司法搜索引擎系统的设计与实现

熊天翼　　严澜
计 95　　建 82
2019011303　　2018010005

1 需求描述

根据给定的中文法律数据集，实现一个司法搜索引擎系统。

• 核心功能：用户界面，关键词检索，案例详情监视。

• 其他功能：高级检索，关键词高亮，标签显示与搜索，基于法条搜索，类似案例推荐，关键词高级检索，以案搜案。

2 框架设计

- **前端**：Vue 框架 + Element UI，基于色色引擎 UI 实现。
  https://github.com/YunYouJun/sese-engine-ui
- **后端** Django + Whoosh + sqlite3

3 核心模块设计

3.1 数据的导入与存储

用 xml.etree.ElementTree 库对 xml 文件进行解析，提取全文、文首、当事人、法律条文、案件基本情况、裁判分析过程、判决结果、文尾、文书名称、审判程序、法庭、审判理由、省份、年份共十四个字段，使用 tree.getroot().find('.//xxxx').get('value') 找到相应字段。剔除不满足上述要求的法律案件，总共成功从 68417 个法律案例中导入了 50548 个案件。

在 django 中，针对案件（Case）和法条（Law）分别建立模型，对应于 sqlite3 数据库中的两张数据表。模型对应的字段如下

```
class Law(models.Model):
    id = models.AutoField(primary_key=True) # 法律id
    name = models.CharField(max_length=50) # 法条名称（在导入数据时保证不出现重复）
    def __str__(self) -> str:
        return self.name

class Case(models.Model):
    id = models.AutoField(primary_key=True)
    qw_value = models.TextField(max_length=500) # 全文
```

```
    head = models.CharField(max_length=200) # 文首
    related_people = models.TextField() # 当事人
    judicial_record = models.TextField() # 诉讼记录
    basic_info = models.TextField() # 案件基本情况
    judgement_process = models.TextField() # 判决分析过程
    result = models.TextField() # 判决结果
    tail = models.TextField() # 文尾
    note_name = models.CharField(max_length=30) # 文书名称
    judge_prop = models.CharField(max_length=30) # 审判程序（一审/二审）
    court = models.CharField(max_length=30) # 经办法院
    case_reason = models.CharField(max_length=30) # 案由
    province = models.CharField(max_length=200) # 行政区划-省份
    year = models.IntegerField(default=2023) # 年份
    laws = models.ManyToManyField(to=Law, related_name="cases") # 法律条文
```

由于一个案件中可能与多个法条相关，且同一法条可能

3.2 使用 Whoosh 框架建立索引与分词

使用 Whoosh 作为搜索引擎框架，Whoosh 是一个用 Python 编写的全文搜索引擎库，支持倒排索引的自动建立，提供简单易用的 API，支持快速搜索、排序和定制。

使用 Whoosh 进行中文分词时，需导入 Whoosh 源代码 whoosh_backend_ZW.py，替换部分不兼容包，并将 analyzer 替换为 ChineseAnalyzer()，最后在 settings.py 中设置对应路径。Whoosh 自动建立分词和倒排索引，存放在 whoosh_index 目录下。

3.3 前端页面

前端页面共有三个，入口首页面、查询页面和详情页。

入口首页和查询页面均支持基础搜索、高级检索和类案检索三种搜索功能。详情页面展示案件具体内容，以及相关法律条文和案例推荐。

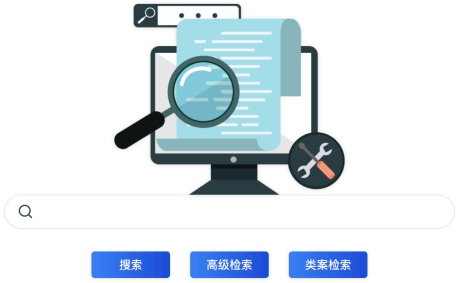

图 1: 搜索引擎首页面

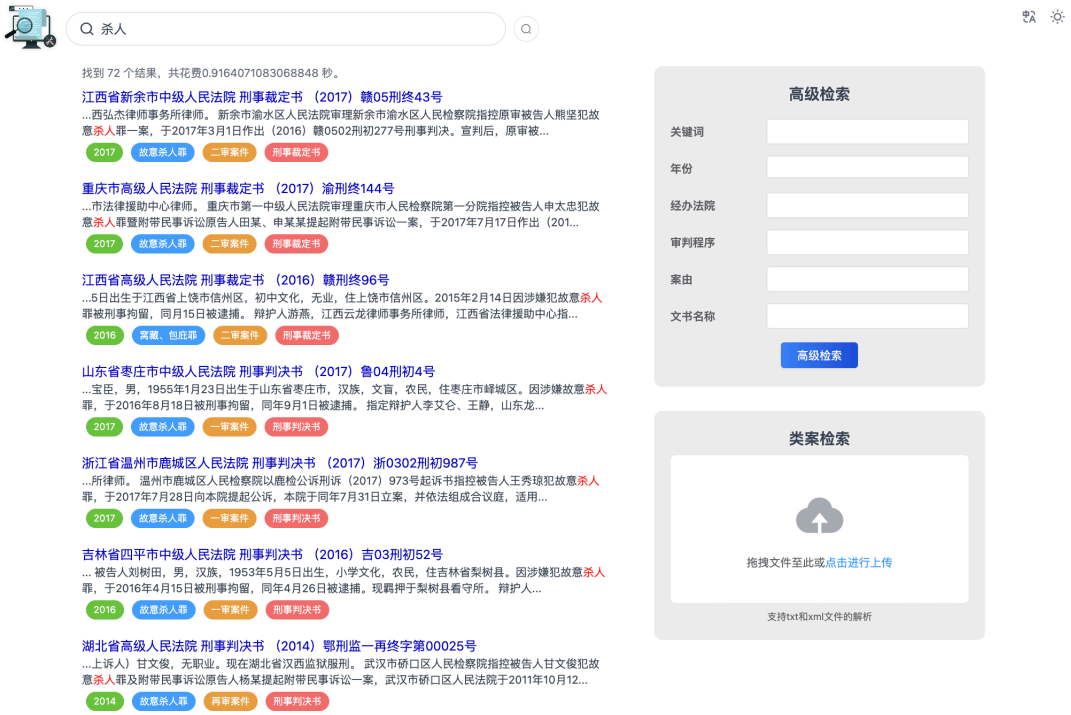

图 2: 搜索引擎查询页面

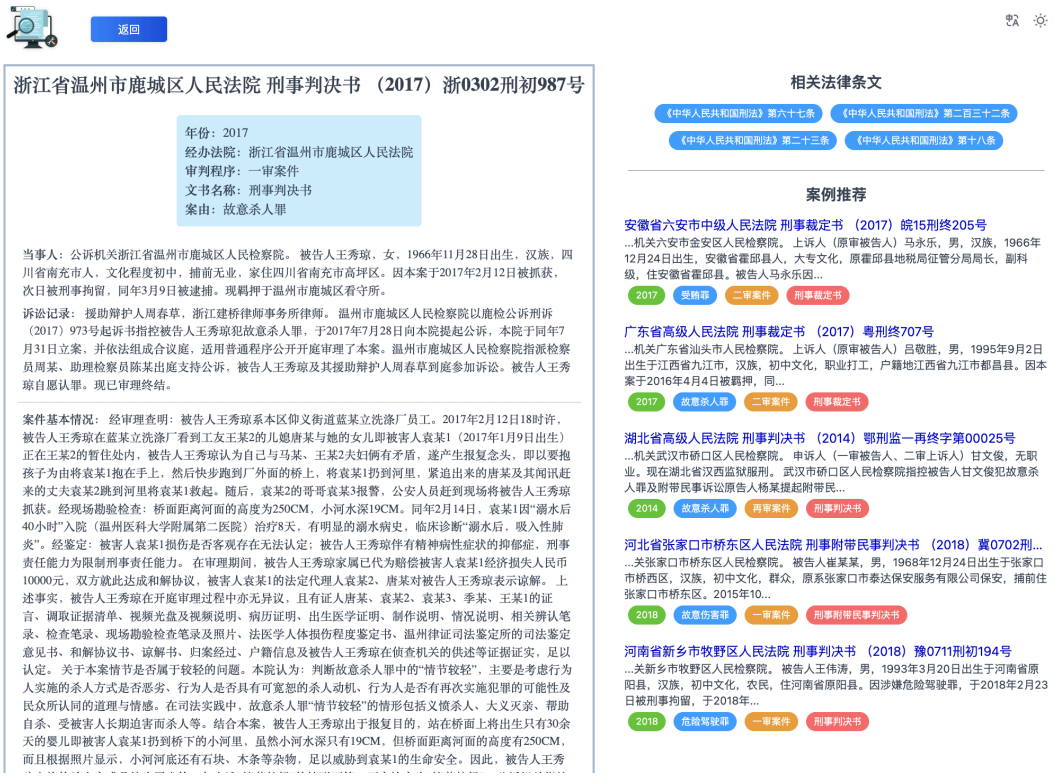

图 3: 搜索引擎详情页面

3.4 基础搜索

基础搜索支持对于给定搜索词句，返回相关的案例。用户可在入口首页面或者查询页面输入需要搜索的词句，随后搜索引擎将采用 Whoosh 框架自动对搜索词句进行分词去除停用词等处理，对于符合要求的返回结果进行随机排序返回，返回的结果采用分页器支持翻页查看。

对于每一条返回结果，支持案件标题的有限长度显示，搜索关键词的高亮显示（红色），法律案件关键标签的显示（彩色 tag），以及对应标签类别的法案的扩展搜索（点击彩色 tag 可自动返回同类案件）。

用户点击感兴趣的法律案件标题即可进入详情页，详情页分为左右两部分，左侧包含案件全文和案件关键标签的汇总展示（包含年份、经办法院、审判程序、文书名称和案由），右侧为案件涉及的法律条文一览，以及案例推荐。

3.5 高级检索

高级检索支持关键词、年份、经办法院、审判程序、案由、文书名称六个字断的共同匹配。

用户可以在查询页面使用高级搜索，选取六个字断中的任意一到六个进行安检搜索，搜索引擎将返回各个字段搜索结果的交集。

3.6 类案检索

类案检索支持上传案件并检索相似案件。

用户可以任意文件格式上传法律案件，如果文件为 xml 文件并包含文首、当事人、法律条文、案件基本情况、裁判分析过程、判决结果、文尾、文书名称、审判程序、法庭、审判理由、省份、年份十四个字段，搜索引擎采用基于案由和文书名称的匹配方式返回相似案件。

对于其他格式的文件或者不满足上述十四个字段的 xml 格式案件，我们收集到了案件关键词，搜索引擎将基于给定的案件关键词匹配相似案件。

//加人工放入的关键词

4 测试结果及样例分析

4.1 关键词测试

//2020 北京盗窃

//强奸

4.2 案例测试

//xml

//江哥

//版权

5 实现功能明细

5.1 搜索

5.2 筛选

5.3 推荐

5.4 界面

6 参考资料

在作业的实现过程中，我们主要参考了 django、Vue、Whoosh、ElementUI 的说明文档，并对于具体问题在参考了 CSDN 和 stackoverflow 上的解决方案。在作业的开始阶段，我们学习了网络上一份 Django + Haystack + Whoosh 实现的搜索引擎代码，了解了搜索引擎的大致实现方式。

7 总结

感谢刘老师和艾老师的辛勤付出和专业教导，以及两位助教学长的耐心答疑解惑，在本学期的搜索引擎课程中，我们对搜索引擎有了全面的认识，不但了解到了各个子系统的原理和技术，也在业界的专家讲座中管窥了推广搜的实战。

在本次法律搜索引擎的大作业中，我们共同研究和实践了不同搜索框架的使用，我们一起分析需求、制定计划，并协作完成了法律搜索系统的搭建和调优。通过大作业，我们深化了索引构建、查询解析、性能优化等方面的实际操作技能，同时也锻炼了团队合作和沟通能力。

7.1 小组分工

- **熊天翼**：前端及部分后端，基础搜索与高级搜索，共同完成代码调试和文档撰写。
- **严澜**：后端及部分前端，类案检索，共同完成代码调试和文档撰写。