Chapter 11: Generative AI in GCP

In the preceding two chapters we have demonstrated AWS and Azure capabilities within the realm of generative AI; we now turn our attention to Google offering in that space. What differentiates the company from its main competitors in the space – Amazon and Microsoft – is that it is currently the only one offering a comprehensive suite of commercial solutions covering every aspect of the AI-focused suite of tools. From dedicated AI chips, through computing power, to development tools and applications – Google has a complete AI stack that serves as a foundation for its offering. Their proprietary Tensor Processing Units (TPU) are designed specifically for machine learning tasks, optimized for use with TensorFlow, and are boasting faster and more efficient performance for AI workloads; while GPU (and CPU) units have not been dethroned across the industry, over the last few years TPU have emerged as formidable rivals.

The Cambrian explosion of generative AI started in the second half of 2022 and the initial impression was that Google had to play catch-up to Microsoft / OpenAI (especially in the NLP space, due to chatGPT parabolic rise). Upon closer examination, it seems like Google strategy has been more of a comprehensive enterprise-grade approach. We begin by discussing the educational offer on GCP: if you, dear reader, have reached this point in the book, chances are you are already familiar with the basics; nevertheless, the series of mini courses offered on the platform can be useful complement to your knowledge.

Having reviewed the educational offer, we will examine the capabilities of Generative AI Studio. The application is built on the Vertex AI platform and allows developers to create their own generative AI apps in text and vision domains. More advanced users can make us of the Model Garden, which gives access to a collection of pretrained models, for advanced exploration and interaction.

If you prefer to hear the summary of Google Generative AI offerings from the company themselves, feel free to check out the introductory videos: <https://www.youtube.com/watch?v=YCZ6nwGnL4o>. Once you have done that, you can come back here and we will embark on the next stage of our journey.

# Learning path

The educational provisions on GCP are designed for a wide range of individuals: from novice enthusiasts eager to expand their understanding to seasoned professionals seeking to refine their expertise. These crash courses allow for an a la carte approach, allowing learners to delve into specific topics – following them sequentially can be helpful in certain cases, but is not required. The first step is to go to the training resources page: <https://cloud.google.com/blog/topics/training-certifications/new-google-cloud-generative-ai-training-resources>

You will be greeted by the screen shown in Figure 11.1 below. Scroll down to “Generative AI Learning Path” and click on the link. This will redirect you to <https://www.cloudskillsboost.google/journeys/118>, where you begin your education.

A picture containing text, screenshot, diagram, design

Description automatically generated

Figure 11.1: Landing page for the Generative AI track from GCP

# A screenshot of a google cloud Description automatically generated with low confidence

Figure 11.2: Generative AI Learning Path page

Each micro-course is composed of three parts: video content, reading list and a quiz to test your newly acquired knowledge. We briefly summarize the modules below:

1. **Introduction to Generative AI**: this course defines Generative AI and explains how it is different from other domains of ML. In addition, it covers major types of models used in the field and describes popular AI applications. Even if you are an experienced ML practitioner, spending half an hour on refreshing the basic definitions is a good idea.  
     
   An important disclaimer needs to be made at this point: *while the structure of the courses is likely to remain stable, the content itself might have changed a bit by the time you are reading these words (e.g. which models are used are integrated, or specific details of the interface); the reason for that is the incredibly fast pace of change in the field: nary a fortnight seems to go by without somebody releasing an ever-more-powerful LLM, beating a previous SOTA. “Change is the only constant” is a rather terrible cliché, but in the context of generative AI it is quite appropriate*.
2. **Introduction to Large Language Models**: this course delves into the concept of LLM, their various applications and utilization of prompt tuning to improve LLM performance. In addition, Google tools useful for personalized Gen AI applications are reviewed. This one is notable for its scope: starting from LLM types (generic, dialog-tuned, instruction-tuned), it discusses also advanced concepts like Chain of Thought.
3. **Introduction to Responsible AI**: This introductory microlearning course is aimed at explaining what responsible AI is, why it's important, and how Google implements responsible AI in their products. It also introduces Google's 7 AI principles.
4. **Introduction to Image Generation:** if population averages are anything to go by, chances are you, dear reader, got into generative AI because you saw an image created with Dall-E or somesuch – which means you have a pretty good idea of what they can do. In this course you will get an introduction into how they do it: diffusion models draw inspiration from thermodynamics and over the last few years they have gained enormous popularity both in research and in the industry. You will get a chance to dive into the theory underpinning diffusion models, as well as learn how to effectively train and deploy them on Vertex AI.
5. **Encoder-Decoder Architecture:** This course provides an overview of the encoder-decoder architecture, a widely used and effective machine learning framework for tasks involving sequences, such as machine translation, text summarization, and question answering. You will gain knowledge about the key elements of the encoder-decoder architecture, including training and deploying these models. During the accompanying lab tutorial, you will code a basic implementation of the encoder-decoder architecture in TensorFlow for generating poetry from scratch.
6. **Attention Mechanism**: in this module you will became acquainted (or re-acquainted – the original paper came out in 2017, after all) with the attention mechanism: a powerful technique enabling DL algorithms to concentrate on segments of an input sequence. You will gain an understanding of how attention operations and how it is applied in enhancing the efficiency of various ML tasks.
7. **Transformer Models and BERT Model:** the course introduces you to \*the\* application of attention mechanism – Transformer architecture, along with its most famous example: Bidirectional Encoder Representations from Transformers (BERT) model. You will learn about the main components (self-attention), and how they comprise the BERT together.
8. **Create Image Captioning Models:** In this course, you will learn to build an image captioning model using deep learning techniques. You will gain knowledge about the various elements comprising an image captioning model, including the encoder and decoder, as well as the process of training and evaluating your model. By the conclusion of the course, you will have the ability to develop your own image captioning models and utilize them for generating descriptive captions for images.
9. **Introduction to Generative AI Studio:** Generative AI Studio is a very interesting offering from GCP, so this final course serves as an excellent segue into our next section.

# Generative AI Studio

Generative AI Studio is a GC console tool for rapidly prototyping and testing generative AI models. You can test sample prompts, design your own prompts, and customize foundation models to handle tasks that meet your application’s needs.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11.3: Generative AI Studio landing page

The Generative AI Studio tutorial (the last course discussed in the previous section)

Asdadsadasdasd

Asdasd

A screenshot of a web page

Description automatically generated with medium confidence

Figure 11.4: XX

Asdasd

A screenshot of a computer

Description automatically generated with low confidence

Figure 11.5: XX

Asd

A screenshot of a computer

Description automatically generated with low confidence

Figure 11.6: XX

The link works, in case you were wondering

What if we chaNGE temperature? – cf menu on the right hand side in 11.5

Asdasd

A screenshot of a computer

Description automatically generated with low confidence

Figure 11.7: XX

As you can see, answer is more creative elaborate blah blah blah

Zero shot example:

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11.8: XX

Can we improve it with a bit of context?

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11.9: XX

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11.10: XX

Got more creative with the later questions, but it still copied the first three

There’s a reason this shit is an art and not a science

We can also use structured prompting for more proper few-shot prompting – see the tutorial from the course list above for an excellent demonstration of this functionality. We now move to the second type of task Generative AI Studio can help us with: conversation.

TODO: or a simple example, you can define a scenario and tell the AI how to respond to help desk

Unless you have been hiding under a rock, you have probably seen countless memes where people demonstrate the role-playing capability of LLM, with chatGPT in the leading role. Google is not likely to promote the product of a competitor, but it has the same functionality

Implemented through the conversation fubnctionality in gen ai studio

Fig 11.11 shows you how to get started thus

A screenshot of a web page

Description automatically generated with low confidence

Figure 11.11: XX

Asdf

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11.12: XX

asdasd

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11.13: XX

Asd

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11.14: XX

Asd

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11.15: XX

Model tuning comes in when few shot tuning does not cut it anymore

Prompt design strategies, such as few shot prompting, may not always suffice for customizing model behavior. Use model tuning to improve a model's performance on specific tasks or help the model adhere to specific output requirements when instructions aren't sufficient. This page shows you how to tune a foundation model in Vertex AI and gives guidance on how to achieve the best results.

Now let’s look at the third feature, tune a language 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.

A screenshot of a computer

Description automatically generated with low confidence

Figure 11.16: XX

TODO:

Language model tuning – we will pay a bit of attention to the bits skimmed over in the tutorial

Get the dataset:

SELECT

CONCAT(q.title, q.body) as input\_text,

a.body AS output\_text

FROM

`bigquery-public-data.stackoverflow.posts\_questions` q

JOIN

`bigquery-public-data.stackoverflow.posts\_answers` a

ON

q.accepted\_answer\_id = a.id

WHERE

q.accepted\_answer\_id IS NOT NULL AND

REGEXP\_CONTAINS(q.tags, "python") AND

a.creation\_date >= "2020-01-01"

LIMIT

10000

asdasd

if you export it from BigQuery like this, default format is correct (input, output, newline) but the extension is .json and not .jsonl => GCP won’t accept it => you need to change it manually

your file should look similar to this

{"input\_text": "question: How many people live in Beijing? context: With over 21 million residents, Beijing is the world's most populous national capital city and is China's second largest city after

Shanghai. It is located in Northern China, and is governed as a municipality under the direct administration of the State Council with 16 urban, suburban, and rural districts.[14] Beijing is mostly

surrounded by Hebei Province with the exception of neighboring Tianjin to the southeast; together, the three divisions form the Jingjinji megalopolis and the national capital region of China.",

"output\_text": "over 21 million people"}

{"input\_text": "question: How many parishes are there in Louisiana? context: The U.S. state of Louisiana is divided into 64 parishes (French: paroisses) in the same manner that 48 other states of the United

States are divided into counties, and Alaska is divided into boroughs.", "output\_text": "64"}

(different if you used the stackoverflow example above)

A screenshot of a computer

Description automatically generated

Figure 11.17: Model tuning

Click “continue” => will take a moment to upload things to the bucket and after that the process follows the tutorial => no point in duplicating it here

# Model Garden

Model Garden is a general Vertex AI functionality, but it is quite handy to mention it around Gen ai

Model Garden offers a unified platform for searching, discovering, and engaging with Google's foundational models. Over time, it will also encompass numerous open-source and third-party models. Users will have more than just text models at their disposal, as they can leverage Google's multimodal models spanning various domains like vision, dialogue, code generation, and code completion. Our aim is to provide a range of options throughout the AI stack, which is why Model Garden will incorporate models from both open-source collaborators and our ecosystem of AI partners. By bringing together a diverse array of model types and sizes in one location, our customers will have the freedom to choose the most suitable resource for their business requirements.

A screenshot of a computer

Description automatically generated

Figure 11.18: Model Garden

# Generative AI App Builder

TODO :

Currently – June 2023 – not available for general access

Gen App Builder is a powerful platform that offers the fastest way to develop chatbots and search engines for both websites and enterprise data. It provides enterprises with the ability to quickly access relevant information and combines organizational data with Google's search technologies. With Gen App Builder, users can build generative AI experiences using text, voice, images, and video.

The key features of Gen App Builder include the combination of generative AI with information retrieval, allowing stakeholders to quickly obtain personalized information. Enterprises can easily ingest data from internal and external sources, enabling applications to understand user intent through natural language processing. The platform also allows for the creation of multimodal apps that incorporate text, images, and videos, enhancing customer interactions.

Gen App Builder goes beyond providing information by enabling complete transactions and seamless integration with third-party apps and services. It allows the creation of digital assistants and bots that can connect to purchasing and provisioning systems, facilitating transactions directly from the conversational user interface. Enterprises maintain control over their applications and data, with the ability to decide which information can be leveraged and ensuring compliance with regulations and data sovereignty laws.

Overall, Gen App Builder offers a comprehensive solution for developing chatbots and search engines, combining the power of generative AI and information retrieval. It provides flexibility for developers with out-of-the-box capabilities and APIs for integration into enterprise workflows. Additionally, it enables the creation of multimodal apps and ensures data control and compliance for enterprises using the platform.

# models on GCP

Welcome to the Introduction to the Generative AI Studio course. In this video, you learn what Generative AI Studio is and describe its options for use. You also explore the Generative AI Studio’s language tool.

[00:14](javascript:;)What is Generative AI? It is a type of artificial intelligence that generates content for you. What kind of content? Well, the generated content can be multi-modal, including text, images, audio, and video.

[00:28](javascript:;)When given a prompt or a request, Generative AI can help you achieve various tasks, such as document summarization, information extraction, code generation, marketing campaign creation, virtual assistance, and call center bot.

[00:42](javascript:;)And these are just a few examples! How does 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

[00:57](javascript:;)of a “foundation model.” An LLM, or large language model, which powers chat bots like Bard, is a typical example of a foundation model. The foundation model can then be used to generate content and solve general problems, such as

[01:11](javascript:;)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.

[01:27](javascript:;)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?

[01:36](javascript:;)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.

[01:47](javascript:;)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,

[02:01](javascript:;)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

[02:14](javascript:;)AI model, you can start from 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.

[02:28](javascript:;)In this course, you focus on Generative AI Studio. Generative AI Studio supports language, vision, and speech. The list grows as you are learning this course. For language, you can design a prompt to perform tasks and tune language models.

[02:43](javascript:;)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. Let’s focus on what you can do with Language in Generative AI Studio.

[02:57](javascript:;)specifically, you can: Design prompts for tasks relevant to your business use case including code generation. Create conversations by specifying the context that instructs how the model should respond. And tune a model so it is better equipped for your use case, which allows you to then

[03:14](javascript:;)deploy it to an endpoint to get predictions or test it in prompt design. Let’s walk through these three features in detail. First is prompt design. To get started experimenting with large language models, or LLMs, click on NEW PROMPT.

[03:32](javascript:;)In the world of Generative AI, a prompt is just a fancy name for the input text that you feed to your model. You can feed your desired input text like questions and instructions to the model.

[03:42](javascript:;)The model will then provide a response based on how you structured your prompt, therefore, the answers you get depend on the questions you ask. The process of figuring out and designing the best input text to get the desired response

[03:53](javascript:;)back from the model is called Prompt Design, which often involves a lot of experimentation. Let’s start with a free-form prompt. One way to design a prompt is to simply tell the model what you want.

[04:04](javascript:;)In other words, provide an instruction. For example, Generate a list of items I need for a camping trip to Joshua Tree National Park. We send this text to the model,

[04:13](javascript:;)And…you can see that the model outputs a useful list of items we don’t want to camp without. This approach of writing a single command so that the LLM can adopt a certain behavior,

[04:23](javascript:;)is called zero shot prompting. Generally, there are 3 methods that you can use to shape the model's response in a way that you desire. Zero-shot prompting - is a method where the LLM is given no additional data on the specific

[04:35](javascript:;)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 prompt "what is prompt

[04:46](javascript:;)design?". 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

[04:58](javascript:;)poem. and 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

[05:11](javascript:;)articles to read. You can use the structured mode to design the few-shot prompting by providing a context and additional examples for the model to learn from. The structured prompt contains a few different components:

[05:24](javascript:;)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

[05:33](javascript:;)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.

[05:42](javascript:;)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

[05:55](javascript:;)Like what does LGM stand for? Or 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

[06:05](javascript:;)want the model to respond. 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 in just a

[06:14](javascript:;)few minutes! Please note a few best practices around prompt design. Be concise Be specific and well-defined Ask one task at a time Turn generative tasks into classification tasks. For example, instead of asking what programming language to learn, ask if Python, Java, or

[06:33](javascript:;)C is a better fit for a beginner in programming. and Improve response quality by including examples. Adding instructions and a few examples tends to yield good results however there’s currently

[06:44](javascript:;)no 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. For more information about prompt design, please check text prompt design in the reading

[06:57](javascript:;)list. So if you designed a prompt that you think is working pretty well, you can save it and return to it later. Your saved prompt will be visible in the prompt gallery, which is a curated collection of

[07:07](javascript:;)sample prompts that show how generative AI models can work for a variety of use cases. Finally, in addition to testing different prompts and prompt structures, there are a few model parameters you can experiment with to try to improve the quality of responses.

[07:22](javascript:;)First, there are different models you can choose from. Each model is tuned to perform well on specific tasks. You can also specify the temperature, top P, and top K. These parameters all adjust

[07:34](javascript:;)the randomness of responses by controlling how the output tokens are selected. When you send a prompt to the model, it produces an array of probabilities over the words that

[07:43](javascript:;)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.

[07:52](javascript:;)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.

[08:04](javascript:;)By controlling the degree of randomness, you can get more unexpected, and some might say creative, responses. Back to the model parameters, temperature is a number used to tune the degree of randomness.

[08:15](javascript:;)Low temperature: Means to select the words that are highly possible and more predictable. In this case, those are flowers and the other words that are located at the beginning of

[08:23](javascript:;)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

[08:34](javascript:;)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.

[08:47](javascript:;)In addition to adjusting the temperature, top K lets the model randomly return a word from the top K number of words in terms of possibility. For example, top 2 means you get a random word from the top 2 possible words including

[09:00](javascript:;)flowers and trees. This approach allows the other high-scoring word a chance of being selected. However, if the probability distribution of the words is highly skewed and you have one

[09:11](javascript:;)word that is very likely and everything else is very unlikely, this approach can result in some strange responses. The difficulty of selecting the best top-k value, leads to another popular approach that

[09:23](javascript:;)dynamically sets the size of the shortlist of words. 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

[09:38](javascript:;)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.

[09:53](javascript:;)This way, the size of the set of words can dynamically increase and decrease according to the probability distribution of the next word on the list. In sum, Generative AI Studio provides a few model parameters for you to play with such

[10:06](javascript:;)as the model, temperature, top K, and top P. Note that, you are not required to adjust them constantly, especially top k and top p. Now let’s look at the second feature, which creates conversations.

[10:19](javascript:;)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,

[10:32](javascript:;)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

[10:42](javascript:;)queries. Your name is Roy. You are a support technician of an IT department. You only respond with "Have you tried turning it off and on again?" to any queries.

[10:52](javascript:;)You can tune the parameters on the right, the same as you do when designing the prompt. To to see how it works, you can type My computer is slow in the chat box and press enter.

[11:02](javascript:;)The AI responds: Have you tried turning it off and on again? Exactly as you told the AI to do. The cool thing is that Google provides the APIs and SDKs to help you build your own application.

[11:15](javascript:;)You can simply click view code. First, you need to download the Vertex AI SDKs that fit your programming language, like Python and Curl. SDK stands for software design kits.

[11:26](javascript:;)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.

[11:38](javascript:;)Now let’s look at the third feature, tune a language 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.

[11:48](javascript:;)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.

[11:59](javascript:;)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.

[12:09](javascript:;)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.

[12:20](javascript:;)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.

[12:29](javascript:;)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.

[12:40](javascript:;)So what’s tuning? Well, one version you might be familiar with is fine-tuning. In this scenario, we take a model that was pretrained on a generic dataset. We make a copy of this model.

[12:51](javascript:;)Then, using those learned weights as a starting point, we re-train the model on a new domain-specific dataset. This technique has been pretty effective for lots of different use cases.

[13:01](javascript:;)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.

[13:12](javascript:;)Compound all of 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.

[13:21](javascript:;)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.

[13:34](javascript:;)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 on some additional layers to the model or an extra embedding

[13:45](javascript:;)to the prompt. If you want to learn more about parameter-efficient tuning and some of the different methods, a summary paper is included in the reading list of this course.

[13:53](javascript:;)But if you just want to get to building, then let's move to Generative AI Studio and see how to start a tuning job. From the language section of Generative AI Studio, Select TUNING.

[14:03](javascript:;)To create a tuned model, we provide a name. Then point to the local or Cloud Storage location of your training data. Parameter efficient tuning is ideally suited for scenarios where you have "modest" amounts

[14:14](javascript:;)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,

[14:29](javascript:;)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. After specifying the path to your dataset, you can start the tuning job and monitor the

[14:42](javascript:;)status in the Google Cloud console. When the tuning job completes, you’ll see the tuned model in the Vertex AI Model Registry and you can deploy it to an endpoint for serving, or you can test it in the Generative AI Studio.

[14:56](javascript:;)In this course, you learned what Generative AI is and the tools provided by Google Cloud to empower your project with Generative AI capabilities. Specifically, you focused on Generative AI Studio, where you can use genAI in your application

[15:10](javascript:;)by quickly prototyping and customizing generative AI models. You learned that Generative AI Studio supports three options: language, vision, and speech. You then walked through the three major features in Language: design and test prompt, create

[15:24](javascript:;)conversations, and tune models. This was a short lesson introducing Generative AI studio on Vertex AI. For more information about natural language processing and different types of language models like decoder-encoder, transformer, and LLM, please check the course titled Natural

[15:42](javascript:;)Language Processing on Google Cloud listed in the reading list.