[Week 2] Prompting and Prompt Engineering

ETMI5: Explain to Me in 5

In this section, we delve into the limitations of general AI models in specialized domains, underscoring the significance of domain-adapted LLMs. We explore the advantages of these models, including depth, precision, improved user experiences, and addressing privacy concerns.

We introduce three types of domain adaptation methods: Domain-Specific Pre-Training, Domain-Specific Fine-Tuning, and Retrieval Augmented Generation (RAG). Each method is outlined, providing details on types, training durations, and quick summaries. We then explain each of these methods in further detail with real-world examples. In the end, we provide an overview of when RAG should be used as opposed to model updating methods.

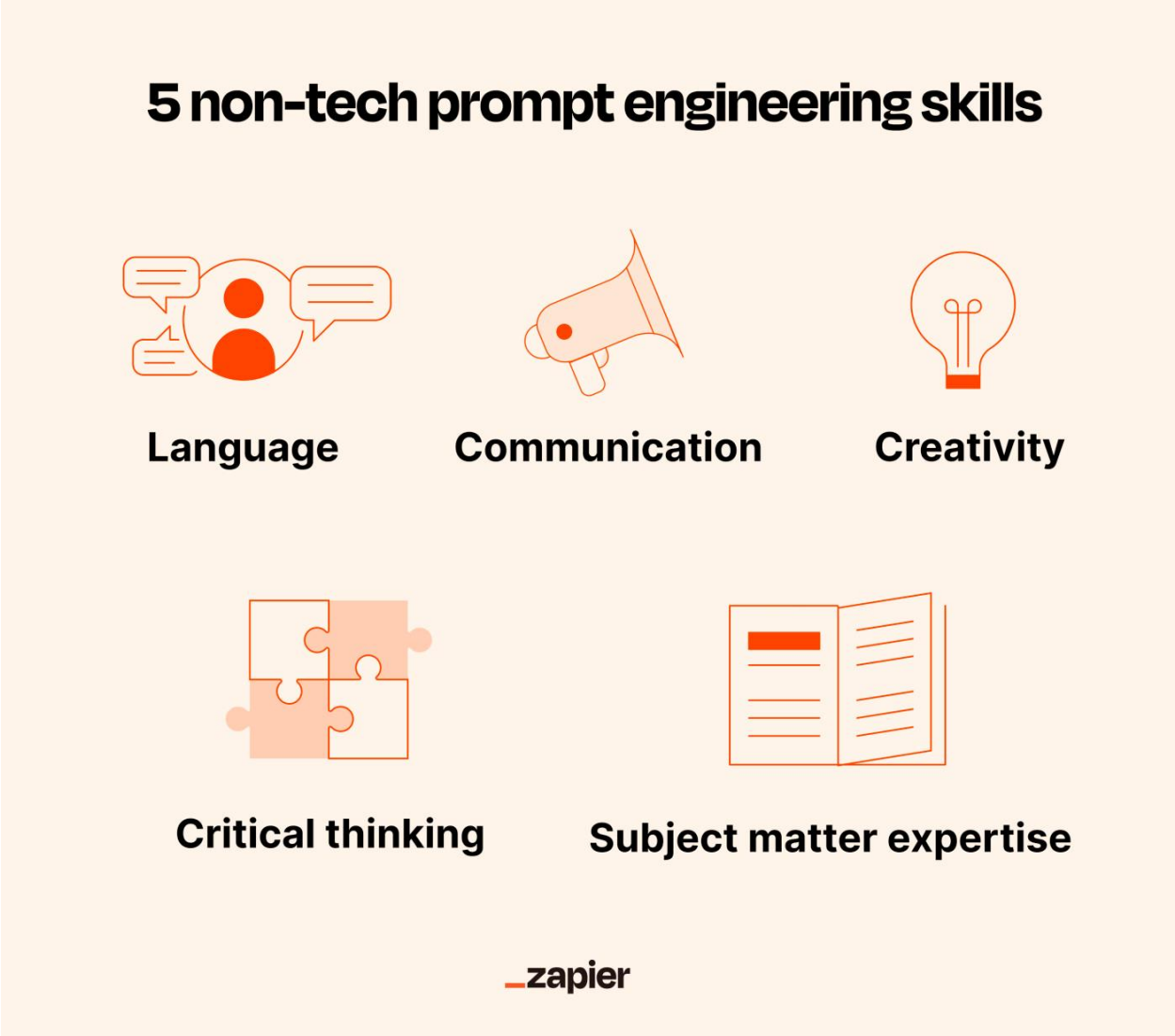
Introduction

Prompting

In the realm of language models, "**prompting**" refers to the art and science of formulating precise instructions or queries provided to the model to generate desired outputs. It's the input—typically in the form of text—that users present to the language model to elicit specific responses. The effectiveness of a prompt lies in its ability to guide the model's understanding and generate outputs aligned with user expectations.

Prompt Engineering

- Prompt engineering, a rapidly growing field, revolves around refining prompts to unleash the full potential of Language Models in various applications.
- In research, prompt engineering is a powerful tool, enhancing LLMs' performance across tasks like question answering and arithmetic reasoning. Users need to leverage these skills to create effective prompting techniques that seamlessly interact with LLMs and other tools.
- Beyond crafting prompts, prompt engineering is a rich set of skills essential for interacting and developing with LLMs. It's not just about design; it's a crucial skill for understanding and exploiting LLM capabilities, ensuring safety, and introducing novel features like domain knowledge integration.
- This proficiency is vital in aligning AI behavior with human intent. While professional prompt engineers delve into the complexities of AI, the skill isn't exclusive to specialists. Anyone refining prompts for models like ChatGPT is engaging in prompt engineering, making it accessible to users exploring language model potentials.



prompting.png

Image Source: https://zapier.com/blog/prompt-engineering/

Why Prompting?

Large language models are trained through a process called unsupervised learning on vast amounts of diverse text data. During training, the model learns to predict the next word in a sentence based on the context provided by the preceding words. This process allows the model to capture grammar, facts, reasoning abilities, and even some aspects of common sense.

Prompting is a crucial aspect of using these models effectively. Here's why prompting LLMs the right way is essential:

1. **Contextual Understanding:** LLMs are trained to understand context and generate responses based on the patterns learned from diverse text data. When you provide a prompt, it's crucial to structure it in a way that aligns with the context the model is familiar with. This helps the model make relevant associations and produce coherent responses.
2. **Training Data Patterns:** During training, the model learns from a wide range of text, capturing the linguistic nuances and patterns present in the data. Effective prompts leverage this training by incorporating similar language and structures that the model has encountered in its training data. This enables the model to generate responses that are consistent with its learned patterns.
3. **Transfer Learning:** LLMs utilize transfer learning. The knowledge gained during training on diverse datasets is transferred to the task at hand when prompted. A well-crafted prompt acts as a bridge, connecting the general knowledge acquired during training to the specific information or action desired by the user.
4. **Contextual Prompts for Contextual Responses:** By using prompts that resemble the language and context the model was trained on, users tap into the model's ability to understand and generate content within similar contexts. This leads to more accurate and contextually appropriate responses.
5. **Mitigating Bias:** The model may inherit biases present in its training data. Thoughtful prompts can help mitigate bias by providing additional context or framing questions in a way that encourages unbiased responses. This is crucial for aligning model outputs with ethical standards.

To summarize, the training of LLMs involves learning from massive datasets, and prompting is the means by which users guide these models to produce useful, relevant, and policy-compliant responses. It's a collaborative process where users and models work together to achieve the desired outcome. There's also a growing field called adversarial prompting which involves intentionally crafting prompts to exploit weaknesses or biases in a language model, with the goal of generating responses that may be misleading, inappropriate, or showcase the model's limitations. Safeguarding models from providing harmful responses is a challenge that needs to be solved and is an active research area.

Prompting Basics

The basic principles of prompting involve the inclusion of specific elements tailored to the task at hand. These elements include:

1. **Instruction:** Clearly specify the task or action you want the model to perform. This sets the context for the model's response and guides its behavior.
2. **Context:** Provide external information or additional context that helps the model better understand the task and generate more accurate responses. Context can be crucial in steering the model towards the desired outcome.
3. **Input Data:** Include the input or question for which you seek a response. This is the information on which you want the model to act or provide insights.
4. **Output Indicator:** Define the type or format of the desired output. This guides the model in presenting the information in a way that aligns with your expectations.

Here's an example prompt for a text classification task:

Prompt:

```
Classify the text into neutral, negative, or positive
Text: I think the food was okay.
Sentiment:
```

In this example:

- **Instruction:** "Classify the text into neutral, negative, or positive."
- **Input Data:** "I think the food was okay."
- **Output Indicator:** "Sentiment."

Note that this example doesn't explicitly use context, but context can also be incorporated into the prompt to provide additional information that aids the model in understanding the task better.

It's important to highlight that **not** all four elements are always necessary for a prompt, and the format can vary based on the specific task. The key is to structure prompts in a way that effectively communicates the user's intent and guides the model to produce relevant and accurate responses.

OpenAI has recently provided guidelines on best practices for prompt engineering using the OpenAI API. For a detailed understanding, you can explore the guidelines here, the below points gives a brief summary:

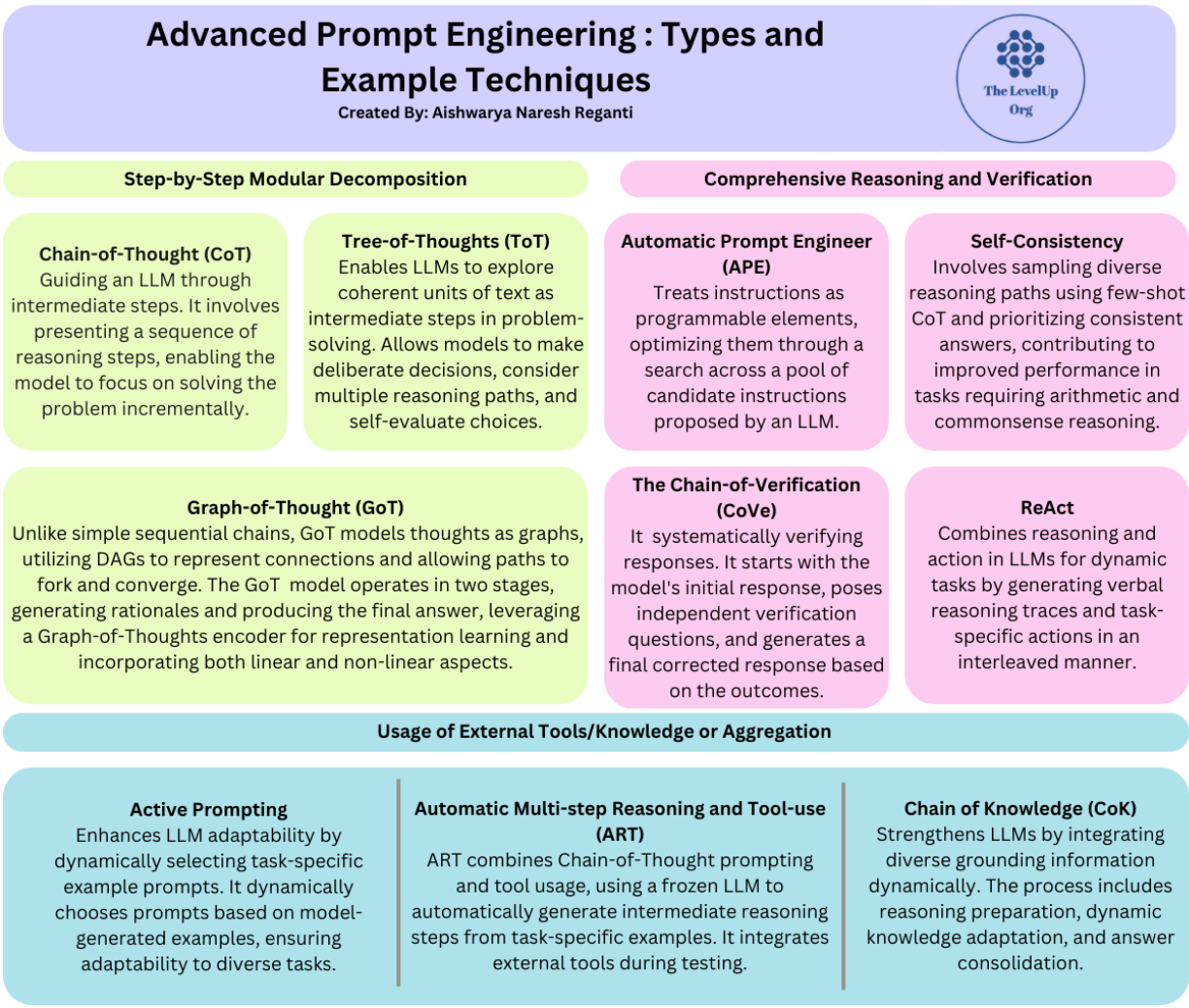
1. **Use the Latest Model:** For optimal results, it is recommended to use the latest and most capable models.
2. **Structure Instructions:** Place instructions at the beginning of the prompt and use ### or """ to separate the instruction and context for clarity and effectiveness.
3. **Be Specific and Descriptive:** Clearly articulate the desired context, outcome, length, format, style, etc., in a specific and detailed manner.
4. **Specify Output Format with Examples:** Clearly express the desired output format through examples, making it easier for the model to understand and respond accurately.
5. **Use Zero-shot, Few-shot, and Fine-tune Approach:** Begin with a zero-shot approach, followed by a few-shot approach (providing examples). If neither works, consider fine-tuning the model.
6. **Avoid Fluffy Descriptions:** Reduce vague and imprecise descriptions. Instead, use clear instructions and avoid unnecessary verbosity.
7. **Provide Positive Guidance:** Instead of stating what not to do, clearly state what actions should be taken in a given situation, offering positive guidance.
8. **Code Generation Specific - Use "Leading Words":** When generating code, utilize "leading words" to guide the model toward a specific pattern or language, improving the accuracy of code generation.

💡 It's also important to note that crafting effective prompts is an iterative process, and you may need to experiment to find the most suitable approach for your specific use case. Prompt patterns may be specific to models and how they were trained (architecture, datasets used etc.)

Explore these examples of prompts to gain a better understanding of how to craft effective prompts in different use-cases.

Advanced Prompting Techniques

Prompting techniques constitute a rapidly evolving area of research, with researchers continually exploring novel methods to effectively prompt models for optimal performance. The simplest forms of prompting include zero-shot, where only instructions are provided, and few-shot, where examples are given, and the language model (LLM) is tasked with replication. More intricate techniques are elucidated in various research papers. While the provided list is not exhaustive, existing prompting methods can be tentatively classified into high-level categories. It's crucial to note that these classes are derived from current techniques and are not exhaustive or definitive; they are subject to evolution and modification, reflecting the dynamic nature of advancements in this field. It's important to highlight that numerous methods may fall into one or more of these classes, exhibiting overlapping characteristics to get the benefits offered by multiple categories.



prompting_11.png

A. Step-by-Step Modular Decomposition

These methods involve breaking down complex problems into smaller, manageable steps, facilitating a structured approach to problem-solving. These methods guide the LLM through a sequence of intermediate steps, allowing it to focus on solving one step at a time rather than tackling the entire problem in a single step. This approach enhances the reasoning abilities of LLMs and is particularly useful for tasks requiring multi-step thinking.

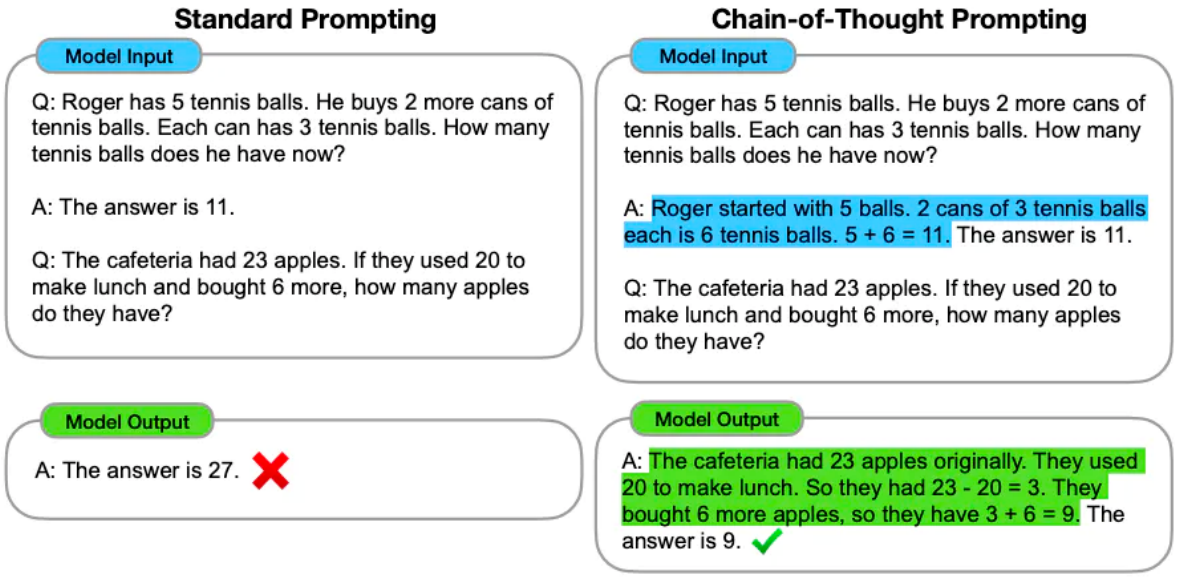
Examples of methods falling under this category include:

1. **Chain-of-Thought (CoT) Prompting:**

Chain-of-Thought (CoT) Prompting is a technique to enhance complex reasoning capabilities through intermediate reasoning steps. This method involves providing a

sequence of reasoning steps that guide a large language model (LLM) through a problem, allowing it to focus on solving one step at a time.

In the provided example below, the prompt involves evaluating whether the sum of odd numbers in a given group is an even number. The LLM is guided to reason through each example step by step, providing intermediate reasoning before arriving at the final answer. The output shows that the model successfully solves the problem by considering the odd numbers and their sums.

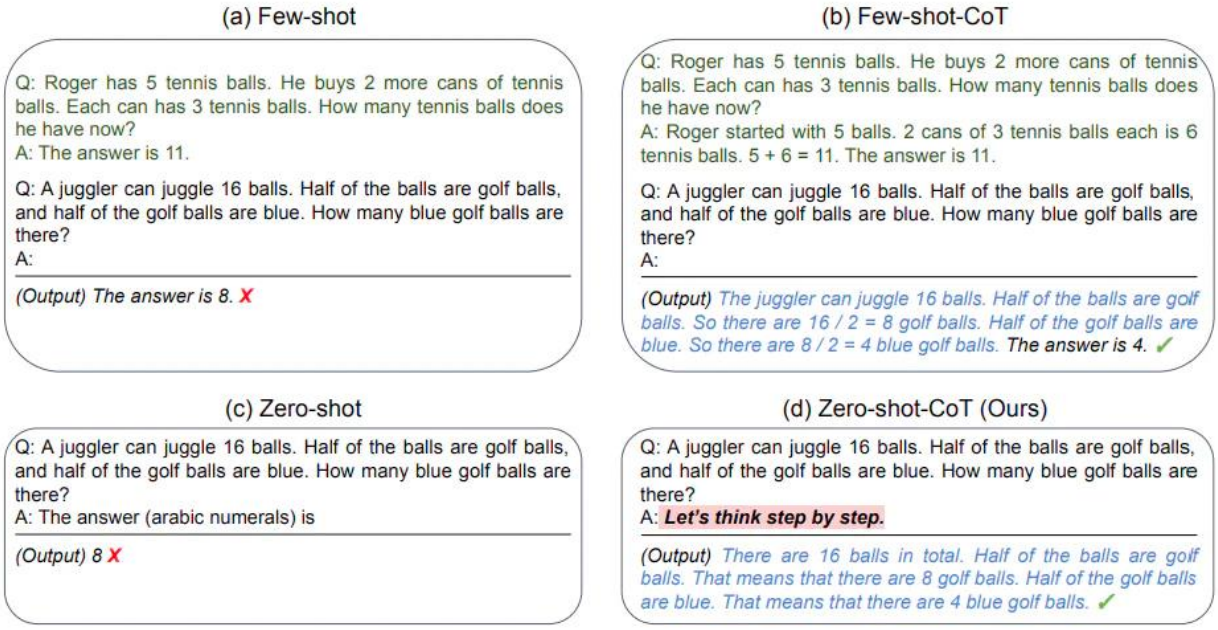


prompting_1.png

Image Source: [Wei et al. (2022)](https://arxiv.org/abs/2201.11903)

1a. **Zero-shot/Few-Shot CoT Prompting:**

Zero-shot involves adding the prompt "Let's think step by step" to the original question to guide the LLM through a systematic reasoning process. Few-shot prompting provides the model with a few examples of similar problems to enhance reasoning abilities. These CoT methods prompt significantly improves the model's performance by explicitly instructing it to think through the problem step by step. In contrast, without the special prompt, the model fails to provide the correct answer.

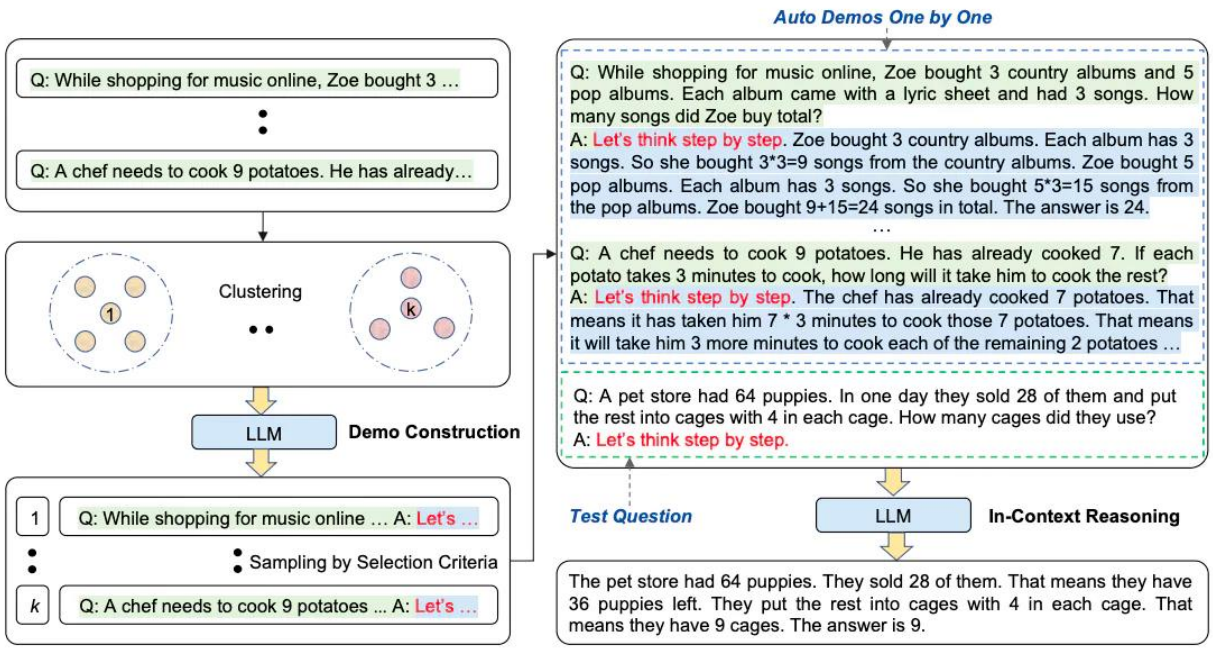


prompting_2.png

Image Source: [Kojima et al. (2022)](https://arxiv.org/abs/2205.11916)

1b. **Automatic Chain-of-Thought (Auto-CoT):**

Automatic Chain-of-Thought (Auto-CoT) was designed to automate the generation of reasoning chains for demonstrations. Instead of manually crafting examples, Auto-CoT leverages LLMs with a "Let's think step by step" prompt to automatically generate reasoning chains one by one.



prompting_3.png

Image Source: Zhang et al. (2022)

The Auto-CoT process involves two main stages:

1. **Question Clustering:** Partition questions into clusters based on similarity.
2. **Demonstration Sampling:** Select a representative question from each cluster and generate its reasoning chain using Zero-Shot-CoT with simple heuristics.

The goal is to eliminate manual efforts in creating diverse and effective examples. Auto-CoT ensures diversity in demonstrations, and the heuristic-based approach encourages the model to generate simple yet accurate reasoning chains.

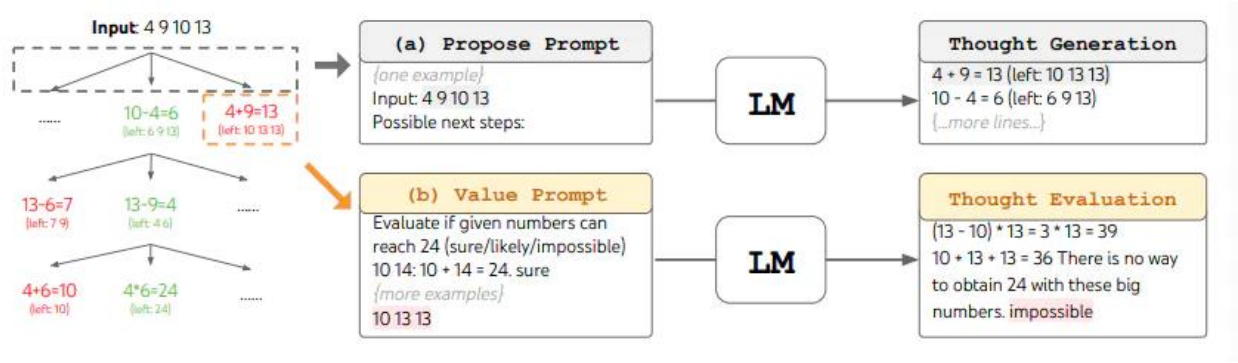
Overall, these CoT prompting techniques showcase the effectiveness of guiding LLMs through step-by-step reasoning for improved problem-solving and demonstration generation.

1. **Tree-of-Thoughts (ToT) Prompting**

Tree-of-Thoughts (ToT) Prompting is a technique that extends the Chain-of-Thought approach. It allows language models to explore coherent units of text ("thoughts") as intermediate steps towards problem-solving. ToT enables models to make deliberate decisions, consider multiple reasoning paths, and self-evaluate choices. It introduces a structured framework where models can look ahead or backtrack as needed during the reasoning process. ToT Prompting provides a more structured and dynamic approach to reasoning, allowing language models to navigate complex problems with greater flexibility and strategic decision-making. It is particularly beneficial for tasks that require comprehensive and adaptive reasoning capabilities.

Key Characteristics:

- **Coherent Units ("Thoughts"):** ToT prompts LLMs to consider coherent units of text as intermediate reasoning steps.
- **Deliberate Decision-Making:** Enables models to make decisions intentionally and evaluate different reasoning paths.
- **Backtracking and Looking Ahead:** Allows models to backtrack or look ahead during the reasoning process, providing flexibility in problem-solving.



prompting_4.png

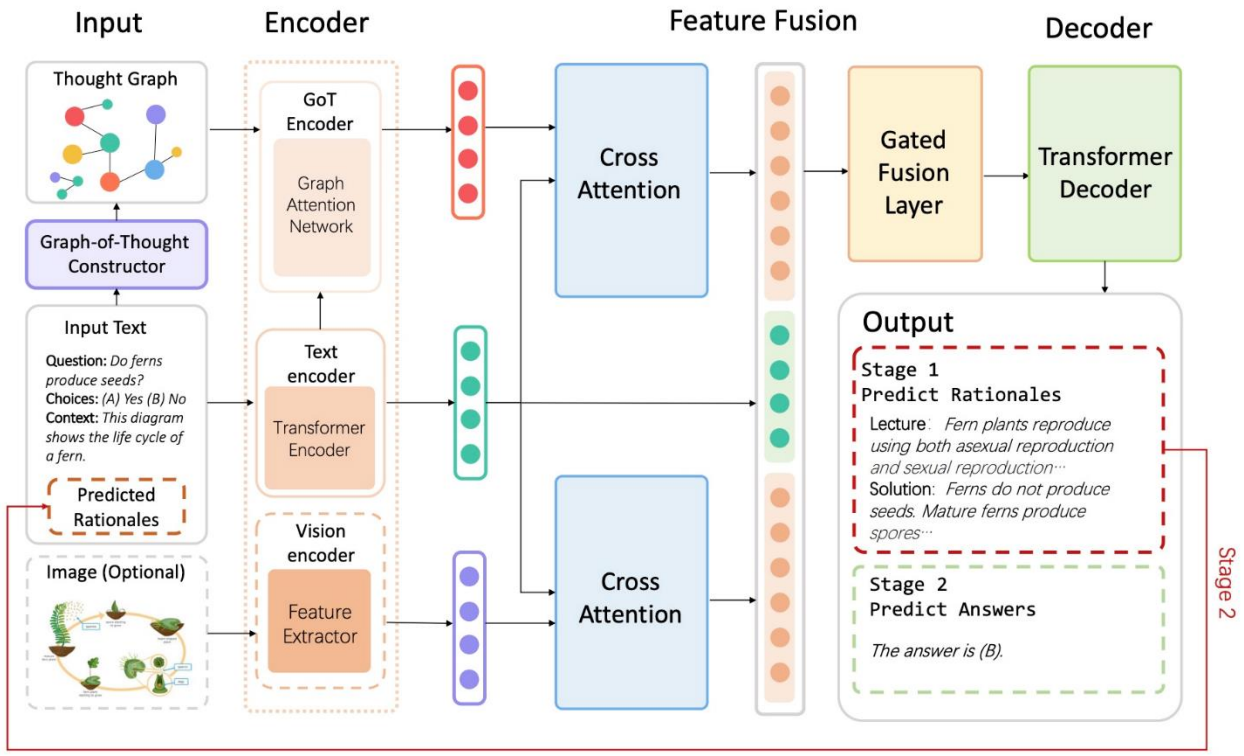
Image Source: Yao et el. (2023)

1. **Graph of Thought Prompting**

This work arises from the fact that human thought processes often follow non-linear patterns, deviating from simple sequential chains. In response, the authors propose Graph-of-Thought (GoT) reasoning, a novel approach that models thoughts not just as chains but as graphs, capturing the intricacies of non-sequential thinking.

This extension introduces a paradigm shift in representing thought units. Nodes in the graph symbolize these thought units, and edges depict connections, presenting a more realistic portrayal of the complexities inherent in human cognition. Unlike traditional trees, GoT employs Directed Acyclic Graphs (DAGs), allowing the modeling of paths that fork and converge. This divergence provides GoT with a significant advantage over conventional linear approaches.

The GoT reasoning model operates in a two-stage framework. Initially, it generates rationales, and subsequently, it produces the final answer. To facilitate this, the model leverages a Graph-of-Thoughts encoder for representation learning. The integration of GoT representations with the original input occurs through a gated fusion mechanism, enabling the model to combine both linear and non-linear aspects of thought processes.



prompting_5.png

Image Source: [Yao et el. (2023)](https://arxiv.org/abs/2305.16582)

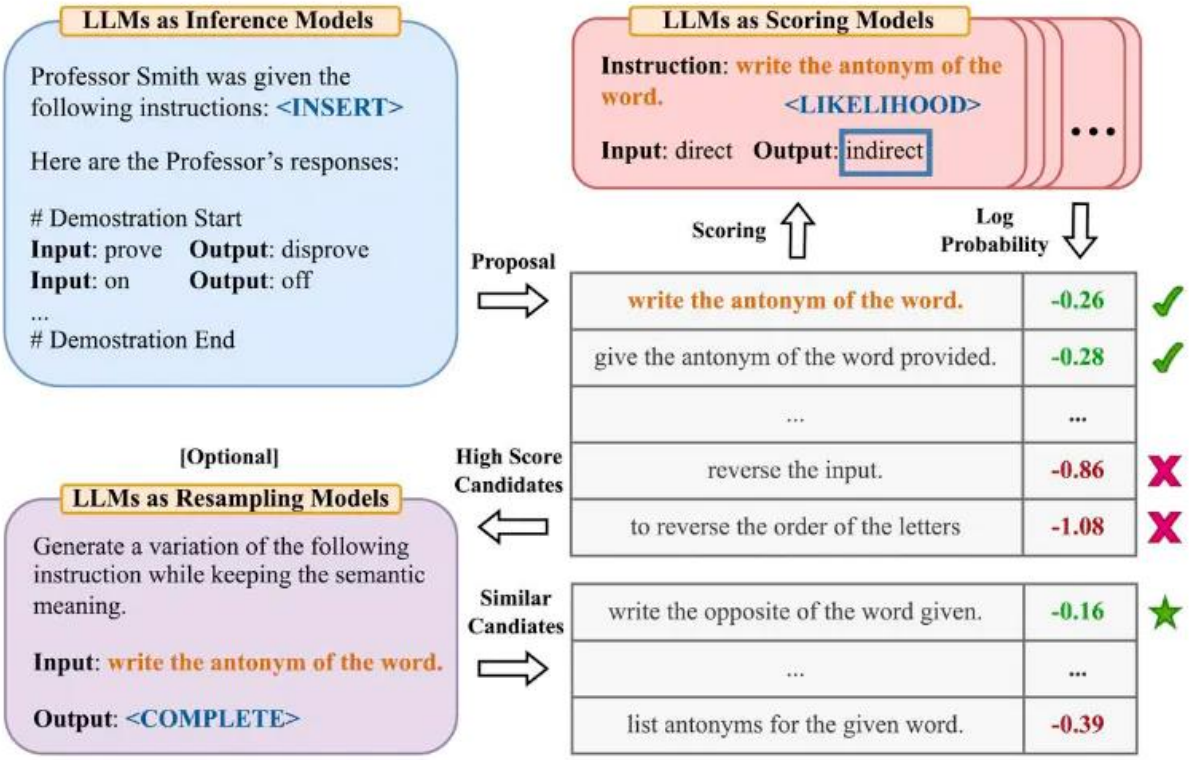
B. Comprehensive Reasoning and Verification

Comprehensive Reasoning and Verification methods in prompting entail a more sophisticated approach where reasoning is not just confined to providing a final answer but involves generating detailed intermediate steps. The distinctive aspect of these techniques is the integration of a self-verification mechanism within the framework. As the LLM generates intermediate answers or reasoning traces, it autonomously verifies their consistency and correctness. If the internal verification yields a false result, the model iteratively refines its responses, ensuring that the generated reasoning aligns with the expected logical coherence. These checks contributes to a more robust and reliable reasoning process, allowing the model to adapt and refine its outputs based on internal validation

1. **Automatic Prompt Engineer**

Automatic Prompt Engineer (APE) is a technique that treats instructions as programmable elements and seeks to optimize them by conducting a search across a pool of instruction candidates proposed by an LLM. Drawing inspiration from classical program synthesis and human prompt engineering, APE employs a scoring function to evaluate the effectiveness of candidate instructions. The selected instruction, determined by the highest score, is then

utilized as the prompt for the LLM. This automated approach aims to enhance the efficiency of prompt generation, aligning with classical program synthesis principles and leveraging the knowledge embedded in large language models to improve overall performance in producing desired outputs.

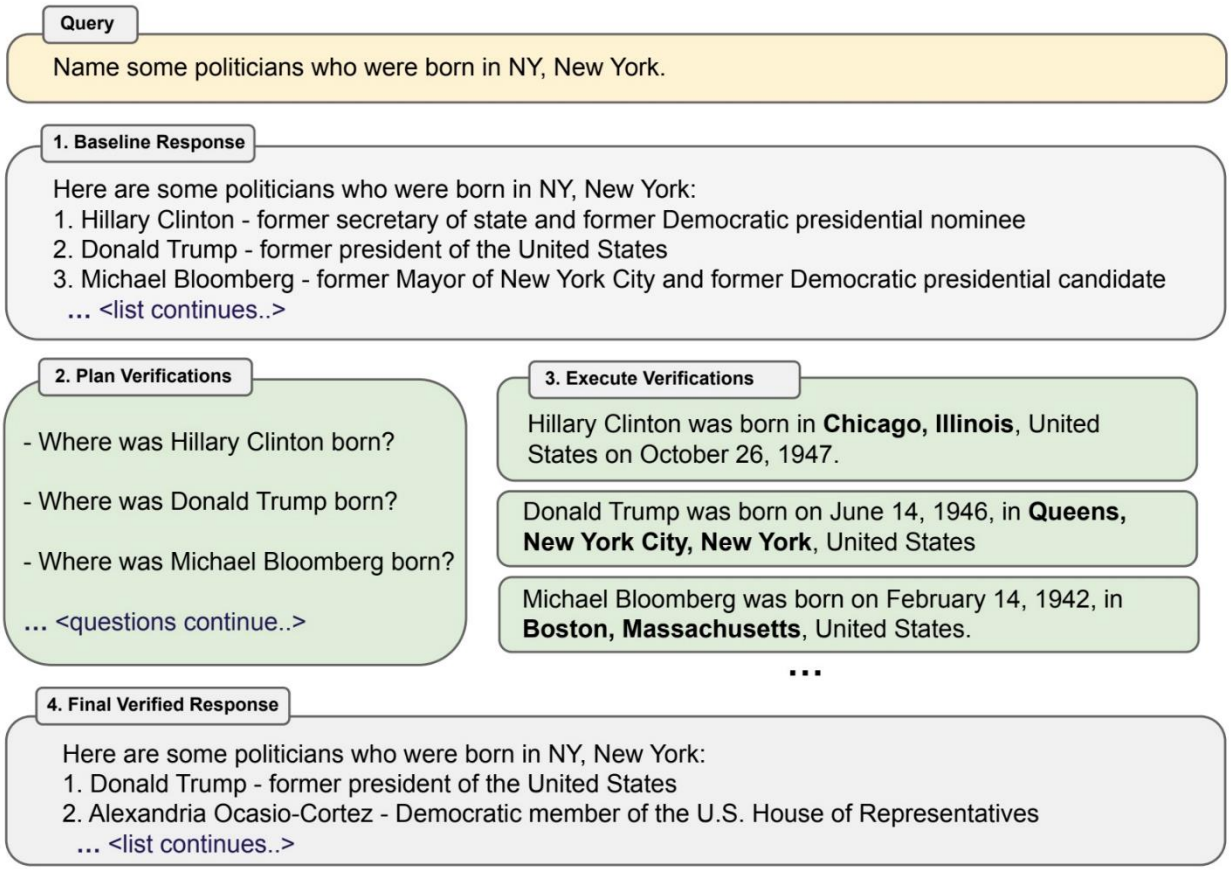


prompting_6.png

Image Source: [Zhou et al., (2022)](https://arxiv.org/abs/2211.01910)

1. **Chain of Verification (CoVe)**

The Chain-of-Verification (CoVe) method addresses the challenge of hallucination in large language models by introducing a systematic verification process. It begins with the model drafting an initial response to a user query, potentially containing inaccuracies. CoVe then plans and poses independent verification questions, aiming to fact-check the initial response without bias. The model answers these questions, and based on the verification outcomes, generates a final response, incorporating corrections and improvements identified through the verification process. CoVe ensures unbiased verification, leading to enhanced factual accuracy in the final response, and contributes to improved overall model performance by mitigating the generation of inaccurate information.

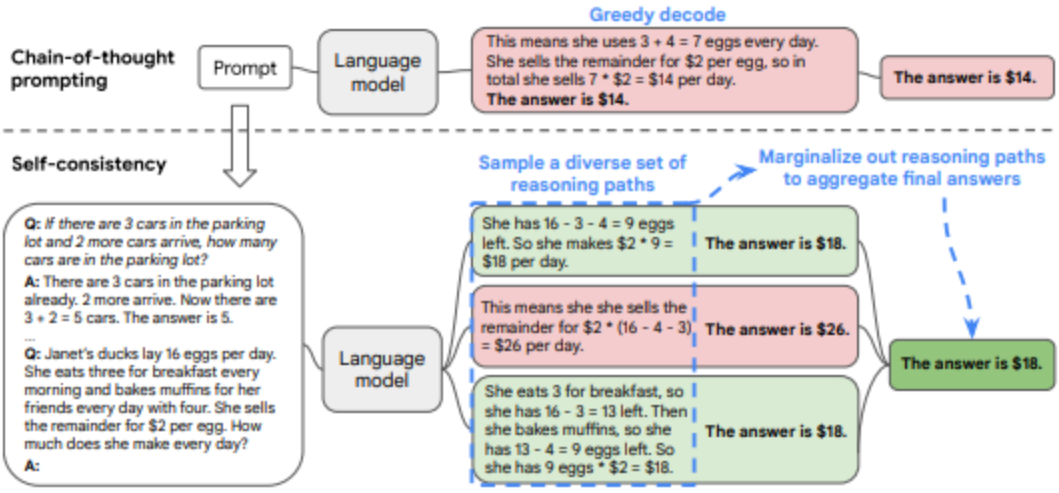


prompting_7.png

Image Source: [Dhuliawala et al.2023](https://arxiv.org/abs/2309.11495)

1. **Self Consistency**

Self Consistency represents a refinement in prompt engineering, specifically targeting the limitations of naive greedy decoding in chain-of-thought prompting. The core concept involves sampling multiple diverse reasoning paths using few-shot CoT and leveraging the generated responses to identify the most consistent answer. This method aims to enhance the performance of CoT prompting, particularly in tasks that demand arithmetic and commonsense reasoning. By introducing diversity in reasoning paths and prioritizing consistency, Self Consistency contributes to more robust and accurate language model responses within the CoT framework.

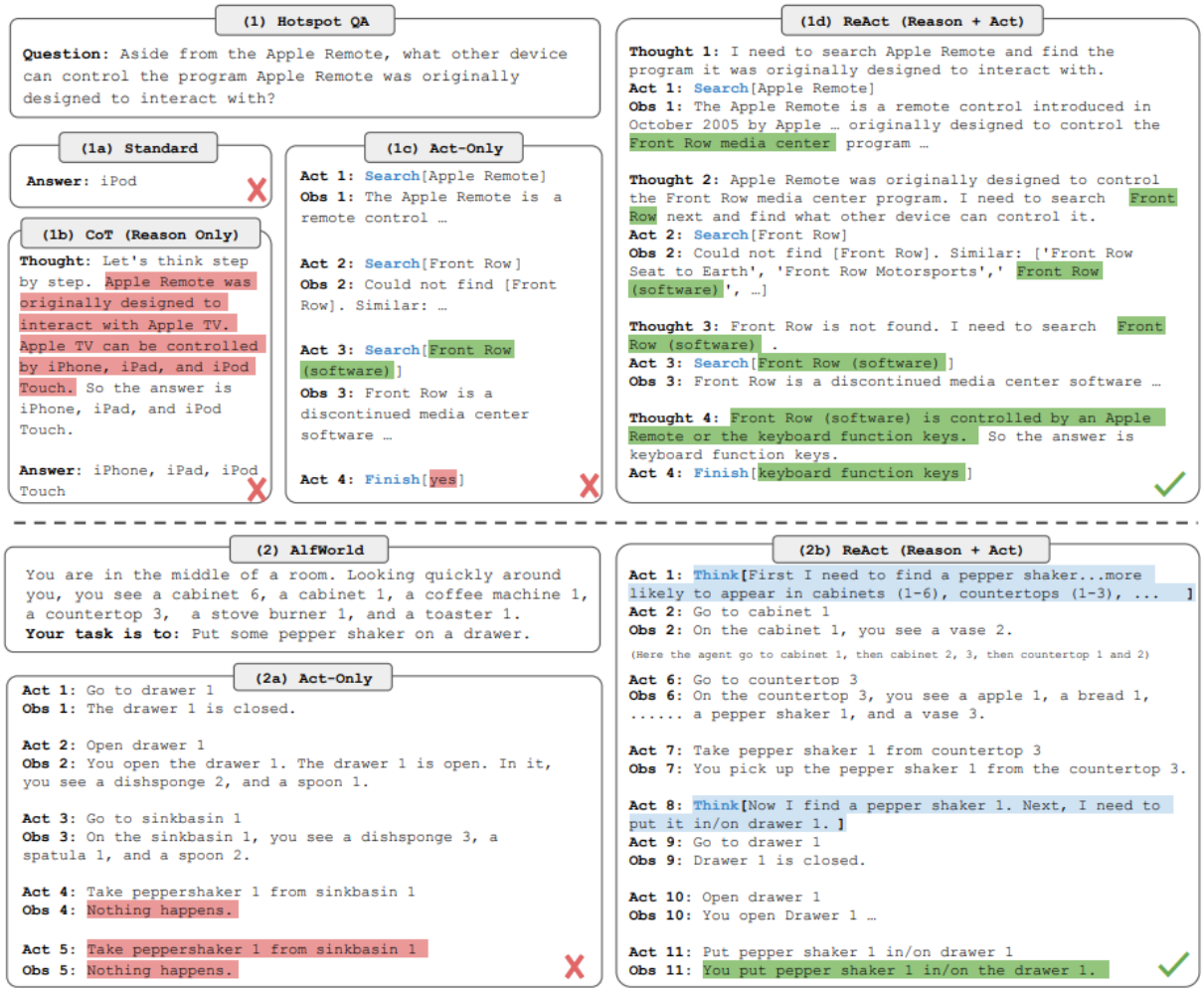


Screenshot 2024-01-14 at 3.50.46 PM.png

Image Source: [Wang et al. (2022)](https://arxiv.org/pdf/2203.11171.pdf)

1. **ReACT**

The ReAct framework combines reasoning and action in LLMs to enhance their capabilities in dynamic tasks. The framework involves generating both verbal reasoning traces and task-specific actions in an interleaved manner. ReAct aims to address the limitations of models, like chain-of-thought , that lack access to the external world and can encounter issues such as fact hallucination and error propagation. Inspired by the synergy between “acting” and “reasoning” in human learning and decision-making, ReAct prompts LLMs to create, maintain, and adjust plans for acting dynamically. The model can interact with external environments, such as knowledge bases, to retrieve additional information, leading to more reliable and factual responses.



Screenshot 2024-01-14 at 3.53.32 PM.png

Image Source: [Yao et al., 2022](https://arxiv.org/abs/2210.03629)

How ReAct Works:

1. **Dynamic Reasoning and Acting:** ReAct generates both verbal reasoning traces and actions, allowing for dynamic reasoning in response to complex tasks.
2. **Interaction with External Environments: T**he action step enables interaction with external sources, like search engines or knowledge bases, to gather information and refine reasoning.
3. **Improved Task Performance:** The framework's integration of reasoning and action contributes to outperforming state-of-the-art baselines on language and decision-making tasks.

4. **Enhanced Human Interpretability:** ReAct leads to improved human interpretability and trustworthiness of LLMs, making their responses more understandable and reliable.

C. Usage of External Tools/Knowledge or Aggregation

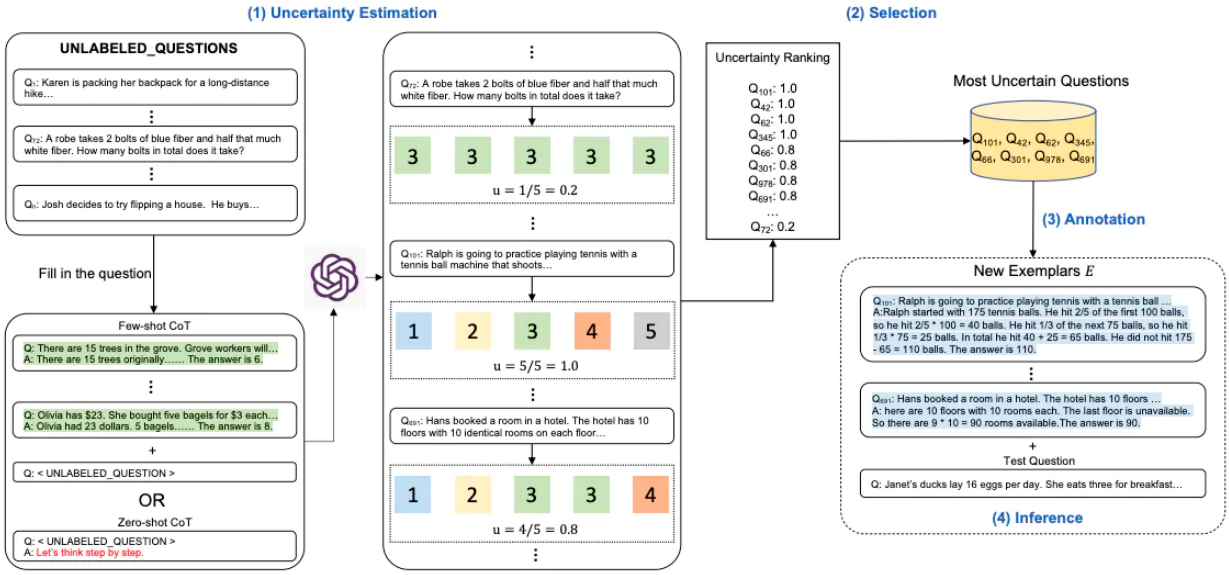
This category of prompting methods encompasses techniques that leverage external sources, tools, or aggregated information to enhance the performance of LLMs. These methods recognize the importance of accessing external knowledge or tools for more informed and contextually rich responses. Aggregation techniques involve harnessing the power of multiple responses to enhance the robustness. This approach recognizes that diverse perspectives and reasoning paths can contribute to more reliable and comprehensive answers. Here's an overview:

1. **Active Prompting (Aggregation)**

Active Prompting was designed to enhance the adaptability LLMs to various tasks by dynamically selecting task-specific example prompts. Chain-of-Thought methods typically rely on a fixed set of human-annotated exemplars, which may not always be the most effective for diverse tasks. Here's how Active Prompting addresses this challenge:

1. **Dynamic Querying:**
   - The process begins by querying the LLM with or without a few CoT examples for a set of training questions.
   - The model generates k possible answers, introducing an element of uncertainty in its responses.
2. **Uncertainty Metric:**
   - An uncertainty metric is calculated based on the disagreement among the k generated answers. This metric reflects the model's uncertainty about the most appropriate response.
3. **Selective Annotation:**
   - The questions with the highest uncertainty, indicating a lack of consensus in the model's responses, are selected for annotation by humans.
   - Humans provide new annotated exemplars specifically tailored to address the uncertainties identified by the LLM.
4. **Adaptive Learning:**
   - The newly annotated exemplars are incorporated into the training data, enriching the model's understanding and adaptability for those specific questions.
   - The model learns from the newly annotated examples, adjusting its responses based on the task-specific guidance provided.

Active Prompting's dynamic adaptation mechanism enables LLMs to actively seek and incorporate task-specific examples that align with the challenges posed by different tasks. By leveraging human-annotated exemplars for uncertain cases, this approach contributes to a more contextually aware and effective performance across diverse tasks.

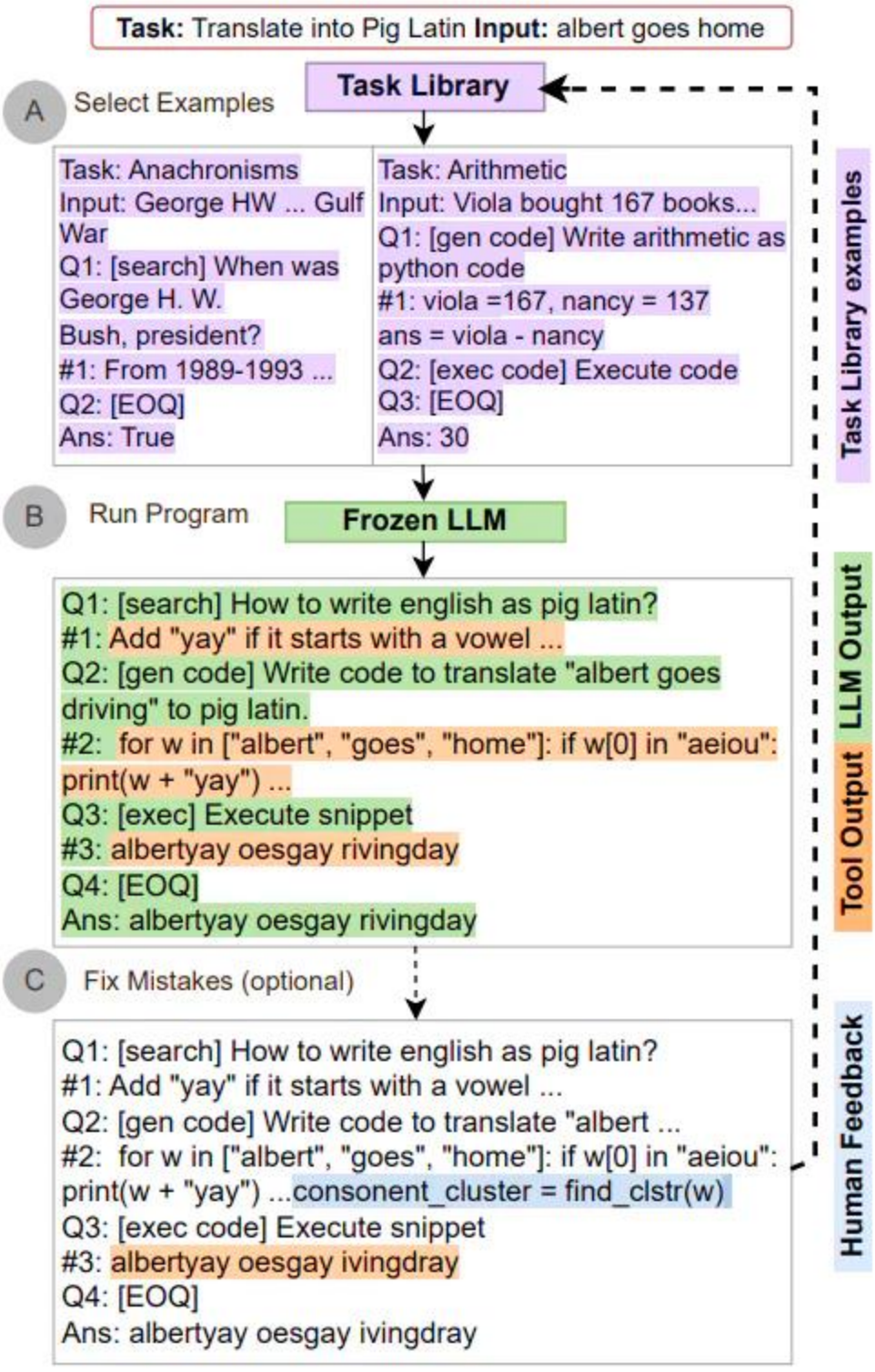


prompting_8.png

Image Source: [Diao et al., (2023)](https://arxiv.org/pdf/2302.12246.pdf)

1. **Automatic Multi-step Reasoning and Tool-use (ART) (External Tools)**

ART emphasizes on task handling with LLMs. This framework integrates Chain-of-Thought prompting and tool usage by employing a frozen LLM. Instead of manually crafting demonstrations, ART selects task-specific examples from a library and enables the model to automatically generate intermediate reasoning steps. During test time, it integrates external tools into the reasoning process, fostering zero-shot generalization for new tasks. ART is not only extensible, allowing for human updates to task and tool libraries, but also promotes adaptability and versatility in addressing a variety of tasks with LLMs.

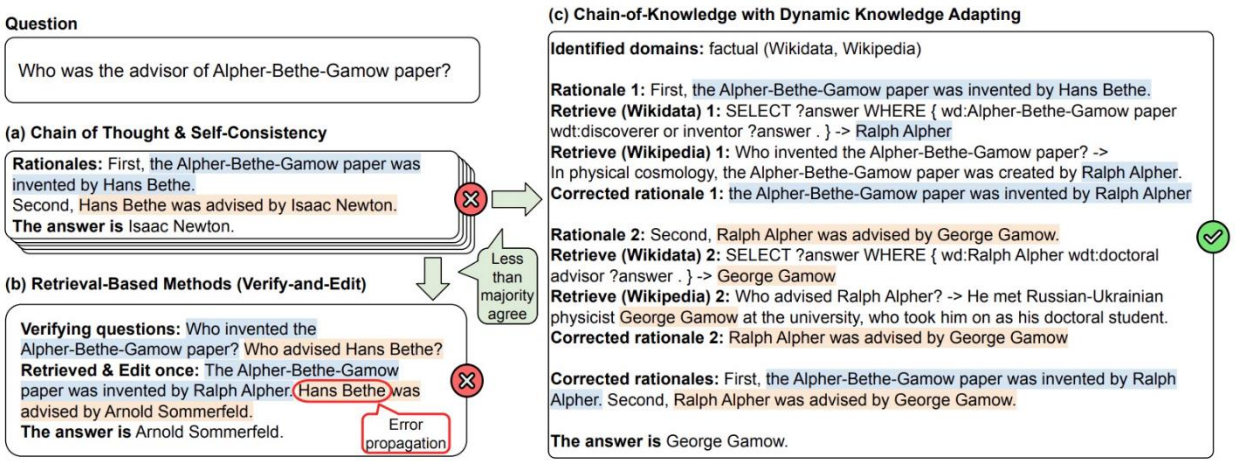


prompting_9.png

Image Source: [Paranjape et al., (2023)](https://arxiv.org/abs/2303.09014)

1. **Chain-of-Knowledge (CoK)**

This framework aims to bolster LLMs by dynamically integrating grounding information from diverse sources, fostering more factual rationales and mitigating the risk of hallucination during generation. CoK operates through three key stages: reasoning preparation, dynamic knowledge adapting, and answer consolidation. It starts by formulating initial rationales and answers while identifying relevant knowledge domains. Subsequently, it refines these rationales incrementally by adapting knowledge from the identified domains, ultimately providing a robust foundation for the final answer. The figure below illustrates a comparison with other methods, highlighting CoK's incorporation of heterogeneous sources for knowledge retrieval and dynamic knowledge adapting.



prompting_10.png

Image Source: [Li et al. 2024](https://arxiv.org/abs/2401.04398)

Risks

Prompting comes with various risks, and prompt hacking is a notable concern that exploits vulnerabilities in LLMs. The risks associated with prompting include:

1. **Prompt Injection:**
   - *Risk:* Malicious actors can inject harmful or misleading content into prompts, leading LLMs to generate inappropriate, biased, or false outputs.
   - *Context:* Untrusted text used in prompts can be manipulated to make the model say anything the attacker desires, compromising the integrity of generated content.
2. **Prompt Leaking:**

- *Risk:* Attackers may extract sensitive information from LLM responses, posing privacy and security concerns.
- *Context:* Changing the user_input to attempt to leak the prompt itself is a form of prompt leaking, potentially revealing internal information.

3. **Jailbreaking:**
   - *Risk:* Jailbreaking allows users to bypass safety and moderation features, leading to the generation of controversial, harmful, or inappropriate responses.
   - *Context:* Prompt hacking methodologies, such as pretending, can exploit the model's difficulty in rejecting harmful prompts, enabling users to ask any question they desire.
4. **Bias and Misinformation:**
   - *Risk:* Prompts that introduce biased or misleading information can result in outputs that perpetuate or amplify existing biases and spread misinformation.
   - *Context:* Crafted prompts can manipulate LLMs into producing biased or inaccurate responses, contributing to the reinforcement of societal biases.
5. **Security Concerns:**
   - *Risk:* Prompt hacking poses a broader security threat, allowing attackers to compromise the integrity of LLM-generated content and potentially exploit models for malicious purposes.
   - *Context:* Defensive measures, including prompt-based defenses and continuous monitoring, are essential to mitigate security risks associated with prompt hacking.

To address these risks, it is crucial to implement robust defensive strategies, conduct regular audits of model behavior, and stay vigilant against potential vulnerabilities introduced through prompts. Additionally, ongoing research and development are necessary to enhance the resilience of LLMs against prompt-based attacks and mitigate biases in generated content.

Popular Tools

Here is a collection of well-known tools for prompt engineering. While some function as end-to-end app development frameworks, others are tailored for prompt generation and maintenance or evaluation purposes. The listed tools are predominantly open source or free to use and have demonstrated good adaptability. It's important to note that there are additional tools available, although they might be less widely recognized or require payment.

1. **PromptAppGPT:**
   - *Description:* A low-code prompt-based rapid app development framework.
   - *Features:* Low-code prompt-based development, GPT text and DALLE image generation, online prompt editor/compiler/runner, automatic UI generation, support for plug-in extensions.

- *Objective:* Enables natural language app development based on GPT, lowering the barrier to GPT application development.

2. **PromptBench:**
   - *Description:* A PyTorch-based Python package for the evaluation of LLMs.
   - *Features:* User-friendly APIs for quick model performance assessment, prompt engineering methods (Few-shot Chain-of-Thought, Emotion Prompt, Expert Prompting), evaluation of adversarial prompts, dynamic evaluation to mitigate potential test data contamination.
   - *Objective:* Facilitates the evaluation and assessment of LLMs with various capabilities, including prompt engineering and adversarial prompt evaluation.
3. **Prompt Engine:**
   - *Description:* An NPM utility library for creating and maintaining prompts for LLMs.
   - *Background:* Aims to simplify prompt engineering for LLMs like GPT-3 and Codex, providing utilities for crafting inputs that coax specific outputs from the models.
   - *Objective:* Facilitates the creation and maintenance of prompts, codifying patterns and practices around prompt engineering.
4. **Prompts AI:**
   - *Description:* An advanced GPT-3 playground with a focus on helping users discover GPT-3 capabilities and assisting developers in prompt engineering for specific use cases.
   - *Goals:* Aid first-time GPT-3 users, experiment with prompt engineering, optimize the product for use cases like creative writing, classification, and chat bots.
5. **OpenPrompt:**
   - *Description:* A library built upon PyTorch for prompt-learning, adapting LLMs to downstream NLP tasks.
   - *Features:* Standard, flexible, and extensible framework for deploying prompt-learning pipelines, supporting loading PLMs from huggingface transformers.
   - *Objective:* Provides a standardized approach to prompt-learning, making it easier to adapt PLMs for specific NLP tasks.
6. **Promptify:**
   - *Features:* Test suite for LLM prompts, perform NLP tasks in a few lines of code, handle out-of-bounds predictions, output provided as Python objects for easy parsing, support for custom examples and samples, run inference on models from the Huggingface Hub.
   - *Objective:* Aims to facilitate prompt testing for LLMs, simplify NLP tasks, and optimize prompts to reduce OpenAI token costs.

Read/Watch These Resources (Optional)

1. https://www.promptingguide.ai/

2. https://aman.ai/primers/ai/prompt-engineering/
3. https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/
4. https://learnprompting.org/courses

Read These Papers (Optional)

1. https://arxiv.org/abs/2304.05970
2. https://arxiv.org/abs/2309.11495
3. https://arxiv.org/abs/2310.08123
4. https://arxiv.org/abs/2305.13626