atmosera®

Gold Microsoft Partner | Azure Expert MSP

OPENAI

Jeff Prosise

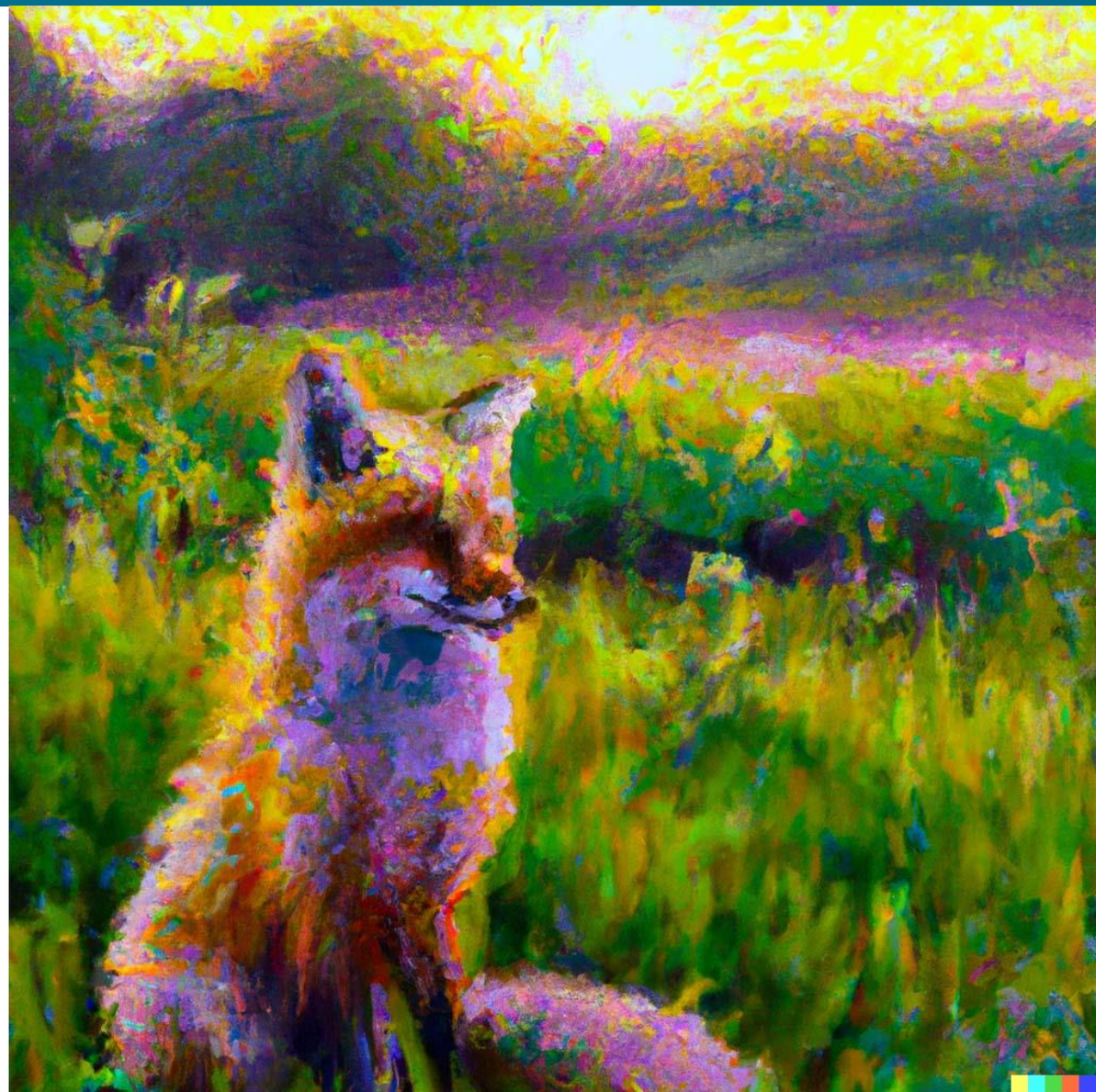
ChatGPT

```
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.

...
```

DALL·E 2

Painting of a fox sitting in a field at sunrise in the style of Claude Monet



OpenAI

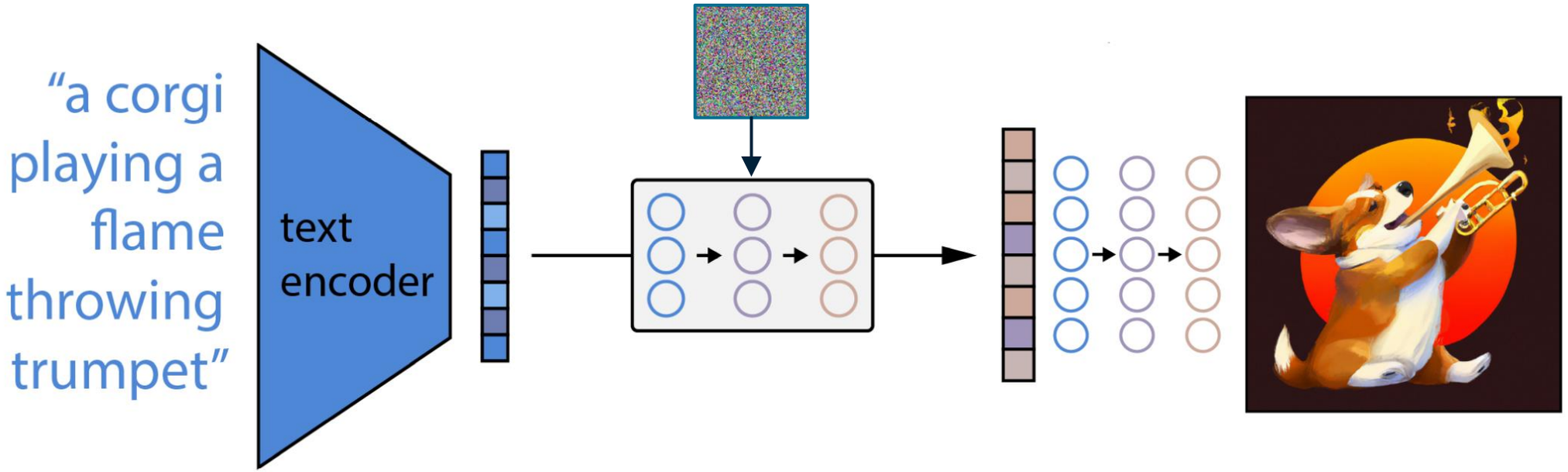
- Creator of ChatGPT, DALL·E 2, and other cutting-edge models
- Offers REST APIs for generating text, images, code, and text embeddings, for converting speech to text, and more

GPT	Codex	DALL·E 2	Embeddings	Whisper
Models that support **text and code generation**. Includes **ChatGPT**.	Models that support **code generation**, converting code to other languages, and more. **Discontinued on March 23, 2023**.	Produces **images from natural-language prompts**. Supports inpainting, outpainting, image variations, and more.	Models that generate **embedding vectors** from text. Used for semantic search, recommender systems, and more.	General-purpose **speech-to-text engine** trained on 680,000 hours of audio. Also available via open source.

DALL·E 2

- Diffusion model from OpenAI featuring 3.5 billion parameters
- Trained on 650 million text-image pairs scraped from the Internet
- Available by REST API
  - Requires an OpenAI account
- Supports image generation and modification, image variations, inpainting, and outpainting

How DALL·E 2 Works



Contrastive Language-Image Pretraining (CLIP) model encodes the prompt, **generating a text embedding** in latent space

"Prior" model generates an **image embedding** from the text embedding and random noise

Decoder uses **reverse diffusion** to **generate a 64x64 image** from the image embedding. CNNs **upsample the image** to 256x256, 512x512, and 1,024x1,024 to produce the final image.

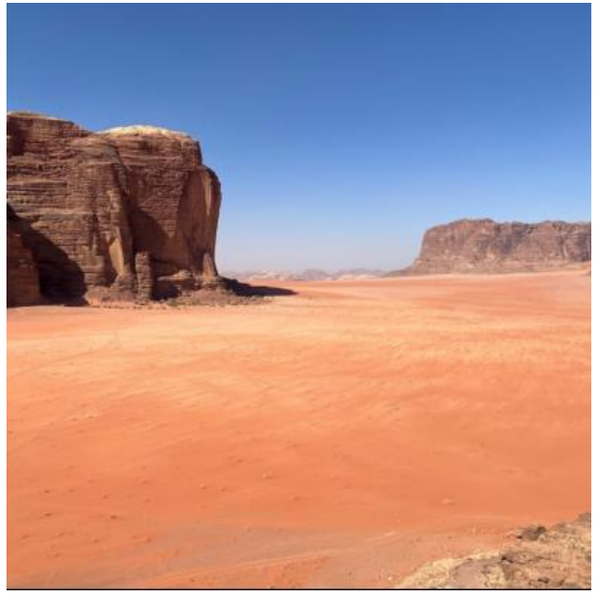
Generating Images with DALL·E 2

```
import openai, base64, PIL, io
openai.api_key = 'OPENAI_API_KEY'

response = openai.Image.create(
    prompt='Photo of a purple unicorn',
    size='512x512',
    n=1,
    response_format='b64_json'
)

image_data = response['data'][0]['b64_json']
image = io.BytesIO(base64.b64decode(image_data))
```

Creating Variations of Existing Images

```
response = openai.Image.create_variation(
    image=open('PATH_TO_IMAGE', 'rb'),
    size='512x512', n=1, response_format='b64_json'
)
```

Original image



Variation created by DALL·E 2

Inpainting

```
response = openai.Image.create_edit(
    image=open('PATH_TO_IMAGE', 'rb'), # Path to original image
    mask=open('PATH_TO_IMAGE_MASK', 'rb'), # Path to same image with transparent pixels
    prompt='Photograph of two people standing on a cliff overlooking the beach',
    n=1, size='512x512', response_format='b64_json'
)
```

Original image with fence in background

Image mask with transparent pixels denoting regions to be inpainted

Outpainting

```
response = openai.Image.create_edit(
    image=open('PATH_TO_IMAGE', 'rb'),
    mask=open('PATH_TO_IMAGE', 'rb'), # Same image
    prompt='Painting of a girl standing in a kitchen',
    n=1, size='512x512', response_format='b64_json'
)
```

Transparent pixels identifying region to be outpainted

Region to be expanded via outpainting

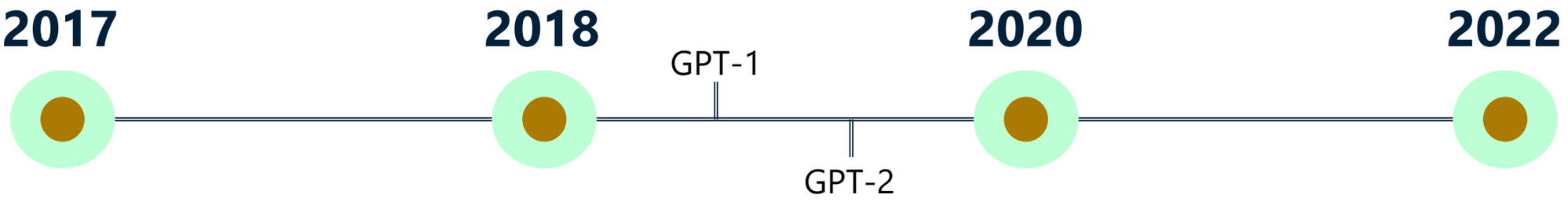
atmosera®

Gold Microsoft Partner | Azure Expert MSP

Demo

DALL·E 2

The Road to ChatGPT



Transformer

Neural architecture for NLP models that **replaced recurrent neural networks** (RNNs). Transformers use **self-attention** to connect related words in a sequence and can **process sequences of any length**.

BERT

Transformer-based model pretrained on more than **3 billion words** from Wikipedia and Google Books using **Masked Language Modeling**. Can be fine-tuned to perform specific NLP tasks such as **Neural Machine Translation**.

GPT-3

Large Language Model (LLM) similar to BERT but trained on **500 billion words** from the Internet. **10 times larger** than any neural network previously built, featuring **175 billion parameters**. Performs certain NLP tasks **without fine-tuning**.

ChatGPT

Fine-tuned version of GPT-3.5. Further refined using **Reinforcement Learning from Human Feedback** (RLHF). Does everything GPT-3.5 does (and more) at **1/10th the cost**. Made available via API on March 1st, 2023.

How ChatGPT was Created

Step 1: Supervised Fine-Tuning

Fine-tune GPT-3 with **13,000 supervised-learning samples**. Each sample consists of a prompt (for example, "write a short story that begins with...") generated manually or selected from **actual inputs to the GPT-3 API** and a response **generated by humans**.

Step 2: Reward Model

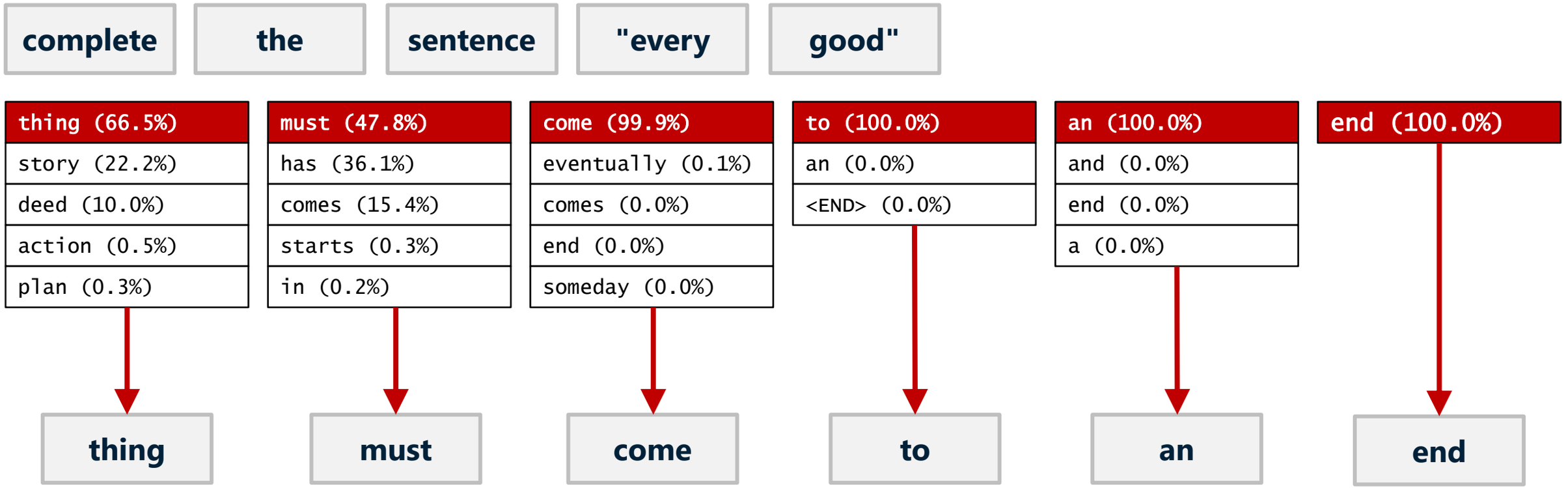
Use the model fine-tuned in Step 1 to generate **several responses** for each of **tens of thousands of prompts**. Manually **rank each set of responses** from best to worst and **train a reward model** with the labeled dataset.

Step 3: Reinforcement Learning

Input a prompt to the model. Use the reward model to **score the resulting response** for quality/desirability. Feed the score back in to the model and use **Proximal Policy Optimization** (PPO) to fine-tune the model's behavior. Do this repeatedly until fine-tuning is complete.

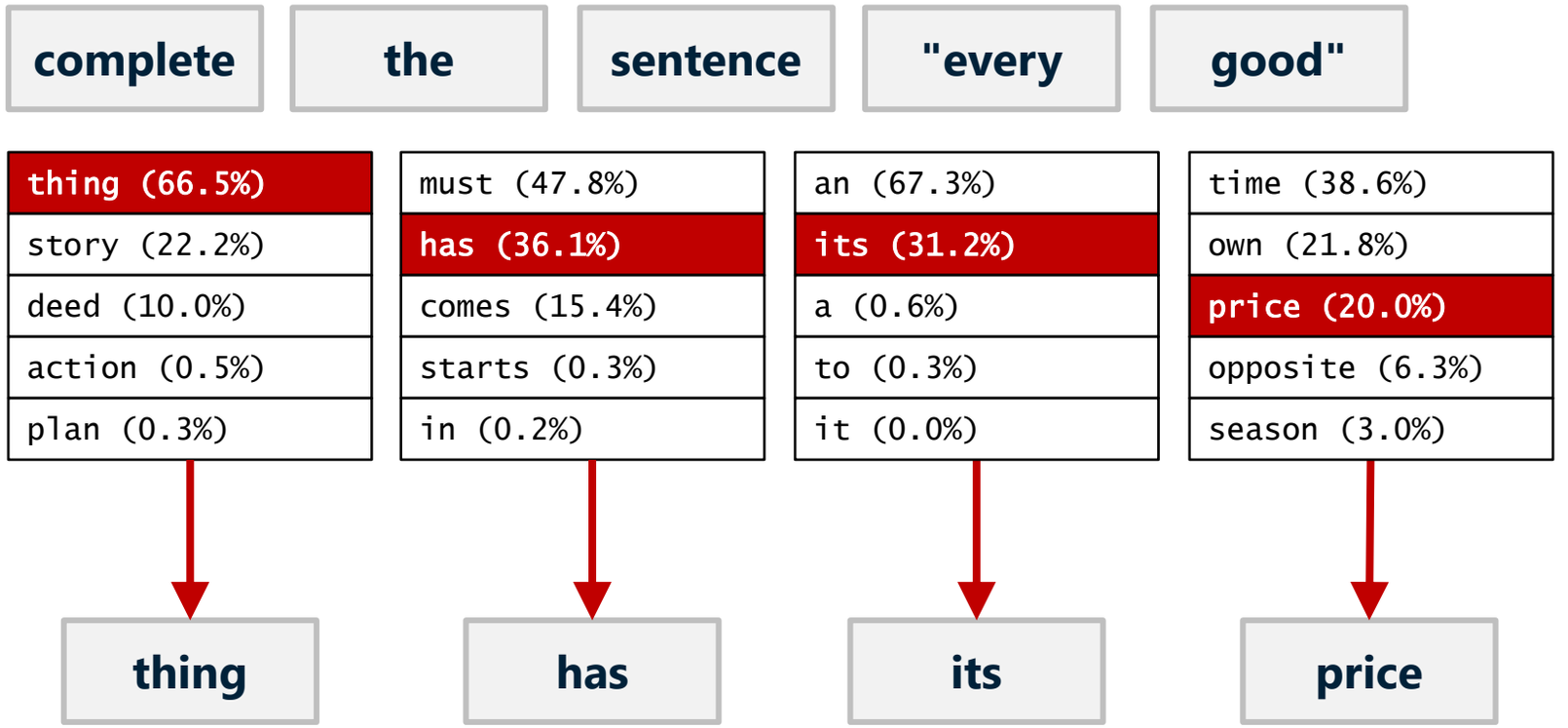
How ChatGPT Works

`temperature=0.0`



How ChatGPT Works, Cont.

```
temperature=0.7
```



Generating Text with ChatGPT

```
messages = [
    { 'role': 'user', 'content': 'Describe molecular biology in the style of Dr. Seuss' }
]

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=messages,
    temperature=0.7
)
```

Streaming the Response

```
messages = [
    { 'role': 'user', 'content': 'Describe molecular biology in the style of Dr. Seuss' }
]

chunks = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=messages,
    stream=True
)

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

Translating Text

```
content = 'Translate the following text from English to French: Best food ever!'

messages = [{ 'role': 'user', 'content': content }]

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=messages,
    temperature=0
)
```

Analyzing Sentiment

```
content = 'Indicate whether the following Tweet\'s sentiment is positive or ' \
          'negative: Great food and excellent service'

messages = [{ 'role': 'user', 'content': content }]

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=messages
)
```

Answering Questions

```
content = f'Answer the following question, and if you don\'t know the answer, ' \
          f'say "I don\'t know."\n\n' \
          f'Q: Who was the first president of the United States?\n\n' \
          f'A: '

messages = [{ 'role': 'user', 'content': content }]

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=messages,
    max_tokens=500
)
```

Answering Contextual Questions

```
content = f'Answer the following question using the provided context, and if the ' \
          f'answer is not contained within the context, say "I don\'t know."\n\n' \
          f'Context: {context}\n\n' \ # Insert text to be searched for an answer
          f'Q: Who was the first president of the United States?\n\n' \
          f'A: '

messages = [{ 'role': 'user', 'content': content }]

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=messages,
    max_tokens=500
)
```

Handling Errors

```
try:
    response = openai.ChatCompletion.create(
        model='gpt-3.5-turbo',
        messages=messages
    )

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

except AuthenticationError as e:
    print('Invalid API key')
except ServiceUnavailableError as e:
    print('ChatGPT temporarily unavailable')
except Exception as e:
    print('Call failed')
```

atmosera®

Gold Microsoft Partner | Azure Expert MSP

Demo

ChatGPT

Code Generation

- ChatGPT does code!
  - Write code (convert natural language into code) and unit tests
  - Comment and explain code (convert code into natural language)
  - Convert code from one programming language to another
  - Complete code, edit/refactor code, and find bugs
- Supports dozens of languages, including Python, C#, and Java
- Trained with billions of lines of code from GitHub
- The basis for GitHub Copilot

Generating Code

```
content = 'Create a Python function that accepts an array of numbers as ' \
          'input, bubble sorts the numbers, and returns a sorted array'

messages = [{ 'role': 'user', 'content': content }]

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=messages,
    temperature=0
)
```

Converting Code to Natural Language

```
content = 'Explain what the following code does:\n' \
          'def bubble_sort(arr):\n' \
          '    n = len(arr)\n' \
          '    for i in range(n):\n' \
          '        for j in range(0, n-i-1):\n' \
          '            if arr[j] > arr[j+1]:\n' \
          '                arr[j], arr[j+1] = arr[j+1], arr[j]\n' \
          '    return arr'

messages = [{ 'role': 'user', 'content' : content }]

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo', messages=messages
)
```

Translating Code

```
content = 'Convert the following Python code into FORTRAN:\n' \
          'def bubble_sort(arr):\n' \
          '    n = len(arr)\n' \
          '    for i in range(n):\n' \
          '        for j in range(0, n-i-1):\n' \
          '            if arr[j] > arr[j+1]:\n' \
          '                arr[j], arr[j+1] = arr[j+1], arr[j]\n' \
          '    return arr'

messages = [{ 'role': 'user', 'content' : content }]

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo', messages=messages, max_tokens=256
)
```

atmosera®

Gold Microsoft Partner | Azure Expert MSP

Demo

Code Generation

Embeddings API

- Generates high-quality embedding vectors from text
  - `text-embedding-ada-002` model is best and most cost-effective
  - Generates vectors of 1,536 floating-point numbers and is applicable to long and short text samples, replacing earlier models that were sensitive to text length
- Compute similarity between two text samples by generating embeddings for each and computing the dot product
  - Useful for semantic-search systems, recommender systems, deduplication systems, and other similarity-based systems
- Combine with vector databases such as **Pinecone** or **Milvus** to create systems that scale to millions of embedding vectors

Generating an Embedding Vector

```
response = openai.Embedding.create(
    model='text-embedding-ada-002',
    input='Four score and seven years ago our fathers brought forth, ' \
          'upon this continent, a new nation, conceived in liberty, and ' \
          'dedicated to the proposition that all men are created equal'
)

embedding = response.data[0].embedding

# [0.01071165315806865, -0.02277084067463874, -0.026825921609997754, ...]
```

Comparing Embedding Vectors

```
x = openai.Embedding.create(
    model='text-embedding-ada-002', input='Jeff is a common name'
).data[0].embedding

y = openai.Embedding.create(
    model='text-embedding-ada-002', input='My name is Jeff'
).data[0].embedding

similarity = np.dot(np.array(x), np.array(y))
# 0.9004762760198348
```

atmosera®

Gold Microsoft Partner | Azure Expert MSP

Demo

ChatGPT Over Custom Data