

浙江大学计算机学院

Java 程序设计课程报告

2023—2024 学年 冬 学期

题目	Scholar Search Engine
学号	3210102037
学生姓名	徐铭
所在专业	计算机科学与技术
所在班级	计科 2101

目 录

1 引言.....	1
1.1 设计目的.....	1
1.2 设计说明.....	1
2 总体设计.....	2
2.1 功能模块设计.....	2
2.2 流程图设计.....	2
3 详细设计.....	4
3.1 网页爬虫的设计.....	4
3.2 索引建立的设计.....	5
3.3 查询的设计.....	9
4 测试与运行.....	11
4.1 程序测试.....	11
4.2 程序运行.....	11
5 总结.....	14
参考文献.....	15

1 引言

本次开发的 Scholar Search Engine 是一个能够爬取文献网站，包括其上的 PDF 文件的爬虫实践，同时实现对爬取的网页内容和 PDF 文件内容进行索引，并能够通过命令行对这些内容进行索引。

1.1 设计目的

作为计算机专业的学生，我们常常需要对计算机领域的技术前沿进行学习，这种学习的方式往往是阅读前沿论文，然而，一个热点研究方向或者重量级的期刊会议某一年的论文可能非常多，在网页上阅读往往需要开很多窗口；

有时，我们还需要将多篇论文放在一起横向比较，此时将论文下载到本地并对这些论文进行检索就十分必要了。

因此，我们具体的设计目的是：

- (1) 写一个 Web 爬虫，爬取文献网站的网页（及 PDF 文件）；
- (2) 解析网页内容，对内容进行结构化，并存储到文件中；
- (3) 解析 PDF 论文内容；
- (4) 为 2 和 3 得到的内容建立索引；
- (5) 通过命令行进行内容检索，并展示内容列表

- ① 可通过作者、标题、摘要、会议来检索论文
- ② 可检索论文图表*

1.2 设计说明

本程序采用 Java 程序设计语言，在 IntelliJ_IDEA 平台下编辑、编译与调试。具体程序由单人开发而成。

开发环境为 Oracle Open JDK 20.0.1，使用的 maven 依赖版本请查看项目文件中的 pom.xml 文件中的<dependencies>

2 总体设计

2.1 功能模块设计

本程序需实现的主要功能有：

- (1) 实现一个能够爬取指定文献网站及相应 PDF 文件的爬虫；
- (2) 能够将爬取到的内容结构化后存储到本地文件中；
- (3) 能够为本地爬取的这些文件建立索引；
- (4) 能够通过建立的索引对本地文件进行检索；

程序的总体功能如图 1 所示：

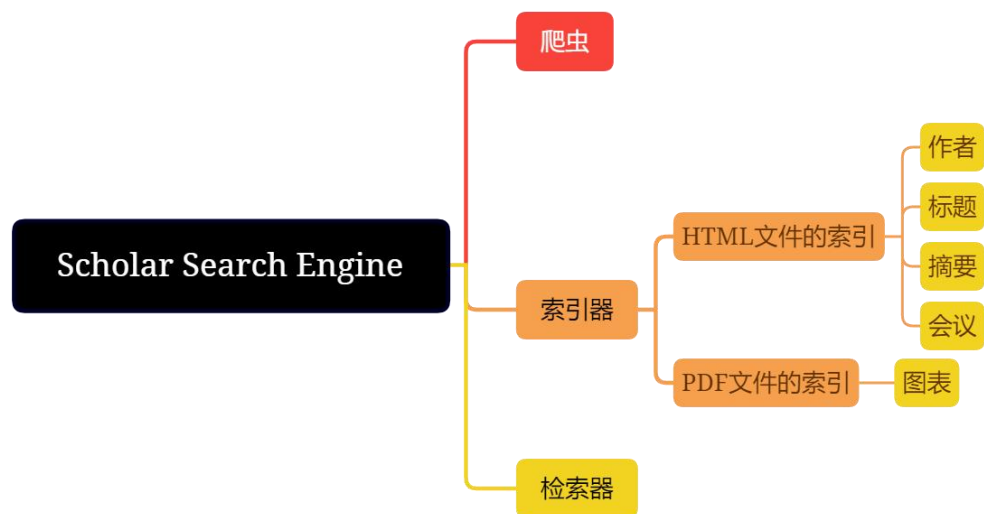


图 1 总体功能图

2.2 流程图设计

程序总体流程如图 2 所示：

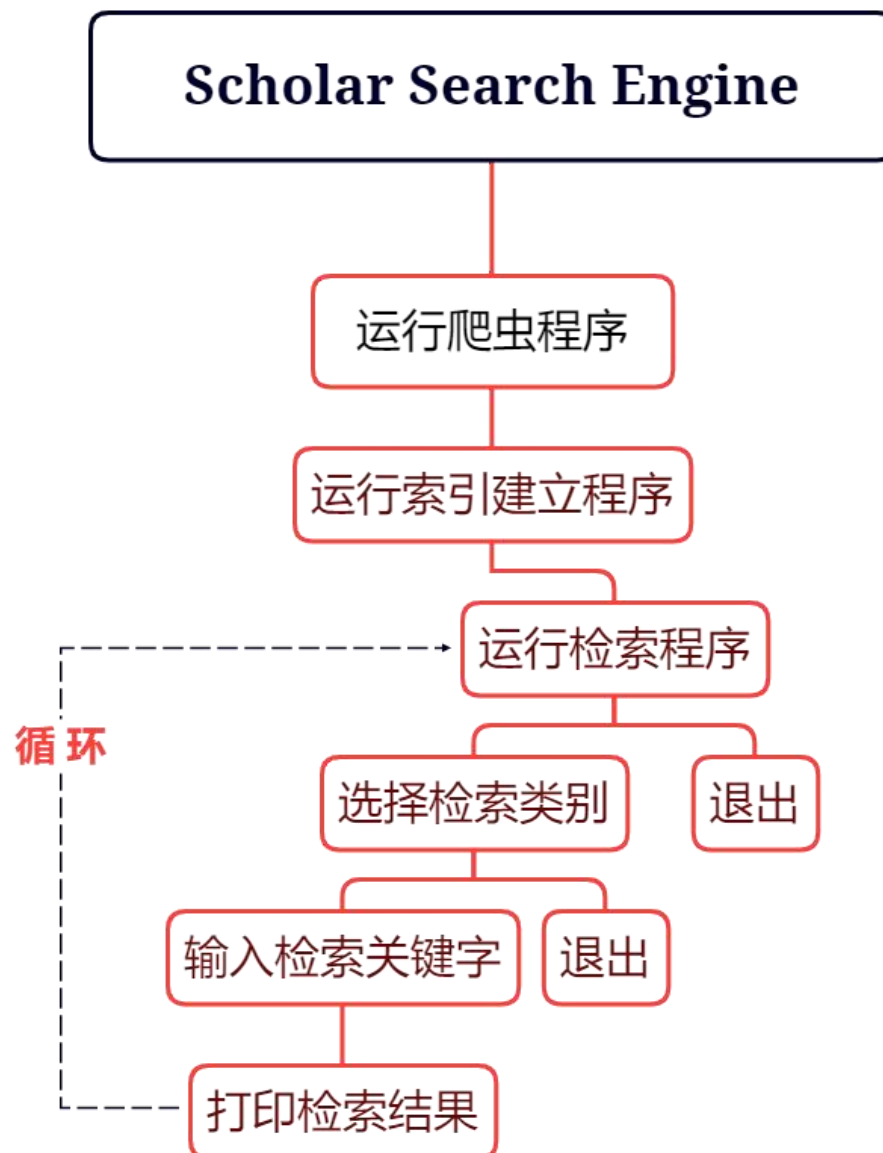


图 2 总体流程图

3 详细设计

3.1 网页爬虫的设计

在这次项目中，我们的网页爬虫使用了 JSOUP 来处理 HTML 内容。

JSOUP 是一款 Java 的 HTML 解析器，可直接解析某个 URL 地址、HTML 文本内容。它提供了一套非常省力的 API，可通过 DOM，CSS 以及类似于 jQuery 的操作方法来取出和操作数据，可以看作是 java 版的 jQuery。

我们将爬取指定网页内容，并获取相应论文 PDF 并将其格式化存储在本地的要求集成在 WebCrawler 类中，主要实现如下 UML 图所示：



图 3 WebCrawler 类的 UML 图

以下是 UML 图中有关数据和方法的详细说明：

① `List<String> parseAndStoreWebContent(String url, String path)`

首先，我们需要确定要爬取的网页是什么，这里我们选择的是“<https://aclanthology.org/>”，由于我们不止要爬取网页内容，还需要下载相应的 PDF 论文文件，所以我考虑通过循环，每次爬取论文的网页内容时，同时下载 PDF 文件。

观察打开论文的详情介绍页面，可以发现规律每篇论文的详情页的 URL 都由“<https://aclanthology.org/> + year.venue-short|long.number/”组成，因此我们想要爬取论文，可以根据这里的“number”来循环；调用 `Jsoup.connect(url).get()` 方法获取网页的 Document 对象，并使用 `BufferedWrite.write(text)` 方法将 html 内容写入本地文件中。

同时，我们需要在这里获取 PDF 文件的下载链接，F12 检查或直接右键查看网页源代码，经过检查可以发现，PDF 论文的链接均以如下形式嵌入：

```
<a href=https://aclanthology.org/2023.acl-long.report.pdf>
```

图 4 网页上的 PDF 链接

因此我们使用 Jsoup 提供的类似 CSS 语法的选择器 select 方法选择属性名为“href”，对应值后缀为“.pdf”的字符串，将该链接 append 到 PDF 下载链接集合中。

② `void downloadAndSavePdfFiles(List<String> pdfLinks, String path)`

此函数功能简单，就是通过遍历 pdfLinks 中提供的下载链接，将对应的 PDF 文件下载到根据 path 构建的相应子目录下。

3.2 索引建立的设计

建立索引首先需要使用 Lucene 对本地的待检索文件添加索引信息，其中包括 HTML 文件和 PDF 文件。

Lucene 是一套用于全文检索和搜寻的开源程式库，由 Apache 软件基金会支持和提供。Lucene 提供了一个简单却强大的应用程序接口，能够做全文索引和搜寻。在 Java 开发环境里 Lucene 是一个成熟的免费开源工具。

何为全文检索：

举个例子，比如现在要在一个文件中查找某个字符串，最直接的想法就是从头开始检索，查到了就 OK，这种对于小数据量的文件来说，很实用，但是对于大数据量的文件来说，就难以为继了。或者说找包含某个字符串的文件，也是这样，如果在一个拥有几十个 G 的硬盘中找那效率可想而知，是很低的。

文件中的数据是属于非结构化数据，也就是说它是没有什么结构可言的，要解决上面提到的效率问题，首先我们得即将非结构化数据中的一部分信息提取出来，重新组织，使其变得有一定结构，然后对此有一定结构的数据进行搜索，从而达到搜索相对较快的目的。这就叫全文搜索。即先建立索引，再对索引进行搜索的过程。

lucene 首先将文档传给分词组件（Tokenizer），分词组件会将文档分成一个个单词，并去除标点符号和停词。然后将词元传给语言处理组件（Linguistic Processor），对于英语，语言处理组件一般会将字母变为小写，将单词缩减为词根形式，如“lives”到“live”等，将单词转变为词根形式，如“drove”到“drive”等，后得到词（Term）。

它的关键字是按字符顺序排列的，因此 lucene 可以用二元搜索算法快速定位关键词。实现时 lucene 将上面三列分别作为词典文件（Term Dictionary）、频率文件（frequencies）和位置文件（positions）保存。其中词典文件不仅保存有每个关键词，还保留了指向频率文件和位置文件的指针，通过指针可以找到该关键字的频率信息和位置信息。

下图是建立索引程序的整体 UML 类图：

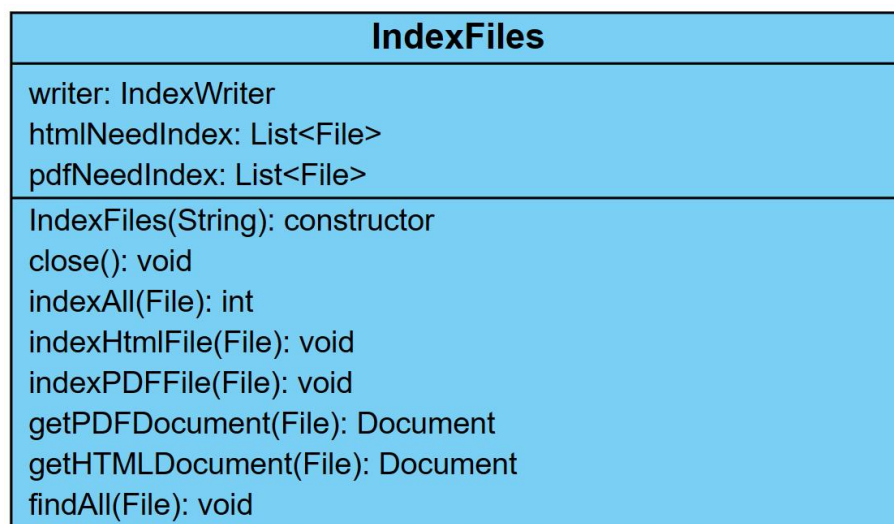


图 5 IndexFiles 类的 UML 图

以下是对各个方法的逻辑的详细说明：

① void findAll(File directory)

负责遍历指定目录下的所有文件和子目录，并将符合条件的 HTML 和 PDF 文件分别加入 htmlNeedIndex 和 pdfNeedIndex 集合中。

② Document getHTMLDocument(File file)

用于处理 HTML 文件，使用 Jsoup 解析 HTML 文件内容，并提取相关信息（文件名、作者、标题、摘要、会议等）。这些信息被保存到 Lucene 的 Document 对象中，然后添加到索引中；

这其中的关键点在于，使用 doc.add() 方法时，如何将需要索引的内容获取到并转化为 String 类型传入参数中。观察各种论文的 HTML 格式，可以总结出标题、作者、会议和摘要信息的特定格式。例如，摘要信息总是放在名为 card-body 的<div>块中名为 acl-abstract 的块中，同时还需要注意去除<h5>块中的标题。如下图所示：


```

<div class="card-body acl-abstract">
  <h5 class="card-title">Abstract</h5>
  <span> == $0
  "We present a summary of the efforts to improve conference peer review that were implemented at ACL'23. This
  includes work with the goal of improving review quality, clearer workflow and decision support for the area
  chairs, as well as our efforts to improve paper-reviewer matching for various kinds of non- mainstream NLP work,
  and improve the overall incentives for all participants of the peer review process. We present analysis of the
  factors affecting peer review, identify the most problematic issues that the authors complained about, and
  provide suggestions for the future chairs. We hope that publishing such reports would (a) improve transparency in
  decision-making, (b) help the people new to the field to understand how the *ACL conferences work, (c) provide
  useful data for the future chairs and workshop organizers, and also academic work on peer review, and (d) provide
  useful context for the final program, as a source of information for meta-research on the structure and
  trajectory of the field of NLP."
  </span>
</div>

```

图 6 Abstract 信息的格式

对于这些已经进行格式化的 HTML 本地文件，使用爬虫时一样的类 CSS 选择器，通过正则化的匹配方式可以获得我们想要的信息。

③ Document getPDFDocument(File file)

用于处理 PDF 文件，使用 Apache PDFBox 库解析 PDF 文件，并提取文本信息中的图表（Figure 和 Table）。同样，将相关信息保存到 Lucene 的 Document 对象中，并添加到索引中。

这里的关键点同样在于如何获取 Figure 和 Table 的信息。询问老师后我了解到，我们实际上是按照图表下方的注释来检索图表，查询结果可以显示图表所在论文的信息。

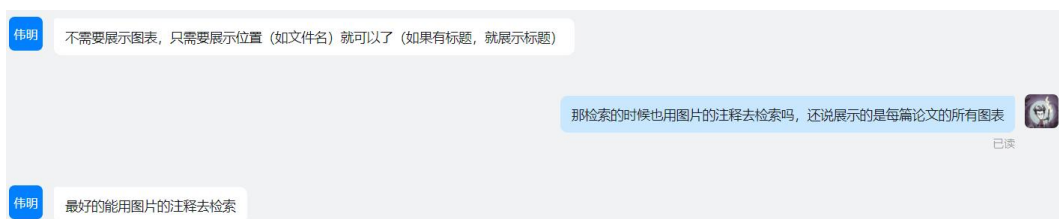


图 7 与老师的交流信息

因此，这里的重点转变为如何获取 PDF 文件中图片下面的注释信息，在本次项目中我使用的用来解析 PDF 文件的工具是 Apache PDFBox。

Apache PDFBox 库是一个开源的 Java 工具，用于使用 PDF 文档。该项目允许创建新的 PDF 文档，操作现有的文档以及从文档中提取内容的能力。

首先，我尝试了最直观的想法，也是老师提及的方法——使用 caption——PDFBox 中的 PDAnnotation 类检索，直观理解为 PDF 文件中的所有注释内容。然而，在经历了很多版本不一致导致的类的方法异常的报错后，我发

现，大部分论文中的注释都属于 PDAnnotationLink，更严重的是打印这些链接注释的内容会发现它们大部分都是空的，这就很难办了；

其次，我想到既然是论文，或许图片下方的注释文本的格式会有严格的要求，因此我希望找出一篇典型论文中图片下注释的文本字体、字号等格式信息，再根据这些格式信息进行匹配；然而，一方面，PDF 文件的格式定义太过于复杂，PDFBox 不同版本之间对字体格式方面的支持差异太大，同样的显示效果其格式信息可能完全不同；另一方面，一篇论文中即使格式相同的，也未必是图片下的注释文本，因此这个方向最后被我舍弃；

最后，我想到关注这些注释信息的文本组织上的格式。观察下面这些例子可以发现：

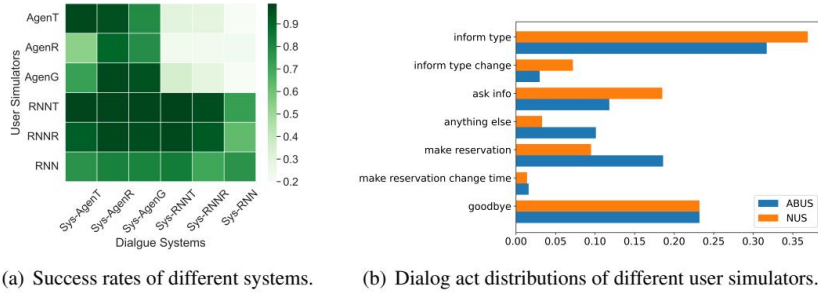


Figure 1: (a) is the heat map on the success rates of system agents tested by different user simulators on 200 dialogues. (b) shows the dialog act distributions of Agenda-based User Simulators (ABUS) and Neural networks-based User Simulators (NUS) provided by Shi et al. (2019). There exist seven user dialog acts annotated in the restaurant search task from MultiWOZ, as shown on the Y-axis.

图 8 图片注释的格式

Dialogue Systems		In-domain evaluation				Out-of-domain evaluation						All	
		Agent	AgentR	RNNT	GPT	AgentG	RNNR	RNN	Avg.↑	Std.↓		Avg.↑	Std.↓
single	Sys-Agent	97.5	54.0	98.5	78.0	72.5	92.5	77.0	80.7	8.6		81.4	14.8
	Sys-AgentR	96.0	90.0	98.5	80.5	97.5	97.5	82.0	92.3	7.3		91.7	7.1
	Sys-RNNT	30.5	23.0	99.0	75.5	35.5	97.5	84.0	72.3	26.6		63.6	30.5
	Sys-GPT	60.5	51.5	97.0	82.0	59.5	94.0	92.0	81.8	15.8		76.6	17.6
MUST	Sys-MUST _{merging}	97.5	83.5	94.5	80.5	97.5	94.0	82.5	91.3	6.4		90.0	6.9
	Sys-MUST _{uniform}	97.5	89.0	97.5	82.5	96.5	96.0	87.5	93.4	4.2		92.4	5.6
	Sys-MUST _{adaptive}	97.5	89.5	97.0	82.5	96.5	97.5	90.0	94.7	3.3		92.9	5.3

[1] The underlined number represents the success rate between a user simulator and its corresponding dialogue system trained by this user simulator. The increasing and decreasing percentages (in red and green colors) use the underlined numbers as the base success rates.

[2] ↓ (↑) indicates by what percentages the success rate has decreased (increased) compared with the base success rate by interacting with the same user simulator.

Table 2: The success rates of system agents testing on various user simulators. Each column represents a user simulator, each row represents a dialogue system trained with a specific simulator, e.g., Sys-Agent means the system trained with Agent. Each entry shows the success rate of a system agent when dealing with a user simulator. We use four simulators (Agent, AgentR, RNNT, and GPT) to implement MUST_{uniform} and MUST_{adaptive}.

图 9 表格注释的格式

不论图片还是表格,其注释的格式都是固定的,因此我们可以直接以Pattern的正则化匹配方式匹配形如“(Figure|Table \d+: (. *?) (\$| [.]))”的文本.

因此,最后我们通过 PDFTextStripper 类获取了 PDF 文件中的所有文本,并通过上面这个正则表达式进行了匹配,将匹配所得的所有文本连接起来即可传入 doc.add() 中.

④ `int indexAll(File dataDir)`

遍历 htmlNeedIndex 和 pdfNeedIndex, 分别调用处理 HTML 和 PDF 文件的方法, 最终返回索引的文件数, close 方法用于关闭 IndexWriter, 确保索引写入完毕.

⑤ `void indexHtmlFile(File file)`: 获取添加完索引信息的 HTML 文件.

⑥ `void indexPDFFile(File file)`: 获取添加完索引信息的 PDF 文件.

⑦ `IndexFiles(String indexDir)`

构造函数, 打开待索引文件目录, 配置标准分词器, 实例化写索引对象.

3.3 查询的设计

索引建立完成后, 查询同样也是使用 lucene 库中的方法, 具体代码结构如下图所示:

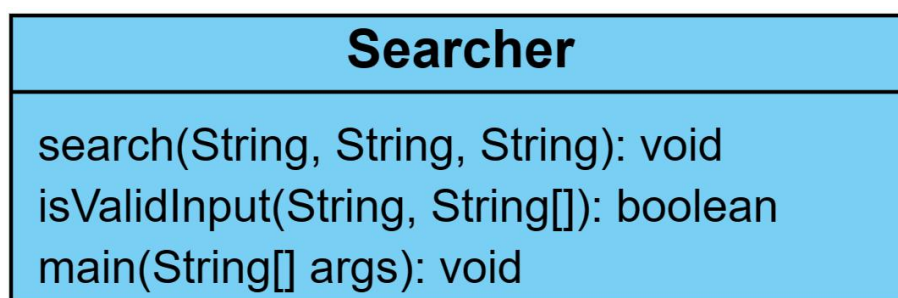


图 10 Searcher 类的 UML 图

下面是对图中各个方法的详细解释:

① `void search(String indexDir, String type, String q)`

打开索引目录, 获取 IndexReader 和 IndexSearcher 对象. 创建标准分词器和查询解析器, 解析查询字符串生成 Query 对象. 记录查询开始时间. 执行查询, 获取匹配的文档 (最多 100 条). 记录查询结束时间, 输出查询耗时和匹配到的记录数. 遍历查询结果, 输出文档信息. 关闭 IndexReader.

② **boolean isValidInput(String input, String[] validStrings)**

判断用户输入的字符串是否在给定的有效字符串数组 `validStrings` 中。如果存在匹配项，返回 `true`，否则返回 `false`。

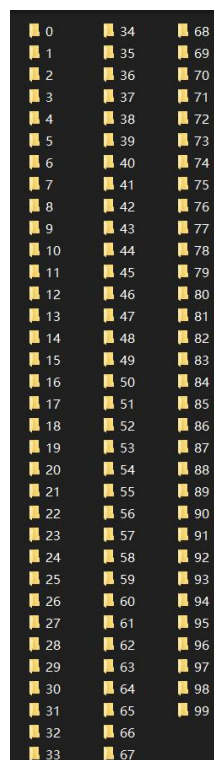
③ **void main(String[] args)**

显示程序标题和查询类型提示。用户选择查询类型。进入查询循环，验证用户输入的类型是否有效，如果无效则提示重新输入。用户输入查询字符串，执行查询并输出结果。显示查询类型提示，等待用户继续输入。如果用户输入 `"exit"`，退出循环，输出结束信息。

4 测试与运行

4.1 程序测试

在程序代码基本完成后，经过不断的调试与修改，最后测试本次所设计的 Scholar Search Engine 能够正常运行，基本功能与画面没有出现明显的错误和漏洞，在功能上已经基本达到要求，其他细节方面有待以后完善。测试数据如下图：



0	34	68
1	35	69
2	36	70
3	37	71
4	38	72
5	39	73
6	40	74
7	41	75
8	42	76
9	43	77
10	44	78
11	45	79
12	46	80
13	47	81
14	48	82
15	49	83
16	50	84
17	51	85
18	52	86
19	53	87
20	54	88
21	55	89
22	56	90
23	57	91
24	58	92
25	59	93
26	60	94
27	61	95
28	62	96
29	63	97
30	64	98
31	65	99
32	66	
33	67	

图 11 测试文件截图

4.2 程序运行

程序运行主界面如图 12 所示：

```
====Scholar-Search-Engine====  
Searching types available:  
Author    Title    Abstract  Venue    Graphs  
Your choice is:
```

图 12 程序主界面

查询作者名中含“Liu”，结果如图 13 所示：

```
====Scholar-Search-Engine====
Searching types available:
Author    Title    Abstract    Venue    Graphs
Your choice is: Author
Your prompt is: Liu
匹配Liu共耗时16毫秒
查询到17 hits条记录
1. Learning with Partial Annotations for Event Detection
2. Fantastic Expressions and Where to Find Them: Chinese Simile Generation with Multiple Constraints
3. Fine-tuning Happens in Tiny Subspaces: Exploring Intrinsic Task-specific Subspaces of Pre-trained Language Models
4. One Cannot Stand for Everyone! Leveraging Multiple User Simulators to train Task-oriented Dialogue Systems
5. When Does Translation Require Context? A Data-driven, Multilingual Exploration
6. Binary and Ternary Natural Language Generation
7. TECHS: Temporal Logical Graph Networks for Explainable Extrapolation Reasoning
8. A Close Look into the Calibration of Pre-trained Language Models
9. WeCheck: Strong Factual Consistency Checker via Weakly Supervised Learning
10. Joint Constrained Learning with Boundary-adjusting for Emotion-Cause Pair Extraction
11. Chain-of-Skills: A Configurable Model for Open-Domain Question Answering
12. DaMSTF: Domain Adversarial Learning Enhanced Meta Self-Training for Domain Adaptation
13. Tailor: A Soft-Prompt-Based Approach to Attribute-Based Controlled Text Generation
14. Test-time Adaptation for Machine Translation Evaluation by Uncertainty Minimization
15. NUWA-XL: Diffusion over Diffusion for eXtremely Long Video Generation
16. AMR-based Network for Aspect-based Sentiment Analysis
17. Layer-wise Fusion with Modality Independence Modeling for Multi-modal Emotion Recognition
```

图 13 查询作者的示例

查询标题中含有“Processing”，结果如图 14 所示：

```
Searching types available:
Author    Title    Abstract    Venue    Graphs
Your choice is: Title
Your prompt is: Processing
匹配Processing共耗时0毫秒
查询到0 hits条记录
```

图 14 查询标题的示例

查询摘要中含有“Processing”，结果如图 15 所示：

```
Searching types available:
Author    Title    Abstract    Venue    Graphs
Your choice is: Abstract
Your prompt is: Processing
匹配Processing共耗时1毫秒
查询到3 hits条记录
1. CLCL: Non-compositional Expression Detection with Contrastive Learning and Curriculum Learning
2. Self-Edit: Fault-Aware Code Editor for Code Generation
3. Post-Abstention: Towards Reliably Re-Attempting the Abstained Instances in QA
```

图 15 查询摘要的示例

查询会议中含有“ACL”，结果如图 16 所示：

```

75. DIONYSUS: A Pre-trained Model for Low-Resource Dialogue Summarization
76. MS-DETR: Natural Language Video Localization with Sampling Moment-Moment Interaction
77. Diverse Demonstrations Improve In-context Compositional Generalization
78. Self-Adaptive In-Context Learning: An Information Compression Perspective for In-Context Example Selection and Ordering
79. ACLM: A Selective-Denoising based Generative Data Augmentation Approach for Low-Resource Complex NER
80. On the Efficacy of Sampling Adapters
81. Cross-Domain Data Augmentation with Domain-Adaptive Language Modeling for Aspect-Based Sentiment Analysis
82. Compositional Data Augmentation for Abstractive Conversation Summarization
83. PMAES: Prompt-mapping Contrastive Learning for Cross-prompt Automated Essay Scoring
84. Marked Personas: Using Natural Language Prompts to Measure Stereotypes in Language Models
85. On Prefix-tuning for Lightweight Out-of-distribution Detection
86. GEC-DePend: Non-Autoregressive Grammatical Error Correction with Decoupled Permutation and Decoding
87. Measuring Progress in Fine-grained Vision-and-Language Understanding
88. Vision Meets Definitions: Unsupervised Visual Word Sense Disambiguation Incorporating Gloss Information
89. Chain-of-Skills: A Configurable Model for Open-Domain Question Answering
90. Natural Language to Code Generation in Interactive Data Science Notebooks
91. Elaboration-Generating Commonsense Question Answering at Scale
92. Neural Unsupervised Reconstruction of Protolanguage Word Forms
93. DaMSTF: Domain Adversarial Learning Enhanced Meta Self-Training for Domain Adaptation
94. On Evaluating Multilingual Compositional Generalization with Translated Datasets
95. FAA: Fine-grained Attention Alignment for Cascade Document Ranking
96. Fine-tuning Happens in Tiny Subspaces: Exploring Intrinsic Task-specific Subspaces of Pre-trained Language Models
97. Facilitating Multi-turn Emotional Support Conversation with Positive Emotion Elicitation: A Reinforcement Learning Approach
98. Query Enhanced Knowledge-Intensive Conversation via Unsupervised Joint Modeling
99. Why Aren't We NER Yet? Artifacts of ASR Errors in Named Entity Recognition in Spontaneous Speech Transcripts
100. Precise Zero-Shot Dense Retrieval without Relevance Labels

```

图 16 查询会议的示例

查询图表注释中含有“heat”，结果如图 17 所示：

```

Searching types available:
Author    Title    Abstract    Venue    Graphs
Your choice is: Graphs
Your prompt is: heat
匹配heat共耗时0毫秒
查询到1 hits条记录
1. D:\Subjects\JLP\homework4\Scholar-Search-Engine\src\main\resources\sources\62\2023.acl-long.62.pdf

```

图 17 查询图表的示例

查询类别输入错误，结果如图 18 所示：

```

Searching types available:
Author    Title    Abstract    Venue    Graphs
Your choice is: Author
Please choose from the types below:
Author    Title    Abstract    Venue    Graphs
Your choice is:

```

图 18 查询类别错误的示例

退出结果如图 19 所示：

```

Searching types available:
Author    Title    Abstract    Venue    Graphs
Your choice is: Author
Please choose from the types below:
Author    Title    Abstract    Venue    Graphs
Your choice is: exit
===Bye Bye===

```

图 19 退出的示例

5. 总结

总体来说，这次的项目还算简单，虽然我觉得它并没有帮助我对 Java 本身有更多的了解，我实际上只是学习了如何使用 Jsoup、PDFBox 以及 Lucene 这三个有力的 Java 工具。当然在这其中我遇到了许多问题，与 Java 语言本身相关的是用 `private` 修饰的类内的方法不能使用 `private` 成员变量；

此外，在使用 Apache PDFBox 时我深刻认识到了，版本更新是一件多么重大的事，以及 2.0.30 和 3.0.1 之间竟有如此大的差异，以及对于 PDF 文件字体的格式获取，竟然是 1.8.17 版本最为方便；

最后，通过该课程设计，我们能全面系统的理解了程序构造的一般原理和基本实现方法。把死板的课本知识变得生动有趣，激发了学习的积极性，能够把课堂上学的知识通过自己设计的程序表示出来，加深了对理论知识的理解。现在通过自己动手做实验，从实践上认识了操作系统是如何处理命令的，课程设计中程序比较复杂，在调试时应该仔细。

参考文献

- [1] [java 爬虫: JSOUP-CSDN 博客](#)
- [2] [Apache PDFBox | A Java PDF Library](#)
- [3] [Apache Lucene - Welcome to Apache Lucene](#)
- [4] [jsoup: Java HTML parser, built for HTML editing, cleaning, scraping, and XSS safety](#)