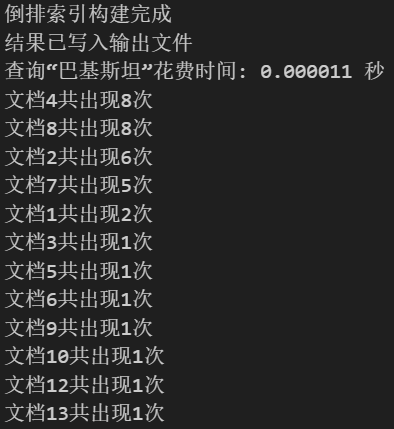
自然语言处理第三次实验报告

唐屹 10235501451

1.代码实现功能

1.1单个词语查询

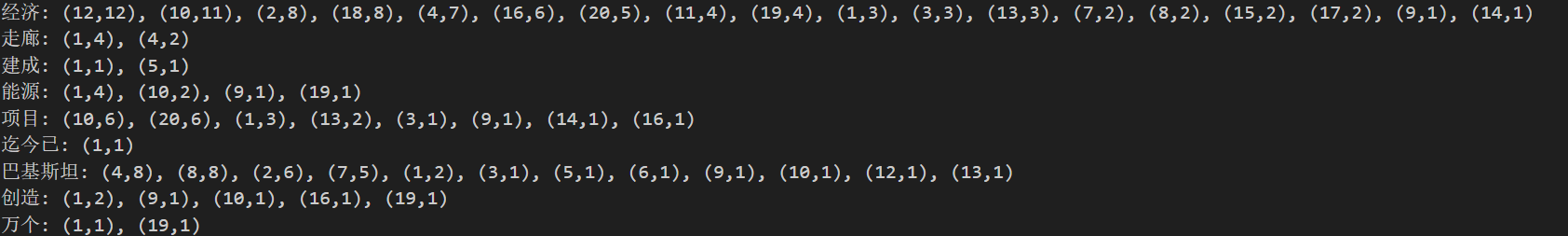


查询成功示例



查询失败案例

1.2 输出倒排索引文件（output.txt）

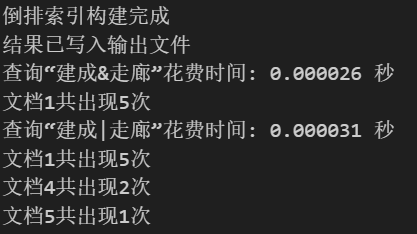


(output.txt部分截图，其中括号记录文档名和词语出现次数，如经济（10，11）代表了“经济”一词在文档10出现了11次)

1.3多个单词进行查找



以“走廊”和“建成”为例，为查询走廊和建成共同出现的文档，输入“走廊&建成”；为查询走廊和建成共同出现的文档，输入“走廊|建成”

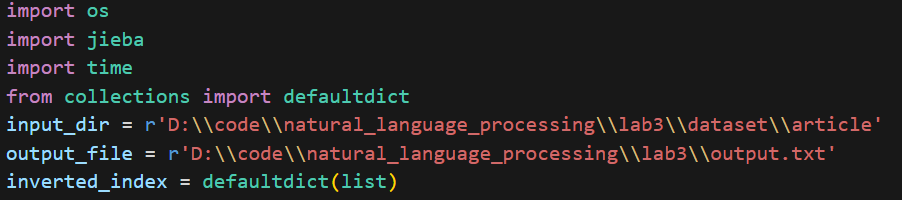


查询结果

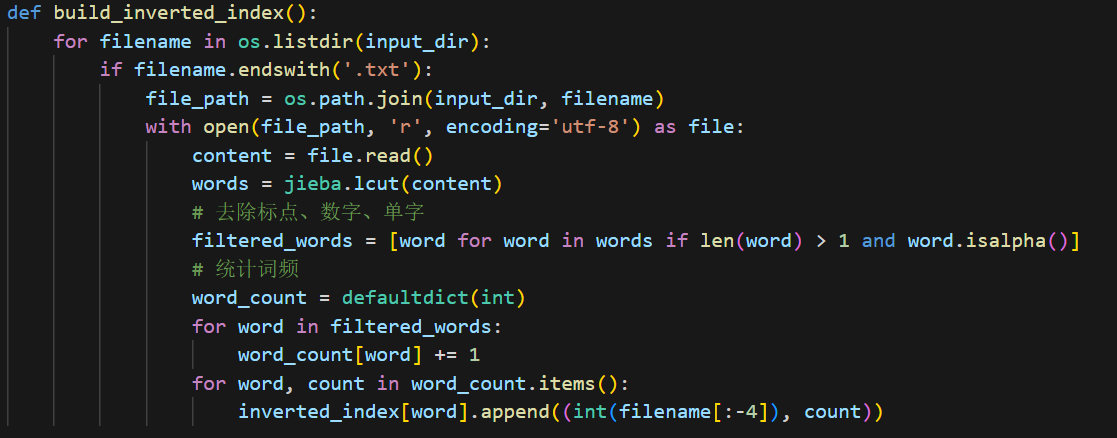
2.代码实现

完整代码可见

[natural\_language\_processing/lab3/lab3.ipynb at main · dingzhen1145141919/natural\_language\_processing](https://github.com/dingzhen1145141919/natural_language_processing/blob/main/lab3/lab3.ipynb)

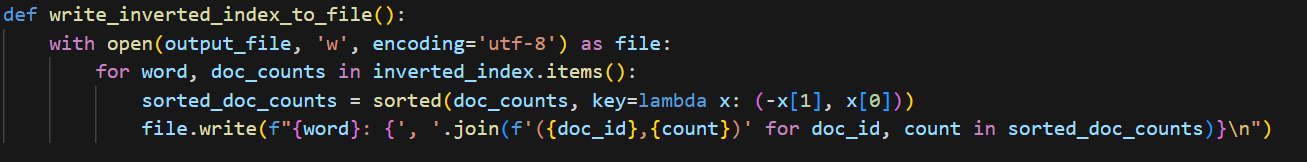


必要的库的导入和文件路径，并创建一个 defaultdict，键为词，值为包含 (doc\_id, count) 元组的列表，用于存储倒排索引。



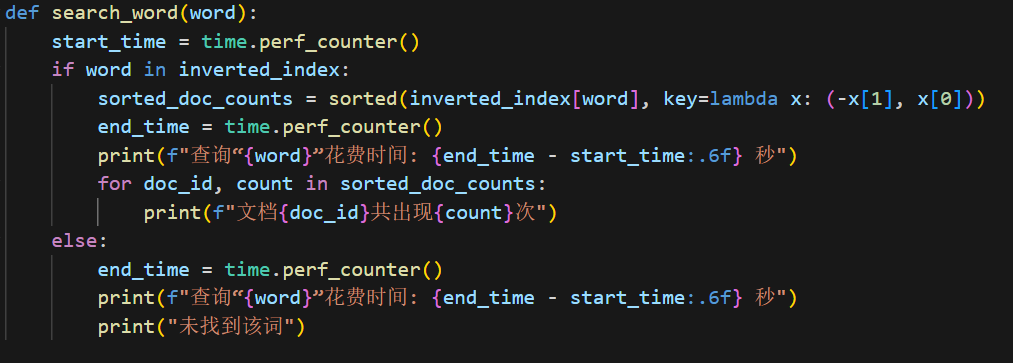
Build\_inverted\_index函数用于构建倒排索引，首先通过os库遍历目录下的所有txt文档，然后调用jieba库分词，去掉单个字（len（word）>1）和去除符号（word.isalpha）,再统计

每个单词再文档中出现的次数，然后将单词和文档id和词频存入全局变量inverted\_index



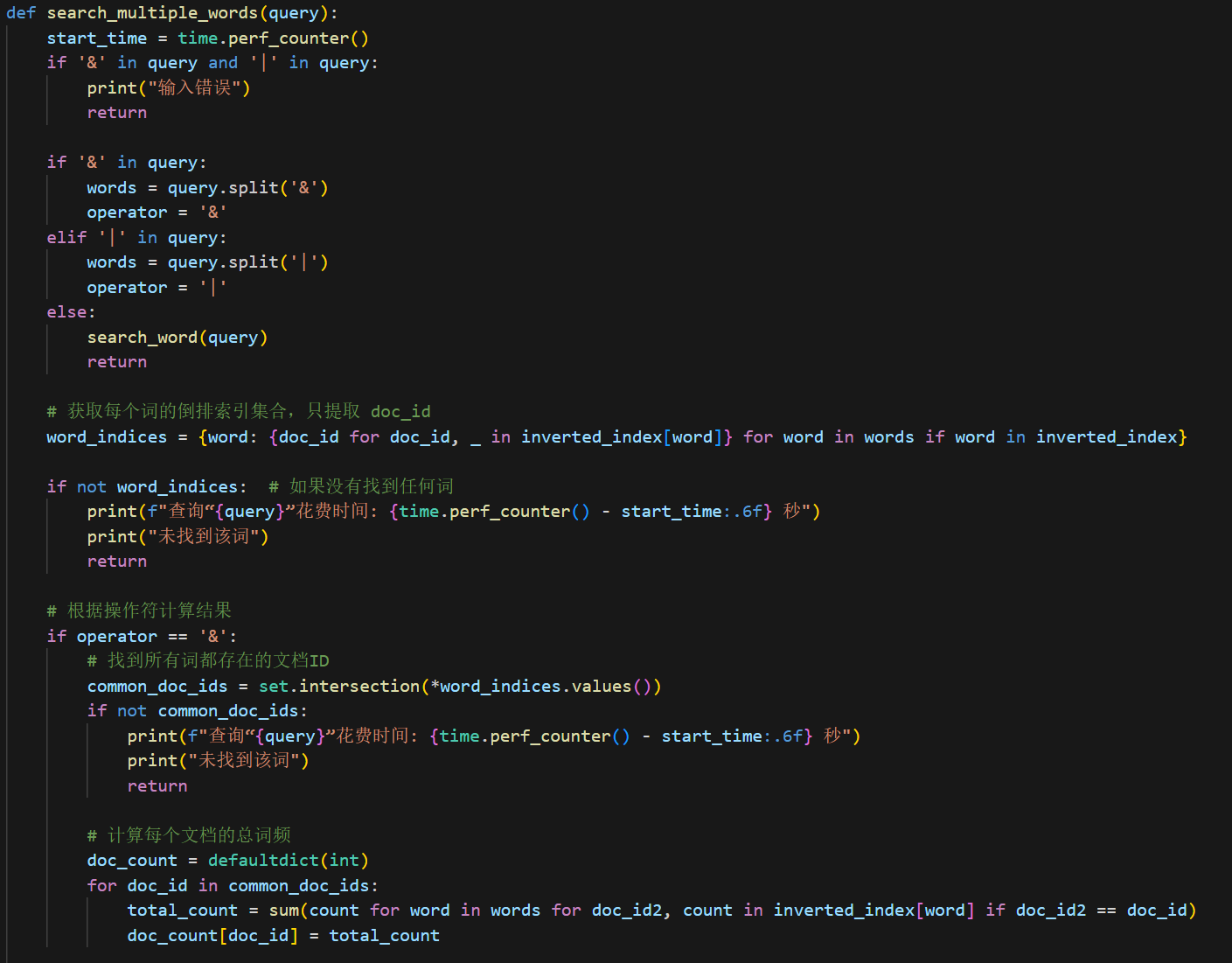
Write\_inverted\_index\_to\_file函数实现将倒排索引写入output.txt中，通过遍历inverted

\_index中的所有单词和对应文档列表，并且按照文档词频降序排序（sorted（doc\_counts）），如果词频相同，则按文档的id升序排列，最后写入output.txt文档



Search\_word函数用于实现单词的查找，如果单词存在于 inverted\_index 中，获取该单词对应的文档列表，并按词频降序排序（如果词频相同，则按文档 ID 升序排序）。

输出每个文档 ID及其对应的词频。如果单词不存在于 inverted\_index 中，输出“未找到该词”。





Search\_multiple\_words函数实现多单词的查询操作，记录查询开始时间（start\_time = time.perf\_counter()）。判断查询字符串中是否同时包含 & 和 | 操作符。如果是，则输出“输入错误”并退出。

如果查询字符串中只包含 &:

分割查询字符串为多个单词（words = query.split('&')）。获取每个单词对应的文档集合（word\_indices = {word: {doc\_id for doc\_id, \_ in inverted\_index[word]} for word in words if word in inverted\_index}）。计算这些集合的交集（即所有单词都存在的文档）：common\_doc\_ids = set.intersection(\*word\_indices.values())。

如果交集为空，则输出“未找到该词”。

否则，对交集中的每个文档，计算这些单词的总词频（total\_count = sum(count for word in words for doc\_id2, count in inverted\_index[word] if doc\_id2 == doc\_id)）。

如果查询字符串中只包含 |：

分割查询字符串为多个单词（words = query.split('|')）。

获取每个单词对应的文档集合（word\_indices = {word: {doc\_id for doc\_id, \_ in inverted\_index[word]} for word in words if word in inverted\_index}）。

计算这些集合的并集（即任意一个单词存在的文档）：all\_doc\_ids = set.union(\*word\_indices.values())。

对并集中的每个文档，计算这些单词的总词频（doc\_count[doc\_id] += count）。

按照总词频降序排序（如果总词频相同，则按文档 ID 升序排序）：sorted\_doc\_counts = sorted(doc\_count.items(), key=lambda x: (-x[1], x[0]))。

输出符合条件的文档及其总词频。

记录查询结束时间（end\_time = time.perf\_counter()），计算查询耗时并输出。



程序的主入口

3.代码的创新点和有待改进之处

代码的创新点：

1. 支持逻辑运算符的多词查询，通过逻辑运算符 & 和 | 实现多词组合查询。

& 表示交集操作，增加了对复杂查询的支持，提升了系统的灵活性和实用性。

2. 按词频排序输出结果

查询结果不仅返回文档 ID 列表，还根据每个文档中查询词的总词频进行排序。

如果多个文档的总词频相同，则按文档 ID 升序排列，提供了更丰富的查询结果展示方式，便于用户快速定位相关性较高的文档。

3. 高效的倒排索引构建与存储

在构建倒排索引时，不仅记录了每个单词出现的文档 ID，还记录了其在每个文档中的词频。

，为后续的排序和统计提供了基础数据。

优化方向：

增加权重计算，可引入 TF-ID算法，为每个文档计算相关性得分，进一步提升查询结果的质量

可使用哈希表加速查找，因为当前代码中，倒排索引存储为一个 defaultdict(list)，每次查询时需要遍历列表来获取文档 ID 和词频。可将每个单词的倒排索引从 list 改为 dict，以文档 ID 作为键，词频作为值。

引入缓存机制，对于高频查询，每次都重新计算会导致不必要的性能开销，可用引入LRU缓存机制。

优化后代码可见

[natural\_language\_processing/lab3/lab3plus.py at main · dingzhen1145141919/natural\_language\_processing](https://github.com/dingzhen1145141919/natural_language_processing/blob/main/lab3/lab3plus.py)