****



**研 究 生 毕 业 论 文**

**（申请工程硕士学位）**

|  |  |
| --- | --- |
| **论文题目** | 裁判文书的文本信息抽取与分类  统计系统的设计与实现 |
| **作者姓名** | 高永伟 |
| **学科、专业名称** | 工程硕士(软件工程领域) |
| **研究方向** | 软件工程 |
| **指导教师** | 葛季栋　副教授 |

**2019年4月30日**

**学 号： MF1732028**

**论文答辩日期： 2019 年 5 月 23 日**

**指 导 教 师： （签字）**

**裁判文书的文本信息抽取与分类统计系统的设计与实现**

|  |  |
| --- | --- |
| **作 者:** | **高永伟** |
| 指导教师: | **葛季栋　副教授** |

|  |
| --- |
| **南京大学研究生毕业论文** |
| **(申请工程硕士学位)** |

|  |
| --- |
| **南京大学软件学院** |
| **2019年4月** |

**Design and Implementation of Text Information Extraction and Classification Statistics System for Judgment Documents**

**Gao, Yongwei**

**Submitted in partial fulfillment of the requirements for the degree of Master of Engineering**

Supervised by

Associate Professor **Ge, Jidong**

Software Institute

**NANJING UNIVERSITY**

Nanjing, China

April, 2019

# 摘 要

裁判文书记载人民法院审理的过程和结果，它是诉讼活动结果的载体。法院经常需要针对不同主题或者案由的案件做专题分类统计，但是相关分类统计项没有记录在审判流程系统的关系数据库中，因此统计的难度非常大，相关统计的数据源只能从裁判文书的文本中获取，所以必须针对裁判文书的文本数据开展深层次的挖掘分析。

本文针对案件分类统计的难点，设计并实现了裁判文书的文本信息抽取与分类统计系统。本文设计了裁判文书通用型文本信息抽取模块，针对各段落的特征词，将裁判文书划分为文首、诉讼参与人、诉讼记录、案件基本情况、裁判分析过程、文尾这六大段落，并在各段落中进一步抽取更详细的信息。本文设计了要素式文本信息抽取模块，在通用型文本信息抽取的基础上进一步抽取裁判文书的要素信息，本文将要素项分为两类，一类建立规则去抽取，另一类转为分类问题，构建TF-IDF与word2vec结合的特征，使用xgboost模型去分类。本文设计了文书分类统计模块，结合中国司法大数据研究院专题报告中的分类统计项，依据上述结果，设计了各统计项分布的计算方法，并将统计报告发布到网络平台展示。本文还设计了文书管理模块，管理信息抽取和统计计算过程中生成的文件。

本系统总体上采用了分层的结构。通用型文本信息抽取模块和要素式文本信息抽取模块作为信息抽取层提供裁判文书的基本信息。文书分类统计模块则作为信息统计层，用来计算统计项的分布。展示层则负责发布展示结果。本文使用Vue.js和Spring MVC框架搭建了Web系统，结合可视化工具Echarts将统计结果展示给用户。本文对系统各个模块的详细设计与实现进行了描述，并通过系统测试验证了系统的性能达到要求。

**关键词**： 信息抽取，Spring MVC，XGBoost，Word2vec，TF-IDF

# Abstract

The judgment documents record the process and results of the people's courts, and it is the carrier of the outcome of litigation activities. The court often needs to do thematic classification statistics for different themes or cases, but we do not record the relevant classification statistics in the relational database of the trial process system, so the statistic is very difficult. The data source of relevant statistics can only be achieved from the text of the judgment documents. Therefore, we must carry out in-depth mining analysis on the text data of the judgment documents.

This thesis designs and implements the text extraction and classification statistical system of the judgment document for the difficulty of case classification and statistics. This thesis designs a general information extraction module for the judgment documents. With the feature words of each paragraph, we divide the judgment documents into six paragraphs: the first sentence, the litigation participant, the litigation record, the basic situation of the case, the judgment analysis process, and the end of the text. Then we further extract detailed information in each paragraph. In this thesis, we also design the elementary information extraction module. Based on the general text information extraction, we extract the element information of the judgment document. This thesis divides the element items into two categories, one class establishes rules to extract and the other class turns the extraction into text classification problems. We construct the combination of TF-IDF and word2vec as the feature of text, and use the xgboost model to classify the documents. This thesis designs the document statistics module, combined with the classification statistics items in the special report of China Judicial Big Data Research Institute, and designs the calculation method of each statistical item distribution. It also publishes the statistical report to the network platform for display. This thesis designs a document management module to manage the files generated during the information extraction and statistical calculation process.

The system adopts a layered structure as general structure. The general information extraction module and the elementary information extraction module provide basic information of the judgment document as the information extraction layer. We use the statistics module as the information statistical layer to calculate the distribution of statistical items. The display layer is responsible for publishing the results of the presentation. This article uses Vue.js and Spring MVC framework to build a Web system, combined with the visualization tool Echarts to show the statistical results to users. This thesis describes the detailed design and implementation of each module of the system, and verifies that the performance of the system meet the requirements through system testing.

**Keywords**: Information extraction, Spring MVC, XGBoost, Word2vec, TF-IDF

**目录**

[摘 要 I](#_Toc9434992)

[Abstract II](#_Toc9434993)

[图目录 VI](#_Toc9434994)

[表目录 VIII](#_Toc9434995)

[第一章 引言 1](#_Toc9434996)

[1.1 项目背景 1](#_Toc9434997)

[1.2 国内外研究现状 2](#_Toc9434998)

[1.3 本文主要的研究工作 3](#_Toc9434999)

[1.4 本文的组织结构 5](#_Toc9435000)

[第二章 技术综述 6](#_Toc9435001)

[2.1 Spring MVC 6](#_Toc9435002)

[2.2 Vue.js 8](#_Toc9435003)

[2.3 TF-IDF 9](#_Toc9435004)

[2.4 Word2vec 10](#_Toc9435005)

[2.5 xgboost 11](#_Toc9435006)

[2.6 本章小结 13](#_Toc9435007)

[第三章 裁判文书的文本信息抽取与分类统计系统的需求分析与设计 14](#_Toc9435008)

[3.1项目总体规划 14](#_Toc9435009)

[3.2 系统需求分析 15](#_Toc9435010)

[3.2.1 系统用例图 15](#_Toc9435011)

[3.2.2 系统的功能需求 15](#_Toc9435012)

[3.2.3 系统的非功能需求 17](#_Toc9435013)

[3.3系统总体设计 17](#_Toc9435014)

[3.3.1 系统的逻辑视图 17](#_Toc9435015)

[3.3.2 系统的开发视图 18](#_Toc9435016)

[3.3.3 系统的进程视图 19](#_Toc9435017)

[3.3.4 系统的物理视图 20](#_Toc9435018)

[3.3.5 系统的技术架构 20](#_Toc9435019)

[3.4 系统模块设计 21](#_Toc9435020)

[3.4.1 通用型文本信息抽取模块的设计 21](#_Toc9435021)

[3.4.2 要素式文本信息抽取模块的设计 26](#_Toc9435022)

[3.4.3 文书分类统计模块的设计 33](#_Toc9435023)

[3.4.4 文书管理模块的设计 34](#_Toc9435024)

[3.5 本章小结 35](#_Toc9435025)

[第四章 裁判文书的文本信息抽取与分类统计系统的实现 36](#_Toc9435026)

[4.1 通用型文本信息抽取模块的实现 36](#_Toc9435027)

[4.1.1 裁判文书段落划分功能的实现 36](#_Toc9435028)

[4.1.2 段落详细信息抽取功能的实现 38](#_Toc9435029)

[4.2 要素式文本信息抽取模块的实现 47](#_Toc9435030)

[4.2.1 通用信息结果加载功能的实现 47](#_Toc9435031)

[4.2.2 规则匹配信息抽取功能的实现 48](#_Toc9435032)

[4.2.3 监督学习信息抽取功能的实现 51](#_Toc9435033)

[4.2.4 要素式文本信息抽取的结果 59](#_Toc9435034)

[4.3 文书分类统计模块的实现 59](#_Toc9435035)

[4.3.1 分类统计记录功能 60](#_Toc9435036)

[4.3.2 网络平台发布功能 62](#_Toc9435037)

[4.4 文书管理模块的实现 64](#_Toc9435038)

[4.5 系统测试 65](#_Toc9435039)

[4.5.1测试目标 65](#_Toc9435040)

[4.5.2 测试过程与结果 65](#_Toc9435041)

[4.6 本章小结 67](#_Toc9435042)

[第五章 总结与展望 68](#_Toc9435043)

[5.1 论文总结 68](#_Toc9435044)

[5.2 进一步工作展望 68](#_Toc9435045)

[参 考 文 献 70](#_Toc9435046)

[致 谢 73](#_Toc9435047)

[版权及论文原创性说明 74](#_Toc9435048)

# 图目录

图2.1 MVC模块关系图 6

图2.2 Spring MVC工作流程图 7

图2.3 Vue.js搭建的界面的组件树 9

图2.4 word2vec的模型原理图 11

图3.1 系统的总体结构图 14

图3.2 系统用例图 15

图3.3 系统的逻辑视图 18

图3.4 系统的开发视图 18

图3.5 系统的进程视图 19

图3.6 系统的物理视图 20

图3.7 系统架构图 20

图3.8 《人民法院民事裁判文书制作规范》的部分信息 21

图3.9 裁判文书段落划分功能的UML图 22

图3.10 文首详细信息抽取的UML类图 23

图3.11 诉讼参与人详细信息抽取的UML类图 23

图3.12 诉讼记录详细信息抽取的UML类图 24

图3.13 案件基本情况详细信息抽取的UML类图 24

图3.14 裁判分析过程详细信息抽取的UML类图 25

图3.15 文尾详细信息抽取的UML类图 25

图3.16 通用型民事判决文书信息抽取模块流程图 25

图3.17 基于监督学习去抽取的要素项 26

图3.18 基于规则匹配去抽取的要素项（部分） 27

图3.19 通用信息结果加载功能的UML类图 27

图3.20 基于规则匹配进行要素信息抽取的UML类图 28

图3.21 模型训练和运行流程图 31

图3.22 基于监督学习进行要素信息抽取的UML类图 32

图3.23 要素式文本信息抽取的流程图 33

图3.24 文书分类统计模块的UML图 34

图4.1 裁判文书段落划分功能的部分代码 37

图4.2 文首段落示例 38

图4.3 文首详细信息抽取的部分代码 39

图4.4 诉讼参与人段落示例 40

图4.5 诉讼参与人详细信息抽取的部分代码 41

图4.6 诉讼记录段落示例 41

图4.7 诉讼记录详细信息抽取的部分代码 42

图4.8 案件基本情况段落示例 43

图4.9 案件基本情况详细信息抽取的部分代码 44

图4.10 裁判分析过程段落示例 44

图4.11 裁判分析过程详细信息抽取的部分代码 45

图4.12 文尾段落示例 46

图4.13 文尾详细信息抽取的部分代码 46

图4.14 通用型文本信息抽取模块的部分结果展示 47

图4.15 抽取“相识时间和如何相识”要素的部分代码 49

图4.16 “夫妻共同债权、债务情况”要素抽取的部分代码 50

图4.17 “是否存在一方下落不明两年并查找不得情形”要素抽取的部分代码 51

图4.18 max\_depth参数测试图 55

图4.19 num\_boost\_round参数测试图 56

图4.20 不同分类器分类结果对比图 57

图4.21 分词、去停用词功能代码 58

图4.22 要素式文本信息抽取模块的抽取结果（部分） 59

图4.23 判决结果统计分布（天津法院部分数据集）展示结果 63

图4.24 婚姻存续时间统计分布（天津法院部分数据集）展示结果 64

图4.25 文件移动方法代码 64

图4.26 Jmeter测试报告 66

图4.27 Jmeter测试聚合图 67

# 表目录

表4.1 裁判文书各段落划分特征词统计表 36

表4.2 裁判文书段落划分功能主要类统计表 37

表4.3 文首段落抽取的信息表 39

表4.4 文首详细信息抽取的主要类统计表 39

表4.5 诉讼参与人段落的BaseSSCYRModel.SSCYR部分属性值表 40

表4.6 诉讼参与人详细信息抽取的主要类统计表 41

表4.7 诉讼记录段落的BaseSSJLModel属性值表 42

表4.8 诉讼记录详细信息抽取的主要类统计表 42

表4.9 案件基本情况段落的BaseAJJBQKModel属性值表 43

表4.10 案件基本情况详细信息抽取的主要类统计表 43

表4.11 裁判分析过程段落的BaseCPFXGCModel属性值表 45

表4.12 裁判分析过程详细信息抽取的主要类统计表 45

表4.13 文尾段落的BaseWWModel属性值表 46

表4.14 文尾详细信息抽取的主要类统计表 46

表4.15 通用信息结果加载功能主要类统计表 48

表4.16 段落详细信息XML保存结果含义对应（部分）表 48

表4.17 混淆矩阵 54

表4.18 两类别某要素分类结果 57

表4.19 监督学习要素抽取结果统计表 58

表4.20 文书分类统计模块的前端内容 63

表4.21 信息抽取负载测试结果 65

# 引言

## 1.1 项目背景

随着法院信息化3.0的进一步发展，各种基于云计算，大数据，人工智能等新技术的工具逐渐进入到法院审判的各个角落。新技术的出现减轻了法院人员工作负担，极大提高了法院审理和执行的透明度和效率[王辰阳, 2018]。经过多年的努力，法院构建了法院专网、移动专网等五大网络基础设施体系，通过中国审判流程信息公开网、中国裁判文书网等公共平台以及其余配套措施，初步达成了构建智慧法院的目标，中国法院的信息化水平总体上达到世界先进水平[最高人民法院, 2018]。

法院裁判文书作为法官审理活动的重要载体，蕴含了丰富的信息。多年以来，中国法院在审理过程中积累了数目庞大的裁判文书，根据中国裁判文书网的数据显示，截止2019年2月25日，网站共收录63517021篇文书**[中国裁判文书网, 2019]。**法院经常需要针对不同主题或者案由的案件做专题分类统计，但是相关分类统计项没有记录在审判流程系统的关系数据库中，因此统计难度非常大，相关统计的数据源只能从裁判文书的文本中获取，所以必须针对裁判文书的文本数据开展深层次的挖掘分析。

法院裁判文书本身有着非常严格的写作体裁。在最高人民法院提供的《人民法院民事裁判文书制作规范》中指出，民事裁判文书必须包含判决时间，案号，辩诉当事人信息，审理经过，双方辩诉内容，判决结果，法官身份等详细而丰富的信息，这些信息在本文中称之为通用信息，裁判文书本身严格的句法句式也为本文接下来的工作奠定了部分基础。

在法院的审理中有一类审理方式是要素式审判。要素式审判，就是根据案件的基本要素进行庭审并制作裁判文书的一种审判方法[中国法院网, 2018]。具体而言，要素式审判就是在法官审理过程中，对那些能够概括总结出该案情要素的案件，进行要素提炼，并对辩诉双方就案件中各种要素是否存在不同意见进行整理归纳[张春波, 2016]。案由要素是案件的重要特征，提取出裁判文书中的案由要素信息将有助于对文书进行专题分类，而该类信息在本文中被称为要素信息。

综合以上背景，本文需要对法院裁判文书进行信息抽取，将裁判文书由非结构化数据转变成半结构化数据。利用抽取出来的信息，帮助法院对裁判文书进行专题分类统计。

## 1.2 国内外研究现状

文本信息抽取技术是从自然语言文本中抽取结构化信息的一项技术，该研究最早始于上世纪60年代中期，以两个长期的、研究性的自然语言处理项目作为它的代表项目。其一是美国纽约大学开展的Linguistic String项目，该项目着力于建立一个英语计算语法规则库，利用其中的语法规则从医疗领域的报告文件以及相关文书中抽取信息[Sager et al., 1981]。这种信息抽取是基于模板（Template）和模式（Pattern）的一种方式，其前提条件是待抽取的自然语言文本必须拥有一定模式和固定语法结构[Grishman, 2015]。另一个项目是耶鲁大学开展的关于故事理解的研究，其研究目的是建立一个基于故事脚本理论的信息抽取系统，利用该系统从新闻报道中抽取信息，新闻报道的场景包含诸多领域，比如地震或者工人罢工等[DeJong, 1982]。信息抽取研究领域的进展得益于信息理解系列会议（MUC，Message Understanding Conference），MUC系列会议推动了信息抽取概念和技术的发展，使其成为自然语言处理领域的一个重要组成部分[李保利等, 2003]。

目前国内外关于文本信息抽取的工作有两种比较主流的方式，一种是基于规则匹配的方法，即使用模板和模式去获取文本信息。另一种是基于监督学习的方法进行信息抽取。所谓的监督学习是机器学习的一个分支，是指事先提供一定规模的已标注文本，然后利用机器学习算法进行自动归纳。究其本质是利用统计学去描述语言模型，通过衡量待匹配短语与目标抽取对象的匹配概率去作出判断[石倩等, 2008]。利用监督学习进行信息抽取有多种途径，使用关系抽取，实体识别等新技术可以抽取信息[Bach N et al，2007]，使用分类算法获取文本类别也是一种抽取信息的方式[刘迁等, 2007]。

目前在文本信息抽取应用领域，有部分研究者研究关于Web内容的信息抽取[Kanhabua et **al.,** 2016]，Web网页因为其独有的标签化语言方式，可以较为容易的建立相对应的模式（Pattern），然后利用模式（Pattern）去获取文本的关键信息[周合明等, 2011]。考虑到法院裁判文书本身所具有的固定写作体裁，这与Web信息抽取的场景非常相似，因此本文在通用型文本信息抽取模块采用建立规则的方式抽取裁判文书的文本信息，而对于要素式文本信息抽取模块所需要抽取的要素信息，一部分采用了规则匹配方式抽取，另一部分由于语法句式的不确定等原因，采用了监督学习的方式抽取。

司法统计是对司法活动中收集的系统性数据，进行分类整理、汇总描述、量化分析和总结的过程，利用统计得出的结果，可以为相关司法领域的问题提供决策和论证的依据与参考[翟全军等, 2017]。目前我国司法统计基本上是以案件信息为核心，以司法统计报表为重要补充的司法统计模式，但总体上信息化的水平较低且信息系统建设不够完善。司法统计包含了许多方面的统计工作，其中有的工作是对案件进行专题统计分析。案件专题统计分析涉及的许多统计项有时并没有被记录在审判流程系统的关系数据库中，相关的统计难度非常大。随着大数据技术的发展，司法统计也开始转变思维，以数据为出发点，让数据说话，利用大数据技术辅助司法统计工作，案件专题统计也更加简单。

本文从裁判文书中抽取的通用信息和要素信息可以有效的促进法院司法专题统计工作，去发掘隐藏在众多裁判文书中丰富的信息。

## 1.3 本文主要的研究工作

本文研究的是法院裁判文书的文本信息抽取与分类统计系统，是以裁判文书写作体裁与特征词为核心，进行通用信息抽取和要素信息抽取，然后对裁判文书进行专题分类统计的系统。本系统主要由通用型文本信息抽取模块、要素式文本信息抽取模块、文书分类统计模块和文书管理模块等功能模块组成，可以对文书进行信息抽取和分类统计，挖掘出隐藏在裁判文书里的事实和经验。

本文设计了通用型文本信息抽取模块去抽取裁判文书的通用信息（即裁判文书制作规范中要求的信息），将裁判文书由非结构化的数据转变成半结构化的数据。本文为该模块设计了两个主要功能，裁判文书段落划分功能和段落详细信息抽取功能。裁判文书段落划分功能是指，由于裁判文书有比较规范的写作结构和特征词，基于这些重要特征词，将某篇文书划分为文首、诉讼参与人、诉讼记录、案件基本情况、裁判分析过程、文尾这六个段落。段落详细信息抽取功能是指从每个段落中去抽取更详细的信息，包括：

（1）在文首部分抽取：文书种类、经办法院、案号等信息；

（2）在诉讼参与人部分抽取：原告、被告、代理人等相关信息；

（3）在诉讼记录部分抽取：受理日期、立案日期、审判组织、开庭审理信息、开庭日期、出庭当事人信息等；

（4）在案件基本情况部分抽取：民事案件的案件基本情况包括：原告诉称段、被告辩称段、原告证据、被告证据、争议焦点、质证结论、查明事实段。由于刑事案件的文书结构与此有着较大的区别，所以无法使用此模板，需在分析文书段落前获取案件性质再作处理；

（5）裁判分析过程部分抽取：审理结论、引用法条、判决结果；

（6）文尾部分抽取：裁判时间、审判组织成员。

本文在通用型文本信息抽取的基础上，设计了要素式文本信息抽取模块，利用相关案由要素表（本文以离婚纠纷案由表为主），制定要素的抽取规则，从而获取裁判文书的要素信息，为文书的分类统计打下基础。该模块需要对离婚纠纷案由的要素抽取，设计了如下相关功能。通用信息抽取结果加载功能，即加载通用型文本信息抽取模块输出的结果，并根据文档节点属性读入到系统中。然后根据上述加载结果并结合离婚纠纷案由要素表去抽取要素信息，这里将案由要素分成两类情况去抽取，一类是使用规则匹配抽取，另一类使用监督学习的方法抽取，所以分别设计了规则匹配信息抽取功能和监督学习信息抽取功能，本文将结合要素项实际内容选择抽取方式。监督学习抽取信息是使用Python语言实现的，而程序主体是Java程序，本文使用socket完成进程通信。

文书分类统计模块的主要设计目的是利用通用型文本信息抽取模块和要素式文本信息抽取模块抽取出来信息，设计各统计项分布的计算方法，并将统计分布的结果发布到网络平台展示出来，本文为该模块设计了两个功能，一是分类统计记录功能，二是网络平台发布功能。本文根据中国司法大数据研究院发布的关于离婚纠纷报告里面的统计项，设计了分类统计记录功能。本文将统计离婚纠纷案件裁判文书的以下相关信息：

（1）根据案件审结情况统计各年的案件量分布；

（2）根据当事人特征统计原告性别分布、原被告年龄差分布；

（3）根据案件特征去统计婚姻存续时间、离婚态度、判决结果、家暴案件省份分布。

本文将依据研究院报告里提供的统计项，利用之前抽取的裁判文书的通用信息和要素信息，计算裁判文书的各统计项分布情况。本文还设计了网络平台发布功能，将统计结果可视化后发布到网络平台，同时也提供数据请求接口给开发者，方便以后将统计结果作它用。

本文还设计了文书管理模块去承担通用信息抽取、要素信息抽取、分类统计计算中过渡文件的管理。

本文对裁判文书的通用型文本信息抽取和要素式文本信息抽取的方式进行了详细介绍，包括规则建立方式和xgboost算法的使用。同时结合具体的分类统计项，阐述如何根据信息抽取的结果去计算统计项的分布。本文也介绍了Spring MVC，Vue.js，Echarts等技术的原理和使用方法，并介绍如何利用这些开发技术去展示裁判文书的分类统计结果。

## 1.4 本文的组织结构

第一章 引言部分。主要介绍了项目的背景，国内外研究现状，本文的主要工作内容等。

第二章 技术综述。主要介绍了裁判文书的文本信息抽取与分类统计系统使用的那些技术，包括Spring MVC、Vue.js、Word2vec、XGBoost等技术。

第三章 裁判文书的文本信息抽取与分类统计系统的需求分析与设计。对裁判文书的文本信息抽取与分类统计系统进行了需求分析，介绍了项目各模块的设计思路，包括通用型文本信息抽取模块，要素式文本信息抽取模块、文书分类统计模块和文书管理模块，并介绍了各个模块之间的关系和模块的构成。

第四章 裁判文书的文本信息抽取与分类统计系统的实现。在需求分析和设计的基础上，详细的介绍了各模块的实现细节，阐述了文书信息抽取的规则构造由来和具体的构建细则，同时也详细介绍了如何使用信息抽取的结果去计算相应统计项的分布，最后测试了系统性能。

第五章 总结与展望。介绍了裁判文书的文本信息抽取与分类统计系统做了哪些工作，并分析项目未来工作的方向和目标。

# 技术综述

本章将对项目中使用到的Spring MVC、Vue.js、Word2vec、TF-IDF以及xgboost等技术进行介绍。

## 2.1 Spring MVC

MVC是Model-View-Controller的简称，即模型-视图-控制器[Johnson R E.,1997]。这是一种被称为模式之王的设计模式，在许多场景都可以利用它来构建软件。在交互系统这一应用场景里面，MVC把软件系统的组成分解成模型、视图、控制器三个部分。Model通常表示的是企业数据和业务规则，包括了数据计算和持久化等功能，同一个Model可以被多个View反复使用。View指应用的表现层，用来展示数据或者提供其它视图。Controller指应用的控制器，控制器从用户那里接收请求，根据请求的参数映射到对应的处理方法，然后执行相应的业务逻辑处理请求，再将执行结果返回给客户端。MVC三部分之间彼此独立，提高了应用程序的可配置性和灵活性。MVC模式框架实现了视图层和业务层的分离，在修改视图层后可以不用再次编译模型与控制器层的代码。这种分层的设计，使得需要修改业务逻辑时，只改动模型层即可[张雪敏, 2018]，MVC模块关系图如2.1所示。



图2.1 MVC模块关系图

Spring MVC是一个以Spring IoC容器为基础的轻量级Web框架，是一种对MVC模式的实现，其架构非常接近MVC模式的核心概念和特性。Spring MVC可以容纳多种视图技术，实现了应用程序的高度解耦，它明确划分了Web项目中各模块的职责，且支持动态配置组件[翟剑锟, 2013]。Spring MVC的工作流程如图2.2所示。

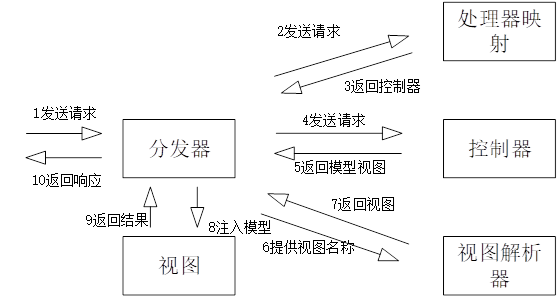


图2.2 Spring MVC工作流程图

由图2.2可以看出，客户发送请求后，请求首先会被Spring MVC的前端请求分发器（Dispatcher Servlet）拦截，所有符合配置中URL样式的访问请求都会由分发器通过查询处理器映射（Handler Mapping）查找相应的控制器（Controller）。分发器在查找到合适的控制器后，会将请求转交给该控制器处理。控制器则进一步调用相应服务类来处理业务逻辑，请求处理完成之后，控制器需返回处理后的结果数据模型以及结果对应的视图名（ModelAndView）至分发器。分发器会通过解析ModelAndView中的视图名，从视图解析器中查找相应视图。分发器获取视图后，将ModelAndView中的返回结果的数据注入到视图中，视图层将二者解析渲染之后再将生成的静态资源页面返回给分发器，由分发器再返回给客户端。Spring MVC的视图解析器支持多视图技术的渲染，包括JSP、HTML、JavaScript等技术的支持[李峰等，2010]。

从Spring MVC的工作流程就可以看出整个流程的核心部分，一是使用哪个控制器来处理请求，由处理器映射来决定；二是最终返回给客户端的结果应该如何渲染，由视图解析器决定。这两者之间的不关联保证了请求处理和结果展示之间的松耦合，更利于程序的开发和维护。使用Spring MVC框架开发过程中需要对一些配置进行管理，如分发器的配置，处理器映射的配置，视图解析器的配置以及控制器的配置等[胡启敏等，2008]。Spring MVC还提供了注解来实现对这些配置的管理，如@Controller注解可以将Java类定义为处理请求的控制器；@RequestMapping注解可以在Controller中的方法与请求URL之间建立映射关系等。这些注解的使用减少了系统文件的数量，大大提升了开发效率。

## 2.2 Vue.js

Vue.js是一套用于构建用户界面的前端开源框架，其发起者是阿里技术顾问尤雨溪。Vue.js是一个可组件化的MVVM库，其结构轻巧、性能高，同时拥有非常容易上手的API。Vue.js是一套构建用户界面的渐进式框架，采用的设计是自底向上增量开发。Vue.js的核心库只关注视图层，且非常容易学习，非常容易与其它库或已有项目整合。另一方面，Vue.js能够驱动采用单文件的组件，可以和 Vue.js生态系统支持的工具开发包一起开发拥有更多功能的单页应用[旷志光等, 2017]。简而言之：Vue.js是一个构建数据驱动的 Web 界面的渐进式框架。

Vue.js 的目标是帮助使用者调用尽可能简单的接口去完成数据与视图的双向绑定，它的核心是一个响应的数据绑定系统。Vue.js使用的是声明式渲染，它允许用户使用一个简单的语法，将数据声明式的渲染进DOM。这使得数据和DOM建立了关联，所有东西都变成响应式的，如果数据发生变化，其绑定的页面元素也会随之发生变化。该功能的使用非常简单，但是其实现的背后，开发者做了大量的工作，它的基本思想是采用了设计模式里面的观察者模式，将每个原生数据都改造成 “可观察对象”。在解析Vue的模板时，也就是在观察者的求值过程中，每一个被取值的可观察对象都会将当前的观察者注册为自己的一个订阅者，并成为当前观察者的一个依赖。当一个被依赖的可观察对象被赋值时，它会通知所有订阅自己的观察者重新求值，并触发相应的更新，即观察者对象中关联的DOM重新渲染。上述过程简单的描述就是DOM元素作为一个观察者在时刻注意着其绑定的数据，当数据值发生变化是，使用观察者的机制，去通知DOM元素重新进行渲染，获取新的值。

Vue.js的另一大优点是组件的可重用性。组件（component）是Vue.js体系里面的一个非常重要的概念，它承担着Vue的基本构成，就像砖块至于房屋一样，没有组件，就没有搭建Vue应用的材料，如图2.3每一个界面都可以被认为是由一个组件树构造而成[焦鹏珲, 2018]。组件可以承担页面构建，逻辑计算、事件监听，数据通信等功能。使用v-on可以将事件绑定到某一个函数，使用v-model可以将元素和数据绑定，这些功能组合在一起就形成了组件的强大功能。但是在Vue里面组件最重要的作用还是在于复用，正如同有了一种砖头之后，就可以把它用到许多地方。当构建好一个组件之后，如果在不同的页面也需要使用它，可以通过import语法，直接导入，方便快捷。因此，外界的第三方组件库也能够非常容易的被使用， Element\_UI就是Vue里面使用最大的一个第三方UI库，本次系统实现中，也使用了里面很多优美漂亮的组件。



图2.3 Vue.js搭建的界面的组件树

Vue.js家族还有许多功能强大的插件，vue-router是view里面用来进行路由跳转控制的插件，vuex是用来进行全局变量操作的插件，axios是用来做数据通信的插件，一些其他的第三方插件，也可以使用Vue.use全局引入到Vue的生态体系中来，因此Vue的可扩展性也是非常强大的。

简而言之，它的优点是：数据驱动、组件化、轻量、简洁、高效、快速、模块友好[刘翔宇, 2018]。

## 2.3 TF-IDF

TF-IDF（term frequency–inverse document frequency）是一种用于信息检索与数据挖掘的常用加权技术。TF意思是词频(Term Frequency)，IDF意思是逆文本频率 (Inverse Document Frequency)，如果有词w与文档d，那么词w在文档d中的词频就是词w在文档d中出现的频率，而逆文档频率表示词w的类别区分能力，包含词w的文档数目越少，则该词的IDF值就越大。TF和IDF的计算公式分别如公式（2.1）和公式（2.2）所示。

（2.1）

（2.2）

其中 表示词w出现在文档d中的次数， 表示语料库中包含词w的文档数目，N表示语料库中全部的文档数量。词条w的TF-IDF权重为：。可以看出，词条w的权重与它在文档中出现的次数成正比，但同时与它在语料库中出现的频率成反比。其中词频代表该词出现在文档中的次数，反映了该词对某个文档的重要程度。比如词W在文档中出现的次数越多，就代表该词对文档的类别贡献越大, 那么这个词越重要。逆文档频率则与包含某个词的文档数目的倒数有关，如果包含某词的文档数目越多, 则该词更有可能是常用词。显然常用词对文档类别的贡献比较小，它的重要程度也较低。

## 2.4 Word2vec

词嵌入（Word Embedding）或者分布式向量（Distributional Vectors）[Cui et al., 2017]是将自然语言的文本描述转换为计算机可以理解的向量或矩阵形式的一种技术。由于要考虑诸多因素例如词的语义（同义词近义词）、词之间的关系（上下文）和向量的维度（处理复杂度）等，我们都希望近义词或者描述同类事物的词语之间的向量空间距离更近，因为只有获得理想的文本表示形式，才能更好进行翻译、问答、信息抽取等工作。Word Embedding是一个词的低维向量表示（一般用的维度可以是几十到几千），它的一个最基本假设就是出现在相同上下文(context)的词意思应该相近，学习Word Embedding的方法就是在用数学的方法建模词和context之间的关系，本系统使用词向量作为要素抽取模块文本特征表示的一部分，法律词向量的引用也可以使用外部的语料库[Chalkidis et **al.**, 2019]以加强效果。

word2vec是Word Embedding的一种实现方式，它的一种模型结构如图2.4所示，它相较于其它算法比如神经网络算法来说，其计算速度要快的多，原因在于他的一种实现采用层级Softmax（Hierarchical Softmax）去加速了计算过程。层级Softmax的基本思想是对语料库里的所有词语根据其出现频率去构建霍夫曼树，所有的词语最终会成为霍夫曼树的叶子节点，霍夫曼树的构建结果是词频较大的节点会成为深度较浅的叶子节点。



图2.4 word2vec的模型原理图

以模型图2.4为例，我们在利用一个词w与2c个上下文Context(w)去训练时，需要构建从2c个上下文Context(w)到目标词w叶子节点路径的概率值，每次路径经过一次中间节点就是一次逻辑回归分类，也就会增加许多参数。显然，词频较大的词会更多涉及到它的计算，因此词频较大的词越靠近根节点，最终的优化目标里的参数就会越少，这是使用霍夫曼树构建层级Softmax加速的原理。

## 2.5 xgboost

xgboost的全称是eXtreme Gradient Boosting[Chen et **al.**, 2016]，即极端梯度提升树。xgboost属于梯度提升机器算法（Gradient Boosting Machine）的扩展与实现。xgboost模型源于Boosting分类器，Boosting分类器是集成学习的一种，其核心思想是把多个分类准确率较低的树模型组合，利用某些投票或者加权的规则，得到一个准确率较高的模型。该模型通过不断迭代，生成一颗颗新的树，模型的核心是如何在每一步生成合理的树。Gradient Boosting Machine算法采用梯度下降的思想去生成计算过程中的每一棵树，它以上一步生成的所有树为基础，向着最小化目标函数的优化方向进行。使用Gradient Boosting Machine算法需要设置合理的参数，并且需要生成足够数量的基础树才能获得较为准确的预测效果，在数据集很庞大的时候，Gradient Boosting Machine算法的需要做大量的计算，效率并不高。xgboost是 Gradient Boosting Machine的一种高效实现，它能够利用CPU的多线程进行并行运算，另一方面，算法本身也进行了改进以提高预测效果。xgboost的基学习器有两种，树（gb-tree）和线性分类器（gb-linear），损失函数利用了二阶泰勒展开去计算，准确度高、可以减少过拟合、扩展性强，它也能分布式处理高维稀疏特征。xgboost在同等条件下，能够拥有比同类算法快10倍以上的性能。

xgboost的本质是GBDT，它的参数训练过程可以由如下过程来表示，公式（2.3）表示模型的构成，它由多个分类器组合而成。

(2.3)

其中表示待分类的一个样本，表示在集成了t个分类器之后对该样本的预测值，则表示第t个分类器，所要训练的参数就是这里面的参数。

公式（2.4）表示第t步的优化目标函数，共有n个样本，是样本i的实际值，损失函数为。

(2.4)

对上述函数进行二阶泰勒展开，可以得到公式（2.5）的结果。

（2.5）

整个优化过程采用贪心策略，所以是已知的，因此可以将目标函数写成公式（2.6），其中，，它们也是已知的。

（2.6）

因此该目标函数只剩下了第t个分类器的参数。

## 2.6 本章小结

本章介绍了裁判文书的文本信息抽取与分类统计系统所采用的一系列技术，包括了Spring MVC、Vue.js、Word2vec、TF-IDF以及xgboost模型。

# 裁判文书的文本信息抽取与分类统计系统的需求分析与设计

本章的主要内容是系统需求分析与设计，首先对项目进行总体上的规划，然后利用用例图对系统用户场景进行分析，分别分析了系统功能性需求和非功能性需求。分析完需求后，给出系统的总体设计，最后描述各个功能模块的详细设计。

## 3.1项目总体规划

裁判文书记载人民法院审理的过程和结果，它是诉讼活动结果的载体[傅郁林, 2000]。法院经常需要针对不同主题或者案由的案件做专题分类统计，相关分类统计项没有记录在审判流程系统的关系数据库中，因此统计的难度非常大，相关统计的数据源只能从裁判文书的文本中获取，所以必须针对裁判文书的文本数据开展深层次的挖掘分析，系统的总体结构如图3.1所示。



图3.1 系统的总体结构图

本系统提供裁判文书文本信息抽取的功能，将抽取的信息按照类型划分成通用信息和要素信息，分别设计了通用型文本信息抽取模块和要素式文本信息抽取模块去抽取裁判文书的深层次信息。本文针对中国司法大数据研究院专题统计报告的统计项，设计了文书分类统计模块，利用信息抽取的结果设计了各统计项分布的计算方法。本文还设计了文书管理模块，用以在上述处理流程中管理裁判文书及其所形成的过渡文件。

## 3.2 系统需求分析

### 3.2.1 系统用例图

系统用例图的作用是为了能够帮助大家更好地理解裁判文书的文本信息抽取与分类统计系统的功能，同时也能够确定系统的各个模块所要承担的职责。本文的裁判文书的文本信息抽取与分类统计系统的系统用例图如图3.2所示，该系统在要素式抽取、通用型抽取和分类统计的过程中，生成了大量有价值的信息，这些信息可以给开发人员和法院统计人员的工作提供大量的帮助。



图3.2 系统用例图

### 3.2.2 系统的功能需求

裁判文书的文本信息抽取与分类统计系统主要包括通用型文本信息抽取、要素式文本信息抽取、文书分类统计、文书管理这四个模块，这四个模块的协同工作，为用户提供了对文书的信息抽取和分类统计功能，为后续的应用奠定了基础，这四个功能模块的主要功能需求如下。

1）通用型文本信息抽取模块：该模块的主要目的是将裁判文书中的结构信息整合抽取，将非结构化的文书转成半结构化或结构化信息，并存储进入XML文档里面。该模块有两个主要功能，裁判文书段落划分功能和段落详细信息抽取功能。裁判文书段落划分功能是指，由于裁判文书有比较规范的写作结构和特征词，基于一些重要特征词，可以将某篇文书划分为6个大段落，分别是：文首、诉讼参与人、诉讼记录、案件基本情况、裁判分析过程、文尾。段落详细信息抽取功能是指从段落中去抽取更详细的信息，包括：

（1）在文首部分抽取：文书种类、经办法院、案号等信息；

（2）在诉讼参与人部分抽取：原告、被告、代理人等相关信息；

（3）在诉讼记录部分抽取：立案日期、审判组织、开庭审理信息、开庭日期、出庭当事人信息等；

（4）在案件基本情况部分抽取：民事案件的案件基本情况包括：原告诉称段、被告辩称段、原告证据、被告证据、质证结论、查明事实段。由于刑事案件的文书结构与此有着较大的区别，所以无法使用此模板，需在分析前判断案件性质再作处理；

（5）裁判分析过程部分抽取：审理结论、引用法条、判决结果；

（6）文尾部分抽取：裁判时间、审判组织成员。

2）要素式文本信息抽取：该模块的主要目的是，利用某类案由的要素表，将该类案由的裁判文书要素项抽取出来，并存储进XML文档里面。本系统结合法院提供的离婚纠纷案由要素表对该案由的裁判文书进行了要素抽取。这里将案由要素分成两类情况去抽取，一类是使用规则匹配抽取，另一类使用监督学习的方法抽取。该模块有三个主要功能，通用信息结果加载功能，规则匹配信息抽取功能和监督学习信息抽取功能。通用信息结果加载功能是指，加载通用型文本信息抽取模块输出的XML结果，并根据文档节点属性读入到系统中来，用于后续抽取工作。规则匹配信息抽取功能是指，使用手工建立规则匹配的方式，对相关要素项进行信息抽取。监督学习信息抽取功能是指，转化信息抽取为文本分类，使用监督学习训练的模型，对相关要素项进行要素抽取，由于监督学习模型是建立在Python语言上的，而程序主体在Java上，所以要设计跨语言模型调用的结构。

3）文书分类统计：该模块的主要目的是对裁判文书按照各种要素信息进行分类统计，共有两个功能，一是分类统计记录功能，二是网络平台发布功能。根据中国司法大数据研究院发布的关于离婚纠纷报告里面的统计项，本文将统计离婚纠纷案件裁判文书的以下相关信息：（1）根据案件审结情况统计各年的案件量；（2）根据当事人特征统计原告性别分布、原被告年龄差分布；（3）根据案件特征去统计婚姻存续时间、离婚态度、判决结果、家暴案件省份分布。本文将依据研究院报告里提供的统计项，利用之前抽取的裁判文书的通用信息和要素信息，计算裁判文书的各统计项分布情况。系统还需要将统计结果可视化后发布到网络平台，同时也提供数据请求接口给开发者，方便以后将统计结果用做别的用途。

4）文书管理：该模块的目的是管理系统里的裁判文书文件以及信息抽取中的生成文件，使抽取过程能够在意外中断后也能接着之前的进度顺利进行。

### 3.2.3 系统的非功能需求

1) 可用性(Usability)需求上，要求系统满足基本功能，且要保证用户使用时操作过程简单、清晰，不需要花费过多的学习成本。

2) 可靠性(Reliability)需求上，要求系统不会因为用户的误操作而崩溃，同时要求系统崩溃后关键的信息不会丢失，且可以迅速恢复系统的正常使用。

3) 可扩展性(Extensibility)需求上，要求系统支持开发或运维人员对功能模块进行扩展。

## 3.3系统总体设计

### 3.3.1 系统的逻辑视图

如图3.3所示是裁判文书的文本信息抽取与分类统计系统的逻辑视图，系统利用通用型文本信息抽取模块和要素式文本信息抽取模块挖掘文书信息，其中要素抽取在逻辑上虽只有一块，但实际由Java和Python两部分组成，在开发视图有展开。文书分类统计模块则利用信息抽取的结果，计算统计项的分布情况，并提供展示统计分布的界面。



图3.3 系统的逻辑视图

### 3.3.2 系统的开发视图



图3.4 系统的开发视图

开发视图如图3.4所示，本系统主要由通用型文本信息抽取、要素式文本信息抽取、文书分类统计、文书管理等四个模块组成。本系统也是采用分层的设计思想，通用型文本信息抽取模块和要素式文本信息抽取模块作为信息抽取层，其中要素抽取模块的Python部分是之前所说的监督学习抽取方法，而Java部分则是规则匹配抽取方法与其他相关功能。文书分类统计模块作为信息统计层，由Vue.js构建的服务器作为信息展示层，去获取信息统计层的数据并渲染。文书管理模块不是独立的一层，但可以为各层模块提供文书管理服务。

### 3.3.3 系统的进程视图



图3.5 系统的进程视图

图3.5是裁判文书的文本信息抽取与分类统计系统的进程视图，系统主要进程还是来源于信息统计层和展示层，信息抽取层一般执行文书信息抽取之后就完成了作用。只有分类统计模块需要维持一个Web界面，用户可以通过各自的浏览器访问分类统计模块提供的统计分布展示界面。

### 3.3.4 系统的物理视图



图3.6 系统的物理视图

图3.6是裁判文书的文本信息抽取与分类统计系统的物理视图，系统的处理程序都可以运行在一个机器上，但是裁判文书集可以放到另一个存储介质里。

### 3.3.5 系统的技术架构



图3.7 系统架构图

裁判文书的文本信息抽取与分类统计系统主要采用的是分层架构的形式，如图3.7所示分为信息抽取层，信息统计层和展示层，其中信息抽取层由通用型文本信息抽取模块和要素式文本信息抽取模块组成，负责解析处理裁判文书。信息统计层由文书统计分类模块构成，其利用信息抽取层的结果，计算统计项的分布数据，然后将数据交由展示层展示给用户。

## 3.4 系统模块设计

### 3.4.1 通用型文本信息抽取模块的设计

根据系统的功能需求小节3.2.2的介绍，通用型文本信息抽取模块需要设计两个功能，裁判文书段落划分功能和段落详细信息抽取功能，以下分别叙述两者的设计。

1. 裁判文书段落划分功能的设计

根据功能需求里的描述，由于裁判文书有比较规范的写作结构和特征词，裁判文书段落划分功能是基于一些重要特征词，将裁判文书划分为六个大段落，分别是：文首、诉讼参与人、诉讼记录、案件基本情况、裁判分析过程、文尾。

文书写作格式来源于最高人民法院下发的《人民法院民事裁判文书制作规范》，规范文件中的部分内容如图3.8所示。

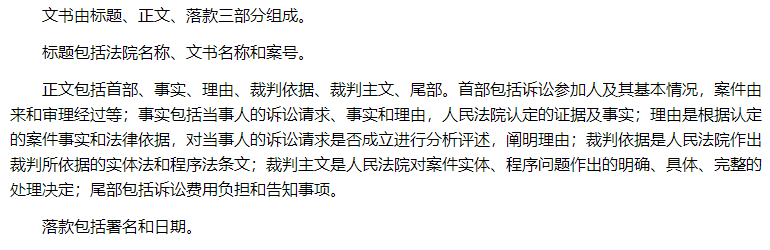


图3.8 《人民法院民事裁判文书制作规范》的部分信息

由图3.8可以看出，中国最高人民法院对民事判决书制定了非常详细的写作规范，在该规范的全文部分，还有更详细的写作要求。

该功能的设计思路是首先将裁判文书读取到内存中去，交由BaseWsAnalyseImpl类进行处理，BaseWsAnalyseImpl会对读入字符串进行预处理，调用各段落枚举类里的段落开头和结尾特征字进行检查，具体的枚举类有WSEnum，SSCYRnum，SSJLEnum，AJJBQKEnum，CPFXGCEnum，WWEnum这六大类，BaseWsAnalyseImpl类会在预处理之后，调用这些枚举类里面记录的特征词，进行裁判文书的段落划分，并最终将段落划分的结果保存为BaseWsAnalyseImpl的属性值，相关特征词会后续实现部分给出具体内容。图3.9是该模块的UML类图。



图3.9 裁判文书段落划分功能的UML图

1. 段落详细信息抽取功能的设计

根据功能需求里的描述，段落信息抽取功能是为了对划分出段落之后的内容进行进一步信息抽取，去获得更为详细的信息，同时由于案件基本情况段落的详细信息抽取会根据民事和刑事案件的差异有所不同，需要在之前作出区分。

该功能的设计思路是由WsHandler类去接收裁判文书段落划分返回的结果，也就是BaseWsAnalyseImpl类里保存的划分后段落信息，然后首先利用BaseWSModelHandler类去解析文首段落信息。先获取案件类型，再决定是否进行进一步信息抽取，如果是民事案件，那么继续调用BaseWSModelHandler，BaseAJJBQKModelHandler，BaseCPFXGCModelHandler， BaseSSCYRMo delHandler， BaseSSJLModelHandler，BaseWWModelHandler等各个段落的处理类，从中获取更加详细的信息。最后将各个段落抽取出来的详细信息保存到分别BaseWSModel，BaseAJJBQKModel，BaseCPFXGCModel， BaseSSCYRModel， BaseSSJLModel，BaseWWModel这些类里，最后利用文书管理模块去保存成为XML文件。

以下是六个段落进行信息进一步抽取的设计过程：

1. 文首段落详细信息抽取设计

一般是指裁判文书里的前三行，包含了经办法院、立案年度、文书类型等。本系统需要抽取出相关信息，文首处理的类图如下图3.10所示，此处交由BaseWSModelHandler类去进一步获取对应的信息。



图3.10 文首详细信息抽取的UML类图

1. 诉讼参与人段落详细信息抽取

诉讼参与人是指参与诉讼的人员信息，一般会从正文第一段开始，通常会有原告、被告、原告代理人、被告代理人等。需要对这些人的姓名、性别、身份证号、居住地等信息进行提取，如果裁判文书中没有提到相应的信息就设置为空。此处交由BaseSSCYRModelHandler类去进一步获取对应的信息，诉讼参与人段落处理的类图如下图3.11所示。



图3.11 诉讼参与人详细信息抽取的UML类图

1. 诉讼记录段落详细信息抽取

诉讼记录一般记录的是本次案件审理的过程，记录着出庭当事人信息，法院立案时间，审理结束时间。此处交由BaseSSJLModelHandler类去进一步获取对应的信息，诉讼记录段落处理的类图如下图3.12所示。



图3.12 诉讼记录详细信息抽取的UML类图

1. 案件基本情况段落详细信息抽取

案件基本情况通常是记载原、被告双方的各种说辞，会包含原告诉称段、被告辩称段等信息。此处交由BaseAJJBQKModelHandler类去进一步获取对应的信息，案件基本情况段落处理的UML类图如下图3.13所示。



图3.13 案件基本情况详细信息抽取的UML类图

1. 裁判分析过程段落详细信息抽取

裁判分析过程会记载着法院法官如何依据事实去评判原、被告双方的说法。此处交由BaseCPFXGCModelHandler类去进一步获取对应的信息，裁判分析过程段落处理的UML类图如下图3.14所示。



图3.14 裁判分析过程详细信息抽取的UML类图

1. 文尾程段落详细信息抽取

文尾通常记录着审判组织成员和审判时间，审判组织成员包含了审判员和书记员，这里抽取时需要区分开。此处由BaseWWModelHandler类去进一步获取对应的信息，文尾段落处理的类图如下图3.15所示。



图3.15 文尾详细信息抽取的UML类图

图3.16是通用型文本信息抽取模块的流程图，在进入段落详细信息抽取之前需要判断是民事还是刑事案件。



图3.16 通用型民事判决文书信息抽取模块流程图

### 3.4.2 要素式文本信息抽取模块的设计

根据系统的功能需求小节3.2.2的介绍，要素式文本信息抽取模块需要设计三个功能：通用信息结果加载功能，规则匹配信息抽取功能和监督学习信息抽取功能。

在仔细研究离婚纠纷案由要素表提供的要素信息基础上，本文归纳总结出要素项的三个特点。

1. 待抽取的要素项是关于时间信息或者文字描述信息；
2. 待抽取的要素项在离婚纠纷案件中出现的数目较少；
3. 待抽取的要素项难以匹配固定句式且抽取结果是二元对立类型的。

这里的二元对立类型就是“是否”问题，其结果只有“是”与“否”这两种。“否”在这里是指裁判文书中未提及相关信息，因为如果存在这种要素中描述的情形，法院在制作判决书的时候一般都会提及该要素。当然对于时间信息和文字描述信息如果没有匹配到，仍然使用“未提及”作为节点描述。

本文对于要素项满足第1、2两点的采用规则匹配的方法去抽取，对于那些满足第3点且不满足第1、2两点的要素项采用监督学习的方法去匹抽取信息。其原因是时间信息比较容易使用正则表达式匹配，方便迅速，而文字描述信息则通常只需将事件描述的句子提取出来，可根据关键字识别，所以这两种信息都使用规则匹配方式，另外如果要素出现在判决书中的数目较少，缺少足够多的样本进行监督学习，那也只能用规则匹配方式。对于那些文书经常提及，但抽取规则建立起来比较复杂的要素，使用监督学习去抽取信息就更加合理。

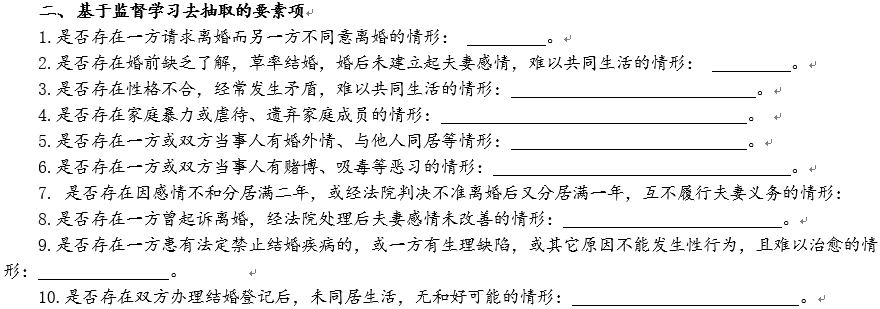


图3.17 基于监督学习去抽取的要素项

根据上述原则，本文将要素项划分成两类，图3.17是使用监督学习去抽取信息的要素项。而图3.18是使用规则匹配去抽取信息的要素项，共有34项要素，这里只截取部分展示。

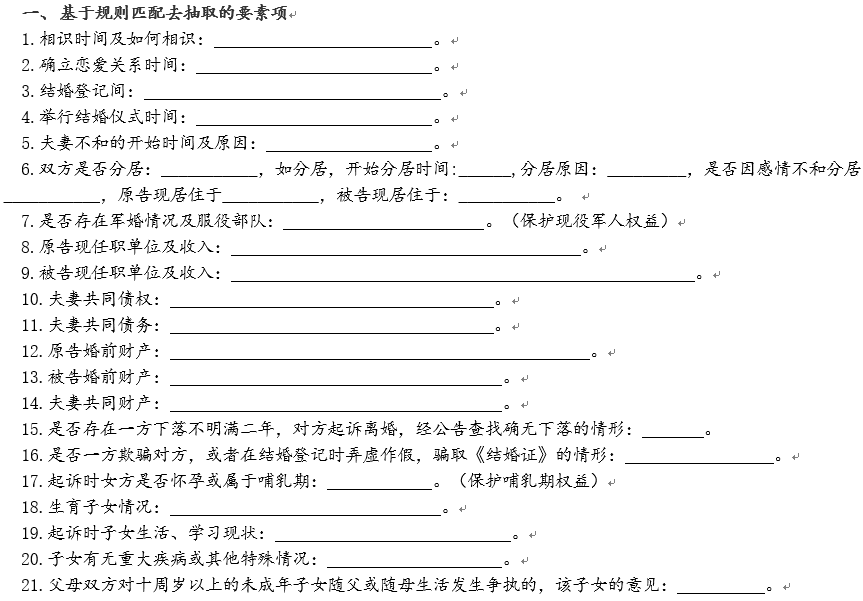


图3.18 基于规则匹配去抽取的要素项（部分）

（1） 通用信息结果加载功能的设计

根据功能需求里的描述，通用信息结果加载功能是指，加载通用型文本信息抽取模块输出的XML结果，并根据文档节点属性读入到系统中来。



图3.19 通用信息结果加载功能的UML类图

该功能的设计思路是调用LoadWSXML工具类，返回读取XML的实例，利用实例获取该XML文件里各节点的属性值，相关节点名称会在实现部分给出。图3.19是通用信息结果加载功能的UML类图。

（2） 规则匹配信息抽取功能的设计

根据功能需求里的描述，规则匹配信息抽取功能是指，使用手工建立规则匹配的方式，对相关要素项进行信息抽取。

对于利用规则匹配去进行要素抽取的那34条要素项，使用ZZElementExtra类去负责。对于每一条要素项的抽取，都要设计一个专门的方法去处理，本文按照上述编号去设计方法名，依次是handleRow1，handleRow2等，每个要素抽取方法会首先调用LoadWSXML类里面的各个节点获取方法，获取通用型文本信息抽取的结果。获取节点信息后，进一步根据要素项的抽取要求设计正则表达式或者字符串匹配，去获取要素信息。图3.20是基于规则匹配进行要素信息抽取的UML类图。



图3.20 基于规则匹配进行要素信息抽取的UML类图

（3） 监督学习信息抽取功能的设计

根据功能需求里的描述，监督学习信息抽取功能是指，转化信息抽取为文本分类，使用监督学习训练的模型，对相关要素项进行要素抽取。

对于利用监督学习去对那些要素项进行要素抽取，本文将分为训练模型和使用模型两个方面来叙述设计思路，不过在此之前，需描述此处的一个结构设计。

在3.2.2小节有提到，系统的监督学习的算法模型是采用了Python语言去训练和调用的，而程序主体部分依然是使用Java语言，所以需要使用Java去调用Python程序训练好的模型以获取结果。实现上述功能有两种比较可行的方式，一是使用Jython工具开发包帮助Java调用Python程序。二是借助socket编程，利用Python实现一个socket服务器，然后在java程序中使用socket请求服务器并调用Python训练好的模型，返回结果。由于第一种方法的程序耦合太高，互相调用的性能受限，一旦程序有修改，变动太大，所以本文采用基于socket的第二种方式来实现模型跨语言调用。

训练模型方面，即利用标注好的数据去训练模型，该类要素项的信息抽取问题被转换成短文本二分类问题，即输入段落节点文本，输出该要素项对应的“是”与“否”中的一个标签。它的合理性在于那些文书的要素更多的是某些语句有与无的区别，而不涉及复杂的语意，类似于主题分类。

对于该类中的每一个要素项，都需要训练一个二分类模型，所以要求模型能够快速训练。本文采用的监督学习模型是xgboost模型，xgboost模型被广泛应用到数据分析竞赛中，并取得了非常优秀的成绩，其体量小，速度快，效果好，被很多企业用做数据分析算法。由于训练输入的是裁判文书信息抽取后的XML文件，所以首先要提取出相应节点的文本内容，然后对文本进行预处理提取特征。对于文本进行预处理有着如下几个步骤：分词，去停用词，特征选择。经过预处理之后的文本变成了计算机可以理解的向量表达形式。之后将文本向量结合该文本的标注类别，转成xgboost算法所需要的文件输入格式。训练完成之后，保存模型以待后续调用。

特征选择是文本预处理的一项重要工作，针对文本分类的研究大多基于传统的向量空间模型（Vector Space Model，VSM），该模型在处理长文本的时候有着比较好的效果，但短文本有着稀疏性的特点，会降低分类准确率。针对本短文本分类的特点，本文结合了TF-IDF和word2vec作为文本的特征向量[Branting, 2017]，以下介绍特征向量的设计。

设有训练语料词库vocab和某篇裁判文书，w是文书中的词语，N是所有词的数目。

（3.1）

使用 Word2vec 模型训练语料，得到文本中单词词向量，将文档中词向量累加得到文本的向量表示。其中，表示词汇t的 word2vec词向量。

（3.2）

接下来引入TF-IDF模型，根据词汇重要程度计算word2vec模型中词汇权重，将加权过的词向量累加得到文档新的向量表示。

（3.3）

最后，合并加权word2vec和TF-IDF两种模型，得到新的文档向量表示 ，其中 表示文档的tfidf向量表示，即文档分词之后的各词的权重向量。

（3.4）

使用这种方法来提取文档的特征向量是因为，word2vec体现的是词与词之间的联系，可以提供一些先验信息[张谦等, 2017]，而TF-IDF则体现了文档本身的关键词特征信息，是文档本身特性的一种体现，两种结合有助于更好的获取文档的向量表示，在实验部分有关于该特征选择对于最终效果的影响情况。

使用模型方面，即利用训练好的模型去抽取裁判文书的要素项，本文将设计一个JDElementExtra类去负责，针对每一条要素项设计一个专门的方法，然后调用LoadWSXML类的方法获取所需要的XML节点中的文本，之后向Python服务器传递参数并请求模型调用，由Python程序载入保存的模型并输入传递来的文本，最后由模型返回标签，即需要抽取的要素信息。本文模型在第一次载入后就保存在内存中，后续调用请求时无需继续载入。

图3.21是模型训练和运行的一个流程图，分为word2vec训练，xgboost训练和xgboost预测，word2vec训练会得到word2vec模型，并参与后续特征计算。xgboost训练使用训练数据得到模型，并使用验证数据测试模型好坏，最后返回一个较好的模型。xgboost预测使用训练好的模型对输入数据进行预测。



图3.21 模型训练和运行流程图

图3.22是基于监督学习进行要素信息抽取的UML类图，类图中既有Java类也有Python类，其中以ws\_classification类与JDElement类之间为划分点，上侧的是Python类，下侧是Java类，Python类里有负责文书预处理的ws\_preprocessing类和进行分类预测的ws\_classification类，而JDElement类则是Java中承担主要任务的类，负责连接Python进程，并提供待处理字符串给Python进程。

综合上述两种抽取方式，可以更有效的抽取出离婚纠纷案由的要素信息。



图3.22 基于监督学习进行要素信息抽取的UML类图

要素式文本信息抽取的总的流程图如图3.23所示，当一篇裁判文书进入流程之后，我们要遍历进行各条要素抽取的抽取方法，通过解析XML文件的节点获取对应的字符串，然后根据要素的特点，选择对应的方法，利用规则匹配或者监督学习方法去进行要素信息抽取，最后将抽取出来的要素信息存入到相应节点并保存。



图3.23 要素式文本信息抽取的流程图

### 3.4.3 文书分类统计模块的设计

根据系统的功能需求小节3.2.2的介绍，文书分类统计模块的主要目的是对裁判文书按照各种要素信息进行分类统计。根据中国司法大数据研究院发布的关于离婚纠纷报告里面的统计项，本文将统计离婚纠纷案件裁判文书的以下相关信息：1)根据案件审结情况统计各年的案件量；2)根据当事人特征统计原告性别分布、原被告年龄差分布；3)根据案件特征去统计婚姻存续时间、离婚态度、判决结果、家暴案件省份分布。依据研究院报告里提供的7个统计项，利用之前抽取的裁判文书的通用型信息和要素式信息，计算裁判文书的各统计项分布情况并提供可视化的网络平台发布。

该模块的设计需要提供这七个统计项分布的计算方法，设计为如下七个方法：calcGnajlDis、calcYgxbDis、calcJbajsfDis、calcPjjgDis、calcLhtdDis、calcYbgnlcDis、calcHycxsjDis，他们由ClassStatisticService类去管理。根据不同的统计项的分布情况，这七个计算方法需要各自维护一个Map数据类型，用以存储各自统计出来的分布值。系统需要将统计出来的分布数据保存到硬盘，以后新添加裁判文书，可以读取该文件，进行增量式的统计。本文还设计一个Web界面用以发布文书分类统计结果，利用可视化工具Echarts提供的组件去渲染数据。这里Web系统采用Spring MVC 作为后端，使用ClassStatisticController去管理分发。本文参考了以往专题报告的形式，将各年的案件量和家暴案件省份这两个分布用柱状图来展示，将原告性别分布、判决结果和离婚态度这三个分布用饼状图来展示，将原被告年龄差和婚姻存续时间这个两个分布用平滑曲线图来展示，实现部分利用了Vue的组件重用技术，提高了前端开发效率，文书分类统计模块的类图如图3.24所示。



图3.24 文书分类统计模块的UML图

### 3.4.4 文书管理模块的设计

根据系统的功能需求小节3.2.2的介绍，文书管理模块的目的是管理系统里的裁判文书文件以及信息抽取中的生成文件，使抽取过程能够在意外中断后也能继续接着之前的进度顺利进行。

该功能的设计思路是使用不同的文件夹来标记裁判文书位于处理流程中的状态。根据之前模块的设计，裁判文书源文件需要经过通用型文本信息抽取过程，形成相关的XML文件保存下来，然后再经过要素式文本信息抽取，将之前的XML文件进一步增加要素信息并存储，之后再经过文书分类统计模块的计算处理，所以需要构建五个文件夹来存储不同状态的文件，一是存储新增文书的文件夹addDir，系统启动时，信息抽取首先会检查该文件夹下是否有新增文书；二是存储通用型文本信息抽取之后所形成XML文件的文件夹halfcompleteDir；三是存储要素式文本信息抽取之后所形成XML文件的文件夹completeDir；四是用来存储经过分类统计模块处理之后裁判文书的文件夹statisticFinishDir；五是存储最初源文件的文件夹sourceDir。

综合上述过程，需要设计一个WSManagerService类，定义一系列文件操作方法，用来处理不同情况下各模块的调用请求，其文件夹地址统一由Constant类里的静态变量存储。

## 3.5 本章小结

本章主要对系统进行了需求进行了分析，通过实际业务逻辑需要给出了系统的功能需求。然后本章对通用型文本信息抽取模块、要素式文本信息抽取模块、文书分类统计模块、文书管理模块的功能进行了细分和具体描述。之后，对整个系统各个模块的工作流程进行了设计，描述了各个模块之间的交互关系。

# 裁判文书的文本信息抽取与分类统计系统的实现

根据上一章对裁判文书的文本信息抽取与分类统计系统的需求分析和设计，本章主要对通用型文本信息抽取模块、要素式文本信息抽取模块、文书分类统计模块以及文书管理模块进行展开，重点描述各个模块的具体实现细节。

## 4.1 通用型文本信息抽取模块的实现

### 4.1.1 裁判文书段落划分功能的实现

根据通用型文本信息抽取模块中裁判文书段落划分功能的设计，需要根据特征词，将某篇文书划分为六个大段落，分别是：文首、诉讼参与人、诉讼记录、案件基本情况、裁判分析过程、文尾。

该功能首先要解析裁判文书的Word文件得到字符串，使用了Apache的POI工具开发包来解析，Apache POI可以提供Java API给用户，通过Java就可以读取和创建MS Word。封装的工具类是POIUtil，使用POIUtil里面的getWqcGString方法去读取Word文件，并将返回的字符串交给BaseWsAnalyseImpl进行处理。BaseWsAnalyseImpl类首先使用removeKG方法去除空格、换行符等。然后读取枚举类里各个段落的开始和结束的特征词进行段落划分。由于有六个段落，所以共有WSEnum，SSCYREnum，SSJLEnum，AJJBQKEnum，CPFXGCEnum，WWEnum六个枚举类，其记载的特征词如表4.1所示。

表4.1 裁判文书各段落划分特征词统计表

|  |  |  |
| --- | --- | --- |
| 枚举类名 | 特征词位置 | 特征词内容 |
| WSEnum | 开始 | 无 |
| 结束 | 数字加“号”字构成的案号 |
| SSCYREnum | 开始 | 原告、上诉人、原告代理人、原告律师等词 |
| 结束 | 诉讼记录开始的特征词 |
| SSJLEnum | 开始 | “一案”等词 |
| 结束 | 审理结束、审理终结、审理完成、  以及案件基本情况的开始特征词 |
| AJJBQKEnum | 开始 | “诉称”、“辩称”等词 |
| 结束 | 裁判分析过程的开始特征词 |
| CPFXGCEnum | 开始 | “本院认为”、“本院经审理认为”、“本院经审查认为”、“主持调解”、“本院审查认为”等词 |
| 结束 | “裁定如下”、“判决如下”、“决定如下”等词 |
| WWEnum | 开始 | “审判长”、“审判员”、“代理审判员”、“书记员”、“速录员”、“人民陪审员”等词 |
| 结束 | 无 |

根据枚举类中的特征词划分完段落之后，将段落内容保存到BaseWsAnalyseImpl的属性值中。

裁判文书段落划分功能主要包含的类如表4.2所示。

表4.2 裁判文书段落划分功能主要类统计表

|  |  |
| --- | --- |
| 分类 | 主要包含的类 |
| 值对象 | WSEnum.java、SSCYREnum.java  SSJLEnum.java、AJJBQKEnum.java  CPFXGCEnum.java、WWEnum.java |
| 业务类 | BaseWsAnalyseImpl.java |
| 工具类 | POIUtil.java |

读取裁判文书Word文件的函数如下图4.1所示。

|  |
| --- |
| public static String getWqcGString(byte[] wsnr,String title) throws Exception{  String suffix=getSuffix(title).toLowerCase();  InputStream is=new ByteArrayInputStream(wsnr);  WsParser parser = null;  String temp = new String(wsnr);  try {  if(StringUtil.equals(temp.substring(0, 5), "{\\rtf") && !StringUtil.equals("rtf", suffix)){  parser=new RtfWsParser(is); }  ……//不同后缀的处理  is.close();  if(parser==null){  return null;}  return parser.getContent();  ……//异常处理  return null; } |

图4.1 裁判文书段落划分功能的部分代码

### 4.1.2 段落详细信息抽取功能的实现

根据通用型文本信息抽取模块中段落详细信息抽取功能的设计，需要从段落中去抽取更细致的信息。

裁判文书段落划分功能最后会将段落划分的结果保存在BaseWsAnalys eImpl类的属性值里面，根据设计要求，首先要解析文书的文首信息去判断案件类型。文首是每一篇裁判文书都有的一个重要信息，解析文首功能使用的是BaseWSModelHandler这个类中的jxWswsModel方法去实现，使用dealFaYuan方法去处理法院信息，使用dealWenShuXingZhi方法去处理文书性质的信息，使用dealAnHao方法去处理案号信息。最后，将处理得到的结果保存在BaseWSModel类的对象实例之中。接着从BaseWSModel类中获取案件基本性质，是否为民事案件，如果是民事案件，则利用WSHandler中dealBase方法去接着依次调用BaseAJJBQKModelHandler，BaseCPFXGCM odelHandler， BaseSSCYRModelHandler， BaseSSJLModelHandler，BaseW WModelHandler中的方法去分别解析各个段落详细信息，将详细信息分别保存在BaseAJJBQKModel，BaseCPFXGCModel， BaseSSCYRModel， BaseSSJLModel，BaseWWModel等值对象中。解析完成后，再调用文书管理模块里的WsManagerService去保存解析出来的信息。

以下是裁判文书各段落信息抽取的详细实现：

1）文首段落详细信息抽取

根据文首段落详细信息抽取部分的设计，该功能由BaseWSModelHandler类来处理，并返回保存了文首信息的BaseWSModel类的对象实例，某篇裁判文书的文首如下图4.2所示。

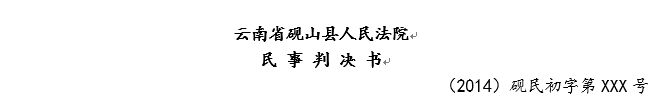


图4.2 文首段落示例

BaseWSModelHandler类有三个方法去解析文首的相关信息，分别是dealFaYuan方法、dealWenShuXingZhi方法、dealAnHao方法。首先利用三个特征词 “法院”、“书”、“号”去匹配出三个句子，然后根据“法院”匹配出来的句子，使用dealFaYuan方法去获取作出判决的法院以及省市信息；根据“书”字匹配出来的句子，使用dealWenShuXingZhi方法去获取文书性质的信息。根据“号”字去匹配出来句子，使用dealAnHao方法去获取案号的信息。表4.3是文首部分抽取出来的相关信息表，存储在BaseWSModel实例中。

表4.3 文首段落抽取的信息表

|  |  |
| --- | --- |
| 字段 | 描述 |
| Jbfy | 经办法院 |
| Wsmc | 文书名称 |
| ah | 案号 |
| land | 立案年度 |
| wszzdw | 文书制作单位 |
| fyjb | 法院级别 |
| xzqhProv | 行政区划（省） |
| xzqhCity | 行政区划（城市） |
| wszl | 文书种类 |
| ajxz | 案件性质 |
| spcx | 审判程序 |
| ajlx | 案件类型 |

段落详细信息抽取功能主要包含的类如表4.4所示。

表4.4 文首详细信息抽取的主要类统计表

|  |  |
| --- | --- |
| 分类 | 主要包含的类 |
| 值对象 | BaseWSModel.java |
| 业务类 | BaseWsAnalyseImpl.java  BaseWSModelHandler.java |

图4.3是文首详细信息抽取中解析文书性质的部分代码。

|  |
| --- |
| public void dealWenShuXingZhi(String wsName,BaseWSModel baseWSModel) {  baseWSModel.setWsmc(wsName);  String ajlx = null;  if (wsName.length() > 3) {ajlx = wsName.substring(0, 2); }  baseWSModel.setAjxz(ajlx + "案件");  String wszl = null;  int index\_shu = wsName.indexOf("书");  if (index\_shu != -1 && index\_shu - 2 < index\_shu + 1 && index\_shu-2 > 0) {  wszl = wsName.substring(index\_shu - 2, index\_shu + 1); }  baseWSModel.setWszl(wszl); } |

图4.3 文首详细信息抽取的部分代码

2）诉讼参与人段落详细信息抽取

根据诉讼参与人段落详细信息抽取部分的设计，该功能由BaseSSCYRM odelHandler类来处理，并返回保存了诉讼参与人信息的BaseSSCYRModel类的对象实例，某篇裁判文书的诉讼参与人段落如下图4.4所示。

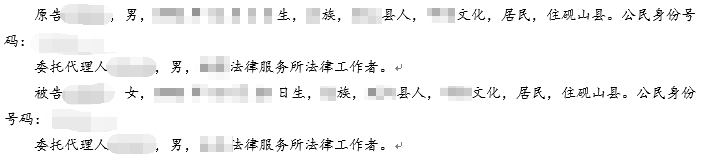


图4.4 诉讼参与人段落示例

BaseSSCYRModelHandler类有三个方法去解析诉讼参与人的相关信息，分别是dealYuanGao方法、dealBeiGao方法、dealDaiLiRen方法。首先利用原告和被告两个关键字去获取描述原被告信息的自然段落，然后将原告与被告之间的自然段落作为描述原告代理律师的段落，将被告之后的自然段落作为描述被告代理律师的段落，需要注意代理律师有时不只有一个人。之后利用dealYuanGao方法抽取原告信息。使用dealBeiGao去抽取被告信息，使用dealDaiLiRen去抽取代理律师的信息。如果不存在相关信息，则默认为空字符串。如此便逐步完成诉讼参与人段落详细信息抽取。表4.5是诉讼参与人段落抽取出来的相关信息表，存储在BaseSSCYRModel.SSCYR实例中，由于有多个诉讼参与人，所以使用内部类SSCRY去存储属性值，然后在BaseSSCYRModel里面声明一个SSCRY构成的列表。

表4.5 诉讼参与人段落的BaseSSCYRModel.SSCYR部分属性值表

|  |  |
| --- | --- |
| 字段 | 描述 |
| sscyrmc | 诉讼参与人名称 |
| sssf | 诉讼身份 |
| ssdw | 诉讼地位 |
| dsrlb | 当事人类别 |
| dsrlx | 当事人类型 |
| xb | 性别 |
| mz | 民族 |
| csrq | 出生年月 |
| gj | 国籍 |

诉讼参与人段落详细信息抽取主要包含的类如表4.6所示。

表4.6 诉讼参与人详细信息抽取的主要类统计表

|  |  |
| --- | --- |
| 分类 | 主要包含的类 |
| 值对象 | BaseSSCYRModel.java |
| 业务类 | BaseWsAnalyseImpl.java  BaseSSCYRModelHandler.java |

图4.5是诉讼参与人详细信息抽取中抽取生日的部分代码。

|  |
| --- |
| for (j = 1; j < contentlist.size(); ++j) {  if ( (contentlist.get(j)).indexOf("年") > 3) {  csrq = (contentlist.get(j)).substring((contentlist.get(j)).indexOf("年") - 4,(contentlis t.get(j)).indexOf("日") + 1); }  String year = null;  String month = null;  String day = null;  if (csrq != null) {  year = csrq.substring(csrq.indexOf("年") - 4, csrq.indexOf("年"));  month = csrq.substring(csrq.indexOf("年") + 1, csrq.indexOf("月"));  day = csrq.substring(csrq.indexOf("月") + 1, csrq.indexOf("日"));  }} |

图4.5 诉讼参与人详细信息抽取的部分代码

3）诉讼记录段落详细信息抽取

根据诉讼记录段落详细信息抽取部分的设计，该功能由BaseSSJL ModelHandler类来处理，并返回保存了诉讼记录信息的BaseSSJLModel类的对象实例，某篇裁判文书的诉讼记录段落如下图4.6所示。

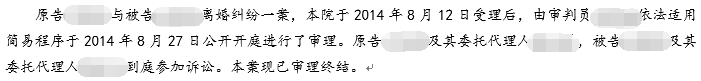


图4.6 诉讼记录段落示例

BaseSSJLModelHandler类匹配案由信息需要依据外部加载的案由代码文件，其余的信息是根据其特征词去匹配。这里所需要抽取的一项出庭当事人信息需要利用一下诉讼参与人抽取的结果。表4.7是诉讼记录段落抽取出来的相关信息表，存储在BaseSSJLModel的实例中。

表4.7 诉讼记录段落的BaseSSJLModel属性值表

|  |  |
| --- | --- |
| 字段 | 描述 |
| ay | 案由 |
| aydm | 案由代码 |
| ktsl | 是否开庭审理 |
| ktslxx | 开庭审理信息 |
| larq | 立案日期 |
| spzz | 审判组织（合议庭） |
| ctdsrxx | 出庭当事人信息(诉讼参与人，诉讼身份） |

诉讼记录段落详细信息抽取主要包含的类如表4.8所示。

表4.8 诉讼记录详细信息抽取的主要类统计表

|  |  |
| --- | --- |
| 分类 | 主要包含的类 |
| 值对象 | BaseSSJLModel.java  BaseSSCYRHandler.java |
| 业务类 | BaseWsAnalyseImpl.java  BaseSSJLModelHandler.java |

图4.7是诉讼记录详细信息抽取中抽取案由的部分代码：

|  |
| --- |
| public String getAYAndDM(List<String> ayList,String ssjl){  String single\_ay;  for(int j = 0; j < ayList.size(); ++j) {  single\_ay = (ayList.get(j)).substring(5, (ayList.get(j)).length());  if (ssjl.contains(single\_ay)) {  return single\_ay+ " "+(ayList.get(j)).substring(0, 4);  }  }  return null;  } |

图4.7 诉讼记录详细信息抽取的部分代码

4）案件基本情况段落详细信息抽取

根据案件基本情况段落详细信息抽取部分的设计，该功能由BaseAJJBQKModelHandler类来处理，并返回保存了案件基本情况信息的BaseAJJBQKModel类的对象实例，某篇裁判文书的案件基本情况段落的写作样例如下图4.8所示。

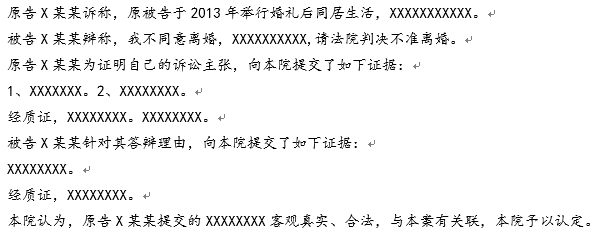


图4.8 案件基本情况段落示例

在案件基本情况段落详细信息抽取的设计中，需要抽取以下内容：原告诉称段、被告辩称段、原告证据、被告证据、质证结论、查明事实段。BaseAJJBQKModelHandler类由“诉称”、“辩称”关键词获取原告诉称段和被告辩称段，可由“证据”配合相关主语得到原告和被告证据段，通过“质证”特征词得到质证结果，通过“予以认定/予以确认”等特征词得到查明事实段，由此获得案件基本情况的详细信息，不过有的文书此段落并没有这么规范的写作格式，对于该情况本文将诉称和辩称之外的案件基本情况段落统一指定为查明事实段。表4.9是案件基本情况段落抽取出来的相关信息表，存储在BaseAJJBQKModel的实例中。

表4.9 案件基本情况段落的BaseAJJBQKModel属性值表

|  |  |
| --- | --- |
| 字段 | 描述 |
| ygscd | 原告诉称段 |
| bgbcd | 被告辩称段 |
| ygzj | 原告证据 |
| bgzj | 被告证据 |
| zyjd | 争议焦点 |
| zzjl | 质证结论 |
| cmssd | 查明事实段 |

案件基本情况段落详细信息抽取主要包含的类如表4.10所示。

表4.10 案件基本情况详细信息抽取的主要类统计表

|  |  |
| --- | --- |
| 分类 | 主要包含的类 |
| 值对象 | BaseAJJBQKModel.java |
| 业务类 | BaseWsAnalyseImpl.java  BaseAJJBQKModelHandler.java |

图4.9是案件基本情况段落详细信息抽取中获取被告辩称段的部分代码。

|  |
| --- |
| public String getBgbcd(List<String> ajjbqk){  for (int i = 1; i < ajjbqk.size(); ++i) {  String dl = ajjbqk.get(i); //存储的是自然段落  List<String> nrtoken = FcUtil.getWholeToken(dl);  int index = nrtoken.size() < 20 ? nrtoken.size() : 20;  for (int j = 0; j < index; ++j) {  String ajnr = dl.length() < 40 ? dl : dl.substring(0, 40);  if (ajnr.contains("辩称")) {  end = i; //找到诉称的结束  …//部分代码段右括号的省略  String ygscd = "";  for (int k = 1; k < end; k++) {  ygscd = ygscd + ajjbqk.get(k); }  } |

图4.9 案件基本情况详细信息抽取的部分代码

5）裁判分析过程段落详细信息抽取

根据裁判分析过程段落详细信息抽取部分的设计，该功能由BaseCPFXGCModelHandler类来处理，并返回保存了裁判分析过程信息的BaseCPFXGCModel类的对象实例，某篇裁判文书的裁判分析过程段落的写作样例如图4.10所示。

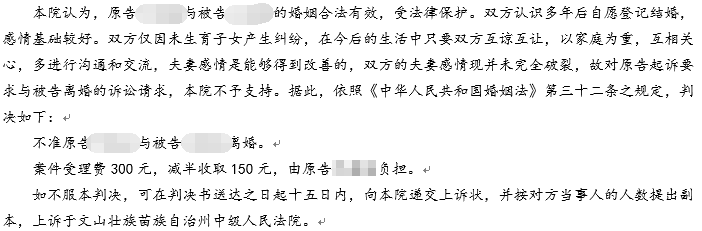


图4.10 裁判分析过程段落示例

在裁判分析过程段落详细信息抽取的设计中，需要抽取以下内容：裁判分析过程部分抽取：审理结论、引用法条、判决结果。为了后续的统计方便，还加入了是否准予离婚这一信息项。BaseCPFXGC ModelHandler类首先匹配引用的法律条文，通过书名号去获取，在第一条法律条文所属的句子前都是审理结论段，将其归入sljl变量里，然后将法律条文一一处理。一般情况，法律条文会有条和款之分，本文对此作了区分。而判决结果段通常跟随在特征词“判决如下”、“裁定如下”、“决定如下”等后面，需要匹配该类关键词以获取详细信息。表4.11是案件基本情况段落抽取出来的相关信息表，存储在BaseCPFXGCModel的实例中，由于通常判决引用的法条是多样的，所以需要使用内部类BaseCPFXGCModel.FT来记录法条的详细信息，而在BaseCPFXGCModel则声明了一个FT类的列表。

表4.11 裁判分析过程段落的BaseCPFXGCModel属性值表

|  |  |
| --- | --- |
| 字段 | 描述 |
| sljl | 审理结论 |
| flft | 法律法条的列表 |
| pjjg | 判决结果 |
| sfzylh | 是否准予离婚 |

裁判分析过程段落详细信息抽取主要包含的类如表4.12所示。

表4.12 裁判分析过程详细信息抽取的主要类统计表

|  |  |
| --- | --- |
| 分类 | 主要包含的类 |
| 值对象 | BaseCPFXGCModel.java |
| 业务类 | BaseWsAnalyseImpl.java  BaseCPFXGCModelHandler.java |

图4.11是裁判分析过程段落抽取法律法条信息的部分代码：

|  |
| --- |
| public void getFlft(String ftString, BaseCPFXGCModel baseCPFXGCModel){  String[] ftfz = ftString.split("《"); for(int j = 0; j < ftfz.length; ++j) {  String content = ftfz[j];  if (content.indexOf("》") != -1) {  BaseCPFXGCModel.FT ft = new BaseCPFXGCModel().new FT();  String flftmc = content.substring(0, content.indexOf("》")); //法条名称  ft.setFtmc(flftmc);  String[] tmString = content.split("条");  HashMap<String, String> ftMap = new HashMap();  for(int i = 0; i < tmString.length - 1; ++i) {  ……//匹配具体的条目和款目}}  ft.setTmkm(ftMap);  baseCPFXGCModel.getFlft().add(ft); }} |

图4.11 裁判分析过程详细信息抽取的部分代码

6）文尾段落详细信息抽取

根据文尾段落详细信息抽取部分的设计，该功能由BaseWWModelHandler类来处理，并返回保存了文尾信息的BaseWWModel类的对象实例，某篇裁判文书的文尾段落的写作样例如下图4.12所示。

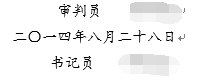


图4.12 文尾段落示例

在文尾段落详细信息抽取的设计中，需要抽取以下内容：文尾部分抽取：裁判时间、审判组织成员。BaseWWModelHandler类需要匹配“审判员”、“书记员”、“审判长”等特征词获取审判组织成员，本文将审判组织成员分为两类，审判人员和书记员，通过HashMap来记录，时间信息则通过年月日进行划分。系统建立一个字符串映射方法，将中文字符串时间映射成阿拉伯数字。表4.13是文尾段落抽取出来的相关信息表，存储在BaseWWModel的实例中。

表4.13 文尾段落的BaseWWModel属性值表

|  |  |
| --- | --- |
| 字段 | 描述 |
| cpsj | 裁判时间 |
| spry | 审判人员 |
| sjy | 书记员 |

文尾段落详细信息抽取主要包含的类如表4.14所示。

表4.14 文尾详细信息抽取的主要类统计表

|  |  |
| --- | --- |
| 分类 | 主要包含的类 |
| 值对象 | BaseWWModel.java |
| 业务类 | BaseWsAnalyseImpl.java  BaseWWModelHandler.java |

图4.13是文尾段落抽取裁判日期信息的部分代码。

|  |
| --- |
| public void getCprq(String content){  if (content.contains("年") && content.contains("月") && content.contains("日")){ //处理裁判日期的部分  boolean hasWsrq = content.length() > 8;  String wsrq = hasWsrq ? DateUtil.convertToCNDate(content) : null;  if (wsrq != null) {  wsrq = DateUtil.convertToCNDate(wsrq);  if (wsrq.indexOf("年") - 4 > -1) {  ……//利用字符串截取方法获取年月日信息并设置  }}}} |

图4.13 文尾详细信息抽取的部分代码

段落详细信息抽取完之后，调用文书管理模块的WSManagerService类将通用型文本信息抽取模块中抽取出来的信息保存为XML文件，并存入halfcompletedir的文件夹，该模块生成的XML文件内容如图4.14所示。

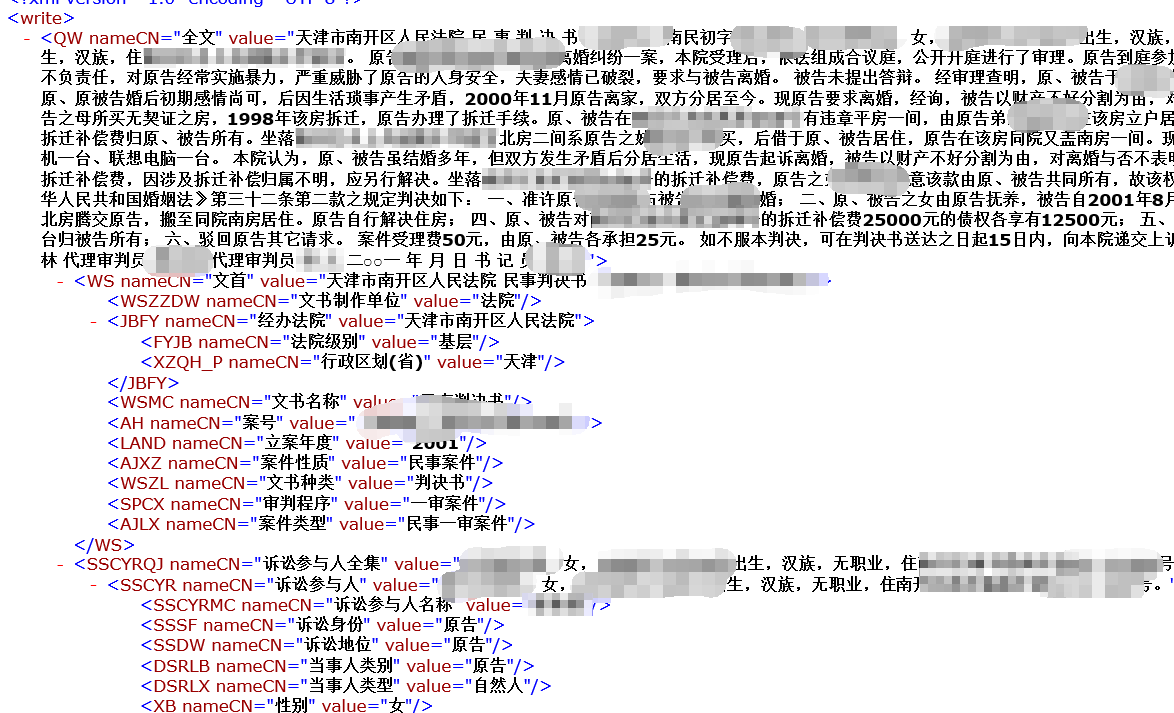


图4.14 通用型文本信息抽取模块的部分结果展示

## 4.2 要素式文本信息抽取模块的实现

### 4.2.1 通用信息结果加载功能的实现

根据要素式文本信息抽取模块中通用信息结果加载功能的设计，该功能需要载入通用型文本信息抽取模块输出的XML结果，并根据文档节点属性读入到系统中来，用于后续抽取工作。

该功能实现读取XML文件使用的是开源的dom4j软件工具开发包，使用dom4j里面提供的SAXReader来读取文件，生成Document实例。dom4j里面也提供DocumentHelper工具类去快速构建XML的文件结构。

根据设计，系统实现了LoadWSXML类，该类里有读取文书XML并返回各字段值的功能，然后将返回的Element根节点实例交由ElementExtraction类去进一步抽取要素。

通用信息结果加载功能主要包含的类如表4.15所示。

表4.15 通用信息结果加载功能主要类统计表

|  |  |
| --- | --- |
| 分类 | 主要包含的类 |
| 业务类 | LoadWSXML.java |
| 工具类 | SAXReader.java |

XML文件部分节点和含义如下表4.16所示。

表4.16 段落详细信息XML保存结果含义对应（部分）表

|  |  |
| --- | --- |
| 节点名 | 存储信息 |
| WS | 文首 |
| WSZZDW | 文书制作单位 |
| WSMC | 文书名称 |
| LAND | 立案年度 |
| SSCYRMC | 诉讼参与人名称 |

### 4.2.2 规则匹配信息抽取功能的实现

根据3.4.2节中要素式文本信息抽取模块的离婚纠纷案由要素抽取功能设计，该功能需要利用上述加载结果并结合离婚纠纷案由要素表去抽取要素信息，将离婚纠纷要素分为了两类，并设计了两种方式去抽取，一种是使用正则表达式等规则匹配的方式去抽取，另一种是使用xgboost模型去做二元分类抽取相关信息。

1. 规则匹配的抽取实现

根据设计部分的介绍，首先利用了ZZElementExtra类去管理该类要素的抽取方法，根据设计，ZZElementExtra类里的要素抽取方法名和要素序号挂钩，方法名为handleRow1表示的是抽取该类第一条要素信息的方法，handleRow2是处理第二条要素信息的方法，以此类推。以下列三条要素抽取方法的实现细节为例，介绍规则匹配方法去进行要素抽取一般性过程：抽取“相识时间以及如何相识”这条要素的handleRow1方法、抽取“夫妻共同债务”这条要素的handleRow11方法、抽取“是否存在一方下落不明满二年，对方起诉离婚，经公告查找确无下落的情形”这条要素的handleRow15方法。

使用handleRow1方法去抽取“相识时间以及如何相识”这一要素。根据裁判文书的写作方式，相识这一事件通常会在案件基本情况的查明事实段里面，所以首先使用LoadWSXML类里的getCmssd静态方法，去获取查明事实段节点，然后利用在BaseWsAnalyseImpl里的getWholeContent方法将查明事实段文本按照句号、逗号和分号划分成句子。之后在所有的句子中匹配关键字，描述相识情况的句子会包含关键字“相识”或者“结识”，将匹配到的句子设为“如何相识”这一要素节点，并用LoadWSXML.setRhss静态方法添加到节点树中。通常该句子中也有相识时间这一信息，所以使用TimePro类里的getDate方法去利用时间正则表达式匹配该句子，即可获取相识时间，然后再使用LoadWSXML.setXssj静态方法将相识时间添加到对应节点。如果没有匹配到时间信息，则将相识时间置为空。根据3.4.2节设计部分的介绍，对于没有相识时间信息的文书，则将该要素节点值设置为“未提及”。其余时间信息和文本信息类要素也是同样的设置。

抽取“相识时间和如何相识”要素的部分代码如下图4.15所示。

|  |
| --- |
| public void handleRow1(Element rootEle){  String cmssd= LoadWSXML.getCmssd (rootEle);  Pattern pattern = Pattern.compile(ZZElementExtra.datePattern);  String[] cmssdArray = BaseWsAnalyseImpl. getWholeContent (cmssd);  ArrayList xsdateList = new ArrayList();  for (String token: cmssdArray) {  Matcher matcher = pattern.matcher(token);  Boolean rs = matcher.find();  if (rs) {  if (qw.contains("相识")|| qw.contains("结识")) {  TimePro.getDate(token, xsdateList, JBSS.datePattern);  LoadWSXML.setXssj(rootEle,xsdateList.get(0));  LoadWSXML.setRhss(rootEle,token);  break; }}} |

图4.15 抽取“相识时间和如何相识”要素的部分代码

使用handleRow11方法去抽取“夫妻共同债务”这一要素。根据裁判文书的写作方式，夫妻共同债务相关描述通常会出现在案件基本情况的查明事实段。仍然使用LoadWSXML类里的getCmssd静态方法，去获取查明事实段节点。然后将句子依据标点符号划分成小句子，依据关键词 “还款”或者“ 偿还”并结合“共同”一词去判断夫妻共同债务情况。然后根据钱币的量词“元”或者“¥”去匹配得到债务数目。最后利用LoadWSXML.setFqgtzw方法将夫妻共同债务信息添加进节点树中。

抽取“夫妻共同债务”要素项的部分代码如下图4.16所示。

|  |
| --- |
| public static void handleRow11 (Element rootEle){  ArrayList<String>loanList=new ArrayList<String>();  boolean flag=false;//检查是否有贷款项  if(root.attribute("value")!=null) {  String cmssd= LoadWSXML.getCmssd(rootEle);  String[] cmssdArray= BaseWsAnalyseImpl. getWholeContent (cmssd);   for(String analyse: cmssdArray) {  if(analyse.contains("共同")&&(analyse.contains("偿还")||analyse.contains ("还款"))){  String[] reason=analyse.split("偿还|还款");  for(String addReason:reason) {  if(!(*checkReapt*(loanList,addReason))) {  matcher=pattern.matcher(addReason);  if(matcher.find()){  if(addReason.contains("元")||addReason.contains("¥")) {  loanList.add(addReason);  flag=true;}} } } }}}  ……//存储相关信息进入节点} |

图4.16 “夫妻共同债权、债务情况”要素抽取的部分代码

使用handleRow15方法去抽取“是否存在一方下落不明满二年，对方起诉离婚，经公告查找确无下落的情形”这一要素信息。根据对文书的分析，该要素的描述通常会出现在查明事实段和审理结论这两处。首先使用LoadWSXML.getCmssd和LoadWSXML.getSljl去载入对应节点，然后使用标点符号分割的办法分割成句子。由于此要素描述通常会在某一句中，因此以句号作为分割。该要素实际可以分为两个信息，一是下落不明两年，二是发公告查找无结果。使用“离家”,“出走”,“下落不明”,“去向不明”等关键字去获取出走这一事件，然后使用TimePro.getRightDate方法获取出走的时间，使用立案年度减去出走时间得到下落不明的时长。然后接着使用“公告”，“无下落”，“未找到”等关键字匹配发布公告无结果这一事件。结合这两种信息可以得到所要抽取的结论，并存入到相应节点。显然现在规则制定已经开始变复杂了，下面介绍的监督学习去抽取信息可以避开这类繁琐的规则制定。

抽取“是否存在一方下落不明满二年，对方起诉离婚，经公告查找确无下落的情形”要素项的部分代码如下图4.17所示，此处将下图名称中的要素简称为“是否存在一方下落不明两年并查找不得情形”。

|  |
| --- |
| public static void handleRow15(Element rootEle){  String ifcz = "否";  String allStr = "";  Element cmssdNode = LoadWSXML.getCmssd (rootEle);  Element sljlNode = LoadWSXML.getSljl (rootEle);  allStr += cmssdNode.attributeValue("value");  allStr += sljlNode.attributeValue("value");  String[] qwStr = allStr.split("。");  ……//出走时间，出走事件，当前时间，公告查找已经两个标记值的声明  String[] keyWords = {"离家","出走","下落不明","去向不明"};  //获取出走时间  for(String qw:qwStr){  if((qw.contains("离家")||qw.contains("出走")&&(qw.contains("下落不明")||qw.contains("去向不明")))){  czsj = TimePro.getRightDate(qw,keyWords , 1, JBSS.datePattern);  flag1 =1;  break; }}  if(!czsj.equals("")){cznf = TimePro.getNumbers(czsj); }  if(flag1==1){ //是否进行公告查找  for(String qw:qwStr){  if(qw.contains("公告")&&(qw.contains("确无下落")||qw.contains("未找到"))){ flagifsz = 1; }}}  Element cpsjNode = LoadWSXML.getCpsj ("WW");  nowdate = TimePro.getDateEach(cpsjNode.attributeValue("value"), ZZElementExtra.datePattern);  ……//利用立案年度去获取当前时间，从而获取出走时长  if(flagm2n\*flagifsz==1){ifcz = "是";}  ……//存入相关节点} |

图4.17 “是否存在一方下落不明两年并查找不得情形”要素抽取的部分代码

### 4.2.3 监督学习信息抽取功能的实现

设计部分按照训练模型与调用模型两方面去描述，这里依然按照这两方面去介绍它的功能实现，为了方便起见，以该类第四条要素“是否存在家庭暴力或虐待、遗弃家庭成员的情形”为例，介绍监督学习过程的具体实现。

1. 功能的实现

根据设计，将模型的训练放在Python端去实现，数据集使用的是标注好的3065条数据，其中存在家暴情形的有1538条，不存在家暴情况有1527条数据。本文选取其中600条作为测试数据，2465条作为训练数据，数据中家暴占比相同。根据标注的类别，分别使用两个文件夹存储XML文件。设计了ws\_preprocessing类去管理数据的载入和预处理。数据载入由load\_ws\_xml函数负责处理，此处使用了Python里面的xml.dom.minidom模块去读取XML文件，在该类中还设计了多个函数读取某些特定节点。以上述第四条要素为例，家庭暴力情况一般会在原告诉称段以及审理查明段有介绍，使用get\_ygsc\_str和get\_slcm\_str函数获取所载入文书中相关节点的value值。在读取完相关文本后之后，需要对文本进行预处理，转为文本向量的形式，以下是训练模型所需的各步骤。

1. 裁判文书中文分词

数据质量是能够训练出优秀效果模型的基础。作为裁判文书预处理的一个重要步骤，分词的效果直接影响了模型的好坏。目前常用的分词工具有jieba、LTP、NLPIR以及ICTCLAS等。本文选择jieba作为分词工具，分词前需要去掉非中文字符，这里利用基本汉字Unicode编码4E00到9FA5去匹配除去非中文字符。

1. 裁判文书去除停用词

某些语气词，通用词不能表示一篇文档的特征，所以需要对这些词语进行去除，还需要考虑到法院判决书会有专业领域常用词，也要一并加以去除。本文的停用词表一部分来源于通用停用词表，另一部分来源于使用IDF（逆文档频率）计算得到的专业领域停用词。

1. 构建语料库和词典文件

由于本文的语料来源于XML文件，如果每次读取会十分麻烦，所以需要单独构建一个语料库，方便以后每次的使用，语料库每一行代表了一篇文档的分词结果，每个词之间以空格隔开。使用files2str函数读取文书集目录下的所有文件，并调用get\_qw\_str方法，读取全文数据返回文书集的list。然后调用dataSet\_word\_split函数，将里面的字符串进行分词，去停用词，最后将语料库存储到文件中去。除了全文语料库，还需要构建各节点语料库，以便后面TF-IDF计算时使用。另外需要使用全文构建词典，它的存储可以使用gensim里面dictionary.save方法实现。

1. 训练word2vec模型

本要素项使用裁判文书集里的查明事实段和原告诉称段作为模型训练的语料库，然后使用gensim提供的word2vec模块，输入分词处理后的词语列表，并载入训练，word2vec模型训练过程需要有如下几个参数设置：

size：是指特征向量的维度，本文设置为200。大的size需要更多的训练数据，但是效果会更好。

min\_count: 最小的统计词频，词频少于min\_count次数的单词会被丢弃掉, 这里设为1，不丢掉任何单词。

window：表示当前词与预测词在一个句子中的最大距离是多少，本文设置为20，可以获取足够多案情事件的上下文信息。

构建完word2vec模型后，使用save方法序列化到指定文件。

1. 计算文本TF-IDF

根据3.4.2小节文本特征向量的设计，需要tfidf作为重要参数。这里使用gensim里面的TfidfModel模块去计算tfidf，首先使用加载之前存储的词典和语料库，构建tfidf模型，利用模型计算训练文本的tfidf值。

1. 计算文本特征向量

根据3.4.2小节文本特征向量的设计，首先要计算训练文本词语的word2vec值。先加载之前保存的word2vec模型，然后利用模型计算训练文本的词向量。之后利用之前构建的tfidf模型，计算出训练文本的tfidf值，根据之前设计部分的公式（3.4），计算文本向量，公式（3.4）是指利用tfidf作为词向量的权值，这样词向量不是简单的累加成文本向量，而是加权之后累加，然后将累加后的文本向量同tfidf一起作为文本的特征表示。

7）模型训练

设计部分选择了xgboost算法，这里使用陈天奇博士等人开发的开源软件包作为xgboost的实现。首先将文本特征向量存储为libsvm格式，由于是二分类问题，将类别列设为1,0来表示“是”与“否”两种情况。对于训练过程，使用ws\_classification类作为职责承担者。ws\_classification类在初始化的过程中需要传入名为params的参数，用作构建xgboost的超参数，这里超参数是指模型本身的结构参数。使用类中的xg\_training方法载入训练数据，并加以训练，然后使用xgboost自带的save函数，将模型序列化，以下是xgboost部分超参数。

booster：选择gbtree，gbtree使用基于树的模型进行提升计算，gblinear使用线性模型进行提升计算。

max\_depth：树的深度，深度越高，分类效果越好，但是也可能会过拟合。

eta：学习率，设置为0.001。

objective：设置一些其余参数，选择binary:logistic表示二分类逻辑回归。

num\_boost\_round：训练时的参数，表示迭代提升的次数。

8）模型调用

在Java部分建立了JDElementExtra类去管理该类要素抽取方法，其方法名同ZZElementExtra类里的类似，为handleRow1，handleRow2等。handleRow1方法是Java端处理该要素的方法，该方法首先调用LoadWSXML类里的getYgscd和getCmssd去获取文书的相关文本字符串，然后通过设置好的socket请求，传递一个Map对象形成的JSON字符串，每次请求的数据包含以下信息：model\_name表示请求的模型名称标识，analysis\_str表示待分类的字符串，比如此条要素抽取模型标识为数字4，待分类字符串为之前提取的文本信息。本文是以“over”字符串作为每次socket请求传输结束的标识。

Python部分使用socket\_main方法启动socket服务器，服务器启动的过程也会同时将那些分类模型一并加载到内存。在接收到请求之后，使用json模块解析字符串，重新转为dict类型。之后读取模型名称并根据它加载对应的模型，输出预测结果返回Java程序。

1. 评估标准

对于分类效果的评价指标，本文采用了准确率（accuracy）作为评价。对于本文的二分类问题，其混淆矩阵如表4.17所示。

表4.17 混淆矩阵

|  |  |  |
| --- | --- | --- |
| 预测  真实 | 正（有家暴） | 负（无家暴） |
| 正（有家暴） | TP | FN |
| 负（无家暴） | FP | TN |

本文定义 TP表示实际为正类、预测也为正类的样本数量 ；FN 表示实际为正类、预测为反类的样本数量 ；FP 表示实际为反类、预测为正类的样本数量 ；TN 表示实际为反类、预测也为反类的样本数量，那么上述评价指标可以由以下公式（4.1）表示。

（4.1）

1. 超参数选择

图4.18 max\_depth参数测试图

这里依然以第四条要素的模型训练过程为例，进行超参数选择的介绍。max\_depth表示xgboost树的深度，深度越高，准确率越好，但是会导致过拟合，本文测试了学习率eta为0.001，迭代次数num\_boost\_round为2000情况下，不同的max\_depth对误差率的影响，图4.18中val-error表示的是验证集中误分类的样本数除以总的验证集样本数。由图4.18可知，max\_depth参数在达到6时，验证集误差已经处于低点了，而参数为7的效果虽然与参数为6相同，但是树的深度越深，越容易过拟合，所以本文选择6作为max\_depth的参数。

num\_boost\_round表示迭代提升的次数，设置为2000，在2000之后就已经不能够获得更大的提升了。如图4.19所示，前期验证集误差较高，但是很快下降，最后在2000左右达到水平线附近，后续波动也较小。

图4.19 num\_boost\_round参数测试图

1. word2vec和tfidf训练选择

本文设计的特征向量涉及到word2vec和tfidf的加权处理，也就是将tfidf作为词向量的权重，加权求和得到文本向量。本文尝试了三种不同的训练语料选择，一是使用裁判文书全文作为word2vec和tfidf语料库，二是使用裁判文书全文作为word2vec语料而将查明事实段与原告诉称段作为tfidf的语料，三是将查明事实段与原告诉称段作为word2vec和tfidf的语料。

实验发现仅使用查明事实段和原告诉称段是一个比较好的选择，因为文书的分类主要依据描述家暴事件的语句，同文书全文的关系有限，后面的特征评估实验也能体现这一点。

1. 对比试验评估

1）特征选择的对比评估

使用以下四种模型作为文本的特征表示进行分类，并使用上述评估标准来判断所选择特征的好坏，此处仍然使用xgboost作为分类算法。TF-IDF、word2vec、加权 word2vec 以及合并加权 word2vec 和 TF-IDF 这四种模型对实验数据集进行分类，并分析停用词对分类准确率的影响。

由表4.18可以看出，单独使用word2vec进行分类的效果并不理想，而选择TF-IDF加权后的word2vec模型的准确率比word2vec模型高，但是略低于选取TF-IDF作为特征的模型。合并加权word2vec和TF-IDF 两种模型，使用合并后的向量作为文本特征的模型准确率最高。另外，去掉停用词同样可以改善模型分类准确率。

表4.18 两类别某要素分类结果

|  |  |
| --- | --- |
| 类别 | accuracy |
| 带停用词word2vec | 0.7617 |
| 去停用词word2vec | 0.7667 |
| 去停用词TFIDF | 0.7983 |
| 带停用词TFIDF | 0.7966 |
| 加权word2vec | 0.7766 |
| 加权word2vec+TFIDF | 0.8100 |

TF-IDF效果高于word2vec，可以得到如下结论，对于这类文本分类问题，其关键词占着较大的比重。这和原问题还是很契合的，一般对于裁判文书而言，出现家暴才会在内容里描述，而不出现则不会描述，这是一个关键词的问题。

对于TF-IDF加权之后的word2vec 模型准确率是高于word2vec模型也是可以理解的，这是加强了关键词的语义空间在向量求和时的权值，使得文本向量更多的朝着TF-IDF值较高的方向偏移。至于该特征模型略低于TF-IDF则说明关键词的作用更大，而只是将TF-IDF作为权值会弱化关键词的作用。

合并加权 word2vec 和 TF-IDF这种特征选择是最好效果的，说明使用word2vec去获取词与词之间的联系能够引入先验信息，提升了仅使用TF-IDF作为特征分类的效果。

2）与其他模型的对比试验

图4.20 不同分类器分类结果对比图

合并加权 word2vec 和 TF-IDF作为特征，组合KNN分类器、支持向量机分类器与 xgboost 分类器进行对比，通过准确率进行性能评估如下图4.20所示，此处的random seed（用于初始化随机数的参数）均设置为0。

由图4.20可以看出xgboost的准确率是高于其他方法的，其中KNN效果较低，支持向量机可以有较好的效果。

（6）监督学习要素抽取结果统计

对监督学习所需要抽取的要素项，有如下表4.19的要素统计结果，数据集来自于天津法院的部分数据。其中正类比例是指存在该要素描述情形，反类是指不存在。

表4.19 监督学习要素抽取结果统计表

|  |  |  |
| --- | --- | --- |
| 要素项名称 | 正类比例 | 反类比例 |
| 是否存在一方请求离婚而另一方不同意离婚的情形 | 0.658 | 0.342 |
| 是否存在婚前缺乏了解，草率结婚，婚后未建立起夫妻感情，难以共同生活的情形 | 0.323 | 0.677 |
| 是否存在性格不合，经常发生矛盾，难以共同生活的情形 | 0.661 | 0.339 |
| 是否存在家庭暴力或虐待、遗弃家庭成员的情形 | 0.253 | 0.747 |
| 是否存在一方或双方当事人有婚外情、与他人同居等情形 | 0.031 | 0.969 |
| 是否存在一方或双方当事人有赌博、吸毒等恶习的情形 | 0.036 | 0.964 |
| 是否存在双方办理结婚登记后，未同居生活，无和好可能的情形 | 0.132 | 0.868 |
| 是否存在一方曾起诉离婚，经法院处理后夫妻感情未改善的情形 | 0.104 | 0.896 |
| 是否存在因感情不和分居满二年，或经法院判决不准离婚后又分居满一年，互不履行夫妻义务的情形 | 0.028 | 0.972 |
| 是否存在一方患有法定禁止结婚疾病的，或一方有生理缺陷，或其它原因不能发生性行为，且难以治愈的情形 | 0.051 | 0.949 |

训练模型方面部分功能代码如下图4.21所示。

|  |
| --- |
| # 对数据集分词、去停用词  def wordSplit(wsContentList):  stopword = stopwordslist(stop\_word\_path) # 创建通用词列表  m = len(wsContentList)  wordList = []  for i in range(m):  rowListRemoveStr = format\_str(wsContentList[i]) # 去特殊符号  rowList = [eachWord for eachWord in jieba.cut(rowListRemoveStr)] # 分词  removeStopwordList = []  for eachword in rowList:  if eachword not in stopword and eachword != '\t' and eachword != ' ':  removeStopwordList.append(eachword)…//添加进wordList并返回 |

图4.21 分词、去停用词功能代码

### 4.2.4 要素式文本信息抽取的结果



图4.22 要素式文本信息抽取模块的抽取结果（部分）

根据上述小节描述的方法，通过规则匹配和监督学习抽取出了离婚纠纷案件裁判文书的要素信息，并将要素信息抽取结果添加到了通用型文本信息抽取所形成的XML文件的节点之中并重新保存。该模块所生成的新XML文件部分节点内容如图4.22所示。

## 4.3 文书分类统计模块的实现

根据文书分类统计模块的设计，该模块的主要目的是对裁判文书按照各种要素信息进行分类统计。根据中国司法大数据研究院发布的关于离婚纠纷报告里面的统计项，本文将统计离婚纠纷案件裁判文书的以下相关信息：1)根据案件审结情况统计各年的案件量；2)根据当事人特征统计原告性别分布、原被告年龄差分布；3)根据案件特征统计婚姻存续时间、离婚态度、判决结果、家暴案件省份分布。依据法院提供的统计项，利用之前抽取的裁判文书的通用型信息和要素式信息，统计裁判文书在统计项里的分布情况，并将分布结果展示出来。文书分类统计模块有分类统计记录功能和网络平台发布功能。

### 4.3.1 分类统计记录功能

文书的分类统计模块需要统计上述的各年案件量、原告性别、家暴案件省份、判决结果、离婚态度、原被告年龄差、婚姻存续时间这七个统计项的分布信息，本文将上述七个统计项的数据分布计算任务交由ClassStatisticService去处理。ClassStatisticService类会调用之前LoadWSXML类里的方法，读取要素抽取的结果并返回所需要节点的Element实例。然后调用calculateDis方法，将读取的结果交由各分布对应的处理方法进行分析，上诉七个分布依次对应的处理方法是calcGnajlDis、calcYgxbDis、calcJbajsfDis、calcPjjgDis、calcLhtdDis、calcYbgnlcDis、calcHycxsjDis，这里采用的是properties去存储之前统计的信息，利用JDK提供的Properties类可以方便的操作文件。

接下来介绍上述七个统计项的分布计算过程：

1. 各年案件量的统计分布计算

calcGnajlDis方法会调用LoadWSXML类里的getLand方法，传入根节点rootEle去进一步获取“立案年度”节点，获得立案年度这一个详细信息。该方法从gnajl.properties文件里获取gnajlMap，该Map的key是年度，value是当前统计到的案件数量，当有新的案件被处理时，先确定是否已经有该立案年度对应的键值对，如果有则对其value加1，否则新增一组键值对，案件的立案年度为key，value设为1，由此可以统计各年案件量。

1. 原告性别的统计分布计算

calcYgxbDis方法会调用LoadWSXML类里的getYGsscyr方法，传入根节点rootEle去进一步获取“原告诉讼参与人”节点，找到诉讼身份为原告的诉讼参与人性别。该方法从ygxb.properties文件里面去获取ygxbMap，该Map有两个key，分别为男和女这两个字符串，读取新的裁判文书后，根据原告性别信息在相应的value上加1，由此可以统计原告性别分布。

1. 家暴案件省份的统计分布计算

calcJbajsfDis方法会调用LoadWSXML类里的getYSJTBL方法，传入根节点rootEle去进一步获取要素项“是否存在家庭暴力或虐待、遗弃家庭成员的情形”这一节点，如果其节点value属性值是“是”，则调用getSJxzqh方法获取文首段落节点的行政区划信息。如果不存在家暴行为则不计算该案例。该方法从jbajsf.properties文件里面去获取jbajsfMap，该Map的key是省份名称，value是该省家暴案件数量。读取新的裁判文书后，如果存在家暴情况，则在相应的省份的value上加1，由此可以统计家暴案件省份分布。

1. 判决结果的统计分布计算

calcPjjgDis方法会调用LoadWSXML类里的getSFZYLH方法，传入根节点rootEle去进一步获取“是否准予离婚”节点的value属性值，属性值为“是”表示解除婚姻。该方法从pjjg.properties里面去获取pjjgMap，该Map有两个key，分别为“解除婚姻关系”和“维持婚姻关系”这两个字符串，读取新的裁判文书后，根据节点值在相应的value上加1，由此可以统计判决结果分布。

1. 离婚态度的统计分布计算

calcLhtdDis方法会调用LoadWSXML类里的getSFZYYFLH方法，传入根节点rootEle去进一步获取“是否存在一方请求离婚而另一方不同意离婚的情形”节点的value属性值，节点值为“是”表示只有一方想离婚。该方法从lhtd.properties文件里面去获取lhtdMap，该Map有两个key，分别为“只有一方想离婚”和“双方都想离婚”这两个字符串，读取新的裁判文书后，根据节点属性值，在对应的value上加1，由此可以统计离婚态度分布。

1. 原被告年龄差的统计分布计算

calcYbgnlcDis方法会调用LoadWSXML类里的getYGSR和getBGSR方法，传入根节点rootEle去进一步获取诉讼参与人各节点的当事人地位的属性值，以此获取原告和被告的信息，并得到原告和被告的出生年月节点，通过出生年月来计算他们的年龄差。该方法从ybgnlc.properties文件里面去获取ybgnlcMap，该Map的key对应着年龄差的数字，value对应中符合该年龄差的案件数目。读取新的裁判文书后，根据读取分析的原被告年龄差是否在键值对的key里面，如果在的话，就在对应的value上加1，否则添加该年龄差作为key，其初始value设为1，由此可以统计原被告年龄差分布。

1. 婚姻存续时间的统计分布计算

calcHycxsjDis方法会首先根据之前判决结果统计项里dealPjjg方法的计算结果来判断是否离婚，如果离婚了，那么调用LoadWSXML类里的getJHDJSJ方法，传入根节点rootEle去进一步获取 “结婚登记时间”这一个要素节点，根据立案年度时间和结婚登记时间这两者的时间差可以计算得到婚姻存续时间。该方法从hycxsj.properties文件里面去获取hycxsjMap，该键值对的key对应着婚姻存续时间，value对应中符合该存续时间的案件数目。读取新的裁判文书后，根据读取分析出的婚姻存续时间是否在键值对的key里面，如果在的话，就在对应的value上加1，否则将该婚姻存续时间作为key，其初始value设为1，添加到hycxsjMap，由此可以统计婚姻存续时间分布。

在分类统计完这些裁判文书之后，需要将要素提取完的XML文件由completeDir文件夹移动到statisticFinishDir文件夹里。最后该模块在统计完信息后，将这些Map重新存储到各自对应的properties文件属性值里面，本文统一使用properties的json属性存储Map的JSON字符串，文件地址由Constant类的静态变量来保存。

### 4.3.2 网络平台发布功能

1. 前端实现

在设计中，本系统将提供一个用以展示裁判文书统计分布情况的界面，本文利用了Vue和Echarts来设计一个展示界面。使用Vue的组件重用技术，对展示用的图标组件进行了重用，使用Echarts去提供数据可视化功能。

在设计中，各年的案件量和家暴案件省份这两个分布是用柱状图来展示，原告性别分布、判决结果和离婚态度这三个分布是用饼状图来展示，原被告年龄差和婚姻存续时间这个两个分布是用平滑曲线图来展示。所以需要有三个基本组件来实现柱状图、饼状图和平滑曲线图，其分别是my\_pillar.vue,my\_pie.vue和my\_line.vue。

这七个统计项分别有六个函数去向后端请求各自的分布数据来渲染界面，分别用getGnajlDis、getYgxbDis、getJbajsfDis、getPjjgDis、getLhtdDis、getYbgnlcDis、getHycxsjDis来请求各年案件量、原告性别、家暴案件省份、判决结果、离婚态度、原被告年龄差、婚姻存续时间这几个统计项的分布情况，家暴案件省份分布这一统计项只展示数目排名靠前的几个省份。

该功能的前端涉及的内容如表4.20所示。

表4.20 文书分类统计模块的前端内容

|  |  |
| --- | --- |
| 分类 | 主要包含的内容 |
| Vue文件 | my\_pie.vue、my\_line.vue  my\_pillar.vue、count\_result.vue |

1. 后端实现

根据后端的设计，由ClassStatisticController类去转发对应的前端请求，由于前端有七个分布请求，后端也有七个与之对应的方法处理这些请求，其名字也和前端的一致，分别为getGnajlDis、getYgxbDis、getJbajsfDis、getPjjgDis、getLhtdDis、getYbgnlcDis、getHycxsjDis。接收请求后，将这些请求转交给ClassStatisticService去处理。它会去读取之前统计分布之后存储到硬盘上的文件，将各自需要的文件读取到内存，重新将各自的JSON字符串转成java对象，再整理好信息后返回给前端。由于前端有三种类型的表示信息，所以本文实现了readForPillar，readForPie，readForLine这三个方法作为元方法整理信息，并分别由各请求处理方法调用，整理好的信息包括用于渲染前端的数字信息和描述统计结果的文字信息，文字信息采取的是固定句式表达，一些具有数据分析性质的信息也一并在此处理，以下是两种统计项的效果演示。

判决结果的统计分布结果如图4.23所示。

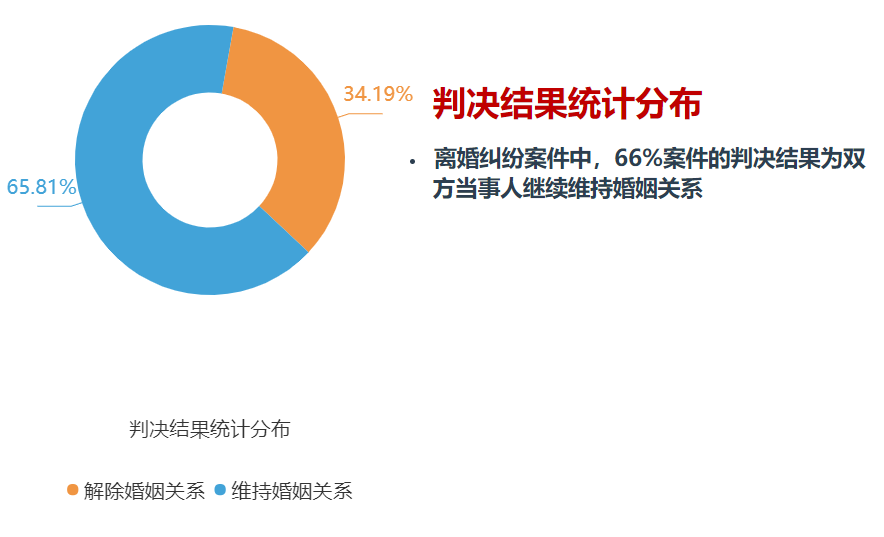


图4.23 判决结果统计分布（天津法院部分数据集）展示结果

婚姻存续时间的统计分布的发布效果如图4.24所示。

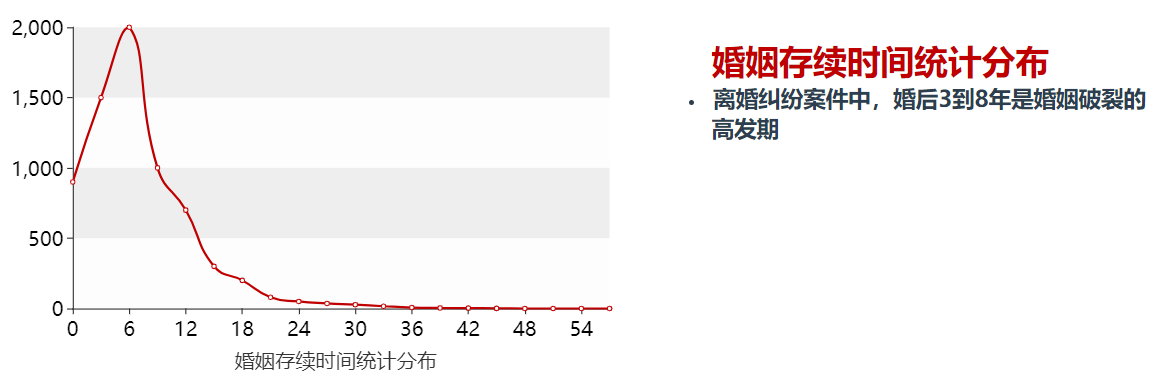


图4.24 婚姻存续时间统计分布（天津法院部分数据集）展示结果

## 4.4 文书管理模块的实现

根据文书管理模块中抽取结果管理功能的设计，抽取结果管理功能是指管理保存通用型文本信息抽取模块和要素式文本信息抽取模块的输出结果。根据之前的设计。抽取结果管理功能共有五个文件夹来存储信息，这五个文件夹是addDir、completeDir、halfCompleteDir、sourceDir和statisticFinishDir，分别存储的是新增的裁判文书、要素式文本信息抽取完之后的裁判文书、通用型文本信息抽取后的裁判文书、源文件和经过统计之后的文件，这五个文件夹的地址由Constant类的静态变量存储。在WSManagerService里面实现了copyAndReplaceFile，copyFile，copyAndNotReplaceFile，moveAndReplace File去进行文件的移动。还实现了saveHalfXML和saveComplXML两个方法去将信息抽取结果利用DocumentHelper存储到halfCompleteDir和completeDir文件夹，部分代码如图4.25。

|  |
| --- |
| public static void copyAndNotReplaceFile(String srcPath, String desPath) throws Exception {  srcPath = separatorReplace(srcPath); //分隔符替换，用于windows和linux平台转换  desPath = separatorReplace(desPath);  File srcFile = new File(srcPath);  File desFile = new File(desPath);  if (!srcFile.isFile()) {  throw new Exception("source file not found!"); }  if (desFile.isFile()) { return; }  copyFile(srcPath, desPath); } |

图4.25 文件移动方法代码

## 4.5 系统测试

### 4.5.1 测试目标

本文主要对系统进行了性能测试，性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。负载测试和压力测试都属于性能测试，两者可以结合进行。通过负载测试，确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。压力测试是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试。

根据上述性能测试的区分，本节设计了两个测试目标：

一是信息抽取负载测试，测试在待处理裁判文书集逐步增加的情况下，两个信息抽取模块处理文书信息的处理时间变化情况。

二是Web压力测试，测试文书分类统计模块提供的Web平台的压力性能测试，判断能否接受大量的访问。

压力性能测试使用Apache JMeter，它是一款基于Java的压力测试工具，使用Apache JMeter可以通过创建进程组配置进程参数对服务器模拟巨大的负载测试系统的强度并分析系统整体性能[佘青, 2012]。

### 4.5.2 测试过程与结果

（1）信息抽取负载测试

本文选择1000篇，2000篇，5000篇文书作为负载测试的三个不同的负载，分别记录程序进行信息抽取的时长，时长的记录源于在控制台打印了函数运行的起始时间和终止时间，时长记录如表4.21所示。

表4.21 信息抽取负载测试结果

|  |  |
| --- | --- |
| 负载程度 | 抽取花费时长 |
| 1000篇 | 65s |
| 2000篇 | 1min49s |
| 5000篇 | 4min22s |

从上述测试结果可以看出，抽取花费时长同文书集数目大致成正比，每篇文书所花费的时间不超过0.08s，文书信息抽取效率能够满足对系统的期待。

（2）Web压力测试

根据设计，选用Apache JMeter作为Web压力测试工具，首先在Apache JMeter中创建进程组Thread Group，将Number of Threads参数设置为500，将Ramp-Up Period参数设置为1，上述两个参数表示，Apache JMeter将在一秒内创建500个虚拟用户模拟现实情况，对测试的系统接口进行访问。然后配置每个测试接口的请求参数，点击运行按钮，运行测试。Apache JMeter对所有的测试接口发送完请求后生成的测试报告如图4.26所示。

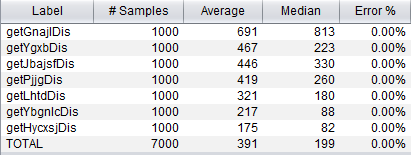


图4.26 Jmeter测试报告

根据测试结果报告的内容，可以看出每一行代表了统计项的请求接口，getGnajlDis代表了各年案件量统计项的接口测试，其平均响应时间为691ms，中位响应时间为813ms，接口异常率为0。getJbajsfDis代表了家暴案件省份统计项的接口测试，其平均响应时间为446ms，中位响应时间为330ms，接口异常率为0。getLhtdDis代表了离婚态度统计项的接口测试，其平均响应时间为321ms，中位响应时间为180ms，接口异常率为0。getYgxbDis代表了原告性别统计项的接口测试，其平均响应时间为467ms，中位响应时间为223ms，接口异常率为0。getPjjgDis代表了判决结果统计项的接口测试，其平均响应时间为419ms，中位响应时间为260ms，接口异常率为0。getYbgnlcDis代表了原被告年龄差统计项的接口测试，其平均响应时间为217ms，中位响应时间为88ms，接口异常率为0。getHycxsjDis代表了婚姻存续时间统计项的接口测试，其平均响应时间为175ms，中位响应时间为88ms，接口异常率为0。

图4.27 Jmeter测试聚合图

根据图4.27所示的测试结果聚合柱状图，其平均响应时间均没有超过900ms，故障率为0，可以满足大部分用户的使用体验。

## 4.6 本章小结

本章主要介绍裁判文书的文本信息抽取与分类统计系统的实现，使用了运行过程描述、关键代码、结果界面截图等方式展现了裁判文书的文本信息抽取与分类统计系统的通用型文本信息抽取、要素式文本信息抽取、文书分类统计、文书管理模块的具体实现，最后对系统性能进行了测试。

# 总结与展望

## 5.1 论文总结

裁判文书记载人民法院审理的过程和结果，它是诉讼活动结果的载体。法院经常需要针对不同主题或者案由的案件做专题分类统计，相关分类统计项没有记录在审判流程系统的关系数据库中，因此统计的难度非常大，相关统计的数据源只能从裁判文书的文本中获取。

本文针对法院裁判文书的写作体裁，设计了通用型文本信息抽取模块，利用特征词将裁判文书划分成文首、诉讼参与人、诉讼记录、案件基本情况、裁判分析过程和文尾这六个段落，并在各段落中进一步抽取了更为详细的信息。

本文还在通用型文本信息抽取模块的基础上，根据不同的案由，设计了要素式文本信息抽取模块。本文以离婚纠纷案由为例，针对离婚纠纷案由的要素项特点，在通用型文本信息抽取的基础上，利用规则匹配和监督学习两种方式去抽取要素信息。

本文以中国司法大数据研究院出具的专题报告中的统计项为例，设计了文书分类统计模块，介绍了如何利用上述信息抽取的结果去计算统计项的统计分布，并结合可视化工具，将统计分布直观的展示出来。

## 5.2 进一步工作展望

本文目前对文书的分析仅限于某篇具体的裁判文书，但是在法院的文书中，对于每一篇判决书都有与之对应的诉讼申请书等法律文书，结合这些描述同一案件的法律文书进行信息抽取可以发掘出案件更复杂和深入的信息，也就可以更好的对案件进行分类统计。

本系统的通用型文本信息抽取目前使用的是关键字词的匹配，它从本质上来说是基于规则的算法，但是在了解到监督学习的过程中，我发现，这种段落划分如果存在标注好的训练数据，也有潜力使用机器学习[陈凯等, 2007]的方法来进行划分。我对这种划分的设计思路是对于已有标记的数据，将它的段落划分点所在的标点符号的序号作为类别标记，然后一篇文书标点符号的数目就是类别数目，由文书词向量作为模型输入，标点序号作为输出，利用训练数据可以得到模型，但是现有的基于规则进行通用型信息抽取已经能够达到要求，暂时可以不对他进行扩展，而且标注数据的花费也是十分大的。

本系统的要素式文本信息抽取涉及到很多原告与被告的关系，以及其余人物实体之间的关系[徐健等, 2008]，我在查阅资料的时候，发现对于这些描述实体与实体之间关系的信息抽取可使用一种名为关系抽取的技术。利用关系抽取技术，可以获取一篇文书中各实体之间的关系，以及事件的关系描述，不过这个技术需要大量的标注数据，还需要对实体进行识别[David et al., 2007]，很多公司都在利用这些技术去构建知识图谱[刘峤等, 2016]，构建知识图谱还需要专业人员去搭建相关领域本体库[Boella et **al.**, 2019]，该方法成本也是极其高昂的。

本系统设计的信息抽取是基于裁判文书的写作体裁，但是在我的观察中发现，写作体裁也不是绝对的，有文章提出了利用句子主题去辅助信息抽取[廉站俊等, 2007]，也就是先判断句子的主题向量是否更符合需要抽取区域的主题向量，这种做法做一定程度上避免了依据写作体裁去抽取信息的局限性，如果对于信息抽取有着更高的要求，可以使用该思路加强效果。

# 参 考 文 献

[Boella et **al.**, 2019] G. Boella, L. D. Caro and V. Leone, Semi-automatic knowledge population in a legal document management system, *Artificial Intelligence and Law*, 2019, 27(2):227-251.

[Branting, 2017] L. K. Branting, Automating Judicial Document Analysis, *ASAIL@ICAIL*, 2017.

[Cui et al., 2017] M. Cui, L. Li and Z. Wang et al., A survey on relation extraction, In *China Conference on Knowledge Graph and Semantic Computing(CCKS’2017)*, pages 50-58*,* 2017*.*

[Chen et **al.**, 2016] T. Chen and C. Guestrin, XGBoost: A Scalable Tree Boosting System, In *Knowledge Discovery and Data Mining(KDD’2016)*, pages 785-794, 2016.

[Chalkidis et **al.**, 2019] I. Chalkidis and D. Kampas, Deep learning in law: early adaptation and legal word embeddings trained on large corpora, *Artificial Intelligence and Law*, 2019, 27(2):171-198.

[DeJong, 1982] G. DeJong, An overview of the FRUMP system, *Strategies for natural language processing*, 1982, 113:149-176.

[David et al., 2007] N. David and S. Satoshi, A survey of named entity recognition and classification, *Lingvisticae Investigationes*, 2007, 30(1):3-26.

[Grishman, 2015] R. Grishman, Information Extraction, *IEEE Intelligent Systems*, 2015, 30(5):8-15.

[Kanhabua et **al.,** 2016] N. Kanhabua and A Anand, Temporal information retrieval, In *International Conference on Research and Development in Information Retrieval(SIGIR’2016)*, pages 1235-1238, 2016.

[Sager et al., 1981] N. Sager, J. S. Siegel and E. A. Larmon, Natural Language Information Processing: a computer grammar of English and its applications, *Reading*, MA: Addison-Wesley, 1981.

[陈凯等, 2007] 陈凯，朱钰，机器学习及其相关算法综述，统计与信息论坛，2007，21(5):105-112。

[傅郁林, 2000] 傅郁林，民事裁判文书的功能与风格，*中国社会科学*，2000，20(4):123-133。

[胡启敏等，2008] 胡启敏，薛锦云，钟林辉，基于Spring框架的轻量级J2EE架构与应用，*计算机工程与应用，*2008, 44(5):115-118。

[焦鹏珲, 2018] 焦鹏珲，*基于 SpringBoot和Vue框架的电子招投标系统的设计与实现，*硕士论文，南京大学，2018。

[旷志光等, 2017] 旷志光，纪婷婷，吴小丽，基于Vue.js的后台单页应用管理系统的研究与实现，*现代计算机(专业版)*，2017，33(30):51-55。

[李峰等，2010] 李峰，刘彦隆，基于SSH框架与jQuery技术的Java Web开发应用，*科技情报开发与经济，*2010，20(6):106-108。

[李保利等, 2003] 李保利，陈玉忠，俞士汶，信息抽取研究综述，*计算机工程与应用，*2003，39(10):1-5。

[廉站俊等, 2007] 廉站俊，吕学强，张玉杰等，基于句子相似度计算的信息抽取，*数据分析与知识发现，*2007，2(6):38-41。

[刘翔宇, 2018] 刘翔宇，*基于Vue的数据可视化系统的设计与实现，*硕士论文，北京邮电大学，2018。

[刘迁等, 2007] 刘迁，焦慧，贾惠波，信息抽取技术的发展现状及构建方法的研究，*计算机应用研究，*2007，24(7):6-9。

[刘峤等, 2016] 刘峤，李杨，段宏等，知识图谱构建技术综述，*计算机研究与发展，*2016，53(3): 582-600。

[佘青, 2012] 佘青，利用Apache Jmeter进行Web性能测试的研究，*智能计算机与应用*，2012，2(2):55-57。

[石倩等, 2008] 石倩，陈荣，鲁明羽，基于规则归纳的信息抽取系统实现，*计算机工程与应用，*2008，44(21):166-170。

[王辰阳, 2018] 王辰阳，人工智能下的智慧法院，*法制与社会，*2018，19(1):95-96。

[徐健等, 2008] 徐健，张智雄，吴振新，实体关系抽取的技术方法综述，*数据分析与知识发现，*2008，24(8):18-23。

[翟剑锟, 2013] 翟剑锟，*Spring框架技术分析及应用研究*，硕士论文，中国科学院大学*，*2013。

[翟全军等, 2017] 翟全军，李明睿，大数据与法院司法统计方式和口径改革，*统计与信息论坛，*2017，32(3):127-128。

[张雪敏, 2018] 张雪敏，浅议 MVC 设计模式在 JavaWeb 中的作用，*科技风，*2018， 30(22):55。

[周合明等, 2011] 周合明，奚建清，基于模板的Web信息提取系统的设计与实现，*计算机技术与发展，*2011，21(11)：105-108。

[张春波, 2016] 张春波，要素式审判法: 庭审方式与裁判文书的创新， *中国审判，*2016， 16(24): 36-38。

[张谦等, 2017] 张谦，高章敏，刘嘉勇，基于Word2vec的微博短文本分类研究，*信息网络安全，*2017, 17(1):57-62。

**[最高人民法院, 2018] http://www.court.gov.cn/zixun-xiangqing-85042.html，最高法工作报告解读系列访谈：加快建设智慧法院，2018。**

**[中国法院网, 2018] https://www.chinacourt.org/article/detail/2017/11/id/ 3089555.html*，*要素式审判在庭审中的运用，2018。**

**[中国裁判文书网, 2019]http://wenshu.court.gov.cn/*，*中国裁判文书网首页，2019****。**

# 致 谢

两年前我有幸考入南京大学软件学院，在这两年里，我遇到了贴心的师长、优秀的同学，享受到了高质量的学习环境和人文环境，这些都极大地激励了我，让我能够努力成为一个更加优秀的自己。两年的学习中，我得到了许多来自老师、同学的帮助，这给我留下了难忘的回忆。

首先我要感谢的是我的导师葛老师，他在我为论文题目迷茫的时候，给我指明了切实可行的方向；同时他在我陷入困顿中，给予了我指导和帮助；而老师平时严谨认真负责的工作作风也让我感受到了作为一名学者的宝贵品质。

我还要感谢周筱羽老师，她在学习与研究上给了我很多的帮助。

我还需要感谢我亲爱的同学们，他们刻苦钻研的态度和用心生活的作风让我找到了学习的榜样，他们优秀的性格让我受益匪浅。

我还想感谢南京大学软件学院，它给了我一个转变自己人生的支点，让我能够朝着我的梦想更进一步，我衷心的感谢南京大学，感谢这个平台给我的机会。

最后我想感谢的是我的父母，他们一直在我的背后默默支持我、鼓励我，伴我成长，助我前行。

# 版权及论文原创性说明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：

日期： 年 月 日