南京大學

研 究 生 毕 业 论 文
（申请工程硕士学位）

论 文 题 目 基于司法多维评估的模型评估系统

作 者 姓 名 贺璐

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 陈振宇教授

2022 年 **5** 月 **23** 日

学　　号：**MF20320063**

论文答辩日期：**2022** 年 **05** 月 **20** 日

指 导 教 师：　　（签字）

Model Evaluation System Based on Judicial Multidimensional

by

He Lu

Supervised by

Professor ZhenYu Chen

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
MASTER OF ENGINEERING
in
Software Engineering

Software Institute
Nanjing University

May 23, 2022

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于司法多维评估的模型评估系统

工程硕士（软件工程领域）专业 2020 级硕士生姓名：贺璐

指导教师（姓名、职称）：陈振宇教授

摘　要

随着大数据时代的到来，各类基于人工智能技术的司法办案辅助系统的使用也越来越频繁。司法办案辅助系统有效地提升了司法办案的效率，缓解了“案多人少”的司法困境，充分发挥了人工智能技术对司法行业的优化重塑效应。但人工智能与司法行业的深入结合也面临着诸多挑战，例如人工智能模型所隐含的“算法黑箱”与司法办案过程中所重视的“可视正义”背道而驰，但完备的司法模型评估体系缺还未建立起来，与此同时现存的司法数据质量也不能保证人工智能模型得到充分的训练，这些问题的存在使得一些学者开始探讨人工智能技术在司法领域的应用边界，但寻找问题解决方案的步伐也从未停止。

本文的主要工作是建立基于司法多维评估的模型评估系统，旨在通过评估模型在司法多维指标下的表现从侧面消解“算法黑箱”带来的隐患。司法多维评估指标分为基础性指标和变异性指标，基础性指标的实现是基于传统的多分类评估指标，在具体实现时为其赋予了特定的司法含义，而变异性指标的计算则基于模型在使用系统默认数据集分类和使用变异数据集分类时的准确性差异，不同领域变异算法的使用决定对应的变异性指标的含义。本文设计的司法多维评估指标包括准确性、泛化性、可参考性、公平性和鲁棒性，其中前三种属于基础性评估指标，后两种属于变异性评估指标，使用了包括基于语法的歧视性文本插入、基于 TF-IDF 的特征裁剪、基于依存句法的特征变换和基于词典的随机插入在内的五种变异算法。

除了司法多维评估指标体系的建立，本文设计并实现了多阶段自动化大规模模型评估筛选系统，在具体实现中系统分为三个阶段对模型进行筛选，通过使用 docker 容器技术保证模型具有持续部署和持续运行的能力，同时通过监听时间点任务实现自动化流程控制，从而降低人力维护成本，提高评估效率；除

此之外系统针对可能存在的大规模模型评估任务设计了易扩展的服务器架构，从而极大地提升了系统的稳健性并拓宽了系统的使用场景，为大规模模型评估筛选系统搭建提供了可行方案。按照系统的功能性需求，系统被划分为了七个模块，包括网关模块、认证授权模块、阶段管理模型、数据集模块、文件管理模块、容器管理模块和异构服务模块。网关模块和认证授权模块协同工作对系统进行访问控制；阶段管理模块对各阶段中的规则、时间、基准线和数据集进行管理；数据集模块提供数据集管理和变异服务；文件管理模块对结果集文件、数据集文件和容器压缩文件进行管理，提供上传下载等功能；容器管理模块对系统中的容器进行管理，调用异构服务中的容器导入和健康检查接口；异构服务模块为其他模块调用异构服务提供接口。

系统以 web 方式提供服务，系统中的用户角色分为比赛管理员和参赛选手，不同角色的权限和可视页面有所区别。司法多维评估系统对参赛选手提交的结果集文件和容器压缩文件进行评估，针对模型分类结果生成评估报告，评估报告包含模型在五种指标中的具体得分和总得分，据此给出模型优化方向，系统各阶段基于总得分使用比赛管理员设定的基准值筛选容器。经测试系统可在多阶段自动化筛选流程后得到质量较高的司法模型，同时易扩展设计帮助系统在负载过大时灵活扩充评估服务器数量，在实际的比赛场景中有较高的实用性和稳定性。除此之外，在系统测试中还针对 fastText 多分类模型、TextCNN 模型及多层 LSTM 模型使用本文提出的司法多维评估指标进行评估，根据各模型在原始数据集和变异数据集上的表现获取各指标的评估分数，并对其进行比较。

关键词： 司法多维指标，司法模型，数据变异，多阶段自动化筛选，大规模模型评估

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Model Evaluation System Based on Judicial Multidimensional

SPECIALIZATION: Software Engineering
POSTGRADUATE: He Lu
MENTOR: Professor ZhenYu Chen

Abstract

With the advent of the era of big data, the application of artificial intelligence in the judicial field is more and more in-depth, and the use of various judicial case handling auxiliary systems is more and more frequent.The judicial case handling auxiliary system effectively improves the efficiency of judicial case handling, alleviates the judicial dilemma of ”many cases but few people”, and gives full play to the optimization and reshaping effect of artificial intelligence technology on the judicial industry. However, the in-depth integration of AI and the judicial industry also faces many challenges. For example, the ”algorithmic black box” implied by the artificial intelligence model contradicts the ”visual justice” that is valued in the judicial case handling process. At the same time, the quality and volume of the existing judicial data cannot guarantee that the artificial intelligence model will be fully trained. The existence of these problems has led some scholars to discuss the application boundary of artificial intelligence technology in the judicial field, but the pace of finding solutions has never stopped.

In this thesis, the main work is to establish evaluation system based on judicial multidimensional evaluation model, aims to eliminate the ”algorithmic black box” from the side by evaluating the performance of the model under the judicial multidimensional indicators. the judicial multidimensional evaluation index is divided into basic index and mutation index, and the realization of the basic index is based on the traditional multi-category evaluation index, while specific judicial meanings are given to them in the specific implementation.The calculation of the mutation index is based on the accuracy difference between the original data set classification and the amplified data set classification of the model. The use of the mutation algorithm in different

fields determines the meaning of the corresponding mutation index. The judicial multi-dimensional evaluation indexes designed in this thesis include accuracy, generalization, reference value, fairness and robustness, among which the first three are basic evaluation indexes and the last two are mutation evaluation indexes. Five mutation algorithms are used, including grammar-based discriminatory text insertion, TF-IDF based feature clip-out, dependency syntactic based feature transformation and dictionary-based random insertion.

In addition to the establishment of the judicial multi-dimensional evaluation index system, this thesis designs and implements a multi-stage automatic large-scale model evaluation and screening system. In the specific implementation, the system is divided into three stages to screen the models, and the docker container technology is used to ensure that the model has the ability to continuously deploy and run, and at the same time it realizes automatic process control by monitoring point-in-time tasks, thereby reducing labor maintenance costs and improving evaluation efficiency; In addition, the system has designed an easily extensible server architecture for possible large-scale model evaluation tasks, thus extremely it greatly improves the robustness of the system and broadens the usage scenarios of the system, providing a feasible solution for the construction of a large-scale model evaluation and screening system. According to the functional requirements of the system, the system is divided into seven modules, including gateway module, authentication and authorization module, stage management model, data set module, file management module, container management module and out-service module. The gateway module and authentication and authorization module work together to control the access of the system. The stage management module manages the rule, time, datum line and data set in each stage. The data set module provides data set management and mutation services; The file management module manages the result set file, data set file and container compressed file, and provides functions such as upload and download. The container management module manages the containers in the system and invokes the interfaces of container import and health check in heterogeneous services. out service modules invoke heterogeneous service submission interfaces for other modules.

The system provides services in the form of web. The user roles in the system are divided into competition administrators and contestants.The permissions and visual

pages of different roles are different. The judicial multi-dimensional evaluation system evaluates the result set files and container compressed files submitted by the contestants, then generates an evaluation report based on the model classification results. The evaluation report contains the specific scores and total scores of the model in five indexes, then the model optimization direction is given accordingly. each stage of the system filters the containers based on the total score using the benchmark value set by the game administrator. After the process mentioned above,the system can obtain some high-quality judicial models . At the same time, the easy-to-expand design helps the system to flexibly expand the number of evaluation servers when the load is too large, and has high practicability and stability in actual competition scenarios.Apart from this, the fastText multi-classification model, TextCNN model and multi-layer LSTM model are also evaluated using the judicial multi-dimensional evaluation indicators proposed in this thesis and compare their result.

keywords: Judicial multidimensional index,Judicial model,Data mutation,Multi-stage automated evaluation screening system,Large-scale model evaluation

目　　录

目　　录 ····· **vi**
插图清单 ····· **ix**
附表清单 ····· **xi**
第一章　引言 ····· **1**
1.1　选题的背景和意义 ····· 1
1.2　国内外研究现状及分析 ····· 3
1.3　本文主要研究工作 ····· 5
1.4　本文的组织结构 ····· 6
第二章　技术综述 ····· **8**
2.1　Spring 框架 ····· 8
2.1.1　Spring Boot 框架 ····· 8
2.1.2　Spring Cloud 微服务 ····· 9
2.2　Docker 技术 ····· 9
2.3　FastDFS 技术 ····· 10
2.4　自然语言处理技术 ····· 10
2.4.1　中文文本分词技术 ····· 11
2.4.2　依存句法分析 ····· 11
2.5　TF-IDF 算法 ····· 13
2.6　本章小结 ····· 14
第三章　系统需求分析与概要设计 ····· **15**
3.1　系统总体规划 ····· 15
3.2　系统需求分析 ····· 16
3.2.1　系统功能性需求 ····· 16
3.2.2　系统非功能性需求 ····· 19
3.2.3　基于司法多维的模型评估系统用例设计 ····· 20
3.3　系统设计 ····· 29
3.3.1　系统架构设计 ····· 29

3.3.2 基于司法多维评估的模型评估系统逻辑设计 …… 31
3.3.3 系统开发设计 …… 32
3.3.4 系统进程设计 …… 33
3.3.5 系统部署设计 …… 35
3.3.6 系统数据库设计 …… 36
3.4 本章小结 …… 45
第四章 数据变异方法与司法模型评估指标的设计 …… 46
4.1 数据变异方法 …… 46
4.1.1 鲁棒性变异 …… 46
4.1.2 公平性变异 …… 49
4.2 司法模型评估指标设计 …… 50
4.3 本章小结 …… 53
第五章 系统详细设计与实现 …… 54
5.1 认证授权模块详细设计与实现 …… 54
5.1.1 认证授权模块详细设计 …… 54
5.1.2 认证授权模块具体实现 …… 55
5.2 文件管理模块 …… 56
5.2.1 文件管理模块详细设计 …… 56
5.2.2 文件管理模块具体实现 …… 59
5.3 阶段管理模块 …… 60
5.3.1 阶段管理模块详细设计 …… 60
5.3.2 阶段管理模块具体实现 …… 62
5.4 数据集管理模块 …… 65
5.4.1 数据集管理模块详细设计 …… 65
5.4.2 数据集管理模块具体实现 …… 67
5.5 容器管理模块 …… 71
5.5.1 容器管理模块详细设计 …… 71
5.5.2 容器管理模块具体实现 …… 73
5.6 本章小结 …… 80
第六章 系统测试与分析 …… 81
6.1 测试环境 …… 81
6.2 功能性测试 …… 82

6.3 非功能性测试 ····· 90
6.4 实验数据及分析 ····· 93
6.4.1 实验数据 ····· 93
6.4.2 实验分析 ····· 95
6.5 本章小节 ····· 96
第七章 总结与展望 ····· **97**
7.1 总结 ····· 97
7.2 展望 ····· 98
致 谢 ····· **99**
参考文献 ····· **100**
简历与科研成果 ····· **104**

插图清单

2-1 依存句法树示例 …… 13

3-1 分阶段评估筛选流程图 …… 15
3-2 基于司法多维评估的模型评估系统用例图 …… 21
3-3 基于司法多维评估的模型评估系统架构图 …… 30
3-4 基于司法多维评估的模型评估系统逻辑视图 …… 31
3-5 系统开发视图 …… 32
3-6 数据集处理进程视图 …… 34
3-7 容器处理进程视图 …… 35
3-8 系统部署设计视图 …… 36
3-9 系统核心实体关系图 …… 37

5-1 认证授权流转时序图 …… 54
5-2 客户端认证关键代码 …… 56
5-3 文件服务器组织结构图 …… 57
5-4 文件上传后端处理流程图 …… 58
5-5 uploadFastDfs 方法代码 …… 59
5-6 fastdfs 配置信息 …… 60
5-7 比赛阶段总流程图 …… 61
5-8 三种阶段权限状态图 …… 62
5-9 阶段流程实现图 …… 63
5-10 阶段时间管理配置项 …… 63
5-11 阶段数据集管理设置项 …… 65
5-12 数据集文件处理流程时序图 …… 66
5-13 数据项格式示例 …… 68
5-14 分批生成词典文件代码 …… 69
5-15 数据集变异实现图 …… 69
5-16 合并文件核心代码 …… 71

5-17 容器导入及健康检查顺序图 ····· 73
5-18 nginx 调度服务配置 ····· 75
5-19 启动待导入文件监听服务 ····· 76
5-20 容器导入触发逻辑流程图 ····· 77
5-21 容器批量导入脚本核心代码 ····· 77
5-22 容器健康检查关键代码 ····· 78
5-23 异构服务器上的阶段数据集记录文件 ····· 79
5-24 结果集监听评估核心代码 ····· 79
5-25 批量运行容器脚本中的容器运行命令 ····· 80

6-1 无权限跳转的 404 页面 ····· 92
6-2 容器导入错误信息 ····· 92
6-3 各模型指标值对比柱状图 ····· 96

附表清单

2-1 依存关系类型表 ····· 12

3-1 基于司法多维的模型评估系统功能性需求列表 ····· 17
3-2 系统非功能性需求列表 ····· 20
3-3 数据集下载用例描述表 ····· 22
3-4 新增结果集用例描述表 ····· 22
3-5 新增容器用例描述表 ····· 23
3-6 容器运行及评估用例描述表 ····· 24
3-7 新增数据集用例描述表 ····· 25
3-8 变异数据集用例描述表 ····· 26
3-9 阶段规则设置用例描述表 ····· 26
3-10 阶段时间设置用例描述表 ····· 27
3-11 阶段基准线设置用例描述表 ····· 28
3-12 阶段数据集设置用例描述表 ····· 28
3-13 file 表字段设计 ····· 37
3-14 dataset 表字段设计 ····· 38
3-15 amplification_dataset 表字段设计 ····· 40
3-16 container 表字段设计 ····· 41
3-17 evaluation 表字段设计 ····· 42
3-18 stage 表字段设计 ····· 42
3-19 stage_time 表字段设计 ····· 43
3-20 stage_dataset 表字段设计 ····· 44
3-21 stage_batch 表字段设计 ····· 44

4-1 基于 TF-IDF 特征裁剪算法变异后的文本对照表 ····· 47
4-2 基于依存句法的特征变换算法变异后的文本对照表 ····· 48
4-3 基于词典的随机插入算法变异后的文本对照表 ····· 49
4-4 基于语义的歧视性文本插入算法变异后的文本对照表 ····· 50

5-1 阶段时间类型任务详情 ····· 64
5-2 数据集模块中异构服务接口调用详情 ····· 67
5-3 容器状态值及含义 ····· 73

6-1 系统测试环境信息表 ····· 81
6-2 数据集下载用例测试结果 ····· 82
6-3 新增结果集用例测试结果 ····· 83
6-4 新增容器用例测试结果 ····· 83
6-5 容器运行及评估用例测试结果 ····· 84
6-6 新增数据集用例测试结果 ····· 85
6-7 变异数据集用例测试结果 ····· 85
6-8 变异数据集用例测试结果 ····· 86
6-9 设置阶段时间用例测试结果 ····· 87
6-10 阶段基准线设置用例测试结果 ····· 89
6-11 阶段数据集设置用例测试结果 ····· 90
6-12 接口响应时长测试结果 ····· 91
6-13 新增容器功能时长测试结果 ····· 91
6-14 新增容器功能并发量与异构服务器数量关系 ····· 91
6-15 fastText 模型在不同数据集上的测试准确率 ····· 93
6-16 TextCNN 模型在不同数据集上的测试准确率 ····· 94
6-17 TextCNN 模型在不同数据集上的测试准确率 ····· 95
6-18 不同模型在各评估指标的分值 ····· 95

第一章　引言

1.1　选题的背景和意义

自 1956 年，“人工智能”概念第一次被提出后 [1]，在起起伏伏的技术革新浪潮中，人工智能逐渐被大众所熟知，其应用也渗透到了医疗、教育、司法、交通等社会生活的方方面面，提升了各领域工作的质量和效率从而有效地改善了人类的生活 [2]。将人工智能技术应用于司法领域最早可以追溯到 20 世纪 70 年代，当时欧美等发达国家就在具体的司法实践中使用了基于人工智能技术开发的法律模拟分析系统及法律推理系统 [3]。21 世纪是大数据的时代，依托于大数据时代数据收集、数据存储、数据分析能力的极速提升，人工智能技术在司法领域的应用也迈向了一个新的台阶，展现出了巨大的技术理性的力量，从目前的实践来看，类案推荐、量刑辅助和离场预警是大数据和人工智能最典型的应用结合 [4]。

2015 年，我国最高法院首次提出“智慧法院”的概念，在 2017 年又对“智慧法院”的建设从国家层面提出了相关要求 [5]，包括促进法官类案同判和量刑规范化，构建应用成效评估改进机制，设计应用评估指标和改进方案等，至此人工智能技术在在我国司法领域的落地越来越深入。2020 年颁布的《法治蓝皮书》指出我国的智慧法院建设已取得诸多成果，基本建成了贯通司法全过程的智慧法院体系 [6]，其中智慧审判是智慧司法的重要一环，其在保障法官基本的自由裁量权的基础上提供量刑意见，为审判过程注入技术理性 [7]。

加速智慧司法的发展是我国司法行业现状的要求，一直以来我国司法行业都面临着“案多人少”的困境，即案例过多而司法办案人员较少的现状，一方面这是公民权力意识增强的体现 [8]，另一方面也反映了“网络化、智能化、数字化”司法的迫切性。人们制定法律的意义就是维护社会秩序，追求公平公正，而“公正在法律中的第二层含义是指效率” [9]，依托于人工智能技术所孵化出来的智慧司法应用有助于解决效率问题。这些应用不仅有助于通过提升一线司法办案人员的工作效率来破解“案多人少”的司法困境，也能借助人工智能技术充分发挥对司法行业的优化重塑效应。除此之外，“同案同判”也是司

法行业在量刑审判过程中关注的重点，在传统的司法领域中，对于案件的裁决往往依赖于法官个人的“法感”，人工智能的引入促进了司法决策从“单一人脑决策”转向“聚合智脑决策” [10]，减少了个人主观性对司法决策公正性的影响，让“同案同判”在一定程度上成为可能。

事实上，人工智能在我国司法行业的发展并不是自顶向下的，众多地方法院根据自身需求推出了能有效提升工作效率、降低司法周期的辅助办案系统，例如上海的案件智能辅助办案系统可抓取和校验证据信息并具备初步的逻辑分析能力，江苏的同案不同判预警系统基于构建的图谱进行类案识别、量刑预测等一系操作达到量刑偏离预警的效果，贵州的智能模拟判决系统针对案件要素检索相关法条并提供判决依据 [11]，这些智能化辅助系统在实际司法场景的落地所取得的成果都揭示了司法行业在人工智能时代背景下的突破。然而，人工智能在司法领域的具体应用一直停留在“辅助”阶段，学术界也在探讨司法人工智能的应用边界，即人工智能技术在不破坏司法特性的情况下可以介入司法判案的最大限度，在推动人工智能与司法深度结合的进程中所面临的问题和挑战主要包括以下两点：

模型在司法层面的表现难以评估。尽管人工智能模型基于大量司法数据巩固了司法过程的客观性，摈除了由于单个司法者的任意性导致的公众对于司法裁决的不信任感。但是人工智能所使用的算法受限于算法设计者对于司法过程和司法事实的理解，很难实现从“接近正义”向“可视正义”的迈进 [12]。模型背后隐藏的“算法黑箱”，其不予公开、不予解释的思想很难保证不会影响司法过程的客观公正。因此，在人工智能模型投入到实际的司法场景中使用之前，应该充分评估其是否符合基本的司法要求、是否恪守基本的司法底线，例如模型在辅助办案时是否具有可参考性的价值，模型审判结果是否基于公平公正的原则等。

模型训练数据不够完备。目前，司法模型训练数据大多取自“中国裁判文书网”上已有的裁判文书 [11]。虽然“中国裁判文书网”上的数据基本涵盖了我国进行司法建设以来的裁判文书，但文书内容的质量却有待商榷，首先部分文书记录的案件事实过于简短，对于影响判决的诸多因素记录不够详实且存在着非结构化用语的问题 [9]，导致模型不可避免地忽略了案件裁决过程中的“关键性信息”，其次裁判文书中的部分案例相关要素的记录不够完整，例如有一些文书中缺少刑期判决结果标签，或是案件类型的分类标签，这些信息在刑期预测和类案推荐中都起到至关重要的作用。总之，现有的裁判文书无论是在内

容上还是结构上都有其自身的限制，但人工智能模型的训练对数据的质量和体量要求又比较高，这一现实困境很大程度上动摇了人工智能在司法行业发挥积极作用的根基。

为了充分释放人工智能在司法领域的潜能、通过技术“克服或至少是瓦解正义之路上的一些障碍”[12]，本文提出建立一套基于司法多维的模型评估体系，并且将数据变异技术引入该模型评估系统。建立基于司法多维的评估体系在一定程度上可以帮助司法相关人员规避掉不符合司法要求的人工智能模型，间接达到防止“算法霸权”的目的，同时模型评估结果也可以为算法设计者提供改进模型的思路和方向。而将数据变异技术引入模型评估系统可以解决前述的司法数据不完备的问题，同时多维评估的实现也依赖于使用不同的数据变异技术定向地为模型在某一指标上的评估提供测试数据。

1.2 国内外研究现状及分析

人工智能在司法领域的应用已有近 60 年的发展 [13]，但对司法模型评估的研究却泛善可陈。在研究司法多维评估模型的相关研究时，首先把基本的模型评估体系建设方面的工作作为基本的切入点，然后会讨论司法领域相关的评估工作，最后会探讨数据变异在司法领域的应用与具体评估指标结合的相关工作。

评估指标体系在设计时应考虑的原则包括科学性、可行性、客观性及可比性 [14]。中国电子技术标准化研究院等单位制提出了对机器学习模型质量进行测试的指标体系，以功能性、可靠性、效率和维护性四个方面作为主要的指标特性，该体系具有领域通用性 [15]，在司法领域使用的人工智能模型也以满足这四个方面的要求为基础，故在建立司法多维评估体系的过程中可以借鉴该标准所提出的质量要素和相关的测试方法。

在司法模型的评估方面，王栋 [16] 针对司法案例筛选模型建立了测试系统，该评估系统从司法模型和案例筛选接口两个角度对案例筛选模型进行测试，提供的评估指标除了传统的 AI 评估指标 (包括准确度、精确度、宏平均、微平均、拟合度、AUC) 以外，还引入了模型在社会层面的评估指标 (包括纠正度、解释性、公平性、一致性、相似性、依赖性)。其工作在建立司法多维评估体系的过程中针对如何扩展司法模型评估指标提供了较高的参考性，但针对各类指标提出的计算方法对各类司法模型没有很好的泛化和迁移能力，同时由于

未能对指标和评估数据之间进行特征关联，所以获得的评估结果受具体的测试数据集的影响比较大。

赵 [17] 等人针对司法决策系统的复杂性和模糊性特点，引入了 TOPSIS 模型，建立了主客观相结合的智能司法决策系统的综合评估模型。在该评估模型的实现过程中首先对指标按照 9 个多属性综合评价指标进行分级，之后使用层次分析法和熵权法对指标的主管权重和客观权重进行计算，最后通过专家强制打分法确定指标的组合权重，根据具体的指标分类构建单个指标的属性度量函数，计算综合属性度量值，并根据属性识别标准判断综合评价等级。基于该工作可以根据模型在具体的指标表现给出模型总体的司法价值评估。

张 [18] 等人参考传统软件测试中数据突变的测试方法引入文本突变法，来达到对审判案件预测模型的鲁棒性进行评估的效果。在实现过程中，基于 fastText、TextCNN 以及多层 LSTM 模型在相关的数据集上进行训练获得相应的司法刑期预测模型，其后从分类性能、词序变化、噪声变化三个方面得出不同模型的敏感程度来反映模型对噪声数据的鲁棒性表现。在该工作中创新性地将数据变异引入司法模型的评估过程中，为后续的评估研究提供了一个新的思路。

在数据变异方面主要专注司法领域的案例文本变异。引入针对司法领域的文本变异技术可以解决高质量司法案例数据较少和分布不均的问题，降低人工整理和标记案例数据的成本。除此之外通过设计数据变异方法可将司法领域中关注的评估指标与变异后的案例数据进行特征关联，使得指标的评估可以与特定的数据之间进行解耦。在大多数的评估任务中，通常只会关注模型在通用的评估指标上的表现，因为通用指标的评估不受数据的限制，使用数据变异技术后可控制数据整体的特性，从而达到特定指标评估的目的。

现有的文本技术主要包括回译和加噪。回译的实现方式是将文本通过机器翻译成其他语言之后再翻译回原语言，这种方式可以有效地增加数据的多样性，保证文本基本的语义不会发生改变，但回译的方式比较依赖机器翻译的质量。加噪是在原数据的基础上通过对词的增删改来达到生成新数据的效果，在原数据集较小的情况下加噪可以有效地扩充训练数据集的量级。

在通用的文本数据变异场景下，回译和加噪都能取得很好的变异效果，但是司法领域的案例文本用词都具有特定的法律属性，对词语的细小改动都会影响文本在法律上的含义，例如将“刑事处罚”修改为“刑罚处罚”就表达了两种不同的含义，前者是伴随法院的有罪判决生效，而后者则表示在违反刑法后

应受到制裁的预期，由于司法领域的严谨性和特殊性通用的文本变异方式并不适用于此。在对司法案例文本进行变异时应该充分考虑其文本领域的特征，李卓阳 [?] 提出了四种针对特定领域场景的文本变异技术包括：基于 TF-IDF 权重的特征裁剪、基于主题模型的特征融合、基于依存句法的特征变换、基于词频词性的特征替换。运用其提出的四种变异方法可以获得符合司法使用场景的案例文本数据。张舒 [19] 设计并实现了司法文本数据自动化生成系统，从基于规则和基于变分编码器两种方式对司法文本进行变异，在实现对应变异方式的同时也设计该变异方式下对模型测试的评估指标，包括对模型抗噪能力及抗对抗攻击能力的测试。

除了关注具体的领域文本数据变异技术以外，变异数据的质量检测也不容忽视，严格 [20] 针对司法领域的数据质量搭建了评估系统，从可解释性、相关性、准确性三个方面对数据质量进行评估，为变异后的司法数据提供质量保障。

1.3 本文主要研究工作

随着人工智能技术的迅速发展，智慧司法在科学赋能的背景下已经催生了众多的司法辅助系统，例如辅助审案的智能判案系统及类案推荐系统等。人工智能与司法行业的深度结合不仅仅归功于人工智能技术的进步，更来源于司法行业内部需求的推动。然而维护司法公信力的“可视正义”在人工智能时代面临着新的风险和挑战，一方面模型在司法层面的表现难以评估，这势必导致了模型在司法层面的不可解释性；另一方面训练和评估模型的司法数据不完备成为了司法人工智能进一步发展的又一障碍。

本文针对人工智能模型在司法领域应用的困境，结合数据变异技术构建一套基于司法多维的模型评估方法。相较于传统的人工智能模型评估指标，司法多维评估指标具有司法层面解释含义，为所评估的模型在司法层面的应用提供指导性价值。为体现模型在司法层面的可解释性一共提出五种司法评估指标，具体包括：准确性、可参考性、一致性、鲁棒性和公平性。其中准确性、可参考性、一致性属于基础评估指标，而鲁棒性和公平性属于变异指标，变异指标的评估会结合相应的数据变异技术，通过调整相应变异方法的参数生成目标测试集，模型在相应测试集上的量化表现就体现了模型在对应指标上的评估得分。

基于上述研究内容，本文的主要工作是搭建基于司法多维评估指标的多阶段大规模模型评估系统。该系统基于 Spring Boot 微服务框架提供前后端交互服务，具体的模型评估任务下沉到多台服务器，在这些服务器上通过 flask 框架提供具体的操作接口，服务器的数量可根据待评估的模型数量进行调整，从而在硬件资源和系统可承载量上达到动态平衡。同时，为规避模型部署和运行失败的风险在该系统中引入 docker 容器技术，针对容器导入、运行设计了一套并发机制，该并发机制可以有效地降低系统的响应时间，同时也能控制并发数量提高系统的稳健性。除此之外系统引入了 fastdfs 技术对文件进行分布式管理，同步不同服务器上的容器和数据集。

1.4 本文的组织结构

本文建立基于司法多维的模型评估系统从五个司法指标对模型进行评估，司法指标分为基础指标和变异指标两类，基础指标在司法领域的背景下被赋予具体的司法含义，而变异指标则基于不同的变异算法在司法领域下变异的领域特征数据。除了对评估指标进行研究以外，本系统设计并实现了一套自动化分阶段评估批量筛选系统。本文组织结构如下：

第一章引言，阐述基于司法多维的评估系统的项目背景及意义、国内外研究现状及本文的主要研究工作。

第二章技术综述，介绍了司法多维评估系统后端选择的开源框架、使模型具备持续部署能力的 Docker 技术和实现系统易扩展效果的 FastDFS 技术，同时对数据变异中所使用的自然语言处理技术进行了说明。

第三章司法多维评估系统的需求分析与概要设计，针对司法多维评估系统的相关的功能性需求和非功能性需求进行详细分析，在需求分析的基础上从多个视图进行司法多维评估系统的概要设计并划分具体的功能模块，在最后对系统的持久化设计和司法多维评估指标设计进行了介绍。

第四章数据变异方法与司法模型评估指标的设计，该部分对系统中使用的四种变异算法进行了详细介绍，并给出了司法多维指标评估体系中的基础性指标和变异性指标的计算方式。

第五章司法多维评估系统的详细设计及实现，该部分在第三章所述的概要分析基础上对司法多维评估系统进行详细设计，并按照详细设计对实现系统。

第六章司法多维评估系统的测试与分析，首先从司法多维评估系统的功能

性需求和非功能性需求两个层面对系统进行测试，确保司法多维评估系统功能的完整性和可靠性，其次针对 fastText 多分类模型使用司法多维评估指标进行有效性评估。

第七章总结与展望，对司法多维评估系统进行总结，并对其不足之处给出具体改进方向。

第二章　技术综述

2.1　Spring 框架

Spring 是一个针对 J2EE 应用的一站式轻量级框架，最早由 Rod Johnson 在 2002 年提出并创建，其产生是为了解决 J2EE 框架在实际生产中存在的种种问题。spring 框架进行优化的两个思路是控制反转 (IoC) 和面向切面编程 (AOP) [21]。控制反转是指将对象创建由开发者转移到系统中，在实现时是通过依赖注入 (DI) 的方式将该控制权交给 IoC 容器，并把对象生命周期的管理托管给容器，这一机制大幅度降低了开发者管理实例对象的复杂性，实现了组件之间的解耦。面向切面编程是对面向对象编程 (OOP) 在时间维度上的补充，通过动态地把代码切入到特定的位置实现系统级功能的抽离从而降低耦合。Spring 为应用开发在表现层、应用层、持久层都提供了相应的服务，使得开发者不需要管理类和资源只需要关注具体的业务逻辑，同时基于策略接口开发者对于具体的 MVC 框架的选择也具有高度的自主性，这些特性使得 Spring 框架广泛应用于企业级应用开发。

2.1.1　Spring Boot 框架

Spring Boot 是基于 Spring4.0 设计的，通过开箱即用、简化配置等方式极大地提升了开发者使用 Spring 的开发体验，将开发者从繁琐的项目配置中解放出来。Spring Boot 框架并不是对 Spring 框架的替代，而是从依赖、配置、部署、监控四个方面简化 Spring 框架 [22]。在简化依赖方面，Spring Boot 集成了大量常用的第三方库配置，在 pom 中添加一个自定义的集成配置就能够达到添加多个依赖的效果，基于这些第三方库的配置 Spring Boot 在简化的依赖同时也规避了依赖包版本冲突、引用不稳定等隐患。在简化配置方面，Spring Boot 引入了 Java Config 的方式来解决 Spring 中繁杂的 XML、Annotation 配置问题，这种方式减少了大量的 XML 配置。在简化部署方面，由于 Spring Boot 内置了 Tomcat 或 Jetty 等 Servlet 容器，所以无需服务器上安装 Servlet 容器，打包成 jar 包后可

直接启动。在简化监控方面，Spring Boot 框架通过使用 spring-boot-start-actuator 提供的服务就可以获取进程的运行参数实现监控功能，但由于其为提供服务发现、注册等功能，故在微服务框架中还需引入 Spring Cloud 配合使用。

2.1.2 Spring Cloud 微服务

Spring Cloud 是提供了一站式解决分布式系统的基础设施的架构方案，其集成了包括服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等功能在内的多种服务框架，这些功能以插拔的形式集成在系统中，开发者在构建时可以按照具体的系统需求引入相应的功能服务，实现低成本、低门槛、轻量级地分布式系统开发。相较于服务化架构 (SOA)，Spring Cloud 将每一个组件独立成对外提供服务的产品，降低服务之间的耦合度，进行整体的服务治理，是微服务架构最佳的落地解决方案。

2.2 Docker 技术

Docker 是 dotCloud 公司开源的应用容器引擎技术，其利用 LXC 技术来实现与虚拟机 (VM) 相类似的功能 [23]，但相较于 VM 在硬件层面上的虚拟，Docker 提供的虚拟化技术是建立在操作系统层面上的，由于其占用的资源远少于虚拟机，可同时启动 Docker 容器数量远远高于在一台机器上可同时启动的虚拟机数量。

镜像和容器是 Docker 中两个很关键的概念，镜像是一种分层结构，它包含了容器运行所需要的代码及所依赖的其他组件，将应用构建成镜像时，会按照 Dockerfile 文件提供的指令分层构建，前一层构建完毕才会构建后一层，并且每一层镜像在系统中都是可复用的，充分利用系统资源降低系统存储负担。而容器是镜像的“实例”，容器的构建是在镜像的基础上增加一个可写层，这个可写层记录了容器运行时的各种变量，实现动态运行的效果。无论是镜像还是容器都可以以文件的形式进行持久化，同时以文件为载体导入到其他的系统中。

Docker 主要解决环境配置问题，通过 Docker 可以保证在软件生命周期中各阶段版本和依赖的统一，因此提供了持续部署的能力。同时 Docker 将容器中运行的进程与宿主机中的进程进行隔离，容器中的服务崩溃不会对宿主机中的

其他服务造成影响，这种高内聚的设计机制使得服务具有很好的隔离性。

2.3 FastDFS 技术

FastDFS 是一个开源的轻量级文件管理系统，提供了在分布式系统文件管理所需要的功能，fastDFS 设计并实现了线性扩容、冗余备份、负载均衡等机制 [24]。相较于广泛应用在大数据领域中处理海量数据的 Hadoop 分布式系统 [25]，FastDFS 在管理体量较小的数据时更有优势，在存储时 FastDFS 不会对文件进行切分，从而保证每个存储节点中文件的完整性。在 FastDFS 运行过程中服务器集群可以划分为三个角色，分别为 Tracker Server、Storage Server 及 Client。

在三种角色中，Storage Server 是具体存储文件的服务器载体，该文件由 Client 上传，在 FastDFS 中会将 Storage Server 分为不同的组 (group)，在每个组中多个 Storage Server 会自动进行同步，在一段时间后同一个组中的节点存储了相同的文件，而文件上传到 FastDFS 系统时由 Tracker Server 进行调度和管理，决定文件具体上传的 group，选定 group 后 Tracker Server 生成文件存储的路径，在 group 中通过负载均衡上传至其中一台存储服务器中，之后由线程进行同步，并将同步进度上传给 Tracker Server。连接服务的发起都是由 Storage server 进行的，Storage Server 会周期性的反馈自己的状态，Tracker Server 据此分配存储任务，Tracker Server 只负责划分组，Storage server 决定自己挂载在哪一个组中，从而实现了线性变异的机制。

FastDFS 技术的使用可以降低系统中文件管理的复杂度，使得系统开发更加专注于业务逻辑，同时该技术的文件同步等特性也为系统中的关键步骤提供了解决方案。

2.4 自然语言处理技术

自然语言，即人们交流所使用的语言，是没有办法直接被计算机程序所理解的，自然语言处理技术 (Nature Language Processing,NLP) [26] 的出现就是为了解决这一困境。本文所研究的基于司法多维的评估系统包括基于已有的司法案例文本的数据变异，其中司法案例文本通过中文语言进行描述，对案例文本的处理包括中文文本分词和依存句法的分析。

2.4.1 中文文本分词技术

分词技术是自然语言处理的基础，计算机要理解语句的含义首先要获取组成该语句的词语，分词就是将语句划分成词语的过程。中文语句从直观意义上来说是由字组成的，但是单个的字不能反映中文语句的具体含义，词语才是构成理解和分析其语义的粒度。

中文语句的词语分割没有确定的规则，不同的分割方法可能会导致语句的含义截然不同，造成分词歧义的问题。除此之外，中文词语有一词多义的特性，词性的变换和语境的差别都可能改变词义，例如在“她原来很好”这句简单短语中，“原来”这个词作为副词表示她很好的意思而作为名词则表示她以前很好的意思。因此中文在语义和语法的复杂性给中文文本分词造成了一定的困难。

目前中文分词技术主要从词典、统计和规则三个方面进行实现 [27]。基于词典的分词对语句中的字序列与词典中的词进行匹配，语句扫描方向和字匹配长度对分词结果有比较大的影响，由于中文词语本身没有标准的定义，故无法构建标准统一的词典，因此基于词典的分词在实际场景的使用中具有较多的限制。基于统计的分词是通过机器学习模型学习已有的语句分词经验来达到对未分词的语句进行分词的目的，在上下文中两个字组合出现的次数越多则将其划分为一个词语的概率也就越高，主要的统计方法有 N 元文法模型、隐 Markow 模型和最大熵模型等 [28]，在实际使用时通常会结合基于词典的分词方式。基于规则的分词方式是通过模拟人脑对语句的理解过程来达到划分的效果，语句的句法、语法以及语义是理解分析语句信息的关键，专家系统分词法和神经网络分词法是目前基于规则分词的两种落地方式，也是基于规则的分词方式的发展方向。

为了将司法案例文本转化成模型可以识别的结构性数据，本文使用哈工大语言技术平台 (Language Technology Platform, LTP) 作为分词工具 [29]，LTP 基于结构化感知器 (Structured Perceptron, SP)，使用最大熵的统计方式进行建模，在解决切分歧义、变异新词方面都有很好的表现。

2.4.2 依存句法分析

依存句法是一种用词语之间的关系来反映句子结构的方法，其最早的现代理论模型由法国语言学家 Lucien Tesniere 在二十世纪提出 [30]。在这种句子结

构中，对处于相邻位置的词与词之间的支配-从属关系进行划分，处于支配地位的词通常被处于从属地位的词修饰和限制。在句子中这种支配-从属关系类似于一种依存关系，因此这种句法关系被称为依存句法，依存关系根据从属词所起到的作用不同具体又可以细分为多种不同的类型，常用的部分类型如表 2-1所示。

表 2-1: 依存关系类型表

关系类型	缩写	含义
主谓关系	SBV	Subject-verb
动宾关系	VOB	Verb-object
定中关系	ATT	Attribute
前置宾语	FOB	Fronting-object
状中关系	ADV	Adverbial
并列关系	COO	Coordinate
核心关系	HED	Head

一般来说，句子的语义结构是没有办法直接通过结构性数据准确表达其含义的，研究句法结构是为了将句子的深层语义结构通过表层的句法结构的状况及条件表现出来，依托于词语之间确定的依存关系，句子的句法结构可以很容易地转化成适合处理的结构性数据。在依存句法中，可以将词语视为节点，从属-支配关系视为一条有向边，则通过一个有向图就可以表示句子的依存句法结构。在有向图结构的基础上，语言学家 J.Robinson 通过以下四条规则将句子依存句法结构的表示转化为树结构 [31]：

(1) 句子中有且只有一个词不从属于任何词，被称为中心词。
(2) 除中心词外，不存在不从属于其他词的词。
(3) 处于从属关系的词语不能从属于其自身。
(4) 位于两个词语之间的词只能直接或间接地从属于这两个词语中的其中一个。

第一条规则保证了单一中心词作为有向图的中心；第二条规则保证了有向图的连通性，并且该有向图包含所有的词；第三条规则保证了该有向图中不存在环。基于这三个限制有向图被转化成了一颗有向树，而第四条规则可以保证该有向树不存在相交弧，因此该有向树又被称为投影树。图 2-1展示了由 LTP 对“北京冬奥会让来自全世界的朋友都齐聚北京”这句话进行句法分析，并使用 Graphviz 可视化工具展示依存句法树的结果。

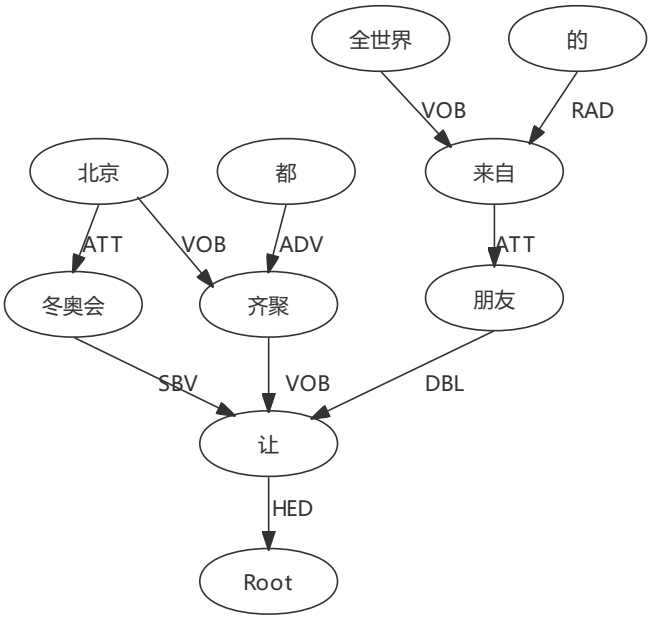

图 2-1: 依存句法树示例

在图 2-1中通过箭头将具有依存关系的节点连接起来，被箭头指向的词是支配词，而箭头另一端则是从属词，而箭头上的标签则代表了依存关系的具体类型，例如在例句中的“北京”与“冬奥会”的箭头标签“ATT”代表两个词语之间的依存关系是定中关系。通过查找“SBV”标签可以找到句子的主语是“冬奥会”，谓语是“让”，同时“让”也是句子的核心词，其指向了依存树的根节点“Root”。通过依存句法树可以很直观的看到句子整体的句法结构。

2.5 TF-IDF 算法

TF-IDF 全称 Term Frequency-Inverse Document Frequency [32]，中文含义是“词频-逆文件频率”，用以在特定语料库中评估字词在其所在文本中重要性的统计方法。

一般来说，一字词在其所在文本中出现的频率越高表示该字词越能反映该文本的主题，也就是相对于该文本来说该字词越“重要”，这是 TF-IDF 算法前半部分 TF(Term Frequency, 词频) 的含义。但是字词的重要性反过来又会受到其在语料库中出现的频率，如果一字词在语料库中出现的频率越高那么该字词对文本的“重要”程度就越低，这就是 TF-IDF 算法后半部分 IDF(Inverse Document Frequency, 逆文件频率) 的含义。因此考虑字词对文本重要性时需要

综合 TF 和 IDF 两方面的影响，对于词语 w 来说，其 TF-IDF 值的计算公式如下所示:

$$TF-IDF(w,d,D)=TF(w,d)\times IDF(w,d,D) \tag{2-1}$$

其中，D 表示语料库，d 表示词语 w 所在的文本，且 $D=\{d_i \mid i\in N且i>0\}$

$TF(w,d)$ 表示词频的计算公式，其具体的计算方式如下所示:

$$TF(w,d)=\frac{count(w)}{\sum_{v\in d} count(v)} \tag{2-2}$$

$count(w)$ 用以表示该词在文本中出现的次数，词频计算中除以 $\sum_{v\in d} count(v)$ 进行归一化处理。

$IDF(w,d,D)$ 表示的是一个词的逆文本频率，若文本中一个词的次数越多表示该词对文本的重要性程度越低，因此是一种词的频率与其在文本的重要性是一种反比关系，计算公式如下所示:

$$IDF(w,d,D)=\log\frac{D}{1+\sum_{d\in D} TF(w,d)} \tag{2-3}$$

D 表示语料库中文本的数量，从公式中可以看出当一个词语在语料库中的词频越大时分母也就越大，分母越大导致取对数的值就越小，对数值也就越小且越接近于 0。

综上只有当某一词其所在文本中的词频较高，在语料库中的词频较低时才能得到较高的 TF-IDF 值，表示该词对文本重要性较高。

2.6 本章小结

本章主要介绍了基于司法多维评估的模型评估系统所使用的相关技术，主要包括搭建评估系统所使用的技术框架、技术方案以及在多维评估时使用的文本处理技术。首先在搭建系统整体框架中介绍了后端框架 Spring Boot 以及微服务框架 Spring Cloud，之后对选择分布式文件管理实现方案 FastDFS 作了介绍，同时对作为评估模型载体的 docker 技术做了简要的介绍，最后对司法多维评估系统中使用到的中文文本相关的处理技术进行了说明，包括文本分词技术、依存句法分析以及反映词在文本中重要性的 TF-IDF 算法。这些技术的介绍为接下来搭建基于司法多维评估的模型评估系统提供技术支撑和理论依据。

第三章　系统需求分析与概要设计

3.1　系统总体规划

本系统主要在比赛场景下针对刑期预期分类系统进行司法多维评估并对模型进行分阶段筛选，图 3-1展示了多阶段评估筛选流程。

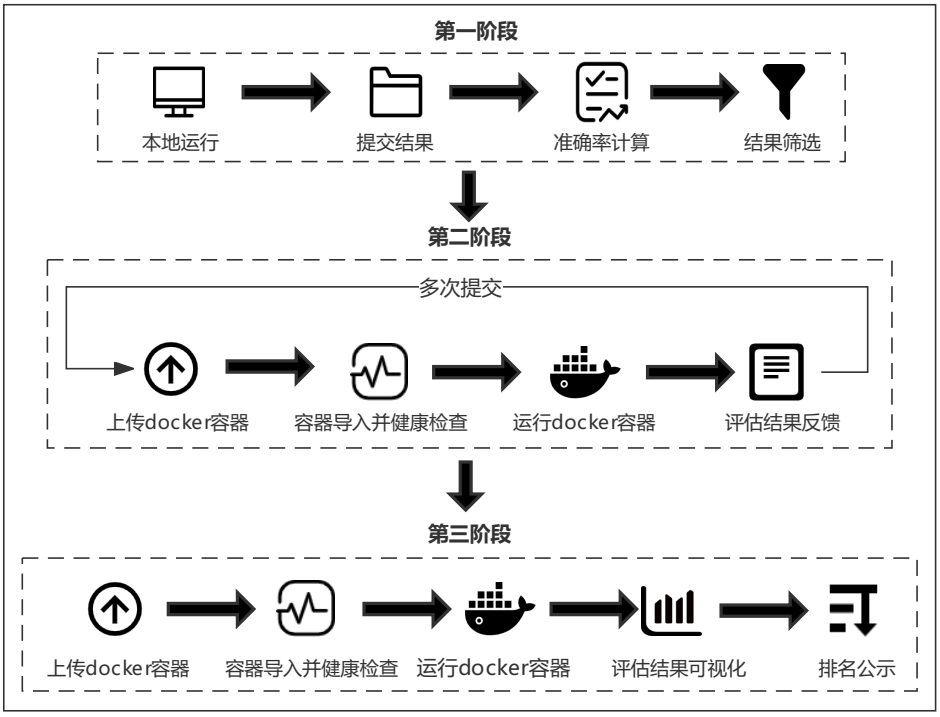

图 3-1: 分阶段评估筛选流程图

第一阶段系统对参赛选手的模型进行初筛，司法多维评估系统提供由案例训练集和测试集，训练集包括案例文本描述和分类结果，而测试集只包含案例文本描述，参赛选手在评估平台上下载案例训练集对模型进行训练后使用评估系统提供的案例测试集进行测试，并将模型的分类结果上传至系统，系统根据分类准确率进行初筛，对于准确率达到基准线之上的模型进入到下一阶段的评估中；第二阶段参赛选手需要上传模型打包后的 docker 容器（系统对模型打包过程提供指导说明），系统导入相应的 docker 容器并进行健康检查（运行容器

中测试命令，检查容器在宿主机中的运行状况和读写能力），在管理员设置的时间节点使用指定变异数据集批量运行容器进行预测，根据预测结果和所使用的变异数据集获得模型的司法多维评分，将评分结果通过前端的可视化反馈给参赛选手，选手可以按照评估结果调整模型，之后在指定时间范围内多次提交模型容器，在第二阶段最后一次提交中具有良好表现的模型将进入第三阶段的评估。第三阶段的流程与第二阶段并无二异，但在第三阶段中具有参赛资格的参赛选手只有一次提交模型的机会且第三阶段所使用的数据集较第二阶段复杂度更高（通过改变相应数据变异算法的参数实现），根据第三阶段的评估结果会获取最终的模型排名。

本系统由网关模块、认证授权模块、文件管理模块、阶段管理模块、数据集管理模块、容器管理模块和异构服务模块七个模块组成。网关模块对负责系统中的请求进行拦截和分发，结合认证授权模块对系按照用户角色对用户进行权限限制，配置网关服务；文件管理模块对系统中上传的文件进行管理包括原始的数据集、容器文件等，提供断点续传、文件下载、文件不重传等功能；阶段管理模块对比赛管理员开发，主要提供设置阶段使用的训练集和测试集、开始提交和截至提交时间点、容器批量运行周期等功能；数据集管理模块管理数据集的增删改查以及数据集的变异；容器管理模块对用户上传的容器进行导入、健康检查、运行、生成评估报告等操作。异构服务模块为系统其他模块提供访问异步服务的接口，异步服务分为本地数据集处理异构服务和 storage 节点中的容器处理异构服务，不同模块根据具体的需求调用服务。

3.2 系统需求分析

3.2.1 系统功能性需求

人工智能技术与司法行业的深度结合促进了司法领域现代化和智能化进程，但人工智能模型所隐含的“算法黑箱”却阻碍了“可视正义”的发展。一些司法辅助办案系统在具体的使用场景中会受到相关方的质疑，影响系统产出结果的权威性。司法多维指标就是总结司法关注的重点问题对模型进行评估，通过全面、完备的评估体系解决“算法黑箱”的问题，从而保证“可视正义”。

司法多维评估系统的核心功能是对刑期预测分类模型进行司法多维指标的评估，司法多维的评估指标包括准确性、泛化性、可参考性、鲁棒性、公平

性。模型评估依赖数据集的质量，本文采用 CAIL2018 [33] 提供的司法案例文本数据作为原始数据集。在评估时不同的评估指标依赖具有不同特性的数据集，因此引入多种数据变异技术对原始数据集进行相应的特征变换。

本系统不适合为单个司法模型提供评估服务，而是在自动化流程下批量筛选和评估模型，适用于比赛场景。系统以 Web 形式提供服务，面向的用户角色包括参赛选手和比赛管理员，针对这两种角色系统需要提供权限管理，确保整个流程安全性。参赛选手提交自己训练好的模型到系统中，根据系统的评估结果优化模型，系统分阶段筛选总评分较高的模型，在第三阶段获取模型评估排名。比赛管理员负责管理各阶段的数据，设置相应时间节点，系统需要根据管理员设置的特定的时间节点与运行周期触发相应事件，完成自动化流程。

为保证模型具有持续部署的能力，用户提交的模型应按照规定的方式打包成 docker，再将 docker 容器导出为压缩文件后上传到系统中。容器上传后进行负载均衡计算在多台宿主机中选择一台进行导入，导入后需要运行对应的健康检查接口，保证容器的读写能力正常，健康检查的结果及时反馈给用户，健康检查有问题的容器应通知用户按照规定重新上传，确保后续批量测试容器的稳健性。

除了上述所提到的司法多维指标评估功能、数据变异功能、权限管理功能、阶段管理功能、自动化流程管理功能、docker 管理功能以外，系统还需提供基础的文件管理模块，数据集文件、容器压缩文件都需要上传到系统并对其进行管理，同时系统应基于易扩展的多服务器架构，故需要提供分布式管理文件的能力。系统的功能性需求列表如表 3-1所示：

表 3-1: 基于司法多维的模型评估系统功能性需求列表

需求 ID	需求名称	需求描述
R1	数据集下载	用户通过点击各阶段流程页面的下载按钮获取相应的训练集或测试集。
R2	新增结果集	在第一阶段流程中，用户可在阶段提交时间范围内提交结果集文件，提交后系统进行结果集解析，通过与标准结果集进行对比后获取结果集准确率，对于解析失败的结果集，系统支持在提交时间范围内再次提交。

R3	新增容器	在第二和第三阶段流程中，用户可在阶段提交时间范围内多次或单次新增容器，容器新增后系统将其上传至 fastdfs 文件管理系统中，并在同步之后进行导入和健康检查操作，将导入和健康检查结果反馈给用户，若导入或健康检查失败用户可以在该提交周期内再次提交，直至容器健康检查成功。
R4	容器运行及评估	对阶段中上传的容器进行批量运行及评估，该任务由系统在提交时间截止或提交周期截止时自动触发，容器评估后可在系统中查看相应的评估报告
R5	新增数据集	比赛管理员可在系统中新增数据集，系统接收数据集后对其进行初始化，初始化后生成该数据集对应的测试集和结果集文件，初始化状态需要反馈给用户，若数据集初始化失败，可在对应的管理页面查看详细的失败原因。
R6	数据集变异	比赛管理员可对系统中初始化成功的数据集进行变异，变异类型包括鲁棒性和公平性，每种类型中包含多种变异方法，系统基于设定的变异参数进行变异，若变异失败，可在对应的管理页面中查看失败原因。
R7	阶段规则设置	比赛管理员可在各阶段的规则管理中设置该阶段对应的规则，规则说明以富文本的形式进行展示，设置后参赛选手可在对应阶段查看阶段规则。

R8	阶段时间设置	比赛管理员可在各阶段的比赛管理中设置阶段时间，阶段时间分为阶段开启时间、提交开启时间、提交截止时间、结果公示时间及提交周期，其中前四项为必填项，在时间节点生效后不能进行更改，最后一项提交周期为选填项，在提交次数设为多次提交时可设置提交周期。阶段时间设置后在对应的时间节点系统自动更新阶段状态，参赛选手基于阶段状态完成相应任务。
R9	阶段基准线设置	比赛管理员可在各阶段的比赛管理中设置基准线，阶段基准线分为两类，一类是按照分值，一类按照比例。按照分值是指参赛选手在该阶段提交的结果集的准确率或容器的评估分数达到该阶段设置基准值后即可进入下一阶段；按照比例指参赛选手的评估分数在该阶段内所有有效提交中排名在该基准比例之上即可进入下一阶段
R10	阶段数据集设置	比赛管理员可在各阶段的比赛管理中设置数据集，数据集设置包括选择该阶段训练集、该阶段测试集、测试集是否公开、鲁棒性测试集及公平性测试集，阶段可使用的数据集必须是成功初始化后的数据集，同时鲁棒性测试集和公平性测试只能在测试集选择不公开的情况下在已选择的测试集所包含的鲁棒性测试集和公平性测试集中进行选择

3.2.2 系统非功能性需求

司法多维评估系统的非功能性需求如表 3-2所示，非功能性需求从司法多维评估系统所要求的时间特性、可扩展性、并发性、安全性、可靠性进行约

束，非功能性需求的实现效果对用户使用体验、系统的生命周期都有很大的影响。

表 3-2: 系统非功能性需求列表

非功能性需求 ID	非功能性需求具体描述
评估系统的时间特性	系统管理员和参赛选手操作页面跳转时响应时间不超过 2s
	参赛选手在上传容器压缩文件时，异构服务导入结果反馈不超过 10s
可扩展性	多维评估系统可进行负载均衡，在参赛选手或提交容器数过多时可以通过增加服务器实现扩展，以防止系统崩溃。
并发性	多维评估系统中的接口至少可以接受 10 个用户同时访问。
安全性	多维评估系统应对敏感性数据加密后进行持久化。
可靠性	多维系统发生系统故障时可及时追溯和恢复。

3.2.3 基于司法多维的模型评估系统用例设计

按照系统权限不同本系统的用户角色主要分为：参赛选手和比赛管理员。参赛选手下载每个比赛阶段的相应训练集，按照各阶段比赛要求提交结果集或模型容器，提交后在特定的时间节点可查看评估报告及在当前阶段评估排名的结果；比赛管理员主要设置各比赛阶段的相关数据，包括指定数据集、时间节点、评估周期、进行下一阶段基准线、阶段规则等信息，另外比赛管理员还需要对整个系统中的数据集进行管理，包括增删改查数据集及变异数据集。系统用例图如图 3-2所示，共涉及到 9 个用例包括数据集下载、新增结果集、新增容器、容器运行及评估、新增数据集、数据集变异、阶段规则设置、阶段时间设置、阶段基准线设置、阶段数据集设置。

系统将整个评估筛选流程分为三个阶段，各阶段的提交物和规则有所差别。第一阶段比赛管理提供训练集和测试集供参赛选手下载，选手下载案例训

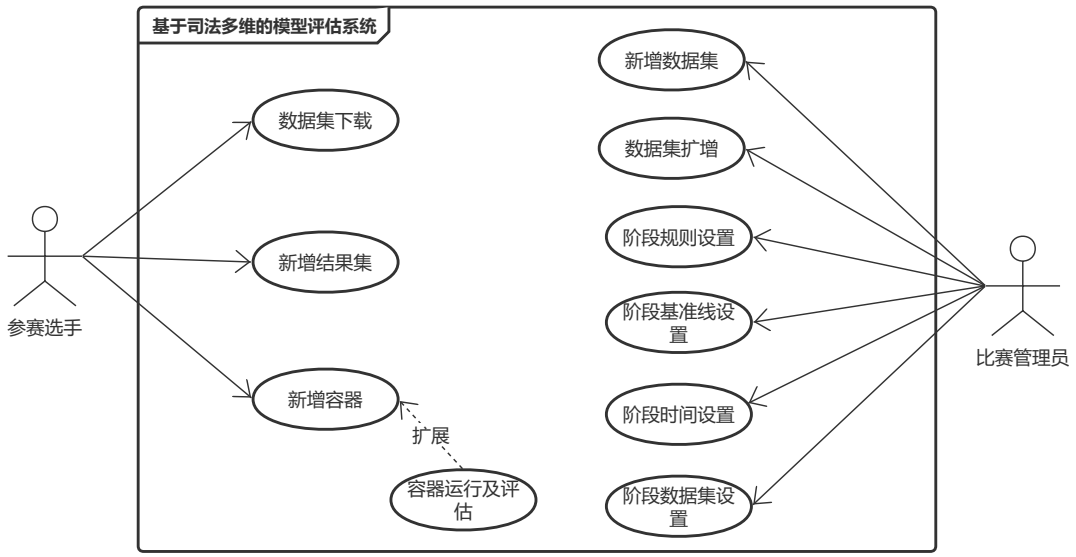

图 3-2: 基于司法多维评估的模型评估系统用例图

练集后对案例分类模型进行训练，之后在本地用下载的案例测试集测试模型获得案例分类预测的结果集文件，用户将结果集文件上传至系统，系统计算结果集文件的准确率，计算完成后用户可在“我的结果集”页面中查看提交结果集的准确率，在第一阶段到达结果公示时间后公布准确率排名，对准确率达到基准线的参赛选手筛选进入第二阶段。参赛选手在第二阶段只可以下载训练集，选手在本地训练模型后用 docker 进行包装（按照系统要求设置启动命令及健康检查命令、读取数据集所在的目录），将 docker 容器导出成压缩文件后上传至系统，在第二阶段的截止提交时间之前可以进行多次提交，但在特定时间周期内有效提交次数为一次，例如将提交周期节点设置为周三和周日，那么从本周的周三到周日可以进行一次有效提交，从本周周日到下周周三也可以进行一次有效提交，有效提交是指提交的容器可以在系统中导入和健康检查成功，若导入和健康检查失败可再次提交直至成功。具体提交规则和时间节点由比赛管理员设置，在每次周期截止提交时间到达之后，系统批量运行该周期内提交的容器，生成评估报告，参赛选手可及时下载模型评估报告，基于报告调整下一周期内提交的模型，第二阶段排名结果以在该阶段所有有效提交的容器评估总分为准。第三阶段与第二阶段的区别是在整个提交时间范围内只有一次有效提交容器的机会，第三阶段是最后一轮的筛选，根据第三阶段的评分结果和第二阶段评估成绩得到最终的评估排名。下面将给出上述流程中涉及的各个用例详细描述。

表 3-3描述了训练集下载的用例详情，用户登录系统后可以查看目前已开放的比赛阶段，若比赛管理员在对应阶段的管理页面设置训练集后用户可以下载该阶段案例训练集到本地，若管理员设置开放测试集时用户也可在比赛阶段中下载案例测试集。

表 3-3: 数据集下载用例描述表

用例 ID	UC01	用例名称	下载数据集
用例描述	用户进行下载数据集操作		
用例参与者	参赛选手		
触发条件	用户点击某阶段训练集下载按钮		
前置条件	比赛管理员已指定该阶段数据集，且该阶段训练集或测试集已开放下载		
后置条件	系统反馈训练集下载状态		
优先级	高		
正常流程	1. 参赛选手登录系统 2. 用户进入某阶段比赛流程页面 3. 用户点击下载训练集或测试集按钮 4. 训练集下载至用户本地		
扩展流程	1a. 比赛阶段暂未开放，无法进入对应阶段 2a. 提交开启时间未到，无法下载测试集		

表 3-4描述了结果集文件上传用例的详情，参赛选手需要在第一阶段提交结果集文件，结果集文件是用户在系统上下载最终测试集后在本地运行模型获得的预测结果集合，结果集文件应以文本文件 (以.txt 结尾) 的形式提交到系统中，对结果集文件的评估只基于准确率一个指标，第一阶段主要起到对模型进行初筛的效果，降低系统后期模型评估的任务量，对于提交的结果集未达到基准值的用户无法进入第二阶段。

表 3-4: 新增结果集用例描述表

ID	UC02	名称	新增结果集
描述	用户进行新增结果集操作		
参与者	参赛选手		
触发条件	用户点击某阶段训练集下载按钮		

前置条件	第一阶段提交结果集已开放
后置条件	系统反馈结果集评估结果
优先级	高
正常流程	1. 用户进入第一阶段流程页面 2. 用户进入点击下一步进入第二个提交结果集流程 3. 用户点击提交结果集按钮，选择结果集文件提交 4. 用户进入我的结果集 tab，查看已上传的结果集文件状态 5. 用户刷新页面，查看结果集评估结果
扩展流程	1a. 比赛阶段暂未开放，无法进入第一阶段 2a. 阶段提交暂未开放或已由提交有效的结果集，无法提交结果集 3a. 结果集文件不是 txt 文本文件，系统不支持上传 5a. 结果集解析失败，无法查看评估结果，需重新上传

表 3-5描述了容器压缩文件上传用例的详情，用户上传的容器压缩文件应以.tar 为后缀，在上传至业务服务器后需上传至 fastDFS，上传成功后返回其存储的路径，基于路径确定 storage 节点所属的 group 和服务器，通过该名称调用对应宿主机的 docker 命令导入 docker 容器，导入成功后执行健康检查命令，若上述任意步骤执行失败用户可以重新上传容器压缩文件。

表 3-5: 新增容器用例描述表

ID	UC03	名称	新增容器
描述	用户进行新增容器操作		
参与者	参赛选手		
触发条件	用户点击提交容器压缩文件按钮		
前置条件	阶段提交容器已开放		
后置条件	系统反馈容器导入及健康检查结果		
优先级	高		
正常流程	1. 用户进入第二阶段或第三阶段流程页面 2. 用户进入点击下一步进入第二个提交容器流程 3. 用户点击提交容器按钮，在弹窗中填写容器信息及上传容器压缩文件 4. 用户进入我的容器 tab，查看已上传的容器文件状态		

	5. 用户刷新页面，查看容器导入及健康检查结果
扩展流程	1a. 比赛阶段暂未开放，无法进入第二阶段或第三阶段 2a. 提交开启时间未到或已提交有效容器，则无法新增容器 3a. 容器文件不是 zip 类型文件，系统不支持上传 5a. 容器导入或健康检查失败，需重新上传
特殊需求	新增容器后应将容器上传至 fastdfs 文件管理系统，待同步完成后进行导入和健康检查操作

表 3-6描述了容器运行及评估用例详情，容器运行和评估是在提交截止时间到达后系统自动触发完成的，评估结束后将评估结果更新到数据库中，用户可查看和下载自己提交的容器评估报告。

表 3-6: 容器运行及评估用例描述表

ID	UC04	名称	容器运行及评估
描述	系统进行容器运行及评估操作		
参与者	系统		
触发条件	到达第二阶段提交周期节点及到达第三阶段提交截止时间		
前置条件	容器导入成功并通过健康检查		
后置条件	系统反馈容器评估报告		
优先级	高		
正常流程	1. 系统触发容器批量运行和评估任务 2. 系统执行容器批量运行脚本 3. 系统根据容器的运行生成的结果集文件运行评估脚本 4. 系统将评估结果写入数据库中 5. 提交截止时间到达后的一段时间内，用户可查看和下载容器评估报告		
扩展流程	1a. 提交周期节点或提交截止时间未到系统无法触发容器批量运行和评估任务 2a. 容器运行失败，记录容器运行失败信息，写入数据库中 3a. 若容器运行后未生成结果集文件，则不进行评估操作		

	5a. 对于运行和评估失败的容器，系统不提供评估报告，可查看相应的错误信息
特殊需求	新增容器后应将容器上传至 fastdfs 文件管理系统，待同步完成后进行导入和健康检查操作

表 3-7描述了新增数据集用例的详情，比赛管理员可以在系统中上传原始数据集，数据集上传后系统自动对文件进行初始化操作，生成测试集和结果集，若初始化失败则该数据集不能在系统中使用。

表 3-7: 新增数据集用例描述表

ID	UC05	名称	新增数据集
描述	用户进行新增数据集操作		
参与者	比赛管理员		
触发条件	用户点击新增数据集按钮		
前置条件	操作用户为比赛管理员		
后置条件	系统反馈数据集上传及初始化结果		
优先级	高		
正常流程	1. 用户进入数据集管理页面 2. 点击新增数据集按钮，在弹出的新增数据集弹窗中上传数据集文件，填写数据集名称及描述等相关信息 3. 点击确定按钮，数据集管理列表刷新，插入新增数据集信息 4. 点击刷新页面可查看数据集初始化状态，初始化完成后可查看结果集和测试集存储路径		
扩展流程	2a. 提交的文件需按照规定的类型，支持 json 和 txt 文件 4a. 数据集初始化失败可查看失败原因		
特殊需求	数据集新增后可进行创建词典文件的操作，创建后列表显示词典文件、向量文件路径		

表 3-8描述了数据集变异用例的详情，比赛管理员在数据集变异页面中选择初始化成功后的原始数据集进行变异，选择变异类型和变异方法，输入变异参数后提交至系统中进行相应的数据集变异任务，单个原始数据集的变异次数不限。

表 3-8: 变异数据集用例描述表

ID	UC06	名称	变异数据集
描述	用户进行变异数据集操作		
参与者	比赛管理员		
触发条件	用户点击变异数据集按钮		
前置条件	操作用户为比赛管理员		
后置条件	系统反馈数据集变异结果		
优先级	高		
正常流程	1. 用户进入数据集管理页面 2. 点击数据集变异按钮，跳转至数据集变异页面 3. 选择待变异的原始数据集 4. 选择变异类型，包括鲁棒性和公平性两种 5. 选择变异方法并填写相关参数		
	6. 点击确定按钮，跳转至数据集管理页面查看数据集变异信息		
扩展流程	2a. 无法选择初始化失败的数据集作为待变异数据集 5a. 当前变异类型下所有的变异方法设置变异数据比例之和不得超过 100%		
特殊需求	执行数据集变异任务时若指定的变异方法需要使用该数据集的词典文件和向量文件则应先生成相应的词典文件和向量文件并将生成的文件路径存储至数据库中		

表 3-9描述了阶段规则设置用例的详情，比赛管理员须在各阶段中设置规则说明，规则说明以富文本的形式保存在系统中，在参赛选手进入对应阶段时可查看该阶段的相应规则，包括提交形式、评分方式等信息。

表 3-9: 阶段规则设置用例描述表

ID	UC07	名称	设置阶段规则
描述	用户进行设置阶段规则操作		
参与者	比赛管理员		
触发条件	用户新增阶段规则说明		
前置条件	操作用户为比赛管理员		
后置条件	以参赛选手身份进入阶段可看到相应规则		

优先级	高
正常流程	1. 用户进入阶段规则页面 2. 在富文本编辑框中修改或新增文本、图片等信息 3. 点击确定按钮，将规则信息更新至数据库中 4. 在阶段开启后，参赛选手可看到该阶段设置的阶段规则

表 3-10描述了阶段时间设置的用例详情，阶段时间设置的目的是自动化控制各阶段流程，时间设置的类型分为多种，包括阶段开启时间、提交开启时间、提交截止时间、提交周期及结果公示时间。系统需监听设置的时间节点，在时间到达后执行特定任务。

表 3-10: 阶段时间设置用例描述表

ID	UC08	名称	设置阶段时间
描述	用户进行设置阶段时间操作		
参与者	比赛管理员		
触发条件	用户点击修改阶段时间按钮		
前置条件	操作用户为比赛管理员		
后置条件	设置阶段时间生效		
优先级	高		
正常流程	1. 用户进入阶段管理页面 2. 点击修改按钮后添加或修改时间节点，时间节点的类型包括阶段开启时间、提交开启时间、提交截止时间、提交周期、结果公示时间 3. 点击确定按钮，系统更新阶段时间，并添加相应的定时任务		
扩展流程	2a. 不能修改已生效的时间节点 2b. 设置提交周期时应设置提交方式为多次，且可指定提交周期节点的准确时间点，若未指定准确时间点则以零点作为默认值 2c. 提交周期无生效一说，修改次数不限		

特殊需求	若修改之前已经添加过的但是暂未生效的时间节点，系统应取消之前的定时任务，同时对在时间节点到达后除了执行特定的任务外，还需将当前时间节点的状态更新为已生效

表 3-11描述了阶段基准线设置的用例详情，阶段基准线由管理员设置。基准线是判断参赛选手是否能进入下一阶段的标准，参赛选手提交的容器或结果集在基准线之上则可以进入下一阶段，否则无下一阶段的权限。在阶段结果公示前必须设置对应的基准线。

表 3-11: 阶段基准线设置用例描述表

ID	UC09	名称	设置阶段基准线
描述	用户进行设置阶段基准线操作		
参与者	比赛管理员		
触发条件	用户点击修改基准线按钮		
前置条件	操作用户为比赛管理员		
后置条件	设置基准线生效，未达到基准线的用户无下一阶段的访问权限		
优先级	高		
正常流程	1. 用户进入阶段管理页面 2. 点击修改按钮 3. 选择基准线类型，并填写对应类型的基准值 4. 点击确定按钮，将基准线更新至系统		
特殊需求	在结果公示时间时查询系统中该阶段所有有效提交后按照基准值设置用户可进入阶段		

表 3-12描述了阶段数据集设置用例的详情，各阶段使用的数据集包括训练集和测试集，训练集始终对用户开放下载，测试集可选择是否对用户开放下载，未开放的测试集可设置相应的变异数据集作为附加测试集来评估容器在该指标上的表现。

表 3-12: 阶段数据集设置用例描述表

ID	UC09	名称	设置阶段数据集
描述	用户进行设置阶段数据集操作		
参与者	比赛管理员		

触发条件	用户点击修改数据集按钮
前置条件	操作用户为比赛管理员
后置条件	设置数据集生效，参赛选手可在阶段流程中下载训练集
优先级	高
正常流程	1. 用户进入阶段管理页面 2. 点击数据集修改按钮 3. 设置训练集、测试集 4. 选择测试集是否公开，若选择不公开可设置鲁棒性测试集和公平性测试集
扩展流程	4a. 设置的鲁棒性测试集和公平性测试集只能在当前测试集中的变异数据集中选择
特殊需求	对于作为测试集的数据集需上传至 fastdfs 中

3.3 系统设计

3.3.1 系统架构设计

本系统架构设计采用前后端分离的设计思想。前端基于渐进式框架 Vue 自底向上逐层搭建，同时引入 Element-ui 公共组件库作为前端页面的基础组件实现方案，前端以简洁明了、安全适用作为基本的设计原则。后端基于 Spring Cloud 微服务框架，提供符合 RESTful [34] 设计原则的接口，按照业务功能对后端模块进行划分，每个功能模块都封装成独立的 Spring Boot 微服务，微服务发现注册等管理服务通过 Eureka 进行治理。为了降低文件管理的成本，引入轻量级分布式文件管理系统 fastDFS，通过 fastDFS 可以达到负载均衡、线性扩容等目的。系统整体的架构图如图 3-3所示。

按照基本的业务功能，后端划分了权限管理微服务、文件管理微服务、阶段管理微服务、数据集管理微服务及容器管理微服务；除此之外，系统还包括 gateway 微服务及 outService 微服务，gateway 微服务基于 zuul 技术实现网关功能，负责拦截所有的前端请求进行服务重分配和监控审查,outService 微服务提供了对访问异构服务的接口，以中间件的方式减少其他模块的配置。

为了保证系统的可扩展性和并发性，系统采用了多服务器架构，前端数据

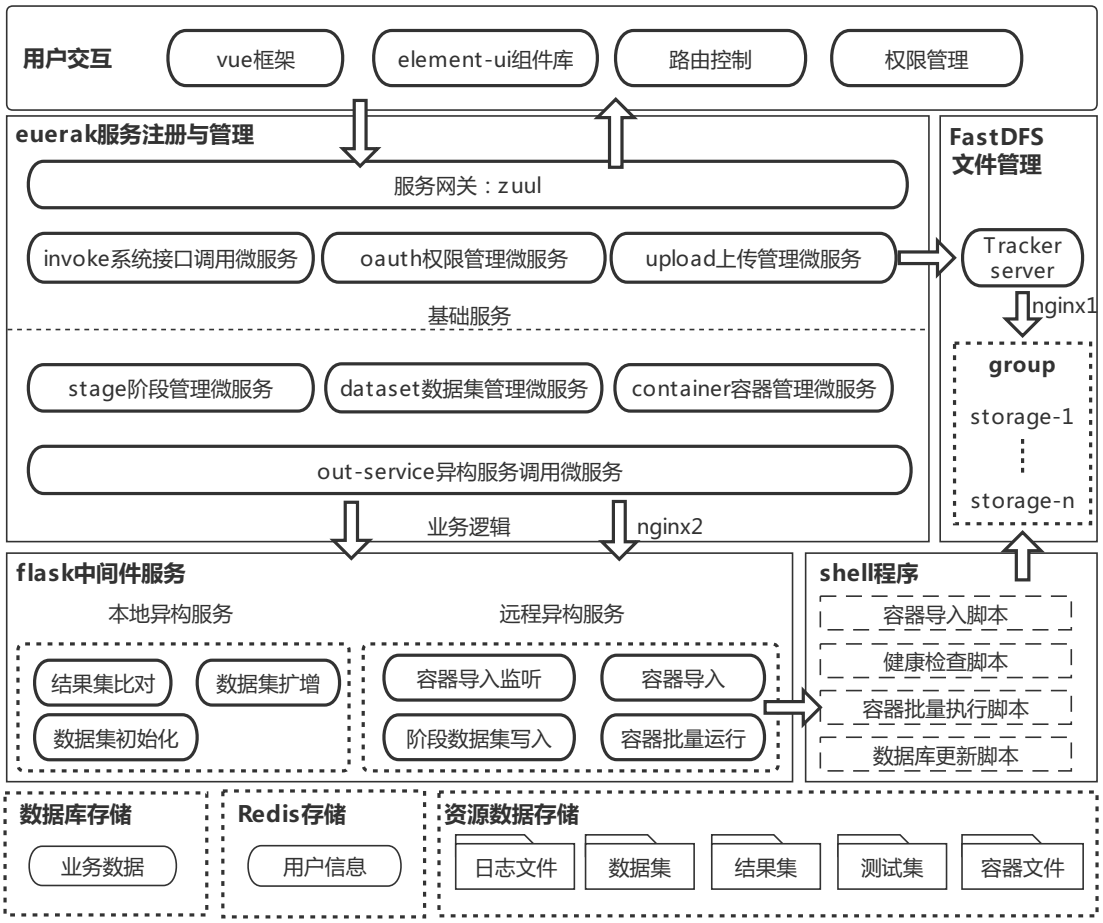

图 3-3: 基于司法多维评估的模型评估系统架构图

交互的基本功能通过 SpringCloud 的微服务集群提供，而核心服务功能例如容器导入、容器批量运行、结果评估则下沉到不同的宿主服务器实现。宿主服务器中使用 flask 框架向上层微服务集群提供核心服务，通过将宿主服务器集群设置为 fastDFS 中的一个 group 实现文件的冗余备份从而达到集群文件实时互通的目的，当用户量增大时，可直接通过增加宿主服务器的数量保证系统并发性。除此之外，在 fastDFS 的 tracker server 中部署了两个 nginx 应用，第一个 nginx 应用通过调度策略为上传到 fastDFS 中的资源选择合适的 storage 初始节点，从而达到负载均衡的目的；第二个 nginx 应用则是利用 url 的 rewrite 功能对业务逻辑服务器发送的请求 url 进行重写，这一方法在发送容器导入请求时及其关键，因为它可以指定只针对初始接收对应容器压缩文件的 storage server 执行容器导入命令，从而使得用户在较短时间内获取容器导入和健康检查的结果，并针对可能的导入或健康检查失败情况做出及时的响应。

3.3.2 基于司法多维评估的模型评估系统逻辑设计

系统的逻辑设计基于上文所述的用例分析及系统的架构设计，系统的逻辑视图如图 3-4所示，该图展示了系统模块的之间依赖关系及对核心外部服务的间接调用关系。

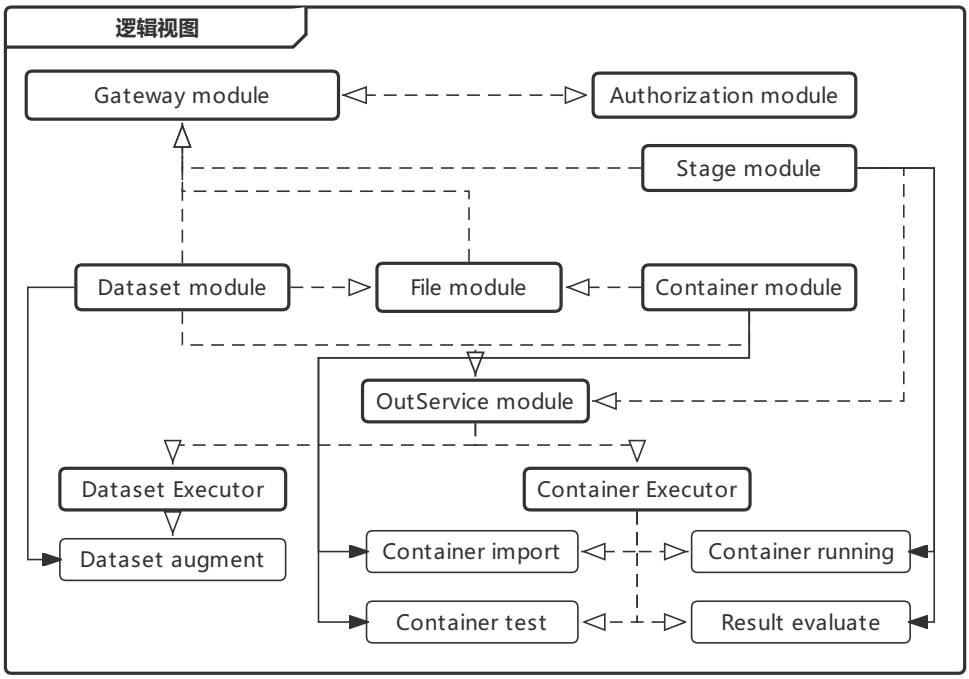

图 3-4: 基于司法多维评估的模型评估系统逻辑视图

系统的逻辑层面由 Gateway module、Authorization module、Stage module、Dateset module、Upload module、Container module、OutService module 七个模块组成：Gateway module 负责接收所有请求进行路由分配，同时该模块内置 Ribbon 的负载均衡功能及 Hystrix 的容错和自我保护功能，Gateway module 保证了系统微服务架构的高可用性；Authorization module 为系统提供用户权限管理服务，通过 Oauth2 引入授权层，向通过身份认证的请求方颁发令牌，对未带有令牌的请求 Gateway module 将对其进行拦截；Stage Module 负责阶段管理任务，包括设置阶段相关数据等功能，除此之外该模块还应提供动态定时任务功能，完成自动化流程控制；Dataset module 是针对系统中的数据集进行管理的模块，包括数据集新增、数据集初始化、数据集变异等功能；Upload module 是文件管理模块，本系统中涉及到的文件管理主要包括结果集文件、数据集文件和容器压缩文件，Upload module 将管理的所有文件视为统一的文件提供上传和下

载等功能；Container module 对用户创建的容器实体进行管理，在容器新增后需将容器上传至 fastdfs 集群中，并将容器信息通过异构服务接口写入容器宿主服务器的待导入清单列表中，等待执行容器导入和健康检查等操作；OutService module 是为其他需要调用外部服务的功能模块提供接口，外部服务主要包括容器导入、容器健康检查、容器批量运行、结果评估及数据集变异。

3.3.3 系统开发设计

系统开发设计从文件组织管理角度体现系统架构，图 3-5展示了系统的开发视图，包括展示层、业务逻辑层、异构服务层三个层面的组织架构。

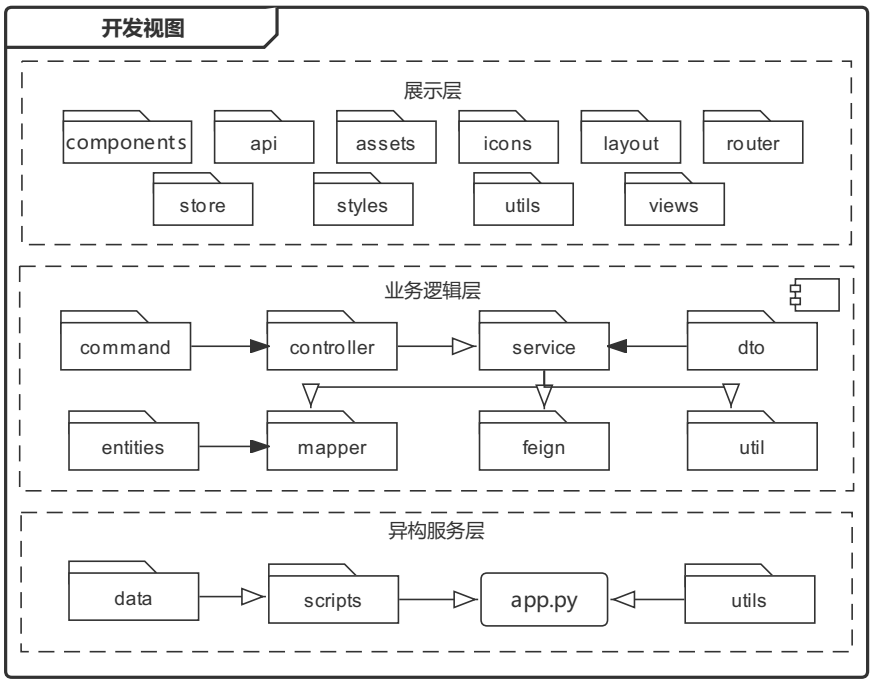

图 3-5: 系统开发视图

展示层是系统的前端部分，由 components、api、assets、icons、layout、router、store、styles、utils、views 七个部分组成。components 中封装了系统复用的组件，例如各阶段中使用的计时组件、文件上传组件、规则介绍编辑的富文本组件等。api 文件夹包含了以模块为粒度构建的请求接口文件，每个文件使用 es6 中的 export 导出具体的请求方法。assets 是展示层的资源文件夹，存放背景图片等资源文件。icons 存放系统图标，图标包括 SVG 形式及 Css 形式。layout 存放与系统框架搭建有关的 Vue 文件，包括系统的顶部导航栏、侧

边导航栏。router 处理前端页面跳转的路由，包括静态路由和动态路由，在用户登录后按照权限生成动态路由，侧边导航栏的选择项取决于具体的动态路由项。store 存放 Vuex 相关文件，Vuex 存放了应用中的大部分状态，例如用户的登录状态等。styles 存放展示层全局的样式文件，采用类 css 语言 sass 来实现。utils 中提供前端所需的工具类文件，包括各类请求的基类 request、token 读取、文件下载等功能。views 中包含了前端所有的页面文件，以模块为单位将具体页面文件组织到文件夹中，同时该文件夹中还包括了该模块中公用的组件文件夹。

业务逻辑层是系统的后端部分，开发视图展示了各微服务中的文件组织形式，按照 MVC 架构风格分为 controller、service、mapper、command、dto、entities、feign、util 八个部分。controller 实现对当前模块的业务逻辑控制，根据路径匹配和拦截请求。service 是业务层逻辑的具体实现，上层被 controller 类调用，下层依赖 mapper 类。mapper 提供对持久化层的访问，通过 mybatis 实现 POJOs 映射。command 中封装了与前端交互的常用数据对象，commad 对象包含了前端传递给后端处理的实体类部分字段，降低了代码的冗余度。dto 的功能与 commad 相对应，其封装了后端需要在响应中传递给前端的对象。entities 包含了模块所有的实体类，每个属性使用 @Column 自动匹配数据库中的字段，对于实体类中的相同字段会将其抽离到实体基类，然后该模块的所有实体类继承自该基类。feign 提供了访问其他微服务模块的接口，通过注解的方式注入 feignClient，在 service 层调用微服务具体服务接口。util 包含系统共用的工具类，例如文件处理、密码加密、缓存等相关功能。

异构服务层提供下沉到宿主服务器的容器导入、容器健康检查、容器批量运行、结果评估等服务，基于 gunicorn+flask 提供接口，app.py 文件中对请求进行拦截和处理，scripts 中包括执行具体服务的脚本文件，data 中包含脚本运行需要的资源文件，utils 中存放 python 语言实现的工具类文件。

3.3.4 系统进程设计

进程视图关注上述逻辑视图中的工作细节，本系统主要涉及对数据集和容器两类资源的管理，本节将从这两类资源的生命周期来描述系统的进程设计。

图 3-6通过时序图的方式刻画了系统中数据集处理的进程视图，数据集在系统中的流转主要包括数据集上传和数据集变异两个操作。数据集上传的是将数据集上传至业务逻辑服务器中，上传后首先在数据库中新增一条数据集记

录，再异步调用数据集初始化接口，初始化操作主要是解析数据集的标签，生成对应纯文本的测试集文件和由每条数据的分类标签组成的结果集文件，初始化完成后会在数据库中记录初始化的相关信息，若初始化成功则记录在数据库中添加测试集文件和结果集文件的存储路径，并将初始状态修改为成功，若初始化失败则更新初始化状态为失败，并记录初始化失败的原因。可进行数据变异的原始数据集需满足初始化成功的条件，业务服务器根据待变异文件在业务服务器的存储路径获取待变异文件并通过运行在业务服务器上的数据集变异异构服务对数据集进行变异，变异后在数据库中记录文件变异的信息及其存储在业务服务器上的文件路径，数据库记录更新后将变异结果反馈给用户。

图 3-6: 数据集处理进程视图

容器处理进程视图如图 3-7所示。容器文件的管理主要分为容器文件上传、容器导入、容器健康检查、容器运行及容器评估结果等步骤。容器文件上传包括从用户本地将文件上传到业务服务以及从业务服务器上分布式存储到异构服务器集群中，冗余存储用户上传的容器可以避免一台异构服务器宕机对评估造成的影响，存储步骤完成后将容器信息更新到数据库中。容器导入是在容器文件上传后实时导入的，在异构服务器集群中通过负载均衡算法选择一台服务器将容器导入的文件路径信息及容器导入后的名称写入该服务器的待导入容器清单文件中，服务器中由进程对该清单文件进行监控一旦发现该清单文件中有写入就会在特定时间段内执行导入容器的脚本，脚本执行后将容器导入结果写入数据库，在此期间客户端可通过刷新页面获取容器导入状态。

容器健康检查是在导入成功后执行的操作，主要是通过运行 docker 容器的接口判断其与宿主机的文件读写能力是否正常，健康检查的结果也直接由异构服务器写入数据库中。容器批量运行的请求由业务服务器中的定时任务发起，每台异构服务器中都会由清单文件记录在该服务器中通过健康检查的容器，异构服务器接收到请求后会启动批量运行脚本扫描清单文件，运行清单文件中的容器，运行的结果存储到异构服务器中，由结果评估脚本在批量运行结束后根据结果文件基于司法多维评估指标进行评估，评估结束后生成评估报告存入数据库中，客户端需要查看评估结果从数据库中读取评估报告。容器文件上传后的流程控制都由系统完成，虽然容器的操作都在异构服务器中执行但对容器状态的更新和查询都是通过数据库作为介质。在系统批量执行容器运行后需要对数据库中的容器状态进行检查，若存在还未运行的容器（可能由于导入该容器

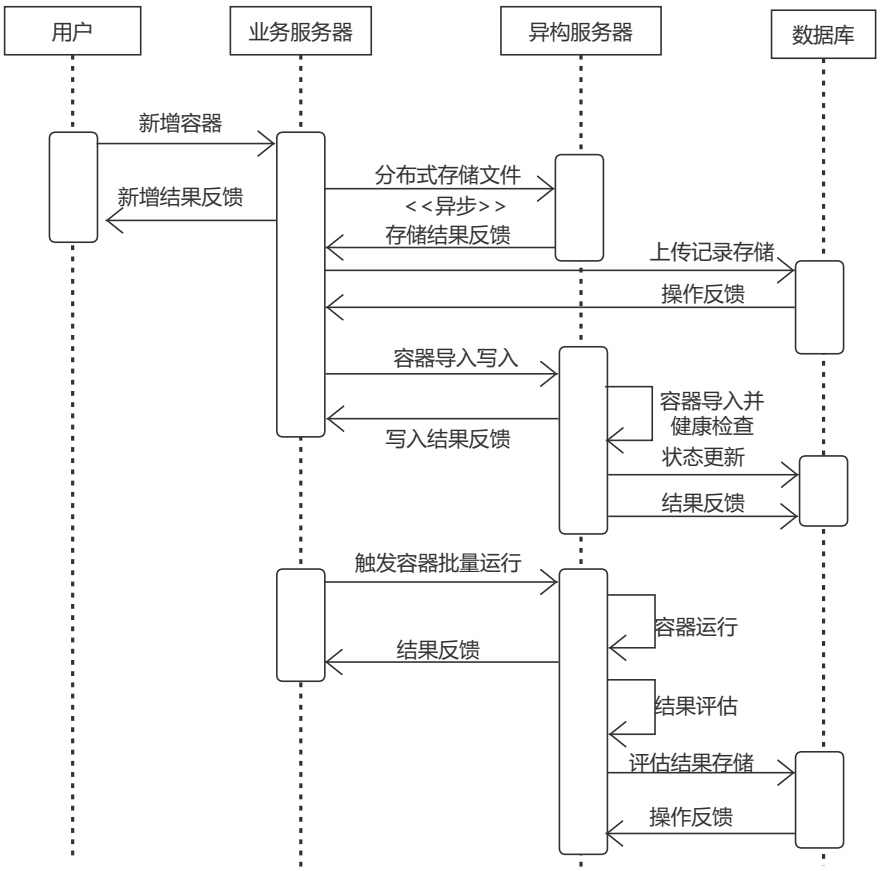

图 3-7: 容器处理进程视图

的异构服务器宕机）需要选择运行状态良好的其他异构服务器重新导入容器后运行。

3.3.5 系统部署设计

图 3-8展示了系统部署结构图，系统服务器部署主要包括前端服务器、业务服务器集群、分布式文件管理系统 tracker、异构服务器集群及数据库。前端服务器负责与客户端进行交互及返回页面展示相关的文件，同时向后端发送 http 请求并对响应数据进行展示。业务服务集群部署了在系统逻辑设计中所述的各模块微服务，微服务之间相互独立，通过 eureka server 对微服务进行管理。fastDFS tracker 管理分布式文件管理系统中的 storage 节点，在数据集文件和容器文件的存储中起到作用。异构服务器集群部署了 fastDFS 中的 storage 服务同时通过 flask 对异构服务提供请求接口。数据库面向业务服务器集群及异构服务器集群提供数据存储服务。

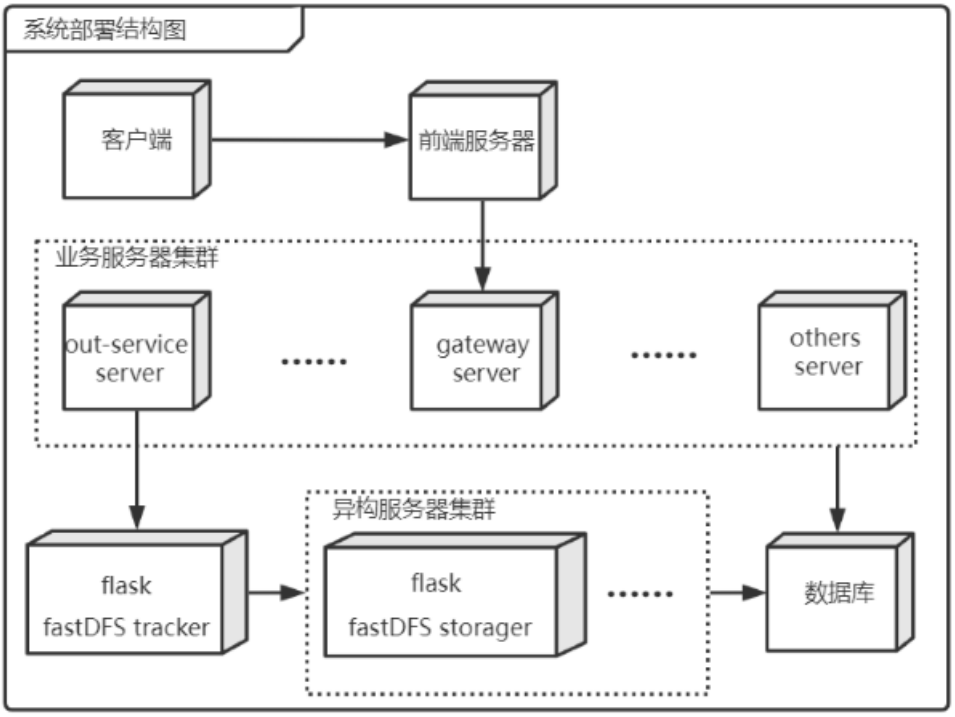

图 3-8: 系统部署设计视图

3.3.6 系统数据库设计

系统实体关系图如图 3-9所示，该图展示了在逻辑设计中的 Stage module、Dataset module、Upload module、Container module 四个模块中的实体关系及实体所包含的关键属性。每个模块中都有核心的实体类，Stage module 的核心实体是阶段记录，Dataset module 的核心实体是数据集，Upload module 的核心实体是文件记录，Container module 的核心实体是容器；除核心实体以外，模块与模块之间的关系也依赖实体表进行记录，阶段数据集就是对各比赛阶段所使用的数据集抽象为实体表。下面将详细说明图 3-9中所涉及的实体含义及其属性。

文件记录是对系统中所管理的所有文件进行持久化的结构性数据存储，在司法多维的评估系统中，系统需要对比赛管理员上传的原始数据集文件、基于原始数据集文件变异后的变异数据集文件、参赛选手上传的结果集文件及容器压缩文件等文件进行管理。文件管理不需要关注这些不同类别的文件细节，只需要对其统一的功能进行实现，故文件记录包含了文件的共同属性，例如文件的在服务器的存储路径、文件大小、文件的 MD5 码 (避免文件重复上传)、文件名称、文件状态、文件后缀等基本信息，除此之外还包括文件在上传时所属的

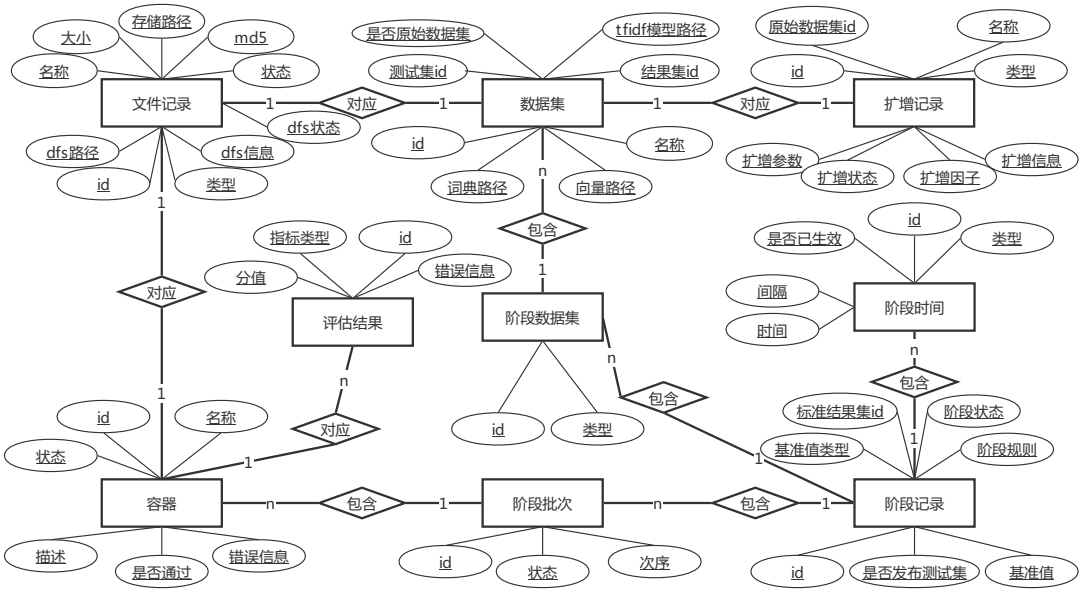

图 3-9: 系统核心实体关系图

类型，包括结果集文件、数据集文件及容器压缩文件，不同取值分别代表不同的类型。除此之外，文件上传至 fastDFS 分布式系统以文件实体为单位，在文件记录表中记录了上传路径、上传状态和上传信息，当文件上传成功时，上传信息字段记录了文件的 dfs 中的路径，该路径包含了文件上传至 dfs 的组名和服务器名，与上传路径的区别是上传路径记录了文件在具体的服务器中的存储路径，在确定的 storage 节点中需要使用该路径访问文件，当文件上传失败时，上传信息记录了 fastDFS 返回的失败原因，方便比赛管理员进行排查。实体在数据库中的设计如表 3-13所示：

表 3-13: file 表字段设计

字段	含义	类型	描述
id	文件 id	int	文件主键
user_id	用户 id	varchar	上传文件的系统用户 id
path	文件路径	varchar	文件在服务器上的存储路径
size	文件大小	int	以字节为单位的文件大小
name	文件名称	varchar	上传的文件本地名称

md5	文件 md5 码	varchar	以文件内容生成的 md5 码
status	文件上传状态	int	不同取值表示不同的文件上传状态
suffix	文件后缀	varchar	上传文件的文件后缀
type	文件类型	int	上传的文件分为结果集、数据集、容器压缩文件
dfs_status	上传 fastdfs 状态	int	状态包括未上传、上传成功及上传失败
dfs_msg	上传 fastdfs 信息	varchar	文件上传至 dfs 后的相关信息，主要保存错误信息或在 dfs 中的存储路径
dfs_path	上传 fastdfs 路径	varchar	文件上传至 dfs 后的存储路径

表 3-14描述了数据集表的存储结构，存储在该表中的数据集信息分为三类，第一类是数据集的基本信息，第二类是数据集的变异信息，第三类是数据集生成的衍生文件信息，各字段具体含义见表 3-14。数据集实体与文件记录实体有一对一的对应关系，数据集上传到系统后，首先是以文件的形式将数据集文件存储在系统中，将文件信息记录在文件记录数据集表中，其次再更新数据集表记录新上传的数据集信息。系统中管理的数据集包括用户上传的原始数据集和基于原始数据集按照变异算法生成的变异数据集，在数据集表中通过 is_original 字段对这两类数据集进行区分，若数据集为变异数据集则通过 amplification_id 管理变异记录表的主键。

表 3-14: dataset 表字段设计

字段	含义	类型	描述
id	数据集 id	int	数据集主键
user_id	用户 id	varchar	上传数据集的系统用户 id

file_id	文件记录 id	varchar	对应的文件记录表的主键
name	数据集名称	varchar	用户设置的数据集名称
is_original	表示是否是原始数据集	int	该字段值为 1 时表示是通过用户上传的原始数据集
amplification_id	变异记录表的主键	int	若该数据集是变异数据集则该条记录与变异记录进行关联
description	数据集描述	varchar	上传数据集时对数据集的描述，更全面描述数据集
init_status	初始化状态	int	数据集初始化的状态，包括未初始化、初始化成功及初始化失败
result_id	结果集文件 id	int	数据集初始化后生成的结果集文件以文件实体的形式在文件记录表中对应的 id
testSet_id	测试集文件 id	int	数据集初始化后生成的测试集文件以文件实体的形式在文件记录表中对应的 id
dictionary_path	词典路径	varchar	由数据集作为语料库生成的词典文件存储路径
bow_path	向量文件路径	varchar	由数据集作为语料库生成的向量文件存储路径

tfidf_path	tfidf 模型路径	varchar	使用词典文件和向量文件生成的 tfidf 模型持久化后的存储路径

表 3-15描述了变异数据集记录表的字段设计，数据集变异以原始数据集为基础，因此在该表中应记录原始数据集 id，在数据集变异成功后系统会生成变异数据集的结果集文件和测试集文件，系统将结果集文件和测试集文件以文件实体的方式记录在系统中，以便后续被设置为阶段测试集时可以统一的使用文件管理模块提供的上传 fastDFS 接口。同时，数据集在变异成功后，系统根据其变异参数记录其变异因子的值存储在数据库中，若该变异数据集作为阶段测试集使用，则将记录的变异因子直接写入异构服务器对应的阶段数据集文件中。除此之外，为方便管理员排查和了解变异进度，该表应记录变异任务所处的状态及变异过程中的相关信息。

表 3-15: amplification_dataset 表字段设计

字段	含义	类型	描述
id	变异记录 id	int	变异记录主键
user_id	用户 id	varchar	变异数据集的系统用户 id
dataset_id	数据集 id	int	原始数据集的主键
type	变异数据集类型	int	变异数据集分为鲁棒性和公平性变异数据集
param_str	变异算法参数	varchar	传入变异算法的参数
amplify_status	变异状态	int	数据集的变异状态，包括未开始、变异中、变异成功及变异失败
amplify_msg	变异信息	varchar	变异过程中的相关信息，主要记录变异失败时的错误信息

amplify_factor	变异因子	float	根据变异参数计算出的变异因子

同数据集持久化存储类似，容器的存储也依赖于文件记录表，容器除文件信息以外的数据存储在容器表中，表 3-16描述了容器实体的字段设计。容器由分布式文件管理上传到异构服务器后，在其对应的文件实体中更新其 dfs 状态、路径及信息，根据 dfs 信息可获取容器当前上传的初始 storage server 的名称，从而指定该容器导入的异构服务器，在该服务器中自动调用服务器中的脚本对容器进行导入、健康检查等操作，容器批量运行则通过定时任务启动脚本来实现，status 字段记录了当前该容器按照导入、健康检查、运行等节点划分容器状态，error_message 字段记录在上述三个操作中任意操作失败的错误信息，只有当容器在上一操作成功时才能进行下一个操作。容器与阶段中的批次 id 是一一对应的关系，用户在一个批次中仅能提交一个有效容器，因此容器记录表中也应包含批次 id 及其是否通过该批次所在阶段的标识。

表 3-16: container 表字段设计

字段	含义	类型	描述
id	容器 id	int	容器主键
user_id	用户 id	varchar	上传容器的系统用户 id
name	容器名称	varchar	用户设置的容器名称
description	描述	varchar	用户设置的容器的补充信息
status	容器状态	int	容器在系统中的当前状态
error_message	错误信息	varchar	容器操作失败时记录容器失败信息
batch_id	批次 id	int	新增容器时所对应的批次 id
is_pass	通过标识	tinyint	是否通过该批次所在阶段测试的标识

file_id	容器文件实体 id	int	在文件实体表中记录的容器压缩文件实体 id

在提交截止时间或提交周期节点到达时，系统会调用异构服务器中的容器批量运行脚本对所有 storage 节点中的对应阶段批次进行批量运行和评估。容器运行成功后自动生成标识文件触发容器评估，不同类型的测试集生成不同结果集评估后对应不同的评估指标，每一条容器评估指标记录都会插入到评估结果表中，评估结果表的字段设计如表 3-17所示。

表 3-17: evaluation 表字段设计

字段	含义	类型	描述
id	评估结果 id	int	评估结果主键
val	评估结果值	float	对应指标类型的评估结果值
container_id	容器 id	int	该评估结果对应的容器 id 值
evaluation_status	评估状态	int	该指标的评估状态
index_type	指标类型	int	指标类型包括准确性、泛化性、可参考性、鲁棒性及公平性
error_message	错误信息	varchar	指标评估过程中的错误信息

司法多维评估系统对模型评估分为多个阶段，系统对阶段管理以阶段记录表为核心，阶段记录表包含的字段如表 3-18。阶段类型也可描述为次序即在评估过程中各阶段启动评估的次序，阶段流程是控制各阶段参赛选手可执行操作的关键字段，在关键时间点到达后会自动更新该字段的值，以达到自动化控制的目的。

表 3-18: stage 表字段设计

字段	含义	类型	描述
id	阶段 id	int	阶段主键

type	阶段类型	int	本系统中包含三种阶段类型
process_status	阶段流程	int	阶段流程分为四个阶段，包括阶段开始、提交开启、提交截止及结果公示
pass_type	基准线类型	int	基准线类型包括按比例和按分数两种
pass_value	基准线值	float	根据基准线类型指定可以进入下一阶段的基准线
result_path	基准结果集文件路径	varchar	指定进行比对的基准结果集路径
test_release	测试集发布标识	tinyint	指定该阶段设置的测试集是否发布
rule	阶段规则说明	varchar	由管理员设置的富文本格式规则说明

阶段时间表记录对应阶段的时间节点，包括开启该阶段时间节点（开启后对用户可见且可下载该阶段训练集）、开启提交时间、关闭提交时间、提交周期等信息，详细的表字段如表 3-19所示。

表 3-19: stage_time 表字段设计

字段	含义	类型	描述
id	阶段时间 id	int	阶段时间主键
stage_id	阶段 id	int	阶段记录主键
time	时间节点	datetime	用户设置的时间节点
interval	周期	varchar	用户设置的阶段评估周期

type	节点类型	int	该阶段时间的节点类型，包括开启阶段时间、开启提交时间、关闭提交时间等
used_status	时间使用状态	tinyint	当值为 1 时表示该时间已生效，无法进行修改

阶段数据集表指定在各阶段中训练集和测试集对应的系统中的数据集，在阶段提交截止时间或提交周期截止时间到达后会将对应阶段中设置的数据集写入异构服务器中，写入信息包括阶段数据集在异构服务器上的测试集路径和结果集路经、数据集类型及变异因子，表 3-20记录了阶段数据集表的各字段及含义。

表 3-20: stage_dataset 表字段设计

字段	含义	类型	描述
id	阶段数据集 id	int	阶段数据集主键
stage_id	阶段 id	int	阶段记录主键
dataset_id	数据集 id	int	数据集主键
type	作用类型	int	数据集在该阶段的作用类型，包括测试集、训练集等

阶段批次表针对设定执行周期的评估阶段，在该情况下，用户可以在该阶段的指定周期中多批次提交容器，每个批次都会对该批次提交的容器批量运行后进行结果评估。阶段批次启动后会调用异构服务中的接口进行初始化，包括创建该批次待导入容器列表文件、处理行列表文件、日志文件等，并会开启对待导入容器列表文件的监听，表 3-21对该表结构进行描述。

表 3-21: stage_batch 表字段设计

字段	含义	类型	描述
id	阶段批次 id	int	阶段批次主键
stage_id	阶段 id	int	阶段记录主键

batch_order	批次次序	int	在阶段中的次序
use_status	批次状态	int	批次包括三个状态，分别为未开始、进行中及已结束
init_status	初始化状态	int	批次开启后对其进行初始化操作，包括未初始化、初始化成功和初始化失败三个状态

3.4 本章小结

本章着重于对司法多维评估系统进行相关的需求分析和概要设计，首先从司法多维评估系统的总体规划出发对评估系统进行需求分析，包括功能性需求和非功能性需求，在司法多维评估系统用例图的基础上详细描述司法多维评估系统功能性需求用例。随后基于司法多维评估系统的需求分析对系统进行了概要设计，通过”4+1” 视图 [35] 的梳理将系统划分为七个模块，为系统详细设计和实现提供关键思路，在本章的最后部分设计了系统的持久化方案。

第四章　数据变异方法与司法模型评估指标的设计

4.1　数据变异方法

数据变异方法按照变异后的司法特性不同分为鲁棒性变异和公平性变异，其变异后的数据集分别对应鲁棒性评估指标和公平性评估指标的测试集，通过比较司法模型在变异测试集上与原始数据集上的预测差异可获得模型对应的评估指标值。

4.1.1　鲁棒性变异

鲁棒性变异专注于对案例描述文本进行增删改等操作，相较于传统文本突变法中随机增加词语、删除文本中的词语以及修改词语顺序，本文引入的鲁棒性变异方法变异生成后的数据更加具有司法领域的特性，使得突变后的文本更加接近实际司法案例数据，更能反映模型在实际应用场景中针对有相关质量问题的数据时的真实效果。鲁棒性变异包括基于 TF-IDF 的特征裁剪、基于依存句法的特征变换以及基于词典的随机插入三种变异方法，下面将对三种算法的实现方式进行阐述。

基于 TF-IDF 的特征裁剪 该算法可模拟实际司法数据中的存在的案例描述不全的问题，词语的 TF-IDF 值是衡量该词语在当前所在文本中的重要性的指标，TF-IDF 值越高则说明该词语对该文本的重要性越高，如果删除该词语就将对文本的语义造成比较大的影响，因此基于 TF-IDF 的特征裁剪是要删除掉对文本重要性较小的元素。该算法的特征裁剪的最小粒度不是词语，而是将文本解析成依存句法树后的树枝，因为直接删除单个词语会对语句的连贯性造成影响，依存句法树枝的 TF-IDF 值等于构成该树枝的所有词语的 TF-IDF 值之和，其中停用词、特殊符号和数字的 TF-IDF 值都取 0，该算法的处理流程如下所示：

(1) 对待变异文件进行初始化操作，包括分词、去停用词。
(2) 基于分词后的结果生成依存句法树。
(3) 使用 TF-IDF 模型计算词语的 TF-IDF 值。
(4) 计算依存句法树中节点的 TF-IDF 值。
(5) 按照节点的 TF-IDF 值进行排序。
(6) 根据排序结果对文本进行剪枝，获得变异后的文本。

基于 TF-IDF 特征裁剪算法包含了 lengthWeight、rangeWeight 以及 quanlity-Weight 三个参数，lengthWeight 表示计算 TF-IDF 时选取树枝的范围，树枝中包含词语的个数在该文本中所有词语的占比小于该值的将不会放在待裁剪树枝列表中；rangeWeight 表示树枝 TF-IDF 排序列表的选择范围，在根据长度筛选树枝列表后将对待裁剪树枝根据 TF-IDF 进行排序，按照从小到大的顺序选取排名在 rangeWeight 之前的树枝作为更新后的待裁剪数组列表；quanlityWeight 用于随机在待裁剪数组列表中选取最终裁剪树枝，以保证每次使用该算法裁剪的树枝有所差异。表 4-1展示了使用基于 TF-IDF 特征裁剪算法对文本进行变异后的内容差异，算法中的三个参数的默认值都为 0.4，从表中可看到变异后的文本内容更加简短，对于一些信息进行了省略，但案件类型和基本的语义信息依然没有改变。

表 4-1: 基于 TF-IDF 特征裁剪算法变异后的文本对照表

初始文本	变异文本
公诉机关指控：2020 年 3 月 23 日 18 时许，被告人龙某在广东省深圳市龙华区龙华文化广场公交车站台，趁被害人张某准备上公交车时，盗窃了张某包中一台黑色 iphone7 型手机（经鉴定价值人民币 2300 元）。	公诉机关指控：被告人龙某在龙华区龙华文化广场公交车站，盗取张某包中手机。

基于依存句法的特征变换 该算法是在保持文本内容不变的情况下，修改语序结构，修改过程中以文本的依存句法关系为依据，因此可以保证文本的领域特征不会发生变化。该方法不依赖文本所在的语料库，只是对文本结构本身进行挖掘，算法处理流程如下所示：

(1) 生成待变异文本的依存句法树。
(2) 根据长度参数筛选待替换的树枝。

(3) 将存在包含关系的树枝合并。
(4) 根据依存关系匹配待交换的树枝集合。
(5) 随机选择树枝集合中的树枝对进行交换生成变异后的文本。

基于依存句法的特征裁剪包含了 lengthWeight 和 selectWeight 两个变异参数，lengthWeight 与基于 TF-IDF 特征裁剪中的 lengthWeight 参数含义相同，表示树枝长度范围的参数；selectWeight 是配对集数量选择参数，在第五步中随机选择的树枝集合的树枝配对集数量是通过 selectWeight*length 获得的，其中 length 表示树枝配对集的总数，该参数可以保证每次选取的配对集不同从而变异后生成的文本也不同。表 4-2展示了使用基于依存句法的特征裁剪算法变异后的文本内容对比，该变异中的两个参数的取值为 0.4，变异后的文本内容和基本的语义没有发生变化，但是语序结构发生了改变。

表 4-2: 基于依存句法的特征变换算法变异后的文本对照表

初始文本	变异文本
南京市玄武区人民检察院指控被告人丁某于 2018 年 12 月 9 日 1 时许，在本市玄武区珠江路 XXX 号新世界百货门口，窃得被害人周某停放于此的白色爱马电动车中的电瓶一台，在逃逸过程中被公安人员人赃俱获。	被害人周某指控被告人丁某于 2018 年 12 月 9 日 1 时许，在白色爱马电动车中的电瓶珠江路 XXX 号新世界百货门口，窃得南京市玄武区人民检察院停放于此的本市玄武区一台，被公安人员在逃逸过程中人赃俱获。。

基于词典的随机插入 该算法是在保证语义基本不变的情况下，向原始文本中随机插入词典文件中的词语，词典文件中的词语来自于文本所在的语料库，并且词语的司法特性较高，变异后的数据可以模拟非结构化用语的质量问题，以此检测司法模型在存在该质量问题的数据集中的表现，算法处理流程如下所示：

(1) 对待变异文本进行分词。
(2) 根据插入比例计算插入词数。
(3) 加载词典文件，从词典中随机选择词语。
(4) 在待变异文本中随机选择位置后插入词语生成变异后的文本。

基于词典的随机插入算法包含了 lengthWeight 一个变异参数，从词典中随机选取的词语数量为 lengthWeight*length，其中 length 表示分词后的文本中包

含的词语总数，选取词语结束后插入的文本中的位置也是随机选取，从而保证每次变异后的文本内容有所区别，表 4-3展示了基于词典的随机插入变异后的文本对比，算法中的 lengthWeight 参数取值为 0.4，变异后的文本内容进行了扩充。

表 4-3: 基于词典的随机插入算法变异后的文本对照表

初始文本	变异文本
本院二审期间上诉人张某委托家属缴纳罚金，审理查明原判认定上诉人张某犯罪事实清楚，证据充分均经原审庭审质证，证据来源合法，内容客观真实且相互印证，本院予以确认。	本院二审期间，上诉人张某委托张某乙连带无名麻将室内不再向周家坤三星大道家和家属缴纳罚金该戒指窃走人民币，审理查明原判笔记本电脑销赃至认定上诉人张某犯四宝香烟犯罪事实，长岭镇一些磨粉清楚证据境外确实的姨父王某某充分大统华超市均经原审庭审质证证据牟县公安局勘验来源合法内容客观真实相互的马路王牌山地印证本院予以确认。。

4.1.2 公平性变异

公平性变异是针对司法评估中关注的公平性问题进行设计的，考察司法模型是否具有排除不相干要素的能力，公平性变异包括基于语义的歧视性文本插入一种变异算法。

基于语义的歧视性文本插入 在司法判案中性别、地区、民族等因素都是影响判决结果的潜在因子，通过这些因子可生成歧视性文本来修饰被告人的身份，从而在不改变案例性质、案例内容的情况下，考察模型在突变后的文本中的预测表现是否有所区别，该算法的变异流程如下所示：

(1) 根据语义确认待插入文本的位置。
(2) 从歧视性语料库中选择歧视性文本。
(3) 将选择的文本插入至对应位置中，生成待变异文本。

根据语义确认待插入文本的位置是获取“被告人”字符串所在的文本位置，歧视性文本只对被告人身份进行修饰。歧视性语料库是按照歧视性因子生成的歧视性语句的集合，后续可随其进行扩充。该算法无需指定相关参数，变

异后的文本差异较小，表 4-4展示了使用该算法变异后的文本与初始文本的比较，从表中可看到变异后的文本在第二行的“被告人”和“张某”之间插入了“性别男”的歧视性文本。

表 4-4: 基于语义的歧视性文本插入算法变异后的文本对照表

初始文本	变异文本
衡水市桃城区人民检察院指控被告人张某多次放任其建立的群组成员传播淫秽视频，截至案发，该群累计发布淫秽视频超过1500 余条，上述指控由公诉机关提供证据，本检察院认为被告人张某利用互联网建立电子信息群组，放任他人传播淫秽视频情节严重，该行为触犯中华人民共和国刑法规定，本院将依法对其进行判处。	衡水市桃城区人民检察院指控被告人性别男张某多次放任其建立的群组成员传播淫秽视频，截至案发，该群累计发布淫秽视频超过 1500 余条，上述指控由公诉机关提供证据，本检察院认为被告人张某利用互联网建立电子信息群组，放任他人传播淫秽视频情节严重，该行为触犯中华人民共和国刑法规定，本院将依法对其进行判处。

4.2 司法模型评估指标设计

基于司法多维评估的模型评估系统将评估指标分为两类，一类是多分类问题的基础性指标，根据评估原理在司法场景中为其赋予司法含义，包括准确性、可参考性和泛化性；另一类是扩展性指标，该类指标基于模型在不同司法特征测试集上分类效果的差异程度进行取值，包括鲁棒性和公平性。

该司法模型评估系统面向刑期预测模型进行评估，我国刑法中规定了有期徒刑、无期徒刑和死刑三类刑期，其中有期徒刑的判刑范围不得超过 20 年 [36]，而被判定为无期徒刑和死刑的罪犯则被剥夺政治权利终身无判刑范围一说，故刑期预测模型可以转化为多分类问题，分类标签为 0 到 21，若刑期预测结果标签为 1 表示刑期在 1 年以内，而标签为 21 则表示无期徒刑或死刑，转化为多分类问题后，评估指标的设计就可参考多分类的评估指标。

准确性 表示分类正确的样本个数占总样本，在司法模型的评估中该指标可以直观的表示刑期模型的分类性能，是最常用的评估指标，其计算公示如 4-1所示，其中 $\hat{y}_i$ 是第 i 个样本的预测值，y_i 是相应的真实值，$1(x)$ 是指示

函数。

$$accuracy(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i) \tag{4-1}$$

在司法层面上该指标可直接反映模型的可用性，如模型分类预测的准确率较低，则说明该模型在分类性能上还有待提升，尚不适用于实际的司法场景中。

可参考性 表示模型分类的总体可参考性，刑期模型的分类标签代表了预测的具体刑期值，当分类标签错误时，具体的标签值偏离程度反映了分类结果的可参考性，例如在一个真实的司法案例中其判刑结果为 5 年零 3 个月，那么正确的分类标签应该是 6，假如有两个不同的模型，第一个模型针对该案例文本描述将其分类为 7，而第二个模型则将其分类为 2，那么显然第一个模型的分类可参考性要高于第二个模型，由此设计的可参考性指标计算公式如所示，该公式的取值范围为 0 到 1，当 $\hat{y}$ 与 y 的距离越大，$refrence(y,\hat{y})$ 越接近于 0，反之则越接近于 1。

$$refrence(y,\hat{y}) = \log_2 \frac{21}{\sqrt{(\hat{y}-y)^2}+21} + 1 \tag{4-2}$$

模型的可参考性越高说明该模型在具体的司法案例中预测的结果越具有参考价值，作为量刑辅助系统，可参考性高的模型能更好的发挥其辅佐作用，帮助相关司法人员提升工作效率，同时维护公平公正。

泛化性 也可以理解为一致性，用以表示模型分类的偏向性，使用 kappa 系数 [37] 计算该指标的值，kappa 系数的计算基于混淆矩阵，取值为-1 到 1，计算公式如 4-3所示。

$$kappa = \frac{p_0 - p_e}{1 - p_e} \tag{4-3}$$

其中 p_0 等于“对角线元素之和”除以“整个矩阵的元素之和”，p_e 等于所有刑期类别对应的“实际与预测数量的乘积”的总和，除以“样本总数的平方”，对 kappa 系数使用公式 4-4进行归一化就可表示泛化性。

$$generalization = \frac{kappa + 1}{2} \tag{4-4}$$

泛化性越高说明模型在多种刑期的分类任务中分类效果越平均，反之则说明其在分类预测过程中具有偏向性，例如某个模型适用于处理判刑年数较少的案例，而处理判刑年数较多的案例时就会存在较大的误差，这种情况下该模型的泛化性结果就比较低，在具体使用该模型时就需要考虑到它处理不同案例文本的能力是不均衡的。

公平性和鲁棒性 变异指标的评估基于模型在变异数据集上分类的准确率与其在原始数据集上准确率的比值，同时变异指标值的计算需要综合变异数据的复杂性，复杂性的计算基于变异算法的参数，在设计时将变异算法的参数分为两类，一类正向参数，一类是逆向参数，正向参数取值越高变异后的数据集复杂性就越高，反之则是逆向参数，变异性指标的计算公式如 4-5所示：

$$amplification = \frac{accuracy_{amplify}}{accuracy_{original}} + \frac{\beta}{10} \tag{4-5}$$

其中 $accuracy_{amplify}$ 是模型在变异数据集的准确率，$accuracy_{original}$ 是模型在原始数据集的准确率，β 是由正向参数和逆向参数计算得到的影响因子，计算公式如 4-6所示。

$$\beta = \frac{\sum_i^x \log_{10}(\frac{p_i - p_{min}}{p_{max} - p_{min}} + 1)}{x} \times \frac{\sum_j^y \log_{10}(\frac{n_{max} - n_j}{n_{max} - n_{min}} + 1)}{y} \tag{4-6}$$

影响因子的取值在 0 到 1 之间，公式 4-6中的前半部分表示正向参数的影响因子，后半部分表示逆向参数的影响因子，在前半部分中 x 表示正向参数的总个数，p_i 表示第 i 个正向参数的取值，p_{min} 表示该正向参数最小取值，p_{max} 表示该正向参数的最大取值，后半部分变量的含义类似。

公平性和鲁棒性都是正向评估指标，即指标值越高说明该司法模型越好。公平性是司法领域中关注的重点，所谓“同案同判”指的就是维护司法判案的公平性，这个“同”表示的不是司法案例过程的完全一致，而是指在去除不相干的司法因素后的司法要素的一致性，因此系统中的公平性指标评估就是关注模型在可能的歧视性文本下是否有不同的表现，其分类的准确率是否有较大的差别，差别越大则说明模型排查不相干因素的能力越弱，则其在公平性指标的得分也就越低，反之则越高。鲁棒性是对模型处理异常的能力的考查，在实际的模型使用场景中案例的描述可能不够完备，模型是否能输出有效结果可作为衡量其司法领域稳定性的标准。在鲁棒性评估中使用三种领域变异方法对数据集进行变异，包括基于 TF-IDF 的特征裁剪、基于依存句法的特征变换和基于

词典的随机插入，相较于随机的文本突变法，三种方法变异出来的司法数据更加具有司法领域特性。

4.3 本章小结

本章对数据突变方法和司法多维评估指标的设计进行了详细的介绍。前半部分介绍的数据突变方法包括基于 TF-IDF 的特征裁剪、基于依存句法的特征变换、基于词典的随机插入以及基于语义的歧视性文本插入，前三种变异方法属于鲁棒性变异方法，最后一种变异方法属于公平性变异方法。后半部分介绍了司法多维评估指标中基础性指标以及变异性指标的计算公式，基础性指标的计算公式有所差异，取值范围是在 0 到 1 之间，而变异性指标的计算公式是统一，在计算中变异的复杂性通过变异因子来体现，变异因子的计算基于变异算法中的相关参数。

第五章　系统详细设计与实现

5.1　认证授权模块详细设计与实现

5.1.1　认证授权模块详细设计

司法多维评估系统中的用户角色分为参赛选手和比赛管理员，不同用户角色具有不同的权限，系统前端需根据用户角色生成动态路由，对用户权限在页面粒度上进行划分，后端则需要提供认证授权模块为 Spring Cloud 架构中的其他微服务提供统一的认证授权服务。用户认证鉴权时序图如图 5-1所示：

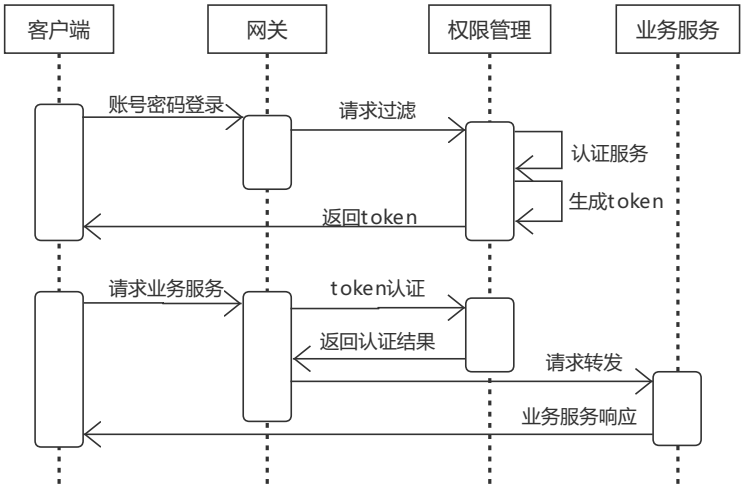

图 5-1: 认证授权流转时序图

系统通过单点登录模式进行认证授权，由 gateway Module 和 Authorization module 共同实现该功能。gateway Module 对客户端发送的所有请求进行拦截，对用户登录和注册接口进行过滤转发至 Authorization Module，Authorization 模块对用户登录提交的信息进行身份认证，认证成功后生成用户在此次登录中身份验证的唯一标识——token，token 生成后存储在服务端同时也需在响应中返回给客户端，客户端将 token 存储在本地，生成的 token 具有标识用户身份的功能，在后续的请求中的 token 都被传递至服务器进行验证。当客户端进行业务

服务请求时，网关拦截该请求，并认证其携带的 token 合法性，网关基于 token 合法性决定是否对请求进行转发，从而实现单点登录的功能。

除了系统后端模块支持，前端针对用户角色对用户权限进行限制。后端返回给前端的 token 是通过 cookie 设置在浏览器中的，在发送请求时前端会先拿到本地 cookie 中的 token，若 token 不存在或失效则需跳转至登录页面重新登录获取 token，获取到 token 后前端会携带 token 访问获取用户详细信息的接口，用户的详细信息中包含了用户的角色字段，基于用户角色在路由表中选择用户可以访问的路由生成动态路由，通过 Vue Router 提供的 addRoutes 方法可动态添加路由，通过这种方式用户在访问其不具有权限的页面时可以跳转到 404 页面，达到认证授权的目的。

5.1.2 认证授权模块具体实现

Authorization Module 的实现基于 OAuth2 标准，该标准制定了授权认证的规范化的设计思路和运行流程，Spring Cloud 根据该标准封装了 spring-cloud-starter-oauth2 这一具体实现，并在实现中引入了 spring-security。OAuth2 提供了多种授权模式，在司法多维评估系统中使用密码模式进行认证授权，在 Authorization Module 中需要配置 spring-security、实现 UserDetailsService 及设置 OAuth2 配置文件。

通过继承 WebSecurityConfigurerAdapter 类重写 config 方法配置 spring-security，config 方法定义了运行匿名访问的接口、密码加密方式及 UserDetailsService 接口的实现类。UserDetailsService 接口实现的关键方法是 loadUserByUsername 方法，其接收传递过来的用户名并返回一个包含用户相关信息的 UserDetail 对象，该对象的属性 copy 自根据用户名查询数据库后返回实体。spring-cloud-starter-oauth2 的核心是配置 OAuth2，在具体实现中通过注解和继承 AuthorizationServerConfigurerAdapter 父类重写三个 configure 方法进行配置，传入 ClientDetailsServiceCondigurer 对象的 configure 方法配置了授权认证客户端，指定授权认证方式、客户端名称、token 有效时长、密码加密方式、客户端访问权限等配置项，传入 AuthorizationServerEndpointsConfigurer 对象的 configure 方法指定配置 token 存储在 redis 中，最后一个重写的 configure 方法限制客户端访问认证接口的权限，关键代码实现如图 5-2所示。

在不同的业务场景中系统可配置多个认证客户端从而实现多种认证方式，在司法多维评估系统中系统实现了使用用户名和密码方式进行验证的客户端，

```
@Override
public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
   //配置客户端用于password认证
   clients.inMemory()
         .withClient("client_1")
         .resourceIds()
         .authorizedGrantTypes("password", "refresh_token")
         .scopes("select")
         .authorities("client")
         .secret("{noop}123456");
}

 @Bean
public RedisTokenStore tokenStore() {
   return new RedisTokenStore(connectionFactory);
}


@Override
public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception {
//配置token存储方式
   endpoints
         .tokenStore(tokenStore())
         .authenticationManager(authenticationManager);
}

  @Override
public void configure(AuthorizationServerSecurityConfigurer oauthServer) throws Exception {
   //允许客户端访问Oauth2授权接口
   oauthServer.allowFormAuthenticationForClients();
}
```

图 5-2: 客户端认证关键代码

系统前端在发送登录请求的时候需通过 query 字段指定认证客户端的信息，包括客户端 id、密钥等信息，指定后使用认证授权模块使用该认证客户端对用户登录请求进行认证。

5.2 文件管理模块

5.2.1 文件管理模块详细设计

司法多维评估系统中的文件管理模块涉及到对选手上传的结果集文件、容器压缩文件及比赛管理上传的原始数据集文件进行管理，主要提供文件上传和文件下载的功能。

文件管理模块封装成独立的微服务运行在业务逻辑服务器中，通过该模块上传的文件直接存储在该业务逻辑服务器中，但容器压缩文件和设置为测试集

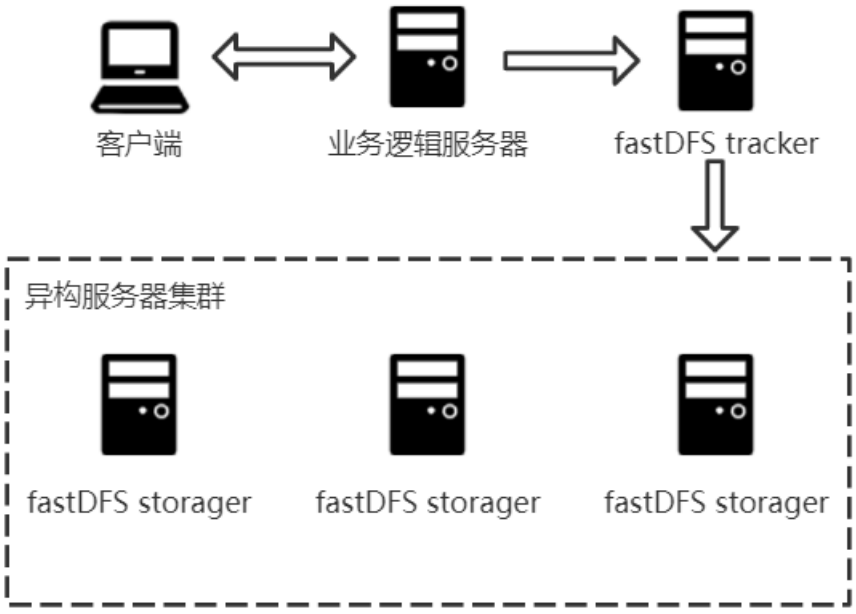

图 5-3: 文件服务器组织结构图

的数据集文件需要通过异构服务进行处理所以在上传至业务逻辑服务器之后还需上传至异构服务器集群中，因此引入 fastDFS 分布式文件管理系统。fastDFS 具有线性冗余的功能，对逻辑上划分为同一组的服务器会进行资源同步，上传的容器压缩文件和数据集文件需要更新到异构服务器集群中的所有服务器中，在 fastDFS 中将异构服务器集群划分为一个组进行自动同步更新，文件上传和同步处理结构图如图 5-3所示。

业务服务器在文件上传时直接与用户进行交互，容器文件及数据集文件作为整体上传耗时较长且成功率较低，故文件管理模块切分文件，分块上传后对文件进行整合，文件的完整性通过 md5 码进行标识，对于已上传的文件或已上传的完整分片不会进行重复上传而是采用直接选中的方式降低交互成本和存储成本。文件分片上传的工作在前端完成。

文件预处理包括检查文件类型、检查文件大小、对整个文件生成 md5 码、将文件分为多个 chunk 并生成每个 chunk 的 md5 码等工作，预处理结束后会将文件的 md5 码传输给后端进行检查，若在数据库中已存储该 md5 码表示该文件之前上传过需检查该文件是否上传完成，若已上传完成则直接为用户选定该文件无需重复上传，若未完成上传则需检查每个分片是否上传完成，对已上传完成分分片不进行上传，在更小的粒度上减少重复。对于未上传的分块分组进行传输，每次发送的 post 请求会携带特定数量的分块，直至文件分块全部传输

完成。

后端在处理文件上传时是通过文件夹来处理中间状态的分片文件，处理流程如图 5-4所示，生成的文件上传路径是以文件的 md5 码作为文件夹名称，若该路径在业务逻辑服务器中已存在，则表示该文件已上传过，根据当前分片的 md5 码可以获得当前分片的存储路径，对于已上传过的文件分片进行检查，若在指定路径下文件存在，则生成该文件的 md5 码，与前端传输的 md5 码进行比对，判断该分片上传的完整性，对于上传不完整的分片在删除其在业务逻辑服务器中存储的对应分片后重新进行分片内容的写入。在写入文件的第一个分片后，需要将文件的信息写入到数据库中，上传状态为上传中，而在最后一个分片上传后需要修改数据库中文件的上传状态为已上传，同时容器压缩文件还需上传至 fastDFS 中，上传成功后修改上传状态。

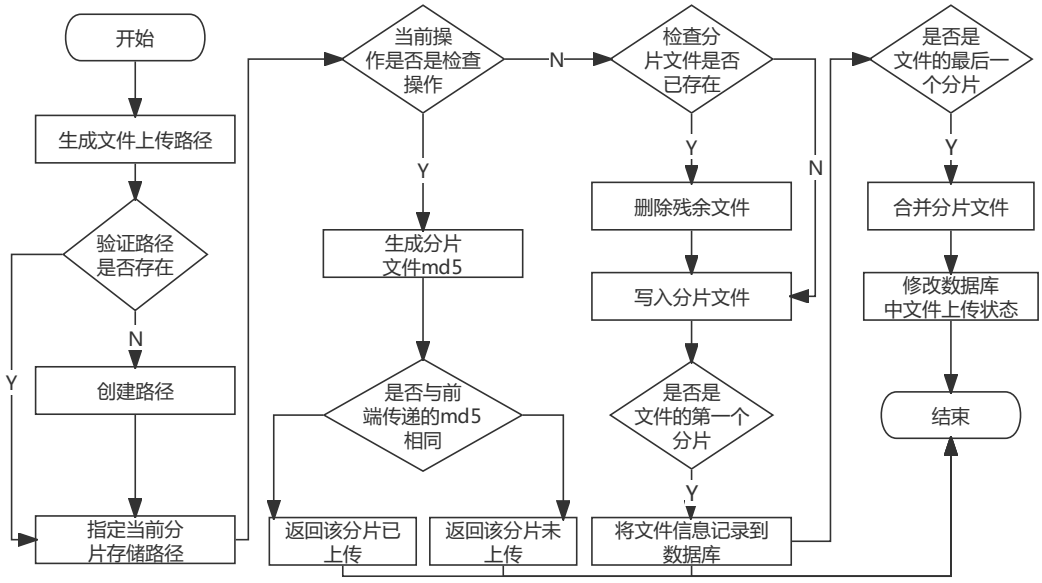

图 5-4: 文件上传后端处理流程图

与文件上传的分片上传不同，文件下载是以整体进行下载的，同时在下载过程中不涉及分布式服务器的处理而是直接从业务逻辑服务器中进行下载，文件下载以流的方式写入 response 对象，用户下载后可直接在对应的文件夹中查看。

5.2.2 文件管理模块具体实现

文件上传功能依赖于前后端协作共同完成，前端通过 spark-md5.js 提供的 append 和 end 方法计算文件 md5 码及 File 原型对象提供的切分方法对文件进行切分，后端则依赖于 java.security.MessageDigest 提供的 getInstance 方法获得生成文件 md5 码的实例对象以及通过合并文件夹下的文件达到还原的目的。 文件

```
private FileResDto uploadFastDfs(FileRes fileRes){
    // 将文件上传到fastdfs
    InputStream is = null;
    try {
        File source = new File(fileRes.getPath());
        is = new FileInputStream(source);
        StorePath storePath = storageClient.uploadFile(is, source.length(), FilenameUtils.getExtension
(source.getName()),null);
        String fullPath = storePath.getFullPath();
        fileRes.setStatus(2);//上传结束后更新上传状态
        fileRes.setDfsPath(fullPath);//记录上传至fastDfs的路径
        fileRes.setModifyTime(new DateTime());
        fileResMapper.updateByExampleSelective(fileRes, example);//将实体更新到数据库中
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } finally {
        try {
            if (is != null){
                is.close();
            }
        }catch (IOException ioe){
            ioe.printStackTrace();
        }
    }
}
```

图 5-5: uploadFastDfs 方法代码

的 md5 码生成和切分在文件预处理阶段完成，文件切片后会计算每一个分片的 md5 码并记录分片文件的起止位置、索引及具体内容，一次请求的多个分片以 arrayBuffer 的形式进行传输，除最后一个分片外每个分片固定大小，当前设定的分片大小是 1MB，前端对上传进度进行可视化的处理，通过已上传的分片数占比表示上传进度。在检查文件是否上传完成之后对于上传未完成或未上传的文件调用 spliceUpload 核心方法进行流程控制，方法中的 num_json 对象存储了此次请求中上传的分片组的最小索引和最大索引，方法传进来的最后一个参数 isHalf 表示当前文件是否上传过，若上传过则在每次上传分片时需将 action 参数赋值为”check” 指示后端上传接口先对该分片的内容完整性进行检查，返回检查结果后再决定是否对文件进行上传，若 isHalf 取值为 false 表示该文件从未上传过在上传分片时无需进行检查直接将 action 参数赋值为“upload”执行上传

操作即可。每当分片上传成功后调用 uploadChunkResultHandle 方法更新上传进度并决定是否再次发起请求传输文件。

后端在处理文件上传时以 action 参数的值决定具体的操作，若为“check”只需检查前端传递的分片 md5 码与后端对应路径下文件的 md5 码是否相同，将比较结果返回给前端，由前端决定是否对该分片进行上传；若为“upload”则需新建对应的分片文件并将前端传递的分片文件写入，并更新数据库中的相应记录，在文件所有的分片全部上传完成后调用 uploadFastDfs 方法将文件上传至异构服务器集群中，uploadFastDfs 方法代码如 5-5所示。

其中 storageClient 通过 Bean 方式进行引入，是 FastFileStorageClient 类的实例化对象，在 application.yaml 中记录了 fastDfs 的配置信息，配置信息如图 5-6所示,tracker-list 是具体的 fastDfs tracker 的 ip 地址和端口。

```
fdfs:
  so-timeout: 3000
  connect-timeout: 1000
  tracker-list:
    - 172.19.240.26:22122
```

图 5-6: fastdfs 配置信息

5.3 阶段管理模块

5.3.1 阶段管理模块详细设计

阶段管理模块实现比赛管理员对各阶段数据进行管理包括规则设置、时间管理、基准线管理及数据集管理。比赛阶段的主干流程由上传测试集、上传相关文件、结果公示三个步骤组成，阶段规则对各阶段的具体流程进行说明，时间管理中设置的有效时间完成各步骤之间的自动化流转，各阶段的基准线管理中设置的基准线决定选手是否可以进行下一阶段的比赛，数据集管理指定该阶段使用的测试集及训练集。

规则设置 以富文本形式对阶段流程进行说明，说明内容包括各阶段的提交规则。对第一阶段来说规则说明应包括上传的结果集文件格式、下载的测试集格式等信息；对第二阶段和第三阶段来说应包含上传的容器压缩文件的打包要求、需要提供的容器命令、容器输出的结果集格式等内容，除此之外第二阶段还应对提交周期进行说明。

时间设置 包括设置阶段开放时间、总提交文件起止时间、可提交次数、提交周期及结果公示时间。阶段开放时间指定对应比赛阶段的对外开放时间，开放时间到达后具有该阶段参赛资格的选手可以访问该阶段具体内容，包括提交规则和相关时间节点等；选手在各阶段提交文件时必须在设置的总提交文件起止时间范围内，提交次数分为多次提交和单次提交，对于多次提交来说还需指定提交周期，提交周期以周为单位指定，设置每周提交截止节点，截止节点之间选手只有一次提交机会；结果公示时间指定在提交截止节点之后的多少时间内对该批次评估结果进行公示，结果公示后选手可根据评估结果和排名结果更新模型，在下一批次或阶段重新提交模型。

基准线设置 用于划分当前阶段进入下一阶段的基准线，基准线设置分为两类，包括按照比例进行划分和按照分数进行划分，按照比例的划分方式需要指定在此次提交中排名在前百分之多少的选手进入下以阶段，按照分数进行划分则划定可以进入下一阶段的评估总分数的最小值。

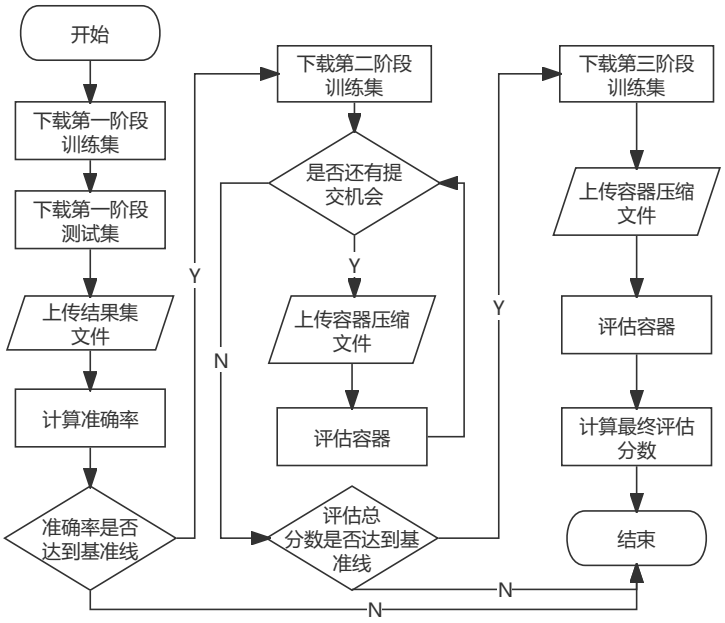

图 5-7: 比赛阶段总流程图

数据集设置 将系统中的数据集指定为训练集和测试集，在每个阶段选手在阶段开放后都可下载该阶段的训练集，而设置的测试集需要通过设置是否发布来决定选手是否可下载，对于发布的测试集在数据管理时无需指定其对应的变异数据集，而未发布的测试集可以选择用来评估鲁棒性及公平性变异数据集。目前司法多维评估系统中的赛程设置分为三个阶段，各阶段相关数据需要分别

设置。第一阶段发布训练集和测试集，选手提交模型在本地运行测试集后的结果集文件，结果集文件的输出需要按照特定的格式，提交结果集文件后，系统后台计算模型分类的准确率并将结果进行持久化存储，在达到设定的结果公示时间后，公布第一阶段排名。第二阶段与第一阶段差别较大，首先第二阶段需要提交容器压缩文件，容器需提供指定的健康检查命令和执行预测命令，其次第二阶段按照周期进行多次提交，按批次运行容器，综合最后一次评估结果决定进入下一阶段的参赛选手。第三阶段在第二阶段的基础上提高测试集的复杂度，同时只提供一次提交机会，最终结果排名综合第二阶段评估结果和第三阶段评估结果，各阶段总流程图如图 5-7所示。

5.3.2 阶段管理模块具体实现

司法多维评估系统中比赛管理员和参赛选手在登录后跳转页面不同，比赛管理员进入带有侧边栏的管理页面而参赛选手则进入比赛首页，比赛首页给出了三个阶段的入口，通过鼠标 hover 后的不同样式区分选手权限，三种不同权限样式表现对比如图 5-8所示，参赛选手访问登录接口登录成功后调用”/account/account” 接口拉取用户详情和”/stage/time” 接口拉取各阶段开放时间，用户详情中记录了选手当前可以进入的最高阶段，初始值都为 1 表示第一阶段。

(a) 可进入. (b) 无权限. (c) 未开放.

图 5-8: 三种阶段权限状态图

进入某一阶段后，页面以 tabs 方式排列三部分内容分别为规则说明，比赛流程和用户在当前阶段提交的文件，图 5-9展示了参赛选手视角的第二阶段比赛流程，以进度条的方式显示当前所处的流程，训练集下载作为比赛流程的第一步，其开放下载的时间以阶段开放时间为准；第二步是上传相关文件，在第

图 5-9: 阶段流程实现图

二阶段中即为上传运行容器压缩页面，在该页面中可查看第二阶段当前所处提交批次，同时使用前端封装的倒计时组件，对剩余提交时间进行提醒，在该组件中计时功能由独立的 worker 线程进行，通过 postMessage 向主线程传递计时信息，同时通过 onMessage 方法监听主线程传递的消息，在计时过程中不会受到主线程的影响保证了计时的准确性；第三步是结果公示步骤，在阶段管理设置的结果公示时间达到后显示选手提交的容器在当前批次截止时的排名。除此之外，用户可在“我的容器压缩文件”中查看已提交容器的状态和评估报告。

比赛管理员对阶段信息的管理主要在阶段管理页面中进行，阶段管理页面由时间设置、基准线设置及数据集设置三个部分组成，阶段管理页面添加阶段时间是实现自动化流程的关键，图 5-10展示阶段时间的设置项， 在 Stage

图 5-10: 阶段时间管理配置项

Module 中基于 Spring Boot 提供的 SchedulingConfigurer 接口封装了集中管理定时任务的工具类 DefaultScheduling，在微服务启动时自动生成实例并注入到系统中，该类提供了 addTriggerTask、cancelTriggerTask、resetTriggerTask、hasTask 等方法，方法对实例中的任务集合进行管理，被管理的任务有唯一 taskId，该 id 由新增定时任务时手动指定，若所指定的 id 已重复则新增任务会

覆盖已有的定时任务，任务新增后在指定的时间会执行相应方法。阶段时间包括阶段开启时间、提交开启时间、提交截止时间、结果公示时间和提交周期，设置定时任务时需基于具体的时间生成 Scheduling 可接收的 Cron 表达式，除提交周期外，其他类型时间的定时任务执行时都需更新对应的阶段状态和修改时间的生效状态，阶段状态控制用户在前端页面中可进行的操作依次实现自动化控制比赛流程的效果，生效状态则决定该类型阶段时间是否可以进行修改，具体时间类型节点需执行的定时任务如表 5-1所示。

表 5-1: 阶段时间类型任务详情

时间类型	Cron 表达式示例	执行任务
阶段开启时间	00 40 21 07 03	更新对应阶段状态为开启；更新生效状态
提交开启时间	00 00 08 08 03	更新对应阶段状态为提交开启；更新生效状态
提交截止时间	59 59 23 22 03	更新对应阶段状态为提交截止；更新生效状态；在异构服务器中写入阶段数据集信息；监听容器运行结束标识文件夹；调用该批次容器批量运行脚本
结果公示时间	00 00 08 29 03	根据基准值修改用户是否具有进入下一阶段的权限；更新生效状态
提交周期	00 00 17 * * SAT	在数据库中插入新的阶段批次记录；执行新批次的初始化任务；除状态更新外同提交截止时间

在新插入提交批次记录时的初始化任务由 storage 节点中的异构服务完成，初始化操作包括新建相关文件和启动该批次待导入容器列表文件的监听任务，初始化请求首先通过异构服务模块发送给 tracker 节点，tracker 节点的服务器中安装的 nginx2 应用可以识别请求是需要转发还是需要广播，判断是广播操作后由 tracker 节点分别调用各 storage 节点的的批次初始化接口，若初始化失败需记录失败原因，需管理员介入进行手动初始化。

阶段管理中的数据集设置与数据集模块相关联，每个阶段会公开该阶段所设置的训练集供用户下载，但阶段测试集的公开是可选择的，在第一阶段中测试集可公开下载，但提供的测试集文件并不是完整的数据集文件，而是由不包含分类结果的数据项的测试集组成，测试集设置后更新阶段记录中的标准结果集为该测试集对应的结果集文件。

第二阶段和第三阶段的测试集是不公开的，在选择对应的数据集文件作为测试集后，可指定该数据集中已经变异的鲁棒性测试集或公平性测试集作为附加测试集，测试集选择后调用 Upload Module 中的上传至 fastDFS 服务器中，上传状态可在数据集管理页面中查看，阶段数据集管理设置项如图 5-11所示。

图 5-11: 阶段数据集管理设置项

5.4 数据集管理模块

5.4.1 数据集管理模块详细设计

数据集管理模块实现比赛管理员对系统上传的数据集进行管理，系统中的数据集分为原始数据集和变异数据集两类，原始数据集是管理员通过文件管理模块上传的数据集，而变异数据集则是通过各类算法对原始数据集进行变异后得到。司法多维评估系统的司法指标包括准确性、泛化性、可参考性、鲁棒性及公平性，前三种评估指标属于基础性评估指标适用于所有数据集，而评估鲁棒性和公平性则需要通过数据变异技术对原始数据集进行符合司法领域的文本

变异。

目前评估鲁棒性引入的算法包括基于 TF-IDF 的特征裁剪、基于词库的随机插入及基于依存句法的特征变换 [38]，三种算法从删除、插入、变换顺序三个角度对数据进行变异，不同分类模型针对不同角度变换的敏感度是不同的，例如 fastText 模型对词序变换不敏感，但是 TextCNN 模型和多层的 LSTM 模型分类性能会受到比较大的影响 [39]，因此考察模型的鲁棒性就是评估模型对数据质量的依赖程度，依赖程度越低说明模型在司法场景中的应用场景就越多。公平性指标的设计主要是针对司法判案中存在的公平、公正问题提出的，相较于实现“个案正义”，“同案同判”才是当前司法模型在司法应用中的最高标准，公平性指标的评估使用的算法是基于语法的歧视性文本插入，在原始数据集中插入可能影响判决的歧视性文本后得到新的数据集，模型在两个数据集中的分类准确率按照公式进行转换的数值就代表了模型在公平性指标上的表现。

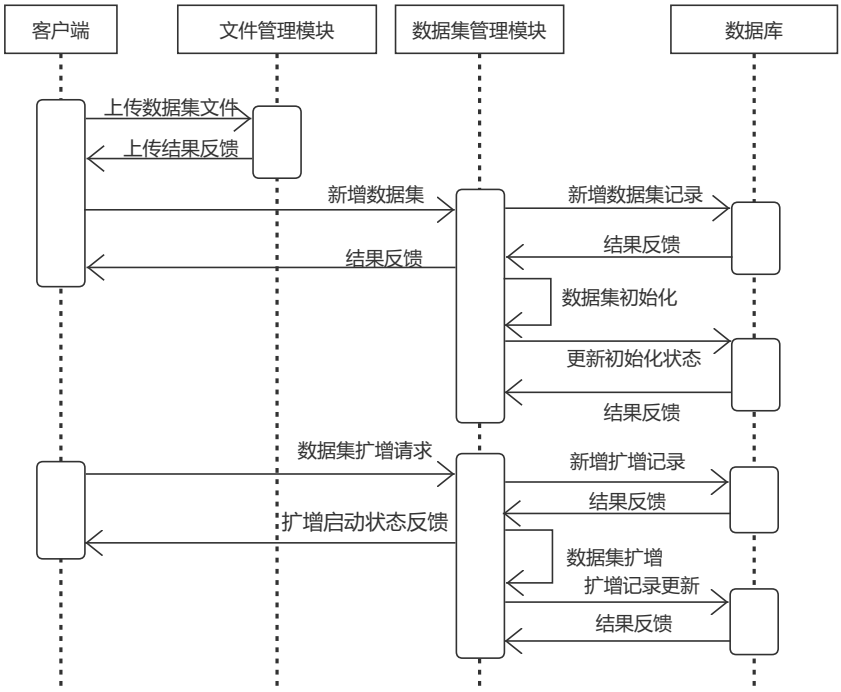

图 5-12: 数据集文件处理流程时序图

图 5-12展示了数据集模块中对数据集文件处理的时序图，数据集管理模块的主要功能包括管理业务逻辑服务器中的数据集文件和变异数据集，比赛管理员通过接口上传数据集文件，在新增数据集操作成功后对数据集文件进行初始化处理，初始化数据集是对数据项标签进行解析后生成测试集文件和结果集文件，可设置为训练集、测试集和可变异的数据集都需满足初始化成功的条件，

否则数据集不能在系统中使用。对数据集的变异操作由运行在业务逻辑服务器中的变异异构服务完成，数据集模块在接收变异请求后会进行一系列处理，包括生成变异参数、文件存放路径、基于不同的变异方法和数据集的状态调用不同的初始化方法处理完成后调用异构服务的接口并将对应参数传递过去，变异完成后异构服务将变异状态返回给数据集模块，数据集模块进行文件合并的相关操作后更新数据库中的记录。

5.4.2 数据集管理模块具体实现

数据集管理模块的具体实现围绕数据集新增和数据集变异两个大功能点展开，表 5-2 展示了两个大功能点中所调用的异构服务中接口详细情况，该异构服务部署在数据集微服务所在的服务器中，对数据集文件直接进行管理和处理。

表 5-2: 数据集模块中异构服务接口调用详情

功能	接口	含义	具体功能
数据集新增	init_dataset	数据集初始化	生成数据集文件对应的结果集和测试集文件
数据集变异	create_dictionary	构建词典文件	以数据集作为语料库生成词典文件和对应的向量文件
	amplify_dataset	变异方法	对业务模块中的参数进行处理调用具体的变异方法，变异方法包括基于 TF-IDF 的特征裁剪、基于依存句法的特征变换、基于词典的随机插入、基于语义的歧视性文本插入
	merge_dir	合并文件	一个指标中所指定的多种变异方法生成的变异集需合并为一个数据集

数据集初始化，系统中上传的原始数据集应符合特定的结构，否则无法初始化成功，数据项格式如图 5-13所示，fact 标签指定了该案例的事实描述

和司法过程，meta 标签中的 term_of_imprisonment 项指定了判刑详情，其中 death_penalty 表示死刑，life_imprisonment 表示无期徒刑，imprisonment 则表示具体的刑期。

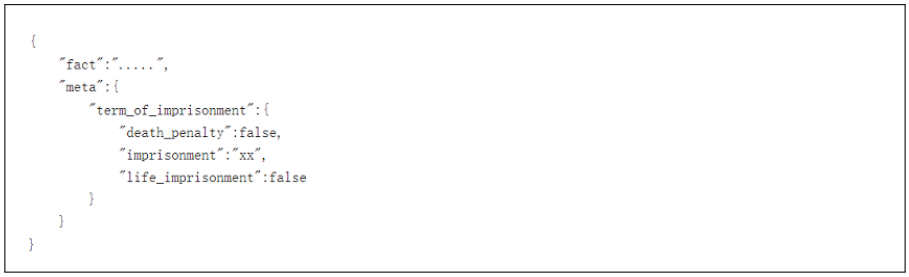

图 5-13: 数据项格式示例

生成测试集时是将每一个案例的 fact 标签中的内容提取处理，不同案例的 fact 以换行符进行分割组成测试集文件，测试集文件中除了案例的描述不包含其他信息，该文件除了作为实际测试集使用外，在变异相应原始数据集时也是以该测试集为准；结果集文件的生成也是遍历原始数据集中的每一个案例，解析 term_of_imprisonment 标签进行生成，若 death_penalty 和，life_imprisonment 对应的值为 true 表示在该案例中罪犯判死刑或无期徒刑，则将其分类归为 21，否则根据 imprisonment 字段获取刑期年数作为分类标签，各案例的分类标签也以换行符作为分割组成结果集文件，结果集文件用评估模型分类的准确率，同时所有变异指标的计算也以该结果集为标准。

原始数据只需初始化一次，异构服务初始化成功后将结果集文件和测试集文件的存储路径返回给数据集模块，数据集模块将其初始化至数据库中，若初始化失败则在数据库中更新该数据集的初始化状态，初始化失败的数据集无法在阶段管理中被设置为测试集和训练集。

词典文件生成, 词典文件在基于 TF-IDF 的特征裁剪和基于词典的随机插入中都会使用到，其中基于 TF-IDF 的特征裁剪也需提供以数据集作为语料库的向量集文件，故在生成词典文件时会同时生成向量集文件持久化至磁盘中。与数据集初始化不同，词典文件的生成采用延迟策略，即需要使用时才会生成相应的词典文件和向量集文件，同时以相同的数据集文件作为语料库生成的词典文件和向量集文件是唯一的，故在生成后可将其路径持久化至数据库中，供多次变异使用。

词典文件的生成基于 python 中提供的 corpora.Dictionary 实现，在进行生成词典之前需对语料库中的句子进行初始化操作，包括分词、去停用词、去非中

文词等操作，其中分词操作调用 ltp 库中提供的接口实现，在生成词典时会对语料库中只出现一次的词语进行过滤，有效降低词典文件的大小。

按照常规流程可对较小的语料库生成词典文件和向量集文件，但当数据集文件过大时很容易造成内存泄漏，作为测试集的数据集文件通常是以 MB 或 GB 为单位的较大文件，所以在生成词典文件之前首先会判断文件大小，若文件大小超过阈值则将其切分为多个等大的语料库文件，对每个文件生成其对应的词典文件和向量集文件，最后使用 merge_with 方法合并词典文件，使用遍历的方式合并向量集文件，合并后以文件的形式存储在系统中，将文件路径存贮在数据库中，对数据集进行多次变异时可以重复使用，分批式生成词典文件的核心代码如图 5-14所示。

```
if split>1:
    dir_name= os.path.dirname(output_path)+'/data_temp'
    # save_temp_file = os.path.dirname(output_path)+'/dic_temp.txt'
    save_temp_file = os.path.join(os.path.dirname(output_path),'dic_temp'+time.time()+'.txt')
    split_file(split,file_path,dir_name)
    file_list = os.listdir(dir_name)
    index = 0

    for file_path in file_list:
        file_name= dir_name+file_path
        if index == 0:
            corpus= create_dic_from_file(file_name,output_path)
        else:
            newCorpus= create_dic_from_file(file_name,save_temp_file)
            out_dic_entity=corpora.Dictionary.load("r"+output_path)
            temp_dic_entity= corpora.Dictionary.load("r",save_temp_file)
            out_dic_entity.merge_with(temp_dic_entity)
            #向量合并
            merged_corpus=[x for x in corpus]+[x for x in newCorpus]
            corpus = merged_corpus
            merged_corpus =None
else:
    corpus=create_dic_from_file(file_path,output_path)
#将向量序列化存储到磁盘中
corpora.MmCorpus.serialize(bow_output_path,corpus)
```

图 5-14: 分批生成词典文件代码

图 5-15: 数据集变异实现图

变异方法，系统中的变异指标包括鲁棒性和公平性，其中鲁棒性指标包括三种变异方法，公平性指标包含一种变异方法。方法参数由比赛管理员在发起变异任务时指定，数据集变异页面如图 5-15所示，在变异时首先选择原始数据集，数据集由比赛管理员上传，初始化成功的数据集可作为原始数据集使用，

数据集选择后可选择相应的评估指标，底部的变异方法由所选择的评估指标所决定，变异方法根据前端设置的 json 文件动态渲染，变异方法参数默认显示最优参数，比赛管理员可通过修改变异参数决定变异数据集的复杂度，数量占比项指定了该变异方法变异的数据在最终变异数据集中的占比，变异方法中数量占比之和不超过 100%，前端对变异参数的合法性进行验证后在提交至后端进行相应的变异任务。

提交至后端的变异参数由业务逻辑服务器中的数据集模块进行处理，若变异任务的评估指标类型为鲁棒性则在该次变异任务中可能包含多种变异方法，数据集模块在解析参数后循环遍历其中所有的变异方法通过 java 类中的反射机制赋值 CreateAmpCommand 对象，该对象是向异构服务中传递数据的载体，与异构服务中的请求对象一一对应。向异构服务发起的变异请求以变异方法为单位，不同的变异方法需要进行不同的初始化操作，例如基于 TF-IDF 的特征裁剪和基于词典的随机插入方法首先需要判断数据集的词典文件是否已生成，若词典文件已生成则将数据库中查询到的词典路径作为变异请求中的参数发送给异构服务，否则需要按照上文所属的步骤生成词典文件，除此之外基于 TF-IDF 的特征裁剪方法还需根据已生成的词典文件向异构服务发起请求构建 TF-IDF 模型，模型生成后将其路径存储在数据库中，在下次发起该方法的变异请求时直接复用模型即可。

数据集变异过程中会读取原始数据集的测试集文件和结果集文件，变异操作主要基于测试集文件，当指定的某一变异算法运行完成后会生成该变异算法的测试集文件，然后根据变异范围截取对应的变异结果集文件。一个变异任务中可包含多种变异算法，当变异任务中的所有变异算法完成后会合并按顺序合并所有的测试集文件和结果集文件，并将这两个文件以文件实体的形式存储在数据库的 file 表中，方便后续被设置为阶段数据集时统一在 dfs 中进行管理。若一个变异任务中的任一变异算法变异失败，则该变异任务的对应状态也会修改为变异失败，需要管理员根据失败信息进行排查。

在业务逻辑服务器中的异构服务实现了四种变异方法，包括基于 TF-IDF 的特征裁剪、基于依存句法的特征变换、基于词典的随机插入和基于语义的歧视性文本插入。

变异方法在执行时按行读取原始数据集对应的测试集中的案例描述作为待变异文本，生成变异文本之后使用换行符作为分隔符存储在文件中，变异文件路径由变异请求中传递的测试集指定，变异成功后生成读取参数中给出的原始

阶段集路径对应的文件，截取变异范围对应的结果集后生成变异结果集存储在指定的路径中。

合并文件，对于鲁棒性评估指标而言，在一次变异任务中可能存在多个变异方法，每个变异方法的目标变异文本是原始数据集中的一部分，生成后的变异文本也不是完整的变异数据集，所以在所有的变异请求返回后，对于包含多种变异方法的变异任务还需进行文件合并，同一次变异任务生成的变异数据集放在一个文件夹下，变异结果后将文件夹下的所有变异方法变异出来的文件合并为一个完成的变异测试集文件，从而生成一个完整的变异数据集，除此之外变异结果集也需要进行合并，变异文件合并的核心代码如图 5-16所示。

```
def merge_file_dir(goal_dir,result_file):
    file_names = os.listdir(goal_dir)
    try:
        with open(result_file,'a') as f:
            for file in file_names:
                file_path = os.path.join(goal_dir,file)
                with open(file_path,'r') as source_file:
                    lines = source_file.readlines()
                    f.writelines(lines)
    except:
        result = result_util.generate_error_result("合并文件失败",500)
        return Response(json.dumps(result),mimetype='application/json')

    result = result_util.generate_success_result("合并文件成功")
    return Response(json.dumps(result), mimetype='application/json')
```

图 5-16: 合并文件核心代码

5.5 容器管理模块

5.5.1 容器管理模块详细设计

在基于司法多维评估的模型评估系统中用户需将模型包装为 docker 容器上传至系统中，通过这种方式保证模型具有持续部署和运行的能力，降低环境不兼容的造成的运行失败的风险 [40]。docker 中有两个核心概念：镜像和容器，容器时镜像的动态运行模式，类似于面向对象语言中的对象和类的概念，docker 导出压缩文件的方式一共有两种，一种是基于镜像使用 docker save 命令，另一种是基于容器使用 docker export 命令，由于使用 save 方式打包的镜像

文件包含了镜像的历史记录和元数据等信息故其打包后的文件一般大于使用 export 命令打包的文件，同时为了避免在容器导入时名称覆盖需要在导入时对容器进行重命名，保证容器名称的独立性，使用 save 方式打包的文件无法对镜像进行重命名，而通过 import 命令可以直接重命名打包的文件，综上在系统中上传的容器需以 export 方式进行导出。

容器压缩文件通过文件管理模块进行上传，上传至业务服务器后需上传至异构服务器进行线性冗余，在异构服务器中分别对容器进行导入、健康检查和运行操作，容器导入是指将容器压缩文件导入到服务器中，容器导入后才能执行容器中的命令，健康检查是针对容器在宿主服务器中文件读写能力的检查，因为在评估模型时容器需要读写宿主服务器中的文件，通过健康检查可以保证在评估运行时数据集的读写没有问题。

具体的业务逻辑对三个操作的执行时机有不同的要求，首先容器导入和健康检查应具有实时交互能力，选手上传容器后应立即对容器进行导入和健康检查，导入和健康检查的结果应及时反馈给用户，若操作结果不成功，需将错误信息反馈给用户，比赛管理员或参赛选手针对错误信息检查系统或容器，在符合实时性的基础上还需考虑服务器中 docker 客户端的负载能力，若并发操作过多将导致 docker 主进程的崩溃，故容器导入和健康检查需采取节流的执行方案；容器运行则在由具体的定时服务触发，当阶段或批次提交时间截止时定时服务调用异构服务器中的接口触发批量运行容器的脚本，因此在详细设计时无需考虑容器运行的实时性。

容器上传异构服务器成功后，异构服务器中提供的导入和健康检查服务由四个步组成。第一步是信息接收，提交容器的用户 id 和存储在数据库中容器实体 id 将传输给异构服务器，同时将信息写入待导入容器列表文件中；第二步是决定是否触发容器批量导入，该部分监听待导入容器列表文件是否变化，在第一步修改了待导入容器列表文件后，将触发第二步的判定，第二步判定的原则是若该导入申请前由其他未完成的导入任务则直接退出，否则则判断是否正在执行导入任务，若有则循环等待其执行完毕，若无则直接执行批量导入任务；第三步是并发执行待导入容器列表文件中未导入容器的导入任务，该步骤由 shell 脚本完成，由于未导入容器数量未定所以需要控制同时执行导入步骤的容器数量；第四步是完成导入成功容器的健康检查，第三步执行容器导入的结果会存储在特定文件中，每一行代表一个容器导入的结果，第四步根据每一行记录的容器状态选择是否进行健康检查，若进行则执行相应命令并判断健康检查

结果是否无误，无论是否进行健康检查及健康检查的结果如何在第四步最后都需要将容器状态更新到数据库中，以上四步顺序图如图 5-17所示。

图 5-17: 容器导入及健康检查顺序图

容器运行是完成模型评估的关键步骤，容器运行命令要求选手将模型分类结果写到宿主服务器中的对应文件内，容器运行由业务逻辑服务其中的定时任务进行流程控制，当设定的提交截止时间到达后将调用异构服务器中的容器批量运行评估接口，容器批量运行与容器导入类似，在考虑服务器中 docker 负载的基础上实现总运行时长最小化的目标，容器运行后需更新数据库中的容器状态以便管理员实时监控运行流程，服务器中容器所处状态及更新时机如表 5-3所示。

表 5-3: 容器状态值及含义

值	含义	更新时机
0	未导入	初始
1	导入及健康检查成功	健康检查后
2	导入及健康检查失败	导入或健康检查后
3	运行及评估成功	结果评估后
4	运行及评估失败	运行或结果评估后

5.5.2 容器管理模块具体实现

容器管理模块的具体实现主要聚焦于容器导入及健康检查、容器运行及结果评估，两个子模块的关键实现都部署在异构服务器中，按照功能实现的相应步骤以 flask 接口和脚本进行实现，下面将依次说明的两个子模块的具体实现。

容器导入及健康检查 在系统中上传容器压缩文件后，首先以文件的形式存储业务逻辑服务器中，之后容器管理模块调用文件管理模块中的“/file/upload/dfs/{fileId}”接口将容器压缩文件上传至 fastDFS 中，上传成功后将返回与“group1/M00/00/00/rBPwGmJ4jkeAHkb5OPd2AHNu9uk373.tar”相似的存储路径，该路径中的“group1/M00”表示初始存储该容器压缩文件

的 fastDFS storage server 所在的组和名称，fastDFS 会对处于相同组的 storage server 中的数据进行同步，但是同步过程是异步的，所以在初始导入该容器压缩文件时必须调用其最初返回的路径存储服务器上的异构服务导入接口。在 outService 微服务中，对所有异构服务器上的异构服务器的调用都是访问的 fastDFS tracker 所在的服务器，在该服务器中安装了两个 nginx 服务，一个 nginx 服务与 fastDFS 相配合，实现 storage 节点的负载均衡，避免同一 storage 节点在短时间内承载过重；而另一个 nginx 服务则是对当前服务器中接收到的请求进行转发和重写。

在业务逻辑服务器发送给异构服务器中的请求分为两种，一种是需要 tracker 所在的服务器广播给其所控制的所有 fastDFS storage server 节点，这种请求的 url 中不会带上具体 storage server 的名称，例如在阶段提交开启或提交周期开启时要启动待导入容器清单文件的监听而调用的“/watch/import”接口，当 nginx 服务接收到请求后将调用所有的 storage server 中的“/watch/import”接口，从而使得所有的 storage server 都处于可导入状态，而另一种是不需要 tracker server 进行广播，其需要对接收到的请求转发给具体的 storage server 并在转发之前对 url 进行重写，这种情况下发送给 tracker server 的 url 中带有具体的 storage server 的名称，例如“/docker/import/{hostName}”请求中的 hostName 就是用上文提到的“group1/M00”进行替换，在 nginx 服务的 config 文件中会带有具体名称的 url 进行拦截，然后使用 nginx 提供的 rewirte 功能将 hostName 从 url 中进行移除，然后将新的 url 发送给具体的 storage server，该 nginx 服务的核心配置如图 5-18所示。

在上传至异构服务器完成后，业务逻辑服务器中的容器管理微服务会调用部署在异构服务器上的 flask 服务的“/docker/import”接口，该接口接收相关容器信息并将其写入待导入容器列表文件中，列表文件的每一行表示一个待导入容器的信息，包括容器 id、容器压缩文件存储路径、用户 id，不同字段以空格进行分割。不同批次的容器写入不同的待导入文件中，在一个提交周期开始时（比赛第二阶段包含一个提交周期，第三阶段包含一个提交周期），会在批次初始化时新建一个该批次待导入文件，并调用异构服务器的接口监听该文件所处的文件夹，启动监听服务代码如图 5-19所示。

当识别到待监听文件夹路径中包含“import”字段时就会将对应的监听处理类设置为 ImportEvenHandler，该类实现了 pynotify 类中 process_IN_MODIFY 接口，在待导入容器列表文件写入容器信息时，将会触发 ImportEvenHandler 中

```
upstream group1_M00 {
    server 172.19.240.26:5000;
}


server {
    listen       8089;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
    }

    location  ~* (group1_M00)$ {
        proxy_pass http://group1_M00;
        rewrite ^/(.*)/group1_M00$ /$1 break;
       # proxy_pass http://group1_M00;
        keepalive_timeout 60s;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

图 5-18: nginx 调度服务配置

的 process_IN_MODIFY 方法，在这个方法中采用“防抖”的方式控制短时间内批量容器导入脚本的调用，该方法的判断逻辑如图 5-20所示， 其主要作用是控制批量导入脚本文件不要重复和过多执行，因为容器批量导入脚本文件会将当前待导入容器列表文件中所有未导入的容器进行导入，所以对于短时间内的多个导入请求会在一次批量导入任务中完成无需多次调用批量导入脚本。通过批量导入的方式可以控制并行导入的容器数量，避免服务器中的 docker 因为短时间的请求过载而崩溃，流程图中“等待 5 秒”的操作就是为了聚集短时间内的多次导入请求，批量导入脚本在每次批量导入后会将最后一个导入容器在待导入列表文件中所处的行序号写入“/data/import/row/{batch_id}.txt”文件中，在下次执行该脚本时会首先读取行序号文件获得当前批量导入的起始位置，从

```
@staticmethod
def start_watch(dir,stage_id):
    eh = ""
    wm = pyinotify.WatchManager()
    print("监听文件名为",dir)
    wm.add_watch(dir, pyinotify.ALL_EVENTS, rec=True)
    if not stage_id is None:
        WatchFileChange.stage_id=stage_id
    if "import" in dir:
    #初始化路径值
        eh = WatchFileChange.ImportEventHandler()
    elif "evaluate" in dir:
        #启动评估操作
        eh = WatchFileChange.EvaluateEventHandler()

    if not eh=="":
        WatchFileChange.notifier = pyinotify.ThreadedNotifier(wm, eh)
        print("before watch")
        WatchFileChange.notifier.start()
        print("after watch")
        return str(1)
    else:
        return str(0)
```

图 5-19: 启动待导入文件监听服务

该起始位置到当前列表文件最后一行所记录的容器记录都会在此次批量导入任务中进行导入，在这个过程中通过文件描述符和命名管道来控制每次并发导入的进程数量，并发数量可根据 docker 客户端的负载能力进行调整，目前脚本中设置的并发数量为 5，表示在容器批量导入脚本执行的过程中最多只有 5 个容器可以并发导入，批量导入脚本文件的关键代码如图 5-21所示。

在批量导入脚本执行过程中，对待导入容器文件的操作是读取，在这个过程中会有新的导入请求发送到异构服务器中，将容器的信息写入待导入列表文件，读取和写入操作是互斥的，并且多个请求同时到达异构服务器时会同时对待导入列表文件进行写入，同一时刻的写入操作也是互斥的，对文件的互斥操作可能会导致文件信息的缺失，因此在调用批量导入脚本前和写入容器信息前应对待导入列表文件加锁，从而保证列表文件的完整性

导入容器后的结果写入导入成功列表文件中，在导入后进行健康检查，按行读取该文件，判断容器的读写能力，健康检查的命令需要参赛选手按照规定写在上传的容器中，健康检查的具体步骤是执行该命令将对应的宿主机中的输

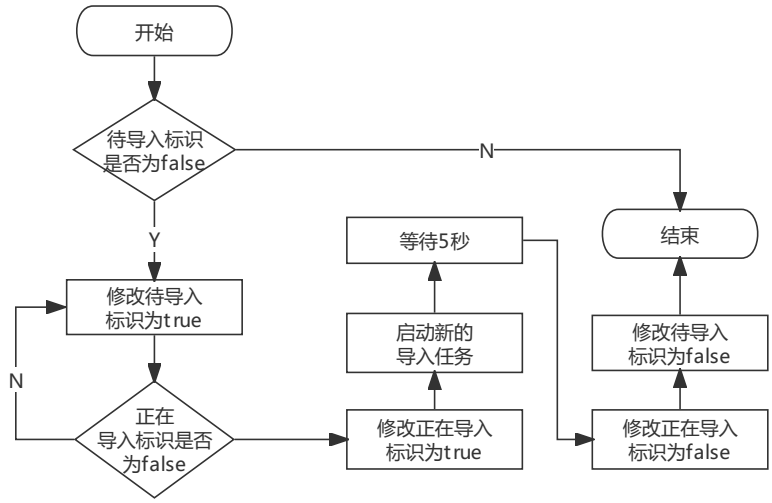

图 5-20: 容器导入触发逻辑流程图

```
while read line
do
  if [ ! $cur_row -lt $START_ROW];then
  read -u12
  {
    array = (${line//,/ })
    user_id = ${array[0]}
    image_id = ${array[1]}
    image_path = ${array[2]}
    new_image_name = "${user_id}_${image_id}"
    result_msg = $(source /data/import_docker.sh ${image_path} ${new_image_name} 2>&1)
    if [[ "$result_msg" =~ ^sha256.* ]]; then
       echo "${image_id} 1 ${result_msg}">$RESULT_FILE
    else
       echo "${image_id} 2 -1">$RESULT_FILE
       echo ${result_msg} > "/data/error/${image_id}.txt"
  }&
  fi
 cur_row = $((cur_row+1))
done<$IMAGE_IMPORT_DES
```

图 5-21: 容器批量导入脚本核心代码

入文件路径和输出文件路径作为参数传入容器中，输入文件中的内容是随机数，而输出文件的内容则为空，执行命令后判断输出文件的内容与输出文件的内容是否一致，若一致则表示容器健康检查成功，否则容器健康检查状态为失败，健康检查脚本的关键代码如图 5-22所示，/data/scripts/health_random.sh 脚本将生成随机数并把该值写入/data/health/random/${image_id}.txt 文件中，写入完成后根据容器名称使用”docker run” 命令执行容器，并将宿主机的/data 目录挂载到容器中，在启动容器时指定完整的容器运行命令，运行容器中

的/code/app/file.py 文件，将”health_check” 指令作为参数传入该程序，该程序会调用”health_check” 方法，并将输入文件中的值写入输出文件中，执行完成后通过” diff $random_input $random_output > /dev/null” 对比输入文件和输出文件的内容是否一致，若内容一致则说明容器健康检查成功，否则容器健康检查失败。判断后调用/data/update_database.sh 脚本对容器状态进行更新，若容器健康检查成功则将该容器名称追加到该批次健康检查成功列表文件中。健康检查的目的可以保证容器在导入成功的前提下可以正常地运行，同时也可以判断容器是否具有读写宿主机文件的能力，从而为后续的容器运行提供保障。

```
array=(${line//,/ })
image_id=$((10#${array[0]}))
import_status=$((10#${array[1]}))
container_name=${array[2]}
random_input="/data/health/random/${image_id}.txt"
random_output="/data/health/output_random/${image_id}.txt"
if [ ! -f "${random_input}" ];then
   touch ${random_input}
fi
if [ ! -f "${random_output}" ];then
   touch ${random_output}
fi
echo "$import_status"
if [[ "$import_status" -eq "1" ]];then
    # do check
    . /data/scripts/health_random.sh ${random_input}

    docker run  -v /data:/data --rm ${container_name} python /code/app/file.py  health_check ${random_input} ${random_output} 2>$1

    diff $random_input $random_output > /dev/null
    if [ $?==0 ];then
        # change database
        #. /data/update_database.sh 2 ${image_id}
        echo "heath check success"
        echo "${container_name}">>${HEALTH_RESULT_SUCCESS_FILE}
    else
        #. /data/update_database.sh 3 ${image_id}
        echo "health check fail"
    fi
else
        echo "import fail"
        #./data/update_database.sh 1 ${image_id}
fi
```

图 5-22: 容器健康检查关键代码

容器运行及结果评估 容器的批量运行是在阶段的提交周期结束或提交截止时间到达后进行的，在批量运行之前首先会调用异构服务器上的”/write/dataset” 接口，将该阶段的数据集文件信息写入异构服务器的”/data/dataset/${stage}.txt” 文件中，该文件中每一行信息由数据集类型、测试集在异构服务器上的存储路径以及结果集在异构服务器上的存储路径组成，如果该数据集是变异数据集那么还会包含该变异数据集的变异因子。文件内容如图 5-23所示，该文件中的第一行数据是原始测试集的信息；第二行数据是鲁棒性测试集的信息；第三行数据是公平性测试集的信息。

阶段数据集信息写入后，通过调用异构服务器上的”/start/exec”flask 接口

```
2 /data/fastdfs/storage/data/00/00/rBPwGmJ5EnmAQf9EAAAiw4L93JI954.txt /data/fast
dfs/storage/data/00/00/rBPwGmJ5EnmALiFjAAAAEzHA7Zo240.txt
3 /data/fastdfs/storage/data/00/00/rBPwGmJ5EnmAR7J1AAAAGSwL2Q8522.txt /data/fast
dfs/storage/data/00/00/rBPwGmJ5EnmAR7J1AAAAGSwL2Q8522.txt 0.0112655
4 /data/fastdfs/storage/data/00/00/rBPwGmJ5EnmAFCyhAAAjGgXcewg286.txt /data/fast
dfs/storage/data/00/00/rBPwGmJ5EnmAGD8YAAAAEzHA7Zo394.txt 0.030103
```

图 5-23: 异构服务器上的阶段数据集记录文件

执行该批次批量运行容器任务，在调用批量运行脚本之前，接口函数会首先调用图 5-19所示的启动监听函数，监听的路径是评估启动标识文件夹，该文件夹中包含 evalate 字段，在启动监听函数中若识别到路径中包括该字段会将对应的监听处理类设置为 EvaluateEvenHandler 类。每个容器会在执行完成后生成评估启动标识存放在启动运行时指定的评估启动标识文件夹下，当 EvaluateEvenHandler 类监听到该文件夹下有新的文件生成时，说明某一容器完成了一个测试集任务，通过解析新生成的文件夹名称可以获得容器 id、批次 id 和测试集类型，由这三个字段拼接获得容器运行后的结果集文件，通过调用评估程序生成对应的指标值，并将指标值更新到数据库中，EvaluateEvenHandler 类的核心代码如图 5-24所示，其中 calculate.init 是根据测试集类型计算指标值的具体函数。

```
class EvaluateEventHandler(pyinotify.ProcessEvent):
    def process_IN_CREATE(self, event):  # 函数名以"process_"开头,后面跟注册的监测类型
        try:
            # os.system('echo '+'create file:%s'%(os.path.join(event.path,event.name))) #之后用于nohup输出
            print("new file: %s " %os.path.join(event.path, event.name))
            # name should include container_name(batch_id container_id) test_type(origin 2 lub 3 fair 4)
            array = event.name.split("_");
            if len(array)==4:
                batch_id = array[0]
                container_id = array[1]
                test_type = array[2]
                #调用扩增方法
                print("========evaluate result========",batch_id,container_id,test_type)
                result = calculate.init(test_type,WatchFileChange.stage_id,batch_id,container_id)
                if not result is None:
                    if result[su.result_success_key] is False:
                        msg = result[su.result_message_key]
                        database_util.update_container_status(container_id,6,msg)
                        if not os.path.exists(EVALUATE_LOG_DIR):
                            os.makedirs(EVALUATE_LOG_DIR)
                        log_file= os.path.join(EVALUATE_LOG_DIR,batch_id+".txt")
                        with open(log_file,'a')as f:
                            f.write(container_id+" -1 "+msg)
        except:
            traceback.print_exc()
            print('evaluate error')
```

图 5-24: 结果集监听评估核心代码

启动监听函数后，将获取该批次健康检查成功列表文件，并判断该文件是否存在，若该文件存在则将该文件路径作为参数传递给批量运行脚本/data/scripts/process_exec_controll.sh，在该脚本中首先会根据该批次所属的阶段数据集文件获取原始测试集和变异数据集，然后遍历健康检查成功列表文件的每一行，根据每一行的容器名称运行容器，容器运行命令如图 5-25所示。与健康检查不同的是，该命令向 file.py 程序中传递的指令是 exec_model 参数，该命令会调用 exec_model 函数，在该函数中会将数据集文件和输出文件进行赋值，参赛选手只需在该函数中调用具体的司法模型即可，模型运行结果输出后，生成对应的评估启动标识文件即可。

```
docker run -v /data:/data  --rm  ${container_name} python /code/app/file.py exec_model ${ORIGIN_TEST_PATH} /data/exec_result/${BATCH_ID}/${image_id}/exec_2.txt ${BATCH_ID}/
${container_name}_2 2>&1
```

图 5-25: 批量运行容器脚本中的容器运行命令

容器模板会在系统中提供给参赛选手，将模型打包成容器压缩文件只需完成以下五个步骤即可：

(1) 将训练好的模型文件放在指定的项目文件夹下。
(2) 在 Dockerfile 文件中添加启动模型必要的依赖。
(3) 在 exec_model 函数的指定地方添加模型运行语句。
(4) 将项目打包成 docker 镜像，并导入至自己本地的系统中。
(5) 在本地进行测试，执行上文所述的健康检查命令和容器运行命令，没有错误后将容器使用 export 命令打包成容器压缩文件。

5.6 本章小结

本章对基于司法多维评估的模型评估系统的认证授权模块、文件管理模块、阶段管理模块、数据集管理模块及容器管理模块的详细设计和具体实现进行阐述，通过关键步骤的流程图和模块交互的时序图解释各模块详细设计细节和具体代码实现的依据，在司法多维评估模型的各模块具体的实现中展示了该模块关键方法的核心代码和部分页面的实现效果图。

第六章　系统测试与分析

本章对上文设计和实现的基于司法多维评估的模型评估系统进行系统测试，测试范围包括功能性测试和非功能性测试。功能性测试根据第三章给出的用例进行测试，包括数据集下载、新增结果集、新增容器、容器运行及评估、新增数据集、数据集变异、阶段规则设置、阶段时间设置、阶段基准线设置和阶段数据集设置等功能；非功能性测试根据非功能性需求进行，主要考察系统的响应时长、负载能力等特性。除此之外，本章使用第三章提出的司法多维评估体系对司法模型进行测试并获取各指标的评估分数，在此基础上给出模型的优化方向。

6.1　测试环境

本系统部署在学校提供的两台云主机上，一台作为部署微服务和前端服务的业务逻辑服务器，另一台作为部署 fastDFS 分布式文件管理服务和 flask 服务的异构服务器。在业务逻辑服务器上部署微服务和前端服务；异构服务器既作为 fastDFS 中的 tracker 节点又作为 storage 节点，tracker 节点在调度时进行负载均衡选择相应 storage 节点，同时 flask 服务通过 Gunicorn 服务器软件部署，使用 nginx 做反向代理和静态文件托管，容器导入等操作在异构服务器上进行，故异构服务器需安装 Docker 客户端，表 6-1展示了系统测试环境配置信息。

表 6-1: 系统测试环境信息表

字段	测试环境信息
云主机	CPU：QEMU Virtual CPU version 2.5+ 内存：4GB 操作系统：Linux CentOS-7.4
数据库	关系型数据库：Mysql 8.0.23 非关系型数据库：Redis 5.0.10
其他软件	Docker 20.10.10、fastDFS 0.9.2、Gunicorn 20.1.0、Maven 3.8.4、Nginx 1.9.9、nodejs 14.15.3

6.2 功能性测试

系统的功能性测试是在系统功能性需求上对系统实现效果进行测试和评估，进行概要设计、详细设计和具体实现后的系统在理论上可以完成系统功能性需求的交互要求和逻辑要求，但在具体用例实现和用例之间的逻辑连贯方面可能存在不可预见的缺陷，寻找和解决缺陷是系统功能性测试的目标。本部分针对 10 个系统用例进行测试，下文将详细说明所有用例的功能测试实现情况。

表 6-2展示了数据集下载用例的测试描述，在三个比赛阶段中都涉及到数据集下载用例，在测试过程中覆盖了三个阶段的测试集或训练集下载。

表 6-2: 数据集下载用例测试结果

测试项	操作	实现效果	测试结果
下载第一阶段训练集	1.参赛选手登录系统 2.进入比赛第一阶段 3.点击下载训练集按钮 4.打开下载后的文件	1.训练集文件下载至本地 2.文件内容包括事实标签及分类结果	通过
下载第一阶段测试集	1.参赛选手登录系统 2.进入比赛第一阶段 3.点击下载测试集按钮 4.打开下载后的文件	1.测试集文件下载至本地 2.文件内容仅包括事实标签	通过
下载第二阶段训练集	1.参赛选手登录系统 2.进入比赛第二阶段 3.点击下载训练集按钮 4.打开下载后的文件	1.训练集文件下载至本地 2.文件内容仅包括事实标签	通过
下载第三阶段训练集	1.参赛选手登录系统 2.进入比赛第三阶段 3.点击下载训练集按钮 4.打开下载后的文件	1.训练集文件下载至本地 2.文件内容仅包括事实标签	通过

表 6-3展示了新增结果集功能相关的测试用例，第一阶段中的提交流程要

求新增结果集文件，用户在提交结果集后可在我的结果集页面中查看提交的结果集和状态，同时管理员用户也可在对应的阶段结果集管理页面中查看结果集详情和实时的排名情况。

表 6-3: 新增结果集用例测试结果

测试项	操作	实现效果	测试结果
新增结果集	1.参赛选手登录系统 2.进入比赛第一阶段 3.进入提交结果集流程 4.点击提交结果集按钮 5.在弹出的文件选择框选择结果集文件 6.进入我的结果集 tab 页面	1.我的结果集 tab 页中可查看提交的结果集 2.刷新页面结果集解析完成，查看准确率	通过

表 6-4描述了新增容器用例的测试详情，在第二阶段和第三阶段都会进行新增容器操作，容器新增后可在我的容器页面查看容器详情，容器上传 fastDFS 状态和导入健康检查状态可通过刷新页面查看。

表 6-4: 新增容器用例测试结果

测试项	操作	实现效果	测试结果
第二阶段新增容器	1.参赛选手登录系统 2.进入比赛第二阶段 3.进入提交容器流程 4.点击提交容器按钮 5.在弹出的对话框中填写容器描述信息，并上传容器压缩文件 6.点击新增确定按钮 7.进入我的容器页面	1.我的容器页中可查看新增的容器 2.刷新页面，显示新增容器的 fastDFS 地址 2.刷新页面，显示新增容器状态为导入及健康检查成功	通过

表 6-5描述了测试容器运行及评估用例详情，容器运行及评估触发由系统设定的时间节点自动触发调用异构服务中的接口，在第二阶段的提交周期节点

及第三阶段的提交截止节点到达时调用批量运行及评估接口，调用后通过刷新页面即可查看运行及评估状态及相关信息。若服务器中的容器导入或者健康检查失败，则容器不在批量运行和评估的列表文件中，无法对其进行评估。除此之外若容器运行失败，也无法调用评估脚本对其进行评估。容器运行评估完成后用户可查看相应的评估报告并下载。

表 6-5: 容器运行及评估用例测试结果

测试项	操作	实现效果	测试结果
第二阶段运行及评估容器	1.比赛管理员登录系统 2.进入第二阶段比赛管理页面 3.设置提交周期为当天并设置时间为十分钟后 4.参赛选手登录系统 5.进入第二阶段 6.在当前提交批次中新增容器	1.我的容器页中可查看新增的容器 2.一段时间后刷新页面显示容器评估运行状态及评估结果	通过
第三阶段运行及评估容器	1.比赛管理员登录系统 2.进入第三阶段比赛管理页面 3.设置提交截止时间为十分钟后 4.参赛选手登录系统 5.进入第三阶段 6.在流程中新增容器	1.我的容器页中可查看新增的容器 2.一段时间后刷新页面显示容器评估运行状态及评估结果	通过

表 6-6描述了测试新增数据集用例的详情，上传后系统进行初始化操作生成测试集文件和结果集文件，上传至系统的原始数据集需按照相应的格式包含对应的字段，否则无法初始化成功，初始化成功是数据集可在系统中使用的关键，初始化成功的数据集可作为阶段的测试集和训练集使用，同时可基于该数据集进行变异，若初始化失败可查看相应的信息修复后重新新增。

表 6-6: 新增数据集用例测试结果

测试项	操作	实现效果	测试结果
新增数据集	1.比赛管理员登录系统 2.进入数据集管理页面 3.点击新增数据集按钮 4.填写数据集信息并上传数据集文件 5.点击新增确定按钮	1.数据集管理列表中可查看新增的数据集 2.一段时间后刷新页面显示测试集和结果集初始化状态为成功	通过

表 6-8描述了变异数据集用例的测试详情，数据集变异基于的原始数据集是初始化成功的原始数据集，变异类型包括鲁棒性评估指标变异和公平性评估指标变异，在测试中分别对两种变异类型进行测试用例的设计和操作。

表 6-7: 变异数据集用例测试结果

测试项	操作	实现效果	测试结果
变异鲁棒性测试集	1.比赛管理员登录系统 2.进入数据集管理页面 3.点击变异数据集按钮 4.选择原始数据集 5.选择变异指标为鲁棒性 6.点击添加变异方法按钮 7.选择基于 TF-IDF 的特征裁剪方法并填写参数 8.点击添加变异方法按钮 9.选择基于依存句法的特征裁剪方法并填写参数 10.点击添加变异方法按钮 11.选择基于词典的随机插入方法并填写参数	1.数据集管理列表中可查看新增的鲁棒性变异数据集 2.一段时间后刷新页面，显示变异数据集存储路径。	通过

变异公平性测试集	1.比赛管理员登录系统 2.进入数据集管理页面 3.点击变异数据集按钮 4.选择原始数据集 5.选择变异指标为公平性 6.点击添加变异方法按钮 7.选择基于语法的歧视性文本插入方法并填写参数	1.数据集管理列表中可查看新增的公平性变异数据集 2.一段时间后刷新页面，显示变异数据集存储路径。	通过

表 6-8描述阶段规则设置用例的测试详情，阶段规则以富文本的形式由比赛管理员设置，参赛选手可在各阶段中查看，规则设置应遵守简洁明了，通俗易懂的风格，在说明时应对重点信息进行强调，避免参赛选手错过或错看重要信息，同时也要注意排版和格式问题。

表 6-8: 变异数据集用例测试结果

测试项	操作	实现效果	测试结果
阶段规则设置测试用例	1.比赛管理员登录系统 2.进入第一阶段规则管理页面 3.编辑阶段规则 4.点击确定按钮 5.参赛选手登录系统 6.进入第一阶段	1.参赛选手可查看阶段规则	通过

表 6-9描述了阶段时间设置用例的测试详情，阶段时间时控制自动化筛选流程的关键，阶段时间测试分为阶段开启时间测试、阶段提交开启测试、阶段提交截止测试、阶段结果公示时间测试、提交周期测试，前四项测试以第一阶段为例，各阶段的测试结果相同，提交周期只在第二阶段中进行了设置，所以对提交周期的测试以第二阶段为例。

表 6-9: 设置阶段时间用例测试结果

测试项	操作	实现效果	测试结果
阶段开启时间设置测试用例	1.比赛管理员登录系统 2.进入第一阶段阶段管理页面 3.设置阶段开启时间为十分钟后 4.点击确定按钮 5.参赛选手登录系统 6.点击进入第一阶段	1.参赛选手无法进入第一阶段，卡片显示暂未开放 2.十分钟后刷新页面后可进入第一阶段，并可下载训练集	通过
提交开启时间设置测试用例	1.比赛管理员登录系统 2.进入第一阶段阶段管理页面 3.设置提交开启时间为十分钟后 4.点击确定按钮 5.参赛选手登录系统 6.点击进入第一阶段 7.进入提交结果集流程	1.参赛选手无法下载测试集 2.进入提交流程显示暂未开放 3.十分钟后刷新页面可下载测试集 4.进入提交流程显示提交结果集按钮	通过
提交截止时间设置测试用例	1.比赛管理员登录系统 2.进入第一阶段阶段管理页面 3.设置提交截止时间为十分钟后 4.点击确定按钮 5.参赛选手登录系统 6.点击进入第一阶段 7.进入提交结果集流程	1.进入提交流程显示提交截止倒计时 2.十分钟后刷新页面可显示提交已截止	通过

结果公示时间设置测试用例	1.比赛管理员登录系统 2.进入第一阶段阶段管理页面 3.设置结果公示时间为十分钟后 4.点击确定按钮 5.参赛选手登录系统 6.点击进入第一阶段 7.进入结果公示流程	1.结果公示流程页显示暂未开放 2.十分钟后刷新页面可查看结果排名列表	通过
提交周期设置测试用例	1.比赛管理员登录系统 2.进入第二阶段阶段管理页面 3.设置提交周期节点为十分钟后 4.点击确定按钮 5.参赛选手登录系统 6.点击进入第二阶段 7.进入提交容器流程 8.点击新增容器按钮 9.填写容器信息并提交	1.提交容器流程页显示当前批次和提交的容器名称 2.提交容器按钮不可点击 3.十分钟后刷新页面，提交容器流程页批次加一，提交容器按钮可点击，可继续新增容器	通过

表 6-10描述了阶段基准线用例的测试详情，基准线是用户可以进入下一阶段的标准，用例测试以第一阶段为例，包括为达到基准线测试用例和达到基准线测试用例，测试用例的实现需管理端和用户端同时进行操作。基准线生效时间以结果公示时间作为节点，在第一阶段中提交结果集的准确率未达到基准值的用户无法进入第二阶段，在点击第二阶段入口时显示暂无权限，否则可以进入第二阶段。第二阶段和第三阶段的基准值类型为比例，在结果公示时间到达后对有效评估后的容器按照评估分数进行排名，根据比例计算通过人数，从而给对应用户修改阶段权限。

表 6-10: 阶段基准线设置用例测试结果

测试项	操作	实现效果	测试结果
未达到基准线测试用例	1.比赛管理员登录系统 2.进入第一阶段管理页面 3.设置基准值为 0.6 4.点击确定按钮 5.设置结果公示时间为十分钟后 6.点击确定按钮 7.参赛选手登录系统 8.进入第一阶段提交流程页 9.新增空的结果集文件	1.十分钟后刷新页面，在结果公示页面中显示未通过 2.第二阶段开启时无法进入，显示无权限	通过
达到基准线测试用例	1.比赛管理员登录系统 2.进入第一阶段管理页面 3.设置基准值为 0.6 4.点击确定按钮 5.设置结果公示时间为十分钟后 6.点击确定按钮 7.参赛选手登录系统 8.进入第一阶段提交流程页 9.新增准确率为 0.6 的结果集文件	1.十分钟后刷新页面，在结果公示页面中显示通过 2.第二阶段开启时可以进入	通过

表 6-11描述了阶段数据集设置用例的测试详情，阶段数据集设置时测试集可以选择发布或不发布，若设置为发布则用户可以下载测试集，测试时以第一阶段为例，若设置为未发布则用户无法下载测试集，测试时以第二阶段为例，设置不发布测试集的阶段可以指定鲁棒性变异数据集和公平性变异数据集，指定后测试集文件的 dfs 状态改变。

表 6-11: 阶段数据集设置用例测试结果

测试项	操作	实现效果	测试结果
第一阶段数据集设置测试用例	1.比赛管理员登录系统 2.进入第一阶段管理页面 3.设置阶段训练集 4.设置阶段测试集 5.点击确定按钮 6.参赛选手登录系统 7.进入第一阶段下载流程页 8.点击下载训练集 9.点击下载测试集	1.下载训练集成功 2.下载测试集成功	通过
第二阶段数据集设置测试用例	1.比赛管理员登录系统 2.进入第二阶段管理页面 3.设置阶段训练集 4.设置阶段测试集 5.选择测试集不发布 6.设置阶段鲁棒性测试集 7.设置阶段公平性测试集 8.点击确定按钮 9.参赛选手登录系统 10.进入第二阶段下载流程页 11.点击下载训练集	1.下载训练集成功 2.比赛管理可在数据集管理页面中查看测试集对应数据集上传 dfs 状态改变	通过

6.3 非功能性测试

基于司法多维评估的模型评估系统的非功能性测试围绕第三章给出的非功能性需求展开，从时间特性、可扩展性、并发性、安全性和可靠性对系统中的关键接口进行测试。

时间特性规定针对不同功能的响应时长，系统中功能的主要分为两类，若某一功能不涉及异构服务则对该功能时间特性的测试以接口的响应时间为准，例如用户登录功能只需考察登录接口的响应时长；若涉及异构服务，则以该功能的整体时长作为测试时间特性的标准，例如新增容器这一功能需调用异构服务器中的接口并运行脚本，所以对这一功能时间特性的考察不以业务逻辑服务器中新增容器的接口响应时长作为其时间特性，而是以从用户发起请求到容器导入成功的完整流程所耗费的时长作为其时间特性。表 6-12显示不同接口在 150 次访问中的平均响应时长；新增容器通过编写脚本得到其平均功能时长，测试结果如表 6-14所示。

表 6-12: 接口响应时长测试结果

接口	平均响应时长	测试结果
用户登录	756.3ms	通过
新增结果集	1297.2ms	通过
新增数据集	1186.9ms	通过
阶段管理	974.1ms	通过
数据集管理	1636.2ms	通过
容器管理	1433.7ms	通过

表 6-13: 新增容器功能时长测试结果

状态	平均响应时长	测试结果
容器导入	8946ms	通过
健康检查	5734ms	通过

可扩展性和**并发性**的测试是结合在一起的，主要针对新增容器功能进行测试，通过增加部署异构服务的服务器数量测试对新增容器功能的并发性影响，测试结果如表 6-14所示，由表可推测该功能的并发量随异构服务器数量的增加而增加，其上限由业务逻辑服务器的新增容器接口的并发量所确定。

表 6-14: 新增容器功能并发量与异构服务器数量关系

服务器数量 (个)	并发量	导入成功率
1	10	90%
1	20	45%

2	10	100%
2	20	95%

安全性测试系统权限管理功能，用户以参赛选手的身份登录到系统中，通过修改 url 地址访问只有比赛管理员才能访问的页面时显示如图 6-1所示的 404 页面，系统安全性测试通过。

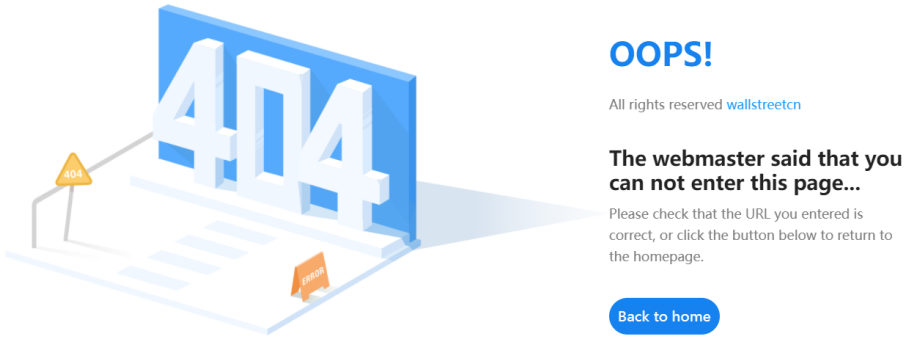

图 6-1: 无权限跳转的 404 页面

可靠性评估的重要指标之一是系统故障的排查能力，业务逻辑服务器上的微服务通过日志记录保障系统可靠性，涉及异构服务的功能对脚本中的异常错误以文件的形式存储在服务器中，同时将错误状态持久化至数据库中，用户可在页面中下载相应的错误信息文件，例如容器导入操作，图 6-2展示在容器管理页面中容器状态为导入失败的错误信息文件下载的具体内容，该信息表示此异构服务器无内存空间，需清空内存或增加其他服务器。

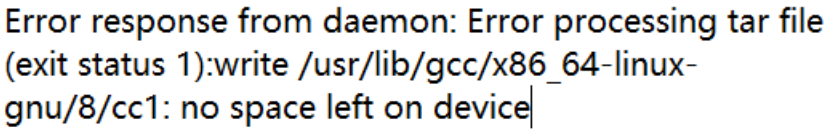

图 6-2: 容器导入错误信息

6.4 实验数据及分析

6.4.1 实验数据

本文选取 fastText 模型 [41]、TextCNN 模型 [42]、多层 LSTM 模型 [43] 进行司法刑期多分类模型的评估。模型训练集和测试集取自 CAIL2018 [33] 提供的真实司法案例数据，分别包含 748203 条和 35927 条案例数据，所涉及案例涵盖多种犯罪类型，具有较高的可用性。针对本文所提出的变异性评估指标，模型需在原始数据集和变异数据集上进行测试，原始数据集是上文所提到的由真实案例构成的测试集，变异数据集是在原始数据集的基础上使用基于 TF-IDF 的特征裁剪、基于依存句法的特征变换、基于词典的随机插入和基于语义的歧视性文本插入四种变异算法变异后的数据集，四种变异算法分别对应四个变异数据集。

在原始数据集上进行测试获取到的准确率是基础性评估指标之一，除此之外还包括可参考性和泛化性，两个指标的评估分数的计算是通过比对原始数据集上的分类结果和原本的案例分类标签。结合本文第三章给出的计算公式可获得三个模型在各指标中评估分数

fastText 模型 fastText 模型训练使用“autotune-validation”参数开启自动超参数优化，该参数可针对训练集自动采用最佳超参数训练模型，从而避免手动调参的繁琐。模型在原始数据集和变异数据集的分类准确率如表 6-15所示，可参考性和泛化性的值分别为 0.947 及 0.62。

表 6-15: fastText 模型在不同数据集上的测试准确率

数据集	准确率	变异参数
原始数据集	71.6%	无
基于 TF-IDF 的特征裁剪变异数据集	68.5%	lengthWeigth:0.4 rangeWeight:0.4 quantityWeight:0.4
基于依存句法的特征变换变异数据集	68.3%	lengthWeigth:0.2 selectWeight:0.4
基于词典的随机插入变异数据集	56.7%	lengthWeigth:0.4

基于语义的歧视性文本插入变异数据集	70.5%	无

TextCNN 模型 该模型的第一部分使用嵌入层对词向量进行训练，第二部分是卷积层，提取句子中相邻词之间的关键信息，其中的卷积滤波器的大小设置为 5，数量为 256，第三部分的池化层取卷积后的得到的几个一维向量的最大值，进行拼接后作为该层的输出，第四部分是进行全连接，提高网络的学习能力，最后使用 softmax 层对分类结果进行输出。该模型在原始数据集和变异数据集的分类准确率如表 6-17所示，可参考性和泛化性的值分别为 0.953 及 0.656。

表 6-16: TextCNN 模型在不同数据集上的测试准确率

数据集	准确率	变异参数
原始数据集	72.2%	无
基于 TF-IDF 的特征裁剪变异数据集	70.5%	lengthWeigth:0.4 rangeWeight:0.4 quantityWeight:0.4
基于依存句法的特征变换变异数据集	65.4%	lengthWeigth:0.2 selectWeight:0.4
基于词典的随机插入变异数据集	66.9%	lengthWeigth:0.4
基于语义的歧视性文本插入变异数据集	72.1%	无

多层 LSTM 模型 该模型的第一部分使用嵌入层对词向量进行训练，第二部分和第三部分通过两层的 LSTM 进行连接，LSTM 向量的长度是 128，并在每一层 LSTM 之后使用了 dropout 以提高模型的泛化能力和防止过拟合，第四部分和第五部分是全连接层，提高网络的学习能力，并且也在两部分之间增加了 dropout，最后是 softMax 层输出分类结果。该模型在原始数据集和变异数据集的分类准确率如表 6-17所示，可参考性和泛化性的值分别为 0.973 及 0.743。

表 6-17: TextCNN 模型在不同数据集上的测试准确率

数据集	准确率	变异参数
原始数据集	72.9%	无
基于 TF-IDF 的特征裁剪变异数据集	71.5%	lengthWeigth:0.4 rangeWeight:0.4 quantityWeight:0.4
基于依存句法的特征变换变异数据集	71.2%	lengthWeigth:0.2 selectWeight:0.4
基于词典的随机插入变异数据集	65.6%	lengthWeigth:0.4
基于语义的歧视性文本插入变异数据集	71.2%	无

6.4.2 实验分析

三种模型的评估指标分值如表 6-18所示，其中准确性、可参考性和泛化性指标直接将原值乘以 100，而鲁棒性和公平性的值则根据第四章提供的变异性指标计算公示根据模型在原始测试集和变异测试集上的准确性比值再加上变异因子获得。

表 6-18: 不同模型在各评估指标的分值

模型名称	准确性	可参考性	泛化性	鲁棒性	公平性
fastText	71.6	94.7	62.0	93.9	98.4
TextCNN	72.2	95.3	65.6	94.75	102
多层 LSTM	72.9	97.3	74.3	95.21	101

图 6-3根据表 6-18的各项指标值给出了各模型指标对比柱状图，从该图中可以看到，各模型在各项指标中的分值差异不大，但多层 LSTM 模型总体表现更好，其在准确性、可参考性、泛化性以及鲁棒性四个指标中的分值最高，而 TextCNN 模型在公平性指标的评估中的表现较好。

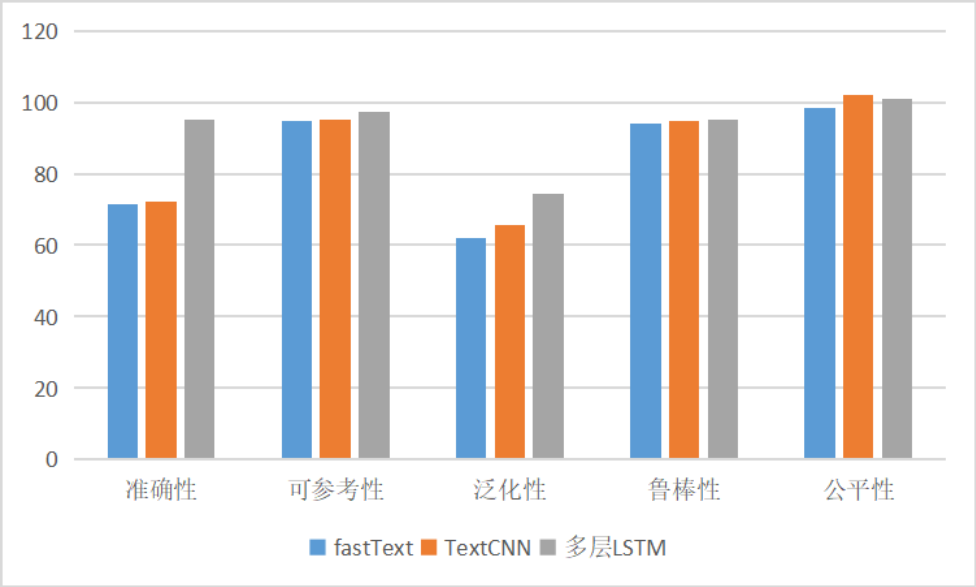

图 6-3: 各模型指标值对比柱状图

6.5 本章小节

本章对基于司法多维评估的模型评估系统进行功能性测试和非功能性测试，第一部分介绍系统部署的测试环境，第二部分和第三部分按照第三章中功能性需求和非功能性需求用例进行测试，测试结果有效地验证了系统的完整性和可用性，最后针对 fastText 多分类模型、TextCNN 模型、多层的 LSTM 模型使用本文提出的司法多维评估指标进行评估，并对评估结果进行对比。

第七章　总结与展望

7.1　总结

基于司法多维评估的模型评估系统针对刑期预测模型进行评估，评估指标分为多分类模型基础评估指标和变异指标，其中基础指标基于相应的计算方式挖掘其在司法领域的评估价值，而变异指标则根据模型在具有不同司法特征的变异数据下分类准确率的高低来评估模型，司法多维评估指标的设计与实现推动司法模型向“可视正义”这一目标迈进。除此之外，在设计之初系统的使用场景就设定为通过竞赛的方式筛选和评估模型，因此系统设计并实现了分阶段批量评估筛选的流程。

本文详细描述了基于司法多维评估的模型评估系统的设计和实现过程。首先，介绍了该系统的课题背景和选题意义，同时对当前司法模型评估工作的现状进行了研究以及对系统所使用的相关技术进行说明。随后，对基于司法多维评估的模型评估系统的功能性需求和非功能性需求进行了阐述，并通过包含两种用户角色的用例图和详细的用例描述对多维评估系统的需求进行展开，在此基础上从“4+1”视图给出了多维评估系统在架构、逻辑、进程、开发和部署等方面的设计，并对系统的持久化设计方案和司法多维评估指标设计进行了阐述。在此之后，本文按照司法多维评估模型功能模块的划分分别介绍了权限管理模块、文件管理模块、阶段管理模块、数据集管理模块和容器管理模块的详细设计和实现，通过统一建模语言和关键方法的实现代码对各个模块和模块间的交互进行具体的说明。最后，本文基于系统的功能性需求和非功能性需求分析设计了具体的测试用例，按照功能性测试用例的描述依次进行系统测试并记录系统相应的实现效果，而非功能性需求的测试从时间特性、可扩展性、并发性、安全性和可靠性几个方面给出相应的实验数据和测试效果，有效地验证了系统的可用性。

本文针对刑期预测模型建立了一套具有司法领域特性的分阶段评估筛选系统，从司法多维指标的设计到批量评估等多个方面为大规模的司法领域评估系统的设计和实现提供参考，以此推动人工智能模型在司法领域更加广泛和深入

的应用。

7.2 展望

本文初步实现了基于司法多维评估的模型评估系统，具备司法多维指标评估和分阶段批量筛选功能，但在实际使用中系统还可以从以下几个方面进行改进：

首先，优化多维评估体系，本系统引入了五种指标的计算方式，其中包括三种基础性指标和两种变异性指标，为司法领域中的模型评估建立了可行的评估体系，在之后的工作中可挖掘更多的司法领域数据变异方法于司法评估指标进行关联，从而建立更完备的司法多维指标评估体系。

其次，自定义和自解析数据集标签，目前系统接收的原始数据集格式是固定的，在之后的工作中应支持数据集格式自定义，数据集初始化时可根据用户定义的规则进行解析，从而扩展系统的应用场景和适用范围。

最后，系统缺乏自检和自恢复能力，系统利用 fastDFS 系统的同步能力对容器压缩文件和测试集文件进行冗余，为系统的自恢复能力奠定基础，在未来的工作中系统应完善自检能力，监听系统中的错误信息，并自动进行恢复工作，提高系统稳定性和降低维护成本。

致　　谢

值此论文完成之际，我首先要感谢我的导师陈振宇教授对我的指导与帮助，感谢刘佳玮学姐帮助我完成论文，感谢我的舍友们这两年对我学习上和生活上的帮助，感谢郭超学长对我的照顾，感谢我的大学舍友和挚友杜月莹同学对我的鼓励和精神上的陪伴，最重要的是我要感谢我的父母和我的两个姐姐，你们给予了我无条件的爱和支持，这些爱和支持支撑着我走过了十八年的学习生涯，你们永远是我最坚强的后盾。

时光荏苒，在南大的两年时间一转眼就过去了，这座我从高一开始便执着的“象牙塔”终究在我心中留下里深深的印记，这些印记或许来源于南大带给我的光环，来源于南京这座城市无与伦比的四季与梧桐树，来源于在万象书坊读着《人性的枷锁》时超越永恒的那一瞬间，又或许来源于在诸多个深夜中的自我怀疑与自我重建，来源于因吃甜食带来的日渐肥胖的身体，来源于无聊、恐惧、未知、焦虑……曾经看过一段话大概意思是说当你觉得痛苦时你就在经历成长，我想我应该是成长了很多，我成长到可以直面自己的缺点和自己的渺小，成长到坦然接受生活向我扔过来的诸多挑战，成长到学会珍惜自己身边至亲至爱的人。四年的北京生活和两年的南京生活让我这个从湘西大山里走出来的姑娘明白了重要的不是我身处何地或是我将去往何地，重要的是我想要成为什么样的人、做什么样的事，在这里过程中我要做的就是挑战、经历、体会、打破、再重构。

参考文献

[1] MOOR J. The Dartmouth College artificial intelligence conference: The next fifty years[J]. Ai Magazine, 2006, 27(4): 87–87.

[2] PANNU A. Artificial intelligence and its application in different areas[J]. Artificial Intelligence, 2015, 4(10): 79–84.

[3] 梅宇轩. 人工智能在法律推理中的应用 [J]. 法制博览 (名家讲坛、经典杂文), 2021(32): 3.

[4] YAO J-J, HUI P. Research on the Application of Artificial Intelligence in Judicial Trial: Experience from China[C] // Journal of Physics: Conference Series: Vol 1487. 2020: 012013.

[5] 陈阳, 荣威. 人工智能在我国司法审判中的应用及限制 [J]. 绥化学院学报, 2021, 41(11): 4.

[6] 光明网. 法治蓝皮书：中国智慧法院体系基本建成 [EB/OL]. 2020. https://m.gmw.cn/baijia/2020-06/08/33896308.html.

[7] 查文龙. 智慧法院建设新进展 [J]. 法制博览, 2021.

[8] 姜峰. 法院" 案多人少" 与国家治道变革——转型时期中国的政治与司法忧思 [J]. 政法论坛, 2015.

[9] 蒋兆康. 法律的经济分析 [J]. 中国大百科全书出版社, 1997.

[10] 赵杨. 人工智能时代的司法信任及其构建 [J]. 华东政法大学学报, 2021, 24(4): 10.

[11] 陈灵峰. 司法人工智能的技术效应和应用边界 [J]. 求索, 2021: 182–190.

[12] 马长山. 司法人工智能的重塑效应及其限度 [J]. 法学研究, 2020, 42(4): 18.

[13] BECERRA S D. The rise of artificial intelligence in the legal field: where we are and where we are going[J]. J. Bus. Entrepreneurship & L., 2018, 11 : 27.

[14] 徐晋. 基于神经网络专家系统的创业企业信用等级评估研究 [J]. 南京理工大学学报（自然科学版）, 2004, 028(006) : 684 – 688.

[15] 王嘉凯, 刘艾杉, 刘祥龙. 人工智能机器学习模型及系统的质量要素和测试方法 [J]. 信息技术与标准化, 2020(1) : 6.

[16] 王栋. 面向司法案例筛选的测试系统的设计与实现 [D]. [S.l.] : 南京大学, 2020.

[17] ZHAO G, WANG J, SHI H, et al. Research on Multiattribute Comprehensive Evaluation of Intelligent Judicial Decision System[J]. DISCRETE DYNAMICS IN NATURE AND SOCIETY, 2021, 2021.

[18] ZHANG S, YAN G, LI Y, et al. Evaluation of Judicial Imprisonment Term Prediction Model Based on Text Mutation[C] // 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C). 2019 : 62 – 65.

[19] 张舒. 司法文本数据自动化生成系统的设计与实现 [D]. [S.l.] : 南京大学, 2021.

[20] 严格. 面向司法数据质量的模型评估系统的设计与实现 [D]. [S.l.] : 南京大学, 2021.

[21] SIRBI K, KULKARNI P J. Stronger enforcement of security using aop and spring aop[J]. arXiv preprint arXiv:1006.4550, 2010.

[22] WALLS C. Spring Boot in action[M]. [S.l.] : Simon and Schuster, 2015.

[23] ANDERSON C. Docker [Software engineering][J]. IEEE Software, 2015, 32(3) : 102 – c3.

[24] LIU X, YU Q, LIAO J. FastDFS: a high performance distributed file system[J]. ICIC express letters. Part B, Applications: an international journal of research and surveys, 2014, 5(6) : 1741 – 1746.

[25] SHVACHKO K, KUANG H, RADIA S, et al. The hadoop distributed file system[C] // 2010 IEEE 26th symposium on mass storage systems and technologies (MSST). 2010 : 1 – 10.

[26] STRUBELL E, GANESH A, MCCALLUM A. Energy and policy considerations for deep learning in NLP[J]. arXiv preprint arXiv:1906.02243, 2019.

[27] 敖盛, 徐岚, 敖清文. NLP 中文分词技术在桥梁报告数据处理中的应用 [J]. 交通世界, 2020(17) : 3 – 5.

[28] 张丹. 中文分词算法综述 [J]. 黑龙江科技信息, 2012(08) : 206.

[29] CHE W, LI Z, LIU T. Ltp: A chinese language technology platform[C] // Coling 2010: Demonstrations. 2010 : 13 – 16.

[30] JARVINEN T, TAPANAINEN P. Towards an implementable dependency grammar[J]. arXiv preprint cmp-lg/9809001, 1998.

[31] J R J. Dependency structures and transformational rules[J]. Language, 1970 : 259 – 285.

[32] J R, OTHERS. Using tf-idf to determine word relevance in document queries[C] // Proceedings of the first instructional conference on machine learning : Vol 242. 2003 : 29 – 48.

[33] XIAO C, ZHONG H, GUO Z, et al. Cail2018: A large-scale legal dataset for judgment prediction[J]. arXiv preprint arXiv:1807.02478, 2018.

[34] RICHARDSON L, RUBY S. RESTful web services[M]. [S.l.] : ” O’Reilly Media, Inc.”, 2008 : 14 – 17.

[35] KRUCHTEN P B. The 4+ 1 view model of architecture[J]. IEEE software, 1995, 12(6) : 42 – 50.

[36] 胡康生, 郎胜. 中华人民共和国刑法释义 [M]. [S.l.] : 法律出版社, 2006.

[37] COOK R J. Kappa[J]. Encyclopedia of biostatistics, 2005, 4.

[38] 李卓阳. 基于领域特征的文本数据扩增技术 [D]. [S.l.] : 南京大学, 2021.

[39] SHI S, WANG Q, CHU X. Performance modeling and evaluation of distributed deep learning frameworks on gpus[C] // 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). 2018 : 949 – 957.

[40] COMBE T, MARTIN A, DI PIETRO R. To Docker or Not to Docker: A Security Perspective[J]. IEEE Cloud Computing, 2016, 3(5) : 54 – 62.

[41] ATHIWARATKUN B, WILSON A G, ANANDKUMAR A. Probabilistic fasttext for multi-sense word embeddings[J]. arXiv preprint arXiv:1806.02901, 2018.

[42] GUO B, ZHANG C, LIU J, et al. Improving text classification with weighted word embeddings via a multi-channel TextCNN model[J]. Neurocomputing, 2019, 363 : 366 – 374.

[43] SALMAN A G, HERYADI Y, ABDURAHMAN E, et al. Single layer & multi-layer long short-term memory (LSTM) model with intermediate variables for weather forecasting[J]. Procedia Computer Science, 2018, 135 : 89 – 98.

简历与科研成果

基本信息

贺璐，女，苗族，1999 年 2 月出生，湖南怀化人。

教育背景

2020 年 9 月 — 2022 年 6 月 南京大学软件工程专业 硕士

2016 年 8 月 — 2020 年 6 月 北京理工大学软件工程专业 本科