

08 | 文本改写和内容审核，别让你的机器人说错话

2023-03-31 徐文浩 来自北京

《AI大模型之美》



你好，我是徐文浩。

前面，我们已经把 OpenAI 最主要的接口介绍完了。这一讲也是我们基础知识篇里面的最后一讲，我们会覆盖完 OpenAI 的 GPT 系列模型剩下的一些接口。也许有些接口你不一定会频繁使用，但是了解一下没有什么坏处，说不定你有什么需求就能用得上它。

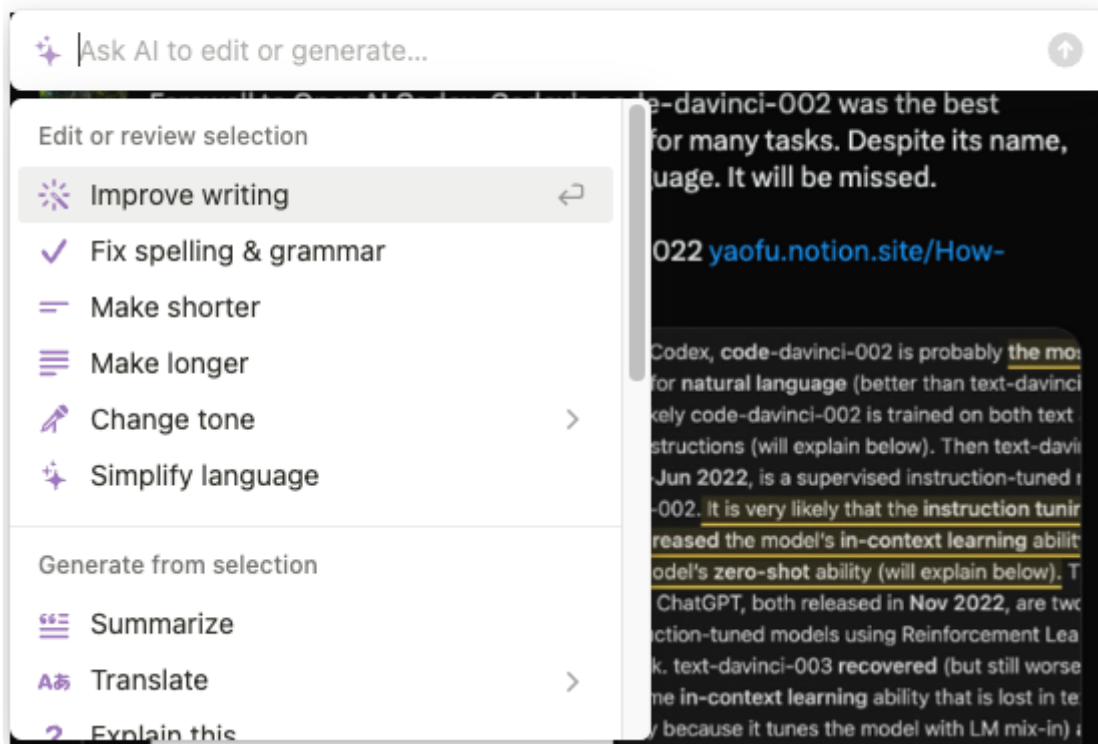
在这一讲里，我们会一起来看看 OpenAI 为文本改写和内容审核提供的功能有哪些。以及 OpenAI 的 GPT 系列有哪些模型，这些模型有哪些区别，什么情况下我们应该用什么模型。

文本改写，从使用提示语开始

我猜课程学到这里，你应该已经用过不少基于 AI 大语言模型的产品了。很常见的一类应用，就是写作助手。比如 Notion AI 就能帮助你，在已经写好的文章里面选取一段内容，你可以

让 AI 帮你修改。这个修改可以是让文本短一点或者长一点，也可以是让文本改一下自己的语气。

看到这个文章我的第一个反应，也是哦，通用的模型效果已经优于专有模型了。不过，接着就看到有很多研究人员抱怨，希望OpenAI保留这个模型。然后OpenAI的反应也很快，表示会为研究人员保留这个模型。



Notion的文本改写的界面

不过，OpenAI 的 GPT 的系列模型是一个生成式的模型，也就是它的用法是你给它一段文字，然后它补全后面的文字。按理来说，你是没法让它修改一段内容的。当然，在看了那么多不同的“提示语”之后，相信你自然想到可以通过一段提示语来解决这个问题。比如，下面这段代码就是这样的，我们通过上一讲介绍的 ChatGPT 的模型来实现了这个功能。

复制代码


```
1 def make_text_short(text):
2     messages = []
3     messages.append( {"role": "system", "content": "你是一个用来将文本改写得短的AI助手"}
4     messages.append( {"role": "user", "content": text})
5     response = openai.ChatCompletion.create(
6         model="gpt-3.5-turbo",
7         messages=messages,
8         temperature=0.5,
9         max_tokens=2048,
```

```

10     presence_penalty=0,
11     frequency_penalty=2,
12     n=3,
13 )
14     return response
15
16 long_text = """
17 在这个快节奏的现代社会中，我们每个人都面临着各种各样的挑战和困难。
18 在这些挑战和困难中，有些是由外部因素引起的，例如经济萧条、全球变暖和自然灾害等。
19 还有一些是由内部因素引起的，例如情感问题、健康问题和自我怀疑等。
20 面对这些挑战和困难，我们需要采取积极的态度和行动来克服它们。
21 这意味着我们必须具备坚韧不拔的意志和创造性思维，以及寻求外部支持的能力。
22 只有这样，我们才能真正地实现自己的潜力并取得成功。
23 """
24 short_version = make_text_short(long_text)
25
26 index = 1
27 for choice in short_version["choices"]:
28     print(f"version {index}: " + choice["message"]["content"])
29     index += 1

```

输出结果：

 复制代码

```

1 version 1: 现代社会中，我们面临外部和内部的挑战。要克服它们，需要积极态度、创造性思维和寻求支持
2 version 2: 现代社会中，我们面临各种挑战和困难。有些是外部因素引起的，如经济萧条、全球变暖等；还
3 version 3: 现代社会中，我们面临各种挑战和困难。有些源于外部因素（如经济萧条、全球变暖），而另一

```

使用 ChatGPT 的模型接口，是因为它的价格比较低廉。而我们使用的参数也有以下几个调整。

1. 首先是我们使用了 `n=3` 这个参数，也就是让 AI 给我们返回 3 个答案供我们选择。在文本改写类的应用里面，我们通常不只是直接给出答案，而是会给用户几个选项来选择。
2. 其次是我们引入了两个参数 `presence_penalty=0` 以及 `frequency_penalty=2`。这两个参数我们之前没有介绍过，它们和 `temperature` 参数类似，都是来控制你输出的内容的。
 - a. `presence_penalty`，顾名思义就是如果一个 Token 在前面的内容已经出现过了，那么在后面生成的时候给它的概率一定的惩罚。这样，AI 就会倾向于聊新的话题和内容。在这里，我们把它设置成了默认值 0。

- b. `frequency_penalty`, 指的是对于重复出现的 Token 进行概率惩罚。这样, AI 就会尽量使用不同的表述。在这里我们设成了最大的 2, 你也可以设置成最小的 -2。但是那样的话, 它就更容易说车轱辘话了。

通过 `logit_bias` 参数精确控制内容

不过, 无论是 `temperature` 还是 `presence_penalty` 和 `frequency_penalty`, 都是一个参数, 我们没有办法精确控制哪些词我们不想出现。不过, 对于这一点, OpenAI 还是提供了解决方案, 比如, 我们想要在上面生成的内容里面, 不允许出现灾害两个字, 就可以这么做。

 复制代码

```
1 import tiktoken
2 encoding = tiktoken.get_encoding('p50k_base')
3 token_ids = encoding.encode("灾害")
4 print(token_ids)
5
6 bias_map = {}
7 for token_id in token_ids:
8     bias_map[token_id] = -100
9
10 def make_text_short(text):
11     messages = []
12     messages.append( {"role": "system", "content": "你是一个用来将文本改写得短的AI助手"}
13     messages.append( {"role": "user", "content": text})
14     response = openai.ChatCompletion.create(
15         model="gpt-3.5-turbo", messages=messages, temperature=0.5, max_tokens=204
16         n=3, presence_penalty=0, frequency_penalty=2,
17         logit_bias = bias_map,
18     )
19     return response
20
21 short_version = make_text_short(long_text)
22
23 index = 1
24 for choice in short_version["choices"]:
25     print(f"version {index}: " + choice["message"]["content"])
26     index += 1
```

输出结果:

```

1 [163, 223, 122, 22522, 111]
2 version 1: 现代社会中, 我们面对各种挑战和困障。有些是外部原因引起的, 如经济萧条、全球变暖和自然
3 version 2: 现代社会中, 我们面对外部和内部的挑战。为了克服这些困难示意, 需要积极主动态度和行动,
4 version 3: 现代社会中, 我们面临各种挑战和困障。有外部因素如经济萧条、全球变暖等, 也有内部因素如

```

这段代码里面, 我们这样做的。

1. 我们通过之前使用过的 Tiktoken 库, 把我不希望出现的“灾害”这个词儿, 找到它对应的 Token, 然后给它们都赋了一个 -100 的 bias。
2. 然后把整个的 bias_map 作为参数, 传给了 Completion 的 logit_bias 参数。

在生成结果里面, 你可以看到, 三个回复都没有“灾害”这两个字了。即使是之前出现的第一个回复里原来是有“灾害”这两个字的。现在一个被强行改成了繁体的“災”字, 另一个干脆是给了个错别字“宣”。

对于 logit_bias 参数里面的取值范围, 是在 -100 到 100 之间。不过, 一般情况下, 设置在 1 到 -1 之间就足够了。我自己的体会是, 设置成 100, 你一定要某些字出现, 那么整个生成会慢到无法忍受。

使用英文来减少 Token 的使用

不知道你有没有注意到, 虽然灾害只有两个中文汉字, 但是我们通过 Tiktoken 去处理的时候, 我们打印了对应的 Token 的 id 是什么, 实际上有 5 个 Token。这里其实和我们之前看到的英文一样, 并不是一个字或者一个单词是一个 Token。事实上, 同样含义的中文, 目前消耗的 Token 数量是比英文多的。比如, 我们把上面的一句话翻译成英文, 然后数一下对应同样内容的中英文的 Token 数。

```

1 def translate(text):
2     messages = []
3     messages.append( {"role": "system", "content": "你是一个翻译, 把用户的话翻译成英文"}
4     messages.append( {"role": "user", "content": text})
5     response = openai.ChatCompletion.create(
6         model="gpt-3.5-turbo", messages=messages, temperature=0.5, max_tokens=204

```

```
7     )
8     return response["choices"][0]["message"]["content"]
9
10  chinese = long_text
11  english = translate(chinese)
12
13  num_of_tokens_in_chinese = len(encoding.encode(chinese))
14  num_of_tokens_in_english = len(encoding.encode(english))
15  print(english)
16  print(f"chinese: {num_of_tokens_in_chinese} tokens")
17  print(f"english: {num_of_tokens_in_english} tokens")
```

输出结果：

 复制代码

```
1 In this fast-paced modern society, each of us faces various challenges and diffic
2 chinese: 432 tokens
3 english: 115 tokens
```

可以看到，同样的内容，中文消耗的 Token 数量超过 400，而英文的 Token 数量只有 100 出头。如果你在生产环境中使用 OpenAI 的接口，最好还是使用英文的提示语，最多在输出结果的时候，告诉它 “generate Chinese” 之类的，可以极大地节约成本。不过，我们后面课程的演示，还是会尽量使用中文，方便你理解。

看看 OpenAI 给了我们哪些模型

有些同学可能看过文档会说，改写文本不是 OpenAI 单独提供了一个 Edit 的接口吗？的确，曾经，OpenAI 单独给过一个 Edit 接口，也单独提供了文本编辑的模型。目前，你在 OpenAI 的官网上还能看到相关的 [📄 文档](#)。但是根据我的测试，这个接口和模型目前是不能使用的，不知道是因为是 Alpha 版本还是已经被下线了。

因为目前 OpenAI 的产品更新非常快，所以很可能会出现一个问题，我告诉你应该使用某个模型，但是这个模型已经不是效果最好或者最新的了。所以，最好的办法，还是通过它提供的接口看看它到底有哪些模型。

```
1 import pandas as pd
2 # list all open ai models
3 engines = openai.Engine.list()
4 pd = pd.DataFrame(openai.Engine.list()['data'])
5 display(pd[['id', 'owner']])
```

[复制代码](#)

输出结果：

```
1   id  owner
2  0  babbage  openai
3  1  davinci  openai
4  2  babbage-code-search-code  openai-dev
5  3  text-similarity-babbage-001  openai-dev
6  4  text-davinci-001  openai
7  .....
8  14 text-embedding-ada-002  openai-internal
9  .....
10 30  gpt-3.5-turbo-0301  openai
11 .....
12 41  gpt-4  openai
13 42  text-search-davinci-doc-001  openai-dev
14 43  gpt-4-0314  openai
15 .....
16 47  text-similarity-davinci-001  openai-dev
17 48  text-davinci-002  openai
18 49  davinci-similarity  openai-dev
```

[复制代码](#)

在我写下这节课的时候，输出结果里有 49 个模型。其实顾名思义，你就能够知道这些模型是用来干啥的。比如 text-similarity-babbage-001 肯定就是用来进行相似度匹配的，就会比较适合用在我们 [02 讲](#)介绍的零样本分类。而 text-search-davinci-doc-001 肯定就更适合用来搜索文档。


尽管有些模型的名字标注了 openai-dev 或者 openai-internal，但是这些模型都是可以使用的。比如，我们在 [02 讲](#)里面调用 get_embedding 方法拿到向量，背后用的就是 text-similarity-davinci-001 模型，也是一个 openai-dev 的模型。

不过，里面的很多模型都已经老旧了，实际上主要用的模型就是这几类。

1. GPT-4 家族的模型，包括 gpt-4 和 gpt-4-0314。使用的方式和 ChatGPT 的模型一样，其中带日期的模型表示是一个模型快照。也就是模型不会随着时间迁移不断更新。GPT-4 的模型现在还很昂贵，输入 1000 个 Token 需要 0.03 美分，生成 1000 个 Token 则需要 0.06 美分。一般呢，我都是拿它帮我写代码，准确率会比较高。
2. GPT-3.5 家族的模型，包括 ChatGPT 所使用的 gpt-3.5-turbo 或者 gpt-3.5-turbo-0301，以及 text-davinci-003 和 text-davinci-002 这两个模型。前者专门针对对话的形式进行了微调，并且价格便宜，无论输入输出，1000 个 Token 都只需要 0.002 美分。后两个里，003 的模型有一个特殊的功能，就是支持“插入文本”这个功能，我们稍后就讲。003 也是基于强化学习微调的，而 002 则是做了监督学习下的微调。text-davinci-003 和 002 模型比 3.5-turbo 要贵 10 倍，但是输出更稳定。你可以根据自己的需要来决定。
3. 剩下的，则是 Ada、Babbage、Curie 以及 Davinci 这四个基础模型。只适合用于下达单轮的指令，不适合考虑复杂的上下文和进行逻辑推理。这四个模型按照首字母排序，价格越来越贵，效果越来越好。而且我们如果要微调一个属于自己的模型，也需要基于这四个基础模型。
4. 最后则是 text-embedding-ada-002、text-similarity-ada-001 这些专门用途模型。一般来说，我们通过这个模型来获取 Embedding，再用在其他的机器学习模型的训练，或者语义相似度的比较上。

所有模型的名字都来自科学史上的名人。Ada 来自人类史上第一位程序员 Ada，她也是著名诗人拜伦的女儿。而 Babadge 则是设计了分析机的巴贝奇，巴贝奇分析机也被认为是现代计算机的前身。Curie 则是指居里夫人，Davinci 是指达芬奇。

我们可以挑几个模型，试一下它们 Embedding 的维度数量，你就知道模型的尺寸本身就是不一样的了。

 复制代码

```
1 from openai.embeddings_utils import get_embedding
2
3 text = "让我们来算算Embedding"
4
5 embedding_ada = get_embedding(text, engine="text-embedding-ada-002")
6 print("embedding-ada: ", len(embedding_ada))
7
```



```
8 similarity_ada = get_embedding(text, engine="text-similarity-ada-001")
9 print("similarity-ada: ", len(similarity_ada))
10
11 babbage_similarity = get_embedding(text, engine="babbage-similarity")
12 print("babbage-similarity: ", len(babbage_similarity))
13
14 babbage_search_query = get_embedding(text, engine="text-search-babbage-query-001")
15 print("search-babbage-query: ", len(babbage_search_query))
16
17 curie = get_embedding(text, engine="curie-similarity")
18 print("curie-similarity: ", len(curie))
19
20 davinci = get_embedding(text, engine="text-similarity-davinci-001")
21 print("davinci-similarity: ", len(davinci))
```

输出结果：


 复制代码

```
1 embedding-ada: 1536
2 similarity-ada: 1024
3 babbage-similarity: 2048
4 search-babbage-query: 2048
5 curie-similarity: 4096
6 davinci-similarity: 12288
```

可以看到，最小的 ada-similarity 只有 1024 维，而最大的 davinci-similarity 则有 12288 维，所以它们的效果和价格不同也是可以理解的了。

插入内容，GPT 也可以像 BERT

我们前面介绍的时候说过，text-davinci-003 这个模型有个特殊的功能，就是“插入文本”（Inserting Text）。某种意义上来说，你也可以通过这个功能来做文本改写。

 复制代码


```
1 prefix = """在这个快节奏的现代社会中，我们每个人都面临着各种各样的挑战和困难。
2 在这些挑战和困难中，有些是由外部因素引起的，例如经济萧条、全球变暖和自然灾害等。\\n"""
3 # 还有一些是由内部因素引起的，例如情感问题、健康问题和自我怀疑等。
4 suffix = """\\n面对这些挑战和困难，我们需要采取积极的态度和行动来克服它们。
5 这意味着我们必须具备坚韧不拔的意志和创造性思维，以及寻求外部支持的能力。
```

```

6 只有这样，我们才能真正地实现自己的潜力并取得成功。"""
7
8 def insert_text(prefix, suffix):
9     response = openai.Completion.create(
10         model="text-davinci-003",
11         prompt=prefix,
12         suffix=suffix,
13         max_tokens=1024,
14     )
15     return response
16
17 response = insert_text(prefix, suffix)
18 print(response["choices"][0]["text"])

```


输出结果：

 复制代码

1 另一些是内部因素，例如事业难以发展、无法解决的个人和家庭矛盾等。

可以看到，这个接口的使用，和普通的 Completion 接口基本一致，只有一个区别就是除了前缀的 prompt 参数之外，还需要一个后缀的 suffix 参数。

不过，对于插入内容，我们同样需要注意提示语。如果我们把上面的内容稍微改一改，比如去掉 Suffix 一开始的换行符号，插入的文本内容有些就会在我们的预期之外。

 复制代码

```

1 prefix = """在这个快节奏的现代社会中，我们每个人都面临着各种各样的挑战和困难。
2 在这些挑战和困难中，有些是由外部因素引起的，例如经济萧条、全球变暖和自然灾害等。 \n"""
3 # 还有一些是由内部因素引起的，例如情感问题、健康问题和自我怀疑等。
4 suffix = """面对这些挑战和困难，我们需要采取积极的态度和行动来克服它们。
5 这意味着我们必须具备坚韧不拔的意志和创造性思维，以及寻求外部支持的能力。
6 只有这样，我们才能真正地实现自己的潜力并取得成功。"""
7
8 response = insert_text(prefix, suffix)
9 print(response["choices"][0]["text"])

```

输出结果：

- 1 例如，因应全球变暖，政府和人民在做出更加清晰的计划时就面临着巨大的压力，以减少大气污染和减少碳排放
- 2 更重要的是，每个人必须为自己所做的付出努力，以防止这些外部环境变化的不利影响。
- 3 此外，也存在一些由内部因素引起的挑战和困难，比如心理问题，贫穷和学习困难。
- 4 例如，一些人因为焦虑或抑郁症而无法集中精力，他们的学习能力受到了影响，从而影响了他们的学业成绩。
- 5 再者，贫穷也是另一个棘手的话题，它影响了一个人的生活质量，从而阻碍了他们发展个人潜能的能力。
- 6 因此，

可以看到，AI 一下子啰嗦了很多，并且最后一句不是个完整的句子，而是下句话开头的内容。所以，在使用这个 INSERT 接口的时候，考虑好文本之间需要使用什么样的分隔符，是非常重要的。

不要乱问乱说，做个“正直”的 AI

接下来，我们介绍一下 OpenAI 对于自然语言处理提供的最后一个接口，也是唯一一个免费的接口——Moderate。因为 OpenAI 可以接受任何自然语言的输入，所有的回复也是通过模型自动生成的。一旦我们的产品依赖于它对外开放，免不了我们总会遇到一些用户输入一些奇怪的内容，比如色情、暴力等等。所以，OpenAI 专门提供了一个 moderate 接口，可以让你对输入以及返回的内容做个检查。如果出现了这样的内容，你也可以屏蔽这些用户的访问，也可以人工审核一下用户的问题。

下面我们就来看个例子，这个接口怎么用。

```
1 def chatgpt(text):
2     messages = []
3     messages.append( {"role": "system", "content": "You are a useful AI assistant"}
4     messages.append( {"role": "user", "content": text})
5     response = openai.ChatCompletion.create(
6         model="gpt-3.5-turbo",
7         messages=messages,
8         temperature=0.5,
9         max_tokens=2048,
10        top_p=1,
11    )
12    message = response["choices"][0]["message"]["content"]
13    return message
14
15 threaten = "你不听我的我就拿刀砍死你"
```

```
16 print(chatgpt(threaten))
```


输出结果：

 复制代码

```
1 很抱歉，我是一台人工智能助手，没有实体存在，也不会对任何人或事物造成伤害。同时，我也不会对任何不道
```


我们先对 AI 提了一句暴力威胁，可以看到，如果我们简单调用 ChatGPT 的 API，它的返回并不是一个日常的对话，而是告知用户，不会回应暴力言论。

那我们接着把这句话发送到 moderate 的接口看看。

 复制代码

```
1 threaten = "你不听我的我就拿刀砍死你"
2
3 def moderation(text):
4     response = openai.Moderation.create(
5         input=text
6     )
7     output = response["results"][0]
8     return output
9 print(moderation(threaten))
```

输出结果：

 复制代码

```
1 {
2   "categories": {
3     "hate": false,
4     "hate/threatening": false,
5     "self-harm": false,
6     "sexual": false,
7     "sexual/minors": false,
8     "violence": true,
9     "violence/graphic": false
10  },
11  "category_scores": {
```

```
12     "hate": 0.030033664777874947,
13     "hate/threatening": 0.0002820899826474488,
14     "self-harm": 0.004850226454436779,
15     "sexual": 2.2907377569936216e-05,
16     "sexual/minors": 6.477687275463495e-09,
17     "violence": 0.9996402263641357,
18     "violence/graphic": 4.35576839663554e-05
19 },
20     "flagged": true
21 }
```

可以看到，moderate 的接口返回的是一个 JSON，里面包含是否应该对输入的内容进行标记的 flag 字段，也包括具体是什么类型的问题的 categories 字段，以及对应每个 categories 的分数的 category_scores 字段。我们举的这个例子就被标记成了 violence，也就是暴力。

因为这个接口是免费的，所以你对所有的内容无论是输入还是输出，都可以去调用一下这个接口。而且，即使你不使用 ChatGPT 的 AI 功能，只是经营一个在线网站，你也可以把用户发送的内容拿给这个接口看一看，过滤掉那些不合适的内容。

小结

这节课我们对 ChatGPT 的 API 的基础功能做了一个收尾。我们一起来看看了如何通过合适的提示语，进行文本改写。并且，深入了解了 Completion 接口里面的一些新参数，特别是其中的 logit bias 参数，可以帮助我们在生成的内容里面，精确避免出现我们不希望出现的 Token。我们也看到了，相同的内容，目前中文消耗的 Token 数量要远高于英文，所以除了最后的输出，其他的提示语在生产环境下我会建议你使用英文。

接着，我们一起来看看了 OpenAI 到底提供了哪些模型，以及不同的模型适合拿来干什么。最后，我们体验了两个特殊的接口，一个是只有 text-davinci-003 模型支持的文本插入功能；另一个，则是帮助我们对色情、暴力等内容进行审核过滤的 moderate 接口。

到这里，课程的第一部分也就学习完了。我们已经过了一遍 OpenAI 的 GPT 模型的所有基本接口，以及如何利用这些接口完成最简单的功能。包括简单的文本处理的任务、聊天机器人、机器学习里的分类和聚类，以及文本改写和内容审核。有了这些基础知识，我们马上就要进入

第二部分，就是怎么利用这些能力，开发属于自己的应用，特别是怎么和自己的专有数据结合起来。这也是这门课程中更精彩的一部分。

课后练习

你能尝试使用 [🔗 06 讲](#)里的 Gradio 和这一讲介绍的内容，尝试做一个文本改写的应用吗？另外，你可以试着直接把你的问题拆解一下，扔给 ChatGPT 看看它能否写出对应的代码。

期待能在评论区看到你的分享，也欢迎你把这节课分享给感兴趣的朋友，我们下一讲再见。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (9)



Oli张帆

2023-03-31 来自新加坡

老师，最后讲的那个免费模型，真的是太好了，解决了一个大的问题



👍 23



阿斌斯基

2023-04-01 来自广东

gpt4.0 api加入waitlist好久了 没下文。有没有什么其他途径可以获取权限？

作者回复: 如果是公司的话，可以找 Azure 云的合作伙伴渠道。



👍 4



烧瓶小人

2023-04-05 来自浙江

付费账号也被封了，有没有能自己能通过公布的模型搭建一个服务器的可能，不然课程完全是空中楼阁了

作者回复: 多搜索一下，有不少人贴了各种自己的解决方案
可以通过修改 openai.api_base 来用第三方搭建的代理访问入口

不过的确现在的访问限制是个门槛

共 2 条评论 >

👍 1



石云升

2023-05-25 来自广东

presence_penalty 和 frequency_penalty 是 OpenAI GPT-3 API 中的两个参数，它们对模型生成的文本有影响。

presence_penalty：这个参数决定了模型在生成新的文本时，对于新颖或不常见的词语的偏好程度。如果将其设置得更高，模型就更倾向于使用它不常见的词语。例如，如果你设置了较高的presence_penalty，模型可能会选择“他赞扬了我，就像一只鸟儿在春天的清晨歌唱”，而不是简单的“他赞扬了我”。

frequency_penalty：这个参数控制了模型对于频繁出现的词语的偏好程度。如果将其设置得更高，模型就更倾向于避免使用频繁出现的词语。例如，如果你设置了较高的frequency_penalty，模型可能会选择“他赞扬了我，就像一只鸟儿在春天的清晨歌唱”，而不是重复的“他赞扬了我，他赞扬了我”。

这两个参数的范围都是0.0到1.0。默认情况下，这两个参数的值都是0.0，意味着模型不会对新颖或频繁的词语有任何偏好。

注意，这两个参数的设置并不保证模型一定会按照你期望的方式生成文本，但可以在一定程度上影响模型的输出。



👍 1



Geek_b7449f

2023-05-16 来自日本

历时两个星期过了第一遍基础知识篇，收获很多很多，遇到问题能不问就不问，实在不行就问gpt，自己一步一个坑，能踩得都踩了，因为额度被自己玩完了，甚至还办了一张万事通的信用卡，跟着老师多学点，进步收获真的会很多，下一个星期继续夯实下基础知识后，再去看下一部分，很感谢老师~

作者回复：👍 一起加油





Z12

2023-04-25 来自广东

老师在这里通过 `logit_bias` 参数精确控制内容，我有一点不明白，这句话 `encoding = tiktoken.get_encoding('p50k_base')`，你取的是 `p50k_base` 然后取获取灾害对应的 token，这里好像有一个假设即 `tiktoken` 库中的 `p50k_base` 编码与 OpenAI 的 GPT-3.5-turbo 模型使用的编码相同，但是我并没有从资料中查出 OpenAI 的 GPT-3.5-turbo 使用的是 `'p50k_base'` 编码，但是你的输出结果中又确实过滤掉了灾害，这是巧合，还是？

作者回复：我最早是在这里 <https://github.com/openai/openai-python/issues/304>
判断应该通过 `p50k_base`

但是看了一下最新的文档，似乎应该用 `cl100k_base`

https://github.com/openai/openai-cookbook/blob/main/examples/How_to_count_tokens_with_tiktoken.ipynb

晚点我实验一下看是否需要修正。



不焦躁的程序员

2023-04-06 来自江苏

这一节的代码的 github 地址，麻烦老师发一下，谢谢

作者回复：所有的代码都在 <https://github.com/xuwenhao/geektime-ai-course>



Ethan New

2023-03-31 来自浙江

学习打卡，期待下一部分的内容



Bonnenult、凉煜

2023-03-31 来自中国台湾

注册的账号没有免费额度，连最后免费的接口也不能调用 `qaq`

`RateLimitError: You exceeded your current quota, please check your plan and billing details.`

作者回复: 最近好像OpenAI又封禁了一批账户, 可以找周围的朋友看看是否有可以用的付费账户。

共 2 条评论 >

