Reader 开发文档

目标:

1, 快速提取信息，代替阅读

2，压缩信息，提升信息密度

3，辅助GPT4 进行信息解析

软件功能v1：

1. 使用GPT系列API进行信息处理
2. 使用终端录入信息，并进行压缩

项目基于面向对象编程范式开发

项目文件：

Setting.json #API秘钥，等各种设定信息

GPT\_API.py #一个用于调用GPT\_API的库

Story\_Board.py #一个用于存储和记录对话信息的库

Main.py #主程序，初始化程序并调用其他库

GPT\_API.py 是一个用于调用gpt3 API的库，其包含网络访问功能，由其他库进行初始化和调用

CLASS GPT\_API

* **\_\_init\_\_(self, api\_key: str)**: 初始化GPT\_API类，需要一个API秘钥。
* **set\_model(self, model: str)**: 设置GPT模型，例如"gpt-4"。
* **query(self, prompt: str, temperature: float, max\_tokens: int) -> str**: 发送请求给GPT模型，根据给定的提示生成回复。接收提示、温度和最大令牌数作为参数，并返回生成的回复。

Story\_Board.py 一个用于存储和记录对话信息的库，它可以将对话信息存储在数据结构中，方便查询和发送给GPT进行请求回复

CLASS StoryBoard

* **\_\_init\_\_(self)**: 初始化StoryBoard类。
* **add\_dialogue\_entry(self, user\_input: str, ai\_response: str)**: 添加对话条目，包括用户输入和AI回应。
* **get\_dialogue\_history(self) -> List[DialogueEntry]**: 获取对话历史记录。