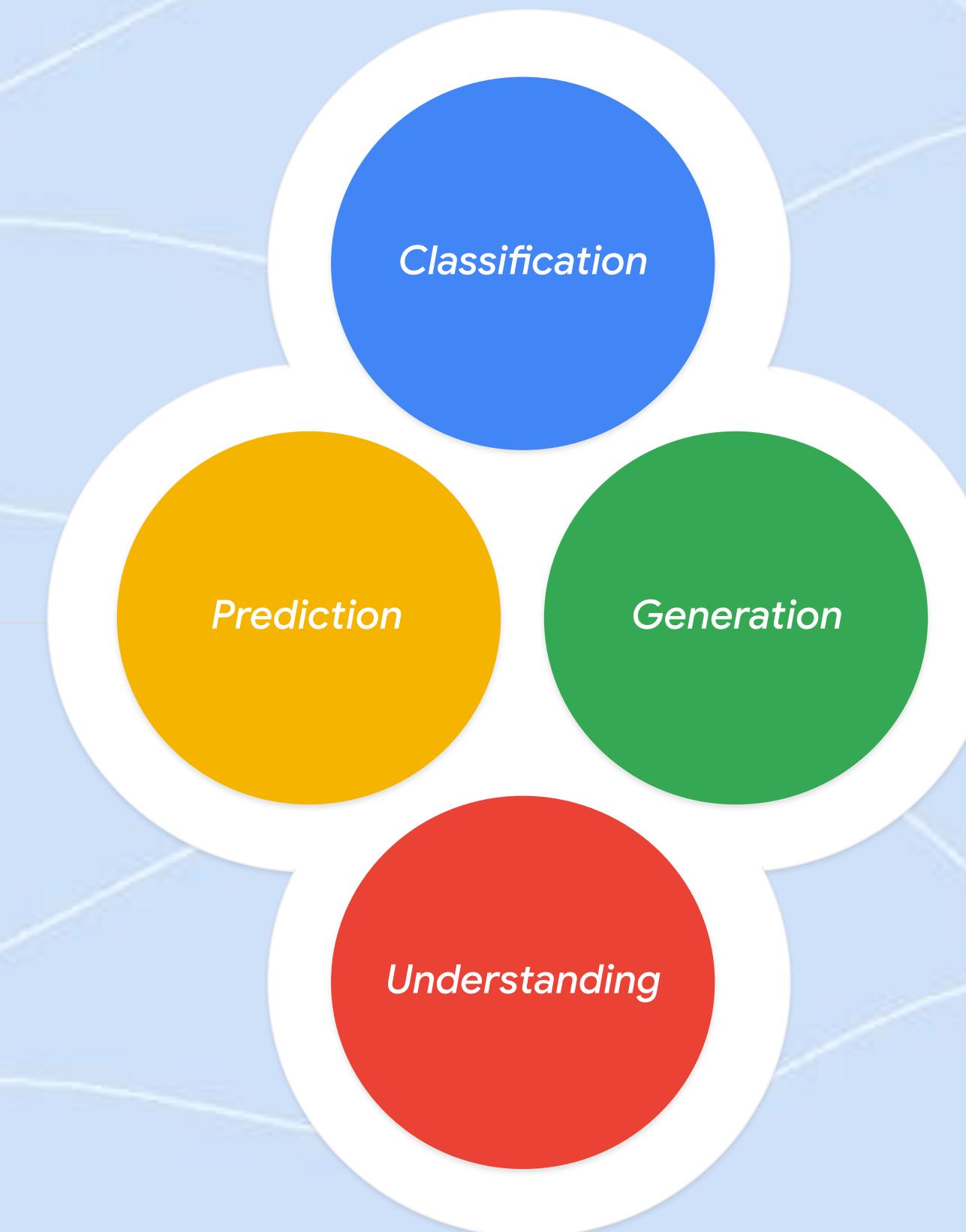


AI is evolving from analysis to creation



Generative AI

- Generative AI is a type of artificial intelligence that can create new content, such as text, images, or music
- It works by learning from large datasets of existing content and then using that knowledge to generate new content that is similar to the training data
- Generative AI has a wide range of potential applications, including:
 - Content creation: Generative AI can be used to create new content, such as articles, blog posts, or even entire books.
 - Data augmentation: Generative AI can be used to augment existing datasets by creating new data that is similar to the existing data. This can be helpful for training machine learning models.
 - Creative applications: Generative AI can be used to create new and innovative art, music, and other creative content.

Bard is a Google consumer-oriented generative AI tool

The
previous
slide!

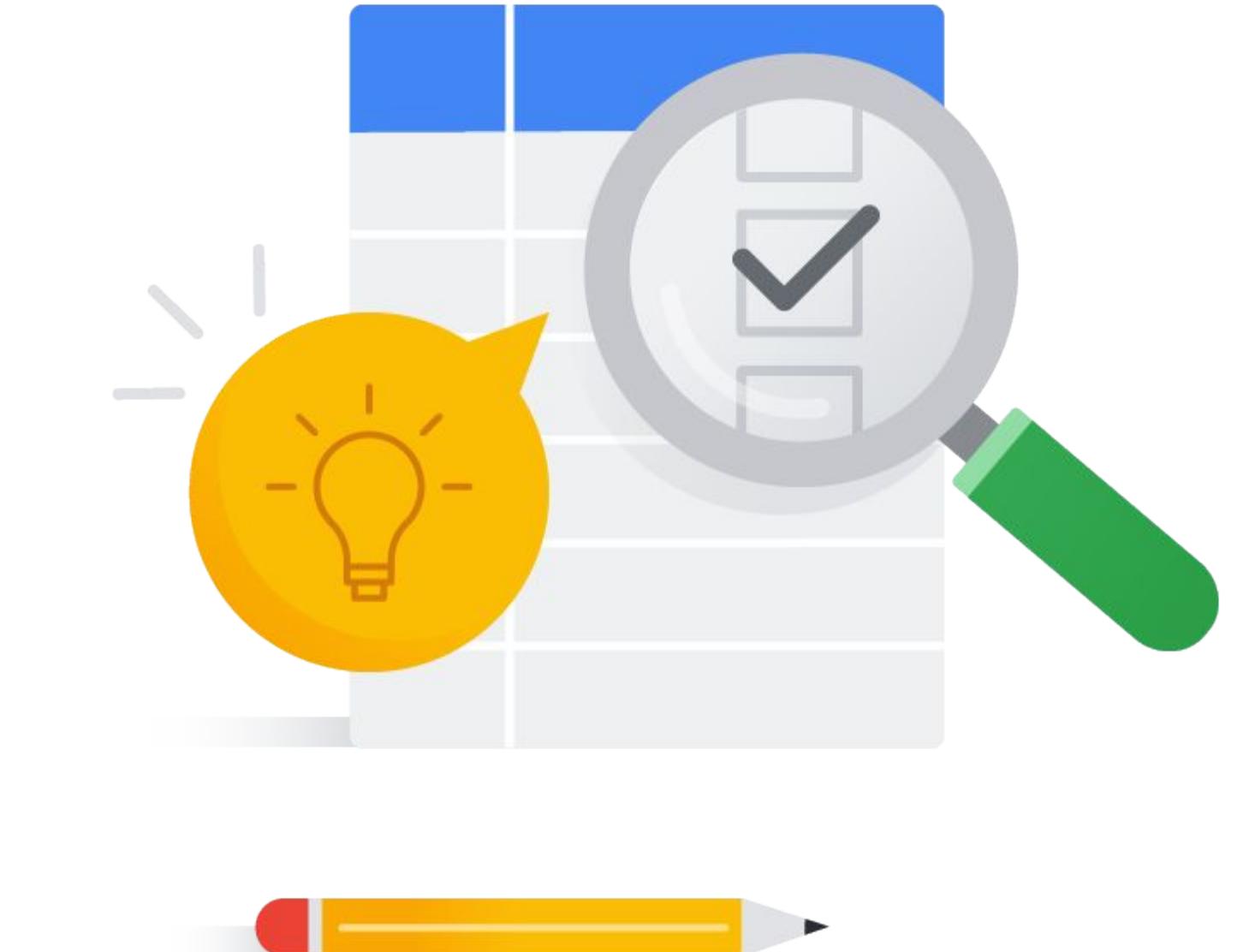
<https://bard.google.com/>

The screenshot shows the Bard AI interface. A blue callout box on the left says "The previous slide!" with a blue arrow pointing to the interface. The interface has a light gray background. At the top, there's a user prompt: "Write me a slide for a training course that describes Generative AI" with a pencil icon. Below it is a "View other drafts" dropdown. The AI response starts with a blue starburst icon and the text: "Sure, here is a slide for a training course that describes Generative AI:". It includes a "Slide Title: Generative AI" section and a "Slide Content:" section with a bulleted list: "• Generative AI is a type of artificial intelligence that can create new content, such as text, images, or music.", "• It works by learning from large datasets of existing content and then using that knowledge to generate new content that is similar to the training data.", "• Generative AI has a wide range of potential applications, including: ○ Content creation: Generative AI can be used to create new content, such as articles, blog posts, or even entire books.", and "○ Data augmentation: Generative AI can be used to augment existing datasets by creating new". At the bottom, there's a text input field "Enter a prompt here" with a microphone icon, and a send button with a right-pointing arrow.

Day 1: Student Sandbox Environment

1 Day

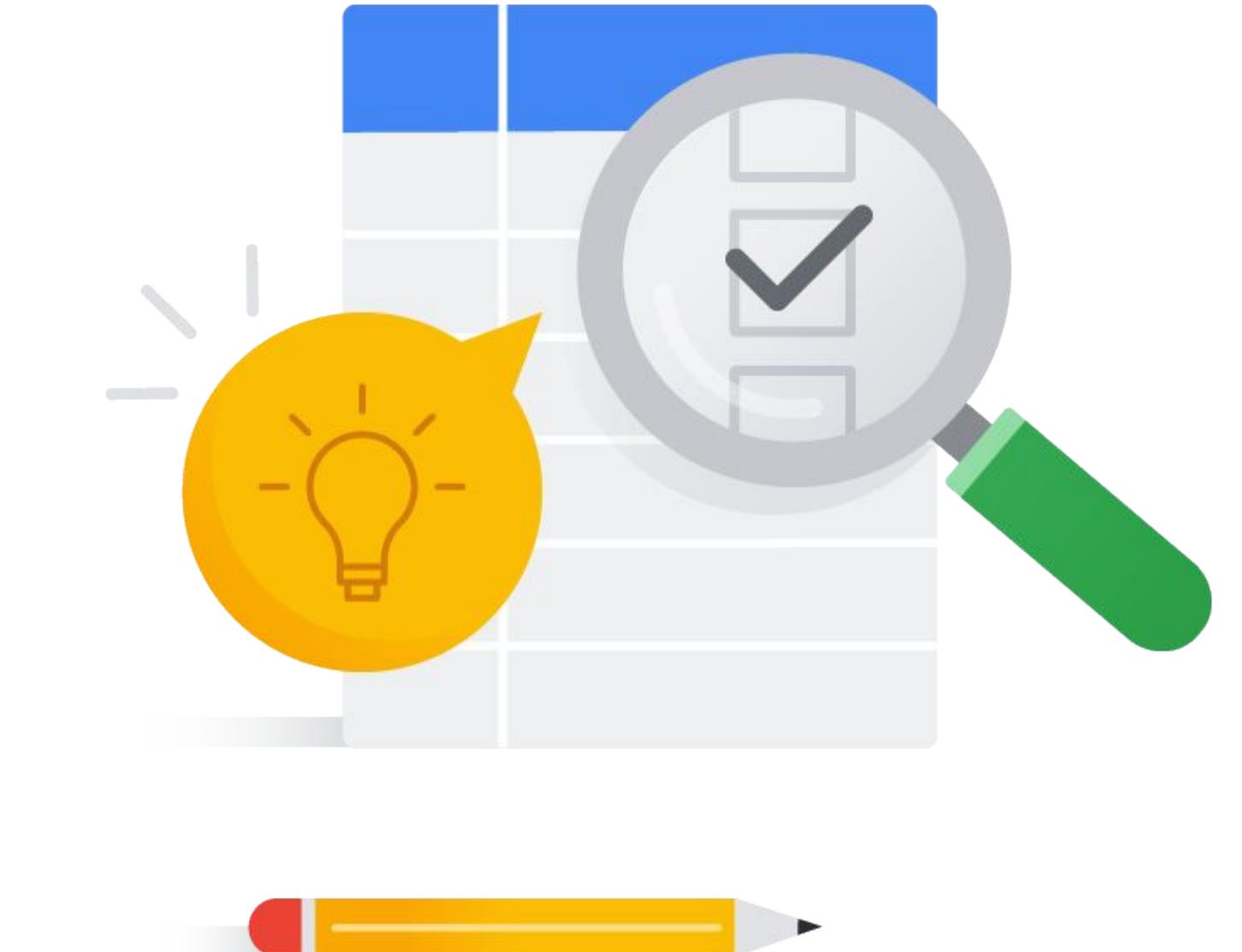
[Day 1: Building Generative AI Applications with Vertex AI Challenge Labs](#)



Day 2: Student Sandbox Environment

1 Day

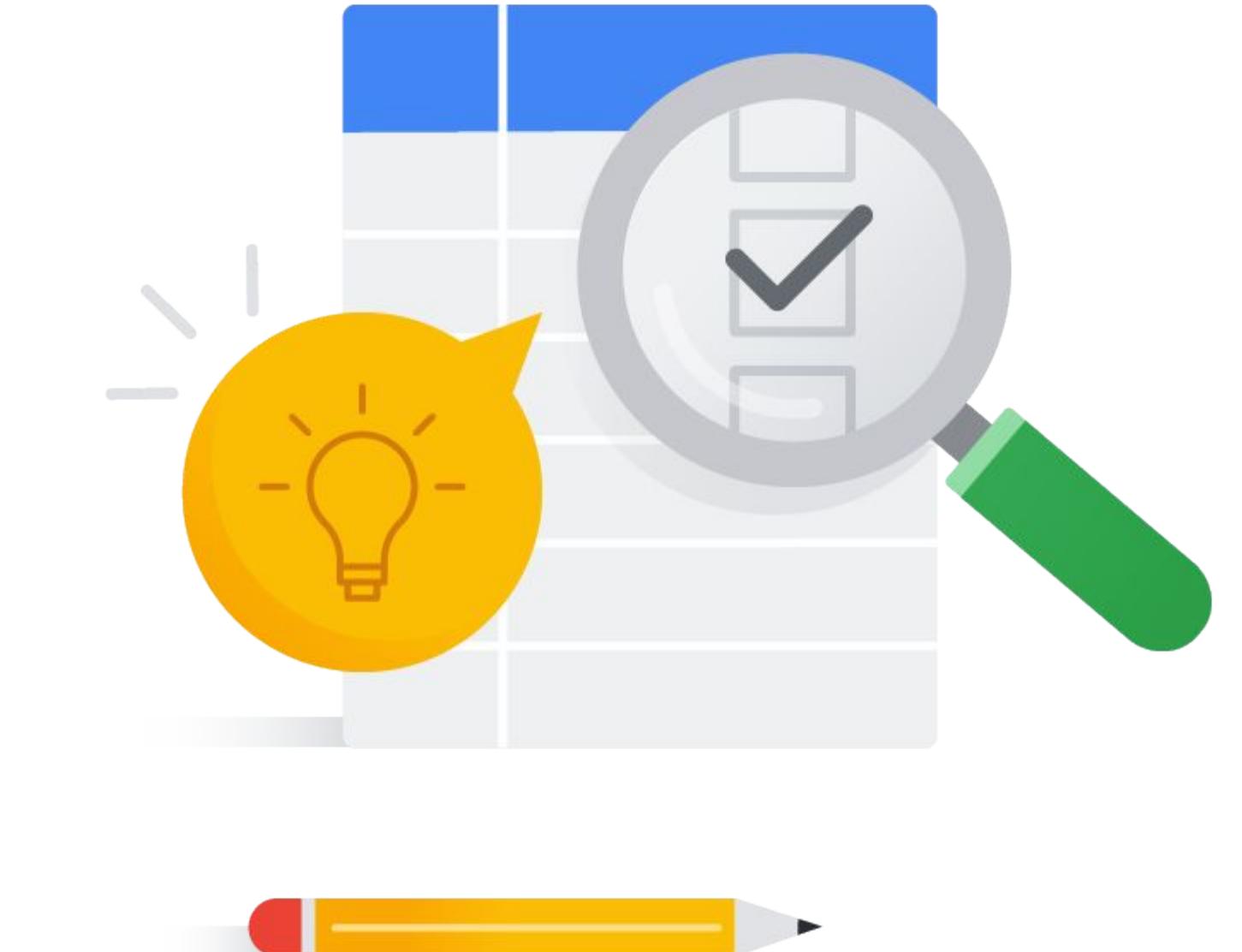
[Day 2: Building Generative AI Applications with Vertex AI Challenge Labs](#)



Day 3: Student Sandbox Environment

1 Day

[Day 3: Building Generative AI Applications with Vertex AI Challenge Labs](#)



Now let's learn to build our own generative AI tools.



Diving Deeper into Generative AI

In this module, you learn to ...

- 01 Leverage foundation models for generative AI
- 02 Use Google's foundational models to optimize generative AI tasks
- 03 Get started building generative AI projects using Vertex AI Studio
- 04 Explore the various use cases for generative AI
- 05 Find and implement models using Vertex AI Model Garden



Topics

- 01 The Benefits of Foundation Models
- 02 Google's Foundation Models
- 03 Vertex AI Studio
- 04 Generative AI Use Cases
- 05 More Foundation Models: Model Garden
- 06 Vertex AI Search



Foundation models

[BERT](#) [Oct 2018]: Pre-text task with ~340M transformer model

[XLNet](#) [June 2019]: Autoregressive Pretraining

[GPT-3](#) [May 2020]: Chatbot model at extreme scales (175B)

[CLIP](#) [Jan 2021]: Image captioning using pre-training tricks inspired by BERT (63M)

[DALL·E](#) [Jan 2021]: Text-to-image generation with a "mini" GPT-3 (12B)

HAI: Human-centered Artificial Intelligence

On the Opportunities and Risks of Foundation Models

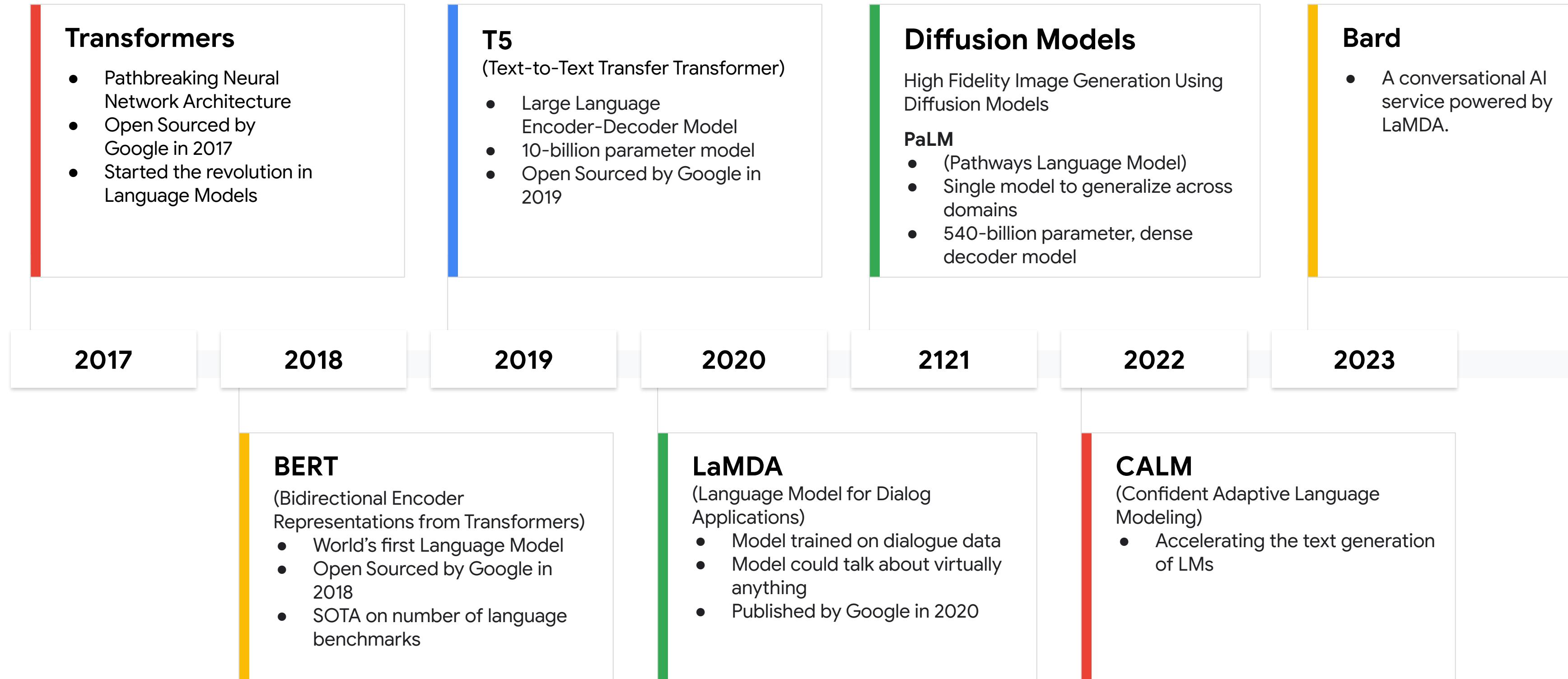
Rishi Bommasani* Drew A. Hudson Ehsan Adeli Russ Altman Simran Arora
 Sydney von Arx Michael S. Bernstein Jeannette Bohg Antoine Bosselut Emma Brunskill
 Erik Brynjolfsson Shyamal Buch Dallas Card Rodrigo Castellon Niladri Chatterji
 Annie Chen Kathleen Creel Jared Quincy Davis Dorottya Demszky Chris Donahue
 Moussa Doumbouya Esin Durmus Stefano Ermon John Etchemendy Kawin Ethayarajh
 Li Fei-Fei Chelsea Finn Trevor Gale Lauren Gillespie Karan Goel Noah Goodman
 Shelby Grossman Neel Guha Tatsunori Hashimoto Peter Henderson John Hewitt
 Daniel E. Ho Jenny Hong Kyle Hsu Jing Huang Thomas Icard Saahil Jain
 Dan Jurafsky Pratyusha Kalluri Siddharth Karamcheti Geoff Keeling Fereshte Khani
 Omar Khattab Pang Wei Koh Mark Krass Ranjay Krishna Rohith Kuditipudi
 Ananya Kumar Faisal Ladhak Mina Lee Tony Lee Jure Leskovec Isabelle Levent
 Xiang Lisa Li Xuechen Li Tengyu Ma Ali Malik Christopher D. Manning
 Suvir Mirchandani Eric Mitchell Zanele Munyikwa Suraj Nair Avanika Narayan
 Deepak Narayanan Ben Newman Allen Nie Juan Carlos Niebles Hamed Nilforoshan
 Julian Nyarko Giray O gut Laurel Orr Isabel Papadimitriou Joon Sung Park Chris Piech
 Eva Portelance Christopher Potts Aditi Raghunathan Rob Reich Hongyu Ren
 Frieda Rong Yusuf Roohani Camilo Ruiz Jack Ryan Christopher Ré Dorsa Sadigh
 Shiori Sagawa Keshav Santhanam Andy Shih Krishnan Srinivasan Alex Tamkin
 Rohan Taori Armin W. Thomas Florian Tramèr Rose E. Wang William Wang Bohan Wu
 Jiajun Wu Yuhuai Wu Sang Michael Xie Michihiro Yasunaga Jiaxuan You Matei Zaharia
 Michael Zhang Tianyi Zhang Xikun Zhang Yuhui Zhang Lucia Zheng Kaitlyn Zhou
 Percy Liang^{*1}

Center for Research on Foundation Models (CRFM)
 Stanford Institute for Human-Centered Artificial Intelligence (HAI)
 Stanford University

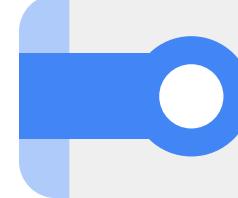
arXiv:2108.07258v3 [cs.LG] 12 Jul 2022

AI is undergoing a paradigm shift with the rise of models (e.g., BERT, DALL-E, GPT-3) trained on broad data (generally using self-supervision at scale) that can be adapted to a wide range of downstream tasks. We call these models foundation models to underscore their critically central yet incomplete character. This report provides a thorough account of the opportunities and risks of foundation models, ranging from their capabilities (e.g., language, vision, robotic manipulation, reasoning, human interaction) and technical principles (e.g., model architectures, training procedures, data, systems, security, evaluation, theory) to their applications (e.g., law, healthcare, education) and societal impact (e.g., inequity, misuse, economic and environmental impact, legal and ethical considerations). Though foundation models are based on standard deep learning and transfer learning, their scale results in new emergent capabilities, and their effectiveness across so many tasks incentivizes homogenization. Homogenization provides powerful leverage but demands caution, as the defects of the foundation model are inherited by all the adapted models downstream. Despite the impending widespread deployment of foundation models, we currently lack a clear understanding of how they work, when they fail, and what they are even capable of due to their emergent properties. To tackle these questions, we believe much of the critical

This revolution started at Google



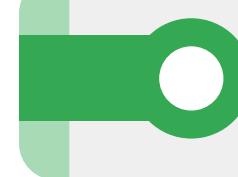
Large Language Models (LLMs)



ML algorithms that can **recognize, predict, and generate** human languages



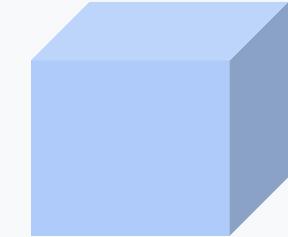
Pre-trained on petabyte scale text-based datasets resulting in large models with **10s to 100s of billions of parameters**



LLMs are normally **pre-trained on a large corpus of text** followed by fine-tuning on a specific task



LLMs can also be called **Large Models** (includes all types of data modality) and **Generative AI** (a model that produces content)

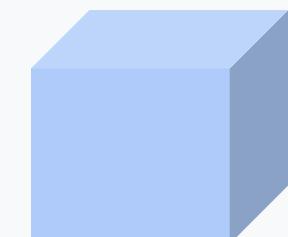


Go read this huuuuuge pile of books.



So, you've learned about cats and millions of other concepts ... what's a cat?

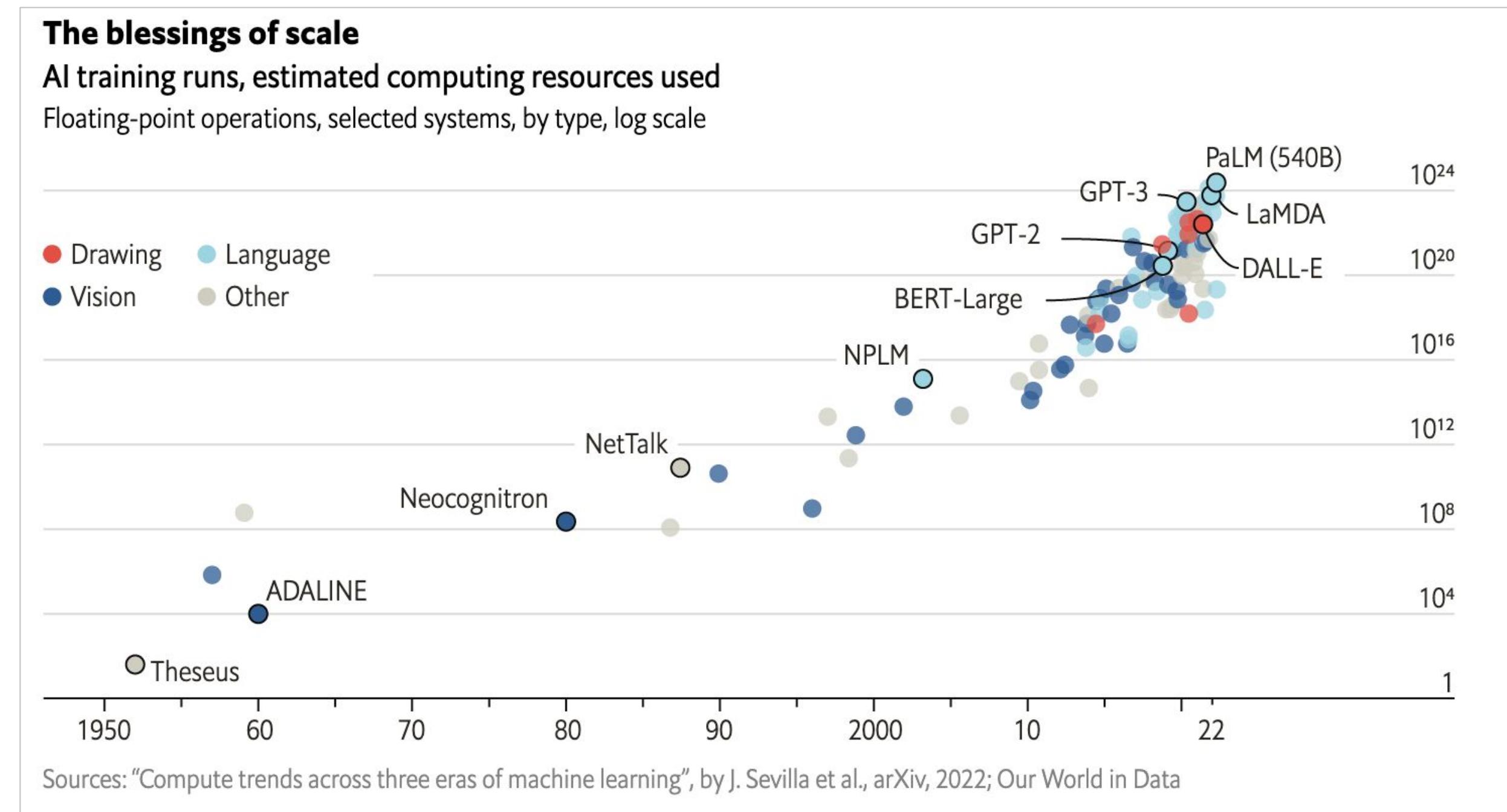
A cat is a small, domesticated carnivorous mammal.



Generative language models

LaMDA, PaLM, GPT-3, etc.

LLMs have driven an explosion in model size



[Source: The Economist, June 2022](#)



Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

[L] 29 Mar 2022

We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4x more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

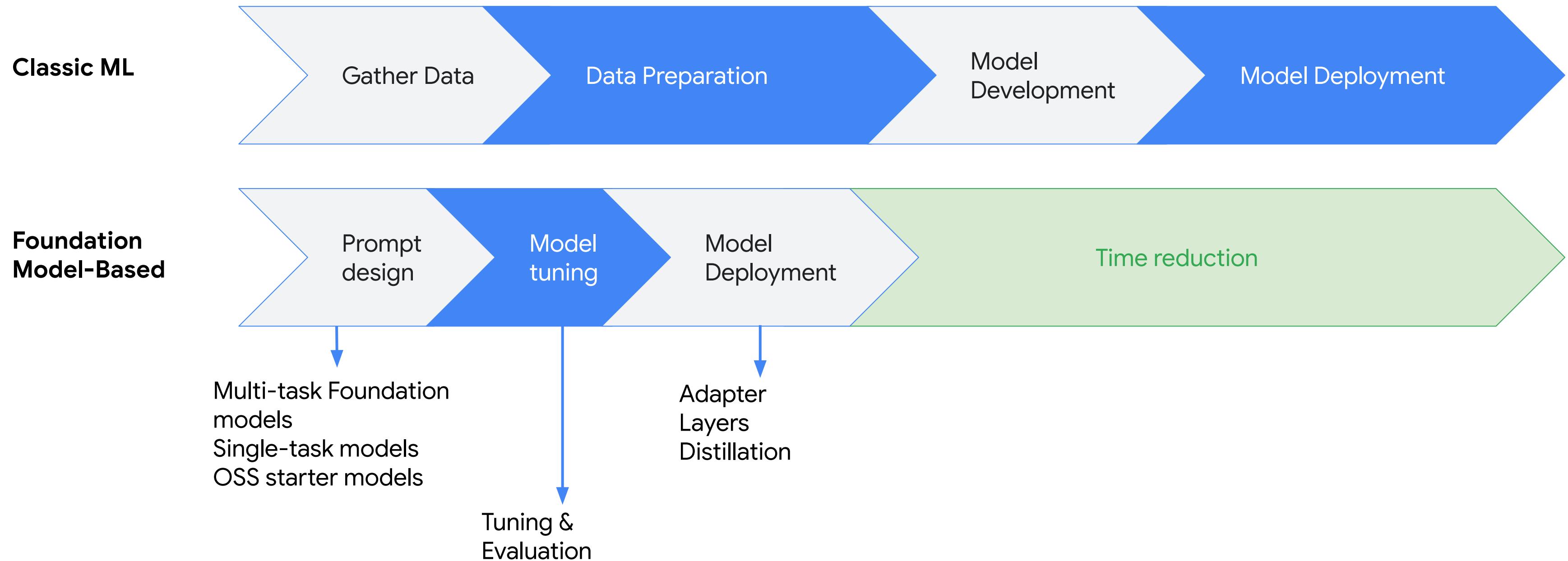
Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
<i>Gopher</i> (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
<i>Chinchilla</i>	70 Billion	1.4 Trillion

“... we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled.”

[1] ([Rae et al., 2021](#); [Thoppilan et al., 2022](#))

Foundational models accelerate time to model deployment



There are also challenges with generative AI

- Can be difficult to control the quality of generated content
- Can be difficult to ensure that generated content is accurate
 - Untrue statements can be presented in a confident manner
 - These are known as hallucinations in generative AI terms
- Can be difficult to ensure that generated content is not offensive or harmful



Topics

01 The Benefits of Foundation Models

02 Google's Foundation Models

03 Vertex AI Studio

04 Generative AI Use Cases

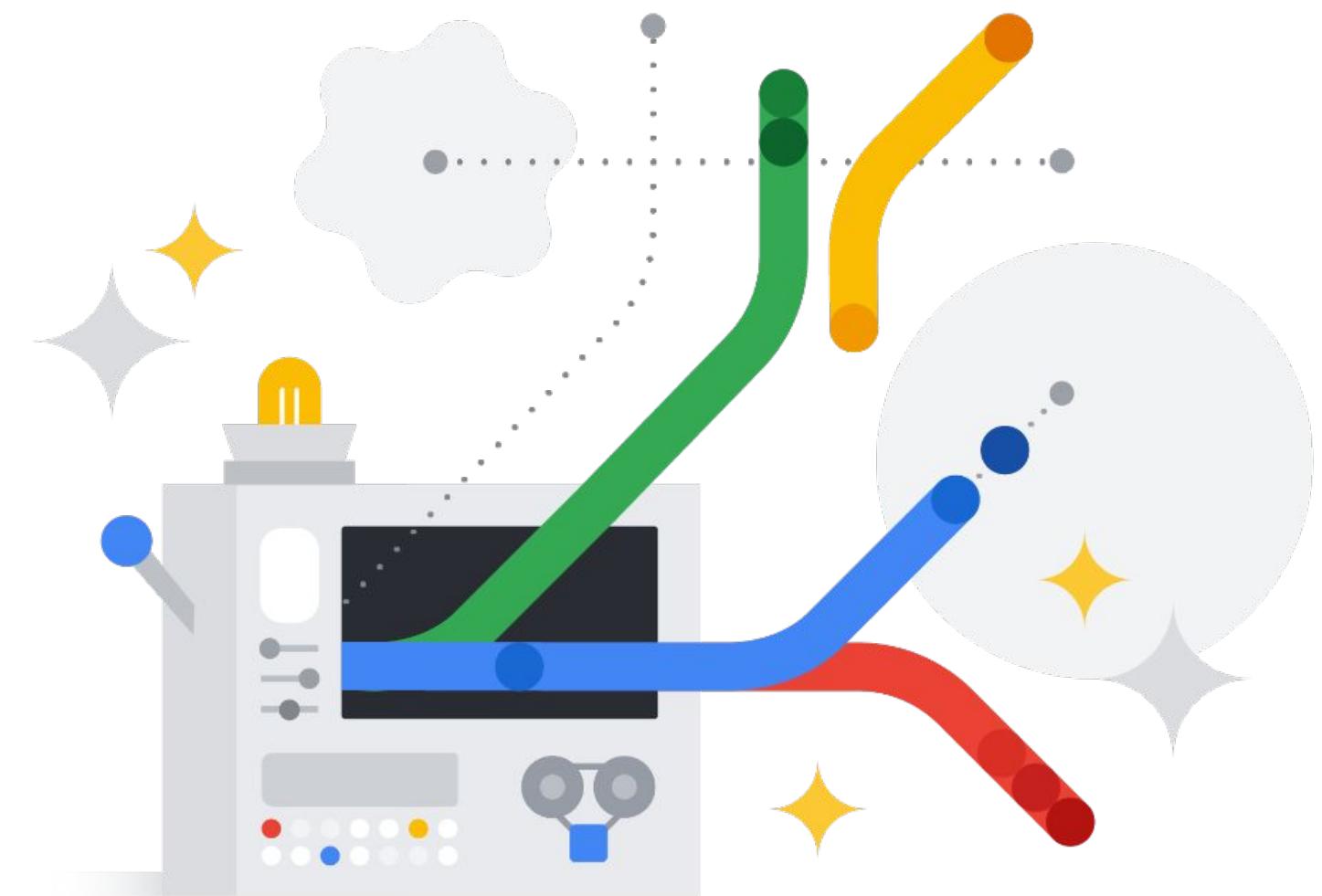
05 More Foundation Models: Model Garden

06 Vertex AI Search



PaLM (Pathways Language Model)

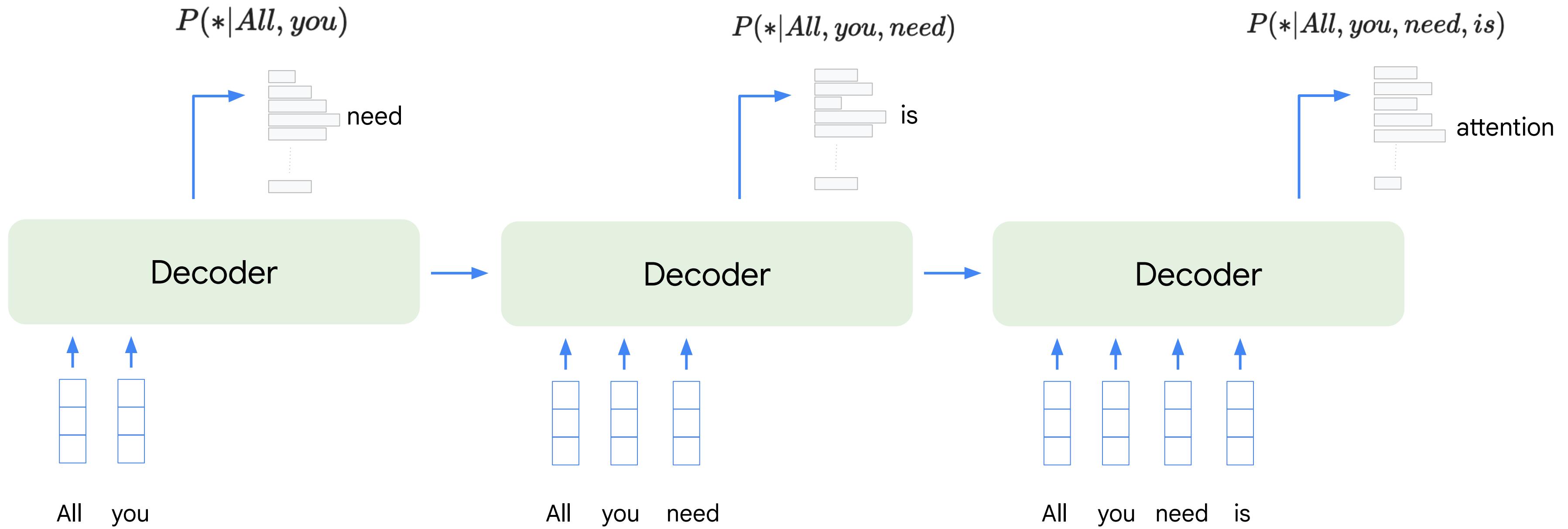
- Google's latest language foundation model
- Trained on a massive dataset of text and code
- Capable of performing a wide range of tasks
 - Text and code generation
 - Reasoning
 - Translation



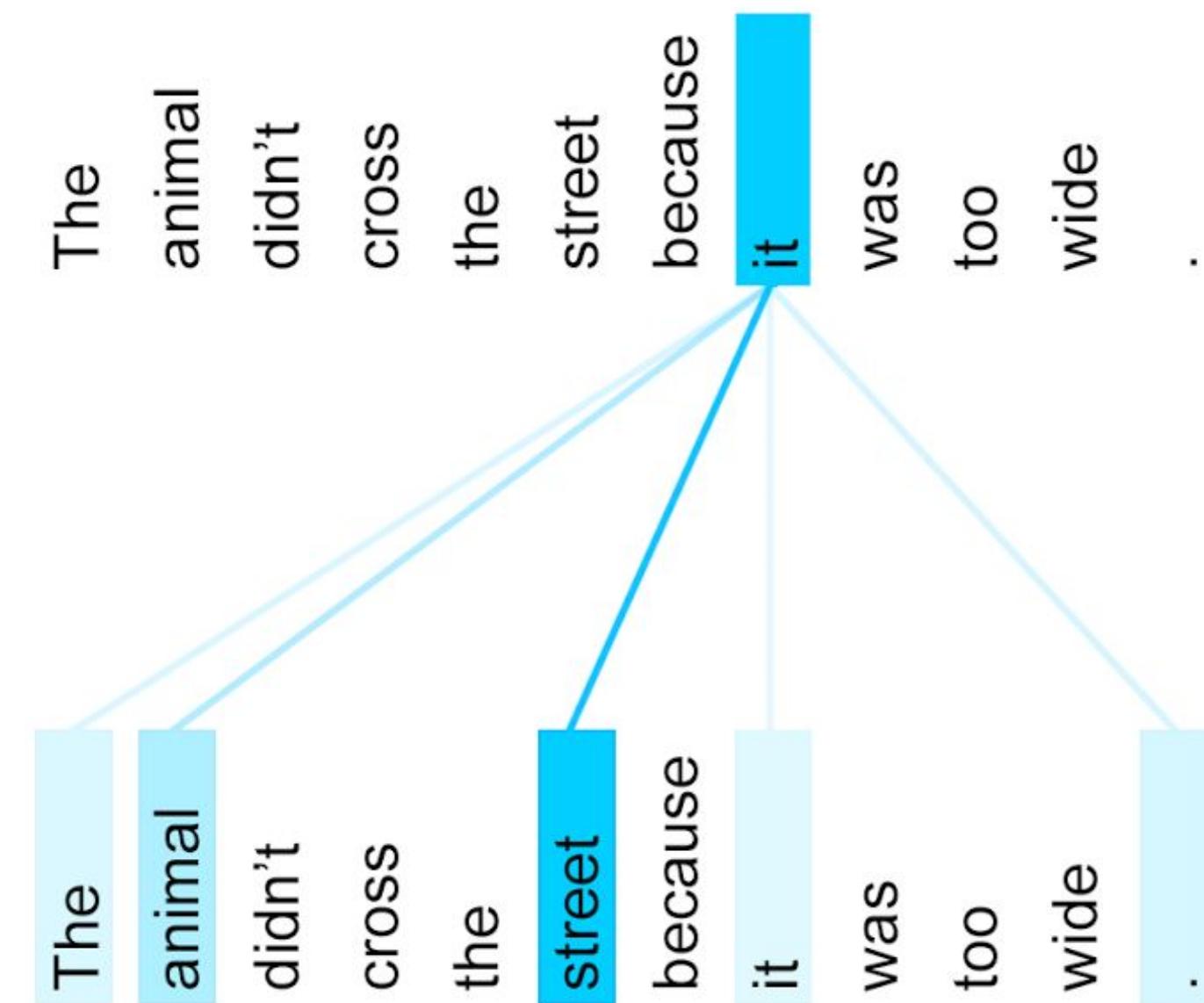
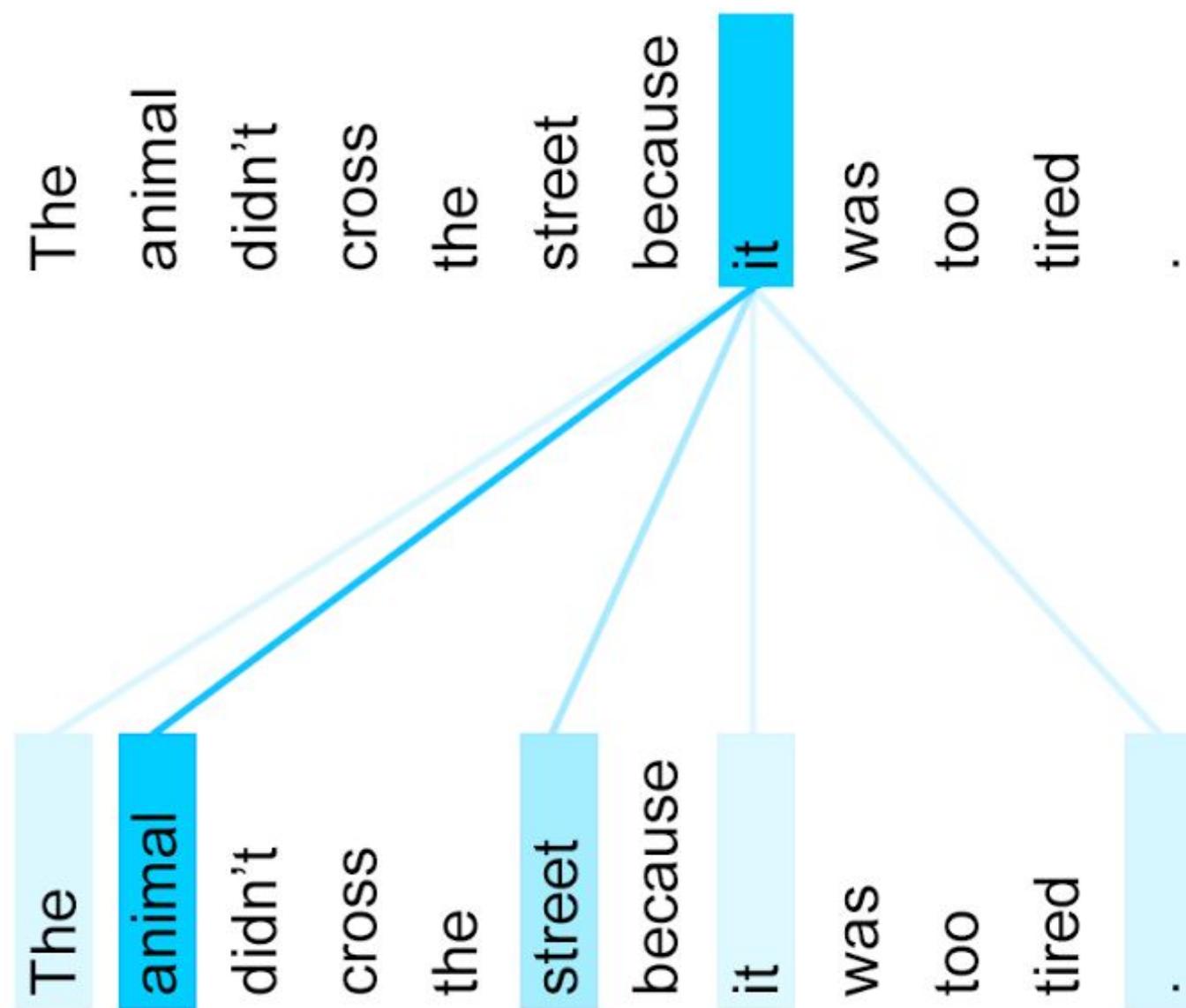
Datasets used to pre-train Google's PaLM model

Dataset	Sampling Proportion
Social media conversations	50%
Filtered web pages	27%
Books	13%
GitHub	5%
Wikipedia	4%
News	1%

PaLM's decoder layer returns a probability distribution



PaLM uses a self-attention layer



PaLM comes in multiple sizes

Gecko

- Small enough to run on a mobile device

Otter

- Medium size model

Bison

- Large model
- Best price - performance value for demanding tasks

Unicorn

- Largest, most advance model
- Computationally, very expensive

PaLM has been fine tuned for various use cases

Text Bison

Chat Bison

Codey

Med PaLM

Sec PaLM

Current model versions

Stable Versions

- Include the version number
- Each version is available for 6 months after subsequent version is released
 - chat-bison@00#
 - code-bison@00#
 - codechat-bison@00#
 - code-gecko@00#
 - text-bison@00#
 - textembedding-gecko@00#
 - text-unicorn@00#

Latest Versions

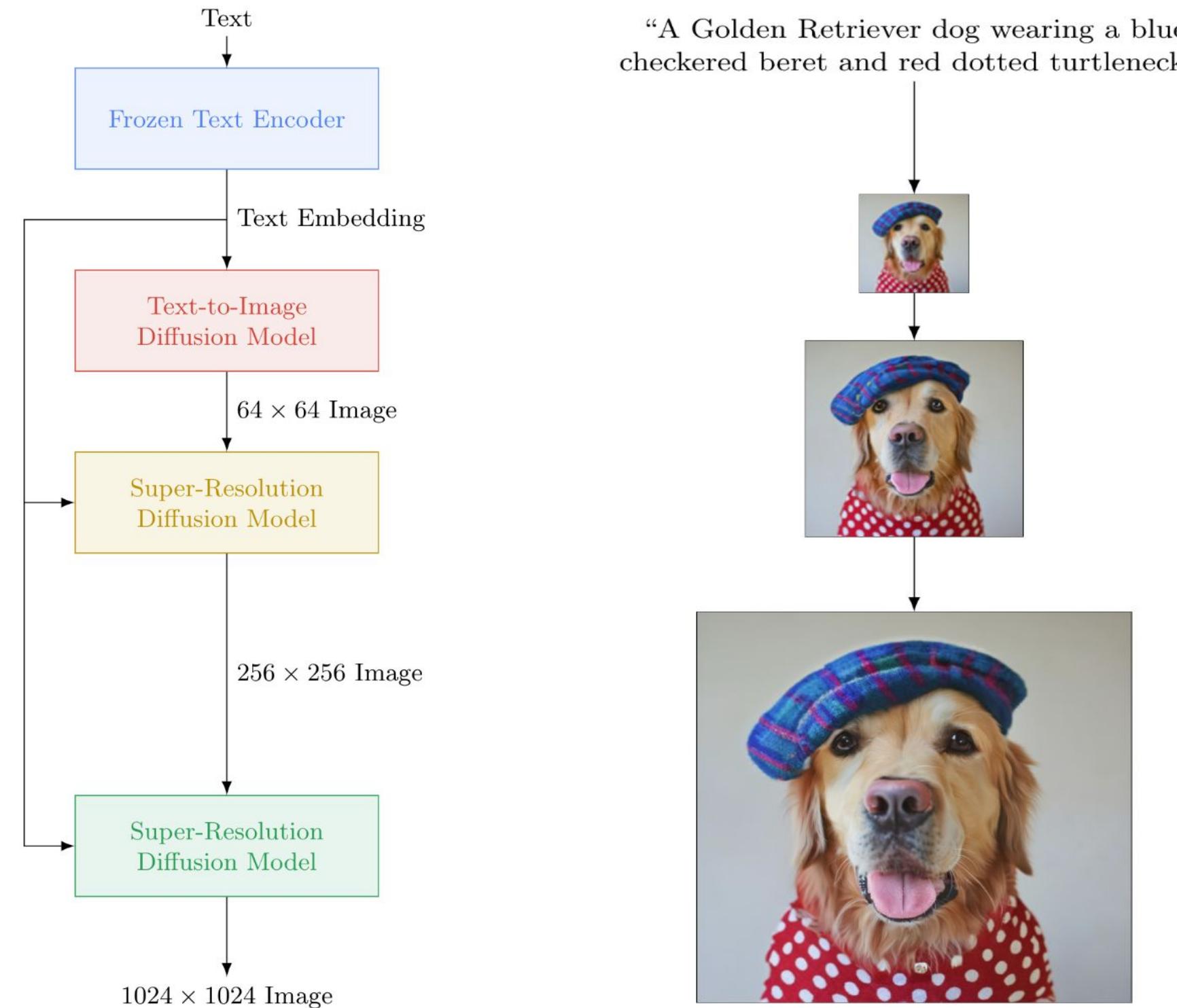
- To use the latest version, provide the name without the version number
 - text-bison, text-bison-32k
 - chat-bison, chat-bison-32k
 - code-bison, code-bison-32k
 - codechat-bison, codechat-bison-32k
 - code-gecko, code-gecko-32k
 - textembedding-gecko
 - textembedding-gecko-multilingual
- 32k versions allow for a larger prompt and response

Imagen is Google's Foundation model for Vision

- The latest version is Imagen 2
- Imagen is capable of performing a wide range of vision-related tasks
 - Generate an image
 - Edit a masked section of an image
 - Caption an image
 - Visual Q&A (Answer questions about an image)
 - The documentation shows a [feature roadmap](#) with more features planned



Imagen uses diffusion-based techniques to generate images

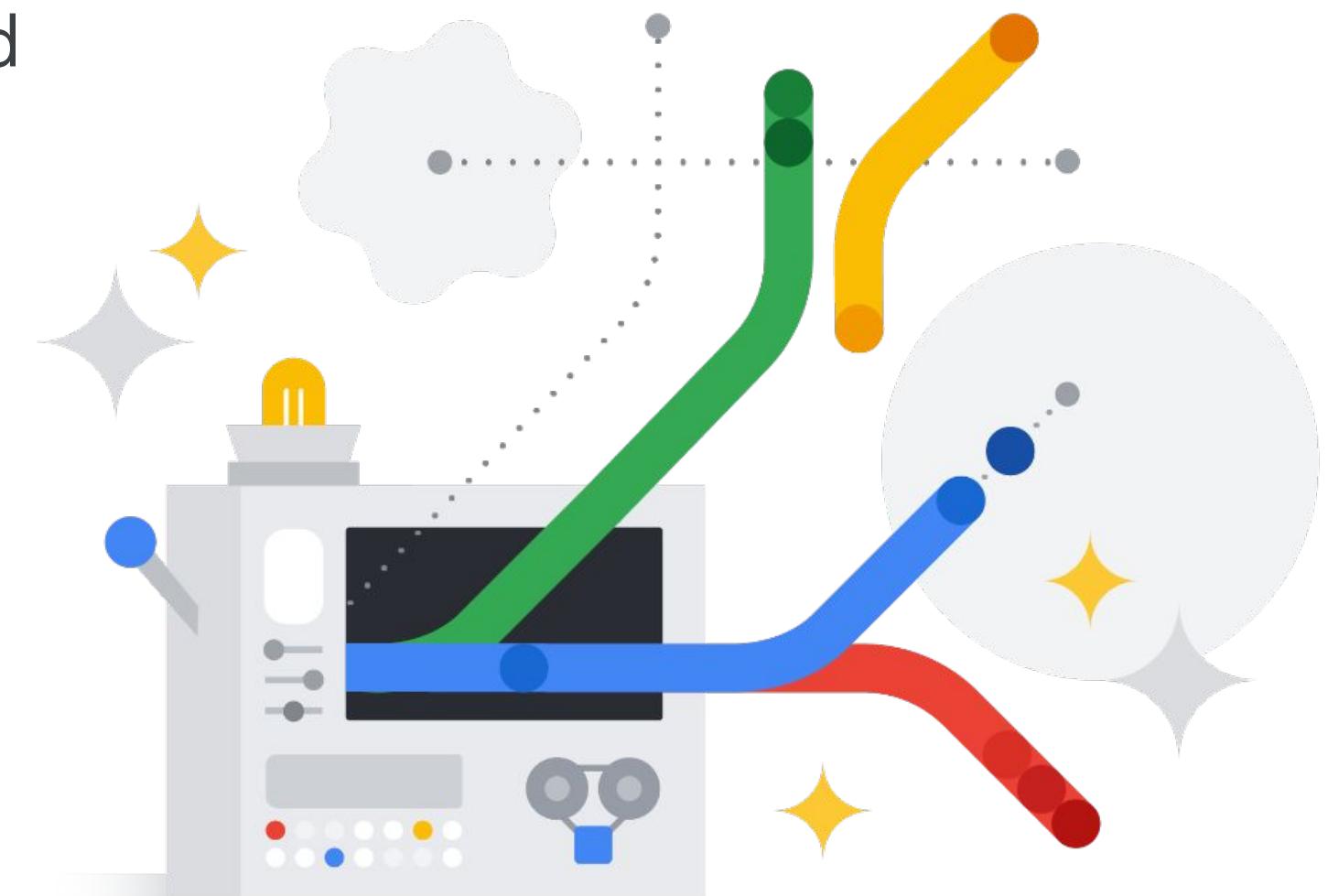


Diffusion models iteratively refine a noise-filled image to approximate the target image distribution.

Gemini is a multimodal foundational model.

It is trained on text, images, video & audio.

- Prompts can contain a combination of text, images, and video
- Capable of performing a wide range of text and vision-related tasks
 - Generate text
 - Extract text from images & video
 - Caption images, video, or audio
 - Understand and respond to questions about video, text & audio



Gemini comes in multiple sizes

Nano

- Most efficient model for on-device tasks

Pro

- Best model for scaling across a wide range of tasks

Ultra

- Largest and most capable model for highly complex tasks

Gemini has been tuned for both language-specific and multimodal use cases

◆ Generative AI Language

Gemini Pro

The best performing Gemini model with features for a wide range of tasks

google/gemini-pro

[VIEW DETAILS](#)

◆ Generative AI Multimodal

Gemini Pro Vision

Created from the ground up to be multimodal (text, images, videos) and to scale across a wide range of tasks

google/gemini-pro-vision

[VIEW DETAILS](#)

Gemini example



Write an exciting and creative story inspired by this image.
Make the dog a talking superhero named Sparks Dino Might

Response

Markdown

Sparks Dino Might is a 3-year-old Great Dane who lives in the Swiss Alps with his owner, a world-renowned scientist. Sparks is a very special dog - he is a super-hero! He inherited his powers from his parents who were both super-heroes. Sparks can fly, he has super strength, and he can shoot lasers from his eyes.

One day, Sparks was playing in the snow when he saw a group of skiers being chased by a pack of wolves. Sparks knew that he had to help, so he flew into action. He used his super strength to knock the wolves away from the skiers, and then he used his lasers to scare them away. The skiers were so grateful to Sparks for saving them, and they all wanted to know who he was. Sparks told them that he was a super-hero, and that he was there to help people.

The skiers were so impressed with Sparks that they decided to name him their honorary mayor. Sparks was so happy to be able to help the skiers, and he was even happier to be named honorary mayor. He knew that he would use his powers to help people all over the world.

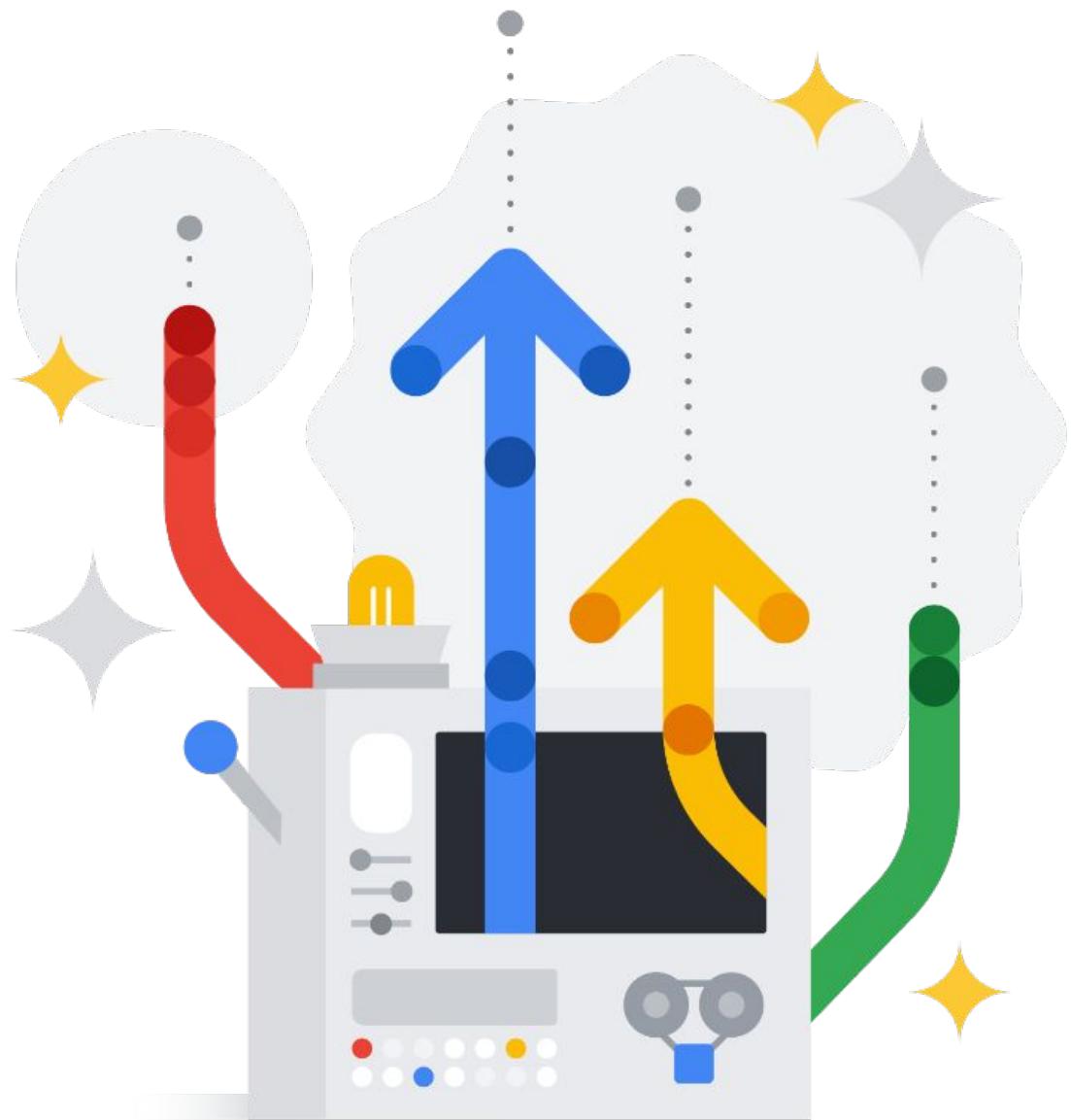
Topics

- 01 The Benefits of Foundation Models
- 02 Google's Foundation Models
- 03 Vertex AI Studio
- 04 Generative AI Use Cases
- 05 More Foundation Models: Model Garden
- 06 Vertex AI Search



Vertex AI Studio with Google foundation models

- Supports Google foundational models
 - Gemini Pro for text generation
 - Gemini Pro Vision for Image and Video Q&A
 - PaLM 2 for text and chat
 - Codey for code generation and chat
 - Chirp for speech to text
 - Imagen for text to image generation
- Allows users to easily experiment with prompts
 - Simple, intuitive design
 - Easily experiment with parameters
 - Add context and examples

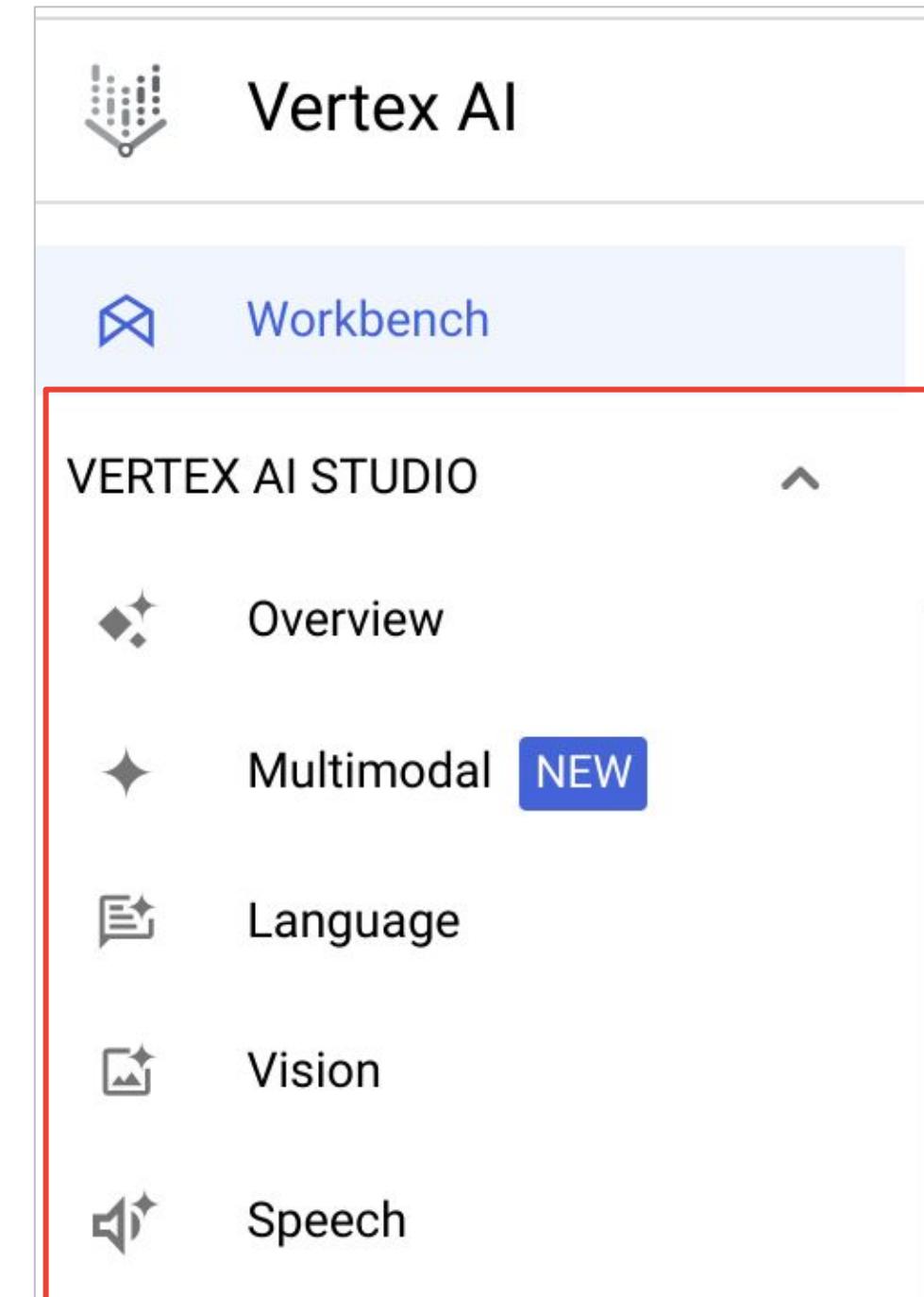


Generative AI use cases with Vertex AI Studio

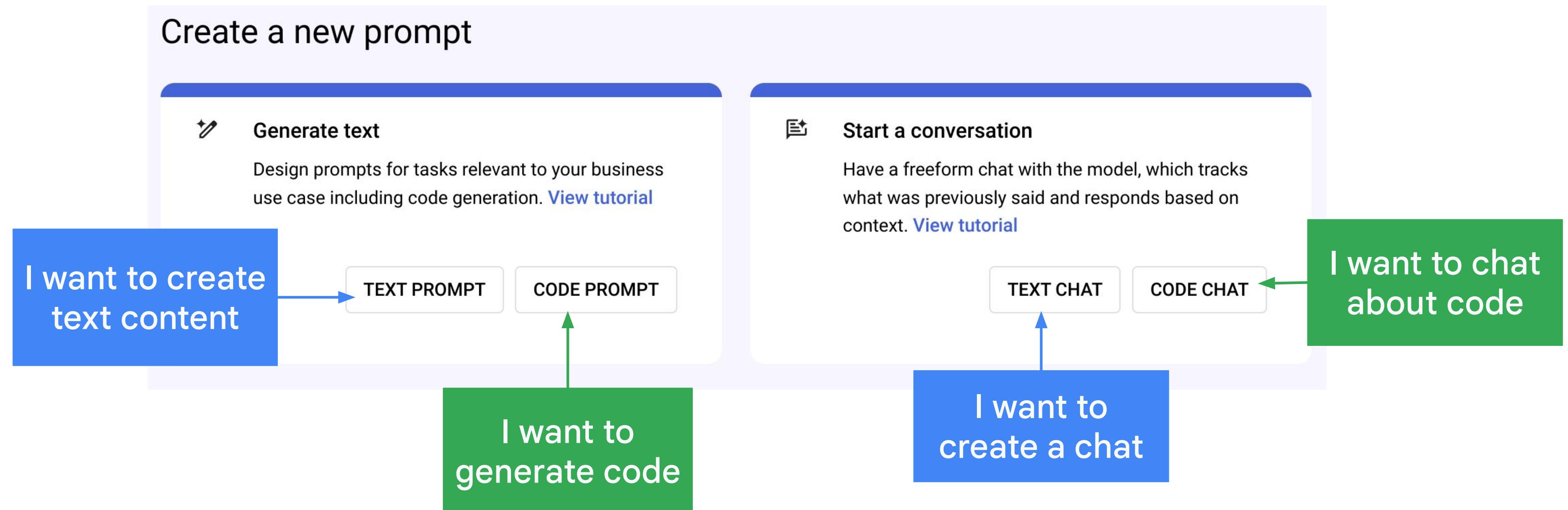
Language	Code	Speech	Vision
<ul style="list-style-type: none">• Writing• Summarization• Ideation• Classification• Sentiment analysis• Extraction• Chat• Search	<ul style="list-style-type: none">• Code generation• Code completion• Code chat• Code conversion	<ul style="list-style-type: none">• Speech to text• Text to speech	<ul style="list-style-type: none">• Image generation• Image editing• Captioning• Image Q&A• Image search• Video descriptions

Vertex AI Studio

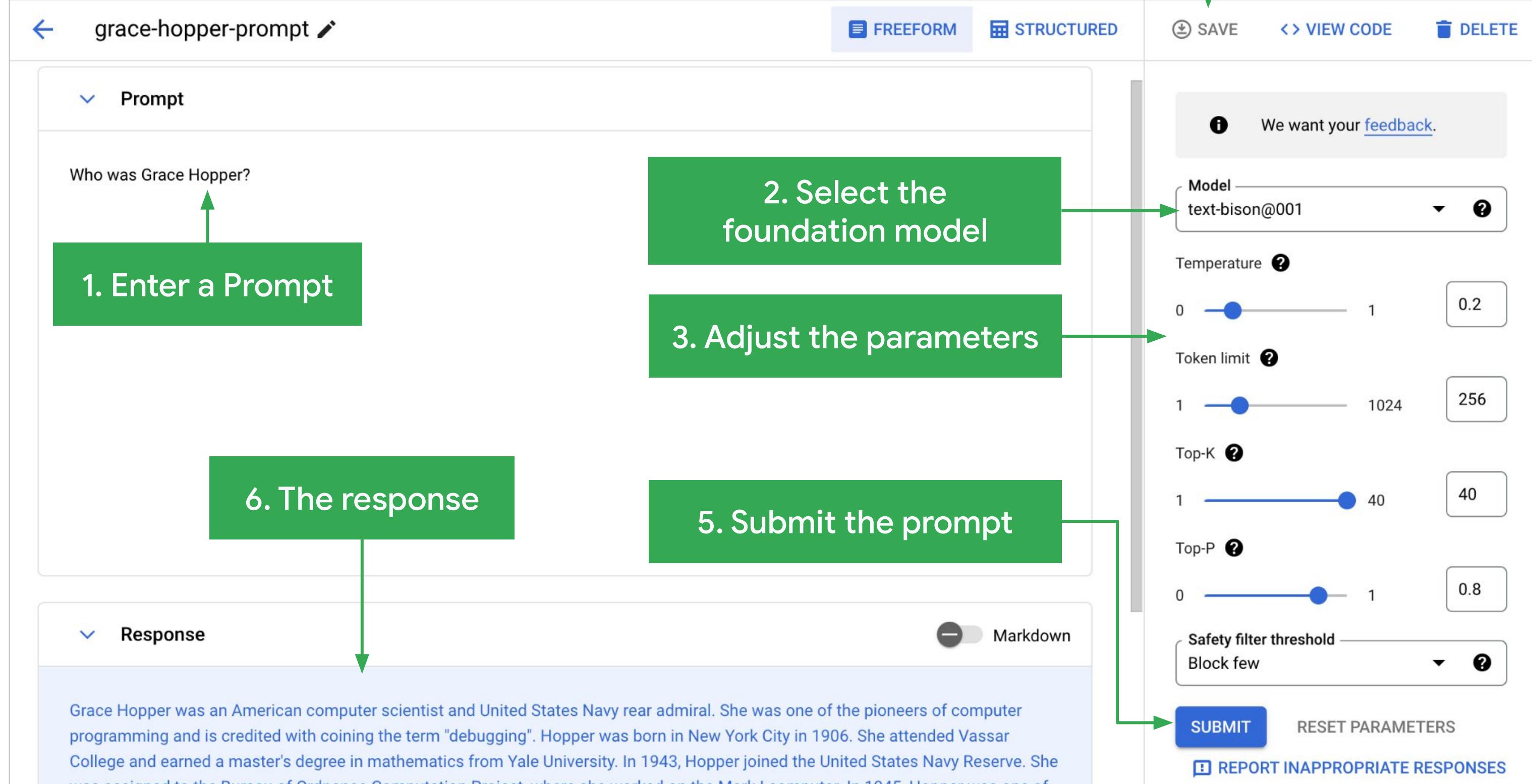
- Vertex AI Studio is available as a feature of Google Cloud Vertex AI
- Choose from Language, Vision, or Speech models



To get started, choose a task based on your goal



Vertex AI Studio UI



Test a simple text prompt to see the results

Prompt

Tell me about Grace Hopper



Response

Grace Hopper was an American computer scientist and United States Navy rear admiral. She was one of the pioneers of computer programming, and is credited with coining the term "debugging". She also helped develop the first compiler, which translates human-readable code into machine code that computers can understand.

Hopper was born in New York City in 1906. She attended Vassar College, where she studied mathematics and physics. After graduating, she taught mathematics at Vassar for a year before joining the United States Navy in 1943.

During World War II, Hopper worked on the Mark I computer, one of the first electronic computers. She was responsible for developing the first compiler, which translated human-readable code into machine code that computers could understand. This work made it much easier for programmers to write code for computers, and helped to make computers more accessible to a wider range of people.

After the war, Hopper continued her work in computer science. She helped develop the COBOL programming language, which is still widely used today. She also served as the director of the Navy's computer science research center.

In 1986, Hopper was awarded the National Medal of Technology for her contributions to computer science

Freeform versus Structured prompts in Vertex AI Studio

- Structured prompts in Vertex AI Studio guide you towards implementing best practices of providing the model context and examples
 - Context tells the model more about your use case characteristics
 - Examples show the model how to structure the response
- You can provide the model the same info in a freeform prompt

Vertex AI Studio UI for structured prompts

The screenshot shows the Vertex AI Studio interface for creating structured prompts. The top navigation bar includes a back arrow, the project name "bbq-examples", and tabs for "FREEFORM" and "STRUCTURED". A green callout box labeled "Select Structured" points to the "STRUCTURED" tab. The main workspace is divided into three sections: "Context", "Examples", and "Test".

- Context:** A green callout box labeled "Context adds information about the use case" points to the "Context" section, which contains a dropdown and a descriptive text about a chef specializing in Southern Barbecue.
- Examples:** A green callout box labeled "Examples demonstrate desired output" points to the "Examples" section. It shows an input row "Write a post about our latest Smoked Brisket video." and an output row "Hey y'all, get ready for a flavor-packed ride down South with my latest YouTube video: Smoked Brisket Low and Slow!".
- Test:** A green callout box labeled "Prompt and response" points to the "Test" section. It shows an input row "Write a post about our new video about smoking Prime Rib for the holidays." and an output row "Y'all ready for some mouth-watering Prime Rib? It's the perfect centerpiece for your holiday feast, and it's so easy".

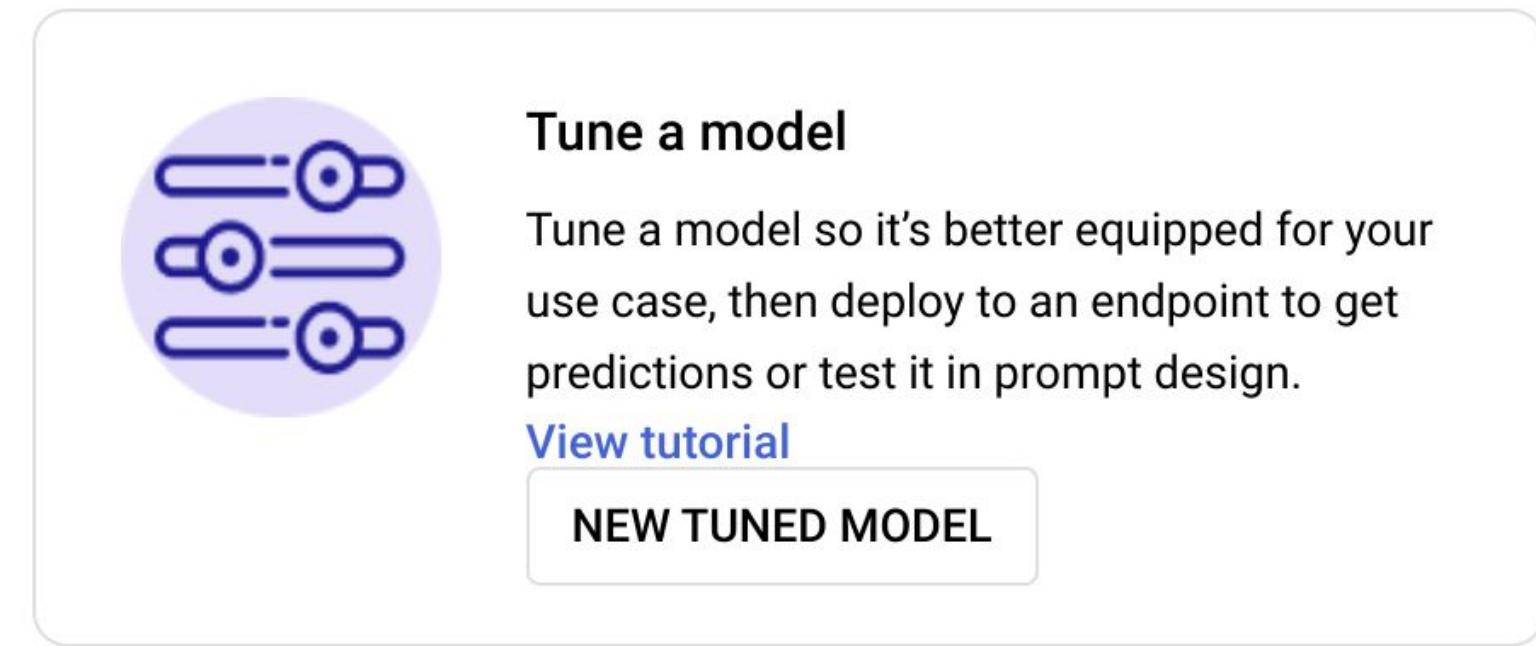
The right sidebar contains configuration options for the prompt, including:

- Model:** text-bison@001
- Temperature:** 0.5
- Token limit:** 256
- Top-K:** 40
- Top-P:** 0.8
- Safety filter threshold:** Block few
- Buttons:** SUBMIT, RESET PARAMETERS, REPORT INAPPROPRIATE RESPONSES

A note at the bottom right states: "Content processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses [but are not visible in the UI](#)".

Use Vertex AI Studio to tune foundational models

- You supply your own training data in JSON-L format
- The foundational model is customized for your particular use case
 - An adapter layer is added on top of the Foundational model
 - The adapter layers adds new parameters to the foundational model
- The training job runs on Google infrastructure
- Your model and data are stored separately in your own project



You can create a library of use cases

Conversational interactions

Document
Search &
Synthesis

Product /
Content Catalog
Discovery

Regulatory
Compliance
Automation

Public Website
Navigation

Internal
Knowledge
Q & A

Documentation
Generation

Business
Process
Automation

Product / Content
Recommendation

Research
Acceleration

Customer
Service
Automation

Creative
Assistance

Developer
Efficiency

Complex data, intuitively accessible

Content generation

Topics

01 The Benefits of Foundation Models

02 Google's Foundation Models

03 Vertex AI Studio

04 Generative AI Use Cases

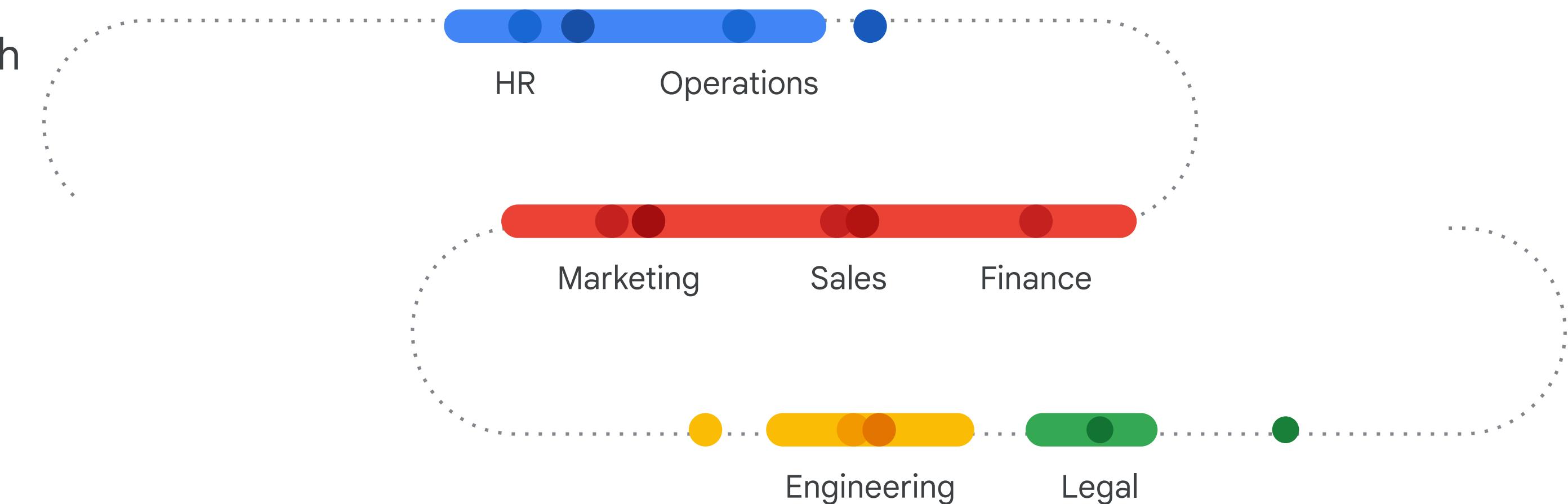
05 More Foundation Models: Model Garden

06 Vertex AI Search



There are many potential use-cases for Generative AI across all aspects of an organization

- Content creation
- Marketing and advertising
- Customer service
- Education and research
- Many more...



Content generation

▼ **Prompt**

Tell me about Grace Hopper



▼ **Response**

Markdown

Grace Hopper was an American computer scientist and United States Navy rear admiral. She was one of the pioneers of computer programming, and is credited with coining the term "debugging". She also helped develop the first compiler, which translates human-readable code into machine code that computers can understand.

Hopper was born in New York City in 1906. She attended Vassar College, where she studied mathematics and physics. After graduating, she taught mathematics at Vassar for a year before joining the United States Navy in 1943.

During World War II, Hopper worked on the Mark I computer, one of the first electronic computers. She was responsible for developing the first compiler, which translated human-readable code into machine code that computers could understand. This work made it much easier for programmers to write code for computers, and helped to make computers more accessible to a wider range of people.

After the war, Hopper continued her work in computer science. She helped develop the COBOL programming language, which is still widely used today. She also served as the director of the Navy's computer science research center.

In 1986, Hopper was awarded the National Medal of Technology for her contributions to computer science

Summarization

Prompt

Summarize the following Wikipedia article about Grace Hopper:

Grace Brewster Hopper (née Murray; December 9, 1906 – January 1, 1992) was an American computer scientist, mathematician, and United States Navy rear admiral. She was one of the first programmers of the Harvard Mark I computer, and she was a pioneer in the development of computer programming who invented one of the first linkers. Hopper was the first to devise the theory of machine-independent programming languages. The FLOW-MATIC programming language she created using this theory was extended to create COBOL, an early high-level programming language used today.

Prior to joining the Navy, Hopper earned a Ph.D. in both mathematics and mathematical physics from Yale University and was a professor of mathematics at Vassar College. Hopper attempted to enlist in the Navy during World War II but was rejected due to her age.

Response

Markdown

Grace Hopper was an American computer scientist and United States Navy rear admiral. She was one of the first programmers of the Harvard Mark I computer, and she was a pioneer in the development of computer programming languages. She is credited with coining the term "debugging".

Hopper was born in New York City in 1906. She earned a Ph.D. in mathematics from Yale University in 1934. She began her career as a professor of mathematics at Vassar College, but she left in 1943 to join the United States Navy Reserve.

Hopper served on the Mark I computer programming staff headed by Howard H. Aiken. She worked on the development of the UNIVAC I computer, and she was a key figure in the development of the COBOL programming language.

Hopper retired from the Navy in 1986, but she continued to work as a consultant for Digital Equipment Corporation. She died in 1992 at the age of 85.

Hopper was a pioneer in the field of computer science, and she made significant contributions to the

Extraction

▼ **Prompt**

Extract the important dates from the following Wikipedia article about Grace Hopper

Grace Brewster Hopper (née Murray; December 9, 1906 – American computer scientist, mathematician, and United [1] One of the first programmers of the Harvard Mark I computer programming who invented one of the first high-level programming languages, FLOW-MATIC, which was later extended to create COBOL, an early high-level programming language used today.

Prior to joining the Navy, Hopper earned a Ph.D. in both mathematics and mathematical physics from Yale University and was a professor at Vassar College.

▼ Response

Important dates in Grace Hopper's life

- * December 9, 1906: Grace Brewster Murray is born in New York City.
 - * 1928: Graduates from Vassar College with a bachelor's degree in mathematics and physics.
 - * 1930: Earns a master's degree from Yale University.
 - * 1934: Earns a Ph.D. in mathematics from Yale University.
 - * 1943: Joins the United States Navy Reserve.
 - * 1944: Serves on the Mark I computer programming staff headed by Howard H. Aiken.
 - * 1949: Joins the Eckert–Mauchly Computer Corporation as a senior mathematician.
 - * 1952: Develops the first compiler, which is written for the A-0 System.
 - * 1954: Named the company's first director of automatic programming.
 - * 1966: Retires from the Naval Reserve with the rank of commander.
 - * 1967: Recalled to active duty for a six-month period that turns into an indefinite assignment

Q&A

▼ Prompt

Document: Grace Brewster Hopper (née Murray; December 9, 1906 – January 1, 1992) was an American computer scientist, mathematician, and United States Navy rear admiral.[1] One of the first programmers of the Harvard Mark I computer, she was a pioneer of computer programming who invented one of the first linkers. Hopper was the first to devise the theory of machine-independent programming languages, and the FLOW-MATIC programming language she created using this theory was later extended to create COBOL, an early high-level programming language still in use today.

Question: How old was Grace Hopper when she died?
Answer:

▼ Response

85 years old

Ideation

▼ Prompt

I wrote an article about Grace Hopper. What are some good titles?



▼ Response

- Grace Hopper: A Pioneer in Computer Science
- Grace Hopper: The Woman Who Invented COBOL
- Grace Hopper: A Life of Ingenuity and Inspiration
- Grace Hopper: A True Renaissance Woman
- Grace Hopper: A Legacy of Leadership and Innovation

Classification

▼ Prompt

Is the following Spam or Not Spam:

Sign up today for the Grace Hopper fan club. For the bargain price of \$9.99 per month you will get new articles every day. Plus, you will be entered into our drawing for a one-of-a-kind Grace Hopper Tee Shirt.



▼ Response

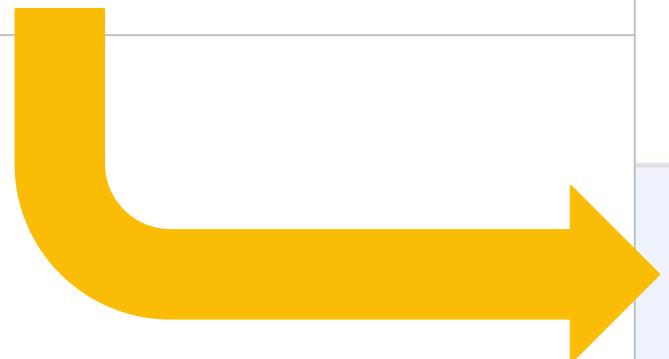
Spam

Sentiment analysis

▼ Prompt

Is the following comment positive or negative:

Your summary of Grace Hopper was interesting and informative. I had no idea how important she was in the the field of computer science!



▼ Response

positive

Language translation

Prompt

Document: Grace Brewster Hopper (née Murray; December 9, 1906 – January 1, 1992) was an American computer scientist, mathematician, and United States Navy rear admiral.[1] One of the first programmers of the Harvard Mark I computer, she was a pioneer of computer programming who invented one of the first linkers. Hopper was the first to devise the theory of machine-independent programming languages, and the FL language she created using this theory was later an early high-level programming language still in

Translate the document above into French

Response



Markdown

Grace Brewster Hopper (née Murray ; 9 décembre 1906 - 1er janvier 1992) était une informaticienne, mathématicienne et contre-amirale de la marine américaine. L'une des premières programmeuses de l'ordinateur Mark I de Harvard, elle fut une pionnière de la programmation informatique qui inventa l'un des premiers éditeurs de liens. Hopper fut la première à concevoir la théorie des langages de programmation indépendants de la machine. Le langage de programmation FLOW-MATIC qu'elle créa à l'aide de cette théorie fut par la suite développé pour créer le COBOL, l'un des premiers langages de programmation de haut niveau encore utilisé aujourd'hui.

Code Generation

Prompt

Write the Terraform code to create a Google Cloud VPC with Subnets in us-central1 and us-east4. Create variables for the IP address ranges.



Response

```
provider "google" {
  project = var.project_id
  region = var.region
}

resource "google_compute_network" "default" {
  name = var.network_name
}

resource "google_compute_subnetwork" "us-central1" {
  name = var.subnetwork_name
  network = google_compute_network.default.name
  region = var.region
  ip_range = var.ip_range_us_central1
}

resource "google_compute_subnetwork" "us-east4" {
  name = var.subnetwork_name
  network = google_compute_network.default.name
  region = var.region
  ip_range = var.ip_range_us_east4
}
```

Chat

USER Write me the code to create a Google Cloud Storage bucket

AI To create a Google Cloud Storage bucket, you can use the following code:

```
import google.cloud.storage as gcs

# Create a client
client = gcs.Client()

# Create a bucket
bucket = client.create_bucket('my_bucket')
```

This code will create a bucket named `my_bucket` in your Google Storage account. You can then use this bucket to store data.

USER How about using a shell script

AI Sure, here is the code for creating a Google Cloud Storage bucket using a shell script:

```
#!/bin/bash

# Create the bucket
gsutil mb gs://my_bucket
```

This code will create a bucket named `my_bucket` in your Google Cloud Storage account. You can then use this bucket to store data.

Image Q&A



Answer the following questions about the picture:

How many people are in the picture?

What are the people doing?

What city are they in?

What landmark in the picture makes you think it is that city?

Return the results in JSON format.

Include both English and French answers.

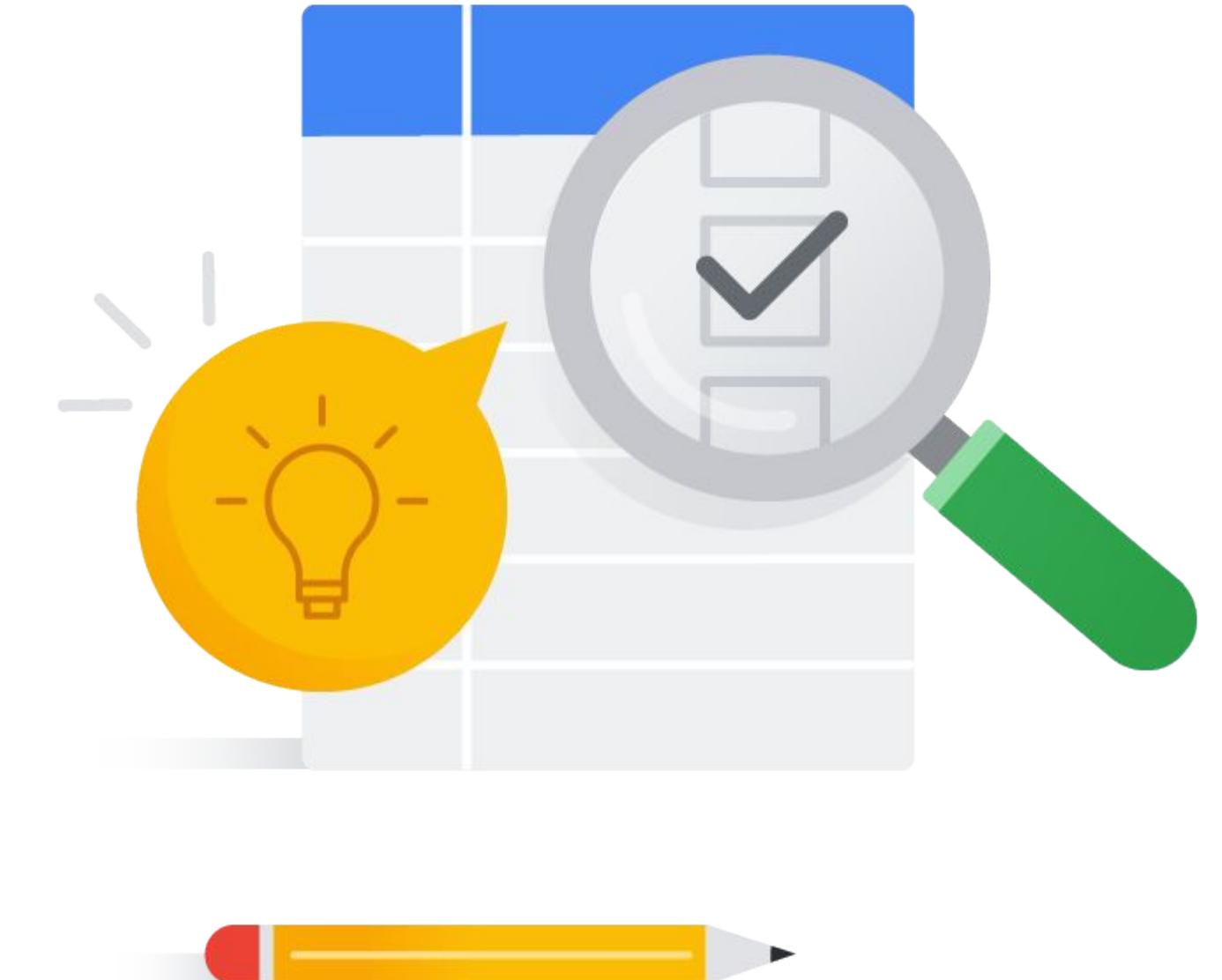
Response

```
```json
{
 "en": {
 "people_count": 2,
 "action": "drinking beer",
 "city": "New York City",
 "landmark": "yellow taxi"
 },
 "fr": {
 "people_count": 2,
 "action": "boire de la biere",
 "city": "New York City",
 "landmark": "taxi jaune"
 }
}
````
```

Lab

⌚ 30 min ⚙️

Lab: Generative AI with Vertex AI: Prompt Design



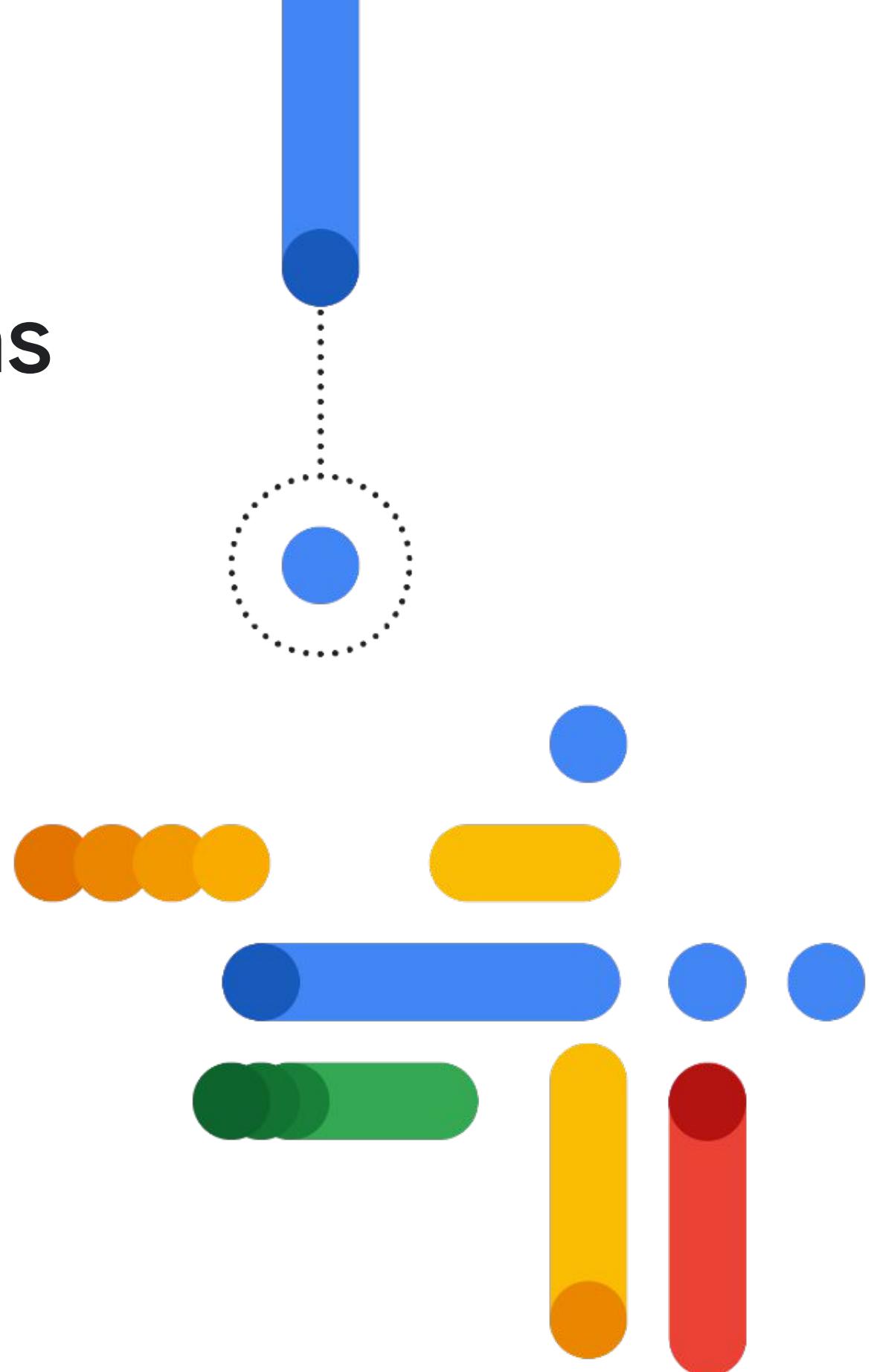
Topics

- 01 The Benefits of Foundation Models
- 02 Google's Foundation Models
- 03 Vertex AI Studio
- 04 Generative AI Use Cases
- 05 More Foundation Models: Model Garden
- 06 Vertex AI Search



Model Garden provides a catalog of pre-trained models to build AI applications

- Foundation models are pre-trained multitask large models that can be tuned or customized for specific tasks
- Fine-tunable models are models that you can fine-tune using a custom notebook or pipeline
- Task-specific solutions are pre-built models that are ready to use and can be customized using your own data
- Model garden provides and easy to use interface for searching for models
 - Documentation and code samples are provided



Navigating Model Garden

Filter models by modality, task, or features

Search for models

Model cards provide access to documentation, code samples, etc.

The screenshot shows the Google Model Garden interface. On the left, there's a sidebar with sections for 'Modalities' and 'Tasks'. The 'Modalities' section lists Language (34), Vision (75), Tabular (2), Documents (2), Speech (1), and Video (3). The 'Tasks' section lists Generation (36), Classification (40), and Detection (27). At the top right, there are buttons for 'EXPLORE GENERATIVE AI' and 'VIEW MY MODELS'. Below them is a search bar with the placeholder 'Search models'. A blue arrow points from the 'Filter models...' callout to the sidebar. Another blue arrow points from the 'Search for models' callout to the search bar. A third blue arrow points from the 'Model cards provide...' callout to the 'VIEW DETAILS' button for the Llama 2 model card.

| Modality | Count |
|-----------|-------|
| Language | 34 |
| Vision | 75 |
| Tabular | 2 |
| Documents | 2 |
| Speech | 1 |
| Video | 3 |

| Task | Count |
|----------------|-------|
| Generation | 36 |
| Classification | 40 |
| Detection | 27 |

Foundation models
Pre-trained multi-task models that can be further tuned or customized for specific tasks. Models marked with ♦ are available in [Generative AI Studio](#).

PaLM 2 for Text
Fine-tuned to follow natural language instructions and is suitable for a variety of language tasks, such as: classification, extraction, summarization and content...
[text-bison](#)

Llama 2
Fine-tune & deploy Meta's Llama 2 models on Vertex AI.

Llama2-7B-001
[VIEW DETAILS](#)

Model Cards provide information on programming, training, and deploying models

The screenshot shows a Model Card for the "Stable Diffusion v2.1" model. At the top left is a back arrow and the text "Stable Diffusion v2.1". The main title "Stable Diffusion v2.1" is bold and centered. Below it is a description: "Latent text-to-image diffusion model taking as input a text prompt, and generates an image." Two buttons are present: a blue "DEPLOY" button and a white "OPEN NOTEBOOK" button with a blue icon. Below these are four navigation links: "OVERVIEW" (underlined), "USE CASES", "DOCUMENTATION", and "PRICING". The "OVERVIEW" section contains an "Overview" heading and a detailed paragraph about the model's history and performance.

← Stable Diffusion v2.1

Stable Diffusion v2.1

Latent text-to-image diffusion model taking as input a text prompt, and generates an image.

[DEPLOY](#) [OPEN NOTEBOOK](#)

[OVERVIEW](#) [USE CASES](#) [DOCUMENTATION](#) [PRICING](#)

Overview

The Stable Diffusion model was first published in the paper "[High-Resolution Image Synthesis with Latent Diffusion Models](#)" by Rombach et al in 2022. To enable diffusion models training on limited computational resources while retaining their quality and flexibility, the authors applied them in the latent space of powerful pretrained autoencoders. Training diffusion models on such a representation allows for the first time to reach a near-optimal point between complexity reduction and detail preservation, greatly boosting visual fidelity. The latent diffusion models achieved a new state of the art for image inpainting and highly competitive performance on various tasks, including unconditional image generation, semantic scene synthesis, and super-resolution, while significantly reducing computational requirements compared to pixel-based diffusion models.

Model Garden includes Google, 3rd-party and open source models

| Model Type | Model Description | Model Details |
|-------------|---|---|
| First Party | <p>Foundation models</p> <p>Leverage Google's multimodal models across vision, dialog, code generation/completion</p> | <ul style="list-style-type: none">• PaLM for Text PaLM for Chat• Imagen for text-to-image• Codey for code completion• Chirp for speech-to-text |
| | <p>Pre-trained APIs</p> <p>Build and deploy AI applications faster with our pre-trained APIs powered by the best Google AI research and technology</p> | <ul style="list-style-type: none">• Speech-to-Text• Natural Language Processing• Translation• Vision |
| Open Source | <p>Open Source</p> <p>Access a wide variety of enterprise-ready open source models</p> | <ul style="list-style-type: none">• StableDiffusion• ViT• EfficientNet |
| 3rd Party | <p>3rd Party</p> <p>Over time Model Garden will support 3rd-party models from partners with foundation models</p> | <ul style="list-style-type: none">• Anthropic's Claude 2 |

Topics

- 01 The Benefits of Foundation Models
- 02 Google's Foundation Models
- 03 Vertex AI Studio
- 04 Generative AI Use Cases
- 05 More Foundation Models: Model Garden
- 06 Vertex AI Search



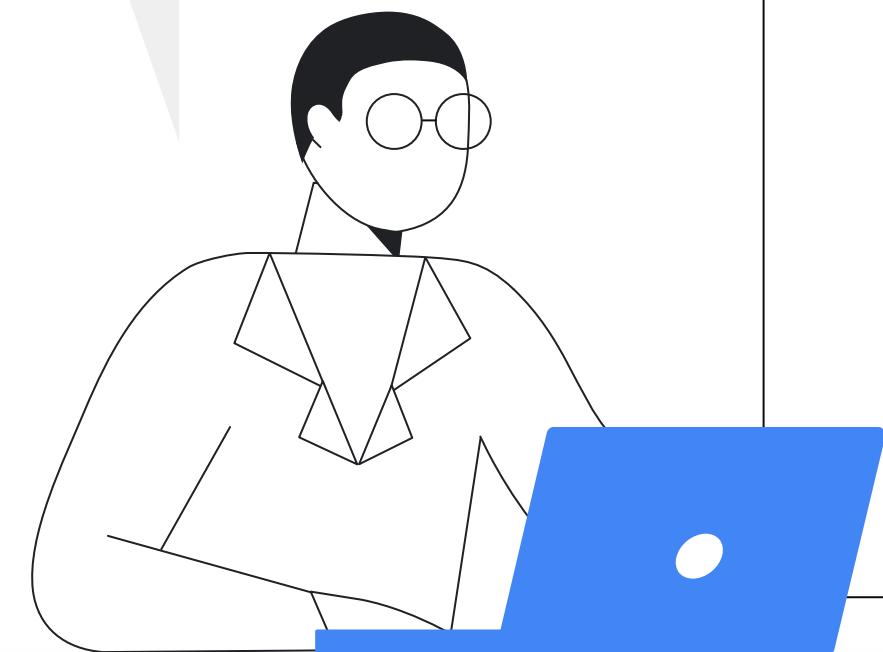
GenAI-enabled search drives faster and better insights

Query and extract insights from public datasets (medical literature) and private datasets (ELN). Quickly and accurately source insights from public and private datasets.

Summarize research into plain language

Example: Searching FDA website and public websites to search for secondary indication for existing pharmaceutical products

“Show me all FDA submissions related to Cymbal drug”



Here is a list of FDA submissions for drugs similar to Cymbal that have been filed for secondary indications:

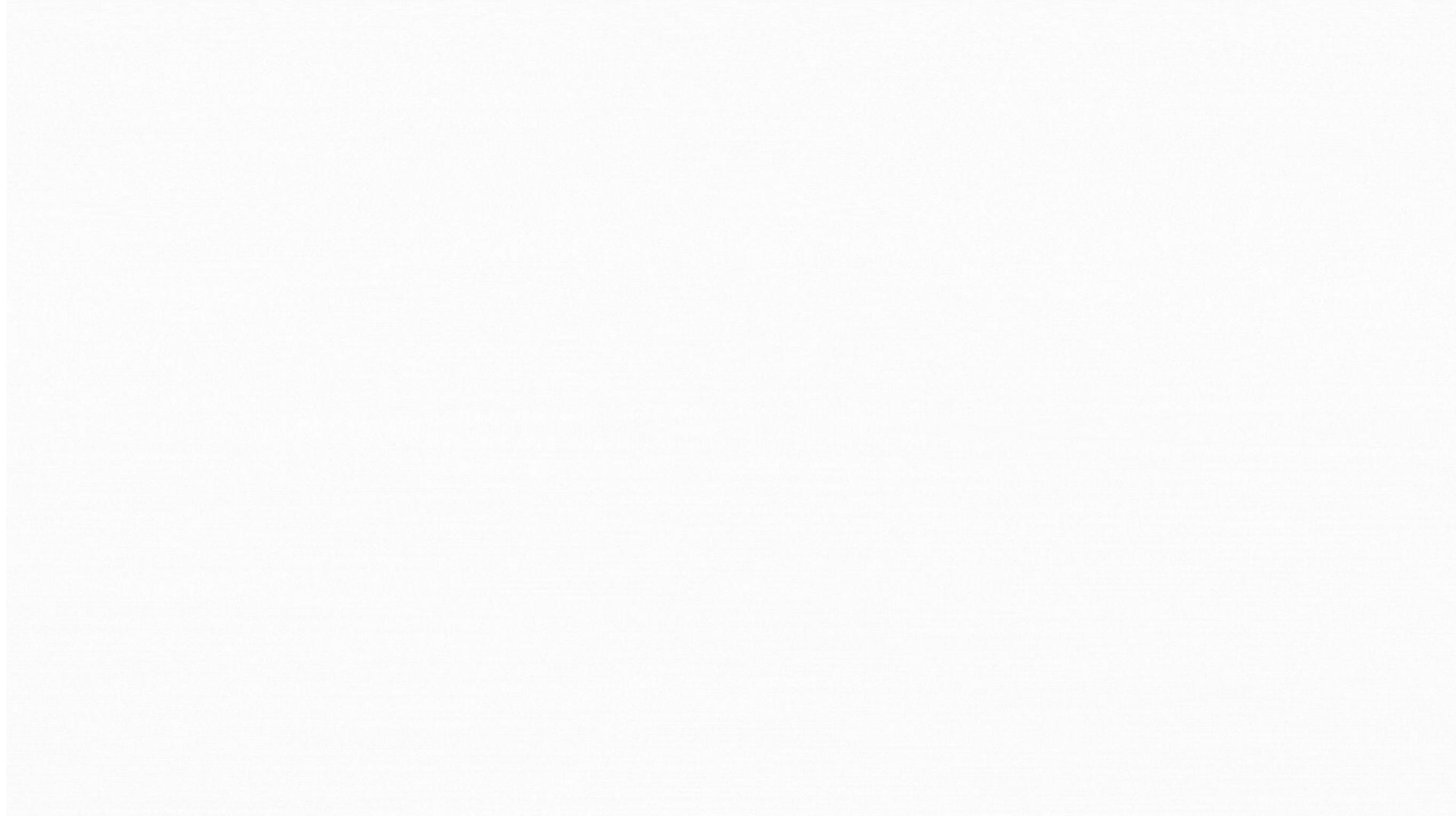
Drug A: Secondary indication for the treatment of pain

Drug B: Secondary indication for the treatment of nausea

Drug C: Secondary indication for the treatment of vomiting



Vertex AI Search allows organizations to index and search their own datasets using natural language



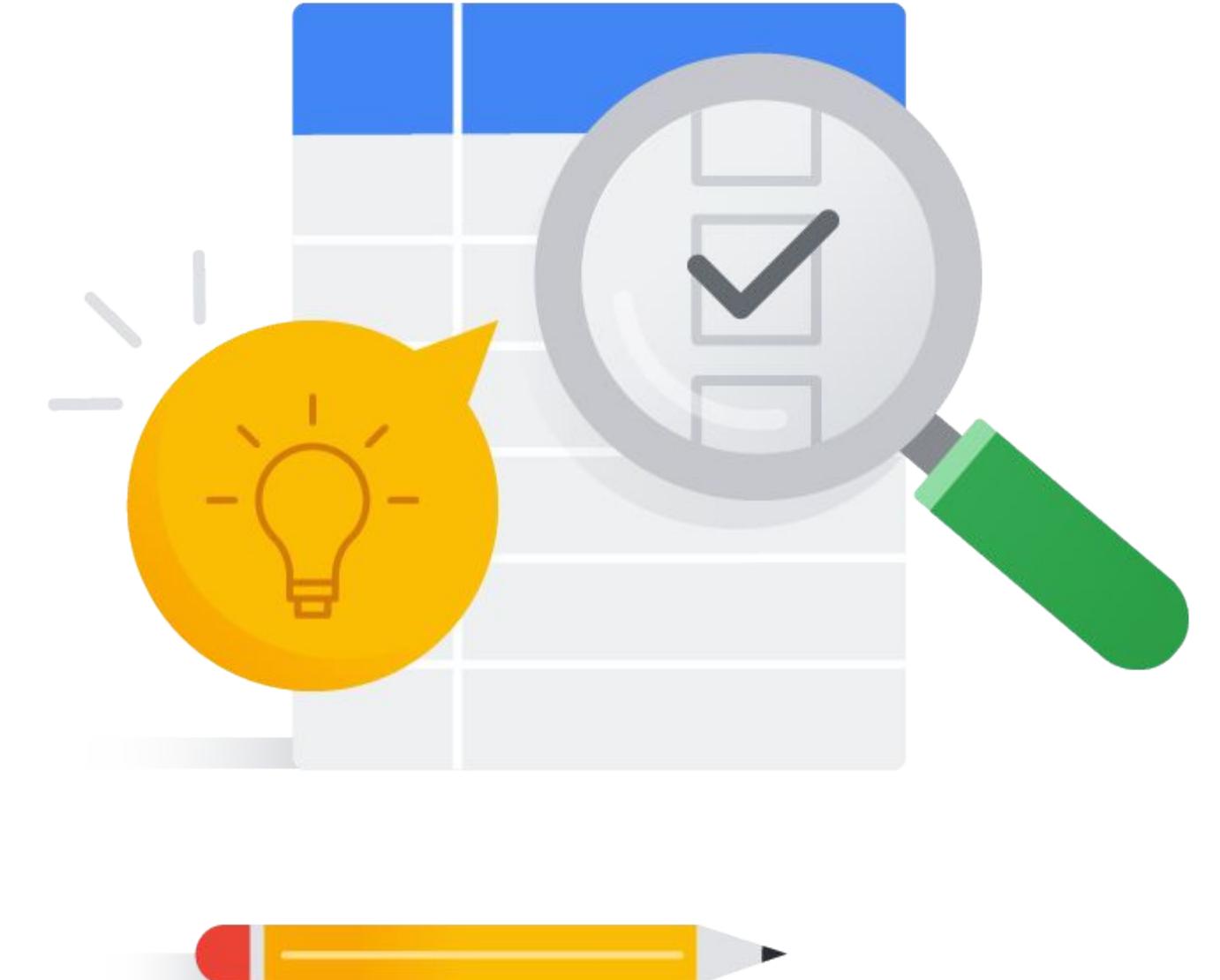
Vertex AI Search can index data in Cloud Storage, BigQuery, Salesforce, your corporate websites, and other sources

Once the data is indexed, query it with natural language

Lab

⌚ 30 min ⚙️

Lab: Exploring Model Garden

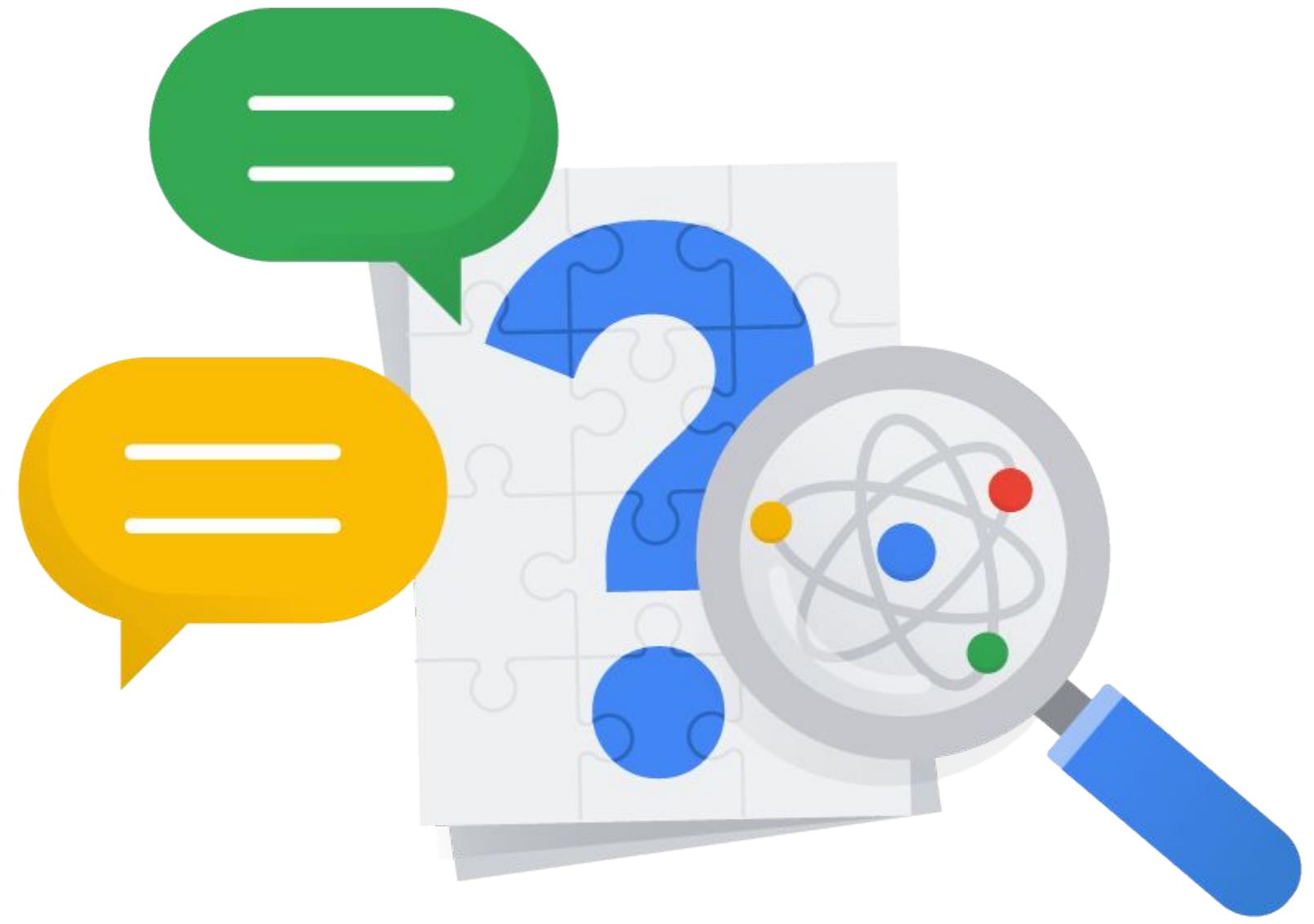


In this module, you learned to ...

- 01 Leverage foundation models for Generative AI
- 02 Use Google's foundational models to optimize generative AI tasks
- 03 Get started building generative AI projects using Vertex AI Studio
- 04 Explore the various use cases for generative AI
- 05 Find and implement models using Vertex AI Model Garden



Questions and answers



Quiz question

Which of the following are tuned versions of the PaLM model, optimized for specific use cases?
(Choose all that apply)

- A: Codey
- B: Chat-Bison
- C: Med PaLM
- D: Sec PaLM
- E: ChatGPT
- F: Imagen
- G: Dall-e

Quiz question

Which of the following are tuned versions of the PaLM model, optimized for specific use cases?
(Choose all that apply)

- A: Codey
- B: Chat-Bison
- C: Med PaLM
- D: Sec PaLM
- E: ChatGPT
- F: Imagen
- G: Dall-e

Quiz question

What tool could you use to find an appropriate ML model based on a use case, and find documentation for a chosen model?

- A: AutoML
- B: Model Garden
- C: Workbench
- D: Vertex AI Studio

Quiz question

What tool could you use to find an appropriate ML model based on a use case, and find documentation for a chosen model?

- A: AutoML
- B: Model Garden
- C: Workbench
- D: Vertex AI Studio

Quiz question

A large language model like Google PaLM or GPT could perform which of the following ML tasks?
(Choose all that apply)

- A: Text generation
- B: Summarization
- C: Image generation
- D: Speech to text
- E. Sentiment analysis

Quiz question

A large language model like Google PaLM or GPT could perform which of the following ML tasks?
(Choose all that apply)

- A: Text generation
- B: Summarization
- C: Image generation
- D: Speech to text
- E. Sentiment analysis

Google Cloud