

20 SEPTEMBER 2024

The Art and Science of AI Prompt Engineering

Artificial Intelligence (AI) is often regarded as a groundbreaking innovation of the modern era, yet its roots extend much further back than many realize. In 1943, neuroscientist Warren McCulloch and logician Walter Pitts proposed the first computational model of a neuron. The term “Artificial Intelligence” was coined in 1956. The subsequent creation of the Perceptron in 1957, the first model of a neural network, and the expert system Dendral designed for chemical analysis demonstrated the potential of computers to process data and apply expert knowledge in specific domains. From the 1970s to the 1990s, expert systems proliferated. A pivotal moment for AI in the public eye came in 1997 when IBM’s chess-playing computer Deep Blue defeated chess world champion Garry Kasparov.

The new millennium brought a new era for AI, with the integration of rudimentary AI systems into everyday technology. Spam filters, recommendation systems, and search engines subtly shaped online user experiences. In 2006, deep learning emerged, marking the renaissance of neural networks. The landmark development came in 2017 with the introduction of Transformers, a neural network architecture that became the most important ingredient for the creation of Large Language Models (LLMs). Its key component, the attention mechanism, enables the model to discern relationships between words over long distances within a text. This mechanism assigns varying weights to words depending on their contextual importance, acknowledging that the same word can hold different meanings in different situations. However, modern AI, as we know it, was made possible mainly thanks to the availability of large datasets and powerful computational hardware. Without the vast resources of the internet and electronic libraries worldwide, modern AI would not have enough data to learn and evolve. And without modern performant GPUs, training AI would be a challenging task.

The LLM is a sophisticated, multilayer neural network comprising numerous interconnected nodes. These nodes are the micro-decision-makers that underpin the collective intelligence of the system. During its training phase, an LLM learns to balance myriad small, simple decisions, which, when combined, enable it to handle complex tasks. The intricacies of these internal decisions are typically opaque to us, as we are primarily interested in the model’s output. However, these complex neural networks can only process numbers, not raw text. Text must be tokenized into words or sub-words, standardized, and normalized — converted to lowercase, stripped of punctuation, etc. These tokens are then put into a dictionary and mapped to unique numerical values. Only this numerical representation of the text allows LLMs to learn the complex relationships between words, phrases, and concepts and the likelihood of certain words or phrases following one another. LLMs therefore process texts as huge numerical arrays without truly understanding the content. They lack a mental model of the world and operate solely on mathematical representations of word relationships and their probabilities. This focus on the answer with the highest probability is also the reason why LLMs can “hallucinate” plausible yet incorrect information or get stuck in response loops, regurgitating the same or similar answers repeatedly.

Based on the relationships between words learned from texts, LLMs also create vast webs of semantic associations that interconnect words. These associations form the backbone of an LLM’s ability to generate contextually appropriate and meaningful responses. When we provide a prompt to an LLM, we are not merely supplying words; we are activating a complex network of related concepts and ideas. Consider the word “apple”. This simple term can trigger a cascade of associated concepts such as “fruit,” “tree,” “food,” and even “technology” or “computer”. The activated associations depend on the context provided by the prompt and the prevalence of related concepts in the training data. The specificity of a prompt greatly affects the semantic associations an LLM considers. A vague prompt like “tell me about apples” may activate a wide array of diverse associations, ranging from horticultural information about apple trees to the nutritional value of the fruit or even cultural references like the tale of Snow White. An LLM will typically use the

association with the highest occurrence in its training data when faced with such a broad prompt. For more targeted and relevant responses, it is crucial to craft focused prompts that incorporate specific technical jargon or references to particular disciplines. By doing so, the user can guide the LLM to activate a more precise subset of semantic associations, thereby narrowing the scope of the response to the desired area of expertise or inquiry.

LLMs have internal parameters that influence their creativity and determinism, such as “temperature”, “top-p”, “max length”, and various penalties. However, these are typically set to balanced defaults, and users should not modify them; otherwise, they could compromise the ability of LLMs to provide meaningful answers. Prompt engineering is therefore the primary method for guiding LLMs toward desired outputs. By crafting specific prompts, users can subtly direct the model’s responses, ensuring relevance and accuracy. The LLM derives a wealth of information from the prompt, determining not only semantic associations for the answer but also estimating its own role and the target audience’s knowledge level. By default, an LLM assumes the role of a helper and assistant, but it can adopt an expert’s voice if prompted. However, to elicit an expert-level response, one must not only set an expert role for the LLM but also specify that the inquirer is an expert as well. Otherwise, an LLM assumes an “average Joe” as the target audience by default. Therefore, even when asked to impersonate an expert role, an LLM may decide to simplify the language for the “average Joe” if the knowledge level of the target audience is not specified, which can result in a disappointing answer.

Consider two prompts for addressing a technical issue with PostgreSQL:

1. “Hi, what could cause delayed checkpoints in PostgreSQL?”
2. “Hi, we are both leading PostgreSQL experts investigating delayed checkpoints. The logs show checkpoints occasionally taking 3-5 times longer than expected. Let us analyze this step by step and identify probable causes.”

The depth of the responses will vary significantly, illustrating the importance of prompt specificity. The second prompt employs common prompting techniques, which we will explore in the following paragraphs. However, it is crucial to recognize the limitations of LLMs, particularly when dealing with expert-level knowledge, such as the issue of delayed checkpoints in our example. Depending on the AI model and the quality of its training data, users may receive either helpful or misleading answers. The quality and amount of training data representing the specific topic play a crucial role.

Highly specialized problems may be underrepresented in the training data, leading to overfitting or hallucinated responses. Overfitting occurs when an LLM focuses too closely on its training data and fails to generalize, providing answers that seem accurate but are contextually incorrect. In our PostgreSQL example, a hallucinated response might borrow facts from other databases (like MySQL or MS SQL) and adjust them to fit PostgreSQL terminology. Thus, the prompt itself is no guarantee of a high-quality answer—any AI-generated information must be carefully verified, which is a task that can be challenging for non-expert users.

With these limitations in mind, let us now delve deeper into prompting techniques. “Zero-shot prompting” is a baseline approach where the LLM operates without additional context or supplemental reference material, relying on its pre-trained knowledge and the prompt’s construction. By carefully activating the right semantic associations and setting the correct scope of attention, the output can be significantly improved. However, LLMs, much like humans, can benefit from examples. By providing reference material within the prompt, the model can learn patterns and structure its output accordingly. This technique is called “few-shot prompting”. The quality of the output is directly related to the quality and relevance of the reference material; hence, the adage “garbage in, garbage out” always applies.

For complex issues, “chain-of-thought” prompting can be particularly effective. This technique can significantly improve the quality of complicated answers because LLMs can struggle with long-distance dependencies in reasoning. Chain-of-thought prompting addresses this by instructing the model to break down the reasoning process into smaller, more manageable parts. It leads to more structured and comprehensible answers by focusing on better-defined sub-problems. In our PostgreSQL example prompt, the phrase “let’s analyze this step by step” instructs the LLM to divide the processing into a chain of smaller sub-problems. An evolution of this technique is the “tree of thoughts” technique. Here,


the model not only breaks down the reasoning into parts but also creates a tree structure with parallel paths of reasoning. Each path is processed separately, allowing the model to converge on the most promising solution. This approach is particularly useful for complex problems requiring creative brainstorming. In our PostgreSQL example prompt, the phrase “let’s identify probable causes” instructs the LLM to discuss several possible pathways in the answer.

Of course, prompting techniques have their drawbacks. Few-shot prompting is limited by the number of tokens, which restricts the amount of information that can be included. Additionally, the model may ignore parts of excessively long prompts, especially the middle sections. Care must also be taken with the frequency of certain words in the reference material, as overlooked frequency can bias the model’s output. Chain-of-thought prompting can also lead to overfitted or “hallucinated” responses for some sub-problems, compromising the overall result.

Instructing the model to provide deterministic, factual responses is another prompting technique, vital for scientific and technical topics. Formulations like “answer using only reliable sources and cite those sources” or “provide an answer based on peer-reviewed scientific literature and cite the specific studies or articles you reference” can direct the model to base its responses on trustworthy sources. However, as already discussed, even with instructions to focus on factual information, the AI’s output must be verified to avoid falling into the trap of overfitted or hallucinated answers.

In conclusion, effective prompt engineering is a skill that combines creativity with strategic thinking, guiding the AI to deliver the most useful and accurate responses. Whether we are seeking simple explanations or delving into complex technical issues, the way we communicate with the AI always makes a difference in the quality of the response. However, we must always keep in mind that even the best prompt is no guarantee of a quality answer, and we must double-check received facts. The quality and amount of training data are paramount, and this means that some problems with received answers can persist even in future LLMs simply because they would have to use the same limited data for some specific topics.

When the model’s training data is sparse or ambiguous in certain highly focused areas, it can produce responses that are syntactically valid but factually incorrect. One reason AI hallucinations can be particularly problematic is their inherent plausibility. The generated text is usually grammatically correct and stylistically consistent, making it difficult for users to immediately identify inaccuracies without external verification. This highlights a key distinction between plausibility and veracity: just because something sounds right it does not mean it is true.

Whether the response is an insightful solution to a complex problem or completely fabricated nonsense is a distinction that must be made by human users, based on their expertise of the topic at hand. Our clients gained repeatedly exactly this experience with different LLMs. They tried to solve their technical problems using AI, but answers were partially incorrect or did not work at all. This is why human expert knowledge is still the most important factor when it comes to solving difficult technical issues. The inherent limitations of LLMs are unlikely to be fully overcome, at least not with current algorithms. Therefore expert knowledge will remain essential in delivering reliable, high-quality solutions even in the future. As people increasingly use AI tools in the same way they rely on Google — using them as a resource or assistant — true expertise will still be needed to interpret, refine, and implement these tools effectively. On the other hand, AI is emerging as a key driver of innovations. Progressive companies are investing heavily in AI, facing challenges related to security and performance. And this is the area where NetApp can also help. Its [cloud AI focused solutions](#)  are designed to address exactly these issues.

(Picture generated by my colleague Felix Alipaz-Dicke using ChatGPT-4.)



ABOUT THE AUTHOR

Josef Machytka

[View posts](#) 