文本处理与应用：AIGC 辅助笔记生成

一、研究背景

在数字化时代，视频已成为我们获取信息和知识的重要途径。随着在线教育平台的兴起，视频内容的丰富性和互动性使其成为学习者的首选资源。然而，视频内容的海量增长也带来了信息筛选和整理的难题。学习者在面对众多视频资源时，往往需要花费大量时间和精力去筛选、整理和吸收关键信息，这一过程不仅效率低下，而且容易遗漏重要知识点。为了解决这一问题，文本处理与应用领域的研究者开始探索如何利用人工智能技术，特别是人工智能生成内容（AIGC）技术，来辅助学习者更高效地从视频中提取和整理信息。

AIGC 技术的核心在于利用机器学习、自然语言处理等技术，模拟人类的创作过程，自动生成文本内容。在视频学习领域，AIGC 技术的应用前景尤为广阔。通过将视频字幕与提示词相结合，输入到文本大模型中，AIGC 能够理解视频内容，并为学习者生成视频的大纲和相关高频词汇。这一过程不仅能够提高学习者的笔记整理效率，而且能够帮助他们快速把握视频的核心要点和关键信息。

为了实现这一目标，首先需要通过爬虫技术获取视频字幕。视频字幕作为视频内容的文字表现形式，包含了视频的核心信息。通过爬虫技术，可以自动地从视频平台获取字幕数据，为后续的文本处理提供基础。获取字幕数据后，需要将这些数据与提示词相结合，输入到文本大模型中。提示词是学习者在整理笔记时关注的关键点，它们可以帮助文本大模型更准确地理解视频内容，从而生成更符合学习者需求的大纲和高频词汇。

文本大模型是 AIGC 技术的核心，它通过深度学习模型，能够理解和生成自然语言文本。在本研究中，文本大模型将根据输入的字幕数据和提示词，自动分析视频内容，提取关键信息，并生成视频的大纲和高频词汇。这一过程不仅能够减轻学习者的负担，而且能够提高笔记的质量和准确性。

二、方法论

2.1 爬虫

爬虫技术是数据采集的基石，尤其在文本处理与应用领域，它的作用不可或缺。爬虫，也称为网络爬虫或网页蜘蛛，是一种自动遍历互联网资源的软件程序。它通过模拟浏览器请求，获取网页内容，并从中提取出有价值的数据。

核心实现原理：爬虫的基本工作流程包括发送 HTTP 请求、接收响应内容、解析 HTML/XML 等格式的数据，并提取所需信息。这一过程通常由以下几个步骤组成：选择目标网站、确定爬取策略、设计爬取规则、实现数据存储。

2.1.1 目标选择

在当前的数字媒体生态中，bilibili 作为一家领先的视频分享平台，以其独特的中长视频内容而脱颖而出。相较于其他视频平台，bilibili 在内容质量上有着显著的优势，尤其是在教程和教学类视频领域。这些视频不仅涵盖了广泛的主题，从编程到烹饪，从艺术到科学，而且往往由领域内的专家或爱好者精心制作，确保了内容的专业性和实用性。

因此在本文中，使用 bilibili 为目标网站，来提升学习者的效率和体验。目标视频选用教程或教学类视频，模拟人通过视频平台学习知识或技能，通过 AIGC 技术，可以自动生成视频内容的大纲，为学习者提供一个清晰的学习路径。这不仅节省了学习者手动整理笔记的时间，而且通过大纲的形式，使得学习过程更加系统化和条理化。

2.1.2 反爬

由于 b 站有许多反爬虫措施，尤其是针对自动化爬虫行为，文中使用 python 库 selenium，通过模拟真实用户的浏览行为来实现数据的获取，避开部分反爬虫措施。

Selenium 是一个强大的自动化测试工具，它能够模拟用户与浏览器的交互，包括点击、滚动、输入等操作。在本方案中，我们选择使用 Selenium 的无头浏

览器模式，即不显示浏览器界面，直接在后台运行，这不仅节省了资源，而且提高了运行效率。

驱动 Chrome 浏览器配置如下：

```
options = Options()
options.add_argument("--headless")  # 启用无头模式
options.add_argument("--disable-gpu")  # 禁用 GPU 硬件加速
options.add_argument('blink-settings=imagesEnabled=false')  # 不加载图片，提升速度
options.add_experimental_option('excludeSwitches', ['enable-automation'])
options.add_argument('log-level=3')
# 初始化 WebDriver，传入配置的选项
driver = webdriver.Chrome(options=options)
```

2.1.3 爬取 B 站字幕原理

B 站的视频字幕分为 CC 字幕、视频内置字幕、AI 生成字幕。目前在网络中可查找到的都是下载视频的 CC 字幕（下载 CC 字幕不需要登陆 b 站，通过 API 传入视频参数直接可以访问下载），但是绝大部视频是没有 CC 字幕的。

图 1 视频内置字幕和 AI 生成字幕

而视频内置字幕提取难度相对较大，所以选择 b 站 AI 生成的字幕，并且绝大部分的视频都有 AI 生成字幕。以为 b 站视频 BV1zX4y1c74i 为例（BV1zX4y1c74i 为 b 站是的 id，即视频唯一索引，视频 URL 为 https://www.bilibili.com/video/BV1zX4y1c74i/）。

结合 B 站开发平台和 github 开源收集的 API，本文使用如下 API 获取视频的信息。

```
https://api.bilibili.com/x/web-interface/view?bvid=
```

这个 API 返回的数据为 json 格式，其中的 bvid、title 为我们需要的数据，

即通过这个 API 通过视频的 BV 号获取其唯一的 bvid 和 title，通过同其中 sections 字段获取合集视频中的每个视频的 cid。

bvid 为每个视频的唯一 id，cid 为视频合集中每个视频的 cid，因此定位合计中每个视频的，需要 bvid 和 cid。

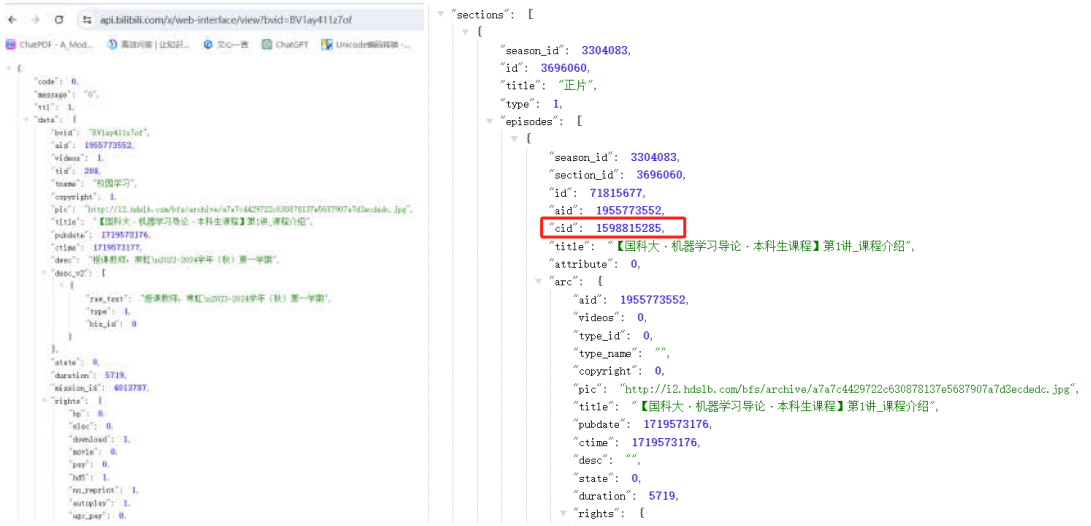

图 2 获取 bvid、title 和 cid

接着我们使用 bvid 和 cid 作为参数，请求下面的 API，即可获取这个视频的 AI 生成字幕地址（也有可能该视频还没有 AI 生成字幕）。

图 3 获取 AI 生成字幕地址

通过上面的流程，我们即可获得视频的 AI 生成字幕地址，使用 requests 库下

载即可。

2.1.4 爬取字幕

由于使用的是无头浏览器，所在调试阶段可以设置显示浏览器。下面是详细的请求 API、登陆 B 站、下载字幕等操作的技术介绍。

1. 获取视频的 bvid、title、cid

由于使用该 API 不需要登陆限制，直接可以获取到返回的 json 文件，对文件进行解析提取即可达到视频对应的 bvid、title、cid 等信息。代码实现是通过 get_video_param 函数实现，传入视频 BV 号，获取的数据写入全局变量 video_title, video_aid, video_total_p, video_cid_list 中，解析使用 xpath 解析（Chrome 自带获取节点 xpath 工具）。

```
def get_video_param(BV):
    global video_title, video_aid, video_total_p, video_cid_list
    driver.get(video_info_api_url.replace("_BV_", BV))
    driver.implicitly_wait(10)
    page_source = driver.page_source
    xml_obj = etree.HTML(page_source)
    json_raw_str = xml_obj.xpath("/html/body/pre/text()")[0]

    if len(json_raw_str) != 0:
        json_data = json.loads(json_raw_str)
        if json_data["message"] != "0":
            logger.error("出现了一些问题：{}".format(json_data["message"]))
            return False
        # 视频数据
        video_title = json_data["data"]["title"]
        video_aid = json_data["data"]["aid"]
        video_total_p = json_data["data"]["videos"]
        if video_total_p > 1:
            for elem_p in json_data["data"]["pages"]:
                video_cid_list.append([elem_p["cid"], elem_p["part"]])
        logger.info("-" * 50)
        logger.success("\n- 视频标题：{}\n- 视频 aid：{}\n- 视频总 P 数：
{}".format(video_title, video_aid, video_total_p))
        logger.info("-" * 50)
    return True
```

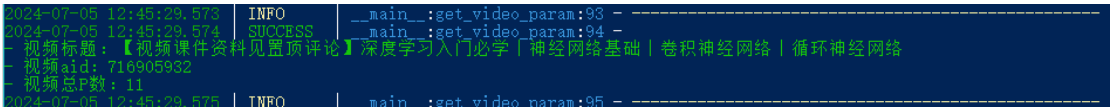

图 4 获取视频信息效果

2. 登陆 b 站

由于通过 bvid 和 cid 请求获取视频信息（包含视频的 AI 生成字幕）的 API 需要登陆（可能是 Cookie 等校验，由于使用无头浏览器就不需要考虑这些了），我们直接通过登陆 b 站首页 https://www.bilibili.com/来进行登陆。详细的流程图如下。

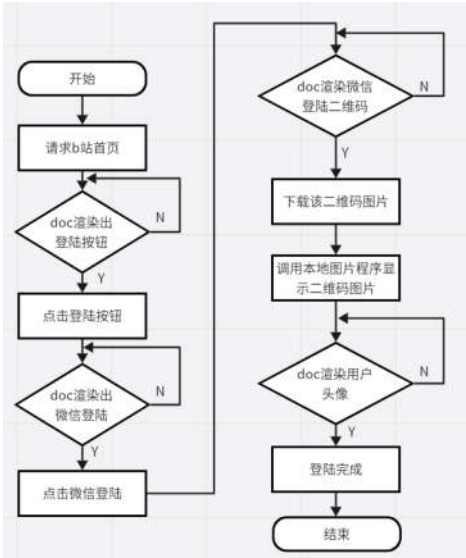

图 5 登陆操作流程图

具体的代码实现如下。主要通过 WebDriverWait(driver, 10).until()等待 document 渲染出我们需要的元素后进行下一步操作。由于采用无头浏览器，无法看到微信登陆的二维码图片，所以将二维码图片下载后显示，用户扫码成功过后即可登陆，登陆成功后浏览器保存了我们后续需要的一系列参数（登陆参数、验证参数等），后续 API 请求可以专注于数据处理。

```
def login_bilibili():
    driver.get("https://www.bilibili.com/")
    driver.implicitly_wait(10)
    # 等待登陆按钮可点击
    element = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.CLASS_NAME, "header-login-entry"))
    )
    element.click()
```

```
    element = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.CLASS_NAME, "login-sns-name"))
    )
    element.click()

    element = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.CLASS_NAME, "web_qrcode_img"))
    )
    # 定位图片元素，这里使用 XPath 作为示例
    image_element = driver.find_element(By.XPATH,
'/html/body/div[1]/span[1]/div[1]/div[3]/div[1]/img')
    # 获取图片的源文件 URL
    image_url = image_element.get_attribute('src')
    response = requests.get(image_url)

    image_path = os.getcwd() + '/b_image.png'
    if response.status_code == 200:
        # 保存图片到本地
        with open(image_path, 'wb') as f:
            f.write(response.content)
    else:
        logger.error(f"图片下载失败，状态码：{response.status_code}")

    subprocess.run(['start', image_path], shell=True)

    logger.info("[info] - 等待登陆...（使用微信扫码登陆，登陆成功后关闭二维
码图片）")
    element = WebDriverWait(driver, 1000).until(
        EC.invisibility_of_element_located((By.CLASS_NAME,
"web_qrcode_img"))
    )
    logger.success("[info] - 登陆成功")
```

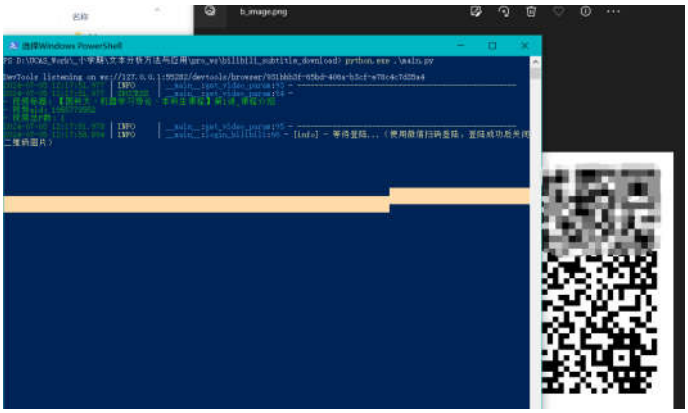

图 6 登陆效果图

登陆成功后，使用 bvid 和 cid 获取视频的 AI 生成字幕地址，将字幕下载保存到当前文件夹下的 subtitle_txt 中。具体代码实现如下，通过解析 json 文件，获取字幕文件 URL，使用 requests 库下载字幕文件。

```
def get_video_subtitle_json(save_json=False):
    global video_cid_list
    total_count, success_count, failed_count = len(video_cid_list), 0, 0
    cur_dir = os.getcwd()
    for idx, elem in enumerate(video_cid_list):
        if str(elem[2]).strip() != "":
            response = requests.get(elem[2])
            subtitle_file_json = cur_dir + "/subtitle_json" + '/{}_&_{}.json'.format(video_title, elem[1])
            subtitle_file_txt = cur_dir + "/subtitle_txt" + '/{}_&_{}.txt'.format(video_title, elem[1])
            logger.info("[info] - 下载 {}_{}".format(video_title, elem[1]))
            if response.status_code == 200:
                if save_json:
                    logger.info("[info] - 写入 json 文件")
                    with open(subtitle_file_json, 'w') as f:
                        f.write(response.text)

                logger.info("[info] - 转为 txt 文件")
                with open(subtitle_file_txt, 'w', encoding="utf-8") as f:
                    for e in json.loads(response.text)["body"]:
                        f.write('{}{}'.format(e["content"], "\n").encode('utf-8').decode("utf-8"))
                success_count = success_count + 1
        else:
            logger.error(f"字幕下载失败")
            failed_count = failed_count + 1
        time.sleep(3)
    return "总计：{}个，下载成功：{}个，下载失败：{}个".format(total_count, success_count, failed_count)
```

图 7 下载字幕文件

字幕文件保存位置和实际内容如图 8。

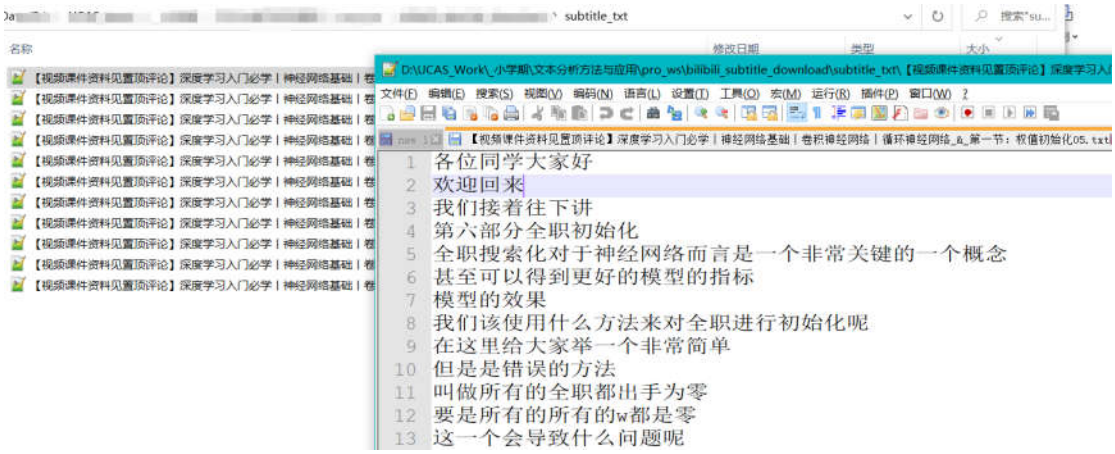

图 8 下载字幕文件

2.2 大模型的使用

本文使用 Kimi 大模型，Python 调用 Kimi 开发平台提供的 API 与 Kimi 实现交互。Kimi 使用 API 交互需要在平台申请 api_key，申请 api_key 后即进行后续请求。

2.2.1 基本配置

使用 openai 库，初始化客户端，传入 base_url 地址和申请的 api_key。

```
from openai import OpenAI
client = OpenAI(
    api_key="xx-xxxx",
    base_url="https://api.moonshot.cn/v1",
)
```

2.2.2 获取可用模型

通过 API 获取可用的模型，判断模型是否可用。代码实现如下，将可用模型数据保存到全部变量 model_data, model_dsc_list 中。

```
def get_ai_model():
    global model_data, model_dsc_list
    model_list = client.models.list()
    model_data = model_list.data

    logger.success("获取到可用模型列表")
    for i, model in enumerate(model_data):
        model_dsc_list.append([i, int(str(model.id).split('-')[-1].replace("k", ""))])
        logger.info("model[{:d}]: {}".format(i, model.id))
  model_dsc_list = sorted(model_dsc_list, key=lambda x: x[1])
```

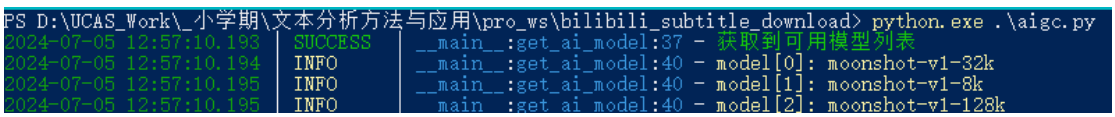

图 9 获取模型类型

2.2.3 生成内容（单次对话）

将提示词和需要解析的字幕数据一起发送给 Kimi，等待 Kimi 的返回，主要是通过 def single_round_session(file_name)函数实现，该函数接收一个字幕数据文件。

1. 选择模型

读取目标字幕文件的大小，并加上提示词大小，然后与可用模型进行比较，选择一个相对合适的模型，加载该模型的参数用于后续使用。代码实现如下：

```
global model_data

logger.info("解析文件: {}".format(file_name))
if not os.path.exists(file_name):
        logger.error("文件不存在")
        return False
selected_model_idx = -1

file_size = os.path.getsize(file_name)
for e_model in model_dsc_list:
    if (file_size + 1000) <= e_model[1] * 1000:
        selected_model_idx = e_model[0]
        break
if selected_model_idx == -1:
    logger.error("文本大小：{:d}B 文本超出 128K 限制，无法请求".format(file_size))
else:
    logger.info("文本大小：{:d}B 选择模型:{}".format(file_size, model_data[selected_model_idx].id))
```

2. 加载提示词和数据

将字幕数据添加到提示词中，通过该对提示词中部分内容进行替换，构成最终要输入到 kimi 的文本数据，主要同 def load_prompt(file_name)函数实现。具体的代码实现如下

```
def load_prompt(file_name):
    file_base_name = str(os.path.basename(file_name))
    course, title = file_base_name.split("_&_")
    global prompt_str
    with open(prompt_file_path, "r", encoding="utf-8") as f:
        raw_prompt_str = f.read()
    with open(file_name, "r", encoding="utf-8") as f:
```

```
        prompt_str = raw_prompt_str.replace("_COURSE_", course).replace("_TITLE_", title).replace("_SUBTITLE_",

f.read())
        return prompt_str
```

提示词文本内容如下：

```
请根据下面的要求，对视频字幕数据进行整理、处理，完成下面的2个任务。下面是课程的基本信息：
- 课程名：_COURSE_
- 当前课程标题：_SUB_TITLE_

1. 任务1：生成大纲任务
    + 任务内容：根据视频字幕数据，形成清晰、准确、规范的课程大纲。
    + 输出的大纲格式要求：
        - 一级标题至四级标题分别使用 #、##、###、#### 标记，确保标题序号连贯。
            - 一级标题需要从"一"开始，二级标题序号从"1."开始，三级标题序号从"1.1. "开始，四级标题序号从"1.1.
            - 精简一级标题数量，以覆盖更多内容。
            - 有必要，请使用三和四级标题
        - 剔除不规范或错误的字幕内容，维持学术严谨性。
        - 细致审查并剔除字幕数据中任何不符合学术规范或存在错误的内容。
        - 使用规范语言，避免非正式表达。
        - 删除与课程内容无关的推广和广告信息。
        - 结合文本主题，直接更正字幕中的错误或不当表达，无需说明。
    + 输出的大纲格式：
        _TASK_1_BEGIN_

        大纲内容

        _TASK_1_END_
2. 任务2：统计词频任务
    + 任务内容：
        - 统计前10高频词
        - 筛选与课程主题密切相关的关键词，排除通用停用词、虚词、无意义。
        - 结合文本主题，自动纠正字幕文本中的错误或不当表达，无需任何方式说明。
        - 移除与课程内容无关的词汇。
        - 仅展示关键词及其出现次数
    + 输出的词频格式：
        _TASK_2_BEGIN_

        - 词汇：出现次数

        _TASK_2_END_

视频字幕如下：
_SUBTITLE_
```

图 10 提示词

3. 请求生成内容

准备合适的模型参数和要输入的文本（提示词+字幕数据），通过 API 请求，等待生成的文本返回。具体的代码实现如下：

```
logger.info("等待生成...")
lock = [True, '生成中...']
try:
    _thread.start_new_thread(loading, (lock,))
except Exception as e:
    print(e)

completion = client.chat.completions.create(
    model=model_data[selected_model_idx].id,
    messages=[
```

```
        {"role": "system", "content": "你是一名老师，善于通过文本内容，总结其中的主要逻辑和大纲"},
        {"role": "user", "content": loaded_prompt_str}
    ],
    temperature=0.3,
)

lock = [False, '']
logger.success("-"*50)
print(completion.choices[0].message.content)
logger.success("-"*50)
```

```
PS D:\UCAS_Work\_小学期\文本分析方法与应用\pro_ws\bilibili_subtitle_download> python.exe .\aigc.py
2024-07-05 13:10:40.400 | SUCCESS  | __main__:get_ai_model:37 - 获取到可用模型列表
2024-07-05 13:10:40.400 | INFO     | __main__:get_ai_model:40 - model[0]: moonshot-v1-8k
2024-07-05 13:10:40.403 | INFO     | __main__:get_ai_model:40 - model[1]: moonshot-v1-128k
2024-07-05 13:10:40.403 | INFO     | __main__:get_ai_model:40 - model[2]: moonshot-v1-32k
2024-07-05 13:10:40.404 | INFO     | __main__:single_round_session:59 - 解析文件: ./subtitle_txt/【视频课件资料见置顶评论】深度学习入门必学丨神经网络基础丨卷积神经网络丨循环神经网络_&_第一节：多层感知机02.txt
2024-07-05 13:10:40.404 | INFO     | __main__:single_round_session:73 - 文本大小: 13868B 选择模型:moonshot-v1-32k
2024-07-05 13:10:40.405 | INFO     | __main__:single_round_session:77 - 等待生成...
⣷ 生成中...
```

```
PS D:\UCAS_Work\_小学期\文本分析方法与应用\pro_ws\bilibili_subtitle_download> python.exe .\aigc.py
2024-07-05 13:10:40.400 | SUCCESS  | __main__:get_ai_model:37 - 获取到可用模型列表
2024-07-05 13:10:40.400 | INFO     | __main__:get_ai_model:40 - model[0]: moonshot-v1-8k
2024-07-05 13:10:40.403 | INFO     | __main__:get_ai_model:40 - model[1]: moonshot-v1-128k
2024-07-05 13:10:40.403 | INFO     | __main__:get_ai_model:40 - model[2]: moonshot-v1-32k
2024-07-05 13:10:40.404 | INFO     | __main__:single_round_session:59 - 解析文件: ./subtitle_txt/【视频课件资料见置顶评论】深度学习入门必学丨神经网络基础丨卷积神经网络丨循环神经网络_&_第一节：多层感知机02.txt
2024-07-05 13:10:40.404 | INFO     | __main__:single_round_session:73 - 文本大小: 13868B 选择模型:moonshot-v1-32k
2024-07-05 13:10:40.405 | INFO     | __main__:single_round_session:77 - 等待生成...
⣷ 生成中...
```

```
2024-07-05 13:10:58.270 | SUCCESS  | __main__:single_round_session:94 - ----------------------------------------⣷ 生成中...
_TASK_1_BEGIN_

# 一、多层感知机概述
## 1.1 多层感知机定义
## 1.2 多层感知机结构

# 二、多层感知机的模型结构
## 2.1 单隐藏层多层感知机
## 2.2 多隐藏层多层感知机

# 三、权重参数与矩阵表示
## 3.1 权重矩阵定义
## 3.2 权重矩阵形状

# 四、前向传播机制
## 4.1 输入与隐藏层的计算
## 4.2 输出层的计算

# 五、激活函数的重要性
## 5.1 激活函数的必要性
## 5.2 激活函数对网络性能的影响

# 六、激活函数的性质与选择
## 6.1 激活函数的性质
## 6.2 常见激活函数介绍

# 七、常见激活函数详解
## 7.1 Sigmoid函数
### 7.1.1 函数特点与应用
### 7.1.2 导数计算与饱和区问题
## 7.2 Tanh函数
### 7.2.1 函数特点与值域
### 7.2.2 导数计算与饱和问题
## 7.3 ReLU函数
### 7.3.1 函数特点与优势
### 7.3.2 导数计算与改进

_TASK_1_END_

_TASK_2_BEGIN_

- 多层感知机：14
- 激活函数：14
- 权重矩阵：7
- 输入：6
- 输出：6
- 隐藏层：6
- ReLU：5
- Sigmoid：4
- Tanh：3
- 神经网络：3

_TASK_2_END_
2024-07-05 13:10:58.272 | SUCCESS  | __main__:single_round_session:96 - ----------------------------------------
PS D:\UCAS_Work\_小学期\文本分析方法与应用\pro_ws\bilibili_subtitle_download>
```

图 11 Kimi 生成文本

三、应用场景

在数字化教育的浪潮中，bilibili 作为一个集合了海量视频内容的平台，已经逐渐成为众多学习者的优选之地。从编程到绘画，从语言学习到科学探索，bilibili 上的视频教程几乎覆盖了所有知识领域。然而，面对如此丰富的信息资源，如何高效地吸收和整理知识，成为了学习者们面临的一大挑战。本文将探讨人工智能生成内容（AIGC）技术在 bilibili 学习过程中的应用，以及它如何帮助学习者提升学习效率和笔记整理的质量。

AIGC 技术在 bilibili 学习过程中的应用主要体现在以下几个方面：

- **视频内容的自动摘要**：AIGC 能够自动分析视频内容，提取关键信息，生成视频的大纲。这不仅帮助学习者快速把握视频的主旨和结构，而且也使得学习者能够更有针对性地进行学习。
- **关键信息的提取**：通过分析视频中的对话和文字，AIGC 能够识别并提取出高频关键词。这些关键词可以作为学习笔记的索引，帮助学习者快速定位到视频的关键部分。
- **学习效率的提升**：自动生成的大纲和关键词索引，使得学习者在编写学习笔记时能够更加高效，节省了手动整理笔记的时间，同时也提高了笔记的质量和可读性。