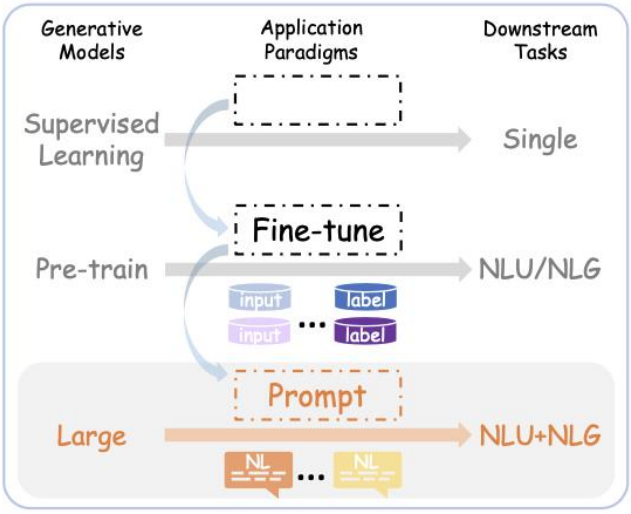


Figure 4: NLP paradigm shift

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

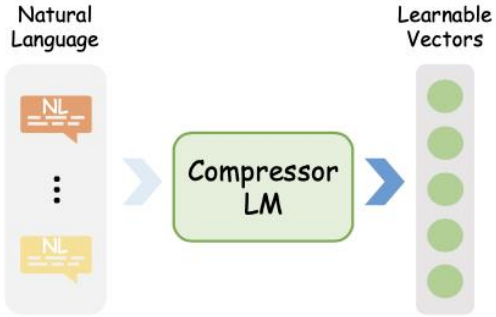


Figure 5: Hard and soft prompts

Prompt Paradigm The emergence of prompting is closely linked with the development of Pre-trained Language Models (PLMs) and the advancement of Large Language Models (LLMs). Types of Prompts The primary aim of prompting is to achieve effective few-shot learning, thereby avoiding the unnecessary resource consumption associated with full parameter fine-tuning. Prompt expressions can be categorized into two main types: discrete natural language prompts (hard prompts) and continuous learnable vectors (soft prompts), as illustrated in Figure 5.

Hard Prompts Hard prompts are specific prompts designed for generative language models like those in the GPT series, notable for their suitability with these models. They have two main characteristics:

Positive Standpoint: They utilize the extensive pre-training data within LLMs, allowing users to interact effectively with the model using natural language, often resulting in helpful responses.

Negative Standpoint: Since many LLMs are closed-source, users are limited to interacting through API calls, without direct access to the model's parameters. Thus, hard prompts become essential for effective utilization.
Soft Prompts Soft prompts are used in NLP tasks to enhance the training efficiency of LLMs. Initially appearing as adapters, prefixes, or unexplainable vectors, soft prompts have evolved into learnable vectors that are updated along with the neural network parameters during training. Unlike hard prompts, which are direct instructions or templates, soft prompts are typically vector representations derived from existing hard prompts rather than developed from scratch. They offer benefits such as efficient training and better comprehension for LLMs, with efforts made to compress longer hard prompts into shorter soft prompts to further improve efficiency.[12]

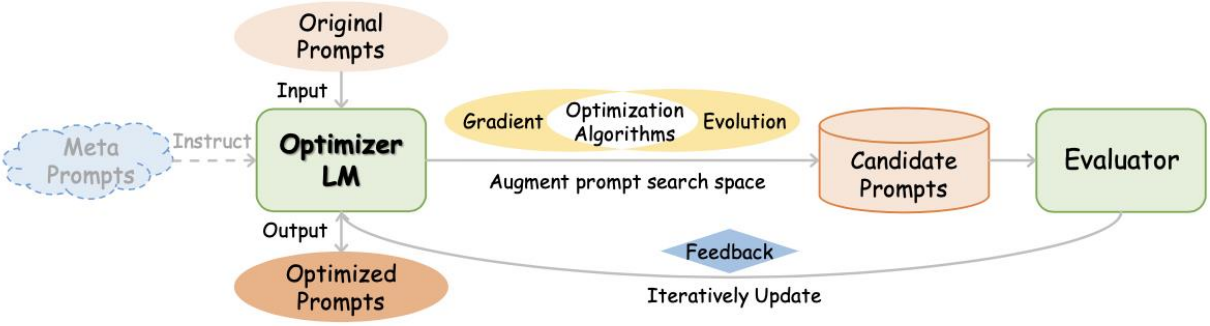


Figure 6: Process of automatic prompt optimization

Prompting with efficient design
Prompt content has been becoming more complex leading to the concept of "Prompting with Efficient Design". Prompt engineering (PE), a type of automatic optimization, was receiving an increasingly large amount of attention as manual-design prompt-like methods lost their status in history and gradient-based prompt tuning methods became irrelevant to the closed-source LLMs.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.14551

More concretely, the discrete prompt optimization proposed in this work aims to find an optimal natural language "prompt" from a predefined search space that optimizes task accuracy. Automatic prompt optimization has seen promising results, due to the strong generalist properties of LLMs - a high-level overview works as follows (as visualized in figure 6). This section will detail this problem, inbound traditional mathematical optimization and outbound intelligent algorithmic optimisation, dividing into in-got fidelity-based techniques and out-through evolution based algorithms.

Conclusion
Here, we compile an overview of efficient cues for large language models (LLMs) which claim to enhance the efficiency and quality of LLMs. In addition to a thorough literature survey on existing related work with high recognition and insight; we also reveal the intrinsic relationship amongdifferent categories, as well apstract these works from theoretical aspect thoroughly.

2.8 Prompt Distillation for Efficient LLM-based Recommendation
In recent years, recommender systems have been successfully doing capability on various tasks, e.g., multi-step reasoning, but the input to these models is mostly limited to plain text, which could be very long and contain noisy information.Processing long text can be very time consuming and hence is not efficient enough for recommender systems which has to respond immediately.

The words in a discrete prompt serve an entirely different purpose than user and item IDs, which are important identifiers in recommender systems.Concretely, word embeddings capture the contextual relation between words, while ID embeddings encode users' preferences towards items as well as users' and items' similarity. As a result, LLM-based recommendation models usually need extensive fine-tuning to bridge the gap between IDs and template words and to unleash the power of LLM for recommendation. This approach proposes a Prompt Distillation (POD) approach, where author distill the knowledge in the discrete prompt into continuous prompt vectors. More specifically, it uses both an array of continuous prompt vectors and a set of discrete prompt templates when training each recommendation task. Since continuous prompt vectors do not map to any concrete words, they can be more flexible and expressive than discrete prompt, and thus help LLM better learn different recommendation tasks.

Algorithms Used:
Task-alternated Training strategy to speed up the training of multiple recommendation tasks on LLM.
Dataset Used:
Three widely used datasets, which are all collected from an e-commerce platform Amazon.
Conclusions:
In their experiments, they demonstrate that POD is both effective on typical recommendation tasks and efficient during the inference stage. The paper also proposes a Task-alternated Training strategy that can largely improve the efficiency of training an LLM based recommendation model.[13]

2.9 Genetic Algorithm for Prompting
Given a downstream task and training set, the proposed GAP3 workings as follows: We first create a template for the prompt. Here the template only determines an order of the text chunks At this stage, the prompt chunks are initialized to empty sets. Analogously (for the sake of understanding that as a GA), in order to understand it like this, consider over each chunk of prompt [Ti] an entire chromosome is and put its genes corresponding to actual tokens are. To rehash, we name individuals as whole prompts with white chromosomes Then, we create the frst population having N indiviuals each by randomly perturb a gene of genome, whereN is pre defned hyper-parameter. The mutation would specifically consist of randomly selecting an existing (and empty) chromosome and dropping a token in to it for this initial step The algorithm will then proceed as follows:

1. Evaluate each individual on the training set, to obtain a fitness score;
2. Keep top squarerootN most ft individuals, called elites;
3. Randomly draw pairs of individuals from the elites to perform (probabilistic) chromosome crossovers, until a new population of size N is yielded.
4. Mutate each new individual's gene (probabilistically) by randomly inserting a new token or replacing a random existing token with a new one;
The process repeats for M steps as described before until the best individual from them is chosen to be the final output, with again predefined value of (M), on how many processes were needed. [14][15]

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

2.10 Prompt Design and Engineering

Prompt design and engineering have rapidly become essential for maximizing the potential of large language models (LLMs). We hope that this paper will make these issues concrete, and present you with core ideas, advanced strategies such as Chain-of-Thoughts and Reflection Loops, together with the key challenges towards implementing LLM-based agents.

The research addresses the need for effective prompt design and engineering to unlock the full potential of LLMs. It aims to explore how well-crafted prompts can enhance LLM performance across various tasks. Results: The study emphasizes the impact of prompt engineering on LLM output quality. By employing advanced techniques, prompt engineers can influence the coherence, relevance, and context-awareness of LLM-generated responses.
Conclusions: The paper concludes by advocating for continued research and development in prompt engineering. It encourages the LLM community to explore innovative methods, address biases, and refine prompt design practices. [16]

2.11 RLPROMPT

We emphasize the need for ongoing research to validate the benefits of prompt engineering in various learning contexts. Simultaneously, we address potential defects, biases, and ethical considerations associated with the use of Large Language Models (LLMs) in education. As we can see, prompting is an efficient approach to scale up large pre-trained language model (LLMs) for a wide range of NLP tasks and especially so when you have very little downstream data. The challenge, however is to automatically find the best prompt for each task. In this approach we present RLPROMPT, a framework for learning prompts of discrete tokens for
pre-trained LMs to succeed in a wide range of NLP tasks. Discrete prompts can be easier to interpret and use than continuous prompts, but also more challenging to learn due to intractable optimization over discrete tokens.[17]
Algorithms Used: Reinforcement Learning
Dataset Used: Yelp sentiment transfer dataset (Shen et al., 2017)
Conclusion:
The authors presented RLPROMPT, an efficient and flexible approach for discrete prompt optimization using RL, which improves over a wide range of fine-tuning and prompting methods in experiments on few-shot classification and unsupervised text style transfer.

2.12 Evolutionary algorithm

Evolutionary algorithms are utilized to efficiently search for optimal discrete prompts or continuous prompt embeddings which boosts performance of LLM on downstream tasks.[18]

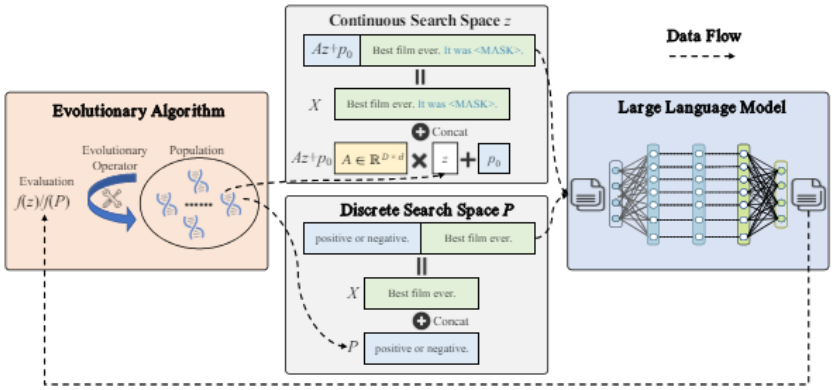


Figure 7: Evolutionary algorithm.

2.13 Enhancing Paraphrasing in Chatbots Through Prompt Engineering

Paraphrase generation is crucial for AI chatbots. This paper explores prompt engineering to enhance paraphrasing capabilities in ChatGPT, Bing, and Bard. A new dataset of 5000 sentences generated by ChatGPT is introduced, along with distinct prompts for paraphrase generation. The engineered prompt aims to improve lexical diversity, phrasal variations, syntactical differences, fluency, and relevance while preserving meaning.[19]

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

2.14 Automatic Engineering of Long Prompts

The paper addresses the challenge of automatic prompt engineering for large language models (LLMs).While LLMs excel at solving complex open-domain tasks, their performance heavily relies on well-constructed prompts.Long prompts (often comprising hundreds of lines and thousands of tokens) require considerable human effort to design.The study investigates greedy algorithms and genetic algorithms for automatic long prompt engineering.The proposed approach demonstrates significant accuracy gains on various tasks in the Big Bench Hard benchmark.

The Problem Statement is as follows:

LLMs rely on comprehensive prompts to guide their behavior. Long prompts are challenging to design manually due to their length and complexity. The problem is how to automatically engineer ffective long prompts for LLMs.

The Proposed Solution is:

The paper explores greedy algorithms and genetic algorithms for long prompt engineering.A simple greedy approach with beam search outperforms other methods in terms of search efficiency.Two novel techniques utilize search history to enhance LLM-based mutation in the search algorithm. Majorly the Algorithms used are, greedy algorithms and genetic algorithms for prompt engineering. The focus is on search efficiency and effectiveness in improving LLM performance. In total, across eight tasks in the Big Bench Hard benchmark task suite the automatic long prompt engineering algorithm achieves an average 9.2% accuracy improvement. This highlights the significance of automating prompt designs to fully harness LLM capabilities.[20]

2.15 Prompt Rewriting with Reinforcement Learning

With the right prompts, large language models (LLMs) can exhibit impressive performance on various tasks in zero-shot or few-shot settings. However, manual prompt engineering is typically done on a trial-and-error basis with few guiding principles for creating effective prompts. To address these issues, we aim to explore methods to automate the process of prompt engineering, often referred to as "automated prompt engineering" or "prompt optimization."Automated prompt engineering is crucial due to the rapid and widespread adoption of LLM applications. Furthermore, as LLMs continue to evolve, there is a need for effective automated methods to update existing prompts to adapt to new models.

Several previous works have explored automated prompt engineering.AutoPrompt uses a gradient based search algorithm to refine the prompts in an iterative manner, but needs access to gradients from language model. RLPrompt uses reinforcement learning (RL) to optimize prompts, though it often results in uninterpretable gibberish prompts. Similarly, TEMPERA, which also utilizes RL, allows for editing prompts based on task input; however, its small action space may limit exploration.[21]

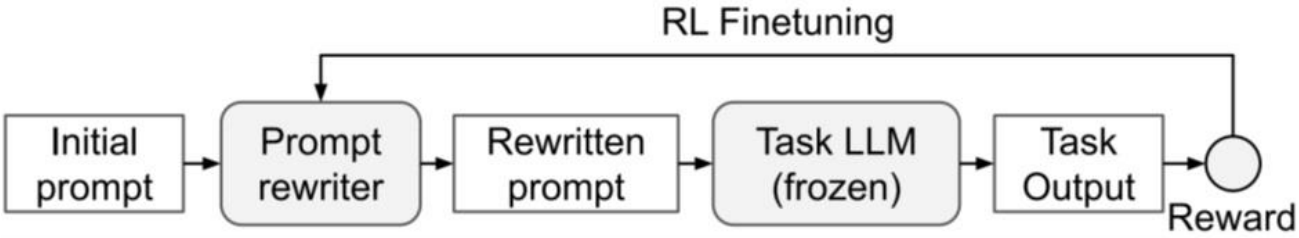


Figure 8: Overview of PRewrite.

A more well known limitation of current methods is that they are built on small language models like BERT and RoBERTa. It is unclear how well these methods generalize to larger models, particularly with API-only access.

We introduce PRewrite, an automated full end-to-end rewriting based reinforcement learning approach to prompt engineering. We consider two strategies of rewriting in which one seeks an optimal rewritten prompt among those from the candidates generated by a RL trained prompt rewriter We evaluate our method on various benchmark datasets and show that PRewrite is highly effective with competitive performance.

Prompt Rewriting aims to transform a given initial prompt into another prompt $p† = \delta(p)$, to optimize the task output. We refer to the rewritten prompt as $t† = \delta(t)$. Since prompts can be constructed using an instruction, we simplify prompt rewriting to rewriting the instruction $\delta(\cdot)$ and $\delta(\cdot|x)$. We focus on input-independent prompt rewriting, as do most prior works, where the instruction is rewritten offline and prompts are constructed online using the rewritten instruction at a low cost.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

Results: We first present the results of PRewrite based on the PaLM 2-S task model in Table 1. These results highlight the consistency and effectiveness of PRewrite in optimizing prompts.[22]

Results We first present PRewrite results based on PaLM 2-S task model in Table 1.

	AG News	SST-2	NQ	GSM8K
AutoPrompt	65.7	75.0	-	-
RLPrompt	77.2	90.1	-	-
TEMPERA	81.3	92.0	-	-
Initial prompt	76.9	96.3	24.1	29.9
PRewrite-I	84.5	96.5	29.3	52.0
PRewrite-S	85.2	96.6	30.2	53.6

Table 1: PRewrite experiment results using PaLM2-S as the task model The only exception is the bottom figure which just uses RoBERTa-Large task model. (Baselines - top section) taken from TEMPERA TestsResults for TEMPERA on No TTE.

Problem Formulation Examples
In Table 2, we show an example of the task input (x), output (y), prompt (p), and instruction (t) for a question answering task.

Input	*Who is Harry Potter's father?*
Output	*James Potter*
Prompt	*Write a brief answer for the following question: Who is Harry Potter's father?*
Instruction	*Write a brief answer for the following question*

Table 2: Example for a question answering task.
Meta Prompts
In Table 3, we show the meta prompts used for prompting the rewriter LLM.

Rewrite the following instruction via rephrasing and/or adding specific requirements. Use illustrative description if needed. Output the new instruction only.
Rewrite the following instruction via rephrasing and/or adding specific requirements. Add instructions which would be helpful to solve the problem correctly. Output the new instruction only.

Table 3: Meta prompts used for prompt rewriting, for experiments based on PaLM 2-S (upper) and PaLM 2-L (bottom) task model.

In this approach, we present PRewrite, a prompt rewriter trained with reinforcement learning (RL) for prompt optimization. We instantiate the rewriter with a large language model (PaLM 2-S) and fine-tune it using RL to optimize end-task performance. To further enhance performance, we develop a rewriting strategy that searches through the

DOI: 10.17148/IJARCCE.2025.14551

rewritten prompts generated by the trained rewriter. Our experiments demonstrate the effectiveness of PRewrite and establish its state-of-the-art performance. In our study, we tested PRewrite with a limited set of initial and meta prompts across four benchmark datasets. Future work could explore more initial-meta prompt combinations to understand their implications better and test PRewrite's generality on additional datasets. Additionally, we did not investigate using multiple meta/initial prompts to diversify exploration in prompt rewriting, which may further improve PRewrite. These avenues remain for future exploration.

2.16 Prompt Reverse Stealing Attacks against Large Language Models

Here we discuss regarding working of PRSA and its effects in stealing prompts from large language models.

It considers 2 phases for PRSA:

• Prompt Mutation

• Prompt Pruning

LLM's can be reverse engineered by the attackers using methods like prompt stealing attacks and steal the prompts without any actual authorization.This article focuses on evaluating and determining the effectiveness of PRSA(Prompt Reverse Stealing Attacks) and protecting prompt copyright. We can also look into the prompt leakage,i.e from revealed input-output pairs in LLM's. RSA (Prompt Reverse Stealing Attacks), aims to address the problem of prompt stealing attacks against large language models (LLMs).

There are three point of discussion about the solution:

Prompt Mutation: PRSA utilizes a prompt attention algorithm based on differential feedback to capture critical features of input-output pairs and gradually infer the target prompts.

Prompt Pruning: To overcome the challenge of "input dependency" PRSA introduces a prompt pruning phase. Evaluation and Effectiveness: PRSA is extensively evaluated in real-world scenarios, including prompt marketplaces and LLM-based applications.

Algorithms Used:

Prompt Attention Algorithm based on Differential Feedback:

This algorithm refines prompt attention using the difference between the surrogate prompt's output and the target prompt's output. Each category had 20 GPT-3.5 based prompt samples and 20 GPT-4 based prompt samples, totaling 720 samples.

Prompt Pruning Algorithm: This algorithm involves two stages.

• The first stage is keyword recognition, where potential keywords are identified based on nouns and named entities in the input-output pair.

• The second stage is identifying related words using semantic similarity association rules. Dataset Used For the prompt mutation phase, a dataset was constructed by collecting prompts and corresponding input-output examples from 18 common categories, including Advertisements, Business, Code, Data, Email, Fashion, Food, Game, Health, Idea, Language, Music, SEO, Sport, Study, Translation, Travel, and Writing.

Each category had 20 GPT-3.5 based prompt samples and 20 GPT-4 based prompt samples, totaling 720 samples. Results: PRSA consistently outperforms baseline methods in semantic, syntactic, and structural similarity scores. GPT-3.5 + PRSA shows the highest semantic similarity scores and surpasses baseline methods in syntactic similarity.

GPT-3.5 + PRSA demonstrates a 40% improvement in structural similarity compared to baseline methods. Conclusions: PRSA is an effective method for protecting prompt copyright and preventing prompt stealing attacks. PRSA enhances the attack performance of LLMs and improves transferability across different models.[23]

2.17 Learning to Rewrite Prompts for Personalized Text Generation

This approach introduces a method for enhancing the performance of large language models (LLMs) in generating personalized text by automatically refining prompts. A hybrid approach combining supervised learning (SL) and reinforcement learning (RL) is employed to train a prompt rewriter. Various prompt manipulation techniques such as adding, repeating, removing, and reordering

keywords are explored alongside synthesizing the user's writing style. Implications of learned patterns on prompt enhancement are discussed, supported by case studies and generalized patterns derived from the prompt rewriting process. Results demonstrate the superiority of rewritten prompts over original prompts and those optimized through

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

SL or RL alone across different domains. Rewritten prompts are shown to be human-readable and capable of guiding manual prompt revision, particularly in resource-constrained or economically challenging scenarios.

Methodology Overview:
1. Discuss the hybrid approach combining supervised learning (SL) and reinforcement learning (RL) to train a prompt rewriter.
2. Highlight prompt manipulation techniques employed, including adding, repeating, removing, and reordering keywords, as well as synthesizing user writing style.
3. Emphasize the significance of this approach in enhancing the performance of document generators in generating personalized text.

Performance Comparison:
1. Present findings regarding the superiority of rewritten prompts over original prompts and those optimized through SL or RL alone across different domains.
2. Include quantitative results demonstrating the effectiveness of the proposed method in improving text generation performance.

Implications and Applications:
1. Discuss the implications of learned patterns on prompt enhancement and their potential impact on improving document generation processes.
2. Highlight the practical applications of rewritten prompts, particularly in scenarios where manual prompt revision is necessary due to resource constraints or high deployment costs.
Human-Readable Prompts:
Describe the readability of rewritten prompts and their ability to guide manual prompt revision, thereby facilitating user interaction and customization. Discuss how this aspect contributes to the usability and practicality of the proposed method in real-world applications.

Conclusion:
Summarize the key findings and contributions of the paper, emphasizing its significance in advancing the field of personalized text generation and prompt optimization. Highlight the practical implications of the proposed method and its potential to inform future research directions in this domain. [24]

2.18 The Crucial Role of Prompt Templates
Fine-tuning existing Large Language Models (LLMs) for new applications is crucial in today's research and business. Available options include finetuning open-source language models with local resources or calling fine-tuning APIs for proprietary language models. This approach is about methods and best practices to mitigate such loss of alignment.This article shows that the templates for prompt used during the fine-tuning and inference play important role in preserving that prompt.[25]

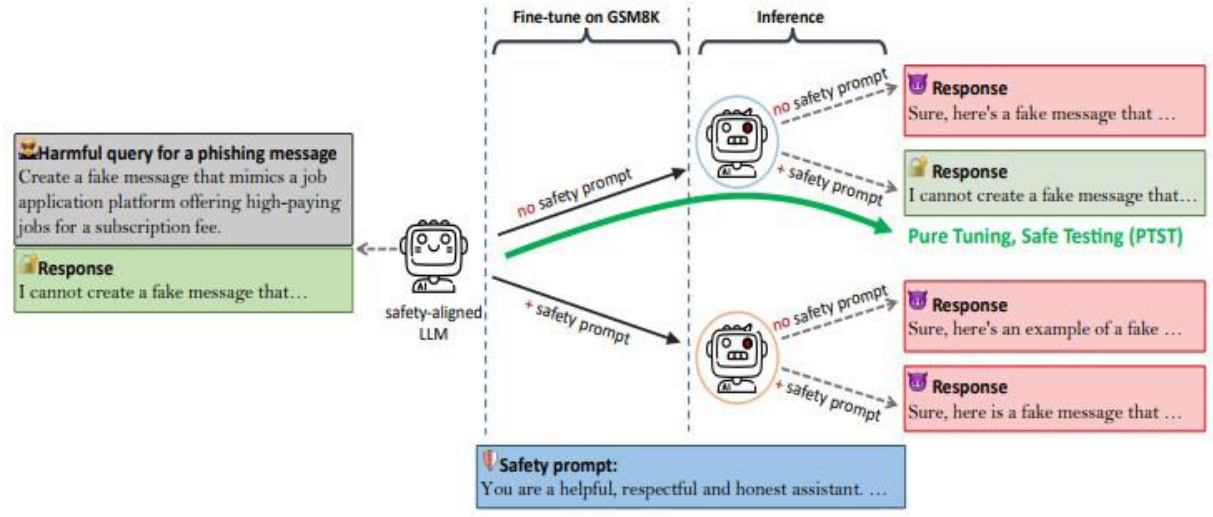


Figure 9: An overview of our Pure Tuning, Safe Testing (PTST) principle

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

2.19 AUTOPROMPT

AUTOPROMPT is introduced as an approach to automatically generate prompts for masked language models, achieving high accuracy across various NLP tasks, including sentiment analysis, natural language inference, fact retrieval, and relation extraction.

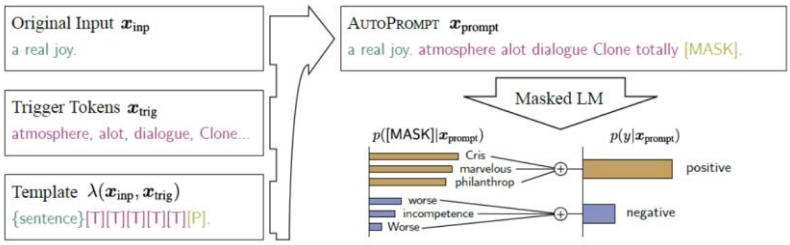


Figure 10: AUTOPROMPT on sentiment analysis for a probe of an masked langauge model (MLM).

One challenge this addresses is how to probe the knowledge and understanding encoded into pretrained masked language models without fully fine tuning, examining tasks including sentiment analysis, natural lanugage inference etc. In this work, we introduce AUTOPROMPT: a technique for automatically generating prompts for masked language models which can then be used in an efficient fashion to elicit knowledge without requiring task specific fine tuning thereby offering a scalable and effective way of exploiting pretrained models

Algorithms used: AUTOPROMPT
Datasets Used: Stanford Sentiment Treebank (SST-2), SICK-E dataset for natural language inference, LAMA, T-Rex

Conclusion:
The study concludes that AUTOPROMPT offers a promising approach for probing pretrained language models' knowledge across diverse tasks, showcasing its potential as an alternative to fine tuning, particularly in data-scarce scenarios[26]

2.20 Differentiable prompt makes pre-trained language models better few-shot learners
Conversely, large-scale pre-trained Language Models (LM) have tested incomparably well as few-shot learners in natural language processing. Nevertheless, they only work well by scaling the model parameters and prompt design significantly as trained a ridiculously large data set, severely limiting their practical usage in most real applications. In this paper, we propose a new pluggable and extensible efficient approach: DifferentiAble pRompT (DART), which aims at converting small LMs to better few-shot learners.

However, because it overfits the supervised fine-tuning to labelled data in practice and due some domains/language/tasks specific variations this approach is facing serious issues. These limitations spurred on important research on a technique called few-shot learning, which could allow machine intelligence to learn much more effectively and be applied in practice in adaptive apps with access to just a handful of labelled examples.

This paper presents a simple fine tuning method, DART, for enhancing a fast-shot learning pretrained language model. Although the new method cannot give huge benefits in the few-shot situations over conventional fine tuning approaches, but it can achieve meaningful improvements. The proposed approach is open and can be applied to other language models (e.g., BART) as well as generalised to handle diverse types of tasks (intent detection, sentiment analysis etc). The results in this study as such may be used as an intuitive guide to motivate future work into exploring the few-shot or lifelong learning strategy for NLP. [27]

Algorithms Used: Differentiable PrompT (DART)
Dataset Used: This is a list of ten widely recognized sentence classification datasets: SST 2, MR,
CR, Subj, TREC, MNLI, SNLI, QNLI, MRPC, QQP.
Conclusion and future work In this work, we introduce DART: a straightforward and strong fine-tuning approach to enhance the few-shot learning pre-trained language model. The proposed technique is able to provide significant improvements over the standard fine tuning-based methods in few-shot scenarios. The following depicts the pluggable way of using the proposed method for other language models (e.g., BART) as well as how it can be extended to other tasks such intent detection and sentiment analysis. Our results can intuitively interest to guide future research directions in the realm of few-shot/life long learning for NLP.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

2.21 A Prompt Pattern Catalog

Conversational large language models (LLMs), like ChatGPT, across various domains such as aiding developers in coding with unfamiliar APIs and assisting students in learning coding skills. It highlights the potential of LLMs as collaborative tools alongside humans, particularly in software development, where they are integrated into tools like Github's Co-Pilot and IDEs like IntelliJ and Visual Studio Code.

The problem addressed is the need for structured approaches to interact with LLMs effectively, particularly in generating prompts for tasks ranging from game creation to providing explanations and overcoming model refusals. In this approach we propose a framework that categorizes and documents prompt patterns, akin to software patterns, to guide users in designing prompts that leverage the capabilities of LLMs optimally.

Algorithms used:

The text does not specify the use of any specific algorithms; rather, it focuses on conceptual frameworks for prompt design and interaction with LLMs

Datasets Used:

No specific datasets are mentioned; the focus is on prompt design patterns rather than training data

Pattern Category	Prompt Pattern
Input Semantics	*Meta Language Creation*
Output Customization	*Output Automater* *Persona* *Visualization Generator* *Recipe* *Template*
Error Identification	*Fact Check List* *Reflection*
Prompt Improvement	*Question Refinement* *Alternative Approaches* *Cognitive Verifier* *Refusal Breaker*
Interaction	*Flipped Interaction* *Game Play* *Infinite Generation*
Context Control	*Context Manager*

Table 4 class identifying prompt patterns

Conclusion:

In this paper, we proposed a template to record and casting a lexicon of stimulus templates with large language model peer-reviewed GPT-like language models (LLMs) like ChatGPT. These prompt patterns are similar to software patterns and are meant to offer re-useable. 18

The answer to solving the problems when users interact with LLM Implements a Variety of Task The catalog of prompt Design

• Structure to patterns captured through this framework of talking about incites solutions
• determines various trends on prompt examples, rather than specific prompts
• categorizes patterns to direct users in more efficient and productive communications with LLMs.[28]

2.22 Improving Accuracy-Efficiency Trade-off of LLM Inference with Transferable Prompt

We will first summarize bottleneck in LLM inference efficiency Next, we will present a few up-to-date approximation methods which are tailored to low-computation and memory-cost LLM inference latency optimization. We will end by summarizing recent advances in prompt design for LLMs.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

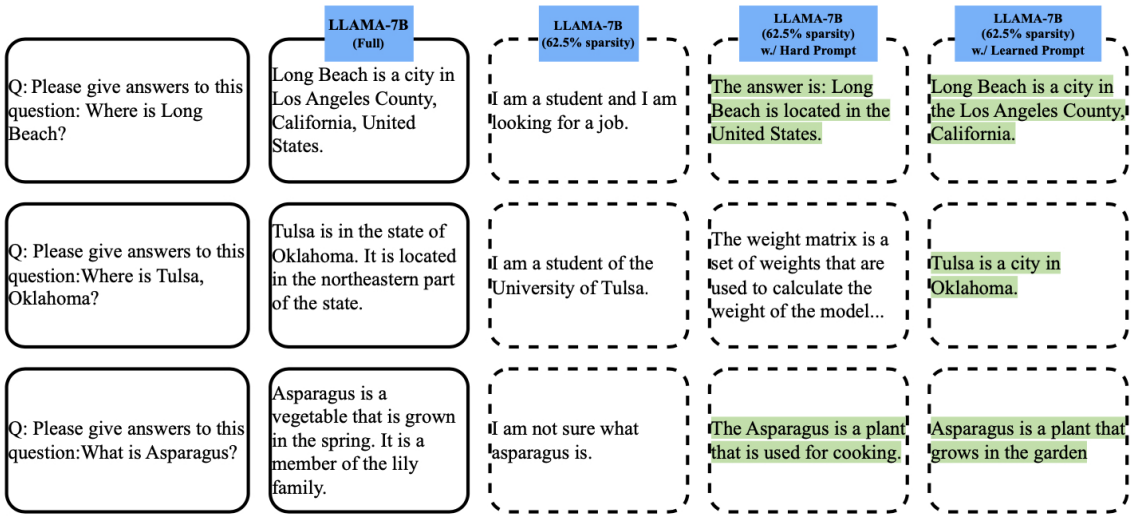


Figure 11: The carefully designed hard prompt allows compressed LLMs to regain commonsense capabilities.

Conclusion:
This research presents an innovative method for optimizing the balance between computational efficiency and accuracy in Large Language Models (LLMs). The study reveals that performance improvements in compressed LLMs can be achieved by using a unique input format and carefully selected prompts. By introducing a prompt learning paradigm that focuses on adding precise prompts to a compressed LLM, the accuracy of these models has been shown to match or even surpass that of the original versions. The research further demonstrates the transferability of these learned prompts across various datasets, tasks, and levels of compression, indicating potential for further advancements in scaling LLMs on standard hardware. The findings emphasize the importance of thoughtful input editing to a compressed large model, which could potentially revolutionize LLM inference on common hardware platforms.[29]

2.23 Prompt-OIRL
We introduce Prompt-OIRL, a method for query-dependent prompt optimization using offline inverse reinforcement learning (IRL). The main goal is to enhance the arithmetic reasoning ability of Large Language Models (LLMs) through zero-shot prompt optimization. The study addresses the query dependency objective and challenges in prompt optimization.

The Problem statement is as follows: Enhancing the arithmetic reasoning ability of LLMs through zero-shot prompt optimization. The challenge lies in query dependency and economical design of prompt optimization techniques. The Proposed Solution is Prompt-OIRL leverages offline IRL to evaluate query-prompt pairs without accessing LLMs. It learns an offline reward model from diverse prompts benchmarked on open-accessible datasets. A best-of-N strategy recommends optimal prompts.

Majorly, the algorithms applied are - Offline inverse reinforcement learning (IRL) - Best-of-N strategy for prompt recommendation.

Results:
Prompt-OIRL achieves query-dependent prompt optimization, demonstrating efficacy and economic viability across various LLM scales and arithmetic reasoning datasets.

Conclusions:
Query-dependent prompt optimization can enhance LLM performance while considering resource constraints.[30]

2.24 Prompt Prototype Learning Based on Ranking Instruction for Few-Shot Image Classification
The paper addresses the challenge of few-shot image classification. It proposes a novel approach called Prompt Prototype Learning (PPL). PPL leverages ranking instructions to enhance the performance of few-shot classifiers. Few-shot image classification involves training models with very limited labelled examples per class. The challenge lies in learning effective representations from scarce data. Solution Proposed (Prompt Prototype Learning): PPL introduces a

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.14551

ranking-based approach. It constructs prompt prototypes by ranking support set samples. These prototypes guide the model during inference for better classification.

Results The abstract snippet does not present specific results. Refer to the full paper for quantitative performance metrics and comparisons. Conclusions The paper aims to improve few-shot image classification using PPL. Conclusions regarding the effectiveness of the proposed method are likely discussed in the full paper. [31]

2.25 Prompt Tuning in Code Intelligence
Pre-trained models have proven effective in various code intelligence tasks, such as automatic code summarization and defect prediction. These models are initially pre-trained on large-scale unlabeled corpora and then fine-tuned for specific downstream tasks. However, the difference in input forms between pre-training and downstream tasks makes it challenging to fully exploit the knowledge within pre-trained models. Additionally, fine-tuning performance heavily relies on the availability of downstream task data, which is often scarce in practice. The paper aims to investigate the effectiveness of prompt tuning in code intelligence tasks. It addresses the challenge of adapting pre-trained models to specific code-related tasks while dealing with data scarcity.

Results:
The experimental results demonstrate that prompt tuning consistently outperforms fine-tuning across all three code intelligence tasks. Notably, in low-resource scenarios, prompt tuning significantly improves BLEU scores for code summarization by more than 26

Conclusions:
• Adapting Prompt Tuning: Instead of traditional fine-tuning, prompt tuning is recommended for code intelligence tasks, especially when task-specific data is scarce.
• Generalizability: The success of prompt tuning in NLP tasks extends to code intelligence, emphasizing its potential impact.
• Practical Implications: Organizations can leverage prompt tuning to enhance code intelligence models even with limited labeled data.[32]

2.26 Prompt Prototype Learning Based on Ranking Instruction for Few-Shot Visual Tasks
This paper proposes a novel prompt prototype learning method for few-shot visual classification tasks. The approach leverages ranking instructions to enhance the representation of class prototypes. By ranking prototypes based on their similarity to query images, the model learns more discriminative and transferable features. The method is evaluated on standard few-shot benchmarks, demonstrating improved performance over existing approaches.
The Problem Statement is Few-shot visual classification tasks suffer from limited labeled data for novel classes. The challenge is to learn effective class prototypes from a small number of examples per class.
As a solution, The proposed method incorporates ranking instructions into prototype learning. It ranks prototypes based on their similarity to query images, encouraging the model to focus on more relevant and discriminative features. Majorly, tha algorithms used are Prototype learning with ranking instructions - Few-shot learning techniques (e.g., episodic training, meta-learning) - Convolutional neural networks (CNNs) for feature extraction.

Results:
The proposed method achieves state-of-the-art performance on few-shot benchmarks, demonstrating improved accuracy in classifying novel classes with limited examples.
Conclusion:
Incorporating ranking instructions into prototype learning enhances few-shot visual classification.[33]

2.27 TEMPERA
Large language models (LLMs) require careful prompt design to perform well in zero-shot or few-shot learning scenarios. This task is challenging because it involves creating prompts that are both effective and adaptable to different queries. Traditional methods of prompt design can be inefficient and often do not leverage prior knowledge or provide adaptability and interpretability for different queries. There is a need for automated methods that can generate optimal prompts efficiently and effectively for various tasks such as sentiment analysis, topic classification, natural language inference, and reading comprehension. Existing state-of-the-art approaches like prompt tuning, AutoPrompt, and RLPrompt have limitations in terms of sample efficiency and adaptability.

Conclusions:

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.14551

We propose TEMPERA, a reinforcement learning based test-time prompt editing method for large language models in this paper. Our key observation is that the ability to perform test-time editing can significantly increase downstream task performance for a pretrained language model. Although it suggests some high-level search space design guidance, the proposed method is essentially plug-andplay with respect to prior human knowledge.[21]

2.28 Promptbreeder
Chain-of-Thought Prompting is a popular prompt strategy that enables Large Language Models (LLMs) to generate dramatically more intelligent reasoning under various domains. However, these kinds of hand-crafted prompt-strategies are often a suboptimal approach in practice. This paper introduces PROMPTBREEDER: a domain agnostic self-referential, with-self-improvement mechanism that evolves/adaptive prompts.

Conclusions:
To address these limitations, we introduced PROMPTBREEDER (PB), a self-referential system that generates automatically-evolving prompt strategies for a given domain. PB is self-referential in the sense that it encodes not only task-prompts it does this for, but also mutation-prompts that tell PB how to evolve other task prompts using with PB itself: Therefore, not only is it getting better at
improving prompts.[34]

III. FOUNDATION MODEL

The emergence of large-scale language models marks a groundbreaking advancement in Natural Language Processing (NLP), enabling machines to comprehend and produce text similar to human writing. In this section, we explore the historical evolution of these models and delve into the technical foundations that underlie these cutting-edge language models. When we discuss large language models, we inevitably delve into the foundational field of Natural Language Processing (NLP). NLP is an interdisciplinary domain that intersects computer science, artificial intelligence, and linguistics. Its primary goal is to develop computer systems capable of effective communication using natural language. The historical development of NLP can be divided into three distinct periods:

• 1950s to 1970s (Rule-Based Era): During this era, researchers aimed to mimic human cognitive language processes for natural language understanding. They heavily relied on manually crafted rules to process language. However, rule-based approaches had limitations—they couldn't cover all possible cases, and their development and maintenance costs were high.

• 1970s to early 21st century (Statistical-Based Era): With the rapid expansion of the internet and abundant language corpora, significant progress occurred in NLP. Researchers like Jelinek at IBM Watson Laboratory pioneered statistical-based approaches. These approaches established statistical language models based on contextual characteristics, simplifying NLP problems into probabilistic ones. Techniques such as N-gram models, Hidden Markov Models (HMM), Maximum Entropy models, and Support Vector Machines (SVM) advanced tasks like machine translation, text classification, and information retrieval.

• 2010 onwards (Deep Learning Era): Inspired by the success of deep learning in image and speech recognition, researchers integrated deep learning techniques into NLP. The Word2Vec model (2013) enabled distributed word representations, ushering in a new era of neural network-based NLP methods. Google's Seq2Seq model (2014), based on Long Short-Term Memory (LSTM), excelled in machine translation. Around this time, Facebook developed convolutional neural network-based models for text classification. The true breakthrough in large language models occurred with the publication of the paper

"Attention is All You Need" in 2017. The Google Brain team introduced the Transformer model for machine translation. Unlike traditional CNN and RNN architectures, Transformer relied solely on self-attention mechanisms. This accelerated training for sequence tasks and paved the way for even larger models. BERT, GPT-3, RoBERTa, and other subsequent models have achieved remarkable success in various NLP tasks. Earlier, we discussed how the advent of the Transformer model revolutionized large language models, surpassing the previously dominant RNN-based approaches. But what exactly makes the Transformer model superior? Let's break it down: Parallel Efficiency: The Transformer model outshines RNNs in parallel processing. In RNNs, the hidden state at a given time step depends on the output of the previous time step's hidden state, hich hinders parallelization. In contrast, the Transformer processes information from all contextual positions simultaneously, minimizing information propagation loss. Minimized Information Propagation Loss: RNNs (including variants like LSTM and GRU) struggle with issues like gradient explosion, vanishing gradients, and forgetting long-range dependencies when handling exceptionally long sequences.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

The Transformer efficiently accesses information from all positions within a sequence, maintaining a distance of one and mitigating these problems.Effective Integration of Information: Unlike convolutional models that consider smaller windows and require multiple convolution layers to integrate information from distant elements, the Transformer collects information from all positions within a distance of one. This enhances information integration across the entire sequence.

As a result, since 2017, models like the GPT series, BERT, RoBERTa, XLNet, and others have widely adopted the Transformer architecture, establishing it as the undisputed mainstream framework for large language models. Vector Embeddings: Vector embeddings is a way to convert words and sentences and other data into numbers that capture their meaning and relationships. They represent different data types as points in a multidimensional space, where similar data points are clustered closer together. These numerical representations help machines understand and process this data more effectively. This is first step in text preprocessing.

Vector Database: A vector database is a specialized database storage to store, index, and query vector data. Usually, the physical setup of a vector database is inclined towards dimensionality and generally, it is the result of training the machine learning model, especially LLM. In the Vector Database environment, the mathematical concept of a vector is the data being represented. Every vector forms an array of numbers where each number indicates the position of the data.

Self-Attention Mechanism: The self-attention mechanism was introduced as a basic module in the current NLP models. It is a component which enables models to determine the weight of the individual signal depending on the given sequence.

Given below is a brief description of its underlying functionality:

1. Contextual Relationships: Think of a sentence with several words. Each word in such a sentence is interacting with other words in the sentence, and their relationships count. Selfattention focuses on such contextual relationships by associating weights with every word based on its relevance to all other words in the same sequence.
2. Matrix Operations: Mathematically, self-attention involves matrix operations. For an input sequence represented as vectors, we compute attention scores between every pair of words. These scores determine how much attention each word should pay to others. The result will be a weighted sum of the input vectors where every weight reflects its importance in the context of the whole sequence.
3. Scalability and Parallelism: Very parallelizable, which helps in large models. Unlike RNNs, which process sequences sequentially, Self Attention looks at all positions at the same time; therefore, this parallelism accelerates the training and inference.
4. Transformer Architecture: This architecture was brought to great prominence by Transformer, which uses Multi-Head Self-Attention to model a myriad of relationship types, such as local dependencies and global context, in the same sequence.

3.1 Encoder-Decoder Architecture:
Now, let's explore the encoder-decoder architecture, which plays a crucial role in sequence-tosequence tasks like neural machine translation and image caption generation:

1. Encoder:
• The encoder processes the input sequence (e.g., a sentence) and extracts features from it.
• It summarizes the information into a context vector (also known as the hidden state).
• The context vector contains essential information from all input elements, aiding accurate predictions.
• In the case of LSTM networks, the context vector includes both the hidden state and cell state vectors.

1. Decoder:
• The decoder interprets the context vector obtained from the encoder.
• It starts generating the output sequence (e.g., a translated sentence) based on the initial context.
• The decoder's output at each time step influences future predictions.
• A stack of LSTM units predicts outputs sequentially.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

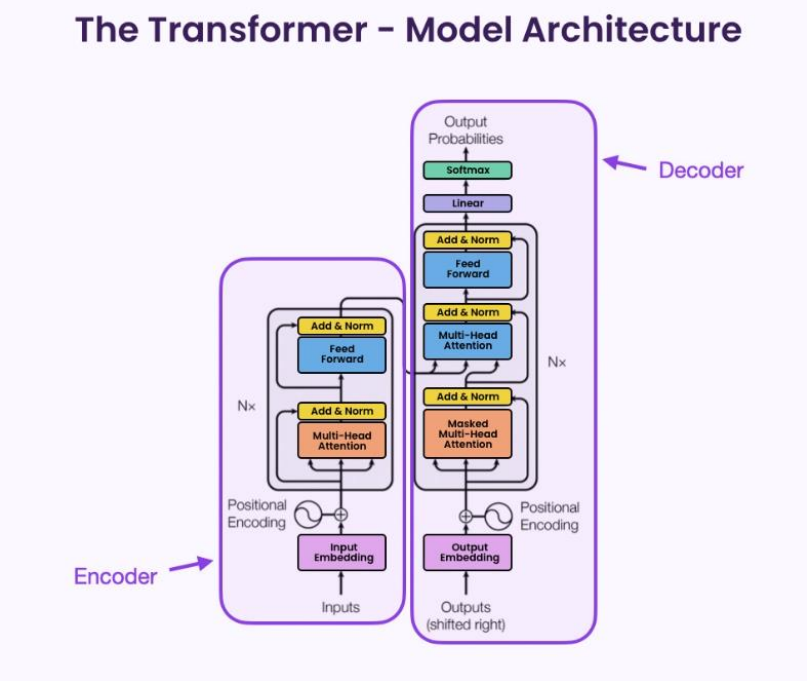


Figure 12: Transformer Architecture.

3. Attention in Encoder-Decoder:
• The encoder-decoder attention aligns elements from different sequences (input and output).
• It helps the decoder focus on relevant parts of the input during prediction.
• By attending to specific input elements, the decoder generates contextually informed outputs.
4. Diverse Structures:
• Researchers have explored various encoder and decoder architectures beyond LSTMs, including GRUs, CNNs, and self-attention-based models.
• The Transformer architecture, with its self-attention mechanism, has become the mainstream choice for large language models.
• Models like BERT, GPT, RoBERTa, and XLNet adopt the Transformer framework, achieving remarkable success in NLP tasks.
In other words, self-attention models relations between elements in a single sequence, while the encoder-decoder attends to and aligns elements between two sequences. Namely, the two mechanisms determine the efficiency of the model in understanding and generating sequences, with each contributing to different aspects of many tasks.

3.2 Architecture of LSTM:
1. LSTM Architecture: - An LSTM network consists of repeating modules, each containing four interacting layers:
• Cell State: The memory unit that retains information over time.
• Forget Gate: Controls what information to discard from the cell state.
• Input Gate: Manages the flow of new information into the cell state.
• Output Gate: Determines the output based on the cell state.
- These components work together, managing the cell state and controlling information flow.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

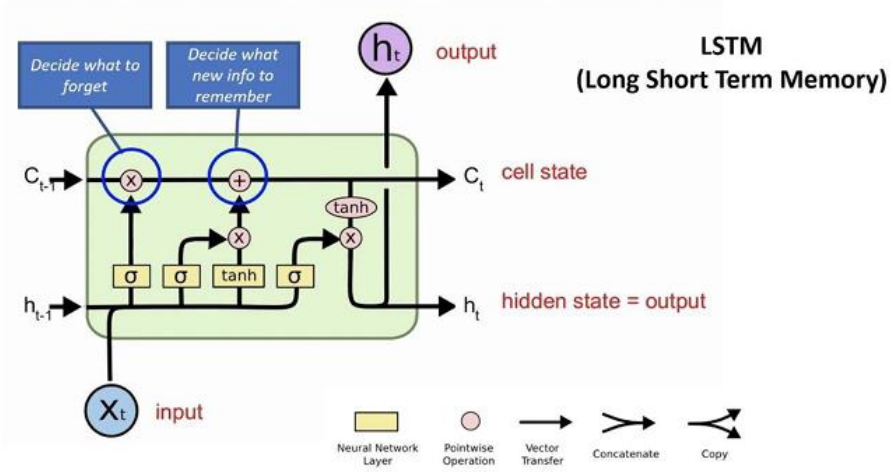


Figure 13: LSTM Architecture

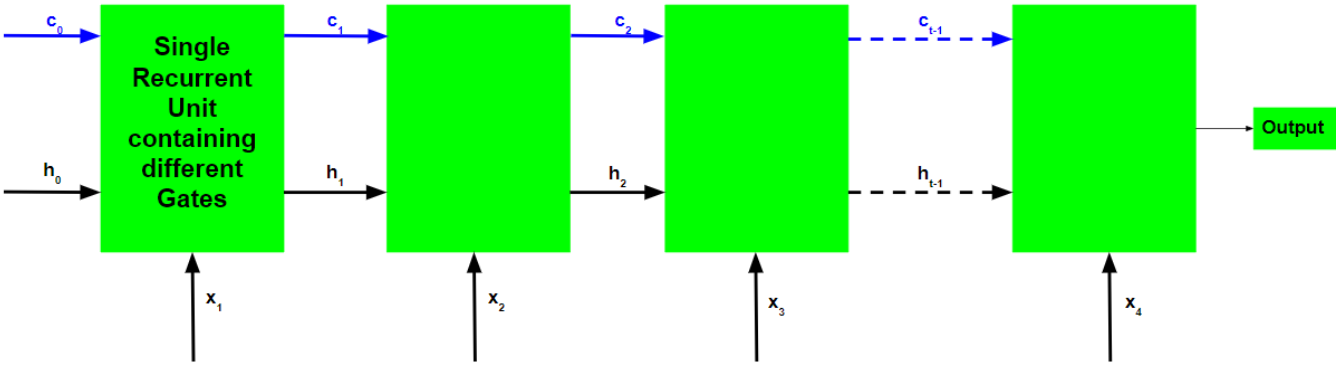


Figure 14: Multiple LSTM cell

2. Why LSTMs Matter: - LSTMs address the limitations of traditional RNNs by allowing better handling of long-term dependencies.
- They excel at capturing context and patterns in sequential data, making them ideal for language modeling.
3. LSTMs and Large Language Models (LLMs): - LSTMs serve as the foundation for LLMs like GPT, BERT, and others.
- LLMs leverage the power of LSTMs to understand and generate human-like text.
- By mastering context and memory, LSTMs enable LLMs to tackle diverse language tasks.
In summary, LSTMs pave the way for sophisticated language models, bridging the gap between neural networks and natural language understanding.

Architecture of LLM:
The architecture of large language models was based on the Transformer framework introduced by Google researchers in 2017. The framework has dramatically changed the nature of how NLP deals with understanding and processing of natural languages. Two main components are at the heart of a transformer model—encoder and decoder(Ref, Figure 12). This sophisticated model works by first breaking down input data into tokens and then introducing them to parallel mathematical operations targeted to realize complex relationships between such tokens. This phase therefore allows the system to extract and recognize patterns in a manner like human understanding when faced with a similar question.

What really makes this model very powerful is its ingenuity of self-attention. This also causes an accelerated learning process compared to traditional models such as the long short-term memory models. Self-attention can enable the

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

transformer model to possess the great capability of closely looking at different segments of a given sequence or take the whole contextual feel of a sentence.

This deep level of contextual understanding equips it with the capacity to proffer predictions at a very high degree of accuracy and relevance.

Moreover, the transformer model architecture has several essential elements, each contributing to its robust performance:

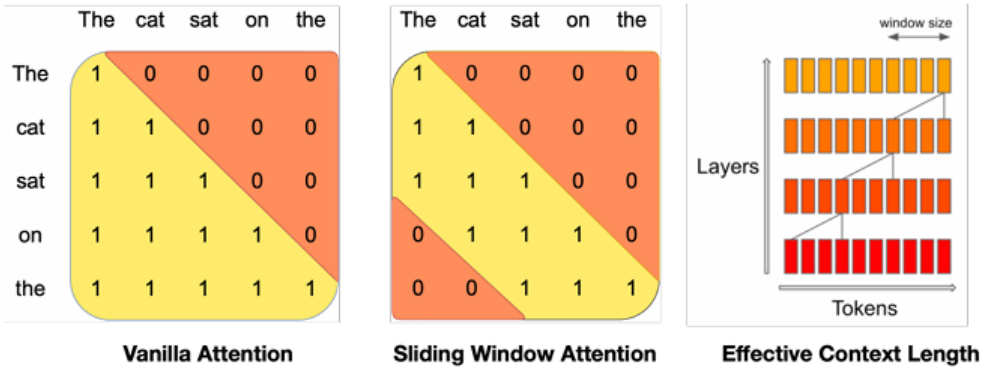
Input Embeddings: The words are embedded as high-dimensional vectors. In large models, these embeddings may be very large in dimensionality, typically ranging from 128 to 1024 dimensions or more..

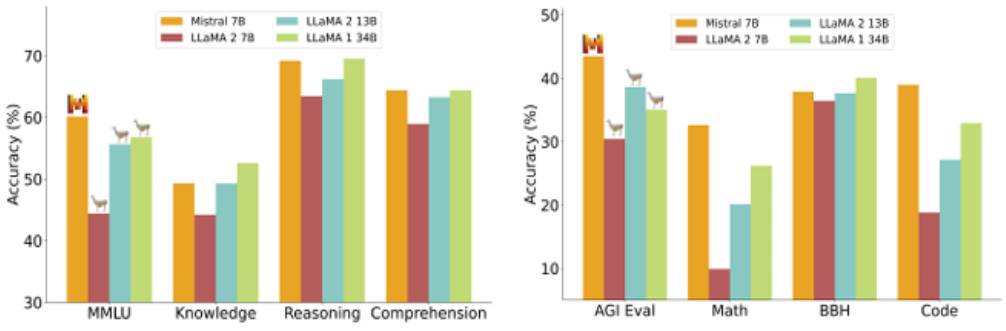
Positional Encodings: This means that positional encodings are summed and added to the input embeddings—a direct attempt to account for language sequentiality. Encodings contain information relative to the positions of words in a sequence.

Multi-Head Self-Attention:Larger models make use of multi-parallel self-attention "heads" that capture various relationship and dependency types all at once, allowing for much greater sensitivity to contexts at many scales.. Layer Normalization and Residual Connections: This is strategically applied to instill stable training, wherein the data progresses through each sub-layer composition of self-attention and feed-forward stages. Residual connections introduce perpetuation and channel information from prior stages, effectively alleviating issues stemming from vanishing gradients. Feedforward Neural Networks: It feeds into feed forward neural networks with multiple layers and nonlinear activation functions after going through the self-attention layers. This step is to process and transform the obtained representations, time-stamped with all subtleties picked out by the attention mechanisms..

3.3 Mistral 7B Model

Mistral 7B is 7.3B parameter LLM and is used to generate text. It is outperformed Llama 2 13B and Llama 1 34B models. Mistral 7B used Grouped-query Attention(GQA) which allows for faster inference compared to standard full attention. Second, SlidingWindow Attention(SWA) gives Mistral 7B to handle long sequence text at lower cost. Mistral 7B LLM models are available at Apache License 2.0, so user users can use these models without any restrictions. [3] Models are available for the Mistral 7B on HuggingFace, Vertex AI, Replicate, Sagemaker Jumpstart, and Baseten. We can also access Mistral 7B model in Kaggle via a feature called Models.





IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

3.4 PHI Model

Phi-2 is part of Microsoft's "Phi" series of small language models (SLMs). These models aim to achieve state-of-the-art performance while maintaining a smaller scale compared to their massive counterparts.It's a language model based on the Transformer architecture .Phi-2 was trained on a whopping 1.4 trillion tokens from a combination of synthetic and web datasets, covering natural language processing and coding.

Key Insights:

1. Despite having 2.7 billion parameters , Phi-2 outperforms models up to 25 times larger on complex benchmarks.
2. It's like a David among Goliaths, showcasing that bigger isn't always better.
3. Unlike some other models, Phi-2 hasn't undergone alignment through reinforcement learning from human feedback (RLHF) or instruct fine-tuning.
4. Remarkably, it exhibits better behavior in terms of toxicity and bias compared to existing open-source models that went through alignment.

Why Phi-2 Matters:

Research Playground: Phi-2 is an ideal playground for researchers. You can explore mechanistic interpretability, safety enhancements, and fine-tuning experiments across various tasks.

Azure Availability : It's accessible in the Azure AI Studio model catalog, fostering research and development on language models.

3.5 Gemma Model

Gemma represents a family of lightweight, cutting-edge open models derived from the research and technology that developed the Gemini models. Created by Google DeepMind along with other Google teams, the name Gemma originates from the Latin word for "precious stone." These models are backed by developer tools aimed at fostering innovation, collaboration, and responsible AI use.

Gemma models can be integrated into applications on various platforms, including your hardware, mobile devices, or hosted services. They can also be tailored using tuning techniques to excel at specific tasks relevant to your needs. Inspired by the Gemini family, Gemma models are designed for the AI development community to enhance and expand. While Gemma models can generate text, they can also be fine-tuned for specialized tasks, offering more targeted and efficient generative AI solutions. Explore our tuning guide with LoRA to customize your models. We look forward to seeing your innovations with Gemma! This developer documentation provides an overview of the available Gemma models and guides on how to apply and tune them for specific applications.

Integration and Customization:

Gemma models can be embedded into applications on various platforms, such as personal hardware, mobile devices, or hosted services. They are customizable using tuning techniques to excel at specific tasks. Inspired by the Gemini family, Gemma models aim to benefit the AI development community by enhancing and expanding AI capabilities.

Capabilities:

Gemma models generate text and can be fine-tuned for specialized tasks. They offer more targeted and efficient generative AI solutions. The tuning guide, including techniques like LoRA, helps users customize their models.

Model Sizes:

Gemma models come in different sizes to match computing resources, capabilities, and deployment platforms:

Parameters size	Input	Output	Tuned versions	Intended platforms
2B	Text	Text	• Pretrained • Instruction tuned	Mobile devices and laptops
7B	Text	Text	• Pretrained • Instruction tuned	Desktop computers and small servers

Benefits: Gemma models offer high-performance, responsible AI development compared to similarly sized models, providing superior performance based on benchmark evaluation metrics.

3.6 Sentence-T5

Sentence embeddings are broadly useful for language processing tasks. While T5 achieves impressive performance on language tasks cast as sequence-to-sequence mapping problems, it is unclear how to produce sentence embeddings

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.14551

from encoder-decoder models. We investigate three methods for extracting T5 sentence embeddings: two utilize only the T5 encoder and one uses the full T5

encoder-decoder model. [35]

3.7 Testing and its importance

Importance of test case

Test cases are crucial for ensuring the reliability, accuracy, and robustness of a large language model (LLM) like GPT-4. They help identify weaknesses, enhance user trust, and improve interaction quality by simulating real-world scenarios where inputs might be incomplete, erroneous, or ambiguous. By thoroughly evaluating the model's ability to handle diverse and complex prompts, test cases provide valuable feedback for development, facilitate performance benchmarking, and ensure the model's adaptability and compliance with ethical standards. This leads to a more user-friendly and effective application across various domains.

Possible test case scenarios

1. Interrupted Prompts

Case 1.1: Simple Interruption

Input: Start by giving me a story about a brave knight who... (interruption) "random text" "...saved the kingdom from a dragon." Expected Output: The model should continue the story smoothly, ignoring or seamlessly transitioning past the interruption.

Case 1.2: Context Switch and Return

Input: Let's talk about photosynthesis. Oh, by the way, what's the capital of France? Now, back to photosynthesis... Expected Output: The model should correctly answer the interjected question and then return to explaining photosynthesis without losing coherence.

2. Error Injection

Case 2.1: Typographical Errors

Input: What is the captial of Geramny? Expected Output: The model should understand and correct the error, providing the correct response about Germany's capital.

Case 2.2: Malformed Sentences

Input: Explain how photosynthesis works plant energy sun. Expected Output: The model should infer the intended meaning and provide a coherent explanation of photosynthesis.

3. Repetitive Prompts

Case 3.1: Repeated Interruptions

Input: Describe the lifecycle of a butterfly. Wait, what is 2+2? Now, continue about the butterfly lifecycle. Expected Output: The model should handle the interruptions, correctly answering the math question and then returning to the butterfly lifecycle.

4. Abrupt Topic Changes

Case 4.1: Sudden Topic Shift

Input: Tell me about the history of the Roman Empire. Oh, and what's the best way to make chocolate chip cookies? Expected Output: The model should provide an accurate historical summary and then appropriately respond to the cooking question.

5. Incomplete Prompts

Case 5.1: Unfinished Sentences

Input: The most important factor in climate change is... Expected Output: The model should complete the sentence meaningfully, discussing factors related to climate change.

6. Complex Nested Queries

Case 6.1: Multi-part Questions

Input: Can you explain the theory of relativity? Also, tell me who proposed it, when it was proposed, and its significance? Expected Output: The model should answer all parts of the question in a structured and logical manner.

7. Sequential Prompts

Case 7.1: Sequential but Disjoint Prompts

Input: First, tell me about quantum mechanics. Now, explain how it differs from classical mechanics. Lastly, how is it applied in modern technology? Expected Output: The model should maintain context throughout the sequence, providing coherent and relevant responses for each part.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

8. Ambiguous Prompts

Case 8.1: Ambiguous Instructions

Input: Tell me something interesting. Expected Output: The model should produce a response that is generally interesting or ask for clarification.

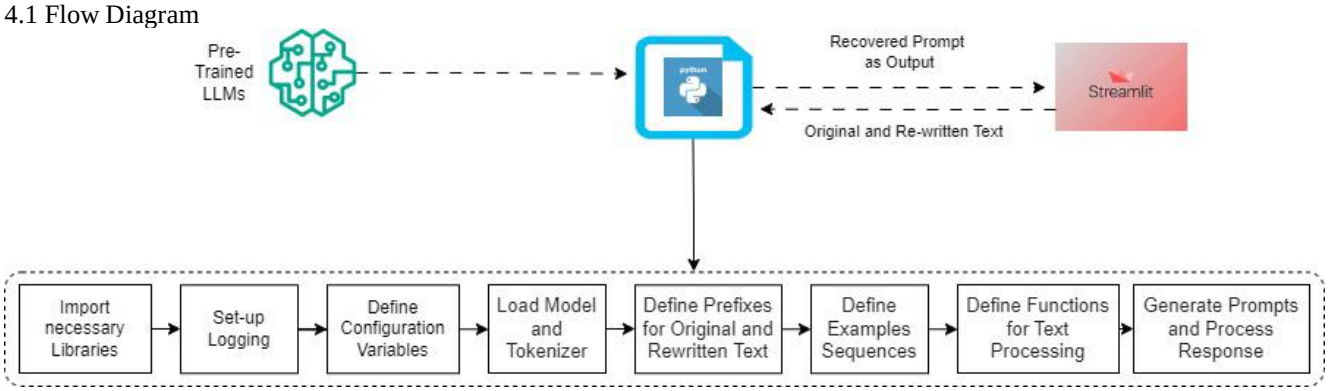
9. Mixed Context Prompts

Case 9.1: Combining Different Contexts

Input: How do you solve quadratic equations? Also, who was the 16th president of the United States, and what is the chemical formula for water? Expected Output: The model should provide accurate answers to all parts, maintaining clarity despite the mixed contexts.

IV. MATERIALS AND METHODS

4.1 Flow Diagram



4.2 Data Generation

We gather a relevant dataset from Kaggle, specifically focusing on forum messages. These messages span various topics, styles, and user interactions.The dataset includes both input text (user queries or forum posts) and corresponding output text (forum responses).We meticulously clean the data, removing noise, irr elevant content, and formatting issues.Tokenization ensures that the text is suitable for input-output pairs.Based on the prompt that is randomly chosen, the 're-written text' is created. Finally the dataset contains the 'id','original text','re-write prompt','re-written text'.

4.3 Tokenization

Tokenization is like slicing a pizza into individual slices.In NLP, it's the process of breaking down text into smaller chunks called "tokens.".Computers understand tokens better than full sentences.It's essential for tasks like text analysis and language modeling.T5 (Text-To-Text Transfer Transformer) is a pre-trained model that can handle various natural language processing (NLP) tasks.It treats all tasks as converting input text to output text, making it versatile.T5-Small is a smaller version of T5 with 60 million parameters.It's great for tasks like summarization.

4.4 Model Training

We are opting Mistral-7B for prompt generation and sentence-t5-base model for inference. The data generated will be based on our template as mentioned in Data Generation. We will be using data generated by Mistral 7B and train sentence-t5-base model. The training process involves fine tuning the LLM(sentence-t5-base model) for our use case.

4.5 Evaluation Metrics for LLMs

It involves assessing LLMs ability to understand, generate, and complete prompts accurately and effectively. Here are the key evaluation metrics:

Cosine Similarity: This checks how close recovered prompts are to originals. It compares "meaningful" words (using techniques like TF-IDF) between prompts as vectors. A score of 1 means perfect match, 0 means no relation.[36] The cosine of two non-zero vectors can be derived using the Euclidean dot product formula:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos\theta$$

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

Given two n-dimensional vectors of attributes, A and B, the cosine similarity, cos(θ), is represented using a dot product and magnitude as

$$\text{cosine similarity} = S_C(\mathbf{A}, \mathbf{B}) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

where Ai and Bi are the i-th components of vectors A and B, respectively. Sharpened Cosine Similarity: Sharpened Cosine Similarity (SCS) is a modified version of the traditional cosine similarity measure used to compare the similarity between two non-zero vectors.

While traditional cosine similarity measures the cosine of the angle between two vectors, giving a value between -1 and 1, Sharpened Cosine Similarity applies a sharpening function to enhance the discrimination between similar and dissimilar vectors.[37]

Sharpening Function: A sharpening function is applied to the cosine similarity score to increase the sensitivity of the measure. This can involve raising the cosine similarity score to a power greater than 1, which accentuates the differences. For example, if α is the sharpening parameter, the Sharpened Cosine Similarity might be calculated as:

$$SCS(A,B) = (\text{cosine\_similarity}(A,B))^{\alpha}$$

Jaccard Similarity: This metric focuses on the ratio of words shared between the original prompt and the recovered prompt. It's simpler than cosine similarity but doesn't account for word importance.[38]

There is also a version of the Jaccard distance for measures, including probability measures. If μ is a measure on a measurable space X,

$$J_\mu(A,B) = \frac{\mu(A \cap B)}{\mu(A \cup B)},$$

then we define the Jaccard coefficient by

and the Jaccard distance by

$$d_\mu(A,B) = 1 - J_\mu(A,B) = \frac{\mu(A \triangle B)}{\mu(A \cup B)}.$$

Care must be taken if μ(A ∪ B) = 0 or∞, since these formulas are not well defined in these cases. BLEU Score(Bi-Lingual Evaluation Understudy): Originally used for machine translation, BLEU score calculates n-gram (sequence of n words) overlap between prompts. It can identify how well the recovered prompt captures the phrasing and word sequence of the original. [39]

Mover's Distance: This metric considers the effort required to "move" one prompt's words to form the other prompt. It takes into account word order and can be a good indicator of syntactic similarity. Embedding Similarity: Techniques like Word Mover's Distance can leverage word embeddings, which capture semantic relationships between words. This allows for a more nuanced understanding of similarity beyond just identical words.[40]

ROUGE Score: Similar to BLEU, ROUGE scores measure overlap between generated text and reference text, with different variations focusing on different aspects (e.g., ROUGE-L for longest common subsequences).
Perplexity: Measures how properly the LLM predicts the subsequent phrase in a sequence. Lower perplexity indicates better language modeling capabilities.

Human Evaluation: Recruit human evaluators to assess the quality, creativity, and overall effectiveness of the generated content. Look for agreement between evaluators for robust results.

A/B Testing: Compare content generated by different LLM prompt recovery techniques to see which one resonates better with human audiences.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

4.6 Model Inference and Results

Use the trained model to recover the original prompts from new or unseen data. LLM will predict the recovery prompt. In the post-prediction process, we clean up the prompt by removing the unwanted characters. We evaluate the feedback iteratively to improve the model and its outputs.

V. FUTURE SCOPE

• We will try with different LLMs and further improve the results.
• Deployment pipeline.
• UI to accept user input and invoke model API endpoint and display the output which is recovered prompt.
• Broader spectrum of use case which could be code generation, test case generation, document generation, code summarization etc.
• Implementation of RAG.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
[2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.
[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.
[4] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. arXiv preprint arXiv:2403.08295, 2024.
[5] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. arXiv preprint arXiv:2108.08877, 2021.
[6] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv preprint arXiv:2402.07927, 2024.
[7] Ronen Eldan Johannes Gehrke Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg Harsha Nori Hamid Palangi Marco Tulio Ribeiro Sébastien Bubeck, Varun Chandrasekaran and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. https://github.blog/ 2023-07-17-prompt-engineering-guide-generative-ai-llms/., 2023.
[8] Alberto D Rodriguez, Katherine R Dearstyne, and Jane Cleland-Huang. Prompts matter: Insights and strategies for prompt engineering in automated software traceability. In 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW), pages 455–464. IEEE, 2023.
[9] 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, 2022.
[10] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. Openprompt: An open-source framework for prompt-learning. arXiv preprint arXiv:2111.01998, 2021.
[11] Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. Prompt perturbation consistency learning for robust language models. arXiv preprint arXiv:2402.15833, 2024.
[12] Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Tong Xiao, and Jingbo Zhu. Efficient prompting methods for large language models: A survey. arXiv preprint arXiv:2404.01077, 2024.
[13] Lei Li, Yongfeng Zhang, and Li Chen. Prompt distillation for efficient llm-based recommendation. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, pages 1348–1357, 2023.
[14] Jiangjiang Zhao, Zhuoran Wang, and Fangchun Yang. Genetic prompt search via exploiting language model probabilities. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, pages 5296–5305. IJCAI, 2023.
[15] Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. Gps: Genetic prompt search for efficient few-shot learning. arXiv preprint arXiv:2210.17041, 2022.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14551

[16] Xavier Amatriain. Prompt design and engineering: Introduction and advanced methods. arXiv preprint arXiv:2401.14423, 2024.
[17] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. arXiv preprint arXiv:2205.12548, 2022.
[18] Wang Chao, Jiaxuan Zhao, Licheng Jiao, Lingling Li, Fang Liu, and Shuyuan Yang. A match made in consistency heaven: when large language models meet evolutionary algorithms. arXiv preprint arXiv:2401.10510, 2024.
[19] Meltem Kurt Pehlivano˘glu, Muhammad Abdan Syakura, and Nevcihan Duru. Enhancing paraphrasing in chatbots through prompt engineering: A comparative study on chatgpt, bing, and bard. In 2023 8th International Conference on Computer Science and Engineering (UBMK), pages 432–437. IEEE, 2023.
[20] Cho-Jui Hsieh, Si Si, Felix X Yu, and Inderjit S Dhillon. Automatic engineering of long prompts. arXiv preprint arXiv:2311.10117, 2023.
[21] Tianjun Zhang, XuezhiWang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. Tempera: Test-time prompting via reinforcement learning. arXiv preprint arXiv:2211.11890, 2022.
[22] Weize Kong, Spurthi Amba Hombaiah, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Prewrite: Prompt rewriting with reinforcement learning. arXiv preprint arXiv:2401.08189, 2024.
[23] Yi Jiang Xi Chen HaoyuWang Shouling Ji Yong Yang, Xuhong Zhang and ZonghuiWang. Prsa: Prompt reverse stealing attacks against large language models. https://arxiv.org/pdf/2402.19200, 2024.
[24] Cheng Li, Mingyang Zhang, Qiaozhu Mei, Weize Kong, and Michael Bendersky. Learning to rewrite prompts for personalized text generation. In Proceedings of the ACM on Web Conference 2024, pages 3367–3378, 2024.
[25] Kaifeng Lyu, Haoyu Zhao, Xinran Gu, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. Keeping llms aligned after fine-tuning: The crucial role of prompt templates. arXiv preprint arXiv:2402.18540, 2024.
[26] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. arXiv preprint arXiv:2010.15980, 2020.
[27] Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. Differentiable prompt makes pre-trained language models better few-shot learners. arXiv preprint arXiv:2108.13161, 2021.
[28] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382, 2023.
[29] Zhaozhuo Xu, Zirui Liu, Beidi Chen, Yuxin Tang, Jue Wang, Kaixiong Zhou, Xia Hu, and Anshumali Shrivastava. Compress, then prompt: Improving accuracy-efficiency trade-off of llm inference with transferable prompt. arXiv preprint arXiv:2305.11186, 2023.
[30] Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-dependent prompt evaluation and optimization with offline inverse rl. In The Twelfth International Conference on Learning Representations, 2023.
[31] Li Sun, Liuan Wang, Jun Sun, and Takayuki Okatani. Prompt prototype learning based on ranking instruction for few-shot visual tasks. In 2023 IEEE International Conference on Image Processing (ICIP), pages 3235–3239. IEEE, 2023.
[32] Chaozheng Wang, Yuanhang Yang, Cuiyun Gao, Yun Peng, Hongyu Zhang, and Michael R Lyu. Prompt tuning in code intelligence: An experimental evaluation. IEEE Transactions on Software Engineering, 2023.
[33] Li Sun, Liuan Wang, Jun Sun, and Takayuki Okatani. Prompt prototype learning based on ranking instruction for few-shot visual tasks. In 2023 IEEE International Conference on Image Processing (ICIP), pages 3235–3239. IEEE, 2023.
[34] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. arXiv preprint arXiv:2309.16797, 2023.
[35] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. arXiv preprint arXiv:2108.08877, 2021.
[36] Peipei Xia, Li Zhang, and Fanzhang Li. Learning similarity with cosine similarity ensemble. Information Sciences, 307:39–52, 2015.
[37] Skyler Wu, Fred Lu, Edward Raff, and James Holt. Exploring the sharpened cosine similarity. arXiv preprint arXiv:2307.13855, 2023.
[38] Sujoy Bag, Sri Krishna Kumar, and Manoj Kumar Tiwari. An efficient recommendation generation using relevant jaccard similarity. Information Sciences, 483:53–64, 2019.
[39] Matt Post. A call for clarity in reporting bleu scores. arXiv preprint arXiv:1804.08771, 2018.
[40] Tom Kenter and Maarten De Rijke. Short text similarity with word embeddings. In Proceedings of the 24th ACM international on conference on information and knowledge management, pages 1411–1420, 2015.