LLM Prompt Engineering — A Needed Skill for Software Engineering Teams

Published 8 November 2023 - ID G00792273 - 9 min read

By Analyst(s): Van Baker, Haritha Khandabattu, Arun Chandrasekaran

Initiatives: Software Engineering Practices; Adopt Modern Architectures and Technologies; Build a World-Class Software Engineering Organization

Large language models offer breakthrough language generation, but they frequently require prompt engineering to adapt output to the organization's context. To build apps that integrate with LLMs, software engineering leaders must ensure that their teams acquire this critical skill.

Overview

Key Findings

- Large language models (LLMs) are being adopted rapidly. However, there are many different model types with varying degrees of capability, fine-tuning, customizability, learning abilities and domain expertise, making it challenging for enterprises to determine which model to use.
- Massive pretraining makes LLMs a general-purpose model for almost all natural language understanding use cases. However, they are prone to hallucinations.
- Many people mistakenly assume that prompt engineering is just about writing the most efficient prompt, but it is actually much more complex. Prompt engineering includes grounding or retrieval of augmented generation, model instructions, automatic prompt generation, prompt chaining, variable insertion and many other elements.
- Hallucinations, bias and factual errors may come from the model algorithm, however, they can also be caused by data inconsistencies. These issues can be mitigated by techniques such as grounding or the use of customized foundation models, but it is challenging to fully eliminate them.

Recommendations

- Build the skills needed to create well-crafted prompts, including prompt constructs — such as chain of thought and prompt chaining — to ensure quality output from the model. To do so, adopt technologies such as prompt engineering frameworks or vector embeddings, and create model instruction libraries.

- Improve the accuracy of LLM-generated output by adding appropriate metadata to your databases to facilitate the creation of efficient indexes. Further, you should use vector embeddings for unstructured data that is relevant to the use cases you want to support.

- Reduce the likelihood of inappropriate LLM output by building a repository of both general model instructions and instructions that are customized to classes of user prompts.

- Consider smaller foundation models for some use cases by customizing these models with added layers. However, assume that costs will be high and might exceed the usage costs of larger models.

Strategic Planning Assumption

By 2027, over 90% of organizations will use prompt engineering techniques when interacting with LLMs.

Introduction

Training LLMs such as GPT-4, PaLM, BLOOM and other broad-spectrum foundation models requires thousands of graphics processing units (GPUs), massive amounts of data and computer memory, and extended periods of time. As a result, LLMs are too expensive to customize and retrain with data from an individual enterprise. This results in limitations for individual organizations when it comes to using these models and generating responses that are relevant to their specific applications and needs.

Due to the elevated cost of training LLMs, they are closed models that are accessed via API. Additionally, LLMs are stateless, meaning that they do not retain information from one session to the next. Customizing the output that is generated by these models requires a process called prompt engineering.

Description

> **Definition**
>
> Prompt engineering is the discipline of providing inputs — in the form of text or images — to generative AI models to specify and confine the set of responses the model can produce. The inputs prompt a set that produces a desired outcome without updating the actual weights of the model (as done with fine-tuning). Prompt engineering is also referred to as "in-context learning," where examples are provided to further guide the model.

A prompt can be constructed to initiate a series of actions, including a series of steps in interaction with the LLM. The user prompt — that is, the portion of the prompt that the user actually enters into the model — can be simple or complex depending on the skills of the user. Tools allow for automatic prompt generation or prompt chaining, but this portion of the prompt is normally the smaller portion of the overall prompt.

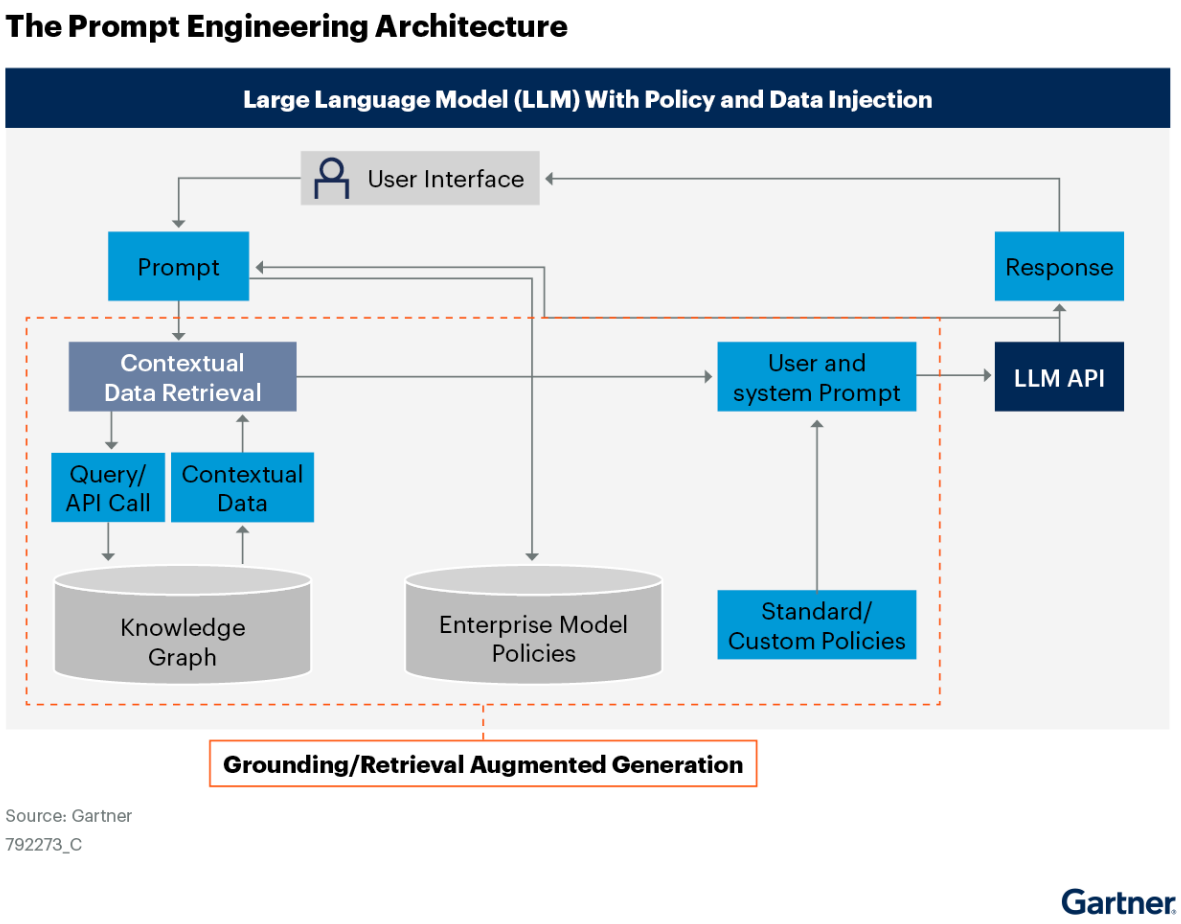
The system prompt is typically the larger portion and is normally invisible to the user. The system prompt can include a process called grounding — or retrieval augmented generation (RAG) — in which contextually relevant data is inserted into the system prompt. This data is then used by the model to compose its response. Grounding significantly reduces and, in some cases, eliminates hallucinations and factual errors.

This data retrieval or grounding process includes data that is relevant to the user prompt. The grounding process relies on searchable data stores to locate and retrieve contextually relevant data in searchable data stores from outside the LLM. This data is then incorporated into the prompt to ground the generative output with factual and new information (see How to Choose an Approach for Deploying Generative AI). Grounding can also include specific instructions to the model on how to operate or how not to operate — for example, data and model policies or instructions on topics to avoid.

In most cases, the user prompt must be supplemented by the system prompt to facilitate accurate output from the LLM model. System prompts normally consist of much more information than user prompts. Prompts are limited based on the number of tokens that the model allows per session. They require near-real-time performance to insert contextual, relevant data and model instructions.

The combination of contextual data and model instructions in the system prompt, as shown in Figure 1, can help minimize hallucinations and factual errors in LLM-generated content. In some cases, the elements of the system prompt can fully eliminate these errors, allowing the output of the model to be used in a fully automated manner.

Figure 1: The Prompt Engineering Architecture


The Prompt Engineering Architecture
Large Language Model (LLM) With Policy and Data Injection
User Interface
Prompt
Response
Contextual Data Retrieval
User and system Prompt
LLM API
Query/ API Call
Contextual Data
Knowledge Graph
Enterprise Model Policies
Standard/ Custom Policies
Grounding/Retrieval Augmented Generation
Source: Gartner
792273_C
Gartner

Gartner

Benefits and Uses

Prompt engineering can be a key differentiator to ensure correct responses and tailor the output to your enterprise's needs.

In addition, prompt engineering facilitates the:

- Customization of large-language-model output.
- Protection of enterprise intellectual property and personally identifiable information.
- Exclusive use of enterprise data to create responses for appropriate use cases.

Risks

Prompt engineering for applications may be subject to several challenges:

- **Prompt hacking can expose sensitive information.** Solutions that use LLMs — especially public-facing solutions — are subject to prompt injection attacks. These can sometimes circumvent input and output filters that are meant to protect sensitive information.
- **Significant investment can be required in infrastructure and software necessary to create indexes of relevant information.** If enterprises do not have the data stores and the metadata required to retrieve very granular data that can be inserted into the model via the system prompt, this can be an expensive effort in both time and materials.
- **Prompt injections and model use bring data security risks that are not easily remediated.** This opens new opportunities for cybersecurity teams to improve and create new products and to align existing solutions. This will help detect and defend against high volumes of prompts and injection attacks on LLM chat and API interfaces.
- **Factual errors are still possible.** LLMs can generate factual errors and hallucinations. Prompt engineering can significantly diminish these, but there is no guarantee that all errors will be eliminated.
- **Prompt engineering skills are deeply technical and rare in the available workforce.** This can lead to the risk of being left behind by your competitors if the skill is not invested in and prioritized properly.

Adoption Rate

Based on a recent Gartner webinar poll, prompt engineering is used by less than 5% of enterprises, but adoption is growing rapidly. [1] Table 1 shows the level of adoption as of July 2023.

Table 1: Prompt Engineering Adoption Rate

Stage of Adoption	Adoption Rate
Currently using in production	3.8%
Currently Piloting	16.7%
Plan to pilot or use in the next six months	11.4%
Plan to pilot or use within the next 12 months	23.5%
No plans to use for now	44.7%
Note: Due to rounding, percentages may not add up to 100%	

Source: Gartner (July 2023)

Alternatives

The use of fine-tuning — for models that allow fine-tuning for the creation of customized domain-specific foundation models — can be an alternative to prompt engineering. Fine-tuning is the process of taking a pretrained LLM as a starting point and further training it on a new dataset.

LLM fine-tuning is emerging as a viable option, with many vendors now offering fine-tuning via their APIs and user interfaces. Fine-tuning is not the only alternative available to create custom applications that align LLMs and incorporate new domain-specific data. Therefore, it is key to understand the alternatives when deciding whether or not to use fine-tuning for each LLM use case.

Organizations, however, should consider LLM fine-tuning if they are trying to achieve at least one of the following objectives:

- **LLM alignment:** Fine-tuning can be used to change the LLM's behavior and better align its underlying capabilities with the task at hand. This will improve the consistency and performance of LLM-based applications. For example, an organization can fine-tune an LLM (via supervised fine-tuning) to match the communication style required, to better moderate conversations, and to be a more useful assistant for the target customer interactions. This typically requires a relatively small dataset of high-quality prompts and completions to display the required behaviors to the LLM.
- **Include additional data:** LLMs can be fine-tuned with additional data to increase their knowledge on certain domain-specific areas that the LLM was not originally trained for. For example, a financial services organization could fine-tune an LLM with its own financial documents to incorporate this knowledge into the model. This typically requires a larger dataset of unlabeled data.

Recommendations

- Build the skills needed to create well-crafted prompts, including prompt constructs — such as chain of thought and prompt chaining — to ensure quality output from the model. You can do so by adopting technologies such as prompt engineering frameworks or vector embeddings, and by creating model instruction libraries. This expertise will span multiple teams, including software engineers, data scientists and database engineers.
- Improve the accuracy of LLM-generated output by adding appropriate metadata to your databases to facilitate the creation of efficient indexes. Further, you can use vector embeddings for unstructured data that is relevant to the use cases you want to support. Consider enterprise search and graph databases as an alternative for grounding solutions where they can provide data with appropriate granularity.
- Reduce the likelihood of inappropriate LLM output by building a repository of model instructions that include both general model instructions and customized instructions that are unique to classes of user prompts. This can not only help with both hallucinations and factual errors, but also prevent discussion of topics where you don't want the model to respond.
- Consider smaller foundation models for some use cases by customizing these models with added layers. However, assume that costs for this will be high and might exceed the usage costs of larger models. Using transfer learning to create a base model that is conversational before adding layers of your specific data to customize the model can reduce costs.

Evidence

This research is based on extensive conversations with vendors providing very large language models. These LLMs require prompt engineering in most use cases where accurate model output is needed.

[1] For more information, see Client Webinar: How Generative AI Is Impacting Software Engineering, conducted 25 July 2023, with 250 attendees.

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

Prompt Engineering With Enterprise Information for LLMs and GenAI

Quick Answer: How Will Prompt Engineering Impact the Work of Data Scientists?

How Large Language Models and Knowledge Graphs Can Transform Enterprise Search

AI Design Patterns for Knowledge Graphs and Generative AI

Table 1: Prompt Engineering Adoption Rate

Stage of Adoption	Adoption Rate
Currently using in production	3.8%
Currently Piloting	16.7%
Plan to pilot or use in the next six months	11.4%
Plan to pilot or use within the next 12 months	23.5%
No plans to use for now	44.7%
Note: Due to rounding, percentages may not add up to 100%	

Source: Gartner (July 2023)