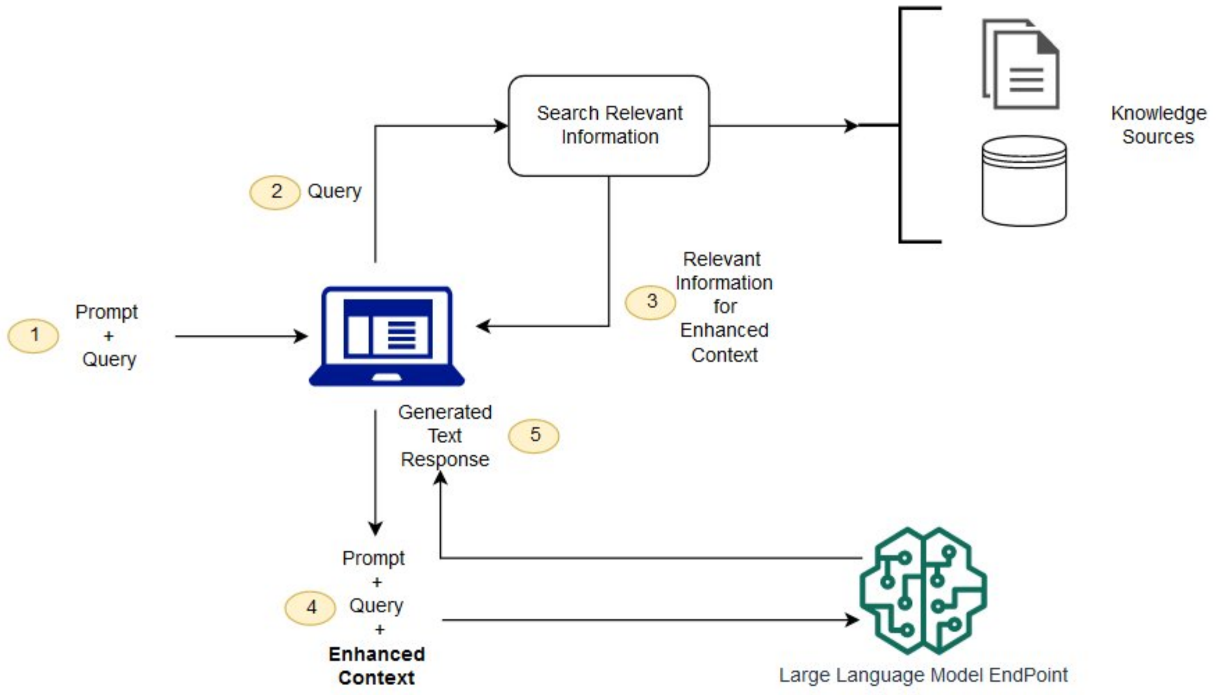
Retrieval Augmented Generation (RAG)

Foundation models are usually trained offline, making the model agnostic to any data that is created after the model was trained. Additionally, foundation models are trained on very general domain corpora, making them less effective for domain-specific tasks. You can use Retrieval Augmented Generation (RAG) to retrieve data from outside a foundation model and augment your prompts by adding the relevant retrieved data in context. For more information about RAG model architectures, see Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.

With RAG, the external data used to augment your prompts can come from multiple data sources, such as a document repositories, databases, or APIs. The first step is to convert your documents and any user queries into a compatible format to perform relevancy search. To make the formats compatible, a document collection, or knowledge library, and user-submitted queries are converted to numerical representations using embedding language models. *Embedding* is the process by which text is given numerical representation in a vector space. RAG model architectures compare the embeddings of user queries within the vector of the knowledge library. The original user prompt is then appended with relevant context from similar documents within the knowledge library. This augmented prompt is then sent to the foundation model. You can update knowledge libraries and their relevant embeddings asynchronously.


Search Relevant Information
Knowledge Sources
2 Query
1 Prompt + Query
3 Relevant Information for Enhanced Context
Generated Text Response 5
4 Prompt + Query + **Enhanced Context**
Large Language Model EndPoint

Prompt engineering for foundation models

Prompt engineering is the process of designing and refining the prompts or input stimuli for a language model to generate specific types of output. Prompt engineering involves selecting appropriate keywords, providing context, and shaping the input in a way that encourages the model to produce the desired response and is a vital technique to actively shape the behavior and output of foundation models.

Effective prompt engineering is crucial for directing model behavior and achieving desired responses. Through prompt engineering, you can control a model's tone, style, and domain expertise without more involved customization measures like fine-tuning. We recommend dedicating time to prompt engineering before you consider fine-tuning a model on additional data. The goal is to provide sufficient context and guidance to the model so that it can generalize and perform well on unseen or limited data scenarios.

Zero-shot learning

Zero-shot learning involves training a model to generalize and make predictions on unseen classes or tasks. To perform prompt engineering in zero-shot learning environments, we recommend constructing prompts that explicitly provide information about the target task and the desired output format. For example, if you want to use a foundation model for zero-shot text classification on a set of classes that the model did not see during training, a well-engineered prompt could be: `"Classify the following text as either sports, politics, or entertainment: *[input text]*."` By explicitly specifying the target classes and the expected output format, you can guide the model to make accurate predictions even on unseen classes.

Few-shot learning

Few-shot learning involves training a model with a limited amount of data for new classes or tasks. Prompt engineering in few-shot learning environments focuses on designing prompts that effectively use the limited available training data. For example, if you use a foundation model for an image classification task and only have a few examples of a new image class, you can engineer a prompt that includes the available labeled examples with a placeholder for the target class. For example, the prompt could be: "`[image 1], [image 2], and [image 3] are examples of` *`[target class]`*`. Classify the following image as` *`[target class]`*". By incorporating the limited labeled examples and explicitly specifying the target class, you can guide the model to generalize and make accurate predictions even with minimal training data.

If prompt engineering is not sufficient to adapt your foundation model to specific business needs, domain-specific language, target tasks, or other requirements, you can consider fine-tuning your model on additional data or using Retrieval Augmented Generation (RAG) to augment your model architecture with enhanced context from archived knowledge sources. For more information, see Fine-tune a foundation model or Retrieval Augmented Generation (RAG).

Fine-tune a foundation model

Foundation models are computationally expensive and trained on a large, unlabeled corpus. Fine-tuning a pre-trained foundation model is an affordable way to take advantage of their broad capabilities while customizing a model on your own small, corpus. Fine-tuning is a customization method that involved further training and does change the weights of your model.

Fine-tuning might be useful to you if you need:

- to customize your model to specific business needs
- your model to successfully work with domain-specific language, such as industry jargon, technical terms, or other specialized vocabulary
- enhanced performance for specific tasks
- accurate, relative, and context-aware responses in applications
- responses that are more factual, less toxic, and better-aligned to specific requirements

There are two main approaches that you can take for fine-tuning depending on your use case and chosen foundation model. If you're interested in fine-tuning your model on domain-specific data, see Domain adaptation fine-tuning. If you're interested in instruction-based fine-tuning using prompt and response examples, see Instruction-based fine-tuning.

Domain adaptation fine-tuning

Domain adaptation fine-tuning allows you to leverage pre-trained foundation models and adapt them to specific tasks using limited domain-specific data. If prompt engineering efforts do not provide enough customization, you can use domain adaption fine-tuning to get your model working with domain-specific language, such as industry jargon, technical terms, or other specialized data. This fine-tuning process modifies the weights of the model.

For more information, see the SageMaker JumpStart Foundation Models - Fine-tuning text generation GPT-J 6B model on domain specific dataset example notebook. You can also follow the domain adaptation dataset format steps in the Fine-tune LLaMA 2 models on SageMaker JumpStart example notebook.

Domain adaptation fine-tuning is available with the following foundation models:

- BloomZ 7b1
- GPT-J 6B
- GPT Neo 2.7B
- LLaMa-2-7b
- LLaMa-2-13b

Instruction-based fine-tuning

Instruction-based fine-tuning uses labeled examples to improve the performance of a pre-trained foundation model on a specific task. The labeled examples are formatted as prompt, response pairs and phrased as instructions. This fine-tuning process modifies the weights of the model. For more information on instruction-based fine-tuning, see the papers Introducing FLAN: More generalizable Language Models with Instruction Fine-Tuning and Scaling Instruction-Finetuned Language Models.

Fine-tuned LAnguage Net (FLAN) models use instruction tuning to make models more amenable to solving general downstream NLP tasks. Amazon SageMaker JumpStart provides a number of foundation models in the FLAN model family. For example, FLAN-T5 models are instruction fine-tuned on a wide range of tasks to increase zero-shot performance for a variety of common use cases. With additional data and fine-tuning, instruction-based models can be further adapted to more specific tasks that weren’t considered during pre-training.

Instruction-based fine-tuning is available with the following foundation models:

- FLAN-T5 XL
- FLAN-T5 Large
- FLAN-T5 Small
- FLAN-T5 Base
- LLaMa-2-7b

- LLaMa-2-13b