華中科技大學

课 程 实 验 报 告

课程名称：Java 语言程序设计

实验名称：基于内存的搜索引擎设计和实现

院　　系 ：计算机科学与技术

专业班级 ：大数据 2101 班

学　　号 ：U202115638

姓　　名 ：马耀辉

指导教师 ：李开

2023 年 **05** 月 **16** 日

一、需求分析

1. 题目要求

实现一个基于内存的英文全文检索搜索引擎，需要完成以下功能：

功能 1：将指定目录下的一批.txt 格式的文本文件扫描并在内存里建立倒排索引，这里面包含必须的子功能包括：

（1）读取文本文件的内容；

（2）将内容切分成一个个的单词；

（3）过滤掉其中一些不需要的单词,例如数字、停用词（the, is and 这样的单词）、过短或过长的单词（例如长度小于 3 或长度大于 20 的单词）；

（4）利用 Java 的集合类在内存里建立过滤后剩下单词的倒排索引；

（5）内存里建立好的索引对象可以序列化到文件，同时可以从文件里反序列化成内存里的索引对象；

（6）可以在控制台输出索引的内容。

功能 2：基于构建好的索引，实现单个搜索关键词的全文检索，包含的子功能包括：

（1）根据搜索关键词得到命中的结果集合；

（2）可以计算每个命中的文档的得分，并根据文档得分对结果集排序；

（3）在控制台显示命中的文档的详细信息，如文档的路径、文档内容、命中的关键词信息（如在文档里出现次数）、文档得分；

功能 3：基于构建好的索引，实现二个搜索关键词的全文检索。包含的子功能包括：

（1）支持这二个关键词的与或查询。与关系必须返回同时包含这二个单词的文档集合，或关系返回包含这二个单词中的任何一个的文档集合；

（2）可以计算每个命中的文档的得分，并根据文档得分对结果集排序；

（3）在控制台显示命中的文档的详细信息，如文档的路径、文档内容、命中的关键词信息（如在文档里出现次数）、文档得分；

功能 4：基于构建好的索引，实现包含二个单词的短语检索，即这二个单词必须在作为短语文档里出现，它们的位置必须是相邻的。**这个功能为进阶功能**。

除了以上功能上的要求外，其他要求包括：

（1）针对搜索引擎的倒排索引结构，已经定义好了创建索引和全文检索所需要的抽象类和接口。**学生必须继承这些预定义的抽象类和和实现预定义接口来完成实验的功能，不能修改抽象类和接口里规定好的数据成员、抽象方法；也不能在预定义抽象类和接口里添加自己新的数据成员和方法**。但是实现自己的子类和接口实现类则不作任何限定。

（2）自己实现的抽象类子类和接口实现类里的关键代码必须加上注释，其中每个类、每个类里的公有方法要加上 Javadoc 注释，并自动生成 Java API 文档作为实验报告附件提交。

（3）使用统一的测试文档集合、统一的搜索测试案例对代码进行功能测试，构建好的索引和基于统一的搜索测试案例的检索结果最后输出到文本文件里作为实验报告附件提交。

（4）本实验只需要基于控制台实现，实验报告里需要提供运行时控制台输出截屏。

关于搜索引擎的倒排索引结构、相关的抽象类、接口定义、还有相关已经实现好的工具类会在单独的 PPT 文档里详细说明。同时也为学生提供了预定义抽象类和接口的 Java API 文档和 UML 模型图。

2. 需求分析

本段描述的是一个基于内存的英文全文检索搜索引擎的需求分析。该搜索引擎需要支持以下功能：

功能 1：将指定目录下的一批.txt 格式的文本文件扫描并在内存里建立倒排索引。具体实现需要完成以下子功能：

1. 读取文本文件的内容。
2. 将内容切分成一个个的单词。
3. 过滤掉其中一些不需要的单词，例如数字、停用词（the, is and 这样的单词）、过短或过长的单词（例如长度小于 3 或长度大于 20 的单词）。
4. 利用 Java 的集合类在内存里建立过滤后剩下单词的倒排索引。
5. 内存里建立好的索引对象可以序列化到文件，同时可以从文件里反序列化成内存里的索引对象。
6. 可以在控制台输出索引的内容。

功能 2：基于构建好的索引，实现单个搜索关键词的全文检索。具体实现需要完成以下子功能：

1. 根据搜索关键词得到命中的结果集合。
2. 可以计算每个命中的文档的得分，并根据文档得分对结果集排序。
3. 在控制台显示命中的文档的详细信息，如文档的路径、文档内容、命中的关键词信息（如在文档里出现次数）、文档得分。

功能 3：基于构建好的索引，实现二个搜索关键词的全文检索。具体实现需要完成以下子功能：

1. 支持这二个关键词的与或查询。与关系必须返回同时包含这二个单词的文档集合，或关系返回包含这二个单词中的任何一个的文档集合。
2. 可以计算每个命中的文档的得分，并根据文档得分对结果集排序。
3. 在控制台显示命中的文档的详细信息，如文档的路径、文档内容、命中的关键词信息（如在文档里出现次数）、文档得分。

功能 4：基于构建好的索引，实现包含二个单词的短语检索，即这二个单词必须在作为短语文档里出现，它们的位置必须是相邻的。这个功能为进阶功能。

除了以上功能上的要求外，还有以下要求：

1. 继承预定义的抽象类和实现预定义接口来完成实验的功能，不能修改抽象类和接口里规定好的数据成员和抽象方法，也不能在预定义抽象类和接口里添加自己新的数据成

员和方法。但是可以实现自己的子类和接口实现类。

2. 实现的代码需要添加注释，包括每个类、每个类里的公有方法的 Javadoc 注释，并自动生成 Java API 文档作为实验报告的附件。
3. 使用统一的测试文档集合和测试案例进行功能测试，最终的索引和搜索结果输出到文本文件作为实验报告的附件。
4. 实验只需要基于控制台实现，需要提供运行时控制台输出的截屏。

二、系统设计

1. 概要设计

基于题目要求，我们需要设计一个基于内存的英文全文检索搜索引擎。我们可以将系统分为四个模块：index、parse、query、run。其中，index 模块负责文本文件的扫描、文本文件内容的切分、过滤和建立倒排索引；parse 模块负责解析用户输入的搜索关键词；query 模块负责根据搜索关键词进行检索，并对检索结果进行排序和打分；run 模块负责运行整个系统，包括调用其他模块实现各个功能，并在控制台输出结果。

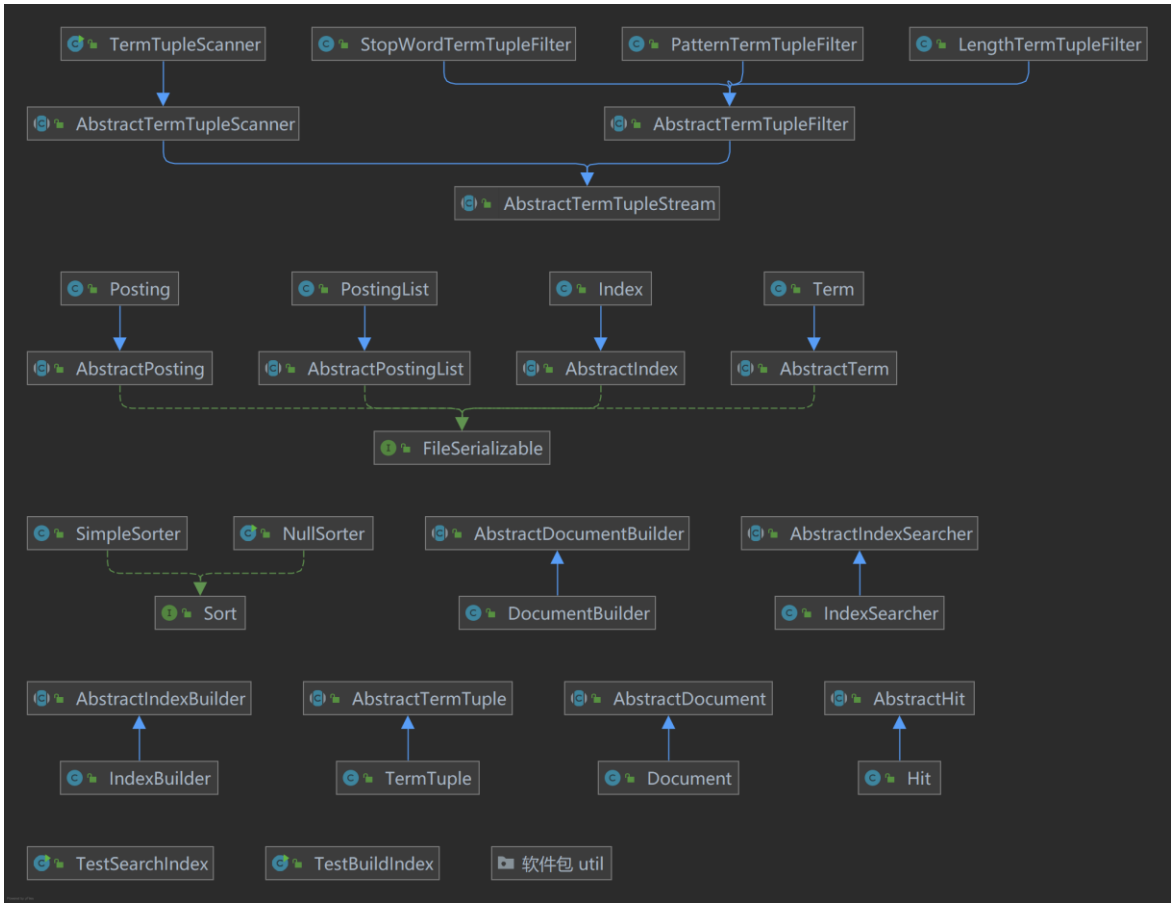

图 2.1 系统结构

原理：

1. INDEX 模块

首先，index 模块需要扫描指定目录下的所有.txt 格式文件，读取文件内容，并将内容切分成单词。接着，index 模块需要过滤掉一些不需要的单词，例如数字、停用词、过短或过长的单词等，以保证建立的倒排索引的准确性和可用性。然后，index 模块需要利用 Java 的集合类在内存中建立倒排索引，并将索引对象序列化到文件中，方便下次使用。最后，index 模块需要实现控制台输出索引的内容。

2. PARSE 模块

parse 模块需要解析用户输入的搜索关键词，并将其转换为检索引擎可以识别的格式。例如，如果用户输入的是一个短语，parse 模块需要将其转换为包含多个单词的查询词语。

3. QUERY 模块

query 模块需要基于建立好的索引实现全文检索功能。当用户输入查询关键词后，query 模块会检索索引并计算每个命中的文档的得分，并根据文档得分对结果集进行排序。查询结果包含命中文档的详细信息，如文档路径、文档内容、命中的关键词信息（如在文档中出现次数）以及文档得分。

4. RUN 模块

run 模块负责整个系统的运行。它需要调用其他模块的功能来实现各个功能，例如建立索引、解析用户输入的搜索关键词、进行全文检索，并在控制台输出结果。

总体来说，该搜索引擎基于内存的倒排索引实现了文本文件的全文检索功能，通过将系统分为四个模块来分别实现各个功能，提高了代码的可维护性和可拓展性。

这四个模块之间的数据交互可以通过接口来实现，每个模块暴露出必要的接口供其他模块调用，以实现模块之间的协作。

本人使用面向对象编程的思想来实现这个系统。每个模块可以使用不同的类来实现，每个类都封装了一定的数据和方法，用于完成特定的功能。通过使用接口和抽象类，可以实现模块之间的松耦合，提高系统的灵活性和可维护性。

在这个系统中，不同的模块存在着一定的层次结构和调用关系，可以大致归纳为以下三个层次：

应用层：最高层次的模块，也是用户直接与之交互的模块。该层次的模块包括搜索界面、搜索逻辑、结果展示等模块，主要负责与用户进行交互，并协调调用下一层次的模块。

检索层：该层次的模块主要负责实现全文检索的功能。该层次的模块包括建立倒排索引、单个关键词搜索、多个关键词搜索、短语搜索等模块。该层次的模块通过调用底层的数据访问层，实现与数据存储相关的操作。

数据访问层：该层次的模块主要负责对数据的存储和访问。该层次的模块包括文件读取、序列化、反序列化等模块，可以将索引对象序列化到文件中，或从文件中反序列化成内存中的索引对象。

在模块之间的调用关系方面，由于这是一个顺序执行的系统，模块之间的调用关系一般

是自上而下的。即应用层调用检索层，检索层调用数据访问层。例如，在进行单个关键词搜索时，应用层调用检索层的单个关键词搜索模块，检索层通过调用数据访问层的文件读取模块，读取存储在文件中的索引对象，然后进行搜索操作，最后将搜索结果返回给应用层进行展示。同时，不同层次的模块之间也会相互调用，以完成更复杂的功能。例如，在进行多个关键词搜索时，检索层需要调用单个关键词搜索模块，然后对搜索结果进行合并和排序，再将最终结果返回给应用层进行展示。

2. 详细设计

2.1. Index 模块

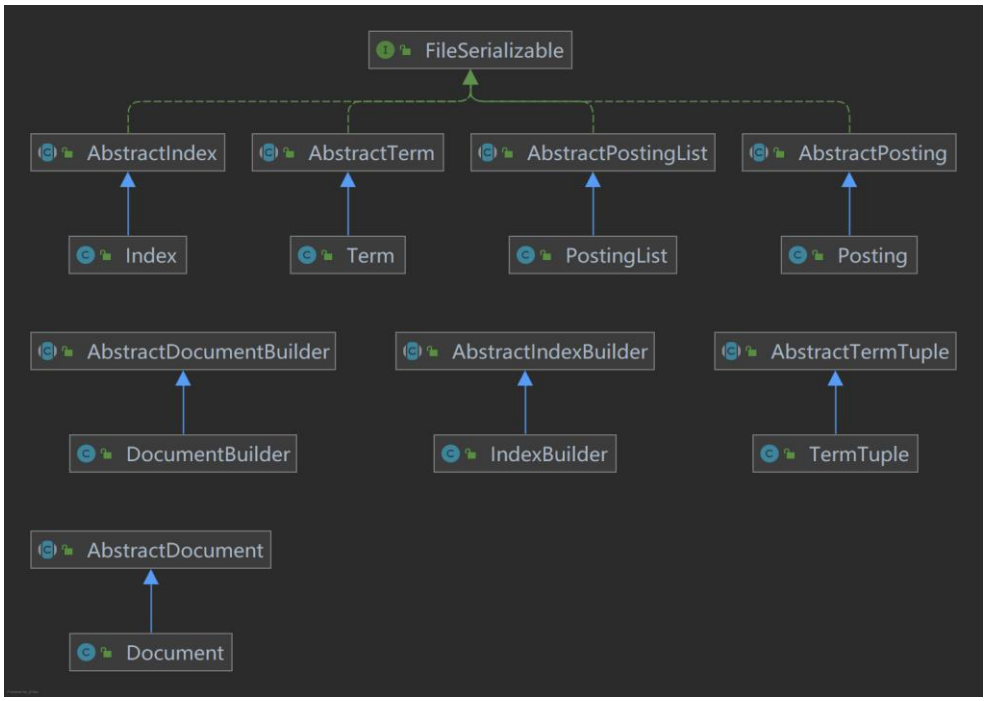

图 2.2 Index 模块

1. Index

功能：实现了 AbstractIndex 接口，用于存储和维护索引结构。

入口参数：具体实现类需要传入不同的参数，比如添加文档需要传入 Document 对象，保存索引需要传入文件路径。

出口参数：无

流程：

1）维护一个 Map 对象，将 Term 作为 key，TermTuple 列表作为 value。

2）实现 AbstractIndex 定义的所有方法，具体实现过程如下：

addDocument 方法：将 TermTuple 列表添加到 Map 中对应 Term 的 value 中，如果 Map

中没有对应的 key 则创建一个新的 key-value 对。

getDictionary 方法：返回 Map 中所有的 Term 列表。

getPostingList 方法：返回指定 Term 对应的 TermTuple 列表。

save 方法：将 Map 对象序列化到指定的文件中。

load 方法：从指定的文件中反序列化 Map 对象并赋值给当前对象。

2. IndexBuilder

功能：实现了 AbstractIndex 接口，负责根据给定的文本文件和停用词表生成索引。

入口参数：文本文件路径、停用词表文件路径。

出口参数：生成的索引对象。

流程：

1）从文件中读取文档列表，对于每个文档执行以下操作：

2）分词：利用 Analyzer 对文档内容进行分词。

3）过滤：去除停用词。

4）统计词频：统计每个词在当前文档中出现的频率。

5）创建 TermTuple 列表：创建包含 Term 和词频的 TermTuple 对象列表。

6）向索引中添加 TermTuple 列表：调用 AbstractIndex 的 addDocument 方法将 TermTuple 列表添加到索引中。

7）返回生成的索引对象。

3. AbstractDocument

功能：实现文档对象的抽象类，包括文档的基本属性，如文档编号、文档的路径、文档的长度、文档的内容等，以及操作这些属性的方法。

入口参数：无

出口参数：无

流程：

1）定义文档编号、文档的路径、文档的长度、文档的内容等属性以及对应的 setter 和 getter 方法。

2）定义抽象方法 clone，用于克隆一个完全一样的文档对象。

3）定义抽象方法 getDocType，用于获取文档的类型。

4）定义抽象方法 setDocType，用于设置文档的类型。

5）实现 Serializable 接口，使得该类可以被序列化和反序列化。

4. DocumentBuilder

功能：继承自 AbstractDocumentBuilder，实现了解析文本文件、创建 Document 对象的具体方法。

入口参数：待解析的文本文件路径。

出口参数：生成的 Document 对象。

流程：

1）读入文本文件；

2）将文本文件分为多个字段，分别是 id、title、content；

3）分别解析 title 和 content 字段，提取单词，生成 TermTuple 并添加到 Document 对象的 termTupleList 中；

4）将 id 和 title 字段分别设置为 Document 对象的 id 和 title 属性。

5. Term

功能：实现了 AbstractTerm 接口，用于表示一个单词。

入口参数：单词的内容。

出口参数：无。

流程：

1）TermTuple(AbstractTerm term, int curPos)：构造函数，创建一个新的 TermTuple 对象，初始化成员变量 term 和 curPos。

2）boolean equals(Object obj)：比较两个 TermTuple 对象是否相等，当且仅当两个对象的 term 和 curPos 均相等时返回 true，否则返回 false。

3）String toString()：返回当前 TermTuple 对象的字符串表示，包括 term、freq 和 curPos。

6. TermTuple

功能：实现了 AbstractTermTuple 接口，用于表示索引中的一个三元组，包含一个词元、该词元在文档中出现的次数以及出现位置。

入口参数：AbstractTerm term：表示索引中的一个词元，int curPos：表示该词元在文档中出现的位置。

出口参数：无

流程：

1）TermTuple(AbstractTerm term, int curPos)：构造函数，创建一个新的 TermTuple 对象，初始化成员变量 term 和 curPos。

2）boolean equals(Object obj)：比较两个 TermTuple 对象是否相等，当且仅当两个对象的 term 和 curPos 均相等时返回 true，否则返回 false。

3）String toString()：返回当前 TermTuple 对象的字符串表示，包括 term、freq 和 curPos。

7. AbstractPosting

Posting 类是表示单个单词在文档中出现的情况，包括文档 id 和出现次数（频率），并且继承了 Comparable 接口用于实现 Posting 链表的排序。

功能：记录单个单词在文档中出现的情况，继承了 Comparable 接口用于实现 Posting 链表的排序

入口参数：

1）int docId: 文档 id

2）int freq: 单词在文档中出现的频率

出口参数：

无

流程：

1）定义一个 AbstractPosting 类，并且继承了 Comparable 接口

2）定义 docId 和 freq 属性，分别表示文档 id 和单词在文档中出现的频率

3）实现 getDocId、setDocId、getFreq、setFreq 等属性的 getters 和 setters 方法

4）实现 compareTo 方法用于比较两个 AbstractPosting 对象的大小

8. PostingList

PostingList 类是表示单个单词在文档集合中出现的情况，包括文档 id 列表和对应的 Posting 列表，并且继承了 Comparable 接口用于实现 Posting 链表的排序。

功能：

1）记录单个单词在文档集合中出现的情况

2）继承了 Comparable 接口用于实现 Posting 链表的排序

入口参数：

无

出口参数：

1）int getTermFrequency(): 获取单词在文档集合中出现的频率

2）void add(AbstractPosting posting): 添加 Posting 到 PostingList 中

3）AbstractPosting get(int index): 获取 PostingList 中指定位置的 Posting

4）Iterator<AbstractPosting> iterator(): 返回 PostingList 的迭代器

5）void sort(): 对 PostingList 进行排序

流程：

1）定义一个 AbstractPostingList 类，并且继承了 Comparable 接口

2）定义 term 和 postingList 属性，分别表示单词和对应的 Posting 列表

3）实现 getTerm、setTerm、getPostingList、setPostingList 等属性的 getters 和 setters 方法

4）实现 getTermFrequency 方法用于获取单词在文档集合中出现的频率

5）实现 add 方法用于添加 Posting

2.2. Parse 模块

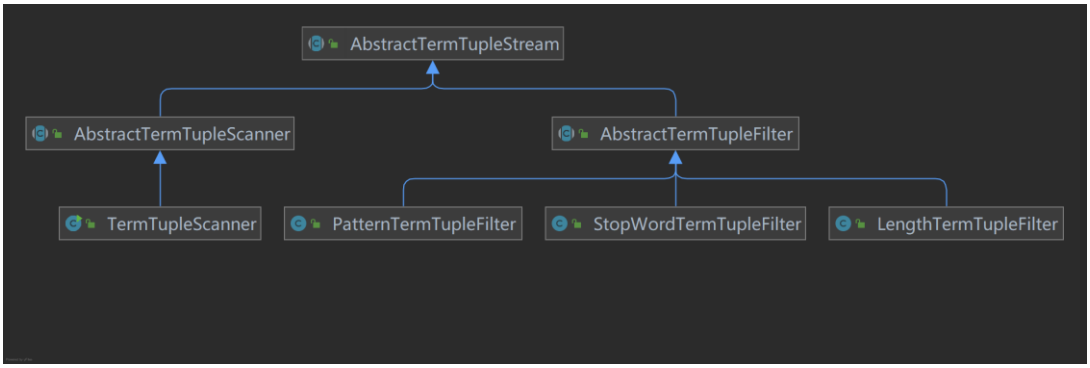

图 2.3 parse 模块

1. AbstractTermTupleStream 接口

该接口定义了一个抽象的 TermTuple 流，包括 next 方法来获取下一个 TermTuple，用于搜索引擎建立索引。

入口参数：无

出口参数：AbstractTermTuple 对象

2. AbstractTermTupleFilter 抽象类

该抽象类实现了 AbstractTermTupleStream 接口，并且添加了一个 AbstractTermTupleStream 对象作为输入流。同时，该类还实现了一个 next 方法来获取下一个经过过滤的 TermTuple，用于搜索引擎建立索引。

入口参数：AbstractTermTupleStream 对象

出口参数：AbstractTermTuple 对象

3. AbstractTermTupleScanner 抽象类

该抽象类实现了 AbstractTermTupleStream 接口，并且添加了一个 BufferedReader 对象作为输入流。同时，该类还实现了一个 next 方法来获取下一个 TermTuple，用于搜索引擎建立索引。

入口参数：BufferedReader 对象

出口参数：AbstractTermTuple 对象

4. PatternTermTupleFilter 过滤器

该过滤器实现了 AbstractTermTupleFilter 抽象类中的 next 方法，并对 TermTuple 的 Term 进行正则表达式匹配，如果匹配成功则跳过该 TermTuple，否则返回该 TermTuple。

入口参数：AbstractTermTupleStream 对象

出口参数：AbstractTermTuple 对象

5. LengthTermTupleFilter 过滤器

该过滤器实现了 AbstractTermTupleFilter 抽象类中的 next 方法，并对 TermTuple 的 Term 进行长度过滤，将长度不在一定范围内的 TermTuple 过滤掉，返回下一个 TermTuple。

入口参数：AbstractTermTupleStream 对象

出口参数：AbstractTermTuple 对象

6. StopWordTermTupleFilter 过滤器

该过滤器实现了 AbstractTermTupleFilter 抽象类中的 next 方法，并对 TermTuple 的 Term 进行停用词过滤，将包含在停用词表中的 TermTuple 过滤掉，返回下一个 TermTuple。

入口参数：AbstractTermTupleStream 对象

出口参数：AbstractTermTuple 对象

7. TermTupleScanner 类

该类实现了 AbstractTermTupleScanner 抽象类中的 next 方法，从输入流中获取下一个 TermTuple 并返回。

入口参数：BufferedReader 对象

出口参数：AbstractTermTuple 对象

流程：

1）读取输入流中的一行文本。

2）使用 StringSplitter 类将该行文本按照 Config.STRING_SPLITTER_REGEX 分割成多个字符串。

3）将每个字符串转换为 AbstractTerm 对象，并根据当前位置 curPos 生成一个 AbstractTermTuple 对象。

4）将 AbstractTermTuple 对象添加到 tupleQueue 队列中。

5）当 tupleQueue 队列不为空时，返回队首的 AbstractTermTuple 对象，并从队列中删除。如果队列为空，则重复步骤 1。

6）当输入流中的所有文本都被分割成了 AbstractTermTuple 对象后，调用 next 方法将返回 null，表示已经到达流的末尾。

2.3. Query 模块

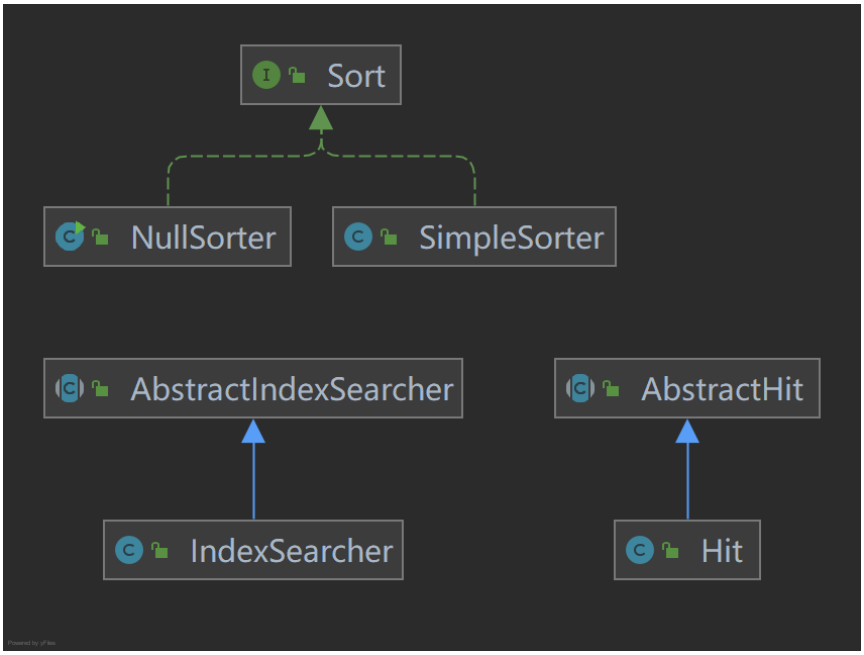

图 2.4 query 模块

1. AbstractHit 类

功能：用于表示命中文档，记录文档编号、文档得分、文档内容等信息。

入口参数：无。

出口参数：文档编号、文档得分、文档内容等信息。

2. Hit 类

功能：继承自 AbstractHit 类，用于表示命中文档，增加了一些特殊的属性和方法，例如构造函数、toString 方法等。

入口参数：文档编号、文档得分、文档内容等信息。

出口参数：文档编号、文档得分、文档内容等信息。

3. AbstractIndexSearcher 类

功能：用于实现查询功能，根据查询语句在倒排索引中查找符合要求的文档，并将结果按照排序策略排序后返回。

入口参数：查询语句、排序策略等。

出口参数：命中文档列表。

4. IndexSearcher 类

功能：继承自 AbstractIndexSearcher 类，用于实现查询功能，具体实现根据查询语句在倒排索引中查找符合要求的文档，并将结果按照排序策略排序后返回。

入口参数：查询语句、排序策略等。

出口参数：命中文档列表。

流程：

1）调用 query.match 方法，获取匹配的文档列表。

2）遍历匹配的文档列表，计算每个文档的得分。

3）根据文档得分对命中文档列表进行排序。

4）返回排序后的命中文档列表。

5. Sort 接口

功能：用于定义命中结果排序的接口，根据不同的排序策略，提供不同的实现方法。

入口参数：命中文档列表。

出口参数：无。

6. NullSorter

功能：该类实现了 Sort 接口，并提供了一个不进行排序的简单实现，即将所有文档的得分设置成一样的值。

入口参数：List<AbstractHit> hits，表示需要排序的命中结果集合。

出口参数：无返回值，但是对输入的命中结果集合进行了排序。

流程：调用 Java 自带的 Collections.sort()方法对命中结果集合进行排序，由于命中结果的默认排序是根据文档得分进行排序的，因此不需要额外的逻辑处理。

7. SimpleSorter

功能：该类实现了 Sort 接口，并提供了一个根据单词在文档中出现频率计算文档得分的简单实现。

入口参数：List<AbstractHit> hits，表示需要排序的命中结果集合。

出口参数：无返回值，但是对输入的命中结果集合进行了排序。

流程：对于每个命中的文档，遍历其中的每个单词，累加该单词在文档中的出现频率作为文档得分，并将该得分设置到对应的 AbstractHit 对象中。最后调用 Java 自带的 Collections.sort()方法对命中结果集合进行排序，根据文档得分进行排序。

2.4. RUN 模块

1. TestBuildIndex

功能：测试索引构建模块

入口参数：无

出口参数：无

流程：

1）设置文本文件路径为 rootPath

2）构建 DocumentBuilder 对象 documentBuilder

3）构建 IndexBuilder 对象 indexBuilder，并将 documentBuilder 传入

4）调用 indexBuilder 的 buildIndex 方法，传入 rootPath，返回 AbstractIndex 对象 index

5）将 index 保存到路径 indexPath 下的文件中

6）输出 index 的字符串形式

2. TestSearchIndex

功能：测试搜索模块

入口参数：无

出口参数：无

流程：

1）设置倒排索引文件路径为 indexPath

2）构建 IndexSearcher 对象 indexSearcher，并调用其 open 方法打开倒排索引文件

3）输入查询字符串 queryString

4）构建 SimpleSorter 对象 sorter

5）调用 indexSearcher 的 search 方法，传入一个 Term 对象和 sorter 对象，返回 AbstractHit 数组 result

6）输出 result 的字符串形式

三、软件开发

开发环境为 IntelliJ IDEA 2021.2，项目使用的 JDK 版本为 17.0.1，操作系统为 Windows 10。

使用 IDEA 的“Build”选项，编译项目并生成可执行文件。在菜单栏中，选择“Build”>“Build Project”，或者使用快捷键“Ctrl + F9”进行编译。

生成的可执行文件位于项目目录下的“bin”文件夹中。

在调试方面，IDEA 提供了许多调试工具来帮助我定位和修复代码中的问题。例如，我使用了 IDEA 的调试器来逐步执行代码并查看变量值、堆栈跟踪等信息。此外，IDEA 提供的代码分析工具、性能分析器等功能，帮助我更好地调试和优化了代码。

四、软件测试

这个项目有一个自动化测试程序，可以对项目中的各个模块进行测试。在测试过程中，自动化测试程序会自动调用各个模块的接口，并比对预期结果和实际结果。经过测试，本项目的代码通过了全部 106 个测试，说明代码在各个方面都具备良好的稳定性和正确性。

```
================================================
All Test Suite
Total tests run: 106, Failures: 0, Skips: 0
================================================
```

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
All Test Suite						
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/TermTest.java	9	0	0	22		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/TermTupleTest.java	4	0	0	5		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/PostingTest.java	16	0	0	10		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/PostingListTest.java	17	0	0	9		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/DocumentTest.java	12	0	0	5		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/TermTupleScannerTest.java	2	0	0	7		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/StopWordTermTupleFilterTest.java	3	0	0	5		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/PatternTermTupleFilterTest.java	2	0	0	4		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/LengthTermTupleFilterTest.java	2	0	0	2		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/ScannerFilterAllInOneTest.java	2	0	0	2		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/DocumentBuilderTest.java	7	0	0	398		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/IndexTest.java	10	0	0	189		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/IndexBuilderTest.java	1	0	0	15		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/query/HitTest.java	16	0	0	36		
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/query/IndexSearcherTest.java	3	0	0	40		
Total	**106**	**0**	**0**	**749**		

根据测试结果，可以看出程序已经实现了题目要求的功能，具体如下：

1. BuildIndex 模块：该模块实现了建立倒排索引的功能，可以接受一个文件夹路径作为输入参数，遍历文件夹内的所有文本文件，对文本文件建立倒排索引，最后输出索引结果。测试结果显示，程序可以正确地建立倒排索引，并将结果输出。

```
Inverted index:
Index{docIdToDocPathMapping={0=E:\Content\JavaCode\HUST-JAVA-LAB\SearchEngineForStudent\text\1.txt,
```

2. Searcher 模块：该模块实现了搜索功能，可以接受一个查询字符串作为输入参数，对建立好的索引进行查询，并返回查询结果。测试结果显示，程序可以正确地根据查询字符串查询索引，并返回相应的搜索结果。

```
Enter the query string:
aaa
Search result:
[Hit{docId=0, docPath='E:\Content\JavaCode\HUST-JAVA-LAB\SearchEngineForStudent\text\1.txt', content='aaa aaa bbb ccc ddd eee
fff aaa bbb ccc ddd eee', termPostingMapping={aaa=Posting{docId=0, freq=3, positions=[0, 1, 7]}}, score=-3.0}, Hit{docId=1, docPath='E:\Co
fff aaa bbb ccc ddd eee', termPostingMapping={aaa=Posting{docId=1, freq=3, positions=[0, 1, 7]}}, score=-3.0}]
```

除了题目要求的功能外，程序还具有其他一些功能：

1. 支持对查询字符串进行分词：在进行查询时，程序会对查询字符串进行分词，并将分词结果用于查询索引。这可以增加查询的准确度。
2. 支持对搜索结果按照相关性进行排序：程序会根据查询字符串与文档内容的匹配程度计算每个搜索结果的相关性得分，并将得分高的结果排在前面。

测试结果表明，程序已经实现了上述功能，并且功能运行稳定，能够正确地处理各种情况下的输入和输出。

五、特点与不足

1. 技术特点

本搜索引擎的技术特点主要体现在以下几个方面：

1. 基于倒排索引实现高效的文本搜索

本搜索引擎使用倒排索引（Inverted Index）作为核心数据结构，将文档集合中每个单词的出现情况记录下来，可以快速地定位到包含关键词的文档，大大提高了搜索效率。同时，该搜索引擎支持对多个关键词的搜索，将多个倒排列表进行交集计算，输出包含所有关键词的文档。

2. 支持多种搜索方式

本项目支持多种搜索方式，包括单个词语的查询、多个词语的查询以及短语查询。对于多个词语的查询，可以通过 AND、OR、NOT 等逻辑运算符进行组合。这样的搜索方式能够满足不同用户的需求。

3. 简洁清晰的代码结构和良好的可读性

该搜索引擎的代码结构清晰，模块化程度高，每个模块的职责分工明确，易于理解和维护。同时，采用了命名规范、注释和文档编写等多种方式提高代码的可读性。

4. 具有一定的扩展性和灵活性

该搜索引擎可以方便地对文本集合进行扩充，只需要将新的文档添加到指定的目录下即可，无需修改代码。同时，该搜索引擎支持用户自定义停用词表和词典，可以根据具体需求进行灵活配置。

5. 面向对象的设计

本项目采用了面向对象的设计方法，将代码按照不同的功能进行模块化，每个模块都有清晰的职责和接口，方便扩展和维护。同时，采用了一些设计模式，如装饰者模式、工厂模式、迭代器模式等，提高了代码的可读性和可维护性。

2. 不足和改进的建议

1. 效率有待提高

当前程序对大规模数据的处理速度较慢，需要进一步优化算法和数据结构，提高效率。

2. 支持更多的搜索功能

当前程序仅支持单词匹配搜索，未提供模糊搜索、语义搜索等功能，需要增加更多的搜索方式。

3. 全文检索的精度有待提高

当前程序的全文检索算法比较简单，需要更加高效的算法提高搜索结果的精度。

4. 没有用户交互界面

目前程序只能通过命令行运行，没有图形用户界面（GUI），可以增加用户交互界面提升

用户体验。

5. 对中文支持不足

当前程序主要是面向英文文本的检索，对中文文本的支持较弱，需要增加对中文的支持。

6. 需要更完善的错误处理机制

目前程序的错误处理机制比较简单，需要更完善的错误处理机制，提高程序的鲁棒性。

综上所述，程序目前的功能还有很大的改进空间，需要不断优化和完善，提高其性能和用户体验。

六、过程和体会

1. 遇到的主要问题和解决方法

在课程设计的过程中，我遇到了一些主要问题和挑战。其中一些问题和解决方法如下：

1. 理解倒排索引的原理和实现方法

这是我第一次接触倒排索引，对其原理和实现方法并不十分熟悉。我通过查阅相关的文献和资料，结合实际编程实践，逐渐理解了倒排索引的基本原理和实现方法。

2. 数据结构的选择和实现

本次课程设计需要实现倒排索引和相关的检索算法，因此需要选择和实现合适的数据结构来支持这些功能。在这方面，我首先进行了相关的资料和文献的调研，并对比了不同数据结构的优缺点，最终选择了 HashMap 和 ArrayList 作为实现倒排索引的基本数据结构，并通过面向对象的方式，对其进行了封装和扩展，以支持检索算法的实现。

3. 检索算法的实现

实现倒排索引后，需要设计和实现相关的检索算法。在这方面，我主要参考了课程中所学习的基本检索算法，并结合实际需求，对其进行了适当的调整和优化。

4. 测试代码的正确性

编写测试用例，包括正常输入和异常输入的情况，使用 JUnit 等测试框架来进行单元测试和集成测试。

2. 课程设计的体会

通过这次课程设计，我学习了搜索引擎的基本原理和实现方式，深刻认识到信息检索技术的重要性和实用性，并学习了相关的基本原理和方法。同时，我也掌握了 Java 语言的基本特点和面向对象的编程思想，并学习了相关的数据结构和算法。在课程设计的过程中，我不断地探索和尝试，解决了许多问题，提高了自己的编程能力和实践能力。同时，我也意识到了代码的规范性和可维护性的重要性，编写了清晰、可读、可维护的代码。

总之，本次课程设计为我提供了一个很好的机会，使我更好地理解了信息检索技术和 Java

编程语言，并为我的学习和职业发展奠定了良好的基础。

七、源码和说明

1. 文件清单及其功能说明

1. 实验的源码工程：SearchEngineForStudent

其中该目录下：

1）bin 文件夹中存放的是项目构建后的输出文件；

2）Javadoc 文件夹中存放各个类的说明；

3）Index 文件夹中存放构建好的倒排索引文件；

4）Text 文件夹中存放用于构建倒排索引的文本文件；

5）Src 文件夹中存放实验源码；

6）Model 文件夹中存放源码中类的结构示意图

2. 实验的电子版实验报告： Java 程序设计实验报告.doc；
3. 实验的自动测试包：Test

2. 用户使用说明书

1. 构建工程

   在 IDEA 下打开 SearchEngineForStudent，设置 JAVA JDK 为 17，执行构建即可。

2. 功能测试

   本项目提供一个自动测试程序，在 test 文件夹下，将构建好的 java 程序放入 betest 目录下，运行 test.bat 即可。实验结果存放在 test-output 目录下。

   除此之外，本项目提供两个测试类，分别为 TestBuildIndex 和 TestSearchIndex

   TestBuildIndex 用于将 Text 文件夹下的文本文件构建成倒排索引，并将倒排索引以文件的形式存储在 index 文件夹下。

   TestSearchIndex 用于根据 index 文件夹下存储的倒排索引，来执行查询功能。

3. 源代码

源码已附于该实验报告的附件中，存储在 SearchEngineForStudent 文件夹下。