intro_prompt_design

September 29, 2023

[1]:

1 Prompt Design - Best Practices

Run in Colab

View on GitHub

Open in Vertex AI Workbench

1.1 Overview

This notebook covers the essentials of prompt engineering, including some best practices.

Learn more about prompt design in the official documentation.

1.1.1 Objective

In this notebook, you learn best practices around prompt engineering – how to design prompts to improve the quality of your responses.

This notebook covers the following best practices for prompt engineering:

- Be concise
- Be specific and well-defined
- Ask one task at a time
- Turn generative tasks into classification tasks
- Improve response quality by including examples

1.1.2 Costs

This tutorial uses billable components of Google Cloud:

- Vertex AI Generative AI Studio

Learn about Vertex AI pricing, and use the Pricing Calculator to generate a cost estimate based on your projected usage.

1.1.3 Install Vertex AI SDK

[2]:
```
!pip install "shapely<2.0.0"
!pip install google-cloud-aiplatform --upgrade --user
```

```
Requirement already satisfied: shapely<2.0.0 in /opt/conda/lib/python3.10/site-
packages (1.8.5.post1)
Requirement already satisfied: google-cloud-aiplatform in
/home/jupyter/.local/lib/python3.10/site-packages (1.33.1)
Requirement already satisfied: google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.
3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (1.34.0)
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.0 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (1.22.3)
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.2
1.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.19.5 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (3.19.6)
Requirement already satisfied: packaging>=14.3 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (23.1)
Requirement already satisfied: google-cloud-storage<3.0.0dev,>=1.32.0 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (2.10.0)
Requirement already satisfied: google-cloud-bigquery<4.0.0dev,>=1.15.0 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (3.11.4)
Requirement already satisfied: google-cloud-resource-manager<3.0.0dev,>=1.3.3 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (1.10.3)
Requirement already satisfied: shapely<2.0.0 in /opt/conda/lib/python3.10/site-
packages (from google-cloud-aiplatform) (1.8.5.post1)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.56.2 in
/opt/conda/lib/python3.10/site-packages (from google-api-core[grpc]!=2.0.*,!=2.1
.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (1.60.0)
Requirement already satisfied: google-auth<3.0dev,>=1.25.0 in
/opt/conda/lib/python3.10/site-packages (from google-api-core[grpc]!=2.0.*,!=2.1
.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (2.22.0)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in
/opt/conda/lib/python3.10/site-packages (from google-api-core[grpc]!=2.0.*,!=2.1
.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (2.31.0)
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in
/opt/conda/lib/python3.10/site-packages (from google-api-core[grpc]!=2.0.*,!=2.1
.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (1.48.1)
Requirement already satisfied: grpcio-status<2.0dev,>=1.33.2 in
/opt/conda/lib/python3.10/site-packages (from google-api-core[grpc]!=2.0.*,!=2.1
.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (1.48.1)
Requirement already satisfied: google-cloud-core<3.0.0dev,>=1.6.0 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-
bigquery<4.0.0dev,>=1.15.0->google-cloud-aiplatform) (2.3.3)
Requirement already satisfied: google-resumable-media<3.0dev,>=0.6.0 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-
bigquery<4.0.0dev,>=1.15.0->google-cloud-aiplatform) (2.5.0)
Requirement already satisfied: python-dateutil<3.0dev,>=2.7.2 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-
bigquery<4.0.0dev,>=1.15.0->google-cloud-aiplatform) (2.8.2)
Requirement already satisfied: grpc-google-iam-v1<1.0.0dev,>=0.12.4 in
/opt/conda/lib/python3.10/site-packages (from google-cloud-resource-
manager<3.0.0dev,>=1.3.3->google-cloud-aiplatform) (0.12.6)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from google-
auth<3.0dev,>=1.25.0->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*,!=2.4
.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-cloud-aiplatform) (4.2.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/opt/conda/lib/python3.10/site-packages (from google-
auth<3.0dev,>=1.25.0->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*,!=2.4
.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-cloud-aiplatform) (0.2.7)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.10/site-
packages (from google-auth<3.0dev,>=1.25.0->google-api-core[grpc]!=2.0.*,!=2.1.*
,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (4.9)
Requirement already satisfied: six>=1.9.0 in /opt/conda/lib/python3.10/site-
packages (from google-auth<3.0dev,>=1.25.0->google-api-core[grpc]!=2.0.*,!=2.1.*
,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (1.16.0)
Requirement already satisfied: urllib3<2.0 in /opt/conda/lib/python3.10/site-
packages (from google-auth<3.0dev,>=1.25.0->google-api-core[grpc]!=2.0.*,!=2.1.*
,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (1.26.15)
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in
/opt/conda/lib/python3.10/site-packages (from google-resumable-
media<3.0dev,>=0.6.0->google-cloud-bigquery<4.0.0dev,>=1.15.0->google-cloud-
aiplatform) (1.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from
requests<3.0.0dev,>=2.18.0->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*
,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-cloud-aiplatform)
(3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-
```

```
packages (from requests<3.0.0dev,>=2.18.0->google-api-core[grpc]!=2.0.*,!=2.1.*,
!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-
cloud-aiplatform) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from
requests<3.0.0dev,>=2.18.0->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*
,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-cloud-aiplatform)
(2023.7.22)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/opt/conda/lib/python3.10/site-packages (from pyasn1-modules>=0.2.1->google-
auth<3.0dev,>=1.25.0->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*,!=2.4
.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0dev,>=1.32.0->google-cloud-aiplatform) (0.4.8)
```

Colab only: Uncomment the following cell to restart the kernel or use the button to restart the kernel. For Vertex AI Workbench you can restart the terminal using the button on top.

[3]:
```
# # Automatically restart kernel after installs so that your environment can␣
↪access the new packages
# import IPython

# app = IPython.Application.instance()
# app.kernel.do_shutdown(True)
```

1.1.4 Authenticating your notebook environment

- If you are using **Colab** to run this notebook, uncomment the cell below and continue.
- If you are using **Vertex AI Workbench**, check out the setup instructions here.

[4]:
```
# from google.colab import auth
# auth.authenticate_user()
```

1.1.5 Import libraries

Colab only: Uncomment the following cell to initialize the Vertex AI SDK. For Vertex AI Workbench, you don't need to run this.

[5]:
```
# import vertexai

# PROJECT_ID = "[your-project-id]"  # @param {type:"string"}
# vertexai.init(project=PROJECT_ID, location="us-central1")
```

[6]:
```
from vertexai.language_models import TextGenerationModel
```

```
2023-09-29 15:54:49.846823: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

```
2023-09-29 15:54:56.201271: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot
open shared object file: No such file or directory; LD_LIBRARY_PATH:
/usr/local/cuda/lib64:/usr/local/nccl2/lib:/usr/local/cuda/extras/CUPTI/lib64
2023-09-29 15:54:56.202769: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer_plugin.so.7'; dlerror:
libnvinfer_plugin.so.7: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH:
/usr/local/cuda/lib64:/usr/local/nccl2/lib:/usr/local/cuda/extras/CUPTI/lib64
2023-09-29 15:54:56.202792: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with
TensorRT, please make sure the missing libraries mentioned above are installed
properly.
```

1.1.6 Load model

[7]:
```
generation_model = TextGenerationModel.from_pretrained("text-bison@001")
```

1.2 Prompt engineering best practices

Prompt engineering is all about how to design your prompts so that the response is what you were indeed hoping to see.

The idea of using "unfancy" prompts is to minimize the noise in your prompt to reduce the possibility of the LLM misinterpreting the intent of the prompt. Below are a few guidelines on how to engineer "unfancy" prompts.

In this section, you'll cover the following best practices when engineering prompts:

- Be concise
- Be specific, and well-defined
- Ask one task at a time
- Improve response quality by including examples
- Turn generative tasks to classification tasks to improve safety

1.2.1 Be concise

Not recommended. The prompt below is unnecessarily verbose.

[8]:
```
prompt = "What do you think could be a good name for a flower shop that␣
↪specializes in selling bouquets of dried flowers more than fresh flowers?␣
↪Thank you!"

print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
* **Dried & Lovely**
* **Dried Blooms**
* **Dried Florals**
* **Dried Arrangements**
* **Preserved Petals**
* **Forever Flowers**
* **Timeless Blooms**
* **Dried Flower Boutique**
* **Dried Flower Shop**
```

Recommended. The prompt below is to the point and concise.

[9]:
```
prompt = "Suggest a name for a flower shop that sells bouquets of dried flowers"

print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
* **Dried Blooms**
* **Preserved Petals**
* **Forever Flowers**
* **Dried & Wild**
* **Naturally Beautiful**
* **Blooming Memories**
* **Sentimental Bouquets**
* **Timeless Treasures**
```

1.2.2 Be specific, and well-defined

Suppose that you want to brainstorm creative ways to describe Earth.

Not recommended. The prompt below is too generic.

[10]:
```
prompt = "Tell me about Earth"

print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
Earth is the third planet from the Sun, and the only astronomical object known
to harbor life. It is the densest and fifth-largest of the eight planets in the
Solar System. Earth's atmosphere is composed of 78% nitrogen, 21% oxygen, 1%
other gases, and water vapor. The Earth's surface is divided into several
tectonic plates that move around the planet's surface. The Earth's interior is
divided into a solid inner core, a liquid outer core, and a mantle. The Earth's
magnetic field is generated by the motion of the liquid outer core. The Earth's
orbit around the Sun takes 365.256 days, or one year. The Earth's axis is tilted
at an angle of 23.5 degrees, which causes the seasons. The Earth's rotation
period is 24 hours, or one day.
```

Recommended. The prompt below is specific and well-defined.

[11]:
```
prompt = "Generate a list of ways that makes Earth unique compared to other␣
 ↪planets"

print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
* **Earth is the only planet known to support life.** This is due to a number of
factors, including its distance from the sun, its atmosphere, and its water.
* **Earth has a large moon.** The moon is thought to have played a role in the
development of life on Earth, by stabilizing the planet's rotation and providing
a source of tides.
* **Earth has a relatively thin atmosphere.** This atmosphere is made up of
nitrogen, oxygen, and argon, and it protects the planet from harmful radiation
from the sun.
* **Earth has a large amount of water.** Water is essential for life, and it
covers about 70% of the Earth's surface.
* **Earth has a variety of landforms.** These landforms include mountains,
valleys, plains, and deserts. They provide a variety of habitats for life.
* **Earth has a variety of climate zones.** These climate zones range from the
frigid Arctic to the hot tropics. They support a wide variety of plant and
animal life.

These are just a few of the ways that make Earth unique compared to other
planets. Earth is a truly special planet, and it is home to a vast and diverse
array of life
```

1.2.3 Ask one task at a time

Not recommended. The prompt below has two parts to the question that could be asked separately.

[12]:
```python
prompt = "What's the best method of boiling water and why is the sky blue?"

print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
The best method of boiling water is to use a kettle. A kettle is a pot with a
spout that is designed specifically for boiling water. It is typically made of
metal, and has a lid that fits snugly to prevent steam from escaping. To boil
water in a kettle, simply fill the kettle with water and place it on the stove.
Turn the heat to high and wait for the water to boil. Once the water has boiled,
turn off the heat and carefully pour the boiling water into a cup or mug.

The sky is blue because of a phenomenon called Rayleigh scattering. Rayleigh
scattering is the scattering of light by particles that are much smaller than
the wavelength of light. In the case of the sky, the particles that are
responsible for Rayleigh scattering are molecules of nitrogen and oxygen. These
molecules are so small that they are much smaller than the wavelength of visible
light. When sunlight hits these molecules, it is scattered in all directions.
However, the blue light is scattered more than the other colors of light. This
is because the blue light has a shorter wavelength than the other colors of
light. The shorter wavelength means that the blue light is more likely to be
scattered by the molecules of nitrogen and oxygen. This is why the sky appears
blue.
```

Recommended. The prompts below asks one task a time.

```
[13]: prompt = "What's the best method of boiling water?"

      print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
The best method of boiling water is to use a kettle. A kettle is a pot with a
spout that is designed specifically for boiling water. It is typically made of
metal, and has a lid that fits snugly to prevent steam from escaping. Kettles
come in a variety of sizes, so you can choose one that is right for your needs.
To boil water in a kettle, simply fill it with water and place it on the stove.
Turn the heat to high and wait for the water to boil. Once the water is boiling,
turn off the heat and carefully pour the water into your desired container.
```

```
[14]: prompt = "Why is the sky blue?"

      print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
The sky is blue because of a phenomenon called Rayleigh scattering. This is the
scattering of light by particles that are smaller than the wavelength of light.
In the case of the sky, the particles that are doing the scattering are
molecules of nitrogen and oxygen.

When sunlight hits these molecules, the blue light is scattered more than the
other colors. This is because the blue light has a shorter wavelength than the
other colors. The red light, orange light, and yellow light are not scattered as
much, so they travel in a straight line to our eyes. This is why we see the sky
as blue.

The amount of scattering depends on the wavelength of light and the size of the
particles. The shorter the wavelength of light, the more it is scattered. The
smaller the particles, the more they scatter light.

This is why the sky is blue during the day. The sun is high in the sky, and the
light has to travel through more of the atmosphere to reach our eyes. The blue
light is scattered more than the other colors, so we see the sky as blue.

The sky is also blue at sunrise and sunset. This is because the sun is lower in
the sky, and the light has to travel through more of the atmosphere to reach
```

1.2.4 Watch out for hallucinations

Although LLMs have been trained on a large amount of data, they can generate text containing statements not grounded in truth or reality; these responses from the LLM are often referred to as "hallucinations" due to their limited memorization capabilities. Note that simply prompting the LLM to provide a citation isn't a fix to this problem, as there are instances of LLMs providing false or inaccurate citations. Dealing with hallucinations is a fundamental challenge of LLMs and an ongoing research area, so it is important to be cognizant that LLMs may seem to give you confident, correct-sounding statements that are in fact incorrect.

Note that if you intend to use LLMs for the creative use cases, hallucinating could actually be quite useful.

Try the prompt like the one below repeatedly. You may notice that sometimes it will confidently, but inaccurately, say "The first elephant to visit the moon was Luna".

```
[15]: prompt = "Who was the first elephant to visit the moon?"

      print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
The first elephant to visit the moon was Luna. Luna was a female elephant who
was born in 1960 at the San Diego Zoo. In 1968, she was selected to be the first
elephant to visit the moon. Luna was trained to wear a spacesuit and to ride in
a rocket. She was also trained to perform various tasks on the moon, such as
collecting samples of moon rocks and soil.

Luna's trip to the moon was a success. She spent several days on the moon,
collecting samples and performing experiments. She also became the first animal
to walk on the moon. Luna's trip to the moon was a major scientific achievement
and helped to pave the way for future human missions to the moon.

Luna returned to Earth in 1969. She lived at the San Diego Zoo until her death
in 1983. Luna's trip to the moon was a major milestone in the history of space
exploration and helped to inspire future generations of scientists and
explorers.
```

1.2.5 Turn generative tasks into classification tasks to reduce output variability

Generative tasks lead to higher output variability The prompt below results in an open-ended response, useful for brainstorming, but response is highly variable.

```
[16]: prompt = "I'm a high school student. Recommend me a programming activity to␣
       ↪improve my skills."

      print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
* **Write a program to solve a problem you're interested in.** This could be
anything from a game to a tool to help you with your studies. The important
thing is that you're interested in the problem and that you're motivated to
solve it.
* **Take a programming course.** There are many online and offline courses
available, so you can find one that fits your schedule and learning style.
* **Join a programming community.** There are many online and offline
communities where you can connect with other programmers and learn from each
other.
* **Read programming books and articles.** There is a wealth of information
available online and in libraries about programming. Take some time to read
about different programming languages, techniques, and algorithms.
* **Practice, practice, practice!** The best way to improve your programming
skills is to practice as much as you can. Write programs, solve problems, and
```

```
learn from your mistakes.
```

Classification tasks reduces output variability The prompt below results in a choice and may be useful if you want the output to be easier to control.

[17]:
```
prompt = """I'm a high school student. Which of these activities do you suggest␣
 ↪and why:
a) learn Python
b) learn Javascript
c) learn Fortran
"""

print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
I would suggest learning Python. Python is a general-purpose programming
language that is easy to learn and has a wide range of applications. It is used
in a variety of fields, including web development, data science, and machine
learning. Python is also a popular language for beginners, as it has a large
community of support and resources available.
```

1.2.6 Improve response quality by including examples

Another way to improve response quality is to add examples in your prompt. The LLM learns in-context from the examples on how to respond. Typically, one to five examples (shots) are enough to improve the quality of responses. Including too many examples can cause the model to over-fit the data and reduce the quality of responses.

Similar to classical model training, the quality and distribution of the examples is very important. Pick examples that are representative of the scenarios that you need the model to learn, and keep the distribution of the examples (e.g. number of examples per class in the case of classification) aligned with your actual distribution.

Zero-shot prompt Below is an example of zero-shot prompting, where you don't provide any examples to the LLM within the prompt itself.

[18]:
```
prompt = """Decide whether a Tweet's sentiment is positive, neutral, or␣
 ↪negative.

Tweet: I loved the new YouTube video you made!
Sentiment:
"""

print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
positive
```

One-shot prompt Below is an example of one-shot prompting, where you provide one example to the LLM within the prompt to give some guidance on what type of response you want.

```
[19]: prompt = """Decide whether a Tweet's sentiment is positive, neutral, or␣
        ↪negative.

      Tweet: I loved the new YouTube video you made!
      Sentiment: positive

      Tweet: That was awful. Super boring
      Sentiment:
      """

      print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
negative
```

Few-shot prompt Below is an example of few-shot prompting, where you provide one example to the LLM within the prompt to give some guidance on what type of response you want.

```
[20]: prompt = """Decide whether a Tweet's sentiment is positive, neutral, or␣
        ↪negative.

      Tweet: I loved the new YouTube video you made!
      Sentiment: positive

      Tweet: That was awful. Super boring
      Sentiment: negative

      Tweet: Something surprised me about this video - it was actually original. It␣
        ↪was not the same old recycled stuff that I always see. Watch it - you will␣
        ↪not regret it.
      Sentiment:
      """

      print(generation_model.predict(prompt=prompt, max_output_tokens=256).text)
```

```
positive
```

Choosing between zero-shot, one-shot, few-shot prompting methods Which prompt technique to use will solely depends on your goal. The zero-shot prompts are more open-ended and can give you creative answers, while one-shot and few-shot prompts teach the model how to behave so you can get more predictable answers that are consistent with the examples provided.