南京大学
NANJING UNIVERSITY

研 究 生 毕 业 论 文

（申请硕士专业学位）

论 文 题 目　面向司法案例筛选的测试系统的设计与实现

作 者 姓 名　王栋

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　刘嘉　副教授

2020 年 05 月 12 日

学　　号：　MF1832154

论文答辩日期：　2020 年 05 月 22 日

指 导 教 师：　  　**（签字）**

The Design and Implementation of the System for Testing Judicial Case Screening System

By

Dong Wang

Supervised by

Associate Professor **Jia Liu**

A Thesis
Submitted to the Software Institute
and the Graduate School
of Nanjing University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

Software Institute

May 2020

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：面向司法案例筛选的测试系统的设计与实现

工程硕士（软件工程领域）专业 2018 级硕士生姓名：王栋

指导教师（姓名、职称）：刘嘉 副教授

摘 要

基于裁判文书的司法案例筛选系统，以案例的描述文本为依据，根据特定的算法分析，从指定的案情数据集合中筛选出相似的司法案例，包括相关的罪行、法律法规、刑期和罚金等关键信息。该系统为司法从业者及需要法律咨询的民众提供了高效便捷的服务。类案推荐的质量水平决定了用户的满意度以及接受度，这影响到该系统的普及程度。本文设计并实现对司法案例筛选系统测试的系统，旨在对司法案例筛选系统的效果建立多维评估体系，并为案例筛选研究人员和测试人员提供参考价值。

本论文的主要工作是建立了一套多维的案例筛选测试标准，搭建了一个可用性强的测试案例筛选系统的测试平台。本文首先对司法案例筛选系统以及司法测试进行了研究现状的介绍。然后，介绍了本系统采用的技术框架，该系统采用 Spring Boot 框架，通过 Spring Cloud 微服务的架构将每个功能模块进行切分，进而详细介绍了项目的用例设计和系统的设计以及实现。

本系统从模型和系统两个方面进行案例筛选的测试：针对案例筛选系统相关的机器学习模型，基于指定数据集进行基础和扩展指标的评估；针对案例筛选系统的系统级接口，对筛选的案例列表进行多维度评估度量。系统的微服务总体设计分为权限管理、上传管理、测试管理和数据处理四个主要的模块。通过权限模块来进行用户认证、权限鉴定以及网关设置，确保服务的安全性；对模型和数据集资源进行上传管理，建立用户专属的资源库；通过测试管理模块，支持对司法模型测试和案例筛选接口测试的新增、查询、删除等操作；通过数据处理模块进行文件解析、校验存储和评估指标的度量。多模块相互协同，最终生成测试报告，并提供评估结果导出等功能。

该系统最终以 web 应用的形式展现给用户，用户可以通过浏览器访问该系统，进行数据集、模型的管理，进行案情描述的文本输入，通过模型或接口运行自动化测评结果，并根据多维评估体系生成对应的测试报告，对司法领域开发者和测试者而言，具有较高的实用性。

关键词：案例筛选，司法测试，自然语言处理，相似性度量

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of the System for Testing Judicial Case Screening System

SPECIALIZATION: Software Engineering

POSTGRADUATE: Dong Wang

MENTOR: Associate Professor Jia Liu

Abstract

Judicial case screening system selects similar judicial cases from the case data set based on the description text of the judicial case, analyzed by the specific algorithm, including related articles, imprisonment, accusation and fines. The system provides efficient and convenient services for judicial practitioners and citizens who need legal advice. The quality of recommendation case list influences user satisfaction and acceptance, which affects the popularity of the system. This thesis designs and implements a system for testing the judicial case screening system. It aims to establish a multi-dimensional evaluation system for the effectiveness of the judicial case screening system and to provide a set of testing criteria for developers and testers.

The main work of this thesis is to set up a series of multi-dimensional case screening test criteria and to build a highly functional testing platform for case screening system. Firstly, this thesis introduces the current state of the judicial case screening system and judicial testing. And then, the developing frameworks and technologies adopted by this system are introduced, such as the Spring Boot framework and some natural language processing technologies. Each functional module is an independent part of the Spring Cloud microservice architecture. Besides, the design and implementation of the project are put out in detail in this thesis.

This system conducts the test of case screening system in two aspects: the model level and the system level. The relevant machine learning models for the case screening system are evaluated by basic and extensive metrics based on the specific data set. Multi-dimensional similarity metrics test the system-level interfaces of the case screening system. The system is divided into four main modules: authorization management, upload management, testing management, data analysis and evaluation modules.

The authorization management module is used to make authorization identification and gateway setting to ensure the security of the service. The upload management module manages the model and dataset resources to establish a user-specific resource library. The testing management module is used to create or query the testing task of this system, and the data analysis module makes sure that all kinds of data facilitated automatic testing by the evaluation module. These modules cooperate to finally generate a test report and also provide the export function.

The testing system eventually works well in the form of a web application. The user can access the system through a browser to manage data set and model files, conduct testing tasks automatically to generate corresponding testing results. This testing system could be highly practical for developers and testers in the legal field.

Keywords: Case Screening, Judicial Testing, Natural Language Processing, Similarity Metrics

目录

表 目 录

图 目 录

第一章　引言

1.1　选题的背景和意义

随着人工智能的迅速发展，司法行业作为社会生活的重要组成部分，正在逐渐拥抱人工智能等新兴技术所带来的综合优势。在人工智能等新技术的协助下，司法工作者得以完成艰巨且繁杂的日常工作，从而实现了提高司法工作效率和质量，推动司法行业发展进步 [1]。基于传统的司法业务场景，人工智能在司法行业产生了诸多应用，包括案例筛选和辅助审判等核心业务场景；为了提高司法工作质量，人工智能也为司法行业拓展了新的业务场景，包括再犯预测和裁判文书质量评估等场景。人工智能在司法行业广泛而深入的应用也带来了新的问题和挑战，如公平性问题 [2][3]。新的挑战和问题急需新的理论和技术方法为人工智能的高效应用提供质量保障。

在目前火热的人工智能和大数据技术的影响下，法律大数据和人工智能技术正在帮助司法界进行技术革新 [4]。在一线司法工作者之中，案例筛选是一项贴近他们工作实务的重要环节，利用机器学习技术，可以一定程度上替代人工，自动地完成从案例库中筛选出关键信息的工作 [5]。司法案例筛选系统，即以案例描述的文本为依据，筛选出与之相关的司法案例信息，包括案例相关的罪行、刑期、法律法规等，已成为智能辅助办案系统的重要组成部分。

目前诸多机构以及各层级法院都开展了相关的研究，开发研究出适应自身业务的案例筛选系统。在地方层面，各省法院先后分别开发出各自的案例筛选系统，并且延伸出类案推荐等业务。但在实际应用中，这样的应用并没有受到司法工作者的普遍欢迎，主要存在的问题包含以下三点：

筛选的结果不够精确，不足以解决法律工作者的实际需要。这一点在类案推荐应用中尤为明显，由于案例筛选的精度不够，虽然从案例中提取处理一定的关键信息作为标签，但是仅仅依据这样的标签进行推荐，往往会推荐出多达上万个相同标签的司法案例。即使在案例筛选的过程中，通过筛选多个标签共同推荐能够将最终推荐的数量削减至百个甚至数十个。即使如此，对于司法工作者而言，精读如此数量的案例仍然非常大的工作量，在这种情况下，案例筛选和类案推荐都未能起到预期的效果。在实际的应用情况中，往往正在困扰司法工作者的是某个细节或者是某个法律难点。此时，他们仅仅是想参考其他从业人员在判案和辩诉时的思路和做法。在这样的情况下，需要提升案例筛选的精确度，使得筛选出的结果能够有效的刻画和表现案例的法律细节，否则过于粗

糙的处理结果无法解决实际中的业务问题。

案例筛选的司法模型文件和测试数据文件较大，当复用资源时通常会进行复制操作占用存储空间，缺乏有效的资源配置管理，导致测试效率低下。同时，测试人员通常只关注测试结果，而忽视司法模型和测试数据集与之的关联，导致测试的可追溯性较差，影响案例筛选模型的测试质量。此外，缺少统一的适配手段来解决司法模型类型和测试数据格式多样性，也对案例筛选的测试造成一定的困难。

各地实践差异较大，水平参差不齐。各地对案例筛选的建设主要都是依据自身的业务需求或创新设想。各层级法院、律师事务所采用的案例筛选模型都是由不同的法律科技公司或机构进行研发设计的，总体而言缺少整体的顶层规划。在这一情况下，各地所实现的案例筛选系统本质上区别很大，也没有明确的迭代和改进的方向，缺乏统一的测试评估方案。

在实际的司法环境下，案例与法律法规，案例与先前判例的关系，以及案例中罚金，量刑的尺度都是关注的焦点，因而筛选文本的上述要素，可以帮助相关司法工作者通过要素快速地定位相关的案例。对于案例筛选，有诸多可以参考的评估模型，然而不同的模型所擅长处理的文本类型不同，需要构建案例筛选评估系统对其进行度量。因此，案例筛选的评估，既要考虑评估模型本身的特性，也需要站在使用者的角度考虑实际语义的特征。本文主要从司法模型和案例筛选接口两个层面来对司法案例筛选系统进行测试，基于现有的评估指标，结合司法领域特征，建立多维的司法案例评估体系，为系统的测试人员和法律从业者提供对应的测试结果报告，来评估司法案例筛选的性能，也为各类的案例系统提供统一的标准，为开发人员提供统一的系统优化方向。

1.2　国内外研究现状及分析

人工智能技术作为新时代下的创新生产力，现已上升为国家级别的技术战略，作为我国攻克技术难题、实现“弯道超车”新的支撑点，各行各业也纷纷响应、支持 [6]。法院自其诞生以来，便一直作为社会关系网格中的重要一环，守护着现代社会的最后一道正义防线，与人民群众的生活息息相关。人工智能的技术发展与司法体制的制度革新相结合，将会给司法工作注入前所未有的创造力 [7]。人工智能介入司法领域是时代的必然选择：一方面，司法机关是国家机关和社会生活领域的重要组成部分，国家战略和社会发展将会不可避免的影响到司法领域的工作；另一方面，司法领域对人工智能具有主动的诉求，即便司法改革的推进已经深入到各基层人民法院，然而当前案多人少的严峻局面并没有得到根本性转变 [8]。人工智能如何介入司法，以及人工智能介入司法行业的程

度或者是如何与司法工作深度融合才是当下急需耕耘的主题，探索的结果将直接关系着人工智能在司法领域未来发展的定位和方向。

将先进的计算机技术与司法相结合并不是最新出现的课题，最早是以信息检索技术为媒介搭建二者结合的桥梁。早在上世纪中后期，美国司法部就已经开发出了包含联邦成文法规、最高法院判例汇编和司法部内部文件等的联网数据库 JURIS 系统 (Juristic Retrieval and Inquiry System)[9]。在我国，以“北大法宝”为代表的法律法规检索系统经过多年来的发展，在信息完整性和搜索智能性等方面的功能已获得显著成就 [10]。2013 年，中国裁判文书网的上线是法律信息检索系统领域的一块新里程碑，利用计算机检索技术，可以从最大程度上促进司法公开，截至 2018 年初，该网站已成为全球最大的裁判文书资源库 [11]。自八十年代中后期开始，我国对于司法系统的研究从法律检索系统进阶为具有决策能力的法律专家系统，这一时期研发的量刑辅助系统属于法律专家系统的范畴，代表性成果包括 1993 年武汉大学法学院赵廷光教授主持开发的实用刑法专家系统 [12]。进入 21 世纪后，随着互联网和人工智能技术的发展以及司法案件数量的激增，法律专家系统的研究得到了迅速发展 [13]。人工智能对司法领域的应用是包括但不限于量刑参考、类案推送等在内的人工智能系统多项功能综合产生的结果，衍生出了包括智慧审判系统、司法案例筛选系统在内的相关智能辅助系统，智慧法院的概念也开始逐步被大众所熟知。

“智慧司法”结合了人工智能技术，其探索的解决方案通过重新审视法学问题，以期获得新的可能路径。首先，人工智能从功能上对法律证成、法律检索、法律解释、法律适用等法律推理的要素和活动进行数字化并加以逻辑分析，将法律推理的研究成果模型化，以实现法律推理知识的机器表达或再现，从而为认识法律推理的过程和规律提供一种实验手段 [14]。在实际的司法过程中，机械地在司法案件中适用法律必然是一个降低司法效率的问题；此外很多人为的记录失误等因素也不可避免的会干扰到司法审判。因此，这些无法依靠人类自身去克服的问题有望利用人工智能技术找到突破口。人工智能技术能够基于以往的案件事实，生成法律语料库，并实现海量存储，与此同时建立分类和分析模型，利用语料库统计方法和统计与规则并举的方法进行法律行业的自然语言“理解”研究 [15]。

作为一种辅助审判的工具，智慧法院系统可以模拟法官，参考案例事实做出决策，在处理类型化案件时，它将发挥更好的司法办案辅助作用。我国目前正在研发的智能辅助办案系统包括法条推荐系统 [16]、刑罚预测系统 [17] 以及案例筛选系统 [18] 等，如最新的人民法院“类案智能推送系统”全面采用新一代深度学习、自然语言处理、知识图谱等人工智能技术，为广大法官提供更便捷、

更精准的案例检索与推送服务 [19]。

随着人工智能技术的不断发展和完善，目前已经投入使用的各类智能辅助办案系统所能发挥的作用将越来越大，然而人工智能可能带来的风险与瓶颈也逐渐成为舆论的焦点，学术界更是对人工智能可能存在地域或性别歧视、算法黑箱、管理失控等问题展开了广泛的讨论 [20]，这些都足以表明利用人工智能辅助法院审判仍然具有争议性。引发这种争议的因素主要包括两方面：一方面，投入使用的人工智能审判系统确实表现出了负面作用。美国威斯康辛州法院采用的 COMPAS 量刑系统是一种基于大数据和人工智能来评估罪犯风险程度的工具 [21]，但是有研究机构发现系统对黑人存在严重的偏见，倾向于高估黑人的再犯可能性；另一方面是由人工智能系统的不透明性造成的社会对其的不信任感。大多数情况下，即使是使用系统的法官本人也无从得知他所使用系统的质量好坏，并且，业界依然对如何刻画一个审判系统的质量缺乏直观、量化的认知。

快速变化的环境与日新月异的技术手段使得公众对法院活动的质量、效率以及经济要求更为严格。人民对法院的期望越来越高，从而需要持续评估并规范法院及其所使用的工具的行为 [22]。一旦这些行为出现了争议，法律将不再能够提供可靠的手段来保障每个公民自由地合法地享用属于自己的权利，司法的发展也将偏离法治的核心价值 [23]。因此，对人工智能审判系统进行质量评估并分析出全面、细节性的评估报告才能打破人工智能司法的瓶颈，进一步释放人工智能带给司法的红利。

目前司法案例筛选系统中的模型主要包括基于多分类技术和线性回归技术的案例筛选模型，以及基于文本相似度技术的案例筛选模型。对于多分类系统的测试以基于混淆矩阵的相关指标为主 [24]，评估所使用的模型分类是否正确。线性回归模型更关注模型的预测分布与实际分布的误差 [25]。但是这些指标鲜有具体的实际意义，无法给出相应的司法解释。此外，案例筛选系统中常使用的推荐算法的测试更多的依赖于用户的交互式反馈结果 [26]，这种测试方式并不适用于司法领域，往往需要利用文本处理技术对筛选结果进行测试 [27]。智慧司法中使用的模型一旦发生了错误，将有极大的概率会导致现实生活中无法弥补的损失，这也是本文系统设计的出发点。

1.3　课题来源及主要研究内容

本文课题来自于国家重点研发计划，是该项目中一个重要的研究内容。随着人工智能和司法大数据的快速发展，将人工智能运用到司法领域已是大势所趋，这也是智慧法院建设的必然要求。人工智能可以促进司法的智能化和自动

化，提高司法过程的效率。用深度学习的方法分析司法文本，用机器学习模型筛选案件，能够为司法判决提供客观有效的辅助。

司法案例筛选主要基于案例的文本描述，通过深度神经网络、依赖性解析、命名实体识别和语言模型在内的一系列自然语言处理和机器学习技术，为司法过程进行辅助，实现相似案例的筛选推荐。智慧司法和法律大数据的发展，也为该业务提供了智能模型和高质量的数据基础，保障了司法案例筛选的研发工作顺利开展。

在实践中，相似案例筛选的评估工作却不尽人意。由于深度学习方法的使用，往往导致整个系统依赖于训练数据，缺乏一定的可解释性，通过传统的测试指标无法充分的评估模型的可靠性。此外，系统通常只考察相似的文本水平，而忽略了案例中关键事实的相似性，也导致了推荐结果在评估过程中有失偏颇。

本文主要研究内容是对司法案例筛选系统建立全面统一的多维评估测试体系，来帮助测试人员和司法从业者评估案例筛选模型的可靠性和合理性，并对案例筛选系统的改进和优化具有一定的指导意义。具体而言，本文的测试方案将基于传统的测试体系的基础上，结合司法领域特征，对模型司法业务可解释性的评估测试，对案例筛选系统进行更为合理可靠的评估。本项目对人工智能与司法领域的交叉研究具有重要的理论意义，对我国人工智能技术更好更广泛更可靠应用具有重要的实际意义。

1.4　本文的工作

本文的研究在最高院提出的“智慧法院”建设目标的背景下展开，旨在推动人工智能更好地在司法实践中服务，提高智慧司法为目标人员的服务水平。目前，应用在司法行业的机器学习模型主要是由大量的司法裁判文书结合机器学习算法构成，由于机器学习模型已被广泛应用于司法实践，因此需要有效的评估方法对司法模型进行评估。此外，为避免单纯基于文本对比的案例筛选方案的片面性，也需要对案例描述的司法特征属性进行分析来综合评估案例筛选结果的性能。

基于上述分析，本文的工作重点在于建立统一全面的案例筛选测试评估体系。于本文而言，分别从司法智能模型和案例筛选结果两个角度建立评估方案：通过对司法模型文件、数据集文件进行管理维护，实现对司法智能模型的测试执行工作；通过系统级的接口调用，基于输入的原始案例描述和筛选案例结果列表数据，实现对案例筛选接口的测试评估工作。基于评估工作进而设计系统功能，展开论述整个系统的具体实现过程，最终实现可运行的原型系统，并对其进行测试验证来确保系统的可用性和稳定性，为目标用户提供优质服务。

1.5　本文的组织结构

文本推荐是自然语言处理的重要应用，司法案例筛选即文本推荐结合领域特征在司法行业的落地应用，然而，在实践中缺乏针对案例筛选的统一评估手段来引导该系统的优化和更新。本文将为司法案例筛选建立全面的多维评估指标体系，从司法智能模型和案例筛选接口两个层面进行评估，并以此衍生为可用的测试系统。本文组织结构如下:

第一章引言，阐述了司法案例筛选测试系统的项目背景及意义、国内外相关的研究现状、本文所属的课题来源、本文的主要研究内容及主要工作。

第二章技术综述，将介绍本文所涉及的技术和开源框架，包括微服务框架、文本处理技术、文本分类模型和司法案例筛选指标等。

第三章系统需求分析与概要设计，将提出项目基本需求，并对项目总体设计思路进行了概述，对项目模块进行了划分、从多个层面对系统进行设计。

第四章系统详细设计及实现，将在需求分析的基础上，重点阐述了系统各个模块的具体实现方法和过程。

第五章系统测试与分析，基于系统需求分析和模块划分对系统功能以及非功能需求进行测试验证，确保系统可用性和稳定性。

第六章总结与展望，对系统项目进行总结描述，并根据系统研究和开发中存在的不足和改进方向。

第二章　技术综述

2.1　Spring 框架

Spring 是一个开源、轻量级、为替代 EJB 应运而生的 Java 企业应用开发框架，提供了构建 Web 应用程序的全功能 MVC 模块 [28]。Spring 最重要的两个核心功能是依赖注入（DI, Dependency Injection）和面向切面编程（AOP, Aspect Oriented Programming）：依赖注入用于管理 Java Bean 对象之间的依赖关系，通过配置在运行时进行注入而非在编译时建立这种依赖，实现了组件间的解耦，提到了编码的灵活性和可维护性；AOP 用于解耦业务代码和公共服务代码（如日志，安全，事务等），实现高内聚开发模式，是对面向对象编程的补充，具有良好的隔离性和源码无关性。Spring 框架最大优势在于其具有模块化的分层架构，开发者可以根据需要使用其中的各个模块，避免引入不必要的部分。Spring 框架对企业应用开发的各类通用问题进行了良好抽象，其非侵入式设计使程序代码对框架依赖最小化，是极其优秀的一站式 Full-Stack 集成框架。

2.1.1　Spring Boot 框架

Spring Boot 是由 Pivotal 团队提供的基于 Spring 的全新框架，其设计目的是用来简化 Spring 项目的编码、配置、部署、监控等工作，帮助开发者快速搭建一个 web 容器 [29]。该框架默认提供 Spring 框架的自动配置信息，可进行通用 Spring Bean 对象的自动装配，额外提供生产就绪型功能，如指标，健康检查和外部配置。此外，Spring Boot 框架支持运行内嵌容器，无需外部依赖 Servlet 容器，可通过 Jar 包启；提供强大的开发包，支持热启动，提高开发和调试效率；支持开发人员快速的开发出 RESTful 风格的微服务架构。Spring Boot 解决的最关键问题，即精简 Spring 配置，并让 Spring 生态圈与其它工具链更方便地整合在一起（比如 redis, email, elasticsearch）。

2.1.2　Spring Cloud 微服务

Spring Cloud 是一系列框架的有序集合，建立在 Netflix 提出的微服务组件的基础上，利用 Spring Boot 的简便性实现的微服务架构的开发工具 [30]。Spring Cloud 将微服务架构设计中所涉及到的服务治理、控制总线、负载均衡、配置管理等功能，通过 Spring Boot 的开发风格做到一键启动和部署。相对于 Dubbo 框架而言，Spring Cloud 提供了一整套企业级分布式应用的完美解决方案，支持快

速高效的开发。Spring Cloud 的服务中心 Eureka 在设计上满足 CAP 理论的可用性和分区容错性，每个服务节点是公平的，确保了微服务的高可用特性，该技术选型符合本项目的高可用需求。微服务架构将业务切分为多模块，降低了单个模块开发难度，增强了应用的持续开发能力，使得每个服务独立扩展。

2.2 React 技术

React 是一个由 Facebook 为解决前端 MVC 架构的扩展需求而开发，用于构建用户界面的 Javascript 开源库 [31]。React 采用声明式设计，纯函数的应用具备极强稳定性，使得前端代码更加可靠，方便调试。通过封装构造组件，React 可复用组件化的设计也提高了开发者的开发效率。相较于传统的前端框架，操作 DOM 一般是直接更新的，效率较低, 而 React 通过抽象提供一个轻量级的 Virutal DOM 来管理真实 DOM 的更新，提高了前端渲染效率。结合 Redux，将 Flux 与函数式编程结合在一起，为 React 提供强大、统一管理的状态管理工具，将数据操作与整体视图逻辑分离，使得前端整体结构更为清晰和容易理解。

2.3 Scikit-learn 库

Scikit-learn（简称 sklearn）是开源的 Python 机器学习库，它基于 Numpy 和 Scipy，提供了大量用于数据挖掘和分析的工具，包括数据预处理、交叉验证、算法与可视化算法等一系列接口。

sklearn 的特征抽取模块可以从原始数据中抽取特征，目前该模块提供了图像和文本特征抽取类 [32]。文本的特征抽取类可以从原始文本中抽取出词语特征，特征数据格式满足所有机器学习算法对输入数据格式的要求。在此需明确特征抽取与特征选择的区别：特征抽取是将文本数据转换成适合机器学习的数值特征而特征选择是一种应用于数值特征的机器学习技术。

2.4 文本处理技术

自然语言处理（Natural Language Process, NLP）研究计算机来处理、理解以及运用人类语言的过程，目标是为解决人机对话中存在的各类困难，使得计算机能够通过人类语言与用户进行交互 [33]。本文主要的研究对象为司法裁判文书中的案例描述内容，即对中文进行文本处理，与英文的文本处理在分词处理、词性标注以及句法结构等方面的细节处理上存在着一定的差异。本部分就针对项目中所涉及到的自然语言处理技术进行具体的展开介绍。

2.4.1　文本分词技术

分词就是将句子、段落、文章这种长文本，分解为以字词为单位的数据结构，方便后续的处理分析工作 [34]。机器学习之所以看上去可以解决很多复杂的问题，是因为它把这些问题都转化为了数学问题，而 NLP 也是相同的思路，文本都是一些“非结构化数据”，我们需要先将这些数据转化为“结构化数据”，结构化数据就可以转化为数学问题了，而分词就是转化的第一步。词是表达完整含义的最小单位。字的粒度太小，无法表达完整含义，比如“鼠”可以是“老鼠”，也可以是“鼠标”，句子的粒度太大，承载的信息量多，很难复用。

英文有天然的空格作为分隔符，但是中文没有。所以如何切分是一个难点，再加上中文里一词多意的情况非常多，导致很容易出现歧义。中文分词需要考虑粒度问题，粒度越大，表达的意思就越准确，但是也会导致召回比较少 [35]。所以中文需要不同的场景和要求选择不同的粒度。这个在英文中是没有的。此外，英文单词存在丰富的变形变换。为了应对这些复杂的变换，英文 NLP 相比中文存在一些独特的处理步骤，即词形还原（Lemmatization）和词干提取（Stemming），而中文则不需要 [36]。

目前对于中文分词没有统一的标准，也没有公认的规范，不同的公司和组织各有各的方法和规则。Jieba 分词算法使用了基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能生成词情况所构成的有向无环图 (DAG)，再采用了动态规划查找最大概率路径，找出基于词频的最大切分组合，对于未登录词，采用了基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法 [37]。

Jieba 分词器还有一个方便的地方是开发者可以指定自定义词典，以便包含词库中没有的词。虽然 Jieba 分词有新词识别能力，但是自行添加新词可以保证分词具备更高的正确率。

2.4.2　文本向量化技术

文本向量化表示就是用数值向量来表示文本的语义 [38]。在自然语言处理中，我们必须将文本转换成机器能够理解的东西，而机器只能理解数值型数据，也就是说，我们需要将字符类型的文本转换成有意义的数字向量 (或数组)。本系统使用 N-gram 和 TF-IDF 技术对案例描述文本进行向量化处理，进而支持相关指标的计算工作。

N-gram 模型将连续出现的 n 个词 ($n \leq N$) 也作为一个单独的特征放入到向量表中去，这样的话就可以解决单词顺序不同所带来的语义不同的问题 [39]。N-Gram 模型基于一个假设：第 n 个词出现与前 n-1 个词相关，而与其他任何词不相关（这也是隐马尔可夫当中的假设）[40]。整个句子出现的概率就等于各个词

出现的概率乘积，各个词的概率可以通过语料中统计计算得到。假设句子 T 是由词序列 $w_1, w_2, w_3...w_n$ 组成，用公式表示 N-Gram 语言模型如下：

$$\begin{aligned} P(T) &= P(w_1)P(w_2)P(w_3)....P(w_n) \\ &= p(w_1)p(w_2|w_1)p(w_3|w_1,w_2)...p(w_n|w_1,w_2,w_3,w_{n-1}) \end{aligned} \tag{2.1}$$

一般常用的 N-Gram 模型是 Bi-Gram 和 Tri-Gram，本文中使用的是 Bi-Gram。Bi-Gram 标注器能够标注训练中它遇到过句子中的所有词，而只要遇到一个新词则无法分配标记。Bi-Gram 不能标注下面的词，即使在训练过程中遇到过的，因为在训练过程中从来没有见过其前面有 None 标记的词，标注器也无法标注句子的其余部分。当 n 增大时，上下文的特异性就会增加，要标注的数据中包含训练数据中不存在的上下文的几率也增大，这被称为数据稀疏问题，在 NLP 中是相当普遍的。因此，研究结果的精度和覆盖范围之间需要有一个权衡。

在处理文本特征的时候，通常一个关键词作为一个特征。这也许在一些场景下可能不够，需要进一步提取更多的特征，这个时候可以考虑 N-Gram，思路如下：以 Bi-Gram 为例，在原始文本中，以每个关键词作为一个特征，通过将关键词两两组合，得到一个 Bi-Gram 组合，再根据 N-Gram 语言模型，计算各个 Bi-Gram 组合的概率，作为新的特征。

TF-IDF 是 Term Frequency-Inverse Document Frequency 的缩写，即“词频-逆文件频率”，分为两部分，第一部分“词频”（Term Frequency，缩写为 TF），第二部分“逆文件频率”（Inverse Document Frequency，缩写为 IDF）。

TF-IDF 是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度 [41]。字词的重要性随着它在文件中出现的次数成正比增加（TF 部分），但同时会随着它在语料库中出现的频率成反比下降（IDF 部分）。因此我们需要将 TF 和 IDF 两部分综合起来表示词的重要性。

词频 (term frequency, TF) 指的是某一个给定的词语在该文件中出现的次数，计算公式如下:

$$TF_{w,D_i} = \frac{\text{count}(w)}{|D_i|}$$

公式中 $|D_i|$ 表示该文档中所有的词条数目，除以该值是为了归一化词频，以防止它偏向长的文件（同一个词语在长文件里可能会比短文件有更高的词频，但它不一定就重要）。

逆文件频率 (inverse document frequency, IDF) 是一个词语普遍重要性的度量，如果包含词条的文档越少, IDF 越大，则说明词条具有很好的类别区分能力。某一特定词语的 IDF，可以由总文件数目除以包含该词语之文件的数目，再将得

到的商取对数得到。计算公式如下：

$$IDF_w = \log \frac{N}{1 + \sum_{i=1}^{N} I(w, D_i)}$$

如果一个词越常见，那么分母就越大，逆文档频率就越小越接近 0。分母之所以要加 1，是为了避免分母为 0（即所有文档都不包含该词）。TF-IDF 的最终值由 TF 乘以 IDF，公式如下：

$$TF - IDF_{w,D_i} = TF_{w,D_i} \times IDF_w$$

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。

2.4.3　文本相似度

中文相似度按照长度可以有字与字的相似度、单词与单词的相似度、句子与句子的相似度、段落与段落的相似度和文章与文章的相似度。传统相似度的衡量计算一般可以使用编辑距离算法、余弦值法、SimHash 法、汉明距离法、最长公共子串法、最长公共子序列法等等。自然语言处理中，文本相似度计算方法分为有监督和无监督两类。

有监督方法，就是用朴素贝叶斯分类器之类的有监督模型来判断文本相似性或者计算相似度。这类方法要求有一定数量的标注语料，构建的代价比较高；由于训练语料通常无法做得很大，模型的泛化性不够，实际用起来会有点麻烦；距离计算环节的复杂度会比较高 [42]。

无监督方法，就是用余弦值、欧氏距离等方法，直接计算文本之间的距离或者相似度。这类方法的特点是：不需要标注语料，特征工程或者参数估计可以使用很大的数据；很多方法对语言的依赖比较小，可以应对多语种混杂的场景；距离计算环节复杂度较低 [43]。本文中采用余弦相似度对案例筛选的结果进行文本相似度度量：一方面计算简便，只需文本向量化数据即可计算；另一方面，余弦相似度相较于欧氏距离等方法，其值域不受维度的限制，方便进行数据归一化。

2.5　本章小结

本章主要介绍了司法类案筛选测试系统的相关技术框架、文本处理技术以及测试评估指标体系。首先介绍了构建系统的后端框架 Spring Boot 以及对应的

微服务框架 Spring Cloud、前端框架 React，然后介绍了本系统数据处理服务中使用的 Scikit-learn 库，提供了丰富的指标计算函数，最后介绍了系统中使用的文本处理技术，包括文本分词技术、文本向量化技术以及文本相似度算法。本章主要目的是为项目实现提供强有力的技术基础和理论依据。

第三章　司法案例筛选测试系统的系统需求分析与概要设计

3.1　系统总体规划

本系统主要针对的是对司法案例筛选系统有测试需求的测试人员和开发人员，如图3.1所示的司法案例筛选系统流程图，系统首先对输入的案例描述进行文本处理操作，包含预处理、分词和向量化等，之后便从案例库中进行筛选相似案例。主流筛选规则包含基于文本相似度排序筛选和基于标签匹配筛选，在实际应用中往往结合使用，其中基于标签匹配筛选规则通常是对案例数据集进行模型训练，进而对向量化输入案例信息进行标签预测。综上，本系统从司法标签预测模型和案例筛选系统级接口两个方面对司法案例筛选系统进行测试，旨在建立起一套完整的多维评估体系。

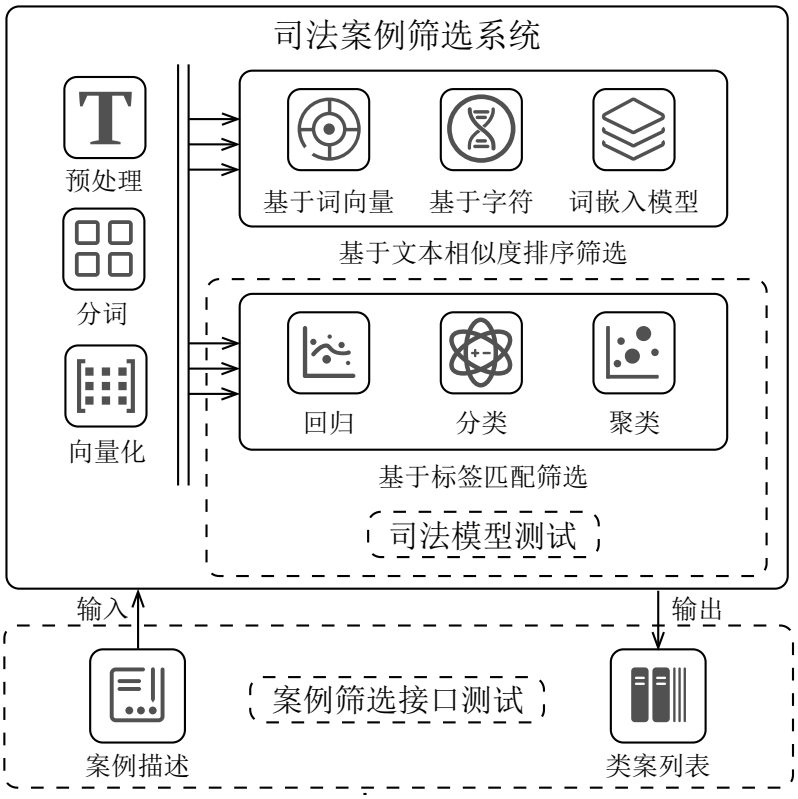

图 3.1: 案例筛选系统流程示意图

本系统计划由权限管理模块，上传管理模块，测试管理模块和数据处理模块四部分组成：通过权限管理模块来进行用户权限鉴定，服务网关配置，确保服务的安全性；上传管理对模型和数据集资源进行管理，建立用户专属的资源库；

测试管理模块对司法模型测试和案例筛选接口测试任务进行管理，实现对测试任务的新建、查询、删除等基本操作；数据处理模块一方面实现对模型和数据集资源的解析、验证、存储，另一方面负责对具体测试任务的执行，计算度量指标工作。多模块相互协同，最终生成测试报告，并提供评估结果导出等功能。本章将分析本系统的需求，明确本系统的系统边界，进而进行对应的需求分析和系统设计。

3.2　司法案例筛选测试系统需求分析

3.2.1　司法案例筛选测试系统功能需求

基于裁判文书的司法案例筛选系统，以案例的描述文本为依据，根据特定的算法分析，从指定的案情数据集合中筛选出相似的司法案例，包括相关的罪行、法律法规、刑期和罚金等关键信息。该系统为司法从业者及需要法律咨询的民众提供了高效便捷的服务。案例筛选推荐的质量水平决定了用户的满意度以及接受度，这影响到该系统的普及程度。因此需要对案例筛选的质量进行评估，为广大的案例筛选服务的使用人员提供多维度的评估的方案，既要考虑评估案例筛选本身的特性，也需要站在使用者的角度考虑实际语义的特征。

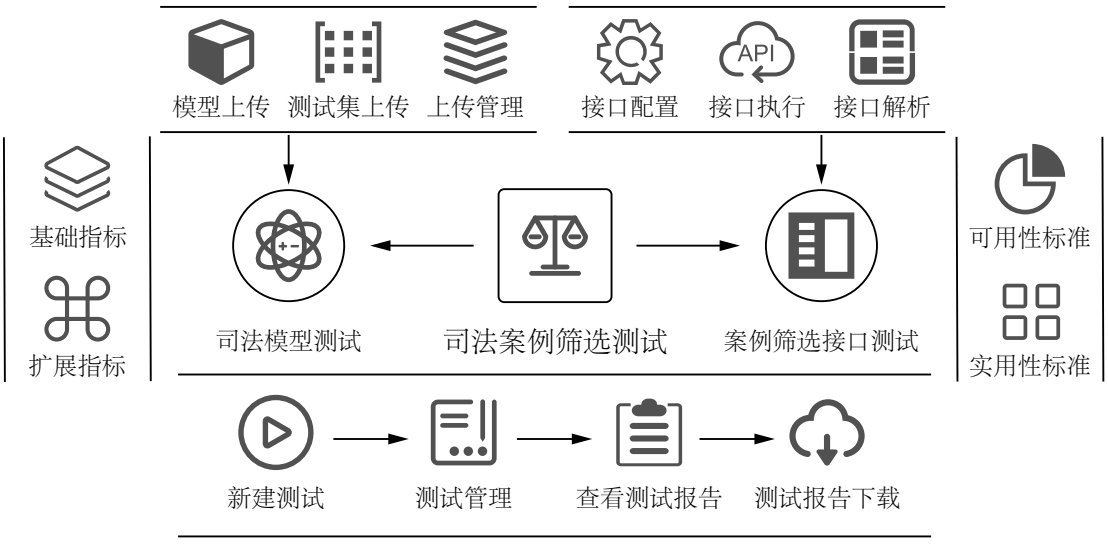

图 3.2: 案例筛选测试系统流程示意图

如图3.2所示，本系统从司法模型和系统级案例筛选接口两个方面来对案例筛选工作进行评估测试。从模型部分来看，本项目支持针对用户上传的司法模型进行评估，模型评估指标体系框架包含两类指标：基础评估指标和扩展评估指标。基础评估指标包括正确率、精确率、宏平均等机器学习领域常见的通用性

评估指标来泛化评估司法模型的稳健性；扩展评估指标结合领域特征，提出一套符合司法领域特征的评估指标，如公平性、一致性、依赖性等指标。基础评估指标用于度量司法案件筛选的模型准确性表现，而扩展指标适用于对司法案例筛选模型的整体的非准确性表现，更偏向于对模型的社会性评估。

从系统角度看，本项目支持用户通过指定系统级的案例筛选接口来评估类案推荐的质量，即给指定系统接口传入案件详情，本项目通过分析系统级接口反馈的案例列表，来评估类案推荐的质量水平。系统接口的评估，主要是通过对文本的分词成分分析，结合文本相似度等基于文本内容案情分析来评估筛选案情的相关性，通过对推荐列表的案件信息进行分析，通过法条、罚金、刑期等标签来评估接口反馈的综合质量。综上所属，系统核心业务功能需求可以总结为以下两点：

上传管理功能，本系统维护用户上传的模型文件和数据集文件，提供下载功能，并且针对用户上传的模型文件需要进行模型检验工作，来确保模型的可运行以及输入输出维度等基础信息的确定，而对于用户上传的数据集文件，需要基于数据模版对数据集进行解析重构为系统可识别的固定格式的数据集，这样有利于提升测试阶段的效率。

案例筛选测试管理功能，包含司法模型测试和案例筛选接口测试。用户发起模型评估任务，可指定上传模型库中的模型或第三方开放的模型接口，以及待测的数据集，进行模型的性能多维评估。用户发起案例筛选系统接口测试任务，输入案情描述，指定接口信息，分析反馈的案例案情筛选列表进行系统接口级的案例筛选质量评估。系统的功能性需求列表整理如表3.1所示：

表 3.1: 系统功能性需求列表

需求 ID	需求名称	需求描述
R1	司法模型上传	用户通过浏览器客户端或系统接口进行司法模型上传功能，用户上传后，后台会对模型类型进行解析，统一格式存储，识别模型的输入输出维度。本系统支持基于 Scikit-learn 库的机器学习模型、基于 Tensorflow 或 Keras 的深度学习模型文件。
R2	数据集上传	用户通过浏览器客户端或系统接口进行司法数据集上传功能，数据集上传后，后台会根据用户上传的数据解析模版进行输入数据和数据标签提取，并统一为固定格式存储。本系统支持向量数据文件和文本数据文件的上传。

R3	上传管理	用户可以查看上传模型和数据集的历史，并能查看上传后数据处理进程，此外支持对上传模型、数据集的下载、删除以及基本信息的修改功能。
R4	司法模型测试	系统支持对司法模型的测评，将用户设置的数据集运行到对应模型上，基于司法模型评估指标体系进行多维评估。
R5	案例筛选接口测试	系统支持对案例筛选接口进行评估，根据用户设置的接口及输入的案情描述，系统通过接口获取案例列表，基于案例筛选评估指标体系进行多维评估。
R6	测试状态查看	系统的测试任务后台自动化执行，支持用户离线，用户重新登录系统只需要查看测试状态即可。
R7	测试历史管理	系统用户可查看发起的测试任务的历史记录，并可查看对应详情。
R8	测试报告管理	测试任务评估结束会自动化生成测试报告展示给用户，并支持用户的下载操作。

3.2.2　司法案例筛选测试系统非功能需求

系统的非功能性需求列表如表3.2所示，主要从时间特性、并发性、可靠性、安全性和可拓展性几个方面进行描述，非功能性需求主要是为提高开发效率，保障软件系统质量角度考虑的。

表 3.2: 司法案例筛选测试系统非功能需求

类型	**非功能需求描述**
时间特性	用户进行页面跳转操作，响应时间不超过 2s
	测试报告生成后，渲染测试报告响应时间不超过 3s
并发性	支持至少 10 位用户并发访问同一系统接口
可靠性	系统故障可追溯，且发生故障扔不影响其他功能使用
安全性	服务仅开放给系统用户使用；密码等敏感数据加密保护，防止数据在传输过程中被窃取
可拓展性	类似组件统一设计；系统新增相似功能可复用核心模块

3.2.3　司法案例筛选测试系统用例设计

按照测试类型进行划分，本系统的用户分为两类：司法模型评估人员，进行司法模型的评估测试任务；系统接口评估人员，建立案例筛选系统推荐接口的评估测试任务。根据上文所述的系统功能需求，本系统用例图如图3.3所示。用例图中共涉及到 8 个用例，分别是模型上传、数据集上传、上传管理、新建司法模型测试任务、新建案例筛选接口测试任务、测试管理、查看测试报告、下载测试报告。

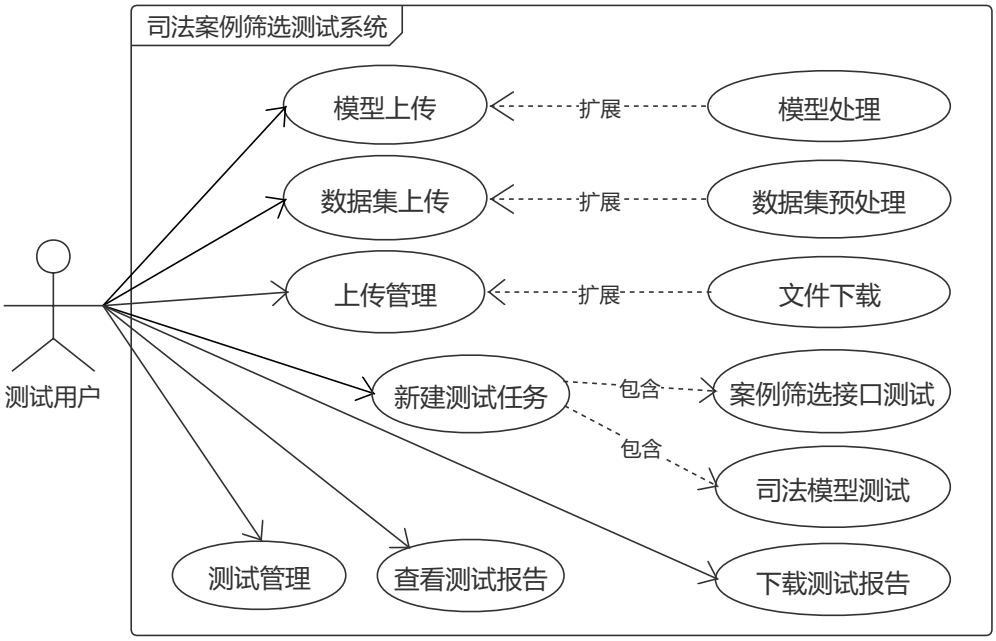

图 3.3: 案例筛选测试系统用例图

用户登录系统后，可通过系统进行模型文件的上传，并配置对应的解析信息，如模型类型、输入输出维度等，同理，也可以对数据集文件进行上传。根据用户权限的设置，用户可查看公开的以及本人上传的数据集和模型列表，有利于保护私人数据的隐私。此外，用户可以根据时间、名称、文件大小等筛选条件快速定位查找对应的模型或数据集文件。模型测试人员可通过本系统新建司法模型测试任务，从上传的模型库中选择待测司法模型，从数据集中选取测试集信息，新建模型测试任务进行评估。案例筛选接口测试人员可新建接口测试任务，指定接口 URL、请求类型和参数配置、结果模版等信息，新建接口测试任务。所有的测试任务均可在测试历史中查看，并显示对应测试任务的测试状态，所有测试任务支持离线运行，用户仅需要在测试管理的测试历史列表中查看运行状态即可。测试任务执行完毕且状态为成功后，用户可查看对应类型的测试报告，测试报告支持用户下载进行本地浏览以供参考。

表3.3描述了模型上传测试用例，用户登录系统后进行上传模型任务，需要指定模型名称，选择上传的模型文件、模型类型、模型标签含义文件、模型输入输出维度信息、自定义的模型个性化标签以及备注等信息，确认上传后，后台会进行模型解析、维度检查、存储等流程确保模型信息的准确性。上传状态可在上传管理的上传模型列表中查看。

表 3.3: 上传模型用例描述

ID	UC01	名称	上传模型
描述	用户进行上传模型操作		
参与者	系统用户		
触发条件	用户进入到模型上传界面		
前置条件	用户需要上传模型文件、新增模型上传记录		
后置条件	系统反馈模型上传状态		
优先级	高		
正常流程	1. 用户进入到上传模型界面 2. 用户选择本地模型文件上传 3. 用户选择本地标签含义配置文件上传 4. 用户配置界面中其他信息，如模型类型、维度信息、备注等 5. 用户点击确认按钮，提交上传模型信息		
扩展流程	2a. 模型文件上传失败，系统反馈失败信息 3a. 标签含义文件上传失败，系统反馈失败信息 5a. 用户点击取消按钮，返回系统首页		
特殊需求	若文件系统中含有对应文件，提示已存在该文件并进行选定，避免二次上传		

数据集上传测试用例如表3.4所示，用户登录系统后进行数据集上传操作，上传数据集文件、标签文件、标签含义文件，配置数据集名称、输入输出维度等信息，确认上传后，后台会进行文件的解析、维度检查、存储等流程确保数据的准确性。上传状态可在上传管理的上传数据集列表中查看。

表 3.4: 上传数据集用例描述

ID	UC02	名称	上传数据集
描述	用户进行上传数据集操作		
参与者	系统用户		
触发条件	用户进入到数据集上传界面		

前置条件	用户需要上传数据集文件、新增数据集上传记录
后置条件	系统反馈数据集上传状态
优先级	高
正常流程	1. 用户进入到上传数据集界面
正常流程	2. 用户选择本地数据集文件上传 3. 用户选择本地数据集对应标签文件上传 4. 用户选择本地标签含义配置文件上传 5. 用户配置界面中其他信息，如维度信息、个性标签、备注等 6. 用户点击确认按钮，提交上传数据集信息
扩展流程	2a. 模型文件上传失败，系统反馈失败信息 3a. 标签文件上传失败，系统反馈失败信息 4a. 标签含义文件上传失败，系统反馈失败信息 6a. 用户点击取消按钮，返回系统首页
特殊需求	若文件系统中含有对应文件，提示已存在该文件并进行选定，避免二次上传

用户可分别对上传的模型和数据集进行管理，可通过名称、标签、时间、维度、权限信息等多种组合条件进行上传列表信息的查询，并可查看对应上传记录中的上传解析状态，支持对上传信息的删除、查看详情操作，并允许用户下载上传状态为已完成的模型或数据集相关文件，上传管理的用例描述如表3.5所示。

表 3.5: 上传管理用例描述

ID	UC03	名称	上传管理
描述	用户进行上传管理操作		
参与者	系统用户		
触发条件	用户进入到上传管理界面		
前置条件	用户对模型或数据集信息进行含查找、删除、下载等操作		
后置条件	系统反馈用户管理操作执行后的信息		
优先级	高		
正常流程	1. 用户进入上传管理界面（模型管理或数据集管理） 2. 用户自定义筛选条件，获取对应上传记录列表 3. 用户查看某条记录上传详情，系统反馈对应详情信息 4. 用户选中记录进行删除，系统反馈删除成功 5. 用户下载列表记录中的文件资源		

正常流程	6. 系统反馈文件资源，建立下载任务
扩展流程	3a. 后台文件不存在，系统反馈资源丢失 4a. 删除失败，系统反馈失败信息 6a. 上传状态未完成，系统反馈无法下载

用户可通过从模型库中选择模型记录和从数据集库中选择数据集记录，新建司法模型的测试任务，后台会获取对应模型和数据集信息，进行维度检查、运行模型指标计算等流程，将最终计算的模型评估指标存储到数据库中。模型测试任务提交后，支持进行后台离线运行，用户可随时在测试历史中查看对应司法测试任务的测试状态。新建司法模型测试任务用例描述如表3.6所示。

表 3.6: 新建司法模型测试任务用例描述

ID	UC04	名称	新建司法模型测试任务
描述	用户新建司法模型测试任务		
参与者	司法模型测试用户		
触发条件	系统用户进入到司法模型测试界面		
前置条件	用户需要进行司法模型测试来评估上传的模型		
后置条件	系统提示任务提交成功，并跳转测试历史界面		
优先级	高		
正常流程	1. 用户进入司法模型测试界面 2. 用户从模型列表中（筛选）选择待测司法模型记录 3. 用户从数据集列表中（筛选）选择待测数据集记录 4. 用户完善基础配置信息，测试名称、标签等信息 5. 用户点击确认按钮，提交司法模型测试任务		
扩展流程	5a. 用户点击取消按钮，返回系统首页		

除了利用模型库和数据集库中的文件进行司法模型测试外，本系统还支持新建案例筛选接口测试任务，用户通过配置请求 URL 地址、参数信息、请求类型、结果解析模版等接口所需信息，后台利用调用请求服务对测试任务中指定的接口进行调用，获取对应案例筛选系统中的案例列表信息，利用结果解析模版进行接口结果关键信息的提取工作，构造筛选的案例对象列表数据，进而基于新建任务时输入的原始案例信息与案例列表数据进行案例筛选指标的计算。新建案例筛选接口测试任务用例描述如表3.7所示。

表 3.7: 新建案例筛选接口测试任务用例描述

ID	UC05	名称	新建案例筛选接口测试任务
描述	用户新建案例筛选接口测试任务		
参与者	案例筛选接口测试用户		
触发条件	系统用户进入到案例筛选接口测试界面		
前置条件	用户需要进行案例接口测试来评估案例筛选结果		
后置条件	系统提示任务提交成功，并跳转测试历史界面		
优先级	高		
正常流程	1. 用户进入案例筛选接口测试界面 2. 用户配置接口相关信息，如 URL、请求类型、参数模版等 3. 用户配置任务基础信息，如名称、备注等 5. 用户点击确认按钮，提交司法模型测试任务		
扩展流程	5a. 用户点击取消按钮，返回系统首页		

用户提交测试任务后，可进入测试历史界面查看用户提交的测试任务列表，包含模型测试和接口测试两种类型，同时提供任务名称、时间、状态、标签等条件筛选功能供测试用户快速查找对应的测试任务，任务列表中展示了测试任务的详情信息，系统支持用户选择对应的测试任务查看详情、删除、停止执行、重新执行等操作。测试历史查看用例描述如表3.8所示。

表 3.8: 测试管理用例描述

ID	UC06	名称	测试管理
描述	测试用户进行测试管理操作		
参与者	系统用户		
触发条件	用户进入到测试历史界面		
前置条件	用户需要查看测试任务列表		
后置条件	系统反馈满足条件的测试任务列表		
优先级	高		
正常流程	1. 用户进入测试任务历史界面 2. 用户设置时间、名称、状态等筛选条件筛选测试任务，系统反馈满足条件的测试历史任务列表 3. 用户选择对应测试任务，选择查看详情 4. 系统进入详情界面，展示测试任务详情信息 5. 用户选择对应测试任务，进行删除操作		

正常流程	6. 系统反馈操作执行成功信息
扩展流程	3a. 获取详情信息失败，系统反馈失败信息 6a. 操作失败，系统反馈失败信息

用户在测试任务列表界面可选择执行成功的测试任务，查看对应的测试报告，测试报告也根据测试类型分为两种报告，在司法模型测试报告中展示了模型信息、数据集信息和指标测试结果展示报告；在案例筛选接口测试报告中展示了用户输入的案例描述信息以及接口反馈的案情列表信息，针对每个推荐的案例信息都进行了案例指标的展示，并最终提供总评报告。测试报告查看用例描述如表3.9所示。

表 3.9: 查看测试报告用例描述

ID	UC07	名称	查看测试报告
描述	用户查看测试任务的测试报告		
参与者	系统用户		
触发条件	用户进入到测试历史界面		
前置条件	用户需要查看对应测试任务的测试报告		
后置条件	无		
优先级	高		
正常流程	1. 用户进入测试任务历史界面 2. 用户筛选查找目标测试任务记录 3. 用户点击对应测试任务记录的查看测试报告按钮 4. 系统跳转到对应测试任务的测试报告界面		
扩展流程	4a. 获取测试报告请求报错，系统反馈失败信息 4b. 测试任务非成功状态，系统反馈测试报告尚未生成		

用户进入测试报告页面后，可进行测试报告下载操作，将对应测试任务的结果报告保存到本地，供用户离线查看，测试报告下载用例描述如表3.10所示。

表 3.10: 下载测试报告用例描述

ID	UC08	名称	下载测试报告
描述	用户下载测试报告		
参与者	系统用户		
触发条件	系统用户进入到测试报告展示界面		
前置条件	系统用户需要将对应测试报告保存到本地		

后置条件	无
优先级	中
正常流程	1. 用户进入测试任务报告界面
正常流程	2. 用户点击下载测试报告按钮 3. 系统反馈文件资源，建立下载任务
扩展流程	3a. 系统请求失败，系统反馈无法下载

3.3　司法案例筛选测试系统设计

3.3.1　系统架构设计

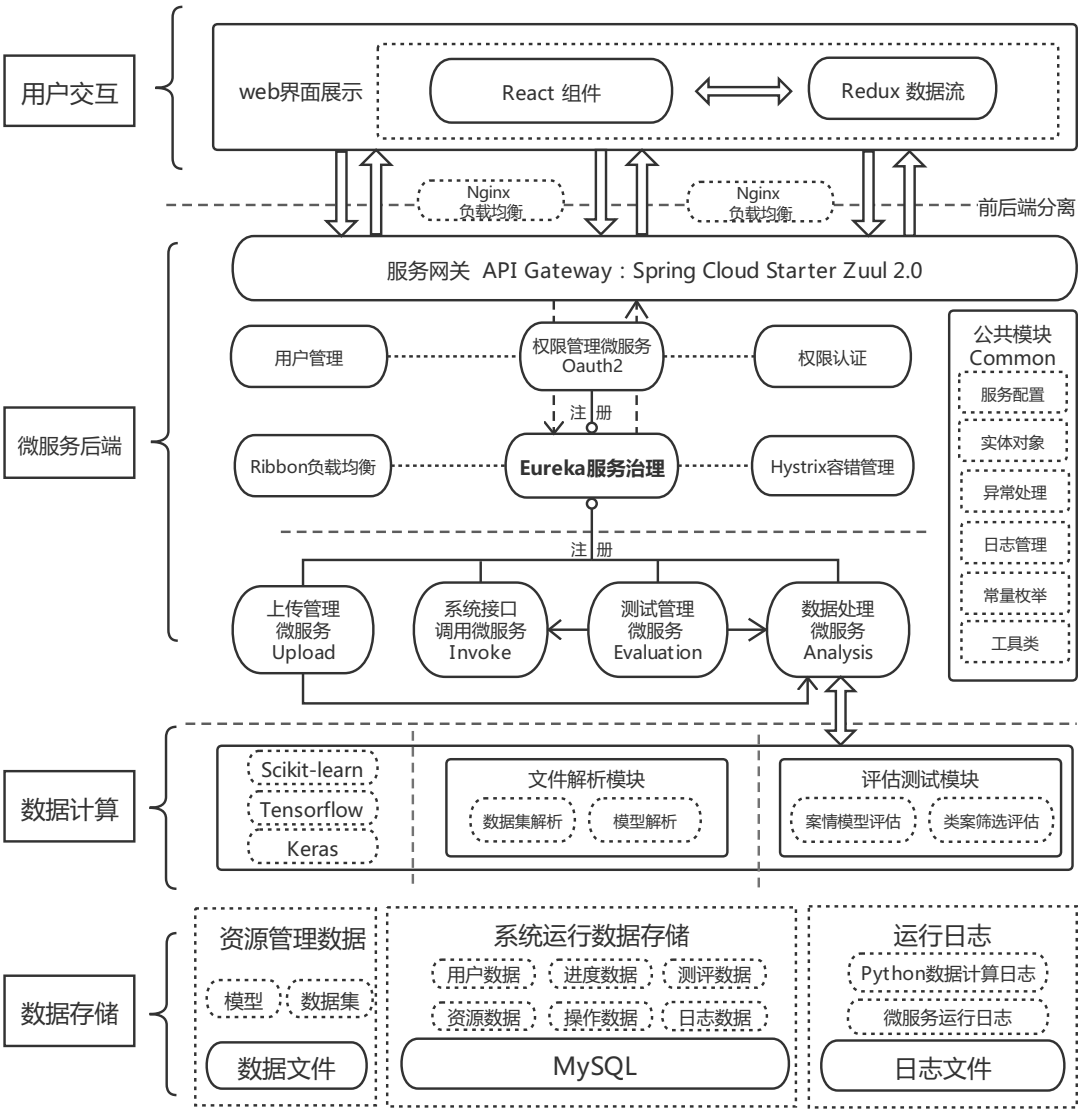

图 3.4: 案例筛选测试系统框架图

本系统采用前后端分离的架构，其中前端采用 React 框架搭建，并通过 Redux 进行整体前端数据流的管理，提供系统的 web 端的用户交互，后端通过微服务的方式基于统一的服务网关对外提供 RESTful 的接口，通过 Spring Cloud 微服务的架构将每个功能模块进行切分为独立的 Spring Boot 微服务，使用 Nginx 来进行前后端交互负载均衡，采用 Zuul2.0 技术提供异步阻塞模型的支持，来加速网关的访问效率。如图3.4所示，系统自顶向下，划分为用户交互、微服务后端、数据计算、数据存储四个部分。

系统后端架构总体设计分为权限管理、上传管理、系统接口调用、测试管理和数据处理五个微服务。通过权限管理 Oauth2 微服务来进行用户管理和权限认证，确保服务的安全性，与系统功能的权限管理模块相对应；上传管理 Upload 微服务对模型和数据集资源进行管理，实现文件上传、查询、删除的管理，与系统功能的相对应；测试管理 Evaluation 微服务负责对司法模型测试和案例筛选接口测试任务的管理，其中通过与系统接口调用 Invoke 微服务交互，进行案例筛选接口的调用、解析、统一格式反馈，与系统功能的测试管理模块相对应；数据处理 Analysis 微服务对应数据处理模块，主要负责与 Upload 微服务和 Evaluation 微服务交互，分别提供对用户上传的模型和数据集文件进行解析、校验、存储和对测试任务执行、指标计算的服务。多模块相互协同，最终生成测试报告，并提供评估结果导出等功能。通过公共模块来进行 Spring 项目配置、POJO 对象和常量定义、异常处理、日志管理、工具类等的公有化实现。

众所周知，Python 在深度学习及数据分析方面有着很强的优势，为了提升数据处理和评估测试的效率，系统架构中增加的数据计算层，主要进行的是微服务进程与 Python 数据处理进程的交互，分为文件解析处理模块和评估测试模块。对于数据存储而言，针对不同类型数据进行分类储存：对于上传模型和数据集文件直接以文件的形式存储；对于微服务运行中的业务数据通过关系型数据库来进行存储；将运行日志以日志文件形式存储，以便于后期的维护和故障排查。

3.3.2　4+1 视图模型

软件架构涉及抽象、分解、组合和美学，由元素、形式和关系组成，用来处理软件高层次结构的设计和实施，从而满足系统的主要功能和性能需求。在软件工程领域，当今普遍应用的方法为“4+1”视图模型 [44]，将软件系统架构分为五个视图：逻辑视图，将系统进行功能抽象分解，进行对象模型的设计，在本文中对应 3.3.3 小节的逻辑视图；开发视图，也称模块视图，考虑软件内部的需求，侧重软件模块的组织和管理，在本文中对应 3.3.4 小节的前后端结构图及实体关系 E-R 图；进程视图，侧重于系统的运行特性和流程，描述对应功能如何

在系统中执行，在本文中对应 3.3.5 小节的时序图；物理视图，用于解决系统拓扑结构、系统安装、通讯等问题，将软件的不同部分映射到硬件节点上，在本文中对应 3.3.6 小节的部署图；场景视图，是重要系统活动的抽象，将前四个视图有机联系起来，在本文中对应的是 3.2.3 小节的用例图。

3.3.3　系统逻辑设计

根据上文中的用例设计以及系统架构设计，本小节对系统的逻辑设计进行描述。系统的逻辑视图如图3.5所示，采用面向对象的设计方法，对系统整体逻辑进行对象模型构建和内部动态协作关系的描述。

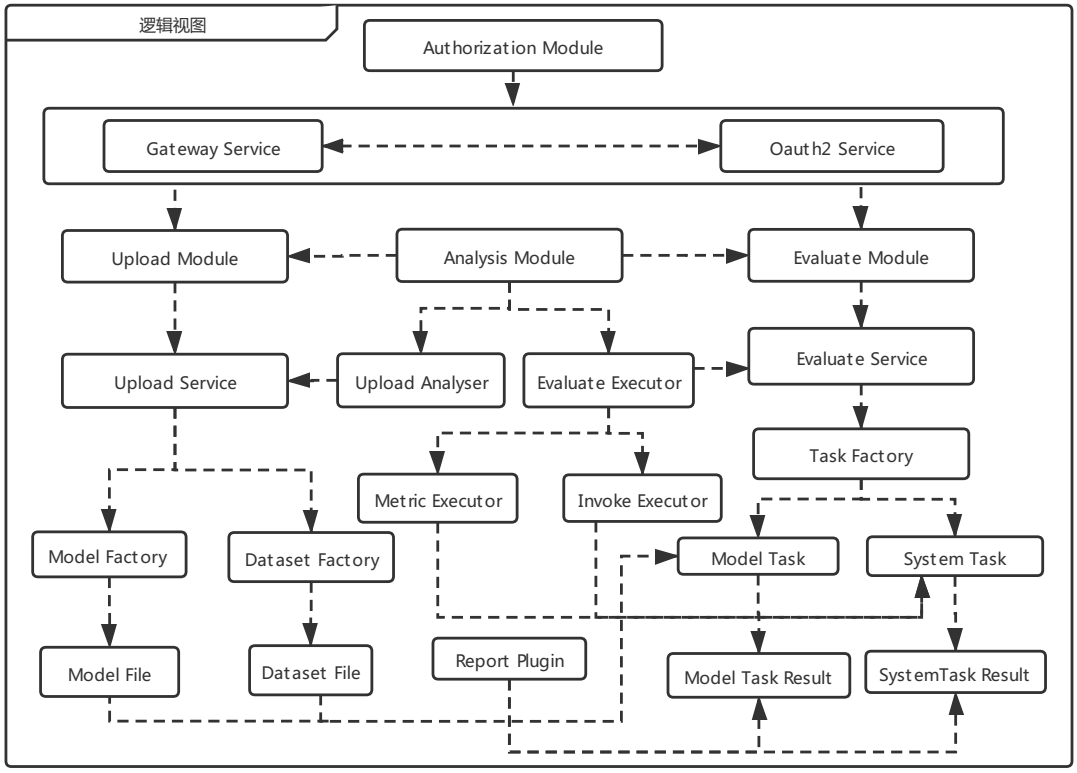

图 3.5: 案例筛选测试系统逻辑视图

系统从逻辑层面主要分为 Authorization Module、Upload Module、Evaluate Module 和 Analysis Module 四个模块：Authorization Module 负责系统统一的权限管理任务，通过 Gateway Service 的网关管理和 Oauth2 Service 的认证管理来对系统用户发起的请求进行鉴权处理，本系统的核心功能仅支持登录系统的用户使用；Upload Module 负责系统的文件管理任务，通过 Upload Service 实现核心功能，本系统进行管理的文件分为两类 Model File 司法模型文件和 Dataset File 司法数据集文件，两类文件通过对应的 Factory 工厂模版实现增删改查等操作；Evaluate Module 通过 Evaluate Service 实现系统测试任务的管理，任务库 Task

Factory 中包含功能描述中的司法模型测试任务 Model Task 和 System Task 测试任务，对应测试任务执行完毕后生成对应的 Result 测试结果实体存储在系统中，供测试人员查看测试报告；Analysis Module 主要负责数据分析功能，分别为 Upload Module 提供司法模型文件和数据集文件的解析、校验、存储等操作，也为 Evaluation Module 提供模型运行支持、案例筛选接口调用解析支持以及测试任务指标计算的服务。

3.3.4　系统开发设计

本系统的前端采用 React 架构，该部分主要由 index、page、component、router、service、redux、util 等模块组成，如图3.6所示。

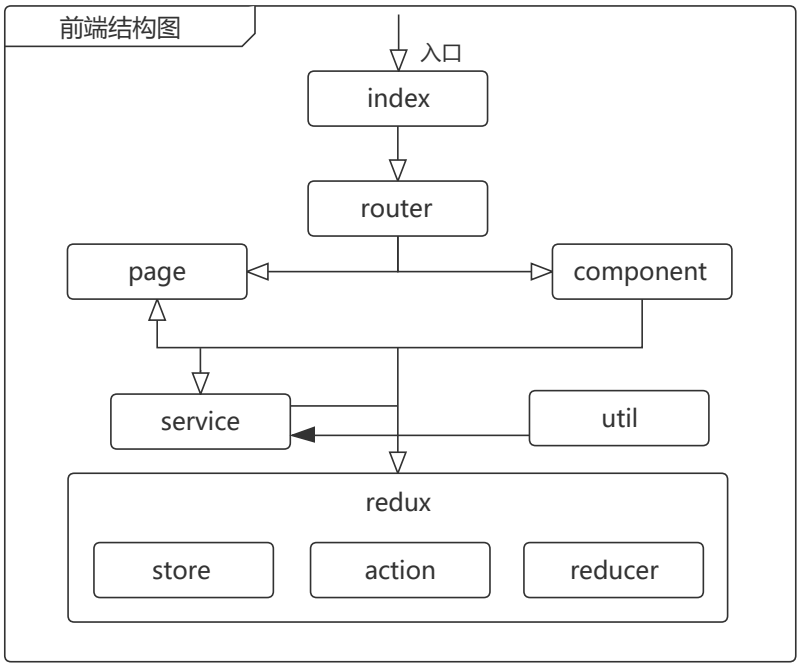

图 3.6: 案例筛选测试系统前端结构图

- index 是前端项目的入口文件，当访问前端网址时，首先渲染 index 资源进入到网站首页。
- page 是前端项目的页面模块，包含前端各个页面的静态文件（通过 JSX 来描述用户界面）、样式文件等。
- component 包含前端页面中使用的组件，是组成页面结构的基础，如导航栏、菜单、图表等自定义组件均在该模块中。
- router 模块统一管理前端项目的路由，通过管理 URL，实现不同组件之间的切换、跳转以及状态的变化。

- service 负责处理页面中的各类请求以及逻辑功能，对 react 的页面数据进行状态的维护工作。
- redux 模块包含 store、action 和 reducer 三个部分，store 为前端提供一套整体的状态信息，action 提供描述界面行为的数据结构，而 recuder 则通过纯函数的形式，负责根据 action 来改变 store 中的状态信息。
- util 中包含模块化的工具函数，供其他模块服务使用，提高代码的复用性。

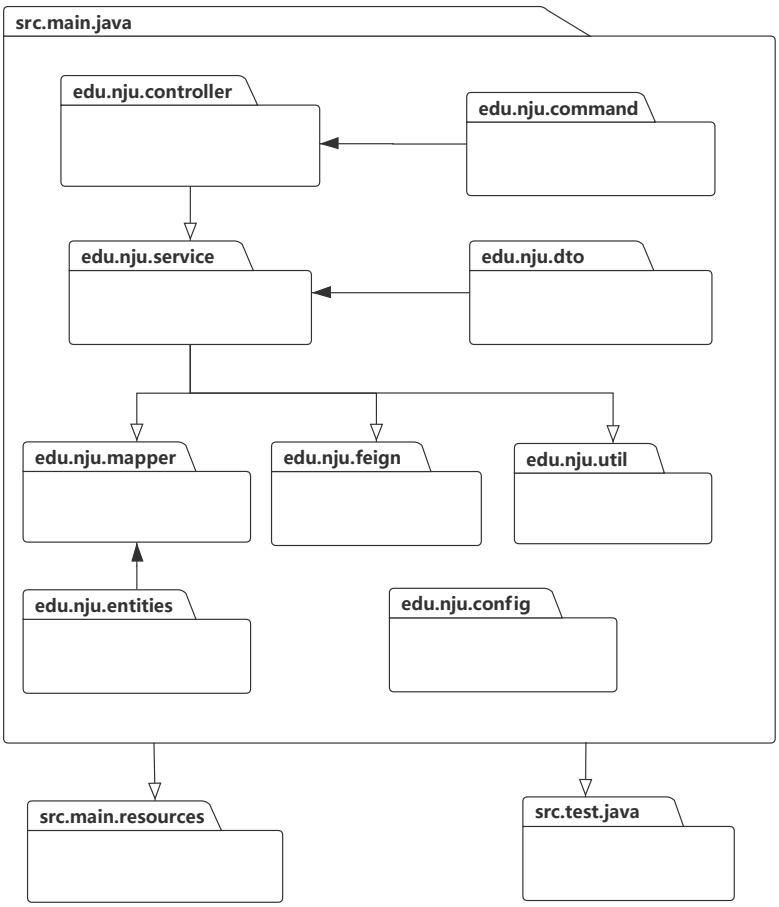

图 3.7: 案例筛选测试系统后端结构图

为遵循 MVC 架构风格，参考 Spring Boot 官方建议项目结构，本系统后端包结构图如图3.7所示。

- controller 包主要负责具体业务模块的流程控制，实现对 HTTP 请求的拦截、转发和解析；service 包主要进行业务逻辑的接口定义和代码实现；mapper 包代表 dao 层，系统使用 Mybatis 进行数据的增删改查操作。以上三个包中的类分层依赖，通过 @Resource 注解进行依赖注入。

- feign 包是负责微服务之间相互调用的 RESTful 风格接口包，通过 @FeignClient 注解注入到使用方微服务中，在 service 层通过 @Autowire 或 @Resource 注解注入进行使用。
- entities 包中对应系统涉及的实体类信息，每个类对应数据库中的一张表。
- command 包中对应的是前端进行表单提交或数据请求的参数对象，由 controller 层解析接收，进而进行后台的逻辑处理。
- dto 包对应的是后端反馈给前端的数据传输对象，基于实体类对象构建，避免传输整个表字段，能够提高信息传递的安全性，字段信息减少也能在一定程度上提高请求效率。
- config 包对应的是后端服务的 Java 代码配置类，如异常处理，类型转换，请求拦截等服务配置项。
- util 包中提供系统通用的工具类，如字符串处理、随机数生成、加密解密操作、文件操作、Json 处理等相关功能。
- resources 中包含各类配置文件信息，如 spring boot 的服务器配置、mysql 连接、redis 连接配置、自定义配置等。此外，还包含项目日志配置文件和 Mybatis 的 mapper 模版文件。
- test 包中主要进行测试用例代码的编写，通过 @SpringBootTest 注入 Spring 环境配置，进行单元测试和系统测试的执行。

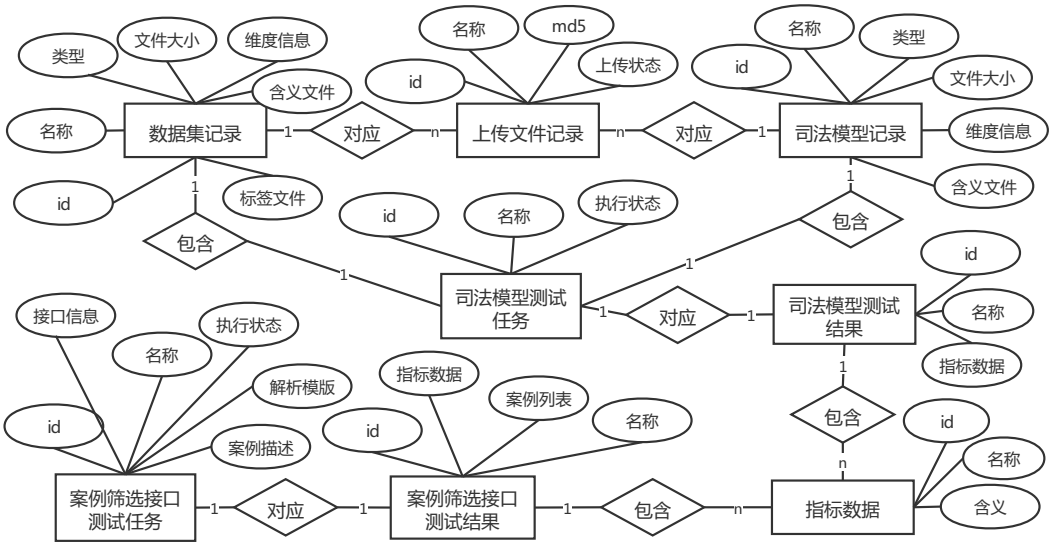

图 3.8: 案例筛选测试系统核心实体关系图

本系统的核心实体关系图如图3.8所示，将上传的文件记录设计为一个实体，与司法模型记录和数据集记录实体关联，对上传文件解析校验成功后，数据库中新增对应的模型或数据集记录。测试任务实体包含司法模型测试任务和案例筛选接口测试任务，其中司法模型测试任务实体包含模型记录和数据集记录，用于模型的测试评估。两类测试任务实体各自关联对应的测试结果实体，用于存储测试结果信息。此外，测试结果实体中的指标信息与指标数据实体关联，便于获取指标含义等基础信息。

根据逻辑视图中的四个模块划分，对涉及到的实体表结构进行设计。权限管理模块 Authorization Module 主要涉及用户 User 和角色 Role 实体，存储基本用户详情信息，设计较为简单，故不列表说明。上传管理模块 Upload Module 涉及到上传文件记录实体 FileRes、司法模型记录实体 ModelFile 和数据集记录实体 Dataset，分别对应数据库中的 t_file_res、t_model_file 和 t_dataset 三张表，数据库表字段设计分别如表3.11、表3.12和表3.13所示。

表 3.11: t_file_res 表字段详细设计

字段	含义	类型	描述
id	上传文件 id	int(11)	上传文件主键
user_id	系统用户 id	varchar(45)	系统用户主键
path	文件相对路径	varchar(200)	文件存储相对路径
size	文件大小	int(20)	文件大小，单位字节
name	文件名称	varchar(45)	上传的源文件名称
md5	文件 md5 标志	varchar(45)	上传文件唯一标识
status	文件上传状态	tinyint(4)	上传状态

其中 md5 字段建立了唯一索引，确保上传文件的唯一性，用于对上传的文件进行鉴定是否存在于文件系统，避免相同文件多次上传导致资源的浪费。status 状态字段描述文件上传的状态，结合前端文件切块和数据库记录可实现文件的断点续传功能。

表 3.12: t_model_file 表字段详细设计

字段	含义	类型	描述
id	模型记录 id	int(11)	模型记录主键
user_id	用户 id	varchar(45)	系统用户主键
filename	模型名称	varchar(45)	指定的模型名称
file_res_id	上传模型文件 id	int(11)	对应 t_file_res 表主键

label_name_file_res_id	上传标签含义文件 id	int(11)	对应 t_file_res 表主键
file_path	模型文件存储路径	varchar(200)	模型解析校验后的存储路径
label_name_file_path	标签含义文件存储路径	varchar(200)	标签含义文件处理后的存储路径
open	是否公开	tinyint(1)	该模型是否公开
size	模型文件大小	int(20)	模型文件大小, 单位字节
input_dim	模型输入维度	varchar(45)	模型输入维度, 用于校验
output_dim	模型输出维度	varchar(45)	模型输出维度, 用于校验
type	模型类型	tinyint(4)	司法模型的类型
status	模型处理状态	tinyint(4)	模型处理的状态
label	自定义 label	varchar(100)	个性化标签，用于查询
is_deleted	模型是否删除	tinyint(1)	逻辑删除标志

表3.12和表3.13分别描述了模型记录和数据集记录数据库表详细设计，关联上传文件表的主键信息。对于模型记录，关联上传模型文件 id 和标签含义文件 id，为模型处理操作指定源文件，而对于数据集记录，则关联上传数据集文件 id、标签文件 id 和标签含义文件 id。type 字段在 t_model_file 表中表示模型类型，与项目开发中定义的 ModelType 枚举类对应，而在 t_dataset 则表示数据集的数据类型，与 DataType 枚举类对应。

表 3.13: t_dataset 表字段详细设计

字段	含义	类型	描述
id	数据集记录 id	int(11)	数据集记录主键
user_id	用户 id	varchar(45)	系统用户主键
filename	数据集名称	varchar(45)	指定的数据集名称
file_res_id	上传数据集文件 id	int(11)	对应 t_file_res 表主键
label_file_res_id	上传标签含义文件 id	int(11)	对应 t_file_res 表主键
label_name_file_res_id	上传标签文件 id	int(11)	对应 t_file_res 表主键
file_path	数据集文件存储路径	varchar(200)	数据集文件处理后的存储路径

label_file_path	标签文件存储路径	varchar(200)	标签文件处理后的存储路径
label_name_file_path	标签含义文件存储路径	varchar(200)	标签含义文件处理后的存储路径
open	是否公开	tinyint(1)	该数据集是否公开
size	数据集文件大小	int(20)	数据集文件大小，单位字节
input_dim	数据集维度	varchar(45)	数据维度，用于校验
output_dim	标签维度	varchar(45)	标签维度，用于校验
type	数据集类型	tinyint(4)	数据集格式类型
status	数据集处理状态	tinyint(4)	数据集处理的状态
label	自定义 label	varchar(100)	个性化标签用于查询
is_deleted	数据集是否删除	tinyint(1)	逻辑删除标志

input_dim 和 output_dim 字段用于对上传文件的输入输出维度校验：对模型记录表来说，输入维度是指模型输入数据的维度，输出维度是指模型预测输出的维度；对数据集记录表而言，输入维度表示上传的数据集文件维度，输出维度表示上传标签文件的维度。status 字段用于标识对应模型或数据集的上传处理状态，根据数据处理服务的反馈信息进行动态更新。本系统对司法模型和数据集记录的删除操作是可恢复的，故设置逻辑删除标志字段 is_deleted，可通过数据库操作恢复误删记录。

表 3.14: t_model_task 表字段详细设计

字段	含义	类型	描述
id	司法模型测试任务 id	int(11)	司法模型测试任务主键
user_id	用户 id	varchar(45)	系统用户主键
name	司法模型测试任务名称	varchar(45)	测试任务名称
model_id	测试模型 id	int(11)	上传模型记录主键
dataset_id	测试数据集 id	int(11)	上传数据集记录主键
state	司法模型测试任务状态	tinyint(4)	任务执行状态
label	自定义 label	varchar(100)	个性化标签用于查询
is_deleted	司法测试任务是否删除	tinyint(1)	逻辑删除标志

评估模块 Evaluation Module 和分析模块 Analysis Module 涉及 t_model_task、t_system_task、t_model_evaluation、t_system_evaluation 和 t_metric 表，分别用于

存储模型测试任务记录、案例筛选接口测试任务记录、模型测试结果记录、案例筛选接口测试结果记录以及指标详情信息，表字段设计如表3.14、表3.15、表3.16、表3.17和表3.18所示。t_model_task 表记录了司法模型测试任务名称、测试模型 id、数据集 id 信息，t_system_task 表则包括案例筛选接口测试任务名称、类按描述、接口请求 URL 地址、接口类型、参数模版、解析模型信息。测试任务两表中 state 字段表示任务执行状态，包含正在执行、成功和失败三种状态。

表 3.15: t_system_task 表字段详细设计

字段	含义	类型	描述
id	案例筛选接口测试任务 id	int(11)	接口测试任务主键
user_id	用户 id	varchar(45)	系统用户主键
name	案例筛选接口测试任务名称	varchar(45)	测试任务名称
fact	案例描述	varchar(1000)	案例描述内容
request_url	接口请求 URL	int(11)	接口请求 URL
request_type	接口类型	int(11)	接口类型
request_template	接口参数模版	int(11)	参数模版，用于参数构建
response_template	反馈解析模版	int(11)	数据反馈的解析模版
state	案例筛选接口测试任务状态	tinyint(4)	任务执行状态
label	自定义 label	varchar(100)	个性化标签用于查询
is_deleted	司法测试任务是否删除	tinyint(1)	逻辑删除标志

当测试任务执行成功后，对应 t_model_evaluation 表或 t_system_evaluation 表中会新增对应测试结果记录。如表3.16和表3.17所示，司法模型测试任务结果 t_model_evaluation 表中主要记录关联的司法模型测试任务 id、测试集测试样本数量、测试分类数量等基础信息；案例筛选接口测试任务结果 t_system_evaluation 表中包含对应测试任务 id、接口反馈的案例数量以及案例列表数据的存储路径信息。测试结果表中 detail_json 字段用于存储测试结果指标 json 数据的存储路径。response_json 字段利用存储路径持久化，是为了避免数据表字段存储内容过多导致查询效率降低。detail_json 字段选择使用结果文件路径进行持久化，是为了更方便灵活解决一对多的映射关系，任务结果指标文件中存储多个对应指标数据，存储文件路径使指标增删更加灵活。

表 3.16: t_model_evaluation 表字段详细设计

字段	含义	类型	描述
id	司法模型测试结果记录 id	int(11)	模型测试结果记录主键
user_id	用户 id	varchar(45)	系统用户主键
model_task_id	司法模型测试任务 id	int(11)	模型测试任务主键
test_size	测试数据集数量	int(11)	数据集测试样本数量
label_size	测试分类数量	int(11)	测试分类数
detail_json	测试结果存储路径	varchar(200)	测试结果 json 存储路径
is_deleted	测试结果是否删除	tinyint(1)	逻辑删除标志

表 3.17: t_system_evaluation 表字段详细设计

字段	含义	类型	描述
id	案例筛选接口测试结果记录 id	int(11)	接口测试结果记录主键
user_id	用户 id	varchar(45)	系统用户主键
system_task_id	案例筛选接口测试任务 id	int(11)	接口测试任务主键
case_list_size	案例筛选列表数量	int(11)	接口反馈筛选案例数量
response_json	接口反馈数据存储路径	varchar(200)	系统对接口返回数据解析处理后的存储路径
detail_json	测试结果存储路径	varchar(200)	测试结果 json 存储路径
is_deleted	测试结果是否删除	tinyint(1)	逻辑删除标志

指标数据 t_metric 表用于存储测试任务执行计算的指标详情，包括指标中英文名称、指标含义、阈值、基线数值、相关性等字段信息。中文名称主要用于测试报告展示用，英文名称在指标计算结果文件（即测试任务结果表中 detail_json 路径指向的文件）中与指标数值构成键值对，供前端解析匹配指标详情使用。指标值域为最大值最小值字段构成的区间 [min,max]，上下限均为具体数值，若指标下限或上限为无穷，在指标计算时利用单调区间映射函数将指标值域映射到有限区间上（一般为 [0,1]），便于指标展示及综合评分计算。standard 字段存储对应指标基线数值信息，在指标值域中划分优劣区间，correlation 字段标识指标值与对应性能优劣的相关性，正相关表示指标值越大对应评估指标性能越优秀，负相关则相反。

表 3.18: t_metric 表字段详细设计

字段	含义	类型	描述
id	指标记录 id	int(11)	指标记录主键
type	指标类型	tinyint(4)	指标类型
name	指标名称	varchar(45)	指标中文名
name_en	指标英文名称	varchar(45)	指标英文名
description	指标含义	varchar(200)	指标含义描述
max	指标上限	decimal(10,4)	指标最大值
min	指标下限	decimal(10,4)	指标最小值
standard	基线数值	decimal(10,4)	指标优劣分界值
correlation	指标相关性	tinyint(1)	指标数值与优劣相关性
is_deleted	指标是否删除	tinyint(1)	逻辑删除标志

3.3.5　系统进程设计

根据上述需求及设计，本节从用户操作角度将系统分为上传管理和测试管理来分析介绍案例筛选测试系统的进程设计。

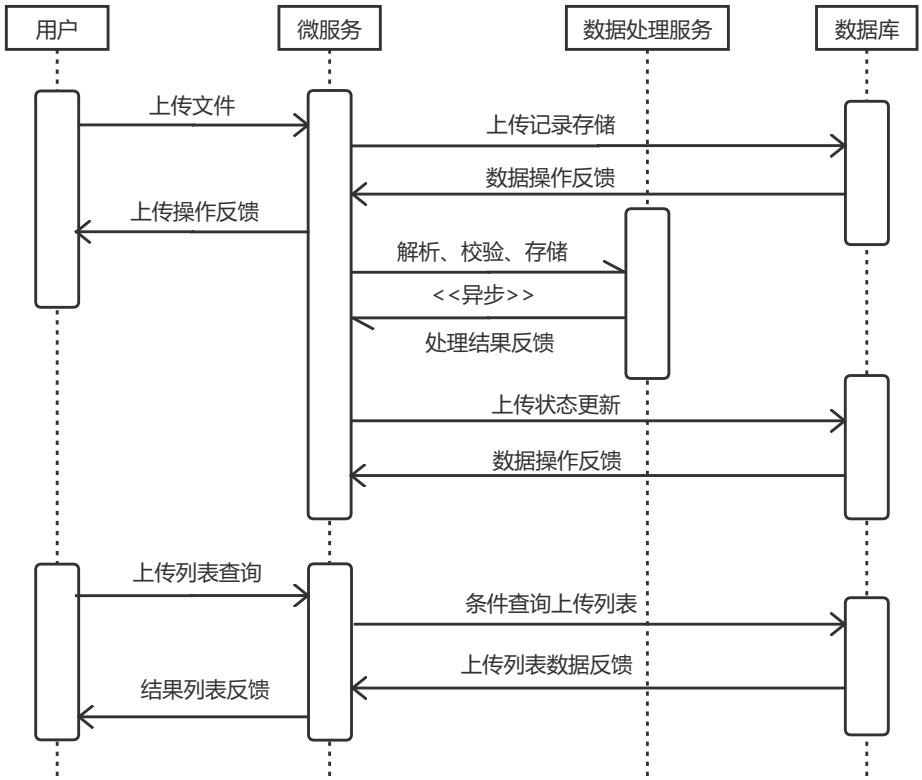

图 3.9: 上传管理进程视图

案例筛选测试系统的上传管理进程视图如图3.9所示，系统用户主要进行上传文件和上传列表查询的操作。用户进行上传模型或数据集，首先将文件上传到微服务指定的文件系统中，将文件上传记录存储到数据库，文件上传后异步开启对上传的模型或数据集的解析、校验、存储工作，该异步操作经由数据处理服务执行，前端无需同步等待，提高响应速度。数据处理服务反馈后微服务便将处理状态更新持久化到数据库中，用户刷新即可见。用户可对上传的模型或数据集信息进行列表查询，请求至微服务向数据库条件查询对应上传列表信息，进而反馈到用户界面，列表信息包含文件的基础信息以及异步处理的状态。同理，用户对上传模型或数据集详情查询或删除某条记录流程均与条件查询相同。

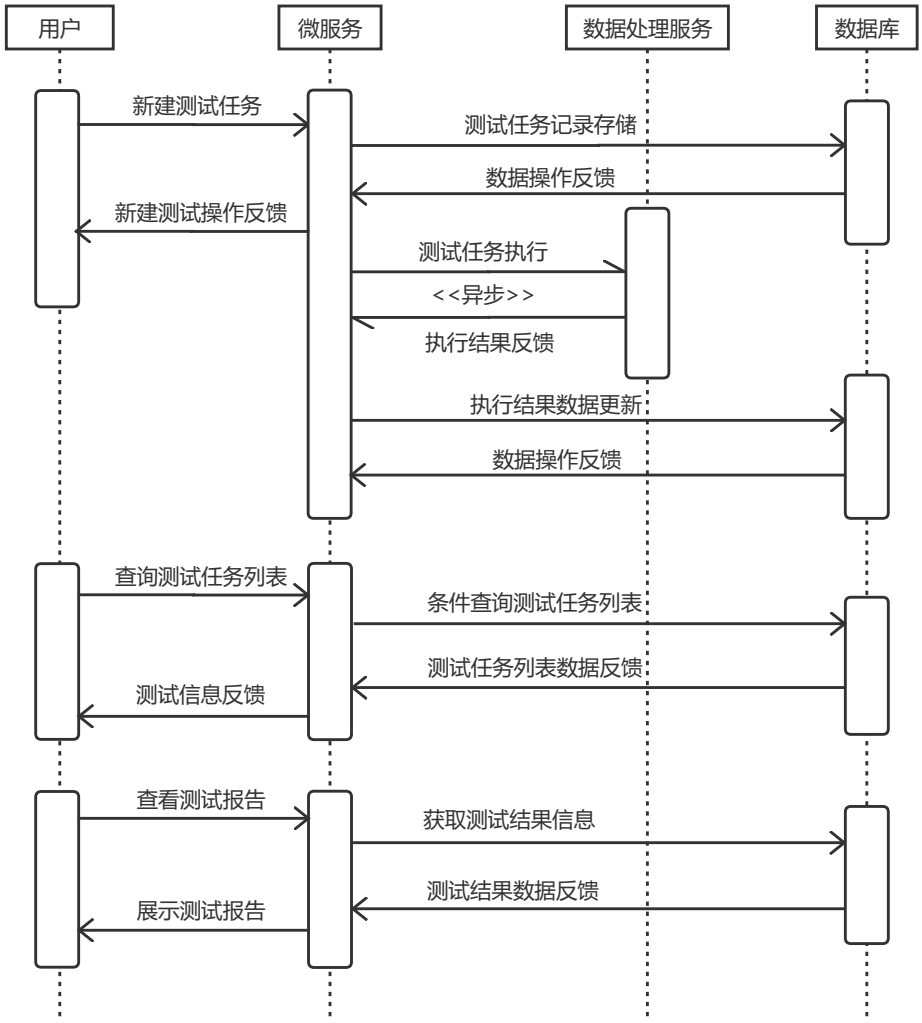

图 3.10: 测试管理进程视图

案例筛选测试系统最重要的核心功能即对司法模型和案例筛选系统接口进行测试，其进程设计如图3.10所示。测试用户根据测试需求新建对应类型测试任

务，后台微服务同步将测试任务记录存储到数据库中，测试记录持久化后即向用户反馈，同时后台调用数据处理服务，异步执行测试任务，计算测试任务的评估指标。根据测试结果反馈信息，微服务更新测试状态到数据库中，测试任务包含正在执行、成功、失败三种状态，执行测试成功同时会将结果相关信息持久化到数据库中。异步机制使本系统测试任务支持离线运行，用户可请求后台进行条件查询获取测试任务列表，来查看任务基础信息及执行状态。此外，用户可查看对应测试任务的测试结果，后台获取数据库持久化的指标数据，前端以测试报告的形式展示。

3.3.6　系统部署设计

系统整体的部署设计如图3.11，前后端分离各自部署在单独节点。后端采用微服务架构，包括 eureka 注册中心微服务、oauth2 权限管理微服务、gateway 网关微服务、upload 上传管理微服务、evaluate 测试管理微服务、invoke 接口调用微服务、analysis 数据处理微服务。其中，analysis 微服务通过插件 Sidecar 融合异构 Flask 框架的 Python 服务，以满足数据处理需求。每个微服务各自部署，可根据资源环境及性能需求进行多实例部署，通过横向扩展服务数量提高接口访问效率，提高系统性能。此外，前端 React 项目和数据处理 Python 服务打包后可使用 Nginx 进行部署服务。

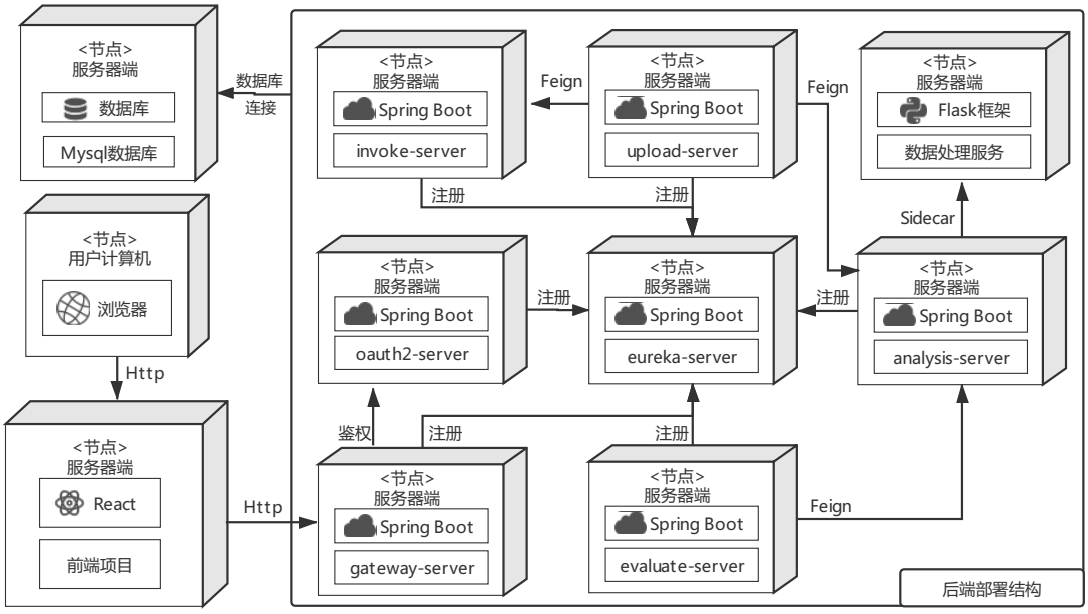

图 3.11: 案例筛选测试系统部署图

3.4　系统评估指标设计

本小节针对模型测试和案例筛选系统接口测试分别进行了评估指标的设计，如图3.12所示，展示了本系统的司法模型评估指标体系，该指标体系分为两个指标层级，一级指标包括基础评估指标、扩展评估指标，共 2 项；二级指标包括准确率、精确率、宏平均、微平均、拟合度、AUC、纠正度、解释性、公平性、一致性、相似性、依赖性，共 12 项。基础评估指标用于评估司法模型在指定数据集下的基本性能表现，如精度，分类能力等；扩展评估指标适用于对司法案例筛选模型领域特征进行评估，如公平性评估在数据质量受到影响时模型预测性能，依赖性用于评估模型对测试集质量的依赖程度。该指标体系可应用于司法多分类模型和回归模型任务中, 涉及到罪名、刑期、罚金、法条预测等领域。

图 3.12: 司法模型评估指标体系框架图

准确率用于考察司法模型的预测准确度，评估司法案例筛选模型的基础精度，适用于各个领域的模型评估，也是司法模型最基础的评估指标。准确率基于

模型在数据集预测结果的混淆矩阵（TP, FP, TN, FN）计算，其计算公式如下：

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{3.1}$$

精确率，用于考察司法模型的精确程度。在传统机器学习指标中精确率（*Precision*）是基于混淆矩阵计算的，而在CAIL2018竞赛中，采用 *Amp_score* 来计算模型预测标签与真实标签的误差范围，本系统的模型精确率基于 *Amp_score* 进行计算，其计算公式如下，其中 n 表示样本数量，y^i_{test} 表示测试样本 i 的真实分类标签，$\hat{y}^i_{test}$ 表示测试样本 i 司法模型预测分类标签，*Normalization* 表示归一化函数：

$$\begin{aligned} Precision &= 1 - Normalization(Amp\_score) \\ Amp\_score &= \frac{\sum_{i=1}^{n} |\log(y^i_{test} + 1) - \log(\hat{y}^i_{test} + 1)|}{n} \end{aligned} \tag{3.2}$$

宏平均用于考察司法模型多分类准确率，其原理与F1度量相同，在传统评估体系中精确率（*Precision*）和召回率（*Recall*）分别用于考察模型的查准和查全的性能，然而在实际应用中，这两个指标往往出现矛盾的情况，因此需要综合考虑精确率和召回率。宏平均则基于模型的每个分类进行相关统计指标的计算，然后对所有类的指标进行算数平均值的计算，该指标能从多分类的宏观角度评估模型在数据集上的性能，计算公式如下，其中 n 表示分类数：

$$\begin{aligned} Macro\_P &= \frac{1}{n}\sum_{i=1}^{n} Precison_i \\ Macro\_R &= \frac{1}{n}\sum_{i=1}^{n} Recall_i \\ Macro\_score &= \frac{2 \times Macro\_P \times Macro\_R}{Macro\_P + Macro\_R} \end{aligned} \tag{3.3}$$

微平均与宏平均相对，也用于评估司法模型的多分类准确率，粒度由每个类的指标变更为针对每个未分类的实例进行指标计算，计算公式如下，其中 n 表示实例数量：

$$\begin{aligned} Micro\_P &= \frac{1}{n}\frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \\ Micro\_R &= \frac{1}{n}\frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \\ Micro\_score &= \frac{2 \times Micro\_P \times Micro\_R}{Micro\_P + Micro\_R} \end{aligned} \tag{3.4}$$

拟合度考察司法模型的拟合优度，在司法案例筛选模型模型的指标中，指的是预测结果与真实结果的拟合程度，该指标基于均方误差（MSE）计算，其

公式如下，其中 n 表示测试样本数量：

$$\begin{aligned} Fitness &= 1 - \frac{MSE}{2} \\ MSE &= \frac{1}{n}\sum_{i=1}^{n}(y_{test}^{i} - \hat{y}_{test}^{i})^2 \end{aligned} \tag{3.5}$$

AUC 用于考察司法模型的整体分类性能，相对于准确率、精确率等精度指标，AUC 在测试样本不均衡的情况下也能反映出模型的性能，AUC 在机器学习中表示的是 ROC 曲线下的面积，与模型性功能呈正相关，其计算公式如下：

$$AUC = \frac{\sum_{ins_i \in P} rank_{ins_i} - M \times \frac{M+1}{2}}{M \times N} \tag{3.6}$$

纠正度用于考察纠正司法模型误差的难度，通过汉明距离来评估由预测结果纠正到正确结果的难度，其计算公式如下，其中 n 表示测试样本数量：

$$\begin{aligned} Correctiveness &= 1 - Normalization(Hamming\_score) \\ Hamming\_score &= \frac{1}{n}\sum_{i=1}^{n} hamming(y_{test}^{i}, \hat{y}_{test}^{i}) \end{aligned} \tag{3.7}$$

解释性用于考察司法案例筛选模型解释性的好坏，越接近于 1 说明测试集越能解释预测结果的方差变化，值越小则说明效果越差。解释性指标通过解释方差（EVS）来进行计算，其公式如下，其中 var 表示方差的计算：

$$EVS = 1 - \frac{var(y_{test}^{i} - \hat{y}_{test}^{i})}{var(y_{test}^{i})} \tag{3.8}$$

公平性用于考察司法案例筛选模型处理缺失值和异常值的能力，即模型是否能在有一定噪声和歧视性文本的情况下实现公平，其计算公式如下，其中 n 表示变异数据集的数量（是指对原始数据集进行数据扰动后生成的数据集），Amp_score 为原始数据集的误差分数，Amp_score_i 为第 i 个变异数据集的误差分数：

$$\begin{aligned} Fairness &= 1 - Normalization(Deviation) \\ Deviation &= \frac{\sum_{i=1}^{n}(Amp\_score - Amp\_score_i)^2}{n} \end{aligned} \tag{3.9}$$

一致性用于度量模型审判结果和预期结果的一致程度，其计算公式如下，其中 p_o 表示数据集的总体分类精度，即准确度，m 表示数据集的分类数量，n 表示数据集的大小，a_i 表示分类 i 下司法样本数量，b_i 表示司法模型预测结果中分类 i 的样本数量：

$$Consistency = \frac{p_o - p_e}{1 - p_e} p_e = \frac{\sum_{i=1} ma_i \times b_i}{n \times n} \tag{3.10}$$

相似性用于考察司法模型审判结果接近预期结果的程度，基于 Jaccard 相似系数度量，计算公式如下，其中 y_{test} 表示司法数据集的真实结果集，$\hat{y}_{test}$ 表示模型预测结果集：

$$Similarity = Jaccard(y_{test}, \hat{y}_{test}) \tag{3.11}$$

依赖性用于考察司法模型对测试数据集的依赖程度，在无法保证数据集的均衡性的情况下评估司法模型的表现效果，基于混淆矩阵进行马修相关系数计算（MCC），计算公式如下：

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.12}$$

对于司法案例筛选接口测试而言，其评估指标主要分为针对单条案例的评估指标以及针对筛选列表的整体评估指标, 如表3.19所示，其中可用性标准用于单条案例评估，实用性标准是对案例筛选列表整体进行评估的指标。

表 3.19: 案例筛选接口评估指标体系表

可用性标准	技术边界相对模式	推荐可靠性
	基础性、关键性的技术	相似度
实用性标准	类案判决标准定轨-司法偏离度	刑期预测偏离度
		罚金预测偏离度
		法条预测偏离度
		罪名预测偏离度
	正向效应-推荐正态性	刑期推荐正态性
		罚金推荐正态性
		法条推荐正态性
		罪名推荐正态性
	统一裁判尺度-案件止讼率	案件刑期止讼率
		案件罚金止讼率
		案件法条止讼率
		案件罪名止讼率

司法案例筛选系统本身存在着一定风险。一方面，该系统对于辅助法官审判的辅助范围和系统实现上的技术边界相对模式存在着一定的模糊性。人工智能介入权威性法律判断的场合可能会削弱法官主体地位、模糊司法责任，甚至在极端情况下可能引发冤假错案，这就要求案例筛选系统具有一定的可靠性。编辑距离（Edit Distance），又称 Levenshtein 距离，是指两个向量化后的案件描述

文本之间，由一个转成另一个所需的编辑操作次数。最小编辑距离，是指所需最小的编辑操作次数，可以用来刻画每一份被推荐的案件用于对原始案件决策时的可用性程度。计算公式如下，其中，当 $A[i] = B[j]$ 时，flag = 0，否则 flag = 1：

$$edit[i][j] = \begin{cases} 0 & i = 0, j = 0 \\ j & i = 0, j > 0 \\ i & i > 0, j = 0 \\ \min(edit[i-1][j] + 1, & \\ edit[i][j-1] + 1, & i > 0, j > 0 \\ edit[i-1][j-1] + flag) & \end{cases} \tag{3.13}$$

另一方面，智慧法院的人工智能方面的实现仍然存在许多的基础性、关键性的技术瓶颈制约。推荐算法的准确性是案例筛选系统契合人民法院智能化要求的前提条件。案例筛选不精准，则会影响类案的参考价值，降低案例筛选系统的可用性。案件相似度计算就是将两个案件描述的特征向量化，然后通过余弦公式计算两者之间的相似性即可。本系统利用 N-Gram 和 TF-IDF 技术对案例文本进行向量化处理，进一步计算筛选案例的相似度指标。

$$Similarity(A, B) = \cos(\theta) = \frac{A \cdot B}{|A| \cdot |B|} = \frac{\sum_{k=1}^{n} A_k \times B_k}{\sqrt{\sum_{k=1}^{n} A_i^2} \times \sqrt{\sum_{k=1}^{n} B_i^2}} \tag{3.14}$$

推荐可靠性和相似度分别以案件为单位评估了案例筛选系统的可用性，除此之外，还需要以单次推荐为单位，评估案例筛选系统的实用性。

通过对类案判决标准定轨，预警和报告类似案件判决偏离度，便于案件监督管理，规范和制约法官自由裁量权，如刑事案件量刑偏离度分析，运用峰度值对单次推荐中生成的类案进行偏差对比，能够保证法官在偏离度阈值范围内进行自由裁量，峰度的计算公式如下：

$$Kurtosis[X] = E[(\frac{X - \mu}{\sigma})^4)] = \frac{E[(X - \mu)^4]}{(E[X - \mu]^2)^2} \tag{3.15}$$

案例筛选系统在司法实践层面应具有正向效应，能够极大化模拟真实世界案件发生的情况。换言之，被推荐的案件应该满足概率统计学当中的正态分布规律，运用偏度值对单次推荐中生成的类案进行分布情况的分析，如刑期推荐正态性，能够辅助法官提高类似案例的运用能力。偏度的计算公式如下，其中 k_2, k_3 分别表示二阶和三阶中心矩：

$$Skewness(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{k_3}{\sigma^3} = \frac{k_3}{k_2^{3/2}} \tag{3.16}$$

人工智能辅助统一裁判尺度，基于案例筛选结果，对司法从业者而言可辅助其进行类案的判决参考，对于当事人而言有利于其选择诉前调解等多元纠纷解决方式进行息诉止讼。案件止讼率通过度量案例筛选列表标签与输入的原始案例标签的匹配比例来评估案例筛选结果的准确程度。

对于上述司法模型评估指标和案例筛选接口评估指标，系统均需对其进行归一化操作，方便对指标数值的统一展示，其中多数指标值域有上下限，采用离差标准化进行归一化处理；值域为 R 实数集的指标使用 $Sigmoid$ 函数进行归一化计算；部分指标值域为 $[0,+\infty)$，如精确率和公平性等计算过程中使用的 $Normalization$ 归一化函数如下所示，利用 $Sigmoid$ 非线性函数进行归一化映射。

$$\begin{aligned} Normalization(x) &= 2 \cdot Sigmoid(x) - 1 \\ Sigmoid(x) &= \frac{1}{1+e^{-x}} \end{aligned} \tag{3.17}$$

3.5　本章小结

本章首先对司法案例筛选测试系统进行总体规划，进一步分析系统的功能性和非功能性需求，并通过用例描述对核心功能进行了总结和归纳，然后对本系统进行架构设计、逻辑设计、开发设计、进程设计和部署设计，展示了 4+1 视图，将系统按功能分为权限管理模块、上传管理模块、测试管理模块和数据处理模块。最后对司法模型测试和案例筛选接口测试的指标设计进行了详细阐述。

第四章　司法案例筛选测试系统的详细设计与实现

本章主要介绍案例筛选测试系统的详细设计与实现。根据第三章中描述的需求分析和概要设计，本系统从逻辑上分为权限管理、上传管理、测试管理和数据处理四个模块，本将分别对这四个模块进行详细介绍。

4.1　权限管理模块详细设计与实现

4.1.1　权限管理模块的详细设计

权限管理模块主要实现对系统用户的认证及请求鉴权的工作，其详细设计类图如图4.1所示，基于实现 Spring Boot 提供的认证模块组件接口 UserDetails 和 UserDetailService，结合慕测主站开放的认证接口进行统一的用户状态管理，适配接入的用户认证操作，通过 WebSecurityConfig 将认证配置应用于服务中。对于系统用户请求利用 Zuul 网关进行拦截认证处理，确保接口使用的安全性。同时配置 AccessFilter 对用户请求可达性进行控制，确保接口转发的信息完整性。

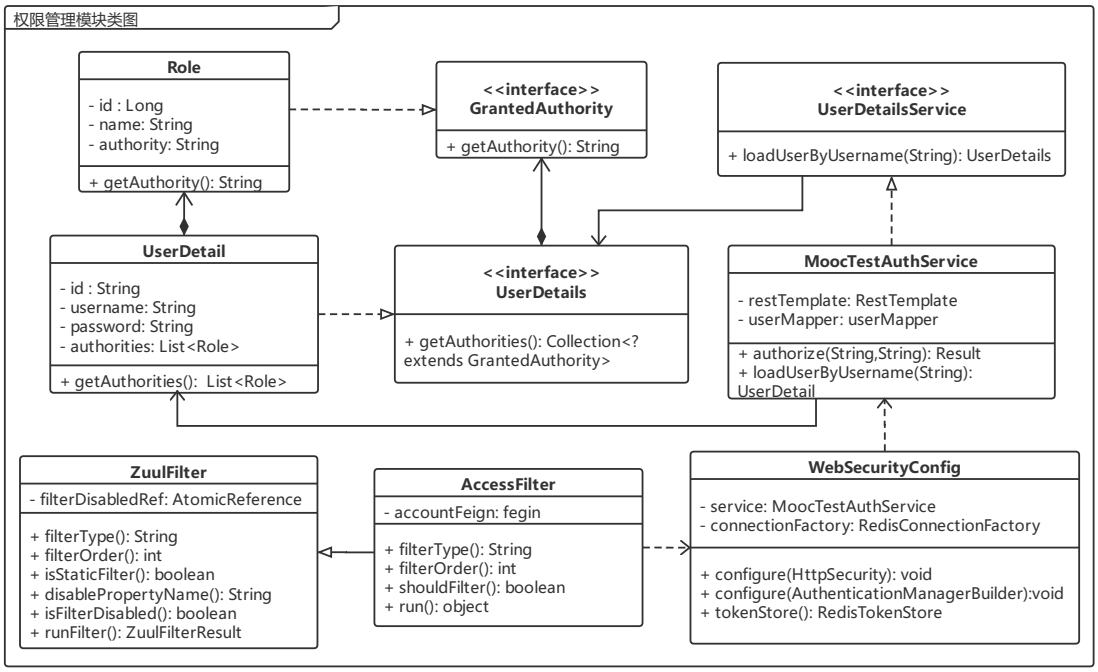

图 4.1: 权限管理模块类图

4.1.2　身份认证的实现

司法案例筛选测试系统用户权限由慕测主站管理，通过调用主站提供的接口进行登录、获取用户信息操作，首次通过慕测主站账号登录的用户系统会将其信息持久化到数据库中。本系统持有用户登录状态，以 access_token 形式维护在认证节点服务器的 redis 中，减轻服务网络请求和数据库请求频率，提高认证服务响应效率。身份认证关键代码如图4.2所示。

```
// 使用 redis 缓存登录状态
@Override
public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws
Exception {
    endpoints.tokenStore(tokenStore())
            .authenticationManager(authenticationManager);
 }
// 用户认证相关代码
@Override
public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
    List<User> userList = accountMapper.selectByName(username);
    UserDetail userDetail = new UserDetail();
    if (userList.size() == 0) {
        // 数据库中未保存当前用户状态信息
        ResponseEntity<UserDetail> entity = restTemplate.
                getForEntity(mooctestAuthURL, UserDetail.class);
        // 解析处理请求信息更新用户状态
        process(entity, userDetail);
    } else {
        // 获取持久化信息
        BeanUtils.copyProperties(userList.get(0), userDetail);
    }
    return userDetail;
}
```

图 4.2: 身份认证处理关键代码

4.1.3　统一网关的实现

系统用户发起的请求均通过网关微服务，该服务采用 Zuul2.0 编程模型，基于 Netty Client 处理后端业务服务 API 请求，支持异步高并发特性，降低请求延迟。本系统前端发起的所有请求均通过网关微服务，进行统一认证处理，并将通过认证后的用户信息装配到请求头中转发到对应微服务。网关认证配置和拦截转发关键代码如图4.3所示。

```
// AccessFilter
@Override
public Object run() throws ZuulException {
    // 获取上下文及 request 对象
    RequestContext currentContext = RequestContext.getCurrentContext();
    HttpServletRequest request = currentContext.getRequest();
    // 验证 token 时候 token 的参数 从请求头获取
    String token = request.getHeader(Strings.ACCESS_TOKEN_KEY);
    String requestURL = request.getRequestURL().toString();
    if (StringUtils.isEmpty(token)) {
        token = request.getParameter(Strings.ACCESS_TOKEN_KEY);
    }
    if (!StringUtils.isEmpty(token)) {
        // 添加用户信息
        Result result = accountFeign.user(token);
        JSONObject jsonObject = JsonUtil.parseObject(result.getData().toString());
        if (Objects.nonNull(jsonObject) && jsonObject.containsKey(Strings.ID_KEY)) {
            currentContext.addZuulRequestHeader(Strings.USER_ID_KEY,
jsonObject.getString(Strings.ID_KEY));
        }
    }
    return null;
}
```

图 4.3: 统一网关处理关键代码

4.2　上传管理模块详细设计与实现

4.2.1　上传管理模块的详细设计

上传管理模块主要实现用户对司法模型文件和数据集上传、查看、删除、下载等操作，主要分为文件上传和上传管理两部分，核心类图设计如图4.4所示。UploadController 负责处理对用户上传和下载模型或测试集相关文件的请求，依赖于业务逻辑层的 UploadService，对于上传的文件通过 ProcessService 异步调用数据处理微服务 AnalysisFeign，对模型或数据集文件进行解析、校验、存储，仅当通过数据处理服务校验的文件才能被测试用户用于测试评估阶段。ManageController 处理对上传列表和详情信息获取的请求，依赖于 ManageService 进行如条件查询，数据组装的操作。FileResMapper 用于对文件上传时相关信息进行持久化处理，ModelFileMapper 和 DatasetMapper 分别表示对上传的司法模型信息和数据集信息进行相关数据库操作。

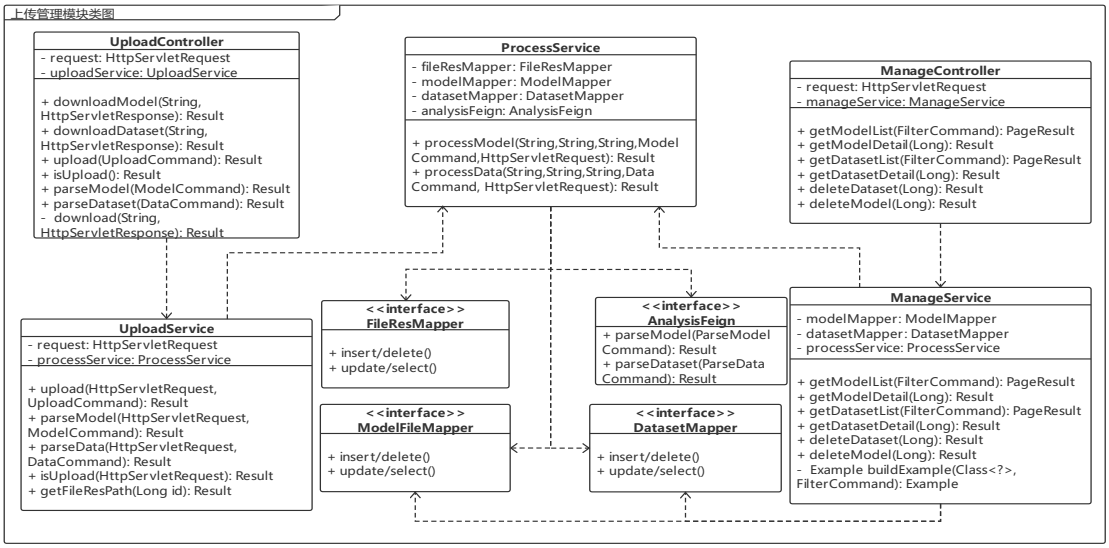

图 4.4: 上传管理模块类图

4.2.2　文件上传的实现

本系统模型上传和数据集上传操作均需进行文件上传，模型和数据文件一般都是几十兆、几百兆甚至 GB 级别的大小，若采用传统的文件上传实现，一方面容易超出后台设置的单个文件上传大小限制，另一方面上传时间过长可能超出服务器响应时间上限导致服务器异常。为解决上述问题，本系统采用对大文件分片处理的方式，将对单个文件的上传请求分为对多个分片的上传请求，对于后端来说，既能避免超出文件大小或响应时间阈值而造成的系统异常，保障上传效率，也能支持切片粒度的断点续传，提高服务稳定性。前端基于分片上传数量占分片总数的比重展示文件上传进度，避免大文件同步请求时间长无反馈的现象，增强用户对文件上传功能的交互体验。

为避免重复文件上传导致带宽和磁盘占用，文件上传前需通过文件的 md5 值在后台进行重复文件校验，若存在对应文件上传记录，系统直接选定对应文件。前端利用 spark-md5.js 库对大文件进行分片，逐个传入 spark.appendBinary() 方法中来计算每个分片的 md5 值，最后通过 spark.end() 方法输出结果，适用于对本地大文件进行 md5 值计算，且不容易出错。上传文件前首先要校验数据库中是否存在对应 md5 标识的上传记录，如代码中所示：若未查询到上传记录或上传状态为未完毕，则需发起上传请求，从对应的分片断点开始上传；若存在上传记录且上传状态已完成，则直接返回上传文件 id 供前端进行上传选定。本系统默认分片大小为 1MB，前端设置长度为 5 的队列对分片文件进行同步传输。

```
// 前端计算上传文件的 md5 标识
totalFileReader.onload = function (e) {
    // 计算整个文件的 fileMd5
    spark.append(e.target.result)
    params.file.fileMd5 = spark.end()
}
// 后端检查文件是否上传
String md5 = request.getParameter("md5");
Example example = new Example(FileRes.class);
Example.Criteria criteria = example.createCriteria();
criteria.andEqualTo("md5", md5);
// 根据 md5 获取上传记录
List<FileRes> list = fileResMapper.selectByExample(example);
String strDate = new DateTime().toString(Strings.DATE_TIME_FORMAT);
if (null == list || list.size() == 0) {
    //没有上传过文件
    dto = FileResDto.generate(1, md5,strDate);
} else {
    FileRes fileRes = list.get(0);
    if (fileRes.getStatus() == 0) {
        //文件上传部分
        dto = FileResDto.generate(2, fileRes.getUuid(), strDate);
        dto.setId(fileRes.getId());
    } else if (fileRes.getStatus() == 1) {
        //文件上传成功，已存在
        dto = FileResDto.generate(3, fileRes.getUuid(), strDate);
        dto.setFilePath(fileRes.getPath());
        dto.setId(fileRes.getId());
    }
}
```

图 4.5: 上传检查关键代码

前端中对文件分片实现的关键代码如图4.6所示，使用 HTML5 File API 技术进行文件分片上传操作，按照浏览器类型进行兼容性处理，获取对应的文件分片 slice 方法，并进一步设置分片大小、计算分片数量等信息。当触发上传事件时，前端将上传文件每个分片的编号、起始位置、结束位置、对应大小等信息进行整合，设置到请求参数 params.chunks 中发起上传切片请求。当确认上传时，遍历请求参数中的分片信息，此时才基于起止位置对文件进行切片操作，执行上传请求。此外，前端通过设置事件监听，支持上传操作的取消操作，中止继续上传分片文件，灵活支持各种断点续传的情形。

```
// 兼容性的处理
let blobSlice = File.prototype.slice || File.prototype.mozSlice ||
File.prototype.webkitSlice, chunkSize = 1024 * 1024 * 1, //分片大为 1M
chunks = Math.ceil(file.size / chunkSize), // 计算切片数量
currentChunk = 0, // 当前上传的 chunk
chunkFileReader = new FileReader(), // 用于处理文件切片信息
totalFileReader = new FileReader(); //用于计算出文件的 md5 标识
chunkFileReader.onload = function (e) {
  // 每一个分片需要包含的信息
  let obj = {   chunk: currentChunk + 1,
                // 计算分片的起始位置
                start: currentChunk * chunkSize,
                // 计算分片的结束位置
                end: ((currentChunk * chunkSize + chunkSize) >= file.size) ?
                file.size : currentChunk * chunkSize + chunkSize }
  // 每一次分片 onload,currentChunk 增加
  currentChunk++;
  params.chunks.push(obj)
}
//遍历分片，发送上传请求
for (let i = 0; i < uploadParams.file.fileChunks; i++) {
  judicialservice.upload(
 // 根据 start 和 end 进行文件切片处理
    file.slice(uploadParams.chunks[i].start,uploadParams.chunks[i].end),
    file.uid,uploadParams.file.fileMd5,uploadParams.file.fileName,
    uploadParams.file.fileSize,uploadParams.chunks[i].chunk,
    uploadParams.chunks[i].chunkMd5).then(response => {...}
  }
}
```

图 4.6: 文件分片关键代码

本系统文件上传支持分片粒度的断点续传，分片发起上传请求时，首先设置 action 操作字段为 check，后台对该分片文件进行存在校验，文件系统若不存在该分片数据，则发起 action 为 upload 的上传请求，将文件分片数据上传存储到服务器上。若文件分片数据存在，则继续检查下一个分片是否存在即可，无需占用网络资源对重复切片进行上传。进行分片上传保存时，代码逻辑如图4.7所示，若分片为上传文件的第一个分片，需将上传记录持久化到数据库中，同时标识对应上传记录状态为正在进行。当系统上传完最后一个分片文件时，需要对所有分片文件进行合并操作，并将对应上传记录状态更改为上传成功，最后进行文件流关闭及切片文件删除处理，避免造成文件系统空间资源的浪费。

```
// check action 请求
if (Strings.CHECK_ACTION.equals(action)) {
    String md5Str = FileUtil.getFileMD5(file);
    if (md5Str != null && md5Str.equals(md5)) {
        return FileResDto.generate(2, uuid);   //分片已上传过
    } else {
        return FileResDto.generate(1, uuid); //分片未上传
    }
// upload action  请求
} else if (Strings.UPLOAD_ACTION.equals(action)) {
    //分片上传过程中出错,有残余时需删除分块后,重新上传
    if (file.exists()) { file.delete(); }
    multipartFile.transferTo(new File(saveDirectory, uuid + "_" + index));
}
//  判断是否上传完毕所有分片，进行合并
if (path.isDirectory()) {
    File[] fileArray = path.listFiles();
    if (index == 1) {
      fileResMapper.insert(fileRes); //文件第一个分片上传时记录到数据库
    }
    if (fileArray.length == total) {
      //分块全部上传完毕,合并
      for (int i = 0; i < total; i++) {
          temp = new FileInputStream(new File(saveDirectory, uuid + "_" + i+1));
          while ((len = temp.read(byt)) != -1) { outputStream.write(byt, 0, len); }
       }
       temp.close(); outputStream.close();//关闭流
       //修改 FileRes 记录为上传成功
       fileResMapper.updateByExampleSelective(fileRes, example);
       String exclude = uuid + "." + suffix;
       FileUtil.delete(saveDirectory, exclude); //上传成功后，删除中间的切片文件
}
```

图 4.7: 分片存储关键代码

4.2.3　上传记录管理的实现

文件上传成功后，后台异步执行对应模型或数据集解析处理操作，确保文件格式、维度等信息的准确性，用户可通过上传记录管理进行上传记录的查看。上传记录管理主要实现对用户上传的模型记录、数据集记录进行条件查询、详情查看、下载、删除等操作，为避免误删，本系统对所有删除操作均为逻辑删除，即对数据表中 is_delete 字段进行更新。如图4.8所示，以模型管理为例，用

户上传模型后进入模型管理界面即可查看上传模型记录列表信息，且可根据上传时间范围、文件处理状态、是否公开、上传记录名称以及标签进行记录筛选查找，对一条记录可进行查看详情、删除操作，当上传处理状态为成功完成时，用户可下载对应的模型文件或数据集文件到本地。

图 4.8: 上传记录管理界面示意图

上传记录管理关键代码如图4.9所示，以模型管理为例，后端通过解析筛选条件对上传记录进行分页查询，利用 PageHelper 插件设置分页参数，根据请求筛选条件 FilterCommand 的字段构建对应条件模版 Example 对象，进行模型上传记录列表的条件查询，其中名称和标签等条件字段是通过模糊匹配来筛选的。获取列表数据后，可根据模型记录的 id 进行详情查看或删除操作。对处理完毕的上传记录，前端可进行对应模型文件的下载，后台通过查询对应文件的相对存储路径，获取文件系统上的文件资源进行下载。

系统网关模块的请求转发和通用日志模块的记录请求日志的实现均涉及对 HttpServletResponse 输出流对象的调用处理。如图4.9所示，为成功实现下载功能，需要对下载请求进行过滤处理，避免因对输出流对象的重复获取，导致下载操作出现输出流被调用的异常。

```
// 查询上传模型记录列表
PageHelper.startPage(command.getPageNum(), command.getPageSize());
Example example = buildExample(ModelFile.class, command);
List<ModelFile> list = modelMapper.selectByExample(example);
PageInfo page = new PageInfo(list);
PageResult result = PageResult.success();
result.setCurrentPage(command.getPageNum()).setSize(command.getPageSize()).set
TotalNum(page.getTotal()).setTotalPage(page.getPages());
// 详情
ModelFile model = modelMapper.selectByPrimaryKey(id);
// 删除
model.setIsDeleted(true);
modelMapper.updateByPrimaryKeySelective(model);
// 下载
FileInputStream fis = = new FileInputStream(file);
BufferedInputStream bis = new BufferedInputStream(fis);
 OutputStream outputStream = response.getOutputStream();
int i = bis.read(buffer);
while(i!=-1){outputStream.write(buffer, 0, i); i=bis.read(buffer);}
// 日志 AOP 配置过滤 download 请求
@Pointcut("execution(public * cn.edu.nju.software.judicialtest.controller..*.*(..))
&& !execution(public * cn.edu.nju.software.judicialtest.controller..*.*download*(..))")
public void webLog() {}
```

图 4.9: 上传记录管理关键代码

4.3　测试管理模块详细设计与实现

4.3.1　测试管理模块的详细设计

测试管理模块主要实现对司法模型测试任务和案例筛选接口测试任务管理、测试报告查看功能，分为测试任务管理和接口调用两个部分，测试任务管理为该模块核心功能，进行测试任务新建、列表查询、详情查看等操作。测试任务包括司法模型测试，用户选定上传的司法模型、数据集及相关配置信息，新建模型测试任务；案例筛选接口测试，用户配置接口请求地址、参数信息、解析模版及自定义指标配置信息，新建案例筛选接口任务。测试任务发送到后台建立后便开始异步执行测试计算指标结果，在测试历史列表可查看测试进度状态、执行完毕的测试报告。接口调用为案例筛选接口测试任务的执行提供测试数据获取解析服务，根据指定请求地址和参数信息构建请求模版，根据解析模版解析构建类案列表信息，用于数据处理服务执行案例筛选接口测试的结果计算。

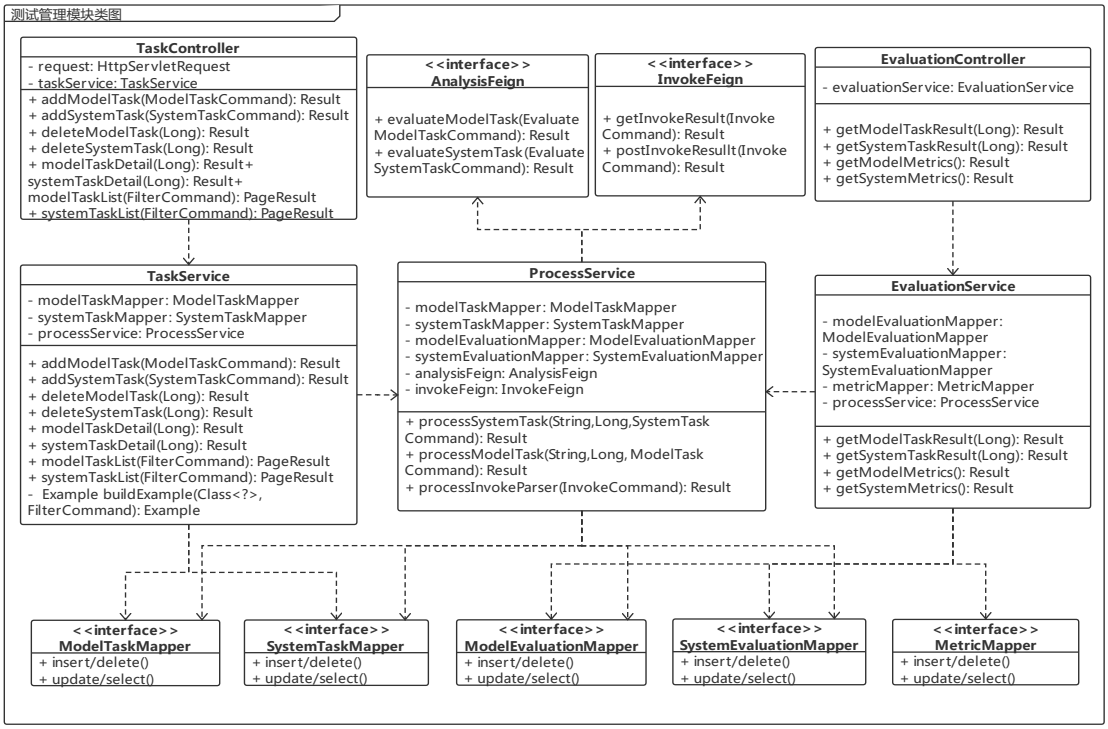

图 4.10: 测试管理模块类图

该模块核心类图如图4.10所示，TaskController 接受测试任务新建、查询、删除等请求，依赖 TaskService 进行业务逻辑处理，EvaluationController 供用户获取测试任务结果信息，用于测试报告展示。ProcessService 在测试管理微服务中进行测试任务执行处理，依赖 AnalysisFeign 调用数据处理服务进行指标计算，依赖 InvokeFeign 进行接口调用解析获取测试数据。

4.3.2　测试任务管理的实现

本系统包括司法模型测试和案例筛选接口测试两类测试任务。司法模型测试，用户指定上传的司法模型和数据集记录信息，加载模型对数据集预测，获取多分类概率输出数据，进而进行模型评估指标计算；案例筛选接口测试，通过用户输入的请求 URL 和参数模版进行网络请求，获取接口反馈信息，利用解析模版对反馈信息分解重构为系统可用格式，基于指定案例描述信息进行案例推荐接口指标计算。测试任务执行相关功能异步执行，支持用户离线运行，该部分具体实现在数据处理模块会进行详细描述。测试任务管理主要实现对测试任务的新增、查询、删除、报告查看等功能。

```
// 新建司法模型测试任务
modelTaskMapper.insert(modelTask);
// 异步执行司法模型测试任务
processService.processModelTask(userId, modelTask.getId(), command);
// 详情
ModelTask modelTask = modelTaskMapper.selectByPrimaryKey(id);
// 删除
modelTask.setIsDeleted(true);
modelTaskMapper.updateByPrimaryKeySelective(modelTask);
// 查询司法模型测试任务列表
PageHelper.startPage(command.getPageNum(), command.getPageSize());
Example example = buildExample(ModelTask.class, command);
List<ModelTask> list = modelTaskMapper.selectByExample(example);
PageInfo page = new PageInfo(list);
PageResult result = PageResult.success();
result.setCurrentPage(command.getPageNum()).setSize(command.getPageSize()).set
TotalNum(page.getTotal()).setTotalPage(page.getPages());
```

图 4.11: 测试任务管理关键代码

如图4.11所示，以司法模型测试任务为例，后端对应测试任务实体 Mapper 类对司法模型测试任务记录进行新增、删除、详情，同上传记录管理，查询任务列表也通过构造筛选条件 Example 对象进行分页查询。测试任务新建后即调用 ProcessService 的异步处理方法，执行测试任务计算测试结果，更新测试执行状态及对应测试结果信息。测试任务记录管理界面如图4.12所示，支持通过时间范围、任务执行状态、任务名称和标签信息对测试任务进行列表查询。

图 4.12: 测试任务记录管理界面示意图

```
// 后端获取测试任务结果 JSON 详情数据 通用
String content = FileUtil.getFileContent(FileUtil.join(taskResultDir,
taskEvaluation.getDetailJsonPath()));
// 前端遍历指标列表，渲染每个指标值及其样式
metrics_data.map((item, index) => {
let list=[item.min, item.standard_a, item.standard_b,
item.standard_c, item.standard_d, item.max];
let data = this.setValue(result,item.name_en)
return (<Descriptions.Item label={item.name}>
  <div style={{display: 'flex', justifyContent: 'center', alignItems: 'center'}}>
  <div style={{flex: 1}}><Progress type="circle" format={percent => `${percent /
  100}`}/></div><div style={{flex: 2}}><MyStep list={list} data={data} /></div></div>
  </Descriptions.Item>)})
```

图 4.13: 测试结果展示关键代码

测试任务执行成功后，可通过测试任务列表查看报告按钮获取测试报告信息进行展示。如图4.13所示，后端获取到对应测试任务结果实体信息后，通过文件工具类获取实体表中 detail_json 字段对应结果文件内容进行反馈。前端对结果 json 进行解析，对获取的模型或接口指标列表进行遍历操作，利用指标英文名称字段作为 key 值检索结果 json 数据中对应指标值，进行渲染。

图 4.14: 司法模型测试报告示意图

图4.14和图4.15分别展示了司法模型测试报告和案例筛选接口测试报告示意图，司法模型测试报告主要展示模型测评体系中的基础评估指标、扩展评估指标共 12 个以及综合评定。案例筛选接口测试报告包括接口反馈案例列表信息，且每条筛选案例展开后向用户提供相似度度量和可靠性指标，对于列表整体进行预测偏离度、推荐正态性、案件止讼率指标展示，最后报告进行综合评定。综合评定的计算以配置文件形式配置到数据处理模块，定义了每个指标权重信息。示意图仅展示报告最关键部分，其中司法模型测试报告中还包括对应司法模型和数据集基本信息，案例筛选接口测试报告也包括输入案例详情信息。此外，前端利用 html 转 pdf 插件，可直接将测试报告转为 pdf 进行本地存储，此功能通过测试报告中下载报告按钮触发。

图 4.15: 案例筛选接口测试报告示意图

4.3.3　测试接口调用的实现

案例筛选接口测试任务执行前，需进行数据准备工作，根据请求 URL、参数、类型等信息进行待测接口请求获取反馈数据，通过反馈解析模版，将接口反馈数据中案例列表关键信息提取重构，为后续评估提供统一格式数据，方便进行数据处理。

```
// 发送 POST 请求，其它类型请求构造类似
JSONObject json = RestUtil.create() //创建一个实例
   //设置 Content-Type，默认为 application/json
   .setContentType(command.getContentType())
   .addHeaders(command.getHeader()) //添加 header
   .addBody(command.getRequestBody()) //添加 body
   .postForObj(command.getUrl(), JSONObject.class); //发送 post 请求
List<CaseDto> dtos = parseResponse(json, command.getResponseTemplate());

// 根据解析模版解析反馈数据构造案例列表数据 List<CaseDto>
List<CaseDto> caseDtos = Lists.newArrayList();
JSONObject tpl= JsonUtil.parseObject(responseTemplate);
// 获取案例列表源数据
List<JSONObject>cases=JSONArray.parseArray(tpl.getString(Strings.CASE_LIST_KE
Y), JSONObject.class);
// 解析案例列表源数据，构造 CaseDto 对象
cases.forEach(obj -> {
CaseDto dto = new CaseDto();
 // 解析设置案例描述、刑期、法条、罪名、罚金信息
 dto.setFact(JsonUtil.getValue(obj, tpl.getString(Strings.FACT_KEY)));
 dto.setImprison(JsonUtil.getValue(obj,tpl.getString(Strings.IMPRISONMENT_KEY)));
 dto.setRelativeArticles(JsonUtil.getValue(obj,tpl.getString(Strings.ARTICLE_KEY)));
 dto.setAccusation(JsonUtil.getValue(obj,tpl.getString(Strings.ACCUSATION_KEY)));
 dto.setMoney(JsonUtil.getValue(obj, tpl.getString(Strings.MONEY_KEY)));
 caseDtos.add(dto); });
```

图 4.16: 测试接口调用关键代码

图4.16为测试接口调用关键代码，以 POST 请求为例，通过工具类构建 REST 请求对象，设置 content-type、header、body 信息，发送请求获取接口反馈 JSONObject 对象数据，然后进行案例列表信息解析工作，其他类型请求构造步骤类似，故不在代码片段中赘述。解析反馈数据时，利用 responseTemplate 模版信息，首先提取案例描述列表数据，进而遍历列表提取每个元素的案例描述、刑期、法条、罪名、罚金信息构建案例对象数据，用于案例筛选接口测试评估。

4.4　数据处理模块详细设计与实现

4.4.1　数据处理模块的详细设计

数据处理模块主要负责对上传的司法模型和数据集文件处理和对测试任务的评估工作，该模块对应架构中的 Analysis 微服务，通过 Sidecar 插件融合 Flask 框架服务，核心功能均由 Python 语言实现。该模块核心类图如图4.17所示，ModelProcessor 和 DatasetProcessor 持有用于模型和数据集文件处理操作的基本信息，依赖 ModelParser 和 DataParser 进行文件解析、校验工作，依赖 ModelSaver 和 DataSaver 进行校验成功后进行文件转存操作。ModelEvaluate 和 SystemEvaluate 分别负责执行司法模型评估任务和案例筛选接口评估任务，利用 MetricUtil 工具类计算相关评估指标。数据处理模块本身不涉及数据库持久化操作，通过与文件管理模块和测试管理模块交互实现对数据表更新操作，具体实现在对应模块均有体现，因此本部分不涉及数据表设计。

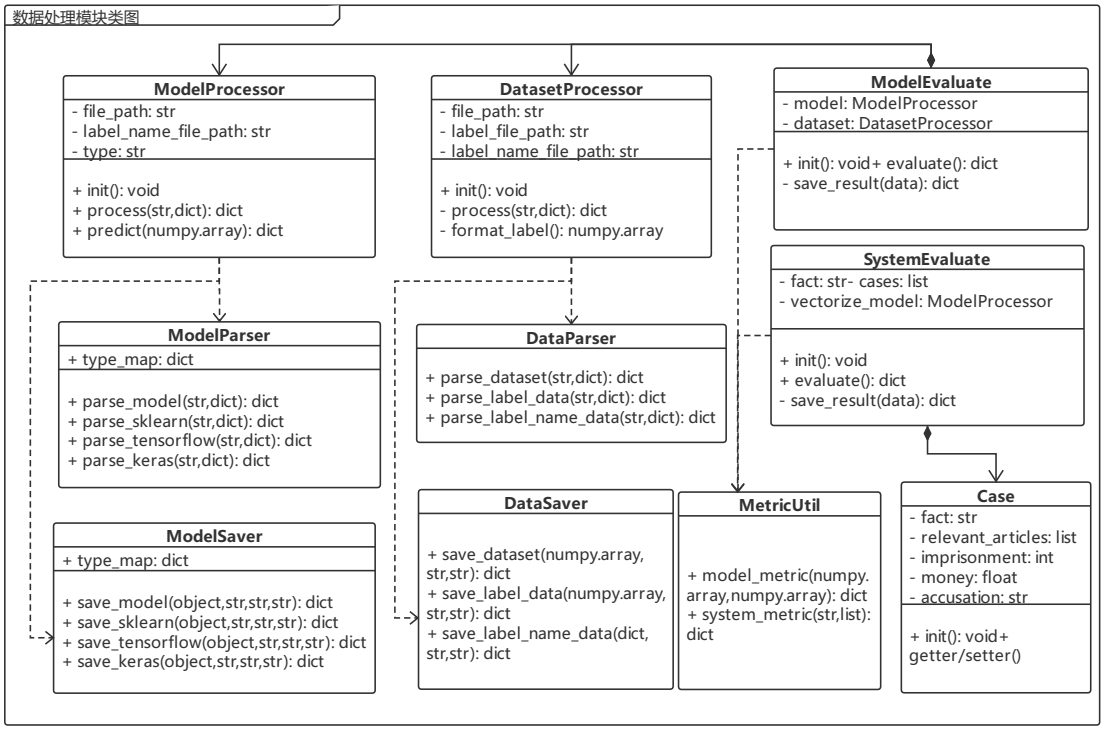

图 4.17: 数据处理模块类图

4.4.2　上传文件处理

上传文件处理部分涉及到司法模型文件、数据文件，其中模型文件包含 Keras、Tensorflow 和 Sklearn 三种类型，数据文件则分为样本数据文件、标签

数据文件和标签含义数据文件。如图4.18所示，type_map 中提供模型类型与解析函数映射关系，模型解析基于对应模型库函数进行加载，相关代码均为最核心代码，异常捕获等代码均不予展示。模型加载完毕后需进行输入输出维度校验，系统构建与输入维度对应的单个样本数据进行模型预测，将预测结果维度与输出维度信息进行校验匹配，确保模型类型和维度信息准确性。

```
# 根据模型类型检测、加载、校验
type_map = {"tensorflow": detect_tensorflow, "keras": detect_keras,
"sklearn": detect_sklearn}
# 解析 tensorflow 模型
import tensorflow as tf
model = tf.train.import_meta_graph(os.path.join(model_file_dir, meta_file))
model.restore(sess, tf.train.latest_checkpoint(model_file_dir))
# 解析 keras 模型
from keras.models import load_model
model = load_model(model_file)
# 解析 sklearn 模型
import pickle
from sklearn.externals import joblib
model = pickle.load(file)
if not model: model = joblib.load(model_file)
# 模型输入输出维度校验
input_dims = attrs[string_util.input_dim_key]
output_dims = attrs[string_util.output_dim_key]
# 检查模型的输入维度是否准确
sample_dim = list(input_dims)
sample_dim[0] = 1
# 生成输入维度匹配的随机样本
x_sample = np.zeros(shape=sample_dim)
# 预测输出
y = model.predict(x_sample)
# 校验输出维度是否匹配
state = dim_util.check_dims_match(np.shape(y), output_dims)
```

图 4.18: 模型文件解析校验关键代码

数据文件解析关键代码如图4.19所示，系统支持对 csv 和 npy 格式数据集文件解析，统一处理为 numpy 的 npy 格式进行校验，对于样本数据文件，检查其数据维度与 input_dim 参数是否匹配，而 output_dim 则用于对标签数据文件做维度维度校验。系统支持 json 和 npy 格式的标签含义文件解析，统一处理为键值对的字典格式，通过标签数据直接获取到对应含义信息。

```
# 解析数据集数据(样本数据 / 标签数据)，支持 csv 和 npy 类型
data = data_parser.parse_dataset(data_path, type)
label_data = data_parser.parse_label_data(label_data_path, type)
# 检查数据集维度是否匹配，样本数据维度 input_dim / 标签数据维度 output_dim
state = dim_util.check_dims_match(data.shape, attrs[string_util.input_dim_key])
state = dim_util.check_dims_match(data.shape, attrs[string_util.output_dim_key])
# 解析标签含义文件，支持 npy 和 json 格式，最终整理为 json 格式
if extension == string_util.extension_npy_key:
    label_name_data = np.load(label_name_data_path)
    label_name_type = string_util.label_name_classification
    result_data[string_util.label_name_type_key] = label_name_type
    for i in range(len(label_name_data)):
        # 配置 json 标签数值：标签名称的 key-value 对
        result_data[string_util.label_name_data_key][str(i)] = label_name_data[i]
elif extension == string_util.extension_json_key:
    with open(label_name_data_path, 'rb') as data:
        result_data = json.load(data)
```

图 4.19: 数据文件解析校验关键代码

文件解析校验成功后，需进行转存操作，对每种类型文件采用统一格式存储，再次加载时无需进行校验处理，提高测试评估效率。如图4.20所示，根据模型类型直接在 type_map 中获取对应存储方法。数据集相关的样本数据和标签数据均以 npy 格式存储，标签含义文件则统一存储为 json 格式。

```
# 根据模型类型选择存储方法
type_map = { "tensorflow": save_tensorflow_model, "keras": save_keras_model,
"sklearn": save_sklearn_model}
# tensorflow 模型存储
file_util.copy_files(model_dir, os.path.join(base_save_path, save_path))
# keras 模型存储
model.save(absolute_file_path)
# sklearn 模型存储
joblib.dump(model, absolute_file_path)
# 数据集样本数据文件存储
np.save(os.path.join(base_save_path, file_path), data)
# 数据集标签数据文件存储
np.save(os.path.join(base_save_path, file_path), data)
# 标签含义文件存储
with open(os.path.join(base_save_path, label_name_file_path), 'w', encoding='utf-8')
as json_file:
    json.dump(json_data, json_file, ensure_ascii=False)
```

图 4.20: 文件存储关键代码

4.4.3　测试任务评估

测试任务评估主要负责司法模型测试和案例筛选接口测试相关指标计算，指标体系在第三章中有具体描述，其中司法模型测试包含基础评估指标和扩展评估指标两大类共 12 个度量指标，案例筛选接口测试指标则包括对单个推荐案例信息评估以及对接口推荐案例列表整体评估的不同粒度的指标。如图4.21所示，展示了测试任务指标计算的关键代码。

```
# 司法模型测试，基于标签结果 t 和模型预测结果 y，计算评估指标
data=[auccuracy(t,y), amp_score(t,y), precision_score(t,y,average="macro"),
precision_score(t,y,average="micro"), 1 - median_absolute_error(t,y), auc_score(t, y),
1/hamming_loss(t,y), explained_variance_score(t,y), eval_fairness(t,y),
cohen_kappa_score(t, y), jaccard_similarity_score(t,y), matthews_corrcoef(t, y)]
# 以公平性为例，公平性偏差
fair_deviation = ((ampscore1 - ampscore0) ** 2 + (ampscore2 - ampscore0) ** 2) / 2
# 公平性归一化
normal_fair = (nfsum / (2 * (max(ampscores) - min(ampscores)))) * 100
# 最终公平性指标结果
fair_score = (-math.log(ampscore0, 10)) * normal_fair

# 案例筛选接口测试，基于输入案例和推荐案例列表，计算评估指标
corpus = [dictionary.doc2bow(doc) for doc in all_doc_list]
doc_test_vec = dictionary.doc2bow(doc_test_list)
# 文本向量化操作
tfidf = models.TfidfModel(corpus)
# 相似度计算
index = similarities.SparseMatrixSimilarity(tfidf[corpus],
num_features=len(dictionary.keys()))
sim = index[tfidf[doc_test_vec]]
edit_dist=[edit_distance(base_case,test_cast) for test_case in case_list]
# 偏离度、正态性、止讼率计算
calculate_deviation(base_case, case_list)
calculate_diversary(base_case, case_list)
calculate_accuracy(base_case, case_list)
```

图 4.21: 测试任务评估关键代码

司法模型测试指标基于测试数据集真实标签和模型预测标签计算，图中以公平性计算为例，其实现基于公式，计算出对应数值后需进行归一化处理。案例筛选接口测试指标基于输入的案例描述以及接口反馈的案例描述列表计算，对文本分词后进行向量化操作，计算相似度指标。遍历案例列表与输入案例信息

度量编辑距离来评估其可靠性。案例筛选接口整体推荐性能从案件刑期、罚金、法条、罪名方面来进行司法偏离度、推荐正态性和案件止讼率的计算。

4.5　本章小结

本章对系统的权限认证模块、上传管理模块、测试管理模块和数据处理模的详细设计与具体实现进行详细阐述，通过核心类图来描述对应模块的设计细节，通过数据表字段设计表来阐述系统持久化层操作的数据结构，通过关键代码和部分页面截图配合说明对应功能的具体实现。

第五章　司法案例筛选测试系统的测试与分析

本章主要基于第三章需求分析以及第四章详细设计和实现对司法案例筛选测试系统进行系统测试，包括测试准备和功能性测试。

5.1　测试准备

5.1.1　测试目标

基于测试环境对司法案例筛选测试系统的有效性进行测试，设计功能测试用例，并执行，检验系统核心功能模块的实现是否可以满足系统对应需求，包括司法模型上传、数据集上传、上传管理、新建司法模型测试任务、新建案例筛选接口测试任务、测试历史查看、测试报告查看及下载等功能，通过测试来保障案例筛选测试系系统的质量。

5.1.2　测试环境

如表5.1所示为系统功能测试环境配置信息，本系统部署在实验室环境下的物理服务器，装有两个 12GB 显存的英伟达显卡为深度学习模型的加载和运行提供加速服务。后端微服务均通过 maven 构建打包部署，数据处理服务 Flask 项目通过 Nginx 的反向代理与 Analysis 微服务交互利用 Sidecar 插件注入 Spring Cloud 体系中，前端 React 项目打包后通过 Nginx 配置部署。为支持物理服务器的公网访问，利用腾讯云服务器对前端进行端口映射，可通过云服务器公网 IP 进行本系统访问，在此条件下进行系统测试。

表 5.1: 系统测试环境

字段	测试环境
服务器	物理服务器 64GB 内存 NVIDIA GeForce GTX 1080Ti * 2
操作系统	Ubuntu 16.04
数据库	Mysql 5.7.29、Redis 5.0.3
Python 库	Python 3.6.2、Flask 1.1.1、Scikit-Learn 0.22.2 Tensorflow-gpu 1.10.0、Keras 2.2.4
测试软件	chrome 浏览器 80.0
其他软件	Nginx 1.15.7、Maven 3.3.9、NodeJS 12.14.1

5.2　功能性测试

系统的功能性测试基于需求分析中用例描述的功能来设计，测试用例由给定输入或操作和期望输出或展示结果组成，通过对比测试用例的期望结果与系统的实际结果，来验证系统功能的正确性，进而对系统中可能存在的缺陷进行发现和修复，以保证系统功能符合预期。本部分主要从文件上传、上传管理、新建测试任务、测试任务执行、测试任务管理、测试报告查看这六个方面展开功能测试，其中上传文件包括司法模型上传和数据集上传，测试任务包含司法模型测试和案例筛选接口测试两种类型，测试报告查看需覆盖到测试报告下载的测试，能充分覆盖用例描述对应的功能点。

表 5.2: 文件上传功能测试用例

测试项	操作/输入	预期结果	测试结果
上传模型	1. 系统用户登录 2. 点击侧边栏或用户下拉框中上传模型按钮 3. 点击模型上传按钮，选择本地模型文件上传 4. 点击模型上传按钮，重新选择相同文件上传 5. 完善模型上传界面表单信息，点击确认按钮	1. 登录成功 2. 进入上传模型界面 3. 上传进度 100% 后，界面提示上传成功 4. 系统提示文件库中已存在该文件，已默认选中 5. 系统提示模型新建成功，并跳转到上传管理界面	通过
上传数据集	1. 点击侧边栏或用户下拉框中上传数据集按钮 2. 点击数据上传按钮选择本地数据文件上传 3. 在上传进度为 100% 前点击取消上传按钮 4. 点击数据上传按钮，重新选择相同文件上传 5. 完善数据集上传界面表单信息，点击确认按钮	1. 跳转到上传数据集界面 2. 系统同步显示上传进度百分比，等待上传结束 3. 上传取消，等待用户操作 4. 上传进度基于之前进度继续上传，完成后提示成功 5. 系统提示数据集新建成功，并跳转到上传管理界面	通过

表5.2描述了文件上传功能的测试用例，分别进行司法模型和数据集上传功能测试，其中覆盖对同一文件进行重复上传和文件断点续传的功能测试，实际测试结果符合预期，对应测试用例全部通过。

表 5.3: 上传管理功能测试用例

测试项	操作/输入	预期结果	测试结果
上传模型记录查询	1. 点击侧边栏或用户下拉框上传管理按钮 2. 设置查询条件后，点击确认按钮	1. 跳转到上传管理界面，默认显示模型管理页面 2. 页面展示满足筛选条件的模型上传记录	通过
上传模型记录详情查看	点击一条模型记录中的详情按钮	页面弹出模型记录详情模态框，展示模型详情	通过
上传模型记录删除	点击一条模型记录中的删除按钮	界面对应记录删除，系统提示删除成功	通过
下载模型文件	在模型详情模态框中点击下载按钮	弹出空白页建立文件获取请求，进行文件下载	通过
上传数据集记录查询	1. 点击上传管理界面测试管理按钮 2. 设置查询条件后，点击确认按钮	1. 显示数据集管理页面 2. 页面展示满足筛选条件的数据集上传记录	通过
上传数据集记录详情查看	点击一条数据集记录中的详情按钮	页面弹出数据集记录详情模态框，展示模型详情	通过
上传数据集记录删除	点击一条数据集记录中的删除按钮	界面对应记录删除，系统提示删除成功	通过
下载数据集文件	在数据集详情模态框中点击下载按钮	弹出空白页建立文件获取请求，进行文件下载	通过

表5.3是上传管理功能的测试用例，主要验证司法模型和测试集上传记录的条件查询、详情查看、删除记录以及对应文件下载功能的正确性，实际测试结果符合预期，对应测试用例全部通过。

表 5.4: 新建测试任务功能测试用例

测试项	操作/输入	预期结果	测试结果
新建司法模型测试任务	1. 选择导航栏中司法测试选项，点击司法模型测试按钮	1. 跳转到新建司法模型测试任务页面	通过

新建司法模型测试任务	2. 点击模型输入框 3. 从司法模型列表中选择待测模型 4. 点击数据集输入框 5. 从司法数据集列表中选择待测数据集 6. 填写名称等基本信息，点击确认按钮	2. 弹出模型上传记录列表信息供用户选择 3. 系统提示选择成功，输入框中显示模型名称 4. 弹出数据集上传记录列表供用户选择 5. 系统提示选择成功，输入框中显示数据集名称 6. 提示新建成功，进入测试历史界面	通过
新建案例筛选接口测试任务	1. 选择导航栏中司法测试选项，点击案例筛选接口测试按钮 2. 输入测试任务相关信息，如接口地址、解析模版，点击确认按钮	1. 跳转到新建案例筛选接口测试任务页面 2. 提示新建成功，进入测试历史界面	通过

表5.4是新建测试任务功能的测试用例，主要验证新建测试任务流程是否正常。测试任务包括司法模型测试任务和案例筛选接口测试任务：新建司法模型测试任务，用户需从模型列表中指定待测模型，从数据集列表中指定测试集；新建案例筛选接口测试任务，需配置接口地址、参数模版和解析模版等关键信息。实际测试结果符合预期，对应测试用例全部通过。

表 5.5: 测试任务执行功能测试用例

测试项	操作/输入	预期结果	测试结果
维度不匹配的司法模型测试	1. 选择导航栏中司法测试选项，点击司法模型测试按钮 2. 选择输入或输出维度不同的模型和测试集记录 3. 填写名称等基本信息，点击确认按钮	1. 跳转到新建司法模型测试任务页面 2. 选择框内显示选定的模型和数据集记录名称 3. 提示新建成功，进入测试历史界面	通过

维度不匹配的司法模型测试	4. 刷新至测试任务执行完毕	4. 对应司法模型测试任务，执行状态为失败	通过
维度匹配的司法模型测试	1. 选择导航栏中司法测试选项，点击司法模型测试按钮 2. 选择维度相匹配的模型和测试集记录 3. 填写名称等基本信息，点击确认按钮 4. 刷新至测试任务执行完毕	1. 跳转到新建司法模型测试任务页面 2. 选择框内显示选定的模型和数据集记录名称 3. 提示新建成功，进入测试历史界面 4. 对应司法模型测试任务，执行状态为成功	通过
GET 请求案例筛选接口测试	1. 选择导航栏中司法测试选项，点击案例筛选接口测试按钮 2. 输入测试任务相关信息，配置接口请求类型为 GET，点击确认按钮 3. 刷新至测试任务执行完毕	1. 跳转到新建案例筛选接口测试任务页面 2. 提示新建成功，进入测试历史界面 3. 对应案例筛选接口测试任务，执行状态为成功	通过
POST 请求案例筛选接口测试	1. 选择导航栏中司法测试选项，点击案例筛选接口测试按钮 2. 输入测试任务相关信息，配置接口请求类型为 POST，点击确认按钮 3. 刷新至测试任务执行完毕	1. 跳转到新建案例筛选接口测试任务页面 2. 提示新建成功，进入测试历史界面 3. 对应案例筛选接口测试任务，执行状态为成功	通过

表5.5是测试任务执行功能的测试用例，主要验证系统对选择模型和数据集维度是否匹配情况下的司法模型测试任务和不同接口请求类型的案例筛选接口测试任务的执行情况，实际测试结果符合预期，对应测试用例全部通过。

表 5.6: 测试任务管理功能测试用例

测试项	操作/输入	预期结果	测试结果
司法模型测试任务查询	1. 点击侧边栏或用户下拉框测试历史按钮 2. 设置查询条件后点击确认按钮	1. 跳转到测试历史界面，默认显示司法模型测试管理页面 2. 页面展示满足筛选条件的司法模型测试记录	通过
司法模型测试任务详情查看	点击一条司法模型测试任务的详情按钮	页面弹出司法模型测试任务记录详情模态框，展示测试基本信息	通过
司法模型测试任务记录删除	点击一条司法模型测试任务的删除按钮	界面对应记录删除，系统提示删除成功	通过
案例筛选接口测试任务查询	1. 点击测试历史界面案例筛选接口测试管理按钮 2. 设置查询条件后点击确认按钮	1. 显示案例筛选接口测试管理页面 2. 页面展示满足筛选条件的接口测试记录	通过
案例筛选接口测试任务详情查看	点击一条案例筛选接口测试任务详情按钮	页面弹出案例筛选接口测试任务记录详情模态框，展示测试基本信息	通过
案例筛选接口测试任务记录删除	点击一条案例筛选接口测试任务删除按钮	界面对应记录删除，系统提示删除成功	通过

表5.6是测试任务管理功能的测试用例，主要验证司法模型测试任务和案例筛选接口测试任务的条件查询、详情查看、删除记录的正确性，实际测试结果符合预期，对应测试用例全部通过。

表 5.7: 查看测试报告功能测试用例

测试项	操作/输入	预期结果	测试结果
司法模型测试报告查看	进入司法模型测试管理界面，点击一条记录的测试报告按钮	跳转到报告界面，展示司法模型测试报告	通过

司法模型测试报告下载	测试报告界面点击下载报告按钮	建立测试报告下载请求，保存司法模型测试报告 PDF 文件到本地	通过
案例筛选接口测试报告查看	进入案例筛选接口测试管理界面，点击一条记录的测试报告按钮	跳转到报告界面，展示案例筛选接口测试报告	通过
案例筛选接口测试报告下载	测试报告界面点击下载报告按钮	建立测试报告下载请求，保存案例筛选接口测试报告 PDF 文件到本地	通过

表5.7是查看测试报告功能的测试用例，主要关注在对司法模型测试任务和案例筛选接口测试任务的测试报告查看和下载功能的验证，实际测试结果符合预期，对应测试用例全部通过。

5.3　非功能性测试

根据第三章设计的非功能需求，本小节对系统的时间特性、并发性进行非功能性测试，非功能需求中的可靠性、安全性、可扩展指标均在系统设计与实现中得以体现：可靠性通过公共模块的日志记录配置得以保障系统故障的可追溯性，系统微服务的架构模式使得系统具备较高的可靠性；权限认证模块的设计保证了用户使用系统的安全性；前端项目中通过复用已有组件进行扩展配置界面元素，增强了前端界面的可扩展性。

表 5.8: 响应时间非功能测试用例

界面	平均响应时间	最大响应时间	最小响应时间	测试结果
用户登录	659.4ms	1256.3ms	524.8ms	通过
新增模型上传记录	1250.1ms	1624.2ms	981.8ms	通过
新增数据集上传记录	1142.7ms	1821.5ms	782.6ms	通过
上传管理	842.4ms	1057.3ms	578.2ms	通过
新建司法模型测试任务	1469.5ms	1924.2ms	852.6ms	通过
新建案例筛选接口测试任务	1325.8ms	1741.9ms	742.1ms	通过

测试历史查看	836.2ms	1472.3ms	627.9ms	通过
司法模型测试报告查看	1721.7ms	2437.2ms	984.8ms	通过
案例筛选接口测试报告查看	2214.7ms	2812.6ms	1498.2ms	通过

通过使用工具 Fiddler 对系统的界面响应时间进行测试，分别对用户登录、新增上传记录、上传管理、新建测试任务、测试历史查看、测试报告查看各进行了 100 次界面响应操作统计其平均响应时间，如表5.8统计结果表明界面响应时间符合时间特性要求。此外，通过测试工具 Jemter 进行并发量为 10 的接口压测，覆盖系统全部接口，均正确反馈结果，满足系统并发性的要求。

5.4　本章小结

本章主要对案例筛选测试系统进行功能性测试和非功能性测试，介绍了测试目标、测试环境、测试设计及执行情况。根据第三章需求分析中模块划分和用例描述来设计系统功能测试用例，验证系统功能的有效性和完整性。根据系统的非功能性需求设计非功能性测试，验证系统的可靠性。

第六章　总结与展望

6.1　总结

司法案例筛选系统，旨在基于案例信息，为司法从业者提供类案信息，用于指导司法审判。该系统实现涉及司法模型和文本处理技术的应用，具有较高领域特征。本文研究了面向司法案例筛选系统测试的系统设计与实现，从司法模型测试和案例筛选接口测试两个角度来进行测试系统的开发。首先，介绍了选题的背景和意义、司法案例筛选系统及其测试的研究现状。紧接着，介绍了系统开发 web 应用框架、自然语言处理等相关技术。

随后，本文详细分析了案例筛选测试系统的功能需求和非功能需求，结合系统用例图进行用例分解和描述，并进一步通过“4+1 视图”对系统从架构、逻辑、进程、开发、部署等方面进行设计描述。在此之后，详细介绍了司法模型测试和案例筛选测试的指标评估体系。本文将系统分为权限管理、上传管理、测试管理和数据处理四个模块。模块间相辅相成，负责用户认证、网关控制、资源管理、测试任务管理及评估工作。

最后，本文介绍了各个功能模块的详细设计与具体实现，基于模块核心类图、涉及数据表字段详细设计和相关关键代码片段具体描述了功能模块的详细实现，同时通过部分页面截图直观展示系统实现情况。完成了以上工作后，本文进一步基于系统功能设计测试用例，并完成相关测试，确保系统的可用性和可靠性。

本系统从司法模型和系统接口两个粒度对案例筛选系统进行测评，建立统一测试指标体系和测评规范，为案例筛选系统开发和测试人员提供测试平台，促进智慧司法在案例筛选领域的改进优化。本人主要负责本系统的后端和前端部分的开发以及司法模型和案例筛选接口测评指标体系建立工作。

6.2　展望

本文初步实现了面向司法案例筛选系统的测试系统，实现了司法模型和案例筛选接口的测试评估功能，但是系统在使用中仍存在一些不足之处，可在以下几方面进行进一步完善：

首先，本系统将上传的模型和数据集文件直接存储到服务器的文件系统中，虽然能够提高文件获取效率，但是容易受到存储空间影响达到性能瓶颈。在进

一步的完善时，可以考虑采用大数据 Hadoop 技术进行文件分布式存储，以横向扩展来解决磁盘空间性能屏障，同时备份机制进一步保障数据安全。

其次，由于司法模型测试评估需进行深度学习框架调用，受限于显卡资源和 Python 语言性能，后台运行时间长，测试队列消费速度低于生产速度。后续可以考虑优化测试任务分发算法、进行测试任务 docker 化运行提升测试效率，如采用 celery 分布式任务调度模块和 nvidia-docker 技术来进行优化。

最后，测试评估可视化程度较低，仅能进行具体指标评估和综合评估，将测试指标与测试整体关联，未能对指标和测试内部数据建立有效关联，尚未进行深层挖掘分析，接下来可以探索对指标评估体系与测试数据进行有机结合，提供更丰富的司法领域相关的可视化手段。

参考文献

[1] 党振兴, 人工智能在司法领域应用研究, 大连干部学刊 (8).

[2] R. Binns, Fairness in machine learning: Lessons from political philosophy, arXiv preprint arXiv:1712.03586.

[3] S. Issacharoff, Judging politics: The elusive quest for judicial review of political fairness, Tex. L. Rev. 71 (1992) 1643.

[4] 缪成, 人工智能与大数据技术在司法领域的发展现状, 法制博览 (2019) 14–17.

[5] 马燕, 论我国一元多层级案例指导制度的构建——基于指导性案例司法应用困境的反思, 法学 (1) (2019) 15.

[6] 国务院, 新一代人工智能发展规划 (2017).

[7] 吴汉东, 人工智能时代的制度安排与法律规制, 法律科学：西北政法大学学报 (05) (2017) 130–138.

[8] 倪寿明, 充分挖掘司法大数据的超凡价值, 人民司法 (应用) (19) (2017) 2.

[9] 张力行, 计算机法律信息检索与计算机法律专家系统——理论与实践, 中外法学 (3) (1989) 41–47.

[10] 赵晓海, 法律检索系统更上一层楼, 中外法学 (1) (1991) 72–72.

[11] 潘庸鲁, 人工智能介入司法领域路径分析, 东方法学 3 (2018) 109–118.

[12] 张保生, 人工智能法律系统的法理学思考, 法学评论 5 (2001) 11–21.

[13] 汤维建, 智慧司法建设开启崭新篇章, 北京观察 (3) (2017) 25–25.

[14] 张保生, 法律推理的理论与方法, 中国政法大学出版社, 2000.

[15] 马治国, 刘宝林, 人工智能司法应用的法理分析: 价值、困境及路径, 青海社会科学 233 (05) (2018) 141–147.

[16] G. Yan, Y. Li, S. Shen, S. Zhang, J. Liu, Law article prediction based on deep learning, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2019, pp. 281–284.

[17] T. He, H. Lian, Z. Qin, Z. Zou, B. Luo, Word embedding based document similarity for the inferring of penalty, in: International Conference on Web Information Systems and Applications, Springer, 2018, pp. 240–251.

[18] 刘鹏, 人工智能辅助裁判理论思考与路径选择——以自然语言处理为例, 法治论坛 (1) (2019) 108–122.

[19] 刘品新, 大数据司法的学术观察, 人民检察 (23) (2017) 29–31.

[20] 徐骏, 智慧法院的法理审思, 法学 (3) (2017) 55–64.

[21] T. Brennan, W. Dieterich, B. Ehret, Evaluating the predictive validity of the compas risk and needs assessment system, Criminal Justice and Behavior 36 (1) (2009) 21–40.

[22] A. Savela, Evaluation of the Quality of Adjudication in Courts of Law: Principles and Proposed Quality Benchmarks: Quality Project of the Courts in the Jurisdiction of the Court of Appeal of Rovaniemi, Finland, Rovaniemi Court of Appeal, 2006.

[23] 廖奕, 当代中国司法改革窘境与均衡路径, 南京社会科学 3.

[24] M. Sokolova, N. Japkowicz, S. Szpakowicz, Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation, in: Australasian joint conference on artificial intelligence, Springer, 2006, pp. 1015–1021.

[25] C. J. Willmott, K. Matsuura, Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance, Climate research 30 (1) (2005) 79–82.

[26] J. N. Gross, Method of testing online recommender system, uS Patent 8,630,960 (Jan. 14 2014).

[27] Z. Qin, T. He, H. Lian, Y. Tian, J. Liu, Research on judicial data standard, in: 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2018, pp. 175–177.

[28] R. Johnson, J. Höller, A. Arendsen, T. Risberg, C. Sampaleanu, Professional Java Development with the Spring Framework, John Wiley & Sons, 2007.

[29] H. Suryotrisongko, D. P. Jayanto, A. Tjahyanto, Design and development of back-end application for public complaint systems using microservice spring boot, Procedia Computer Science 124 (2017) 736–743.

[30] I. Cosmina, Spring microservices with spring cloud, in: Pivotal Certified Professional Spring Developer Exam, Springer, 2017, pp. 435–459.

[31] A. Banks, E. Porcello, Learning React: functional web development with React and Redux, " O'Reilly Media, Inc.", 2017.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (Oct) (2011) 2825–2830.

[33] X. Qiu, Q. Zhang, X.-J. Huang, Fudannlp: A toolkit for chinese natural language processing, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2013, pp. 49–54.

[34] Z. Wang, et al., Segment-based fine-grained emotion detection for chinese text, in: Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing, 2014, pp. 52–60.

[35] F. Peng, F. Feng, A. McCallum, Chinese segmentation and new word detection using conditional random fields, in: Proceedings of the 20th international conference on Computational Linguistics, Association for Computational Linguistics, 2004, p. 562.

[36] G. Nicolai, G. Kondrak, Leveraging inflection tables for stemming and lemmatization., in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016, pp. 1138–1147.

[37] J. Sun, Jieba chinese word segmentation tool, Accessed: Jun 25 (2012) 2018.

[38] S. Bai, H. J. P. Wu, H. Li, G. Loudon, System for chinese tokenization and named entity recognition, uS Patent 6,311,152 (Oct. 30 2001).

[39] R. Kang, H. Zhang, W. Hao, K. Cheng, G. Zhang, Learning chinese word embeddings with words and subcharacter n-grams, IEEE Access 7 (2019) 42987–42992.

[40] J.-Y. Nie, J. Gao, J. Zhang, M. Zhou, On the use of words and n-grams for chinese information retrieval, in: Proceedings of the fifth international workshop on on Information retrieval with Asian languages, 2000, pp. 141–148.

[41] J. Li, K. Zhang, et al., Keyword extraction based on tf/idf for chinese news document, Wuhan University Journal of Natural Sciences 12 (5) (2007) 917–921.

[42] C. Banea, S. Hassan, M. Mohler, R. Mihalcea, Unt: A supervised synergistic approach to semantic text similarity, in: Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Association for Computational Linguistics, 2012, pp. 635–642.

[43] Y. Song, D. Roth, Unsupervised sparse vector densification for short text similarity, in: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015, pp. 1275–1280.

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

简历与科研成果

基本情况 王栋，男，汉族，1995 年 07 月出生，山东省莱州市人。

教育背景

2018.9 ~ 2020.6 南京大学软件学院 硕士

2014.9 ~ 2018.6 南京大学软件学院 本科

读研期间的成果（包括发表的论文及参与的专利）

1. **Wang D**, Wang Z, Fang C, et al. DeepPath: Path-Driven Testing Criteria for Deep Neural Networks[C]//2019 IEEE International Conference On Artificial Intelligence Testing (AITest). IEEE, 2019: 119-120.
2. Chen Y, Wang Z, **Wang D**, et al. Behavior Pattern-Driven Test Case Selection for Deep Neural Networks[C]//2019 IEEE International Conference On Artificial Intelligence Testing (AITest). IEEE, 2019: 89-90.
3. Chen Y, Wang Z, **Wang D**, et al. Variable Strength Combinatorial Testing for Deep Neural Networks[C]//2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2019: 281-284.
4. 陈振宇、鲍泓、蔡彦、**王栋**、夏志龙、万俊，“一种基于精度误差的模型稳定性评估方法”，申请号：2019107238704，已受理。
5. 陈振宇、鲍泓、蔡彦、章许帆、**王栋**，“一种基于深度学习模型的测试数据分布多样性的评估方法”，申请号：2019107238719，已受理。
6. 陈振宇、**王栋**、王子元、陈炎杉、钱航，“一种基于路径状态的神经网络充分性评估方法”，申请号：201910268447.X，已受理。

致　　谢

时光如白驹过隙，两年的工程硕士研究生生活即将落下帷幕，在论文完成之际，我想向硕士就读生涯中所有帮助和关心过我的老师、同学、朋友、亲人致以衷心的感谢。

首先，感谢我的指导老师刘嘉老师，刘老师认真负责，不仅在专业知识和项目实践以及毕业设计中给予我很多建议和帮助，也在生活中教会我很多做人道理和人生哲学。同时，还要感谢实验室的陈振宇老师、房春荣老师、何铁科老师们的悉心指导，毕设期间在提纲拟定、内容编排等方面提出很多改进建议，使我获益良多！

其次，我要感谢在毕设期间帮助过我的所有同学和朋友，在项目开发中提供很多实践经验指导帮助我解决遇到的技术难题，感谢你们的热心帮助，从你们身上我学到很多项目开发技术和对技术执着认真的态度。

最后，我要感谢父母对我学习和生活无私的奉献和支持，在我情绪低沉时，对我悉心呵护开导，让我走出阴霾重拾信心，在我骄傲浮躁时，告诫我自律慎独，让我戒骄戒躁稳打稳扎，养育之恩无以为报！同时，向论文评阅及论文答辩委员会的老师致以最诚挚的谢意，感谢各位老师的辛勤工作。

版权与原创性说明

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

作者签名：王栋

日期：2020 年 05 月 22 日

《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名：王栋

2020 年 05 月 26 日

论文题名	面向司法案例筛选的测试系统的设计与实现				
研究生学号	MF1832154	所在院系	软件学院	学位年度	2020
论文级别	☐学术学位硕士 ☑专业学位硕士 ☐学术学位博士 ☐专业学位博士 (请在方框内画钩)				
作者 Email	mf1832154@smail.nju.edu.cn				
导师姓名	刘嘉				

论文涉密情况：

☑ 不保密

☐ 保密，保密期（_____年_____月_____日 至 _____年_____月_____日）