基于TextRank的新闻自动摘要软件设计与实现

主讲人 陶思羽

指导老师 陈景强

时间 2025.5.29

Catalogue 目录

1. 项目简介
2. 系统整体架构
3. TextRank算法实现
4. 主要功能界面展示

01

项目简介

自动摘要技术研究

01

项目背景

由于互联网技术的发展，海量新闻信息不断产生，给用户带来了极大的阅读压力。如何从众多的信息中快速提取出关键内容，成为了一个亟待解决的问题。

02

研究意义

自动摘要技术能从冗长的文本中提取核心信息，极大地提高了用户的阅读效率。这一技术在应对信息过载现象中，具有重要的实际应用价值。

03

TextRank算法优势

TextRank 是一种无监督的抽取式摘要算法，不依赖人工标注语料，通过构建句子之间的相似图，并利用图排序算法提取出重要句子，特别适合新闻这类篇幅较长、结构松散的文本类型，能够有效提取核心信息。

04

项目目标

本项目旨在设计并实现一个支持中英文新闻文本的自动摘要系统，能够根据用户输入的文本内容，快速生成简洁、准确的摘要，帮助用户快速获取所需信息，减轻阅读负担。

02

系统整体架构

技术栈

前端技术

前端采用 Vue 3 框架构建，实现用户界面的交互与展示，通过 Axios 调用 API 与后端实时通信。

后端技术

基于 Flask 框架，负责实现 API 接口，处理业务逻辑和摘要算法，响应前端请求并返回结果。

数据库

采用 MySQL 数据库，负责持久化存储用户和摘要数据，保障数据安全和高效访问。

开发环境

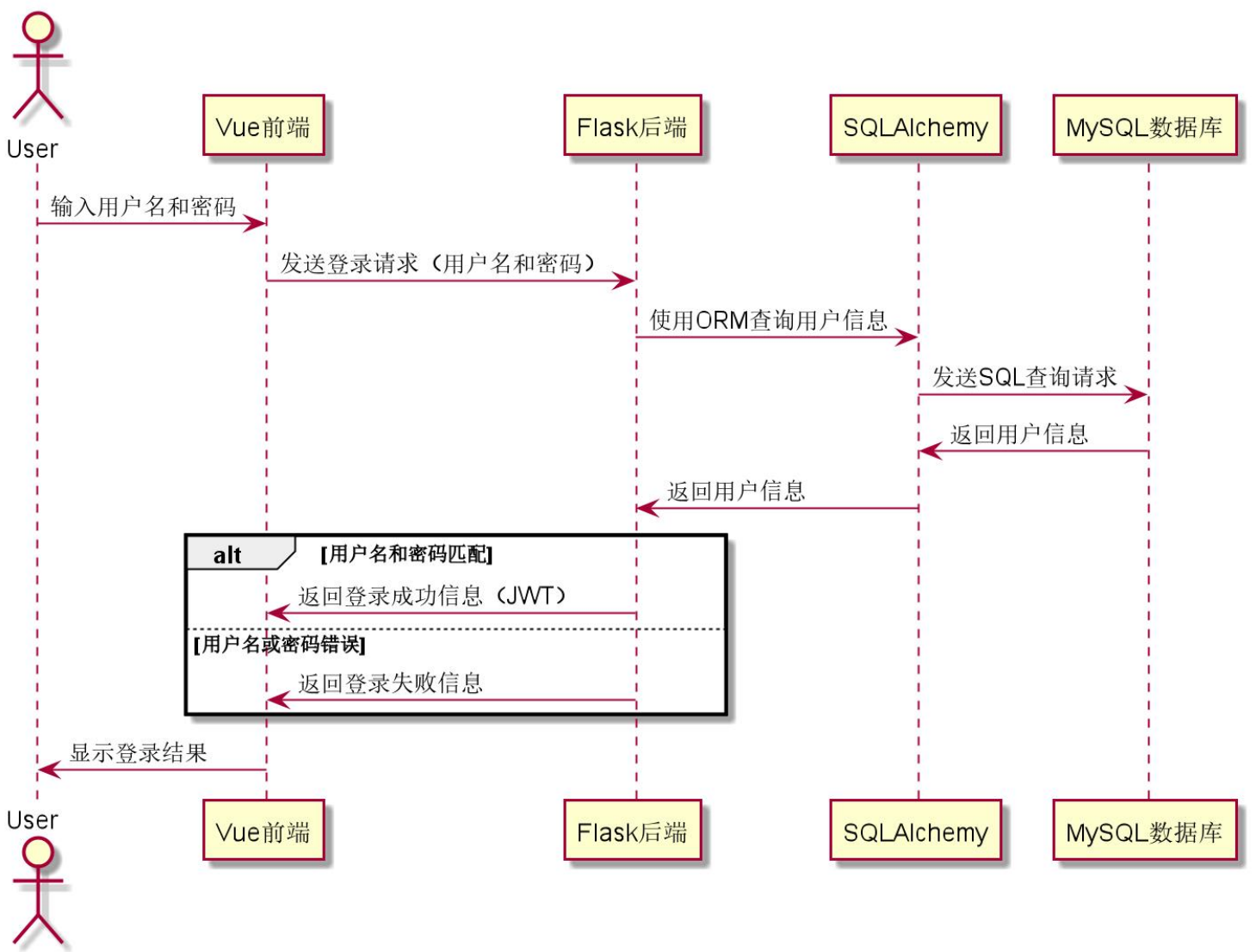
开发工具：PyCharm

语言：python3.11

操作系统：Win11

浏览器：Chrome、Firefox等

以用户登录的时序图为例展示前后端交互



1. 用户在Vue前端界面输入用户名和密码 。
2. 前端将包含用户名和密码的登录请求发送给后端 。
3. Flask后端接收到请求后，使用SQLAlchemy（一种对象关系映射工具，可将Python对象与数据库表进行映射操作 ）来准备查询用户信息。
4. SQLAlchemy向MySQL数据库发送SQL查询请求，查找对应的用户信息 。
5. MySQL数据库将查询到的用户信息返回给SQLAlchemy 。
6. SQLAlchemy把用户信息返回给Flask后端 。
7. 验证身份信息并返回对应的结果给Vue前端 。
8. Vue前端接收并显示登录结果给用户。

03

TextRank算法实现

TextRank基本流程

①文本预处理

首先将待处理文本进行分句，将每个句子视为图中的一个节点。随后对句子进行分词、去除停用词等处理，得到用于相似度计算的词集合。

②图构建

依据句子间的相似度构建图。相似度大于阈值的句子两两连接，形成无向加权图。

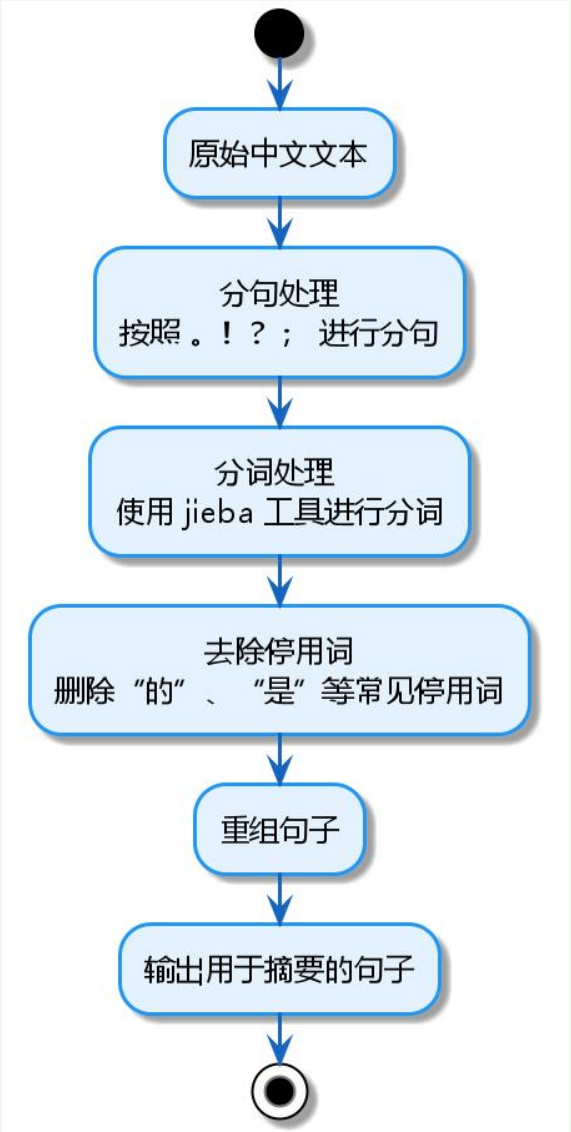
③迭代计算权重

将相似度高于一定阈值的句子之间连边，然后对每个节点（句子）初始化得分，并使用类PageRank公式进行迭代计算直至收敛。得分越高的句子被认为越具代表性。

④排序并生成摘要

根据计算出的得分对句子排序，选择得分前k 个句子作为摘要摘要内容，并保持其在原文中的顺序。

（1）文本预处理（以中文为例）



1.分句处理：
按照“。”，“！”，“？”等标点符号，将原始文本划分成一个个独立的句子，便于后续处理。

2.分词处理：
运用 jieba 分词工具，将分好的句子进一步切分成一个个词语，把连续的文本转化为词的集合，为后续的去停用词、向量计算做准备。

3.去除停用词：
从分词后的词语中，删除像“的”“是”这类在文本中频繁出现但对语义表达贡献较小的停用词，避免它们在 TF-IDF 中干扰相似度计算。

4.重组句子：
将去除停用词后的词语重新组合，构建新的句子结构，整合关键信息，用于构建相似度图。（注：这不是语法意义上的完整句子，而是一个用于向量表示的文本片段。）

（2）构建相似度图

1.使用TfidfVectorizer 对每个句子进行向量化

TF-IDF是一种常用的权重评估方法，用于衡量某个词在语料中某一文本中出现的重要程度。该方法综合考虑了词在单个文本中的频率（TF）以及其在整个语料中出现的稀有程度（IDF）。

tfidf = TfidfVectorizer().fit_transform(processed_sentences)

- 首先会根据与处理后的句子生成所有出现词语的词汇表
- 然后将每个句子向量化为一组TF-IDF值，形成一个矩阵，每一列对应一个词，每一行表示一个句子在各个词上的重要性分布。

（2）构建相似度图

2.计算句子之间的余弦相似度，并得到相似度矩阵

余弦相似度的计算公式如下：

$$CosineSim(\vec{v_i}, \vec{v_j}) = \frac{\vec{v_i} \cdot \vec{v_j}}{\|\vec{v_i}\|\|\vec{v_j}\|}$$

通过公式可以看出余弦相似度的值在0到1之间，越接近1表示越相似。

cosine_matrix = (tfidf * tfidf.T).toarray()

由于这些句子向量的维度均与词汇表的维度相同，并且TF-IDF向量已经被归一化为单位向量了，所以可以通过矩阵乘法快速计算所有句子之间的相似度。

（2）构建相似度图

3.构建句子间相似度图

将上述计算得到的相似度矩阵转化为图结构，其中每个节点代表一个句子，每对相邻节点之间的边权重即为它们的余弦相似度。

graph = nx.from_numpy_array(cosine_matrix)

使用 NetworkX 将相似度矩阵转换为无向图 graph，图结构将作为 PageRank 的输入，用于后续评分排序。

（3）迭代计算权重

PageRank原理介绍

PageRank 算法思想的核心是：一个节点（句子）的重要性由它连接的其他节点的重要性决定。PageRank 使用如下公式迭代更新每个节点的分数：

$$S(V_i) = (1-d) + d \cdot \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} S(V_j)$$

在 TextRank 算法中，这个得分由与该句子相似的其他句子的得分共同决定，一个句子越是与多个重要句子相似，它本身就越重要。在每一轮迭代中，系统会根据所有其他句子的当前得分，以及它们与目标句子的相似度，对该句子的得分进行更新。整个过程会重复多次，直到所有句子的得分趋于稳定，此时得分最高的若干句子就是最能代表全文核心内容的句子。

（4）排序并生成摘要

1.对句子按得分从高到低排序

将每个句子的 PageRank 分数与其在原文中的编号进行配对，然后根据分数进行降序排列，得分高的排在前面。

```
ranked = sorted(((score, idx) for idx, score in scores.items()), reverse=True)
```

2.获取排名靠前的前n个句子

根据用户希望生成的摘要长度选出前 n个得分最高的句子编号，并在原始句子列表中取出这些句子.

```
top_sentences = [original_sentences[idx] for _, idx in ranked[:top_n]]
```

3.生成摘要

将选出的句子合并成一个字符串。

```
return ''.join(top_sentences)
```

04

主要功能界面展示

用户生成摘要界面

- **语言选择：**支持中文和英文（下拉框选择）。
- **摘要长度选择：**用户可选择摘要句数。
- **文件上传功能：**支持txt、pdf、docx、doc格式。
- **文本输入区域：**用户直接粘贴新闻文本。
- **生成摘要按钮：**调用后端摘要服务。
- **摘要显示区：**展示生成摘要。

恳请各位老师指正

主讲人 陶思羽

指导老师 陈景强

时间 2025.5.29