Prompt Engineering With Enterprise Information for LLMs and GenAI

Published 12 July 2023 - ID G00792014 - 23 min read

By Analyst(s): Darin Stewart

Initiatives: Digital Workplace and CRM for Technical Professionals

Training and fine-tuning custom large language models for generative artificial intelligence is beyond the capabilities of most organizations. Technical professionals can use prompt engineering and in-context learning to meet the needs of most generative AI scenarios.

Overview

Key Findings

- Training and fine-tuning custom large language models (LLMs) is impractical for most organizations. The technical and computational requirements call for very mature data management practices and significant data science expertise.
- Most enterprise use cases can be accomplished with a combination of prompt engineering and in-context learning to ground LLM responses in enterprise information and constraints. Several approaches to prompt creation are available to support generative AI (GenAI) applications.
- Retrieval augmented generation is a practical approach to prompt engineering that incorporates private and proprietary enterprise information into GenAI applications without requiring the underlying model to be modified in any way.

Recommendations

Technical professionals planning to incorporate LLMs and GenAI into enterprise applications should:

- **Adopt prompt engineering and in-context learning as the primary means to interact with LLMs.** Prompt engineering and in-context learning are capable of meeting the requirements of most applications and should be the main method for utilizing an LLM. Tuning and training models should be reserved for high-value use cases that cannot be accomplished otherwise.
- **Use retrieval augmented generation to incorporate enterprise information into LLM interactions.** Incorporating private and proprietary information into prompts is fundamental to in-context learning for enterprise applications. A systematic and programmatic approach to finding and including this information is necessary for ongoing, scalable solutions.
- **Create GenAI application knowledge bases of curated enterprise content.** Optimize content for prompts by copying relevant enterprise information into application specific repositories and splitting it into more granular document or file fragments with a systematic naming convention and metadata.

Analysis

GenAI will find its way into the enterprise. In some form or other, its introduction is inevitable. The debut of ChatGPT caught the imagination of the public and set digital leaders scrambling to find ways to use the underlying technology, LLMs, to their greatest advantage. Microsoft and Google have both announced the embedding of GenAI in their core offerings. NVIDIA crossed the trillion-dollar valuation threshold on the wave of GenAI enthusiasm due to the need for graphics processing units (GPUs) to support LLMs. Niche players and fly-by-night startups are proliferating with the promise of transforming productivity and quality in every aspect of the enterprise. Early results seem to indicate that this enthusiasm may be justified.

A recent study by the Massachusetts Institute of Technology found that workers using ChatGPT completed tasks ten minutes faster than their colleagues working without the service. This represented a 37% improvement in productivity. Outside evaluators determined that the product of ChatGPT-supported workers was also of a significantly higher quality. [1] As GenAI matures and enterprises become more adept at their application, these results can only be expected to improve and expand. Researchers at the University of Pennsylvania estimate that around 80% of the U.S. workforce could have at least 10% of their work tasks affected by the introduction of LLMs, while approximately 19% of workers may see at least 50% of their tasks impacted. [2]

> **"The biggest workplace challenge so far this year across industries is how to adapt to the rapidly evolving role of AI in office work."**
>
> — *The Wall Street Journal*

Technical professionals are struggling to understand how to interact with these models and adapt them for the enterprise. Their options fall into three categories:

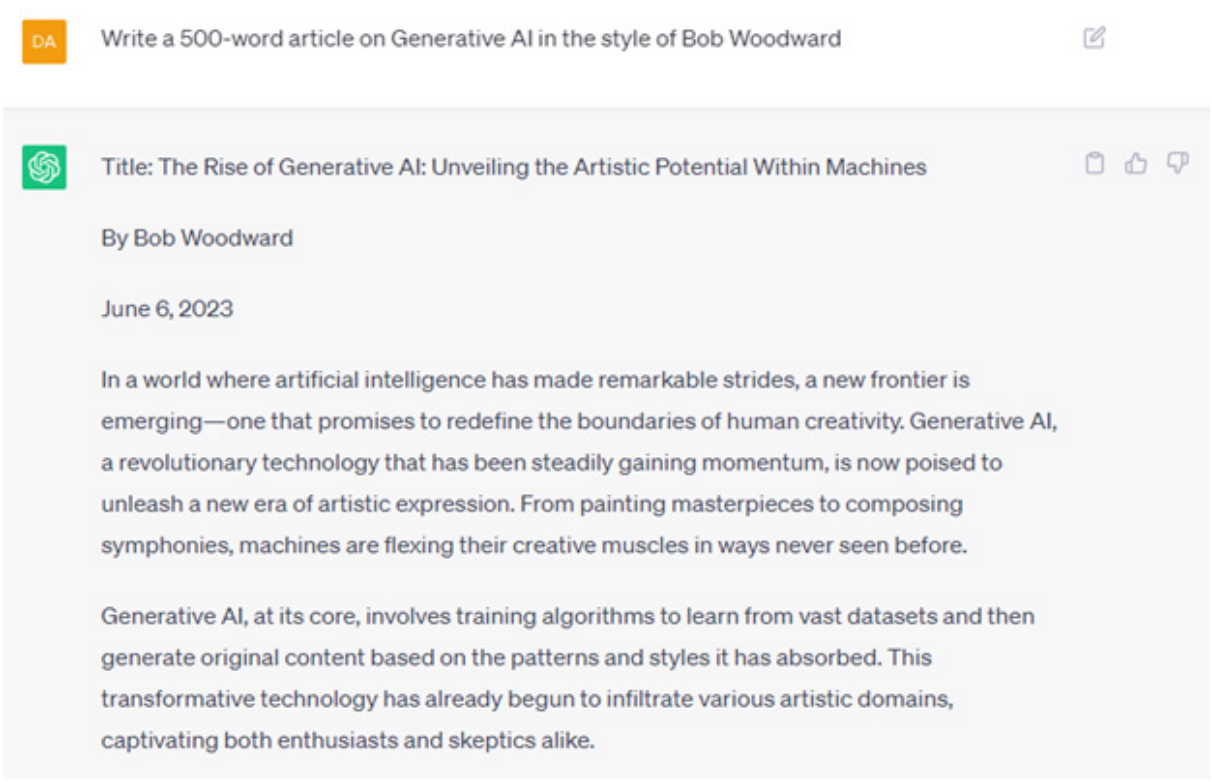
- **Prompt engineering:** A description of the desired task is provided directly to a GenAI solution, often in natural language. Information to be considered in responding to the request along with instructions or examples of how to carry out the task may also be included.
- **Model fine-tuning:** The dimensions of a pretrained LLM are adapted to perform new tasks or respond to new patterns without extensive retraining. No new knowledge is added to the model. It is just taught a new way of processing the information it contains.
- **Model training:** A model is provided with a large set of training data in order to learn the appropriate weights and biases for the dimensions in that model to accomplish a particular task or set of tasks.

Each of these processes must happen at some point in the GenAI life cycle. Every LLM must be trained and tuned if it is to be able to respond in useful ways to the requests and instructions it receives as prompts. This does not mean that all three processes must be performed by an enterprise and its developers in order to make use of LLMs and GenAI. Most use cases will be able to leverage pretrained models rather than requiring the creation or fine-tuning of a custom LLM. Prompt engineering is often enough to use a generalized LLM such as a generative pretrained transformer (GPT) in an enterprise-specific way. This is especially true when prompts incorporate enterprise information and in-context learning. The remainder of this research will explain how.

Prompt Engineering

In GenAI, a “prompt” is simply the instructions provided to an LLM or other AI when instigating an interaction or making a request. When provided directly by an end user, prompts are usually natural language requests such as “list things to do in Portland, Oregon” or “write a 500-word article on generative AI in the style of Bob Woodward.” This is known as a “zero-shot” prompt. The LLM is given a task with no further guidance as to how to perform that task (see Figure 1).

Figure 1: A Zero-Shot Prompt to ChatGPT 3.5



The LLM will respond to such requests using the information that was used to train the model. For example, the GPT 3.5 model was trained on a wide variety of websites, books, articles and other publicly available information. Any information not included in that training data or that is more recent than when the model was trained (in the case of GPT 3.5, September 2021) is not available to the LLM when responding to requests. In other words, from a content perspective, an LLM is effectively out of date as soon as it is released.

Incorporating Enterprise Information

To make GenAI truly useful to the enterprise, the LLM must have access to current, private information. This could be any enterprise content germane to the use case or application at hand. A collection of closed help desk tickets with successful resolutions could be passed to the LLM along with instructions to summarize the solution from those tickets. Earnings reports could be supplied to identify trends. Résumés of job applicants could be analyzed to identify top candidates for a particular position. Whatever relevant information is available can be supplied directly to the LLM along with instructions to restrict its analysis to the content provided. Doing so will make the intent behind the prompt clear to the LLM while also grounding any response in the context of the enterprise.

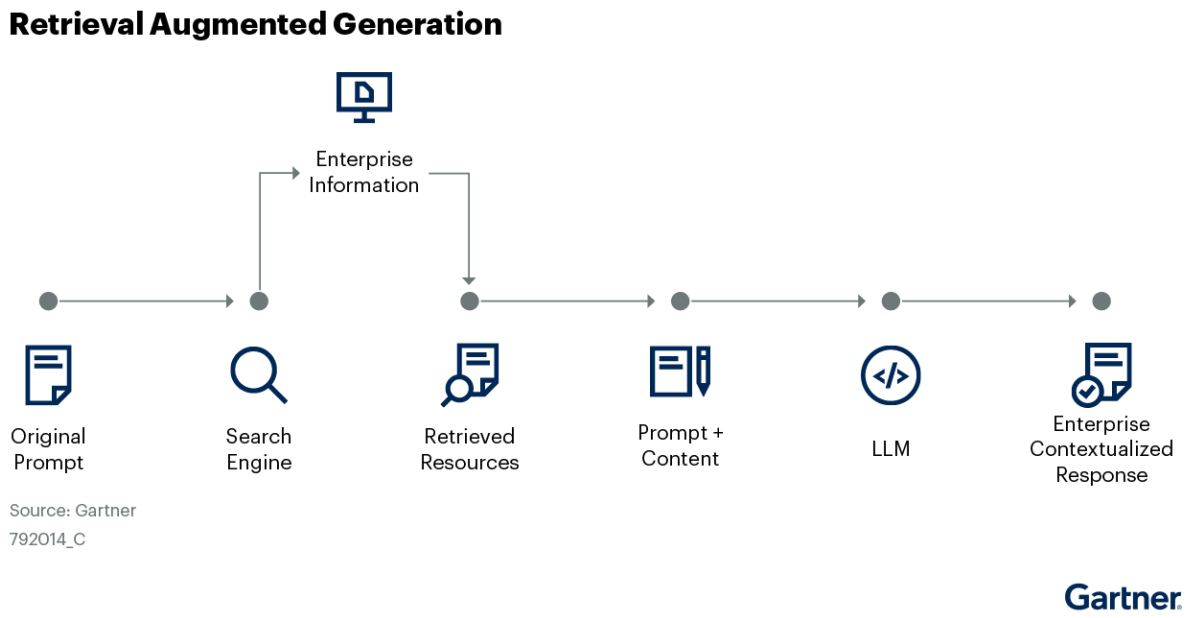
The simplest way to incorporate private information into an LLM interaction is to simply copy the content into the prompt, either as plain text or more commonly as JSON. While this approach is straightforward and available to nontechnical users, it has several drawbacks.

First and foremost is the issue of privacy. While OpenAI currently states that information contained in prompts is not used to train the model, that does not mean prompt information is not retained. OpenAI's data usage policy states: "For non-API consumer products like ChatGPT and DALL-E, we may use content such as prompts, responses, uploaded images, and generated images to improve our services." [3] Even API user data is retained "for 30 days for abuse and misuse monitoring purposes." [4] While this is not an unreasonable policy, it may also be unacceptable to some organizations with particularly sensitive or regulated information.

A more pragmatic issue is prompt limits. Any content added to the prompt counts against the maximum number of tokens allowed in a single prompt and response. A token is the basic unit of input a LLM uses for interpretation and understanding. It may be a word, part of a word, an individual character, or some other segment of the input. GPT 3.x allows up to 4,000 tokens or approximately 3,100 words. GPT 4.x increases this limit to 8,000 tokens with an option to expand to 32,000 tokens. While this represents a considerable amount of text, it includes both the prompt and any generated response. To work within these limits, the content provided to the prompt must be selected judiciously and programmatically. This is best accomplished with retrieval augmented generation.

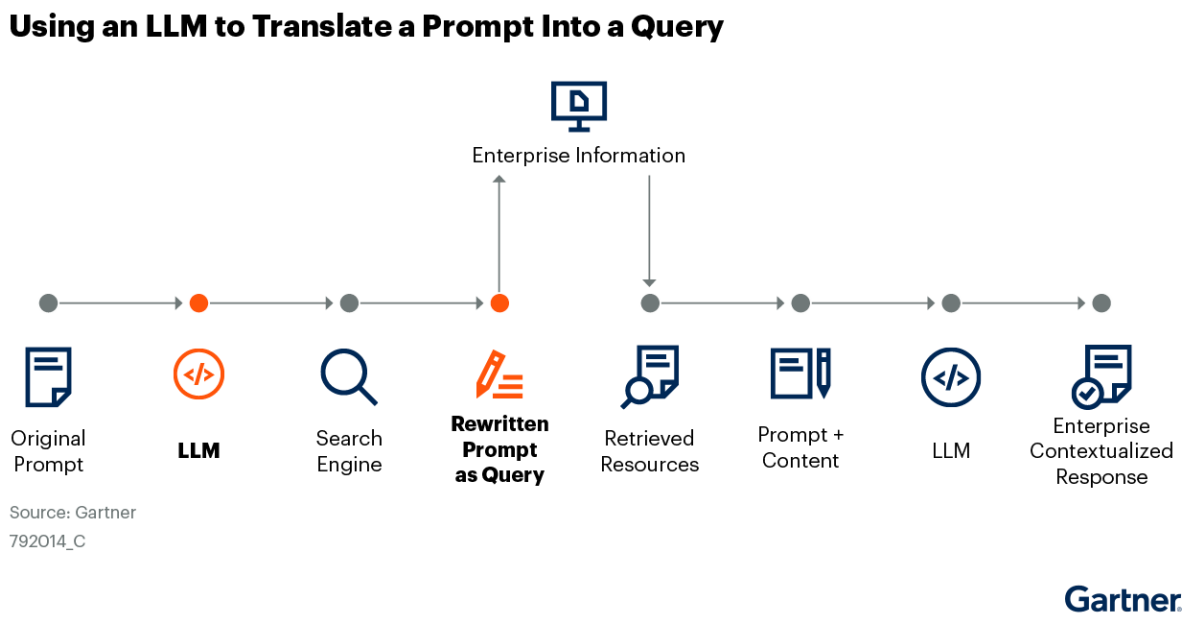
Retrieval augmented generation uses a search engine to locate and retrieve any information necessary to construct a prompt including proprietary information (see Figure 2). The initial user or application request, itself a basic prompt, is used as a query for the search engine which finds the appropriate content to add to the final prompt to be passed to the LLM. This allows contextual information to be drawn from across the enterprise and provided to the LLM for use in the current interaction. Token limits still apply, but the included content can be much more targeted, and prompts can be chained to partially circumvent these limits as discussed below. This is a much more practical approach for incorporating enterprise information into GenAI than attempting to train or retrain a model. A search-based approach has the added advantage of respecting permissions and enforcing access controls on content.

Figure 2: Retrieval Augmented Generation



A drawback of this approach is that a natural language request to an LLM or even a well-structured prompt is not well-suited to querying a search engine. Remember, an LLM is a language model, not a knowledge base or search index. If a prompt is to use a search engine to retrieve information from the enterprise, the prompt must be reformulated as a suitable query. Fortunately, LLMs are very good at this sort of refactoring and can be used to rewrite the prompt. Just as the LLM can be instructed to write an essay in a particular style, it can be made to generate a query in a certain format, for a specific platform, drawing from particular repositories (see Figure 3).

Figure 3: Using an LLM to Translate a Prompt Into a Query

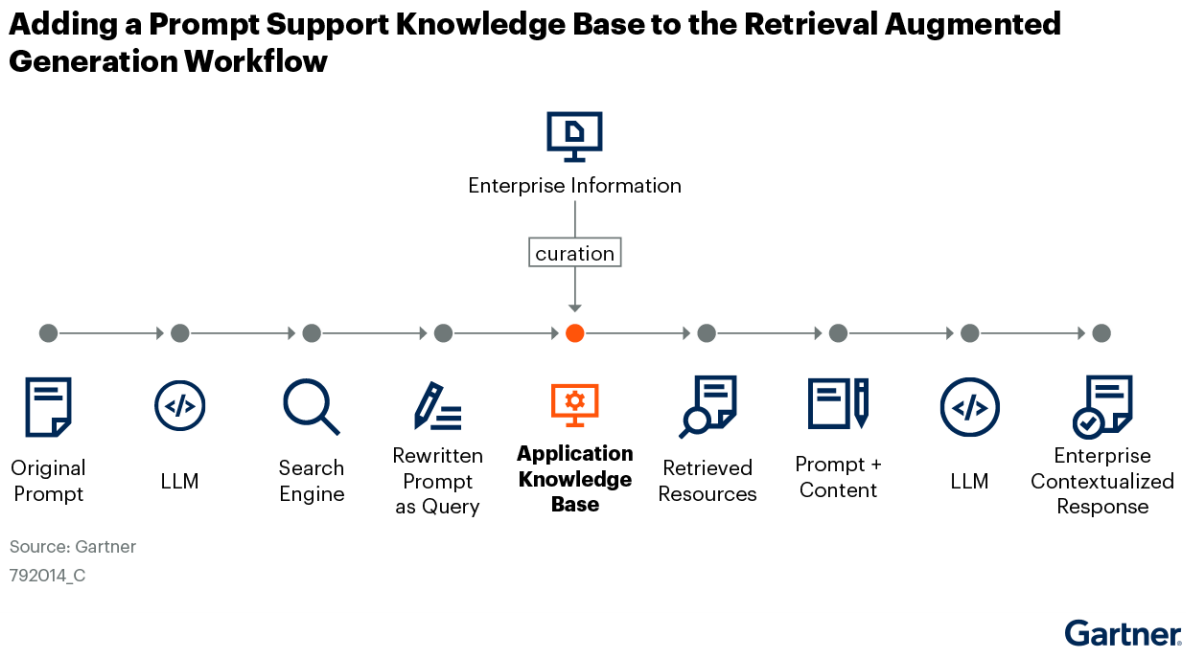


Optimizing content repositories for GenAI applications is the final step in developing a retrieval augmented generation ecosystem. Raw enterprise content is not created or managed to be fodder for LLMs. It is created to support primary business processes. Sales reports, process documents and help desk tickets are all potentially invaluable sources of contextual information for prompts, but they are not stored in forms conducive to this use. Passing an entire PDF or spreadsheet into a prompt, when only a particular section is relevant to the interaction at hand, is both inefficient and unnecessary. Only the relevant portions of the document, those that indicate relevance to the search engine, should be included in the prompt. This will help keep the prompt within token limits while enabling more precise citations in the final output.

To optimize content for prompts, enterprise information should be managed more granularly than is typically the case for regular business documents. Enterprise information resources that are to be used in prompts should be stored and managed as document fragments. For example, a single 50-page PDF document could be split into 50 single-page PDFs. A complex excel file could be divided into individual worksheets. These fragments can be indexed, and only the most relevant will independently be retrieved for a given prompt.

Document fragmentation of this sort is rarely practical at the enterprise level. Content is generally created as a byproduct of business processes and takes the form most appropriate for those processes. To support retrieval augmented generation and provide contextual information for prompts, a dedicated knowledge base of fragmented content should be created (see Figure 4). This will allow content to be selected, cleaned and structured in a manner most appropriate to the GenAI application at hand.

Figure 4: Adding a Prompt Support Knowledge Base to the Retrieval Augmented Generation Workflow



A single repository of curated content can be used to support prompt engineering across the enterprise, but multiple, targeted knowledge bases are a more effective approach. It is easier to collect, structure and maintain content for a specific application or use case than to support a one-size-fits-all repository. A targeted collection will also facilitate more effective search both by focusing the content and allowing the search engine to be tuned for that particular application. The prompt context knowledge base can be further optimized by vectorizing the content in the repository.

For a full discussion of vectorized content and semantic vector search, see How Large Language Models and Knowledge Graphs Can Transform Enterprise Search.

In-Context Learning

Even with enterprise-specific information made available through a prompt, an LLM will not know how to process that information in an enterprise-specific way. All of the reasoning abilities of an LLM are derived from its training data. In other words, it only knows how to process information and complete tasks in a generalized way. If an LLM is asked to classify a set of documents, it will organize them according to the examples it encountered during training. In the case of ChatGPT and other public LLMs, this means output will reflect the consensus of public sources and services. While the results may be technically accurate, they may not reflect the needs or context of the enterprise.

In-context learning does not change the underlying LLM in any way, and the new capability is not retained from session to session.

In order to get enterprise-oriented output from an LLM, guidance must be provided as to how the request is to be executed. Just as with the context data described above, this guidance is added directly to the prompt. This is best accomplished by providing an example of the desired response. For example, if I wanted a meeting transcript summarized and formatted in a particular manner, I would include in the prompt not only the template but an example of a completed template. This is known as “in-context learning.” It teaches the LLM how to complete a task in a particular way. This type of learning does not change the underlying LLM in any way, and the new capability is not retained from session to session.

In most cases, more than a single example is necessary to achieve the desired result. “Few-shot learning” is an AI framework that teaches a model how to deal with data and scenarios not encountered during training with only a few examples. This is more formally known as “N-way K-shot learning” where N is the number of new categories or classes the model must learn, and K is the number of labeled examples provided for each target category (N). If I want to assign a sentiment to text as either “positive,” “neutral” or “negative” I would have an N of three. If I give one example for each sentiment, I have a K of one.

N = 3, K = 1

The food was wonderful! //positive

The service was okay. //neutral

The atmosphere was terrible! //negative

N = 3, K = 2

The food was wonderful! //positive

I loved the restaurant. //positive

The service was okay. //neutral

It was good enough. //neutral

The atmosphere was terrible! //negative

Worst place ever! //negative

Few-shot learning is the most common approach to in-context learning and will meet the needs of most common enterprise applications. It eliminates the need for large amounts of labeled training data and extensive human testing and feedback. Values for K generally run between one and five examples. The best value can be determined through simple comparison testing. Begin with one or two examples per target class, and add examples until results are consistently acceptable. A library of training pairs and patterns can be created to support prompt creation for common and recurring scenarios. This will enable both programmatic prompt generation and the simplification of manual prompt creation.

If a problem proves too complex for simple few-shot learning prompts, a final approach known as “chain of thought” (CoT) prompting can be used. CoT enables a LLM to break down a complex problem into multiple steps, each providing an intermediary solution that feeds into the next step. Similar to few-shot prompting, CoT essentially shows the LLM how to solve a particular type of problem. Consider the following example from the original Google research report describing the approach. [5]

Prompt

Q: *Roger has five tennis balls. He buys two more cans of tennis balls. Each can has three tennis balls. How many tennis balls does he have now?*

A: *Roger started with five balls. Two cans of three tennis balls each is six tennis balls. 5 + 6 = 11. The answer is 11.*

Q: *The cafeteria had 23 apples. If they used 20 to make lunch and bought six more, how many apples do they have?*

Response

A: *The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought six more apples, so they have 3 + 6 = 9. The answer is nine.*

This sort of reasoning is generally beyond the capabilities of an LLM. The LLM is only able to process a standard prompt, few-shot or otherwise. A CoT prompt walks the LLM, in this case the Pathways Language Model (PaLM), through the necessary reasoning to arrive at the correct response. The LLM can then generalize the approach to similar problems. CoT prompting has also been shown to be effective in so-called commonsense and symbolic reasoning tasks such as knowing what location is appropriate for a particular activity. In principle, CoT prompting can be applied to any problem that can be solved with language.

CoT prompts are more complex than standard prompts and as a result, more difficult to generate programmatically. The thought process described in the prompt must be systematic, clearly articulated and above all, correct. Again, a library of CoT prompts and templates will facilitate their creation and application. CoT prompting also has the caveat that they only work with very large models. Researchers describe CoT as an “emergent property” of LLMs, and CoT only yields benefits in models of over 100 billion dimensions.

Regardless of the prompt style adopted, it is essential to ensure every prompt exhibits three fundamental qualities: clarity, context, constraints.

- **Clarity:** Accuracy is more important than eloquence. The objective of the prompt must be clearly and concisely stated. Avoid extraneous wording that might deviate from the core objective. Example pairs should be accurate and directly demonstrate the desired behavior.
- **Context:** Make sure the LLM has everything it needs to produce the desired results. Include any necessary content that is likely not present in the model along with instructions and examples of how that information is to be processed and interpreted.
- **Constraints:** If any boundary conditions apply to the desired operation, they should be included in the prompt. Permissions are usually applied and enforced by the search mechanism retrieving contextual content, but any additional constraints should be made explicit. This often includes a statement to the effect that only the information below should be used.

Prompts constructed according to these three principals will be sufficient to satisfy the requirements of the majority of enterprise use cases, but not all. If an application requires a specialized skill, extensive industry or domain-specific terminology, or must follow a complex pattern, prompt engineering may not be enough. It can be impractical or unreliable to try to consistently explain these processes with few-shot or CoT learning. In these instances, fine-tuning a pretrained model or even training a new model may be required. These are complex and expensive processes that should be reserved for high-value applications that cannot be addressed in any other manner.

Strengths

Prompt engineering and in-context learning have many advantages over other methods of manipulating LLMs.

- **Prompt engineering does not require fine-tuning or retraining a pretrained model.** Generalized LLMs such as GPT, the Language Model for Dialogue Applications (LaMDA), and the Large Language Model Meta AI (LLaMA), contain an extraordinary amount of information and exhibit remarkable reasoning abilities. A well-constructed prompt takes advantage of these abilities while shaping them to meet specific enterprise requirements without requiring retraining or fine-tuning the model. This dramatically reduces both the complexity and expense of incorporating LLMs and GenAI into the enterprise.
- **In-context learning allows private and proprietary enterprise information to be processed by an LLM.** In-context learning and retrieval augmented generation incorporate enterprise content directly into the LLM session. Instructions can be provided to the LLM to only consider the provided content rather than its own training data. Guidance can also be provided as to how to evaluate and analyze the provided information. As new enterprise information is created, it can be made immediately available to GenAI applications, reducing, if not eliminating, the lag time between content creation and LLM utilization.
- **Prompt style and sophistication can be matched to use cases.** Prompt styles range from simple, natural language, “zero-shot” questions to sophisticated CoT instructions. Patterns can be adopted for each type of prompt and templates developed to systematize their creation. Prompts and prompt templates can be organized into a library that users and developers can select from to match the most appropriate prompt type to the situation at hand. This also promotes quality and consistency in LLM interactions while ensuring guidelines are followed and guardrails enforced.

Weaknesses

Prompt engineering and in-context learning have limitations and drawbacks that must be taken into account.

- **In-context learning is not retained from session to session.** Because prompt engineering and in-context learning does not alter the underlying model in any way, content and context is not retained by the LLM once an interaction has concluded. Each new session is essentially starting from scratch in terms of the LLM's enterprise-specific knowledge and abilities. This can be mitigated by external retention mechanisms, but these are solely the responsibility of the developer.
- **Prompt engineering may be insufficient for specialized or complex reasoning.** While few-shot learning and CoT prompting can yield sophisticated results from a pretrained model, some processes are too complex for a prompt-based approach. If an application requires a specialized skill set or extensive specialized knowledge, the fine-tuning of the pretrained model may be required. Because fine-tuning does not add new knowledge to the LLM, only adjusting existing dimensions, retraining the model or training a custom model may be necessary. These are both complex and expensive propositions.
- **Prompt engineering requires changes to enterprise search and content management.** Enterprise content is created to support primary activities and business processes. It is generally not suited to retrieval augmented generation. To achieve optimal results, enterprise content must be refactored into smaller fragments. These document or file fragments then need to be managed in a manner conducive to search and prompt engineering. In addition, vector embeddings should be generated for this kind of content and stored in a GenAI-specific knowledge base. This is a new capability for most organizations and has a considerable learning curve.

Guidance

Technical professionals wishing to incorporate large language models and generative AI into enterprise applications should take the following actions.

Adopt prompt engineering and in-context learning as the primary means to interact with LLMs.

Prompt engineering and in-context learning are capable of meeting the requirements of most applications and should be the main method for utilizing an LLM. Fine-tuning pretrained models and training custom models should be considered a last resort for interacting with LLMs. There are cases where these options are appropriate and necessary, but they should be reserved for high-value use cases that cannot be accomplished otherwise.

Begin by experimenting with zero-shot prompting to understand the capabilities and limitations of the pretrained model being used. From there, develop a set of patterns for both zero-shot and few-shot prompts appropriate to the applications being developed or planned. Implement the patterns as a library of prompt templates available to users and developers. Create training pairs, both as templates and data, for recurring tasks. If CoT prompting is determined to be necessary, create examples articulating common processes.

Numerous prompt libraries are emerging online including repositories from Google, GoDaddy and numerous GenAI vendors and developer communities. Explore these as examples and starting points, but adopt available prompts with caution. Ensure that they conform to your requirements, respect established guardrails and abide by enterprise data handling standards.

Use retrieval augmented generation to incorporate enterprise information into LLM interactions.

Incorporating private and proprietary information into prompts is fundamental to in-context learning for enterprise applications. Content can be manually copied into prompts in limited or one-off uses, but a more systematic and programmatic approach is necessary for ongoing, scalable solutions. This requires good content management and a properly implemented search tool.

Begin by identifying enterprise content and data that will be useful to the application being developed or the overall GenAI program. Secure any necessary approvals from the content owners, and ensure that appropriate access controls are in place. Evaluate the granularity of the files and documents to be used. Use a tool, like a PDF splitter or something similar, to divide larger documents into single page (or smaller) files. Adopt a naming convention that will adequately identify the original information source in a citation generated by the LLM. Consistent metadata will enhance both retrieval and ongoing content management.

Identify the search mechanism to be used. General enterprise search will not be suitable for GenAI applications. In most organizations, search accuracy and performance is poor to begin with. Even properly implemented enterprise search applications are tuned for general information retrieval rather than application-specific uses. This does not mean an existing search platform cannot be used, but a GenAI-specific index and application will need to be created apart from the existing intranet or other search instances.

Index the content and expose the search API to the GenAI application or LLM service. Ensure licenses and usage agreements comport with enterprise data privacy, handling and ownership requirements.

Create application knowledge bases of curated enterprise content.

Enterprise content is created to support business processes instead of informing prompts. To optimize content for prompts, relevant enterprise information should be copied to a separate repository and split into more granular document or file fragments. Numerous commercial and open-source file splitting tools are available for this purpose. Select the tool that best fits your workflow. The size of these fragments should be determined through testing with the target use case or application in mind, but single-page documents are a good baseline to start with.

Consistent metadata will facilitate both the retrieval and management of the fragmented content. At a minimum, adopt a file naming convention that will adequately identify the original information source in a citation generated by the LLM. This is essential to transparency and trust as well as testing and diagnostic processes.

Content hygiene is essential to retrieval augmented generation. Ensure that the fragmented content is not only accurate and relevant to the use case, but appropriately structured and organized for search and prompt creation. A single repository of curated content can support prompt engineering across the enterprise, but multiple, targeted knowledge bases are more effective. Create a common repository as a staging area for initial collection and processing of information from enterprise systems. Use that staging area to select use-case-specific information and structure it according to the needs of the specific application at hand. Store and manage that content in a dedicated knowledge base for that application. This will both simplify the management of that content and improve search and retrieval for the LLM.

Evidence

[1] Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence (PDF), Massachusetts Institute of Technology.

[2] GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models, arXiv, Cornell University.

[3] Data Usage for Consumer Services FAQ, OpenAI.

[4] API Data Usage Policies, OpenAI.

[5] Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, arXiv, Cornell University.

Recommended by the Author

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

AI Design Patterns for Large Language Models

Best Practices for the Responsible Use of Natural Language Technologies