

# An Introduction to Large Language Models: Prompt Engineering and P-Tuning

Apr 26, 2023

By [Tanay Varshney](#) and [Annie Surla](#)

+29

Like

[Discuss \(0\)](#)



ChatGPT has made quite an impression. Users are excited to use the AI chatbot to ask questions, write poems, imbue a persona for interaction, act as a personal assistant, and more. [Large language models](#) (LLMs) power ChatGPT, and these models are the topic of this post.

Before considering LLMs more carefully, we would first like to establish what a language model does. A language model gives a probability distribution of a word being valid in a sequence of words. Essentially, the job of a language model is to predict which word is the best fit in a sentence. Figure 1 provides an example.

**Hey, I need to chop this onion, can you pass me the [?]**  
**Model's prediction[?] : Knife**

*Figure 1. Simple word prediction using a language model*

While language models like BERT have been effectively used to tackle many downstream tasks like text classification, it has been observed that with an increase in the scale of these models, certain additional abilities emerge.

This increased scale typically comes with a commensurate increase in the following three dimensions: the number of parameters, training data, and the computational resources required to train the model. For more information, see [Emergent Abilities of Large Language Models](#).

LLMs are deep learning models that can recognize, summarize, translate, predict, and generate content using large datasets. There is no one set demarcation for what is considered an LLM, but for the purposes of this discussion, we use this term to refer to any GPT-scale model or models with 1B or more parameters.

This post explains the benefits of using LLMs over a set of model pipelines built using smaller language models. It also covers the following basics:

- LLM prompts
- Prompt engineering
- P-tuning

## **Why use large language models?**

Chatbots are typically built with an ensemble of BERT models and a dialog manager. This method has some advantages such as smaller-sized models, which can result in lower latencies and compute requirements. This, in turn, is more cost-efficient. So why not use ensembles over LLMs?

- Ensembles, by their very design, are not as flexible as LLMs. This flexibility comes from the generation capabilities and the fact that said models are trained on a large corpus of data that entails a wide variety of tasks.
- In many cases, obtaining enough data to address challenges is not feasible.
- With each ensemble comes its own MLOps pipeline. Maintaining and updating a multitude of complex ensembles is difficult, as each model in every ensemble has to be fine-tuned regularly.

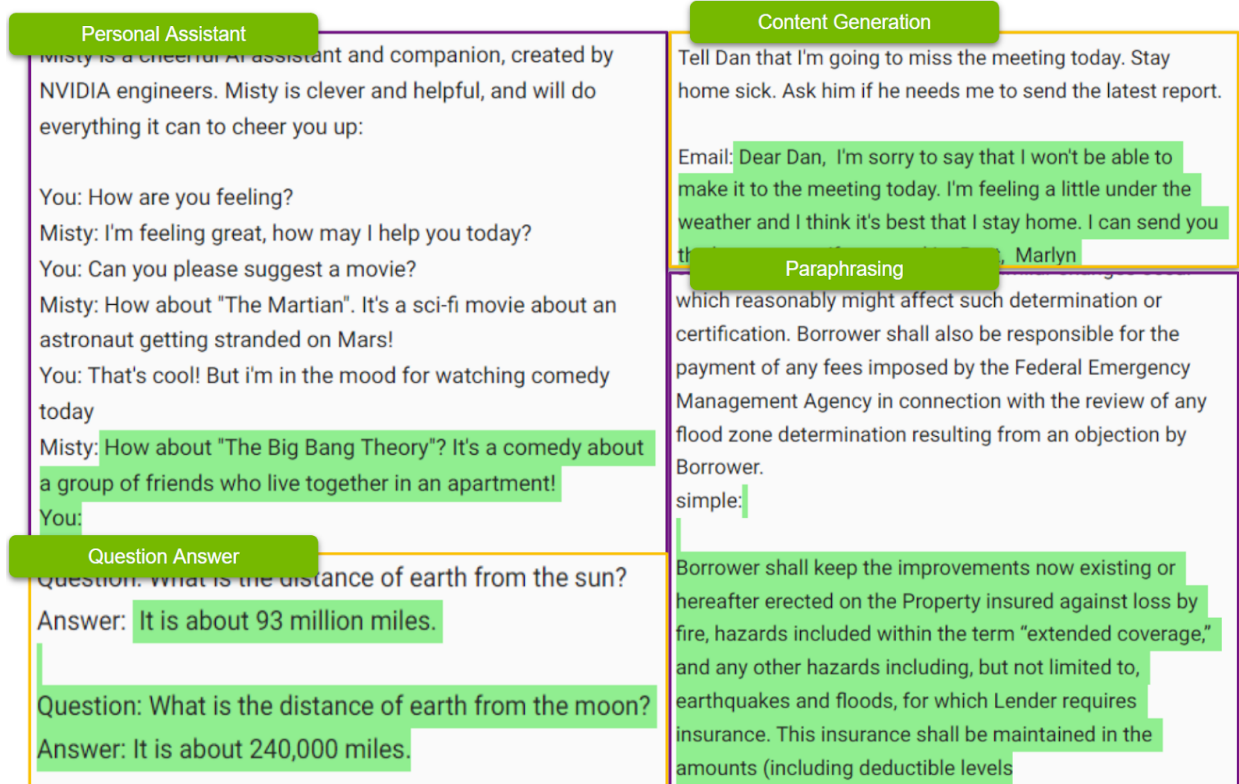


Figure 2. A subset of possible tasks that can be handled by a single LLM

## Value of LLMs over multiple ensembles

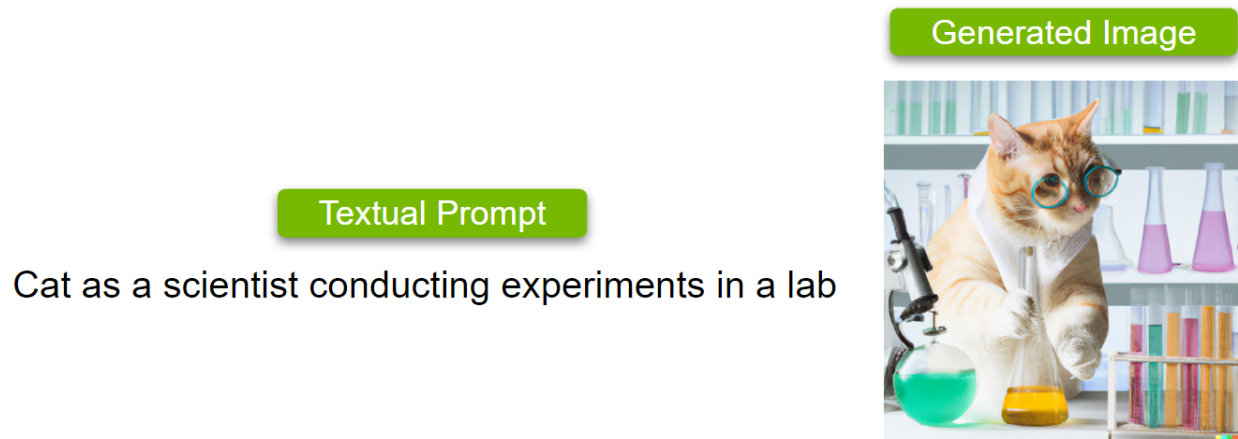
It can be argued that an ensemble of models can be less expensive than LLMs. However, considering just the inference costs, this assumption leaves out the following considerations:

- **Engineering time and cost savings:** Building, maintaining, and scaling an ensemble is a complex challenge. Each of the component models must be fine-tuned. AI infrastructure for model inference and scaling to accommodate traffic requires considerably more time to build. And this is looking at a single skill. To emulate LLMs, multiple skill sets must be built.
- **Shorter time to ship features:** The time required to build a new pipeline for a new skill is typically more than the time required to p-tune an LLM (more on this later). This means that TTM is much longer.
- **Data acquisition and quality maintenance:** Any purpose-built ensemble needs a large volume of case-specific data, which is not always available. This data must be collected on a per-model basis. In other words, in addition to the I/O from the ensemble, a data set for each of the individual models used in the ensemble is needed. Additionally, all models drift over time, and maintenance costs for fine-tuning add up quickly when working with multiple models.

These considerations show the value of using LLMs over multiple ensembles.

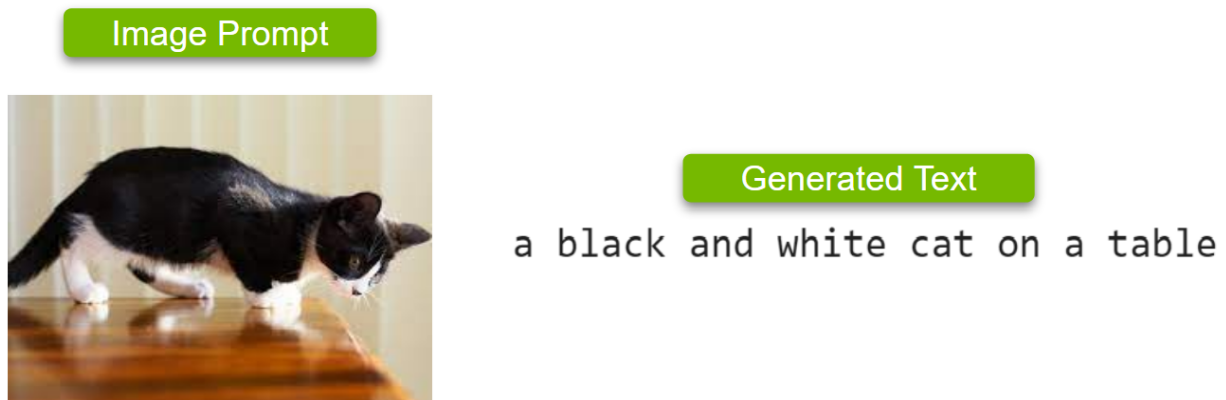
## Prompting LLMs

Prompts are used as a means to interact with LLMs to accomplish a task. A prompt is a user-provided input to which the model is meant to respond. Prompts can include instructions, questions, or any other type of input, depending on the intended use of the model. In the case of Stable Diffusion models, for example, the prompt is the description of the image to generate.



*Figure 3. A DALL-E 2 textual prompt (left) and the generated image (right)*

Prompts can also take the form of an image. With this approach, a generated textual output describes the image prompt. This is commonly used for tasks like image captioning.



*Figure 4. An image prompt (left) and the generated text (right)*

With models such as GPT-3, a text prompt can be a simple question like, “How many colors are in the rainbow?” Or, the prompt can take the form of a complicated problem, data, or an instruction such as, “Compose a motivational poem to make me happy.”

Simple Prompt	Complicated Prompt
Q: How many colors are in the rainbow? A: There are seven colors in the rainbow.	Company Name   Number of Employees   Year Established   IPO Date   Share Price     Providence Inc.   250   1990   25th August 1990   \$0.90     Grant Corporation   2000   1890   21st September 1920   \$115.90     Rusty Metalworks   12459   1946   12th September 1986   \$15.23
Instruction Prompt	
Compose a motivational poem to make me happy: Life is full of ups and downs, But don't let it get you down. You can't always get what you want, But if you try sometimes, you might find, You get what you need. If you want to make a difference, Start by doing what's right.	Q: Which company has the most employees? A: Rusty Metalworks Q: What is the share price and IPO date of Grant Corporation? A: \$115.90 and 21st September 1920

Figure 5. Examples of different types of prompts: simple, complicated, and instruction

Prompts can also include specific constraints or requirements like tone, style, or even desired length of the response. For example, a prompt to write a letter to a friend can specify tone, word limit, and specific topics to include.

The quality and relevance of the response generated by the LLM is heavily dependent on the quality of the prompt. So, prompts play a critical role in customizing LLMs to ensure that the responses of the model meet the requirements of a custom use case.

## Prompt engineering for better prompts

The term *prompt engineering* refers to the process of carefully designing the prompts to generate a specific output. Prompts play a critical role in obtaining optimal results from the model, and how you write one can make a whole lot of difference in the output generated. The following examples discuss three different strategies:

- Zero-shot prompts
- Few-shot prompts
- Chain-of-thought prompts

Zero-shot means prompting the model without any example of expected behavior from the model. For example, a zero-shot prompt asks a question.

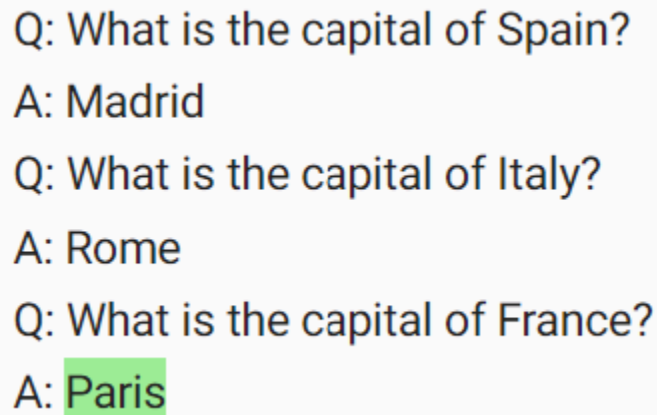
Q: What is the capital of France?  
A: France

of a zero-shot prompt

Figure 7. A simple question is an example

In Figure 7, the answer is incorrect, as Paris is the capital. Judging from the answer, the model may not understand the use of the word “capital” in this context.

A simple way to overcome this issue is to give some examples in the prompt. This type of prompt is referred to as a *few-shot prompt*. You provide a few examples before posing the actual question.



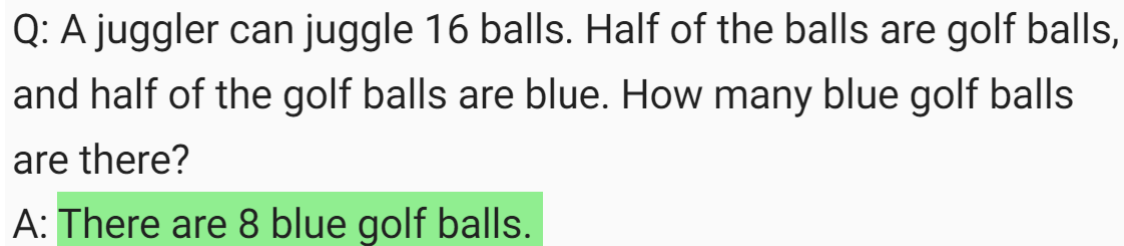
Q: What is the capital of Spain?  
A: Madrid  
Q: What is the capital of Italy?  
A: Rome  
Q: What is the capital of France?  
A: Paris

Figure 8. Example questions for a few-shot

*prompt*

A few-shot prompt enables the model to learn without training. This is one way of engineering a prompt.

How do you get the model to answer a question logically? To explore this, start with a zero-shot prompt that is more complicated.



Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A: There are 8 blue golf balls.

Figure

9. A complicated logical question zero-shot prompt

In Figure 9, the answer is once again incorrect. (The correct answer is four blue golf balls.) To help develop reasoning, use a prompting method called chain-of-thought prompting. Do this by providing some few-shot examples, where the reasoning process is explained. When the LLM answers the prompt, it shows its reasoning process as well.

Q: A juggler can juggle 4 balls. Half of the balls are golf balls, and half of the golf balls are red. How many red golf balls are there?

A: Step1: There are 4 balls. Step 2: There are 2 golf balls.  
Step3: There is 1 red golf ball. So there is 1 red golf ball.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: Step1: There are 16 balls. Step 2: There are 8 golf balls.  
Step3: There is 4 blue golf balls. So there is 4 blue golf balls.

*Figure*

*10. An example of a few-shot chain-of-thought prompt, enabling the LLM to answer accurately with reasoning*

While the example shown in Figure 10 is a few-shot chain-of-thought prompt, you can also give a zero-shot chain-of-thought prompt. This type of prompt includes phrases such as, “Let’s think about this logically.”

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there? Let's think about this logically.

1. There are 16 balls.
2. Half of the balls are golf balls.
3. Half of the golf balls are blue.

Now we can write an equation.

There are 8 golf balls.

There are 4 blue golf balls.

So, the answer is 4.

Figure

11. Zero-shot chain-of-thought prompt, enabling the LLM to answer accurately with reasoning

With this approach, the LLM generated a chain of thought that answers the question accurately. It is useful to try a range of different prompts.

## P-tuning to customize LLMs

As discussed earlier, prompt engineering is one way to customize a model's response. However, this method has shortcomings:

- A small number of examples can be used, limiting the level of control.
- The examples must be pre-appended, which affects the token budget

How can you work around these limitations?

Transfer learning is an obvious candidate: start with a base model and use the use case-specific data to fine-tune the model. This approach works well when dealing with regular models, but fine-tuning a model with 530B parameters (about 5,300x larger than a BERT model) consumes considerable time and resources.



P-tuning, or *prompt tuning*, is a parameter-efficient tuning technique that solves this challenge. P-tuning involves using a small trainable model before using the LLM. The small model is used to encode the text prompt and generate task-specific virtual tokens.

These virtual tokens are pre-appended to the prompt and passed to the LLM. When the tuning process is complete, these virtual tokens are stored in a lookup table and used during inference, replacing the smaller model.

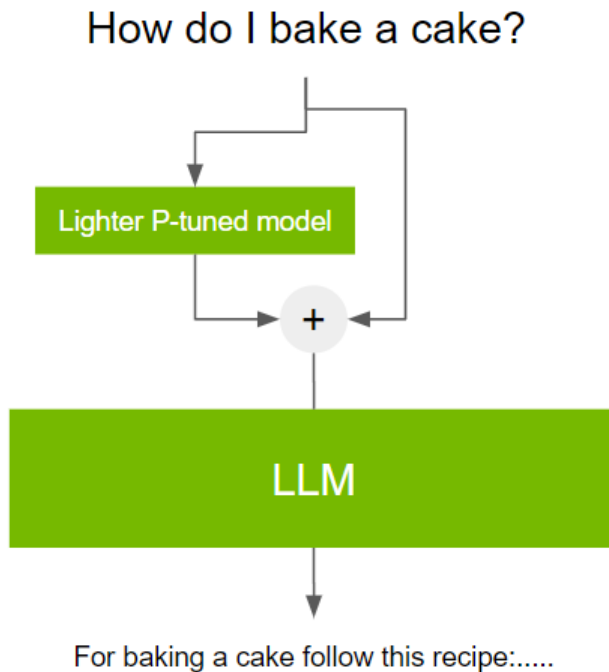


Figure 12. The general flow of prompt

tuning

This process is beneficial for the following reasons:

- Considerably fewer resources are required (compared to fine-tuning the LLM) to customize the model pipeline to achieve the desired results.
- The time required to tune a smaller model is much less (as fast as ~20 minutes).
- Models p-tuned on different tasks can be saved, without the need for large amounts of memory.

The [NVIDIA NeMo cloud service](#) simplifies this process. For more information, see [p-tuning the models in the NeMo service](#) (you must be a member of the early access program).

## Conclusion

This post discussed LLMs and outlined cases for their use. It also covered the basic concepts involved in customizing the behavior of LLMs, including various types of prompts, prompt engineering, and p-tuning.

For more information, see [more posts about LLMs](#).