FUNDAMENTALS OF PROMPT ENGINEERING

Break up Complex Tasks

Some examples to give to the model – Add to the end of the prompt

- Think step by step
- Divide the task into subtasks
- Approach the problem systematically
- Reason through the problem one step at a time.

Determinism parameters

Choosing lower values for each parameter provides factual results, and choosing higher values provides diverse and creative results. The following parameters control determinism:

- **Temperature** controls randomness. Lower values focus on probable tokens, and higher values add randomness and diversity. Use lower for factual responses and higher values for creative responses.
- **Top_p** adjusts determinism with "nucleus sampling." Lower values give exact answers, while higher values give diverse responses. This value controls the diversity of the model's responses.
- **Top_k** is the number of the highest-probability vocabulary tokens to keep for top-k-filtering. Like the Top_p parameter, Top_k defines the cutoff where the model no longer selects the words.

Token count

Token count parameters include the following:

- **MinTokens is the minimum number of tokens to generate for each response.**
- **MaxTokenCount is the maximum number of tokens to generate before stopping.**

Stop Sequences

StopSequences is a list of strings that will cause the model to stop generating.

Number of results

numResults is the number of responses to generate for a given prompt.

Comparing parameters

The following table lists the different parameters associated with the three models discussed in this lesson.

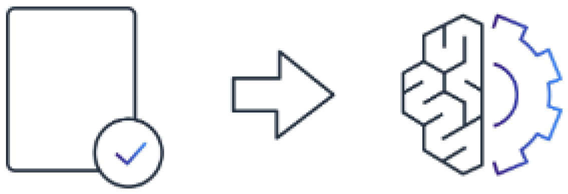
Model Provider	Model Name	Parameters
Amazon	Amazon Titan	temperature topP maxTokenCount stopSequences
Anthropic	Claude	temperature top_p top_k max_tokens_to_sample stop_sequences
AI21 Labs	Jurassic-2	temperature topP topKReturn maxTokens stopSequences numResults minTokens frequencyPenalty presencePenalty countPenalty

Instruction: Describes what you are asking of the model, such as summarization or question answering.

- **Context:** Sets the scene for the model, such as making suggestions like this one: "The following data is a medical text."
- **Input:** Provides the actual input text for summaries or retrieval tasks.
- **Output indicator:** Constrains the output from the model. For example, you may want the model to limit the output to a certain number of words or sentences while summarizing a passage.

Basic Prompt Techniques (BPT)

Zero-shot prompting

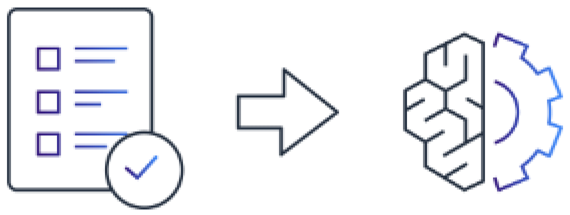


Zero-shot prompting is a prompting technique where a user presents a task to an LLM without giving the model further examples. Here, the user expects the model to perform the task without a prior understanding, or *shot*, of the task. Modern LLMs demonstrate remarkable zero-shot performance.

Tips for using a zero-shot prompting technique include the following:

- The larger the LLM, the more likely the zero-shot prompt will yield effective results.
- Instruction tuning can improve improve zero-shot learning. You can adopt reinforcement learning from human feedback (RLHF) to scale instruction tuning, aligning modern LLMs to better fit human preferences.

Few-shot prompting



Few-shot prompting is a prompting technique where you give the model contextual information about the requested tasks. In this technique, you provide examples of both the task and the output you want. Providing this context, or a *few shots*, in the prompt conditions the model to follow the task guidance closely.

Tips for using a few-shot prompting technique include the following:

- The labels in a few-shot prompt do not need to be correct to improve model performance. Usually, applying random labels outperforms using no labels at all. However, the label space and distribution of the input text specified by the demonstrations are important. The use of the term label in this context refers to the output of the prompt examples. The sentiment expressed by a statement in a "prompt example" is an example of a label.
- If you have access to a large set of examples, use techniques to obey the token limits of your model and dynamically populate prompt templates. You can use an example selector that is based on semantic similarity to help. To try out an example selector, refer to [Python Langchain Example Selectors].

Zero-shot prompt

Consider the following zero-shot prompt.

Prompt	Output
Tell me the sentiment of the following social media post and categorize it as positive, negative, or neutral: Don't miss the electric vehicle revolution! AnyCompany is ditching muscle cars for EVs, creating a huge opportunity for investors.	Positive

Note: This prompt did not provide any examples to the model. However, the model was still effective in deciphering the task.

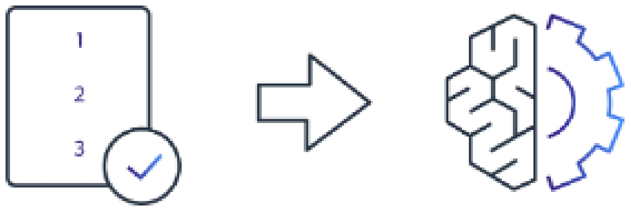
Few-shot prompts

Consider the following few-shot prompts.

Prompt	Output
Tell me the sentiment of the following headline and categorize it as either positive, negative, or neutral. Here are some examples: Research firm fends off allegations of impropriety over new technology. Answer: Negative Offshore windfarms continue to thrive as vocal minority in opposition dwindles. Answer: Positive Manufacturing plant is the latest target in investigation by state officials. Answer:	Negative

Note: The Amazon Titan Text model was used in this example. For other models, please refer to the 'Model-Specific Prompt Techniques' lesson.

Chain-of-thought prompting



Chain-of-thought (CoT) prompting breaks down complex reasoning tasks through intermediary reasoning steps. You can use both zero-shot and few-shot prompting techniques with CoT prompts.

Chain-of-thought prompts are specific to a problem type. You can use the phrase "Think step by step" to invoke CoT reasoning from your machine learning model.

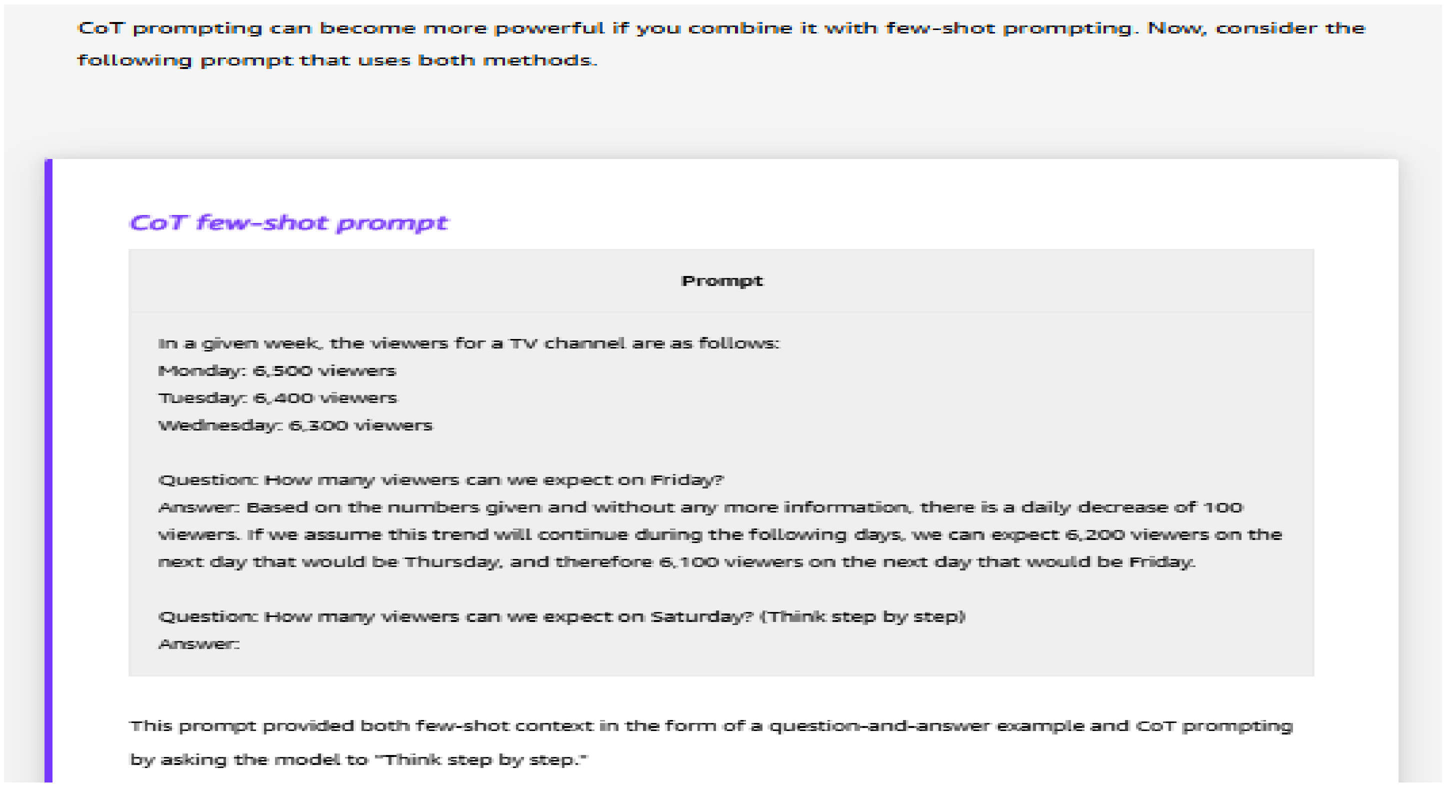
Tip: Use CoT prompting when the task involves several steps or requires a series of reasoning.

Review the following CoT prompt examples that use Amazon Titan, a large language model.

CoT zero-shot prompt

Prompt	Output
Which vehicle requires a larger down payment based on the following information? The total cost of vehicle A is $40,000, and it requires a 30 percent down payment. The total cost of vehicle B is $50,000, and it requires a 20 percent down payment. (Think step by step)	The down payment for vehicle A is 30 percent of $40,000, which is (30/100) * 40,000 = $12,000. The down payment for vehicle B is 20 percent of $50,000, which is (20/100) * 50,000 = $10,000. We can see that vehicle A needs a larger down payment than vehicle B.

Note: This prompt did not provide any examples to the model. However, the model was still effective in deciphering the task.



CoT prompting can become more powerful if you combine it with few-shot prompting. Now, consider the following prompt that uses both methods.

CoT few-shot prompt

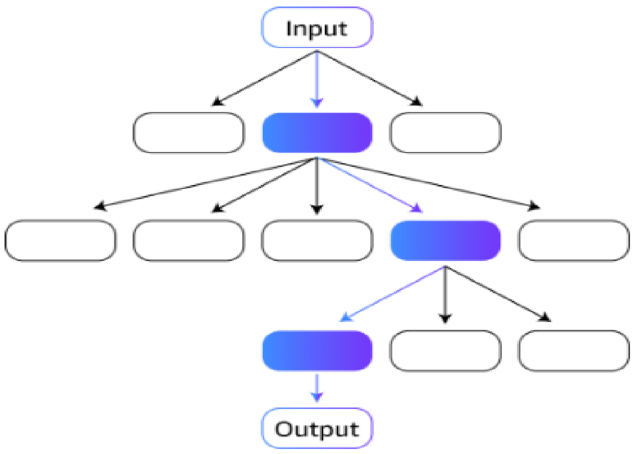
Prompt
In a given week, the viewers for a TV channel are as follows: Monday: 6,500 viewers Tuesday: 6,400 viewers Wednesday: 6,300 viewers Question: How many viewers can we expect on Friday? Answer: Based on the numbers given and without any more information, there is a daily decrease of 100 viewers. If we assume this trend will continue during the following days, we can expect 6,200 viewers on the next day that would be Thursday, and therefore 6,100 viewers on the next day that would be Friday. Question: How many viewers can we expect on Saturday? (Think step by step) Answer:

This prompt provided both few-shot context in the form of a question-and-answer example and CoT prompting by asking the model to "Think step by step."

Advanced Prompt Techniques (APT)

Self-consistency is a prompting technique that is similar to chain-of-thought prompting. However, instead of taking the obvious step-by-step, or *greedy* path, self-consistency prompts the model to sample a variety of reasoning paths. Then, the model aggregates the final answer based on multiple data points from the various paths. According to the article "Self-Consistency Improves Chain of Thought Reasoning in Language Models" by Xuezhi Wang and others, self-consistency improves CoT reasoning prompting when used in a range of common arithmetic and common-sense reasoning benchmarks.

Tree of thoughts



Tree of thoughts (ToT) is another technique that builds on the CoT prompting technique. CoT prompting samples thoughts sequentially, but ToT prompting follows a tree-branching technique. With the ToT technique, the LLM can learn in a nuanced way, considering multiple paths instead of one sequential path.

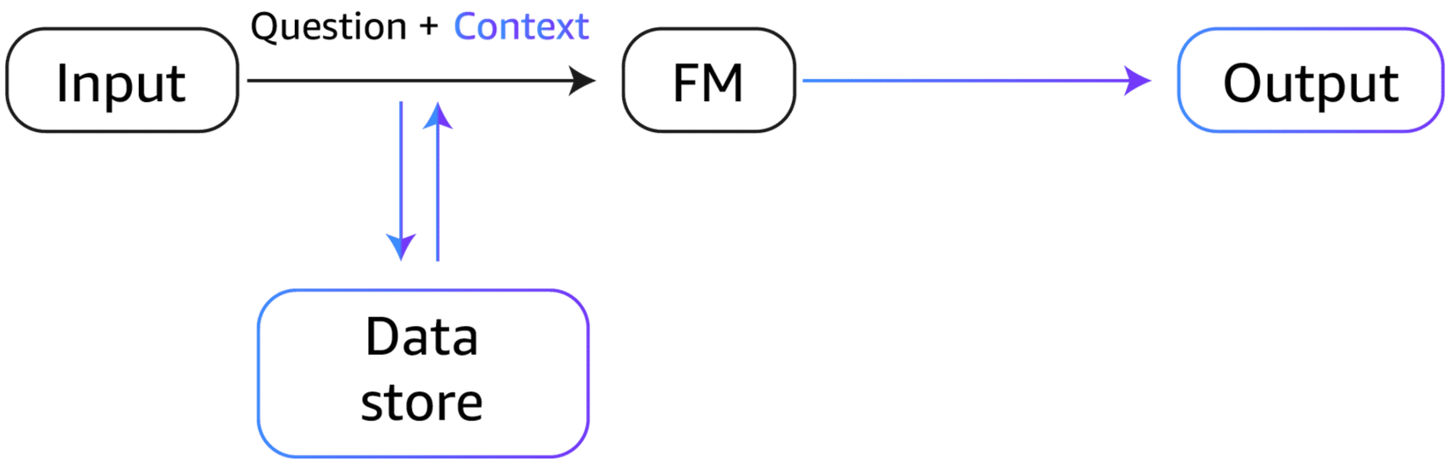
Tree of thoughts (ToT) is another technique that builds on the CoT prompting technique. CoT prompting samples thoughts sequentially, but ToT prompting follows a tree-branching technique. With the ToT technique, the LLM can learn in a nuanced way, considering multiple paths instead of one sequential path.

ToT prompting is an especially effective method for tasks that involve important initial decisions, strategies for the future, and exploration of multiple solutions. Most LLMs make decisions by following a standard left-to-right token-level inference, but with ToT, LLMs can self-evaluate choices.

According to the article, "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," by Shunyu Yao and others, ToT significantly improves performance on tasks that require nontrivial planning. Yao and other researchers tested the ToT method on three tasks: creative writing, mini crosswords, and Game of 24, a mathematical learning game. For Game of 24, Generative Pre-trained Transformer 4 (GPT-4) achieved a 4 percent success with CoT prompting. However, the model reached 74 percent success with a ToT prompting method.

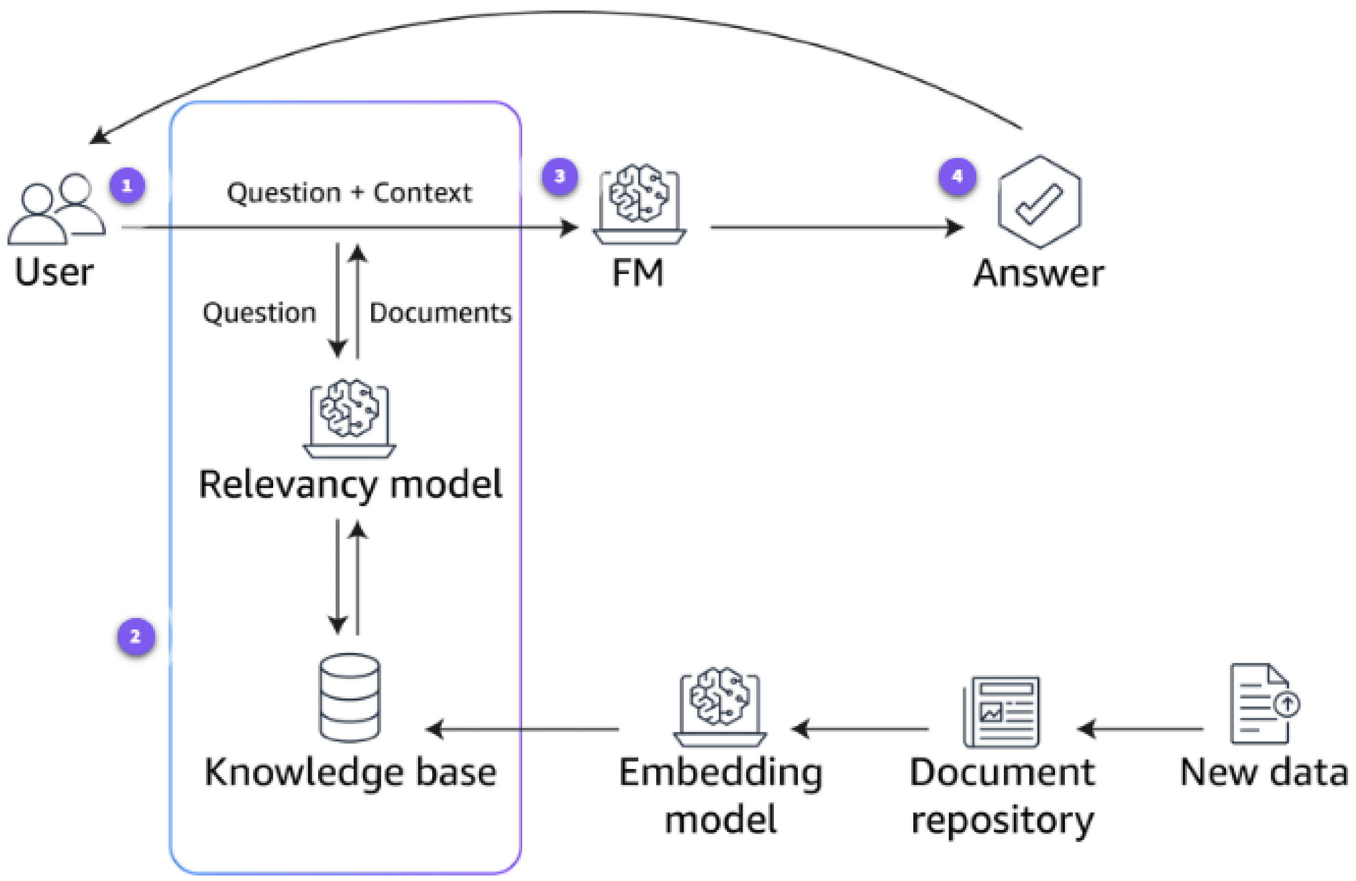
Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) is a prompting technique that supplies domain-relevant data as context to produce responses based on that data and the prompt. This technique is similar to fine-tuning. However, rather than having to fine-tune an FM with a small set of labeled examples, RAG retrieves a small set of relevant documents from a large corpus and uses that to provide context to answer the question. RAG will not change the weights of the foundation model whereas fine-tuning will change model weights.

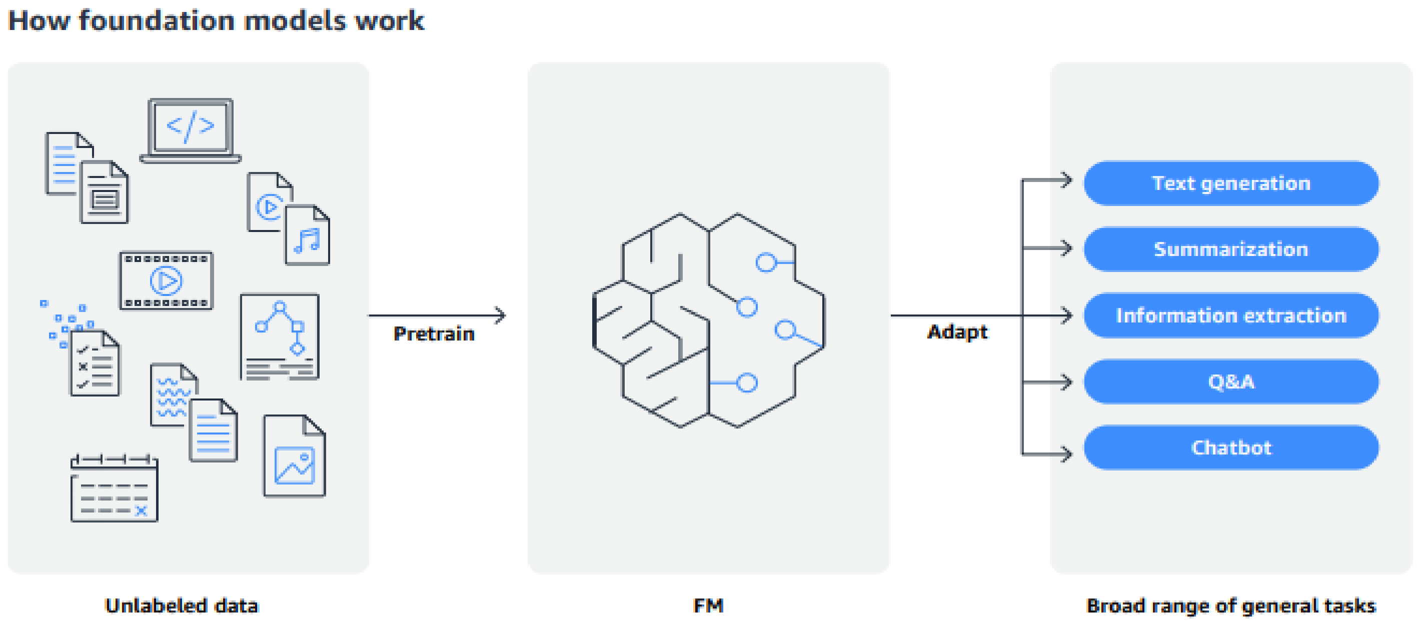


This approach can be more cost-efficient than regular fine-tuning because the RAG approach doesn't incur the cost of fine-tuning a model. RAG also addresses the challenge of frequent data changes because it retrieves updated and relevant information instead of relying on potentially outdated sets of data.

In RAG, the external data can come from multiple data sources, such as a document repository, databases, or APIs. Before using RAG with LLMs you must prepare and keep the knowledge base updated. The following diagram shows the conceptual flow of using RAG with LLMs. To see the steps the model uses to learn once the knowledge base has been prepared, choose each of the four numbered markers



Retrieval-Augmented Generation for Knowledge-


How foundation models work
Pretrain
Adapt
Text generation
Summarization
Information extraction
Q&A
Chatbot
Unlabeled data
FM
Broad range of general tasks