lab3

1. 目录结构

```
lab3/
└─src
    └─main
        ├─java
        │      task1.java                    // 执行实现倒排索引任务
        │      task2.java                    // 执行任意词检索任务
        │
        └─resources
            ├─article                        // 存放文档集
            │      1.txt
            │      2.txt
            │      3.txt
            │      4.txt
            │      5.txt
            │      6.txt
            │      7.txt
            │      8.txt
            │      9.txt
            │      10.txt
            │      11.txt
            │      12.txt
            │      13.txt
            │      14.txt
            │      15.txt
            │      16.txt
            │      17.txt
            │      18.txt
            │      19.txt
            │      20.txt
            │
            └─result                         // 存放实验结果
                    document_frequency.txt   // 文档频率表
                    inverted_index.txt       // 倒排索引表
```

2. 实现细节

2.1. task1

- `List<String> readTextFile(String filePath)`：**读取文档**

  对于文档集中的 20 个文档逐个读取，逐行读取文本，去除换行符和空行。

- `Set<String> tokenize(List<String> sentences)`：**分词**

  调用 jieba 库，对于每个文档的每个句子进行分词，对分词结果进行去除标点，数字和单字的处理。

- `Map<String, List<Integer>> buildInvertedIndex(List<Set<String>> articles)`：**建立倒排索引**

  每个词的倒排索引都由一个列表保存。在建立过程中，顺序访问每篇文章的分词结果，得到的倒排索引即已按照 docID 升序排列。

- `void saveInvertedIndex(Map<String, List<Integer>> invertedIndex, String filepath)`：**保存倒排记录表**

  保存为”词 – 倒排索引“的形式，每一行为一个词和它的倒排索引列表。

- `void saveDocumentFrequency(Map<String, List<Integer>> invertedIndex, String filepath)`：**保存文档频率表**

  保存为”词 – 文档频率“的形式，每一行为一个词和它的文档频率。此处的文档频率（DF）其实并不是”频率“，由于对于每个词而言语料库中的文档总数是相同的，所以可以省去除以文档总数的步骤，计算公式如下：

  $$DF(t, D) = \text{语料库 } D \text{ 中包含词 } t \text{ 的文档数}$$

  所以对于一个词，它的文档频率等于它的倒排索引列表的长度。

2.2. task2

- `Map<String,Integer> loadDocumentFrequency(String filepath)`：**加载文档频率表**

  加载 task1 保存的文档频率表。

- `Map<String, List<Integer>> loadInvertedIndex(String filepath)`：**加载倒排记录表**

  加载 task1 保存的倒排记录表。

- `List<Integer> combineInvertedIndex(Map<String, List<Integer>> indexes, Map<String, Integer> frequencies)`：**合并倒排记录表**

处理多个词项的倒排记录表的 and 合并。先将这些词项按照文档频率升序排列，然后每次从文档频率最小的两个词项开始合并。合并过程中，首先取出两个词项的倒排记录表，然后对每个倒排记录表分别维护一个定位指针，两个指针都是从前往后扫描，每次比较当前指针对应倒排记录，即 docID。如果两个 docID 相同，则将其加入检索结果，然后同时向后移动两个指针；如果两个 docID 不同，则向后移动指向较小 docID 的那个指针。

- `List<Integer> exactMatch(String[] keywords, Map<String, Integer> documentFrequency, Map<String, List<Integer>> invertedIndex)`：**严格检索**

  检索包含所有输入关键词的文章的 docID。具体实现为，首先在文档频率表或倒排记录表中寻找输入关键词，如果有一个关键词未找到，则可以立即返回。然后分为两种情况：如果关键词个数为 1，则直接返回其倒排记录表中的所有 docID；如果关键词个数大于等于 2，则先从文档频率表取出它们的文档频率，然后调用 `combineInvertedIndex` 函数进行处理。

- `List<Integer> moderateMatch(String[] keywords, Map<String, Integer> documentFrequency, Map<String, List<Integer>> invertedIndex)`：**适中检索**

  检索含有至少一半输入关键词的文章的 docID。具体实现为，首先在文档频率表或倒排记录表中寻找输入关键词，如果有超过一半的关键词未找到，则可以立即返回。然后统计各文章含有的关键词数，按含有的关键词数对文章进行降序排序（为了使得最终检索结果按照相关度由高到低排列），最后只保留含有至少一半关键词的文章的 docID。

- `List<Integer> looseMatch(String[] keywords, Map<String, List<Integer>> invertedIndex)`：**宽松检索**

  检索含有至少一个输入关键词的文章的 docID。具体实现为，遍历所有关键词的倒排记录表，将 docID 全部加入一个集合中。

- `void respond(List<Integer> docIDs)`：**返回检索结果**

  如果检索到的 docID 数量为 0，则打印“抱歉，没有与此相关的检索结果”；否则，先将检索到的 docID 排序，然后依次打印出来，同时打印检索到的 docID 的总数。

- `void main(String[] args)`：**main 函数**

  使用一个 while (true) 循环，首先询问用户选择检索风格（严格/适中/宽松），然后接收用户输入的关键词，如果是多个关键词需要以空格分隔，接下来记录检索开始时间，并开始进行相应风格的检索，检索完成后返回检索结果，记录检索结束时间，计算检索时间，打印本次检索用时，最后询问用户是否想要继续检索（是/否），如果用户输入“否”，则跳出循环，否则重复该流程。

3. 创新点

- 提供了三种不同的检索风格（严格/适中/宽松），可供用户自由选择。

- 进行了查询优化，大幅降低了检索时间。

```
robot：您好，请选择检索风格（严格/适中/宽松）
user：严格
robot：好的，请输入关键词
user：生物 中新社 人民日报
robot：抱歉，没有与此相关的检索结果
robot：本次检索用时 0.5627 毫秒

robot：请问您是否想要继续检索（是/否）
user：是

####################################

robot：您好，请选择检索风格（严格/适中/宽松）
user：适中
robot：好的，请输入关键词
user：生物 中新社 人民日报
robot：共 1 个检索结果： 17.txt
robot：本次检索用时 2.9679 毫秒

robot：请问您是否想要继续检索（是/否）
user：是

####################################

robot：您好，请选择检索风格（严格/适中/宽松）
user：宽松
robot：好的，请输入关键词
user：生物 中新社 人民日报
robot：共 9 个检索结果： 2.txt  5.txt  7.txt  9.txt  11.txt  12.txt  15.txt  17.txt  18.txt
robot：本次检索用时 0.3521 毫秒

robot：请问您是否想要继续检索（是/否）
user：否
```