南京大學

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目 基于小样本学习的测试用例分类系统

作 者 姓 名 唐昊杰

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 05 月 20 日

学　　号：**MF20320136**

论文答辩日期：**2022 年 05 月 20 日**

指 导 教 师：　　**（签字）**

Test Case Classification System Based on Few-Shot Learning

By

Haojie Tang

Supervised by

Professor Zhenyu Chen

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 2022

学 位 论 文 原 创 性 声 明

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

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不句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。

论文作者签名：_____

日期：　　年　　月　　日

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：基于小样本学习的测试用例分类系统

工程硕士（软件工程领域）专业 2020 级硕士生姓名：唐昊杰

指导教师（姓名、职称）：陈振宇 教授

摘 要

由于软件版本迭代开发的速度加快，软件测试工作需要频繁开展。众包测试、抽样测试等方法被用于降低测试成本、提高测试效率。众包测试过程中会产生大量测试用例。测试人员需要从测试用例库中挑选合适的用例进行测试。通过类别标签为测试用例分类能够帮助测试人员挑选合适的测试用例。然而现实场景中带标签的测试用例获取困难。项目中难以直接获得大量带标签的测试用例，为测试用例手动标注类别标签的成本也十分高昂，同时带标签的实际测试用例集还存在着标签分布不均匀的问题，为存量测试用例的挑选和复用等实际应用带来了一定的挑战。

本文针对测试用例实际应用中的上述问题，提出了基于小样本学习的测试用例分类技术，开发了测试用例管理与分类系统。在管理测试用例的过程中，测试人员上传测试用例集，系统提取测试用例相关信息，使用文本数据扩增技术扩大测试用例集规模，使用语言技术平台和词向量模型提取测试用例关键词并向量化，使用带有注意力机制的双向长短期记忆模型收集测试用例的类别词库并获取测试用例的分类。系统将标记测试用例的分类和关键词作为标签，测试人员可以查看测试用例标签、获取同类别测试用例并下载测试用例。

本系统使用 Vue2 作为测试用例分类系统的前端框架，SpringBoot 作为测试用例分类系统的后端框架，Flask 作为测试用例特征提取和分类服务的后端框架。使用 MySql 作为关系型数据库，MongoDB 作为 NoSql 数据库提供测试用例模型对象的存储。使用 Nginx 实现前后端负载均衡，提高系统可用性和可扩展性。使用 Docker 容器技术虚拟化服务，提高系统的可移植性和服务间的松耦合，保障服务的高性能、轻量级、跨平台部署。

本系统已经部署在真实项目中并投入试用，为企业提供稳定的测试用例管理和分类服务。本系统的可用性评估实验选取了针对 3 款软件的上千条测试用例进行用例分类。实验结果经过人工验证，标签准确率接近 60%，优于其他基线算法。本系统使测试人员无需手工标记纷繁复杂的测试用例，呈现测试用例的分类结果和标签信息，提高测试用例使用效率。

关键词：测试用例，小样本学习，分类算法，类别标签

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Test Case Classification System Based on Few-Shot Learning

SPECIALIZATION: Software Engineering

POSTGRADUATE: Haojie Tang

MENTOR: Professor **Zhenyu Chen**

Abstract

As the iterative development of software versions accelerates, testing needs to be performed frequently. Methods such as crowdsourced testing and sample testing are used to reduce testing costs and improve testing efficiency. A large number of test cases are generated during the crowdsourced testing process. Testers need to select suitable test cases from the test case library for testing and reusing. Categorizing test cases by category labels can help testers select right test cases. However, it is difficult to obtain labeled test cases in real-world scenarios. Directly obtaining a large number of labeled test cases in the project is hard, and the cost of manually labeling the test cases is also very high. At the same time, the actual test case set with labels still suffers from an uneven distribution of labels, which poses certain challenges for the practical application of the test cases.

Aiming at the above problems in the practical application of test cases, this paper proposes a test case classification technology based on Few-shot Learning, and develops a test case management and classification system. In the process of managing test cases, testers upload test case sets, the system extracts test case related information, uses text augmentation to increase the size of test case sets, uses language technology platform and Word2Vec model to extract test case keywords and vectorize them, collects the category lexicon of test cases using bi-directional long and short-term memory model with attention mechanism and obtains the classification of test cases. The system will tag the test cases with their categories and keywords, and testers can view the tags, obtain test cases of the same category and download test cases.

This system uses Vue2 as the front-end framework of the test case classification system, SpringBoot as the back-end framework of the test case classification system,

and Flask as the back-end framework of the test case feature extraction and classification service. Use MySql as relational database and MongoDB as NoSql database to provide storage of test case model objects. Use Nginx to achieve front-end and back-end load balancing to improve system availability and scalability. Use Docker container technology to virtualize services to improve system portability and loose coupling between services, guaranteeing high performance, lightweight and cross-platform deployment of services.

This system has been deployed and put into trial use in real projects, providing enterprises with stable test case management and classification services. In the usability evaluation experiment of this system, thousands of test cases for three softwares are selected for keyword extraction and test case classification. The experimental results have been manually verified, and the label accuracy rate has been close to 60%. This system makes it unnecessary for testers to manually mark complicated test cases, presents the classification results and label information of test cases, and improves the use efficiency of test cases.

Keywords: Test Case, Few-Shot Learning, Classification Algorithm, Category Label

目录

表 目 录

图 目 录

第一章　引言

1.1　项目背景及意义

21 世纪是互联网的时代，万物互联、万物上云是时代的发展趋势，随着互联网基础资源的建设速度加快和数字应用基础服务的日益丰富，以 PC 端和移动端为主的软件服务深入到消费流通、生活服务、文娱内容、医疗教育、工业互联网等各个领域，为群众带来快捷的数字便利。根据中国互联网络信息中心 2022 年发布的第 48 次《中国互联网络发展状况统计报告》[1]统计，截至 2021 年 6 月，在我国注册的网站数量达到 422 万个，APP 软件在架数量达到 302 万款，我国网民规模达 10.11 亿，互联网普及率较 2020 年 12 月增长 1.2%，网络生态的复杂和多样对软件产品与服务的质量和性能提出了更高的要求。

软件测试 [1] 是保障软件应用稳定性、改善应用质量的重要环节。软件测试指在软件交付使用前需要针对软件可能出现的问题进行检测，并通过技术手段对测试中遇到的问题加以消除，检验软件是否满足规定的需求，避免软件系统投入使用后出现安全或质量隐患。近年来，敏捷开发模式的兴起导致软件开发的节奏变快，敏捷带来的持续测试要求期望测试人员高效的完成测试，为软件测试带来了很多新的挑战。一方面，快节奏开发迭代下软件测试的人力成本愈发高昂，传统的软件测试手段依靠测试人员手动设计测试用例进行相应的手工测试，测试时间成本约占总开发时间成本的 40% 甚至 60%[2]。也不能适应敏捷开发的持续集成需求。另一方面，在开发过程完成后调用测试团队的传统测试方法不能满足敏捷开发持续集成、持续部署的需求，严重限制了测试效率。

持续集成 [3] 将测试周期前移到开发周期，利用自动化测试 [4] 有效解决了一部分捷测试成本和效率的问题。在持续集成中，测试人员开发和运行自动化脚本，通过在软件开发周期中反复运行自动化的接口功能测试用例和部分 UI 功能测试用例，降低了手工测试的投入成本，保证了已覆盖自动化测试的模块质量，最终达到降低测试的多次投入成本的目的。然而开发和运维人员往往难以按计划执行自动化测试，据统计，只有 4% 的公司能做到 90% 以上的测试实现自动化 [5]，主要原因有需求变更频繁、产品的可测试性差、单元测试不足、测试人员能力和测试工具功能不足等。

众包测试 [6] 是软件测试的一个新兴趋势。它充分利用众包和云平台的优

[1]https://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/202109/t20210915_71543.htm

势，具有测试速度快、测试成本低等特点，备受业界的青睐，十分适用于缺乏专业测试人员和规范测试手段的公司和个人。众测过程中会产生大量的测试用例，由于众测工人的水平不一，测试人员往往会重新从测试用例集中挑选合适的用例进行检验并供以后使用。

众包测试的测试用例通常具有类别标签稀少、类别标签分布不均匀、类别标签标注错误等特点。测试人员有时难以分辨用例的用途，难以判断用例分类的准确性，导致众包测试测试用例重复利用率不高。人工重新标注或更改这些标签虽然准确，但当测试用例集较大时效率低下且难以维持较高的准确率。针对这些问题，我们提出通过生成合理的标签提炼测试用例内容、标识测试用例类别，辅助测试人员更好的挑选和利用已有测试用例集。

本文旨在设计一个测试用例管理和分类系统，针对众包测试中测试用例缺少类别标签或类别标签不准确的问题，能够完成基于测试用例文本分析和小样本学习的测试用例标签提取和类别分类。本文就是系统后续管理和分类服务流程的具体说明。服务基于众包测试的测试用例及其关联的测试报告等真实数据，利用文本特征提取和深度学习等方法对其进行深度挖掘，生成具有代表性的标签，表明该测试用例的类别和场景，帮助测试人员管理测试用例，降低测试成本。

1.2 国内外研究现状

1.2.1 软件测试应用现状

软件测试是在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程 [7]。简而言之，它描述了一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。常见的软件测试阶段包括单元测试、集成测试、系统测试和回归测试。其中，单元测试是对软件组成单元进行测试 [8]，其目的是检验软件基本组成单位的正确性，测试的对象是软件设计中的函数，单元测试还有助于开发人员编写更好的代码。集成测试也称综合测试、联合测试，将程序模块采用适当的集成策略组装起来，对系统的接口及集成后的功能进行正确性检测的测试工作，其主要目的是检查软件单位之间的接口是否正确，集成测试的对象是已经经过单元测试的模块。系统测试主要包括功能测试、界面测试、可靠性测试、易用性测试、性能测试。回归测试指在软件维护阶段，为了检测代码修改而引入的错误所进行的测试活动 [9]。回归测试是软件维护阶段的重要工作，有研究表明，回归测试带来的耗费占软件生命周期的 1/3 总费用以上 [10]。与普通的测试不同，在回归测试过程开始的时候，测试者有一个完整的测试用例集可供使用，因此，如何根据代码的修改情况

对已有测试用例集进行有效的复用是回归测试研究的重要方向。回归测试的主要目的是测试原有功能以及测试新加入的功能是否有副作用。

然而伴随着软件研发模式的改变和人工智能、大数据、云计算、区块链、物联网等软件新技术的发展和普遍应用，如今的软件测试面临着各种各样的挑战。例如，在针对 AI 软件的测试中，测试往往是黑盒的：从测试的样本来看，样本通常具有较大的局限性；从测试的过程来看，整个过程具有较长的周期且难以实现自动化测试；从测试的结果来看，结果也具有不可解释性和低鲁棒性。针对物联网的测试又面临着测试环境复杂性、高可靠性、实时性、兼容性等严格的要求。在未来，软件测试会向着敏捷化、高度自动化、云化、服务化、智能化的方向进步 [5]，如何快速反馈软件质量、提高测试效率和资源使用效率、降低成本是软件测试前进道路上永恒的主题。

测试用例是软件测试中的核心，保障了软件的测试质量。关于软件测试用例的研究，主要集中在测试用例生成、测试用例复用、测试用例约简等方面。测试用例生成是软件测试领域研究的重要方向，根据是否需要被测软件的源码，测试用例生成技术可以被分为白盒测试用例生成技术和黑盒测试用例生成技术 [11]。测试用例复用指把已经执行的测试用例用在该软件新的版本或者其他软件的测试中，主要体现在测试用例服用的设计思路、实现思路、实现过程等方面 [12]。测试用例集约简指在仍然满足测试准则的前提下，通过删除所有冗余测试用例得到测试用例集的最小约简测试用例集 [13]。测试用例相关技术的发展，解决了测试人员由于经验不足难以设计测试用例的问题，保障了测试用例的设计质量，能够有效提高软件测试效率和质量。

1.2.2 文本分类研究现状

文本分类 [14] 是指按照一定的分类体系或规则使用机器对文本集实现自动划归类别的过程，是自然语言处理任务中的一项基础性工作，主要是根据已知文本集找到文本特征和文本类别之间的关联，再利用这种关联对新的文本进行分类。文本分类流程上一般包括文本预处理、文本特征提取、分类模型构建和分类器训练等过程 [15]。

根据是否需要标记数据，文本分类的方法可以被划分为无监督文本分类、半监督文本分类、有监督文本分类三种 [16]。无监督文本分类不需要带类别标记的训练数据辅助。有监督的机器学习方法需要大量有标签训练数据，但分类结果高度依赖人工标记，成本较高。半监督文本分类只需要输入少量的有标记数据，通过学习少量标记数据和大量无标记数据的潜在特征，建立分类模型，并对新数据做出预测。传统的分类算法包括朴素贝叶斯算法、K-近邻算法、决策树算

法、随即森林算法、支持向量机算法等。

相对于传统的分类模型，非传统的文本分类方法往往能够获取文本中更高层更抽象的语义特征，弥补传统模型中人为设计特征的缺陷。非传统的文本分类方法包括：深度学习、集成学习、迁移学习、强化学习等。集成学习通过融合多个文本分类器的分类结果提高文本分类的准确率，既保留了各个基分类器之间的差异性，也提高了模型的鲁棒性；迁移学习在大规模通用语料库上训练语言模型并将其在具体数据集上微调，能够降低模型的训练难度并节省训练的计算资源；强化学习通过试错和延迟回报的核心思想将文本分类问题建模成离散有序的决策过程，具有较高的可解释性。

文本分类可以被应用于多个领域。在搜索推荐领域，文本分类方法生成的准确分类标签为资源检索和个性化推荐提供了有力的支撑。在情感分类领域，文本分类可以分析用户评论的情感倾向，帮助分析热点话题、理解用户习惯、监控危机舆情。在对话系统领域，分类用户提出的问题使得智能客服能够代替人工客服向用户提供问题的解答，在有效降低了运营成本的同时也改善了用户的咨询体验。

1.2.3 小样本学习研究现状

随着大数据时代的到来，机器学习和深度学习在许多数据密集型领域和服务中取得了成功。机器凭借强大的计算设备、完善的大型数据集、先进的模型和算法做到许多人类做不到的事情。然而成功的人工智能应用基本依赖于大规模标注数据集的输入，现实世界的真实场景中某些类别只有少量数据甚至少量的标注数据。人类能够通过利用他们过去所学的知识来迅速学习新任务，而相比之下人工智能却很难从少数样本中准确又迅速的归纳新知识，这是人工智能和人类学习之间存在的差距。

Li 等人在 2003 年正式提出了小样本学习的概念 [17]，他们认为，当新的类别只有一个或几个带标签的样本时, 已经学习到的旧类别可以帮助预测新类别。小样本学习是一种为了从有限的有监督信息的例子中学习而提出的新的机器学习范式，在每个类别只给定几个标记的样本的情况下学习分类器。除了想令人工智能像人类一样学习，小样本学习能够在很难或不可能获得足够的样本时为这些罕见情况学习模型，例如在一个药物发现任务中，小样本学习可以预测一个新成分是否具有毒性。小样本学习还可以帮助减轻收集大量有监督信息的样本的负担，减少数据收集和计算的成本，因为对无标签数据进行标注会消耗大量的时间和人力。

小样本学习方法最早从计算机视觉领域兴起，近年来同样逐渐应用于机器

人学、自然语言处理、声学信号处理、医疗、金融、随机搜索等各个领域。它在自然语言处理领域发展较缓慢，原因在于当样本量较少时图像能提取比文本更多的特征，同时由于文本的语义中含有更多的变化和噪声，图像特征的提取也更加容易。在自然语言处理应用中，通常使用小样本方法实现对话系统、情感分类、句子完成、口语理解、单词相似性或 WordEmbedding 等 NLP 基本任务。

1.3 本文的主要工作

针对以上研究现状和当前众包测试与回归测试中测试用例复用所面临的标注数据稀少的问题，本文提出了基于小样本学习的测试用例分类技术，开发了测试用例管理和分类系统，实现了在测试用例管理阶段对无标签测试用例的标记，并为测试人员在管理和挑选需复用的测试用例提供辅助的判断依据。本文的主要工作如下：

（1）基于前期调研和现状分析，针对当前测试用例管理和分类面临的困难，在考虑系统部署、使用和后期扩展的情况下，设计了包括测试用例管理模块、特征提取模块、分类模块、类别词库收集模块在内的测试用例管理和分类系统，满足了用户管理、复用测试用例和自动化标注无标签测试用例的需要。

（2）实现了测试用例管理模块。本模块中用户只需打开指定的网址并输入账号验证身份即可访问测试用例管理和分类系统，用户在本模块中可以完成创建测试用例、上传测试用例集、生成用例标签、导出测试用例、审核测试用例、审核用例标签等操作。

（3）实现了测试用例分类服务。本系统通过文本数据扩增、关键词提取、词向量化等方式实现了对测试用例的特征提取模块，首先从数据库中获取历史测试用例的测试名称、测试过程、测试需求和测试模块等属性数据，输入特征提取模块得到历史测试用例词向量集，根据词向量集使用基于小样本学习的分类方法训练出分类模型。之后，当获取到测试用例管理模块的调用请求时，使用特征提取模块处理请求信息，根据分类模型结果返回预测的测试用例类别。

（4）实现了一种在带标签测试用例数据量较少的情况下对大部分无标签测试用例分类的方法。实现了一个带注意力机制的双向长短期记忆网络模型以训练基础的测试用例文本基础分类器。实现了基于该网络模型的类别词库收集机制，构建词库收集每个类别的代表性关键词。根据基础分类器和词库的匹配规则共同预测无标签数据的类别，扩大带标签训练集，从而重新训练模型以提供更准确的分类器。

（5）系统测试和实验分析。本文完成并展示了对本系统功能性测试、性能测试的结果，验证了本系统功能的可用性和可靠性。在真实测试集上完成了对分类方法有效性和优越性的验证。

1.4 本文的组织结构

本文共分为六个章节，组织结构如下：

第一章，引言部分。首先介绍了软件测试和众包测试的发展现状，然后对软件测试、测试用例、文本分类和小样本学习的国内外研究现状进行了说明，针对上述情况，本文提出了基于小样本学习的测试用例分类技术。

第二章，相关技术综述。对于系统开发和部署过程中使用到的相关工程技术以及算法理论进行了简单的介绍。

第三章，系统需求分析与概要设计。首先分析了系统需求，从涉众分析，架构设计，系统模块划分和“4+1”视图等多个角度对系统进行了说明。接着对系统使用到的分类算法和类别词库收集算法进行了设计，使用流程图对系统的四个模块进行了概要设计和说明。最后设计了系统的数据模型。

第四章，系统详细设计与实现。采用顺序图，核心类图，关键代码和文字描述相结合的方式，阐述了测试用例管理模块、测试用例特征提取模块、测试用例分类模块、类别词库收集模块的具体实现。

第五章，系统测试与实验分析。通过对系统进行功能和性能测试，验证了系统的可用和可靠。通过在真实众包测试测试用例集上的实验分析，验证了基于小样本学习的测试用例分类技术在已分类数据在数据集中占比较少的情况下分类的提升效果。

第六章，总结与展望。总结了项目开发和论文撰写过程中的相关工作，分析了系统和算法服务的不足，对基于小样本学习的测试用例分类系统的未来发展方向做出展望。

第二章　相关技术概述

2.1　文本预处理相关技术

2.1.1　语言技术平台

本系统使用语言技术平台 LTP 对测试用例文本进行预处理。LTP[18] 是一个综合中文处理平台，包括一套高性能的自然语言处理模块和相关语料库。

LTP 使用在线学习算法框架处理分词、词性标注、依存句法分析等结构化学习任务。在线学习指每次通过一个训练实例学习模型的学习方法，每当完成一次正确的预测，其结果可以直接被用来修正模型。

2.1.2　Word2Vec

本系统使用 Word2Vec 模型获取测试用例文本关键词列表的词向量。Word2Vec 是一群用来生成词向量的相关模型，由 Tomas Mikolov 于 2013 年在 Google 带领研究团队创造 [19][20]。Word2vec 主要依赖 Skip-gram 模型或连续词袋模型 CBOW 来建立神经词嵌入。其中，连续词袋模型 [21] 以一个中心词语在文本序列中的上下文作为输入预测该中心词，而 Skip-gram 模型 [22] 以一个中心词作为输入预测该词在文本序列中的上下文。

Word2vec 是一个两层神经网络，通过向量化单词来处理文本。它的输入是一个文本语料库，输出是一组用以表示该语料库中单词的特征向量。Word2Vec 采用层次 softmax 或负采样加速训练神经网络。其中，层次 softmax 使用霍夫曼树代替传统的神经网络，降低了模型复杂度，提高了模型训练效率 [23]。负采样使每个训练样本被输入神经网络后只更新一小部分的神经元权重，降低了梯度下降过程的计算量，提高了计算效率 [24]。

Word2Vec 广泛应用于个性化推荐、相似度计算、快速检索、情感分析等领域，也可以将生成的词向量当作特征作为另一个模型的输入来使用。

2.1.3　SimBERT

本系统使用 SimBERT 模型对原始测试用例集进行文本数据扩增。SimBERT[25] 是基于 UniLM 思想、融检索与生成于一体的 BERT 模型，由追一科技的苏剑林老师于 2021 年开源。

SimBERT 的文本生成能力来源于 UniLM 模型 [26]，UniLM 模型具有前缀双向可见的注意力机制，利用特定的自我注意掩码来控制预测条件的上下文并

使用共享的多层 Transformer 网络在大量文本上进行预训练，可以为自然语言生成任务进行微调。

SimBERT 主要应用于相似文本生成和相似文本检索。通过 SimBERT 根据现有的已分类文本数据生成相似文本，或构建测试用例语料库库从语料库库中检索相似文本获取语义相关性较好的伪标签测试用例文本数据，可以有效提升分类器的效果。本系统在历史数据预处理的过程中使用 SimBERT 扩增测试用例的伪标签数据，扩大了众包测试真实测试用例集的规模，为测试用例管理和分类系统提供性能更好的分类服务。

2.2 小样本学习相关技术

2.2.1 小样本学习

现有的小样本学习工作可以被分为数据、模型和算法三个方向 [27]。从数据角度来看，小样本学习方法主要使用先验知识来增强训练数据，增强样本集以丰富训练的监督信息。可以通过人为定义的规则进行数据增强，例如图像领域中的平移 [28]、翻转 [29]、裁剪 [30]、缩放 [31] 等。然而这些扩增规则很大程度上取决于领域知识，并需要昂贵的人力成本。它们首先很难在其他数据集产生作用，其次也很难列举出所有可能的扩增规则，因此人工数据增强不能完全解决小样本问题 [32]。还有三类更高级的数据增强方法。（1）利用无标签数据对小样本数据集进行扩充 [33]。当目标任务或类别存在大量的弱监督或未标记的样本时，可以选中其中部分样本赋予伪标签来增强训练集。（2）通过增强样本特征对训练集进行扩充 [34][35]。利用辅助数据集或辅助属性进行样本的特征增强，提高由于小样本学习中样本量过少而导致的低特征多样性。（3）通过聚合和适应来自其他数据集的样本来增强训练集。使用生成对抗网络 [36] 判断新生成的样本是否属于训练集中的样本类别，通过损坏和生成来自辅助数据集的样本将新生成的样本合成到原先的数据集中去。

从模型角度看，小样本学习主要专注于如何利用旧知识来学习新知识，将已经学会的源领域知识迁移到一个新的具有一定相关性的目标领域中帮助训练分类模型。根据其方法不同可分为基于度量学习、基于元学习和基于图神经网络的方法三类 [37]。

在基于度量学习的方法中，Koch 等人在 2015 年最先提出使用孪生神经网络进行单样本图像识别 [38]，从数据中学习度量并利用学习到的度量比较和匹配未知类别的样本。Snell 等人在 2017 年提出了原型网络 [39]，对于同属一个类别的样本求得样本向量的平均值作为该类别的原型，通过不断训练模型和最小

化损失函数使得同类别内样本距离更靠近, 不同类别的样本更远离。Gao 等人和 Sun 等人在原型网络后续的工作中添加了注意力机制 [40][41]，证明了不同的样本和特征对分类任务的重要性不同。

从算法角度看，小样本学习主要关注如何提供模型最好的初始化参数 [42] 和后续微调优化该参数的策略 [43]。当源大数据集和目标小样本数据集分布较类似时，通常在大规模数据上预训练模型，在小样本数据集上对神经网络模型的连接层进行参数微调，最终得到微调后的模型。然而在现实世界的真实场景中，目标数据集和源数据集的数据分布往往并不相似，采用微调方法会导致模型在目标数据集上过拟合。

2.3 相关工程技术

2.3.1 SpringBoot

SpringBoot 是基于 Spring Framework 4.0 派生的生产级别的 Spring 应用框架，可以以最小的依赖引入来构建一个 Spring 应用 [44]。SpringBoot 框架具有如下几种特点：

（1）拥有嵌入式的 Tomcat, Jetty, Undertow 或者 Reactor Netty。Springboot 在 web 应用模版中提供了内置的网络容器，致使 Web 工程可以不用封装 war 包访问外部的网络容器运行，而可以直接使用 maven 打包后的 jar 包运行。

（2）尽可能地自动配置（@AutoConfiguration）Spring 和第三方库。SpringBoot 会自动为 jar 包里面的类解析、生成和配置 Bean，供开发者直接调用，因此在使用 Springboot 开发时开发者可以不考虑如何获取和配置这些环境，只需要按照 Springboot 既定的框架准备即可。

（3）提供指标、运行状态检查和外部化配置等生产环境中的功能。例如，SpringBoot 提供对运行时项目监控的功能。

（4）无需麻烦而冗余的 XML 配置，一切都可以使用 Java 配置。可以使用 @Service、@Bean 等注解为程序注入 bean。

本系统选用 SpringBoot 框架开发基于 Java 的后端服务。

2.3.2 Vue

Vue.js 是一套主张渐进式的前端框架，大大降低了入门难度。开发者可以根据需要由浅到深的学习 Vue 的不同模块，而不需要学习过完整的技术模块后再进行开发。

Vue.js 采用了声明式渲染，DOM 状态只是数据状态的一个映射。基于数据

双向绑定的原理，降低了数据层和视图层的耦合度，也保证了数据和视图可以同步更新。

Vue.js 引入了虚拟 Dom 的概念 [45]。前端通过 JS 处理 DOM 更新视图数据，而大量的 DOM 更新一定会影响系统性能。Vue.js 在内存中生成与真实 DOM 与之对应的数据结构，这个在内存中生成的结构便称之为虚拟 DOM。当数据发生变化时，能够智能的计算出重新渲染组件的最小代价并应用到 DOM 操作上，减少了不必要的渲染，提升了系统性能。

Vue.js 采用了组件化的思想。Vue.js 将构成的元素拆分成一个个组件。这样的好处也是显而易见了，提高了代码维护性，可复用性，可测试性。在项目中我们还使用了其他的 Vue 模块。我们用 vue-router 来实现页面之间的无刷新跳转，用 vue-router 的导航守卫来解决前后端分离下的认证和权限控制问题，同时也用了 vuex 作为状态管理，存储了如用户信息等多组件间的共有数据信息。

本系统选用 Vue 框架开发前端页面服务。

2.3.3 Flask

Flask 是一个使用 Python 编写的轻量级 Web 应用框架 [46]。Flask 使用简单的核心但保留了扩增的弹性，使用扩展增加诸如 ORM、文件上传、身份验证等其他功能。微框架 Flask 基于了 Pocoo 计划的 Werkzeug 和 Jinja2。Flask 框架具有如下特点：

（1）内置开发服务器和调试器。和 SpringBoot 类似，Flask 自带的开发服务器使开发者在调试程序时无需再安装其他任何诸如 Tomcat、JBoss 或 Apache 等网络服务器。程序启动后 Flask 默认处于调试状态，任何信息或错误都会被打印在控制台或客户端上。

（2）集成的单元测试支持。Flask 通过 test_client 函数向开发者提供测试接口并能够与自带的单元测试框架 unitest 衔接。通过该函数，测试程序可以模拟进行 HTTP 访问的客户端来调用 Flask 路由处理函数，并获取函数的输出来进行自定义的验证。

（3）使用 Jinja2[47] 模板引擎。Flask 通过使用 Jinja2 模版技术关联前端页面和后端程序。Jinja2 是一个非常灵活的 html 模版技术，能够自动抗击 XSS 跨站攻击并且易于调试。

（4）完全兼容 WSGI1.0[48] 标准。WSGI 标准要求框架具有强伸缩性并能运行于多线程或多进程环境下。正是由于遵守了这一标准，Flask 能够自由配置倒各种大型网络服务器中。

（5）基于 Unicode 编码。Flask 默认会在 HTTP 请求头中自动设置编码格式

为 UTF-8，开发者在正常情况下无需刻意关注编码问题。

本系统选用 Flask 框架开发基于 Python 的后端服务。

2.3.4 Docker

Docker[49] 是一个允许使用者快速构建、测试、部署应用程序的软件平台。它使得用户可以将原本部署在基础设施中的软件应用独立出来，将应用和应用的原型环境封装成容器这种能够轻松迁移的较小粒度，以此提高软件的交付速度。

Docker 利用 Linux 核心中的资源分离机制，使得多个容器可以共同部署并运行在同一操作系统中，避免启动虚拟机造成的额外负担 [50]。Linux 核心对命名空间的支持完全隔离了工作环境中应用程序的视野，包括进程树、网络、用户 ID 与挂载文件系统，而核心的 cgroup 提供资源隔离，包括 CPU、存储器、block I/O 与网络。

Docker 的核心组件包括 Docker Client、Docker Daemon、Docker Registry Docker Container。其中 Docker Client 负责与 Docker Daemon 进行连接，发出命令。Docker Daemon 负责处理命令，主要包括创建、运行、保存容器等任务。Docker Registry 负责对 Docker 镜像进行存储和管理。Docker Container 是从镜像创建而来的，是运行中的镜像实例。

本系统选用 Docker 容器部署前后端服务。

2.3.5 本章小结

本文介绍了项目中涉及到的相关技术和算法。首先对文本预处理相关技术进行了介绍，包括 LTP 语言技术平台、词向量模型 Word2Vec 和文本数据扩增技术 SimBERT；然后从数据、模型、算法三个角度对小样本学习相关技术进行了介绍；最后对系统实现和部署的相关工程技术进行了介绍，包括服务端 SpringBoot 和 Flask 框架、前端 Vue 框架、部署运维工具 Docker，说明了这些技术的优势。

第三章　系统需求分析和概要设计

3.1　系统整体概述

本系统在测试用例上传过程中向测试人员提供创建单个测试用例、测试用例集上传和测试用例存储功能。在测试用例分类和筛选的过程中向测试人员提供测试用例标签生成功能、测试用例下载功能和测试用例修改、审核、标签审核功能。在系统管理过程中向管理员账号提供测试人员账号创建和删除功能。目的是在管理测试用例集的同时，智能分析缺少类别标签的测试用例并生成相应标签，这些标签包括了用例关键词和潜在的用例类别，在测试用例类别分部不均衡和大部分用例缺少描述性标签的情况下，辅助测试人员更好的了解和挑选测试用例，优化测试用例的使用体验。同时，对于某一测试用例智能生成的标签列表，测试人员可以手动为该测试用例在标签列表中选择更加正确的标签或手动输入新标签，提高标签准确率。

在一场众包测试中，对于某一指定的测试目标，众包工人会手动创建测试用例，一个测试用例集就是同一测试目标下所有测试用例的集合。每条测试用例包含了测试用例名称、测试过程、测试模块、测试用例描述、标签和测试目标类别等多个属性，其中标签属性表示该条测试用例的辅助类别和关键词集合，测试目标类别属性指该条测试用例所测的系统模块类别如功能完整性、用户体验、页面布局、不正常退出等。本系统中泛指的标签通常包括了标签和测试目标类别这两个具体属性。

本章将侧重针对基于小样本学习的测试用例标签生成和分类技术进行需求分析，对于系统实现的前后端架构、用例分类和类别标签生成算法、实体类和数据库等关键内容进行概要设计和说明，对于系统的五个模块进行详细的设计。

3.2　系统需求分析

3.2.1　涉众分析

本系统作为闭环的测试用例管理和分类平台，采用了基于小样本学习的测试用例分类技术实现测试用例的类别标签的自动生成。系统的涉众分析如表3.1所示，主要涉众包括测试人员、评审专家、系统管理员。

测试人员作为测试用例的管理人员，可以通过系统上传测试用例集，填写测试用例，筛选测试用例，查看测试用例详情，修改测试用例，生成测试用例标

签，挑选测试用例并下载等。测试人员负责管理和维护由众测工人编写的测试用例，测试人员不一定需要拥有很高的代码开发能力，但他们需要掌握被测软件的功能及测试需要的软硬件环境，深入分析被测软件的测试需求和验收标准，精简和修改众测工人不专业的口语化表达，防止测试用例的歧义。同时，他们需要掌握正确的测试用例挑选策略，在有测试用例和缺乏测试资源的情况下能够挑选合适的测试用例保证测试质量。希望系统提供良好的用户体验，帮助他们提高测试用例管理和挑选效率。

评审专家作为测试用例和测试用例标签的评审方,可以审核测试用例和审核生成的测试用例标签，为每条测试用例确定正确的分类标签并去除不恰当的关键词标签，间接影响测试人员最终挑选测试用例的质量。评审专家在测试人员拥有能力的基础上，还具有丰富的测试用例评审经验，能够充分评估测试用例的可执行性和全面性，准确判断测试用例的类别。希望系统提供良好的标签选择功能、评审功能和用户体验。

系统管理员作为测试用例管理平台的系统管理者，可以添加和删除测试人员账号和评审专家账号，为帐号授予不同系统操作如查看测试用例、挑选用例标签等的权限。他们拥有一定的管理能力。希望保障系统的安全性和便捷性，提高账户的使用体验。

表 3.1: 系统涉众分析表

涉众类别	涉众特征与期望
测试人员	作为测试用例的管理人员，他们希望在管理用例过程中获得系统的帮助，进行测试用例集的上传和便捷管理。测试人员应具有一定的专业测试能力，应该具有一定的测试用例编写和管理能力，能够进行用例的修改和筛选。应该具有良好的测试用例挑选策略，能够为待测软件挑选合适的用例。
评审专家	作为测试用例和用例标签的评审方，他们希望测试用例标签尽可能包括用例的准确类别、概括用例的特征，希望系统提供良好的测试用例审核和标签审核体验。评审专家应该具有精深的测试能力，拥有丰富的测试用例评审经验，准确判断标签和测试用例的匹配程度。
系统管理员	作为测试用例管理和分类系统的管理员，熟悉系统业务和功能，有丰富的维护、管理系统的经验，了解互联网法律法规并确保系统稳定健康的运营，同时对系统资源进行管理和维护以防系统过载。期望能对用例系统进行较好的管理，避免测试资源浪费、恶意提交和权限混乱。

3.2.2 功能性需求

功能需求是对用户需求的分解，是产品中必须实现的核心软件需求，是软件创造价值的基础。根据系统背景和涉众分析，从系统输入测试用例、系统输出

用例标签、用例管理以及用户管理等角度出发，我们总结了如下表3.2所示的测试用例管理分类系统功能需求列表。

表 3.2: 系统功能需求表

需求编号	需求名称	需求描述
RQ1	创建单条测试用例	系统应该允许测试人员创建测试用例。测试人员进行测试用例基本信息填写，包括用例名称、用例类别、测试方法、用例描述、输入及操作步骤等。填写完成后点击按钮创建并存储测试用例。
RQ2	上传测试用例集	系统应该允许测试人员上传测试用例集。测试人员上传包含多条测试用例的 Excel 表格，表格每行代表一条测试用例，每列代表测试用例的一条属性。
RQ3	查看测试用例详情	系统应该允许测试人员查看测试用例详情。测试人员点击按钮查看详细信息，包括测试用例新生成的标签和分类。
RQ4	查看测试用例列表	系统应该允许测试人员进入列表界面查看测试用例列表，并提供列表筛选功能，筛选条件包括用例标签、测试方法、用例类型等。
RQ5	修改测试用例	系统应该允许测试人员进入测试用例详情界面，点击编辑按钮后任意修改测试用例内容。
RQ6	生成测试用例标签	系统应该允许测试人员进入测试用例界面，点击标签生成按钮生成用例标签提供挑选用例的参考信息。
RQ7	审核测试用例	系统应该允许评审专家审核测试人员提交的测试用例，修改测试用例并审核通过或不通过测试用例。
RQ8	审核测试用例标签	系统应该允许评审专家在测试用例界面操作某条用例新生成的测试用例标签，可以删除或保留标签。
RQ9	导出测试用例	系统应该允许测试人员在测试用例界面自由勾选并以 Excel 表格形式或 Json 形式导出测试用例以供测试使用。
RQ10	查看账号列表	系统应该允许系统管理员进入账户管理界面查看账户列表。
RQ11	创建删除账号	系统应该允许系统管理员在账户管理界面创建或删除测试人员账号或审核账号。
RQ12	调整账号权限	系统应该允许系统管理员在账户管理界面为测试账号增加或取消评审权限。

在测试人员的业务流程中，测试人员首先需要登录验证具有相应管理权限的测试账号。接着测试人员上可以分别选择以特定 Excel 表格的形式上传待管理的测试用例集或手动编写测试用例的详细内容上传单挑测试用例，此时测试用例处于用例待审核状态。待测试用例审核通过后，测试人员可以查看具体测试用例并点击生成标签按钮，通知系统运行算法模块为单个测试用例生成高概率匹配该用例特征和分类的标签。此时生成标签的用例进入标签待审核状态。待标签审核通过后，测试人员可以自由挑选满足测试需求的测试用例并以 Excel 表

格形式导出。

在评审专家的业务流程中，评审专家首先需要登录验证具有相应审核权限的评审账号。在测试人员上传测试用例后，评审专家可以查看或修改审核中的测试用例并决定用例是否通过审核。在测试人员生成用例标签后，评审专家可以删除认定不匹配测试用例的标签，保存较符合用例情况的标签，也可手动添加标签，随后通过标签审核。

在系统管理员的业务流程中，验证身份后的管理员可以添加或删除测试账号与评审账号。同时，可以修改两种账号的管理权限，当急需下载测试用例时，测试账号也可以临时审核通过用例和用例标签。

3.2.3 非功能性需求

非功能性需求，是指软件产品为满足用户业务需求而必须具有且除功能需求以外的特性，包括性能、安全性、可用性、可靠性、可拓展性、易用性、伸缩性等。非功能性需求反映了软件系统质量和特性的额外需求。作为一款面向专业测试人员的辅助工具，测试用例管理和分类系统在可用性、可靠性、安全性、性能等方面均有较高的要求。

性能：软件性能是指软件在尽可能少地占用系统资源的前提下,尽可能高地提高运行速度。本系统在正常网络环境下，应保证所有的简单查询导出请求从操作到得到反馈的时间均控制在 200ms 以内，其他复杂查询请求、创建删除等写请求或生成标签请求性能要求可放宽至 500ms 以内。

安全性: 安全性是指软件运行不引起系统事故的能力。本系统应确保访问和数据安全，防止用户权限错乱，防止用户信息和测试用例发生信息泄漏。

可靠性：可靠性是指在规定的条件下和规定的时间内，软件不引起系统失效的能力。本系统应具有良好的异常处理机制与数据备份和恢复机制，妥善处理程序运行时可能出现的错误，妥善保管与备份系统存储的测试用例与标签数据。

可用性：可用性是指在某段考察时间内系统能够正常运行的概率或时间占有率期望值。本系统应保障全年 99.9% 的可用性。从技术角度上减少故障次数、缩短故障恢复时间。当系统发生崩溃或服务不可用时，应保证快速告警发现，及时使用负载均衡等手段切换备用系统，通过回滚、重启、扩容等手段恢复故障系统。

可扩展性：可扩展性是指在保持软件不变的情况下，系统性能可以随系统规模扩充而提高的特性。本系统应具有较高的可扩展性，采用低耦合的架构和模块设计，对用例管理模块、标签生成模块、用例分类模块进行抽象，分离数

据提取部分和程序运行部分。在为新模块增加系统功能时，无需修改已有模块。在旧模块中重构新增功能时，不影响其他模块。

可维护性：可维护性指的是维护人员对该软件进行维护的难易程度。本系统应具有较高的可维护性。系统需要结构合理并具有详细的设计文档和日志记录模块、代码注释方便运维人员、维护人员查阅理解。

易用性：易用性指用户使用系统提供的资源的便利程度。本系统应具有较强的易用性，符合通用的人机交互规范。本系统的主要使用者是专业测试人员，应聚焦其关注的用例管理功能，系统界面简洁、布局合理、易于理解，具有丰富的信息提示引导和反馈。

3.2.4 系统用例图

根据系统功能性需求的分析结果，本文将系统分为如图3.1所示的 12 个用例。本系统的用户可分为测试人员、评审人员和系统管理员。测试人员负责测试用例在系统中的大部分生命周期活动，包括创建测试用例、上传测试用例集、查看测试用例详情、查看测试用例列表、修改测试用例、生成测试用例标签、导出测试用例。

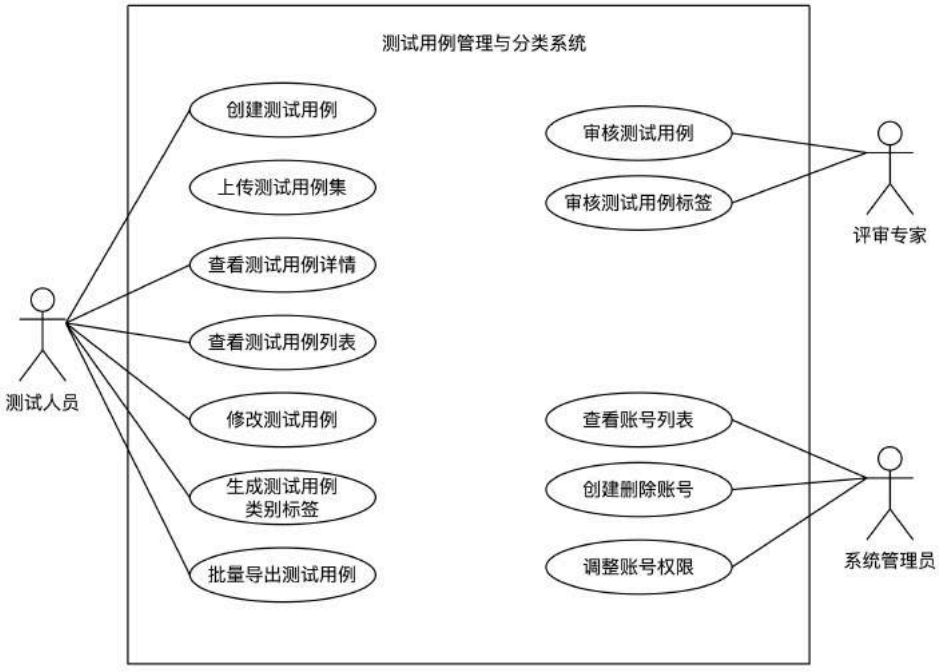

图 3.1: 系统用例图

评审人员负责审核和修改测试人员创建或上传的测试用例与生成的测试用例标签，只有本身审核通过或包含标签被审核通过的测试用例才能被测试人员

导出。评审人员和测试人员共有修改测试用例用例，另外拥有审核测试用例、审核测试用例标签用例。

系统管理员负责系统账号和权限的管理。查看账号列表用例、创建删除账号用例、调整账号权限用例属于系统管理员。

3.2.5 系统用例描述

本小节详细描述了系统用例图中展示的用例，介绍用例的优先级、触发条件、前置条件、后置条件、正常流程、扩展流程等信息。

UC1 创建单条测试用例的用例描述如表3.3所示。测试人员登录系统后，点击左侧菜单栏选项即可进入测试用例创建与上传页面，进行单条测试用例信息填写，包括用例名称、用例类型、测试方法、设计人员、设计日期、测试用例描述、输入及操作步骤、预期结果和可选的前提与约束、备注、用例标识等信息。输入完成后点击创建按钮后，测试用例即创建成功，并处于“待审核”状态。其中，系统在输入的属性值不符合规范或输入的用例名称重复时给出相应的提示。

表 3.3: 创建单条测试用例用例描述表

ID	UC1
用例描述	创建测试用例
参与者	测试人员
触发条件	测试人员点击菜单项进入测试用例创建与上传页面
前置条件	测试人员必须已被识别和授权
后置条件	1. 评审专家能在审核测试用例页面看到该用例； 2. 测试人员能在查看测试用例列表页面检索到该用例，并处于“待审核”状态。
优先级	高
正常流程	1. 测试人员点击菜单项创建测试用例； 2. 系统显示测试用例创建与上传页面； 3. 测试人员输入测试用例基本信息，包括必填的用例名称、用例类型、测试方法、设计人员、设计日期、测试用例描述、输入及操作步骤、预期结果和可选的前提与约束、备注、用例标识等属性，点击创建按钮； 4. 系统显示创建成功，保存测试用例信息，回显输入并补充用例创建时间。
拓展流程	3a. 测试人员创建测试用例校验不通过； 1. 系统提示测试用例属性值不规范，请重新填写测试用例 3b. 测试人员创建测试用例重名； 1. 系统提示测试用例命名重复
特殊需求	1. 测试人员编辑测试用例信息后未创建，系统不自动保存其草稿

UC2 上传测试用例集的用例描述如表3.4所示。当测试人员登录系统并进入测试用例创建与上传页面，点击上传测试用例集按钮后，系统弹出文件选择页面。测试人员从本机目录中选择内含测试用例数据的 Excel 文件，点击上传确认

按钮将文件上传到系统。系统在校验内容无误后显示测试用例集上传成功提示，逐条存储文件内的所有测试用例并使其全部处于“待审核”状态。特别的，当被上传文件格式不正确或被上传文件内测试用例属性名不正确时，系统会给出相应的错误提示。

表 3.4: 上传测试用例集用例描述表

ID	UC2
用例描述	上传测试用例集
参与者	测试人员
触发条件	测试人员点击菜单项进入测试用例创建与上传页面
前置条件	测试人员必须已被识别和授权
后置条件	1. 评审专家能在审核测试用例页面看到包含在测试用例集内的全部测试用例； 2. 测试人员能在查看测试用例列表页面检索到包含在测试用例集内的用例，并处于“待审核”状态。
优先级	高
正常流程	1. 测试人员点击菜单项创建测试用例； 2. 系统显示测试用例创建与上传页面； 3. 测试人员点击上传测试用例集按钮 4. 系统弹出文件选择与上传页，提示请选择待上传的文件； 5. 测试人员选择 Excel 表格，点击上传按钮； 6. 系统显示上传成功，保存全部测试用例信息。
拓展流程	5a. 测试人员上传文件种类不正确； 1. 系统提示上传文件格式错误，请上传 Excel 文件 5b. 测试人员上传文件中的首行属性不正确； 1. 系统提示上传测试用例集属性名有错误，请修改属性名后重新上传
特殊需求	无

UC3 查看测试用例列表的用例描述如表3.5所示。测试人员进入系统后点击左侧菜单栏选项即可查看具体的测试用例列表页面。系统会分页显示所有测试用例的用例名称、用例描述、创建时间、上传人员等信息。同时系统支持列表从用例类型、测试方法、测试类型、上传人员四个维度进行筛选，并支持从创建时间进行正序倒序排列。

UC4 查看测试用例详情的用例描述如表3.6所示。测试人员在列表页面对应测试用例条目处点击查看按钮，进入测试用例详情页面查看测试用例详细信息，包括用例名称、用例类型、测试方法、设计人员、设计日期、测试用例描述、输入及操作步骤、预期结果和可选的前提与约束、备注、用例标识、上传日期、状态、标签。当测试人员创建的测试用例通过评审专家审核后，系统在测试用例列表页面显示该测试用例为“审核通过”状态，允许通过标签选择器批量导出该条测试用例。系统在测试用例详情页面显示用例为“审核通过”状态，允许导出该

表 3.5: 查看测试用例列表用例描述表

ID	UC3
用例描述	查看测试用例列表用例描述
参与者	测试人员
触发条件	测试人员点击菜单项进入测试用例列表页面
前置条件	测试人员必须已被识别和授权
后置条件	系统显示所有筛选结果
优先级	高
正常流程	1. 测试人员点击菜单项测试用例列表； 2. 系统显示测试用例列表页面； 3. 测试人员点击列表顶部属性筛选按钮，分别有测试用例类型、测试方法、测试类型、上传人员、状态等按钮； 4. 系统弹出筛选条件选择框，可选内容为每个属性的枚举值； 5. 测试人员点击某一枚举值选项； 6. 系统显示属性符合选择的测试用例列表。
拓展流程	无
特殊需求	无

条测试用例。测试人员在详情页面点击导出测试用例操作，系统将测试用例下载到测试人员机器上。

表 3.6: 查看测试用例详情用例描述表

ID	UC4
用例描述	查看测试用例详情
参与者	测试人员
触发条件	测试人员在测试用例列表页面点击某条测试用例的查看按钮
前置条件	1. 测试人员必须已被识别和授权
后置条件	无
优先级	高
正常流程	1. 测试人员点击查看测试用例； 2. 系统跳转至当前测试用例详情页，显示包括用例名称、用例类型、测试方法、设计人员、设计日期、测试用例描述、输入及操作步骤在内的用例基本信息；
拓展流程	1a. 测试人员点击导出测试用例操作； 1. 系统将测试用例下载到测试人员机器上。
特殊需求	无

UC5 修改测试用例的用例描述如表3.7所示。测试人员进入测试用例详情界面点击编辑按钮，该条测试用例即可进入可编辑状态。测试人员修改用例信息后点击保存按钮，系统保存修改后的测试用例信息，提示“保存成功”并在测试用例详情页面回显更新信息，同时修改后的用例也在测试用例列表页面刷新后

更新。此时该条测试用例进入“待审核”状态，等待评审专家审核。评审专家在测试用例评审详情页面也可以点击编辑按钮，该条测试用例也可进入可编辑状态。特别的，当修改后的测试用例属性值未通过校验时，系统给出相应的错误提示。

表 3.7: 修改测试用例用例描述表

ID	UC5
用例描述	修改测试用例
参与者	测试人员
触发条件	测试人员点击编辑按钮
前置条件	1. 测试人员必须已被识别和授权 2. 测试人员进入测试用例详情页面
后置条件	系统保存修改后的测试用例信息
优先级	高
正常流程	1. 测试人员点击编辑按钮； 2. 系统显示测试用例编辑页面； 3. 测试人员修改测试用例属性信息； 4. 测试人员点击保存按钮； 5. 系统保存修改后的测试用例信息。
拓展流程	3a. 修改后的测试用例属性值校验不通过； 1. 系统提示属性值未通过校验，请重新填写。
特殊需求	无

UC6 生成测试用例标签的用例描述如表3.8所示。测试人员进入测试用例列表页面，点击右上角生成标签按钮，列表中的测试用例统一补全用例类型和生成用例标签。测试人员在用例列表页面点击查看按钮进入用例详情页面，点击右上角生成标签按钮，系统为该条测试用例生成测试用例标签，并提示测试用例标签生成成功。

UC7 审核测试用例的用例描述如表3.9所示。评审专家登录系统后，点击左侧菜单栏选项即可进入测试用例评审页面，系统显示待评审的测试用例列表。评审专家点击查看按钮进入待评审的测试用例详情页面，点击编辑按钮进入测试用例编辑页面，修改测试用例后点击保存按钮，系统保存评审专家修改过的测试用例信息。评审专家点击审核通过或不通过按钮，系统在测试用例详情页面和列表页面更新测试用例的状态为“审核通过”或“审核不通过”。

UC8 审核测试用例标签的用例描述如表3.10所示。评审专家登录系统后，点击左侧菜单栏选项即可进入测试用例标签评审页面，系统显示待评审标签的测试用例列表。评审专家点击查看按钮进入待评审标签的测试用例详情页面，点击编辑按钮进入测试用例编辑页面，修改或删除测试用例新生成的标签后点击

表 3.8: 生成测试用例标签用例描述表

ID	UC6
用例描述	生成测试用例标签
参与者	测试人员
触发条件	测试人员进入测试用例详情页面
前置条件	1. 测试人员必须已被识别和授权 2. 测试人员进入测试用例详情页面
后置条件	1. 评审专家能在测试用例标签审核列表看到该用例和对应新生成的标签； 2. 测试人员能在测试用例列表和测试用例详情界面看到标签，表示处于“标签等待审核”状态。
优先级	高
正常流程	1. 测试人员点击生成标签按钮； 2. 系统用实线框显示新生成的标签，表示该标签待审核； 3. 系统保存新生成的标签信息。
拓展流程	1a. 测试人员在测试用例列表页面点击生成标签按钮； 1. 系统为每条测试用例补充测试用例类别并生成测试用例标签。
特殊需求	无

表 3.9: 审核测试用例用例描述表

ID	UC7
用例描述	审核测试用例
参与者	评审专家
触发条件	有新的测试用例集或测试用例被上传到系统中
前置条件	评审专家必须已被识别和授权
后置条件	1. 测试人员能在测试用例列表页面看到最新测试用例更新为审核通过状态 2. 点击查看按钮可查看测试用例详情
优先级	高
正常流程	1. 评审专家进入到待审核的测试用例列表页面； 2. 系统按时间倒序分页显示全部待审核的测试用例； 3. 评审专家选择某个测试用例点击查看详情按钮； 4. 系统跳转至测试用例详情页面； 5. 评审专家点击审核通过按钮； 6. 测试用例更新为“审核通过”状态并更新详情页面。
拓展流程	5a. 评审专家点击拒绝通过； 1. 测试用例更新为“审核不通过”状态并更新详情页面。
特殊需求	无

保存按钮，系统保存评审专家修改过的标签信息。评审专家点击标签审核通过或不通过按钮，系统在测试用例详情页面和列表页面更新测试用例的状态为“审核通过”或“标签审核不通过”。

UC9 批量导出测试用例的用例描述如表3.11所示。

表 3.10: 审核测试用例标签用例描述表

ID	UC8
用例描述	审核测试用例标签
参与者	评审专家
触发条件	测试用例新生成了标签
前置条件	评审专家必须已被识别和授权
后置条件	无
优先级	高
正常流程	1. 评审专家进入到待审核标签的测试用例列表页面； 2. 系统按时间倒序分页显示全部待审核标签的测试用例； 3. 评审专家选择某个测试用例点击查看详情按钮； 4. 系统跳转至测试用例详情页面； 5. 评审专家修改标签并点击保存；6. 测试用例更新为标签“审核通过”状态并更新详情页面。
拓展流程	5a. 评审专家点击拒绝通过； 1. 测试用例更新为“标签审核不通过”状态并更新详情页面。
特殊需求	无

表 3.11: 批量导出测试用例用例描述表

ID	UC9
用例描述	批量导出测试用例
参与者	测试人员
触发条件	测试人员点击批量导出按钮
前置条件	1. 测试人员必须已被识别和授权 2. 测试人员进入到测试用例列表页面
后置条件	测试人员够打开并查看导出后的 Excel 文件中的测试用例内容
优先级	高
正常流程	1. 测试人员选择多条测试用例； 2. 测试用例左侧的选择框上呈现被选中样式； 3. 测试人员点击批量导出测试用例按钮； 4. 系统下载测试用例集为 Excel 表格形式并保存到测试人员的电脑上。
拓展流程	5a. 评审专家点击拒绝通过； 1. 测试用例更新为审核未通过状态并更新详情页面。
特殊需求	无

测试人员登录系统后，通过左侧菜单栏进入测试用例列表页面，通过列表中的选择器选择想要导出的测试用例，点击导出按钮将测试用例转化为固定测试用例格式的 Excel 文件，下载至测试人员本地机器。

3.3 系统总体设计

3.3.1 系统总体架构设计

本系统是一个采用小样本学习方法，面向真实业务场景下的少部分带标签测试用例和大部分无标签测试用例，为它们提供类别标签补全功能的测试用例分类系统。

系统在架构上采用了前后端分离的开发方式。前端采用 Vue.js 框架开发单页应用，后端测试用例管理和分类服务使用 Springboot 框架开发 Java 后端，以 Restful API 的形式向前端提供调用接口。本系统使用 MySql 数据库存储原始测试用例数据和系统沿伸的标签、测试用例类型、账户等管理数据，Java 后端使用 Spring JPA 查询和使用数据库数据。

考虑到基于小样本学习的测试用例分类技术需要使用深度学习和机器学习相关技术，Python 语言中的 Pytorch 库和 Sklearn 库以及基础的数据处理和数学计算库对深度学习和机器学习的支持较好。测试用例关键词提取和分类预测服务使用轻量级 Web 开发框架 Flask 开发 Python 后端并为 Java 后端提供调用接口，使用 Simbert 提供测试用例数据扩增服务，使用基于词库的双向长短期记忆人工神经网络 + 注意力机制提供预训练后的测试用例分类模型。最后，使用 Docker 容器技术打包各个服务，提供在各个复杂实际服务器环境中的无差别部署能力。

基于小样本学习的测试用例分类系统整体架构设计图如图3.2所示。

1）使用 Vue.js 框架进行前端开发，采用单页面的开发模式，以 JavaScript 作为业务逻辑的开发语言，使用 Element UI 作为前端开发组件库，使用 npm 管理前端项目依赖并打包前端项目为静态文件部署，同时使用 nignx 分发前端流量。

2）使用 Springboot 框架进行后端主体测试用例管理和分类系统开发，基于 Spring mvc 的分层思想将项目分为控制器层（controller）、业务模型层（service）、数据库访问层（dao）、域模型层（model）进行分层开发。其中，控制器层负责处理请求、响应和前后端交互，调用业务模型层的接口。业务模型层存放业务处理逻辑，调用数据库访问层持久化业务数据。数据库访问层将数据存放在持久化介质中，同时提供增删改查操作。以上三层的控制和实现信息通过配置文件导入。域模型层负责存放实体，主要定义和数据库对象对应的属性和提供附带方法。使用 maven 作为包管理和打包工具，将项目打包为 jar 包后部署。

3）使用 Flask 框架进行分类服务接口开发，采用了小样本学习技术训练后保存的分类模型，为测试用例管理和分类系统提供测试用例的分类服务。分类服务以微服务的形式向主体系统后端提供服务，测试用例管理和分类系统的业务模型层访问分类服务提供的 Restful API 获取服务。

4）使用 Pytorch 深度学习框架进行神经网络模型开发。使用 Simbert 模型生成相似的测试用例文本进行数据扩增。使用 csv 库和 pymysql 库读取历史测试用例数据，使用 LTP 中文自然语言处理工具对测试用例文本进行分词、词性筛选、去停用词、同义词转换处理。使用 Embedding 技术对分词后的文本进行词向量化。使用 torch 库构建网络层和定义前向传播、反向传播，获取词向量的注意力权重并与具体词汇关联、构建类别专属词库。深度学习训练出的模型保存后供分类服务调用。

5）使用 MySql 作为后端和分类服务的持久化数据库，使用阿里云 OSS 存储历史测试用例文本和图像。

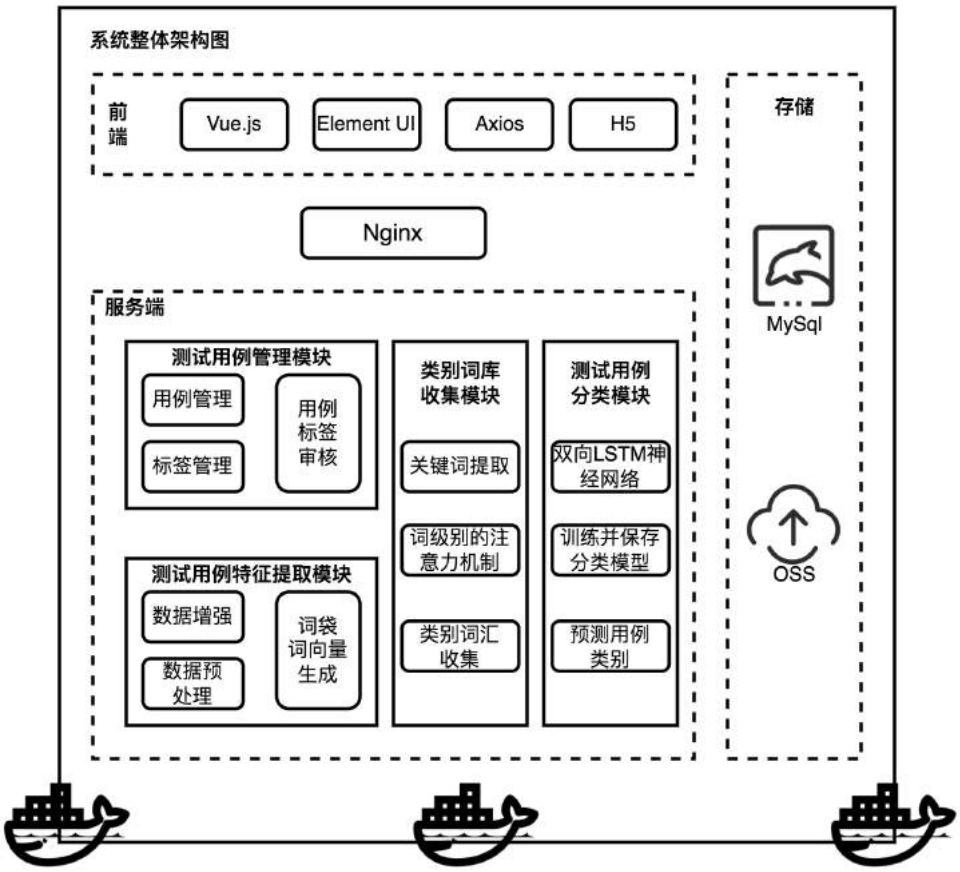

图 3.2: 系统整体架构设计图

3.3.2 系统模块划分

如图 3.2 所示，本系统可以划分为测试用例管理模块、测试用例特征提取模块、类别词库收集模块、测试用例分类模块四个部分。其中测试用例管理模块主要位于测试用例管理后端服务中，其余三个模块位于分类服务中。

测试用例管理模块的主要功能是使用友好的人机交互界面收集测试用例数据至服务端并进行持久化，同时为不同的用户角色提供用例、用例标签的管理和审核功能。在一次完整的测试用例分类流程中，测试人员在本模块创建测试用例，系统将用例数据保存至 MySql 中，当测试用例审核通过后，测试人员点击生成标签按钮获取用例的类别标签，系统调用分类服务获取当前测试用例的类别。

测试用例特征提取模块的主要功能是使用 Simbert 模型对测试用例数据进行数据扩增和增强，使用 LTP 工具对扩增后的数据集进行分词、筛选词性、去停用词、同义词转换等预处理操作，对预处理后的关键词文本进行词向量和词袋模型的训练。在一次完整的测试用例分类流程中，本模块将测试用例处理为能够表示其数据特征的词向量，将历史测试用例数据扩增、词向量化后用于训练分类模型并将模型保存至本地或阿里云 OSS 中，将实时测试用例数据词向量化后输入训练好的分类模型并得到模型返回的分类结果。

测试用例分类模块的主要功能是训练分类模型并提供分类结果。在一次完整的测试用例分类流程中，本模块首先构建双向长短期记忆人工神经网络，定义输入历史测试用例的词向量形式数据，训练得到测试用例分类器。对于实时测试用例的词向量形式数据输入，分类器返回该条测试用例的分类结果。

类别词库收集模块的主要功能是收集分类器训练过程中注意到的能够代表某个类别的词汇，形成词库并辅助分类器判断测试用例类别。在一次完整的测试用例分类流程中，针对历史数据，本模块关注双向长短期记忆人工神经网络训练出的分类器对每条测试用例分类的置信度，挑选高置信度的测试用例中具有最高注意力权重的词汇，把收集到的词集当作置信度最高的类别的词库。针对实时测试用例数据，本模块结合分类器，根据分类器的置信度和该条测试用例与各类别词库的匹配程度共同判断测试用例数据的类别。

系统采用 http 协议实现前后端服务、后端 Java 服务与后端 Python 服务间的交互。OSS 主要负责存储历史测试用例数据中页面截图和分类模型等大文件数据。Docker 主要负责部署各个服务，降低系统部署成本，提高系统部署效率。

3.3.3 4+1 视图

软件架构需要分析的内容往往复杂繁多，单一的场景视图具有一定的表达限度，难以捕捉所有的系统架构要点。Philippe Kruchten 教授在其论文《The 4+1 View Model of Architecture》中首次提出 4+1 视图的方法，分别从逻辑视图、开发视图、进程视图、物理视图、场景视图这五个不同的视角描述软件架构设计。每个视图仅用来描述一个特定的所关注方面的问题集合，多个并发视图结合起

来才能完整的反映系统设计。其中，场景视图已在上文用例分析中进行过详细描述，本节仅从其他四个视图的角度描述本系统的软件架构。

（1）逻辑视图

逻辑视图主要从最终用户的角度表述整个系统的抽象结构，关注系统提供最终用户的功能，不涉及具体的编译即输出和部署，常用类图、交互图、时序图表示。本文采用 UML 类图的方式展示逻辑视图。本系统的逻辑视图如图3.3所示

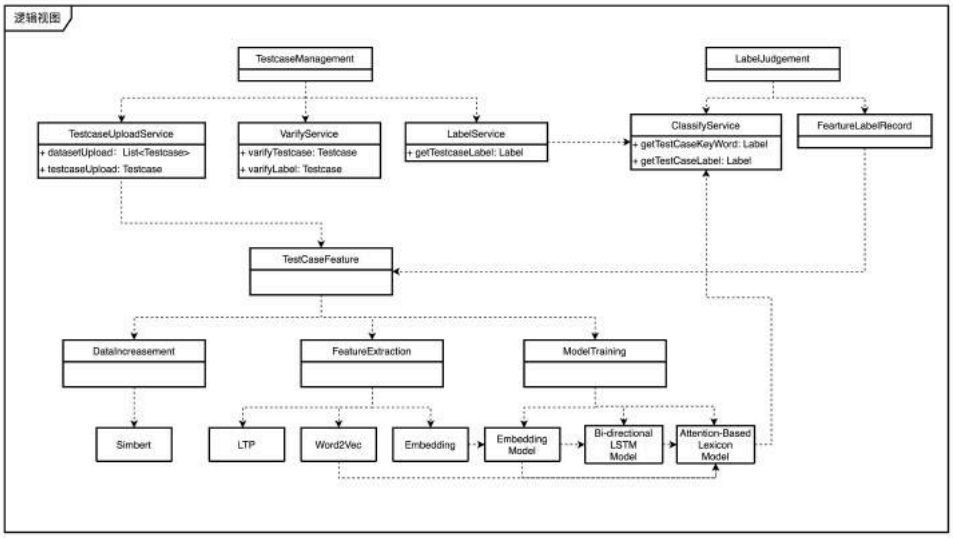

图 3.3: 逻辑视图

TestcaseManagement 模块进行测试用例的管理，TestcaseUpload 负责测试用例的上传，包括单条测试用例具体内容的填写和上传与测试用例数据集的上传，将数据存储至 MySql；VarifyService 负责测试用例的校对，分为测试用例内容的校对和标签分类的校对，将校对结果写进 MySql；LabeService 负责调用 LabelJudgment 模块的分类服务为测试用例获取分类。TestcaseFeature 模块负责测试用例数据的扩增、测试用例特征的提取和特征文本分类模型的构建，使用历史测试用例数据进行特征模型训练和分类模型训练，使用训练后的模型对实时上传的测试用例进行特征提取并分类，其中 DataIncreasement 负责使用 Simbert 对历史测试用例数据进行相似语义的数据扩增，丰富训练集；FeatureExtraction 负责特征提取，使用 LTP 将句子分词并过滤为词汇文本，使用 Word2Vec 提取词汇特征并构建词袋模型，使用 Word Embedding 将词汇文本转化为表示能够句子特征的词向量；ModelTraining 负责构建和训练双向 LSTM 模型和基于注意力机制的词库模型对转化后的词向量判断类别。LabelJudgement 模块分类测试用例，其中 ClassifyService 负责保存和调用分类模型、获取测试用例的特征词，

FeatureLabelRecord 对分类记录进行存储，反馈给 TestcaseFeature 模块，用于优化模型。

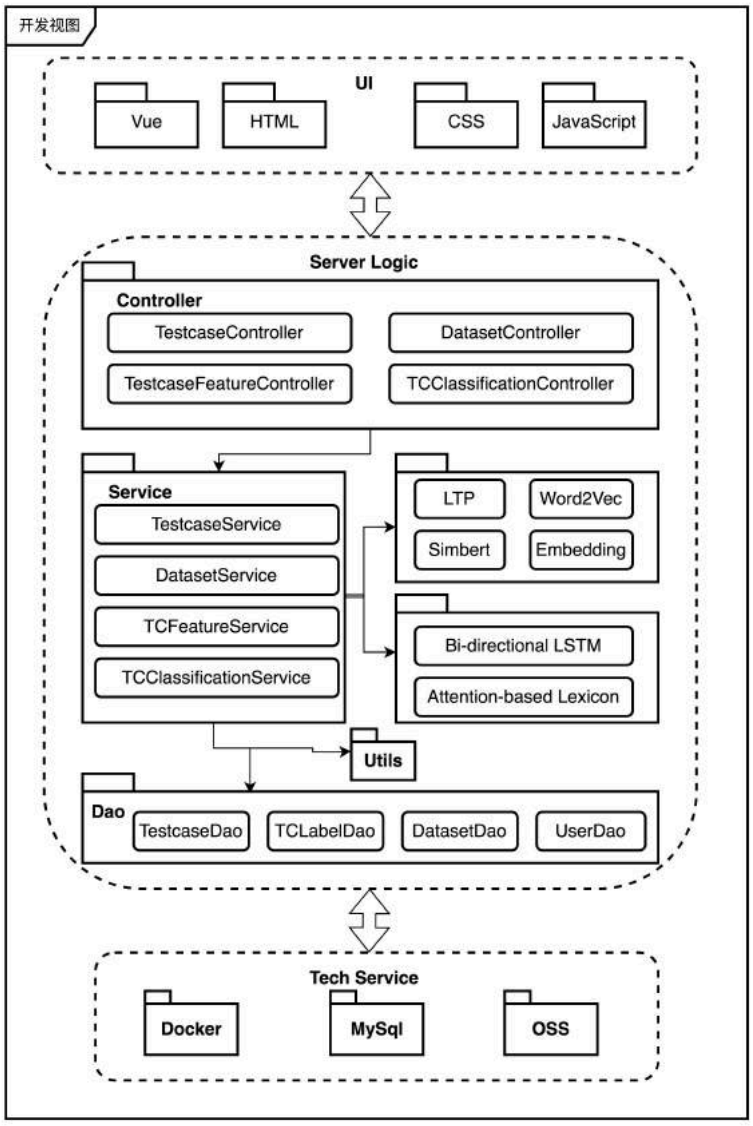

图 3.4: 开发视图

（2）开发视图

开发视图主要从开发者的角度表述软件系统架构，关注软件开发的静态组织结构，不仅包括要编写的源程序，还包括可以直接使用的第三方 SDK 和现成框架、类库，以及开发的系统将运行于其上的系统软件或中间件。本系统的开发视图如图3.4所示，主要分为三个部分，分别是 UI、Server Logic 以及 Tech Service。

UI 是展示层，由 Vue 文件、HTML 文件、CSS 文件、Javascript 文件、各方依赖包和用于配制或传输的 Json 文件组成。

Server Logic 描述了本系统中服务的逻辑，采用了分层架构，自上而下包括了 Controller 包、Service 包、Utils 包、Dao 包、Python 后端的服务包以及在本图中未展示出的表示对象实体的 Model 包或 DTO 包。其中，Controller 包主要负责接收和解析 http 请求，包含了 TestcaseController、TestcaseClassicifationController 等具体类，根据解析后的请求调用不同的 Service 包。Service 包是业务逻辑的构建器和服务的直接提供者，负责接收和存储上传的数据集、处理测试用例数据，通过调用 Dao 包、Utils 包和 Python 端服务包为 Controller 提供服务。Python 端服务包提供对中文文本数据的处理功能、测试用例的数据扩充功能、词袋构建和词向量转化功能与分类模型的构建、训练和使用，为 Service 包服务。Utils 包是工具类的集合，提供实体对象的格式转换与 Json 数据处理等常用的 JAVA 工具类。Dao 包提供了与底层数据库的连接与交互操作，包括对数据库对象的存储和增删改查功能，和 Model 包一起为 Service 包服务。

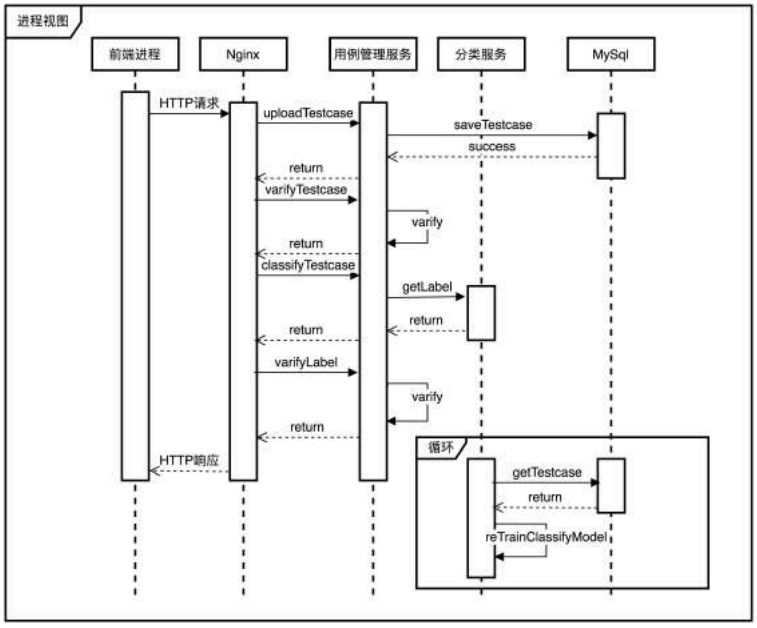

图 3.5: 进程视图

Tech Service 层描述了本系统中使用过的第三方工具和类库，包括容器技术 Docker、数据库 MySql、阿里云文件存储服务 OSS、前端组件库 Element UI。

（3）进程视图

进程视图主要从系统集成人员的角度表述软件系统架构，关注进程、线程相

关的并发、同步、通信等问题，描述系统内的请求。本系统的进程视图如图3.5所示。

用户在测试用例管理和分类系统前端进行 UI 操作，触发前端方法，发送 HTTP 请求至 Nginx 服务器，Nginx 根据配置解析收到的请求，根据自定义的负载均衡算法将请求发送至对应的服务端进程。测试用例管理和分类系统后端采用 MVC 设计模式，以 Controller 层接收请求，service 层处理请求。如果请求涉及对数据的增删改查，服务端进程会和 MySql 进程进行交互，对于部分持久化数据，MySql 进程会将数据持久化到数据库并进行备份。如果请求是关于生成测试用例标签分类，测试用例管理和分类服务端会以 Restful API 的形式请求分类服务进行处理。分类服务进程会每隔固定时间与 MySql 进程交互，读取存储在数据库中的历史测试用例数据，重新训练模型，保证分类的准确率。当测试用例管理和分类服务端处理完成业务请求后，将响应数据返回给 Nginx 服务器，Nginx 服务器再将响应数据返回给前端，前端拿到响应数据后解析数据并展示。

（4）物理视图

物理视图主要从运维人员的角度表述软件系统架构，主要关注系统安装、部署、物理节点之间的通信问题。本系统的物理视图如图3.6所示。

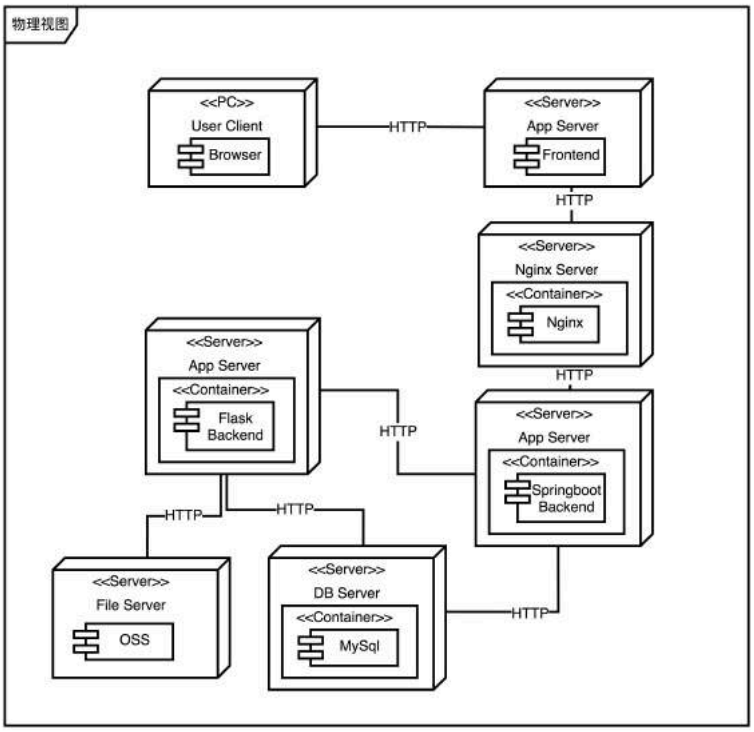

图 3.6: 物理视图

用户使用自己的计算机，通过浏览器客户端输入指定的 URL 即可访问测试用例管理和分类系统前端网页，前端以静态文件的形式部署在某一服务器上。用户对前端网页的操作请求将流向反向反向代理服务器，经过请求解析和负载均衡转发给 Springboot 服务端进行处理。Springboot 服务端通过 TCP 的方式访问 MySql 数据库，通过 HTTP 请求以 Restful API 的形式调用 Flask 服务端的接口提供分类服务，Flask 服务端通过 HTTP 请求将大文件存储在阿里云存储服务 OSS 中。本系统中的 Nginx、MySql 和后端服务都使用 Docker 容器部署运行，提供部署环境的一致性，提高部署效率，降低部署成本。

3.4 基于小样本学习的测试用例分类技术

针对目前众包测试等场景中收集到的测试用例需要分类、但实际上大多数测试用例缺乏分类标签的问题，本文从测试用例管理系统的角度出发，提出了基于小样本学习的测试用例分类技术，为众包测试中产生的测试用例提供分类服务。

基于小样本学习的测试用例分类技术主要由分类算法和类别词库收集算法两部分组成。分类算法为测试用例训练尽可能准确的分类器，向它们提供类别的预测。类别词库收集算法可以收集到每个测试用例类别的关键词，这些关键词或多或少在属于这个类别的测试用例中出现，一定程度上能够体现类别的通性特征。

测试人员在管理众包测试等场景收集到的测试用例的过程中，使用测试用例分类服务获取用例的类别标签，还能查看测试用例的关键词等辅助信息，从而更大程度上了解测试用例的相关信息，更好更准确的挑选测试用例，提高测试用例的复用能力，同时也减少了评审专家标记测试用例类别的工作量，解放生产力，满足业务在类似场景中的业务需求。

基于小样本学习的测试用例分类技术的具体流程如图3.7所示。首先，测试人员上传测试用例集，这一步主要由测试用例管理模块负责。系统定时获取历史测试用例数据和实时获取增量测试用例数据，构建分类模型的初始数据集，使用 Simbert 扩增数据、LTP 提取测试用例的文本特征词、训练 Word2Vec 模型并使用 Embedding 方法将由关键词文本转化为词语对应的词向量，这一步主要由测试用例特征提取模块负责。接着使用带有注意力机制的双向 LSTM 模型根据历史测试用例词向量的训练集和测试集训练初始分类器，获取测试用例的预测分类，这一部分由测试用例分类模块负责。最后使用类别词库收集算法，根据第三部分的初始分类器收集测试用例词向量发展集中的类别词语，使用分类器

和模式匹配的方式并行预测发展集中测试用例的类别，将类别伪标签赋予给发展集中的测试用例，将其从不带标签的发展集数据转化为带标签的训练集数据，重新利用扩大后的训练集优化初始分类器，这一部分由类别词库收集模块负责。

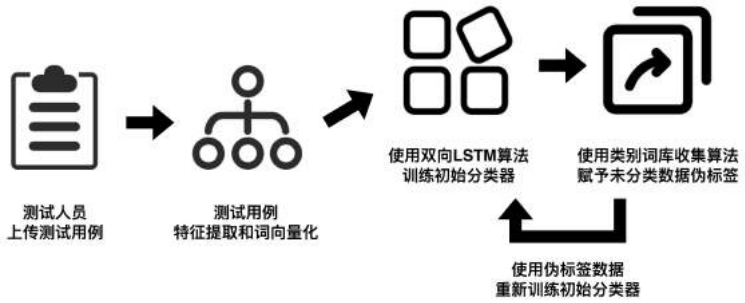

图 3.7: 基于小样本学习的测试用例分类技术流程图

3.4.1 分类算法

本系统选取双向 LSTM+Attention 模型作为测试用例文本的分类算法。模型的训练输入为特征处理模块输出的测试用例词向量训练集，模型网络由输入层、Embedding 层、LSTM 层（线性变换层、使用 ReLu 激活函数的激活层）、注意力层、输出层组成，模型输出为每条词向量的预测类别、模型预测该类别的置信度、这条词向量中每个词的注意力权重。

双向 LSTM 全称 Bi-directional Long Short Term Memory，双向长短期记忆网络，是一种卷积神经网络的特殊类型，更是一种特殊的循环神经网络，是 LSTM 长短期记忆网络的变体。图3.8是双向长短期记忆网络模型的示意图。LSTM 引入了遗忘控制门、输入控制门和输出控制门的概念，通过门控状态来控制传输状态，记住需要长时间记忆的前面若干输入的信息，忘记前面若干输入中不重要的信息，主要解决了长序列训练过程中的梯度消失和梯度爆炸问题。相对于普通的 RNN，LSTM 能够在更长的序列中有更好的表现。然而有些时候预测可能需要由前面若干输入和后面若干输入共同决定，双向 LSTM 在 LSTM 中 Forward 层的基础上增加了一层 Backward 层，保留了后面若干输入的重要信息，这样网络的预测结果会更加准确。本模型选取预测概率最大的类别作为被预测测试用例的类别标签，同时将这个预测概率称为模型预测该类别的置信度。

Attention 机制模仿了生物观察行为的内部过程，人的视觉在处理一张图片时，会通过快速扫描全局图像，获得需要重点关注的目标区域，也就是注意力焦点。然后对这一区域投入更多的注意力资源，以获得更多所需要关注的目标的细节信息，并抑制其它无用信息。在本文的分类模型中，注意力机制通过为

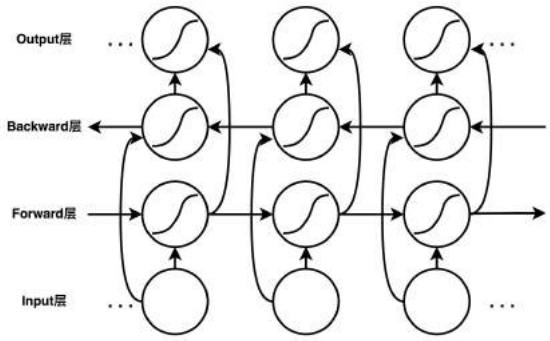

图 3.8: 双向长短期记忆网络模型示意图

每个词获取一个权重以保证当前识别的词对原测试用例中各个词的关注点不同，同时使用 softmax 函数对各词的注意力权重归一化，表示测试用例词向量里的每个维度（每个词）在这次预测类别中的重要程度。将 LSTM 层输入的向量集合表示为 $H[h_1, h_2, \cdots, h_T]$。其 Attention 层得到的权重矩阵 r 由下面的方式得到：

$$M = tanh(H) \tag{3.1}$$

$$\alpha = softmax(w^T M) \tag{3.2}$$

$$r = H^T \tag{3.3}$$

其中，$H \in R^{d^w T}$，d^w 为词向量的维度，w^T 是一个训练学习得到的参数向量的转置。

3.4.2 类别词库收集算法

本系统使用一种半监督的自学习框架进行小样本分类，在训练中同时使用少量带标签数据和大量无标签数据。通过使用基于 LSTM 训练出的分类器中的注意力权重来获得更大的可靠训练集，将无标签的样本转化为带有算法赋予的伪标签的样本当作训练集使用。

算法的大致流程是首先从带标签的数据集中训练一个基于带 Attention 机制的 LSTM 的初始分类器，这一步从测试用例分类模块实现。然后，从无标签数据集中通过初始分类器的注意力机制为每个类别收集一组词称为词库，词库中的词是每个类别的代表词集。最后，使用初始分类器和基于词库的模式匹配规则共同标记无标签样本，最终使用扩大后的带标签数据集训练出新分类器供分类服务使用。

根据以往的研究，重要的词在分类中往往具有较高的注意力权重，注意力机制也为能够代表输入数据类别的词分配更高的权重。即使初始分类器的性能很差，分类器也会对预测的数据中具有高置信度的重要词汇分配更高的权重。因

此本算法将深度学习分类器和词库匹配结合起来，可以解决一定程度上的标注数据稀少的问题。

算法的具体流程如下：

（1）从带标签的测试用例集中创建初始分类器，该测试用例集的数据量非常少（小样本）。该分类器必须包括注意力机制以收集每个分类的关键词。

（2）在无标签测试用例集 D 上重新运行具有注意力机制的初始分类器。

（3）获得一组 D 中测试用例数据的关键词，充当每个预测类别的代表词集（词库）。

（4）使用初始分类器和词库预测 D 中测试用例的标签。

（5）将新的标签测试用例加入训练集，再次训练分类器，流程跳转至步骤（1）。

（6）重复上述过程，直到不再有伪标签测试用例被添加到训练集中。

基于词库的模式匹配规则如下，其中 t1、t2 是当前测试用例在词库中匹配到的词的数量且 t1<t2：

（1）如果分类器以高置信度预测了无标签的测试用例，并且词库中至少有 t1 个匹配词，那就根据分类器的预测来标记测试用例。

（2）如果词库中至少有 t2 个匹配词，那就根据词库的预测对数据进行标注。

3.5 测试用例管理模块流程设计

测试用例管理模块的主要功能是上传测试用例数据、生成测试用例标签、审核测试用例和用例标签，本模块直接与用户操作界面交互，流程如图3.9所示。本模块的主要步骤如下：

（1）用户使用账户登录系统，系统验证账户及其权限，验证不通过则返回账户登录步骤。

（2）登录验证通过后，用户上传测试用例，此时系统保存测试用例并等待验证用例审核，审核不通过则返回上传测试用例步骤并更改数据库中测试用例状态。

（3）用例审核通过后，用户点击生成用例标签按钮获取用例的类别，此时新生成的用例标签等待审核。

（4）标签审核通过后，系统保存标签信息并进入第（5）步。

（5）用例审核通过后，用户挑选并导出测试用例，流程结束。

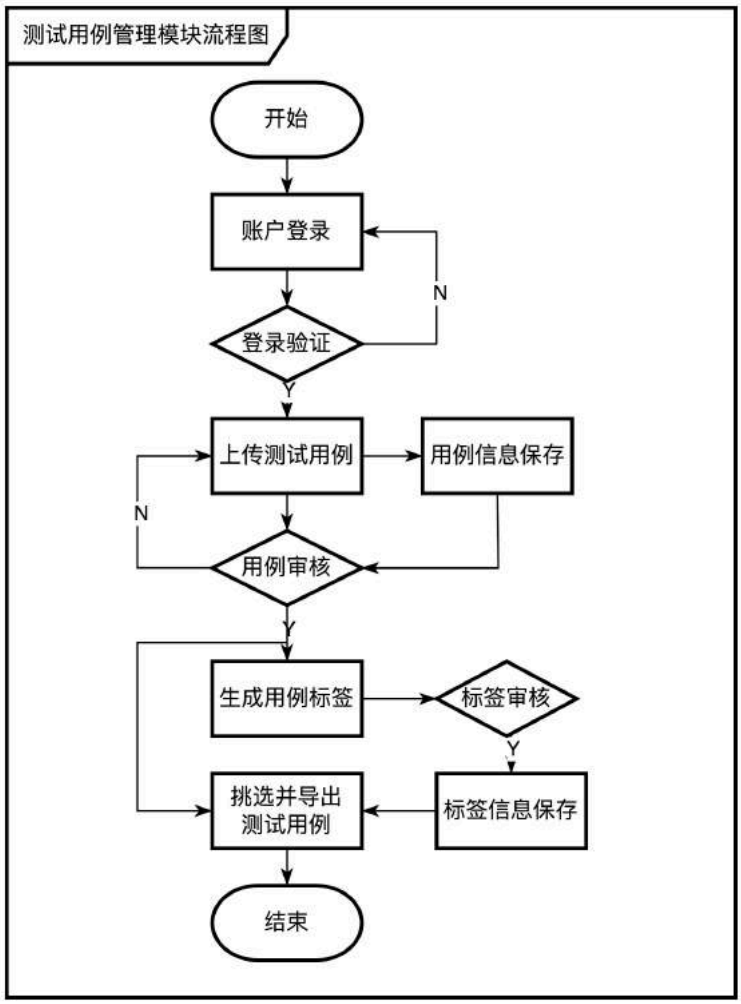

图 3.9: 测试用例管理模块流程图

3.6 测试用例特征提取模块流程设计

测试用例特征提取模块的主要功能是对测试用例管理模块获取的测试用例数据进行处理，将处理后的中间数据保存在文件系统中。获取到的原始测试用例数据难以被计算机理解，因此需要经过各种条件筛选出脏数据后提取测试用例特征再供分类模型使用。对于存量测试用例，本模块首先用它们做一轮数据扩增以丰富数据集，接着使用 LTP 工具、Word2Vec 方法、Word Embedding 方法将它们转化为词向量供分类模型训练。对于增量测试用例，本模块直接将它们转化为词向量供输入分类模型获得分类结果。本模块流程如图3.10所示，主要步骤如下:

（1）通过 tcp 请求连接 MySql 数据库获取历史测试用例数据。

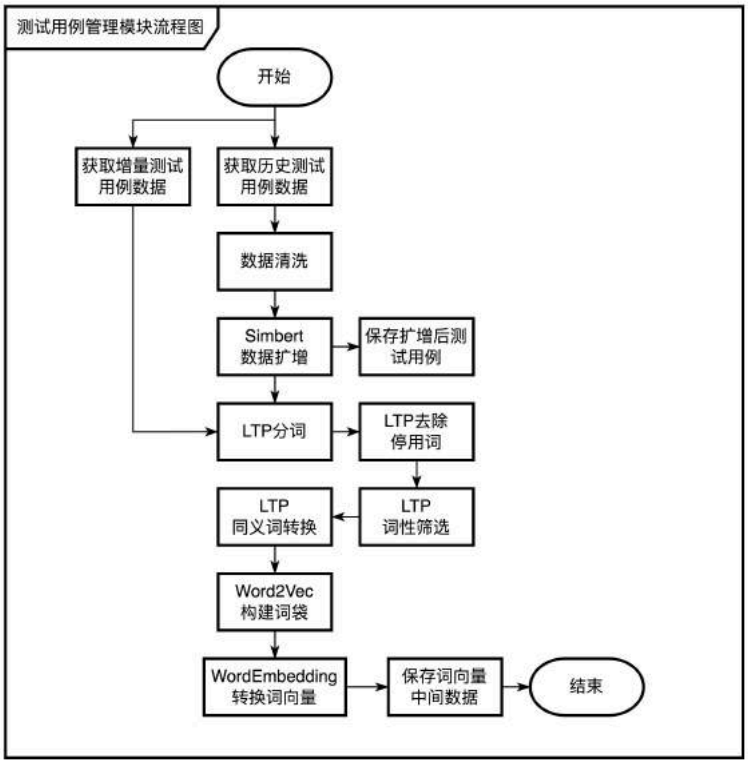

图 3.10: 测试用例特征提取模块流程图

（2）对测试用例数据进行筛选处理，如过滤用例类别异常或测试过程、测试需求为空的测试用例、删除用例状态为等待审核的测试用例。

（3）使用 Simbert 模型生成与测试用例语义相似的文本，扩充用例数量，方便后续训练神经网络，保存扩充后的测试用例文本至文件系统。

（4）使用语言技术平台 LTP 对测试用例文本数据进行分词、去除停用词操作，过滤词性不是动词或名词的词汇，统一同义词方便特征提取。

（5）训练关于历史测试用例数据的 Word2Vec 模型，构建词袋并保存。

（6）使用 nn.embedding() 获取测试用例关于 Word2Vec 模型的词嵌入向量，保存测试用例对应的词嵌入向量至文件系统，流程结束。

（7）通过接收测试用例管理模块的 http 请求获取增量测试用例数据，跳转至步骤（4）。

3.7 测试用例分类模块流程设计

测试用例分类模块的主要功能是使用从测试用例特征提取模块获得的历史测试用例词向量训练双向 LSTM 神经网络模型当作测试用例的初始分类模型，

使用从测试用例特征提取模块获得的增量测试用例词向量作为初始分类模型的输入，得到模型的输出为该增量测试用例对应的类别。本模块流程如图3.11所示，主要步骤如下：

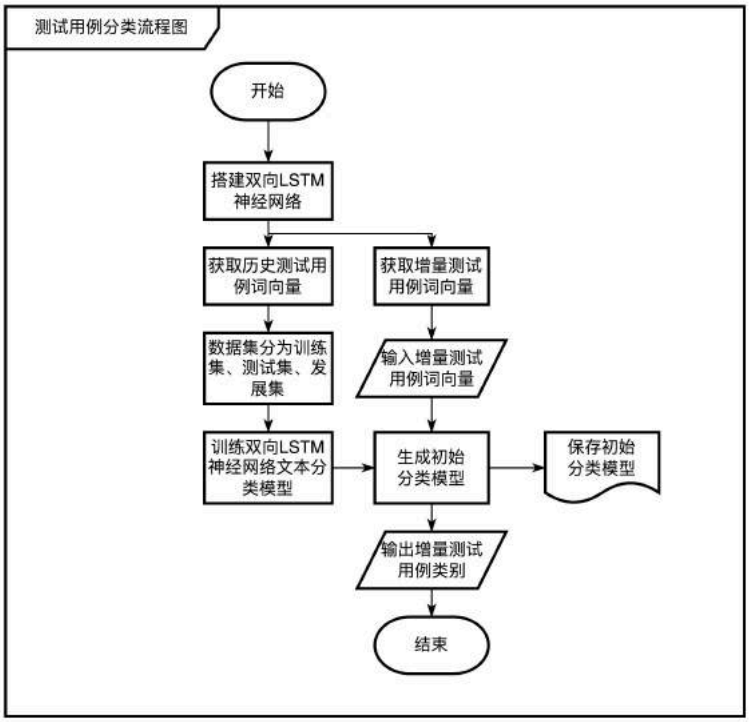

图 3.11: 测试用例分类模块流程图

（1）使用 PyTorch 开源机器学习库搭建双向长短期记忆神经网络，定义输入层、隐藏层、模型的前向计算。

（2）在文件系统中获取历史测试用例数据的词向量数据集，将数据集分为由带标签测试用例词向量组成的训练集、测试集与由无标签测试用例词向量组成的发展集，其中发展集供类别词库收集模块使用。

（3）使用训练集经过多轮训练得到在测试集上分类准确率最高的初始分类模型，模型保存至阿里云 OSS 中。

（4）从测试用例特征提取模块获得增量测试用例词向量作为初始分类模型的输入，输出初始分类模型对该增量数据预测的类别并记录预测置信度，流程结束。

3.8 类别词库收集模块流程设计

类别词库收集模块的主要功能是在双向 LSTM 神经网络中增加 Attention 机制，可视化注意力层，关注具体测试用例中的词在分类模型计算时的注意力权

重，为测试用例的每个类别收集代表词汇，在初始分类模型预测置信度较低的情况下通过模式匹配方法判断测试用例类别，提高分类准确率。本模块流程如图3.12所示，主要步骤如下：

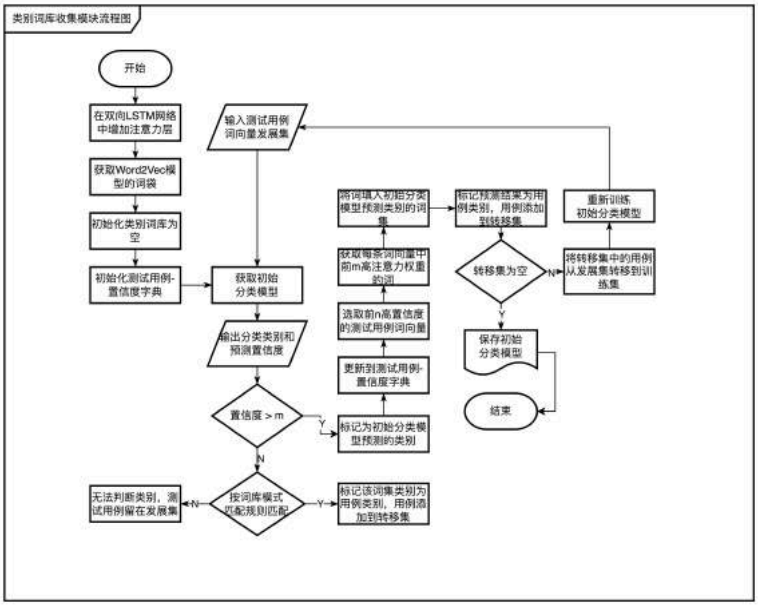

图 3.12: 类别词库收集模块流程图

（1）使用 PyTorch 开源机器学习库在双向长短期记忆神经网络中增加注意力层，希望获取词向量的注意力权重。

（2）获取测试用例特征提取模块中保存的 Word2Vec 模型的词袋，获取词袋中词向量和真实词汇的对应关系。

（3）初始化类别词库，每个类别的词集均初始化为空。初始化测试用例-置信度字典为空，后续该字典按照置信度高低从高到低排序。

（4）获取测试用例分类模块中保存的初始分类模型，获取测试用例分类模块中分配的测试用例词向量发展集。

（5）将发展集作为初始分类模型的输入，分类模型为发展集中测试用例一一预测类别，获取分类类别和预测置信度。

（6）当预测置信度大于等于 p 时，将该条测试用例类别标记为初始分类模型预测的类别，将它加入到测试用例-置信度字典中。

（7）选取测试用例-置信度字典中前 n 高置信度的测试用例词向量，结合词袋挑选出每条词向量中注意力权重前 m 高的词，将这些词加入该条词向量对应的类别的词集中。

（8）将获取到预测类别的词向量加入到转移集。

（9）如果转移集不为空，将转移集中的用例词向量丛发展集转移到训练集，根据训练集重新训练初始分类模型，接着跳转至步骤（5）。

（10）如果转移集为空，保存当前的初始分类模型为最终分类模型，流程结束。

（11）当预测置信度小于 p 时，按照模式匹配规则为词向量在词集中匹配词，根据匹配到词的数量预测当前词向量的类别，跳转至步骤（8）。

（12）如果按照模式匹配规则仍然不能预测当前词向量的类别，则该词向量仍保留在发展集中。跳转至步骤（9）。

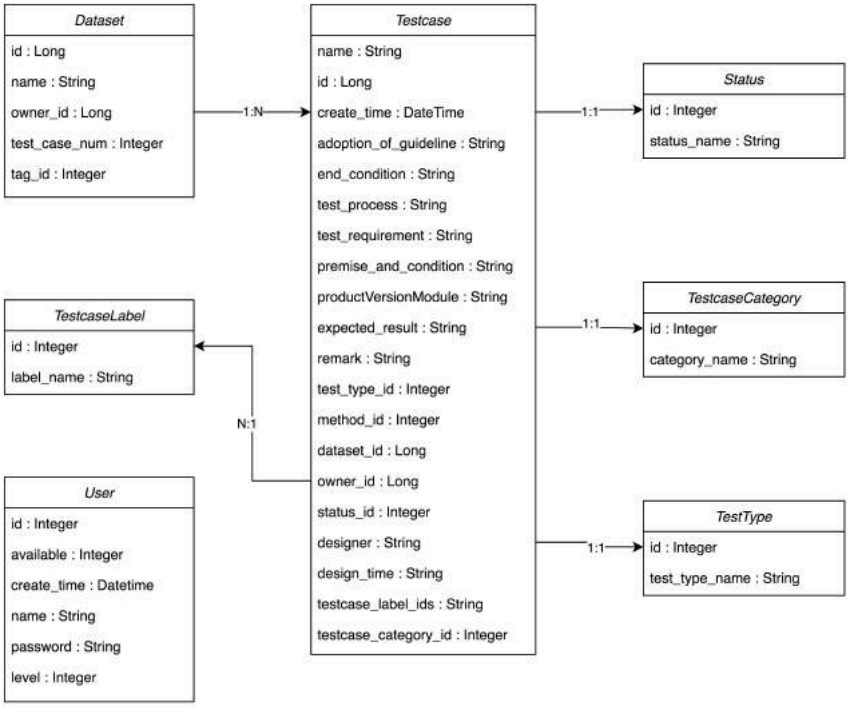

图 3.13: 实体类设计图

3.9 数据模型设计

在基于小样本学习的测试用例分类系统中，重要的实体类包括测试用例类 Testcase、数据集类 Dataset、测试类型类 TestType、测试用例状态类 Tcstatus、测试用例类型类 TcCategory、测试用例标签类 TcLabel 和用户类 User 等，实体类的设计和联系如图3.13所示。一个数据集内可以包含 0 到 N 条测试用例，对于每条测试用例，都有对应的一种测试类型、一种测试用例状态、一种测试用例类型和 0 到 N 条测试用例标签。

Testcase 类：Testcase 类存储测试用例的相关信息。测试人员通过上传数据集或填写测试用例属性上传测试用例，经过评审专家审核后测试人员可查看测试用例列表。Testcase 类具体内容如表3.12所示。

表 3.12: Testcase 类详情列表

字段	类型	含义
id	Integer	testcase 的唯一标识，标识唯一确定的测试用例
name	String	测试用例名称
create_time	Datetime	测试用例上传日期
adoption_of_gudeline	String	评估准则
end_condition	String	终止条件
test_process	String	测试过程
test_requirement	String	测试需求
premise_and_condition	String	前提和约束
product_version_module	String	产品版本模块
expected_result	String	预期结果
remark	String	备注
test_type_id	Integer	测试类型 id
method	Integer	测试方法，黑盒测试 0 白盒测试 1
dataset_id	Long	所属数据集 id
owner_id	Long	上传者 id
status_id	Integer	测试用例状态 id
designer	String	测试用例设计者
design_time	String	测试用例设计日期
testcase_label_ids	String	测试用例标签集
testcase_category_id	String	测试用例类型 id

TestType 类：TestType 类存储测试用例的测试类型，当前已存储的测试类型有 23 种，分别为：文档审查、代码审查、静态分析、代码走查、逻辑测试、功能测试、性能测试、接口测试、人机交互界面测试、强度测试、余量测试、可靠性测试、安全性测试、恢复性测试、边界测试、数据处理测试、安装性测试、容量测试、互操作性测试、敏感性测试、标准符合性测试、兼容性测试、中文本地化测试。TestType 类具体内容如表3.13所示。

表 3.13: TestType 类详情列表

字段	类型	含义
id	Long	testtype 的唯一标识，标识唯一确定的测试类型
name	String	测试类型名称
create_time	Datetime	测试类型创建日期

Dataset 类：Dataset 类存储数据集的相关信息，每个数据集对应上传的 Excel 文件，Excel 文件第一行的每列代表测试用例的属性，Excel 文件每一列代表一条测试用例。测试人员上传数据集，系统存储数据集中的测试用例数据，同时存储数据集信息。Dataset 类具体内容如表3.14所示。

表 3.14: Dataset 类详情列表

字段	类型	含义
id	Long	dataset 的唯一标识，标识唯一确定的数据集
name	String	数据集名称，Excel 文件的名称
create_time	Datetime	数据集上传日期
owner_id	Long	数据集上传人员 id
tag_id	Long	数据集标签
test_case_num	Integer	数据集包含的测试用例数目

Tcstatus 类：Tcstatus 类存储测试用例的状态信息。每条测试用例在某一时段都有且仅有唯一的状态，不同状态之间可以相互转换。当前定义的测试用例状态共有 5 种，分别为：等待审核、标签等待审核、审核不通过、标签审核不通过、审核通过。测试用例最先被测试人员上传后首先处于“等待审核”状态，经过评审专家的审核可以转化为“审核通过”或“审核不通过”状态。当测试用例处于“审核通过”状态时，测试人员可以点击生成标签按钮使测试用例转化为“标签等待审核”状态，再次经过评审专家的审核可以转化为“审核通过”或“标签审核不通过”状态。Tcstatus 类具体内容如表3.15所示。

表 3.15: Tcstatus 类详情列表

字段	类型	含义
id	Long	tcstatus 的唯一标识，标识唯一确定的测试用例状态
tcstatus_name	String	测试用例状态名称
create_time	Datetime	测试用例状态创建日期

Tclabel 类：Tclabel 类存储测试用例的标签信息。每条测试用例在生成标签后都可以拥有 0 到 N 个标签，出现过的标签会存储在数据库中，方便后续对标签个数、标签对应的测试用例个数的统计。Tclabel 类具体内容如表3.16所示。

Tccategory 类：Tccategory 类存储测试用例的类别信息。每条测试用例都有且仅有一种它希望测试的场景，我们称之为测试用例类别。有些测试用例上传时没有标注类别，测试用例管理和分类系统调用分类服务为这些没有标注类别的测试用例分类并将分类情况保存。当前定义了 6 种测试用例类别，分别是：不

表 3.16: Tclabel 类详情列表

字段	类型	含义
id	Long	tclabel 的唯一标识，标识唯一确定的测试用例标签
tclabel_name	String	测试用例标签名称
create_time	Datetime	测试用例标签创建日期

表 3.17: Tccategory 类详情列表

字段	类型	含义
id	Long	tccategory 的唯一标识，标识唯一确定的测试用例类别
tccategory_name	String	测试用例类别名称
create_time	Datetime	测试用例类别创建日期

正常退出、功能不完整、用户体验、页面布局缺陷、性能、安全。Tccategory 类具体内容如表3.17所示。

表 3.18: User 类详情列表

字段	类型	含义
id	Long	user 的唯一标识，标识唯一确定的账户
name	String	账户名
password	String	账户密码
level	Integer	账户类型
availability	Integer	账户可用性
online	Integer	账户登录情况
create_time	Datetime	账户创建日期

User 类：User 类存储本系统的账户相关信息。终端用户使用账户登录本系统，本系统的账户类型有三种，使用 level 字段表示，0 表示管理员账户、1 表示评审专家账户、2 表示测试人员账户。availability 字段表示了账户当前的使用情况，0 表示被禁用、1 表示可用。User 类具体内容如表3.18所示。

考虑到系统需求和以上实体类设计，测试用例管理系统和分类服务需要对部分数据对象进行持久化处理，主要包括了测试用例及其派生出的一系列对象。本系统的服务端采用关系型数据库 MySql 实现数据的持久化，存储业务数据，本系统的数据库 ER 图如图3.14所示。

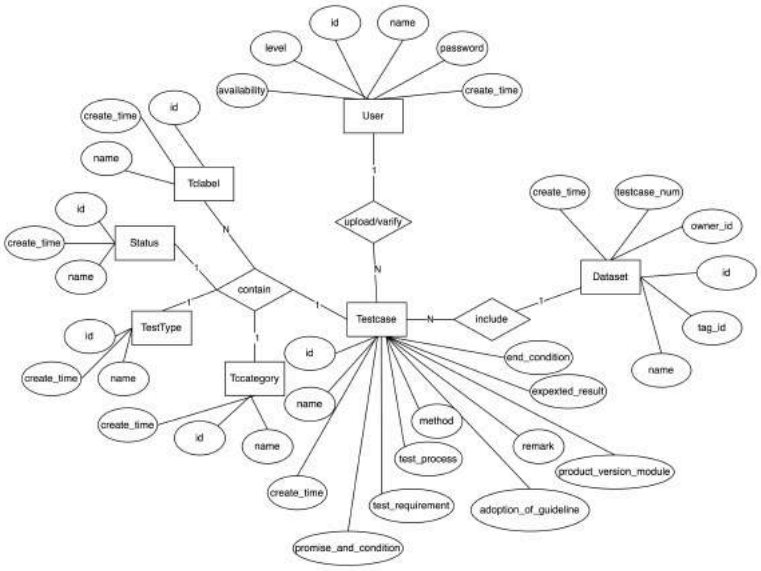

图 3.14: 数据库 ER 图

3.10 本章小结

本章是系统需求分析和概要设计过程的文字表征。首先介绍了测试用例管理和分类系统的整体情况，然后对系统进行整体的需求分析工作，完成了系统的涉众分析，功能需求和非功能需求的分析归纳，并通过系统用例图和系统用例描述进行阐述。其次，介绍了系统的总体设计架构，将系统模块划分为测试用例管理模块、测试用例特征提取模块、测试用例分类模块和类别词库收集模块四个模块。接着，使用“4+1”视图从不同角度对系统的架构和实现进行了整体概要设计，使用流程图和公式对本系统使用的分类算法和类别词库收集算法进行了说明和设计，使用流程图对系统的四个模块进行了流程上的简要说明。最后使用类图、ER 图设计了系统的数据模型。为下一章的代码实现提供了坚实的技术基础。

第四章　系统详细设计与具体实现

4.1　测试用例管理模块详细设计与实现

4.1.1　测试用例管理模块核心类图

测试用例管理模块核心类图如图4.1所示。测试用例构建模块的核心是测试人员上传测试用例并生成类别标签和评审专家审核，系统更新并保存测试用例状态，因此分为测试用例管理和标签管理两部分。

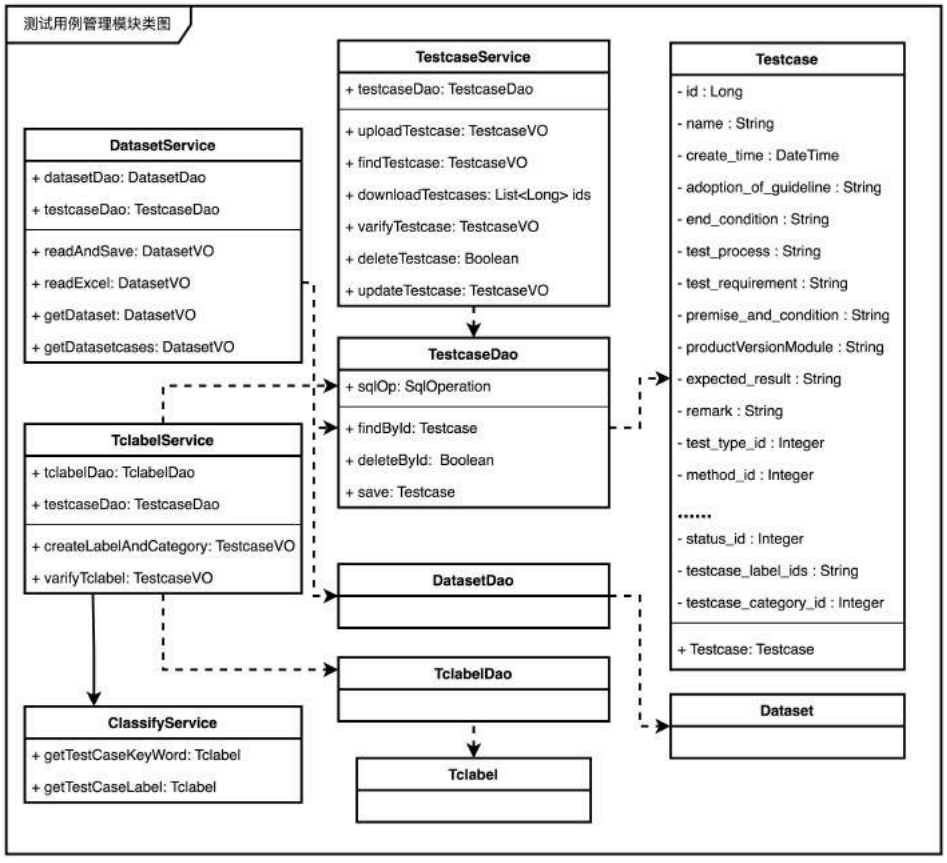

图 4.1: 测试用例管理模块核心类图

测试用例管理部分包含了 TestcaseService 类和 DatasetService 类两个服务。TestcaseService 类对外提供五个接口，包括上传单个测试用例（uploadTestcase）、查看测试用例详情（findTestcase）、下载测试用例集（downloadTestcases）、审核测试用例（varifyTestcase）、删除测试用例（deleteTestcase）和更新测试用例

（updateTestcase）。用户在 UI 界面填写测试用例相关信息上传单个测试用例，系统使用 TestcaseDao 调用 save 接口将数据保存至 MySql 数据库。用户选取多条测试用例下载，系统调用 downloadTestcases 接口使用 Dao 层的 findById 接口获取测试用例信息并下载至定义好的 Excel 文件模版中。用户删除某一测试用例，系统调用 deleteTestcase 接口使用 Dao 层的 deleteById 接口删除数据库中的信息。

DatasetService 类对外提供四个接口，包括读取 Excel 中的测试用例信息并存储至数据库中（readAndSave）、不同 Excel 版本的读取函数（readExcel）、获取数据集信息（getDataset）、获取数据集中的所有测试用例（getDatasetcases）。DatasetDao 提供数据集查找（findById）、更新（update）和删除（deleteById）方法来支持 DatasetService。用户在 UI 界面上传符合规定格式的 Excel 文件，系统调用 readAndSave 接口创建数据集对象读取 Excel 文件并获取测试用例存储至数据库。

标签管理部分包含了 TclabelService 类，对外提供审核测试用例标签（varifyTclabel）和生成分类标签（createLabelAndCategory）接口。TclabelService 依赖 Python 端的 ClassifyService 类提供的 getTestcaseKeyWord 和 getTestcaseLabel 方法，ClassifyService 调用其他模块训练出来的分类模型提供预测的测试用例分类标签。

图中的 Testcase、Dataset、Tclabel 均为实体类，分别表示测试用例、数据集、测试用例标签。

4.1.2 测试用例管理模块顺序图

测试用例管理模块的顺序图如图4.2所示。测试人员点击上传数据集按钮后，出发前端的 uploadDataset 方法，向 Java 后端发送 POST 请求，传输符合预定义格式的 Excel 文件，DatasetController 的 uploadByExcel 方法获取传输来的 Excel 文件，先调用 DatasetService 的 readExcel 方法判断 Excel 文件的版本，该方法调用同类中的 readAndSave 方法执行 Excel 文件的具体解析操作，将 Excel 文件映射成一个数据集对象、文件中的行映射成一条测试用例对象，分别调用 TestcaseDao 和 DatasetDao 的 save 方法将实体对象存储至 MySql 数据库中。测试人员点击查看数据集列表后，DatasetController 的 getDatasetList 方法接收并解析前端传来的 GET 请求，调用 DatasetService 的 getDatasetList 方法，该方法调用 DatasetDao 的 findAll 方法查找并返回全部的数据集信息。测试人员点击查看某一数据集的具体相关信息，DatasetController 调用 DatasetService 的 getDataset 方法，该方法调用 DatasetDao 的 findById 方法根据数据集的 id 查找返回某一数据集的全部信息。当测试人员想要查看数据集内包含的所有测试用例时，点击某一数据集

条目，前端跳转至该数据集的测试用例列表界面，此时调用 Java 后端 DatasetController 的 getCaseList 方法，该方法调用 DatasetService 的 getDatasetcases 方法，getDatasetcases 方法根据数据集的 id 使用 TestcaseDao 的 findByDatasetId 方法在数据库的测试用例表中查找属于这个数据集的所有测试用例信息并返回。

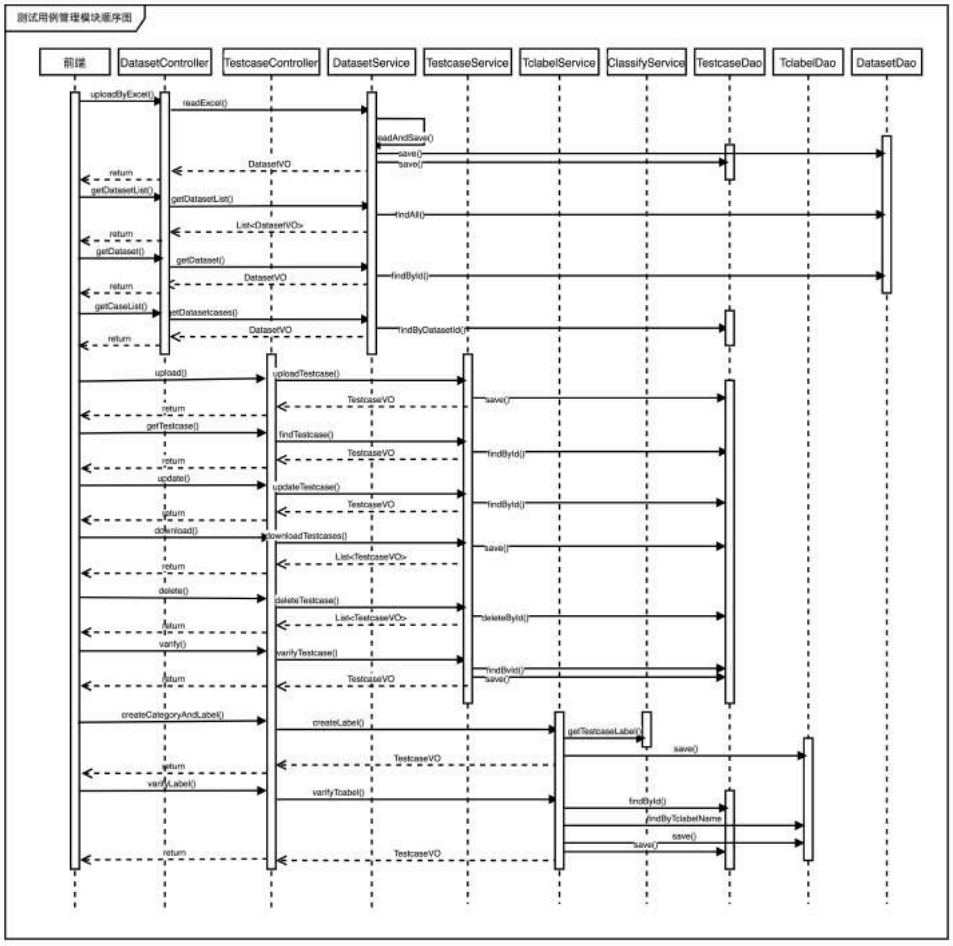

图 4.2: 测试用例管理模块顺序图

测试人员填写测试用例相关信息后，点击上传按钮上传单条测试用例信息，前端调用 Java 后端 TestcaseController 的 upload 方法解析 POST 请求，依次调用 TestcaseService 的 uploadTestcase 方法、TestcaseDao 的 save 方法在校对测试用例的基本信息正确后将单条测试用例存储至数据库中。类似的，前端调用 TestcaseController 的 getTestcase 方法查看某条测试用例的具体信息、调用它的 update 方法更新某条测试用例的具体信息。

当测试人员选中多条测试用例并点击导出按钮时，前端发送带有被选中测试用例 id 列表的 POST 请求，调用 Java 后端 TestcaseController 的 download 方法，

该方法调用 TestcaseService 层的 downloadTestcases 方法，首先生成一个 Excel 文件对象，接着设置文件中第一行的测试用例属性说明，最后循环调用 TestcaseDao 的 findById 方法逐个将测试用例填充进 Excel 文件的每一行中并返回文件流给前端，前端浏览器此时自动下载对应的 Excel 文件到本地。

测试人员点击测试用例详情页的标签生成按钮，前端发送 GET 请求调用 Java 后端 Testcasecontroller 的 createCategoryAndLabel 方法，该方法调用 TclabelService 的 createLabel 方法，createLabel 中使用 restTemplate 类向测试用例分类服务发送 POST 请求获取测试用例的类别并返回。

评审专家可以登录评审账号审核测试用例和用例标签。在审核测试用例时，评审专家点击通过或不通过按钮，前端调用 Testcasecontroller 的 varify 方法进而调用 TestcaseService 的 varifyTestcase 方法，varifyTestcase 首先在数据库中查找该测试用例是否存在，以免由于并发而被删除，接着修改测试用例的状态，保存并返回。同样的，评审专家在评审用例标签通过时，Java 后端的 Testcasecontroller 通过 varifyLabel 方法调用 TclabelService 的 varifyTclabel 方法，首先检查测试用例是否存在，接着检查测试用例的状态是否为评审通过，如果前提都正确，varifyTclabel 将通过 TclabelDao 存储新的标签信息并更新测试用例状态，否则返回异常并不予以评审通过标签。

4.1.3 测试用例管理模块的实现

```
public DatasetVO readExcel(String name, long ownerId, MultipartFile file, long tagId) throws IOException {
    DatasetVO datasetVO = null;
    log.info("before 判断是 xls 文件还是 excel 文件");
    if(file.getOriginalFilename().split("\\.")[1].equals("xls")) {
        log.info("before 读取 xls 文件");
        datasetVO = readAndSave(name, ownerId, tagId, file, true);
    } else if(file.getOriginalFilename().split("\\.")[1].equals("xlsx")) {
        log.info("before 读取 xlsx 文件");
        datasetVO = readAndSave(name, ownerId, tagId, file, false);
    }
    return datasetVO;
}
```

图 4.3: 上传数据集关键代码图

图4.3是测试用例管理模块中数据集上传功能的关键代码。readExcel 函数的输入参数是命名的数据集名称、上传者 id、上传文件标签和文件本身 MultipartFile 类的对象。首先使用 getOriginalFilename 方法获取上传文件的文件名，接着使用字符串的 split 方法分割文件名，获取文件名的后缀以判断 Excel 文件的版本，根据版本选择相应的处理函数解析 Excel 文件以生成新的数据集对象。

```
public TestCaseVO createTclabelAndCategory(long id) {
        Optional<TestCase> testCaseOptional = testCaseRepository.findById(id);
        if (!testCaseOptional.isPresent()) {
            log.info("当前 id 的测试用例在数据库中不存在！ ");
            return null;
        }
        TestCase testCase = testCaseOptional.get();
        LabelGenerationRequestDTO labelGenerationRequestDTO = new LabelGenerationRequestDTO();
        labelGenerationRequestDTO.setTestProcess(testCase.getTestProcess());
        labelGenerationRequestDTO.setTestRequirement(testCase.getTestRequirement());
        labelGenerationRequestDTO.setProductVersionModule(testCase.getProductVersionModule());

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<LabelGenerationRequestDTO> httpEntity = new HttpEntity<>(labelGenerationRequestDTO,
headers);
        ResponseEntity<String> responseEntity = null;
        try {
            log.info("分类服务的 url 地址为：" + labelGenerationUrl);
            responseEntity = restTemplate.postForEntity(labelGenerationUrl + "get_category", httpEntity,
String.class);
        } catch(RestClientException e){
            log.info("上游测试用例分类系统 在 restTemplate 的过程中出错");
            return null;
        }
        if(responseEntity.getBody() == null) {
            log.info("上游测试用例分类系统 返回值为空");
            return null;
        }

        LabelGenerationResponseDTO responseDTO = new Gson().fromJson(responseEntity.getBody(), new
TypeToken<LabelGenerationResponseDTO>(){}.getType());
        log.info("上游测试用例分类系统返回值为: {}", responseDTO);
        String categoryName = responseDTO.getCategory();
        testCase.setTccategoryId(tccategoryRespository.findByTccategoryName(categoryName).getId());
        String labels = "";
        for(String keyword: responseDTO.getKeyword()) {
            labels = keyword + "&&";
        }
        labels = labels.substring(0,labels.length() - 2);
        testCase.setTclabels(labels);
        TestCaseVO testCaseVO= new TestCaseVO(testCase);
        return testCaseVO;
}
```

图 4.4: 标签生成关键代码图

图4.4是测试用例管理模块中生成测试用例标签功能的关键代码。createTcLabelAndCategory 函数实现了测试用例标签生辰的功能，输入参数是需要生成分类标签的测试用例 id。首先调用 testcaseDao 层的 findById 方法，确定想要获取分类标签的测试用例数据存在于数据库中。接着新建 LabelGenerationRequestDTO 对象，该对象中存放着 Python 端分类服务需要获取的测试用例分类的依据信息。代码中使用 RestTemplate 包支持 Java 后端发起 HTTP 请求，新建 HttpHeader 对象并设置 HTTP 请求首部格式，设置 HTTP 请求的 body 部分为 LabelGenerationRequestDTO 对象的 Json 格式，调用 RestTemplate 包的 postForEntity 方法向分类服务的 URL 发送 POST 请求调用它的 get_category 方法。代码随后使

用 ResponseEntity 类获取分类服务的响应，判断了是否出现 RestClientException 异常以及响应返回的主体部分是否为空。最后使用 LabelGenerationResponseDTO 承接分类服务返回的数据，获取新生成的分类标签和关键词返回给前端。

```
public HSSFWorkbook downLoadTestCases(List<Long> ids) {
        HSSFWorkbook workbook = new HSSFWorkbook();
        HSSFSheet sheet = workbook.createSheet("测试用例批量导出");
        HSSFRow row = sheet.createRow(0);
        for(int i=0;i<TESTCASE_OUTPUT_HEADERS.length;i++){
            HSSFCell cell = row.createCell(i);
            HSSFRichTextString text = new HSSFRichTextString(TESTCASE_OUTPUT_HEADERS[i]);
            cell.setCellValue(text);
        }

        int rowNum = 1;
        for(int i = 0; i < ids.size(); i++) {
            Optional<TestCase> testCaseOptional = testCaseRepository.findById(ids.get(i));
            if(!testCaseOptional.isPresent()) {
                continue;
            }
            TestCase testCase = testCaseOptional.get();
            HSSFRow row1 = sheet.createRow(rowNum);
            row1.createCell(0).setCellValue(testCase.getName());
            row1.createCell(2).setCellValue(testCase.getTestType());
            row1.createCell(3).setCellValue(testCase.getProductVersionModule());
            row1.createCell(4).setCellValue(testCase.getMethod());
            row1.createCell(5).setCellValue(testCase.getDesigner());
            row1.createCell(6).setCellValue(testCase.getDesignTime());
            row1.createCell(7).setCellValue(testCase.getPremiseAndConstraint());
            row1.createCell(8).setCellValue(testCase.getEndCondition());
            row1.createCell(9).setCellValue(testCase.getTestRequirement());
            row1.createCell(10).setCellValue(testCase.getTestProcess());
            row1.createCell(11).setCellValue(testCase.getExpectedResult());
            row1.createCell(12).setCellValue(testCase.getAdoptionOfGuideline());
            row1.createCell(13).setCellValue(testCase.getRemark());
            rowNum++;
        }
        return workbook;
    }
```

图 4.5: 测试用例下载关键代码图

图4.5是测试用例管理模块中测试用例批量导出功能的关键代码。downLoad-TestCases 函数的输入参数是需要导出的测试用例 id 的列表，返回值为内含需要导出的测试用例集信息的 Excel 文件。首先，使用 HSSFWorkbook 类创建空白 Excel 文件和 Sheet 页，命名 sheet 页为“测试用例批量导出”。接着遍历静态常量字符串数组 TESTCASE_OUTPUT_HEADERS，其值为” 用例名称”,” 用例标识”,” 测试类型”,” 产品/版本/模块”,” 测试方法”,” 设计人员”,” 设计日期”,” 前提和约束”,” 测试终止条件”,” 测试用例描述”,” 输入及操作步骤”,” 预期结果”,” 评估准则”,” 备注”，将它们设置为 Excel 文件中每列的列头，意为测试用例中的属性名。最后，遍历测试用例 id 列表，调用 testCaseRepository 对象的 findById 方法从数据库中获取相应的测试用例信息，设置到 Excel 文件的每一行中，最终返

回 HSSFWorkbook 类对象。

4.2 测试用例特征提取模块详细设计与实现

4.2.1 测试用例特征提取模块核心类图

测试用例特征提取模块的核心类图如图4.6所示。测试用例特征提取模块使用历史测试用例数据扩增数据集，使用扩增后的数据集获取历史测试用例关键词数据，使用历史测试用例关键词数据训练 Word2Vec 模型，获取测试用例对应的词向量。

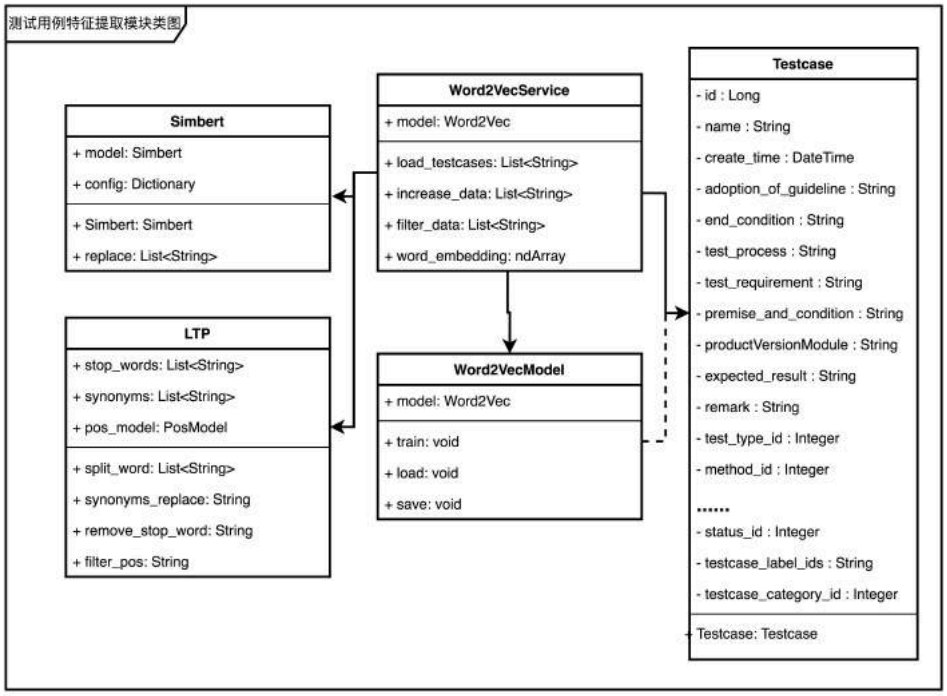

图 4.6: 测试用例特征提取模块核心类图

本模块包含核心类 Word2VecService 类，对外提供五个接口，分别是载入历史测试用例数据（load_testcases）、扩增测试用例数据（increase_data）、获取关键词实体（filter_data）、根据历史关键词向量训练 Word2Vec 模型（train_model）和获取词向量（word_embedding）。Simbert 类包含 Simbert 语言词汇模型和 replace 接口，为 Word2VecService 类提供生成语义相似的测试用例文本服务，increase_data 函数通过指定 Simbert 类配置和提供测试用例文本进行数据扩增。

fliter_data 函数调用 LTP 库对测试用例进行分词、去除停用词、同义词替换、词性筛选，最终得到测试用例的关键词实体。LTP 类对外提供提供 remove_stop_word

方法根据停用词表去除文本中没有意义的词。提供 synonyms_replace 方法根据同义词词林进行同义词替换，将多个语义相同的词语转换成同一个关键词。提供 filter_pos 方法根据词性模型筛选掉词性为名词和动词以外的词，因为这两个词性的词最能保留文本含义。提供 split_word 方法进行分词，将一个完整的句子分成能够组成该句子的多个关键词。

Word2VecService 类的 word_embedding 函数根据 Word2VecModel 类获取测试用例关键词实体对应的词向量。Word2VecModel 对外提供模型训练（train）、模型加载（load）、模型存储（save），其中 train 函数提供生成词袋功能，后续的增量测试用例关键词实体借助词袋转化为词向量。Testcase 是实体类，对应测试用例。

4.2.2 测试用例特征提取模块顺序图

测试用例特征提取模块的顺序图如图4.7所示。Word2VecService 类调用自己的 load_testcase 方法从数据库中载入历史测试用例数据，接着调用 increase_data 方法，increase_data 中使用 Config 字典定义 Simbert 类初始化所需的参数，调用 Simbert 方法初始化 Simbert 模型，然后使用 Simbert 的 replace 方法对每条测试用例生成语义相似的五条测试用例文本并返回，扩增测试用例数据集数据，提高之后分类模型的准确率。Word2VecService 类还调用自身的 filter_data 方法，使用 LTP 工具处理原始测试用例文本，将它们转化为由关键词组成的列表。首先读取 LTP 的 cws.model 分词模型初始化 LTP 处理类，接着读取 stop_word.txt 文件、同义词词林 txt 文件和 pos.model 文件，初始化 LTP 处理类的停用词列表、同义词列表和词性模型。停用词是指在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据之前或之后会自动过滤掉的某些字或词。同义词词林是将同义或近义的词收排在一起的类义词典。词性标注模型将汉语词类系统分为 7 类体词，4 类谓词、5 类虚词、代词和感叹词，为每一个词语标注它的词性。filter_data 方法依次调用 LTP 类的 split_word 函数进行分词、调用 remove_stop_word 方法去除分词后的停用词、调用 synonyms_replace 方法将同义或近义的词转化为同一个词语、调用 filter_pos 方法筛选掉除了词性为动词或名词外的所有词语，最终返回最能体现文本特征的关键词列表。

最后，Word2VecService 类调用 Word2VecModel 类的 train 方法，根据历史测试用例关键词数据训练出词袋模型并保存，将文本转化成词向量。在预测实时测试用例数据的类别时，ClassifyService 类首先调用 Word2VecService 的 word_embedding 方法，先通过 filter_data 方法将测试用例文本转化为关键词列表，再加载 Word2VecModel 的词袋模型将关键词列表转化为单条词向量供之后

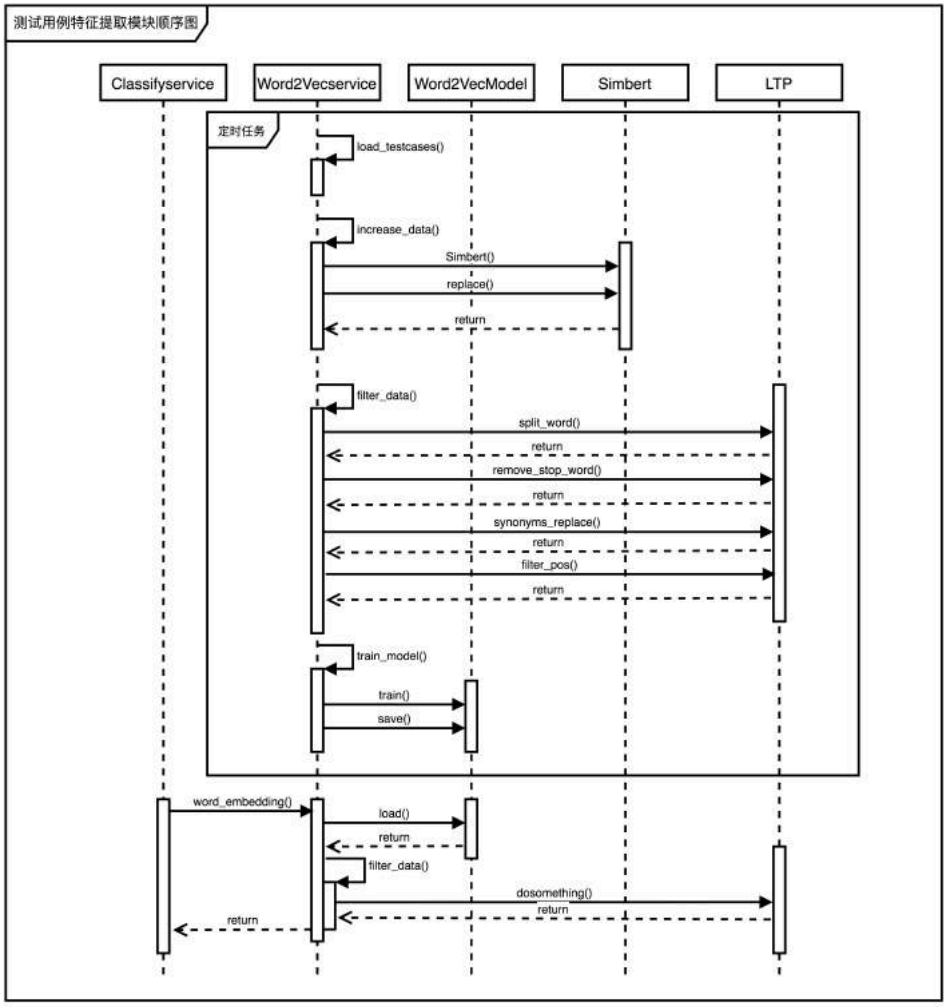

图 4.7: 测试用例特征提取模块顺序图

的分类模块使用。

4.2.3 测试用例特征提取模块的实现

图4.8是测试用例特征提取模块中数据扩增功能的关键代码。首先定义 Simbert 模型的配置参数，model_path 参数指定了训练好的中文相似语义文本生成模型的路径参数，CUDA_VISIBLE_DEVICES 参数选择要用的 CPU，max_len 参数指定要返回的文本长度，seed 定义了文本替换的随机种子。process_file 方法是可以遍历指定目录下的所有文件和目录，系统使用文件系统中的 txt 文件存储测试用例的测试过程和测试需求文本信息，当 process_file 方法遍历到 txt 文件时，读取文件信息，使用 Simbert 类的 replace 方法生成相似文本，replace 方法的参数 sent 为原测试用例文本、create_num 为指定的生成原文本的相似文本的个数，

返回值为包含 create_num 个扩增文本的列表。

```
from nlpcda import Simbert
config = {
        'model_path': '/Users/xxx /chinese_simbert_L-12_H-768_A-12',
        'CUDA_VISIBLE_DEVICES': '0,1',
        'max_len': 32,
        'seed': 1
}
def process_file(root_path):
    dir_or_files = os.listdir(root_path)
    for dir_file in dir_or_files:
        dir_file_path = os.path.join(root_path,dir_file)
        if os.path.isdir(dir_file_path):
            process_file(dir_file_path)
        else:
            if dir_file_path.split(".")[1] == "txt":
                id = dir_file_path.split(".")[0]
                description = read_file(dir_file_path)
                description = description.replace("\n"," ")
                synonyms = simbert.replace(sent=description, create_num=5)
                for i in range(5):
                    create_file(id + "+" + str(i) + ".txt", synonyms[i][0])

bug_type = ["不正常退出", "功能不完整", "用户体验", "页面布局缺陷", "性能", "安全"]
base_path = "/Users/xxx /手动标记后的数据/决赛自主可控众测 web 自主可控运维管理系统/train/不正常退出"

simbert = Simbert(config=config)
process_file(base_path)
```

图 4.8: 数据扩增关键代码图

```
def stop_word_list(self, filepath):
        stop_word = [line.strip() for line in open(filepath, "r", encoding = "utf-8").readlines()]
        return stop_word

    def clean_word_list(self, origin_data, stop_word):
        clean_word = [word for word in origin_data if word not in stop_word]
        postags = list(self.postagger.postag(clean_word))
        clean_word_after_postagger = []
        for i in range(len(postags)):
            if postags[i] == "n" or postags[i] == "v":
                clean_word_after_postagger.append(clean_word[i])
        return clean_word_after_postagger
```

图 4.9: LTP 分词部分代码图

图4.9是测试用例特征提取模块中LTP分词功能的部分代码。其中stop_word_list方法根据传入的文件路径读取停用词文件,获取生成停用词列表。clean_word_list方法首先去除停用词，接着根据读取到的词性模型获取测试用例文本中每个词的词性，保存词性为“n”（名词）和“v”（动词）的词语，返回筛选掉停用词和其他词性词语的关键词列表。

```
def word_count(self, datas):
        #统计单词出现的频次，并将其降序排列，得出出现频次最多的单词
        dic = {}
        for data in datas:
            data_list = data.split()
            for word in data_list:
                word = word.lower() #所有单词转化为小写,中文没有小写 todo
                if(word in dic):
                    dic[word] += 1
                else:
                    dic[word] = 1
        word_count_sorted = sorted(dic.items(), key=lambda item:item[1], reverse=True)
        return   word_count_sorted # 键是词，值是出现的次数

    def word_index(self, datas, vocab_size):
        #创建词表
        word_count_sorted = self.word_count(datas)
        word2index = {}
        #词表中未出现的词，因为词表大小有限，所以有些句子中的词不在词表中
        word2index["<unk>"] = 0
        #句子添加的 padding，whats this
        word2index["<pad>"] = 1

        #词表的实际大小由词的数量和限定大小决定
        vocab_size = min(len(word_count_sorted), vocab_size)
        for i in range(vocab_size):
            word = word_count_sorted[i][0]
            word2index[word] = i + 2 # 键是 词，值是在 word2index 列表中的位置

        return word2index, vocab_size

for data in train_datas:
            feature = []
            data_list = data.split() # 分割 默认为所有的空字符
            for word in data_list:
                word = word.lower() #词表中的单词均为小写
                if word in word2index:
                    feature.append(word2index[word])
                else:
                    feature.append(word2index["<unk>"]) #词表中未出现的词用<unk>代替
                if(len(feature)==max_len): #限制句子的最大长度，超出部分直接截断
                    break
            #对未达到最大长度的句子添加 padding
            feature = feature + [word2index["<pad>"]] * (max_len - len(feature))
```

图 4.10: 训练词向量模型关键代码图

图4.10是测试用例提取模块中训练 Word2Vec 模型的关键代码。其中 word_count 方法统计输入参数 datas 中各个单词出现的频次并将单词按频次降序排列，返回排序后的字典，字典的键是单词，值是单词在语料中出现的次数。word_index 方法用来创建词表并返回词表和词表的大小，首先初始化空字典 word2index，如果单词是词中未出现的，则统一用“<unk>”表示，对应值为 0，如果句子的长度不够规定值，则后面的空统一用“<pad>”表示，对应值为 1。根据词频字典创建词表字典 word2index，词表字典的键为词频字典中的键，词表字典的值为键在词表中的位置。第三部分是利用词表将关键词列表转化为特征向量的代码，特征向量中的每一个数是关键词列表中同位置的关键词在词表中对应的值。

4.3 基于双向 LSTM 的测试用例分类模块详细设计与实现

4.3.1 测试用例分类模块核心类图

测试用例分类模块核心类图如图4.11所示。主要是依赖 BiLSTM 类进行测试用例初始分类模型的训练与服务。

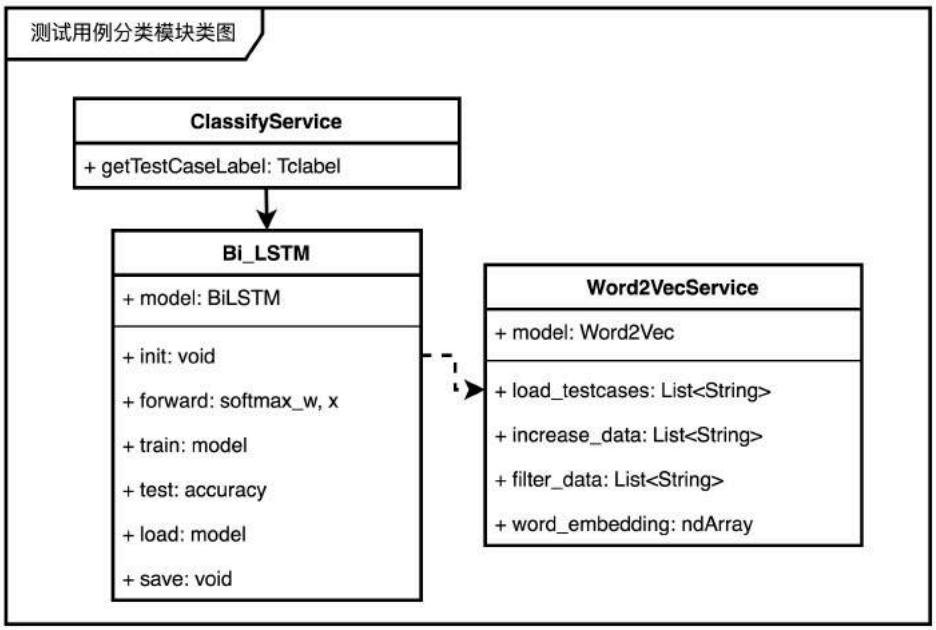

图 4.11: 测试用例分类模块核心类图

本模块包含 ClassifyService 类和 BiLSTM 类。其中，ClassifyService 类包含 getTestcaseLabel 接口，接收 Java 后端 TcLabel 类服务 createCategory 接口的调用，对外提供测试用例的类别标签预测服务。

BiLSTM 类是实际上的分类服务提供者，供有 6 个函数分别是模型初始化（init）、前向计算函数（forward）、模型训练（train）、模型准确率测试（test）、保存训练好的模型（save）和加载训练好的模型（load）。BiLSTM 类依赖 Word2VecService 类获取测试用例词向量，最终训练出一个初始分类器并通过阿里云 OSS 服务保存。其中，init 函数初始化双向 LSTM 模型的输入层、embedding 层、lstm 层和输出层。forward 函数定义模型的前向计算，根据输入 x 计算返回所需要的模型输出，可以把 forward 理解成常规的 predict 函数，返回的是预测出的类别。train 函数使用多批训练集数据反复训练模型，调用 test 函数查看当前模型在测试集上的准确率，保存在测试集上表现最好的模型参数。

4.3.2 测试用例分类模块顺序图

测试用例分类模块的顺序图如图4.12所示。当测试人员点击获取分类按钮后, Java 后端调用 Python 后端 ClassifyService 类的 getTestcaseLabel 方法, getTestcaseLabel 首先调用 Word2VecService 类的 word_embedding 方法将原始文本词向量化，接着加载双向 LSTM 分类模型，调用 forward 方法获取分类模型对测试用例词向量的类别预测，返回给 Java 后端。而对于双向 LSTM 分类模型，本系

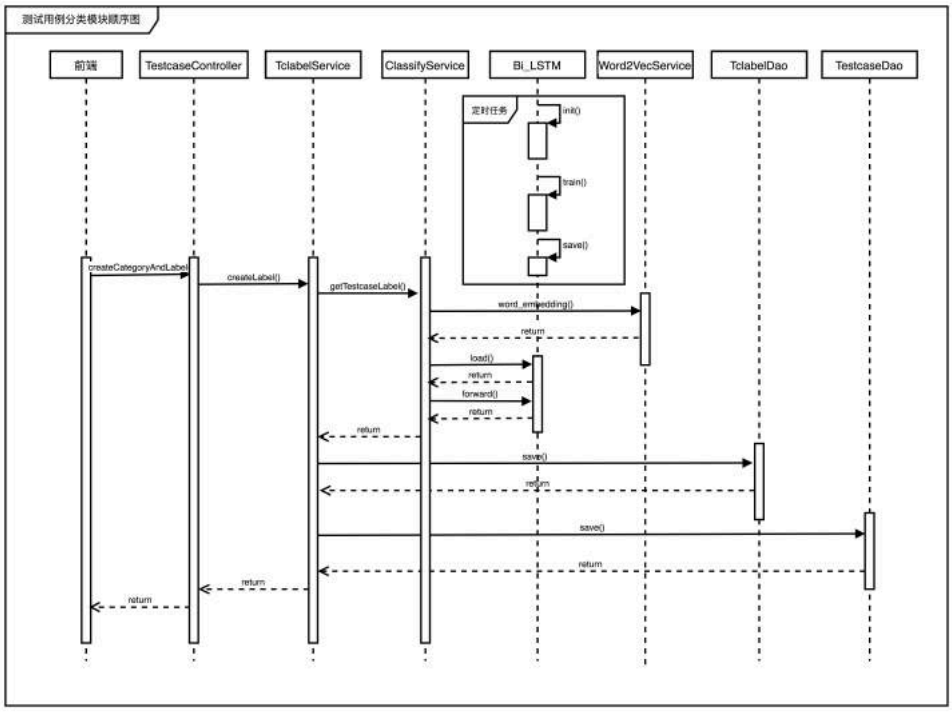

图 4.12: 测试用例分类模块顺序图

统设置了一个定时任务，定时初始化分类模型并更新历史测试用例词向量数据，通过 train 方法重新训练并调用 save 方法保存最新的分类模型。

4.3.3 测试用例分类模块的实现

图4.13是测试用例分类模块中定义双向 LSTM 模型前向计算的关键代码。forward 方法根据输入参数 x 返回所需要的模型输出。输入参数 x 是一个维度为 [batch_size, sentence_max_length, embedding_size] 的三维数据，batch_size 是数据集的批尺寸，sentence_max_length 是单个输入向量的长度，embedding_size 是词向量的维度。由于 lstm 层的输入维度为 [seq_len, batch, input_size]，方法首先设置 lstm 最初的前项输出，接着将双向 LSTM 的输出拆分为前向输出和后向输出。双向 LSTM 中通常会使用最后一个时序的输出向量作为特征向量，然后

进行特征分类。因此为了使用到 lstm 最后一个时间步时每层 lstm 的表达，代码使用 h_n 生成 attention 的权重，最后得到 softmax_w 的值是注意力机制的权重向量。forward 方法返回值为注意力权重 softmax 和输入向量的分类预测结果 x。图4.14是测试用例分类模块中训练双向 LSTM 模型的关键代码。train 方法中根据

```
# 定义模型的前向计算，即如何根据输入 x 计算返回所需要的模型输出
    def forward(self, x):
        x = x.permute(1, 0, 2)
        batch_size = x.size(1)

        #设置 lstm 最初的前项输出
        h_0 = torch.randn(self.num_layers * self.num_directions, batch_size, self.hidden_size).to(device)
        c_0 = torch.randn(self.num_layers * self.num_directions, batch_size, self.hidden_size).to(device)

        #out[seq_len, batch, num_directions * hidden_size]   out 只保存最后一层每个时间步 t 的输出 h_t
        out, (h_n, c_n) = self.lstm(x, (h_0, c_0))

        #将双向 lstm 的输出拆分为前向输出和后向输出
        (forward_out, backward_out) = torch.chunk(out, 2, dim = 2)
        out = forward_out + backward_out   #[seq_len, batch, hidden_size]
        out = out.permute(1, 0, 2)   #[batch, seq_len, hidden_size]

        h_n = h_n.permute(1, 0, 2)   #[batch, num_layers * num_directions,   hidden_size]
        h_n = torch.sum(h_n, dim=1) #[batch, 1,   hidden_size]
        h_n = h_n.squeeze(dim=1)   #[batch, hidden_size]

        attention_w = self.attention_weights_layer(h_n)
        attention_w = attention_w.unsqueeze(dim=1)
        attention_context = torch.bmm(attention_w, out.transpose(1, 2))   #[batch, 1, seq_len] [16 ,1, 32]

        softmax_w = F.softmax(attention_context, dim=-1)   #[batch, 1, seq_len],权重归一化  [16, 1, 32]

        x = torch.bmm(softmax_w, out)   #[batch, 1, hidden_size]
        x = x.squeeze(dim=1)   #[batch, hidden_size]
        x = self.liner(x)
        x = self.act_func(x)
        return softmax_w, x
```

图 4.13: 前向计算关键代码图

当前模型参数下 test 方法返回的测试集上模型的准确率，保存最好的训练模型参数 best_model_params。首先初始化模型在训练集上最高的准确率为 0，接着使用 model.train 方法记录批训练时模型内参数的变化情况，使用 model.forward 方法和 torch.argmax 方法获取最大可能性的预测类别，最后使用 model.load_state_dict 方法存储模型。

4.4 基于注意力机制的类别词库收集模块详细设计与实现

4.4.1 类别词库收集模块核心类图

类别词库收集模块核心类图如图4.15所示。主要进行类别词库的收集和利用基于双向 LSTM 的类别词库收集算法赋予无标签测试用例数据伪标签以扩大训练集，训练性能更好的分类器。本模块的核心类是 BiLSTM_with_Attention 类，

```
def train_origin(model, train_loader, test_loader, optimizer, loss_func, epochs):
    best_val_acc = 0.0
    best_model_params = copy.deepcopy(model.state_dict())
    for epoch in range(epochs):
        model.train()
        loss_val = 0.0
        corrects = 0.0
        for datas, labels in train_loader:
            datas = datas.to(device)
            labels = labels.to(device)
            attention_w, preds = model.forward(datas) # 使用 model 预测数据
            loss = loss_func(preds, labels)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            loss_val += loss.item() * datas.size(0)

            #获取预测的最大概率出现的位置
            preds = torch.argmax(preds, dim=1)
            labels = torch.argmax(labels, dim=1)
            corrects += torch.sum(preds == labels).item()
        train_loss = loss_val / len(train_loader.dataset)
        train_acc = corrects / len(train_loader.dataset)
        if(epoch % 2 == 0):
            print("Train Loss: {}, Train Acc: {}".format(train_loss, train_acc))
            test_acc = test_origin(model, test_loader, loss_func)
            if(best_val_acc < test_acc):
                best_val_acc = test_acc
                best_model_params = copy.deepcopy(model.state_dict())
    model.load_state_dict(best_model_params)
    return model
```

图 4.14: 双向 LSTM 模型训练关键代码图

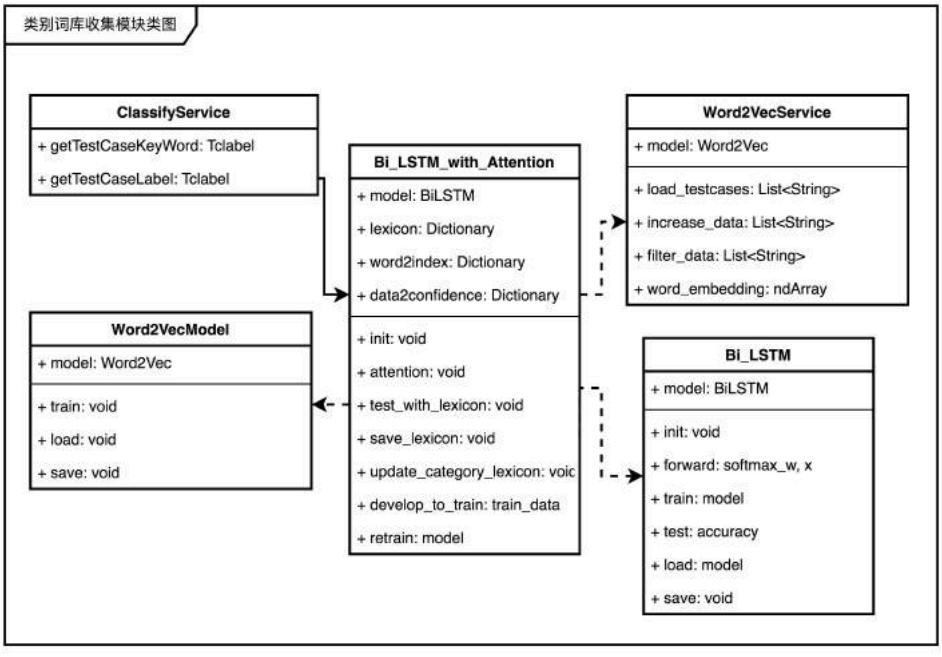

图 4.15: 类别词库收集模块核心类图

主要目的是向双向 LSTM 模型中添加 Attention 机制，关注每个类别的代表性词汇。BiLSTM_with_Attention 类主要包含 7 个函数，分别是初始化双向 LSTM 模型和类别词库（init）、添加 Attention 机制（attention）、利用发展集收集词库（test_with_lexicon）、更新词库（update_category_lexicon）、保存词库（save_lexicon）、转移数据集（develop_to_train）和重新反复训练测试用例分类器（retrain）。其中，test_with_lexicon 函数在测试用例词向量发展集上重新运行初始分类器，根据预测出的类别、置信度和词语的注意力权重，调用 update_category_lexicon 函数更新在发展集上收集到的类别词库，同时记录下发展集中被新赋予伪标签的数据，调用 develop_to_train 函数将该条数据从发展集中剔除，加入到训练集中。

BiLSTM_with_Attention 类依赖 Bi_LSTM 类获取初始的训练完的分类器模型，通过在 init 函数中使用 attention 函数并调用 Bi_LSTM 类的 init 函数初始化出带有注意力机制的双向 LSTM 网络。

BiLSTM_with_Attention 类依赖 Word2VecService 类获取测试用例词向量。依赖 Word2VecModel 获取从数据集上构建出的词袋，得到词语和词向量中每个位置的数字的对应关系，可视化模型的注意力机制。

4.4.2 类别词库收集模块顺序图

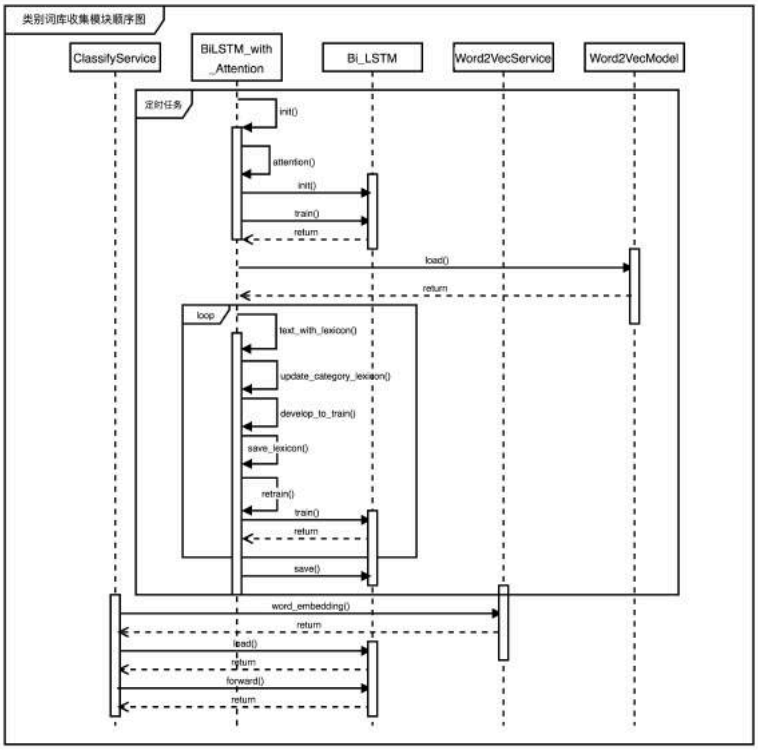

图 4.16: 类别词库收集模块顺序图

```
for i in sorted(confidence_dict, reverse = True):
        lexicon_key = category_dict[confidence_dict[i]]
        if lexicon_num[lexicon_key] <= n: # 每个类别取置信度最高的前 n 条数据
            # print(str(lexicon_key) + ":" + str(i))
            lexicon_num[lexicon_key] += 1
            lexicon_value_attention = attention_dict[confidence_dict[i]]
            lexicon_value_word = develop_feature_origin[confidence_dict[i]]
            attention2word = dict(zip(lexicon_value_attention,lexicon_value_word))

            word2attention = {}
            for j in sorted(attention2word, reverse = True):
                word = list(word2index.keys())[list(word2index.values()).index(attention2word[j])]
                if word != "<unk>" and word != "<pad>":
                    if word in word2attention.keys():
                        word2attention[word] += j
                    else:
                        word2attention[word] = j
            q = 0
            for k in sorted(word2attention.items(), key = lambda kv:(kv[1], kv[0]), reverse = True):
                if q < m and k[0] not in lexicon[lexicon_key]:
                    lexicon[lexicon_key].append(k[0])
                q += 1
```

图 4.17: 类别词库收集部分代码图

类别词库收集模块的顺序图如图4.16所示。BiLSTM_with_Attention 类是 BiLSTM 类的子类，在功能上是它的强化和扩展，首先在调用 init 方法时为 BiLSTM 类的 init 方法增加了注意力层，训练出一个带有注意力机制的双层 LSTM 模型。BiLSTM_with_Attention 类同时加载了 Word2VecModel 类中训练好的词袋模型以获取注意力机制关注到的词向量对应的实际关键词。在初始化步骤完成后，BiLSTM_with_Attention 通过调用自身的 text_with_lexicon 方法向发展集中的词向量标注伪标签，调用 update_category_lexicon 方法在标注伪标签时将前 m 高置信度的词向量中具有前 n 高的注意力权重的词向量对应的关键词收集到对应预测类别的词库中，调用 develop_to_train 方法将被标注了伪标签的词向量从发展集中剔除，转移到训练集中，使用扩增后的新训练集调用 retrain 方法重新训练初始分类器并保存训练后的分类模型。上述反复扩增训练集的步骤将不断循环重复，直到不再有发展集中的词向量被赋予伪标签。

4.4.3 类别词库收集模块的实现

图4.17是类别词库收集模块收集词集功能 text_with_lexicon 方法中的部分代码。这部分代码首先对每条测试用例词向量的预测置信度排序，从中获取每个预测类别的前 n 高置信度的测试用例词向量。随后，获取这条词向量的注意力权重和注意力权重所在位置对应的关键词数字以及 Word2Vec 模型的词表。构造键为关键词数字、值为关键词权重的字典，当关键词数字对应的关键词不为“<unk>”和“<pad>”时，将该关键词放入这个字典中。最后根据权重排序这个字典，获取前 m 高注意力权重的且不在当前词库中的关键词。

4.5 本章小结

本章是对基于小样本学习的测试用例分类系统的详细设计与实现的阐述，包括测试用例管理模块、特征提取模块、分类模块和类别词库收集模块。本章通过核心类图、顺序图以及代码片段等展现形式对各模块的设计思路和实现进行解读。

第五章　系统测试与实验分析

5.1　测试准备

5.1.1　测试目标

根据第三章对系统的需求分析，本部分对系统的测试主要包括功能测试和性能测试两个方面。

功能测试是指根据应用系统应该做出的功能按照规范和要求编写测试用例，根据测试用例对应用系统中的各个功能进行测试，验证应用系统是否满足功能性需求。针对本系统的功能测试主要包括系统各页面间能否正常查看和跳转、能否创建测试用例、能否上传数据集、能否生成并查看测试用例标签、能否审核测试用例和用例标签、能否导出测试用例等。

性能测试是指验证软件性能在正常环境和系统条件下经过重复使用是否还能满足性能指标的测试，验证应用系统是否满足性能需求。针对本系统的性能测试主要包括检查在高并发情况下系统的各个接口能否正常运行并及时反馈。

5.1.2　测试环境

根据第三章中系统的物理部署图，系统的测试环境说明如表5.1所示，涵盖了相关设备配置和部分软件的版本信息。同时考虑到成本因素，本系统实际部署了包括用于运行 Vue 程序和 Nginx 的前端服务器、用于运行 Java 后端程序和 Python 后端服务的后端服务器、用于运行 MySql 和 MongoDB 的数据服务器。

5.2　功能测试

5.2.1　测试设计

功能测试的测试设计部分主要根据本文第三章系统需求分析中分析出的功能性需求设计测试用例，目标是保证基于小样本学习的测试用例分类系统功能符合预期，能够满足用户的业务需求。具体的系统测试用例如下所示。

表5.2展示了创建单条测试用例的测试用例，测试的是测试人员登陆账号后填写单条测试用例相关信息并创建测试用例的功能，主要关注点是测试人员能否正常填写测试用例相关信息并保存，需要测试的具体功能包括填写测试用例、保存测试用例、测试用例列表展示新创建的测试用例。

表 5.1: 系统测试环境表

设备与软件	描述信息
用户计算机	配置：2C8G 系统：MacOS10.13.6 Chrome 浏览器：99.0.4844.74
前端服务器	配置：4C8G 系统：Ubuntu16.04 带宽：外网 4M
后端服务器	配置：4C8G 系统：Ubuntu16.04 带宽：外网 4M
数据服务器	配置：4C8G 系统：Ubuntu16.04 带宽：外网 4M
SpringBoot	版本：2.4.4
Flask	版本：1.1.1
Java	版本：1.8.0_131
Python	版本：3.6.3
Vue.js	版本：2.6.10
MySql	版本：5.7
Mongo	版本：4.0.9
torch	版本：1.0.1
tensorflow	版本：2.1.0

表 5.2: 创建单条测试用例测试用例表

测试用例编号	TC1
测试名称	创建单条测试用例测试
前置条件	用户已经得到授权和认证
测试步骤	1. 点击菜单栏“创建测试用例”按钮 2. 填写部分必填属性，缺省部分必填属性，点击“保存”按钮 3. 填写测试用例名称、测试过程、预期结果、标签等全部必填属性，点击“保存”按钮 4. 点击菜单栏“测试用例管理”按钮
预期结果	1. 展示测试用例创建页面 2. 系统弹出“填写错误”提示，需重新填写测试用例相关系信息 3. 系统弹出“创建成功”提示 4. 新创建的测试用例在测试用例列表中被展示

表5.3展示了上传测试用例集的测试用例，测试的是测试人员登陆账号后创建测试用例集的功能，主要关注点是测试人员能否正常填写数据集相关信息、选取 Excel 文件并上传，需要测试的具体功能包括填写数据集信息、选取数据集文

件上传、创建数据集并展示。

表 5.3: 上传测试用例集测试用例表

测试用例编号	TC2
测试名称	上传测试用例集测试
前置条件	用户已经得到授权和认证
测试步骤	1. 点击菜单栏“上传数据集”按钮 2. 缺省数据集名称，选取文件系统中的 Excel 文件，点击“保存”按钮 3. 填写数据集名称和其他非必填属性，选取文件系统中的非 Excel 文件，点击“保存”按钮 4. 填写数据集名称和其他非必填属性，选取文件系统中的 Excel 文件，点击“保存”按钮 5. 点击菜单栏“数据集管理”按钮
预期结果	1. 展示上传数据集页面 2. 系统弹出“填写错误”提示，需重新填写数据集相关系信息 3. 系统弹出“文件错误”提示，需重新选取文件 4. 系统弹出“上传成功”提示 4. 新上传的数据集在数据集列表中被展示

表5.4展示了查看测试用例详情的测试用例，测试的是测试人员登陆账号后查看一条测试用例内包含的所有信息的功能，主要关注点是测试人员能否根据测试用例列表查看测试用例详情，需要测试的具体功能包括通过测试用例列表页面查看用例和通过数据集关联的测试用例列表页面查看用例。

表 5.4: 查看测试用例详情测试用例表

测试用例编号	TC3
测试名称	查看测试用例详情测试
前置条件	用户已经得到授权和认证
测试步骤	1. 点击菜单栏“测试用例管理”按钮 2. 点击测试用例列表中某条测试用例的“查看”按钮 3. 点击菜单栏“数据集管理”按钮 4. 点击数据集列表中某个数据集的“查看”按钮 5. 点击数据集的测试用例列表中某条测试用例的“查看”按钮
预期结果	1. 展示测试用例列表页面 2. 展示测试用例详情页面 3. 展示数据集列表页面 4. 展示数据集内的测试用例列表页面 5. 展示测试用例详情页面

表5.5展示了修改测试用例详情的测试用例，测试的是测试人员登陆账号后修改一条测试用例内包含信息的功能，主要关注点是测试人员能否正常编辑测

试用例信息并保存，需要测试的具体功能包括编辑测试用例、取消编辑测试用例、保存测试用例。

表 5.5: 修改测试用例测试用例表

测试用例编号	TC4
测试名称	修改测试用例测试
前置条件	用户已经得到授权和认证
测试步骤	1. 点击测试用例列表中某条测试用例的“查看”按钮 2. 点击测试用例详情页面中的“编辑”按钮 3. 修改测试用例相关信息或增删测试用例标签后点击“取消”按钮 4. 修改测试用例相关信息或增删测试用例标签后点击“保存”按钮
预期结果	1. 展示测试用例详情页面 2. 测试用例详情页面进入编辑模式 3. 回到测试用例详情页面正常模式，显示编辑前信息 4. 回到测试用例详情页面正常模式，显示保存后信息

表5.6展示了生成测试用例标签的测试用例，测试的是测试人员登陆账号后使用标签分类服务为测试用例生成分类标签的功能，主要关注点是系统能否在测试用例详情页面正常展示新的分类标签。

表 5.6: 生成测试用例标签测试用例表

测试用例编号	TC5
测试名称	生成测试用例标签测试
前置条件	用户已经得到授权和认证
测试步骤	1. 点击测试用例列表中某条测试用例的“查看”按钮 2. 点击测试用例详情页面中的“标签生成”按钮
预期结果	1. 展示测试用例详情页面 2. 测试用例详情页面中展示新标签

表5.7展示了审核测试用例的测试用例，测试的是评审专家登陆账号后审核测试用例的功能，主要关注点是评审专家能否正常完成对待审核测试用例的审核，需要测试的具体功能包括待审核的测试用例列表展示、待审核的测试用例详情展示、审核通过或不通过功能。

表5.8展示了审核测试用例标签的测试用例，测试的是评审专家登陆账号后审核测试用例标签的功能，主要关注点是评审专家能否正常完成对待审核标签的测试用例的审核，需要测试的具体功能包括待审核标签的测试用例列表展示、待审核标签的测试用例详情展示、编辑待审核标签的测试用例的标签、审核完成功能。

表 5.7: 审核测试用例测试用例表

测试用例编号	TC6
测试名称	审核测试用例测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录评审专家账号 2. 点击菜单栏“审核测试用例”按钮 3. 点击测试用例列表某条测试用例的“查看”按钮 4. 点击“通过”按钮 5. 在新的待审核测试用例的页面中点击“不通过”按钮
预期结果	1. 展示评审专家权限特有的菜单栏 2. 展示待审核的测试用例列表 3. 展示待审核的测试用例详情页面 4. 系统弹出“审核完成”提示 5. 系统弹出“审核完成”提示 6. 该条测试用例状态更改为“审核通过”或“审核不通过”，从待审核列表中消失

表 5.8: 审核用例标签测试用例表

测试用例编号	TC7
测试名称	审核用例标签测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录评审专家账号 2. 点击菜单栏“审核测试用例标签”按钮 3. 点击测试用例列表某条测试用例的“查看”按钮 4. 点击“编辑标签”按钮 5. 修改或不修改标签后点击“完成”按钮
预期结果	1. 展示评审专家权限特有的菜单栏 2. 展示待审核标签的测试用例列表 3. 展示待审核标签的测试用例详情页面 4. 测试用例详情页面进入编辑模式，此时可编辑标签信息，但不能编辑其他测试用例相关信息 5. 系统弹出“审核完成”提示 6. 该条测试用例状态更改为“审核通过”或“标签审核不通过”，从待审核标签的列表中消失

5.2.2 测试执行

测试执行指一个测试用例在测试周期中的独立的执行实例。本文严格按照测试用例的测试步骤执行用例，表5.9是测试设计小节测试用例的执行结果记录，从最终测试结果得出结论，本系统实现的功能可以满足此前分析的功能性需求。

表 5.9: 功能测试用例执行结果表

测试用例 ID	用例描述 ID	测试结果
TC1	UC1	通过，单条测试用例创建成功。
TC2	UC2	通过，测试用例集上传，数据集创建成功。
TC3	UC3、UC4	通过，系统显示测试用例详情。
TC4	UC5	通过，测试用例修改成功。
TC5	UC6	通过，测试用例类别标签生成成功。
TC6	UC7	通过，测试用例审核成功。
TC7	UC8	通过，测试用例标签审核成功。

5.3 性能测试

5.3.1 测试设计

性能测试的测试设计部分主要根据本文第三章分析出的非功能性需求，针对系统的接口进行高并发测试。目标是在高并发情况下验证系统的能力状态，通过测试系统各个接口的响应时间、成功率，了解系统的运行速度，满足用户的性能需求。本系统需要进行性能测试的接口如表5.10所示。

表 5.10: 性能测试接口列表

测试接口编号	接口描述	接口类型	接口路径
I1	创建单条测试用例	POST	/testcase
I2	上传测试用例集	POST	/dataset/excel
I3	查看测试用例列表	GET	/testcase/list/{statusId}
I4	查看测试用例详情	GET	/testcase/{testCaseId}
I5	查看数据集详情	GET	/dataset/{datasetId}
I6	导出测试用例	GET	/testcase/download/{testCaseId}
I7	审核测试用例	POST	/testcase/check
I8	审核测试用例标签	POST	/testcase/checkLabel
I9	生成测试用例标签	POST	/testcase/createLabel

本文采用 Jmeter 对系统进行性能测试。Jmeter 是一款使用 Java 语言开发的、基于多线程并发模型的压测工具，也是目前最流行的开源压测工具。以 I2 上传测试用例集接口为例，Jmeter 的具体配置步骤如下：

（1）在测试计划中创建线程组，命名为“文件上传”，设置线程属性中线程数 =200、Ramp-Up 时间（秒）=10、循环次数 =5。其中线程数意为并发现成的数目，每个线程完全独立互不打扰，多个线程模仿对服务器的并发访问；Ramp-Up 时间为启动所有线程所需要的时间；循环次数意为每个线程在 Ramp-Up 时间内

执行测试计划的次数。上述设置的含义为，10 秒钟内有 200 个线程（虚拟用户）分别对测试接口循环发送 5 次请求。

（2）右击线程组，创建 HTTP 信息头管理器，将名为 “Content-Type”、值为 “application/octet-stream” 的键值对存储在信息头管理器中。HTTP 信息头管理器用于发出的 HTTP 请求的请求头内容。

（3）右击线程组，创建名为 “上传测试用例” 的 HTTP 请求，配置内容包括 Web 服务器栏的协议、服务器 IP、端口号，HTTP 请求栏的请求类型、路径、内容编码。特别的，选中 “对 POST 使用 multipart/form-data” 以允许上传文件和参数，对应的在 “参数” 栏新建 name 和 ownerId 两个参数、在 “文件上传” 栏新建 file 参数将本地文件 “/Users/tanghaojie/Desktop/test.xls” 以 “application/octet-stream” 类型上传。I2 接口的配置详情如图5.1所示。

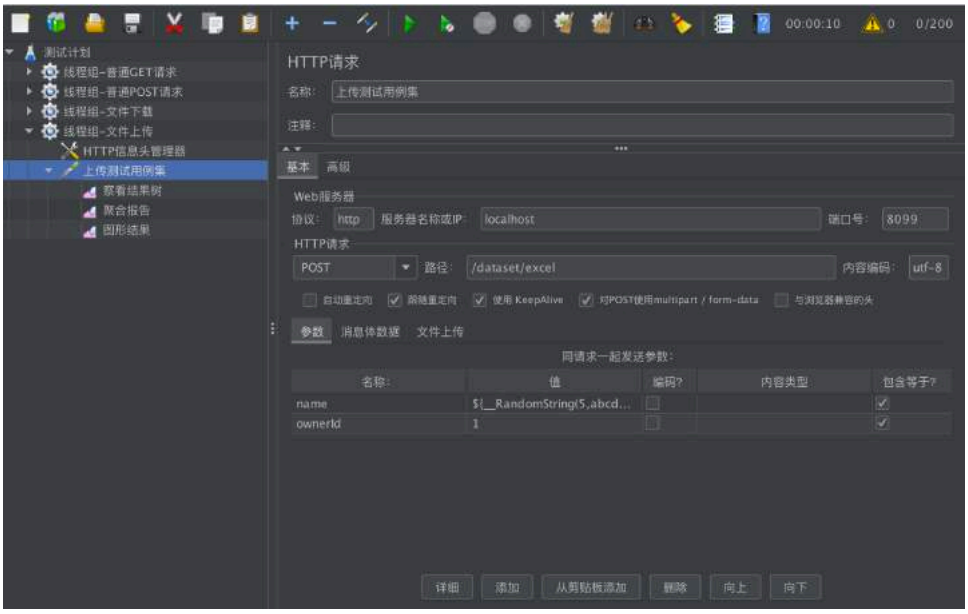

图 5.1: 上传测试用例集 HTTP 请求配置截图

5.3.2 测试执行

各个接口的性能测试结果如表5.11所示。其中，平均响应时间的计算方式为总运行时间除以发送到服务器的总请求数，95% 响应时间指有百分之九十五的请求小于这个值，99% 响应时间指有百分之九十九的请求小于这个值。

以 I2 上传测试用例集接口为例，该接口在所有接口中的响应时间相对较长，但仍然满足用户的性能需求（复杂接口响应时间放宽到 500ms 以内）。如图5.2所示，一千次模拟请求的平均响应时间为 82 毫秒，90% 的请求在 229 毫秒内得到响应，99% 的请求在 458 毫秒内得到响应，发生异常的请求数目为 0。

表 5.11: 性能测试结果列表

测试接口编号	异常请求百分比	平均响应时间	95% 响应时间	99% 响应时间
I1	0%	23ms	85ms	177ms
I2	0%	82ms	320ms	458ms
I3	0%	35ms	99ms	158ms
I4	0%	31ms	128ms	253ms
I5	0%	20ms	82ms	227ms
I6	0%	89ms	356ms	498ms
I7	0%	9ms	23ms	43ms
I8	0%	11ms	22ms	41ms
I9	0%	92ms	374ms	503ms

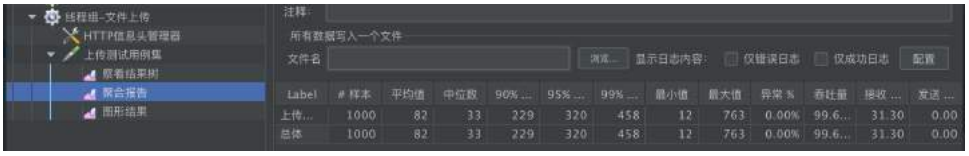

图 5.2: 上传测试用例聚合报告截图

5.4 效果测试

在验证了测试用例管理和分类系统的可用性和可靠性之后，我们还需要对测试用例分类服务的分类效果进行评估，检查由基于小样本学习的测试用例分类技术生成的分类器的效果。检查内容具体包括该分类器在小样本场景下生成的类别标签是否合理准确，生成的类别标签能否为专家审核测试用例类别提供帮助。

5.4.1 测试对象

为了探究上述问题，我们选取了三个从真实众包测试比赛中获取的测试用例集合作为我们的实验对象，开展我们的系统评估实验。这三个测试用例数据集分别来自娱乐和工具两个不同的领域，数据集中测试用例的数量如图5.3所示。

5.4.2 测试设计

（1）为验证本系统采用的分类器在小样本场景下生成的类别标签是否足够合理准确，我们设置对比实验，分别使用基于小样本学习的测试用例分类算法、支持向量机算法、k 最近邻算法和朴素贝叶斯算法在三个数据集上建立分类器。这些分类器使用数据集 20% 的数据充当训练集训练，使用数据集 80% 的数据作为测试集，来模拟测试用例集在真实场景下已分类样本数占总样本数较少的情

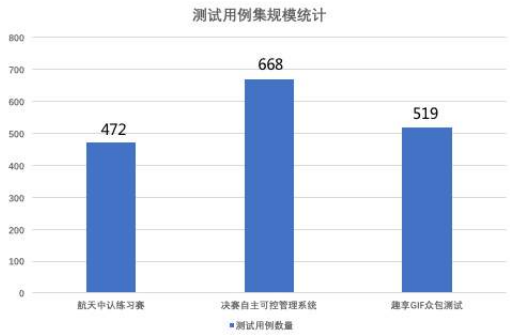

图 5.3: 众包测试测试用例集合规模统计

况，统计每个分类器的准确率，查看我们的算法是否具有优越性。

（2）为验证分类器生成的类别标签能否为专家审核测试用例类别和测试人员手动分类测试用例提供帮助。我们设置对比实验，将 12 名测试人员平均分为 AB 两组，其中 A 组人员使用测试用例分类服务生成类别标签作为手动分类测试用例的参考，B 组人员不使用分类服务直接手动分类测试用例。我们要求测试人员手动为三组测试用例集中的测试用例填写类别并统计他们手动完成分类的时长，探究分类器能否帮助降低测试用例的分类成本。同时，我们邀请三名经验丰富的测试人员充当评审专家，对测试人员手动分类测试用例的准确率进行评分，探究分类器能否帮助提升测试用例的分类质量。

5.4.3 测试结果

表5.12展示的是测试设计（1）的测试结果。其中，行数据表示的是每种分类算法训练出的分类器在不同数据集上的准确率，列数据表示的是每种数据集在不同分类算法训练出的分类器上的分类准确率。可以看到，本系统算法的最高分类准确率接近 60%，相对于其他分类算法而言，准确率均有约 5% 到 25% 的提升。

表 5.12: 分类准确率结果列表

分类算法	航天中认练习赛	决赛自主可控管理系统	趣享 GIF 众包测试
本系统算法	**0.51**	**0.57**	**0.48**
支持向量机	0.47	0.51	0.44
k 最近邻	0.33	0.42	0.32
朴素贝叶斯	0.20	0.24	0.17

表5.13展示的是 AB 两组测试人员被要求对不同的测试用例集中未分类的测试用例进行分类时的平均时长统计情况。可以看到，在分类器生成的类别标签的辅助下，A 组测试人员的平均统计时长均比 B 组测试人员的时长缩短超过 6 分钟，大大提高了手工分类测试用例的效率。

表 5.13: 测试人员分类时长统计表

被分类数据集名称	A 组平均分类时长	B 组平均分类时长
航天中认练习赛	19.3min	25.1min
决赛自主可控管理系统	25.2min	34.7min
趣享 GIF 众包测试	20.2min	27.9min

表5.14展示的是三位评审专家对每个测试人员对每个测试用例集中测试用例的分类质量的评审情况，采用打分计算均分的方法体现分类质量，分数可以为 1 到 10 分。其中可以看到，A 组测试人员在同一数据集上的平均得分均高于 B 组测试人员的平均得分，说明了分类器生成的类别标签可以辅助提高手工分类测试用例的质量。

表 5.14: 测试人员分类质量评审得分表

评审专家编号	1-A 组	1-B 组	2-A 组	2-B 组	3-A 组	3-B 组
E1	8.2	7.4	8.3	7.7	8.1	7.2
E2	8.0	7.2	8.5	8.1	7.9	7.1
E3	8.5	8.3	9.1	8.6	8.3	7.7

5.5 本章小结

本章对系统的功能、性能和效果进行了充分的测试。功能和性能测试的结果表明，系统能够满足用户的功能性需求，能够提供预定的功能，也能够在高并发的场景下正常运行，满足用户的非功能性需求。效果测试的结果表明，基于小样本学习的测试用例分类技术可以帮助测试人员自动生成较为合理的测试用例类别标签，也可以有效辅助测试人员手工分类，提高手工分类的分类效率和分类质量，有效降低了手工分类成本。

第六章　总结与展望

6.1　总结

众包测试具有测试速度快、测试成本低等特点，因此倍受业内关注。在众包测试的过程中，众包工人为了提交缺陷报告会产出大量的测试用例，这些测试用例通常具有类别标签稀少、类别标签分布不均匀、类别标签标注错误等问题，导致测试用例集的管理人员难以分辨测试用例的类别并充分复用测试用例。本文针对以上问题，结合测试用例管理平台和协作式众包平台现状，提出了基于小样本学习的测试用例分类技术，开发了测试用例管理和分类系统。

测试人员在登录本系统后，可以主动上传测试用例集或单条测试用例，可以根据测试用例的测试类型、测试方法、产品/版本/模块、测试过程、测试需求等属性筛选并导出测试用例，更加智能化和规范化的管理测试用例。同时，系统为每个测试用例集提供该集合内的测试用例分类功能，测试人员可以点击测试用例详情页面的获取分类按钮得到由带有注意力机制的双向 LSTM 分类器预测的用例类别，作为对众包工人标记的测试用例原始分类的补充或纠正，进而辅助测试人员了解和挑选测试用例。为了保证测试用例的准确性和自动生成的测试用例标签的有效性，系统还提供了评审专家账号用于审核测试用例和用例标签，评审专家可以结合自身经验判断用例标签是否合理并进行修改，为分类器提供了可靠的训练数据，提高分类质量，间接帮助复用测试用例，降低测试成本。

本文首先介绍了项目的背景和意义，通过调研测试用例、文本分类和小样本学习的研究现状，受到相关工作的启发，将测试用例特征和小样本文本分类相结合，提出了基于小样本学习的测试用例分类技术，明确了该技术在当前情景下的可实现性。同时，介绍了项目设计和开发过程中涉及到的技术框架和算法理论，为后续工程实现奠定了基础。其次，通过对系统进行需求分析，借助用例图和系统用例描述，明确了系统的涉众、功能性需求和非功能性需求，再根据分析结果对系统进行架构设计、模块划分和数据模型设计，采用 4+1 视图从不同角度对系统进行了描述，然后对系统使用的分类算法和类别词库收集算法进行了设计与描述。接着使用流程图、核心类图、关键代码进一步阐述测试用例管理、测试用例特征提取、测试用例分类和类别词库收集四个模块的详细设计与实现。最后，在系统的部署环境中，使用功能测试、性能测试的方法验证了系统的可用性和可靠性，同时在真实的众包测试测试用例集上通过实验分析比较和

说明了小样本情景下测试用例分类技术的效果。

6.2 展望

尽管基于小样本学习的测试用例分类技术已在真实测试用例管理平台中投入使用并取得了一定的效果。支持在历史测试用例集中已分类的测试用例数目较少的情况下，为属于该集合的增量测试用例提供较为合理的分类，但系统和实验的实现目前仍然存在许多需要改进和继续研究的地方：

首先，囿于现实因素，现阶段用于实验的测试用例集数据规模较小，更加适用机器学习进行处理。而基于小样本学习的测试用例分类技术属于深度学习和神经网络类别，比较适用于更大的数据集，我们暂时无法观察到本技术在大规模测试用例集上的表现。后续我们需要扩大测试用例集规模，在测试用例分类领域为类似的研究工作提供可信的大规模数据集。

其次，尽管在数据集中已分类数据较少的情况下，本技术的分类表现优于其余诸如 SVM、DBM 等模型，但分类器准确率仍然有待提高。本文在系统层面上引入了类别标签审核机制为分类器的训练输入正反馈，但在算法层面上还需要改进类别词库收集算法，建立公共词库收集在每个类别中都经常出现的关键词，避免这些词汇在词库模式匹配机制中的误导作用。

最后，系统需要对不同主题的测试用例集都各自训练各自的分类器，当前我们仍然没有找到使用单分类器对所有测试用例准确分类的方法。后续的思路可以是对测试用例集聚类或分类，对相似的测试用例集可以训练共同的分类器。

参考文献

[1] ORSO A, ROTHERMEL G. Software testing: a research travelogue (2000–2014)[J], 2014.

[2] 李明树, 何梅, 杨达, et al. 软件成本估算方法及应用 [J]. 软件学报, 2007, 18(4): 21.

[3] MOHAMMAD S M. CONTINUOUS INTEGRATION AND AUTOMATION[J]. SSRN Electronic Journal, 2016, 4(3): 938–945.

[4] BERNER S, WEBER R, KELLER R K. Observations and lessons learned from automated testing[C] // Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on. 2005.

[5] 朱少民. 软件测试面临的挑战与发展趋势 [J]. 测控技术, 2020, 39(1): 4.

[6] 章晓芳, XIAOFANG Z, 冯洋, et al. 众包软件测试技术研究进展 [J]. 软件学报, 2018, 29(1): 20.

[7] 何琼月. 谈软件工程中软件测试的重要性及方法 [J]. 电子元器件与信息技术, 2020.

[8] DAKA E, FRASER G. A Survey on Unit Testing Practices and Problems[C] // IEEE International Symposium on Software Reliability Engineering. 2014: 201–211.

[9] YOO S, HARMAN M. Regression testing minimization, selection and prioritization: a survey[J]. John Wiley Sons, Ltd, 2012, 22(2): 67–120.

[10] 丁志军, 周泽霞, 丁志军. Web 服务组合测试综述 [J]. 软件学报, 2018, 29(2): 21.

[11] 王廷永, 黄松. 测试用例自动生成技术综述 [J]. 电子技术与软件工程, 2021(18): 3.

[12] 姚佳瑜. 软件测试中的测试用例及复用研究 [J]. 数字技术与应用, 2018(1): 2.

[13] 章晓芳, 陈林, 徐宝文, et al. 测试用例集约简问题研究及其进展 [J]. 计算机科学与探索, 2008, 2(3) : 13.

[14] KOWSARI, MEIMANDI J, HEIDARYSAFA, et al. Text Classification Algorithms: A Survey[J]. Information, 2019, 10(4).

[15] 于游, 付钰, 吴晓平. 中文文本分类方法综述 [J]. 网络与信息安全学报, 2019, 5(5) : 8.

[16] 汪岿, 刘柏嵩. 文本分类研究综述 [J]. 数据通信, 2019(3) : 11.

[17] FE-FEI L, OTHERS. A Bayesian approach to unsupervised one-shot learning of object categories[C] // proceedings ninth IEEE international conference on computer vision. 2003 : 1134 – 1141.

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

[19] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient Estimation of Word Representations in Vector Space[J], 2013.

[20] TOMXE, MIKOLOV, SUTSKEVER I, et al. Distributed representations of words and phrases and their compositionality[J], 2013.

[21] LUO Q, XU W, GUO J. A Study on the CBOW Model's Overfitting and Stability[C] // Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning. 2014 : 9 – 12.

[22] GUTHRIE D, ALLISON B, LIU W, et al. A closer look at skip-gram modelling.[C] // LREC : Vol 6. 2006 : 1222 – 1225.

[23] MOHAMMED A A, UMAASHANKAR V. Effectiveness of hierarchical softmax in large scale classification tasks[C] // 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2018 : 1090 – 1094.

[24] GOLDBERG Y, LEVY O. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method[J]. arXiv preprint arXiv:1402.3722, 2014.

[25] SU J. SimBERT: Integrating Retrieval and Generation into BERT[R/OL]. 2020. https://github.com/ZhuiyiTechnology/simbert.

[26] DONG L, YANG N, WANG W, et al. Unified language model pre-training for natural language understanding and generation[J]. Advances in Neural Information Processing Systems, 2019, 32.

[27] WANG Y, YAO Q, KWOK J T, et al. Generalizing from a few examples: A survey on few-shot learning[J]. ACM computing surveys (csur), 2020, 53(3): 1 – 34.

[28] BENAIM S, WOLF L. One-shot unsupervised cross domain translation[J]. advances in neural information processing systems, 2018, 31.

[29] QI H, BROWN M, LOWE D G. Low-shot learning with imprinted weights[C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 5822 – 5830.

[30] SHYAM P, GUPTA S, DUKKIPATI A. Attentive recurrent comparators[C] // International Conference on Machine Learning. 2017: 3173 – 3181.

[31] ZHANG Y, TANG H, JIA K. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data[C] // Proceedings of the european conference on computer vision (ECCV). 2018: 233 – 248.

[32] KOZERAWSKI J, TURK M. Clear: Cumulative learning for one-shot one-class image recognition[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 3446 – 3455.

[33] PFISTER T, CHARLES J, ZISSERMAN A. Domain-adaptive discriminative one-shot learning of gestures[C] // European Conference on Computer Vision. 2014: 814 – 829.

[34] MILLER E G, MATSAKIS N E, VIOLA P A. Learning from one example through shared densities on transforms[C] // Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662): Vol 1. 2000: 464 – 471.

[35] LIU B, WANG X, DIXIT M, et al. Feature space transfer for data augmentation[C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 9090 – 9098.

[36] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[J]. Advances in neural information processing systems, 2014, 27.

[37] 赵凯琳, 靳小龙, 王元卓. 小样本学习研究综述 [J]. Journal of Software, 2021, 32(2).

[38] KOCH G, ZEMEL R, SALAKHUTDINOV R, et al. Siamese neural networks for one-shot image recognition[C] // ICML deep learning workshop : Vol 2. 2015 : 0.

[39] SNELL J, SWERSKY K, ZEMEL R. Prototypical networks for few-shot learning[J]. Advances in neural information processing systems, 2017, 30.

[40] GAO T, HAN X, LIU Z, et al. Hybrid attention-based prototypical networks for noisy few-shot relation classification[C] // Proceedings of the AAAI Conference on Artificial Intelligence : Vol 33. 2019 : 6407 – 6414.

[41] SUN S, SUN Q, ZHOU K, et al. Hierarchical attention prototypical networks for few-shot text classification[C] // Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP). 2019 : 476 – 485.

[42] YU M, GUO X, YI J, et al. Diverse few-shot text classification with multiple metrics[J]. arXiv preprint arXiv:1805.07513, 2018.

[43] HOFFMAN J, TZENG E, DONAHUE J, et al. One-shot adaptation of supervised deep convolutional models[J]. arXiv preprint arXiv:1312.6204, 2013.

[44] WEBB P, SYER D, LONG J, et al. Spring boot reference guide[J]. Part IV. Spring Boot features, 2013, 24.

[45] 戴志诚, 程劲草. 基于虚拟 DOM 的 Web 前端性能优化研究 [J]. 计算机应用与软件, 2017, 34(12) : 21 – 25.

[46] GHIMIRE D. Comparative study on Python web frameworks: Flask and Django[J], 2020.

[47] LOKHANDE P, ASLAM F, HAWA N, et al. Efficient way of web development using python and flask[J], 2015.

[48] GARDNER J. The web server gateway interface (wsgi)[J]. The Definitive Guide to Pylons, 2009 : 369 – 388.

[49] RAD B B, BHATTI H J, AHMADI M. An introduction to docker and analysis of its performance[J]. International Journal of Computer Science and Network Security (IJCSNS), 2017, 17(3): 228.

[50] 王鹃, 胡威, 张雨菡, et al. 基于 Docker 的可信容器 [J]. 武汉大学学报 (理学版), 2017, 63(2): 102–108.

简历与科研成果

基本情况

唐昊杰，男，汉族，1998 年 7 月出生，江苏省盐城人。

教育背景

2020 年 9 月–2022 年 6 月 南京大学软件学院 硕士

2016 年 9 月–2020 年 6 月 大连理工大学软件学院 本科

致　　谢

时间过的飞快，行文将至终点，无论是刚入学时的憧憬，还是本文提笔前的踌躇，于南京大学求学生涯中那一幕幕的心情激荡起伏都仿若眼前。在这里我渡过了人生中最青葱的岁月，留下了太多美好回忆，足够多年后的我回味。

首先我要感谢我的导师陈振宇教授，感谢陈老师在我硕士期间给予我学业上的指导和生活中的关心，在此表达最诚挚的感谢。陈老师在生活中率真且风趣，祝他厨艺越来越好。

其次我要感谢赵源学长，还记得在我惴惴不安第一次来实验室时赵源学长不辞辛苦的引领我入住和请我吃饭。在之后的两年里，无论是熟悉项目还是确立学习目标，赵源学长都给予了我足够的帮助，真的很感谢他。

接着我要感谢实验室的门铎学长、韩奇学姐、薛晓波学长、郭超学长在我最初接手项目时对我的指导，谢谢他们不厌其烦的被我打扰。感谢钱瑞祥、刘智彪、王睿智、吴青衡和其他有幸相遇的同学们，感谢那段在五台花园我们一起努力学习一起畅想未来的日子。

我还要感谢我的父母和女朋友。他们是我最依赖的人，支撑着我前进。感谢我的女朋友，我们在实习和毕设的时光里一起经历各种事情，一起变得成熟和坚强，一起相互理解、鼓励和陪伴，感谢有你。

感谢所有参与毕业论文和答辩评审的专家老师们，你们辛苦了，向你们的专业和付出致敬。

季子正年少，匹马黑貂裘。感谢曾经的自己，能够来到南大这个美丽的校园，能够遇见相伴一生的人，能够选择自己心仪的岗位。希望未来的自己不忘初心，敢于任事，踏实做事，勤恳待人。