研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目 基于深度玻尔兹曼机的众包测试缺陷报告分类系统

作 者 姓 名

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

2022 年 05 月 20 日

学　　号：**MF20320153**

论文答辩日期：**2022 年 05 月 20 日**

指 导 教 师：　　**（签字）**

The Deep Boltzmann Machine Based Crowdsourcing Test Defect Report Classification System

By

(Author Name)

Supervised by

(Supervisor's position)**(Supervisor's Name)**

A Thesis
Submitted to the XXX Department
and the Graduate School
of XXX University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

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

研究生毕业论文中文摘要首页用纸

毕业论文题目：基于深度玻尔兹曼机的众包测试缺陷报告分类系统

工程硕士（软件工程领域）专业 2020 级硕士生姓名：

指导教师（姓名、职称）：

摘　要

近年来，为了更好地满足用户需求，软件应用的复杂度越来越高，软件测试的需求和规模也随之增加。在这种环境下，众包测试以其低成本、高时效的优势得以快速发展。不同于传统基于实验室的软件测试模式，众包测试通过互联网招募工人执行测试任务，参与测试的人员可能缺乏专业的领域知识且对待测应用缺乏了解，这导致众包测试报告存在质量良莠不齐、重复率高、审核难度大等问题，处理众包测试结果仍需要大量专家参与。对众包测试报告进行自动化分类能够很大程度减轻报告审核人员的工作负担，减少众测过程中的专家需求。

众包测试通常依据测试人员提交的故障数量和及时性给予奖励，使得测试人员倾向于快速提交测试报告，并且众包测试任务需要在移动设备进行，大量的文字输入并不便捷，因此，众包测试报告具有文本简短、截图丰富等特点。本文将文本和图像信息融合，设计并实现了一种基于深度玻尔兹曼机的众包缺陷分类系统，实现众包缺陷报告自动化分类。首先，利用自然语言处理技术对文本进行预处理和特征提取，利用 OpenCV 和 SPM 对图像数据进行预处理和特征提取。其次，利用深度玻尔兹曼机从每种数据模态中取样，提取文本和图像的高维特征表示，并进行特征融合。最后，利用支持向量机对高维融合特征向量进行分类，将缺陷报告分为不正常退出、功能不完整、用户体验、页面布局缺陷、性能和安全六种类别。本方法通过高维融合特征向量进一步挖掘不同数据模态之间的内在联系，使得分类结果更加准确。

本系统使用 React 作为前端框架，SpringBoot 作为后端框架，Flask 作为分类服务框架，MySQL 作为数据库提供数据存储功能，Restful 通讯协议提供数据通讯服务，Nginx 反向代理实现数据访问，Docker 实现服务虚拟化，实现了服务间的松耦合。系统主要划分为数据查看和报告分类两个功能模块。分类服务基于慕测平台的众包测试数据进行报告融合模型和分类模型的预训练。本文使用 3 个来自工业界的移动应用进行实验，实验结果表明，基于深度玻尔兹曼机的众包测试缺陷报告分类系统分类准确率达到 50.6%，证明了对该方法对众包测试场景下缺陷报告分类的有效性。

关键词：众包测试，特征融合，深度玻尔兹曼机，报告分类

研究生毕业论文英文摘要首页用纸

THESIS: The Deep Boltzmann Machine Based Crowdsourcing Test Defect Report Classification System

SPECIALIZATION: Software Engineering

POSTGRADUATE: (Author Name)

MENTOR: (Supervisor's position)(Supervisor's Name)

Abstract

In recent years, the complexity of software applications has been increasing in order to better meet user needs, and the demand and scale of software testing has also increased. In this environment, crowdsourced testing has been able to develop rapidly with its advantages of low cost and high timeliness. Unlike the traditional lab-based software testing model, crowdsourced testing recruits workers to perform testing tasks through the Internet, and the participants may lack professional domain knowledge and understanding of the application to be tested, which leads to problems such as uneven quality of crowdsourced testing's reports, high repetition rate, and difficulty in auditing, etc. The processing of crowdsourced testing's results still requires the participation of a large number of experts. The automated classification of crowdsourced testing's reports can largely reduce the workload of report reviewers and the need for experts in the process of crowdsourced testing.

Crowdsourced testing is usually rewarded based on the number and timeliness of faults reported by testers, which makes testers tend to submit test reports quickly, and a large number of crowdsourced testing tasks need to be performed on mobile devices, and a large amount of text input is not convenient, therefore, crowdsourced testing's reports are characterized by short text and rich screenshots. In this thesis, we fuse text and image information, design and implement a crowdsourcing defect classification system based on Deep boltzmann Machine, and realize the automated classification of crowdsourcing defect reports. First, the text is preprocessed and feature extracted using natural language processing techniques, and the image data is preprocessed and feature extracted using OpenCV and SPM. Second, a Deep boltzmann Machine is used to sample from each data modality, extract high-dimensional feature representations

of text and images, and perform feature fusion. Finally, a support vector machine is used to classify the high-dimensional fused feature vectors into six categories of defect reports: abnormal exit, incomplete functionality, user experience, page layout defects, performance and security. This method further explores the intrinsic connection between different data modalities through high-dimensional fused feature vectors, which makes the classification results more accurate.

This system uses React as the front-end framework, SpringBoot as the back-end framework, Flask as the classification service framework, MySQL as the database to provide data storage function, Restful communication protocol to provide data communication services, Nginx reverse proxy to realize data access, Docker to realize service virtualization, Loose coupling between services is achieved. The system is mainly divided into two functional modules: data viewing and report classification. The classification service performs pre-training of the report fusion model and the classification model based on the crowdsourced testing's data of the mooctest platform. In this thesis, we use three mobile applications from the industry to conduct experiments. The experimental results show that the Deep Boltzmann Machine based classification system for crowdsourcing test defect reports achieves 50.6%, which proves its effectiveness in classifying defect reports in crowdsourcing testing scenarios.

Keywords: Crowdsourced Testing, Feature Fusion, Deep Boltzmann Machine, Report Classification

目录

表 目 录

图 目 录

第一章　引言

1.1　研究背景

在 2006 年，Howe Jeff 首次提出众包（crowdsourcing）的概念 [1]，众包是由混合群众（crowd）和外包（outsourcing）词义结合从而产生的混成词 [2]，表示公司或机构把原本由员工执行的工作任务，以自由自愿的形式外包给非特定的（并且通常是大型的）大众网络的做法 [3]。随着众包概念的广泛传播，众包模式 [4] 也逐步发展为互联网上一种流行的商业模式。众包平台作为众包工作中的沟通桥梁，协助问题解决者和发包方顺利达成合作。众包平台促成了众包工作更好的完成，将众包模式推广到了更高的高度。

近年来，随着软件复杂度的不断提升，软件测试需求的规模逐渐增大，众包模式在软件测试领域得到飞速发展。传统基于实验室的软件测试已无法满足软件测试的需求。众包模式已经在可用性测试、GUI 测试、性能测试等方面得到广泛应用。如图1.1所示，众包测试通常由任务发起者、众包工人以及众包平台共同参与。首先由任务发起者在众包平台上发布待测软件和测试任务，之后众包工人会自由选择待测应用，执行测试任务，并在规定时间内提交测试报告，最终，审核人员审核测试报告，整合后交付给任务发起方。相比于传统测试模式，

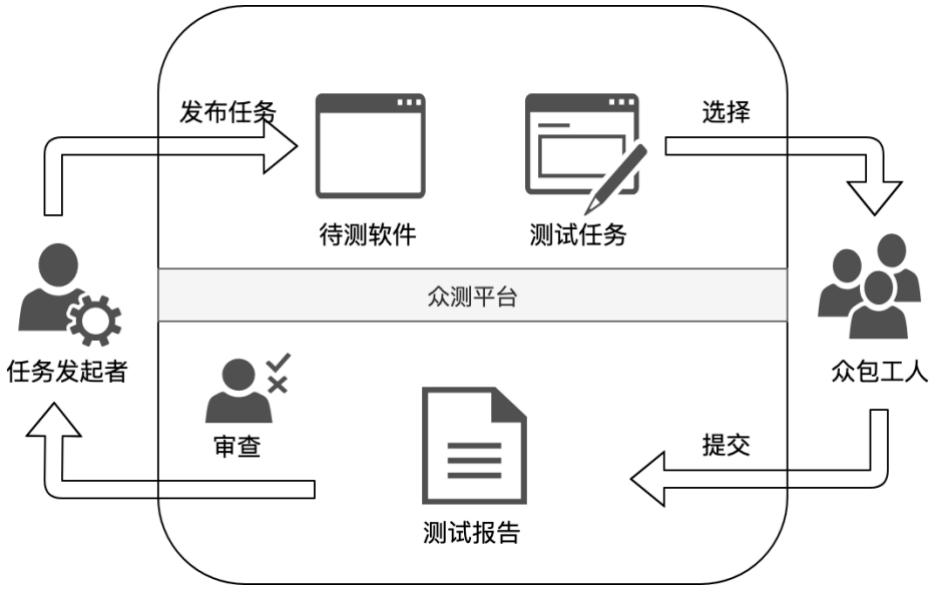

图 1.1: 众包测试流程图

众包测试具有以下显著优势：

（1）更丰富的测试环境。众包平台通过互联网招募的大量众包工人能够提供更为丰富的测试设备、操作系统、网络环境和地理位置等信息，保障了测试环境的充分性。

（2）更快速的测试迭代。随着应用软件的不断发展，快速迭代成为重要的软件开发需求。众包采用“短平快”的测试模式，招募大量的众包工人同时执行测试任务，在快速完成测试任务的同时，能够获得终端用户的使用感受，把更多问题暴露于软件开发初期，缩短软件投入应用市场的周期。

（3）更低的测试成本。众包测试平台招募终端用户作为测试人员，其成本远低于专家用户，且激励机制往往规定为有效缺陷付费，进一步降低了测试成本。

（4）更真实的用户体验。众包工人作为终端用户，除了功能鼓掌外，能够反馈更多用户体验问题，在辅助缺陷复现的同时，能够提供更真实的优化建议。

当前，国内外陆续出现了大量众包测试平台，常见的包括“百度众测”[1]、“先知平台”[2]、“testin 平台”[3]、“alltesting 平台”[4]等 [5]。然而，随着测试软件规模不断增加，众包测试参与人员也急剧上升，众包测试中存在的问题不断暴露：

（1）测试报告质量难以保障。缺陷的提交时间和有效性是众包测试平台支付报酬的重要依据，因此众包工人倾向于迅速提交测试报告，而忽略报告的质量和完整性。在实际应用过程中，这种提交方式使得测试报告数量过多，而质量参差不齐。

（2）众包工人难以协作。当前较为流行的众包测试平台仍采用竞争式众包模式，即众包工人独立执行测试任务，彼此之间没有信息交互和协作，这导致大量重复测试报告被提交至众测平台，极大程度上浪费了测试资源和报告审核成本。

为缓解上述问题，测试报告分类受到学者们的广泛关注。首先，测试报告划分为不同的类别能够判断报告的严重程度并推荐合适的开发人员。例如，功能错误（程序中存在未实现功能或功能错误）的严重程度通常远大于用户体验问题（能够提升用户体验的优化建议），应优先修复。其次，对大量测试报告分类，能够根据类别对测试报告进行筛选和分派，降低审核人员的审核压力，更好地辅助软件应用的开发。另外，为了节省成本，大多数的众包测试工人都是非专业的测试人员，仅依靠测试人员进行 Bug 分类难以保障分类的准确性。因此，如何更好的对众包测试产生的报告分类 [6] 成为了一个新的研究方向。

[1]https://test.baidu.com/

[2]https://www.bizsn.com/

[3]https://www.ztestin.com/

[4]https://zc.alltesting.cn/

目前，针对众包测试报告分类 [7] 这一领域，学术界和工业界已经有了较为成熟的研究和应用。由于众包测试中存在多模态数据，大多数传统的做法是通常将缺陷报告中的文本和图像分别处理后再进行加权（如通过朴素贝叶斯、K 最近邻（k-nearest neighbor，KNN）、支持向量机（support vector machines，SVM）、RNN（循环神经网络）处理文本分类，通过 KNN、CNN（convolutional neural network）处理图像分类）。然而，在一份由文本和图像信息组成的多模态报告中，文本和图像通常描述相同的 Bug，具有极强的关联性。这样分别处理后进行加权的做法难以捕捉的多模态数据之间的内在联系。另一方面，未来本系统可能在 Bug 报告的描述中引入其他的模态信息，如视频信息，日志信息等。由于不同模态的信息处理方式不同，如果每次都对不同模态的信息都要分别进行处理，这种方式无疑是非常低效的。因此，本系统实现了一种基于深度学习框架的众包缺陷分类系统，利用深度玻尔兹曼机实现多模态数据融合 [8]，并针对融合后的高维特征向量分类，从而提升众包测试报告的分类准确率。

1.2 国内外研究现状及分析

1.2.1 众包测试

Mao 等人 [9] 概括总结了众包测试的应用。众包测试往往依托于众测平台，测试工作由大量参与众包测试的工作人员共同完成。众包测试的发展，使得市面上出现了越来越多的众测平台。众包测试已经被广泛应用于解决软件工程问题。例如，Dolstra 等 [10] 提出了一种应用于 GUI 测试的众包测试方法，通过在虚拟机中实例化待测系统实现 CUI 测试，这些虚拟机服务于分散在各地的测试人员。Chen 和 Kim[11] 提出了 PAT, 将复杂的对象突变和约束求解问题分解成众包工人能够解决的子问题，实现众包测试。Gomide 等人 [12] 提出了一种基于众包的可用性测试流程，可以减少与传统可用性测试的时间开销和经济成本。

还有许多学者专注于解决众包测试带来的新问题，例如，Wang 等人 [13] 提出了重复测试报告检测技术，减轻测试报告审核工作。2015 年，Feng 等人 [14] 对测试报告优先级进行排序，在每次迭代中动态地为检查选择风险最大的和多样化的测试报告，辅助众包测试报告审核和开发人员的修复分派。Wang 等人试图通过采用基于聚类的分类方法 [15] 和主动学习 [16] 从原始测试报告中识别非故障报告。同样，我们的研究旨在解决众包测试问题。本文提出了一种基于 DBM 的测试报告分类算法，将测试报告中的文本和图像信息进行融合，以融合结果作为分类器的输入，以实现测试报告的自动化分类。

1.2.2 众包测试报告

众包测试与传统测试模式的重要区别体现在众包工人和测试报告两方面。不同于传统的人工测试，众包测试的参与人员是来自互联网的未知大众，对待测应用缺乏足够的了解，且不具备专业的测试技能，因此测试报告的表现形式与传统报告存在明显差异。

众包工人：传统的软件测试通常基于实验室进行，根据产品规模的大小配备一名或多名测试工程师，根据模块划分，独立完成测试任务。颜炯等人 [17] 的综述中指出，软件测试依赖于测试人员经验。基于模型的软件测试需要测试人员具备相关的专业技能、具有一定的理论基础和必要的测试经验。许静等人 [18] 分析了常见的软件测试方法，分析结果表明，测试质量与测试工程师的经验具有紧密关联。测试工程师能够为软件测试提供能多领域知识，基于一定的理论基础提供更多专业意见。然而，长期的独立测试工作易导致思维固化，难以多角度探索新的故障。众包测试人员往往不是长期从事软件测试的专业人士，且人员成多样化分布，不同的知识背景、地理位置、年龄差异会提供多样的测试方式和角度，在一定程度上缓解个人经验带来的思维固化。与此同时，这也带来了一些问题，如众包工人可能不具备专业的知识技能，提交的测试报告质量难以保障，文本描述不够精准全面，缺陷复现存在一定困难。

测试报告：测试报告是软件开发人员重现缺陷的重要基础，已有许多学者进行测试报告文本规范研究。Bettenburg 等人 [19] 提出，复现步骤、预期行为和实际行为是开发人员使用最多的信息。在传统的基于实验室的测试模式中，测试工程师会使用特定的模块和格式详细描述这些信息。由于众包测试中的激励机制设定，众包工人倾向于提交简短的文字描述以快速提交测试报告，获取更多奖励。特别是对于移动应用的测试通常只在移动设备端进行，不便于进行大量的文本输入。为了更加清晰的描述缺陷，众包工人可以提交屏幕截图，以弥补文本描述的不足。因此，众包测试报告呈现文本简短、截图丰富的特征。

1.2.3 缺陷分类系统

软件缺陷，又称为 Bug，为计算机软件或程序中存在的某种破坏软件或应用正常运行能力的错误或隐藏的功能缺陷。缺陷的存在会破坏软件产品的使用感，甚至可能直接导致产品无法使用。根据 IEEE729-1983 对缺陷的定义：从产品内部看，缺陷是软件产品开发或后期维护过程中存在的错误或其他问题；从产品外部看，缺陷是系统所需要实现的某种功能的冲突。越是在软件开发的后期，修复检测到的软件错误的成本也就会越高。

软件缺陷不可避免，甚至可以当作是软件产品的一部分。虽然现在国内外

已经有较多的众包测试平台，但是市面上缺乏专门针对众包测试报告分类的相关系统。一方面因为现有的众包平台大多只关注测试报告的产生，忽略了对后续报告分类，修复人员分派等工作。另一方面，缺陷分类系统较为依赖于众包平台的业务属性，通常只能作为平台内部的某一子系统或子功能，不易通用化。于是，本文主要关注缺陷报告的分类，而非众包测试报告的产生，旨在提高众包测试报告的分类准确率。通过分类，报告审核人员可以将任务分配给最适合解决的开发人员，进而提高缺陷修复效率，保障软件产品的质量。

重复报告检测是测试报告分类的重要相关基础工作。在缺陷存储库中，一些测试报告被标记为重复报告，这是因为这样的测试报告与其他已处理的测试报告非常相似。Runeson 等人 [20] 和 Wang 等人 [13] 提出基于自然语言处理技术的重复报告检测技术。Wang 等人 [15] 提出了一种结合自然语言和执行信息检测重复报告报告的方法。但是，他们只是删除了重复的报告，并且未对其进行分类，这意味着他们不能为测试报告的审核提供更多支持。

许多研究也集中在测试报告分类上。Zanetti 等人 [21] 提出了一种自动分类方法并将其集成至 Bug 跟踪平台中。Tian 等人 [22] 提出了一种多因素分析技术 DRONE，用于对测试报告的优先级进行分类。Zhou 等人 [23] 提出了一种结合文本挖掘和数据挖掘相关方法的混合方法实现测试报告分类。但是，改方法不能处理包含屏幕截图的测试报告，因此不能应用于众包测试报告分类。

总体来讲，目前现有的缺陷分类方法主要针对单一模态进行分类，且主要是针对文本特征分类。通过对测试报告中的文本信息（通常包含报告标题，缺陷描述等）进行数据清洗，分词提取，词频统计，特征提取 [24] 后，对提取的文本特征按照某一分类方法分类。但是一份标准的众包测试报告往往不仅仅由文本信息组成，通常包含了文本信息，图像信息（如缺陷截图），未来还可能会有视频信息等。因此这种单一模态并不能很好的表示一份报告中包含的信息，从而造成分类结果并不理想。此外，由于不同数据模态的数据具有不同的表示和相关结构，多模态数据的融合也变成了一个研究热点。结合众包测试自身包含文本和图像的特点，基于多模态特征融合的分类，是最适合本项目中缺陷报告分类的处理方法。

1.3 本文的工作内容

本文主要针对众包测试中缺陷报告分类的场景展开分析和研究，力图降低报告审核人员的压力，保证众包测试服务和软件开发工作更好的开展。本文主要阐述的分类服务，是利用分类器对从缺陷报告中提取到的文本和图像数据进行特征提取和融合后的高维融合向量进行分类，将收集到的无规律的缺陷报告

映射到已定义好的缺陷类别中。本文的主要工作内容为:

(1) 分析研究现状，明确研究方向，拆解工作内容。基于当下的研究现状做好充分的调研和分析，明确工作主要面向多模态的高维特征向量分类问题开展，设计并实现了包括缺陷报告管理模块、缺陷报告特征提取模块、缺陷报告特征融合模块以及缺陷报告分类模块四个模块，分离出系统的功能性和非功能性需求。

(2) 设计和实现了缺陷报告管理模块。缺陷报告管理模块是整个系统的基础模块，负责响应客户端的请求以及调度其他服务。主要处理登录授权，数据查看以及分类服务。其中，分类服务依赖服务器部署的另一服务模块。另一方面，缺陷报告管理模块直接对接数据库，满足对数据的增删查改需求。

(3) 设计和实现了缺陷报告特征提取模块。缺陷报告特征提取模块是缺陷报告特征融合模块的前置模块，为特征融合做下了准备工作。主要对原始的缺陷报告中的文本数据和图像数据进行预处理，最终以特征向量的方式对数据进行描述。

(4) 设计和实现了缺陷报告特征融合模块。缺陷报告特征融合模块是系统的核心模块，负责将缺陷报告特征提取模块中提取到的文本向量和图像向量进行融合，得到融合特征向量，以高维特征向量描述了缺陷报告数据。同时，融合向量作为分类模块的分类输入。

(5) 设计和实现了缺陷报告分类模块。缺陷报告分类模块是系统的重要功能模块，负责将缺陷报告特征融合模块中得到的融合向量映射到已定义好的分类标签上，最终提供给用户分类结果。

综合上述工作内容，整个系统将以缺陷报告管理模块基础，以缺陷报告分类模块为核心功能，保证了对系统需求的完整响应，并通过服务隔离，实现了不同服务之间的低耦合。系统以 MySQL 作为数据存储，确保了数据持久化，以浏览器作为客户端入口，满足了用户的基本使用场景。

1.4 本文的组织结构

本文主要分为七个章节，具体的组织结构如下: 第一章为引言。主要介绍了众包缺陷分类系统的研究背景，并从众包测试，众包测试报告和缺陷分类三个方面分析国内外研究进展。最后简要介绍了众包缺陷分类系统的主要工作和论文的组织结构。

第二章为相关技术。主要介绍了实现众包缺陷分类系统需要使用到的相关技术框架和核心算法。总体包含了本方法中需要的文本预处理相关技术，图像预

处理相关技术，用于特征融合的深度玻尔兹曼机，用于分类的 SVM，以及工程实现中需要的前端框架 React，后端框架 SpringBoot，数据库 MySQL 以及 Nginx 服务器。

第三章为缺陷报告分类系统的需求分析与设计。首先对系统进行了整体概述，之后从涉众分析开始，明确了系统需求，并通过用例图和用例说明对功能性需求进行了详细描述。此外还对系统实现展开设计，从系统的整体架构开始分析，通过“4+1”视图进行了辅助说明。再逐步对系统的各个模块进行分析设计，并提供了各个模块的流程设计图。

第四章为缺陷报告分类系统的详细设计与实现。采用交互顺序图，数据类图，关键代码相结合的方式，分别对缺陷报告管理模块，缺陷报告特征提取模块，缺陷报告特征融合模块和缺陷报告分类模块的具体实现进行了详细说明。

第五章为系统测试与实验分析。通过对系统进行功能测试、性能测试和效果测试，验证了系统的可用性和可靠性，并说明了系统的实用价值。

第六章为总结与展望。总结了本文所有的工作内容，针对系统现有的不足提出了改进方向。

第二章　相关技术

2.1　文本处理及其相关技术

从众包测试中获取到的测试报告通常会包含报告标题，缺陷描述和缺陷截图这些重要信息 [25]。本系统将测试报告中的报告标题和缺陷描述进行拼接，提取作为测试报告的文本信息。之后根据下图2.1示意的流程对测试报告的文本信息进行预处理。主要包含文本预处理，特征选择 [26]，以及特征提取 [27] 过程。对于从众包测试数据集中获取到的一份报告数据，首先通过 LTP（Language Technology Platform）库进行预处理，包含去除停用词 [28]、同义词转换 [29] 和文本分词等自然语言处理 [30] 工作。再将输出后的文本数据通过 TF-IDF（Term Frequency–Inverse Document Frequency) 方法进行词频统计 [31]，得到中间阶段的文本向量，并进行归一化，以便将归一化后的文本向量输入深度玻尔兹曼机（Deep Boltzmann Machine，DBM）得到提取后的高维文本向量。

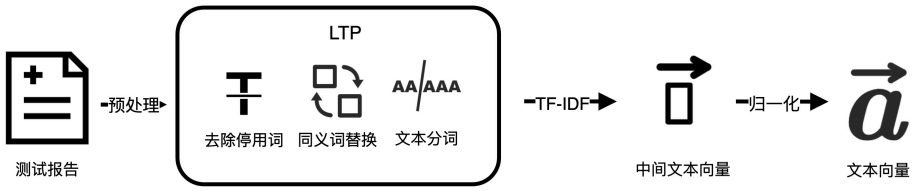

图 2.1: 文本处理流程图

2.1.1　LTP

本系统的文本处理工作借助 LTP 库完成。LTP（Language Technology Platform）作为当前流行的自然语言处理系统，基于 GPL 协议，由哈尔滨工业研究大学社会计算与信息检索研究中心研发。LTP[32] 的语言处理结果表示基于 XML 体系，其自然语言处理模块方便快捷，功能丰富，又高效准确，同时还包含多项常见的中文的语言处理分析技术。并且，LTP 提供了便捷的使用接口。开发者还可以像使用流水线一样的方式将 LTP 支持的各项工具结合使用，为需要进行自然语言处理工作的开发者提供了一个很好的选择，可以更加高效的处理各项自然语言处理任务。

本系统使用 LTP 进行了“去除停用词”、“同义词转换”、“文本分词”操作。首先根据哈工大提供的停用词词库去掉文本中意义不大的停用词，停用词通常是指连词、虚词和语气词等无意义词，对文本分析往往没有帮助。之后进行同义

词转换，提高特征计算的可靠性。最后进行分词，把语句拆分成词汇。

2.1.2 TF-IDF

TF-IDF（term frequency–inverse document frequency) 是一种加权技术，常用于信息检索与数据挖掘场景 [33]。本系统中用于进行词频统计。TF-IDF 有两方面的含义，TF(Term Frequency) 表示词频，IDF(Inverse Document Frequency) 表示逆文本频率指数。词频指的是全文段中该词条出现的频率。通常会用词频除以文章的总词数对词频进行归一化, 以确保它不会偏向长文件。TF-IDF 可根据 TF 和 IDF 计算得到，如下公式所示。对一段文本来说，如果文本中的某个词的 TF-IDF 值越大，一般可以理解为这个词在这段文本中的重要性更高。所以可以通过对报告中的文本计算 TF-IDF 词，根据大小排序后选择出该报告的关键词。

$$TF_w = \frac{\text{在某一类中词条 w 出现的次数}}{\text{该类中所有的词条数目}} \tag{2.1}$$

$$IDF = \log \frac{\text{在某一类中词条 w 出现的次数}}{\text{该类中所有的词条数目} + 1} \tag{2.2}$$

$$TF - IDF = TF * IDF \tag{2.3}$$

例如文段：

成功新增邮件，提示操作成功，用户体验友好。

成功删除邮件，提示操作失败，用户体验糟糕。

经过分词处理后的词汇表为：[成功，邮件，新增，提示，操作，成功，友好，用户，体验，删除，失败，糟糕]

经过 TF-IDF 得到的词频：

表 2.1: 文档集 TEXT1 的 TF-IDF 表

text1	成功	邮件	新增	提示	操作	完成	友好	用户	体验
TF	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
IDF	-0.0005	-0.0005	0.692	-0.0005	-0.0005	0.692	0.692	-0.0005	-0.0005
TF-IDF	-0.00006	-0.00006	-0.07681	-0.00006	-0.00006	-0.07681	-0.07681	-0.00006	-0.00006

表 2.2: 文档集 TEXT2 的 TF-IDF 表

text2	成功	邮件	删除	提示	操作	失败	糟糕	用户	体验
TF	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
IDF	-0.0005	-0.0005	0.692	-0.0005	-0.0005	0.692	0.692	-0.0005	-0.0005
TF-IDF	-0.00006	-0.00006	-0.07681	-0.00006	-0.00006	-0.07681	-0.07681	-0.00006	-0.00006

2.2 图像处理及其相关技术

测试报告中还包含了对缺陷的截图，这些图像信息也需要进行预处理。图像信息首先需要从云端下载，之后依次进行尺寸调整、特征提取和高维特征提取，如下图2.2所示。首先，我们需要从云上拿到所有的图像信息，并存储到本地。由于不同的测试工人提供饿截图大小并不一致，因此我们需要调整报告中所有图像的尺寸。接着，采用 SPM[34] 提取出图像信息中的特征并归一化，最后通过 DBM 进行高维特征提取，作为 DBM 网络的输入向量。

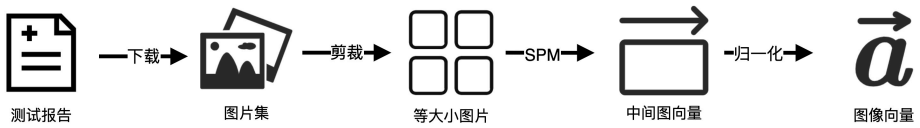

图 2.2: 图像处理流程图

2.2.1 OpenCV

由于众包测试工人设备不一致性等原因，提交的测试报告中缺陷截图尺寸也不一致，为了统一计算，需要对所有的截图数据进行尺寸剪裁的操作。本系统采用 OpenCV 统一处理图像数据。OpenCV[35] 是常用的基于 Python 的计算机视觉库。OpenCV 基于开源协议，并且可以运行在 Linux、Mac OS、Windows 和 Andriod 操作系统上。同时，OpenCV 轻量便捷，提供了很多的图像处理和计算机视觉方向想换的常用算法，支持 Python、Java 等多种语言的开发和调用，为不同语言的开发者提供了便捷的调用接口，可以更好得辅助开发者进行图像相关的数据处理工作。这使得开发者可以将更多的精力专注于开发算法本身。最终，本系统将所有的图像数据统一处理为相同像素。

2.2.2 SPM

SPM(Spatial Pyramid Matching)[36] 是一种基于空间金字塔的图像处理算法，由 Lazebnik 在 2006 年提出 [37], 是对 BOF(Bag Of Features) 算法的改进。

SPM 的本质是将图像看作空间金字塔的第一层，然后对图像进行划分，反

复一步一步划分后，最终就会形成空间金字塔模型。BOF 根据整张图像计算特征点的分布特征，得到全局直方图，因此不能得到整张图像的全部信息，对图像的识别程度低。而 SPM 考虑了空间信息，将图片划分为不同的子区域，支持在不同分辨率上统计图像特征点的分布，进而获取图像的局部信息。因此，SPM 获取到的图像信息更细节，更全面，因而可以对图像进行更精准的描述。

2.3 深度玻尔兹曼机

2.3.1 多模态学习概述

事物发生或体验的方式可以被称为一种模态。现实中也经常会遇到非单一模态的信息。例如音视频和文字结合可以更全面的进行信息描述，也可以覆盖更多的场景。在本系统中，测试报告中也包含了文字和图像两种模态信息，图像对缺陷进行了反馈，文字则是对缺陷的进一步描述。不同模态信息之间的统计特性不同，导致发现不同模态之间的潜在关系变得更加困难。如何对多模态数据提取出好的特征表示也变成了机器学习领域关注的重要问题。多模态学习 [38] 在多模态的场景下建立了一种模型，目的在于尽可能的挖掘不同模态之间的隐藏信息，得到信息的精确表示。

多模态表示 [39] 主要包括两大研究方向：联合表示和协同表示。联合表示类似于“多对一”，最终会用统一的模态描述出不同的模态信息，常用在预测时所有模态的数据全都出现的情况，而协同表示更像是“一对一”，不同模态会在满足某种约束的前提下被映射到自己本身的向量空间中，如线性相关约束等，常用在预测时仅有一种模态出现的情况。

多模态融合 [40] 通过整合多种模态的信息得到向量的高维表示。一方面，从多种模态中提取信息更具备多样性，覆盖到的信息更全面，可以挖掘到很多潜在的辅助信息，这样通常会得到更具鲁棒性的结果 [8]。另一方面，多模态融合可以在单一模态缺失时仍正常运行。目前常见的特征融合方法有按照人工规则线性融合，计算多个向量相似度矩阵后按照相似度进行融合，直接拼接特征向量等。

2.3.2 深度玻尔兹曼机概述

玻尔兹曼机（Boltzmann Machine,BM）[41] 是一种基于能量的网络。从机器学习的角度来说，我们只需要知道这种能量网络通过调节参数，维持保持能量最低的趋势即可 [42]。玻尔兹曼机对应的联合概率分布公式可以如下表示。

$$P(x) = \frac{exp(-E(x))}{Z} \tag{2.4}$$

其中，能量 E 越小，则对应状态发生的概率的越大。Z 是用于归一化的分配函数。这种基于能量的模型是一种学习概率分布的通用方法，尤其是当并不清楚给定数据集的潜在分布形式时。

受限玻尔兹曼机（Boltzmann Machine，RBM）[43] 是一种特殊的玻尔兹曼机，它包含一个输入可视层 v 一个隐藏层 h，两层之间用 w 表示连接权重，如下图 2.3所示。可见层 v 用来表示观测数据，这个观测数据也是实数，也是二值。一个标准受限玻尔兹曼机的可视层和隐藏层单元通常也都二值。

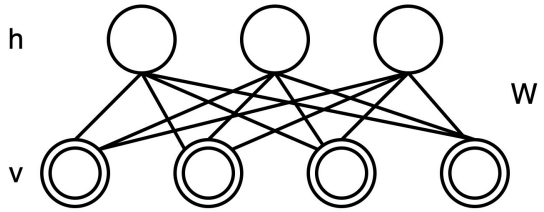

图 2.3: RBM 示意图

如图 2.3，v_T 表示可见输入层，h_T 表示隐藏层，W为单元连接权值，表示隐藏层和可见层之间的权值矩阵。

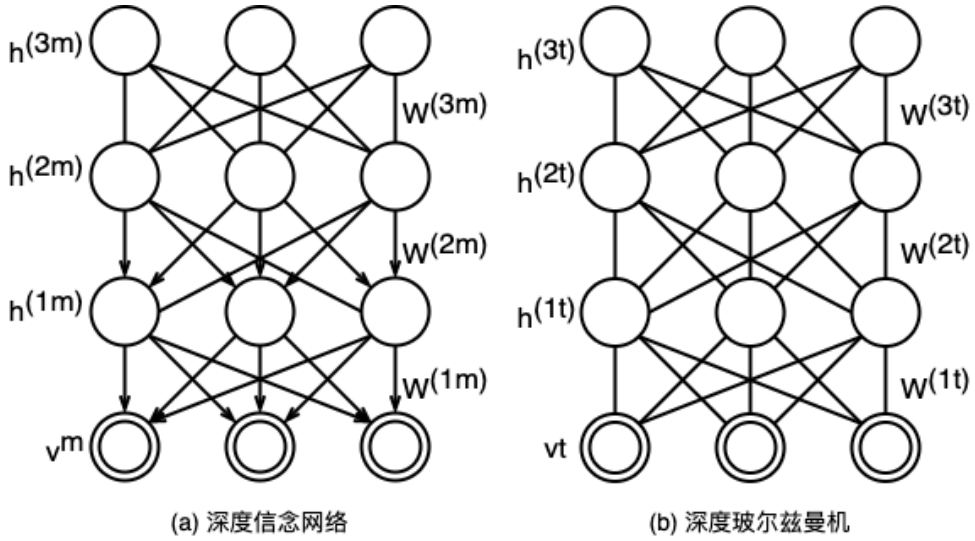

图 2.4: DBN-DBM 对比图

深度玻尔兹曼机（Deep Boltzmann Machine，DBM）[44] 是一种以受限玻尔兹曼机为基础的深度学习模型，其本质仍然是一种特殊构造的神经网络。可以理解为深度玻尔兹曼机由多个受限玻尔兹曼机堆叠而成 [45]。二者的相似之处在于只有相邻的两层神经元之间才可以连接，同一层或者非相邻层之间的神经

元之间不存在连接。二者的不同之处在于，受限玻尔兹曼机只包含一个隐藏层，而 DBM 可以有多个隐藏层。

从层级结构出发，深度玻尔兹曼机在层级结构上又和深度信念网络相似，二者都是生成式的多层级模型，相似之处在于都以受限的玻尔兹曼机为组件。二者的区别在于深度玻尔兹曼机网络是无向图模型，其层级结构图中的所有连接都是无向的，而深度信念网络图巧妙结合了有向图和无向图的生成模型，只有最高的两层是无向连接 [46]。因为 DBN 是有向且逐层训练的，所以前层的分布并不依赖于后层。如图2.4表现出了二者的差异，其中 v 表示可见输入层，h 表示隐藏层，W为单元连接权值。左图代表了深度信念网络，右图代表了深度玻尔兹曼机模型。可以看到左图最上两层是无向连接的 [47]。

图2.5描述了用于进行多模态特征融合的三层 DBM 示意图。其中 v 表示可见输入层，h 表示隐藏层，W为单元连接权值。左图表示提取图像特征向量的 DBM，中间图表示提取文本特征向量的 DBM，右图表示进行特征融合的 DBM。

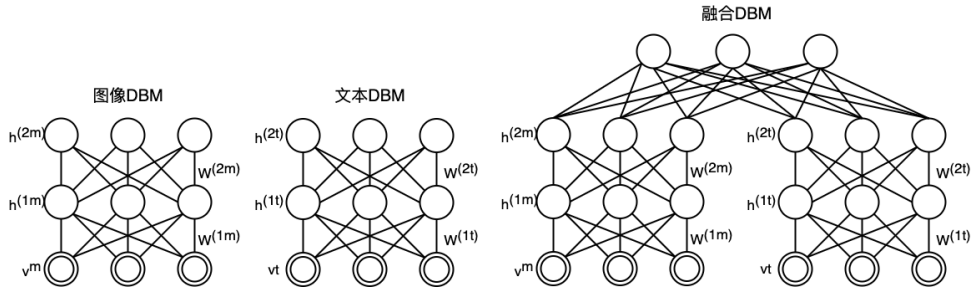

图 2.5: 三层 DBM 示意图

构建一个简单的三层 DBM 的步骤为：

（1）首先进行预训练，通过对比散度算法训练出两个受限玻尔兹曼机；

（2）根据逐层计算隐藏层数据，每相邻两层都可以认为是一个 RBM，前一个 RBM 的隐藏层可以看作后一个 RBM 的可见层；

（3）结合两个受限玻尔兹曼机，添加二进制隐藏单元网络进行特征融合；

（4）通过反向传播算法调整模型权重向量和偏置向量。

包含三层隐藏层的深度玻尔兹曼机，联合概率分布为：

$$P(v,h^{(1)},h^{(2)},h^{(3)}) = \frac{1}{Z(\theta)}e^{-E(v,h^{(1)},h^{(2)},h^{(3)};\theta)} \tag{2.5}$$

其能量形式为：

$$E(v,h^{(1)},h^{(2)},h^{(3)};\theta) = -v^T W^{(1)}h^{(1)} - h^{(1)T}W^2h^2 - h^{(2)T}W^{(3)}h^{(3)} \tag{2.6}$$

其中，E 为可见层与输入层之间的能量函数；v 表示可见输入层（T 表示转置），h 表示隐藏层，θ 表示整个模型的参数；v 为可见节点状态；$h^{(1)}$ 为第一层隐单元状态；$W^{(1)}$ 为第一组单元连接权值；$h^{(2)}$ 为第二层隐单元状态；$W^{(2)}$ 为第二组单元连接权值；$h^{(3)}$ 为第三层隐单元状态；$W^{(3)}$ 为第三组单元连接权值；联合概率分布表示映射到某一合成向量上的概率，能量 E 确保模型稳定性，最终输出得到所述文本和图像的合成向量。

2.4 支持向量机

支持向量机（Support Vector Machine,SVM）[48] 是一种常见的用于分类与回归分析中的分类学习算法，最初是为了求解二分类问题提出，现在已经被用于小样本学习。当需要解决的问题包含多分题时，就需要另外训练多分类的 SVM。SVM 的主要思想可以概括为两点:

(1)SVM 针对线性可分情况进行分析。当遇到线性不可分的情况时, 使用非线性映射算法将低维输入空间中线性不可分的样本转化为高维特征空间，使其线性可分, 这就促使高维特征空间采用线性算法对样本的非线性特征进行线性分析成为可能;

(2) SVM 基于结构风险最小化理论。SVM 在特征空间中建构最优分割超平面, 使学习器得到全局最优化, 并且在整个样本空间的期望风险以某个概率满足一定上限。

SVM 不但算法简单, 而且具有较好的鲁棒性, 而且具有较高的准确率。如今已经广泛被用于求解分类问题。

2.5 工程技术

2.5.1 React

React 是由 Facebook 创建的 JavaScript 库 [49]，是目前前端三大框架之一，因其声明式、组件化的特点受到前端开发者的喜爱。无论当前项目的技术栈是什么，都可以在无需重构现有代码的前提下，通过引入 React 来快速搭建新的功能。

React 采用声明式渲染，视图层的 DOM 只是数据状态的映射。当数据变动时，React 可以高效的更新并渲染组件，基于数据绑定的原理，降低了视图层和数据层的耦合，保证了数据和视图的一致性。

React 采用组件化的思想。React 将页面和构成页面的元素都可以称之为组件。构建管理自身状态的单个组件，通过组合多个组件形成复杂组件。组件只需

要关心自己的逻辑，彼此独立。这种基于组合模式的开发方式大大不仅便于代码复用，还降低了代码的维护难度，实现了组件之间的解耦。同时，React 的生命周期钩子允许开发人员在组件实例化、完成渲染、属性更新、组件销毁等不同阶段操作组件。

React 引入了 Virtual DOM 的概念。前端本身需要通过 js 处理 DOM 来更新试图数据，而大量的 DOM 更新一定会影响到系统的性能。React 在内存中生成了与浏览器中真实 DOM 相对应的数据结构，这种在内存中生成的 DOM 便成为 Virtual DOM。通过 React 实现的 DOM Diff 算法，可以计算出 VIrtual DOm 与真实 DOM 之间的差异，最大程度上的减少页面的重排和重绘，尽可能减少 DOM 操作，实现高性能的 DOM 渲染。

React 采用单向数据流。使用单向数据流使得代码逻辑更加清晰，数据变动的来源易于追溯，降低了数据管理成本，便于快速定位问题。

在项目中我们还使用了其他的 React 模块。我们用 react-router 来实现页面之间的无刷新跳转，采用了 react-redux 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。

2.5.2 SpringBoot

SpringBoot 是由 Spring 社区提供的开源框架，用于辅助开发者快速简单的搭建起项目框架，简化了项目的开发，配置，部署等流程，降低了项目开发的难度和复杂度，使开发者可以更多的关注项目本身的逻辑而非开发环境的准备。

SpringBoot 继承自 Spring 框架，是 Spring 框架的增强。使用 Spring 生态开发的主要优势在于它对 Java 对象的生命周期管理，强大的控制反转，灵活简单的切面编程 (AOP)，方便的注解驱动和完善的数据访问方式等，为 Java 应用程序的开发提供了全面的技术支撑和保障。另一方面，不同于 Spring 需要大量的 XML 配置文件，SpringBoot 提供了配置模版来辅助快速搭建启动项目，简化了项目的配置和启动过程。SpringBoot 是构建 Web 应用最先进的 Web MVC 框架。

控制翻转是设计模式中提到的软件开发原则，Spring Core 中的 Ioc 部分是一个标准的依赖反转实现框架。依赖翻转可以将 Java 对象交给 Spring 容器管理，而不需要用户手动将依赖通过属性设置器或者构造函数进行配置。控制翻转将创建对象以及查找依赖对象的权限交给了 Ioc 容器，并通过容器实现了对象的有序组合，这么做的优点是既能够使对象间都是松耦合，方便进行测试，也便于功能复用，还可以使程序的整个体系结构变得较为灵活。

AOP 虽然不在 Spring Core 体系中，却是一个非常好用的附加模块。如果使用 AOP 功能，需要额外导入 AOP 相关的依赖。AOP 的含义是面向切面编程，其

主要思想为利用横切技术将需要涉及到多个类或方法的通用行为封装到通用的模块，再动态插入到我们需要使用的方法或者是类上。这些通用模块也可以称为增强点，需要插入的类或者方法称为连接点。这样处理的好处是可以将与业务无关的代码抽取到公共的地方进行统一维护，尽可能复用代码，实现系统的低耦合，提高系统的可操作性，也使得系统后续便于开发者维护。

JPA 是为了整合第三方 ORM 框架而建立的一种标准。Mybatis 和 Hibernate 是两个目前比较流行的基于 JPA 标准的框架，其中 Hibernate ORM 是最成熟的 JPA 实现之一，通过将 Hibernate 整合到 Spring 中，开发者只需要通过编写具有特殊格式的接口就可以完成数据库的增删改查，减少了 SQL 语句的编写工作，提高了开发效能和 SQL 质量。

SpringBoot 提供了用于简化构建配置的 starters 启动器，简化了开发者对 Maven 依赖的配置过程，减少了版本依赖之间的冲突，方便用户更好地集成各种第三方依赖和框架，使整合一些常用功能变得更加便捷。SpringBoot 具有强大并且易用的数据库，也可以很方便的完成和各类数据哭之间的交互。

2.5.3 MySQL

MySQL 是目前最流行的开源关系型数据库之一，具有运行速度较快、性能较高等特点。作为一款完全免费的产品，被广泛地应用在互联网上的各类网站开发中。

MySQL 支持多种数据库存储引擎。这些引擎能够适应于各种的使用场合，开发人员可以在不同的需求下选用不同的引擎以达到最高的效率和性能。同时在陆续的版本中 MySQL 逐步添加了事务、视图、触发器等特性。

MySQL 支持多种开发平台，如主流 Linux，Windows 等，具备良好的平台移植性。这使得在任何平台下编写的程序都可以更便捷的移植，而不需要做太多的修改。同时，MySQL 可以和多种语言进行很好的结合，如 Java，C，C++，PHP 等，同时提供了高效便捷 API 供开发者使用，提升开发效率。

MySQL 的运行速度极快。它的索引结构使用了 B 树来存储数据节点，可以有效的节约内存空间、减少查询数据时访问磁盘的次数。另一方面，MySQL 函数依赖的类库高效稳定，具有极高的稳定性。

MySQL 的安全性较高。MySQL 拥有完备的密码系统，同时对不同的使用者划分了不同的权限级别，为 MySQL 的安全提供了保障。MySQL 链接数据库服务时，所采用的密码传输均为加密传输，严格确保了访问安全性。

2.5.4 Nginx

Nginx 是当前最为流行的轻量级 HTTP 和反代理服务器之一，以其自由、开源、低内存、高并发的特点受到了开发者的喜爱。Nginx 可以作为一个 HTTP 服务器发布处理网站系统，也可以作为反向代理服务器进行负载均衡的实现。Nginx 的技术架构如图 2.6所示。

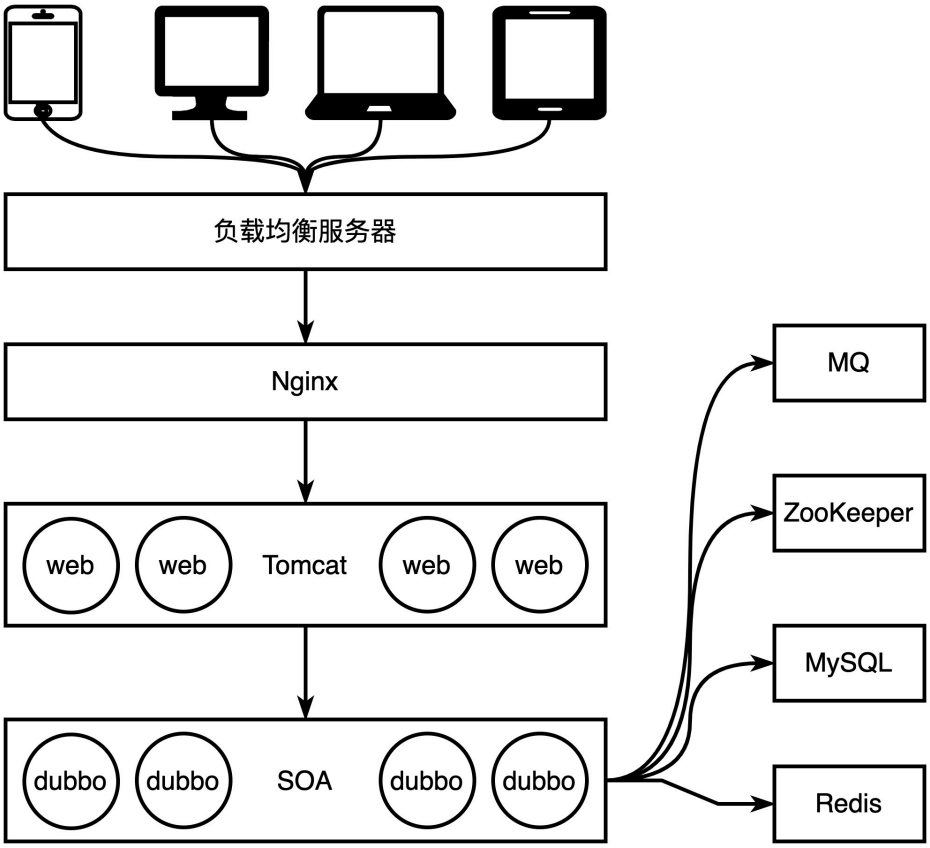

图 2.6: Nginx 技术架构图

当 Nginx 作为反向代理使用时，面向的对象是服务端，它代替服务端接受请求，主要应用于服务器集群分布式部署的情景。反向代理的情况下，由于客户端并不需要清楚具体的服务器地址，所以 Nginx 将具体的服务器信息隐藏，只向客户端暴露公网访问地址。多个客户端给服务器发送请求，Nginx 在接收到客户端请求后，根据既定分发策略交给不同的服务器处理。此时，请求的客户端信息是已知的，但是用于处理请求的服务器信息则是未知的。这样既可以保证内网安全，也可以做负载均衡，优化网站的加载性能。

当 Nginx 作为正向代理使用时，面向对象是客户端，它代替客户端发送请求，主要用于访问原来无法访问的资源的场景。正向代理的情况下，客户端清楚

请求的服务器信息，而服务器清楚代理请求的代理服务器信息，并不清楚发起请求的客户端信息。多个客户端给代理服务器发送请求，再由代理服务器将请求转发到目标服务器上，然后代理服务器将获得的请求内容返回给客户端。Nginx代替客户端发出请求，充当一个位于客户端和原服务器之间的服务器角色。这样既可以访问到本来不可访问的内容，也可以做缓存来加速资源访问的速度，代理还可以记录用户访问的记录，但是对外隐藏用户信息。

2.6 本章小节

本章主要介绍了基于深度玻尔兹曼机的众包测试缺陷报告系统主要使用的算法理论和技术框架。首先对文本和图像处理的相关技术进行了介绍，之后对多模态学习和深度玻尔兹曼机进行介绍，最后对系统实现依赖的工程技术进行介绍，包括前后端技术框架，和运维部署相关的工具。

第三章　众包缺陷分类系统的需求分析与设计

3.1　系统整体概述

由于众包测试工人自身非专业性，且众包平台环境无法统一的原因，使得产生的大量测试报告审核和分类困难，因此亟需一种自动化分类方式来辅助进行报告分类，降低众包审核人员的审核压力，也可以为后续的缺陷修复和任务分配工作提供帮助。本系统作为众包缺陷分类系统，针对众包场景下的测试报告分析问题，先利用深度玻尔兹曼机进行文本和图像模态的特征融合，再针对高维 MySQL 的特征向量进行分类，将测试报告划分为不正常退出、功能不完整、用户体验、页面布局缺陷、性能和安全六类。

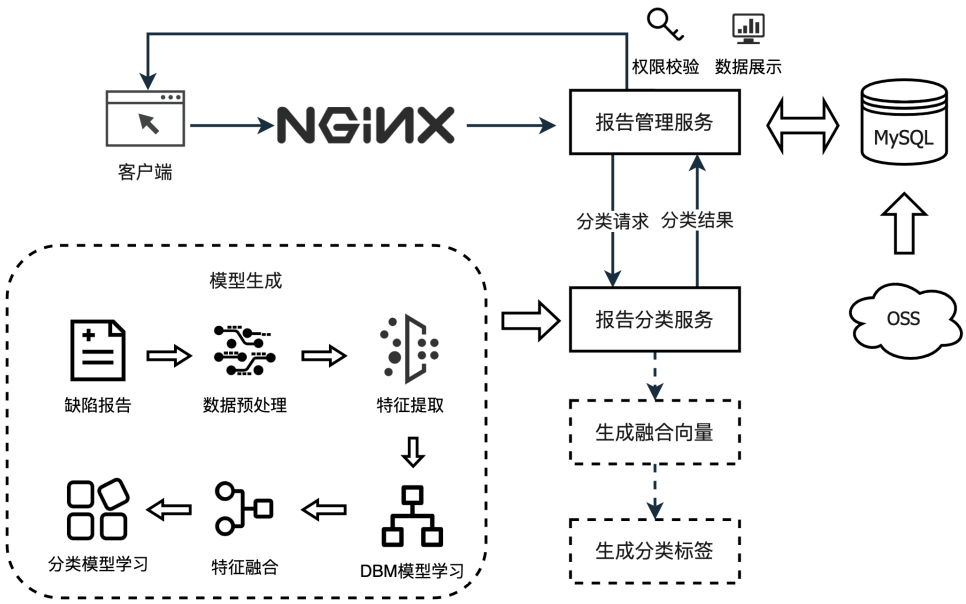

图 3.1: 系统整体概述图

如图3.1所示本系统的整体概述。用户通过浏览器端登录系统，通过 nginx 服务器转发到对应的系统后端开始处理用户请求。报告管理服务作为后端基础，负责响应前端请求，与数据库进行数据交互，MySQL 的初始数据从 OSS 中下载得到。当用户进行分类请求时，报告管理服务会调用报告分类服务，报告分类服务会先将生成报告得融合向量，再将融合向量映射到已定义好的分类标签上。其中，特征融合模型和分类模型都是提前训练过的。

报告分类服务会利用提前分类过的历史报告进行模型学习。首先，会对初始

的缺陷报告中的文本数据和图像数据进行数据预处理。文本数据会先借助 LTP 完成去除停用词、同义词转换和分析，再借助 TF-IDF 完成文本特征提取，并进行特征向量归一化。图像数据会先借助 CV2 进行尺寸裁剪和灰度化处理，再借助 SPM 完成图片特征提取，并进行特征归一化。通过预处理后的数据进行 DBM 模型学习，得到收敛稳定的 DBM 模型参数。再通过确定参数的 DBM 进行特征融合，最后借助高维融合向量学习出 SVM 分类模型。

3.2 系统需求分析

3.2.1 涉众分析

本系统主要基于众包测试报告进行分类，因此涉众人员包括参与分类的众包审核人员以及系统管理员。根据自身的涉众特征与期望，审核人员主要对众包测试报告进行分类，管理员则需要对系统的测试报告数据，用户数据进行管理。下表3.1列举出本系统的涉众人员分析。

表 3.1: 系统涉众分析表

涉众类别	涉众特征与期望
审核人员	作为众包分类系统的审核人员，同样也是该系统的主要用户，需要具备软件工程和软件测试的专业知识；了解众包测试；对软件的缺陷分类有一定理解；具备良好的工作沟通能力和合作能力；
系统管理员	作为众包分类系统的管理员，需要了解专业的软件工程专业知识；了解众包测试；熟悉系统的所有功能；具备管理系统的所有数据的能力，主要包括用户数据和测试报告数据；具备一定的系统管理和维护能力

3.2.2 功能性需求

软件需求是指用户为了解决某种问题或者为了达到某个目标需要具备的软件功能。软件的需求文档一般会包括所开发软件的功能性需求和非功能性需求。功能性需求是指系统显而易见需要实现的主要功能以及需要提供的主要服务。功能需求是软件产品必须包含的核心需求，是软件创造价值的基础。随着软件产品的不同，需要满足的业务需求不同，功能需求一般也大不相同。

本系统的涉众主要是众包测试平台的系统审核员。审核员用户（下文统一称为用户）首先需要登录系统以获取对系统功能的使用权限。接着，用户需要查看所有的应用列表。对经过众包测试的软件应用进行报告分类，因此需要可以查看应用列表。在应用列表中选择任一应用后就可以查看到该应用下的所有测试报告，即查看测试报告的功能。系统根据功能将测试报告又分为已分类测试报告和未分类测试报告。从任一测试列表可以点击进入测试报告的详情界面，

查看测试报告的标题、缺陷描述、缺陷作者、缺陷截图等信息。不同的是，已分类测试列表中的测试报告已经分好类，不支持再次修改类别。而未分类测试报告列表中的报告并未分类，用户可以从详情界面逐个手动分类，也可以直接在未分类列表中一次选择多个测试报告，统一进行自动化分类。最后，用户可以查看当前应用下所有报告的分类统计数据。

表 3.2: 系统功能需求列表

需求编号	需求名称	需求描述
RQ1	用户登录	用户登录系统以获取操作系统的权限。
RQ2	查看应用列表	用户可以查看所有应用，且应用列表按分类进度排序。
RQ3	查看已分类测试报告列表	用户选择具体的应用后，可以看到该应用下的所有已分类的测试报告。已分类的报告可以看到归纳过的类别。另外还包含缺陷名称、缺陷详情、缺陷截图等信息。
RQ4	查看未分类测试报告列表	用户选择具体的应用后，可以看到该应用下的所有未分类的测试报告。另外包含缺陷名称、缺陷详情、严重程度、缺陷截图等信息。
RQ5	查看报告详情	用户在所有列表、已分类报告列表和未分类报告列表都可以点击后查看报告详情。包含报告标题，描述，作者，文本描述，截图等。
RQ6	手动分类单个报告	用户进入未分类报告列表后，可以在未分类列表中选择任一报告手动分类。
RQ7	自动分类多个报告	用户进入未分类列表后，可选择多项测试报告后自动分类。
RQ8	查看测试报告分类情况	用户进入应用列表后，可以选择任一应用点击查看该应用下测试报告当前的分类情况。

3.2.3 非功能性需求

表 3.3: 系统非功能需求列表

类别	需求描述
可用性	系统需要保障可用性。当系统主库出现问题时，能及时切换备份，在 5min 内恢复数据库访问。
可靠性	系统应该具备一般场景下的异常处理机。当出现可预见异常的情况时，妥善保管并备份已有数据，同时提供给用户直观易懂的提示信息。
性能	系统在正常网络场景下，应当确保所有简单查询请求的相应时间控制在 300ms 以内，复杂查询、大数据量请求和复杂写请求的性能可适当放宽至 500ms 以内。
可拓展性	系统应该采用插拔式低耦合的模块设计，保证后续功能可以以最小成本扩展。
安全性	系统应确保系统安全性，确保数据的查看权限和操作权限仅限管理员账号。
易用性	系统需要满足通用的人机交互规范，符合用户正常使用习惯，界面简洁易操作，并提供给用户直观友好的提示和引导。

非功能性需求，一般会称为系统的“品质”，是指软件产品为了满足用户业

务需求而必须具有的除功能需求以外的其他特性。系统需要具备可用性、可靠性。需要满足用户功能的基础上，保障用户的使用。可以通过多种数据备份手段，数据恢复机制等进行基本的使用保障。在此基础上，系统还需要具备高性能，尤其在复杂数据查询或复杂写请求的场景下，系统的快速响应也是用户体验的一大关键因素。为了满足后期可能的功能迭代，系统的每一个功能模块需要进行可插拔式设计，以保障后期功能的可拓展性。安全性也是必不可少的一大特性。系统必须确保数据访问和功能操作的安全性，保障数据安全和资源安全。另外，易用性也是我们需要考虑的因素。系统越容易上手，用户就能更短时间能开始使用，体验也就越好。上表3.3列举出了系统的非功能性需求。

3.2.4 系统用例图

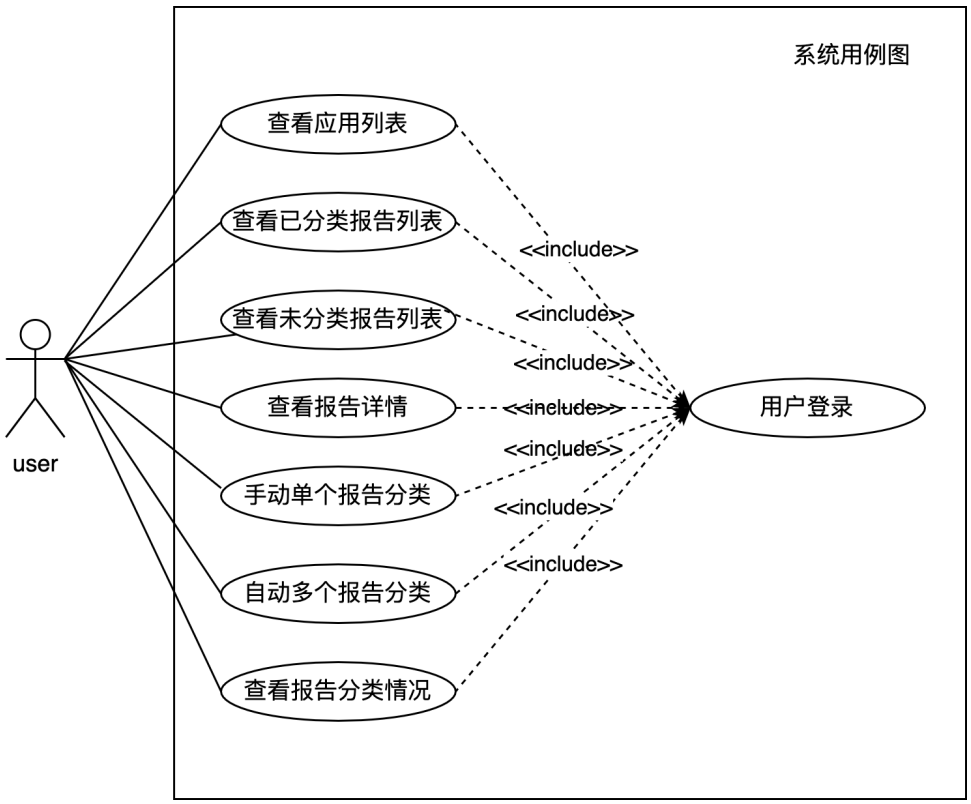

图 3.2: 系统用例图

本系统的主要用户为众包测试的审核员。通过分析审核员在缺陷分类系统中需要完成的操作，基本可以分为查看应用列表、查看已分类报告列表、查看未分类报告列表、查看报告详情、手动单个报告分类、自动多个报告分类以及查看报告分类情况。如图 3.2所示。

3.2.5 系统用例描述

用例描述是对用例的进一步说明，通过文本的方式将用例的参与者、触发条件、流程等信息描述出来。根据用例图可以整理出如下用例列表:

表 3.4: 系统用例列表

用例编号	用例名称	功能需求编号
UC1	查看应用列表	RQ1、RQ2
UC2	查看已分类报告列表	RQ1、RQ3
UC3	查看未分类报告列表	RQ1、RQ4
UC4	查看报告详情	RQ1、RQ5
UC5	手动单个报告分类	RQ1、RQ6
UC6	自动多个报告分类	RQ1、RQ7
UC7	查看报告分类情况	RQ1、RQ8

UC1 查看应用列表如表3.5所示，审核人员登录系统后，进入应用列表页面，包括应用名称、应用简介、应用类别、开始时间、结束时间、处理进度等信息。其中处理进度表示当前已完成分类的报告的占比。点击选择列表任一应用，查看该应用下的全部测试报告，处于“所有报告”状态。

表 3.5: 查看应用列表用例描述表

ID	UC1
用例描述	查看应用列表
参与者	审核人员
触发条件	审核人员登录后进入应用列表页面
前置条件	审核人员必须已经成功登录系统
后置条件	无
优先级	高
正常流程	1. 审核人员登录系统，进入应用列表页面 2. 显示应用信息，包含应用名称、应用简介、应用类别、开始时间、结束时间、处理进度等信息;

UC2 查看已分类报告列表如表3.6所示，审核人员登录系统后，进入应用列表页面，点击任一应用进入该应用下的所有测试报告。点击已分类，筛选出所有已分类的测试报告。

UC3 查看未分类报告列表如表3.7所示，审核人员登录系统后，进入应用列表页面，点击任一应用进入该应用下的所有测试报告。点击未分类，筛选出所有未分类的测试报告。

表 3.6: 查看已分类报告列表用例描述表

ID	UC2
用例描述	查看已分类报告列表
参与者	审核人员
触发条件	审核人员登录后进入应用列表页面并选择已分类
前置条件	审核人员必须已经成功登录系统
后置条件	无
优先级	高
正常流程	1. 审核人员登录系统，进入应用列表页面 2. 系统显示应用信息，包含应用名称、应用简介、应用类别、开始时间、结束时间、处理进度等; 3. 点击任一应用进入该应用下的所有报告页面 4. 点击已分类报告 5. 系统显示当前应用下的所有已分类测试报告

表 3.7: 查看未分类报告列表用例描述表

ID	UC3
用例描述	查看已分类报告列表
参与者	审核人员
触发条件	审核人员登录后进入应用列表页面并选择已分类
前置条件	审核人员必须已经成功登录系统
后置条件	无
优先级	高
正常流程	1. 审核人员登录系统，进入应用列表页面 2. 系统显示应用信息，包含应用名称、应用简介、应用类别、开始时间、结束时间、处理进度等; 3. 点击任一应用进入该应用下的所有报告页面 4. 点击未分类报告 5. 系统显示当前应用下的所有未分类测试报告

UC4 查看报告详情如表3.8所示，审核人员登录系统后，进入应用列表页面，点击任一应用进入该应用下的所有测试报告。通过所有测试报告列表（或未分类、已分类测试报告列表）点击进入某一测试报告详情页，系统展示测试报告详情信息。

UC5 手动单个报告分类如表3.9所示，审核人员登录系统后，进入应用列表页面，点击任一应用进入该应用下的所有测试报告。点击“未分类”筛选出当前未分类的所有测试报告。选择任一测试报告，在报告详情界面分类拦点击下拉菜单，选定合适的分类即可。

UC6 自动多个报告分类如表3.10所示，审核人员登录系统后，进入应用列表

表 3.8: 查看报告详情用例描述表

ID	UC4
用例描述	查看报告详情
参与者	审核人员
触发条件	审核人员登录后进入应用列表页面后选择某一个应用点击进入该应用下的报告列表
前置条件	审核人员必须已经成功登录系统
后置条件	无
优先级	高
正常流程	1. 审核人员登录系统，进入应用列表页面 2. 系统显示当前应用信息，包含应用名称、应用简介、应用类别、开始时间、结束时间、处理进度等; 3. 点击任一应用进入该应用下的所有报告列表 4. 点击任一报告进入报告详情界面（或通过当前应用下已分类或未分类报告列表进入某一报告详情界面） 5. 系统显示报告标题、创建时间、文本介绍、屏幕截图等信息

表 3.9: 手动单个报告分类用例描述表

ID	UC5
用例描述	手动单个报告分类
参与者	审核人员
触发条件	审核人员选择某一个应用点击进入该应用下的报告列表，并进入某一报告的详情界面
前置条件	审核人员必须已经成功登录系统
后置条件	所选报告进入已分类报告列表
优先级	高
正常流程	1. 审核人员登录系统，进入应用列表页面 2. 系统显示当前应用信息，包含应用名称、应用简介、应用类别、开始时间、结束时间、处理进度等; 3. 点击任一应用进入该应用下的报告列表并点击“未分类”筛选出所有未分类测试报告 4. 点击任一报告进入报告详情界面 5. 点击“手动分类”，选择适当的分类标签 6. 系统弹出确认框，询问是否确定将该报告分类为所选中的分类 7. 确定分类 8. 系统提示分类成功
特殊需求	1. 当上述流程 7 中选择取消分类时，不进行分类请求

页面，点击任一应用进入该应用下的所有测试报告。点击“未分类”删选出当前未分类的所有测试报告。选择任一测试报告，操作栏中点击下拉框，选定合适的分类。

表 3.10: 自动多个报告分类用例描述表

ID	UC6
用例描述	自动多个报告分类
参与者	审核人员
触发条件	审核人员选择某一个应用点击进入该应用下的报告列表，删选进入未分类报告列表
前置条件	审核人员必须已经成功登录系统
后置条件	所选报告进入已分类报告列表
优先级	高
正常流程	1. 审核人员登录系统，进入应用列表页面 2. 系统显示当前应用信息，包含应用名称、应用简介、应用类别、开始时间、结束时间、处理进度等; 3. 点击任一应用进入该应用下的报告列表并点击“未分类”筛选进入未分类报告列表 4. 点击“分类”按钮进入分类模式 5. 选择将要用于自动分类的测试报告 6. 点击“自动分类”按钮 7. 系统弹出确认框，询问是否确定将选中报告自动分类 8. 确定分类 9. 系统提示分类成功
特殊需求	1. 当上述流程 8 中选择取消分类时，不进行分类请求

UC7 查看报告分类情况如表3.11所示，审核人员登录系统后，进入应用列表页面，点击任一应用后，点击“分类统计”查看当前报告

表 3.11: 查看报告分类情况用例描述表

ID	UC7
用例描述	查看报告分类情况
参与者	审核人员
触发条件	审核人员选择某一个应用点击进入该应用下的报告列表，点击分类统计
前置条件	审核人员必须成功登录系统
后置条件	无
优先级	高
正常流程	1. 审核人员登录系统，进入应用列表页面 2. 系统显示当前应用信息，包含应用名称、应用简介、应用类别、开始时间、结束时间、处理进度等; 3. 点击任一应用进入该应用下的报告列表并点击“分类统计”查看当前状态下分类信息 4. 系统显示当前分类统计信息，包含各个类别当前报告数，报告所占比等

3.3 系统总体设计

3.3.1 系统整体架构设计

本系统为基于深度学习框架——深度玻尔兹曼机的众包测试报告分类系统，采用了前后端分离的开发架构。前端使用 React 开发 SPA 应用，系统后端采用 SpringBoot 开发，前后端通过 Restful Api 通信，获取到众包测试中应用的缺陷报告数据，如缺陷标题，缺陷描述等信息。系统中的分类功能包含手动单个分类和自动多个分类，手动单个分类时直接选择缺陷对应的类别即可，自动多个分类时需要用到基于 Python 的分类服务，该服务通过 Flask 提供的 Restful 和 SpringBoot 通信。具体设计如图3.3所示。图中服务均采用 Docker 打包部署到服务器。

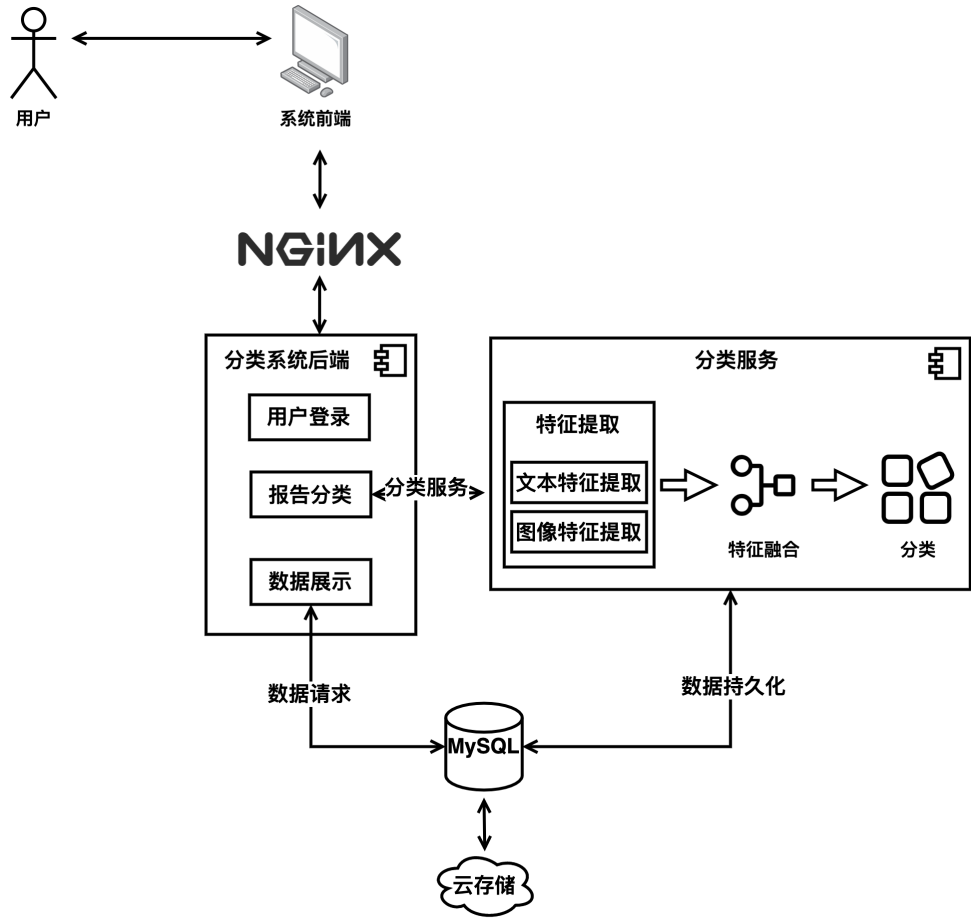

图 3.3: 报告分类系统架构图

1）使用 React 进行前端开发，采用了单页面的开发模式，采用单页面开发的好处是不会因为用户的操作而进行页面的重新加载，配合 JavaScript 动态的完

成和用户的交互，结合 React-Router 实现路由的跳转，保证了良好的交互体验。使用 Ant-Design 作为前端组件库，实现了前端页面风格统一化，同时简化了自建组件的开发过程。使用 Echarts 进行数据的统计展示，便捷开发的同时也保证了图标的美观性。使用 npm 进行前端包管理，很好的维护了不同包版本之间的依赖。使用 webpack 前端打包，便于前端打包部署。

2）使用 SpringBoot 进行后端开发，采用了 MVC 的开发模式。Controller 作为控制层，控制具体的业务模块的流程。Sercive 层作为业务层，服务各个业务模块的逻辑应用设计。DAO 层作为持久层，主要负责和数据库进行交互。在 SpringBoot 中，Controller 层调用 Service 层的方法，Service 层调用 DAO 层的方法。Controller 层处理前后端交互，接受前端请求，调用 Service 层的方法，接收 Service 层返回的数据，最后返回数据给前端。Service 包含了放接口的类和实现的类，并在配置文件中进行管理。Controller 层的调用到 Service 层后，Service 层调用 DAO 层的接口，接收 DAO 层返回的数据并返回给 Controller。DAO 层定义了实际会使用到的方法，比如数据的增删改查。可以在配置文件中配置 DAO 层的数据源和数据库连接的参数。

3）使用 Flask 进行分类服务的开发。Flask 作为 Python 最灵活的框架之一，可以很方便的扩展其他功能。Flask-Restful 是 Flask 的一个扩展，提供了 Rest API，实现了与系统后端的 api 交互。将基于 Python 的分类服务和基于 Java 的后端服务打通。

4）使用 MySQL 作为后端和分类服务的持久化数据库，存储了众包测试中不同应用下的缺陷报告数据。将原始的报告数据从云服务器下载到本地 Excel 中，在 Excel 进行数据清洗与标记后，持久化存储在 MySQL 中，再将后端服务与 MySQL 连接，进行数据操作。

3.3.2 系统模块划分设计

如图3.3所示，本系统主要由缺陷分类系统后端和分类服务构成。其中，缺陷分类基于 Java 语言开发，用于和前端交互，负责处理前端请求，并返回数据给前端。主要包括用户登录，报告分类和数据展示三个功能模块。用户需要登录系统确保被授权系统的使用和数据的操作权限，主要通过前后端增加 token 字段在后端校验前端是否有操作权限，或者前端的权限是否过期。对于众包测试工人提交的测试报告，审核人员具有分类报告的权限，分类报告可以手动对单个报告进行分类，也可以选择在列表中一次选择多个报告统一一次性进行分类，这种多选分类依赖分类服务。数据展示功能中，报告可以按照是否分类进行展示，也可以通过分类统计页面统一查看报告分类情况，包含已分类和未分类报

告的数量，比例，以及各类别缺陷报告的数量和比例。

分类服务包括数据预处理，DBM 特征融合，报告分类三个部分。首先会对所选报告中的文本数据和图像数据进行预处理，并对提取后的文本和图像特征进行归一化；接着通过 DBM 对归一化后的文本和图像特征向量进行特征融合，得到高维合成向量；最后用 SVM 对合成后的高维合成向量分类。

3.3.3 “4+1”视图

“4+1”视图是 1995 年由 Philippe Kruchten 教授首次在《IEEE Software》提出的概念，是一种使用多个视图同时描述软件架构的模型方法，以不同参与者的角度分别从某一特定视角给出了对系统架构的特有描述，从而更好的得到了不同角度的系统架构刻画。“4+1”视图从五个不同的视角描述了软件的体系结构，分别为场景视图、逻辑视图、开发视图、进程视图和物理视图。每一个试图只从一个角度关注系统，五个视图的结合才可以完整的反映出系统完整的软件体系结构。

（1）场景视图

场景视图连接了其他四个视图，是重要系统活动的抽象，或者说是重要需求的抽象，描述了现实生活中系统运用的场景，包括牵扯到的参与者及其功能。这里对应了 UML 中的用例图，如上文图3.2所示。

（2）逻辑视图

逻辑视图从最终用户的角度描述了系统架构，主要描述了系统的功能模块。在逻辑视图中，通过逻辑分层，系统被分解为功能的抽象，功能模块被封装成类。可以通过 UML 中的类图进行描述。

如图3.4描述了系统的逻辑视图。BugReportManagement 模块是基础的缺陷报告管理模块，包含了 BugUploaderService 模块，BugReportDisplayService 模块和 BugReportClassifyService 模块。其中，BugUploaderService 模块负责缺陷报告的数据获取工作，主要从之前的众包测试报告中获取需要的缺陷报告数据并存储在 MySQL 服务器上。BugReportDisplayService 模块负责系统的相关展示工作，如应用列表展示，应用中的缺陷报告展示等。BugReportClassifyService 模块负责缺陷报告分类功能，这一功能依赖另一 BugClassification 模块。BugClassification 模块是专门处理缺陷报告分类的功能模块，包含 ModelTraningService 模块和 BugClassifyService 模块。其中，ModelTraningService 模块用于模型训练，这里主要训练了用于特征融合的深度学习模型和用于分类的分类器模型。BugClassifyService 模块作为处理缺陷报告分类的功能模块。另外还有 BugFeature 模块专门用于模型训练，包括了 DataPreprocessing 模块用于数据预处理，FeatureExtracion

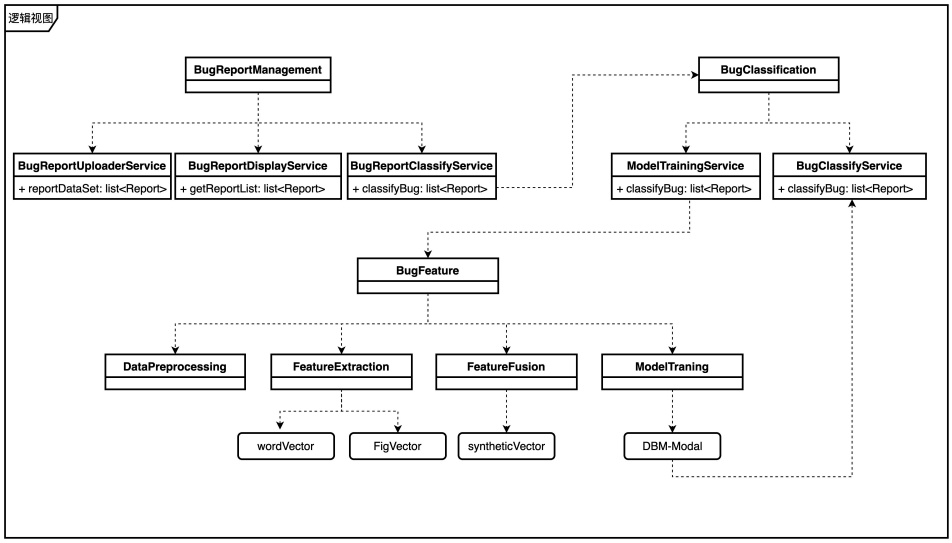

图 3.4: 逻辑视图

模块用于特征提取，FeatureFusion 模块用于特征融合，以及最后的 ModelTraning 用于模型训练，最终训练的模型服务于分类服务。

（3）开发视图

开发视图从软件开发者和项目经理的角度描述了软件实际开发环境下模块组织结构。在开发中视图中，关注点更多在程序包，如开发者自己编写的源程序包以及第三方的 SDK、现成框架等，关注软件模块的组织和管理，在 UML 中通过组件图、包图来表示。开发视图和逻辑视图之间一般可能会存在一定的映射关系，比如逻辑层一般会映射到多个程序包等。

如图3.5描述了系统的开发视图。前端部分由 React 完成，配合 React-Router 采用了 SPA 开发模式，使用 Echarts 进行数据图表展示，npm 作为包管理工具，Webpack 作为打包构建工具，Ant-Design 作为组件样式库，JavaScript 作为开发语言，css 编写样式，less 进行 css 预编译。

后端部分由 SpringBoot 完成，配合 Maven 完成项目管理和构建工作。Controller 层包含 BugControllery 用于处理 Bug 相关的请求，AppController 处理 App 相关的相求。Service 层包含 BugService 处理 Bug 相关的数据，AppService 处理 App 相关数据。Dao 层包含 BugDao 和 AppDao 用于和 MySQL 交互。

同时，缺陷报告分类的请求还依赖于独立的 Classify Service。该服务由 Flask 开发完成。包含了数据预处理，特征的提取和融合，分类三个功能模块。其中，LTP 用于文本数据预处理，TF-IDF 用于文本特征提取，CV2 用于图像尺寸调整，SPM 用于图像特征提取，DBM 用于特征融合，SVM 用于特征分类。该模块用

于深度学习模型和分类模型的训练以及缺陷报告分类功能。

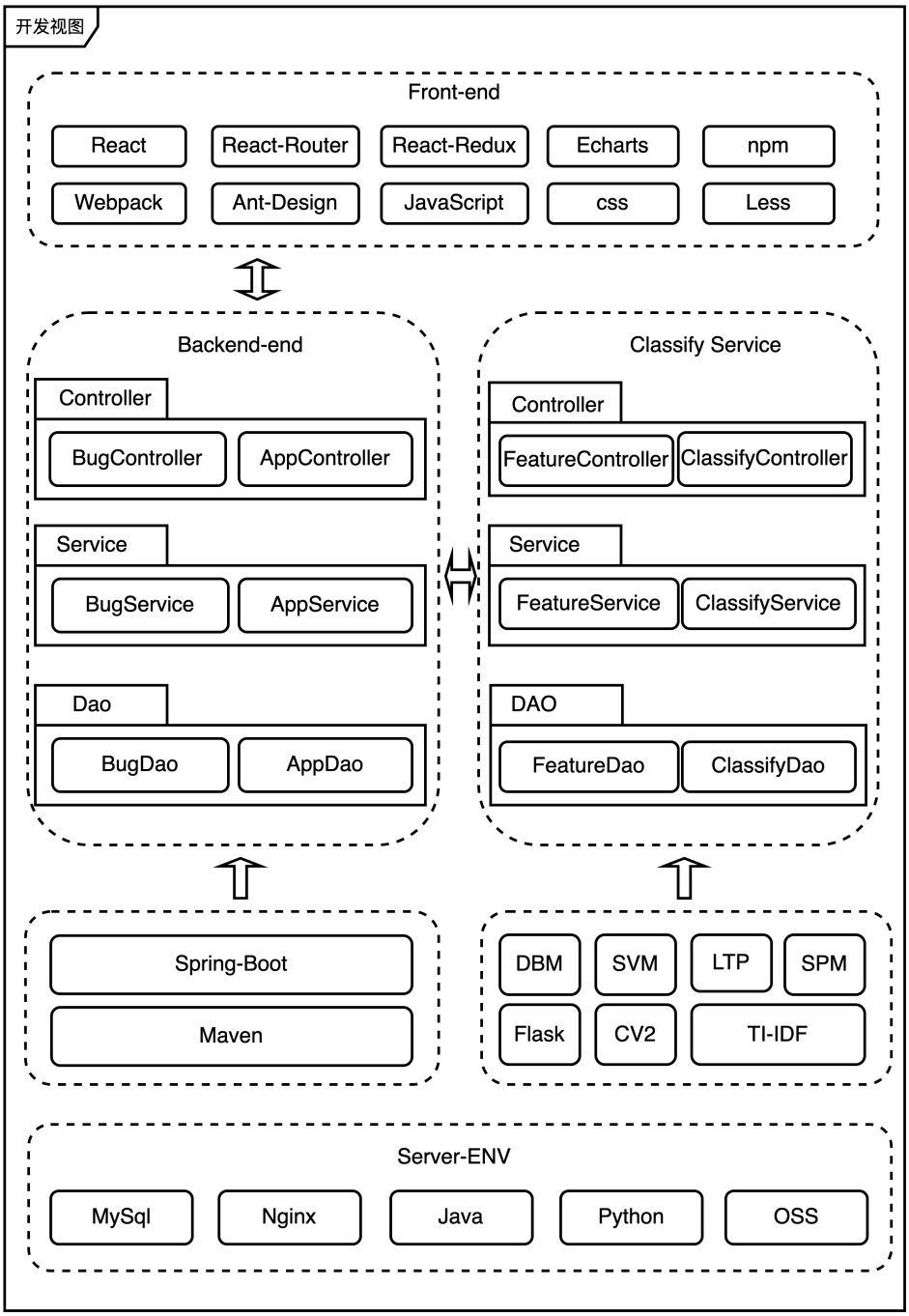

图 3.5: 开发视图

（4）进程视图

进程视图从系统集成者和系统设计人员的角度描述了系统并发和信息同步的过程，主要关注与系统中主要的进程和线程之间的通信过程，关注系统的运

行特性。进程视图更强调系统的并发性、分布性、容错性和昔日集成性，定义了逻辑视图中各个类的操作具体在哪个线程中被执行的，是对逻辑视图中细节的详细描述。

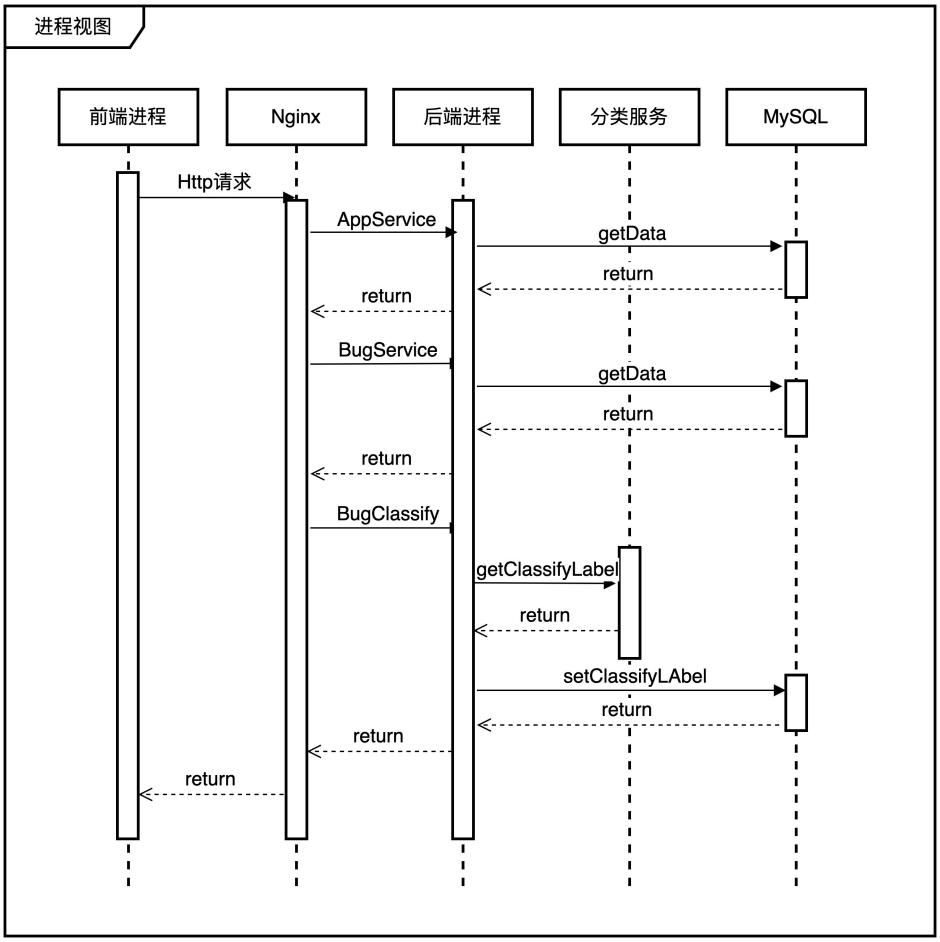

图 3.6: 进程视图

如3.5所示进程视图。由用户在前端界面发起请求，开启进程。Nginx 代理会根据服务端配置开始解析请求，之后根据负载均衡策略将请求分发到对应的服务端进行解析。后端进程采用了分层模式，Controller，Service，Dao 层相互配合，处理请求，返回数据。如果是缺陷报告分类的服务，后端会将请求转发到特定的分类服务上，由分类服务所在的服务端解析请求并返回。对于数据的增删查改，服务端会与 MySQL 数据库进行交互，进行数据持久化。最终，请求结果返回给客户端。

（5）物理视图

物理视图从系统工程师和运维人员的角度描述了系统的部署结构和网络通

信策略。物理视图主要关注如何把软件映射到硬件上，通常需要考虑到系统的性能、规模、可靠性等等，需要解决系统的物理拓扑、系统安装和网络通信等问题，可以用 UML 中的部署图描述。物理视图就是对物理资源的分配进行的描述和展示。

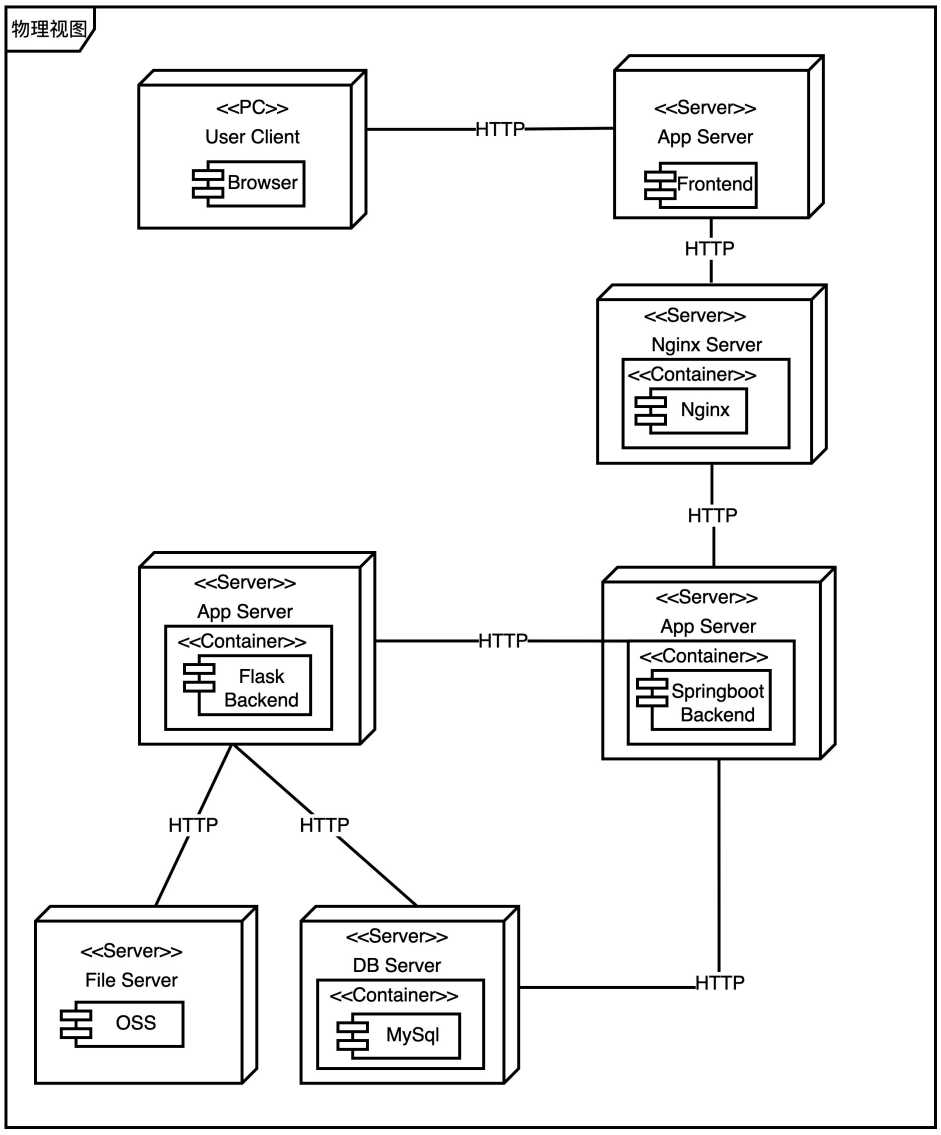

图 3.7: 物理视图

如3.7所示物理视图。用户一般通过自己本地计算机上的浏览器访问系统的前端页面，登录系统并成功完成授权认证后，进行系统操作。系统的前端作打包成静态文件部署在服务器上。用户通过浏览器向服务端发送 Http 请求，请求经过 Nginx 代理，根据既定好的负载均衡策略分发到特定的服务器上。同时，Nginx

也避免了前端浏览器的跨域问题。基于服务器成本和运维成本等因素，本系统相关的前端，后端，分类服务，Nginx 环境，MySQL 环境均以 Docker 容器的形式部署在同一台服务器上，并以不同端口向外提供服务。服务端之间通过 Restful API 形式通信。同时，数据不仅存储在 MySQL 中，还会在 OSS 上持久化存储。

3.4 系统数据模型设计

3.4.1 实体类设计

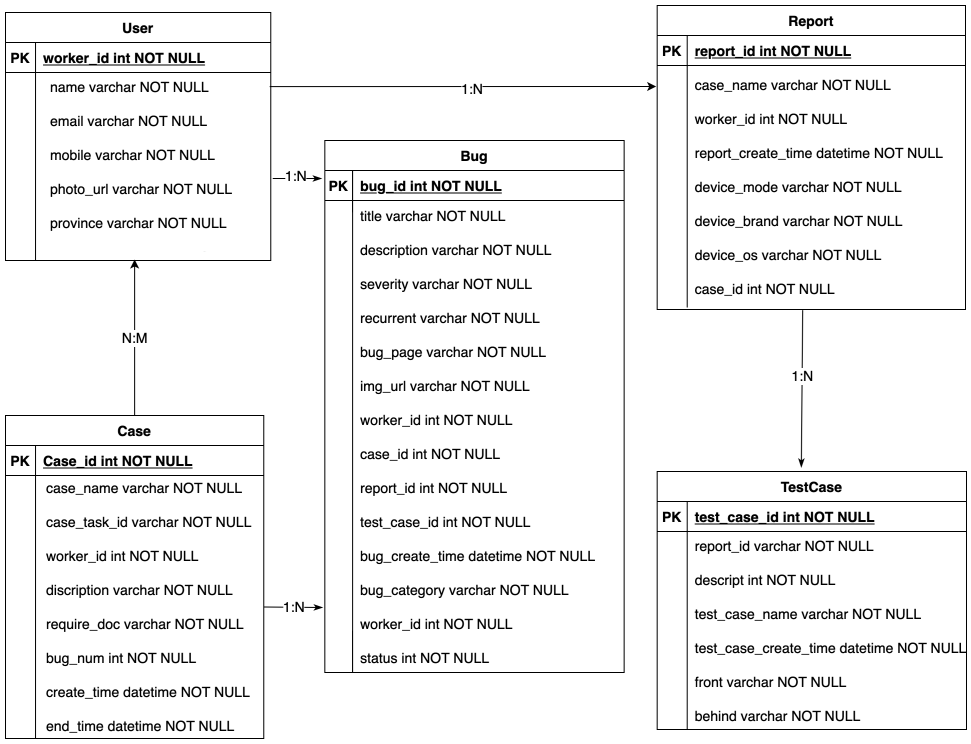

图 3.8: 实体类设计图

如图3.8描述了系统中存在的各个实体类之间的关系。重要的实体类包括本系统的用户即审核员 Worker，测试目标类也叫做待测 App 类 Case，测试报告类 Report，测试用例类 TestCase，以及缺陷报告类 Bug。一个审核人员可以选择参与 0 到 N 场众包测试任务，一场众包测试任务可以有 0 到 N 个审核工人。审核工人可以在一场众测中审核 0 到 N 个测试报告。每一份测试报告中包含了 1 到 N 个测试用例，一个测试用例中可以提出 0 到 N 个测试缺陷，对应 0 到 N 份缺陷报告。

Worker 类：Worker 类主要存储系统主要用户的相关信息。众包测试工人提交测试报告后，审核员有权限对测试报告进行分类和审核操作，该系统中主要针对分类操作展开。Worker 类的具体内容如表3.12所示。

表 3.12: Worker 类详情列表

字段	类型	含义	备注
name	varchar	昵称	当前用户昵称
email	varchar	邮箱	当前用户邮箱
mobile	varchar	手机号	当前用户手机号码
photo_url	varchar	用户头像	当前用户头像
province	varchar	省份	当前用户所在省份
create_time	datetime	创建时间	当前账户创建时间

Case 类：Case 类主要存储待测应用的相关信息。任务发布方会在众包测试平台上上传测试应用，测试需求，奖励机制，并等待任务接受方领取。Case 类的具体内容如表3.13所示。

表 3.13: Case 类详情列表

字段	类型	含义	备注
case_name	varchar	report 名称	当前应用的名称
worker_id	int	众测工人 id	确定了创建当前众测报告的众测工人
description	varchar	应用描述	对当前应用的功能等进行的简要说明
require_doc	varchar	需求文档	当前应用的需求说明文档
bug_num	int	缺陷数量	当前应用下测出的缺陷数量
create_time	datetime	创建时间	对当前应用进行众包测试的开始时间
end_time	datetime	结束时间	对当前应用进行众包测试的结束时间
no_type_bug_num	int	未分类缺陷数	用于衡量当前应用的分类进度

表 3.14: Report 类详情列表

字段	类型	含义	备注
report_name	varchar	report 名称	当前报告的名称
report_create_time	datetime	创建时间	当前测试报告的创建时间
worker_id	int	众测工人 id	确定了创建当前众测报告的众测工人
device_mode	varchar	设备型号	当前报告所用的测试设备型号
device_brand	varchar	设备品牌	当前报告所用的测试设备品牌
device_os	varchar	操作系统号	当前报告所用的测试设备操作系统
case_id	int	相关联的 case	当前报告所属的相关应用

Report 类:Report 类主要存储众包测试报告的相关信息。测试报告是在众包测试中测试用例和测试报告的载体。测试报告中主要记录了测试工人执行测试任务时用到的设备信息，便于后期缺陷的复现以及后续缺陷修复工作的开展。Report 类的具体内容如表3.14所示。

TestCase 类:TestCase 类主要存储了测试用例的相关信息。测试用例在测试报告中记录，每个测试用例可能会产生多个缺陷信息。TestCase 类的具体内容如表3.15所示。

表 3.15: TestCase 类详情列表

字段	类型	含义	备注
report_id	int	testCase 关联的 repord_id	唯一确定了相关联的报告
create_time	datetime	创建时间	当前测试用例的创建时间
name	varchar	测试用例 name	当前测试用例的名称
description	varchar	测试用例描述 name	当前测试用例的描述
front	varchar	测试用例前置条件 name	当前测试用例的前置条件
behind	varchar	测试用例后置条件 name	当前测试用例的后置条件

Bug 类:Bug 类主要存储了缺陷报告的相关信息。在众包测试的过程中，测试工人创建测试报告，选择具体的测试用例后创建并提交缺陷信息。缺陷信息主要包括描述缺陷的页面，严重等级等。Bug 类中的信息是本系统分类的主要目标。Bug 类的具体内容如表3.16所示。

表 3.16: Bug 类详情列表

字段	类型	含义	备注
title	varchar	bug 的标题	缺陷标题
description	varchar	bug 的描述	缺陷描述
severity	varchar	bug 的严重等级	缺陷的严重等级
recurrent	varchar	bug 的可复现程度	缺陷的严重等级
bug_page	varchar	bug 存在的页面	当前缺陷所在的 App 页面
img_url	varchar	bug 截图地址面	缺陷截图所在的 url 连接
case_id	int	对应的 case	缺陷相关联的应用 App
test_case_id	int	对应的测试用例	缺陷相关联的测试用例
report_id	int	对应的 report	缺陷相关联的测试报告
bug_category	varchar	bug 的类别	缺陷对应的分类类别
bug_create_time	datetime	bug 的创建时间	缺陷的创建时间
worker_id	int	众测工人 id	提出当前缺陷的众测工人
status	int	分类状态	当前报告的状态，是否分类，是否审核

3.4.2 数据库设计

ER 图 (Entity Relationship Diagram) 也可以称为实体-联系图，由实体、属性、关系三个核心部分组成。实体由长方形表示，属性由椭圆形表示，而关系用菱形表示。实体指的是数据模型中的数据对象，属性是的是实体中的特征，一般来讲实体至少都会有一个属性，关系描述了实体之间的联系。ER 图是现实世界的一种抽象概念，是表述概念关系模型的一种方式，属于需求数据库设计的需

求分析阶段。针对具体的问题。通过分析现实关系，确定实体，属性，制作 ER 图，辅助进行数据库建设。

通过分析系统功能，得出系统主要有审核工人 Worker，待测应用 Case，测试报告 Report，测试用例 TestCase 和缺陷 Bug 五个实体。在每个应用 Case 下，可以有多个审核工人 Worker，Worker 审核报告 Report，每个 Report 下包含一个或多个测试用例 TestCase，每个测试用例又由一个或多个缺陷 Bug 组成。下图3.9描述了本系统的各个实体之间的关系。

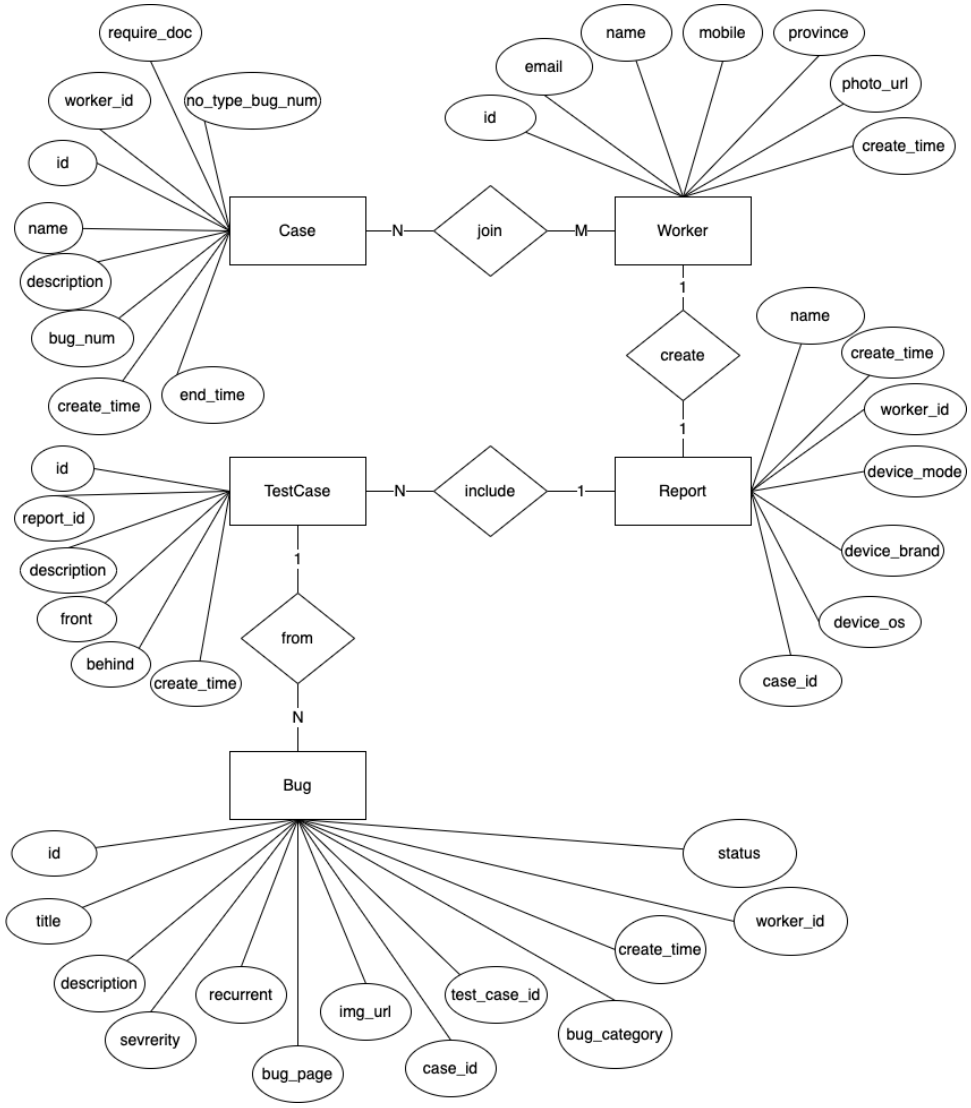

图 3.9: E-R 图

3.5 系统模块流程设计

3.5.1 缺陷报告管理模块流程设计

缺陷报告管理模块是整个系统的基础模块，承载了整个系统的全部功能。缺陷报告管理模块也是服务端的 Java 主程序模块，负责系统不同服务之间的互相调用。用户需要从登录系统开始获取系统的操作授权。完成登录授权后，可以进行数据查看和缺陷报告分类的两个主要操作。包括查看测试目标 App，测试报告，缺陷分类情况等信息。报告分类后，对分类的结果进行持久化存储。如图3.10描述了缺陷报告管理模块的流程。

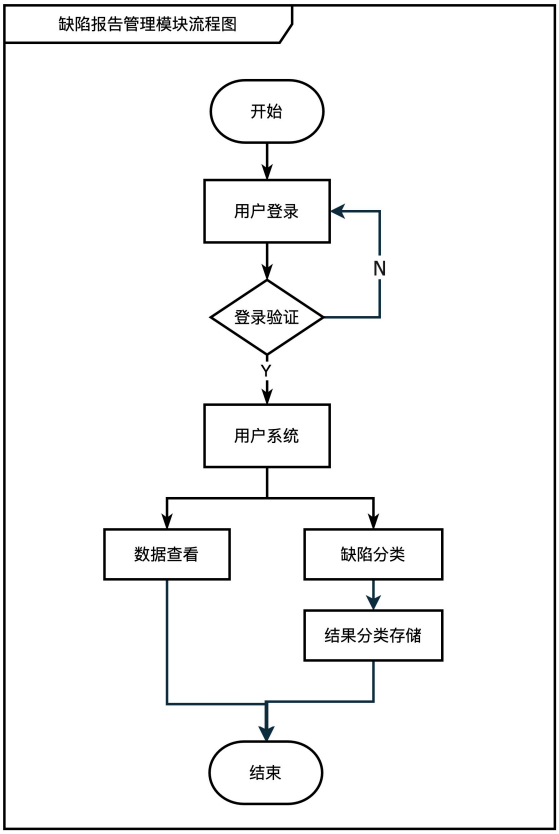

图 3.10: 缺陷报告管理模块流程图

3.5.2 缺陷报告特征提取模块流程设计

缺陷报告特征提取模块是数据预处理模块，也是系统融合模块前置依赖的一个重要模块。该模块在进行特征融合之前进行了重要的特征提取工作，主要对原始缺陷报告中的文本和图像信息进行预处理，并输出归一化后的特征向量。

首先对缺陷报告中的文本信息和图像信息进行提取，其中图像信息需要从对应的 OSS 云端提前下载。对于文本数据，主要提取了测试报告中的缺陷标题和缺陷描述,并进行文本拼接，然后通过 LTP 去除无意义的停用词，进行同义词转换，以及文本分词，再通过 TF-IDF 进行特征提取，最后进行归一化。对于图像数据，需要提前借助 CV2 库完成尺寸剪裁和灰度化处理，之后通过 SPM 提取出图像特征向量，并进行归一化。最后保存处理过的文本和图像向量，为融合模块做好准备工作。

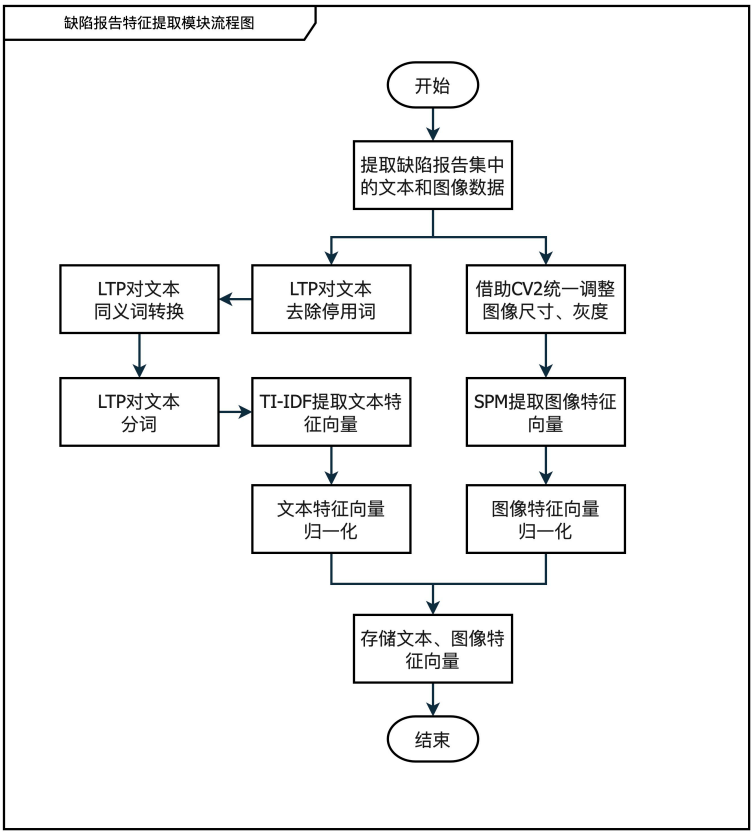

图 3.11: 缺陷报告特征提取模块流程图

3.5.3 缺陷报告特征融合模块流程设计

缺陷报告特征融合模块是系统依赖的一个关键服务模块，该服务模块在 Flask 分类服务中完成，被缺陷报告管理模块调用。主要作用是通过生成用于融合的 DBM 模型，将缺陷报告中的文字和图像信息进行特征提取，特征融合后，以高维合成向量的形式输出。在进入特征融合模块之前，首先根据特征提取模块中的步骤，对缺陷报告中的文本和图像数据进行预处理，得到文本和图像特征向量。

对于系统中的每一个待测应用，一共需要预训练三个 DBM 模型，分别是用于提取文本高维向量的 DBM 模型，用于提取图像高维向量的 DBM 模型，以及用于生成融合向量的 DBM 模型。这三个 DBM 模型都包含一个可视层和两个隐藏层，每个 DBM 都需要经过对比散度算法得到参数的最优解。如图3.12描述了缺陷报告特征融合模的主要流程。

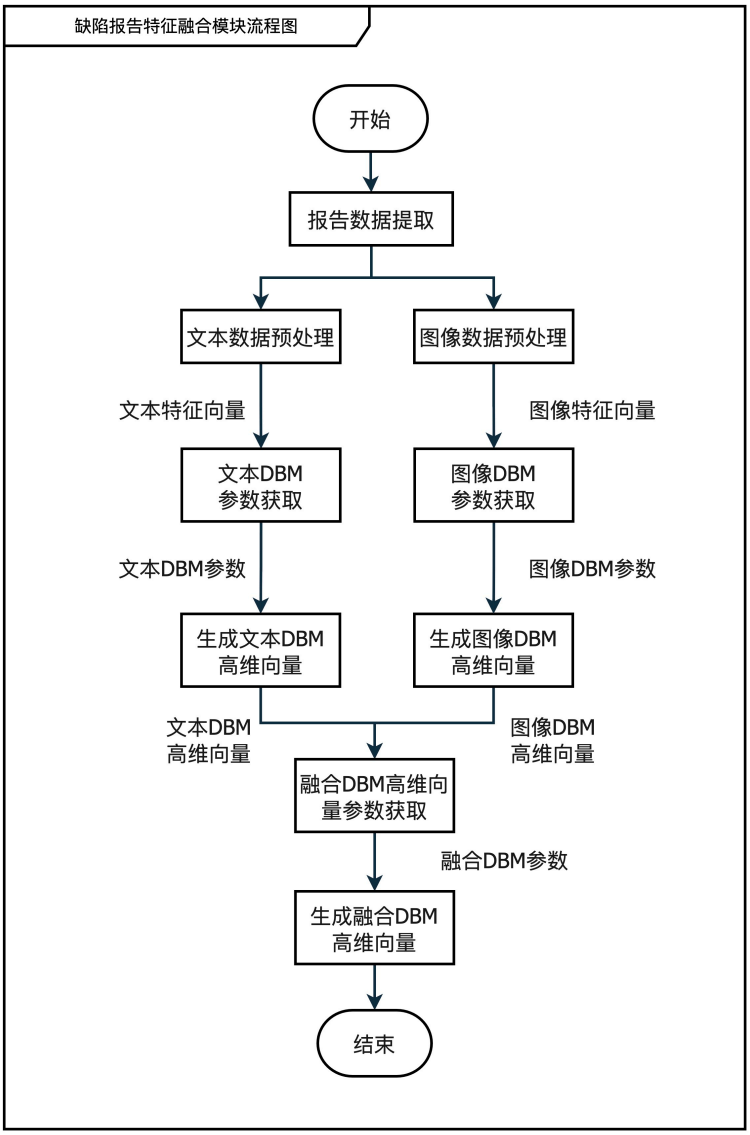

图 3.12: 缺陷报告特征融合模块流程图

图3.13描述了文本 DBM 向量获取的流程。文本向量输入预定义好的 DBM 模型中根据对比散度算法训练得到文本的 DBM 参数。之后将文本向量输入参数化后的 DBM 模型中得到文本高维向量。

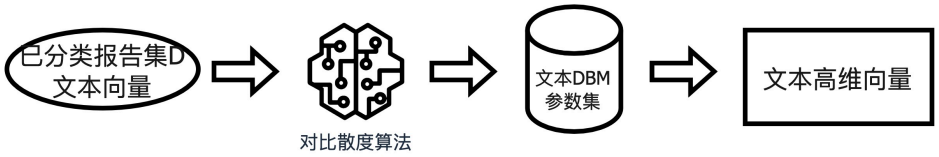

图 3.13: 文本 DBM 向量获取图

图3.14描述了图像 DBM 向量获取的流程。图像向量输入预定义好的 DBM 模型中根据对比散度算法训练得到图像的 DBM 参数。之后将图像向量输入参数化后的 DBM 模型中得到高维向量。

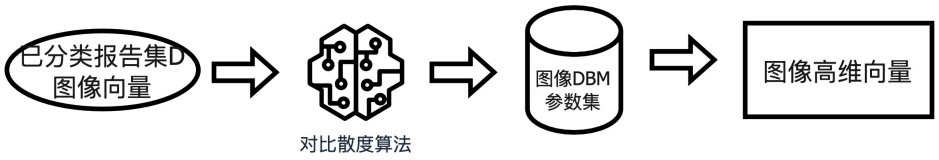

图 3.14: 图像 DBM 向量获取图

图3.15描述了缺陷报告特征融合模块的流程。将高维文本和图像向量进行拼接后根据对比散度算法训练得到融合的 DBM 参数，从而得到融合后的高维融合向量。这个融合向量就是对缺陷报告的特征表示。

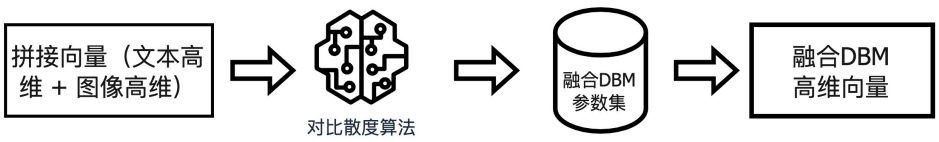

图 3.15: 融合 DBM 向量获取图

3.5.4 缺陷报告分类模块流程设计

缺陷报告分类模块是系统重要的核心模块。为了实现分类功能，首先需要进行分类模型的学习。将历史已有标签的数据集分为训练集和测试集，并搭建出三层 DBM 模型，利用 DBM 模型对训练数据集求出融合向量并根据训练集生成分类模型。后续可以经过测试验收准确率并优化模型。

得到 DBM 模型和 SVM 模型参数后存储起来，便于后续对缺陷报告进行分类。当待分类的报告进入分类流程后，根据 DBM 模型得到融合向量，再将融合向量输入 SVM 模型得到分类结果。

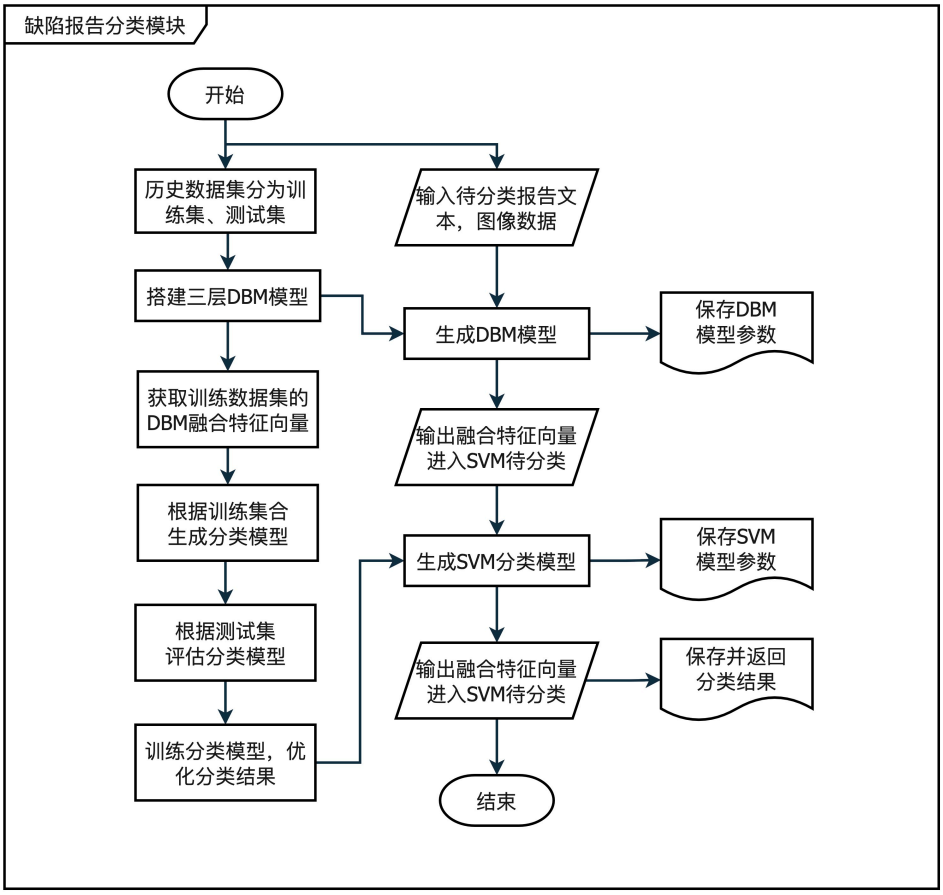

图 3.16: 缺陷报告分类模块流程图

3.6 本章小节

本章针对基于深度玻尔兹曼机的众包测试缺陷报告系统进行了需求分析和系统设计。首先通过涉众分析明确系统的功能性需求和非功能性需求，并结合用例图和用例描述详细说明。接着，借助“4+1”视图从不同角度对系统的架构和实现进行分析和设计。然后使用类图，E-R 图设计系统数据类。接着，通过流程图对系统不同模块的工作流程进行描述，为第四章详细设计做好准备。

第四章　众包缺陷分类系统的详细设计与实现

4.1　缺陷报告管理模块设计与实现

缺陷报告管理模块作为整个缺陷分类系统的基础模块，负责调度不同的功能服务，协助各个服务共同完成整个系统功能，主要包括测试应用列表查看，缺陷报告列表查看，缺陷报告分类，缺陷分类结果查看等。其中，对报告的分类需要进行持久化存储。用户进行系统操作和数据查看的前提是经过权限验证，所以用户必须先登录系统，完成操作授权。

4.1.1　交互顺序设计

缺陷报告管理模块主要处理的功能逻辑可以分为数据查看和报告分类。如图 4.1描述了缺陷报告管理模块数据查看功能的交互流程。用户需要先从前端登录系统，经过后端权限验证，在获取授权后取得系统的数据查看和操作权限。用户进入首页后前端对应用列表数据发起 http 请求，请求通过 Nginx 代理到相应 java 后端的报告管理服务开始响应用户请求，后端 SpringBoot 会根据三层 MVC 层层调用，进行请求响应。在获取应用列表请求中，根据请求路径调用 AppController 的 getAppList 方法，AppController 会调用 AppService 的 getAppList 方法，AppService 调用 AppDao 层的 findAll 方法，Dao 层的会直接对数据库 MySQL 进行数据请求，完成数据查询，最终层层返回到前端进行应用列表展示。展示内容包含应用名称，应用描述，应用类别，应用需求文档链接，应用缺陷报告分类进度情况等。

从应用列表点击任一应用后进入到应用的缺陷报告页面，默认停留在所有报告页面，用户可以通过选择已分类或者未分类报告进行报告筛选，查看指定状态下的报告，展示内容包括缺陷标题，缺陷描述，缺陷可复现程度，缺陷严重程度，缺陷提出人等等。在默认状态下，前端发起 getBugList 请求，由 ngxin 代理到后端 BugController 处理，BugController 调用 BugDao 的 findAll 方法请求数据库中所有的缺陷报告数据，并返回给前端进行展示。

当选择指定报告状态，或根据指定字段搜索含特定报告标题的缺陷报告时，指定的条件会被包裹成一个 condition 对象，这个 condition 对象会随着 post 请求在请求体中一起传递到后端。后端解析请求条件并处理后，BugService 带着参数调用 BugDao 层对应方法请求数据。这里，后端会根据请求条件选择具体调用 Dao 层的哪个方法，如根据标题搜索时会调用行 findByTitle 方法，根据分类状

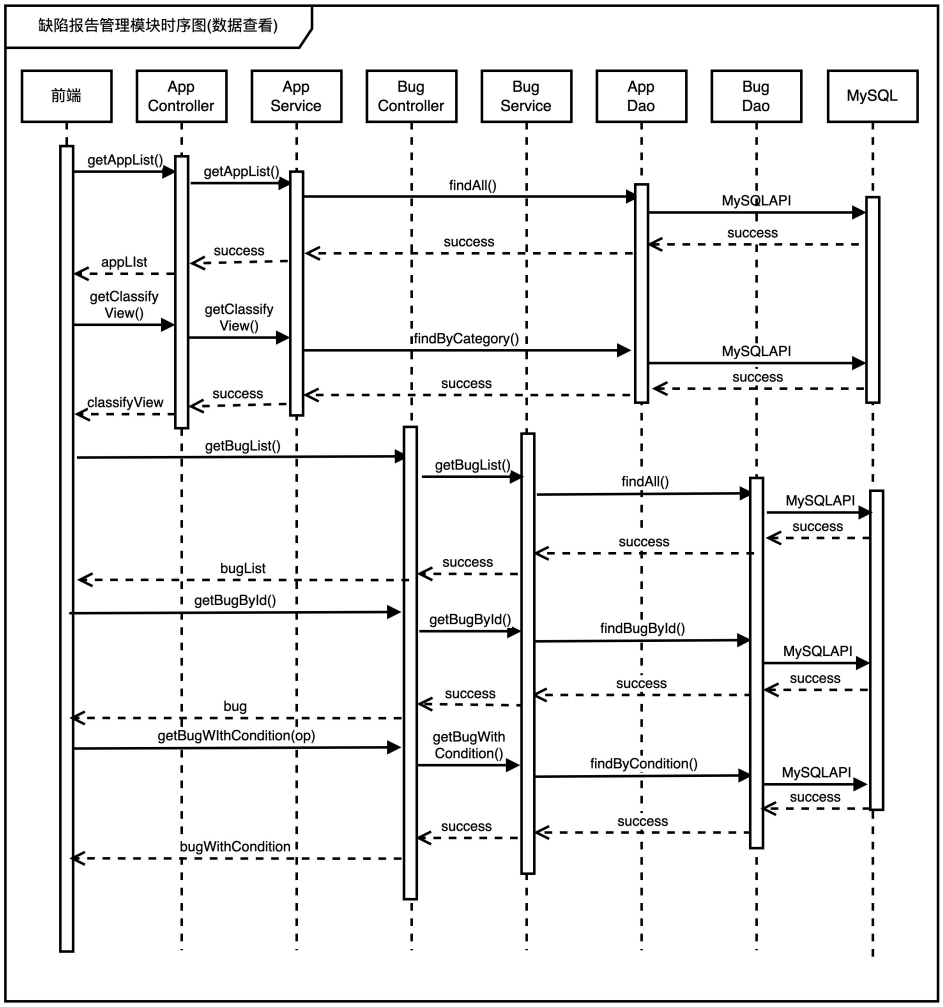

图 4.1: 缺陷报告管理模块时序图 (数据查看)

态搜索时会调用 findByStatus 方法。最终，后端将拿到的请求数据逐层返回给前端进行展示。

在报告列表页面点击任一报告，即可查看缺陷报告详情。前端会向后端发 get 请求获取 getDetailById 以获取对应缺陷报告的完整信息，后端接受到请求后 BugController 层调用 BugDao 层的 findBugById 方法，在数据库中获取到指定 Bug 报告完整信息并返回给前端。

从应用列表页面点击进入到应用的缺陷报告页面后，还可以从左侧选择查看当前应用下报告的分类情况, 如不同类别的缺陷报告分布图，当前应用的分类进度等。进入统计页面后，前端发起 getClassifyView 请求，后端 AppController 接收请求后调用 AppDao 层的 findBycategory 方法从数据库获取到当前应用的缺陷报告统计数据。

如图 4.2描述了缺陷报告管理模块报告分类功能的交互流程。当用户进入系统后，从应用列表页面点击任一应用进入该应用对应的缺陷报告列表页后，可以从页面左侧筛选进入未分类的报告列表页。用户可以选择逐个对报告分类或者自动对多个报告进行分类。

当单个报告手动分类时，用户在进入报告详情界面后，在分类栏中选择自定义类别，会触发更新报告的请求，后端此时由 BugController 接收到分类请求，调用 BugService 层的 UpdateBug 方法，再由 BugService 层调用 BugDao 的 UpdateBug 方法，最终更新数据库，该报告分类成功。

当多个报告自动分类时，用户在未分类报告列表界面选择多个报告，点击分类按钮，触发自动分类请求，后端此时由 BugController 层调用 BugService 层的 predict 方法，BugService 层会调用 ClassifyService 服务的 predict 方法，最终返回分类标签。接着 BugService 将带有标签的报告数据进行组织，调用 BugDao，完成多个报告分类功能。

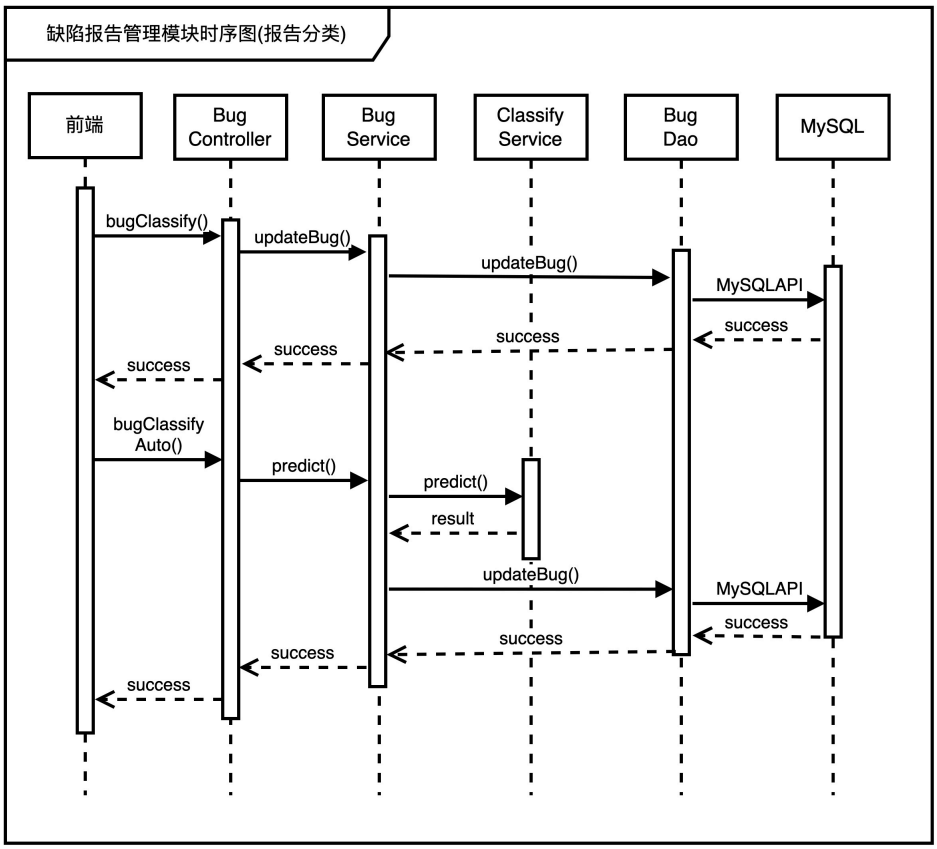

图 4.2: 缺陷报告管理模块时序图 (报告分类)

4.1.2 数据与类设计

缺陷报告管理模块的核心类图如 4.3所示。缺陷报告管理模块中组织协调所有系统的功能实现。BugService 调用 BugDao 层，BugDao 对接数据库，操作 Bug 类，完成 Bug 数据的有关增删改查操作，如获取所有缺陷报告 (findAll)，获取单个缺陷报告 (findBugById)，以及根据状态获取缺陷报告列表 (findAllByStatus)。最终，BugService 向外暴露出获取缺陷报告列表 (getBugList)，获取缺陷报告详情 (getBug)，更新缺陷报告 (updateBug)，根据状态获取缺陷报告列表 (getBugListByStatus)，以及缺陷报告分类 (predict) 接口。其中，分类接口依赖分类服务，分类服务中需要提前训练用于分类的深度学习模型。本系统中需要提前根据历史的测试报告数据训练出用于特征融合的 DBM 模型和用于分类的 SVM 模型，因此分类服务中会包含通过历史报告模型的 trainDBM 和 trainSVM 方法，以及用于自动化分类的 classifyBug 方法。

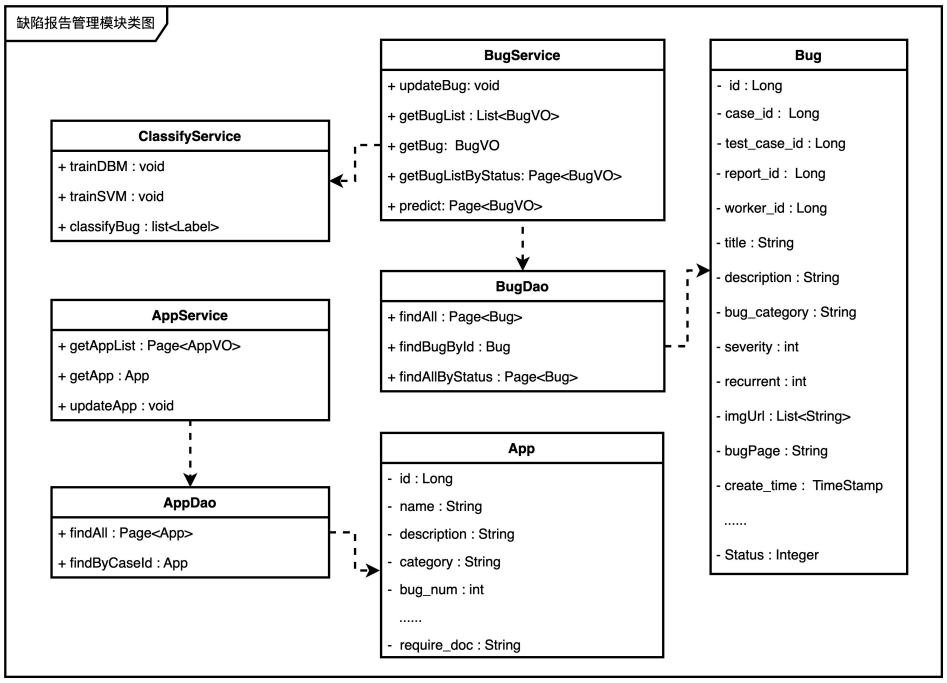

图 4.3: 缺陷报告管理模块类图

同理，AppService 调用 AppDao 层，AppDao 对接数据库，操作 App 类，完成 App 相关的数据增删改查，如获取所有测试应用列表 (findAll)，获取单个用用详情 (findByCaseId)。最终，AppService 向外暴露出获取应用列表 (getAppList)，获取单个应用详情 (getApp)，更新应用信息 (updateApp) 接口。

4.1.3 关键代码

图 4.4展示了缺陷报告管理模块获取 Python 端分类服务的部分代码。

```
private ClassificationResponseDTO postClassifyService(ClassificationRequestDTO classificationRequestDTO){
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<ClassificationRequestDTO> httpEntity = new HttpEntity<>(classificationRequestDTO, headers);
        ResponseEntity<String> responseEntity = null;
        try {
            log.info("缺陷报告分类服务的 url 地址为：" + classifyServiceURL);
            responseEntity = restTemplate.postForEntity(classifyServiceURL + "get_category", httpEntity,
String.class);
        } catch(RestClientException e){
            log.info("上游缺陷报告分类系统 在 restTemplate 的过程中出错");
            return null;
        }
        if(responseEntity.getBody() == null) {
            log.info("上游缺陷报告分类系统 返回值为空");
            return null;
        }

        ClassificationResponseDTO classificationResponseDTO = new
Gson().fromJson(responseEntity.getBody(), new TypeToken<ClassificationResponseDTO>(){}.getType());
        log.info("上游缺陷报告分类系统返回值为: {}", classificationResponseDTO);
        return classificationResponseDTO;
    }

    @Override
    public Boolean predict(String classificationMethod, List<Integer> ids) {
        ClassificationRequestDTO classificationRequestDTO = new ClassificationRequestDTO();
        classificationRequestDTO.setClassificationMethod(classificationMethod);
        List<BugDTO> bugDTOList = new ArrayList<>();
        for(Integer id: ids) {
            Bug bug = bugRepository.findBugById(id);
            BugDTO bugDTO = new BugDTO();
            bugDTO.setId(id);
            bugDTO.setTitle(bug.getTitle());
            bugDTO.setDescription(bug.getDescription());
            bugDTOList.add(bugDTO);
        }
        ClassificationResponseDTO classificationResponseDTO =
postClassifyService(classificationRequestDTO);
        List<BugDTO> bugDTOList1 = classificationResponseDTO.getBugs();
        for(BugDTO bugDTO: bugDTOList1){
            Bug bug = bugRepository.findBugById(bugDTO.getId());
            bug.setBugCategory(bugDTO.getCategory());
            bugRepository.save(bug);
        }
        return true;
    }
```

图 4.4: 缺陷报告管理模块关键代码截图

其中，predict 方法根据传入的测试方法和缺陷报告 id 列表构造新的 ClassificationRequestDTO 对象，ClassificationRequestDTO 对象是 Java 后端调用分类服务时传输的对象，将需要获取分类的缺陷报告集的标题和描述传递给分类服务。predict 方法在同包内调用 postClassifyService 方法使用 Resttemplate 类间接调用 Python 端的 RestfulAPI 接口，获取 postClassifyService 方法返回的 ClassificationResponseDTO 类对象，拿到 ClassificationResponseDTO 对象的 BugDTO

集的类别属性，存储到数据库中。postClassifyService 方法负责具体和 Python 端的交互，向给定的 classifyServiceURL 地址发送 POST 请求，请求附带的实体设置为 ClassificationRequestDTO，使用 Gson 类读取 Python 端的返回值，解析为 ClassificationResponseDTO 类型。

4.2 缺陷报告特征提取模块设计与实现

缺陷报告缺陷报告特征提取模块是特征融合模块依赖的前置模块，主要负责对原始的测试报告数据做预处理，主要包括文本和图像信息的数据读取，数据处理，特征提取以及特征向量归一化步骤。其中，图像信息需要提前从云端 OSS 下载。

4.2.1 交互顺序设计

缺陷报告缺陷报告特征提取模块时序图主要进行了测试报告的数据预处理工作。主要功能是输入测试报告数据，输出提取后的文本和图像特征向量。无论是模型训练还是增量数据的分类请求，都需要对报告数据进行特征提取，这里以增量数据分类请求为例说明。在用户选择测试报告后进行自动分类请求时，通过 Nginx 转发，后端主程序中的 BugController 接收到前端 Http 请求，并调用 BugService 的 predict 方法进行分类标签预测，BugService 调用第三方依赖的分类服务 ClassifyService。ClassifyService 开始调用特征提取服务 ExtractionService，进行文字和图片的特征提取工作。文本和图像的特征提取工作可以同时进行，可以分别抽取为 WordPropressService 和 FigPropressService，在时序图中进行了标注。

文本特征提取时，需要调用 wordRead 方法进行文本数据读取，之后借助 LTP 服务进行文本数据预处理。首先，调用 remove_stop_words 方法去除文本中无意义的停用词，再调用 synonyms_replace 方法进行同义词替换，随后调用 split_words 方法进行文本分词。这时候完成了基本的数据处理工作，借助 TF-IDF 服务进行特征提取。先调用 tf_idf_count_vec 方法进行词频统计，再通过 tf_idf_cvt_vec 生成文本向量，最后调用 neomalized_vec 方法进行文本向量归一化，完成文本向量的特征提取。

图像特征提取时，需要调用 figRead 方法进行图像数据读取，之后借助 CV2 服务进行图像数据预处理。首先，调用 resize 方法剪裁图像，统一图像像素为 140*140，之后调用 cvtColor 方法对图像统一进行灰度化处理。再借助 SPM 服务，调用 spm2Vector 方法完成图像向量提取，最后调用 neomalized_vec 方法进行图像向量归一化，完成图像向量的特征提取。缺陷报告特征提取模块时序图

如下图 4.5所示。

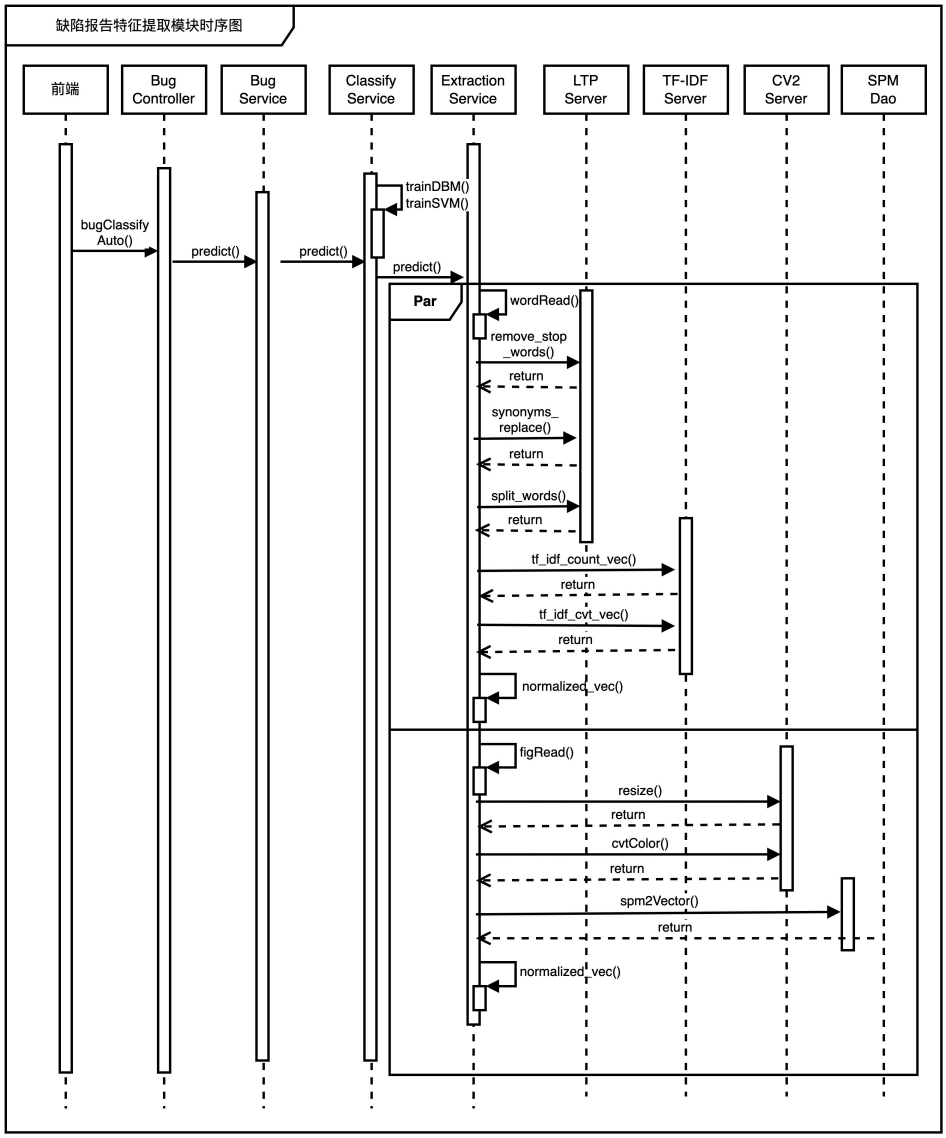

图 4.5: 缺陷报告特征提取模块时序图

4.2.2 数据与类设计

缺陷报告特征提取模块被缺陷报告分类功能依赖的分类服务调用，主要用于提取原始缺陷报告中的文本和图像信息，并生成对应的特征向量表示。这一操作用于根据历史的缺陷报告数据训练 DBM 模型。特征提取服务 (FeatureExtractionService) 包含用于加载缺陷报告数据的 loadBugData 方法，生成文本向量的 getWordVector 方法以及生成图片向量的 getFigVector 方法。

WordPropressService 用于预处理缺陷报告中的文本信息并生成文本向量。为了生成文本向量，首先需要读取文本信息 (wordRead)，之后去除停用词 (re-

move_stop_words), 并进行同义词替换 (synonyms_replace)，接着进行文本分词 (split_words)。这些预处理方法都依赖于第三方 LTP 提供的相关接口服务。文本向量生成则依赖 TF_IDF 服务。首先需要进行词频统计 (tf_idf_count_vec), 之后生成文本向量 (tf_idf_cvt_vec) 并进行归一化 (normalized_vec)。

FigProcessService 用于预处理缺陷报告中的图像信息并生成图像向量。为了生成图像向量，首先需要读取图像信息 (figVec)，之后统一尺寸调整 (resize)，图片灰度化处理 (cvtColor)。以上预处理方法都依赖第三方 CV2 哭提供的相关接口服务。图像向量生成需要借助 SPM 服务，通过 SPM 得到图片向量 (spm2vector)，并进行归一化 (normalized_vec)。缺陷报告特征提取模块核心类图如图 4.6所示。

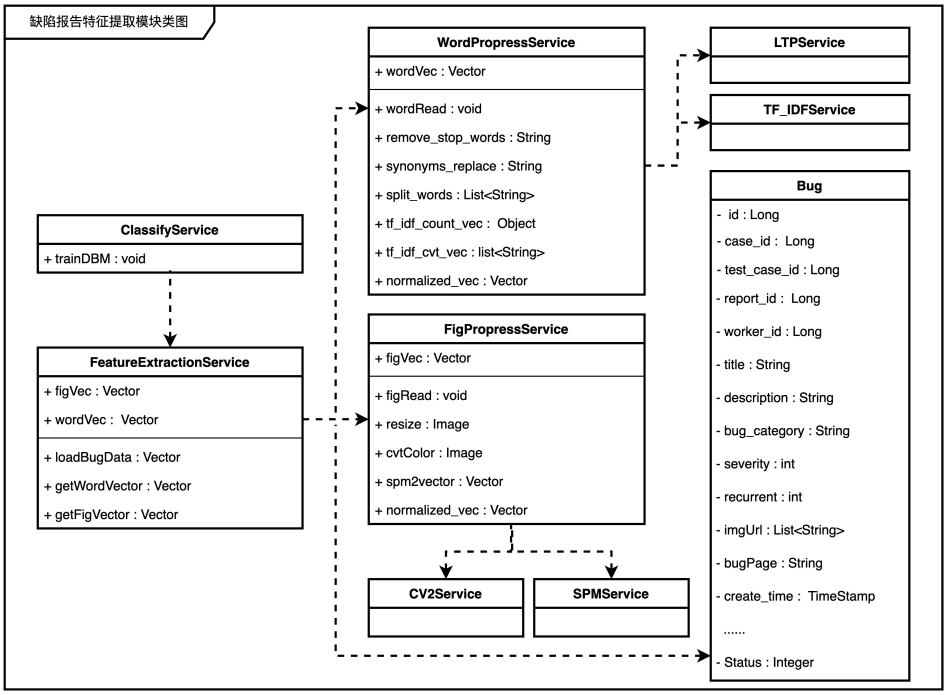

图 4.6: 缺陷报告特征提取模块类图

4.2.3 关键代码

图 4.7展示了文本特征提取模块关键代码，主要使用 LTP 工具进行文本分词。其中 ltp_segmentor 方法的输入参数为原始的缺陷报告文本，首先初始化分词类 Segmentor，在按照指定文件地址加载完分词模型 cws.model 后，调用 Segmentor 类的 segment 方法对原始文本分词，返回分词后的关键词列表。类似的，ltp_postags 方法的输入参数为分词后的关键词列表，通过载入词性模型 pos.model 和自定义的词性表，通过调用 Postagger 类的 postag 方法返回关键词列表中每个关键词对应的词性列表。前人的工作显示文档中的名词和动词更能

反映文档的主要内容，因此我们利用 ltp_parser 方法，根据关键词列表和对应的词性列表处理缺陷描述文本，筛选保留文本中词性为动词和名词的关键词。

```
def ltp_segmentor(sentence):
    segmentor = Segmentor()
    segmentor.load('..\\ltp_data\\cws.model')
    words = segmentor.segment(sentence)
    segmentor.release()
    return list(words)

def ltp_postags(words):
    postagger = Postagger()
    model_path = '..\\ltp_data\\pos.model'
    lexicon_path = '..\\ltp_data\\posLexicon.txt'
    postagger.load_with_lexicon(model_path, lexicon_path)  # 加载自定义词性表
    postagger.load(model_path)
    postags = postagger.postag(words)
    postagger.release()
    return list(postags)

def ltp_parser(words, postags):
    words_filter = []
    for i in range(len(postags)):
        if postags[i] == "n" or postags[i] == "v":
            words_filter.append(words[i])
    return words_filter
```

图 4.7: 文本特征提取模块关键代码截图

如图 4.8展示了图像特征提取模块中的关键代码。首先，我们借助 Python 的 request 库从指定的网址中读取缺陷报告的图像内容并存储在文件系统中。接着使用 cv2 图像处理库的 imread 方法读取 jpg 图像为 ndarray 数组格式，使用 resize 方法剪裁图像尺寸，这个大小足以保存图像的特征信息，同时方便后续使用 SPM 模型提取图像特征。在将图像读取为统一大小的 numpy 数组后，我们使用 extract_DenseSift_descriptors 方法获取每张图像的 SIFT 描述符，为使用 SPM 空间金字塔匹配模型做预处理。extract_DenseSift_descriptors 方法的输入参数是 BGR numpy 数组，返回图像的密集 SIFT 描述符，如果在图像中没有检测到描述符，则返回 None。最后，通过 build_codebook 方法创建图像的 “词袋” 模型，以 codebook、图像本身和图像的 SIFT 描述为参数，调用 spatial_pyramid_matching 方法将图像划分为不同的子区域，并分别计算每个子区域上描述符对应关键点的特征。

4.3 缺陷报告特征融合模块设计与实现

缺陷报告特征融合模块是分类模块依赖的前置模块，也是本系统的核心模块。主要负责对特征提取模块输出的文本向量和图像向量进行融合，输出融合

```
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 "_" in dir_file_path and dir_file_path.split("_")[1] == "after.txt":
                img_base = dir_file_path.strip("after.txt")
                img0_path = img_base + "0.jpg"
                img0 = cv2.imread(img0_path)
                img0 = cv2.resize(img0,(96,96))
                imgs.append(img0)

def extract_DenseSift_descriptors(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    sift = cv2.xfeatures2d.SIFT_create()
    disft_step_size = DSIFT_STEP_SIZE
    keypoints = [cv2.KeyPoint(x, y, disft_step_size)
            for y in range(0, gray.shape[0], disft_step_size)
                for x in range(0, gray.shape[1], disft_step_size)]
    keypoints, descriptors = sift.compute(gray, keypoints)
    return [keypoints, descriptors]

img_feature = [extract_DenseSift_descriptors(img) for img in imgs]
img_kp, img_des = zip(*img_feature)
codebook = build_codebook(img_des, VOC_SIZE)
import pickle
with open('./spm_lv1_codebook2.pkl','wb') as f:
    pickle.dump(codebook, f)
imgs = [spatial_pyramid_matching(imgs[i],img_des[i], codebook, level=PYRAMID_LEVEL) for i in
range(len(imgs))]
imgs = np.asarray(imgs)
```

图 4.8: 图像特征提取模块关键代码截图

后的高维特征向量。通过融合的多模态向量，对缺陷报告数据进行更确切的表达，从而得到更准确的分类结果。

4.3.1 交互顺序设计

特征融合依赖 DBM 服务的 DBM 模型，因此需要提前用历史报告数据训练出可供使用的 DBM 模型。DBM 模型主要用于处理向量的融合，这里抽取出了 FusionService 用于融合操作。由第三方分类服务 ClassifyService 调用 trainDBM 方法触发 DBM 模型的训练。融合服务 FusionService 调用 DBMServer 服务中 preTrain 方法，根据默认参数进行预训练，并通过 RBM 的对比散度算法进行参数调整，达到每相邻两层之间的稳定。这里调用 rbm_contrastive_divergence 方法实现参数调整。之后通过调用 fineTuningParam 方法实现通过历史报告数据进行模型的优化调整，这里会调用 DBM 的对比散度算法 dbm_contrastive_divergence 进行参数调整，达到整个三层 DBM 的稳定。最终将预训练好的模型参数通过 saveParam 存储起来，再后续进行特征融合时候载入即可调用模型。

当前端触发分类请求后，后端 BugController 接收请求并调用 BugService 处

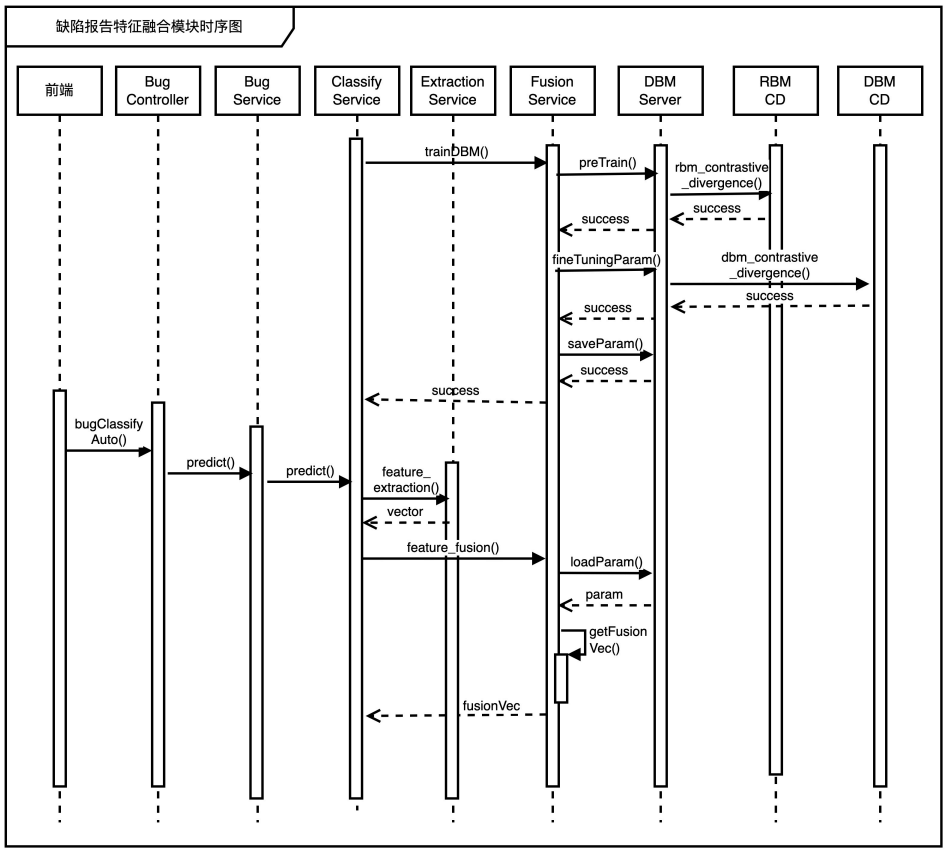

图 4.9: 缺陷报告特征融合模块时序图

理，BugService 调用第三方 ClassifyService 进行分类。ClassifyService 首先需要调用 ExtractionService 进行预处理与特征提取，调用 FusionService 进行特征融合。FusionService 会调用 DBMSever 中 loadParam 方法加载预训练过的模型参数，拿到参数后对 ExtractionService 中提取到的文本向量和图像向量进行融合。最终输出融合后的高维特征向量。缺陷报告特征融合模块时序图如图 4.9所示。

4.3.2 数据与类设计

缺陷报告特征融合模块被缺陷报告分类功能依赖的分类服务调用，主要用于将特征提取模块生成的文本向量和图像向量融合成高维特征向量。这一操作同样用于根据历史的缺陷报告数据训练 DBM 模型。

特征融合服务 (FeatureFusionService) 包含用于特征融合的 getFusionVector 方法，生成高维融合向量 fusionVec，这一融合操作依赖特征提取服务 (FeatureExtractionService) 中生成的文本向量和图像向量。

DBMService 主要用于生成用于融合向量的 DBM 模型，并将模型的有关参

数持久化存储。首先根据默认参数进行预训练 (preTrain), 如默认隐藏层节点个数，可视层节点个数，学习效率，迭代次数等，并根据对比散度算法进行参数调整相邻两层参数 (rbm_contrastive_divergence), 之后根据历史缺陷报告数据进行训练微调 (fineTuningParam)，此时调用的是 DBM 的对比散度算法调整三层 DBM 参数 (dbm_contrastive_divergence), 并将稳定后的参数进行持久化存储 (saveParam)。当需要使用 DBM 模型时，再导入存储好的参数数据即可 (loadParam)。缺陷报告特征融合模块核心类图如图 4.10所示。

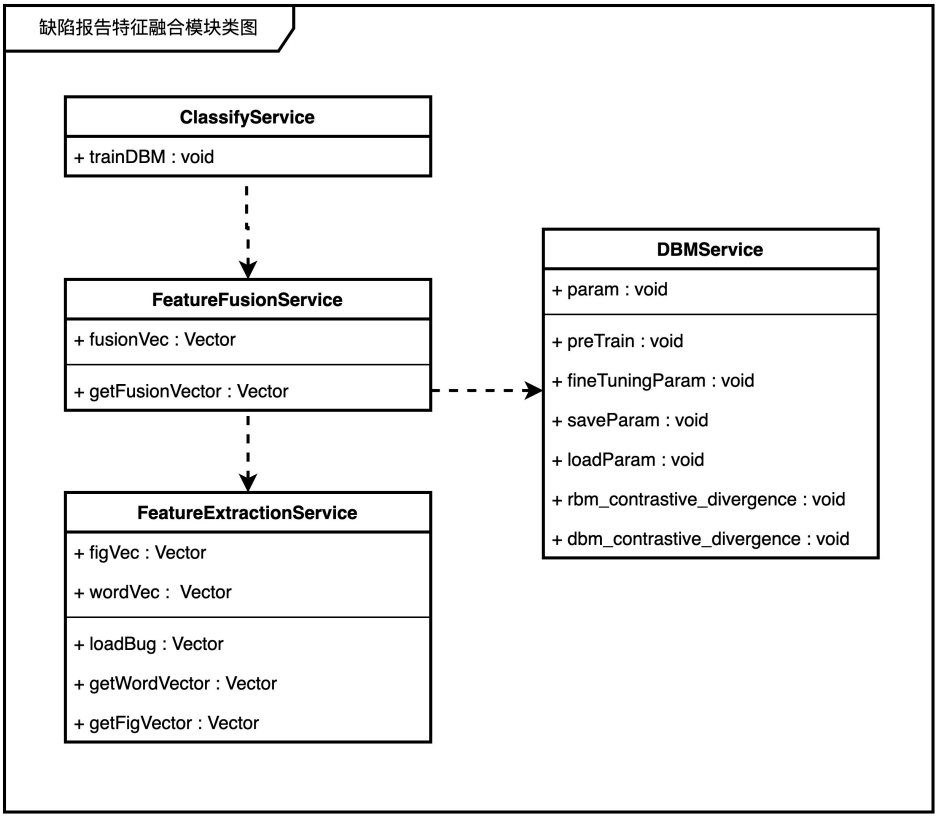

图 4.10: 缺陷报告特征融合模块类图

4.3.3 关键代码

图 4.11展示了特征融合模块中使用对比散度算法微调 DBM 模型参数的部分代码。代码的首先使用不同批次的训练数据调用 dbm_contrastive_divergence 方法获得这轮更新的参数步长，上轮迭代值加上步长乘以训练学习率可以更新得到本轮 DBM 模型中的所有参数。dbm_contrastive_divergence 方法实现了三层 DBM 的对比散度算法，输入参数是上一轮数据 data 和上一轮训练中 DBM 模型的所有连接权重和偏置参数，其中调用 np.random.binomial 方法使用吉布斯采样

模拟数据的分布，重新计算出被吉布斯采样后的隐藏层数据和可见层数据，返回本轮训练拟合出的数据和参数的更新。在本模块，我们初步将图像 DBM 模型可视层神经元节点个数设置为 9000 个，对应图像特征提取出的维度也为 9000 个，图像第一层隐藏层神经元节点个数设置为 1000 个，第二个隐藏层神经元节点个数设置为 500 个。将文本 DBM 模型可视层神经元节点个数设置为 815 个，对应文本特征提取出的特征维度为 815 维，第一层隐藏层神经元节点个数设置为 500 个，第二个隐藏层神经元节点个数设置为 500 个。而对于融合 DBM 模型，我们将可视层神经元节点个数设置为 1000 个，对应图像 DBM 模型输出特征数和文本 DBM 模型输出特征数的相加，第一层隐藏层神经元节点个数设置为 800 个，第二个隐藏层神经元节点个数设置为 500 个。三个模型训练后的参数全部存储为 npy 文件，供增量缺陷报告特征向量调用，获取高维融合向量。

```
for i in range(train_epochs):
    # Calculate gradient
    update_b, update_c1, update_c2, update_w_vh1, update_w_h1h2 \
                                    = dbm_contrastive_divergence(data, b, c1, c2, w_vh1, w_h1h2)
    # Update parameters 更新所有的偏置以及连接权重
    b += train_learning_rate * update_b
    c1 += train_learning_rate * update_c1
    c2 += train_learning_rate * update_c2
    w_vh1 += train_learning_rate * update_w_vh1
    w_h1h2 += train_learning_rate * update_w_h1h2


def dbm_contrastive_divergence(data, b, c1, c2, w_vh1, w_h1h2, num_sample=100):
    m_vis = data
    m_h1 = np.random.uniform(size=(len(data), len(c1)))
    m_h2 = np.random.uniform(size=(len(data), len(c2)))
    for i in range(num_sample):
        m_h1 = expit( np.dot(m_vis, w_vh1) + np.dot(w_h1h2, m_h2.T).T + c1 )
        m_h2 = expit( np.dot(m_h1, w_h1h2) + c2 )
    # Gibbs sample 吉布斯采样
    s_vis = np.random.binomial(1, m_vis)
    s_h1 = np.random.binomial(1, 0.5, size=(len(data), len(c1)))
    s_h2 = np.random.binomial(1, 0.5, size=(len(data), len(c2)))
    for i in range(num_sample):
        sm_vis = expit( np.dot(w_vh1, s_h1.T).T + b )
        s_vis = np.random.binomial(1, sm_vis)
        sm_h1 = expit( np.dot(s_vis, w_vh1) + np.dot(w_h1h2, s_h2.T).T + c1 )
        s_h1 = np.random.binomial(1, sm_h1)
        sm_h2 = expit( np.dot(s_h1, w_h1h2) + c2 )
        s_h2 = np.random.binomial(1, sm_h2)
    return np.mean(m_vis - s_vis, axis=0), np.mean(m_h1 - s_h1, axis=0), np.mean(m_h2 - s_h2, axis=0), \
                ( np.dot(m_vis.T, m_h1) - np.dot(s_vis.T, s_h1) ) / len(data), ( np.dot(m_h1.T, m_h2) -
np.dot(s_h1.T, s_h2) ) / len(data)
```

图 4.11: 特征融合模块关键代码截图

4.4 缺陷报告分类模块设计与实现

缺陷报告分类模块是系统最重要的功能模块，主要负责将特征融合模块提取到的高维融合向量映射到定义好的分类标签上，实现自动化缺陷报告分类功能。

4.4.1 交互顺序设计

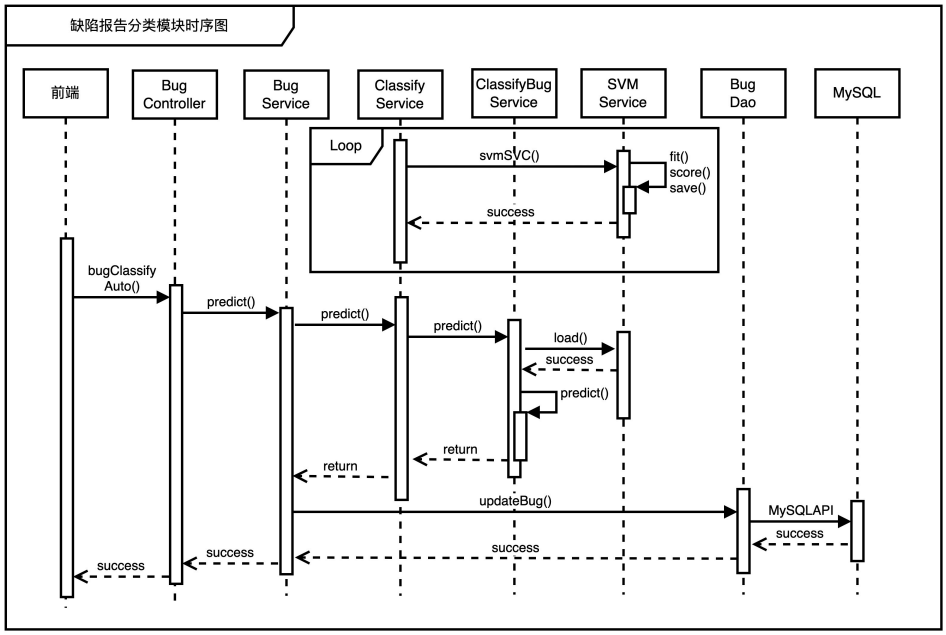

图 4.12: 缺陷报告分类模块时序图

缺陷报告分类模块时序图如图 4.12所示。缺陷报告分类模块在用户选择自动分类时触发，自动分类模型需要提前进行预训练。根据历史缺陷报告数据，由第三方服务 ClassifyService 调用 SVMService 完成分类模型的预训练。首先，ClassifyService 调用 SVMService 的 svmSVC 方法进行参数参数化，之后 SVMService 内部调用 fit 方法进行数据拟合，调用 score 方法根据拟合结果对当前模型评分，并保存当前参数。持续这一操作，直到模型得分满意为止，此时也就得到了用于自动分类的分类模型。

当前端触发自动分类请求时，BugController 调用 BugService 处理，BugService 调用第三方 ClassifyService 分类。ClassifyService 此时已经完成了模型训练。首先，调用 ClassifyBugService，ClassifyBugService 会调用 SVMService 中 load 方法导入分类模型，通过导入的模型进行分类标签预测，并将分类结果返回到 BugService。再由 BugService 调用 BugDao 的 updateBug 方法，将分类后的报告

信息更新至数据库中，完成分类操作。

4.4.2 数据与类设计

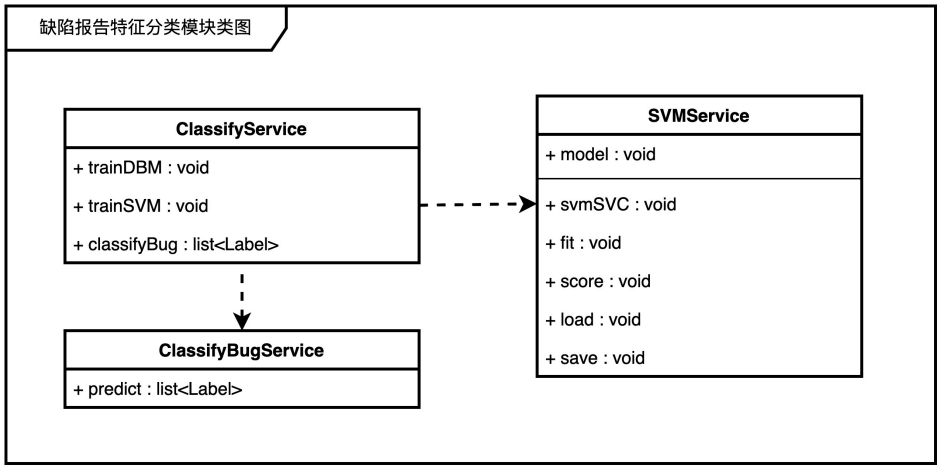

图 4.13: 缺陷报告分类模块类图

缺陷报告分类模块核心类图如图 4.13所示。缺陷报告特征分类模块被缺陷报告分类功能依赖的分类服务调用，主要用于对指定的缺陷报告进行自动化分类。这一操作用于用户请求自动化分类缺陷报告数据的场景。

缺陷报告分类服务 (ClassifyBugService) 包含用于自动化分类的 predict 方法，返回给用户对应缺陷报告的类别标签信息。

SVMService 主要用于生成缺陷报告分类的 SVM 模型，并将模型进行持久化存储。首先进行模型初始化 (svmSVC)，之后根据历史报告进行模型训练 (fit)，然后对分类结果进行打分 (score)。将训练后的模型进行持久化存储 (save)，在需要调用 SVM 模型时，再导入模型即可 (load)。

4.4.3 关键代码

如图 4.14展示了分类模块中使用 SVM 算法分类缺陷报告高维特征向量的部分代码。我们调用 sklearn 机器学习库的 train_test_split 方法分割训练集和测试集的比例为 8 比 2。调用 svm.SVC 方法初始化 SVM 分类器，输入参数 C 表示错误项的惩罚系数，C 越大对分错样本的惩罚程度越大，因此在训练样本中准确率越高，反而泛化能力越低。输入参数 kernel 表示算法中采用的核函数类型，“rbf”意为高斯核。接着，我们调用 fit 方法根据训练集训练 SVM，调用 predict 方法基于之前的训练对测试集中的每个样本进行类别预测。

```
train_data,test_data,train_label,test_label =sklearn.model_selection.train_test_split(union, labels_new,
random_state=1, train_size=0.8,test_size=0.2)

classifier=svm.SVC(C=2,kernel='rbf',gamma=10,decision_function_shape='ovr')
classifier.fit(train_data,train_label.ravel())

print("训练集：",classifier.score(train_data,train_label))
print("测试集：",classifier.score(test_data,test_label))
```

图 4.14: 缺陷报告分类模块关键代码截图

4.5 本章小节

本章是全文的核心章节，展示了系统缺陷报告管理模块，缺陷报告特征提取模块，缺陷报告特征融合模块和缺陷报告分类模块的详细设计与实现。借助时序图，类图，界面截图，关键代码等形式，通过交互流程分析，数据类分析，关键实现分析介绍了系统实现的不同方面。

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

本章节主要介绍了众包缺陷分类系统的功能测试和性能测试。功能测试需要对系统的各项功能用例展开测试，验证系统功能是否达到用户的使用需求，满足功能验收标准。这里根据第三章需求分析得到用例和功能性需求，测试系统在查看应用列表，查看已分类报告列表，查看未分类报告列表，查看报告详情，手动单个报告分类，自动多个报告分类，和查看报告分类情况等功能下是否满足功能标准。性能测试用于衡量在不同的系统环境下，系统的各项性能指标是否可以达到正常标准。这里主要考虑到多用户使用系统的情况，检查系统在高并发环境下的性能情况。

5.1.2　测试环境

下表5.1总结了系统测试所需的测试环境信息。

表 5.1: 系统测试环境说明表

设备与软件	程序	详细信息
用户计算机	Chrome 浏览器	配置：4 核 8G 系统：MacOS 10.15.7
React 前端服务器	React 17.0.0	配置：4 核 8G 系统：Ubuntu 16.04
Java 后端服务器	Java 1.8.0_027	配置：4 核 8G 系统：Ubuntu 16.04
Python 分类服务器	Flask 1.1.1	配置：4 核 8G 系统：Ubuntu 16.04
MySQL 数据服务器	MySQL 5.7.0	配置：4 核 8G 系统：Ubuntu 16.04
Nginx 服务器	nginx	Nginx 1.19.0
Docker 服务器	docker	Docker 19.03.6

根据上文第 3.3.3 节设计的物理部署图可知，系统的测试环境主要包括用户用于安装运行 Chrome 浏览器的本地计算机，主要用来进行系统交互；用于运行前端 React 程序的前端服务器；用于部署运行 Nginx 代理和 Docker 容器的 Java 后端服务器；用于部署运行 Flask 程序的 Python 分类服务器；用于部署运

行 MySQL 的数据服务器。考虑到成本以及产品社区友好情况，本系统所有服务统一托管在阿里云服务器上。根据服务运行所需的不同环境要求，不同服务部署的服务器配置有所差别。

5.2 功能测试

5.2.1 测试设计

功能测试是对产品功能进行逐项验证以检测是否符合用户需求。本小节根据第三章系统需求分析后得出的系统功能需求进行测试用例设计，目的是保证系统能够完整地提供预期功能，满足用户业务需求，达到功能验收的标准。缺陷分类系统的具体测试用例设计如下：

表5.2展示了查看应用列表的测试用例，测试的是用户登录进入系统后进入的首页面，主要关注的是应用列表能否正确展示，需要注意的是鼠标悬浮应用详情时描述信息是否会展示完整，鼠标悬浮分类进度是否会正确展示报告总数和已分类数。

表 5.2: 查看应用列表测试用例表

测试用例编号	TC1
测试名称	查看应用列表测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录审核员用户账号
预期结果	1. 进入应用列表页面，包括应用名称，应用描述，分类信息等，鼠标悬浮在应用详情，展示完整详情信息，鼠标悬浮在分类进度，展示应用报告数和已分类报告数

表5.3展示了查看已分类报告列表测试用例，测试的是用户登录进入系统选择应用后的已分类缺陷报告列表界面，主要关注的是缺陷列表及其关键能否正确展示，需要注意鼠标悬浮时缺陷描述和标题信息是否会展示完整，以及是否正确展示缺陷类别信息。

表5.4展示了查看未分类报告列表测试用例，测试的是用户登录进入系统选择应用后的未分类缺陷报告列表界面，主要关注的是缺陷列表及其关键能否正确展示，需要注意鼠标悬浮时缺陷描述和标题信息是否会展示完整，与“查看已分类报告列表”不同的是，此时并不展示缺陷类别信息，并且提供了多选分类的入口。

表5.5展示了查看报告详情测试用例，测试的是用户选择任一缺陷报告后进入的报告详情界面，主要关注的是详情信息是否完整展示，对空字段的展示是

表 5.3: 查看已分类报告列表测试用例表

测试用例编号	TC2
测试名称	查看已分类报告列表测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录审核员用户账号 2. 点击左侧侧边栏“已分类报告” 3. 点击任一已分类报告
预期结果	1. 进入应用列表页面 2. 展示出所有已分类报告列表，包括缺陷标题，缺陷类别，缺陷等级等 3. 展示出选中缺陷报告详情信息

表 5.4: 查看未分类报告列表测试用例表

测试用例编号	TC3
测试名称	查看未分类报告列表测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录审核员用户账号 2. 点击左侧侧边栏“已分类报告”
预期结果	1. 进入应用列表页面 2. 展示出所有已分类报告列表，但是列表不包含缺陷列表信息

否合理，需要注意已分类报告包含缺陷分类字段，未分类报告不包含测试用例字段。

表 5.5: 查看报告详情测试用例表

测试用例编号	TC4
测试名称	查看报告详情测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录审核员用户账号 2. 点击选择“所有报告”下任一缺陷报告，或者点击“未分类报告”后选择报告，或者点击“已分类报告”后选择报告
预期结果	1. 进入应用列表页面 2. 展示出报告详情信息，包括缺陷标题，缺陷描述，缺陷可复现程度等，已分类报告包含缺陷类别，不包含未分类报告

表5.6展示了手动单个报告分类测试用例，测试的是用户选择手动对单个缺陷报告进行分类的功能，主要关注的是分类后报告是否还在未分类报告列表中，已分类列表中是否更新，需要注意的是相应应用分类进度以及应用的分类统计信息是否进行了相应数据的改变。

表5.7展示了自动多个报告分类测试用例，测试的是用户在未分类列表选择

表 5.6: 手动单个报告分类测试用例表

测试用例编号	TC5
测试名称	手动单个报告分类测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录审核员用户账号 2. 点击“未分类报告”进入未分类报告列表 3. 点击选择想要分类的缺陷报告 4. 点击选择指定的缺陷类别
预期结果	1. 进入应用列表页面 2. 展示未分类报告列表，以及关键属性信息 3. 展示所选择报告的详细信息 4. 提示分类成功，返回未分类报告列表

多个缺陷报告并进行自动化分类的功能，主要关注的是分类后报告是否还在未分类报告列表中，已分类列表中是否更新，同样需要注意的是相应应用分类进度以及应用的分类统计信息是否进行了相应数据的改变，以及确认框中的待分类缺陷报告数目是否准确。

表 5.7: 自动多个报告分类测试用例表

测试用例编号	TC6
测试名称	自动多个报告分类测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录审核员用户账号 2. 点击“未分类报告”进入未分类报告列表 3. 点击选择多个想要分类的缺陷报告 4. 点击分类按钮 5. 弹出提示框后，点击确认按钮
预期结果	1. 进入应用列表页面 2. 展示未分类报告列表，以及关键属性信息 3. 弹出确认框，提示“是否确定对所选的多个缺陷报告进行自动分类” 4. 如果点击了确认按钮，则对所选择的报告进行分类，否则则返回列表页 5. 提示分类成功

表5.8展示了查看报告分类结果测试用例，测试的是用户在选择应用后进入的分类统计界面，主要关注的是各类分类展示图表是否正确展示，数据是否正确，描述信息是否完整，需要注意的是在保证数据准确的同时也要保证图标可以合理并尽量清晰的反映出数据状态。

表 5.8: 查看报告分类结果测试用例表

测试用例编号	TC7
测试名称	查看报告分类结果测试
前置条件	用户已经得到授权和认证
测试步骤	1. 用户登录审核员用户账号 2. 点击“分类统计”
预期结果	1. 进入应用列表页面 2. 展示当前应用分类信息，包括不同类别的缺陷报告数量，分类进度等

5.2.2 测试结果

在上述测试环境下严格按照测试用例步骤测试后，将测试结果与预期结果进行对比，并记录差异点。表5.9展示了上述测试用例的测试结果。

表 5.9: 系统功能测试结果表

测试用例 ID	用例描述 ID	测试结果
TC_1	UC1	通过，查看应用列表成功
TC_2	UC2	通过，查看已分类报告列表成功
TC_3	UC3	通过，查看未分类报告列表成功
TC_4	UC4	通过，查看报告详情成功
TC_5	UC5	通过，手动单个报告分类成功
TC_6	UC6	通过，自动多个报告分类成功
TC_7	UC7	通过，查看报告分类情况成功

5.3 性能测试

5.3.1 测试设计

本小节主要测试缺陷报告分类系统的性能在实际使用环境下是否能够满足用户需求。

表 5.10: 系统测试环境说明表

测试接口编号	接口描述	接口路径
I1	查看 app 列表	/api/case
I2	查看缺陷报告列表	/api/bug
I3	查看缺陷报告详情	/api/bug/{id}
I4	手动分类缺陷报告	/api/bugClassify
I5	自动分类缺陷报告	api/bugClassifyAuto

通过 JMeter 模拟实际开发环境以及高并发使用场景，测试本系统能否如期稳定运行，验证本系统能否满足实际使用需求以及是否具备实用价值。通过分析本系统的功能需求，结合系统实际的开发接口，需要测试的主要接口如表5.10所示。

以查看缺陷报告详情接口为例，JMeter 配置如下：

（1）通过 JMeter 创建线程组并配置参数，Number Of Thread（线程数）为 200，Ramp Up Period（启动所有线程所需要的时间）为 10，Loop Count（循环测试）为 5。在一个测试计划中所有的测试任务都是基于线程组的，这里的线程组配置模拟了 2000 个用户线程数，每个用户会在 10s 内循环发送 5 次请求，一共发送 1000 次。图5.1展示了具体的配置信息。

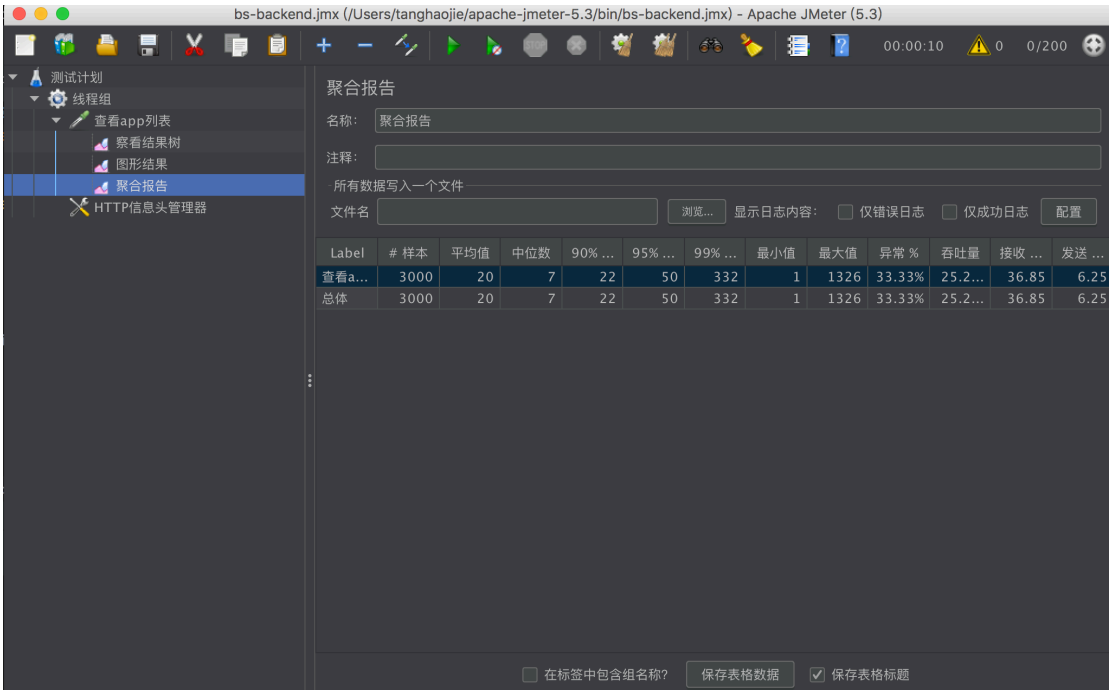

图 5.1: JMeter 配置截图

（2）在线程组中添加接口对应的 HTTP 请求进行压力测试。需要配置请求名称，请求协议，IP 地址，端口号，请求方法，请求路径，参数信息，内容编码规则等等。当接口使用 POST 方法请求时，需要额外配置请求参数信息作为请求体，请求体会随着请求一同转发到服务器。GET 请求接口具体的配置信息如图5.2所示。

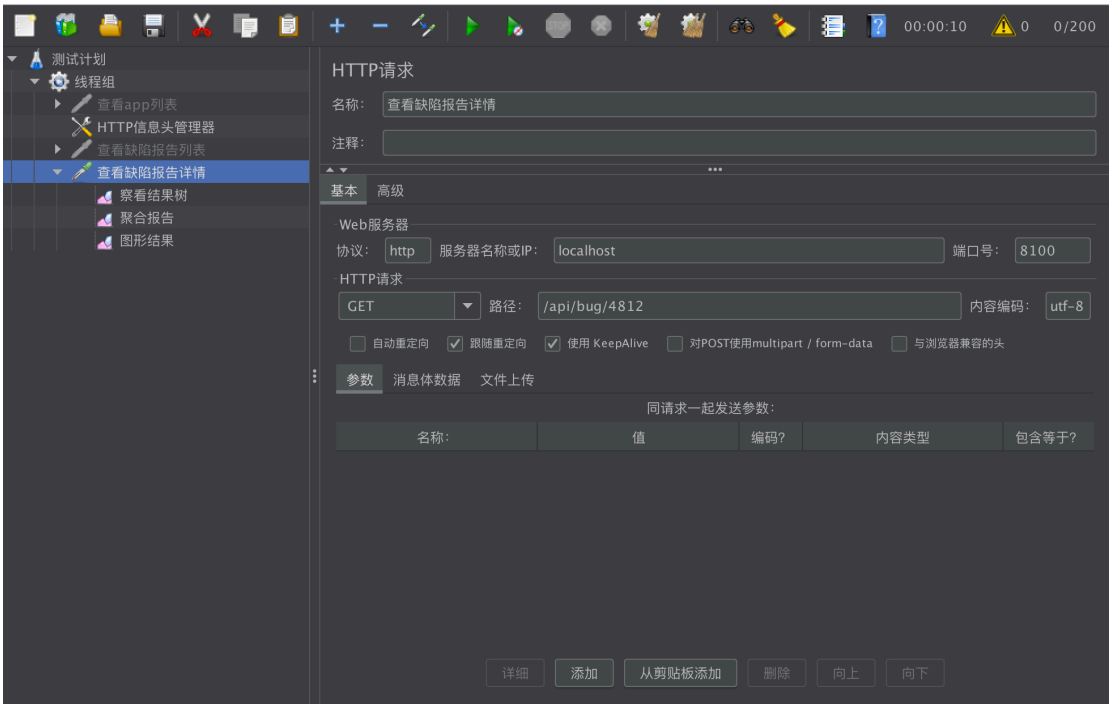

图 5.2: GET 请求 JMeter 参数配置截图

5.3.2 测试结果

根据上述性能测试设计方案严格实行后，对比并记录测试结果。表5.11展示了本系统接口的测试结果。

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%	20ms	50ms	332ms
I2	0%	8ms	18ms	51ms
I3	0%	11ms	31ms	62ms
I4	0%	30ms	124ms	248ms
I5	0%	77ms	242ms	488ms

5.4 效果测试

在验证了系统的可用性和可靠性之后，我们还需要对分类效果进行评估。检查由基于深度玻尔兹曼机的众包测试缺陷报告分类器的效果。

5.4.1 测试对象

为了验证系统效果，我们选取了三场真实众包测试比赛，提取测试报告作为实验对象，开展系统效果测试。

5.4.2 测试设计

为验证本系统是否具备使用价值，我们设置了两个对比实验。针对自动分类和手动分类时长进行对比，验证系统在提升分类效率方面的价值。针对不同算法的准确率进行对比，分别使用基于随机分类算法、拼接向量分类和融合向量分类。随机算法使用随机函数对报告进行分类，拼接向量分类中直接拼接了文本和图像特征向量后输入 SVM 分类器，融合向量分类则在提取了文本和图像向量后，使用 DBM 进行特征融合，最终将得到的高维融合向量输入 SVM 分类器。在三个数据集上建立分类器，这些分类器使用 20% 的数据充当训练集训练，使用 80% 的数据作为测试集，来模拟测试报告集在真实场景下已分类样本数较少的情况，统计每个分类器的准确率，查看算法效果，验证系统在报告分类方面的价值。在实验中，我们随机划分数据集 15 次，并进行了 15 次实验，最终取平均值。

5.4.3 测试结果

表5.12展示了 AB 两组测试人分别通过手动单个报告分类和自动多个报告分类时的平均分类时长统计情况。根据统计结果，A 组测试人员的平均分类时长均比 B 组测试人员的时长超出大约 30 分钟。

表 5.12: 测试人员分类时长统计表

被分类数据集名称	报告数	A 组平均分类时长	B 组平均分类时长
航天中认练习赛	618	58.0min	29.5min
决赛自主可控管理系统	668	62.5min	30.5min
趣享 GIF 众包测试	519	50.5min	27.0min

表5.13展示的了本系统的效果测试结果。行数据表示同一数据集下采用不同算法分类的准确率，列数据表示同一算法在不同数据集上的分类准确率。可以看到，基于 DBM 特征融合的分类算法在不同数据集上的准确率均为最佳，平均分类准确率约 50.6%。图5.3更加清晰的体现了同一数据集下使用不同算法的分类效果差异。

表 5.13: 分类准确率结果统计表

被分类数据集名称	随机分类	拼接向量分类	融合向量分类
航天中认练习赛	0.166	0.500	0.502
决赛自主可控管理系统	0.172	0.496	0.538
趣享 GIF 众包测试	0.163	0.443	0.479

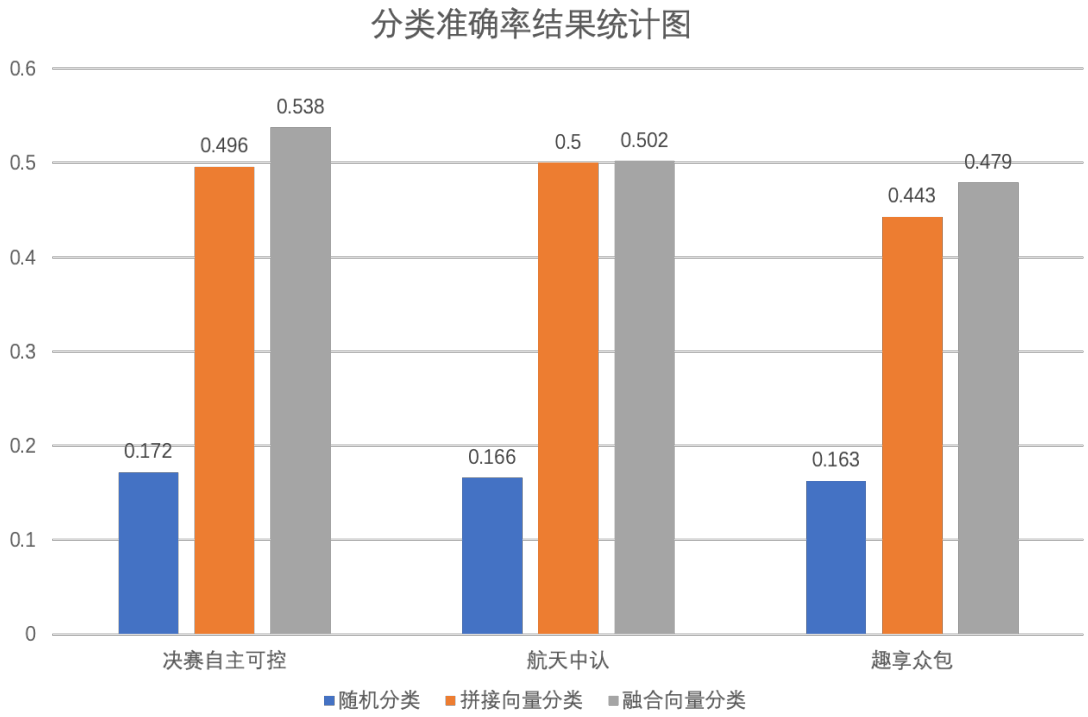

图 5.3: 分类准确率结果统计图

5.5 本章小节

本章主要对系统进行系统测试和实验分析。其中，功能测试基于第三章需求分析得到的功能性需求和用例开展，验证用例是否满足用户需求。性能测试通过使用 JMeter 模拟实际高并发场景开展，验证系统的性能在实际使用环境下是否能够满足用户需求。效果测试基于分类结果开展，验证系统分类结果的准确性，系统的使用价值。

第六章　总结与展望

6.1　总结

日益复杂的软件需求促进了测试领域的发展，众包测试作为软件测试领域的新兴趋势，结合了众包和云平台的优势，成本低、效率高且不受平台限制，受到了越来越多的关注。然而由于众包测试人员的非专业性和庞大基数，导致产生的测试报告数量大且重复率高，增加了审核人员的压力，也不利于后期的缺陷分配工作。另一方面，测试人员自身对缺陷的分类也不一定正确。针对上述问题，结合众包平台的特点，本文提出了基于深度玻尔兹曼机的测试报告分类方式，并开发了众包缺陷分类系统。

本系统提供了两种测试报告分类模式，单报告手动分类和多报告自动分类。在手动分类场景中，审核人员在测试报告详情界面选择合适的缺陷类别即可。在自动分类场景中，需要提前根据历史报告数据集进行融合模型和分类模型的预训练。审核人员在进入测试报告列表界面后，选择需要分类的测试报告集，点击分类即可。被选中的测试报告会经过数据预处理，特征提取和特征融合过程，并最终落入预定义好的缺陷类别中。自动化分类避免了单一的分类操作，提高了分类效率。系统的应用列表界面可以查看不同应用的分类进度，在应用的分类统计页面可以查看当前应用的分类统计信息，包含不同类别的缺陷数量，不同类别下缺陷严重等级的划分数据等。这些统计信息可以对开发成果进行反馈，从而辅助软件开发工作更好的开展。高效准确的报告分类不仅便于审核工作更好地划分，也便于将缺陷分配到更适合的开发人员。

本文首先介绍了系统的研究背景，研究意义以及国内外众包测试和缺陷分类系统的研究现状，提出了基于深度玻尔兹曼机的众包缺陷分类系统的开发目标，介绍了系统开发需要的算法理论基础和技术框架，并分析阐述了技术选型的理由。接着，对系统进行整体说明后进行了需求分析，明确了系统的功能性需求和非功能性需求，并提取出系统用例，对每个用例进行了详细描述。之后根据用例进行系统架构设计，并划分为缺陷报告管理模块，缺陷报告特征提取模块，缺陷报告特征融合模块和缺陷报告分类模块。为了详细描述系统的设计模型，采用“4+1”视图从不同视角进行了更加详细了分析说明，并完成了数据模型设计，并通过流程图说明不同模块的工作流程。接着，对不同的功能模块，完成交互顺序设计，数据与类设计，并展示出实现的关键代码。为了确保系统的稳定运行，保证高质量的用户体验，本文还对系统进行了充分的测试，主要针对

系统功能，性能和效果展开。测试结果表明，本系统可以在高并发和大数据量的场景下平稳运行。最后展示了系统的部分实例截图，以及系统的最终效果。

6.2 展望

基于深度玻尔兹曼机的众包测试缺陷报告系统的目标是高效的完成测试报告的分类工作并尽可能的准确，降低审核人员的审核压力，辅助后续缺陷分配工作的开展。该系统还有诸多可以进行优化的方面，具体有下述几点：

(1) 模型方面，深度玻尔兹曼机的参数需要手动调整至最优。每个应用的模型都需要单独训练，单独调参。后期可以通过统一的机器学习算法进行自动调参，降低人力调参成本，提升模型训练效率。

(2) 数据方面，由于受到实验设备限制，在图像预处理步骤提取的图像像素为 140*140，像素较低，损失信息较多。后期可以提升图像的像素采集，提升图像特征准确率。

(3) 应用方面，目前实验的数据集较小，单一应用的数据集规模在约在 600 份报告。后期可以构建更大的样本数据集，通过扩大模型的训练集规模，提升模型性能。同时，可以通过分析缺陷的分类统计数据，得到缺陷的分布报告，对软件开发过程进行相应的反馈和调整。

参考文献

[1] HOME W, TECH M, REVIEWS P, et al. The Rise of Crowdsourcing[J], 2012.

[2] PENG X, BABAR M A, EBERT C. Collaborative Software Development Platforms for Crowdsourcing[J]. IEEE Software, 2014, 31(2): 30–36.

[3] 刘安战, 郭基凤. 软件众包开发者的能力价值率模型研究 [J]. 计算机应用研究, 2020.

[4] BRABHAM D C. Crowdsourcing as a model for problem solving: Leveraging the collective intelligence of online communities for public good[J]. Convergence the International Journal of Research Into New Media Technologies, 2010, 14(1): 75–90.

[5] 徐瀛, 程广明. 众测模式现状概述及生态效益分析 [J]. 科技创新导报, 2018, 15(29): 3.

[6] JAWERIA, KANWAL, ONAIZA, et al. Bug Prioritization to Facilitate Bug Report Triage[J]. 计算机科学技术学报：英文版, 2012, 27(2): 16.

[7] ANTONIOL G, AYARI K, DI PENTA M, et al. Is it a bug or an enhancement?[J], 2008: 304.

[8] 何俊, 张彩庆, 李小珍, et al. 面向深度学习的多模态融合技术研究综述 [J]. 计算机工程, 2020, 46(5): 11.

[9] MAO K, CAPRA L, HARMAN M, et al. A Survey of the Use of Crowdsourcing in Software Engineering[J]. Journal of Systems and Software, 2017: 57–84.

[10] DOLSTRA E, VLIEGENDHART R, POUWELSE J. Crowdsourcing gui tests[C] // 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation. 2013: 332–341.

[11] CHEN N, KIM S. Puzzle-based automatic testing: Bringing humans into the loop by solving puzzles[C] // 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. 2012: 140–149.

[12] GOMIDE V H, VALLE P A, FERREIRA J O, et al. Affective crowdsourcing applied to usability testing[J]. International Journal of Computer Scienceand Information Technologies, 2014, 5(1) : 575 – 579.

[13] WANG X, ZHANG L, XIE T, et al. An approach to detecting duplicate bug reports using natural language and execution information[C] // Proceedings of the 30th international conference on Software engineering. 2008 : 461 – 470.

[14] FENG Y, CHEN Z, JONES J A, et al. Test report prioritization to assist crowdsourced testing[C] // Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. 2015 : 225 – 236.

[15] WANG J, CUI Q, WANG Q, et al. Towards effectively test report classification to assist crowdsourced testing[C] // Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. 2016 : 1 – 10.

[16] WANG J, WANG S, CUI Q, et al. Local-based active classification of test report to assist crowdsourced testing[C] // Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. 2016 : 190 – 201.

[17] 颜炯, 王戟, 陈火旺, et al. 基于模型的软件测试综述 [D]. [S.l.] : [s.n.], 2004.

[18] 许静, 陈宏刚, 王庆人. 软件测试方法简述与展望 [J]. 计算机工程与应用, 2003, 39(13) : 75 – 78.

[19] BETTENBURG N, JUST S, SCHRÖTER A, et al. What makes a good bug report?[C] // Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. 2008 : 308 – 318.

[20] RUNESON P, ALEXANDERSSON M, NYHOLM O. Detection of duplicate defect reports using natural language processing[C] // 29th International Conference on Software Engineering (ICSE'07). 2007 : 499 – 510.

[21] ZANETTI M S, SCHOLTES I, TESSONE C J, et al. Categorizing bugs with social networks: a case study on four open source software communities[C] // 2013 35th International Conference on Software Engineering (ICSE). 2013 : 1032 – 1041.

[22] TIAN Y, LO D, SUN C. Drone: Predicting priority of reported bugs by multi-factor analysis[C] // 2013 IEEE International Conference on Software Maintenance. 2013 : 200 – 209.

[23] ZHOU Y, TONG Y, GU R, et al. Combining text mining and data mining for bug report classification[J]. Journal of Software: Evolution and Process, 2016, 28(3) : 150 – 176.

[24] YIN X, NG B W-H, ABBOTT D. Feature Extraction and Selection[G] // Terahertz Imaging for Biomedical Applications. [S.l.] : Springer, 2012 : 95 – 118.

[25] TAO Z, GAO J, JING C. Crowdsourced Testing Services for Mobile Apps[C] // Service-oriented System Engineering. 2017.

[26] CHANDRASHEKAR G, SAHIN F. A survey on feature selection methods[J]. Computers Electrical Engineering, 2014, 40(1) : 16 – 28.

[27] KHALID S, KHALIL T, NASREEN S. A survey of feature selection and feature extraction techniques in machine learning[C] // 2014 science and information conference. 2014 : 372 – 378.

[28] HAO L, HAO L. Automatic identification of stop words in chinese text classification[C] // 2008 International conference on computer science and software engineering : Vol 1. 2008 : 718 – 722.

[29] JIN X, ZHANG S, LIU J. Word semantic similarity calculation based on word2vec[C] // 2018 International Conference on Control, Automation and Information Sciences (ICCAIS). 2018 : 12 – 16.

[30] CHOWDHARY K. Natural language processing[J]. Fundamentals of artificial intelligence, 2020 : 603 – 649.

[31] LUO Y, ZHAO S, XIAOCHAO L I, et al. Text keyword extraction method based on word frequency statistics[J]. Journal of Computer Applications, 2016.

[32] CHE W, LI Z, LIU T. LTP: A Chinese Language Technology Platform[C] // The 23rd International Conference on Computational. 2010.

[33] DADGAR S, ARAGHI M S, FARAHANI M M. A novel text mining approach based on TF-IDF and Support Vector Machine for news classification[C] // IEEE International Conference on Engineering Technology. 2016.

[34] LI J. A Brief Introduction to the spm Package[J]. idp, 2017, 3(1) : 0 – 6.

[35] CULJAK I, ABRAM D, PRIBANIC T, et al. A brief introduction to OpenCV[C] // MIPRO, 2012 Proceedings of the 35th International Convention. 2012.

[36] YANG J, YU K, GONG Y, et al. Linear spatial pyramid matching using sparse coding for image classification[C] // 2009 IEEE Conference on computer vision and pattern recognition. 2009 : 1794 – 1801.

[37] LAZEBNIK S, SCHMID C, PONCE J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories[C] // Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. 2006.

[38] RAMACHANDRAM D, TAYLOR G W. Deep multimodal learning: A survey on recent advances and trends[J]. IEEE signal processing magazine, 2017, 34(6) : 96 – 108.

[39] GUO W, WANG J, WANG S. Deep multimodal representation learning: A survey[J]. IEEE Access, 2019, 7 : 63373 – 63394.

[40] ATREY P K, HOSSAIN M A, EL SADDIK A, et al. Multimodal fusion for multimedia analysis: a survey[J]. Multimedia systems, 2010, 16(6) : 345 – 379.

[41] HINTON G E. Boltzmann machine[J]. Scholarpedia, 2007, 2(5) : 1668.

[42] 刘建伟, 刘媛, 罗雄麟. 玻尔兹曼机研究进展 [J]. 计算机研究与发展, 2014, 51(1) : 16.

[43] ZHANG N, DING S, ZHANG J, et al. An overview on restricted Boltzmann machines[J]. Neurocomputing, 2018, 275 : 1186 – 1199.

[44] SALAKHUTDINOV R, LAROCHELLE H. Efficient learning of deep Boltzmann machines[C] // Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010 : 693 – 700.

[45] ESLAMI S, HEESS N, WILLIAMS C K, et al. The shape boltzmann machine: a strong model of object shape[J]. International Journal of Computer Vision, 2014, 107(2): 155–176.

[46] BALTRUAITIS T, AHUJA C, MORENCY L P. Multimodal Machine Learning: A Survey and Taxonomy[J]. IEEE, 2019.

[47] 王功明, 乔俊飞, 关丽娜, et al. 深度信念网络研究现状与展望 [J]. 自动化学报, 2021, 47(1): 15.

[48] NOBLE W S. What is a support vector machine?[J]. Nature biotechnology, 2006, 24(12): 1565–1567.

[49] INC F. React –A JavaScript library for building user interfaces[J], 2015.