# **北京信息科技大学课程报告**

报告题目：开源项目建设发布

课程名称：开源软件开发

任课教师：李宁

项目导师：王绎豪

小组组长：张景元

## **团队成员**

（请按照 姓名，学号，班级的格式分行给出，组长写在第一行）

张景元 2023011295 软工2301

陈佳圻 2023011287 软工2301

张曦文 2023011214 计科2302

韩欣雨 2023011219 计科2303

胡雯宇 2023011221 计科2303

## **报告要求**

建立一个开源软件项目，并在项目托管平台Github上发布。

报告内容包括以下方面:

\*项目名称

\*项目描述

\*项目许可证

\*项目人员和分工

\*项目开发流程

\*代码库，网站，Wiki，邮件列表，缺陷追踪系统，文档编制

\*项目计划（版本、用户、推广)

\*其他

报告篇幅为1000字以上。格式清晰完整，标点正确，无错别字。

## **项目计划**

以下各项均为必填

项目名称

小说生成器

项目描述

智能小说生成器是一个基于自然语言处理技术的应用，旨在帮助用户快速生成小说的章节内容和大纲。通过用户提供的指令和参数，系统能够利用预训练的语言模型生成连贯且具有逻辑性的文本，支持用户在创作过程中的灵感激发和内容构建。

技术选型

1. 编程语言和框架: 项目主要使用Python语言，利用异步编程库`asyncio`处理并发任务。

2. 数据验证与模型定义: 使用`pydantic`库定义数据模型，确保生成的小说结构合法和清晰。

3. 文本生成: 基于自定义模块`metagpt`中的`ActionNode`和`LLM`，利用预训练语言模型生成文本内容。

关键功能与流程

1. 用户输入: 用户通过界面输入小说名称、背景设定、角色等创作指令。

2. 数据模型填充: 系统使用`pydantic`将用户输入映射到数据模型中，如`Novel`和`Chapters`。

3. 智能生成: 基于用户提供的信息和预训练语言模型，生成小说的章节内容和大纲。

4. 输出结果: 将生成的小说章节内容和大纲以指定格式输出，方便用户查阅或进一步编辑。

许可证（以及为什么选择这个许可证）

BSD 3-Clause License

BSD许可证的灵活性使得开发者可以将其与其他许可证结合使用，以满足特定的需求。例如，开发者可以将BSD许可证的代码与GPL许可证的代码结合使用，从而创建一个混合许可证的软件。

人员（仅需姓名）和分工

张景元：项目文档 项目构思 整合报告

陈佳圻：制作ppt 选择许可证 项目描述和计划

张曦文：代码开发

韩欣雨：代码开发 项目汇报

胡雯宇：代码开发

开发流程（项目构思，环境部署，代码编写，代码测试，文档撰写，项目发布等过程）

**项目构思：**

项目目标

开发一个智能系统，能够根据用户提供的指令和参数生成小说的章节内容和大纲。

技术选型

1.语言和框架: Python语言，利用异步编程库asyncio处理并发任务。

2.数据验证与模型: 使用pydantic库定义数据模型，确保生成的小说结构清晰、合法。

3.生成文本: 基于自定义模块metagpt中的ActionNode和LLM，利用预训练语言模型生成文本内容。

关键功能与流程

1.用户指令输入: 用户输入写作指令，例如小说名称、背景设定、角色等信息。

2.数据模型填充: 使用pydantic将用户输入的指令映射到数据模型中，如Novel和Chapters。

3. 文本生成与填充: 使用ActionNode和LLM模块，基于用户输入和模型填充生成小说内容。

4. 结果输出: 将生成的章节内容和大纲以指定格式输出，可用于展示或进一步处理。

技术实现步骤

1. 定义数据模型:

- 创建`Novel`、`Chapter`和`Chapters`类，包含小说的名称、章节内容、大纲等信息。

2.编写异步函数:

- 实现`generate\_novel()`函数，使用`ActionNode`和`LLM`填充数据模型，生成小说内容和章节。

3. 用户界面设计:

- 可以考虑实现一个简单的命令行界面或Web界面，接收用户的写作指令和参数输入。

4.异常处理与优化:

- 考虑用户输入错误或异常情况的处理，如缺失必要信息或格式错误的指令处理。

5. 测试与部署:

- 对系统进行全面测试，确保生成的小说内容符合预期。

- 部署到合适的环境中，确保系统可用性和稳定性。

预期成果

开发完成后，用户可以通过输入指令和参数，使用该系统生成定制化的小说章节和大纲。系统通过结合用户提供的信息和预训练语言模型，生成符合逻辑和情节连贯的文本内容，从而支持用户的创作和写作过程。

这个项目不仅可以作为创作辅助工具，也可以为学习自然语言处理和预训练语言模型的开发者提供一个有趣的实践场景。

**环境部署：**

Python3.9

MetaGPT

**代码编写：**

1. Shebang和编码声明：

```python

#!/usr/bin/env python

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

```

- `#!/usr/bin/env python` 表示这是一个可执行的Python脚本。

- `# -\*- coding: utf-8 -\*-` 声明了Python源文件使用UTF-8编码。

2. 文档字符串和时间戳：

```python

"""

@Time : 2024/2/1 12:01

@Author : alexanderwu

@File : write\_novel.py

"""

```

- 这段文档字符串提供了代码的说明，包括时间戳和作者信息。

3. 导入模块：

```python

import asyncio

from typing import List

from pydantic import BaseModel, Field

from metagpt.actions.action\_node import ActionNode

from metagpt.llm import LLM

```

- `asyncio` 是Python的异步编程库。

- `BaseModel` 和 `Field` 是 `pydantic` 模块中的类，用于数据验证和解析。

- `ActionNode` 和 `LLM` 来自自定义模块 `metagpt` 中的部分，用于生成小说内容。

4. 定义数据模型：

```python

class Chapter(BaseModel):

name: str = Field(default="Chapter 1", description="The name of the chapter.")

content: str = Field(default="...", description="The content of the chapter. No more than 1000 words.")

class Chapters(BaseModel):

chapters: List[Chapter] = Field(

default=[

{"name": "Chapter 1", "content": "..."},

{"name": "Chapter 2", "content": "..."},

{"name": "Chapter 3", "content": "..."},

],

description="The chapters of the novel.",

)

class Novel(BaseModel):

name: str = Field(default="The Lord of the Rings", description="The name of the novel.")

user\_group: str = Field(default="...", description="The user group of the novel.")

outlines: List[str] = Field(

default=["Chapter 1: ...", "Chapter 2: ...", "Chapter 3: ..."],

description="The outlines of the novel. No more than 10 chapters.",

)

background: str = Field(default="...", description="The background of the novel.")

character\_names: List[str] = Field(default=["Frodo", "Gandalf", "Sauron"], description="The characters.")

模型定义：

Chapter, Chapters, Novel 是基于 BaseModel 的数据模型，使用 pydantic 库进行定义。它们描述了小说的结构，包括章节名称、内容，小说名称、背景、角色等重要元素。

异步生成函数 generate\_novel()：

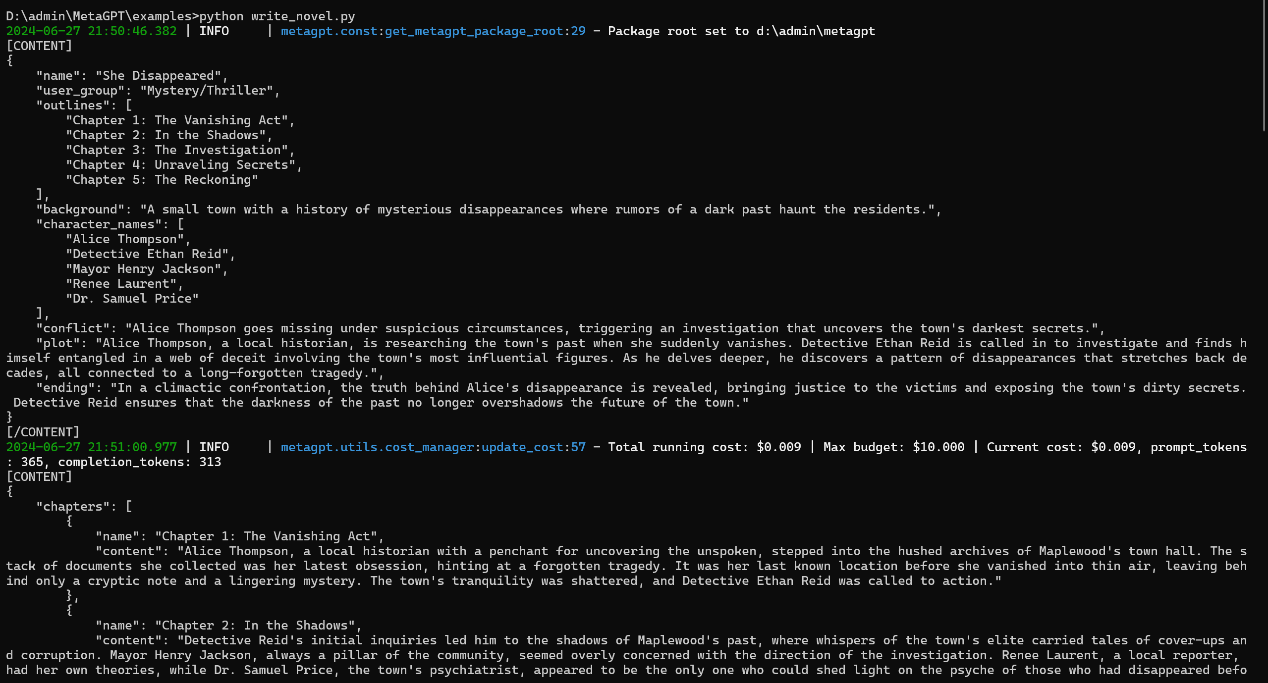
定义了一个异步函数 generate\_novel()，用于生成小说的章节和大纲。

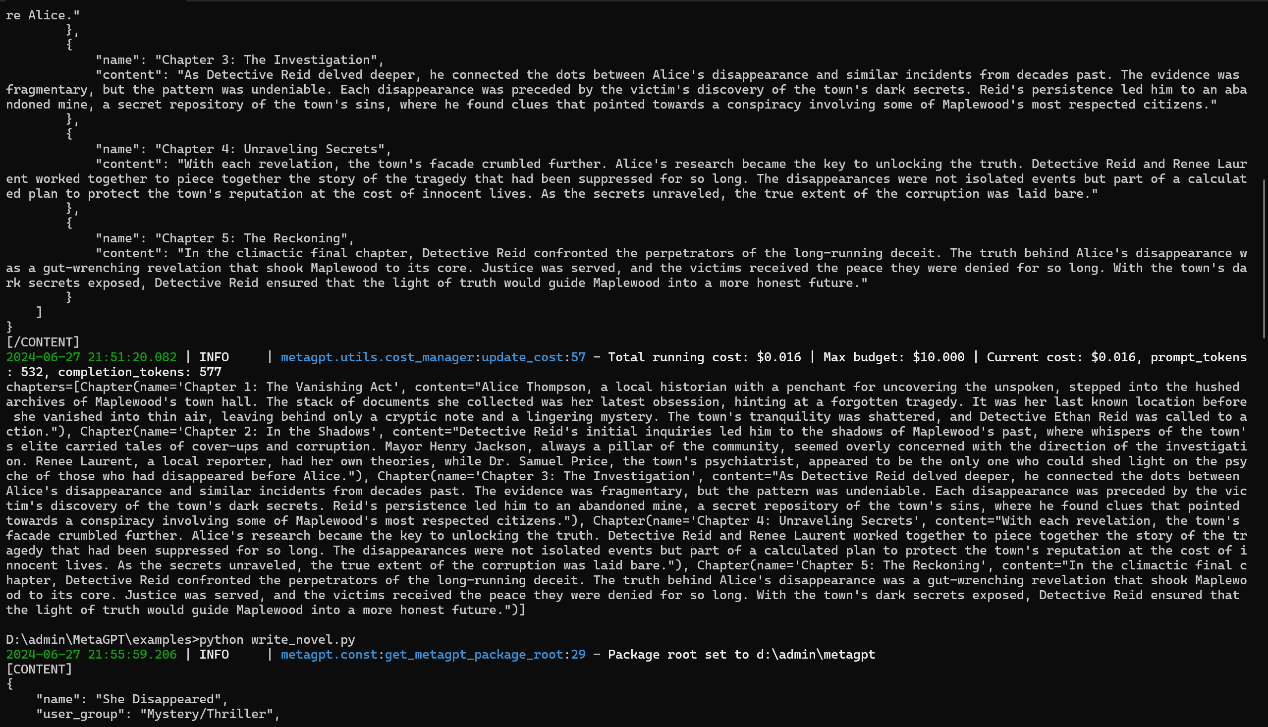
instruction 变量包含了生成小说的指令和激励信息。

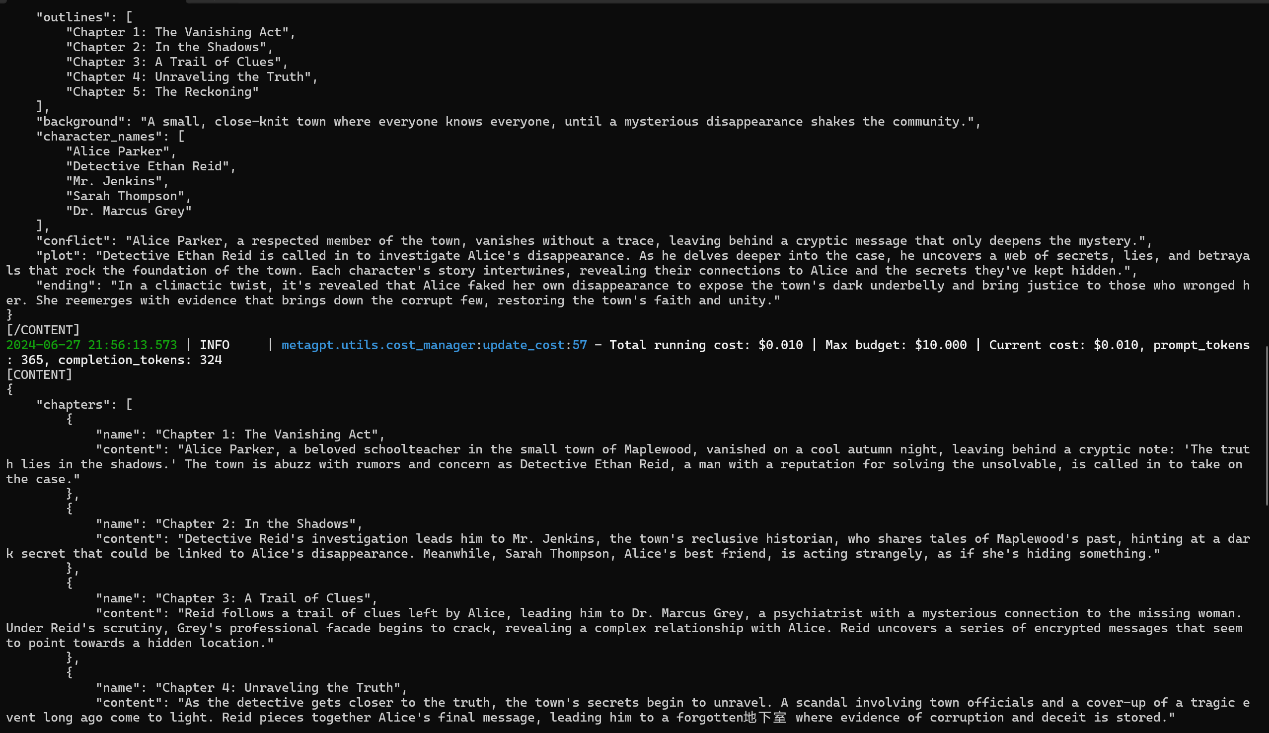
ActionNode.from\_pydantic() 方法利用 LLM 对象填充数据模型，并生成相应的内容。首先填充 Novel 模型以获取完整的小说内容，然后将其作为指令内容填充到 Chapters 模型中。

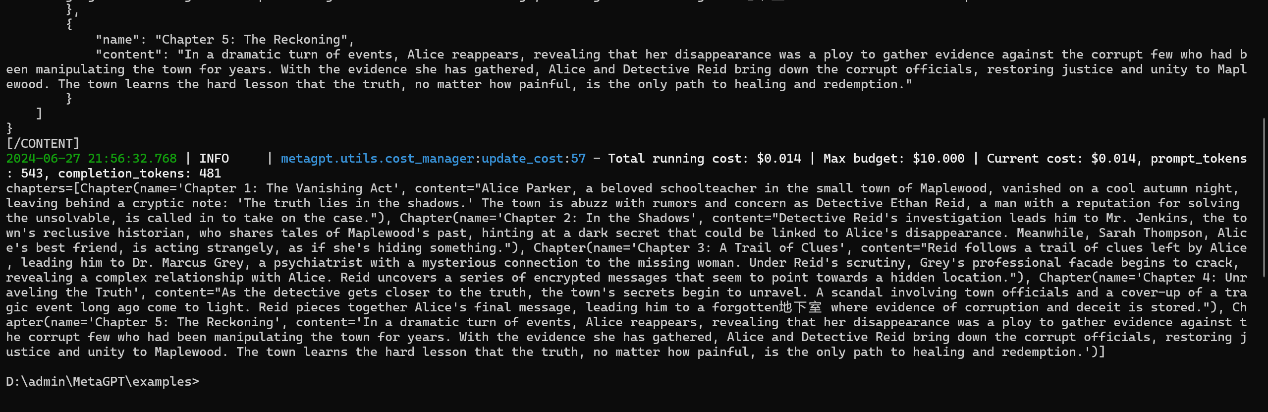
最后，打印出生成的章节和大纲内容 (chap\_node.instruct\_content)。

**代码测试：**









项目信息（给出各项所用的平台、软件及地址）

代码库：https://github.com/Bistu-OSSDT-2024/6-Novel-generator/blob/main/write\_novel.py

网站：GitHub

Wiki：无

邮件列表：无

缺陷追踪系统：无

文档：无

项目计划（版本发布的计划、面向的用户、如何推广)

面向对象：初级用户：提供直观的用户界面和简洁的操作流程，确保用户能够快速上手。

专业用户：提供高级功能和定制选项，满足专业用户在工作中的特殊需求。

开发者：提供清晰的API文档和开发者社区支持，鼓励开发者基于本项目进行二次开发或贡献代码。

教育机构：提供教育版或免费授权，支持学校和教育机构的教学和研究工作。

如何推广：1）技术社区合作：与GitHub、GitLab等技术社区合作，发布项目信息、接受社区贡献并举办技术交流活动。

（2）博客和文章：撰写博客和文章分享项目经验、技术细节和使用心得等内容，吸引更多潜在用户了解和使用本项目。同时，鼓励用户分享自己的使用体验和心得到社交媒体平台。

（3）案例展示：收集并展示成功的用户案例和应用场景，展示项目在解决实际问题中的价值和应用前景。这有助于增强用户对项目的信任和兴趣。

（4）定期更新和维护：保持项目的持续更新和维护状态，及时修复已知问题和添加新功能。这有助于提升用户体验和忠诚度，并吸引更多用户长期使用和推荐给他人使用。

## **项目文档**

以下内容无需在此填写，请将内容填写到代码仓库中的对应文件。

如果没有内容则文件可以为空，但必须有这些文件存在。

INSTALL

配置、编译和安装该项目的说明信息

FAQ

关于本项目的纯文本格式的常见问题解答。

CREDITS

本项目所有贡献者的列表。

HISTORY

本项目的历史发展演变记录。

LICENSE

本项目的许可证条款文件。

README

本项目的说明。

## **项目开发**

选择一个熟悉的语言开发项目。参考课堂派上提供的范例。

遵守课程讲义中的注意点：

* 按规则命名事物
* 注重编码风格
* 选好工具、语言和平台
* 写好文档
* 做好发布前的准备

本部分无需填写。代码直接提交到代码仓库。

## **项目发布**

1. 注册Github账号。
2. 建立Github代码仓库。
3. 将项目代码发布到Github。
4. 发布项目文档到Github。建议使用Markdown格式。
5. 项目成员的协作需要体现到Github上，也就是小组中的每个成员要加到项目里，并都要有提交。
6. 请在以下填写你发布后的github项目地址（必须），项目必须为公开项目。

（特别重要）

https://github.com/Bistu-OSSDT-2024/6-Novel-generator

## **参考文献**

请列出本报告参考的资料来源。