

Large Language Models

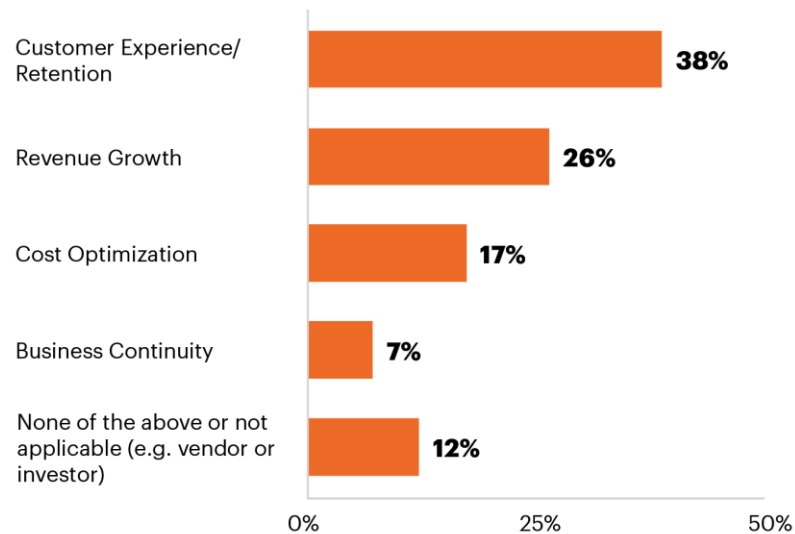
Jeff Prosis

@jprosis



Why Generative AI?

Primary Focus of Generative AI Initiatives



gartner.com

Source: Gartner
© 2023 Gartner, Inc. and/or its affiliates. All rights reserved. 2323718

Gartner.

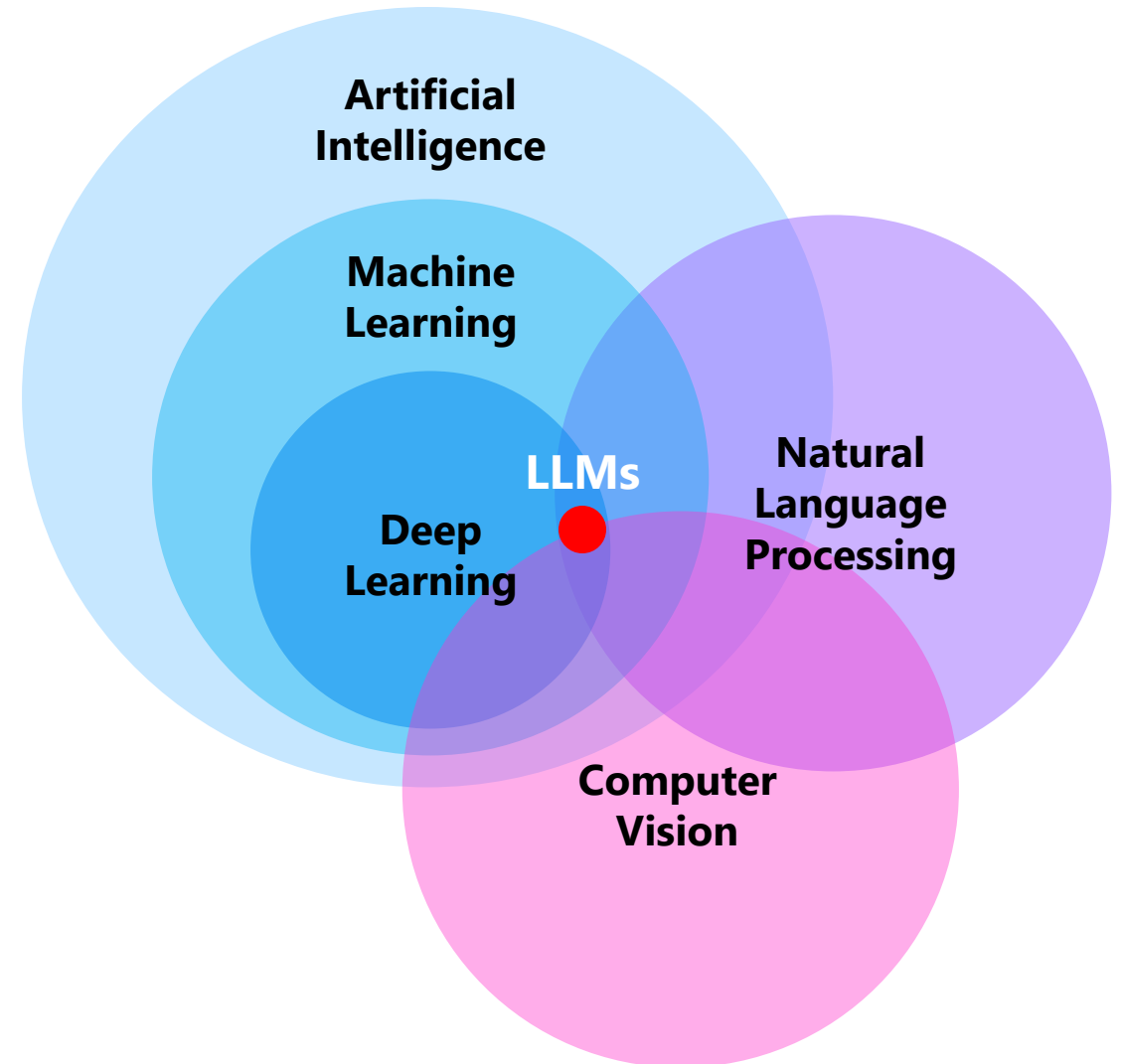


Gartner sees generative AI becoming a **general-purpose technology** with an impact similar to that of the **steam engine, electricity and the internet**. The hype will subside as the reality of implementation sets in, but the **impact of generative AI will grow** as people and enterprises discover more innovative applications for the technology in daily work and life.



Large Language Models (LLMs)

- Deep-learning models (neural networks) that generate text
 - All accept text as input
 - "Write a poem about deep learning"
 - Multimodal LLMs (MLLMs) accept text, images, and sometimes audio and video as input
 - "Describe what you see on this map"
- Billions of parameters trained on trillions of words



Generating Text

write a short story that begins with "It was a dark and stormy night."

It was a dark and stormy night. The wind was howling through the streets, making it difficult for anyone to stay on their feet. The rain was coming down in sheets, drenching anyone who dared to step outside. Inside, a young woman sat huddled in a corner of her small apartment, listening to the storm rage on outside.

She had always been afraid of storms, ever since she was a child. But tonight, there was something different about this one. It seemed to be getting closer and closer, and she could feel the panic rising in her chest. She tried to shake it off, telling herself that it was just a normal storm, but she couldn't shake the feeling that something was coming.

...

Translating Text

Translate the following into French: what time does the train leave for Paris?

À quelle heure part le train pour Paris?

Generating Code

Create a Python function that accepts an array of numbers as input, bubble sorts the numbers, and returns a sorted array

```
def bubble_sort(numbers):  
    # Get the length of the array  
    n = len(numbers)  
  
    # Traverse through all the array elements  
    for i in range(n-1):  
        # Last i elements are already in place  
        for j in range(0, n-i-1):  
            # Traverse the array from 0 to n-i-1  
            # Swap if the element found is greater than the next element  
            if numbers[j] > numbers[j+1]:  
                numbers[j], numbers[j+1] = numbers[j+1], numbers[j]  
  
    return numbers
```

Translating Code

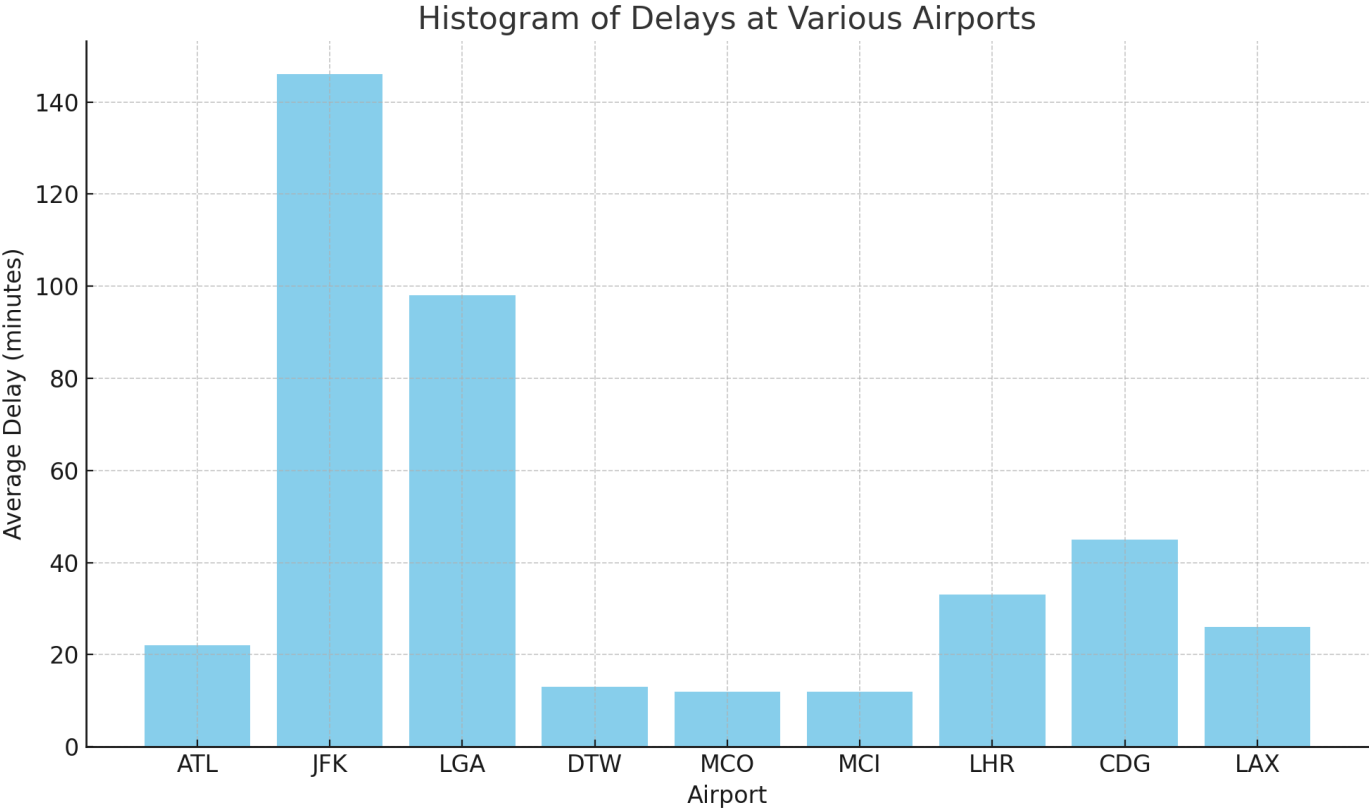
Translate the bubble_sort function into FORTRAN

```
FUNCTION bubble_sort(arr, n) RESULT(result)
  ...
  DO i = 1, n
    DO j = 1, n-i-1
      IF (arr(j) > arr(j+1)) THEN
        temp = arr(j)
        arr(j) = arr(j+1)
        arr(j+1) = temp
      END IF
    END DO
  END DO

  result = arr
END FUNCTION bubble_sort
```

Generating Charts and Graphs

Generate a histogram from the provided data showing delays at various airports



Airport	Average Delay
ATL	22
JFK	146
LGA	98
DTW	13
MCO	12
MCI	12
LHR	33
CDG	45
LAX	26

Analyzing Charts

Explain the chart on the preceding slide

This chart is a histogram depicting the average delays (in minutes) at various airports. Here are some key observations:

- JFK has the highest average delay, approaching 140 minutes.
- LGA follows with a slightly lower average delay, around 100 minutes.
- Most other airports, such as ATL, DTW, MCO, and MCI, have much lower average delays, typically around 20 minutes or lower.
- CDG and LHR, two major international airports, show moderate delays (CDG with around 70 minutes and LHR with about 50 minutes)

The light blue bars indicate the magnitude of the delay for each airport, and the title of the chart indicates it's a "Histogram of Delays at Various Airports."

Demo

LLMs in Action



OpenAI GPT Models

- Available by REST API with an OpenAI account
- SDKs available for Python, Node.js, .NET, Java, Rust, and more



GPT-3.5 Turbo

Groundbreaking LLM with **175B parameters**. Fast and cost-effective. Also known as **ChatGPT**.



GPT-4o

Multimodal model ("o" for "omni") with **128K context window**. Knowledge cutoff of October 2023.



GPT-4o mini

Distilled version of GPT-4o with **128K context window**. Costs more than **30 times less than GPT-4o**.



GPT-4o1

Latest GPT model capable of **complex problem solving**. Currently in preview.

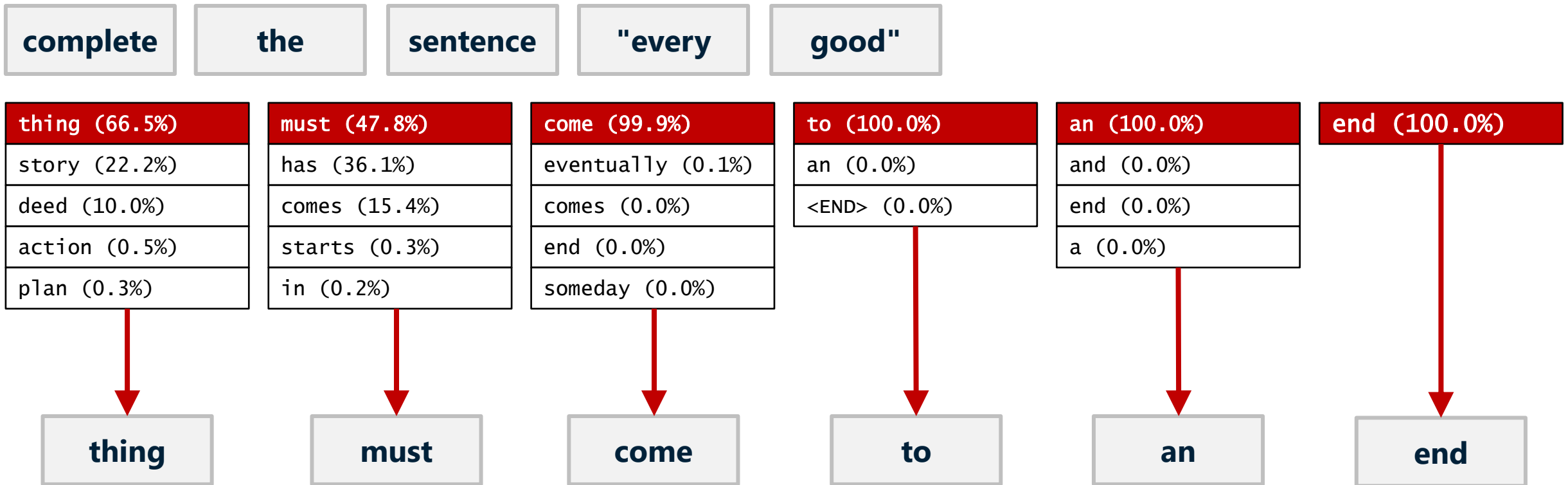


GPT-4o1 mini

Distilled version of GPT-4o1. Currently in preview.

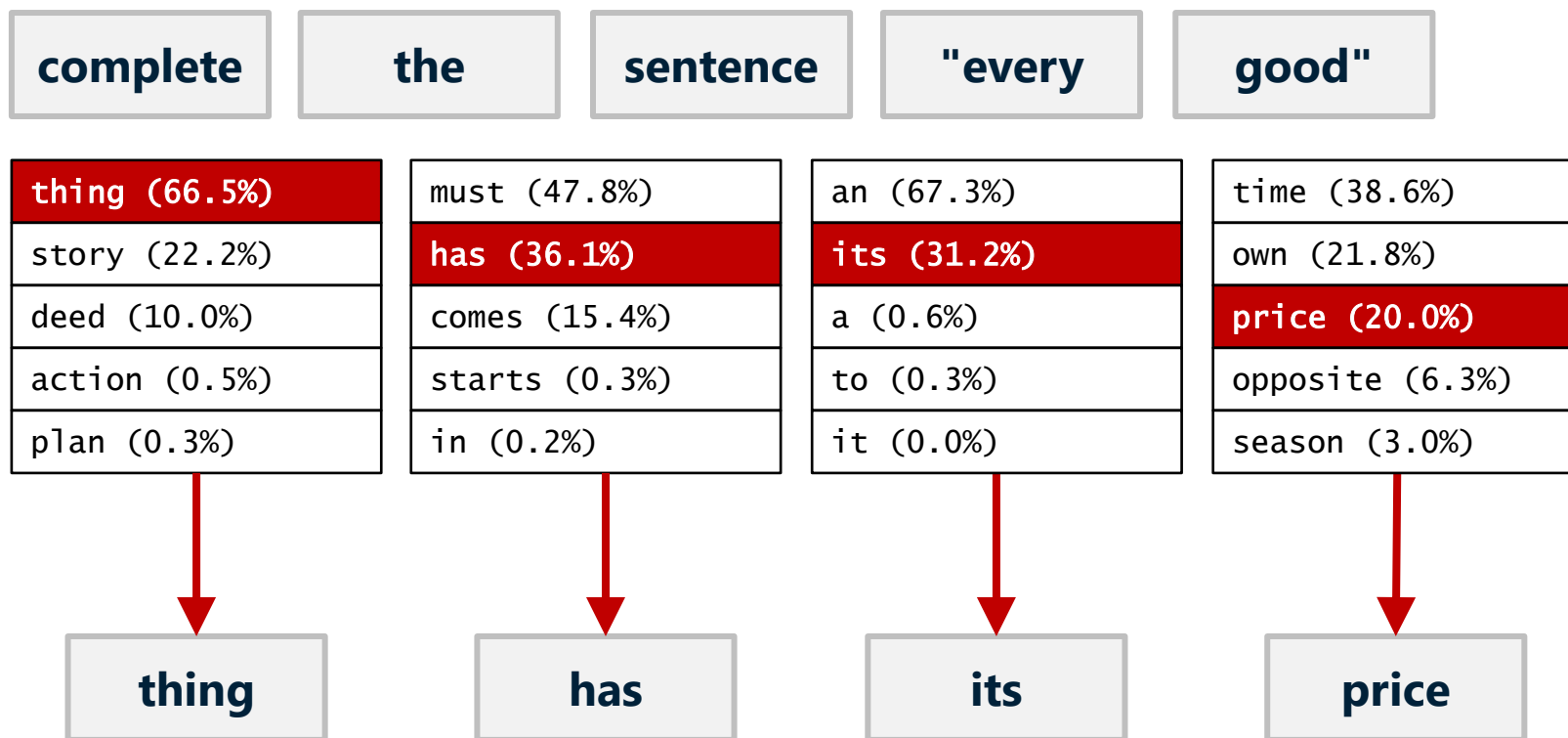
How GPT Models Work

temperature=0.0



How GPT Models Work, Cont.

temperature=0.7



Generating Text with GPT-4o

```
from openai import OpenAI

client = OpenAI(api_key='OPENAI_API_KEY') # Or use OPENAI_API_KEY environment variable
messages = [{ 'role': 'user', 'content': 'Why is the sky blue?' }]

response = client.chat.completions.create(
    model='gpt-4o',
    messages=messages
)

print(response.choices[0].message.content)
```

Streaming the Response

```
client = OpenAI(api_key='OPENAI_API_KEY')
messages = [{ 'role': 'user', 'content': 'Why is the sky blue?' }]

response = client.chat.completions.create(
    model='gpt-4o',
    messages=messages,
    stream=True
)

for chunk in response:
    content = chunk.choices[0].delta.content
    if content is not None:
        print(content, end='')
```

Specifying the Temperature

```
client = OpenAI(api_key='OPENAI_API_KEY')
messages = [{ 'role': 'user', 'content': 'Why is the sky blue?' }]

response = client.chat.completions.create(
    model='gpt-4o',
    messages=messages,
    temperature=0.2
)

print(response.choices[0].message.content)
```


Generating JSON Output

```
prompt = '''
    Which books comprise the Harry Potter series? Please respond
    in JSON using the following format for each book:
    {
        "title": "Harry Potter and the Prisoner of Azkaban",
        "year": 1999
    }
    ...

client = OpenAI(api_key='OPENAI_API_KEY')
messages = [{ 'role': 'user', 'content': prompt }]

response = client.chat.completions.create(
    model='gpt-4o',
    messages=messages,
    response_format={ 'type': 'json_object' }
)

print(response.choices[0].message.content)
```

Generating Structured Output

```
from pydantic import BaseModel

class Book(BaseModel):
    title: str
    year: int

class Books(BaseModel):
    books: list[Book]

client = OpenAI(api_key='OPENAI_API_KEY')
messages = [{ 'role': 'user', 'content': 'Which books comprise the Harry Potter series?' }]

response = client.beta.chat.completions.parse(
    model='gpt-4o',
    messages=messages,
    response_format={Books}
)

print(response.choices[0].message.content)
```

Generating Text Asynchronously

```
from openai import AsyncOpenAI

client = AsyncOpenAI(api_key='OPENAI_API_KEY')
messages = [{ 'role': 'user', 'content': 'Why is the sky blue?' }]

response = await client.chat.completions.create(
    model='gpt-4o',
    messages=messages
)

print(response.choices[0].message.content)
```

Google Gemini

- Available by REST API with Google account
- SDKs available for Python, Node.js, Go, Dart, Swift, and more



Gemini 1.5 Flash

Fastest, most cost-efficient model that accepts **text, images, audio, and video** as input. Features a **1 million token context window** and an estimated 20 billion parameters.



Gemini 1.5 Pro

Google's **most advanced multimodal LLM** with an estimated 120 billion parameters. Features a context window that accepts up to **2 million tokens**.

Generating Text with Gemini

```
import google.generativeai as genai

genai.configure(api_key='GOOGLE_API_KEY')
model = genai.GenerativeModel('gemini-1.5-flash')
response = model.generate_content('Why is the sky blue?')
print(response.text)
```

Streaming the Response

```
import google.generativeai as genai

genai.configure(api_key='GOOGLE_API_KEY')
model = genai.GenerativeModel('gemini-1.5-flash')
response = model.generate_content('Why is the sky blue?', stream=True)

for chunk in response:
    print(chunk, end='')
```

Specifying the Temperature

```
import google.generativeai as genai

genai.configure(api_key='GOOGLE_API_KEY')
model = genai.GenerativeModel('gemini-1.5-flash')
config = genai.types.GenerationConfig(temperature=0.2)

response = model.generate_content(
    'Why is the sky blue?',
    generation_config=config
)

print(response.text)
```

Generating JSON Output

```
prompt = '''
    Which books comprise the Harry Potter series? Please respond in JSON
    using the following format for each book. Do not output markdown.
    {
        "title": "Harry Potter and the Prisoner of Azkaban",
        "year": 1999
    }
    ...

genai.configure(api_key='GOOGLE_API_KEY')
model = genai.GenerativeModel('gemini-1.5-flash')
response = model.generate_content('Which books comprise the Harry Potter series?')
print(response.text)
```


Generating Structured Output

```
import TypedDict from typing_extensions

class Book(TypedDict):
    title: str
    year: int

config=genai.types.GenerationConfig(
    response_mime_type='application/json',
    response_schema=list[Book]
)

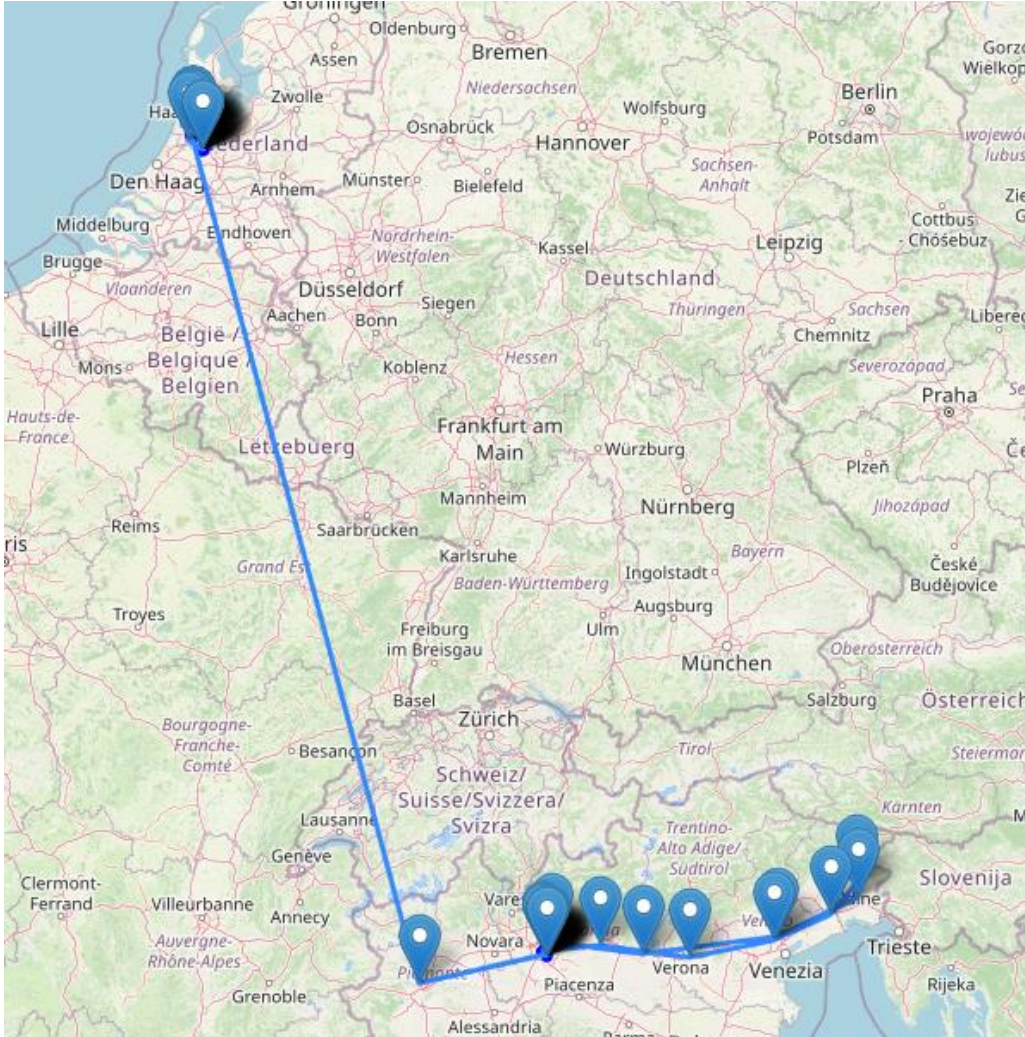
genai.configure(api_key='GOOGLE_API_KEY')
model = genai.GenerativeModel('gemini-1.5-flash')
response = model.generate_content(prompt, generation_config=config)
print(response.text)
```

Demo

Text Generation



Multimodal LLMs (MLLMs)



Describe the shipment plotted on this map.

The shipment began in the Netherlands and made its way to northeastern Italy using a combination of air and ground transportation.

The initial segment went by air, originating from a location near the coast of the Netherlands, possibly near Rotterdam, and ending in northeastern Italy, likely near Milan, as the shipment appears to fly across the Netherlands, over Germany, and into Italy.

The subsequent legs of the journey were traveled by road. The shipment passed through several cities in northern Italy, eventually reaching its destination near Venice.

The map does not specify exact highways, but the route within Italy likely involved major roads such as the A4 motorway, which connects Milan to Venice.

Submitting an Image to GPT-4o

```
messages = [{  
    'role': 'user',  
    'content': [  
        { 'type': 'text', 'text': 'Describe what you see in this image' },  
        { 'type': 'image_url', 'image_url': { 'url': 'IMAGE_URL' } }  
    ]  
}]
```

```
response = client.chat.completions.create(  
    model='gpt-4o',  
    messages=messages  
)
```

Submitting Multiple Images to GPT-4o

```
messages = [{  
    'role': 'user',  
    'content': [  
        { 'type': 'text', 'text': 'Describe similarities between these images' },  
        { 'type': 'image_url', 'image_url': { 'url': 'IMAGE_URL_1' } },  
        { 'type': 'image_url', 'image_url': { 'url': 'IMAGE_URL_2' } }  
    ]  
}]  
  
response = client.chat.completions.create(  
    model='gpt-4o',  
    messages=messages  
)
```

Submitting an Image to Gemini

```
image = Image.open('IMAGE_PATH')  
prompt = 'Describe what you see in this image'  
response = model.generate_content([image, prompt])
```

Submitting Multiple Images to Gemini

```
image1 = Image.open('IMAGE_PATH_1')  
image2 = Image.open('IMAGE_PATH_2')  
prompt = 'Describe similarities between these images'  
response = model.generate_content([image1, image2, prompt])
```


Transcribing Audio with Gemini

```
import time

audio = genai.upload_file(path='AUDIO_PATH')

while audio.state.name == 'PROCESSING':
    time.sleep(1)
    audio = genai.get_file(audio.name)

if audio.state.name == 'ACTIVE':
    response = model.generate_content([audio, 'Transcribe this audio'])
```


Transcribing Video with Gemini

```
import time

video = genai.upload_file(path='VIDEO_PATH')

while video.state.name == 'PROCESSING':
    time.sleep(1)
    video = genai.get_file(video.name)

if video.state.name == 'ACTIVE':
    response = model.generate_content([video, 'Transcribe this video'])
```

Demo

Multimodal LLMs



Whisper

- OpenAI's speech-to-text model
 - Convert speech to text in more than 50 languages (transcription)
 - Convert speech in more than 50 languages to English text (translation)
- Accepts audio files in seven formats
 - MP3, MP4, MPEG, MPGA, M4A, WAV, and WEBM
- Trained on 680,000 hours of multilingual audio
- Not for real-time transcription or translation
- More information at <https://cdn.openai.com/papers/whisper.pdf>

Converting Speech to Text

```
from openai import OpenAI

client = OpenAI(api_key='OPENAI_API_KEY')
audio_file= open('PATH_TO_AUDIO_FILE', 'rb')

response = client.audio.transcriptions.create(
    model='whisper-1',
    file=audio_file,
    response_format='text' # Default == JSON
)

print(response.text)
```

Translating Speech into English

```
from openai import OpenAI

client = OpenAI(api_key='OPENAI_API_KEY')
audio_file= open('PATH_TO_AUDIO_FILE', 'rb')

response = client.audio.translations.create(
    model='whisper-1',
    file=audio_file,
    response_format='text' # Default == JSON
)

print(response.text)
```

TTS

- OpenAI's text-to-speech model
- Supports six neural voices, five audio output formats, and more than 50 spoken languages
 - Voices: Alloy, Echo, Fable, Onyx, Nova, and Shimmer
 - Output formats: MP3 (default), AAC, Opus, FLAC, WAV, and PCM
- Supports block output and streaming output
- Voices are currently optimized for English

Converting Text to Speech

```
from openai import OpenAI

client = OpenAI(api_key='OPENAI_API_KEY')

response = client.audio.speech.create(
    model='tts-1', # Or tts-1-hd for higher quality at the expense of increased latency
    voice='fable', # alloy, echo, fable, onyx, nova, or shimmer
    input='Tranquility base here. The Eagle has landed.'
)

with open('output.mp3', 'wb') as audio_file:
    audio_file.write(response.content)
```

Streaming the Response

```
from openai import OpenAI

client = OpenAI(api_key='OPENAI_API_KEY')

response = client.audio.speech.create(
    model='tts-1', # Or tts-1-hd for higher quality at the expense of increased latency
    voice='fable', # alloy, echo, fable, onyx, nova, or shimmer
    input='Tranquility base here. The Eagle has landed.'
)

response.stream_to_file('output.mp3')
```