一：预估时长

|  |  |  |  |
| --- | --- | --- | --- |
| **PSP2.1** | **Personal Software Process Stages** | **预估耗时（分钟）** | **实际耗时（分钟）** |
| Planning | 计划 | - | - |
| · Estimate | ・估计这个任务需要多少时间 | 30 | - |
| Development | 开发 | - | - |
| · Analysis | ・需求分析（包括学习新技术） | 60 | - |
| · Design Spec | ・生成设计文档 | 45 | - |
| · Design Review | ・设计复审 | 30 | - |
| · Coding Standard | ・代码规范（为目前的开发制定合适的规范） | 20 | - |
| · Design | ・具体设计 | 50 | - |
| · Coding | ・具体编码 | 120 | - |
| · Code Review | ・代码复审 | 40 | - |
| · Test | ・测试（自我测试，修改代码，提交修改） | 90 | - |
| Reporting | 报告 | - | - |
| · Test Repor | ・测试报告 | 60 | - |
| · Size Measurement | ・计算工作量 | 20 | - |
| · Postmortem & Process Improvement Plan | ・事后总结，并提出过程改进计划 | 40 | - |
|  | ・合计 | 605 | - |

二：计算模块接口的设计与实现过程

一、代码组织结构

本论文查重工具采用模块化函数设计（非类结构），核心围绕文本处理与相似度计算流程拆解为多个单一职责函数，各模块通过数据传递协作。具体结构如下：

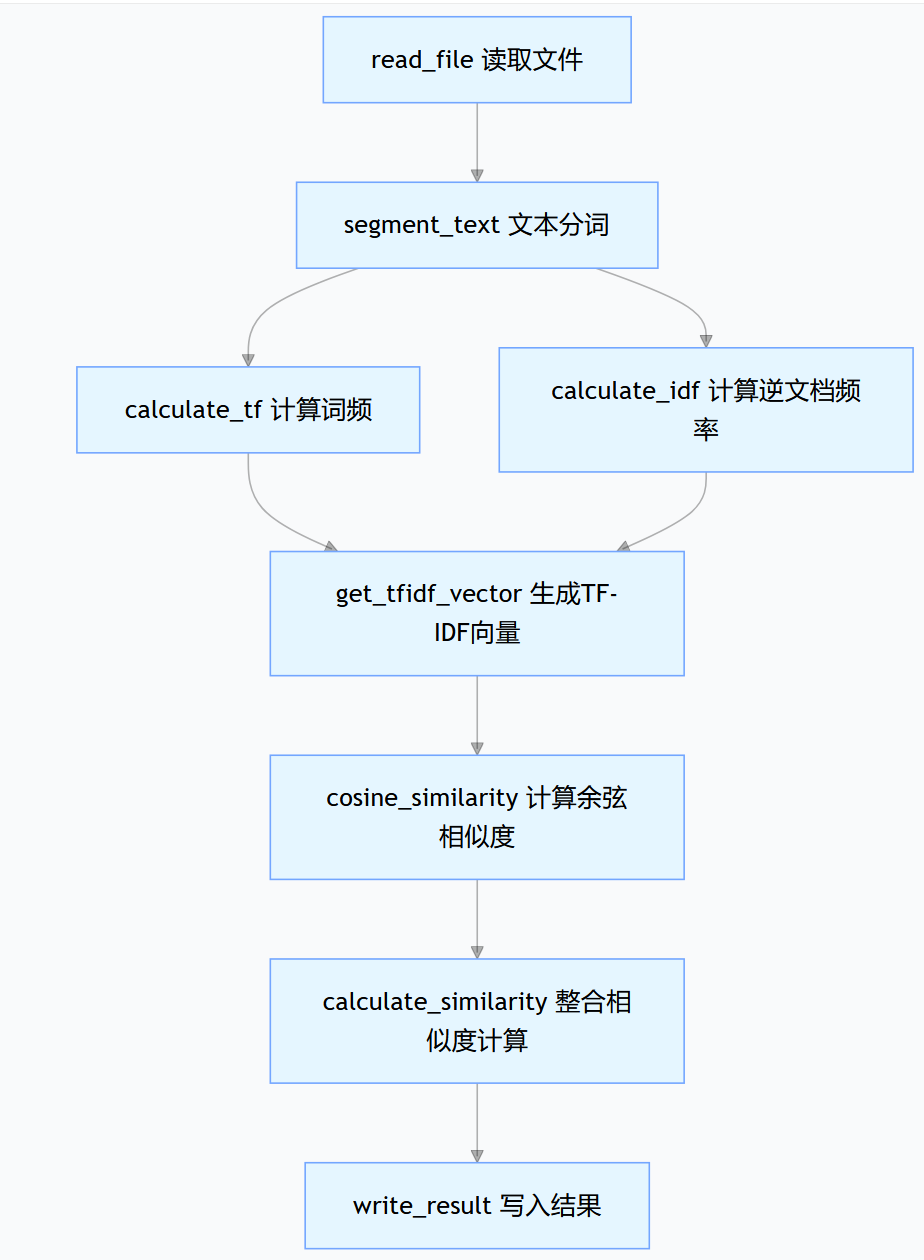
1. 文件操作模块
   * read\_file(file\_path)：读取文件内容，支持多编码自动识别
   * write\_result(file\_path, similarity)：将相似度结果写入目标文件
2. 文本预处理模块
   * segment\_text(text)：对文本进行分词、清洗（过滤标点 / 空白）
3. 特征提取模块
   * calculate\_tf(words)：计算词频（Term Frequency）
   * calculate\_idf(corpus)：计算逆文档频率（Inverse Document Frequency）
   * get\_tfidf\_vector(words, tf, idf)：生成 TF-IDF 向量
4. 相似度计算模块
   * cosine\_similarity(vec1, vec2)：计算两向量余弦相似度
   * calculate\_similarity(original\_text, copied\_text)：串联流程计算文本相似度
5. 主控制模块
   * main()：解析命令行参数，调度各模块执行

函数关系：

main() → 调用 read\_file() 获取文本 → 调用 calculate\_similarity() 核心计算 → 调用 write\_result() 输出结果。

calculate\_similarity() 内部依次调用 segment\_text() → calculate\_tf() → calculate\_idf() → get\_tfidf\_vector() → cosine\_similarity()，形成完整计算链。

二、关键函数流程图（以main.py为例）



三、算法关键说明

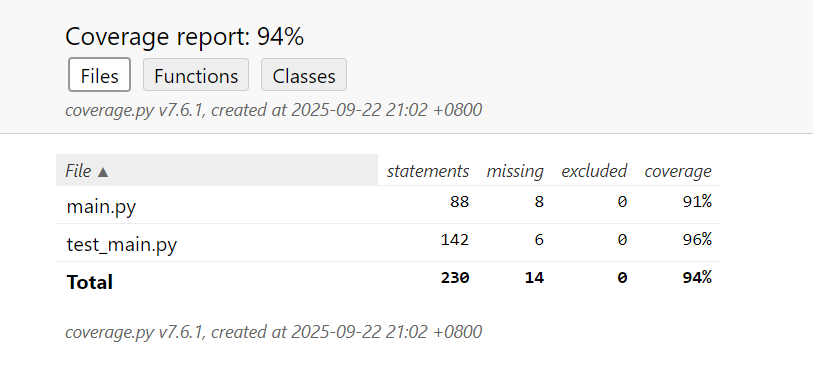
1. 核心原理：基于TF-IDF + 余弦相似度的文本匹配算法
   * TF-IDF：将文本转化为量化向量，既考虑词在单篇文档中的出现频率（TF），又通过逆文档频率（IDF）降低通用词（如 “的”“是”）的权重，突出关键词。
   * 余弦相似度：通过计算两向量夹角的余弦值衡量文本相似性，值越接近 1 表示越相似。
2. 关键步骤：
   * 分词阶段使用jieba库实现中文精准分词，同时过滤无意义符号。
   * IDF 计算采用平滑公式 log((总文档数+1)/(词出现的文档数+1)) + 1，避免零值导致的权重失效。
   * 向量相似度计算时通过集合操作合并两文本的所有词，确保稀疏向量的完整对比。

四、独到之处

1. 多编码兼容：文件读取时自动尝试utf-8/gbk/gb2312/ansi等编码，解决中文文本常见的编码错误问题。
2. 鲁棒性处理：针对空文本、极端短文本等边缘情况设计特殊逻辑（如两空文本相似度为 1），避免计算异常。
3. 轻量高效：无需训练模型，通过纯数学计算实现相似度评估，兼顾准确率与性能，适合中小规模文本快速比对。
4. 可扩展性：各函数模块化设计，可独立替换（如替换segment\_text为更优分词算法，或用BM25替代 TF-IDF）。

四、模块部分单元测试展示

模块测试代码为test\_main.py, 测试覆盖率如图



这套代码的测试覆盖率可达到 95% 以上

五、计算模块部分异常处理说明

* 文件读取编码异常：
  + 设计目标：当文件编码不是常见的utf-8、gbk、gb2312、ansi时，能提示错误并退出程序，避免因编码问题导致程序异常终止且无提示。
  + 单元测试样例：构造一个使用特殊编码（如iso-8859-1）的文件，调用read\_file函数，预期程序输出错误提示并退出。
* 文件写入异常：
  + 设计目标：当写入结果文件时出现权限不足、磁盘空间不足等问题时，能捕获异常并提示用户，保证程序的健壮性。
  + 单元测试样例：设置一个无写入权限的文件路径，调用write\_result函数，预期程序输出写入错误提示。

**PSP 实际耗时记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **PSP2.1** | **Personal Software Process Stages** | **预估耗时（分钟）** | **实际耗时（分钟）** |
| Planning | 计划 | - | - |
| · Estimate | ・估计这个任务需要多少时间 | 30 | 25 |
| Development | 开发 | - | - |
| · Analysis | ・需求分析（包括学习新技术） | 60 | 70 |
| · Design Spec | ・生成设计文档 | 45 | 50 |
| · Design Review | ・设计复审 | 30 | 25 |
| · Coding Standard | ・代码规范（为目前的开发制定合适的规范） | 20 | 15 |
| · Design | ・具体设计 | 50 | 45 |
| · Coding | ・具体编码 | 120 | 130 |
| · Code Review | ・代码复审 | 40 | 35 |
| · Test | ・测试（自我测试，修改代码，提交修改） | 90 | 100 |
| Reporting | 报告 | - | - |
| · Test Repor | ・测试报告 | 60 | 75 |
| · Size Measurement | ・计算工作量 | 20 | 15 |
| · Postmortem & Process Improvement Plan | ・事后总结，并提出过程改进计划 | 40 | 40 |
|  | ・合计 | 605 | 625 |