**FAQs on Gen AI**

*(Bulk of the Generative AI Learning path is handwritten.)*

**What is Generative AI?**

It is a type of artificial intelligence that generates content for you.

**What kind of content?**

The generated content can include text, images, code, audio(speech), video, and 3D.

**What are some of the tasks Gen AI can help you automate or achieve?**

* Document summarization
* Information extraction
* Code generation
* Marketing campaign creation
* Virtual assistance
* Call center bot
* etc.

**How does Gen AI generate new content?**

It learns from a massive amount of existing content. This includes text, audio and video. The process of learning from existing content is called training, which results in the creation of a “foundation model.” An LLM, or large language model, which powers chat bots like Bard AI and ChatGPT, is a typical example of a foundation model. The foundation model can then be used to generate content and solve general problems, such as content extraction and document summarization. It can also be trained further with new datasets in your field to solve specific problems, such as financial model generation and healthcare consulting. This results in the creation of a new model that is tailored to your specific needs.

**How can you use the foundation model to power your applications, and how can you further train, or tune, the foundation model to solve a problem in your specific field?**

Google Cloud provides several easy-to-use tools that help you use generative AI in your projects with or without an AI and machine learning background. One such tool is Vertex AI. Vertex AI is an end-to-end ML development platform on Google Cloud that helps you build, deploy, and manage machine learning models. With Vertex AI, if you are an app developer or data scientist and want to build an application, you can use Generative AI Studio to quickly prototype and customize generative AI models with no code or low code. If you are a data scientist or ML developer who wants to build and automate a generative AI model, you can start with Model Garden. Model Garden lets you discover and interact with Google’s foundation and third-party open-source models and has built-in MLops tools to automate the ML pipeline.

A screenshot of a computer

Description automatically generated with medium confidence

**What is generative AI studio all about?**

Generative AI Studio supports language, vision, and speech. For language, you can design a prompt to perform tasks and tune language models. For vision, you can generate an image based on a prompt and further edit the image. For speech, you can generate text from speech or vice versa.

A screenshot of a computer

Description automatically generated with medium confidence

**What can you do with language in Gen AI Studio?**

1. Design a prompt.
2. Start a conversation.
3. Tune a model.

A screenshot of a video

Description automatically generated with medium confidence

**DESIGN A PROMPT**

\*The first feature you can do in Gen AI Studio using language is design a prompt. In the world of Generative AI, a prompt is just a fancy name for the input text you feed into your model. So, in short, the answers or responses you get from the model depend on the questions or prompts you ask the model.

A screenshot of a video

Description automatically generated with medium confidence

**What is prompt design?**

*Prompt design* is the process of figuring out and designing the best input text to get the desired response back from the model.

**What are the three methods a user can use to shape a model's response in a way that you desire?**

1. Zero-shot prompting – is a method where the LLM is given no additional data on the specific task that it is being asked to perform. Instead, it is only given a prompt that describes the task. For example, if you want the LLM to answer a question, you just enter the prompt "what is prompt design?". And the output will be produced on the definition of prompt design based on the number of tokens a user has set.
2. One-shot prompting – is a method where the LLM is given a single example of the task that it is being asked to perform. For example, if you want the LLM to write a poem, you might provide a single example poem to help paint the theme you are looking for.
3. Few-shot prompting – is a method where the LLM is given a small number of examples of the task that it is being asked to perform. For example, if you want the LLM to write a news article, you might give it a few news articles to read. You can use the structured mode in GCP to design the few-shot prompting by providing a context and additional examples for the model to learn from.

A screenshot of a video

Description automatically generated

**How do you prototype (sample) a Q&A system in Generative AI Studio?**

If using the Generative AI studio, under language a user needs to switch over to structured prompt, also known as Few-shot prompting. The structured prompt contains a few different components:

* First, we have the context, which instructs how the model should respond. You can specify words the model can or cannot use, topics to focus on or avoid, or a particular response format. And the context applies each time you send a request to the model. Let’s say we want to use an LLM to answer questions based on some background text. In this case, a passage that describes changes in rainforest vegetation in the Amazon. We can paste in the background text as the context. Then, we add some examples of questions that could be answered from this passage like:
* What does LGM stand for?
* What did the analysis from the sediment deposits indicate?

We’ll need to add in the corresponding answers to these questions, to demonstrate how we want the model to respond.

A screenshot of a computer

Description automatically generated

Then, we can test out the prompt we’ve designed by sending a new question as input. And there you go; you’ve prototyped a Q&A system based on background text.

A screenshot of a computer

Description automatically generated

The area used for testing allows a user to ask the LLM questions about the context given above and receive answers that were not thought of.

**What are some of the best practices for prompt design?**

1. Be concise.
2. Be specific and well-defined.
3. Ask one task at a time.
4. Ask to classify, instead of generate. (Turn generative tasks into classification tasks. For example, instead of asking what programming language to learn, ask if Python, Java, or C is a better fit for a beginner in programming.)
5. Include examples. (Improve response quality by including examples. Adding instructions and a few examples tends to yield good results.)

\*\*\*There’s currentlyno one best way to write a prompt. You may need to experiment with different structures, formats, and examples to see what works best for your use case.

When you send a prompt to an LLM, it produces an array of probabilities over the words that could come next. And from this array, we need some strategy to decide what to return. A simple strategy might be to select the most likely word at every timestep. But this method can result in uninteresting and sometimes repetitive answers. On the contrary, if you randomly sample over the distribution returned by the model, you might get some unlikely responses. By controlling the degree of randomness, you can get more unexpected, and some might say creative, responses.

\*The panel to the right of where a user enters their prompts is known as the model parameters. The first parameter is known as temperature. Temperature is a number used to tune the degree of randomness. Low temperature means to select the words that are highly possible and more predictable.

A screenshot of a computer

Description automatically generated

In this case, those are flowers and the other words that are located at the beginning of the list. This setting is generally better for tasks like Q&A and summarization where you expect a more “predictable” answer with less variation. High temperature means to select the words that have low possibility and are more unusual. In this case, those are bugs and the other words that that are located at the end of the list. This setting is good if you want to generate more “creative” or unexpected content.

\*Think of temperature as a measure of entropy in your model.

Entropy - is defined as a measure of randomness or disorder of a system.

**What does the model parameter Top K do? (number)**

Top K lets the model randomly return a word from the top K number of words in terms of possibility. In other words, if a user enters a prompt and only a total of 10 words could possibly fill the gap, with a Top K of 3, the user would only be selecting from the top 3 best possible words to fill the gap. Top 2 means you get a random word from the top 2 possible words including flowers and trees. This approach allows the other high-scoring word a chance of being selected because the LLM does not have to choose from a list of 10 words anymore, once you have shrunk the list down to 3 or a Top K of 3 in this case.

A screenshot of a computer

Description automatically generated

**What does the model parameter Top P do? (probability)**

Top P allows the model to randomly return a word from the top P probability of words. With top P, you choose from a set of words with the sum of the likelihoods not exceeding P. For example, p of 0.75 means you sample from a set of words that have a cumulative probability greater than 0.75. In this case, it includes three words: flowers, trees, and herbs. Remember, you are not required to adjust Top K and Top P constantly.

A screenshot of a computer

Description automatically generated

**START A CONVERSATION**

The second feature you can do in Gen AI Studio using language is start a conversation. First, you need to specify the conversation context. Context instructs how the model should respond. For example, specifying words the model can or cannot use, topics to focus on or avoid, or response format. Context applies each time you send a request to the model.

For a simple example, you can define a scenario and tell the AI how to respond to help desk queries. Here is a sample prompt you can enter in the context area below:

“*Your name is Roy. You are a support technician in an IT department. You only respond with "Have you tried turning it off and on again?" to any queries*.”

You can tune the parameters on the right, the same as you do when designing the prompt. To see how it works, you can type ‘My computer is slow’ in the chat box below and press enter. The AI responds: Have you tried turning it off and on again? Exactly as you told the AI to do. Starting a conversation is very effective when using an LLM, if you are looking for the LLM to produce an automated response to anything that may be asked in a call center environment. You can also structure a more complex prompt for other queries a user may have. All in all, the goal here is to tell the LLM what to say, when asked a specific question.

A screenshot of a computer

Description automatically generated

\*The cool thing is that Google provides the APIs and SDKs to help you build your own application. You can simply click view code. First, you need to download the Vertex AI SDKs that fit your programming language, like Python and Curl. They implement the functions and do the job for you. You can use them like you call libraries from the code. You then follow the sample code and the API and insert the code into your application.

**SDKs**

**What is an SDK?**

\*SDK stands for software design kits. Also known as a devkit, the SDK is a set of software-building tools for a specific platform, including the building blocks, debuggers and, often, a framework or group of code libraries such as a set of routines specific to an operating system (OS).

**APIs**

**What is an API?**

API stands for application programming interface. Whether working as a standalone solution or included within an SDK, an API facilitates communication between two platforms. It does this by allowing its proprietary software to be leveraged by third-party developers. The developers can then enable their own users to indirectly use the service or services provided by the API solution.

You can also think of an API as a kind of agreement between two parties. The API not only allows for on-demand information exchange, but it also stipulates how that information should be exchanged. Because some APIs provide the interface directly, the terms “API” and “interface” are sometimes used interchangeably.

**What are a few well-known popular APIs used?**

* Web APIs, which are used to reach web browsers and devices or as their own web services application.
* SOAP APIs, which are a popular choice in cases of heightened data privacy and security.
* Open APIs (or Public APIs) and REST (or RESTful) APIs, which are a popular choice for ease of use and maximizing bandwidth.
* JSON-RPC, a go-to for cases where asynchronous server calls are needed.
* Custom APIs, for maximum agility with all the moving parts of software development.

**TUNE A MODEL**

If you’ve been prototyping with large language models, you might be wondering if there’s a way you can improve the quality of responses beyond just prompt design. So, let’s learn how to tune a large language model and how to launch a tuning job from Generative AI Studio.

As a quick recap, the prompt is your text input that you pass to the model. Your prompt might look like an instruction… And maybe you add some examples… Then you send this text to the model so that it adopts the behavior that you want.

A screenshot of a computer

Description automatically generated

\*Prompt design allows for fast experimentation and customization. And because you’re not writing any complicated code, you don’t need to be an ML expert to get started. But producing prompts can be tricky. Small changes in wording or word order can affect the model results in ways that aren’t totally predictable. And you can’t really fit all that many examples into a prompt. Even when you do discover a good prompt for your use case, you might notice the quality of model responses isn’t totally consistent. One thing we can do to alleviate these issues is to tune the model.

**What is Tuning?**

Well, one version you might be familiar with is fine-tuning. As you recall, when fine-tuning:

* A user takes a model that was pretrained on a generic dataset.
* We make a copy of this model.
* Then, using those learned weights as a starting point, we re-train the model on a new domain-specific dataset.

\*This technique has been effective for lots of different use cases.

But when we try to fine tune LLMs, we run into some challenges. LLMs are, as the name suggests, large. So, updating every weight can take a long training job. Compound all that computation with the hassle and cost of now having to serve this giant model… And as a result, fine-tuning a large language model might not be the best option for you.

But there’s an innovative approach to tuning called parameter-efficient tuning.

This is a super exciting research area that aims to reduce the challenges of fine-tuning LLMs, by only training a subset of parameters. These parameters might be a subset of the existing model parameters. Or they could be an entirely new set of parameters. For example, maybe you add some additional layers to the model or an extra embedding to the prompt.

When referring to a subset of parameters, think of it as a column of parameters in a neural network, as shown below in the image:

A screenshot of a computer

Description automatically generated with medium confidence

Ideally, the average user will not be tuning an LLM. Parameter efficient tuning is ideally suited for scenarios where you have "modest" amounts of training data, say hundreds or maybe thousands of training examples. Your training data should be structured as a supervised training dataset in a text-to-text format. Each record or row in the data will contain the input text, in other words, the prompt, which is followed by the expected output of the model. This means that the model can be tuned for a task that can be modeled as a text-to-text problem.

Below is an example of text-to-text syntax, as you can see the dictionary has the input and output values labeled accordingly:

A screenshot of a computer

Description automatically generated