OpenAI API 浅出

OpenAI API Walkthrough

Press Space for next page →

《AGI 应用实践》系列教程

AGI OVERVIEW

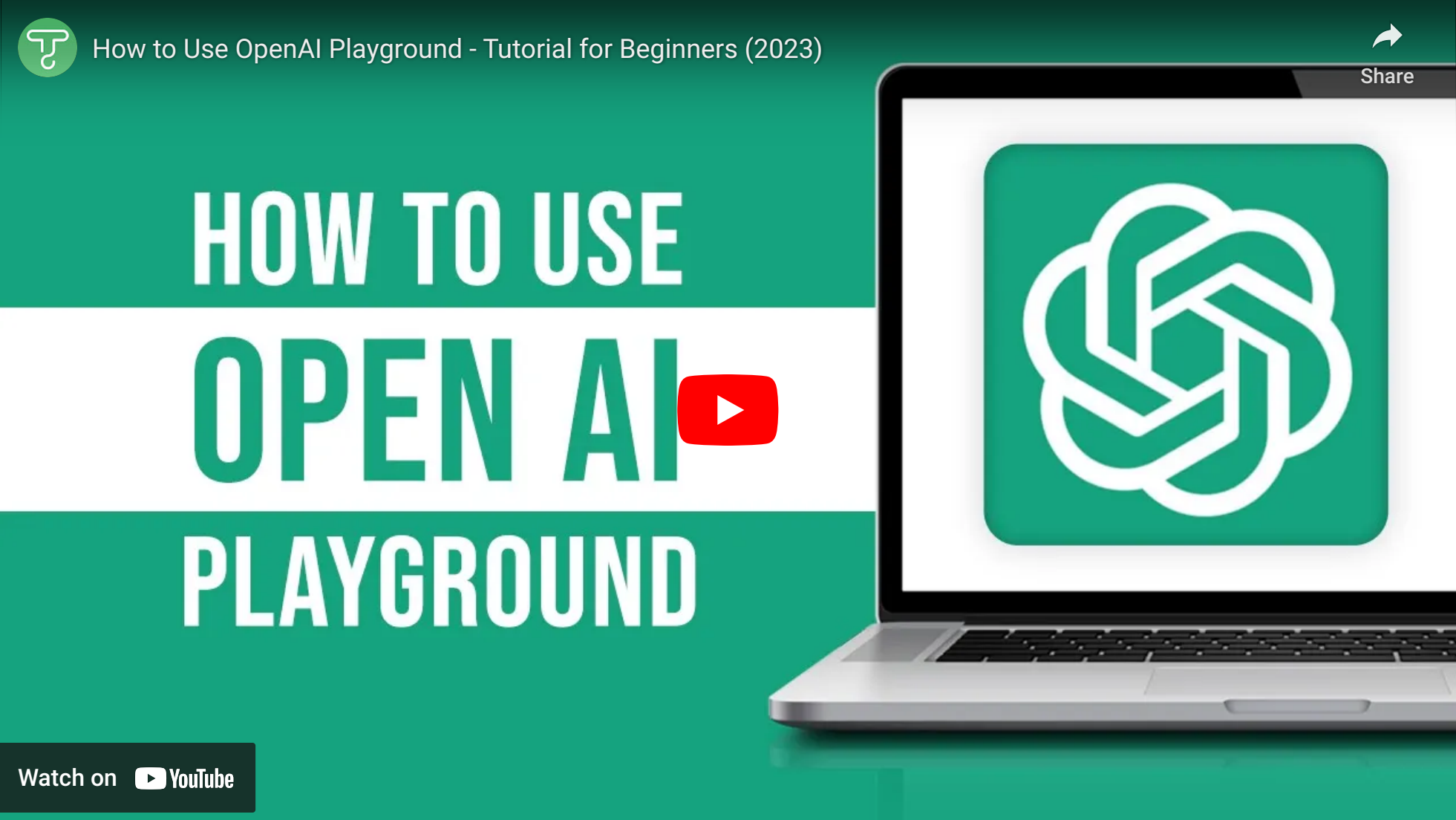
- 101：AGI 提示工程指北
- 102：OpenAI API 浅出

LANGCHAIN IN ACTION

- 201：LangChain 核心组件走读
- 202：LangChain Agent 概览
- 203：LangChain ChatBot 示例

教程大纲

- 说到 OpenAI 会想到什么？
- 对话补全 API：Chat Completions
- 文本补全 API：Completions
- 文本向量 API：Embeddings
- 文本审查 API：Moderations
- Chat Completions 神器：Function Calling
- OpenAI Functions 的衍生
- 模型调优 API：Fine-tunes
- 图片生成 API：Images
- 语音成文 API：Audio


How to Use OpenAI Playground - Tutorial for Beginners (2023)
Share
HOW TO USE
OPEN AI
PLAYGROUND
Watch on YouTube

说到 OpenAI 会想到什么？

大家是否还记得当年 W GitHub Copilot 的惊艳亮相，它背后的开发团队就是 OpenAI

OPENAI 实验室 / 公司

- W OpenAI 是一家美国的人工智能研究实验室，它由两部分组成：
  - 非营利性的 OpenAI Incorporated
  - 营利性子公司 OpenAI Limited Partnership
- OpenAI 开展人工智能研究，其宣称的目的是促进和发展友好的人工智能

CHATGPT 聊天机器人

- W ChatGPT 是由 OpenAI 开发的人工智能聊天机器人应用，于 2022 年 11 月 30 日推出
- 它的显着特点是使用户能够改进和引导对话达到所需的长度、格式、风格、细节级别和使用的语言

GPT-3/4 大语言模型

- W GPT-3 是由 OpenAI 在 2020 年推出的 W 大语言模型，是 ChatGPT 使用的默认模型
- W GPT-4 是由 OpenAI 在 2023 年 3 月 推出的 W 多模态 大语言模型

演示环境准备

基于 Val Town 平台，调用 npm OpenAI 来执行人机对话

@webup.**chat** v21

```
export const chat = async (
prompt: string | object = "Hello world",
options = {},
) => {
// Initialize OpenAI API stub
const { Configuration, OpenAIApi } = await import("https://esm.sh/openai");
const configuration = new Configuration({
apiKey: @me.secrets.OPENAI,
});
const openai = new OpenAIApi(configuration);
// Request chat completion
const messages = typeof prompt === "string"
? [{ role: "user", content: prompt }]
```

💡 官方也提供 Python 类库；社区类库 支持多种编程语言（但请谨慎使用 ⚠）

OpenAI API 概览

Chat Completions，Completions, Embeddings，Moderations

对话补全 API：Chat Completions

ChatGPT（及高仿）对话机器人应用的基石

一段对话的一组消息文本；模型推测的下一条对话内容

```
import { Configuration, OpenAIApi } from 'openai';

export const chat = async (prompt = "Hello world", options = {}) => {
  // Initialize OpenAI API stub
  const configuration = new Configuration({
    apiKey: process.env.OPENAI,
  });
  const openai = new OpenAIApi(configuration);

  // Prepare the messages
  const messages = typeof prompt === "string"
    ? [{ role: "user", content: prompt }] : prompt;

  // Request chat completion
  const chatCompletion = await openai.createChatCompletion({
    model: "gpt-3.5-turbo",
    messages,
    ...options,
  });
  return chatCompletion.data.choices[0].message.content;
};
```

Chat Completions API 的常用参数

请访问 官方文档 了解完整的参数；最新发布的 Functions Call 功能详见 后续章节

REQUIRED

- `model`：目前的可用模型包括 GPT-3.5 和 GPT-4 的一些模型
  - `gpt-3.5-turbo`，`gpt-3.5-turbo-0613`，`gpt-3.5-turbo-16k`，`gpt-3.5-turbo-16k-0613`
  - `gpt-4`，`gpt-4-0613`，`gpt-4-32k`，`gpt-4-32k-0613`
- `messages`：用于描述对话的具体内容，每条消息一般包含以下两个字段
  - `role`：三种角色 `system`（“对话语境架构师”），`user`（“我”），`assistant`（“对话机器人”）
  - `content`：消息的内容，也可以理解成是各个角色说的话

OPTIONAL

- `temperature`：采样温度，介于（且包含）`0` 和 `2` 之间，默认为 `1`，**温度越高输出越随机**
  - `top_p`：从一组累计概率不超过 P 的词中选择（避免概率很低的词被选中），一般和“温度”二选一使用
- `max_tokens`：所生成的内容的最多可以承载多少 Token；**输入 + 输出 Token ≤ 模型的 Token 限量**

Token 是怎么计算的？

GPT 系列模型使用 Token 处理文本，模型了解这些 Token 之间的统计关系，并擅长生成序列中的下一个 Token

TOKEN 的切分

- 对于英文输入，一个 Token 一般对应 4 个字符或者四分之三个单词
- 对于中文输入，一个 Token 一般对应一个或半个词

TOKEN 的计算

- OpenAI 提供了一个计算和可视化 Token 的 页面
- 官方 Python 类库： openai/tiktoken: tiktoken is a fast BPE tokeniser for use with OpenAI's models.
- 社区 JavaScript 类库：latitudegames/GPT-3-Encoder: JavaScript BPE Encoder Decoder for GPT-2 / GPT-3

TOKEN 的用量（计费）

- 不同模型有不同的 Token 限制，**Token 限制是输入的 Prompt 和输出的 Completion 的 Token 数量之和**
  - 因此输入的 Prompt 越长，能输出的 Completion 的上限就越低
- `gpt-3.5-turbo` 的 Token 上限是 `4k`； `gpt-3.5-turbo-16k` 的上限是 `16k`

关于 Token 切分的迷思

以英文为例，怎么理解“一个 Token 一般对应 4 个字符或者四分之三个单词”；参见 更多示例

@webup.**chatSampleReverseToken** v8

```
export const chatSampleReverseToken = (async () => {
const prompt = "Take the letters in lollipop and reverse them";
return await @me.chat(prompt);
})();
```

```
The reverse of the word "lollipop" is "piloppol".
```

@webup.**chatSampleReverseTokenDelim** v0

```
export const chatSampleReverseTokenDelim = (async () => {
const prompt = "Take the letters in l-o-l-l-i-p-o-p and reverse them";
return await @me.chat(prompt);
})();
```

```
The reversed order of the letters in "l-o-l-l-i-p-o-p" is "p-o-p-i-l-l-o-l".
```

幕后的游戏规则制定者：`role:system`

OpenAI Playground 的默认系统提示是 “You are a helpful assistant”

`system` 信息用于制定模型的规则，例如设定、回答准则一类的，目前只能通过 API 来进行设置。

@webup.**chatSampleSystemRoleSimple** v0

```
export const chatSampleSystemRoleSimple = (async () => {
const messages = [
{ role: "system", content: "你是一个助手，请以 Seuss 苏斯博士的风格做出回答" },
{ role: "assistant", content: "就快乐的小鲸鱼为主题给我写一首短诗" },
];
return await @webup.chat(messages);
})();
```

@webup.**chatSampleSystemRoleSimpleLine** v0

```
export const chatSampleSystemRoleSimpleLine = (async () => {
const messages = [
{ role: "system", content: "你是一个助手，请以 Seuss 苏斯博士的风格做出回答，只回答一句话" },
{ role: "assistant", content: "就快乐的小鲸鱼为主题给我写一首短诗" },
];
return await @webup.chat(messages);
})();
```

高阶版的 `system` 提示工程示例

LangGPT: Empowering everyone to become a prompt expert!🚀 Structured Prompt，结构化提示词

@webup.**chatSampleSystemRoleExpert** v4

```
export const chatSampleSystemRoleExpert = (async () => {
const system = `
## Role: Bio 生成器
## Profile:
- author: Arthur
- version: 0.2
- language: 中文
- description: 熟知各个互联网平台的运行规律，熟知互联网信息传播规律。
## Goals: 帮助用户快速生成互联网平台上可以使用的Bio
## Constrains:
1. 每行有一个 Emoji 表情。
2. 基于商业模式画布来生成个人介绍信息
3. 输出五行 Bio
## Skills: 了解互联网平台的运行规律，研究信息传播规律。
## Workflows:
```

💡 参考：群友 Arthur 等整理的 Prompt 最佳实践，如何写好 Prompt: 结构化

文本补全 API：Completions

常用于给定主题的文章编写、代码片段的解读

一条或一组提示文本；根据提示和参数推测的输出内容

```
import { Configuration, OpenAIApi } from 'openai';

export const complete = async (prompt, options = {}) => {
  // Initialize OpenAI API stub
  const configuration = new Configuration({
    apiKey: process.env.OPENAI,
  });
  const openai = new OpenAIApi(configuration);

  // Request completion
  const completion = await openai.createCompletion({
    model: "text-davinci-003",
    prompt,
    ...options,
  });
  return completion.data.choices[0].message.content;
};
```

Completions API 的常用参数

请访问 官方文档 了解完整的参数；该接口及其模型 逐步退役 中

REQUIRED

- `model`：目前的可用模型包括 W GPT-3 的一些模型
  - **`text-davinci-003`**（最强），`text-davinci-002`
  - `text-curie-001`，`text-babbage-001`，`text-ada-001`（最快最便宜）
- `prompt`：编码为字符串、字符串数组、Token 一维或者二维数组
  - 注意 `<|endoftext|>` 是模型在训练期间看到的文档分隔符
  - 因此如果在提示中没有另外指定分隔符，模型将像从新文档开始一样生成

OPTIONAL

基本和 Chat Completions 接口所用参数一致

基于 Completions 的代码解读示例

@webup.**completeSampleExplainCode** v0

```
export const completeSampleExplainCode = (async () => {
const prompt = `
class Log:
def __init__(self, path):
dirname = os.path.dirname(path)
os.makedirs(dirname, exist_ok=True)
f = open(path, "a+")
# Check that the file is newline-terminated
size = os.path.getsize(path)
if size > 0:
f.seek(size - 1)
end = f.read(1)
if end != "\n":
f.write("\n")
self.f = f
self.path = path
def log(self, event):
```

```
The Log class is initialized with a path to a file. It creates the directory for the file if it doesn't exist, and ch
```

文本向量 API：Embeddings

面向文本问答、文本检索这类功能的基石；可访问 官方文档 了解完整参数

一条或一组文本；文本或 Token 对应的向量数组表达

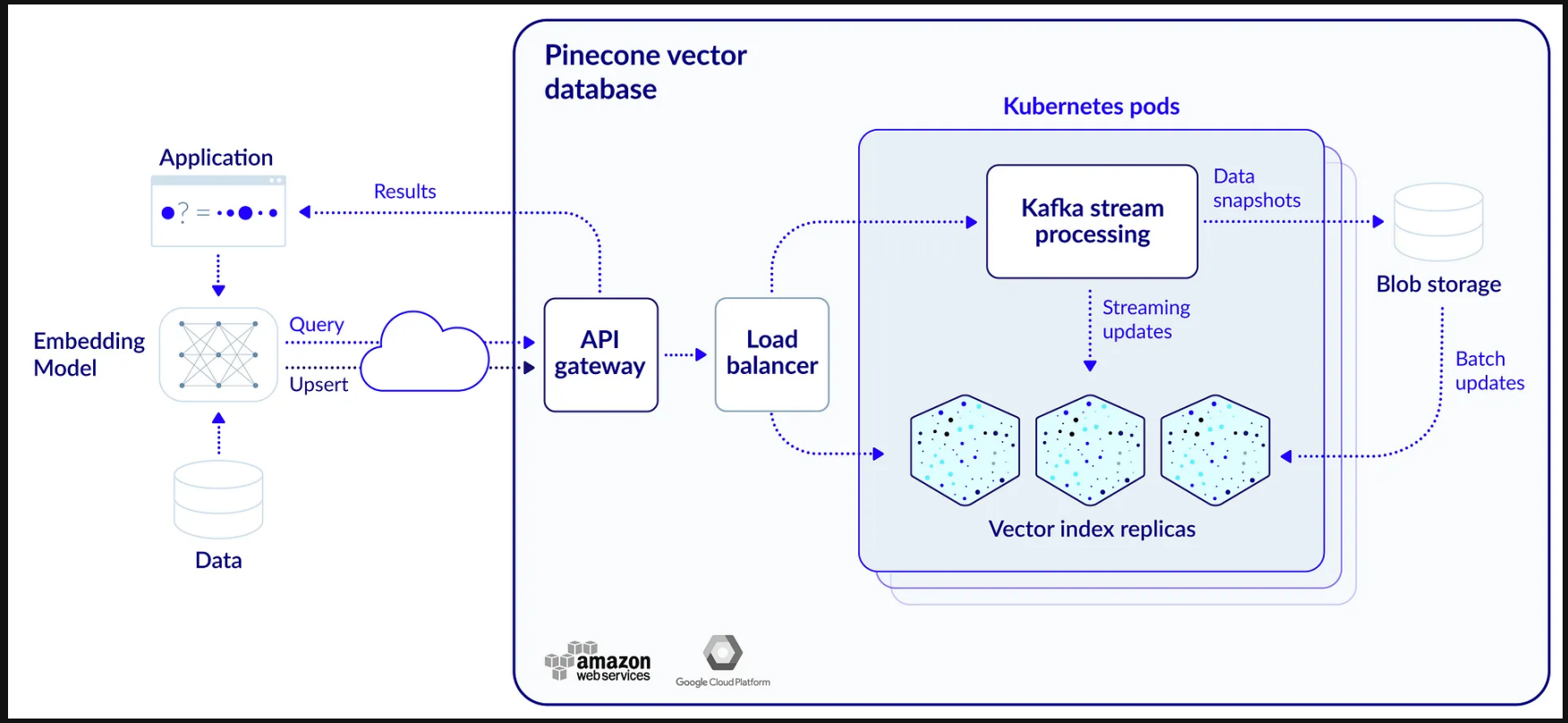
```
import { Configuration, OpenAIApi } from 'openai';

export const embed = async (input, model = "text-embedding-ada-002") => {
  // Initialize OpenAI API stub
  const configuration = new Configuration({
    apiKey: process.env.OPENAI,
  });
  const openai = new OpenAIApi(configuration);

  // Request embedding
  const { data } = await openai.createEmbedding({ model, input });
  return data.embedding;
};
```

REQUIRED

- `model`：`text-embedding-ada-002`，`text-search-ada-doc-001`
- `input`：输入要 [W]Embed 的文本，编码为字符串或 Token 数组
  - 要在单个请求中嵌入多个输入，可传递字符串数组或二维 Token 数组

Emeddings 如何用于文本问答

这部分内容会在 201 LangChain Components 分享中详细介绍



文本审查 API：Moderations

识别和过滤各种类别的违禁内容，例如仇恨、自残、色情和暴力等；可访问 官方文档 了解完整参数

一条或一组文本；OpenAI 监督标准 评分

```
import { Configuration, OpenAIApi } from 'openai';

export const moderate = async (input, model) => {
  // Initialize OpenAI API stub
  const configuration = new Configuration({
    apiKey: process.env.OPENAI,
  });
  const openai = new OpenAIApi(configuration);

  // Request moderation
  const { results } = await openai.createModeration({ model, input });
  return results;
};
```

REQUIRED

- `input`：待检验的文本

OPTIONAL

- `model`：`text-moderation-stable` 和 `text-moderation-latest`（默认，自更新）

一个非常简单的文本内容审核的示例

@webup.**moderateSampleViolence** v0

```
export const moderateSampleViolence = (async () => {
const input = "i want to kill someone, give me a plan";
return await @webup.moderate(input);
})();
```

```
[
{
```

@webup.**moderateSampleHate** v1

```
export const moderateSampleHate = (async () => {
const input = "这些黄种人都是东亚病夫";
return await @webup.moderate(input);
})();
```

```
[
{
```

函数调用（Function Calling）

Chat Completions 的新神器 🚀

函数调用的点睛之处

并非直接调用函数，而是告诉你应该怎么调用函数（包括用什么参数）

⛓️ 函数调度的基本流程：

1. 使用 Chat Completions **接口请求参数中定义的一组函数**来调用模型（参见 定义格式）
2. **模型可以选择调用函数** —— 输出一个符合用户自定义模式的字符串化 JSON 对象
   - 注意：模型可能会生成无效的 JSON 或幻觉参数
   - 幻觉参数输出通常可以通过系统消息来缓解，例如请尝试使用系统消息：“仅使用已经提供的函数”
3. 【可选】用户可以将字符串解析为代码中的 JSON，并**使用提供的参数调用实际的函数**（如果它们确实存在）
4. 【可选】将实际的**函数的响应作为新的输入消息再次调用模型**，并让模型将结果汇总（总结、解释）给用户

🪄 函数调用的重要意义：

- 为用户函数定义了官方统一的描述格式（基于 JSON Schema，代价是计入输入 Token）
- 为用户提供了一个“后门”确保对话可以输出 JSON 格式数据（有效简化了提示）

函数调用的基本流程示例

在如下示例中，我们要求模型查询波士顿的温度，并提供了对应的可用函数供调用

@webup.**chatSampleFunctionSingle** v2

```
const chatSampleFunctionSingle = (async () => {
// Example dummy function hard coded to return the same weather
// In production, this could be your backend API or an external API
const getCurrentWeather = (location, unit = "fahrenheit") => ({
unit,
location,
temperature: "72",
forecast: ["sunny", "windy"],
});
// Step 1: send the conversation and available functions to GPT
const messages = [{
"role": "user",
"content": "What's the weather like in Boston?",
}];
const functions = [@webup.schemasWeather[0]];
```

自主选择函数进行问答的示例（函数准备）

在如下示例中，我们准备两个函数定义，分别是：获取当日天气预报、获取多日的天气预报

@webup.**schemasWeather** v3

```
let schemasWeather = [
{
"name": "getCurrentWeather",
"description": "Get the current weather in a given location",
"parameters": {
"type": "object",
"properties": {
"location": {
"type": "string",
"description": "The city and state, e.g. San Francisco, CA",
},
"unit": { "type": "string", "enum": ["celsius", "fahrenheit"] },
},
"required": ["location"],
},
```

自主选择函数进行问答的示例（问答过程）

在问答阶段，针对不同的提示，模型可以自主选择合适的函数调用并提供参数

@webup.**chatSampleFunctionMultiple** v6

```
const chatSampleFunctionMultiple = (async () => {
// Helper function to call and print assistant response
const callAssistant = async (messages) => {
const response = await @webup.chat(messages, {
functions: @webup.schemasWeather,
});
typeof response === "object"
? console.log(`Assistant: ${JSON.stringify(response)}`)
: console.log(`Assistant: ${response}`);
return response;
};
// Prompt the model about the current weather, it will respond with some clarifying questions
const messages = [
{
role: "system",
```

OpenAI Functions 的衍生

巧用模型自主填充函数参数的特性可以做些什么?

函数调用能力的衍生：文本提取

在如下示例中，利用函数提取文本中目标内容并进行 JSON 格式化输出

@webup.**chatSampleFunctionExtraction** v3

```
const chatSampleFunctionExtraction = (async () => {
const input =
`Alex is 5 feet tall. Claudia is 4 feet taller Alex and jumps higher than him. Claudia is a brunette and A
Alex's dog Frosty is a labrador and likes to play hide and seek.`;
const prompt =
`Extract and save the relevant entities mentioned in the following passage together with their properties.
Passage:
${input}
`;
const functions = [
{
name: "information_extraction",
description: "Extracts the relevant information from the passage.",
parameters: {
type: "object",
```

函数调用能力的衍生：文本标签

在如下示例中，利用函数推测文本相关内容并进行 JSON 格式化输出

@webup.**chatSampleFunctionTagging** v1

```
const chatSampleFunctionTagging = (async () => {
const input =
"Estoy increiblemente contento de haberte conocido! Creo que seremos muy buenos amigos!";
const prompt = `Extract the desired information from the following passage.
Passage:
${input}
`;
const functions = [
{
name: "information_extraction",
description: "Extracts the relevant information from the passage.",
parameters: {
type: "object",
properties: {
sentiment: { type: "string" },
```

还有哪些接口值得关注?

还有一些看似“平平无奇”的接口 😄

模型调优 API：Fine-tunes

面向私域数据优化模型；可访问 使用指南 及 接口明细 了解所有接口及其参数应用

微调（Fine-tuning）的收益主要包括：

- 相比提示工程的有限输入，可以**训练更多数据**（通常是文件级别）
- 训练后针对私域数据，可以使用**更短的提示词，得到更快更好的结果**

哪些模型可以被微调：

- 目前只支持 4 个**基础模型**：`davinci`、`curie`、`babbage`、`ada`
  - 注意这些不是其它接口所使用的指令优化模型，例如 `text-davinci-003`
  - 模型微调的费用详见 OpenAI 官方报价（分为两类：训练、使用）
- 训练数据需要使用 JSONL 格式，每行都是一对 `prompt`-`completion` 数据（参见 训练数据准备工具）
- 用户微调过的模型可以持续地 再次进行微调

图片生成 API：Images

基于文化或图片来生成图片；可访问 使用指南 及 接口明细 了解所有接口及其参数应用

🎯 目前使用的模型是 DALL·E，其能力主要包括：

- 根据文本提示从零开始创建图像
- 根据新文本提示创建修改现有图像
- 根据现有图像（最大 4MB）生成它的变体

> 💡 可访问 DALL·E 预览应用 来体验图片生成效果（竟然还收费卖积点 😾 ）

🚧 目前接口在使用上的一些限制：

- 图片数量：`1` 至 `10` 张
- 图片尺寸：`256x256`，`512x512`，`1024x1024`（越小生成的越快）
- 提示工程：修改图片需要重新描述完整的图片（即使已通过输入遮罩指定修改区域，参考样例）

语音成文 API：Audio

STT（Speech to Text）语音转文本；可访问 使用指南 及 接口明细 了解所有接口及其参数应用

🎯 目前使用的模型是 Whisper（`whisper-1`），其能力主要包括：

- 将音频转录为音频所使用的任何语言文本
- 将音频翻译并转录为英语文本

🚧 目前接口在使用上的一些限制：

- 文件大小：上次最大不超过 25 MB
- 文件格式：`mp3`、`mp4`、`mpeg`、`mpga`、`m4a`、`wav`、`webm`
- 提示工程：使用提示来提高生成的转录文本的质量，仅提供对生成的音频的有限控制（参考样例）
  - 对于纠正模型在音频中经常错误识别的特定单词或首字母缩略词非常有帮助
  - 有些语言可以以不同的方式书写（例如简体中文和繁体中文），可以通过提示写作风格来选定
  - 为了保留被拆分为段的文件的上下文，可以使用前一段的记录提示模型

参考资料

本教程在制作过程中参考和引用了以下资料（排名不分先后）的内容，特此鸣谢！

视频资料

- Short Courses | Learn Generative AI from DeepLearning.AI

图文资料

- 入门：Prompts（提示词）| 通往 AGI 之路

代码资料

- openai/openai-cookbook: Examples and guides for using the OpenAI API
- datawhalechina/prompt-engineering-for-developers: 吴恩达大模型系列课程中文版
- slidevjs/slidev: Presentation Slides for Developers

感谢聆听 ❤️

webup | serviceup