Model Fine Tuning

In this module, you learn to ...

01 Fine tune models using customer datasets

02 Create a tuned model using Supervised Tuning in Vertex AI

03 Create a tuned model using Reinforcement Learning from Human Feedback

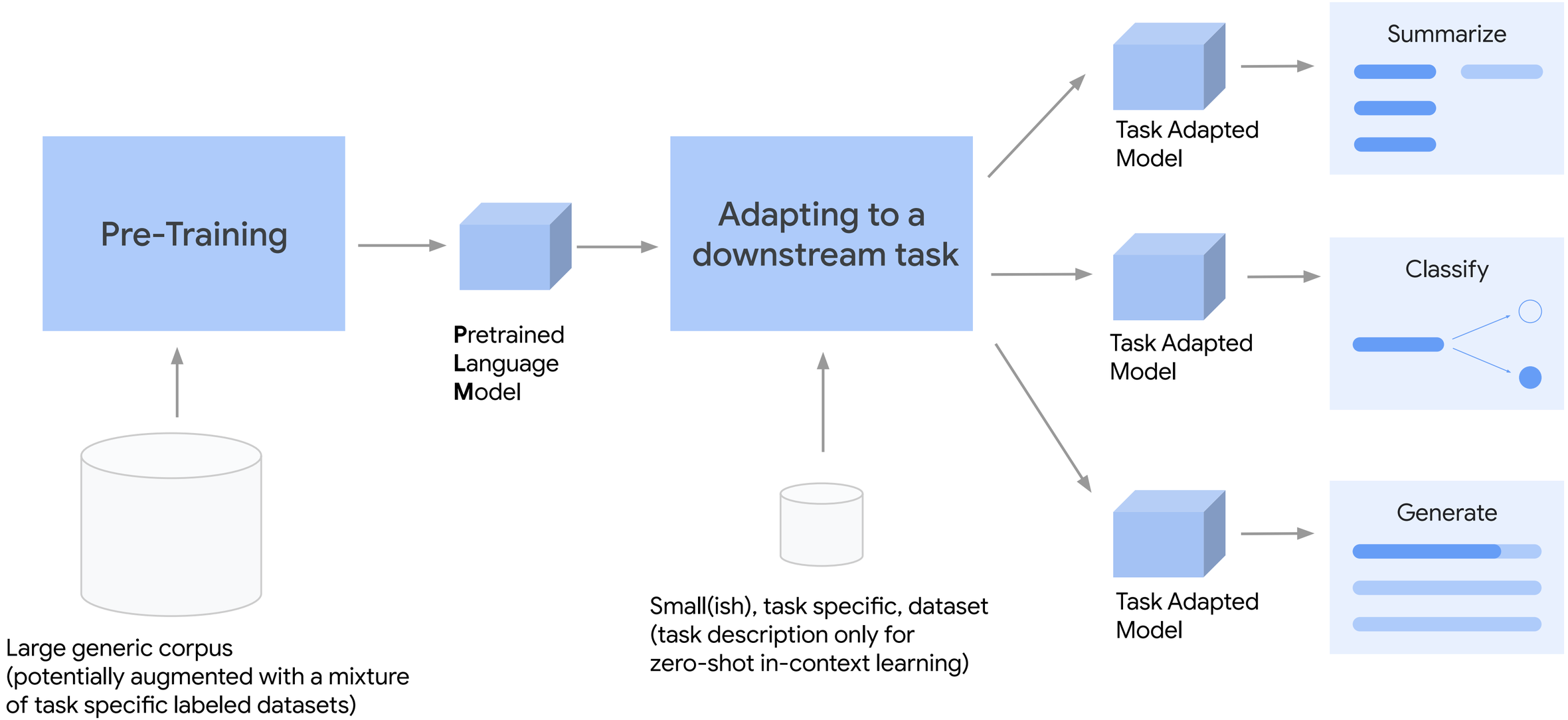
Topics

01	Model Fine Tuning
02	Vertex AI Supervised Tuning
03	Vertex AI RLHF Tuning

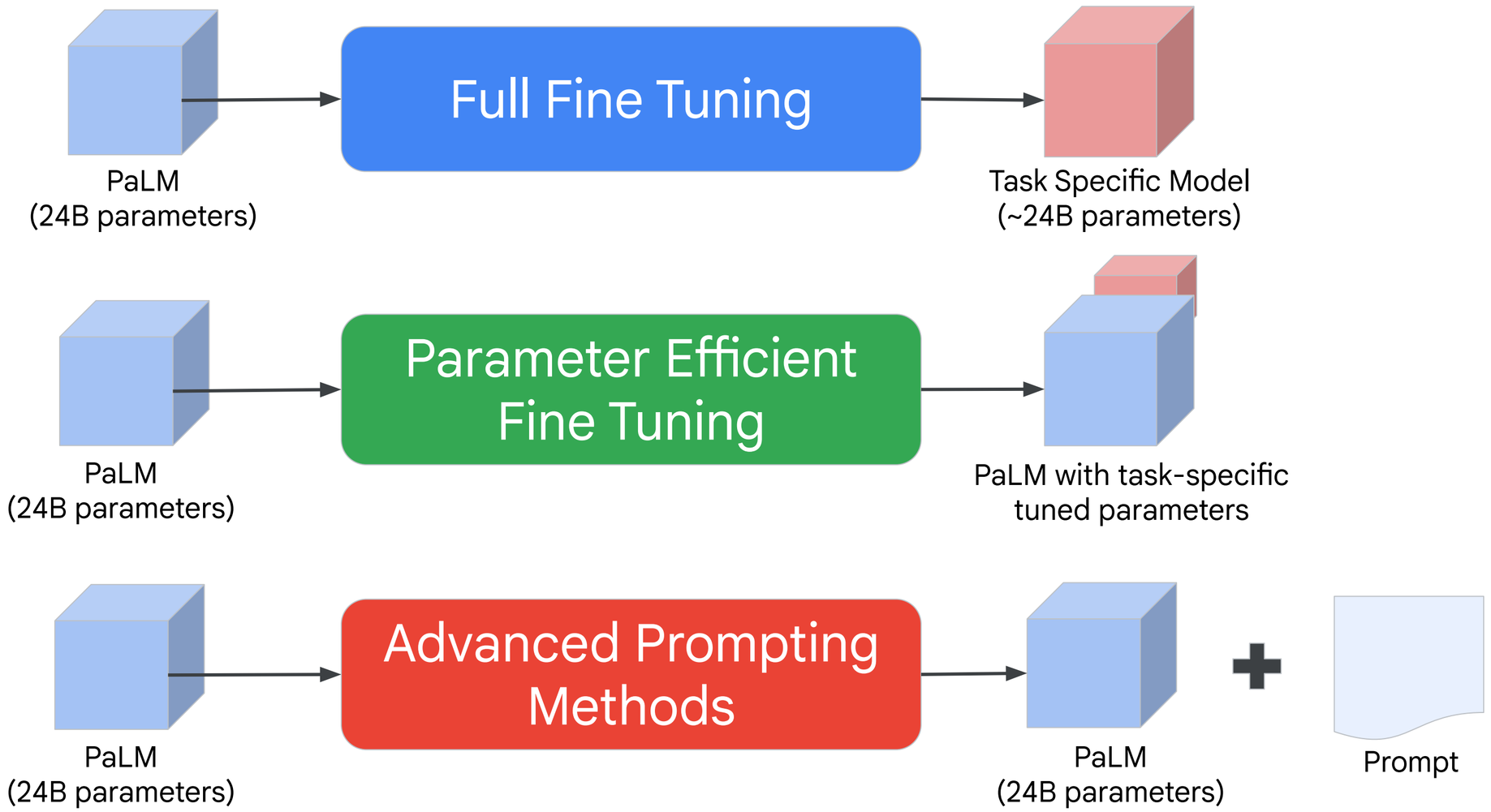
Use model tuning to improve model performance on specific tasks

- When few-shot prompting and adding context are not adequate for your use case
- Allows you to teach the model more about what your expected output should be
- You specify a custom dataset which includes prompts along with the expected output
  - Like adding examples, but more of them with custom training
- The custom training jobs learn the outputs (called weights)

Adapting Large Language Models to downstream tasks



There are different ways to tune models



Tuning strategies

- Full-fine tuning results in a completely new model
  - Very expensive
  - Requires a huge amount of data
  - Med-PaLM is an example of a fully fine tuned model
  - Full fine tuning is currently not available in Vertex AI
- Parameter-efficient fine tuning adds weights to the foundational model
  - Requires much less data (100s to 1000s of examples)
  - Codey is an example of a PEFT model
  - Automated using Vertex AI
- Before fine tuning a model, try to use advanced prompt techniques
  - Add context and examples
  - Use chain of thought prompting

Tuning may be required when you want output that deviates from general language patterns

- Specific structures or formats for generating output
- Specific behaviors such as when to provide a terse or verbose output
- Specific customized outputs for specific types of inputs

When tuning may be helpful: Classification

- Classification with custom classes (groups)
  - Give the model examples, with the correct answers

```
input_text:
Classify the following text into one of the following classes:
[HR, Sales, Marketing, Customer Service].
Text: Are you currently hiring?

output_text:
HR
```

When tuning may be helpful: Summarization

- Summaries that require specific output
- In the example below, you want to remove personally identifiable information (PII) in a chat summary

```
input_text:
Summarize:
Jessica: That sounds great! See you in Times Square!
Alexander: See you at 10!

output_text:
#Person1 and #Person2 agree to meet at Times Square at 10:00 AM
```

When tuning may be helpful: Question answering

- The question is about a context and the answer is a substring of the context

```
input_text:
context: There is evidence that there have been significant changes in
Amazon rainforest vegetation over the last 21,000 years through the Last
Glacial Maximum (LGM) and subsequent deglaciation.
question: What does LGM stand for?

output_text:
Last Glacial Maximum
```

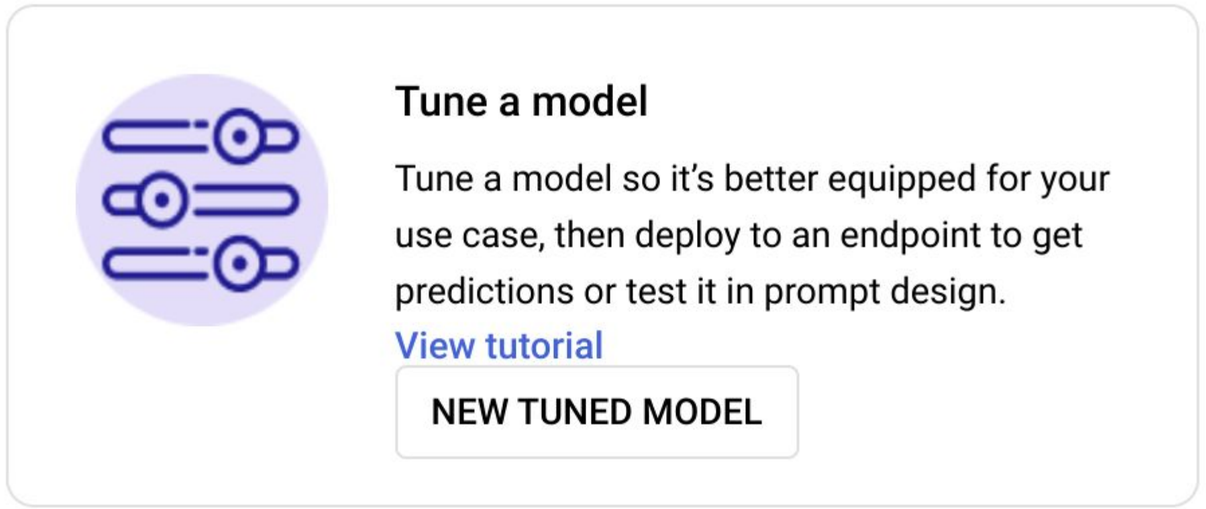
Including context in your training data

- In the example below, the **input_text** consists of both a **context** section and a **question** section

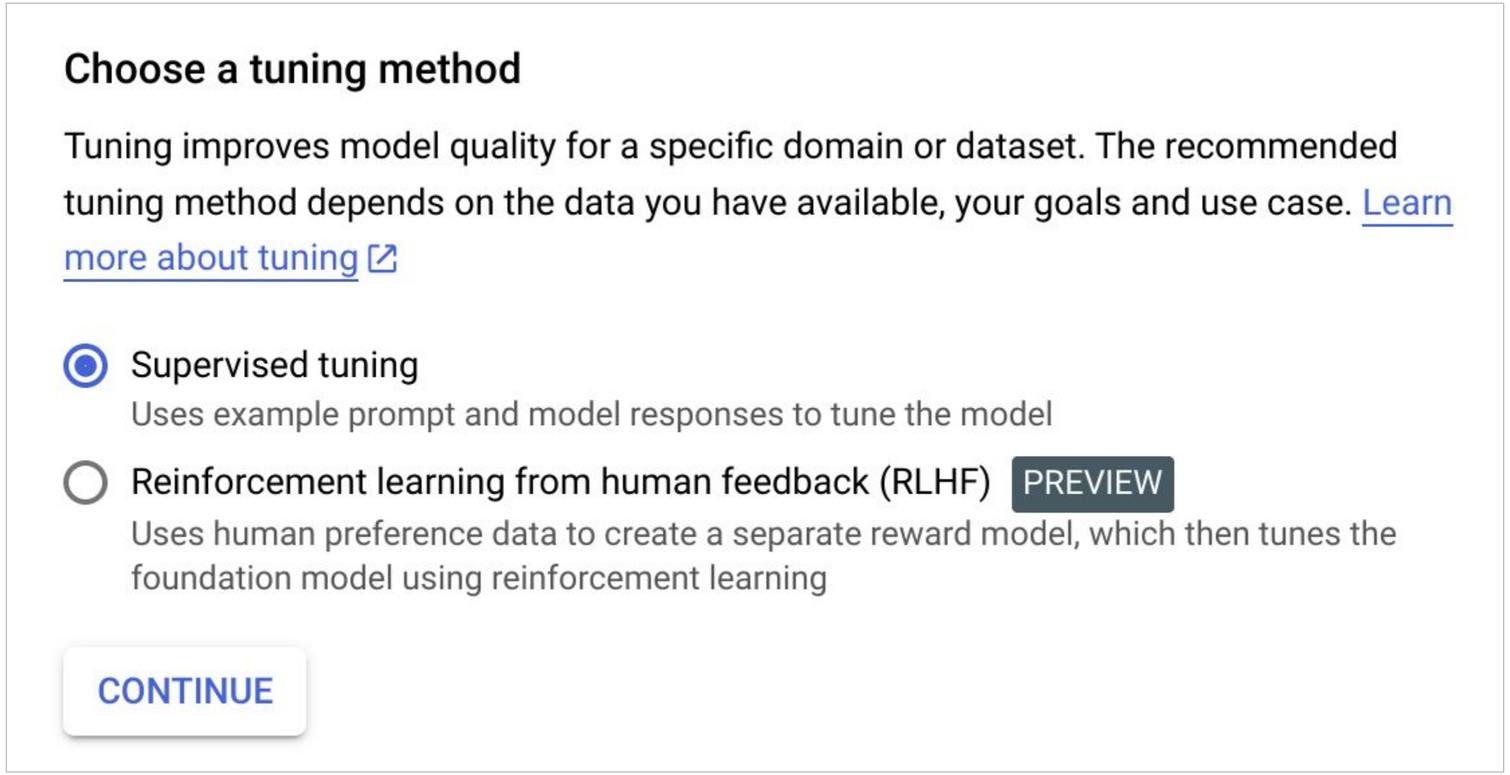
```
input_text:
context: There is evidence that there have been significant changes in
Amazon rainforest vegetation over the last 21,000 years through the Last
Glacial Maximum (LGM) and subsequent deglaciation.
question: What does LGM stand for?
```

- The context provides additional information for answering the question
- If your training data is formatted in this way, you must format prompts in the same way when using your model for inference
  - I.e. Whatever sections your training data has as input, must be included in prompts in the same order when using the model

Select the Tune a model task in Vertex AI Studio



Vertex AI currently supports 2 types of tuning



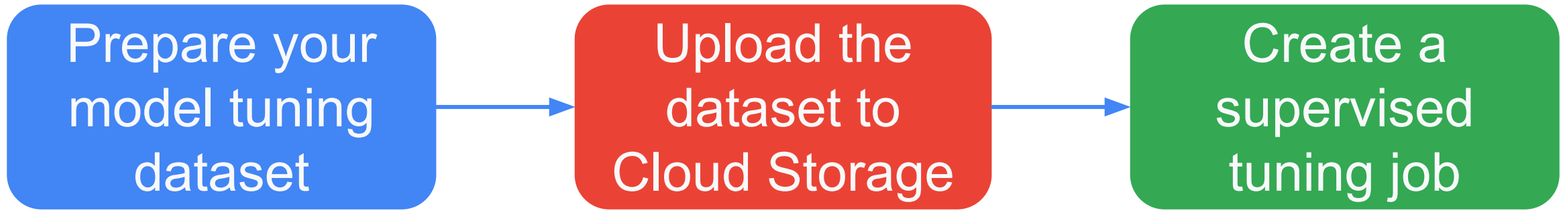
Topics

01	Model Fine Tuning
02	Vertex AI Supervised Tuning
03	Vertex AI RLHF Tuning

Supervised tuning

- Supervised tuning improves the performance of a model by teaching it to format its output more specifically:
  - Uses hundreds of examples to teach the model to mimic a desired output pattern
  - Examples demonstrate what you want the model to output during inference
- Learns additional parameters that help it encode the necessary information to perform the desired task or learn the desired behavior
  - Uses these parameters during inference
- Outputs a new model with layers that sit on top of the original model
  - The original model is untouched (PEFT)
- Can be used with the following models (*currently*)
  - text-bison, chat-bison, code-bison, chatcode-bison, textembedding-gecko

Supervised model tuning workflow on Vertex AI



- After tuning, the model is automatically deployed to a Vertex AI endpoint using the name you provide in the tuning job
- The model is also available in Vertex AI Studio when creating prompts

Prepare your model tuning dataset for supervised tuning

- The training data must be in JSONL format
  - The “L” is for “Line”
  - Each line in the JSONL file is one example
  - It is not an array of objects, it is one object per line
- Each object must have the properties `input_text` and `output_text`

{} custom-training-data.jsonl ×

Users > doug > {} custom-training-data.jsonl

```
1  {"input_text": "question: How many copies of Gears of War 3 were sold ? context: Like
2  {"input_text": "question: How many parishes are there in Louisiana ? context: The U .
```

It is important to include the same instructions you will use at prediction time in your training data

- The following has no instructions, so it is not a good example

```
{"input_text": "5 stocks to buy now","output_text": "business"}
```

- The following has instructions, so it is a better example

```
{"input_text": "Classify the following text into one of the following classes:
[business, entertainment] Text: 5 stocks to buy now","output_text": "business"}
```

Include context within the input text

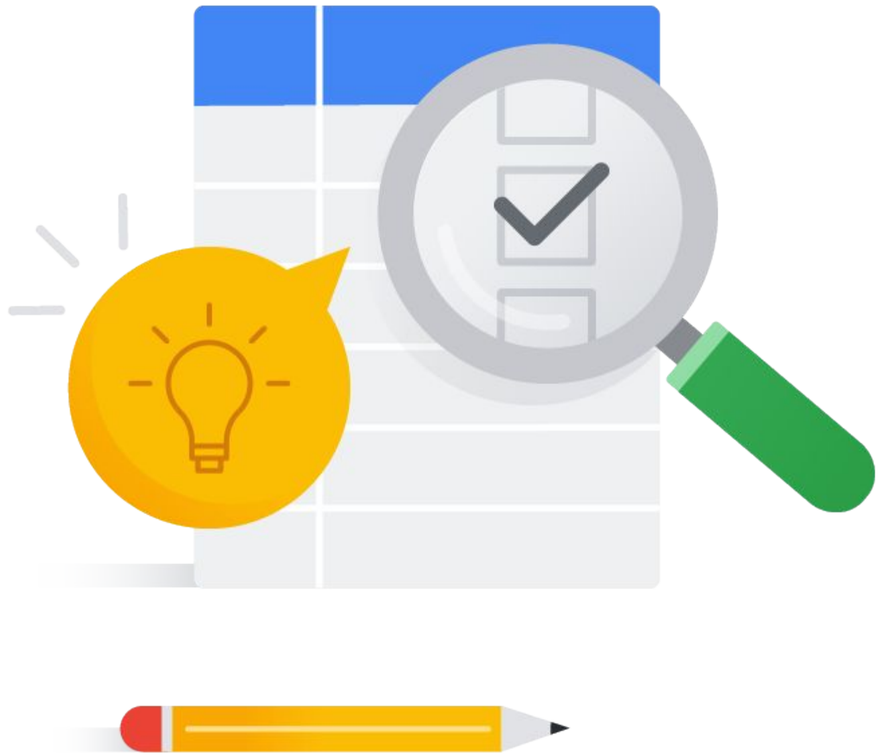
- Notice that the following `input_text` has question and context sections
  - When using the model, remember that prompts need to be formatted the same way
  - Be consistent

```
{"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"}
```

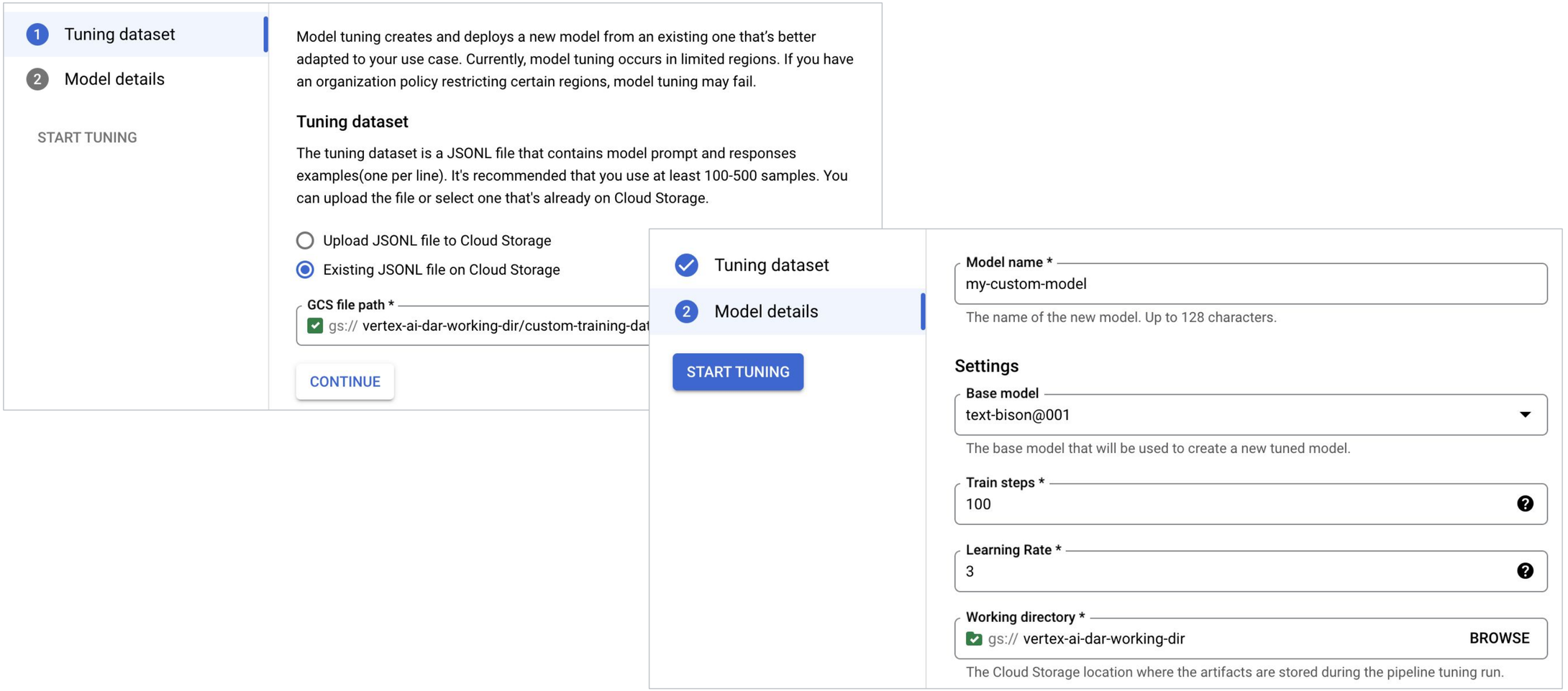
Do Now: Exploring Sample Training Data

5 min

1. Go to: https://github.com/roitraining/genai-model-tuning-examples
2. You will find some example fine-tuning datasets
3. Click on a couple of them and explore the examples
   a. Each file has 1 example per line
   b. Each example hss input_text and output text attributes



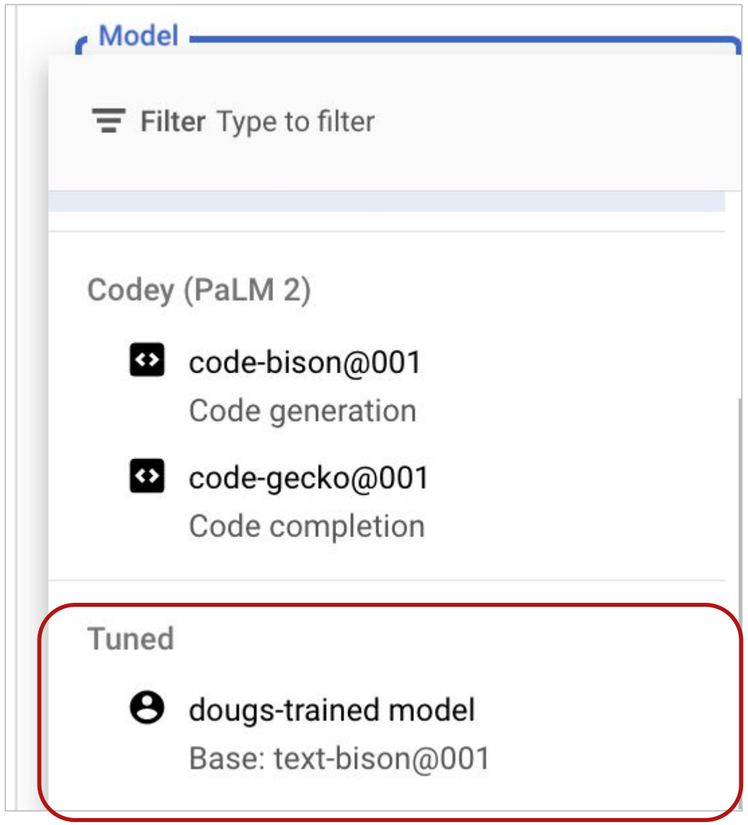
Specify the location of the data and the job parameters



The number of training examples and training steps needed depends on the task

Task	Suggested # of examples	Training steps
Classification	100+	100-500
Summarization	100-500+	200-1000
Extractive QA	100+	100-500

Tuned models are available from Vertex AI Studio



Vertex AI Studio will generate the code to use tuned models

View code

PYTHON | PYTHON COLAB | CURL

Use this script to request a model response in your application.

1. Set up the Vertex AI SDK for Python
2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.preview.language_models import TextGenerationModel

vertexai.init(project="982785856251", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
model = model.get_tuned_model("projects/982785856251/locations/us-central1/models/167990643268360
response = model.predict(
    """,
    **parameters
)
print(f"Response from Model: {response.text}")
```

Topics

01	Model Fine Tuning
02	Vertex AI Supervised Tuning
03	Vertex AI RLHF Tuning

Reinforcement Learning provides a way for a model to learn to achieve a given objective without examples

- The foundational principle of RL to train an agent so it maximizes the cumulative reward it receives (for its actions) in the long run
- The reward function can be learned: reward model
- The reward model can be trained to capture human preferences and intents
- The reward signal generated by the reward model can be used tune an LLM to better align with human preferences and intents

Reinforcement Learning from Human Feedback (RHLF)

- RLHF optimizes language models using human-specified preferences.
  - Improves model alignment with human preferences.
  - Reduces undesired outcomes in tasks with complex human intuitions.
- For example, RLHF helps with ambiguous tasks like writing creative content
  - It involves presenting two options to a human and letting them choose their preferred one
- A dataset of prompts, options, and the preference is created and used for training
- RLHF is currently supported with the text-bison model

Reward Model training data: which completion is better?

Input

Explain the moon landing to a 6 year old in a few sentence

Completion 1

The Moon is a natural satellite of the Earth. It is the fifth largest moon in the Solar System and the largest relative to the size of its host planet.

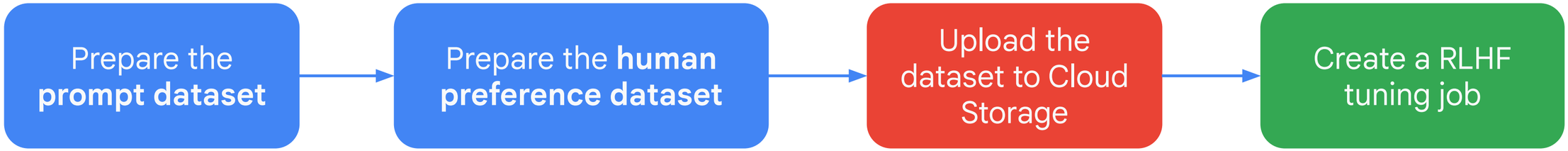
<

Completion 2

People went to the moon, and they took pictures of what they saw, and sent them back to earth so we could all see them

Source: Hyung Won Chung - NYU CSCI 2590 lecture

RHLF model tuning workflow on Vertex AI



- After tuning, the model is automatically deployed to a Vertex AI endpoint using the name you provide in the tuning job
- The model is also available in Vertex AI Studio when creating prompts

Prompt dataset

- Contains unlabeled prompts
  - Each line contains one `input_text` field that specifies the unlabeled prompt
  - Can be the same or different as the `input_text` fields in the human preference dataset
- Example:

```
{"input_text": "Create a description for Plantation Palms."}
```

Human preference dataset

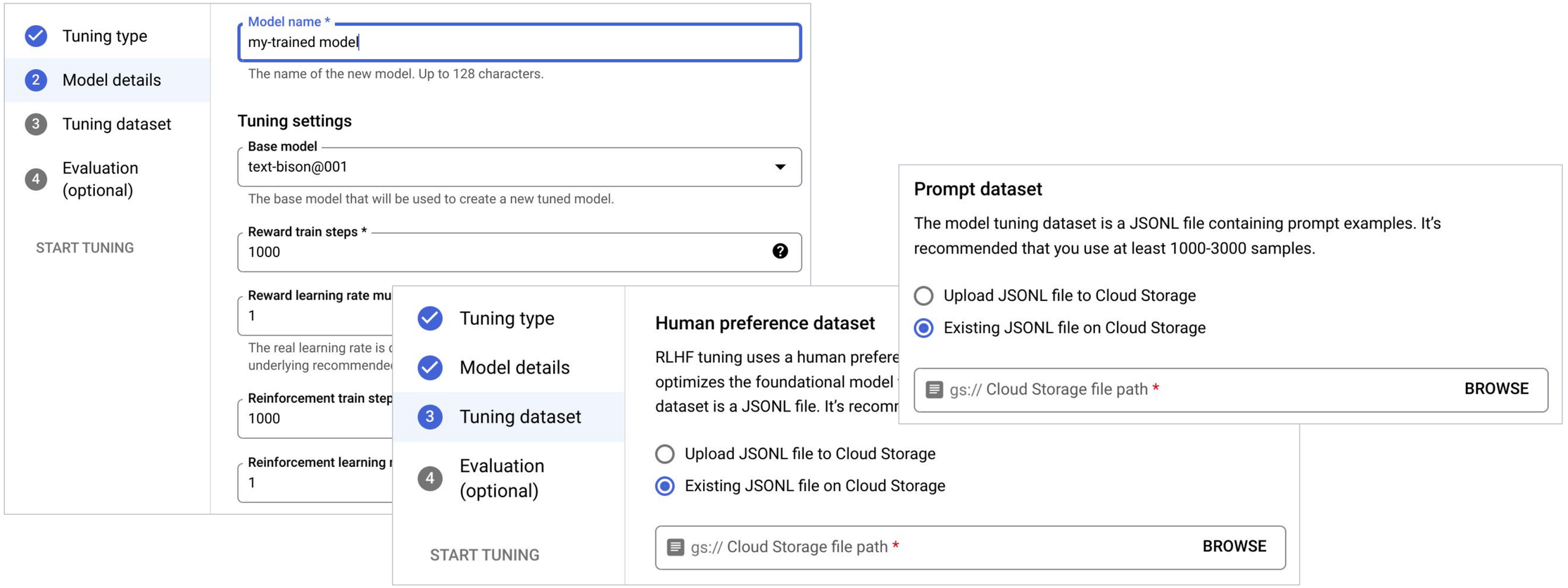
- Contains labeled prompts
- Each line contains:
  - One **`input_text`** field
  - Two **`candidate`** responses
  - The preferred response (**`choice`**)

```
{
  "input_text": "Create a description for Plantation Palms.",
  "candidate_0": "Enjoy some fun in the sun at Gulf Shores.",
  "candidate_1": "A Tranquil Oasis of Natural Beauty.",
  "choice": 0
}
```

You can optionally train an evaluation dataset

- Same format as the prompt dataset
- During tuning, a reward model is created
  - If the model generates a preferred response it is rewarded

Set training job parameters and run it



Creating an RLHF pipeline

```
import google.cloud.aiplatform as aiplatform
from google_cloud_pipeline_components.preview.llm import rlfh_pipeline
from kfp import compiler

compiler.Compiler().compile(
    pipeline_func=rlfh_pipeline, package_path="rlfh_pipeline.yaml"

job = aiplatform.PipelineJob(
display_name="my-rlhf-tuned-model",
template_path="rlfh_pipeline.yaml",
pipeline_root="gs://my_bucket/my_rlfh_pipeline_root",
parameter_values = {
    "preference_dataset": "gs://my_bucket/data/preference/*.jsonl",
    "prompt_dataset": "gs://my_bucket/data/prompt/*.jsonl",
    "large_model_reference": "text-bison@001",
    "reward_model_train_steps": 100,
    "reinforcement_learning_train_steps": 100,
    "kl_coeff": 0.1,...})
job.run()
```

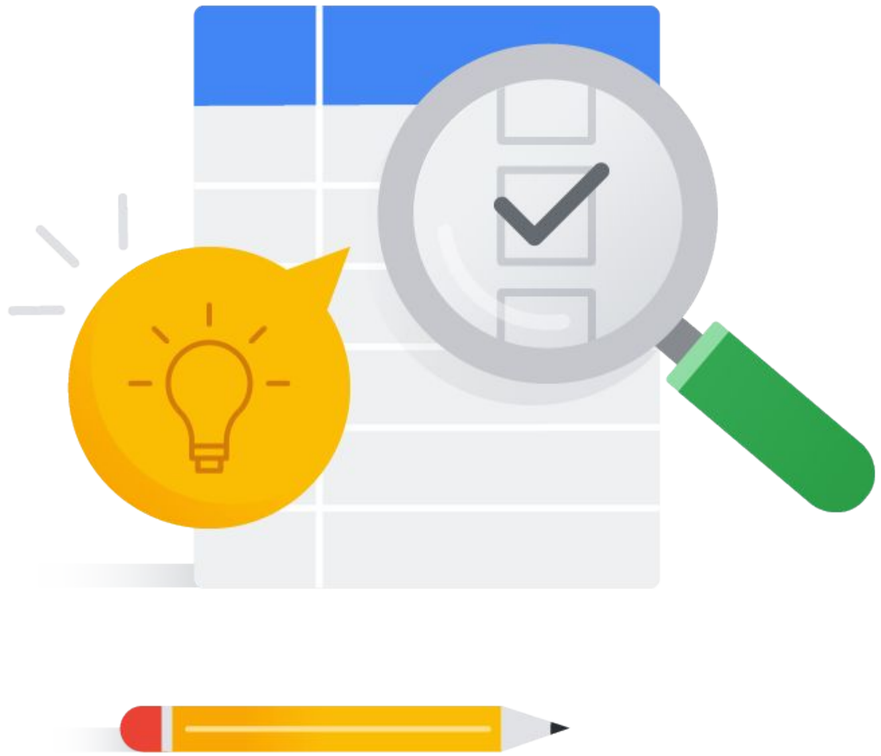
Creating a high quality preference dataset is extremely hard

- OpenAI hired a team of 40 contractors to label data. They were selected based on their performance on a sophisticated screening test
- They had access to a large corpus of real prompts submitted by their users through OpenAI API
- In addition to rating responses from the models, the labelers also created a large sets of both prompts and expected completions
- They used “held-out” labelers that did not produce any training data for evaluation. They also had automated evaluation instrumentation for standard OSS datasets.
- The bottom line: the process of creating a good quality preference dataset can be very labor intensive and expensive

Lab

45 min

Lab: Generative AI Workshop Model Tuning



In this module, you learned to ...

01 Fine tune models using customer datasets

02 Create a tuned model using Supervised Tuning in Vertex AI

03 Create a tuned model using Reinforcement Learning from Human Feedback

Questions and answers

Quiz question

Which of the following methods can help you tune a model for a specific task?

A: Full fine tuning

B: Parameter efficient fine tuning

C: Prompt engineering

D: All of the above

Quiz question

Which of the following methods can help you tune a model for a specific task?

A: Full fine tuning

B: Parameter efficient fine tuning

C: Prompt engineering

D: All of the above

Quiz question

What tuning methods are supported in Vertex AI? (Choose all that apply)

A: Full fine tuning

B: Supervised parameter efficient fine tuning

C: Reinforcement Learning from human feedback

D: Unsupervised tuning

Quiz question

What tuning methods are supported in Vertex AI? (Choose all that apply)

A: Full fine tuning

B: Supervised parameter efficient fine tuning

C: Reinforcement Learning from human feedback

D: Unsupervised tuning

Quiz question

If you are working on a specific task and need fine control over what the LLM returns, what method should you try first?

A: Full fine tuning

B: Supervised parameter efficient fine tuning

C: Reinforcement Learning from human feedback

D: Prompt engineering

Quiz question

If you are working on a specific task and need fine control over what the LLM returns, what method should you try first?

A: Full fine tuning

B: Supervised parameter efficient fine tuning

C: Reinforcement Learning from human feedback

D: Prompt engineering


Google Cloud