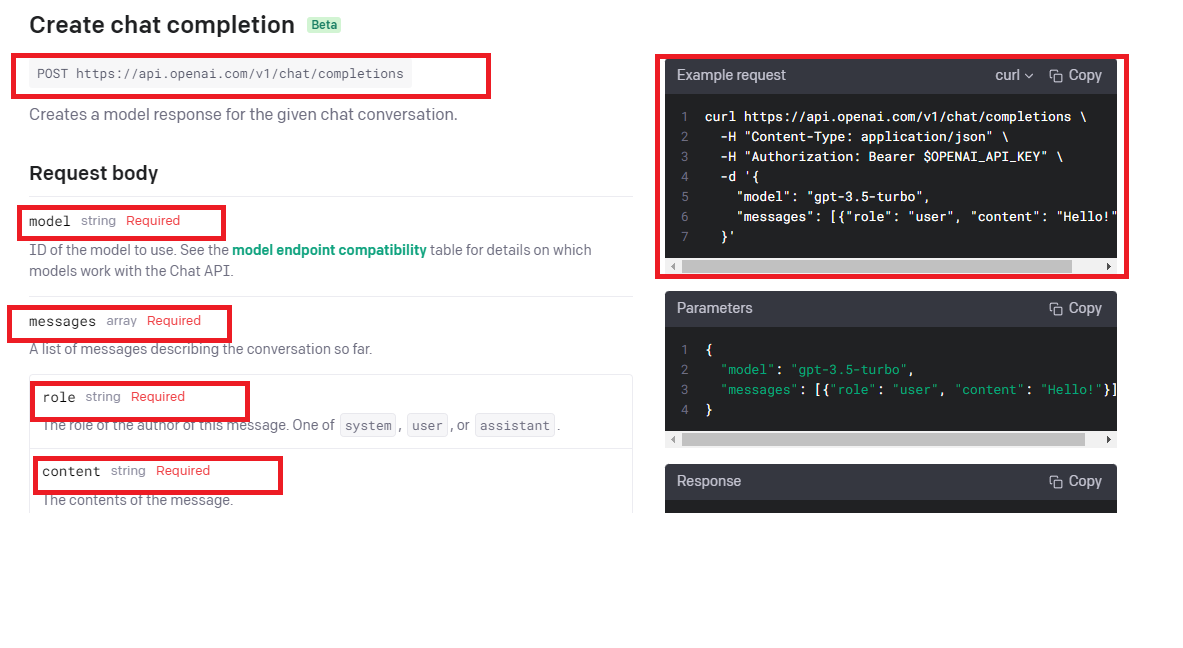
ChatGPT在Unity中的调用方法及流式返回

1. 阅读[官方文档](https://platform.openai.com/docs/api-reference/completions/create)，查看调用API所需传入的参数，重要部分如图所示



* 1. url

右侧最上面表示以POST形式向  
<https://api.openai.com/v1/chat/completions>发送。

* 1. Header部分

首先查看右侧示例中的信息“-H”为Header部分，其中“Content-Type: application/json”为固定值，表示以Json格式发送，“Authorization: Bearer $OPENAI\_API\_KEY”中的$OPENAI\_API\_KEY部分为API Key。

* 1. Body部分

该部分可从左侧“Request body”中查看，标注部分为重要部分。其中model为模型名称，GPT3.5的模型为“gpt-3.5-turbo”。Messages内包含发送的信息，role通常为“user”即可，content为要发送的内容，Messages为数组，可以发送历史记录以结合上下文回答。

1. JsonUtility是Unity自带的Json解析库，使用方法可参考：[Unity | 使用JsonUtility解析Json](https://blog.csdn.net/weixin_39766005/article/details/103926877?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-103926877-blog-120602106.235%5Ev35%5Epc_relevant_increate_t0_download_v2&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-103926877-blog-120602106.235%5Ev35%5Epc_relevant_increate_t0_download_v2&utm_relevant_index=2)，接下来创建实体类，用来存储待发送的数据和接受的数据

2.1发送前的数据

Json格式为

文本

描述已自动生成

需要创建两个类来存储，messages为数组，因此也需要使用数组来存储。

文本

描述已自动生成

注意：若不带[Serializable]则无法解析！！！

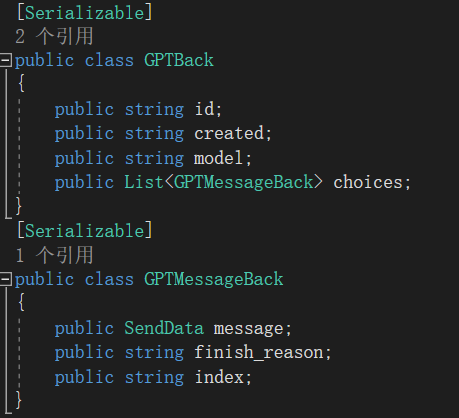
* 1. 发送后的数据存储

GPT3.5返回的数据Json格式为：

文本

描述已自动生成

这里只存储id、created、model和choices部分，其余部分若有需要请自行添加字段。



1. 构建发送与接收
   1. 首先在脚本创建所需变量

图形用户界面, 文本

描述已自动生成

apiurl根据官方文档即为<https://api.openai.com/v1/chat/completions>。

key为Open API Key。

datalist用来存储信息历史记录.

* 1. 发送

创建一个协程GetPostData用来发送和接受，参数为一个string类型的postWord，及要发送的文字。



在协程中首先将postWord存储到datalist中，role为“user”，content为“postWord”

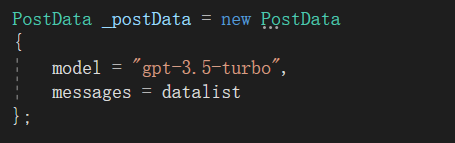
手机屏幕的截图

中度可信度描述已自动生成

使用using创建一个UnityWebRequest的对象，用来向HTTP发送信息，使用using可以确保能够自动释放，不必手动调用Dispose()方法。



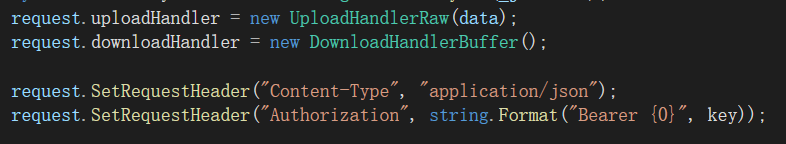
接下来将需要发送的信息转换为Json格式，首先实例化PostData实体类将数据存储在其中。



然后使用JsonUtility.ToJson（）方法将\_postData转换为Json格式，用string类型的\_jsonText来存储，再将\_jsonText转换为byte[]格式。



接下来准备发送信息，request.uploadHandler为data，并设置Header信息。



最后发送信息

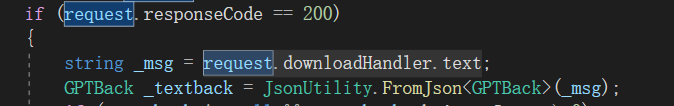


* 1. 接受信息

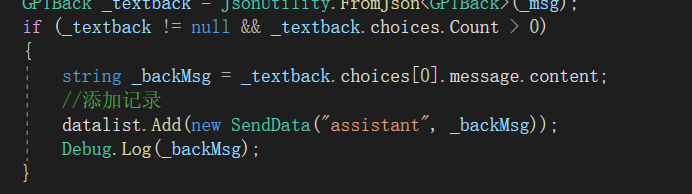
首先判断request.responseCode，为200时即为成功 。



由于返回类型为Json格式的，需要将其解析，使用request.downloadHandler.text将返回的数据取出，存储在string类型的变量\_msg中，然后实例化实体类GPTBack类，将Json数据解析到其中。

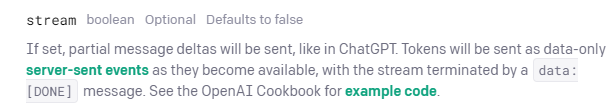


即可使用\_textback.choices[0].message.content提取出GPT的回答，并作出处理。



1. 流式返回

OpenAI的官方文档中有表明可以将数据进行流式返回。

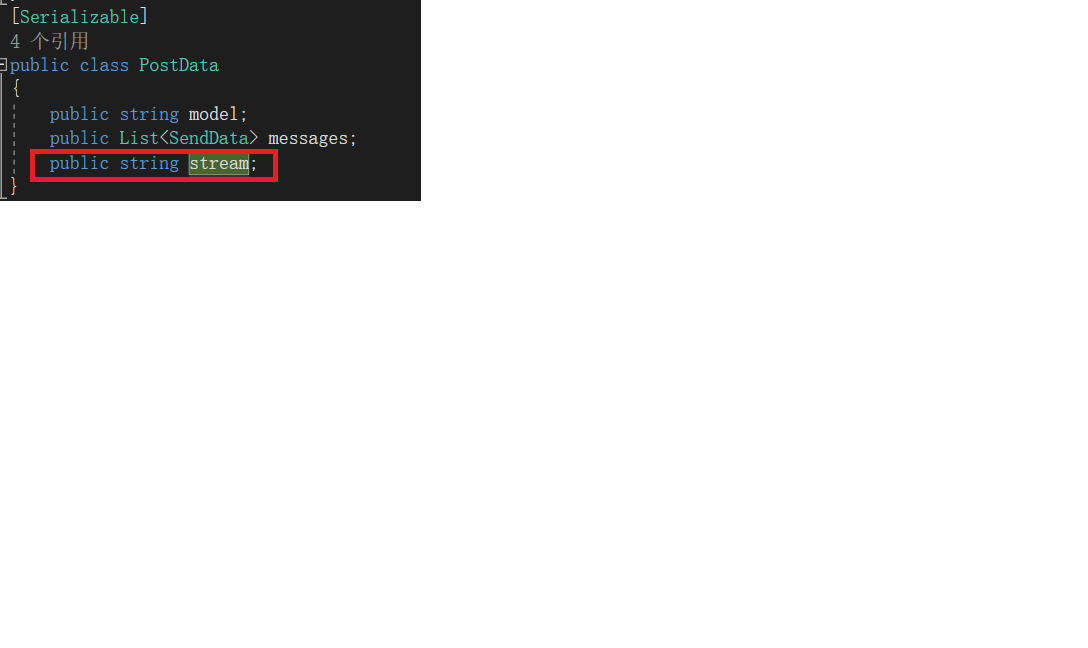


翻译：

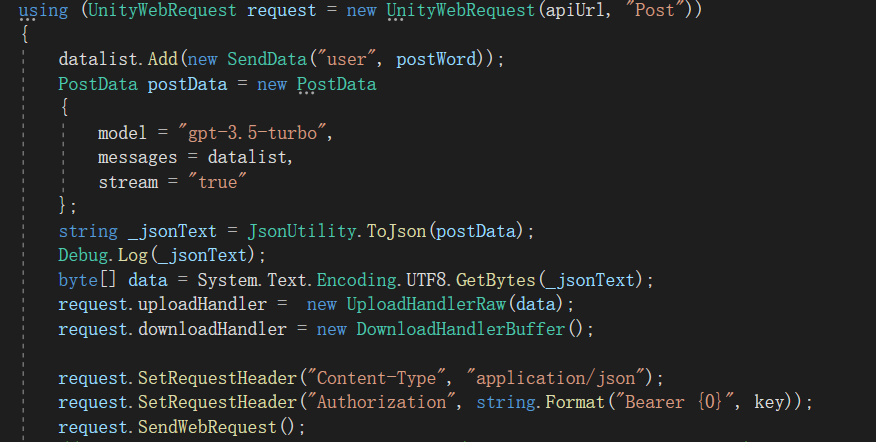
文本

中度可信度描述已自动生成

* 1. 在发送数据时加上stream：”true”即可将数据进行流式返回，在PostData实体类中加上stream



在发送创建PostData对象时加上stream = "true"，需要注意流式返回为分数次返回，因此不可使用yield return request.SendWebRequest()等待，直接使用request.SendWebRequest()发送。

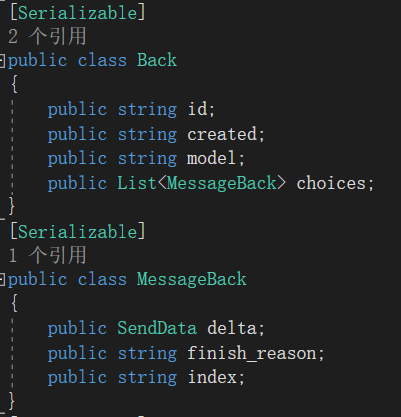


* 1. 接受信息类的处理

流式返回时会被Unity存储为如下图所示：

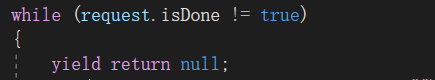


流式返回可以使用循环分多次提取，但由于为分为多个Json形式返回，所以这里只管Json部分，不管“data：”部分，由于Json格式在名称方面与之前并不相同，所以需要重新构建接受数据的实体类，如图：



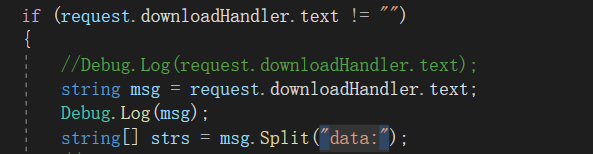
* 1. 接受信息

由于流式返回，所以使用while循环不断提取，可使用yield return new WaitForSeconds()来控制每次提取的时间，这里使用yield return null即每帧提取一次，运行条件为request.isDone != true

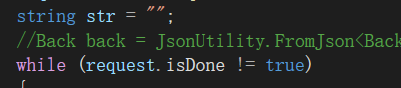


提取时首先判断request.downloadHandler.text是否为空，不为空即可继续。

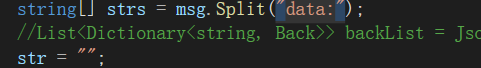
使用string类型的变量msg存储返回的信息，再使用msg.Split("data:")可将msg以"data:"为间隔分成数组。这样每个数组存储的就是一个Json格式的数据。



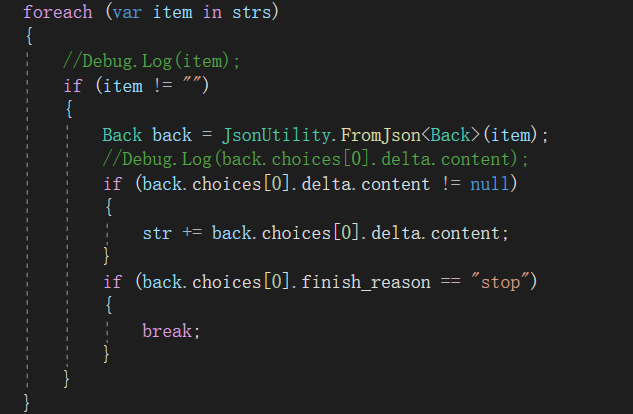
在循环体创建一个string类型的变量str，用来存储GPT已经返回的回复文字。



在分化为数组后将str清除，准备存储目前已经返回的数据



接下来遍历strs数组即，存储不同Json的数组，判断数组中存储的数据是否为空，若不为空，则开始解析Json数据，解析后提取出回复内容并将其追加在str中。



在结束后，即while循环执行完后，使用将信息存储在历史记录中。



str即为当前已经返回的数据，每次foreach结束对str处理即可。