

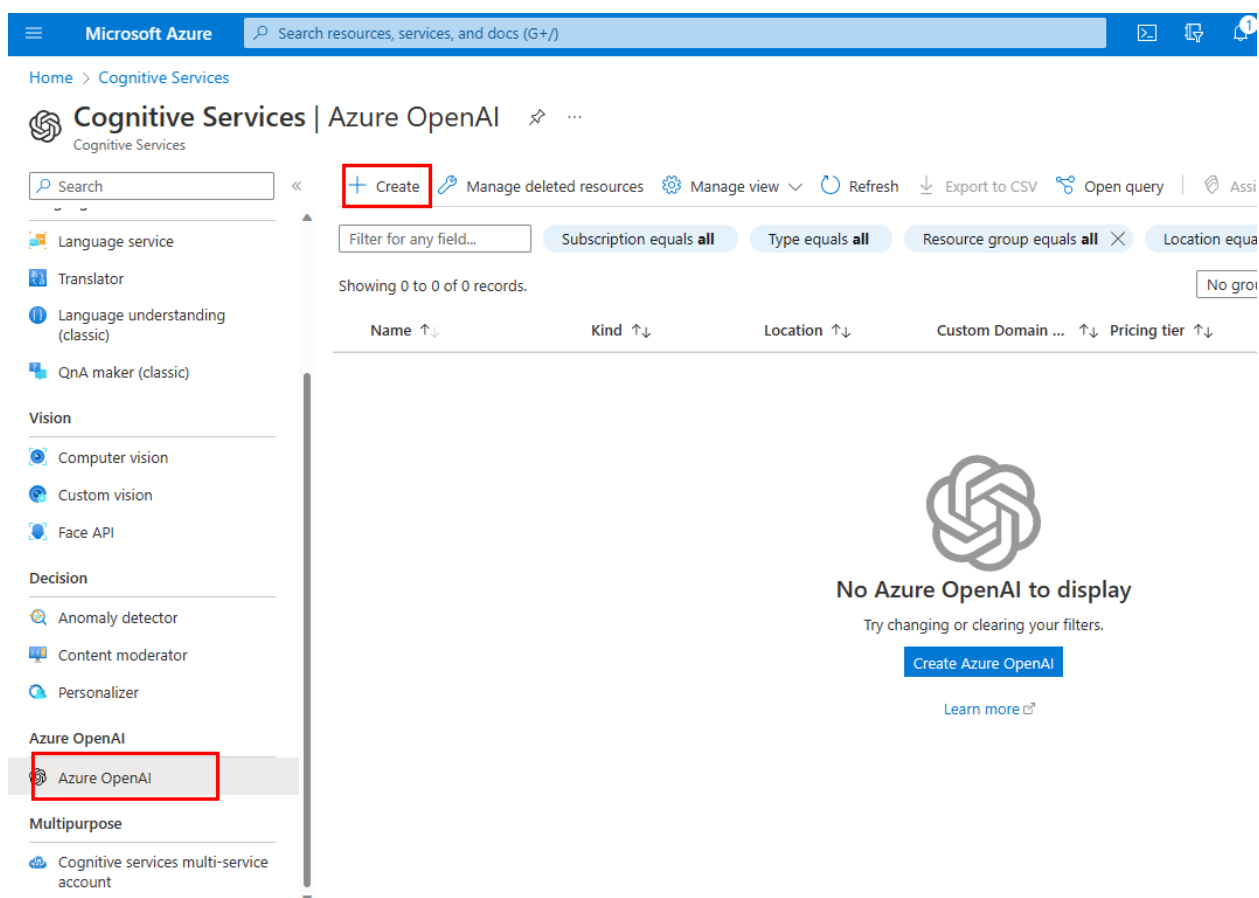
# 1 AZURE OPENAI

## 1.1 概述

Azure OpenAI 服务允许通过 REST API 访问 OpenAI 的强大语言模型，包括 GPT-3、Codex 和 Embeddings 模型系列。这些模型可以轻松适应特定的任务，包括但不限于内容生成、汇总、语义搜索和自然语言到代码的转换。用户可以在 Azure OpenAI Studio 中通过 REST API、Python SDK 或基于 Web 的界面访问该服务。

## 1.2 创建 AZURE OPENAI 服务资源

浏览器访问【[Azure Portal](#)】- 搜索框搜索【Azure OpenAI】- 【创建】



定义资源信息后创建 - 【Review + Create】，稍等片刻大概 2~5 分钟即可创建完成

# Create Azure OpenAI ...

Basics   Tags   Review + create

Enable new business solutions with OpenAI's language generation capabilities powered by GPT-3 models. These models have been pretrained with trillions of words and can easily adapt to your scenario with a few short examples provided at inference. Apply them to numerous scenarios, from summarization to content and code generation.

## Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

OpenAI Test Subscription   选择 OpenAI 加入应用程序中使用的 Azure 订阅   ▼

Resource group \* ⓘ

test-resource-group   包含 OpenAI 资源的 Azure 资源组。可以创建新组或将其添加到预先存在的组。   ▼

Create new

## Instance details

Region \* ⓘ

South Central US   实例的位置。不同位置可能会导致延迟，但不会影响资源的运行时可用性。   ▼

Name \* ⓘ

azure-openai-test-001   资源的描述性名称，自定义   ✓

Pricing tier \* ⓘ

Standard S0   目前只有 1 个定价层可用于该服务   ▼

[View full pricing details](#)

[Review + create](#)   [< Previous](#)   [Next : Tags >](#)

## 1.3 在 AZURE OPENAI 服务资源下部署模型

在该资源下部署一个模型，azure 上必须先部署模型，然后才能进行调用

通过 Azure portal，选择创建好的 Azure OpenAI 的资源名称 - 【模型部署】 - 【创建】 - 自定义模型的名称、选择一个基模型。 - 【保存】

Microsoft Azure 搜索资源、服务和文档(G+/)

主页 > 认知服务 | Azure OpenAI > rnd-test-01

**rnd-test-01 | 模型部署** ☆ ...

Azure OpenAI

搜索

概述

活动日志

访问控制(标识和访问管理)

标记

诊断并解决问题

资源管理

密钥和终结点

**模型部署**

定价层

网络

标识

成本分析

属性

锁

监视

警报

指标

诊断设置

+ 创建 修改 删除 刷新 转到 Azure OpenAI Studio

筛选... 所有模型部署状态

模型部署名称	状态	模型
txt-davinci003-1	Succeeded	text-davinci-003
txt-curie001-1	Succeeded	text-curie-001

**创建模型部署**

创建模型部署

设置模型部署以对提供的基模型或自定义模型进行 API 调用。已完成的模型部署可供使用。模型部署完成且可供使用时，模型部署状态将转为成功。

模型部署名称 \* ①

text-d002-deploy001 部署模型的名称 ✓

模型 ①

text-davinci-002 选择一个基模型，一个服务资源下，一种模型只能部署一次 ✓

**擅长:** 复杂意图, 因果关系, 受众总结

Davinci 是功能最强大的模型系列，通常只需较少的指令即可执行其他模型可以执行的任何任务。对于需要理解大量内容的应用程序(例如特定受众的摘要和创意内容生成)，Davinci 将产生最佳结果。

[了解详细信息](#)

标准模型部署提供完全托管的体验，将按使用模型处理的令牌向你收费。微调模型将额外收取每小时托管费用

保存

也可通过 [Azure OpenAI Studio](#) 部署模型，[参考文档](#)

## 1.4 获取密钥和终结点

浏览器访问 [【Azure Portal】](#) - 搜索框搜索 **【Azure OpenAI】** - 选择创建好的 Azure OpenAI 的资源名称 - **【密钥和终结点】**，复制一个密钥和终结点以供后面调用 API 时使用。

🔑 rnd-test-01 | 密钥和终结点
 ☆ ⋮

«
🔄 重新生成密钥1
🔄 重新生成密钥2

- 📄 概述
- 📅 活动日志
- 👤 访问控制(标识和访问管理)
- 📌 标记
- 🔧 诊断并解决问题

资源管理

🔑 密钥和终结点

📦 模型部署

💰 定价层

🌐 网络

🏷️ 标识

💵 成本分析

📋 属性

🔒 锁

监视

📊 警报

ⓘ 这些密钥用于访问认知服务 API。请勿共享你的密钥。请安全地存储这些密钥(例如，使用 Azure Key Vault)。另外，建议定期重新生成这些密钥。执行 API 调用只需要一个密钥。在重新生成第一个密钥时，可以使用第二个密钥继续访问服务。

显示密钥

密钥 1

.....

密钥 2

.....

位置/区域 🕒

southcentralus 📄

终结点

https://rnd-test-01.openai.azure.com/ 📄

## 2 模型介绍

---

### 2.1 GPT-3

### 2.2 CODEX

### 2.3 EMBEDDINGS

### 2.4 DELL-E

### 2.5 WHISPER

### 2.6 CHATGPT

## 3 提示工程和 FINE-TUNE

---

### 3.1 提示工程 (CONTEXT LEARNING)

Prompt Engineering 是短期/一次性改变输出的方法


#### 3.1.1 零样本学习 (zero-shot)

在模型输入中只提供任务描述（通常包括任务类型、输入数据的格式、预期输出的格式和数据类型等信息）和测试样例输入，得到测试样例输出

示例 1:

没有提示的输出

QW 今天出门比较晚，路上会不会堵车？

 很抱歉，我无法确定今天出门比较晚路上是否会堵车，因为这取决于很多因素，如交通状况、路况、天气等。

加了提示学习的输出

QW 今天出门比较晚，路上会不会堵车？只回答会或不会，不需要其他的答案

 会。

示例二：

QW 按时间倒序列出，只列出时间点，openAI和微软的合作历程

 1. 2020年7月22日  
2. 2019年6月  
3. 2018年6月  
4. 2017年11月

### 3.1.2 单样本学习 (one-shot)

在模型输入中提供任务描述、一条训练样例和测试样例输入，得到测试样例输出；

示例 1：

QW

按照以下格式输出，不要解释

今天出门比较晚，路上会不会堵车？ ==> 不堵

话卡流量用超了，会不会多扣费



话卡流量用超了，会不会多扣费？ ==> 不会

示例 2:

QW

输入：一只黑白相间的动物，身上有长长的条纹，喜欢吃肉，会爬树，通常生活在非洲的草原上。

输出：这只动物是一只豹子。

输入：一只灰色的动物，长长的尾巴和细长的腿，喜欢吃老鼠，是夜行动物，通常生活在北极地区。



输出：这只动物是一只极地狐。

### 3.1.3 少量样本学习 (few-shot)

在模型输入中提供任务描述、少量训练样例和测试样例输入，得到测试样例输出。

示例

QW

我有以下几个动物类别

狗：拉布拉多、贵宾、哈士奇、柯基、松狮

猫：波斯猫、暹罗猫、布偶猫、英国短毛猫、美国短毛猫

鸟：鹦鹉、金丝雀、乌鸦、鸵鸟、企鹅

鱼：金鱼、鲨鱼、海马、鲸鱼、鲤鱼

蛇：眼镜蛇、蟒蛇、水蛇、蝮蛇、蝰蛇

根据描述为每个动物提供一个标签

这是一只四肢很短的狗，有很长的毛，非常可爱。标签：狗

这是一只非常漂亮的波斯猫，有一只大而明亮的眼睛。标签：猫

这是一只很大的鸟，有一对非常漂亮的翅膀。标签：鸟

这是一条很长的蛇，它的皮肤有很多颜色。标签：蛇

现在起，请帮我分类

这是一只有羽毛的动物，它可以飞行

这是一种淡水鱼，它有许多鳞片，可以长到很大



这是一只有羽毛的动物，它可以飞行。标签：鸟

这是一种淡水鱼，它有许多鳞片，可以长到很大。标签：鱼

QW

这是一只有白色猫的动物，它会抓老鼠



这是一只有白色猫的动物，它会抓老鼠。标签：猫

### 3.2 微调

Fine-tuning 主要是用来改善长期输出形式，Fine-tuning 可以用来限定范围，提高 domain 相关度。



### 3.2.1 [API 接口调用](#)

### 3.2.2 [Azure OpenAI Studio](#) 操作

## 4 [定价](#)

## 5 [配额和限制](#)

## 6 模型调用 DEMO

---

### 6.1 接口参数说明

#### 6.1.1 GPT-3

#### 6.1.2 Codex

#### 6.1.3 Embeddings

#### 6.1.4 DELL-E

#### 6.1.5 Whisper

#### 6.1.6 ChatGPT

#### 6.1.7 tiktoken

### 6.2 通过 PYTHON SDK 调用

#### 6.2.1 获取可用模型、部署信息（将红色字体部分换成自己的配置）

##### 6.2.1.1 获取所有模型列表清单

```
# *_ coding: utf-8 *_
```

```
# 先安装 python 模块
```

```
#pip install openai num2words matplotlib plotly scipy scikit-learn transformers
```

```

# 导入需要用到的模块
import os
import json
import openai

# 设置 Azure OpenAI 的 API key , 可参考上述 1.4 步骤
openai.api_key = "<Azure OpenAI 服务资源的密钥>"
# 设置 Azure OpenAI 的 API 终结点 , 可参考上述 1.4 步骤
openai.api_base = "<Azure OpenAI 服务资源的终结点>"
# 设置 API 类型为 Azure
openai.api_type = 'azure'
# 设置 API 的版本号, 此版本号可能会在未来更改
openai.api_version = '2022-12-01'

# 获取可支持的模型列表
response = openai.Model.list()
# res = json.loads(response)
# print(json.dumps(res, indent=4))
for i in response["data"]:
    print(i["id"])

```

示例:

```

# _ _ coding: utf-8 _ _

import openai
from secret import AOAI_KEY, AOAI_ENDPOINT

# 设置Azure OpenAI的API key , 可参考上述1.4步骤
openai.api_key = AOAI_KEY
# 设置Azure OpenAI的API 终结点 , 可参考上述1.4步骤
openai.api_base = AOAI_ENDPOINT
# 设置API类型为Azure
openai.api_type = 'azure'
# 设置API的版本号, 此版本号可能会在未来更改
openai.api_version = '2022-12-01'

# 获取可支持的模型列表
response = openai.Model.list()
for i in response["data"]:
    print(i["id"])

```

```

ada
babbage
curie
davinci
text-ada-001
text-babbage-001
text-curie-001
text-davinci-001
text-davinci-002
text-davinci-003
code-cushman-001
code-cushman-fine-tune-002
text-similarity-ada-001
text-similarity-babbage-001
text-similarity-curie-001
text-similarity-davinci-001
text-search-ada-doc-001
text-search-ada-query-001
text-search-babbage-doc-001
text-search-babbage-query-001
text-search-curie-doc-001
text-search-curie-query-001
text-search-davinci-doc-001
text-search-davinci-query-001
code-search-ada-code-001
code-search-ada-text-001
code-search-babbage-code-001
code-search-babbage-text-001
text-embedding-ada-002

```

### 6.2.1.2 获取已部署的模型清单

```
# *_ coding: utf-8 *_

import json
import openai

# 设置 Azure OpenAI 的 API key , 可参考上述 1.4 步骤
openai.api_key = "<Azure OpenAI 服务资源的密钥>"
# 设置 Azure OpenAI 的 API 终结点 , 可参考上述 1.4 步骤
openai.api_base = "<Azure OpenAI 服务资源的终结点>"
# 设置 API 类型为 Azure
openai.api_type = 'azure'
# 设置 API 的版本号, 此版本号可能会在未来更改
openai.api_version = '2022-12-01' # this may change in the future

# 获取已部署的模型
response = openai.Deployment.list()
for each in response["data"]:
    print(f"部署 ID: {each['id']}, 基模型: {each['model']}")
```

示例:

```
# *_ coding: utf-8 *_

import openai
from secret import AOAI_KEY, AOAI_ENDPOINT

# 设置Azure OpenAI的API key , 可参考上述1.4步骤
openai.api_key = AOAI_KEY
# 设置Azure OpenAI的API 终结点 , 可参考上述1.4步骤
openai.api_base = AOAI_ENDPOINT
# 设置API类型为Azure
openai.api_type = 'azure'
# 设置API的版本号, 此版本号可能会在未来更改
openai.api_version = '2022-12-01'

# 获取已部署的模型
response = openai.Deployment.list()
for each in response["data"]:
    print(f"部署ID: {each['id']}, 基模型: {each['model']}")
```

azure\python-for-ai\openai\_text.py  
部署ID: txt-davinci003-1, 基模型: text-davinci-003  
部署ID: txt-curie001-1, 基模型: text-curie-001  
PS C:\MSWORK\MyCode>

## 6.2.2 GPT-3

### 6.2.2.1 单次对话

```
# *_ coding: utf-8 *_

# 先安装 python 模块
#pip install openai num2words matplotlib plotly scipy scikit-learn transformers

# 导入需要用到的模块
import os
import json
```

```
import openai

# 设置 Azure OpenAI 的 API key , 可参考上述 1.4 步骤
openai.api_key = "<Azure OpenAI 服务资源的密钥>"
# 设置 Azure OpenAI 的 API 终结点 , 可参考上述 1.4 步骤
openai.api_base = "<Azure OpenAI 服务资源的终结点>"
# 设置 API 类型为 Azure
openai.api_type = 'azure'
# 设置 API 的版本号, 此版本号可能会在未来更改
openai.api_version = '2022-12-01'
# 设置部署名称, 参考上述 1.3 步骤
deployment_name = "<Azure OpenAI 服务资源下自定义的模型的名称>"

# 定义输入的 Prompt 文本的内容
input_context = 'Write a tagline for an ice cream shop. '
#print("Ask: ", input_context)

# 发送完成请求以生成答案
response = openai.Completion.create(
    engine=deployment_name,
    prompt=input_context, # 定义输入的 Prompt 文本的内容
    temperature=0,
    frequency_penalty=0,
    presence_penalty=0,
    max_tokens=10
) # 创建完成请求, 设置模型、输入、参数等, 可参考上述 6.1.1 步骤

# 从响应中获取生成的文本内容
text = response['choices'][-1]['text'].strip()
print("Ans: ", text)
```

示例：

```
# -*- coding: utf-8 -*-

# 先安装python模块
#pip install openai num2words matplotlib plotly scipy scikit-learn

# 导入需要用到的模块
import openai
from secret import AOAI_KEY, AOAI_ENDPOINT, AOAI_DEPLOYMENT_ID

# 设置Azure OpenAI的API key，可参考上述1.4步骤
openai.api_key = AOAI_KEY
# 设置Azure OpenAI的API 终结点，可参考上述1.4步骤
openai.api_base = AOAI_ENDPOINT
# 设置API类型为Azure
openai.api_type = 'azure'
# 设置API的版本号，此版本号可能会在未来更改
openai.api_version = '2022-12-01'
# 设置部署名称，参考上述1.3步骤
deployment_name = AOAI_DEPLOYMENT_ID

# 定义输入的Prompt文本的内容
input_context = 'OpenAI与微软的合作时长'
print("Ask: ", input_context)

response = openai.Completion.create(
    engine=deployment_name,
    prompt=input_context, # 定义输入的Prompt文本的内容
    temperature=0,
    frequency_penalty=0,
    presence_penalty=0,
    max_tokens=100
) # 创建完成请求，设置模型、输入、参数等，可参考上述6.1.1步骤

# 从响应中获取生成的文本内容
text = response['choices'][-1]['text'].strip()
print("Ans: ", text)
```

Ask: OpenAI与微软的合作时长  
Ans: OpenAI与微软的合作始于2016年，至今已有4年的合作历史。  
PS C:\MSWORK\MyCode> []

### 6.2.2.2 连续对话

```
# -*- coding: utf-8 -*-

# 先安装 python 模块
#pip install openai num2words matplotlib plotly scipy scikit-learn transformers

# 导入需要用到的模块
import os
import json
import openai

def Azure_test_d003_test_py(query):
    # 设置 Azure OpenAI 的 API key，可参考上述 1.4 步骤
    openai.api_key = "<Azure OpenAI 服务资源的密钥>"
    # 设置 Azure OpenAI 的 API 终结点，可参考上述 1.4 步骤
    openai.api_base = "<Azure OpenAI 服务资源的终结点>"
    # 设置 API 类型为 Azure
    openai.api_type = 'azure'
    # 设置 API 的版本号，此版本号可能会在未来更改
    openai.api_version = '2022-12-01'
    # 设置部署名称，参考上述 1.3 步骤
```

```

deployment_name = "<Azure OpenAI 服务资源下自定义的模型的名称>"
response = openai.Completion.create(
    engine=deployment_name,
    prompt= ,
    temperature=0,
    frequency_penalty=0,
    presence_penalty=0,
    max_tokens=100
)
text = response['choices'][-1]['text'].strip()
# print("Ans: ", text)
return response

def Azure_test_d003_test_py_batch():
    exit_conditions = ("quit", "exit")

    coversation = []

    while True:
        user_input = input("Ask: ")
        if user_input not in exit_conditions:
            coversation.append("Ask: " + user_input)
            context = "\n".join(coversation)
            """
            context 的内容
            context = "Ask: 写一句赞美中国的七言诗\n
Ans: 春风千里共芳菲， 中华气象万千秋。 \n
Ask: 再来一句\n
Ans: Ans: 山川河流奔流淌， 桃李春风正芳芬。 "
            """
            response = Azure_test_d003_test_py(context)
            # 打印所有的返回
            # print(json.dumps(response.to_dict(), ensure_ascii = False, indent=4))
            res_text = str(response.to_dict()["choices"][0]["text"]).strip()
            coversation.append("Ans: " + res_text)
            print("Ans: ", res_text)
            print("\n")
        else:
            break

if __name__ == '__main__':
    Azure_test_d003_test_py_batch()

```

示例：

```
python for ai 7 azure_openai_text.py 7 ...
prompt=query,
temperature=0,
frequency_penalty=0,
presence_penalty=0,
max_tokens=100
)
text = response["choices"][-1]["text"].strip()
# print("Ans: ", text)
return response

def Azure_test_d003_test_py_batch():
    exit_conditions = ("quit", "exit")

    conversation = []

    while True:
        user_input = input("Ask: ")
        if user_input not in exit_conditions:
            conversation.append("Ask: " + user_input)
            context = "\n".join(conversation)
            # print(context)
            response = Azure_test_d003_test_py(context)
            # print(json.dumps(response.to_dict(), ensure_ascii = False, indent=2))
            res_text = str(response.to_dict()["choices"][-1]["text"]).strip()
            conversation.append("Ans: " + res_text)
            print("Ans: ", res_text.lstrip('\n\n').lstrip("Ans: ").strip())
            print("\n")
        else:
            break

if __name__ == '__main__':
    Azure_test_d003_test_py_batch()
# Azure_test_d003_test_py("写一句赞美中国的七言诗")

Ask: 写一句赞美中国的七言诗
Ans: 春风千里共芳菲，中华气象万千秋。

Ask: 再写一句
Ans: 山川河流恩深情，满目芳草满国家。

Ask: 两句连起来
Ans: 春风千里共芳菲，中华气象万千秋；山川河流恩深情，满目芳草满国家。

Ask: exit
PS C:\WORK\MyCode>
```

### 6.2.2.3 GPT-3 与认知服务语音服务、语言服务结合使用

```
# _*_ coding: utf-8 _*_

# 安装包
# pip install openai num2words matplotlib plotly scipy scikit-learn transformers
# pip install azure-cognitiveservices-speech

import os
import json
import openai
import azure.cognitiveservices.speech as speechsdk
from datetime import datetime

def text_d003_test_one(query):
    # 设置 Azure OpenAI 的 API key，可参考上述 1.4 步骤
    openai.api_key = "<Azure OpenAI 服务资源的密钥>"
    # 设置 Azure OpenAI 的 API 终结点，可参考上述 1.4 步骤
    openai.api_base = "<Azure OpenAI 服务资源的终结点>"
    # 设置 API 类型为 Azure
    openai.api_type = 'azure'
    # 设置 API 的版本号，此版本号可能会在未来更改
    openai.api_version = '2022-12-01'
    # 设置部署名称，参考上述 1.3 步骤
    deployment_name = "<Azure OpenAI 服务资源下自定义的模型的名称>"
    response = openai.Completion.create(
        engine=deployment_name,
        prompt=query,
        temperature=0.1,
```

```

max_tokens=200,
top_p=1,
frequency_penalty=0,
presence_penalty=0,
# logprobs=0,
stop=None
)
# print(json.dumps(response.to_dict(), ensure_ascii = False, indent=4))
res_text = response.to_dict()["choices"][-1]["text"]
print("Ans: ", res_text)
return response

# 短时实时语音识别 默认 15s 时长
def once_recognize_from_mic():
    # 设置语音服务的订阅信息，配置密钥、区域
    COGNITIVE_KEY = "<Azure 认知服务的密钥>"
    COGNITIVE_REGION = "<Azure 认知服务的区域>"
    speech_config = speechsdk.SpeechConfig(subscription=COGNITIVE_KEY,
region=COGNITIVE_REGION)
    # 支持的语言 https://learn.microsoft.com/zh-cn/azure/cognitive-services/speech-service/language-support?tabs=stt
    speech_config.speech_recognition_language="zh-CN"
    # 有不恰当的内容删除掉
    speech_config.set_profanity(speechsdk.ProfanityOption.Removed)

    # 使用麦克风输入音频
    audio_config = speechsdk.audio.AudioConfig(use_default_microphone=True)
    # 从文件输入音频
    # audio_config = speechsdk.audio.AudioConfig(filename="10086-2302072138.wav")

    # 创建一个使用麦克风作为音频输入的语音识别器
    speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config,
audio_config=audio_config)

    print("Speak into your microphone: ")
    # 识别一次，长时间识别使用 start_continuous_recognition()
    speech_recognition_result = speech_recognizer.recognize_once()

    out_text = ""
    if speech_recognition_result.reason == speechsdk.ResultReason.RecognizedSpeech:
        out_text = speech_recognition_result.text
        print("Recognized: {}".format(out_text))
    # 15s 内未识别到语音
    elif speech_recognition_result.reason == speechsdk.ResultReason.NoMatch:
        print("No speech could be recognized:
{}".format(speech_recognition_result.no_match_details))
    elif speech_recognition_result.reason == speechsdk.ResultReason.Canceled:

```



```

        cancellation_details = speech_recognition_result.cancellation_details
        print("Speech Recognition canceled: {}".format(cancellation_details.reason))
        if cancellation_details.reason == speechsdk.CancellationReason.Error:
            print("Error details: {}".format(cancellation_details.error_details))
            print("Did you set the speech resource key and region values?")
        return out_text

def synthesizer_VoiceToText_out_speaker(t_text):
    # 设置语音服务的订阅信息，配置密钥、区域
    speech_config = speechsdk.SpeechConfig(subscription=COGNITIVE_KEY,
        region=COGNITIVE_REGION)
    speech_config.speech_synthesis_language = "zh-CN"
    # 受支持的声音 https://learn.microsoft.com/zh-cn/azure/cognitive-services/speech-service/language-support?tabs=tts#prebuilt-neural-voices
    speech_config.speech_synthesis_voice_name = "zh-CN-XiaomengNeural"

    speech_config.set_profanity(speechsdk.ProfanityOption.Removed)

    # print("Enter some text that you want to speak >")
    # text = input()

    # 设置读取文本后使用扬声器播放音频 - begin
    audio_config = speechsdk.audio.AudioOutputConfig(use_default_speaker=True)
    # 实例化一个语音合成器，
    speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config,
        audio_config=audio_config)
    # 设置输入文本
    speech_synthesis_result = speech_synthesizer.speak_text_async(t_text).get()
    # 设置读取文本后使用扬声器播放音频 - end

    # 将读取到文本合成音频文件 - begin
    now = datetime.now() # 获取当前时间
    current_time = now.strftime("%Y%m%d%H%M%S") # 将时间格式化为字符串
    audio_config = speechsdk.audio.AudioOutputConfig(filename='files/'+current_time +
        '.wav')
    # 实例化一个语音合成器，
    speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config,
        audio_config=audio_config)
    # 输入文本，执行语音合成
    speech_synthesis_result = speech_synthesizer.speak_text_async(t_text).get()
    # 将读取到文本合成音频文件 - end

    if speech_synthesis_result.reason ==
speechsdk.ResultReason.SynthesizingAudioCompleted:
        print("Speech synthesized for text [ {} ]".format(t_text))
    elif speech_synthesis_result.reason == speechsdk.ResultReason.Canceled:

```

```

cancellation_details = speech_synthesis_result.cancellation_details
print("Speech synthesis canceled: {}".format(cancellation_details.reason))
if cancellation_details.reason == speechsdk.CancellationReason.Error:
    if cancellation_details.error_details:
        print("Error details: {}".format(cancellation_details.error_details))
        print("Did you set the speech resource key and region values?")

if __name__ == '__main__':
    print("\n 一. 语音转文本，从麦克风识别语音=====")
    input_content = once_recognize_from_mic()
    # content = "Are you ok?"

    print("\n 二. 调用 GPT-3 生成查询=====")
    print("Ask: ",input_content)
    response = text_d003_test_one(input_content)
    res_text = str(response.to_dict()["choices"][-1]["text"]).strip()

    print("\n 三. 将 GPT-3 生成的结果文本转成语音，用扬声器播放并合成音频文件=====")
    synthesizer_VoiceToText_out_speaker("\n 问: " + input_content + "\n 答: " + res_text)
    print("\n")

```

示例：

<pre> current_time = now.strftime("%Y%m%d%H%M%S") # 将时间格式化为字符串 audio_config = speechsdk.audio.AudioOutputConfig(filename='files/'+current_time + '.wav') # 实例化一个语音合成器。 speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config, audio_config=audio_config) # 输入文本，执行语音合成 speech_synthesis_result = speech_synthesizer.speak_text_async(t_text).get() # 将读取到文本合成音频文件 - end  if speech_synthesis_result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:     # print("Speech synthesized for text [ {} ]".format(t_text))     print("Speech synthesized for text [ {} ]".format(t_text)) elif speech_synthesis_result.reason == speechsdk.ResultReason.Canceled:     cancellation_details = speech_synthesis_result.cancellation_details     print("Speech synthesis canceled: {}".format(cancellation_details.reason))     if cancellation_details.reason == speechsdk.CancellationReason.Error:         if cancellation_details.error_details:             print("Error details: {}".format(cancellation_details.error_details))             print("Did you set the speech resource key and region values?")  if __name__ == '__main__':     # content = input("Ask: ")     # input_content = "写一句赞美中国的七言诗"     print("\n 一. 语音转文本，从麦克风识别语音=====")     input_content = once_recognize_from_mic()     # content = "Are you ok?"      print("\n 二. 调用GPT-3 生成查询=====")     print("Ask: ",input_content)     response = text_d003_test_one(input_content)     res_text = str(response.to_dict()["choices"][-1]["text"]).strip()      print("\n 三. 将GPT-3生成的结果文本转成语音，用扬声器播放并合成音频文件=====")     synthesizer_VoiceToText_out_speaker("\n 问: " + input_content + "\n 答: " + res_text)     print("\n") </pre>	<pre> python-for-ai\my_openal_and_cogitveservice.py 一. 语音转文本，从麦克风识别语音===== Speak into your microphone: Recognized: 把大象装冰箱总共分几步？  二. 调用GPT-3 生成查询===== Ask: 把大象装冰箱总共分几步？ Ans:  一般来说，把大象装冰箱的步骤大致可以分为以下几步：  1. 准备冰箱：打开冰箱门，清理内部，检查冰箱的尺寸，确保大象可以容纳其中。 2. 准备大象：给大象洗澡，把它的毛发剪短，给它涂上抗菌剂，以防止细菌滋生。 3. 把大象放入冰箱：把大象放入冰箱，确保它的头部和身体都能完全放入冰箱。 4. 关闭冰箱门：关闭冰箱门，确保大象完全被封闭在冰箱里。 5. 加入冰块：把冰块放入冰箱，确保大象能够得到足够的冷却。 6. 加入水：把水放入冰箱，确保大象能够得到足够的水分。  三. 将GPT-3生成的结果文本转成语音，用扬声器播放===== Speech synthesized for text [ 问: 把大象装冰箱总共分几步？ 答: 一般来说，把大象装冰箱的步骤大致可以分为以下几步：  1. 准备冰箱：打开冰箱门，清理内部，检查冰箱的尺寸，确保大象可以容纳其中。 2. 准备大象：给大象洗澡，把它的毛发剪短，给它涂上抗菌剂，以防止细菌滋生。 3. 把大象放入冰箱：把大象放入冰箱，确保它的头部和身体都能完全放入冰箱。 4. 关闭冰箱门：关闭冰箱门，确保大象完全被封闭在冰箱里。 5. 加入冰块：把冰块放入冰箱，确保大象能够得到足够的冷却。 6. 加入水：把水放入冰箱，确保大象能够得到足够的水分。 ] </pre>
---	---

### 6.2.3 Codex

### 6.2.4 Embeddings

### 6.2.5 DELL-E

目前 Azure 上的 DELL-E 产品还没有上线，以下是 OpenAI 的 api 调用

```
# *_ coding: utf-8 *_

import json
import requests
import openai
from datetime import datetime


openai.api_key = "<OpenAI 的 API Key>"
input_content = "一只粉色的狐狸"
response = openai.Image.create(
    prompt=input_content,
    n=1, # 返回给用户多少张图片
    size="1024x1024" # 生成的照片大小
)
# 打印所有的返回
#print(json.dumps(response.to_dict(), ensure_ascii = False, indent=4))

for i in response.to_dict()["data"]:
    image_url = i["url"] #返回的图片是一个 azure blob 链接，可直接浏览器访问

    # 可将图片保存到本地
    img_data = requests.get(url=image_url).content
    now = datetime.now() # 获取当前时间
    current_time = now.strftime("%Y%m%d%H%M%S") # 将时间格式化为字符串
    with open("files/"+current_time+".png", 'wb') as handler:
        handler.write(img_data)
```

示例：

```
1 # -*- coding: utf-8 -*-
2
3 import requests,json
4 import openai
5 from datetime import datetime
6 from secret import OPENAI_API_KEY
7
8 openai.api_key = OPENAI_API_KEY
9 input_content = "一只长着兔耳朵的粉色的狐狸，正在天上飞"
10 response = openai.Image.create(
11     prompt=input_content,
12     n=1, # 返回给用户多少张图片
13     size="1024x1024" # 生成的照片大小
14 )
15 # 打印所有的返回
16 print(json.dumps(response.to_dict(), ensure_ascii = False, indent=4))
17
18 for i in response.to_dict()["data"]:
19     image_url = i["url"] #返回的图片是一个azure blob链接，可直接
20
21     # 可将图片保存到本地
22     img_data = requests.get(url=image_url).content
23     now = datetime.now() # 获取当前时间
24     current_time = now.strftime("%Y%m%d%H%M%S") # 将时间格式化
25     with open("files/"+current_time+".png", 'wb') as handler:
26         handler.write(img_data)
27
```



```
y"
{
  "created": 1678247258,
  "data": [
    {
      "url": "https://oaidalleapiprodscus.blob.core.windows.net/private/org-X18KdkRA9bt1loq9KJZ1UIUJ/user-PNOU6Nj1U6DC6UpkbuVEXqFJ/img-8YncnutZN3EvclP79FHYhGPv.png?st=2023-03-08T02%3A47%3A38Z&se=2023-03-08T04%3A47%3A38Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rsc=image/png&skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&skt=2023-03-07T21%3A44%3A21Z&ske=2023-03-08T21%3A44%3A21Z&skv=b&skv=2021-08-06&sig=u65cqr9%2BcznJGVZARObolQwodXJ6ukmoCSarAjNYSQg%3D"
    }
  ]
}
```

## 6.2.6 Whisper

## 6.2.7 ChatGPT

```
# -*- coding: UTF-8 -*-
import openai

# 设置 Azure OpenAI 的 API key，可参考上述 1.4 步骤
openai.api_key = "<Azure OpenAI 服务资源的密钥>"
# 设置 Azure OpenAI 的 API 终结点，可参考上述 1.4 步骤
openai.api_base = "<Azure OpenAI 服务资源的终结点>"
# 设置 API 类型为 Azure
openai.api_type = 'azure'
# 设置 API 的版本号，此版本号可能会在未来更改
```

```

openai.api_version = '2022-12-01'
# 设置部署名称, 参考上述 1.3 步骤
deployment_name = "<Azure OpenAI 服务资源下自定义的模型的名称>"
input_content = "OpenAI 与微软的合作历程, 按时间倒序列出"

sys_message = """<|im_start|>system
Provide some context and/or instructions to the model.
<|im_end|>
"""

user_message = """<|im_end|>
<|im_start|>user
""" + input_content + """
<|im_end|>
"""

assistant_message = """<|im_start|>assistant"""

prompt_context = sys_message + user_message + assistant_message
response = openai.Completion.create(
    engine=deployment_name,
    prompt=prompt_context,
    temperature=0.7,
    max_tokens=800,
    top_p=0.95,
    stop=["<|im_end|>"]
)
print(response['choices'][0]['text'])

#聊天标记语言 ChatML

```

示例:

```

6 from secret import AOAI_KEY, AOAI_ENDPOINT, AOAI_DEPLOYMENT_CHAT_ID
7
8 def Azure_text_chat_test_one_py(query):
9     openai.api_key = AOAI_KEY
10    openai.api_base = AOAI_ENDPOINT
11    openai.api_type = "azure"
12    openai.api_version = "2022-12-01"
13    """ChatML聊天标记语言
14    <|im_start|>system
15    Provide some context and/or instructions to the model.
16    <|im_end|>
17    <|im_start|>user
18    The user's message goes here
19    <|im_end|>
20    <|im_start|>assistant
21    """
22    sys_message = """<|im_start|>system
23    Provide some context and/or instructions to the model.
24    <|im_end|>
25    """
26    user_message = """<|im_end|>
27    <|im_start|>user
28    """ + query + """
29    <|im_end|>
30    """
31    assistant_message = """<|im_start|>assistant"""
32
33    prompt_context = sys_message + user_message + assistant_message
34    response = openai.Completion.create(
35        engine=AOAI_DEPLOYMENT_CHAT_ID,
36        prompt=prompt_context,
37        temperature=0.7,
38        max_tokens=800,
39        top_p=0.95,
40        stop=["<|im_end|>"]
41    )
42    print(response['choices'][0]['text'])
43
44 if __name__ == '__main__':
45     # input_content = input("Ask: ")
46     input_content = "OpenAI与微软的合作历程,按时间倒序列出"
47     Azure_text_chat_test_one_py(input_content)
48

```

以下是OpenAI与微软的合作历程，按时间倒序列出：

1. 2020年9月 - OpenAI与微软合作开发新一代语言模型GPT-3，并将其作为Azure云服务的一部分推出。
2. 2019年5月 - OpenAI宣布与微软扩大合作，将使用Azure云服务来开发和部署其AI技术。
3. 2018年6月 - 微软投资OpenAI 10亿美元，并宣布双方将在AI研究和开发方面进行合作。
4. 2017年11月 - OpenAI与微软合作推出了OpenAI Gym，这是一个用于开发和比较强化学习算法的平台。
5. 2016年12月 - 微软宣布向OpenAI提供1亿美元的云服务，以帮助其研究开发人工智能技术。

以上就是OpenAI与微软的合作历程，按时间倒序列出。

○ PS C:\VSMWORK\MyCode> █

## 6.3 通过 REST API 调用

### 6.3.1 获取可用模型、部署信息

#### 6.3.1.1 获取所有模型列表清单

```

# *_ coding: utf-8 *_

import requests,json

url = "{aoai_res_name}openai/models?api-version={api_version}".format(
    aoai_res_name = "<Azure OpenAI 服务资源的终结点>", # Azure 上认知服务或 Azure
    OpenAI 服务的站点
    api_version="2022-12-01" # api 的版本
)

headers = {"api-key": "<Azure OpenAI 服务资源的密钥>", # Azure 上认知服务或 Azure
    OpenAI 服务资源的密钥
    "Content-Type": "Application/json"
}

# 发送完成请求以生成答案

```

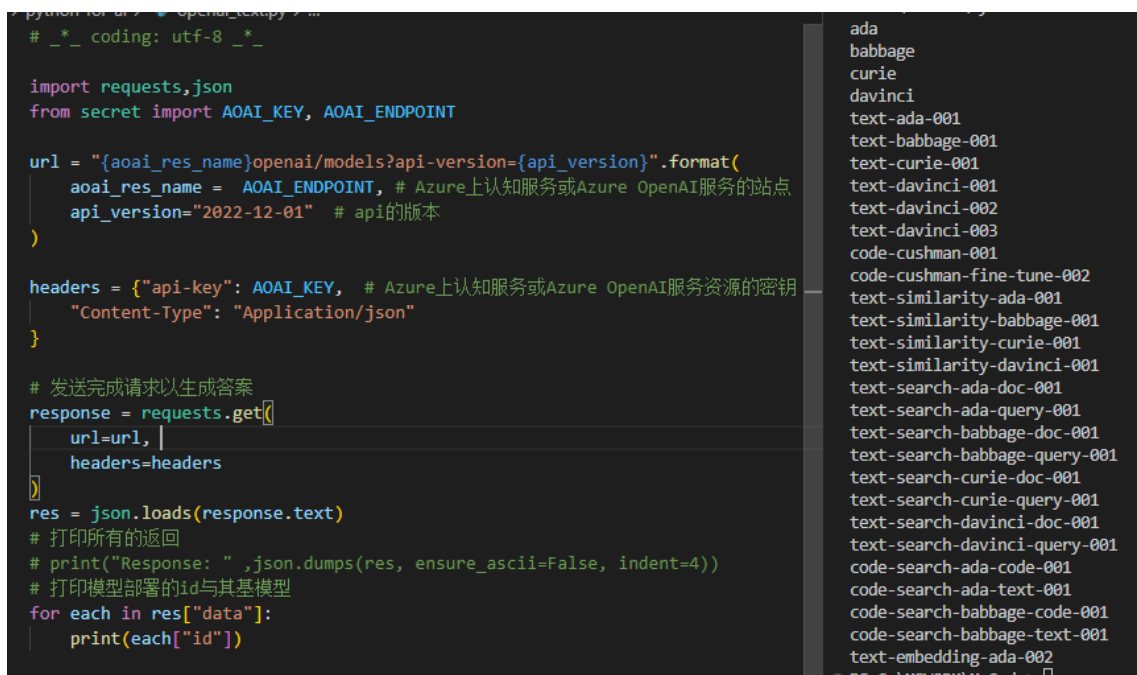
```

response = requests.get(
    url=url,
    headers=headers
)
res = json.loads(response.text)
# 打印所有的返回
# print("Response: ", json.dumps(res, ensure_ascii=False, indent=4))
# 打印模型 id
for each in res["data"]:
    print(each["id"])

```

#<[参考文档](#)>

示例



```

# -*- coding: utf-8 -*-

import requests, json
from secret import AOAI_KEY, AOAI_ENDPOINT

url = "{aoai_res_name}openai/models?api-version={api_version}".format(
    aoai_res_name = AOAI_ENDPOINT, # Azure上认知服务或Azure OpenAI服务的站点
    api_version="2022-12-01" # api的版本
)

headers = {"api-key": AOAI_KEY, # Azure上认知服务或Azure OpenAI服务资源的密钥
           "Content-Type": "Application/json"}

# 发送完成请求以生成答案
response = requests.get(
    url=url,
    headers=headers
)
res = json.loads(response.text)
# 打印所有的返回
# print("Response: ", json.dumps(res, ensure_ascii=False, indent=4))
# 打印模型部署的id与基模型
for each in res["data"]:
    print(each["id"])

```

ada  
babbage  
curie  
davinci  
text-ada-001  
text-babbage-001  
text-curie-001  
text-davinci-001  
text-davinci-002  
text-davinci-003  
code-cushman-001  
code-cushman-fine-tune-002  
text-similarity-ada-001  
text-similarity-babbage-001  
text-similarity-curie-001  
text-similarity-davinci-001  
text-search-ada-doc-001  
text-search-ada-query-001  
text-search-babbage-doc-001  
text-search-babbage-query-001  
text-search-curie-doc-001  
text-search-curie-query-001  
text-search-davinci-doc-001  
text-search-davinci-query-001  
code-search-ada-code-001  
code-search-ada-text-001  
code-search-babbage-code-001  
code-search-babbage-text-001  
text-embedding-ada-002

### 6.3.1.2 获取已部署的模型清单

```

# -*- coding: utf-8 -*-

import json
import requests

url = "{openai_resource_name}openai/deployments?api-version={api_version}".format(
    openai_resource_name = "<Azure OpenAI 服务资源的终结点>", # Azure 上认知服务或
    Azure OpenAI 服务的站点
    api_version="2022-12-01" # api 的版本
)

```

```

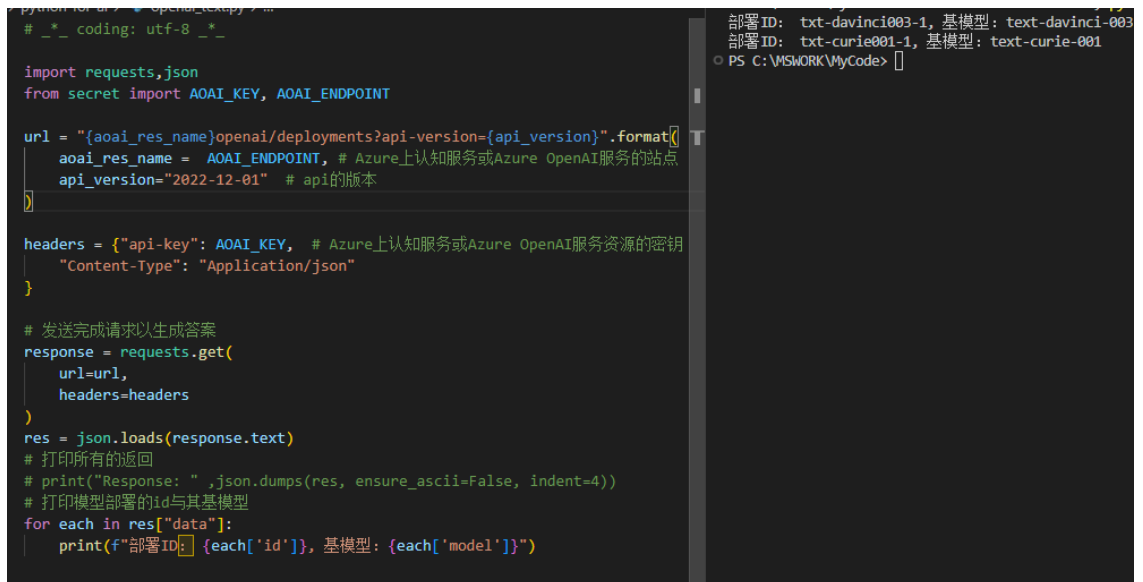
headers = {"api-key": "<Azure OpenAI 服务资源的密钥>", # Azure 上认知服务或 Azure
OpenAI 服务资源的密钥
    "Content-Type": "Application/json"
}

# 发送完成请求以生成答案
response = requests.get(
    url=url,
    headers=headers
)
res = json.loads(response.text)
# 打印所有的返回
# print("Response: ", json.dumps(res, ensure_ascii=False, indent=4))
# 打印模型部署的 id 与其基模型
for each in res["data"]:
    print(f"部署 ID: {each['id']}, 基模型: {each['model']}")

#<参考文档>

```

示例：



```

python for ai / openai_text.py / m
# -*- coding: utf-8 -*-

import requests, json
from secret import AOAI_KEY, AOAI_ENDPOINT

url = "{aoai_res_name}openai/deployments?api-version={api_version}".format(
    aoai_res_name = AOAI_ENDPOINT, # Azure上认知服务或Azure OpenAI服务的站点
    api_version="2022-12-01" # api的版本
)

headers = {"api-key": AOAI_KEY, # Azure上认知服务或Azure OpenAI服务资源的密钥
    "Content-Type": "Application/json"
}

# 发送完成请求以生成答案
response = requests.get(
    url=url,
    headers=headers
)
res = json.loads(response.text)
# 打印所有的返回
# print("Response: ", json.dumps(res, ensure_ascii=False, indent=4))
# 打印模型部署的id与其基模型
for each in res["data"]:
    print(f"部署ID: {each['id']}, 基模型: {each['model']}")

部署ID: txt-davinci003-1, 基模型: text-davinci-003
部署ID: txt-curie001-1, 基模型: text-curie-001
PS C:\MSWORK\MyCode>

```

### 6.3.2 GPT-3

```

# -*- coding: utf-8 -*-

# 导入需要用到的模块
import json
import requests

url = "{openai_resource_name}openai/deployments/{deploymen_id}/completions?api-
version={api_version}".format(

```



```

    openai_resource_name="<Azure OpenAI 服务资源的终结点>", # Azure 上认知服务或
    Azure OpenAI 服务的站点
    deploymen_id="<Azure OpenAI 服务资源下部署的模型的名称>", # Azure 上部署的模
    型的名称
    api_version="2022-12-01" # api 的版本
)

headers = {"api-key": "<Azure OpenAI 服务资源的密钥>", # Azure 上认知服务或 Azure
OpenAI 服务资源的密钥
    "Content-Type": "Application/json"
}
data = {
    "prompt": "微软与 OpenAI 的合作时长", # 定义输入的 Prompt 文本的内容
    "temperature": 1,
    "max_tokens": 200, # 模型最大生成的文本长度, 输入 + 输出
    "top_p": 1,
    "frequency_penalty": 0, # 减少生成文本中重复词语出现的概率
    "presence_penalty": 0 # 减少生成文本中重复片段出现的概率
}

# 发送完成请求以生成答案
response = requests.post(
    url=url,
    data=json.dumps(data, ensure_ascii=False).encode(encoding="UTF-8"),
    headers=headers
)
# print(response)
# 打印所有的返回
res = json.loads(response.text)
print("Response: ", json.dumps(res, ensure_ascii=False, indent=4))
# 打印生成的答案
text= str(json.dumps(res["choices"][-1]["text"], ensure_ascii=False, indent=4)).strip()
print("\nAns: " , text )

```

示例：

```
1 # -*- coding: utf-8 -*-
2
3 # 先安装python模块
4 #pip install openai num2words matplotlib plotly scipy scikit-learn transformers
5
6 # 导入需要用的模块
7 import json
8 import openai
9 import requests
10 from secret import AOAI_KEY, AOAI_ENDPOINT, AOAI_DEPLOYMENT_ID
11
12 url = "(your_resource_name)openai/deployments/{deployment_id}/completions?api-version={api_version}"
13 your_resource_name=AOAI_ENDPOINT, # Azure上认知服务或Azure OpenAI服务的站点
14 deployment_id=AOAI_DEPLOYMENT_ID, # Azure上部署的模型名称
15 api_version="2022-12-01" # api的版本
16
17
18 headers = {"api-key": AOAI_KEY, # Azure上认知服务或Azure OpenAI服务资源的密钥
19           "Content-Type": "Application/json"}
20
21 data = {
22     "prompt": "微软与OpenAI的合作时长", # 定义输入的Prompt文本的内容
23     "temperature": 1,
24     "max_tokens": 200, # 模型最大生成的文本长度, 输入 + 输出
25     "top_p": 1,
26     "frequency_penalty": 0, # 减少生成文本中重复词语出现的概率
27     "presence_penalty": 0 # 减少生成文本中重复片段出现的概率
28 }
29
30 # 发送完成请求以生成答案
31 response = requests.post(
32     url=url,
33     data=json.dumps(data, ensure_ascii=False).encode(encoding="UTF-8"),
34     headers=headers
35 )
36 # print(response)
37
38 res = json.loads(response.text)
39 print("Response: ", json.dumps(res, ensure_ascii=False, indent=4))
40
41 text= str(json.dumps(res["choices"][-1]["text"], ensure_ascii=False, indent=4)).
42 print("\nAns: ", text )
```

```
on-ai\openai_text.py
Response: {
  "id": "cmpl-6rJlKvH8p354VullUctkUGtrpsb",
  "object": "text_completion",
  "created": 1678165396,
  "model": "text-davinci-003",
  "choices": [
    {
      "text": "\n\n微软和OpenAI之间的合作始于2018年7月，之后的合作一直持续到2020年9月，整个合作持续了大约2年半的时间。", 返回的文本的内容
      "index": 0,
      "finish_reason": "stop",
      "logprobs": null
    }
  ],
  "usage": {
    "completion_tokens": 89, 返回的文本消耗的token
    "prompt_tokens": 18, 输入的文本消耗的token
    "total_tokens": 107 输入 (prompt) + 输出 (completion) 总共消耗的token
  }
}
Ans: "\n\n微软和OpenAI之间的合作始于2018年7月，之后的合作一直持续到2020年9月，整个合作持续了大约2年半的时间。"
PS C:\VISMORR\VlyCode>
```

### 6.3.3 Codex

### 6.3.4 Embeddings

### 6.3.5 DELL-E

目前 Azure 上的 DELL-E 产品还没有上线，以下是 OpenAI 的 api 调用

```
# -*- coding: utf-8 -*-

import json
import requests
from datetime import datetime

url = "https://api.openai.com/v1/images/generations"

headers = {
    "Content-Type": "application/json",
    "Authorization": "Bearer <OpenAI 的 API Key>"
}

input_content = "一只粉色的狐狸和一只傻狗哈士奇"
data = {
    "prompt": input_content,
    "n": 2,
    "size": "1024x1024"
```

```

}
response = requests.post(url=url,
    data=json.dumps(data, ensure_ascii=False).encode(encoding="UTF-8"),
    headers=headers
)

res = json.loads(response.text)
# 打印所有的返回
# print(json.dumps(res, ensure_ascii = False, indent=4))

for i in res["data"]:
    image_url = i["url"] #返回的图片是一个 azure blob 链接，可直接浏览器访问
    # 可将图片保存到本地
    img_data = requests.get(url=image_url).content
    now = datetime.now() # 获取当前时间
    current_time = now.strftime("%Y%m%d%H%M%S") # 将时间格式化为字符串
    with open("files/"+current_time+".png", 'wb') as handler:
        handler.write(img_data)

```

### 6.3.6 Whisper

### 6.3.7 ChatGPT

示例：

## 6.4 使用 POSTMAN 调用（也可[网页访问](#)）

### 6.4.1 获取可用模型、部署信息

#### 6.4.1.1 获取所有模型列表清单

①请求方式： GET

②请求的 URL： {{AOAI\_ENDPOINT}}openai/models?api-version=2022-12-01  
 (AOAI\_ENDPOINT 是 Azure OpenAI 资源的站点，参考上述步骤 1.4，类似于  
<https://xxx01.openai.azure.com/>)

③Headers： api-key : {{AOAI\_KEY}} (AOAI\_KEY 是 Azure OpenAI 资源的密钥，参考上述步骤 1.4)

④Pre-request Script：  
 pm.environment.set("AOAI\_KEY", "<Azure OpenAI 资源的密钥>")

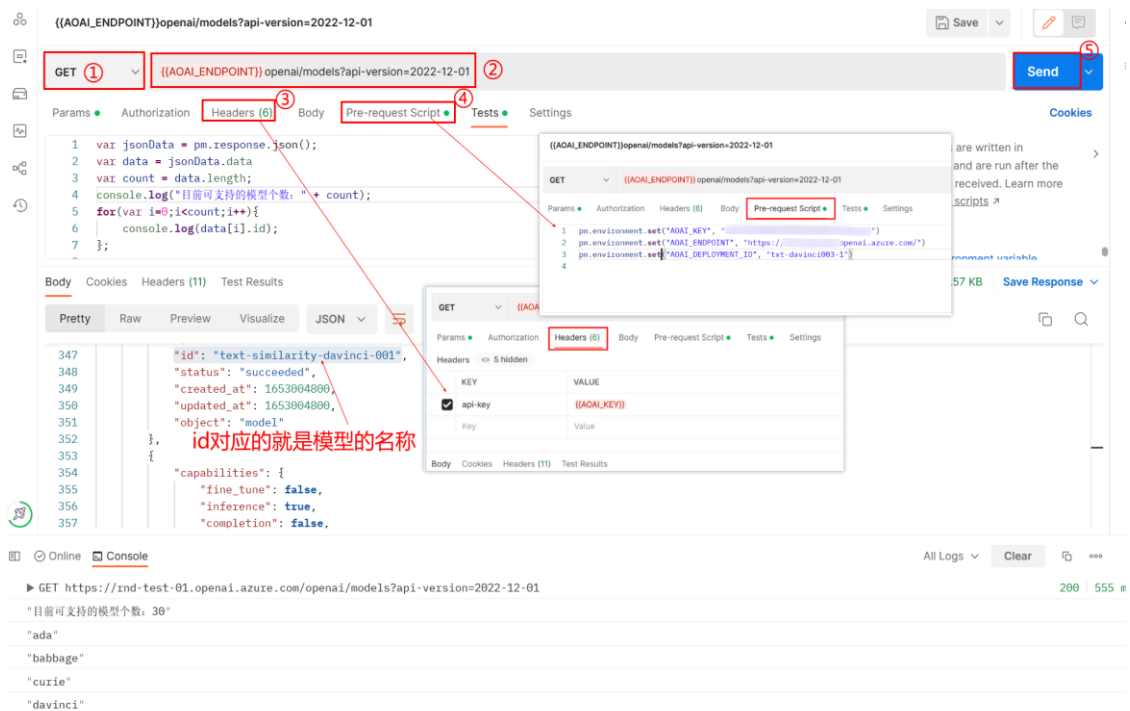
```
pm.environment.set("AOAI_ENDPOINT", "https://xxx01.openai.azure.com/")
pm.environment.set("AOAI_DEPLOYMENT_ID", "<Azure OpenAI 资源内模型的部署名称>")
```

Tests（可选）：

```
var jsonData = pm.response.json();
var data = jsonData.data
var count = data.length;
console.log("目前可支持的模型个数：" + count);
for(var i=0;i<count;i++){
    console.log(data[i].id);
};
```

⑤ send

#<参考文档>



#### 6.4.1.2 获取已部署的模型清单

①请求方式： GET

②请求的 URL： `{{AOAI_ENDPOINT}}openai/deployments?api-version=2022-12-01`  
(AOAI\_ENDPOINT 是 Azure OpenAI 资源的站点，参考上述步骤 1.4，类似于 <https://xxx01.openai.azure.com/>)

③Headers： `api-key`： `{{AOAI_KEY}}` (AOAI\_KEY 是 Azure OpenAI 资源的密钥，参考上述步骤 1.4)

#### ④Pre-request Script:

```
pm.environment.set("AOAI_KEY", "<Azure OpenAI 资源的密钥>")
pm.environment.set("AOAI_ENDPOINT", "https://xxx01.openai.azure.com/")
pm.environment.set("AOAI_DEPLOYMENT_ID", "<Azure OpenAI 资源内模型的部署名称>")
```

Tests (可选):

```
var jsonData = pm.response.json();
var data = jsonData.data
var count = data.length;
console.log("目前已部署的模型个数: " + count);
for(var i=0;i<count;i++){
    console.log(data[i].id);
};
```

#### ⑤ send

#<参考文档>

The screenshot displays the Postman interface for a REST client request. The main request is a GET call to '({{AOAI\_ENDPOINT}})openai/deployments?api-version=2022-12-01'. The 'Pre-request Script' tab is active, showing a JavaScript snippet that logs the number of deployed models and their IDs. The 'Headers' tab shows an 'api-key' header with the value '({{AOAI\_KEY}})'. The 'Body' tab shows a JSON response snippet. The console at the bottom shows the execution results of the request.

Annotations in the image:

- ① GET method
- ② URL: '({{AOAI\_ENDPOINT}})openai/deployments?api-version=2022-12-01'
- ③ Headers tab
- ④ Pre-request Script tab
- ⑤ Send button

JSON response snippet:

```
{
  "scale_type": "standard",
  "model": "text-davinci-003",
  "owner": "organization-owner",
  "id": "text-davinci003-1",
  "status": "succeeded",
  "created_at": 1678023593,
  "updated_at": 1678023593,
  "object": "deployment"
}
```

Console output:

```
GET https://xxx01.openai.azure.com/openai/deployments?api-version=2022-12-01 200
"目前已部署的模型个数: 3"
"text-davinci003-1"
"txt-curie001-1"
"codex-c001-deploy001"
```

## 6.4.2 GPT-3

### ①请求方式: POST

②请求的 URL:

{{AOAI\_ENDPOINT}}openai/deployments/{{AOAI\_DEPLOYMENT\_ID}}/completions?api-version=2022-12-01

(AOAI\_ENDPOINT 是 Azure OpenAI 资源的站点, 参考上述步骤 1.4, 类似于

<https://xxx01.openai.azure.com/>

AOAI\_DEPLOYMENT\_ID 是 Azure OpenAI 服务资源模型部署的名称)

③Pre-request Script:

pm.environment.set("AOAI\_KEY", "<Azure OpenAI 资源的密钥>")

pm.environment.set("AOAI\_ENDPOINT", "https://xxx01.openai.azure.com/")

pm.environment.set("AOAI\_DEPLOYMENT\_ID", "<Azure OpenAI 资源内模型的部署名称>")

Tests (可选):

```
var jsonData = pm.response.json();
```

```
var data = jsonData
```

```
console.log(data.choices[0].text)
```

④Headers: api-key : {{AOAI\_KEY}} (AOAI\_KEY 是 Azure OpenAI 资源的密钥, 参考上述步骤 1.4)

⑤Body

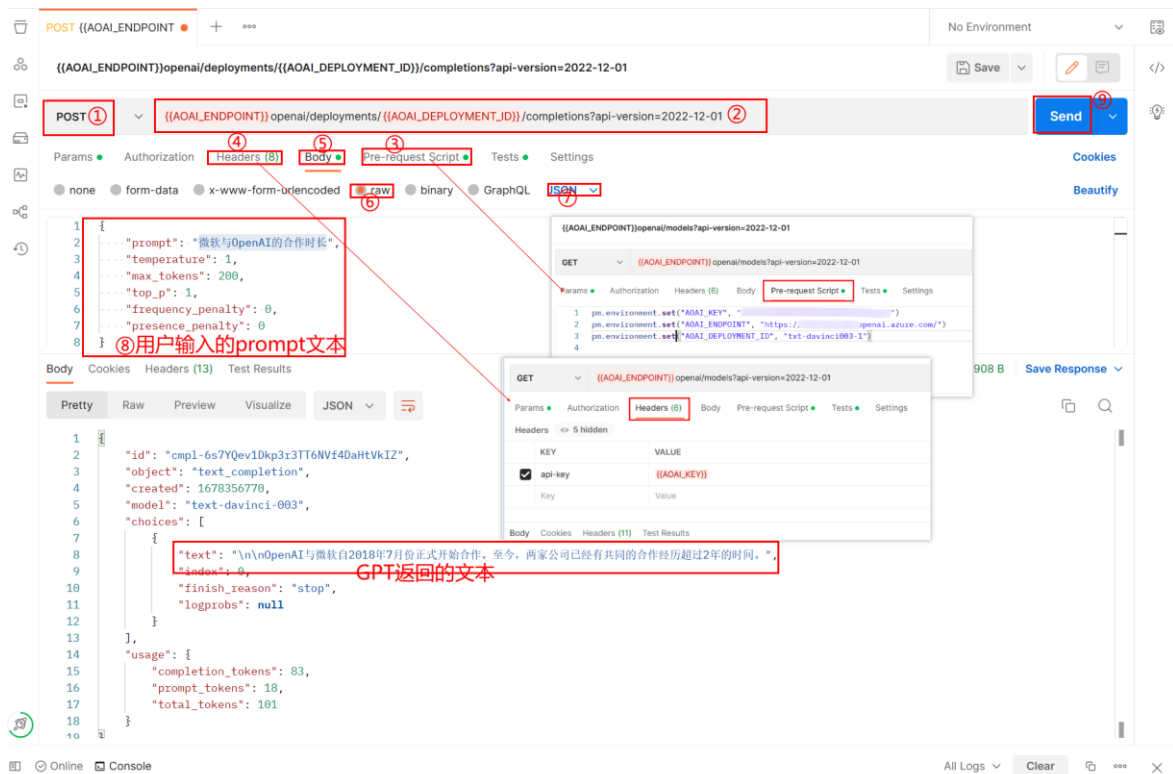
⑥raw

⑦JSON

⑧Body 的内容

```
{
  "prompt": "微软与 OpenAI 的合作时长",
  "temperature": 1,
  "max_tokens": 200,
  "top_p": 1,
  "frequency_penalty": 0,
  "presence_penalty": 0
}
```

⑨send



### 6.4.3 Codex

### 6.4.4 Embeddings

### 6.4.5 DELL-E

目前 Azure 上的 DELL-E 产品还没有上线，以下是 OpenAI 的 api 调用

①请求方式： POST

②请求的 URL： <https://api.openai.com/v1/images/generations>

③Authorization： BearerToken： <OpenAI 的 API Token>

④Body

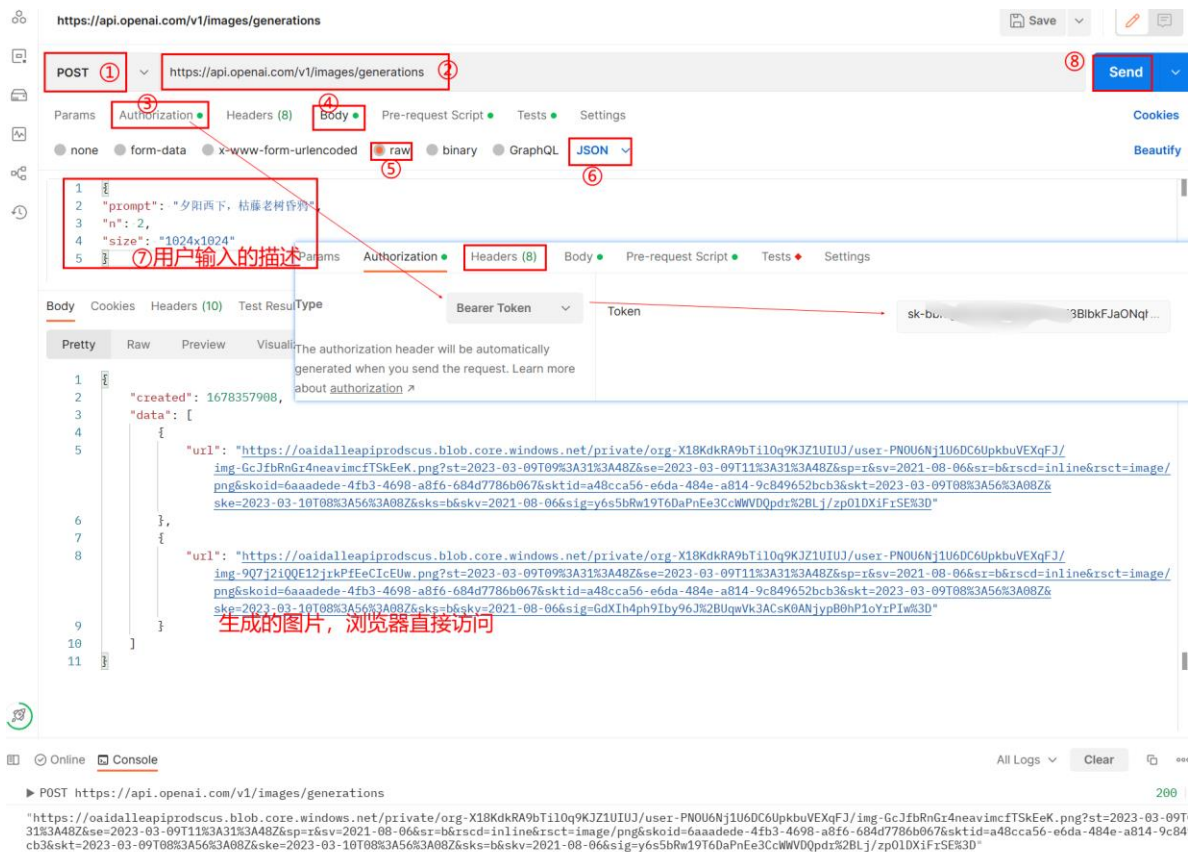
⑤raw

⑥JSON

⑦Body 的内容

```
{
  "prompt": "夕阳西下，枯藤老树昏鸦",
  "n": 2,
```

```
"size": "1024x1024"
}
⑧send
```



#### 6.4.6 Whisper

#### 6.4.7 ChatGPT

### 6.5 通过 CHATGPT 或 CODEX 模型生成代码

#### 6.5.1 Codex 通用工具

##### GitHub Copilot

#### 6.5.2 ChatGPT

输入 prompt 文本提示后，等待回复，将生成的代码中涉及到的密钥替换成自己的





以下是一个简单的 Python 代码示例，演示如何使用 OpenAI 的 GPT 模型实现多轮实时对话。在这个示例中，我们使用了 OpenAI 提供的 GPT-3 模型，但是你需要有一个 OpenAI API 密钥才能运行它。



```
python Copy code

import openai
import time

# 设置 OpenAI API 密钥
openai.api_key = "YOUR_API_KEY"

# 初始化 GPT-3 模型
model_engine = "text-davinci-002"
prompt = "Hello, I'm a chatbot. What can I help you with today?"
chat_history = ""

# 进入对话循环
while True:
    # 获取用户输入
    user_input = input("> ")

    # 将用户输入添加到对话历史中
    if user_input.strip():
        chat_history += f"\nUser: {user_input.strip()}"

    # 组合文本输入
    text_input = prompt + chat_history

    # 生成响应
    response = openai.Completion.create(
        engine=model_engine,
        prompt=text_input,
        max_tokens=1024,
        n=1,
        stop=None,
        temperature=0.5,
    )

    # 解析响应文本
    chat_history += f"\nChatbot: {response.choices[0].text.strip()}"
    print(f"Chatbot: {response.choices[0].text.strip()}")

    # 等待一会儿，以避免过于频繁的请求
    time.sleep(1)
```

在上面的代码中，我们定义了一个包含 GPT-3 模型初始化和对话循环的主程序。在循环中，我们首先获取用户输入，然后将其添加到对话历史中。接下来，我们组合文本输入，其中包括先前的对话历史和当前用户输入。然后，我们使用 OpenAI API 发送生成响应的请求。最后，我们将响应解析为文本并将其添加到对话历史中，然后将其打印到控制台上。

在每个循环迭代中，我们等待一秒钟，以避免过于频繁的请求。这有助于确保我们不会超出 OpenAI API 的配额限制。

请注意，上面的代码示例是一个简单的 GPT-3 对话引擎，它只是在每个用户输入后简单地生成一个响应。如果你想更进一步地定制你的对话引擎，你需要考虑如何处理上下文、回答选项等。

## 7 其他文档参考

---

- [Azure OpenAI 服务管理 – CLI](#)
- [OpenAI 提供的示例](#)