“星火杯”大模型创新大赛

创新应用组

队名：AIread 爱阅读队

组员：侯红日 SA23168040

目录

概要

AIread 是一款基于讯飞星火大模型 V4.0 和智文大模型的 API 接口的针对 RSS 和论文助手的创新应用。主要目标是利用大模型的总结能力来对 RSS 发送的消息进行总结从而节省用户自身的阅读成本，同时对于论文阅读场景，可以通过简单拖拽上传的方式让星火大模型进行总结和对话，与此同时利用智文大模型的 PPT 生成功能，仅需几分钟就可以为用户建立一个关于论文的汇报模板，从而大大降低了制作成本。

引入

RSS 简介

RSS，全称为 Really Simple Syndication（真正简单的整合），是一种用于网页内容更新通知的技术。它通过标准化的格式将网站内容发布出去，使用户可以通过 RSS 阅读器订阅这些内容，并在有新内容发布时接收到通知。这种技术的出现，极大地改变了人们获取信息的方式，使得信息的传递更加高效、及时。

RSS 的主要特点包括实时更新、个性化定制、跨平台支持和易于集成等。首先，RSS 能够实现网页内容的实时更新，用户可以及时获取到订阅内容的更新，无需手动刷新网页。其次，RSS 允许用户根据自己的兴趣选择订阅不同的网站或博客，实现个性化的内容推送。此外，几乎所有的主流浏览器和操作系统都支持 RSS 阅读器的安装和使用，这使得 RSS 具有广泛的适用性。最后，网站管理员可以轻松地将 RSS 功能集成到自己的网站中，提高用户体验和访问量。

随着互联网技术的不断发展和普及，RSS 作为一种高效的信息推送技术，其应用前景将越来越广阔。常见的 RSS 阅读器包括 Inoreader、Feedly、The Old Reader、NewsBlur、Netvibes 和 Flipboard 等等，它们帮助用户集中管理和阅读多个来源的 RSS 订阅内容。

星火大模型

在人工智能领域，大规模预训练语言模型已经成为了推动技术进步和应用创新的关键力量。科大讯飞作为中国领先的智能语音和人工智能公司，其推出的星火大模型自首次亮相以来，便引起了广泛关注。

2023 年 5 月 6 日，科大讯飞正式发布讯飞星火认知大模型并开始不断迭代；6 月 9 日，星火大模型 V1.5 正式发布；8 月 15 日，星火大模型 V2.0 正式发布；9 月 5 日，星火大模型正式面向全民开放；10 月 24 日，星火大模型 V3.0 正式发布；2024 年 1 月 30 日，星火大模型 V3.5 正式发布。4 月 26 日，讯飞星火大模型 V3.5 更新。5 月 22 日，讯飞星火 Lite 版永久免费。6 月 27 日，讯飞星火 V4.0 正式发布。

在关键技术突破方面，星火大模型通过优化算法和增加训练数据，显著提高了语义理解、指令跟随以及多轮对话的能力。此外，其实时互动的特性也是其一大亮点，用户可以即时获得模型的反馈，这在在线客服、智能助手等领域具有重要的应用价值。星火大模型的发展不仅体现在技术层面，还对社会产生了深远的影响。它在教育、医疗、金融等多个行业的应用，极大地提高了工作效率和服务质量。

本文介绍的应用的主要功能就建立在讯飞星火 V4.0 的 API 接口上，通过对接大模型的强大能力，获得非同寻常的用户体验。

智文大模型

智文大模型采用了先进的 Transformer 架构，这是一种基于注意力机制的深度学习模型，能够有效捕捉文本中的长距离依赖关系。这种架构的优势在于它能够并行处理数据，大大提高了训练效率和模型的性能。通过大规模的语料库进行预训练，智文大模型能够理解和生成高质量的文本内容，这在多个 NLP 任务中都得到了验证。

除了基本的文本生成和理解能力外，智文大模型还具备多模态处理能力，能够理解和生成包括图片在内的多种类型的信息。这一特性使得智文大模型在应用场景上更加广泛，能够满足不同行业和领域的需求。

智文大模型在 PPT 生成方面的作用尤为突出。传统的 PPT 制作过程通常需要大量的手动操作和时间投入，而智文大模型能够根据用户提供的主题和内容要求，自动生成结构合理、内容丰富的 PPT 演示文稿。这不仅大大节省了用户的时间，还提高了 PPT 的专业性和吸引力。通过自然语言处理技术，智文大模型能够理解用户的意图，并据此选择合适的模板、布局和设计元素，从而生成符合用户需求的高质量 PPT。

AIread 爱阅读

但在具体使用过程中可以发现 RSS 存在一些影响用户体验的问题，一方面是当一段时间没有查看 RSS 的订阅源网站后，RSS 网站会堆积大量信息（999+），而用户无法很快的浏览相关内容。另一方面是订阅单一的 RSS 订阅源后，针对论文网站情景，会有大量与用户感兴趣的领域无关的内容，从而大幅降低了用户的使用效率。

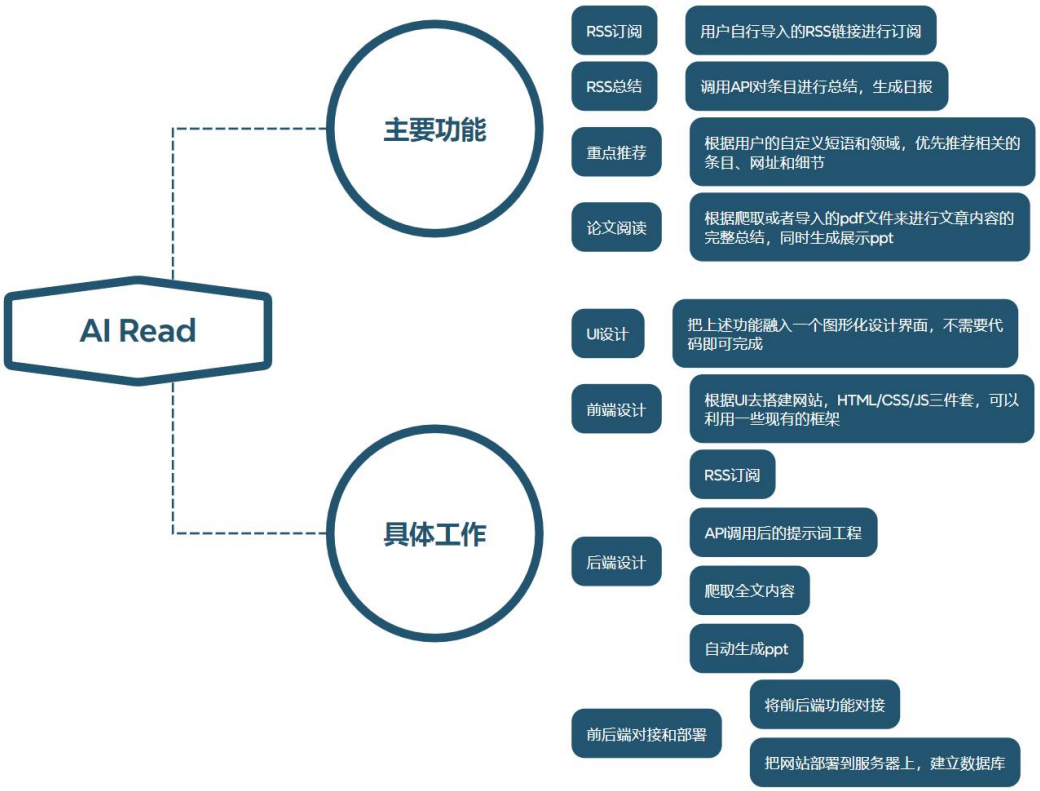
本应用的主体思想来自于作者于 23-24 年春季学期所上的罗昭锋老师的《文献管理与信息分析》的课程。我们日常获取信息的方式主要有搜索和推荐，而 RSS 则是一种较好的推荐手段，能够让我们主动了解不知道自己不知道的内容，同时可以一定程度上避免信息茧房。而如果能与大模型的总结能力进行有机融合，则可以较好的解决上面提到的两个使用目前 RSS 阅读器遇到的主要问题，从而提升用户体验并且助力科研。

在论文阅读方面的另一个常见痛点场景是对于长篇学术论文的总结和汇报，通过智文大模型的多模态能力则可以轻松解决。用户上传论文后一键生成就可以生成和论文对应的汇报 PPT，从而在该论文基础上进行进一步的修改，同时还可以询问大模型有关的细节问题。

AIread 则是一个整合了上述两个主要功能的阅读网站，旨在为用户提供更方便快捷的体验，在论文资讯获取和阅读汇报上减少不必要的成本。

具体实现

功能设计

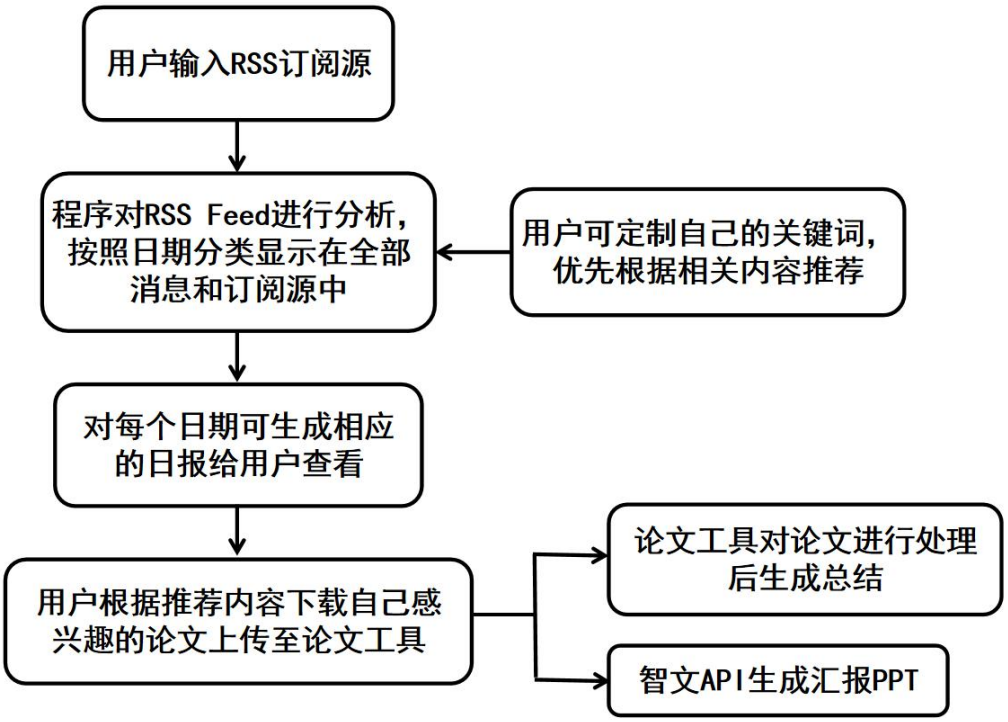


AIread 主要设计功能有 RSS 订阅、RSS 总结、重点推荐和论文阅读几部分，具体需要通过 UI 设计、前端设计、后端设计以及前后端对接几方面工作来具体实现。主要针对的就是之前提到的目前 RSS 遇到的问题。

技术路线

前端主要使用 HTML/CSS/JS 网页三件套进行设计，图标方面的素材来源于 Google Material 网站。后端通过 Flask 来搭建，使用 Python 语言进行前后端对接和 API 调用，数据存储目前是通过文本文件和 json 文件的形式进行存储和查询。

用户操作逻辑

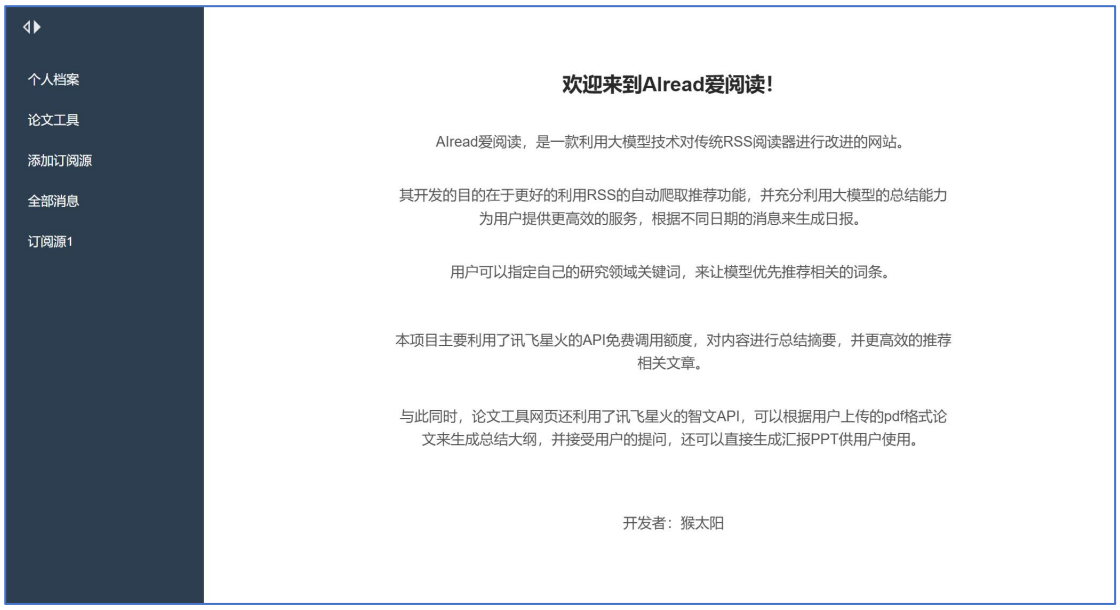


网站设计的用户操作逻辑如上面的流程图所示，用户首先输入 RSS 订阅源网址，让程序对 Feed 文档进行分析并显示消息，根据对应的日期可以生成相应的总结报告，还可以根据用户自己定制的关键词来优先推荐相关内容。

用户在阅读总结或推送消息后自行下载相关论文，上传至论文工具页面，论文工具可以对论文进行总结然后回答用户进一步的细节问题，同时还可以生成相应的 PPT 汇报文档的下载地址。

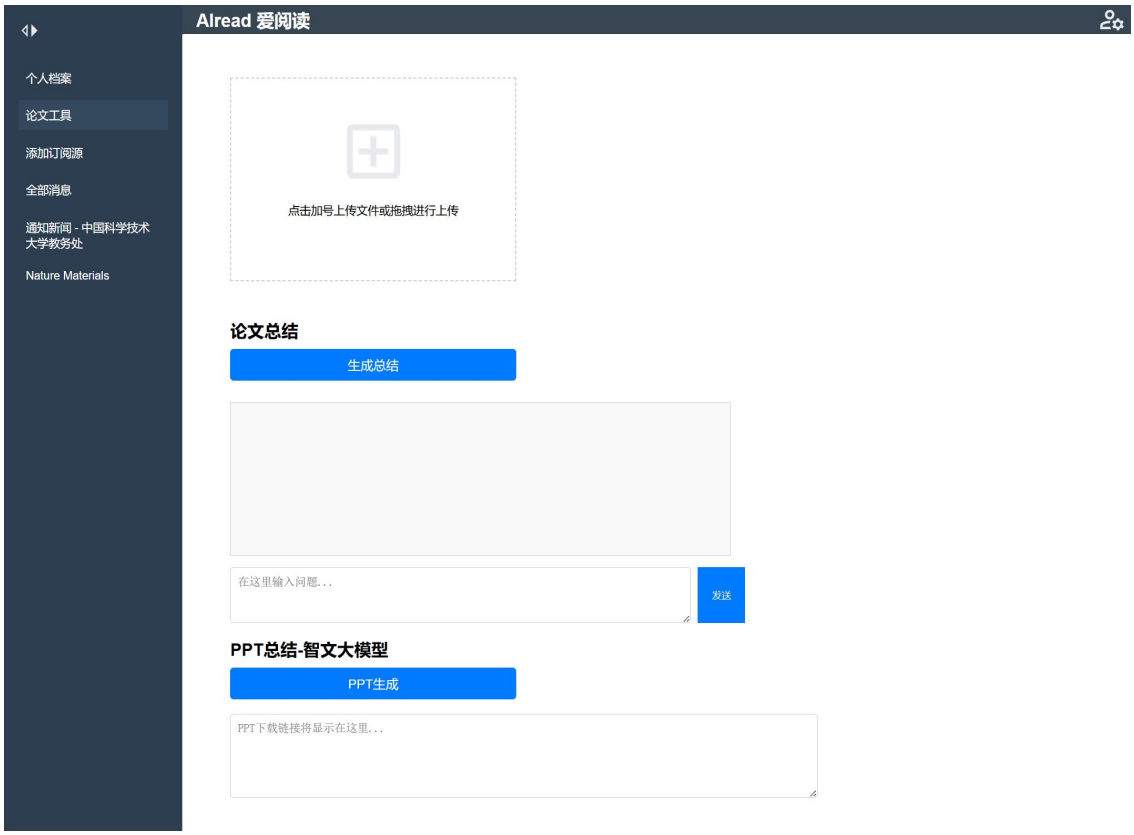
前端页面

前端主页面



前端主页面使用 HTML、CSS 和 JavaScript 进行构建，并将该模板作为母模板提供给后续的页面使用从而保证页面的统一性。

论文助手



论文助手中的拖拽上传功能使用 JavaScript 的动态监听方法来对拖拽上传操作进行识别，将论文上传到本地指定路径后调用后端代码对其进行处理。

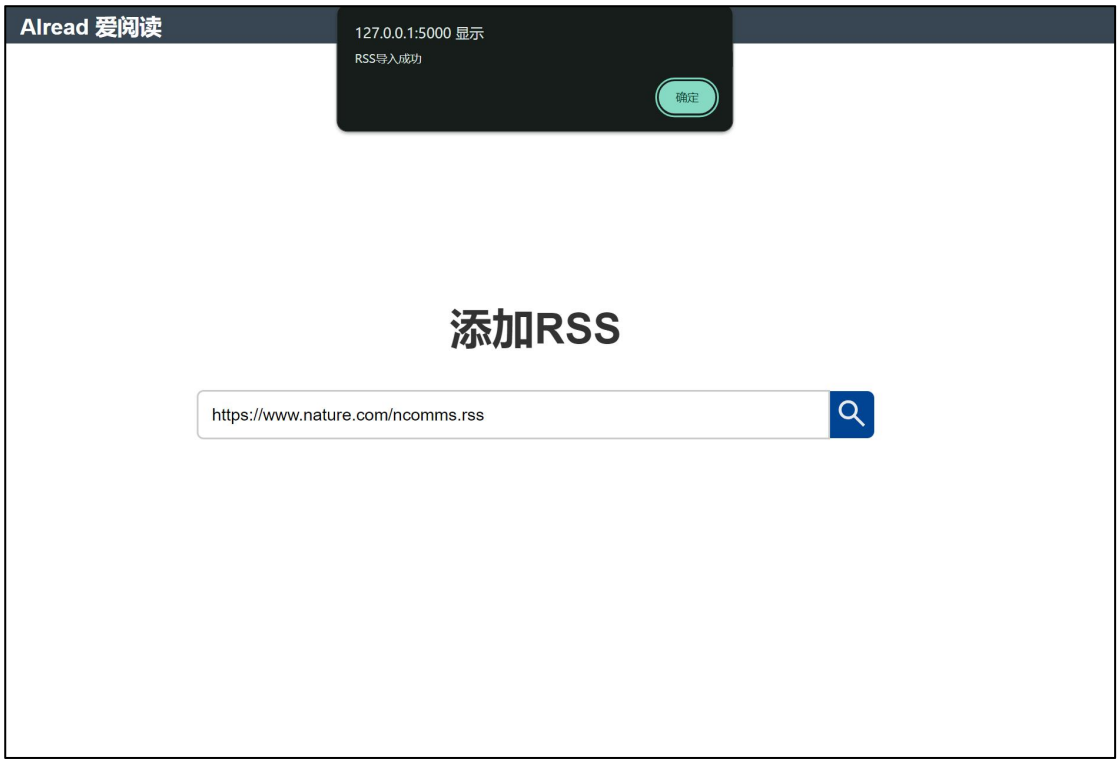
论文总结部分点击“生成总结”按钮后，星火大模型会总结上传的论文内容，然后用户可以具体进行详细的细节提问。

而 PPT 总结部分，点击“PPT 生成”按钮后，后端会调用智文大模型进行 pdf 的 PPT 生成，等待几分钟后会将下载链接上传至前端进行更新。

个人档案

个人档案目前就仅有用户名和关键词两部分内容，主要是为了让用户进行关键词的自主编辑，从而实现总结推送时达到更精准的推送用户感兴趣的领域。

添加订阅源

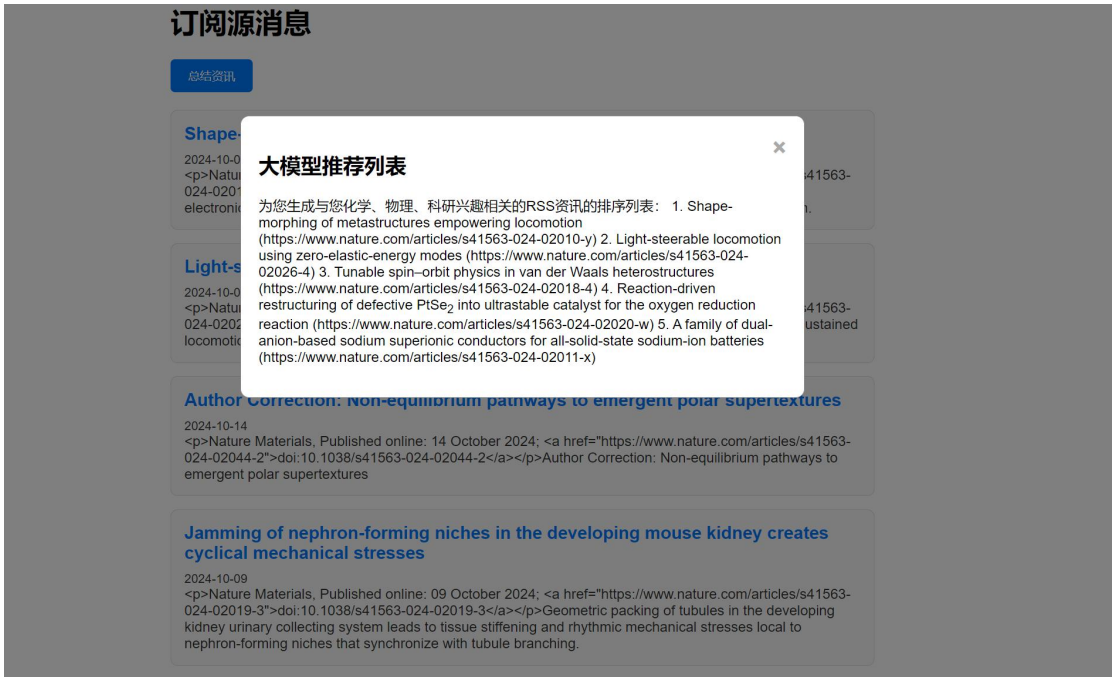


添加订阅源的页面是让用户自主输入感兴趣的 RSS 订阅网址，然后导入到后端进行分析处理，导入成功会显示弹窗消息。

订阅源界面

当订阅源被添加后，用户可以点击相应的新生成的选项卡进入相应的订阅源消息列表。

总结和询问



可以点击总结按钮对整个页面进行总结和推荐兴趣相关的。对特定的消息，若消息过长，也可以点击询问 AI 的功能来询问具体细节。

目前还没有开发对同一个订阅源的更新功能，即对已经导入的订阅源的消息进行更新，增加新增的信息内容，需要对重复的资讯进行筛选。

后端实现

后端总体是通过 Flask 进行构建的，Flask 是一个轻量级、模块化的 Python Web 框架，适合快速构建简单而功能强大的 Web 应用，并提供灵活的扩展选项。通过给每个页面建立路由页面来管理不同页面的后端操作逻辑。

论文助手的 PPT 生成

论文助手中的 PPT 生成功能是利用智文大模型的 PPT 文档生成 API，使用 demo 代码中的`createByDoc.py`文件，修改其参数列表，即可进行文档生成，生成完成后会提供下载链接供用户进行下载。

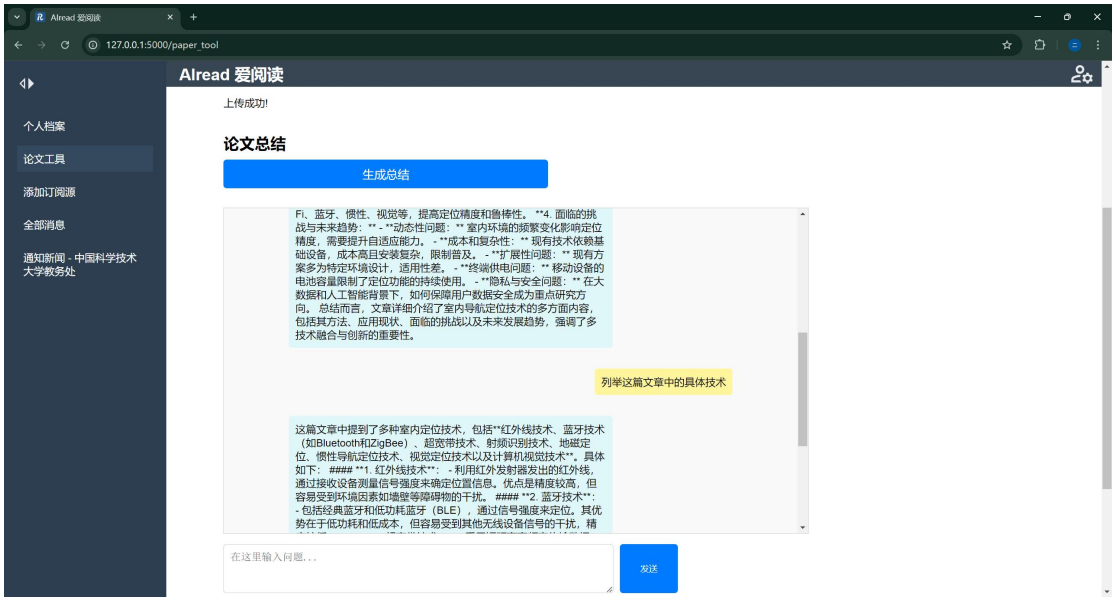
```python
params = {
    "file_name": self.filename,
    "query": "这是一篇学术文献，请用这篇文献生成一个文献汇报的PPT，要求逻辑清晰，图文并茂，重点突出。",
    "file": (
        self.output + '.pdf', open(self.filepath, 'rb'),
        'application/pdf'
    ),
    "is_figure": "true",
}
data = MultipartEncoder(fields=params)
```


```
{'appId': 'c29cd28d', 'timestamp': '1726382841', 'signature': 'JvLO391p8uS3mwNcqmfmCx7EqdY=', 'Content-Type': 'multipart
/form-data; boundary=468fcc23c04a42ae9bb2dd4c7a309769'}
{"flag":true,"code":0,"desc":"成功","count":null,"data":{"process":70,"pptUrl":null,"errMsg":null}}
sid:3109332f669f4db7ae165845055813b8
{'appId': 'c29cd28d', 'timestamp': '1726382841', 'signature': 'JvLO391p8uS3mwNcqmfmCx7EqdY=', 'Content-Type': 'multipart
/form-data; boundary=468fcc23c04a42ae9bb2dd4c7a309769'}
{"flag":true,"code":0,"desc":"成功","count":null,"data":{"process":70,"pptUrl":null,"errMsg":null}}
sid:3109332f669f4db7ae165845055813b8
{'appId': 'c29cd28d', 'timestamp': '1726382841', 'signature': 'JvLO391p8uS3mwNcqmfmCx7EqdY=', 'Content-Type': 'multipart
/form-data; boundary=468fcc23c04a42ae9bb2dd4c7a309769'}
{"flag":true,"code":0,"desc":"成功","count":null,"data":{"process":70,"pptUrl":null,"errMsg":null}}
sid:3109332f669f4db7ae165845055813b8
{'appId': 'c29cd28d', 'timestamp': '1726382841', 'signature': 'JvLO391p8uS3mwNcqmfmCx7EqdY=', 'Content-Type': 'multipart
/form-data; boundary=468fcc23c04a42ae9bb2dd4c7a309769'}
{"flag":true,"code":0,"desc":"成功","count":null,"data":{"process":70,"pptUrl":null,"errMsg":null}}
sid:3109332f669f4db7ae165845055813b8
{'appId': 'c29cd28d', 'timestamp': '1726382841', 'signature': 'JvLO391p8uS3mwNcqmfmCx7EqdY=', 'Content-Type': 'multipart
/form-data; boundary=468fcc23c04a42ae9bb2dd4c7a309769'}
{"flag":true,"code":0,"desc":"成功","count":null,"data":{"process":70,"pptUrl":null,"errMsg":null}}
sid:3109332f669f4db7ae165845055813b8
{'appId': 'c29cd28d', 'timestamp': '1726382841', 'signature': 'JvLO391p8uS3mwNcqmfmCx7EqdY=', 'Content-Type': 'multipart
/form-data; boundary=468fcc23c04a42ae9bb2dd4c7a309769'}
{"flag":true,"code":0,"desc":"成功","count":null,"data":{"process":100,"pptUrl":"https://bjcdn.openstorage.cn/xinghuo-pr
ivatedata/%2Ftmp/apiTempFile3109332f669f4db7ae165845055813b8418298708515571204/LLaMA%E6%A8%A1%E5%9E%8B%E6%80%A7%E8%83%BD
%E4%B8%8E%E5%8F%82%E6%95%B0%E9%87%8F%E7%A0%94%E7%A9%B6.pptx","errMsg":null}}
生成的PPT请从此地址获取：
https://bjcdn.openstorage.cn/xinghuo-privatedata/%2Ftmp/apiTempFile3109332f669f4db7ae165845055813b8418298708515571204/LL
aMA%E6%A8%A1%E5%9E%8B%E6%80%A7%E8%83%BD%E4%B8%8E%E5%8F%82%E6%95%B0%E9%87%8F%E7%A0%94%E7%A9%B6.pptx
```


论文助手的总结生成

总结对话页面通过 pdfplumber 库函数对上传的 PDF 文件进行信息抽取，然后整体传入到讯飞星火的 API 调用接口进行总结和对话。

为了对话过长避免传入的论文内容被溢出，每次对话并不保存上下文对话信息。这样每次都能尽可能多的保存论文的内容来完整回答问题，对于长文本（溢出上下文窗口）如何进行完整保存，还需要进一步的开发。




```python
def pdfRead(self, filepath):
    import pdfplumber
    print('Reading pdf...')
    with pdfplumber.open(filepath) as pdf:
        self.pdf_content = ''
        # 循环读取每一页的内容
        for page in pdf.pages:
            self.pdf_content += page.extract_text()
    with open("./data/process_file.txt", 'w', encoding='utf-8') as file:
        file.write(self.pdf_content)
    print('Reading Completed, processing...')

def generate_summary(self, filepath):
    self.pdfRead(filepath)
    sys_txt = self.pdf_content + "请根据上述文章，用精炼的中文对其进行总结和概括，着重介绍其工作过程和创新点。"
    self.getText("user", sys_txt)

    SparkApi.answer =""
    print("星火:",end ="")
    SparkApi.main(self.appid,self.api_key,self.api_secret,self.Spark_url,self.domain, self.text)
    return SparkApi.answer

def ask_question(self, question):
    # 受限于上下文，为了不过度累计历史对话导致超额，我们每次都清空历史记录
    # 只输入文章内容和用户的问题
    self.text = []
    with open("./data/process_file.txt", 'r', encoding='utf-8') as file:
        pdf_content = file.read()
    print(pdf_content)
    sys_txt = pdf_content + "\n请根据文章内容回答用户的问题。"
    self.getText("user", sys_txt)
    self.getText("user", question)

    SparkApi.answer =""
    print("星火:",end ="")
    SparkApi.main(self.appid,self.api_key,self.api_secret,self.Spark_url,self.domain, self.text)
    return SparkApi.answer
```

拖拽上传

而拖拽上传功能则是通过 JS 脚本和后端路由的配合实现的。当用户点击上传按钮时，会打开文件选择框；选择文件后，触发 change 事件并调用上传处理

函数；同时，监听 dragover、dragleave 和 drop 事件，以便在文件被拖入指定区域时高亮显示，最终在放置文件时调用上传函数处理文件上传。后端路由则接收上传请求，验证文件并将其保存到指定位置。

```
/**********************/
/* 论文工具中的拖拽上传 */
/**********************/
document.getElementById('uploadImage').addEventListener('click', function () {
    document.getElementById('fileInput').click(); // 打开文件选择框
});

// 监听文件选择变化
document.getElementById('fileInput').addEventListener('change', function (event) {
    uploadFiles(event.target.files);
});

// 监听拖拽
const uploadBox = document.getElementById('uploadBox');
uploadBox.addEventListener('dragover', (event) => {
    event.preventDefault();
    uploadBox.classList.add('drag-over');
});

uploadBox.addEventListener('dragleave', () => {
    uploadBox.classList.remove('drag-over');
});

uploadBox.addEventListener('drop', (event) => {
    event.preventDefault();
    uploadBox.classList.remove('drag-over');
    const files = event.dataTransfer.files;
    uploadFiles(files); // 上传拖拽的文件
});
```

添加订阅源

添加订阅源的功能主要由 feedparser 库提供，对获取的 XML 文档进行分析并进行信息存储，目前是存入本地文本文件和 json 文件中，也可以考虑使用 xml 格式或 SQL 数据库进行存储。目前遇到的一个问题是很多订阅源的存储键值不统一，同时还有的没有摘要信息，所以会导致在存储数据时可能会出现部分缺失的情况。

```python
try:
    feed = feedparser.parse(rss_url)
except Exception as error:
    print(error)
    # if feed.bozo:
    return jsonify({"status": "error", "message": "RSS解析失败"})

# 解析成功，开始存储内容
rss_title = feed.feed.title
rss_hash = hashlib.md5(rss_title.encode('utf-8')).hexdigest()

# 对每条消息进行遍历，这个键值比较麻烦
messages = []
for entry in feed.entries:
    message = {
        "title": entry.title,
        "link": entry.link,
    }
    summary = entry.get('summary') or entry.get('description')
    message['summary'] = summary if summary else "No summary available"

    details = entry.get('content') or entry.get('description')
    message['details'] = details if details else "No details available"

    date = entry.get('date') or entry.get('pubDate') or entry.get('published')
    message['date'] = date if date else "No date available"

    messages.append(message)
```

动态更新侧边栏

添加订阅源后，后端会根据存储的 RSS 订阅源名称对侧边栏进行动态更新，调用 JS 脚本对侧边栏进行动态更新，创建新的<a>标签作为新增的侧边栏标题。

选定范围进行总结

用户可以选择一个总结范围，后端对该范围内的信息进行整合并传递给星火大模型的 API，要求其根据用户的兴趣领域进行针对性的推荐，从而快速获取相关性最高的有关条目，将总结显示在前端页面上。

```javascript
//////////////////////////
// 总结推荐列表
//////////////////////////

// 获取按钮元素和弹出框元素
const recommendModal = document.getElementById('recommend-modal');
const closeModalBtn = document.getElementById('close-modal');
const generateButton = document.getElementById('generate-recommend');

// 监听点击“总结资讯”按钮，打开弹出框并调用API生成总结
generateButton.addEventListener('click', function() {
    fetch('/generate-recommend', {
        method: 'POST'
    })
    .then(response => response.json())
    .then(data => {
        // 在弹出框内显示生成的总结
        document.getElementById('recommend').innerHTML = '<p>' + data.recommend +
        '</p>';
        // 显示弹出框
        recommendModal.style.display = 'block';
    })
    .catch(error => console.error('Error:', error));
});

// 监听关闭按钮点击事件，隐藏弹出框
closeModalBtn.addEventListener('click', function() {
    recommendModal.style.display = 'none';
});
```

```python
def jsonRead(self, filepath):
    print('Reading json...')
    self.json_content = []
    # 读取对应的RSS订阅源消息
    files = glob.glob(f'{filepath}/*.json')
    for i in range(8):
        file = files[i]
        # 读取每个JSON文件
        with open(file, 'r', encoding='utf-8') as f:
            message = json.load(f)
            self.json_content.append(message)  # 将消息添加到列表中

def getKeyword(self):
    with open("./data/keyword.txt", 'r', encoding='utf-8') as file:
        self.keyword = file.read()

def generate_recommend(self, filepath):
    self.getKeyword()
    self.jsonRead(filepath)
    sys_txt = str(self.json_content) + f"请根据上面的资讯的json列表里的内容，为用户提供
    一个相关性和{self.keyword}从高到低的排序的资讯的title和link属性的排序，回答格式为'为
    您生成与您{self.keyword}兴趣相关的RSS资讯的排序列表：1.title1(link1) 2.title2
    (link2)...'"
    self.getText("user", sys_txt)

    SparkApi.answer =""
    print("星火:",end ="")
    SparkApi.main(self.appid,self.api_key,self.api_secret,self.Spark_url,self.
    domain, self.text)
    return SparkApi.answer
```

后续开发特征

之后可以发展的特征有以下几点：

1.建立用户管理系统和数据库系统；

2.部署服务器进行自动抓取和 RSS 动态订阅更新；

3.对 API 调用进行优化，文本采用流式输出减少响应时间；

4.前端和显示动画还有待改良。