**8. Prompt Engineering Techniques**

Aarushi Kansal[**1**](https://learning.oreilly.com/library/view/building-generative-ai-powered/9798868802058/html/610509_1_En_8_Chapter.xhtml#Aff2)

(1)

Melbourne, Australia

In this day and age, it’s easy to make use of ChatGPT and other LLMs as a super-powered search engine and ask for information or even small tasks such as summarization. However, prompt engineering goes beyond this and is increasingly becoming a booming and interesting area – with new research and styles of prompting being proposed regularly. Prompt engineering or becoming a prompt engineer is an emerging but highly relevant role in the new wave of generative AI and AI-powered applications.

In this chapter, you’re going to dive into the fascinating world of prompt engineering and learn some of the most recent developments.

**What Is Prompt Engineering?**

Prompt engineering is an emerging field in the realm of artificial intelligence (AI), particularly in the context of language models like GPT-4, Llama 2, and other similar technologies. At its core, prompt engineering involves crafting inputs (prompts) to an AI in a way that elicits the most useful, accurate, or creative responses. It’s a blend of art and science, requiring an understanding of both the technical workings of AI models and the nuances of human language.

**The Role of a Prompt Engineer**

A prompt engineer is akin to a translator or a guide, bridging the gap between human questions or tasks and the AI’s understanding of them. They design prompts that effectively communicate the task at hand to the AI. This role involves the following:

1. 1.

**Understanding the Model’s Capabilities**: Knowing what the AI can and cannot do is crucial. This includes an awareness of its training data, limitations, biases, and strengths.

1. 2.

**Crafting Effective Prompts**: This involves the strategic use of language to guide the AI toward producing the desired outcome. It could be as simple as rephrasing a question or as complex as designing a multi-part prompt with context and instructions.

1. 3.

**Iterative Testing and Refinement**: Prompt engineers often employ a trial-and-error approach, tweaking their prompts based on the AI’s responses to hone in on the most effective formulations.

**Skills and Techniques in Prompt Engineering**

* **Linguistic Skills**: A strong grasp of language and syntax is essential. Understanding how different phrasings can lead to different outcomes is a key part of the job.
* **Technical Knowledge**: Familiarity with AI and machine learning concepts helps in understanding how the model processes information.
* **Creativity and Problem-Solving**: Often, the best prompts come from out-of-the-box thinking, especially when dealing with complex or abstract tasks.
* **Analytical Skills**: Assessing the effectiveness of different prompts requires a methodical approach, often involving data analysis.

**Challenges in Prompt Engineering**

* **Unpredictability**: AI models, especially sophisticated ones like GPT-4, can sometimes produce unexpected or inconsistent results.
* **Model Limitations**: The AI’s knowledge is limited to its training, and it might struggle with concepts or information it hasn’t been trained on.
* **Bias and Ethical** **Considerations**: Prompt engineers must be aware of and work to mitigate biases in AI responses, ensuring ethical use of the technology.

**Future of Prompt Engineering**

As AI models continue to evolve, the field of prompt engineering is likely to grow in importance. It will become more nuanced and possibly even specialized, with prompt engineers working in specific domains like health care, law, or creative writing. Additionally, as models become more sophisticated, the role of a prompt engineer might evolve to include more complex interactions and even dialogue management with AI systems.

Prompt engineering is at the forefront of maximizing the potential of language models in AI. It represents a unique intersection of technical skill and creative language use, making it a vital and intriguing field in the age of advanced AI. As we continue to integrate AI into various aspects of life and work, the skills of a prompt engineer will become increasingly valuable, shaping how effectively we can communicate with and utilize AI technologies.

**Chain of Thought**

**What Is It?**

Chain-of-thought (CoT) prompting is one of the oldest “chain of” methods for improving LLM performance – in particular in the context of queries or tasks that need complex, human-like reasoning to reach an answer.

This approach involves structuring prompts so that the LLM breaks down complex problems into a series of logical, intermediate steps, similar to how a human would when thinking through a problem. The idea is to make the reasoning process of the LLM more transparent and interpretable.

Imagine you’re faced with a complex puzzle, one that requires you to untangle a web of intricate reasoning and abstract thinking. Now, picture a sophisticated AI system equipped with the power of CoT prompting, acting like a detective piecing together clues in a Sherlock Holmes novel. That’s the kind of transformative impact CoT prompting is having on large-scale language models like PaLM, which boasts hundreds of billions of parameters.

In this AI-driven detective story, mathematical problems turn into fascinating mysteries. The AI system, with CoT prompting, will meticulously dissect each part of the problem, laying out calculations step by step, similar to a mathematician explaining a complex theorem on a whiteboard. It’s not just about reaching the answer; it’s about understanding the journey there, with each step unfolding like a chapter in a gripping novel.

But the prowess of CoT prompting isn’t limited to the realm of numbers and equations. It steps into the real world through commonsense reasoning. Here, the system navigates through scenarios filled with human interactions and everyday logic, akin to a wise sage pondering over life’s many riddles. It’s about connecting the dots in a multistep logical reasoning process, mirroring how we, as humans, process and interpret the world around us.

And when it comes to symbolic reasoning, CoT transforms these AI models into abstract thinkers, capable of unravelling logic puzzles and conceptual conundrums that once seemed insurmountable. It’s akin to a philosopher contemplating existential questions, but in the realm of AI.

Basically, CoT prompting isn’t just a technical upgrade; it’s a leap toward making AI systems think and reason more like us – with depth and a nuanced understanding of complex problems.

**Design**

The essence of CoT prompting is to lead the AI through a sequence of reasoning steps. It’s akin to solving a puzzle by laying out each piece methodically rather than trying to visualize the completed image all at once.

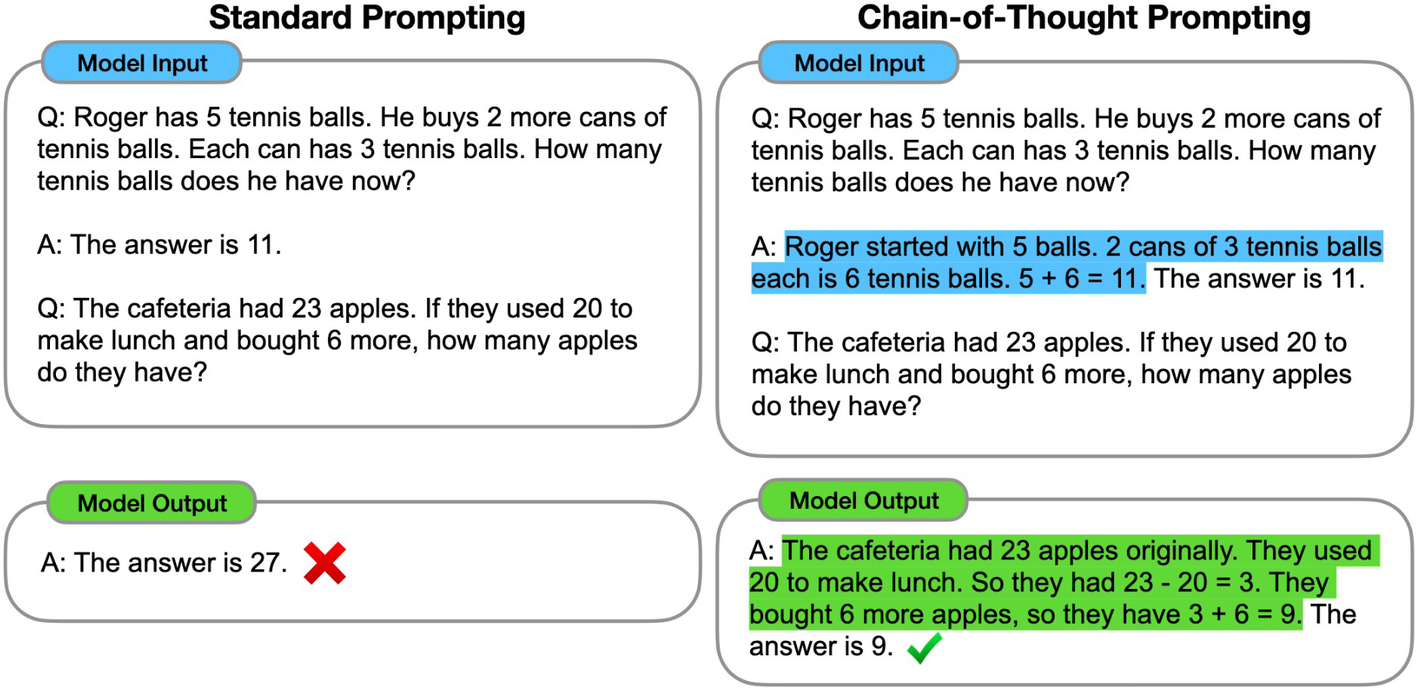
**Few-Shot Exemplars**: A key strategy in CoT prompting is using few-shot exemplars.

**Example**

* **Question**: “A baker has ten loaves of bread. She bakes five more. How many loaves does she have now?”
* **Answer**: “The baker starts with 10 loaves. She bakes 5 more. 10 + 5 = 15. So she now has 15 loaves.”
* **Question**: “Amy had 23 scarves. She knits 13 more, how many scarves does she have now?”

In this case, your prompt consists of a sample question, a sample answer that contains the reasoning, and your actual question for the LLM. This allows the LLM to “understand” how to reason, the same way you as a human would for complex problems, such as arithmetic.

Figure [8-1](https://learning.oreilly.com/library/view/building-generative-ai-powered/9798868802058/html/610509_1_En_8_Chapter.xhtml#Fig1) shows the comparison of a standard prompt and a chain-of-thought prompt, directly from the original paper that proposed CoT.



***Figure 8-1***

Example of chain of thought vs. no chain of thought used in prompt (source: [*https://arxiv.org/pdf/2201.11903.pdf*](https://arxiv.org/pdf/2201.11903.pdf))

An interesting and very useful variant on top of CoT that has emerged is **Zero-Shot CoT**.

**Zero-Shot CoT**: This variant involves adding phrases like “Let’s think step by step” to the original prompt, enhancing the model’s ability to reason even when there are no examples provided.

It mimics how a human being might approach a new problem they haven’t seen before. For example, imagine yourself sitting down with a complex puzzle; you wouldn’t usually solve it in one leap. Instead, you’d approach it step by step, considering different aspects methodically. That’s the essence of Zero-Shot CoT – it’s about instilling this methodical, step-by-step thought process in AI.

This style can be very useful when you don’t have a lot of examples to feed into your prompt.

**Zero-Shot CoT**

1. 1.

**Strategic Cues for AI Reasoning**

* + At the heart of Zero-Shot CoT is the introduction of simple yet powerful cues like “Let’s think step by step.” These phrases are like subtle nudges, encouraging the AI to unpack a question or a problem gradually, akin to how a detective might piece together clues at a crime scene.

1. 2.

**Mimicking Human Cognitive Processes**

* + This approach mirrors how we, as humans, tackle complex issues. We often find it easier to break down a daunting task into smaller, more digestible steps. By incorporating this human-like approach, Zero-Shot CoT essentially guides an LLM to follow a similar path.

1. 3.

**Deepening AI’s Interpretive Skills**

* + In scenarios where a direct or straightforward answer isn’t evident, Zero-Shot CoT is like giving the AI a compass to navigate through the problem’s intricacies. It helps the AI interpret the question thoroughly, deliberate on different elements, and then, step by step, build up to a conclusion.

So taking our original CoT example, with Zero-Shot CoT, it becomes “Amy had 23 scarves. She knits 13 more; how many scarves does she have now? Think step-by-step.” And your LLM answers something to the effect of

“To solve this problem, let’s go through it step by step:

1. 1.

**Starting Amount**: Amy initially has 23 scarves.

1. 2.

**Additional Scarves**: She knits 13 more scarves.

1. 3.

**Total Scarves**: To find out how many scarves she has now, we add the number of scarves she knitted to her initial amount.

So the calculation is

Total Scarves=Initial Scarves+Scarves Knitted

Total Scarves=23+13

Now, let’s do the math.

Amy now has a total of 36 scarves after knitting 13 more.”

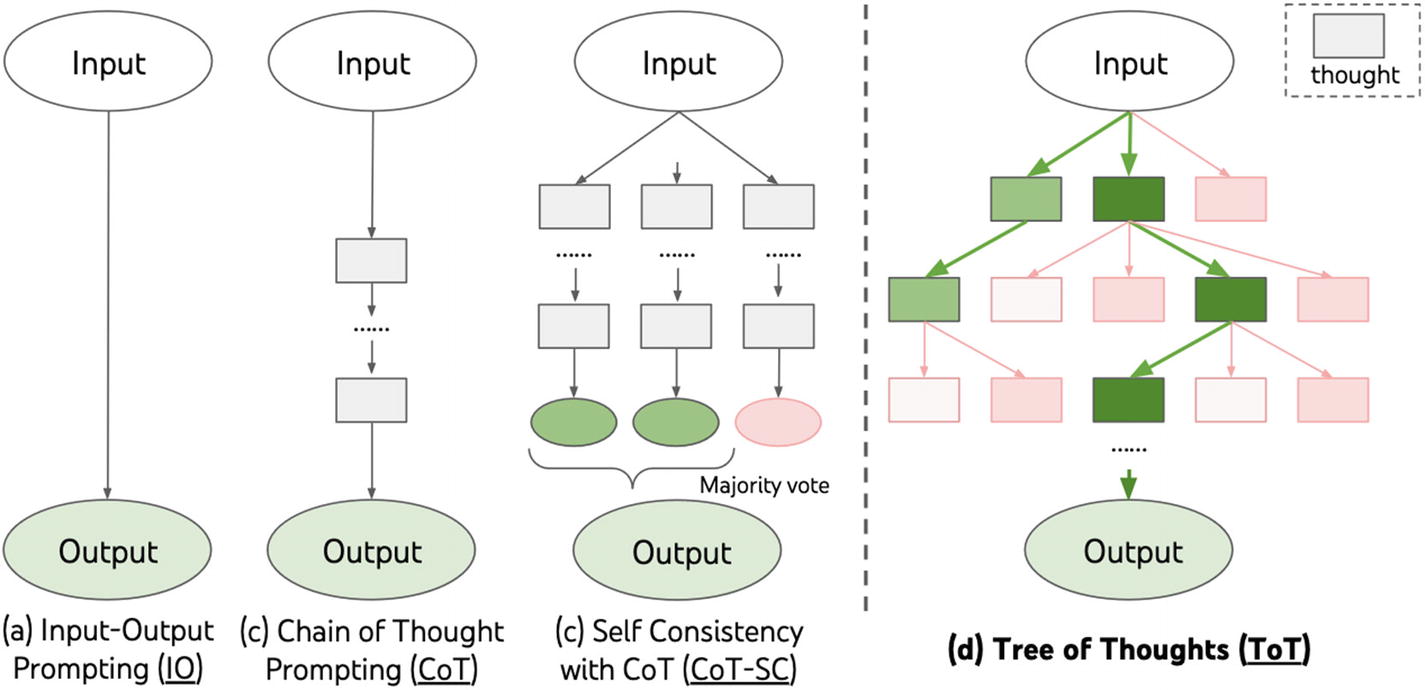
Overall, CoT prompting symbolizes a future where AI can not only replicate but also mirror the depth and complexity of human thinking, a future where AI becomes not just a tool, but a thinking partner.

**Tree of Thought**

Chain of thought has been a groundbreaking development in the prompting engineering space – allowing for LLMs to go from pure text generation tools to problem-solving tools – with almost human-like capabilities.

Tree of thought (ToT) is an advancement on chain-of-thought prompting. The latter essentially instructs the model to break down a complex problem into smaller problems and walk through each problem iteratively. This allows the model to think logically as well as mimics “scratch pad” behavior. Tree of thought takes this style of breaking a problem down further and allows the model to generate multiple thoughts and prune them one by one and eventually arriving at the final, most optimal solution. In tree of thought – the model is able to evaluate thoughts and then backtrack or look forward for better decision-making.

You can see this in Figure [8-2](https://learning.oreilly.com/library/view/building-generative-ai-powered/9798868802058/html/610509_1_En_8_Chapter.xhtml#Fig2).



***Figure 8-2***

Tree of thought representing thoughts as a tree (source: <https://arxiv.org/pdf/2305.10601.pdf>)

**Design**

**Structure of the ToT Framework**

**Initial Thought Generation**: The process begins by generating multiple initial thoughts or solutions, analogous to the root nodes of a tree. Each of these nodes can branch into further thoughts or steps.

**Hierarchical Layering**: The ToT maintains a hierarchical structure, where each layer represents a deeper level of thought or solution refinement.

**Self-Evaluation and Critique**

After generating initial thoughts, the AI model evaluates each thought in relation to the input prompt. This self-critique involves assessing how well each thought or step aligns with the overall problem-solving objective.

This phase could involve ranking each thought or assigning scores based on their utility and relevance to the problem.

**Thought Decomposition and Expansion**

**Decomposition**: The ability to break down problems into smaller segments, allowing the model to address each part individually and iteratively build upon each solution.

**Expansion**: After the initial evaluation, the model expands upon the remaining thoughts, generating further steps and delving deeper into the problem-solving process.

**The Role of the Evaluator**

A critical component of ToT is the evaluator, which assesses potential solutions at each intermediate step. This helps the model determine the viability of potential solutions or whether alternative paths should be explored.

**Deliberate Reasoning**

The ultimate goal is to enable the large language model to deliberately reason its way to a solution. This is achieved through creating models that can propose and evaluate methods contextually.

**Backtracking in the ToT Process**

Backtracking is essential in instances where all generated thoughts for a node are evaluated as unsuitable. The model then returns to a previous layer of the tree to explore alternative nodes, enhancing the effectiveness and efficiency of the problem-solving process.

**Tree Search Techniques**

ToT employs search algorithms like breadth-first search (BFS) and depth-first search (DFS) for systematic exploration. This structure allows for efficient searching through potential solutions, with the model consistently focusing on the most promising paths.

**Dual Roles of the AI Model**

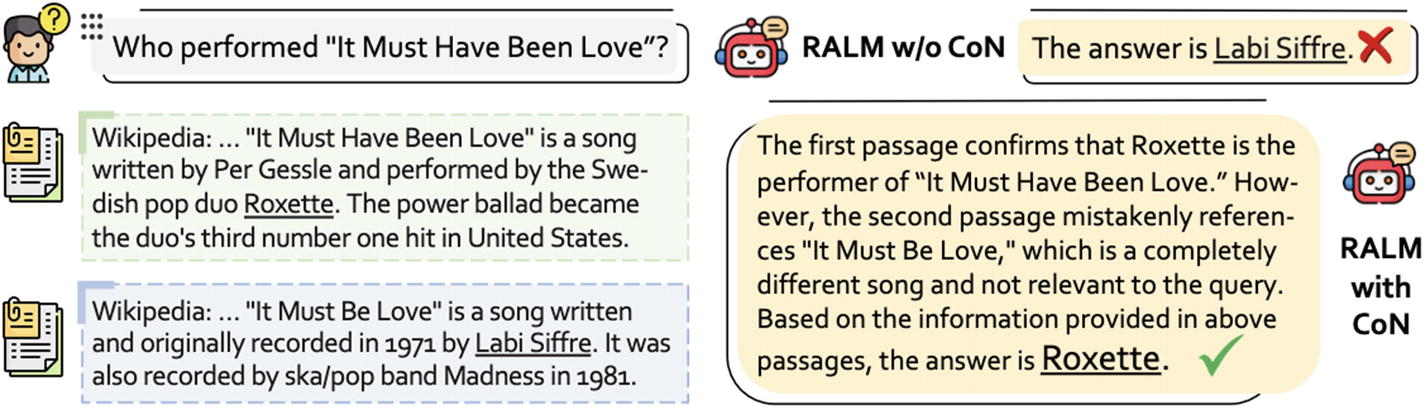
The AI model in ToT performs two distinct roles: the thought generator and the critic. It generates intermediate steps based on the input and previous thoughts and then critiques these for relevance and efficacy.

The tree-of-thought prompting method represents a leap forward in the capabilities of large language models for complex problem-solving. It combines hierarchical thought generation, self-evaluation, and strategic backtracking with a dual role of generation and critique, enabling models to tackle problems with unprecedented depth and efficiency. This method’s ability to iteratively refine and explore a multitude of possibilities before settling on an optimal solution showcases its potential in a variety of applications, from mathematical reasoning to creative writing.

**Chain of Note**

**What Is It?**

In the “chain-of-note” framework, the innovation lies in its ability to generate sequential reading notes for each retrieved document, enhancing the robustness of Retrieval-Augmented Language Models (RALMs). This process allows the model to critically evaluate and filter out irrelevant or misleading information. You can see the core idea of generating summary reading notes compared to not doing so in Figure [8-3](https://learning.oreilly.com/library/view/building-generative-ai-powered/9798868802058/html/610509_1_En_8_Chapter.xhtml#Fig3).



***Figure 8-3***

Example of creating reading notes vs. not and the resulting answers (source: [*https://arxiv.org/pdf/2311.09210.pdf*](https://arxiv.org/pdf/2311.09210.pdf))

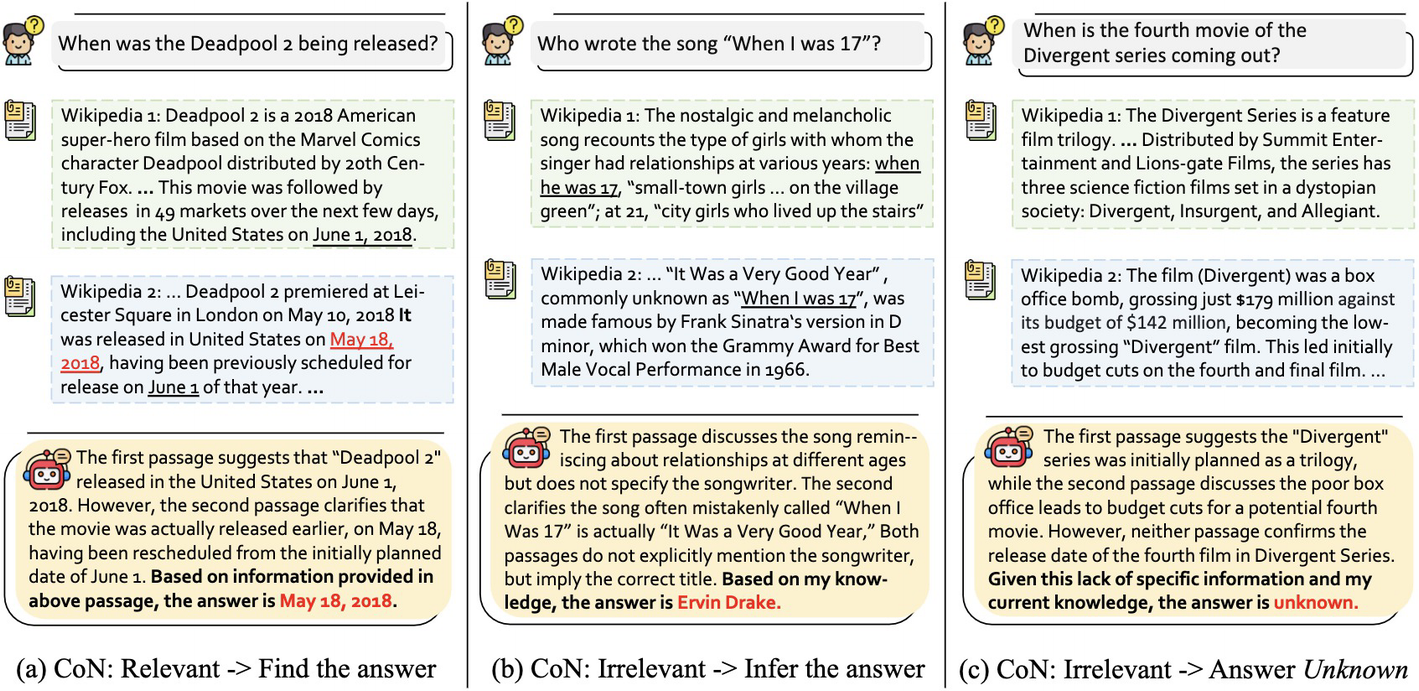
Imagine a scenario where the model is tasked to answer a complex historical question. Instead of directly using the retrieved data, the model creates reading notes, akin to a researcher jotting down key points and their relevance to the question. This method ensures that only pertinent information is considered for the final response.

In cases where the retrieved document is only tangentially related, the model cleverly integrates this context with its built-in knowledge, showcasing an advanced level of comprehension and inference. This is like a historian piecing together facts from different sources to form a coherent narrative.

This approach significantly improves the model’s performance in open-domain question-answering tasks, particularly in handling ambiguous or complex queries. The “chain of note” thus represents a leap forward in creating more reliable and contextually aware AI systems, particularly for applications demanding high accuracy and precision in information retrieval and processing.

**Design**

The crux of this method includes three types of note design as shown in Figure [8-4](https://learning.oreilly.com/library/view/building-generative-ai-powered/9798868802058/html/610509_1_En_8_Chapter.xhtml#Fig4). The first being when a retrieved document clearly contains an answer to the query, the bot creates its own response based on that very document. The second being when the document or documents retrieved don’t contain an answer but do provide enough context so that the model can then make use of the context plus its own baseline knowledge to craft an answer. The third and final one being no relevant answer in the docs retrieved and not enough baseline knowledge in the model to answer – in this case, default answer is unknown.



***Figure 8-4***

Three types of note creations (source: [*https://arxiv.org/pdf/2311.09210.pdf?*](https://arxiv.org/pdf/2311.09210.pdf?))

**Prompt Template**

The base prompt that you can use:

1. 1.

Understand the users’ question and read <documents>.

1. 2.

Write reading notes, with the most important points from these <documents>.

1. 3.

Consider the relevance of the <documents> to the users’ question.

1. 4.

If some documents give you relevant context to the users’ question, give a brief answer based on the passages.

1. 5.

If no document is relevant, give the user a default “Unknown” answer.

Taking into account this is the base prompt, in reality, you can plug in a database or some other data source rather than hard-coding the documents in the prompt – that is, RAG with chain of note.

**Fine-Tuning**

While chain of note is a prompt engineering technique – it does require some fine-tuning to actually give a foundational model such as Llama 2, Falcon, etc., the ability to craft reading notes.

Specifically, in the chain-of-note paper, the researchers used Llama-2 7B to give it note-taking abilities for this framework.

In your own work – you can use another model and fine-tune it on your own data to really make it adaptable to your own niche domain.

**Generated Knowledge Prompting**

**What Is It?**

Generated knowledge prompting is another way to improve the reasoning abilities and reduce hallucination within an LLM. First introduced in the paper “Generated Knowledge Prompting for Commonsense Reasoning” (<https://arxiv.org/pdf/2110.08387.pdf>), this style of prompting started as a way to answer the question of whether extra knowledge within a prompt actually helps improve an LLM or not.

As the name suggests, this entails first generating knowledge via the LLM itself and then incorporating that knowledge with the query, to reason and come up with a reasonable answer.

For example, if you wanted to write an article about LLMs, you would get the LLM to generate a few facts about LLMs and then, based on these facts, get the LLM to write the article.

You can think of this as being quite similar to how you might approach mentoring a junior engineer, without spoon-feeding them solutions. Imagine in a situation you are the tech lead, pair programming with a junior software engineer. You are both working to optimize the performance of a database system in your application. The junior engineer is relatively inexperienced with database optimization.

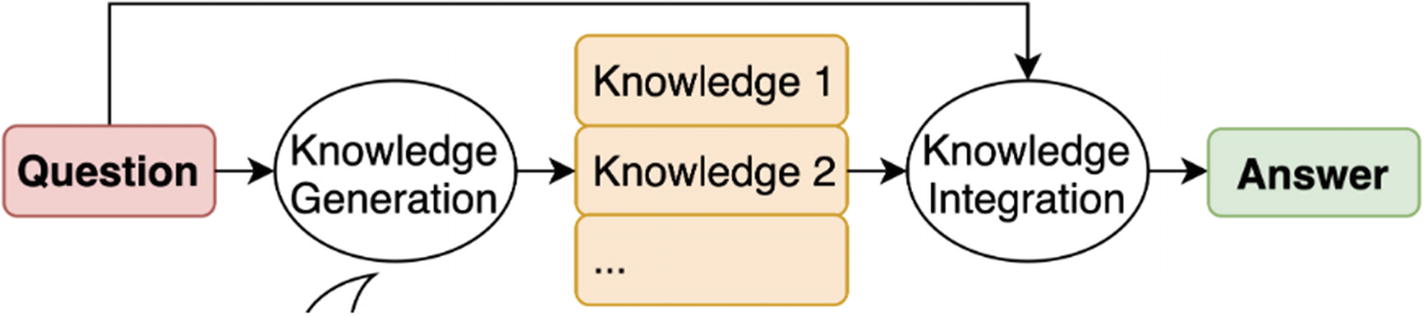
In this case, you might get them to answer questions such as “What factors can affect database performance?” or “Can you name any database optimization techniques you know?” Based on these facts, they might be able to more accurately come up with a solution/answer to optimizing a DB, rather than if they were to go into it without first thinking through the facts already sitting in their brain.

**Design**

As mentioned before, this prompting style involves two steps:

* Knowledge generation
* Knowledge integration

A user queries the LLM; the LLM then generates facts or knowledge; this knowledge is integrated into the query and used to generate an answer, the pipeline you can see in Figure [8-5](https://learning.oreilly.com/library/view/building-generative-ai-powered/9798868802058/html/610509_1_En_8_Chapter.xhtml#Fig5).



***Figure 8-5***

Generated knowledge prompting pipeline (source: [*https://arxiv.org/pdf/2110.08387.pdf*](https://arxiv.org/pdf/2110.08387.pdf))

**Knowledge Generation**

**Objective**: The goal is to generate knowledge statements related to a question that aid in answering it, without directly providing the answer.

**Methodology**

1. 1.

**Preparing the Prompt**: The process begins by creating a prompt for the language model. This prompt includes the following:

* + **Instruction**: A clear directive explaining what is expected from the language model.
  + **Demonstrations**: These are human-written examples specific to the task at hand. Each demonstration includes a question reflective of the task’s style and a knowledge statement that helps in answering such questions.
  + **Question Placeholder**: A spot in the prompt where new, task-related questions can be inserted.

1. 2.

**Demonstration Content**: The demonstrations are carefully crafted. Each consists of the following:

* + A representative question of the task’s challenges. This essentially means choosing questions that require the same skill, reasoning, knowledge, or problem-solving to answer the question, as achieving the task in question. The paper focuses on Numerical Commonsense and Scientific Commonsense:
    1. i.

**Numerical Commonsense**: Questions that require understanding and reasoning about numbers, quantities, and their relationships in real-world contexts. For example, “If a recipe for a cake serves 4 people and uses 2 eggs, how many eggs are needed for 12 people?”

* + 1. ii.

**Scientific Commonsense**: Questions that need an understanding of basic scientific principles or concepts. For instance, “Why do objects feel lighter in water?”

* + A knowledge statement that transforms the problem posed by the question into an explicit reasoning process. It’s crucial that this statement aids in reasoning toward the answer but doesn’t directly answer the question.

**Examples**

Let’s take the example (from the paper) of a question: “Penguins have <mask> wings”.

* **Poor Knowledge Statement**: “Penguins have two wings.” (This directly answers the question, which is not the objective.)
* **Effective Knowledge Statement**: “Birds have two wings. Penguin is a kind of bird.” (This statement facilitates deductive reasoning without directly answering the question. It provides the necessary information for someone to conclude how many wings penguins have, without stating it outright.)

**Generating Knowledge for New Questions**

When a new question q is presented, it is inserted into the placeholder of the prompt. The language model then generates various continuations of this prompt, resulting in a set of knowledge statements Kq = {k1, k2, ..., kM}. Each of these statements offers a piece of information that can be used to infer the answer to the question, aligning with the concept of aiding reasoning rather than providing direct answers.

**Knowledge Integration**

**Concept**: After generating a set of knowledge statements relevant to a particular question, the next step is to use these pieces of knowledge to reach a well-supported answer. This is the essence of knowledge integration.

**The Role of the Inference Model**

* **Function**: This is a language model tasked with making predictions or inferences. It uses the knowledge statements as inputs to help find the most suitable answer to the original question.
* **Operation**: The model processes each knowledge statement alongside the original question. This combination creates new, enriched questions that are augmented with additional context.

**Creation of Augmented Questions**

* **Technique**: This involves appending each knowledge statement to the original question, creating a series of new, expanded questions. Each of these questions contains the original query plus one of the knowledge statements, broadening the context for the answer.

**Determining the Best Answer**

* **Scoring**: For every possible answer, the model calculates a score based on how well each augmented question supports it. The higher the score, the stronger the support the knowledge statement offers for that answer.
* **Selection**: The answer that garners the highest overall score from among these augmented questions is chosen as the most probable or accurate.

**Final Outcome and Selected Knowledge**

* **Prediction**: The end result is the selection of the answer that is best validated by the knowledge statements.
* **Key Information**: The process also identifies which particular knowledge statement provided the most substantial support for the chosen answer, marking it as the most influential or relevant piece of information.

**Flexibility and Application**

* **Model Variability**: This integration step can utilize various forms of language models, ranging from those used straight out of the box (zero-shot models) to those specially tailored or fine-tuned for the task at hand.

Think of knowledge integration as a decision-making process in which an AI system consults a series of expert opinions (the knowledge statements) to answer a question. Each piece of advice is weighed and considered in the context of how well it supports a potential answer. The system then picks the answer best backed up by these expert opinions. This approach ensures a well-informed and substantiated decision, leveraging the AI’s analytical capabilities to sift through complex information and extract the most pertinent insights.

**Food for Thought**

So far I’ve introduced you to a few prompt engineering techniques. There are a lot more such as

* Emotion-based prompting
* Self-consistency
* Multimodal prompting

I recommend you keep yourself up to date on these styles by reading different research papers and keeping up with the open source community (e.g., LangChain repo) as that is where research goes from theory to production ready.

**Conclusion**

Prompt engineering is an up and coming field – not only is it becoming increasingly sought after as a skill, it’s also incredibly fascinating from a technical point of view. Research in this space is moving at a rapid speed, and there are regularly new ways of prompting that are discovered, investigated, and increasingly show improvements in LLM’s capabilities. In this chapter, you were introduced to the fascinating and booming world of prompt engineering. You learned chain of thought, tree of thought, chain of note, and generated knowledge prompting and how they work in detail. These are some of the popular prompting techniques; however, there are plenty more for you to investigate and tailor to your own needs and domain.