南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论文题目	基于测试报告融合的众包审核交付系统的设计与实现
作者姓名	梅 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	刘嘉　副教授

2019 年 5 月 27 日

学　　号：　**MF1732099**

论文答辩日期：　**2019** 年　**5** 月 **13** 日

指　导　教　师：　　（签字）

基于测试报告融合的众包审核交付系统的设计与实现

作　　者：　　梅杰

指导教师：　　刘嘉　副教授

南京大学研究生毕业论文

（申请工程硕士学位）

南京大学软件学院

2019 年 05 月

The Design and Implementation of Crowdsourced Feedback Review and Delivery System Based on Test Report Summary

Mei, Jie

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

Supervised by

Associate Professor **Liu, Jia**

Software Institute

NANJING UNIVERSITY

Nanjing, China

May, 2019

摘 要

众包测试是指雇佣大量线上众包工人在一个短周期内对测试目标进行测试和反馈。因为可以提供丰富的测试条件、真实用户场景和较低人力成本,近年来众包测试已逐渐成为一种流行的软件测试方法。不同于传统测试，众包测试人员的特点是数量多，大多为非专业的测试人员。该特点导致所提交的测试报告呈现出数量多、重复率高和质量低的问题。审核人员要在较短周期内审查这些大量且重复低质量的测试报告，并及时交付反馈的问题，无疑面临较大困难。为改善业务流程，降低报告审核难度，众测平台亟需一个针对众包测试特点的报告审核交付系统来协助工作。

本文依托于南京慕测信息科技有限公司的众包测试平台，设计实现了基于测试报告融合的众包审核交付系统。本系统创新地引入基于融合的报告处理机制，来解决传统众包审核中的问题。首先，系统基于测试报告的文本和截图的特征进行聚类，从而将相似报告聚合到同一类簇中。其次，基于 PageRank 算法，系统分别计算每个类簇中报告的权重，取最高权重者为该类簇的主报告。然后，在每个类簇中，系统将报告拆分成句子和截图，分别与主报告对比，找出差异点。最后，系统对差异点进行聚类，每个类簇作为主报告的一个补充点。通过对主报告和补充点的识别和组织，报告信息变得更加丰富和有层次，报告质量得到提升。

本系统划分为审核交付模块、文本处理模块、图片处理模块、聚合模块和融合模块。为方便系统水平扩展，本系统充分利用负载均衡和进程外缓存 Redis 实现服务端无状态化。为实现系统松耦合，本系统使用事件引擎 EventBus 进行异步化和事件化设计。为保证数据可靠性，系统采用主从同步来实现数据实时备份。

目前该系统已在慕测公司的众包测试平台上线，用于支撑公司众包业务的发展。该系统有效聚合了相似报告，并对冗余信息进行合理地组织和展示，从而使审核人员更加快速精准地识别有效信息。最终，该系统提高了报告审核效率，保证了报告的交付质量，帮助平台更好地将测试产物交付给任务发布方。

关键词：众包测试，测试报告聚合，测试报告融合，测试报告管理

Abstract

Crowdsourced testing involves hiring a large number of online crowd workers to test application and give feedback in a short period of time. Over the past few years, crowdsourced testing has gradually become a popular method for software testing as it brings abundant test conditions and real user scenarios, which also has a relatively low labor cost. However, unlike the traditional testing, crowd workers are consisted of a large amount of people, mostly non-professional testers, which leads to large quantities, high repetition rates and low quality of the test reports. It is difficult for the reviewers to inspect numberous and repetitive test reports in a short period of time and deliver feedback in a timely manner. In order to improve business processes and reduce the difficulty of the review, the crowdsourced test platform requires a report review and delivery system, which is specified for crowdsourced testing features to assist the work.

By utilizing the crowdsourced test platform, this thesis has designed and implemented a crowdsourced review and delivery system based on the test report summary. The system has introduced a novel mechanism, which is based on reporting summarization to solve the problems of the crowdsourced feedback review. First, the system clustered reports based on the features of text and screenshots, thereby aggregating similar reports into the same cluster. Second, based on the PageRank algorithm, the system weighted reports from each cluster, and then the one which had the highest weighted value from each cluster was chosed to be main report of the cluster. Third, the system splited reports into sentences and screenshots from each cluster, which were then compared with the main report to find the different points. Last, the system clustered the different points. Each cluster served as a supplementary point to the main report. Through the identification and organization of the main report and supplementary points, the report information is enriched and more hierarchical, so that the quality of reports is improved.

The system was divided into review and delivery module, text process module, image process module, aggregation module and summary module. The system has made full use of load balancing, out-of-process cache Redis, etc., to achieve server statelessness, which has allowed it to expand horizontally. Moreover, making full use of asynchronous and

event-based design has improved system loose coupling. Master-slave synchronization has been used to ensure data reliability.

Currently, this system has been launched on the testing platform to support the crowdsourcing business of the company. The system has effectively aggregated similar reports and reasonably organized redundant information, which enables the reviewers to identify the helpful information more quickly and accurately. Finally, the system has improved the efficiency of the report review, guaranteed the quality of the report delivery and provided a better delivery of the test product to the task issuer.

Keywords：Crowdsourced Testing, Test Report Aggregation, Test Report Summary, Test Report Management

目 录

图目录

表目录

第一章 引言

1.1 项目背景与意义

众包测试指雇佣大量线上众包工人在一个短周期内对测试目标进行测试和反馈。近年来，众包测试已逐渐成为一种流行的程序测试方法，得到众多商业测试平台的采纳，例如 uTest[1]、Testin[2]、百度众测[3]、阿里众测[4]、testIO[5]等。众包测试具有很多的优点，首先，具备丰富的测试条件，例如多种多样的设备型号、操作系统、网络环境等。其次，基于用户真实的使用场景，开发者可以得到软件产品最有效的反馈。再次，利用群体智慧探索多种多样的测试路径。最后，能够在花费相对较低的人力成本的同时完成主要的测试任务。

图 1.1 展示了工业界众包测试的主要流程。主要参与者包括任务发起者、众包工人和众包平台[Mao et al., 2017]。任务发起者（通常是软件开发方）会在众包平台上传待测软件和布置测试任务。众包工人会选择自己擅长的或分配好的任务去完成，并在规定时间内提交测试报告，以反馈自己测试过程中发现的问题。最终，会由平台审核人员审核并整理测试报告交付给任务发起方。

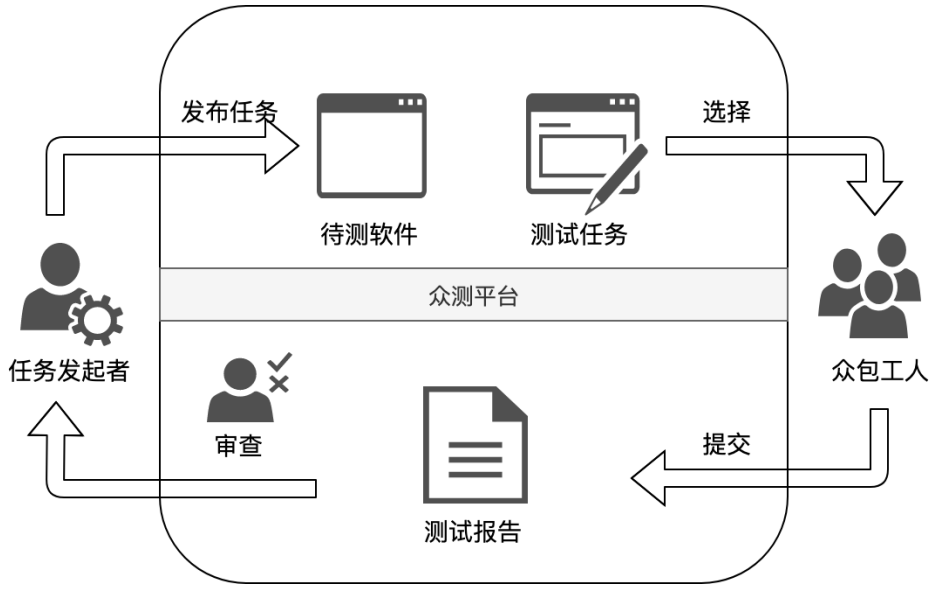

图 1.1 众包测试流程

[1] https://www.utest.com
[2] http://www.testin.io
[3] http://test.baidu.com
[4] https://mqc.aliyun.com/crowdtest/index.htm
[5] https://test.io/

不同于传统测试，众包测试人员有数量多，大多为非专业测试人员的特点，同时，众包机制大多为竞争式的模式，测试人员提交的有效 Bug 数将直接与奖励挂钩，彼此之间不互通信息。因此测试人员通常以多提交报告为目标，而对质量和重复率的关注度则较低。基于上述情况，众包测试报告呈现数量大、重复率高、质量相对较低的问题。在较短周期内，审核人员要审查这些大量且重复低效的报告，并及时交付测试人员反馈的问题，无疑面临较大困难。

在实践中，现有众包测试平台也面临同样的问题。平台的众包测试任务往往产生数量巨大的 Bug 报告，由于审核人力有限，经常只能交付全部的原始报告。由于报告重复率高，质量低，导致客户满意度差，严重影响公司业务发展。目前市面上又缺乏专门针对众包测试报告进行审核和交付的系统，因此公司更倾向于选择自己实现一套带有完整报告审核交付流程的系统。

本系统面向众包测试报告审核交付业务，针对众包测试报告数量大、重复率高和质量低的问题，引入基于融合的报告处理机制，构建了围绕报告融合的众包审核交付系统。该系统对相似报告进行有效聚合，并在此基础上，对相似报告的冗余信息进行合理地组织和展示，从而使审核人员能够集中处理相似报告，更加快速精准地识别 Bug。最终，本系统为企业降低人力成本和提高经济效益作出贡献。与此同时，对相似报告的检测和融合，至今处于实验室研究论证的阶段，工业界缺乏通用解决方案，本系统也作为相似报告检测和融合的研究在工业界的一次实践，为相关科研工作提供验证和参考。

1.2 国内外研究现状及分析

1.2.1 缺陷管理系统分析

国内外虽然有较多众包测试平台，但是市面上缺乏专门针对众包测试报告审核和交付的相关系统，原因是此类系统较为依赖于众包平台的业务属性，不易通用化，通常只能作为平台内部的子系统来使用。于是，本文针对国内外较为知名的一些缺陷管理追踪系统进行了调研，如 Atlassian 公司的 JIRA，开源的 Bugzilla 等。主要关注这些系统在对相似报告上的处理方法，如何提高测试报告审核和交付的效率。

JIRA 和 Bugzilla 通过第三方插件或自带的功能，在用户创建 Bug 报告时，基于关键字检索来发现是否有重复报告，并让用户选择是否过滤筛选。该处理方式在面对众包测试时存在两个问题，问题一是众包测试人员独立测试，彼此不互通消息。目前主流的众包测试平台多采用竞争式的众包机制，以用户找到的有效 Bug 数作为考核标准，为防止互相抄袭，测试人员各自独立地进行测试任务，因此测试人员通常没有权利直接查看他人的报告来进行去重。问题二是没有解决报告质量低的问题。众测报告不同于专业测试报告，其质量相对较低，不易理解，因此仅通过去重无法解决众测报告质量低的问题。

综上所述，JIRA 和 Bugzilla 的处理方式不适合于众包测试报告。Bettenburg 等人发现额外信息有助于审核人员对软件缺陷的理解[Bettenburg et al., 2008]。众测报告有重复率高的特点，通过利用多份相似报告的冗余信息来对信息进行增强，可以有效提高报告质量。为适应众包测试的特点，本系统借鉴了以聚合和融合相似报告的方式来提升审核人员审核和交付 Bug 的效率[Hao et al., 2019]。

1.2.2 相似报告检测研究现状

目前科研人员已经提出了许多方法来检测相似错误报告。比如基于纯粹的自然语言处理技术[Runeson et al., 2007]、机器学习技术（分类模型[Lazar et al., 2014]）、信息检索技术（n-gram 模型[Sureka et al., 2010]、程序执行信息[Wang et al., 2008]）等。Runeson 通过使用错误报告的自然语言信息提出一种相似报告检测方法[Runeson et al., 2007]。Klein 等人[Klein et al., 2014]提出基于主题分布的方法。Nguyen 等[Nguyen et al., 2012]利用基于术语和基于主题的特征进行重复检测。Rocha 等人[Rocha et al., 2016]采用错误组件和错误描述的相似性来检测重复报告。Yang 等人[Yang et al., 2016]采用信息检索和词嵌入技术，计算时，除文本描述外还考虑了产品和组件信息。Banerjee 等人[Banerjee et al., 2013]提出融合的分类方法来检测重复报告。Sun 等人[Sun et al., 2010]利用判别模型和 BM25F 相似性测量来提高检测性能，并在此基础上提出 REP 模型[Sun et al., 2011]。Tian 等人[Tian et al., 2012]基于聚类算法和 REP 模型改进报告相似度计算。Hindle 和 Alipour 等人[Alipour et al., 2013]合作利用上下文信息和信

息检索技术来提高重复报告准确率，并在此基础上进行了改进[Hindle et al., 2016]。

在相似 Crash 报告检测的领域，Dang 等人[Dang et al., 2012]考虑了一种名为 ReBucket 的新型 bucketing 方法，它使用 Position Dependent Model（PDM）基于异常的堆栈信息来聚类崩溃报告。Jiang 等人[Jiang et al., 2018]将聚类技术应用于众包测试报告，他们提出 TERFUR 方法来将冗余测试报告进行聚类，以减少需要手动检查的报告数量。Jiang 等人的想法在设计上给本系统一定的启发。

上述研究从各个技术角度对检测重复报告进行了尝试，不过都主要集中于对报告文本信息的处理，有少量研究在计算测试报告相似度时考虑了随报告上传的图片。Feng 等人[Feng et al., 2016]提出对报告的文本信息和图片信息分别建立相似度矩阵，最后按照“平衡公式”合并两个矩阵，生成最终的报告之间的相似度矩阵。Hao 等人[Hao et al., 2019]也采用这种方法来检测重复报告。Wang 等人[Wang et al., 2018]则采用不同思路，他们首先对截图和文本分别提取特征，截图特征采用颜色和结构，而文本特征采用 TF-IDF 和词向量。基于上述特征分别计算报告的图片相似度和文本相似度，在合并两种相似度时，没有采用“平衡公式”，而是先判断图片相似度是否大于设定的阈值，如果大于，则按照文本相似度进行排序；如果小于则按照两种相似度的平均值进行排序。

公司目前以移动应用众包测试为主，这类报告基本都包含屏幕截图，结合公司的业务数据情况，仅仅依靠文本信息进行报告处理是远远不够的，因此本系统采用文本描述与屏幕截图相结合的方式来计算测试报告的相似度。

1.2.3 测试报告摘要研究现状

摘要技术主要被分为两类：抽取式[Erkan et al., 2004]和生成式。对于 Bug 报告来说，最主要的摘要技术是基于句子级别的抽取模型，即通过抽取中心句并进行一定比例的压缩。近年来，为从 Bug 报告中提取有价值的信息，科研人员已经提出一些测试报告摘要技术。Ferreira 等人[Ferreira et al., 2016]提出基于评论的摘要方法。Jiang 等人[Jiang et al., 2017a]通过挖掘写作特点发现写作风格一致性和报告质量之间的关系，并据此提出一个新的摘要方法名叫 Authorship Characteristics based Summarization (ACS)，除此之外，他们还提出一种基于

PageRank 的摘要技术（PRST）[Jiang et al., 2017b]，它利用主报告中的信息和相关的重复项来总结主报告。Lotufo 等人提出一种无监督的 Bug 报告摘要方法，该方法评估用户在阅读过程中可能会给予哪些句子的更高注意力，以此来评判句子重要性[Lotufo et al., 2012]。Yeasmin 等人采用 Lotufo 的模型来构建了一个可视化的摘要模型[Yeasmin et al., 2014]，之后对该模型的算法进行改进，并进行详细的评估和解释[Yeasmin et al., 2015]。

上述方法主要集中于研究对单份测试报告的自动摘要，Hao 等人提出对多份相似报告进行摘要的方法[Hao et al., 2019]，她提出的方法是利用 PageRank 算法查找出多份报告中的主报告，然后将其他报告拆分成句子和图片，与主报告依次对比，查找出与主报告差异度较大的内容作为“补充点”。本系统在多报告融合阶段将会采用该方法。

1.3 本文主要工作内容

本文所设计和实现的众包测试报告审核交付系统是众包测试平台的一个子系统，其主要目标是针对众包测试报告数量多、重复率高和质量低的特点，帮助审核人员解决审核工作量大、重复工作多和报告阅读困难的问题。

本系统考虑通过以下两个方面的设计来实现该目标：

一方面集中于如何解决与利用冗余信息，实现报告的自动化处理和关键信息的提取。审核人员在审核报告时常常受到报告数量大和冗余度高的困扰。结合查阅的相关资料和业务现状，本系统没有采用传统的过滤重复报告的方案，而是引入基于融合的报告处理机制来自动化处理测试报告[Hao et al., 2019]，该机制包括聚合和融合两个阶段。聚合阶段的功能主要集中于相似报告的识别和聚合，这一阶段的难点在于文本、图片各自特征如何选择，文本和图片信息如何结合。融合阶段的功能则集中于有用信息的提取，这一阶段的难点在于融合机制的选择，有用信息的识别和组织。该机制是为了针对众包测试报告的特点，通过聚合相似报告，让相似报告得到集中处理，同时融合冗余信息，解决单份报告描述不足、质量较低的问题。

另一方面是围绕报告融合设计审核交付的业务流程，方便审核人员在聚合报告的基础上开展审核工作。本系统提供给用户众包任务查询视图、原始报告管理

视图、聚合报告管理视图、审核进度提示、聚合报告展示页面和预交付报告管理视图。聚合报告展示页面需要将融合阶段提取的信息进行合理的展现，能够突出主题，对报告的聚合关系有可视化处理，对有用的辅助信息进行重点突出。

结合上述需求和方案，本系统基于 Spring Boot 框架开发，作为服务独立部署，与其他服务实现松耦合。系统使用 HTTP 协议与其他服务进行数据交换，优势是可以快速应对数据格式的变化，简单快捷。为保障数据可靠性，系统使用稳定的数据存储软件，并进行主从同步，以实现数据实时备份。为实现系统水平扩展，使用负载均衡和缓存中间件实现服务端无状态化。为提高松耦合特性，系统采用 EventBus 对系统进行异步化事件化设计。

对于报告自动化处理阶段，本系统将分解为聚合和融合两个阶段。聚合阶段中，系统会提取报告的文本和图片的特征，然后使用凝聚式层次聚类算法来对相似报告进行聚类。融合阶段中，首先，系统采用 PageRank 算法给每个类簇的报告计算权重，并选取具有最大权重的作为该类簇的主报告。其次，系统将每个类簇的其他报告拆分为句子和截图，并分别与主报告进行对比，筛选出差异点。然后，考虑到差异点之间存在重复，系统对每个类簇的差异点进行聚类，每个类簇作为主报告的一个补充点。最后，通过对主报告和补充点的合理组织，形成聚合报告，提高了报告的可读性，便于审核人员快速精准地发现有效信息。

1.4 本文的组织结构

本文的组织结构如下：

第一章 引言部分。本章概述众包测试的背景和众包测试报告审核过程中面临的问题，国内外工业系统现状，国内外相似测试报告检测和摘要领域的研究现状，以及本文在解决该问题上的主要工作思路。

第二章 技术综述。本章概述项目中使用的 Word2Vec、相似度计算方法、凝聚式层次聚类算法、PageRank 算法、可视化组件 D3、缓存中间件 Redis、事件发布订阅组件 EventBus。

第三章 系统需求分析与设计。本章描述了本文的提出背景和开发目标，分析和总结出本系统的功能性和非功能性需求，并描述了系统总体设计思路和框架，对项目模块进行划分，并分别详述文本处理模块、图片处理模块、聚合模块、融

合模块和审核交付模块的架构设计、流程设计、核心类图和数据库设计等。

第四章 系统的具体实现。在第三章设计的基础上，本章对文本处理模块、图片处理模块、聚合模块、融合模块、审核交付模块的实现细节进行详细分析，展示调用顺序图和关键代码，以及系统的主要界面。

第五章 系统测试。本章主要依据第三章需求分析的产品规格，对系统进行功能测试、水平扩展性测试和数据库主备切换测试。具体包括测试环境说明、测试用例设计和测试结果分析与评价。

第六章 总结与展望。首先，本章总结众包审核工作中的问题和本文的主要解决方案及创新之处。其次，本章总结了本文在系统开发前后的主要工作，包括论文综述、需求分析、工程设计与实现和系统测试等。最后，本章对系统可后续改进的功能作了进一步分析。

第二章 相关技术概述

本章对系统中采用的相关技术进行简要概述。系统在自动化处理过程中需要基于词向量来提取文本特征，利用余弦相似度计算相似度，利用凝聚式层次聚类算法聚合相似报告，并基于 PageRank 完成融合算法。在展示聚合关系时，系统需要 D3 组件来完成。系统基于 Redis 和 EventBus 来构建高效和可靠的系统。

2.1 Word2Vec 技术简介

本系统在报告处理过程中，需要计算两段短文本之间的相似性。要判断文本的相似度，首先要提取文本的特征值，然后根据特征向量计算二者的相似度。因此如何表示文本的特征向量将极大影响到文本相似度的计算。由于 Word2Vec 训练的词向量维度低，资源消耗低，同时考虑了词的语义，精确度更高，因此本系统选择 Word2Vec 技术来训练词向量，计算文本相似性。

2.1.1 词的表示方法

在自然语言处理中词的表示方法主要有两种。

第一种是 one-hot representation。这种方法的思路是将所有词去重以后进行编号，排成一列较长的向量，每个词占据该向量的一个位置。当表示一个词时，该向量只有这个词的位置是 1，其他位置均设置为 0。举例如下，

“按钮”表示为 [0,0,0,1,0,0,…,0]

“点击”表示为 [0,1,0,0,0,0,…,0]

这种方法优点是简洁、直观。缺点是向量维度会随着句子不同词数的增加而增加，并且任意两个词之间彼此独立，缺乏语义层面的信息。

第二种是 distributed representation。为将语义融入到词表示中，相关学者提出的分布假说为这一设想提供了理论基础：上下文相似的词，其语义也相似。这种方法的思路是利用低维度空间的密集向量来表示词，同时利用模型将语料中相似的词映射到相似的坐标。词向量，又被称作词嵌入，就是基于该思想来把语料中的词映射到低维度空间中的方法。

2.1.2 Word2Vec 主要流程

词向量模型本质是对词表示方法进行降维的一种思想，具体实现方法可以有多种。考虑到性能和存储开销，目前最常用的是神经网络模型，该领域有两种最常见的模型，基于 CBOW 模型和基于 Skip-Gram 模型[Mikolov et al., 2013]。两种模型都由 Mikolov 等人提出，训练方法上也基本相近。两种模型的细微差别是前者的输入是一个词的上下文关系词，通过上下文来进行预测，而后者正好与前者相反，输入是一个词的词向量，输出是上下文词的词向量。Word2Vec 是用来训练词嵌入或者低维向量表示的工具。

Word2Vec 主要流程如下。

首先，分词、词干提取和词形还原。中文里需要分词，英文则要考虑时态还原，本系统主要面向中文分词。

其次，构造词典，统计词频。这一步需要遍历所有语料的词，统计出各个词的词频。

第三，构造 Huffman 树，生成 Huffman 编码[Huffman, 1952]。依照出现概率构造生成 Huffman 树，并生成 Huffman 编码。

第四，初始化 Huffman 树各节点的向量。向量的长度由训练方给定，通常叶结点的向量是随机生成是，而非叶节点的向量都为 0。叶节点代表最终的词向量，非叶结点存储的向量值相当于神经网络模型中隐藏层的参数。

最后，训练中间向量和词向量。这步基于改进的 CBOW 模型或 Skip-Gram 模型，读取语料库中的词，通过梯度下降不停迭代调整中间向量，并得到最终的词向量。

2.2 相似度计算方法

本系统在报告处理过程中，需要依据文本和图片特征来计算相似度。由于余弦相似性在高低维度的计算结果都更加稳定[Huang, 2008]，因此本系统选择它作为特征相似度计算方法。同时，本系统在融合模块中，需要计算类簇之间相同 Bug 数占总 Bug 数的比例，以此决定是否进行差异点类簇合并。Jaccard 相似性是用来度量集合相似度最常用的方法，因此本系统选择它计算差异点类簇相似性。

2.2.1 余弦相似性

余弦相似性使用余弦公式来对两个向量夹角的余弦值进行计算，该结果用来对它们之间的相似性进行度量。当两个向量的夹角越小时，说明它们越接近，可以理解为越相似，极端情况下，当向量间夹角为零时，两个向量重合，说明它们重合度达到最高，这时余弦值为 1。反之，当两个向量完全不相似时，它们的余弦值为 0。

给定两个向量 A, B，其余弦公式如公式(1)所示：

$$\cos\theta = \frac{A \cdot \mathrm{B}}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \quad (1)$$

其中，A_i 和 B_i 分别代表向量 A 和 B 的各个分量。

2.2.2 Jaccard 相似性

Jaccard 相似性可以理解为计算两个集合的相似性，定义为两个集合元素的交集和并集的比值。如公式(2)所示：

$$J(A,B) = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A} \cup \mathrm{B}|} = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A}| + |\mathrm{B}| - |\mathrm{A} \cap \mathrm{B}|} \quad (2)$$

其中，A 和 B 表示任意两个集合，且$0 \le J(A,B) \le 1$。

2.3 凝聚式层次聚类技术简介

本系统在聚合模块中，需要对相似报告进行聚类。层次聚类算法的一大优点就是不用特意指定聚类数目，同时距离和规则的相似度容易定义，限制少，可以发现类的层次关系。这些特点很适合用于相似报告的聚类。因为凝聚式层次聚类算法相较于分裂式更加稳定，因此本系统采用它作为聚类算法。

2.3.1 层次聚类

层次聚类算法主要有两种，可以分为凝聚（agglomerative）式和分裂（divsive）式，取决于层次分解的方向。分裂的层次聚类使用自顶向下的策略，一开始所有的元素放置于同一个 group，然后将根划分为多个小 group，递归划分直到每个

group 中只有一个元素为止。凝聚的层次聚类使用自底向上的策略，一开始每个元素都是一个 group，通过迭代每次将距离最近的一对 group 进行合并，直到所有元素都在同一个 group 为止[Patel et al., 2015]。分裂式层次聚类不容易判断如何在整个集合进行划分，较为复杂，因此使用较少，凝聚式层次聚类更为常用。

2.3.2 聚类过程简述

为说明聚类过程，假定有 5 个待聚类的元素，它的主要步骤如下：

(1) 初始状态，每个元素都是一个类簇。

(2) 计算类簇之间的相似度，有多种计算方法，具体如 2.3.3 节所述。

(3) 每次迭代将相似度最高的两个类簇合并。

(4) 重复步骤(2)和(3)，直到所有元素在同一个类簇中。

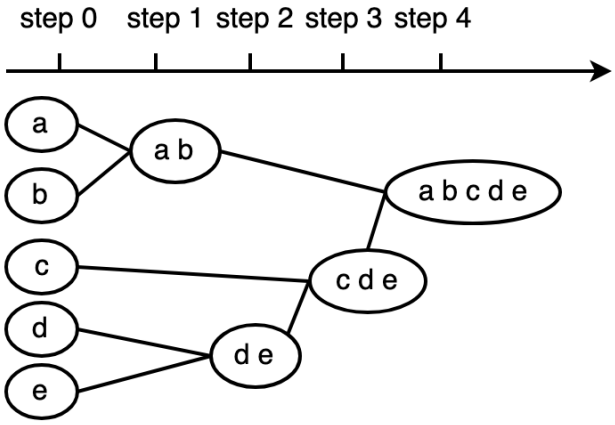

图 2.1 凝聚式层次聚类样例图

整个过程就是建立一棵树，如图 2.1 所示，在建立的结束中，以树的高度作为切割条件，可以对树进行切割，其结果就是最终的聚类结果。

2.3.3 簇间距离度量算法

在凝聚层次聚类算法运算过程中，需要度量不同聚类之间的相似度，以决定是否需要合并，在算法中该度量方法被称为 Linkage。Linkage 的实现有多种算法，较为常见的算法如下：

Single 方法：又称为单连接算法，使用不同类簇元素之间的最小距离来衡量两个类簇之间的距离。当使用该算法时，层次聚类也被称为最近邻聚类算法。具

体如公式(3)所示：

$$D(X,Y) = \min_{x\in X, y\in Y} d(x,y) \tag{3}$$

其中，X和Y是任意两个类簇的元素集合，$d(x,y)$代表元素x和y的距离。

Complete 方法：又称为全连接算法，使用不同类簇元素之间的最大距离来衡量两个类簇之间的距离。具体如公式(4)所示：

$$D(X,Y) = \max_{x\in X, y\in Y} d(x,y) \tag{4}$$

其中，X和Y是任意两个类簇的元素集合，$d(x,y)$代表元素x和y的距离。

Average 方法：使用不同类簇元素之间的平均距离来度量簇间距离，具体如公式(5)所示：

$$D(X,Y) = \frac{1}{|\mathrm{X}|\cdot|\mathrm{Y}|}\sum_{x\in X}\sum_{y\in Y} d(x,y) \tag{5}$$

其中，X和Y是任意两个类簇的元素集合，|X|和|Y|表示两个类簇的元素个数，$d(x,y)$代表元素x和y的距离。

Ward 方法：使用两个类簇合并时导致的平方误差的增量来度量两个类簇之间的距离。计算两个类簇合并后的平方和公式如公式(6)所示：

$$TD_{C1\cup C2} = \sum_{x\in C1\cup C2} D(x,\mu_{C1\cup C2})^2 \tag{6}$$

其中，$\mu_{C1\cup C2}$表示合并后新类簇的中心点，$D(x,\mu_{C1\cup C2})$表示合并后任一元素到中心点的距离。最终距离公式如公式(7)所示：

$$D(C1,C2) = TD_{C1\cup C2} - (TD_{C1}+TD_{C2}) \tag{7}$$

2.4 PageRank 算法简介

本系统在融合模块中，使用 PageRank 算法计算报告之间权重，选择最大权重者作为每个类簇的主报告。由于 PageRank 算法是依据链接关系来计算权重，因此最大权重者意味着在网络中得到的认可度最高，也最有代表性。而主报告也是每个类簇中最有代表性的一份报告，因此该算法适用于本系统对主报告的选取。

2.4.1 算法概述

PageRank 算法[Page et al., 1999]，是依据页面之间的超链接关系计算网页权重的技术，本质上是一种基于有向图关系来计算结点权重的方法。

PageRank 的最终目的是给每一个页面计算一个权重。其主要思想是，分析结点之间链入链出的关系来计算出各个结点的权重，简单来说，可以理解为结点之间的相互投票行为，到达一个结点的链接就相当于给该结点投了一票。那么依据这个逻辑，得到票数较多的结点，说明它重要等级就要更高，因此具有较高的权重，而对于没有任何链入的结点，则具有较低的权重。

该算法有两个前提假设。

第一个是一个结点如果被链入数量越多，那么说明这个页面越重要。

第二个是结点的权重是不同的，指向结点 N 的入链质量也就不同，权重高的结点会通过链出边向其他结点分配更多的权重。所以如果结点 N 被越多的权重高的页面链入，那么结点 N 越重要。

综上，PageRank 算法初始阶段会平均分配每个结点相同的权重，经过多次迭代计算来不停刷新各个结点的权重，直至两次计算之间的结果差收敛。

2.4.2 过程简述

PageRank 算法的基本步骤如下：

初始阶段：根据链接关系构建有向图，所有结点平均分配 PageRank（以下简称 PR）值。

计算过程：在每一轮计算中，各个结点把它当前的 PR 值平均分配到本结点所有的链出边上，然后每个结点将所有链入边的 PR 值求和，结果就是本结点最新的权重。

结束条件：按照步骤二不停迭代直到收敛，收敛的标准就是两次计算的 PR 值之差是一个非常小的值。

下面以一个简单例子进行描述：

假设世界上存在四个页面 A、B、C 和 D，且 B、C、D 都链向 A，那么 A 的 PR 值是 B、C、D 的 PR 值之和。

$$\mathrm{PR}(A) = PR(B) + PR(C) + PR(D) \tag{8}$$

现在重新假设 B 页面有两个链出页面，分别是 A 和 C，假设 D 有 3 个链出页面分别是 A、B、C。那么 B 将自己的 PR 值平分为两份，其中一份给 A，同理，D 的 PR 值得$\frac{1}{3}$给 A，那么 A 的 PR 值计算如下：

$$\mathrm{PR}(A) = \frac{PR(B)}{2} + PR(C) + \frac{PR(D)}{3} \tag{9}$$

下面给出该公式的通用形式，考虑到没有任何出链的页面并经过数学统计，最后将给每个页面一个最小值$(1-d)/N$，通常 d 的值设置为 0.85。公式修正后如公式(10)所示：

$$\mathrm{PR}(p_i) = \frac{1-d}{N} + d \times \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \tag{10}$$

其中，p_1，p_2，p_3 …，p_N指被研究的页面，$M(p_i)$指链入 p_i 页面的集合，$L(p_j)$指 p_j 链出页面的数目，N指所有页面数。

2.5 可视化工具 D3

2.5.1 D3 简述

本系统使用 D3 来对报告之间的聚合关系进行可视化。D3 的全称是 Data-Driven Documents，即数据驱动的文档[Bostock et al., 2011]。D3 基于 JavaScript 编写，可以直接将获取的数据绑定到 DOM 上，并能根据数据变化驱动页面展示的变化。D3 完全基于 HTML、SVG 和 CSS 来对图形进行绘制，没有引入其他视觉表示方法。D3 对动画效果的支持也非常强大，可以自定义图形变换的过渡效果，并且对现代主流浏览器的兼容性较好。

D3 在实现数据可视化通常要经历几个步骤：首先，把数据加载到浏览器的内存空间；其次，根据需要创建新元素，或者直接将数据绑定到 DOM 元素；然后，解析每个元素的边界范围，并设置相应可视化属性，实现元素的变换（transforming）；最后，响应用户输入，实现元素状态的过渡。

2.5.2 使用 D3 的优势

现在有很多开源的图表库 Echarts、HighCharts、G2.js 等，D3 相比于这些组件的优势如下：

第一，基于 svg。D3 生成的图像是 svg 图形，svg 称为可缩放矢量图形，用户对图像的放大不会导致失真。

第二，数据转换与绘制独立。一般图形库直接提供一个函数让用户直接输入数据绘制图形，但 D3 提供函数转换用户提供的数据，并由用户决定如何绘制图形。

第三，代码简洁。D3 的代码采用 JQuery 的设计，采用大量链式语法，增加代码可读性和简洁度。

综上，本系统使用 D3 作为可视化工具。

2.6 缓存组件 Redis

2.6.1 Redis 简述

Redis 是目前主流的缓存中间件之一，本系统使用 Redis 来缓存用户会话和相关业务数据。Redis 是 Remote Dictionary Server 的缩写，是一个基于内存的数据存储服务[Carlson, 2012]。Redis 支持非常丰富的数据类型，包括 String、Hash、List、Set、ZSet、hyperloglogs[Heule et al., 2013]等数据类型。基于这些数据类型，可以直接实现高速缓存和消息队列。Redis 是单进程单线程架构，所有指令都是串行执行，不需要传统数据里的串行控制，性能因此得到提升。Redis 还提供了分布式集群和高可用机制的实现，便于系统的水平扩展。

Redis 支持两种数据持久化机制，第一种是快照，第二种是 AOF 日志。快照是一次全量备份，AOF 日志是将所有指令都缓存在日志中，通过指令重放来恢复数据。快照存储较为紧凑，是所有数据的二进制备份，缺点是可能会丢失最新的的数据；而 AOF 日志能保存最新的数据，缺点是文件会逐渐膨胀。为解决两种备份方式的问题，通常会将二者结合起来，AOF 日志只记录最近一次快照后新增的数据，这样 AOF 文件的大小得到控制。

2.6.2 Redis 数据结构

表 2.1 Redis 数据结构

数据类型	使用数据结构	特性	应用场景
String	动态字符串	可以包含任何数据，比如 jpg 图片或者序列化的对象，一个键最大能存储 512M	缓存字符串数据、计数、位图等
List	ziplist、quicklist	增删快，提供操作某一段元素的 API	最新消息排行等功能、消息队列等
Set	哈希表	添加、删除、查找的复杂度都是 O(1)，为集合提供求交集、并集、差集等操作	计算共同好友、利用唯一性，统计访问网站的所有独立 IP 等、点赞统计
Hash	哈希表	适合存储对象，并且可以像数据库中 update 一个属性一样只修改某一项属性值	存储、读取、修改用户属性
ZSet	skiplist	数据插入集合时，已经进行天然排序	积分排行榜

如表 2.1 所示是 Redis 的五种基本数据类型。Redis 区别于 Memcached 等缓存中间件的一大特点就是拥有丰富的数据类型。Redis 支持 String、List、Hashe、Set 及 Ordered Sets 数据类型操作，基于这些丰富的数据类型，系统可以方便地实现对在线用户数、排行榜、关注关系、点赞关系的缓存。

2.7 事件引擎 EventBus

2.7.1 EventBus 简述

EventBus 是目前主流的提供进程内发布订阅[Eugster et al., 2003]服务的组件之一,本系统使用 EventBus 来作为系统内部的事件引擎。EventBus 是一个使用发布-订阅模式构建并且低耦合的 Java 开源库。EventBus 通过简洁的代码即可实现系统模块之间通信解耦。通过该框架，系统可以实现简洁的发布订阅模式，驱动系统的事件化和异步化。

EventBus 有如下三个要素：

(1) Event：事件。

(2) Subscriber：事件订阅者，接收特定的事件。

(3) Publisher：事件发布者，生成 Subscriber 关心的事件。

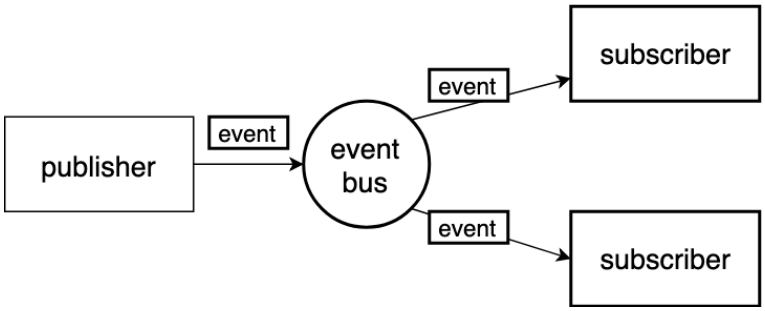

图 2.2 EventBus 原理图

如图 2.2 所示，Subscriber 首先注册到 EventBus 上，Publisher 则是将事件推到 EventBus 上，当 EventBus 收到消息时，会依据事件类型查找相关的订阅者集合，并回调订阅者的方法。

2.7.2 使用 EventBus 的优势

简化组件间的通信：（1）对发送和接受事件解耦；（2）可以在主线程和后台线程间执行；（3）简化依赖关系，降低系统复杂度。

基于注解，编写便捷：通过注解直接在订阅方法上声明即可，避免了显式的注册和订阅。

事件订阅者继承：事件设计遵循面向对象设计原则，如 A 事件是 B 事件的父类，当事件 B 发生时，关注事件 A 的订阅者，也同样会收到事件通知。

可配置：EventBus 的特性可以在创建的时候通过参数进行控制，比如采用同步还是异步的线程池，线程池的大小等。

小巧：EventBus 的 JAR 包仅仅 50KB 左右，功能强大的同时消耗极小。

综上，本系统使用 EventBus 作为系统事件引擎。

2.8 本章小结

本章主要概述开发本系统所使用的相关算法、技术和框架。首先，对提取短文本特征所使用的 Word2Vec 技术进行概述。第二，对用于计算相似度的两种方法进行概述，一种是余弦相似度，另一种是度量集合相似度的 Jaccard 相似系数。第三，对在聚合相似报告时使用的凝聚式层次聚类算法进行概述，它可以不用指定类簇的数量，这点适用于相似测试报告的聚合。第四，对 PageRank 算法进行

概述，该算法主要用于融合阶段的主报告选取。第五，概述可视化组件 D3 的特性和优势，该组件主要用于对聚合报告的聚合关系进行可视化。第六，概述缓存中间件 Redis 的特性和数据结构，主要用于提供进程外缓存服务。最后，对事件组件 EventBus 进行概述，该组件主要用于进程内多线程之间异步通信。

第三章 众包审核交付系统的需求与设计

本章将要概述众包审核交付系统的需求分析和设计。首先，本章描述系统的开发目标和总体功能。其次，本章分析系统的功能和非功能需求，围绕功能需求进行用例设计。然后，本章对系统进行总体设计和模块划分，并描述系统的 4+1 视图和测试报告自动化处理的总体设计。最后，本章对各个模块的总体设计、流程设计、核心类图和数据库设计进行详述。

3.1 系统整体概述

由于众包测试参与人员和自身机制的原因，导致平台在测试报告审核和交付时遇到工作量大、报告重复率高和质量低等困难。本系统作为众包测试平台下的子系统，针对传统众包测试报告审核过程中的问题和难点，利用报告自动化处理技术和高效方便的交互系统以降低审核人员的工作难度，帮助众包测试平台更好地将测试产物交付给任务发布方。

如图 3.1 是本系统基于报告融合的测试报告处理流程。原始报告通过特征提取、相似报告聚合和融合、人工审查整理，形成最终的交付报告，交付给任务发布方。通过聚合相似报告，审核人员可以减少重复工作，从而降低工作量。通过融合相似报告，使冗余信息按照主要点和补充点的形式进行组织和展示，报告质量将得到有效提升，从而使审核人员可以更快速精准地识别有效信息。同时，本系统从审核人员实际工作中的难点出发，对交互方式进行简单、直观、快捷的设计，尽可能使界面布局合理的同时有保证足够的信息量。最终，通过上述措施，系统显著改善了传统众包测试报告审核过程中的问题，降低人力成本，保证了交付报告质量，为公司业务发展创造价值。

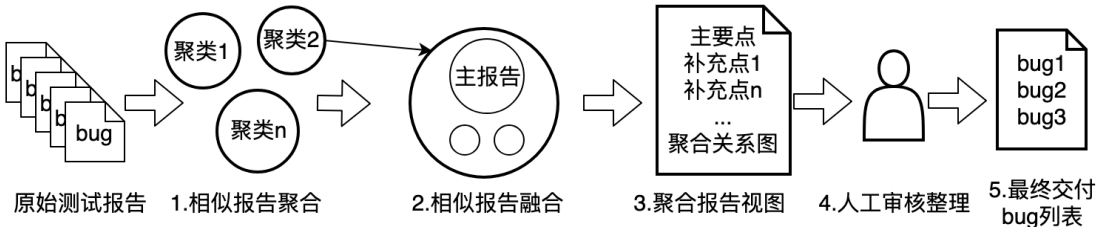

图 3.1 基于融合的测试报告处理流程

3.2 系统需求分析

3.2.1 功能性需求

功能需求按照两个部分来进行分析，分别是测试报告自动化处理和围绕报告融合的审核交付流程。主要需求如表 3.1 所示。

表 3.1 功能需求列表

需求 ID	需求名称	需求描述
R1	报告聚合	系统基于测试报告的文本和图片特征，对相似报告进行聚合，并且能够基于聚类结果，对各个类簇进行融合分析，得到主报告和差异点，使信息主次分明。
R2	查看众包任务列表	用户能查看所有众包任务，未审核完成的任务按时间顺序置顶，每个任务能看到审核进度。
R3	查看众包任务详情	用户选择具体的众包任务后，能看到该任务的报告审核进度，所有原始报告信息和触发报告聚合按钮。
R4	查看聚合报告列表	用户选择“聚合视图”，能看到所有聚合报告列表，每一项能看到内容摘要和审核状态。同时根据审核状态进行过滤，并且能返回原始报告视图。
R5	查看聚合报告详情	用户选择任一聚合报告，能进入报告详情。主要内容为主报告内容，补充点内容，补充点需要展示原始报告内容，并且对重点内容进行着色。除此以外，对该聚合报告的聚合关系以图的形式进行可视化，并且每个点被点击后能看到细节。
R6	审核报告	用户在聚合报告详情页，选择“审核”，所有原始报告状态修改为已审核。
R7	预交付报告管理	用户可以在聚合报告详情页，选择“创建预交付报告”，右侧出现报告编辑页面，拖拽图片可实现图片增删，选择保存即可创建报告。 用户选择删除指定报告即可直接删除。用户选择修改报告，右侧变为报告编辑页面。上述操作在预交付报告列表也同样都可以。

在测试报告自动化处理流程里，首先，系统需要从外部服务的数据库里获取一次任务的所有报告。接着，系统需要对报告进行一系列的预处理。对于文本内容，系统需要进行分词、清洗、加载模型等。对于图片，系统需要进行下载、入库、特征抽取和索引等，方便后续查询和相似度计算。处理完毕后，系统将对所有报告进行聚类，以找到相似报告并进行聚合，这一步系统需要考虑文本和图片的结合。在得到所有报告的聚类结果后，系统需要分别对每个聚类进行融合处理，

目标是能尽量将其信息得到一个合理的融合，识别出主要点和补充点，既能突出主要信息，又能提供辅助信息。

在审核交付流程中，审核人员首先需要能够查看所有的众包测试任务，并能看到尚未完成审核的任务处于置顶位置，以及具体的审核进度，方便审核人员快速进入审核任务。

在众包任务详情页面，首先应该能看到报告的审核进度，其次是所有的原始报告，以及能够触发报告自动化处理的按钮，原始报告列表应该支持相应的排序、检索等基础管理功能。

触发自动化处理功能结束以后，能看到“聚合视图”的按钮，进入到聚合报告列表，每个列表项应该能看到融合后的一些摘要信息，方便审核人员快速了解报告大概的内容，并且显示每个报告的审核状态。

聚合报告详情页面展示主报告和补充点内容，并且对聚合关系进行可视化。在聚合报告详情页能审核报告，一旦审核，其对应的原始报告状态也对应变为已审核。同时该页面能直接创建预交付报告，方便审核人员直接根据报告内容进行整理，提高效率。

预交付报告列表页能查看报告内容，有基本管理操作，并展示出派生该预交付报告的聚合报告 ID。

3.2.2 非功能性需求

表 3.2 非功能需求列表

可扩展性	系统应该对自动化报告处理流程内部阶段做好接口抽象，方便算法增加、升级、替换。
	保持系统服务端水平扩展的能力。
易用性	系统界面简洁美观，布局合理，容易理解，人机交互体验应该满足大部分用户的要求。
	用户在接触该系统 10 分钟以后就可以熟练使用系统。
可用性	应对数据进行备份，主库出现问题时，能及时切换备份，保证数据库访问。
	Web 服务器部分宕机时，能通过负载均衡等方法，保证其他 Web 服务器继续提供服务。

本系统的非功能需求如表 3.2 所示。为应对报告处理方法的迭代更新，本系统需要做好接口抽象，方便算法的升级和替换，增强系统的可扩展性。为应对随

着业务增长而带来的访问压力，系统需要保持一定的水平扩展能力，即在增加服务器时，能保证负载接近于线性下降。本系统要求交互简洁、便于使用，普通用户能在较短时间内熟悉系统操作流程。在系统可用性上，本系统要求数据访问和业务访问都保证在部分服务器出现问题时，其余的可以继续提供服务，使系统具备一定的灾备能力。对于数据库访问，系统应提前对数据做好备份，保证在主库宕机时，从库能继续提供服务。

3.2.3 系统用例图

根据上述功能性需求的描述，得到系统用例图如图 3.2 所示的 7 个用例，分别是查看众包任务列表、查看众包任务详情、触发报告聚合、查看聚合报告列表、审核报告、查看聚合报告详情、管理预交付报告。

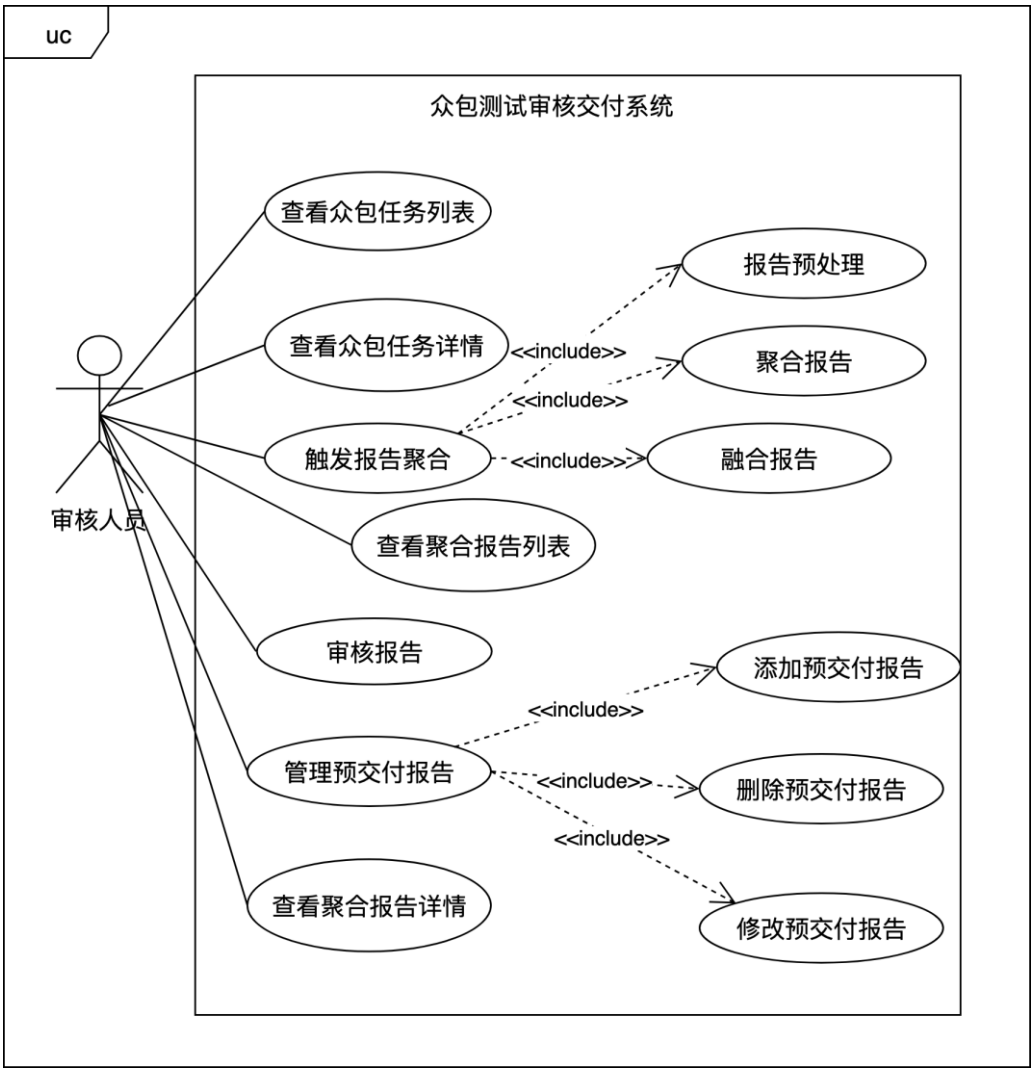

图 3.2 用例图

3.2.4 系统用例描述

查看众包任务列表是系统的入口，审核人员通过该功能快速发现和进入未完成的审核任务，因此在设计上要使未完成审核的任务尽可能放置在显眼位置，同时有进度条帮助审核人员更加直观了解审核进度。用例描述如表 3.3 所示。

表 3.3 查看众包任务列表用例描述

ID	UC1	名称	查看众包任务列表
参与者	审核人员，目的查看有哪些待审核的众包任务		
触发条件	有新的众包任务结束		
前置条件	审核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. 审核人员登录完成； 2. 系统显示所有众包任务，并且把审核中的任务按照时间升序置顶显示； 3. 审核人员查看进度条； 4. 系统显示该任务已审核报告数和未审核报告数，并且显示百分比；		
特殊需求	进度条已审核和未审核数量的颜色不同。		

查看众包任务详情主要展示众包测试报告的审核进度和原始报告的基本信息，同时能对原始报告列表进行一定的排序和检索，具备基本的管理功能。用例描述如表 3.4 所示。

表 3.4 查看众包任务详情用例描述

ID	UC2	名称	查看众包任务详情
参与者	审核人员，目的查看某个待审核众包任务的原始报告和审核进度		
触发条件	审核人员在中报任务列表选择一个众包任务		
前置条件	审核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. 审核人员在众包任务列表选择一个任务； 2. 系统显示测试报告审核进度，系统显示所有测试报告列表； 3. 审核人员根据测试报告严重程度升降序排列； 4. 系统显示对应排序的原始报告； 5. 审核人员根据测试报告可复现程度升降序排列； 6. 系统显示对应排序的原始报告； 7. 审核人员检索		

触发报告聚合是一个同步计算的操作，虽然用户进行的操作很简单，但是进行的工作十分重要，是整个系统的核心之一，因为聚合报告的展示内容都是基于

计算和融合的结果而来的，而这部分计算是独立于用户的，因此力争使用简便，清晰易懂。用例描述如表 3.5 所示。

表 3.5 触发报告聚合用例描述

ID	UC3	**名称**	触发报告聚合
参与者	审核人员，目的是触发聚合报告命令，开启自动化处理报告		
触发条件	审核人员选择“触发聚合报告”		
前置条件	审核人员必须已被识别和授权		
后置条件	数据计算完毕，结果持久化存储		
优先级	高		
正常流程	1. 审核人员在众包任务详情选择“触发聚合报告”； 2. 系统提示正在计算，结束后显示聚合视图入口，列表显示原始报告所属的聚合报告 ID； 3. 审核人员点击聚合报告 ID； 4. 系统显示对应聚合报告详情页面；		

查看聚合报告列表功能是为帮助审核人员查看聚合报告审核情况。除需要显示报告列表及基本信息外，其每一项还要展示报告的内容摘要，方便审核人员快速了解主题。除上述功能外，还要能根据审核状态对报告进行过滤。用例描述如表 3.6 所示。

表 3.6 查看聚合报告列表用例描述

ID	UC4	**名称**	查看聚合报告列表
参与者	审核人员，目的是查看聚合报告结果，选择要审核的聚合报告		
触发条件	审核人员选择聚合视图		
前置条件	审核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. 审核人员在众包任务详情选择“聚合视图”； 2. 系统显示聚合报告列表，每一项是主报告的内容，并显示审核状态； 3. 审核人员选择更多； 4. 系统显示对应聚合报告非主报告的其他报告信息； 5. 审核人员选择“已审核报告”； 6. 系统显示对应所有已审核状态的聚合报告； 7. 审核人员选择“未审核报告”； 8. 系统显示对应的所有未审核状态的聚合报告；		

查看聚合报告详情功能的目的是帮助审核人员快速理解信息、识别 Bug。系统通过主报告展示主要的缺陷内容，在主报告下方展示补充点，通过补充点进一步完善细节信息。系统通过可视化聚合关系帮助审核人员迅速理解聚合过程。除

此之外，词云、缺陷分类、缺陷严重性等信息的展示也将从侧面对审核人员产生帮助。用例描述如表 3.7 所示。

表 3.7 查看聚合报告详情用例描述

ID	UC5	**名称**	查看聚合报告详情
参与者	审核人员，目的是查看聚合报告显示的缺陷内容		
触发条件	审核人员选择一个聚合报告		
前置条件	审核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. 审核人员在聚合报告列表选择一个聚合报告； 2. 系统显示聚合报告详情页，显示主报告、补充点关键内容、词云、聚合关系图； 3. 审核人员展开补充点； 4. 系统显示该补充点对应的原始报告，包括文字和图片，对于和主报告不同的部分用红色标记出来； 5. 审核人员点击聚合关系图的结点； 6. 系统在关系图旁边分别显示对应结点的详细内容；		

审核报告功能主要帮助审核人员标记报告的审核状态，防止其他人重复审查。该功能主要关注审核结束是否能及时更新页面状态，提示信息是否正确。用例描述如表 3.8 所示。

表 3.8 审核报告用例描述

ID	UC6	**名称**	审核报告
参与者	审核人员，目的是确认报告审核完毕		
触发条件	审核人员选择报告已审核		
前置条件	审核人员必须已被识别和授权		
后置条件	被选中聚合报告下所有原始报告变更为已审核状态		
优先级	高		
正常流程	1. 审核人员在聚合报告详情选择“已审核”； 2. 系统提示审核成功，标签更改为“已审核”；		
扩展流程	2a. 系统显示失败，提示失败原因；		

管理预交付报告功能主要是对预交付报告的创建、编辑和删除，主要关注的是操作是否足够快捷方便，比如图片的增删操作是否简便，文字修改是否足够快捷等。本功能要求在聚合报告详情和预交付报告列表都提供操作，以提高用户使用效率，方便审核人员对不满意的预交付报告及时进行修改。审核人员在编辑报告时，可以选择报告的严重性、可复现程度和缺陷分类。用例描述如表 3.9 所示。

表 3.9 管理预交付报告用例描述

ID	UC7	名称	管理预交付报告
参与者	审核人员，目的是对预交付报告增删改查		
触发条件	审核人员选择对预交付报告进行增删改查		
前置条件	审核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. 审核人员在聚合报告详情选择“创建预交付报告”； 2. 系统在聚合报告详情右侧显示报告编辑页面； 3. 审核人员选择可复现程度、严重程度、Bug 类型，并填写 Bug 描述和图片，审核人员可以快速复制报告的描述内容，图片可以通过快速拖拽来增删，并点击保存； 4. 系统提示保存成功，并展示已创建的预交付报告列表； 5. 审核人员选择编辑预交付报告； 6. 系统显示报告编辑页面，并填充该报告内容； 7. 审核人员修改报告内容，点击保存； 8. 系统提示保存成功，并刷新预交付报告列表； 9. 审核人员选择删除报告； 10. 系统提示删除成功，并刷新预交付报告列表；		
扩展流程	3a. 审核人员点击取消； 4a. 系统取消更改； 7a. 审核人员点击取消； 8a. 系统取消更改；		

3.3 系统总体设计

3.3.1 系统整体架构设计

众包审核交付系统的系统架构图如图 3.3 所示，下面从工程设计的角度进行描述。

系统前端页面模板引擎采用 Thymeleaf，它集成于 Spring Boot 框架，能够完成相对复杂的前端逻辑，同时保持快速高效的渲染能力。UI 组件库选择的是比较成熟的 Bootstrap，通过预定义好的样式和简便的方式就可快速使用。前后端除页面请求时采用 HTTP 同步请求外，其他页面局部更新的请求均采用 AJAX 进行异步请求，刷新页面，可有效降低页面的延迟。图形可视化的工作则交由 D3 组件来完成，D3 组件提供深度定制图形能力，更加灵活，但使用难度相对 ECharts 等常用组件更高。

系统服务端采用 Spring Boot 框架开发，由于其自身丰富的可扩展性和强大的社区支持，能够快速和其他服务组件进行集成。服务端需要承担与审核人员交互时的一系列业务逻辑，因此服务端需要和外部服务进行交互，获取众包测试任务的原始报告数据，对用户进行认证和授权，获取众包测试任务列表和被测应用的信息。由于用户服务等一些服务只支持 Dubbo 调用，因此服务端集成了 Dubbo 客户端，并通过 ZooKeeper 进行服务发现和服务注册。

前端和服务端采用 Nginx 进行负载均衡，可以保持比较好的水平扩展性，同时能保证服务器资源得到充分地利用。服务端使用 Redis 作为第三方缓存中间件，用于存储用户 Session 等一些需要在服务器之间共享的临时数据。相比于进程内缓存，共享的缓存可以让数据不固定于一台服务器，使服务器做到无状态，有利于水平扩展。为保证数据的可靠性，存储层采用主从备份的做法。应用层和存储层采用数据库连接池来管理连接，防止资源的重复开销，提高使用效率。

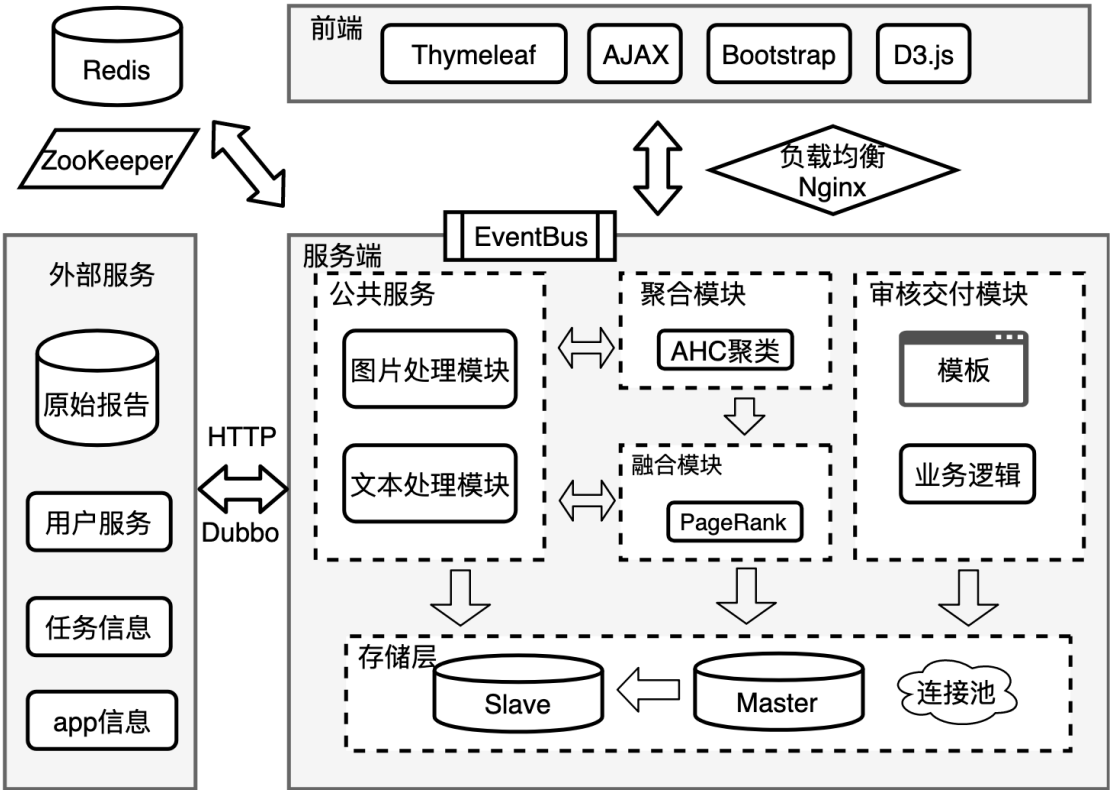

图 3.3 系统整体架构图

由于测试报告自动化处理时间较长，如果使用 HTTP 会导致超时重试，为防止这种情况，服务端采用让任务进入异步处理的队列 EventBus，任务结束后使用回调函数将任务结束的标记写入 Redis，前端则轮询任务结果，拿到结果以后修改前端状态，并请求结果数据。

3.3.2 系统模块划分

如图 3.3 所示，本系统从功能模块的角度总体可划分两大部分，第一部分是自动化报告处理，进一步细分为文本处理模块，图片处理模块，聚合模块，融合模块；第二部分为审核交付模块。下文会首先概述自动化报告处理的总体设计，然后会进一步描述几个细分模块的设计。

自动化报告处理主要完成从原始报告列表到聚合报告的生成过程。其中，文本处理模块的主要职责包括文本分词、清洗、Word2Vec 模型加载，文本相似度计算等。图片处理模块的主要职责包括图片下载，特征提取和索引，相似度计算等。聚合模块负责整合文本和图片特征，利用聚类算法对相似报告进行聚合。融合模块负责相似报告的合理融合，其主要步骤包括主报告识别、报告拆分、差异点识别、差异点聚类、差异点排序等。

审核交付模块主要承担审核人员的主要工作和一系列交互、业务逻辑，包括众包任务查看、众包任务详情、触发报告聚合、聚合报告详情、预交付报告生成和管理等。其设计上主要采用传统的 MVC 方案，并结合一些缓存和异步等技术进行一定优化。

3.3.3 4+1 视图

本小节从场景视图、逻辑视图、进程视图（Process View）、开发视图和物理视图来详述系统总体设计。其中场景视图描述的是用户场景，是其他四种视图的根本，在 UML 中对应用例图，本系统的场景视图如前文中图 3.2 所示。

逻辑视图面向的是最终用户视角，描述的是系统最终提供给用户的服务，在 UML 中可用类图来描述，如图 3.4 所示，系统被分解为一系列关键抽象。

TaskService 主要提供众包任务列表和详情的查询。BugReportService 主要提供原始报告数据获取的功能。Aggregator 提供报告聚合处理功能，完成测试报告自动化处理操作。Aggregator 需要调用多个功能来实现报告处理，其中，DistMatrix 提供报告相似度计算功能，返回相似度矩阵，ClusterAnalyzer 提供聚类分析功能，MasterReportService 提供筛选聚类中的主报告及其管理的功能，SupplementService 提供筛选主报告的补充点及其管理的功能。DistMatrix 调用

的 NLPService 和 ImgProcessService 分别提供文本和图片的特征提取和相似度计算。FinalReportService 提供管理预交付报告的功能。

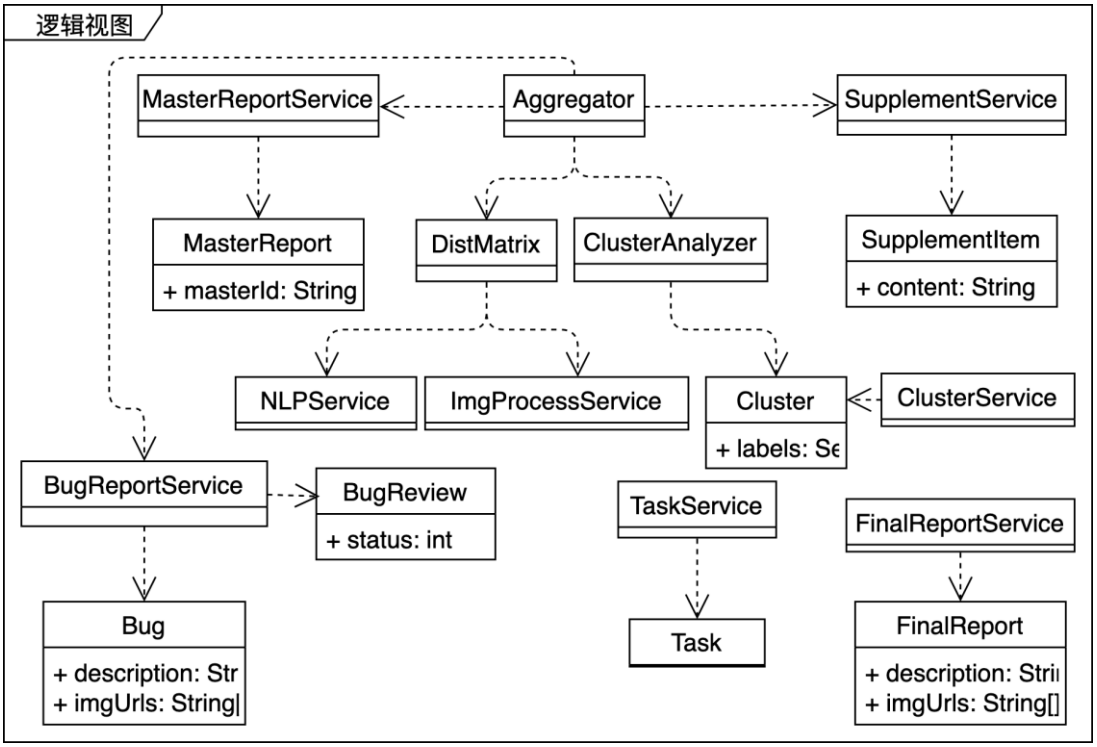

图 3.4 逻辑视图

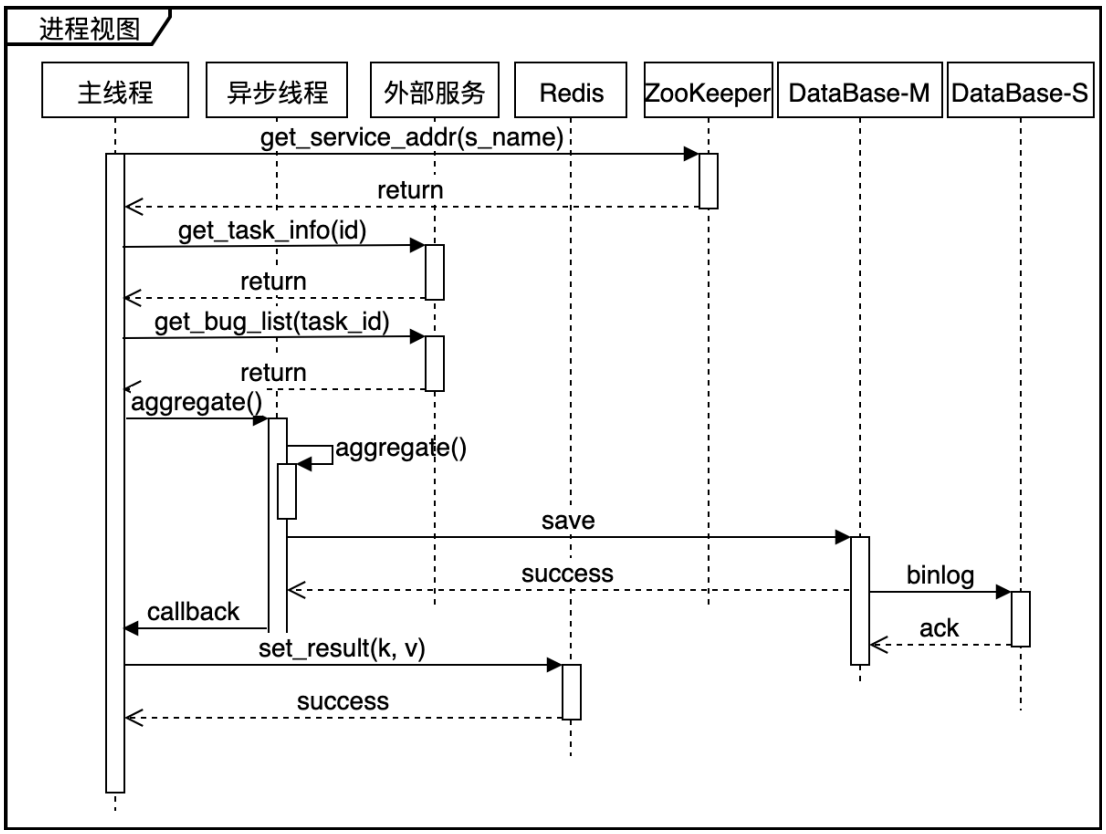

图 3.5 进程视图

如图 3.5 所示，进程视图面向系统集成者视角，主要描述系统相关进程、线程实体及其通信。当系统需要访问外部服务获取数据时，主线程会向服务注册中心 ZooKeeper 查询服务地址，并根据地址调用相关服务。系统主线程会向外部服务进行同步调用，并获取众包任务和原始报告等信息。当触发报告聚合时，主线程会启动新线程异步地对报告进行计算，异步线程会将结果存储到数据库，计算完成后回调主线程的事件处理函数，将任务结果存储到 Redis。与此同时，数据库的 Master 进程会将二进制日志传给 Slave，对数据进行实时同步备份。

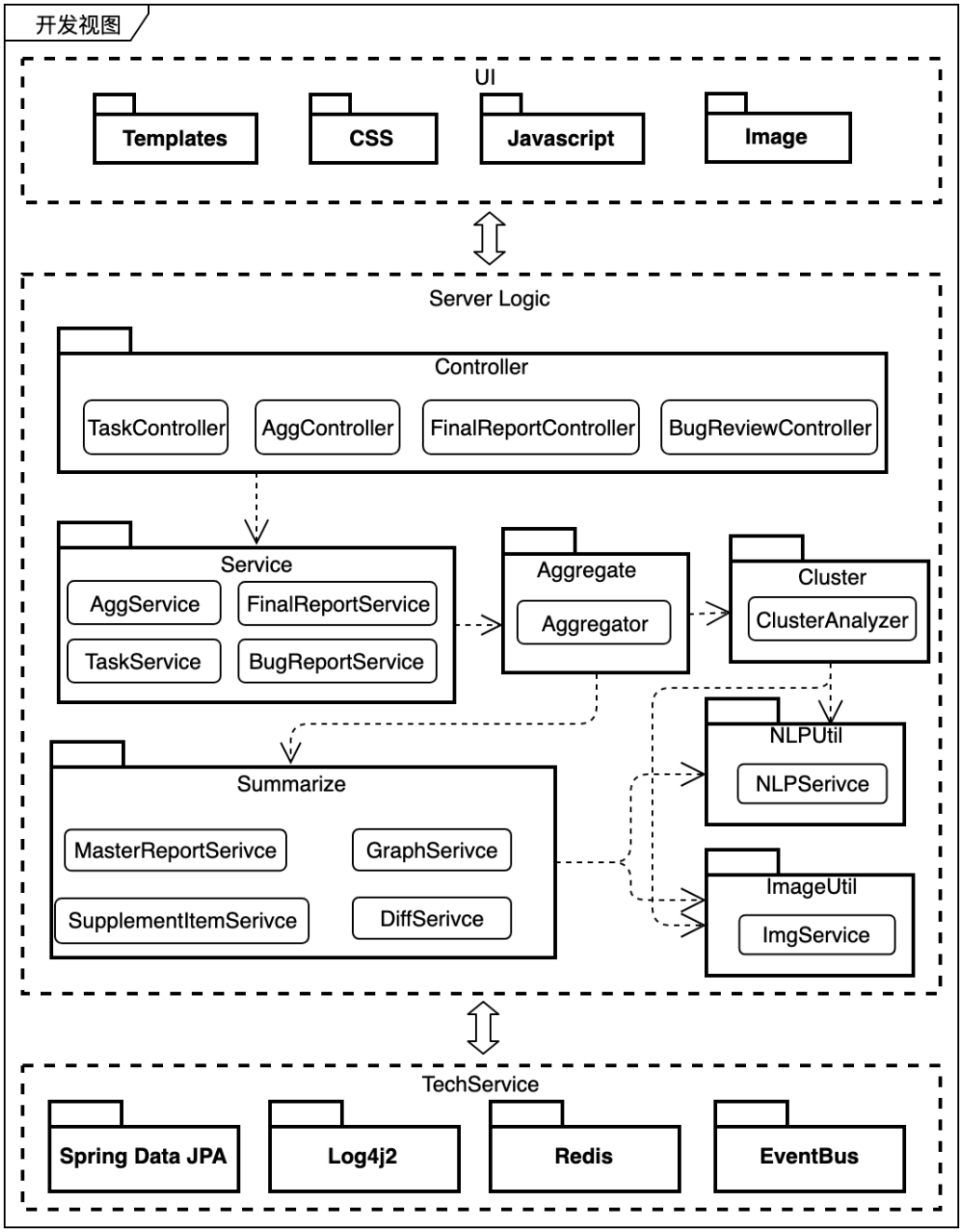

图 3.6 开发视图

如图 3.6 所示，开发视图面向开发人员视角，描述开发环境下的软件模块组织和管理。UI 部分的管理分为模板文件和静态资源管理（JavaScript、CSS、图片）。服务端业务逻辑按照功能和层级来划分。Controller 包负责控制器管理，面向所有外部请求。Service 包负责对接 Controller 并实现审核交付相关的业务逻辑。Aggregate 包负责自动化报告处理的总体逻辑。Cluster 包负责聚类分析的实现。Summarize 包负责报告融合的实现。NLPUtil 包和 ImageUtil 包负责文本和图片的特征提取和相似度计算。技术服务主要是业务独立的基础技术，包括日志 Log4J2，缓存 Redis 访问的客户端封装等。

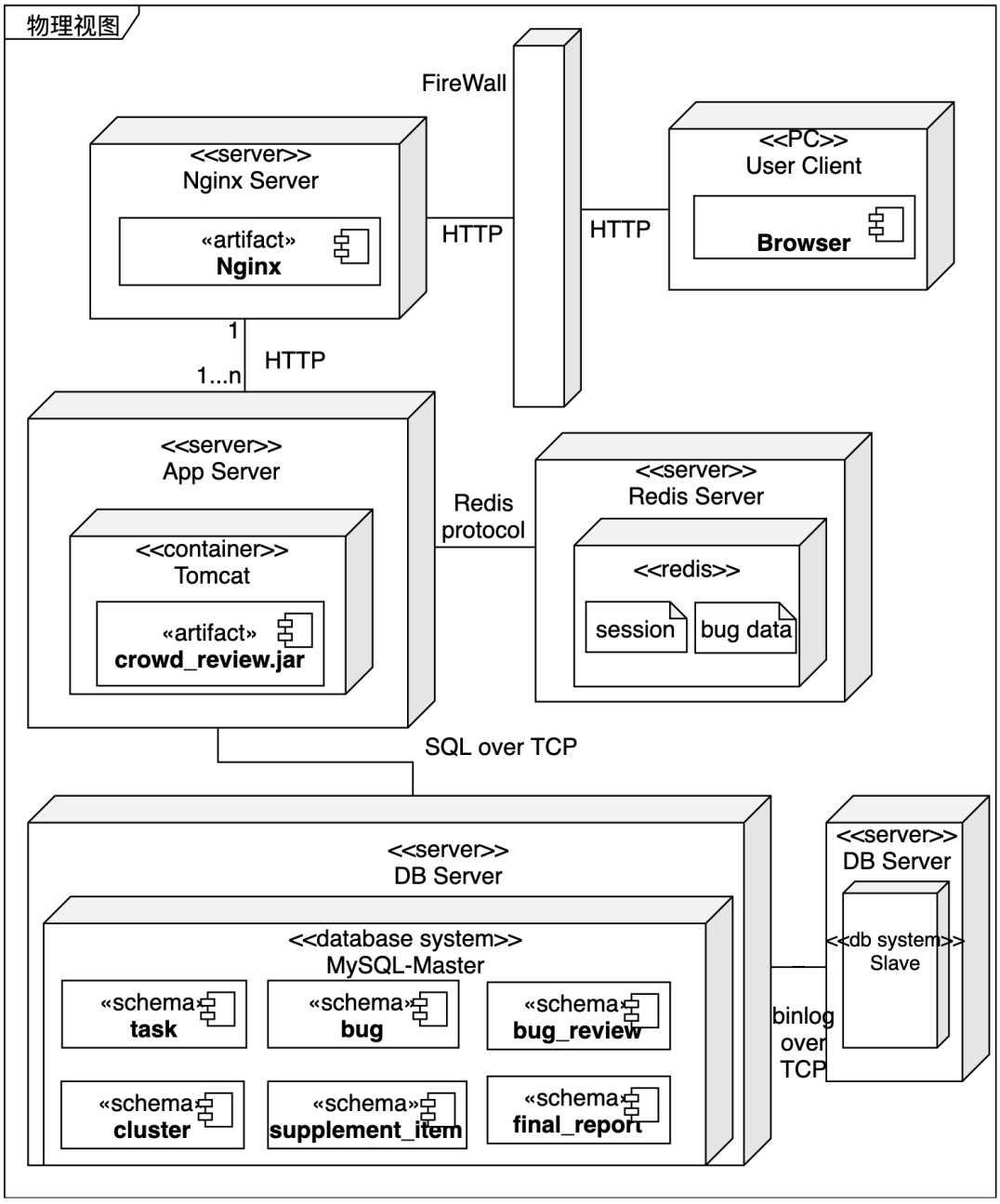

图 3.7 物理视图

如图 3.7 所示，物理视图面向运维人员视角，主要描述系统物理节点间的通信和部署情况。用户通过浏览器即可对系统发起 HTTP 访问，中间需要经过防火墙对黑名单的过滤。验证通过后会首先访问到 Nginx 服务器，进行负载均衡，请求被均分到多个应用服务器。同时，将服务器内部的 Session 托管到 Redis 保存，使用户请求可以随意落到不同应用服务器，保证系统的水平扩展性。应用服务器与 Redis 之间通过 Redis 自定义协议通信。应用服务器与数据库之间分开部署，通过 TCP 传递 SQL 查询。数据库主库与从库分开部署，通过二进制日志进行实时同步。

3.3.4 测试报告自动化处理总体设计

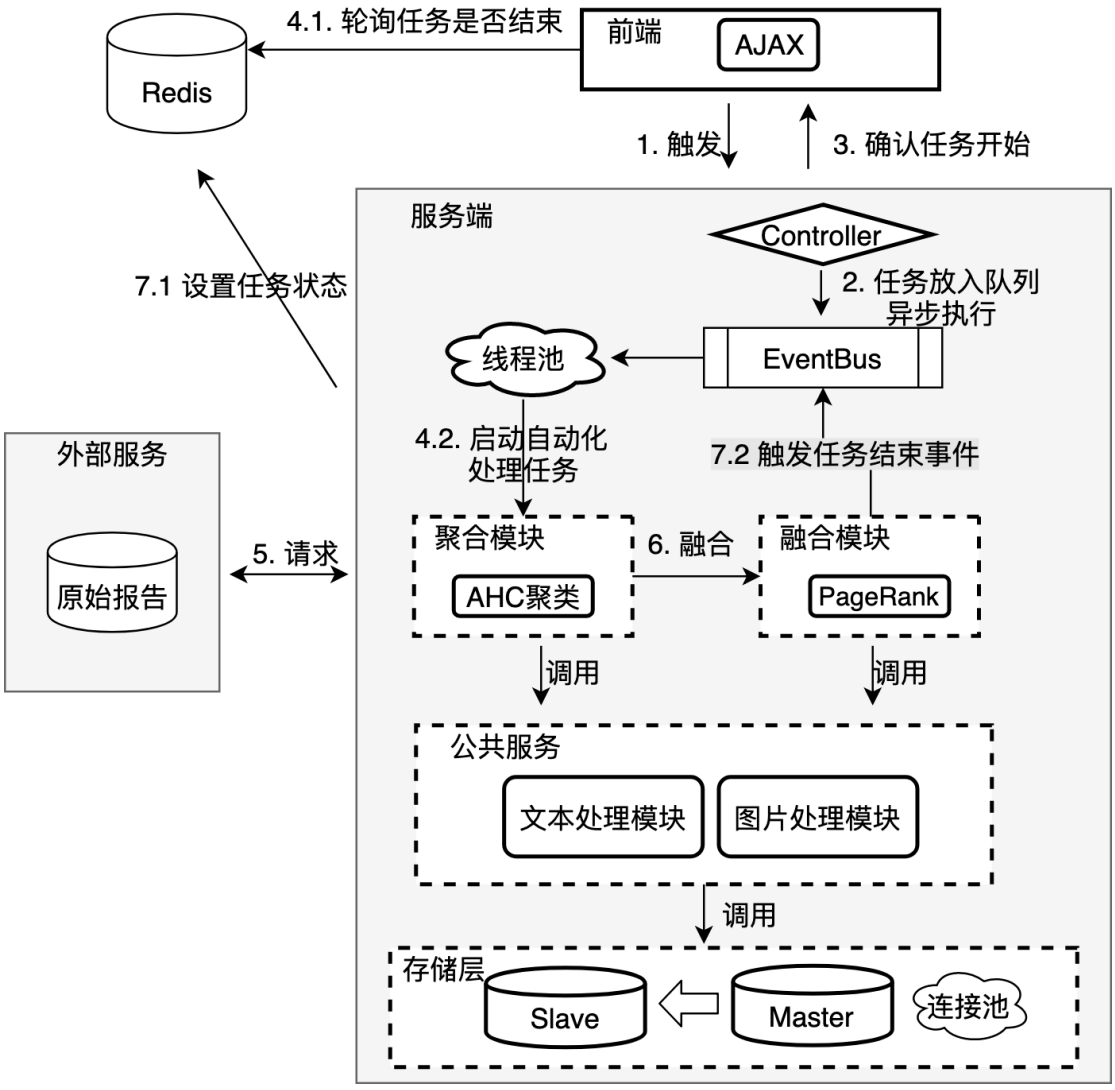

图 3.8 测试报告自动化处理总体架构

测试报告自动化处理的总体设计如图 3.8 所示。主要步骤如下：

(1) 前端通过 AJAX 发起测试报告自动化处理请求；

(2) 服务端的控制器处理 HTTP 请求，把任务提交至 EventBus；

(3) 控制器返回请求成功，意味着任务开始；

(4) 4.1 前端间隔一段时间后开始轮询任务结果；4.2 EventBus 把任务提交给异步线程池并调用聚合模块开始处理任务；

(5) 聚合模块向外部服务请求原始报告，并调用文本处理模块和图片处理模块进行相应处理，并聚类，中间结果均存储在数据库；

(6) 融合模块处理聚合模块的结果，进行信息提取，中间结果及最终结果存储于数据库；

(7) 7.1 通过预先设置好的回调函数，设置 Redis 的任务标志位，修改成功后前端获得结果，展示上做出对应的处理；7.2 触发任务结束事件。

至此，测试报告自动化处理的主要步骤结束。

由于报告处理时间偏长，如果直接使用 HTTP 来同步请求，会因为超时重试机制导致任务被重复执行，解决方法也比较多，本系统主要使用 EventBus 来异步处理任务，通过使用一个新的线程池来执行添加的任务。同时对任务设置好回调函数，当任务结束时可以通知 Redis 修改任务状态位。修改之后，前端通过轮询机制就能获得任务结束的通知，从而请求数据，改变对应的展现。具体的报告处理模块将在后续小节进行分析。

3.4 文本处理模块设计

3.4.1 架构设计

如图 3.9 所示，文本处理模块提供基本的中文分词分句功能，分词功能是指将一句中文，如“我爱北京天安门”分为“我”,“爱”,“北京”,“天安门”。分句功能指将一段中文，根据标点符号分解为几句。这里将采用 HanLP[6]工具来完成。接着是文本清洗，这里主要是停用词过滤，即读取本地停用词文件，将分词或分句结果进行一一匹配，并保留有效部分。特征提取部分提供文本特征提取，

[6] https://github.com/hankcs/HanLP

目前提供 TF-IDF 向量和 Word2Vec 词向量。相似度计算部分目前提供两种算法，分别是余弦相似度和 Jaccard 相似系数。分词、分句、特征提取和相似度计算的功能都有对外提供的接口。

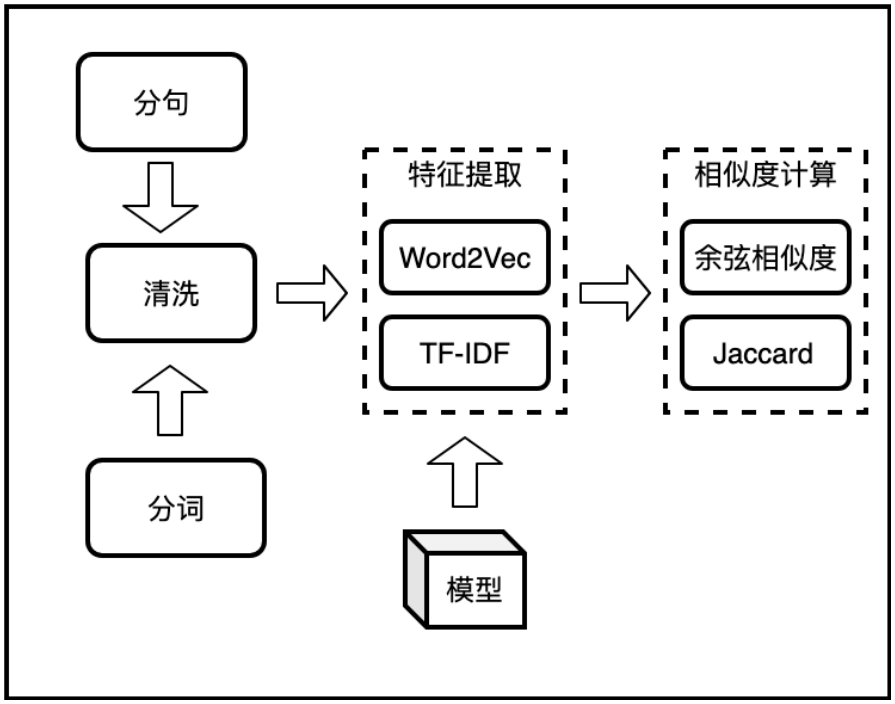

图 3.9 文本处理模块架构图

3.4.2 核心类图

如图 3.10 所示是文本处理模块的核心类图设计，NLPService 是主要对外提供服务的类，暴露的三个接口主要对应文本处理模块所提供的三个功能，分别是分词分句（segment 和 seg2sentence 方法），特征提取（getVector 方法）和相似度计算（similar 方法）。

分词分句主要依赖 NotionalTokenizer 类，这个类主要依赖 HanLP[6] 完成具体功能。文本清洗工作主要由 StopWordFilter 类完成，通过 loadStopWordFile 方法加载本地项目的停用词表对 NotionalTokenizer 分词结果进行过滤。Item 类负载记录分词结果，其中 word 代表分词结果，nature 代表词性。

文本特征提取主要通过 Doc2VecModel 类提供服务，主要提供单文本向量查询（query 方法）、文本相似度计算（similar 方法）、最相似文本查询（nearest 方法）的功能。Doc2VecModel 主要依赖于 Word2VecModel 类，Word2VecModel 定义在 HanLP 组件中，通过 Word2VecUtil 加载项目下预训练的词向量模型生

成 Word2VecModel 实例。Vector 主要是词向量结果的实例，提供对结果数组的一些计算方法，方便在计算过程中的一些数值处理。相似度计算主要通过 Similar 类提供的余弦相似度和 Jaccard 相似系数。

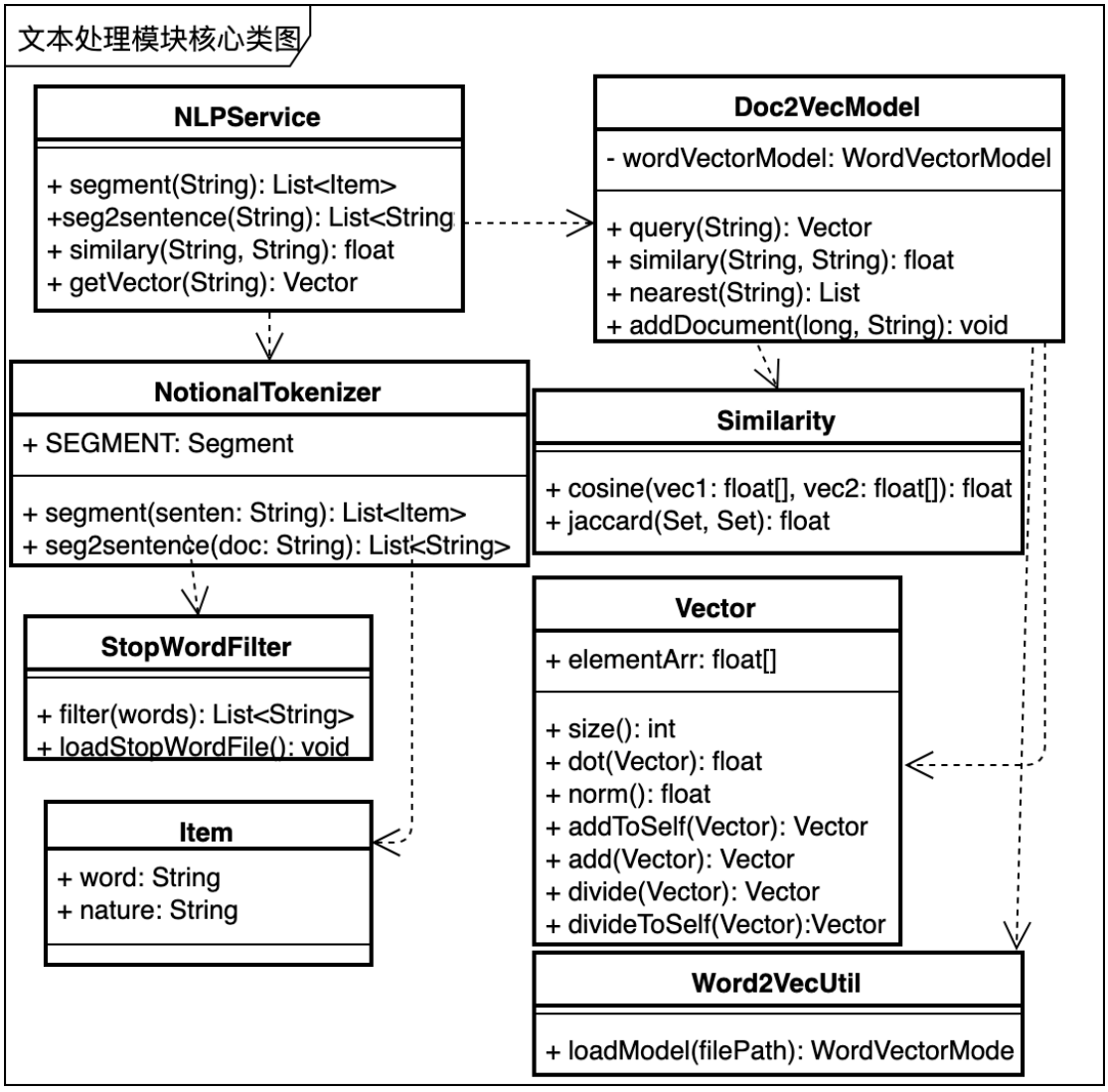

图 3.10 文本处理模块核心类图

3.5 图片处理模块设计

3.5.1 架构设计

如图 3.11 所示，图片处理模块主要分为图片下载、特征提取和相似度计算三个部分。图片下载主要通过访问阿里云对象存储进行下载。接下来通过特征提取模块进行特征抽取，支持多种特征，本系统采用 CEDD[Chatzichristofis et al.,

2008]特征，但保留其他特征提取的能力，便于以后算法替换和升级。提取特征以后会先将特征向量缓存在本地文件中，通常是任务 id 命名的目录下。相似度计算时会根据传入的图片 id 读取检索任务目录下的文件，加载向量至内存。

在融合模块计算结束后，系统会发送任务结束事件，EventBus 会收到事件通知，启动对应的回调函数清理缓存在本地的特征向量文件，以降低本地磁盘容量，避免磁盘容量耗尽。

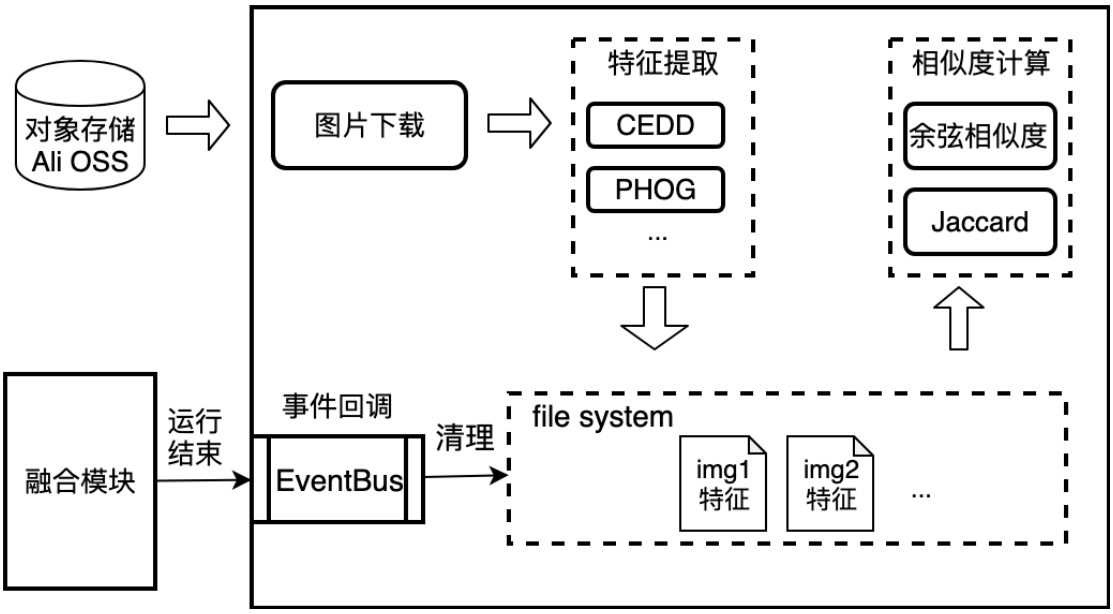

图 3.11 图片处理模块架构图

3.5.2 核心类图

如图 3.12 所示是图片处理模块的核心类图设计。ImgProcessService 主要负责对外提供的功能是图片下载与特征抽取（downloadAndExtractImg 方法）和图片相似度计算（similarity 方法）。

图片下载主要依赖 ImageDownload 类，通过 RestTemplate 发起 HTTP 请求依次下载图片到本地。图片地址则通过 BugReportService 获取，首先获得所有报告信息，再依次取出它们的图片地址。

特征抽取主要通过 FeatureExtractor 实现，参数是选用的特征类型和图片的二进制流对象，返回值是 GlobalFeature。具体抽取过程主要依赖于 LIRE[7]框架提供的各种描述符完成。GlobalFeature 保存了特征向量，并提供距离计算方法。

[7] http://www.lire-project.net/

距离也可以通过 Similarity 类计算。特征的本地存储和访问则主要通过 FileUtil 类来实现。

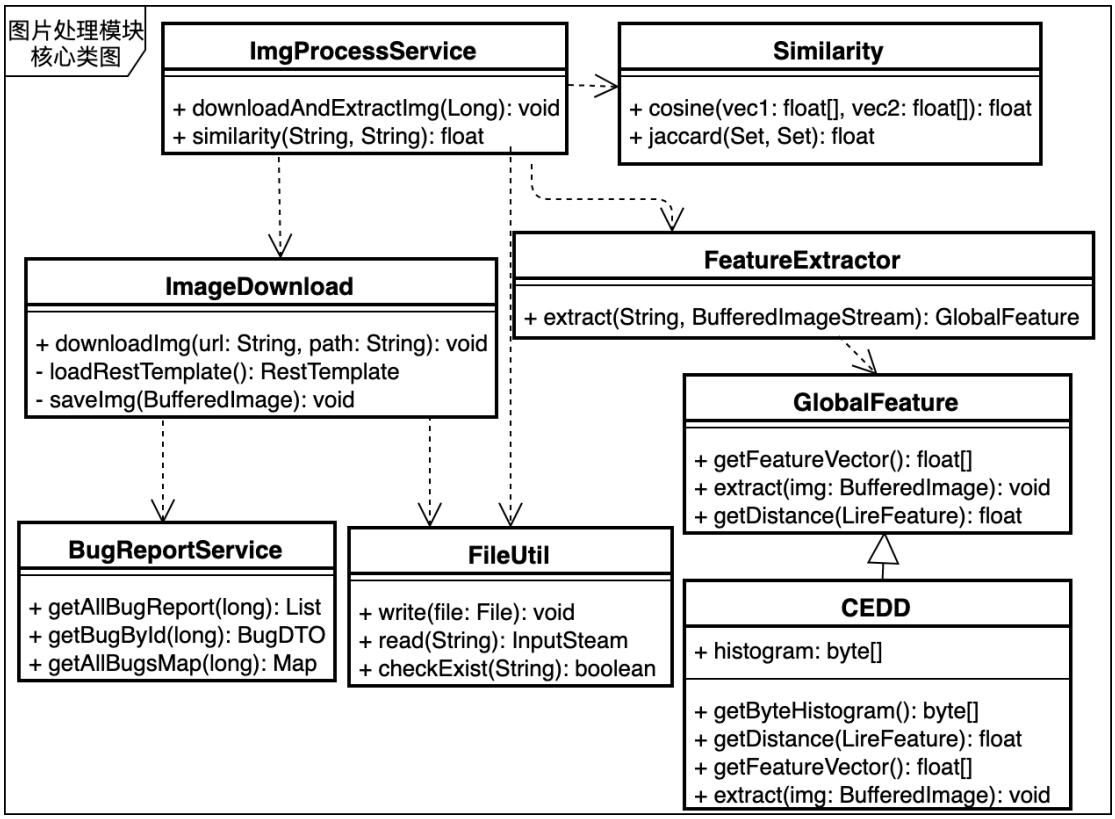

图 3.12 图片处理模块核心类图

3.6 聚合模块设计

3.6.1 架构设计

聚合模块架构设计如图 3.13 所示。聚合模块的主要功能是作为自动化报告处理的入口，完成测试报告第一阶段的聚类处理。聚类处理的主要步骤是：首先，访问外部服务，获取任务对应的所有原始报告；其次，调用文本处理模块和图片处理模块完成对原始报告预处理，获得文本和图片的相似矩阵；然后，将两个矩阵合并成统一的相似矩阵；最后，调用凝聚式层次聚类算法进行聚类，并将结果存储于数据库中。

为应对文本和图片特征的替换和升级，文本处理模块和图片处理模块屏蔽了特征抽取和相似度计算的细节，而是将最终结果以相似矩阵传递给聚合模块。聚合模块可以通过传递参数来控制文本和图片所抽取的特征类型。目前本系统的文

本特征采用词向量，图片特征采用 CEDD 特征。

为降低相似矩阵的存储规模，在实际的计算过程中，使用的是相似矩阵的下三角矩阵进行计算，避免二维矩阵带来的冗余。

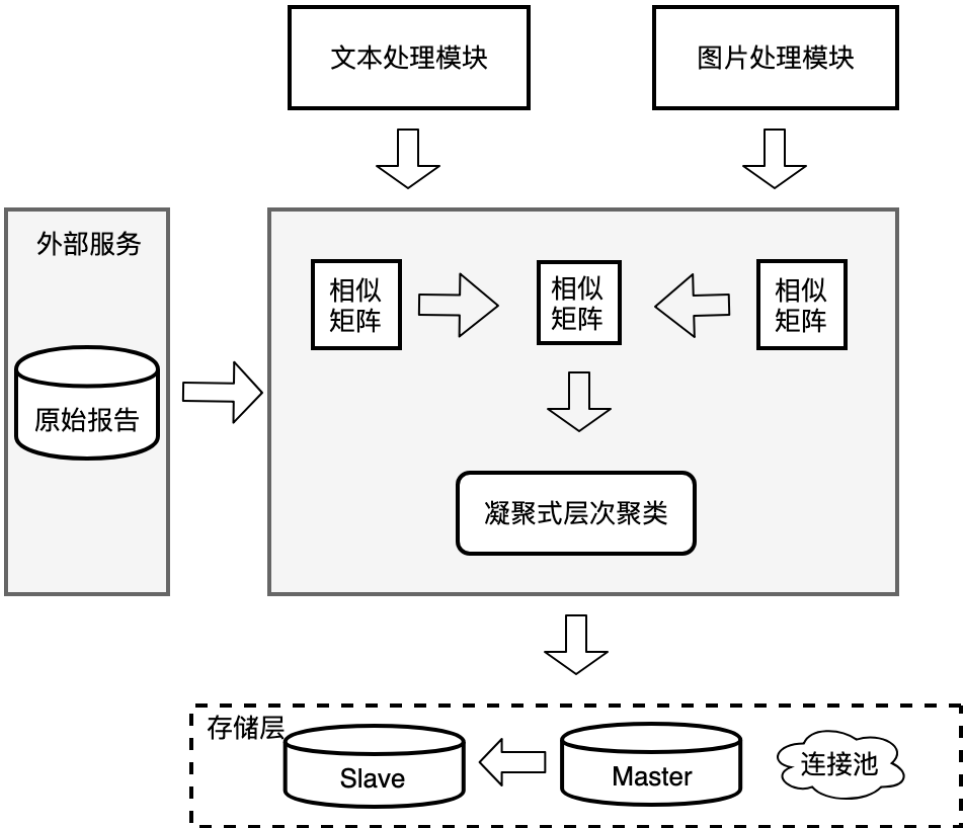

图 3.13 聚合模块架构图

3.6.2 核心类图

聚合模块核心类图如图 3.14 所示。在聚合模块里，Aggregator 类提供了 aggregate 接口来触发报告自动化处理。

BugReportService 可以获取所有原始报告，为降低网络开销，该类会首先查询数据库中是否已存在原始报告，如果没有，再访问外部服务获取后存到数据库中。

DistanceMatrix 主要提供获取距离矩阵的接口，其中 getHybridDistMatrix 是获得混合距离矩阵的接口，会分别调用 NLPService 和 ImageProcessService 的相关接口进行计算。为节省内存空间，NLPService 和 ImageProcessService 类的 similar 方法将相似矩阵的下三角部分压缩成一维数组后返回。

ClusterAnalyzer 主要负责聚类分析，对外提供凝聚式层次聚类方法的接口 AHC。genClusters 方法则是一个私有方法，主要作用是处理聚类结果数据并以符合业务数据的格式返回。HierachricalClustering 提供凝聚式层次聚类算法的主要实现，并且保存了计算的中间过程和最终结果，可以通过 getTree 和 getHeight 方法来访问这些数据。Linkage 类提供聚类过程中，度量类簇之间相似度的算法。WardLinkage 继承 Linkage，实现基于 ward 方法的类簇相似度算法。为了保证 Linkage 算法的可扩展性，可以通过继承 Linkage 类来实现其他度量方法。

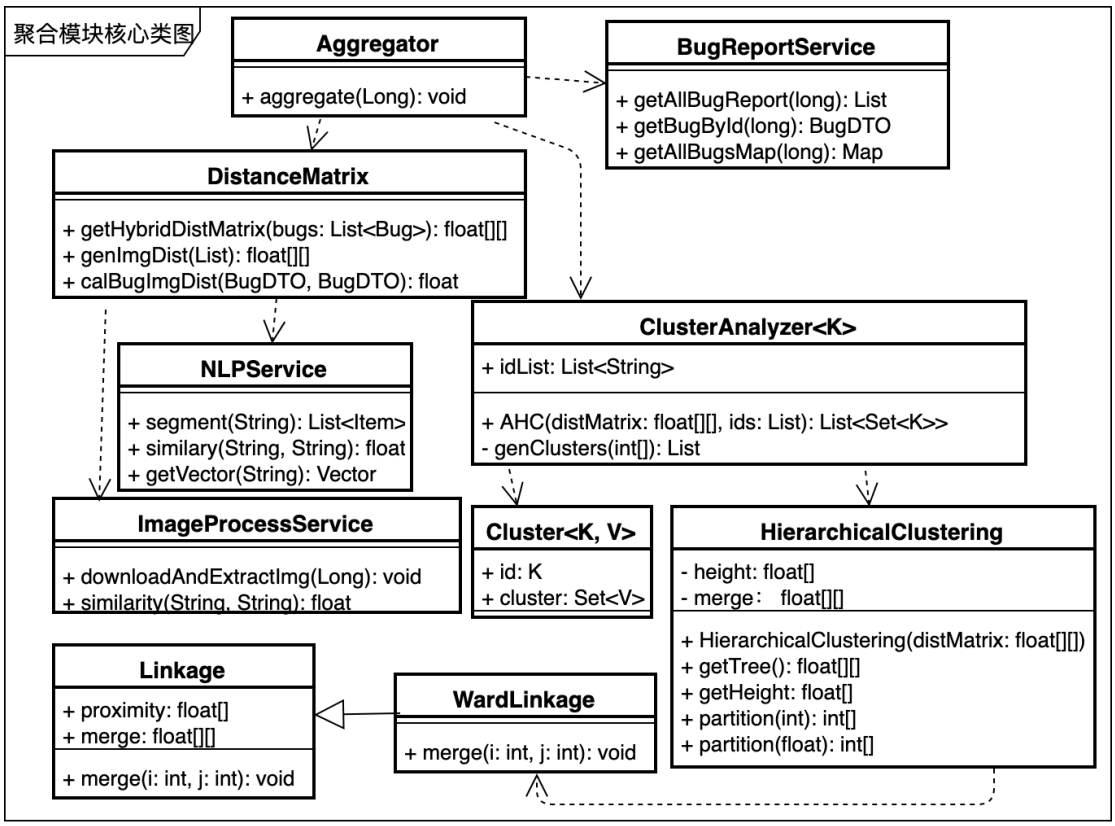

图 3.14 聚合模块核心类图

3.6.3 数据库设计

聚合模块数据库 ER 图如图 3.15 所示。图中描述了聚合模块涉及到的三个实体，其中 task 代表众包任务，bug 代表原始缺陷报告，cluster 代表聚类结果。根据图中分析，task 和 bug 是一对多关系，一次众包任务对应多份缺陷报告；task 和 cluster 是一对多关系，一次众包任务的报告可以产生多个聚类；cluster 和 bug 是一对多关系，一个类簇由多个缺陷报告组成。

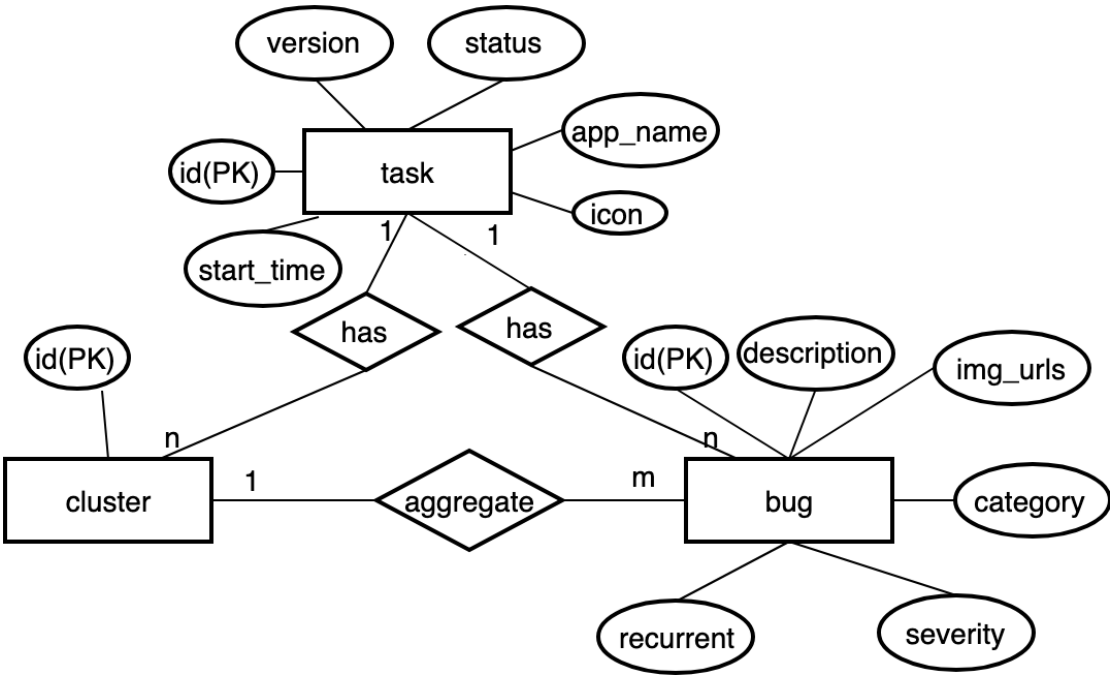

图 3.15 聚合模块 ER 图

如表 3.10 所示为 bug 表的数据库表字段说明。该表主要保存原始报告的主要信息。虽然可以从外部服务获取原始报告，但是本系统出于兜底策略的考虑，同时保证访问速度，故选择本地存储一份原始报告的数据。

表 3.10 bug 表

字段名	类型	描述
id	BIGINT	bug 唯一 id，bug 表主键
task_id	BIGINT	任务 id，对应众包任务的唯一标识
cluster_id	BIGINT	类簇 id，用于聚类唯一标识
description	VARCHAR	bug 描述，报告中对 bug 的描述
img_urls	VARCHAR	图片地址，报告上传的图片存储地址，多张图片地址以逗号间隔
category	INT	bug 所属类型： 1：安全；2：功能不完整；3：性能；4：页面布局缺陷；5：用户体验；6：不正常退出；7：其他
severity	INT	bug 严重性： 1：待定；2：较轻；3：一般；4：严重；5：紧急；
recurrent	INT	bug 可复现程度： 1：其他；2：无规律复现；3：小概率复现；4：大概率复现；5：必现

如表 3.11 所示是 task 表的数据库表字段说明。该表主要保存众测任务信息。虽然可以从外部服务获取相关数据，但建此表的理由和 bug 表相同，也是兜底和加速访问。

表 3.11 task 表

字段名	类型	描述
id	BIGINT	任务 id，对应众包任务的唯一标识
app_name	VARCHAR	应用名
icon	VARCHAR	应用图标地址
version	VARCHAR	被测应用的版本号
status	INT	任务审核状态 0：审核中；1：审核结束
start_time	TIMESTAMP	任务开始时间
end_time	TIMESTAMP	任务结束时间

由于 cluster 表还涉及到融合模块的分析，其数据库表设计将在融合模块的数据库设计中给出。

3.7 融合模块设计

3.7.1 架构设计

融合模块的架构设计如图 3.16 所示。融合模块是在聚合模块的结果之上进行的，是报告处理的第二阶段。

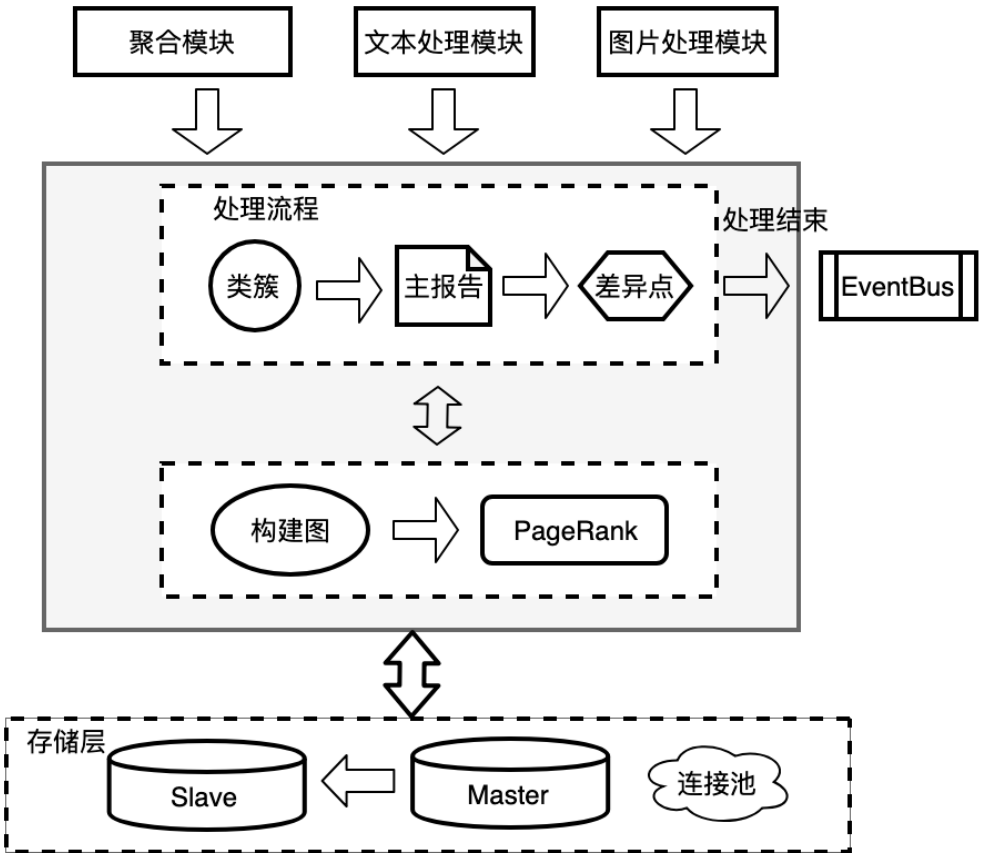

图 3.16 融合模块架构图

融合模块在处理过程需要调用文本处理模块、图片处理模块和聚合模块的相关接口，因为计算过程中涉及到文本处理、图片处理、相似度计算和聚类。除此之外，计算过程还涉及到两次图的构建和 PageRank 算法的运行，一次是从类簇筛选主报告的过程；另一次是从类簇的其他报告分离出与主报告的差异点之后需要通过 PageRank 算法进行排序。由于图相关的计算和逻辑处理是独立且可复用的，因此独立出来作为一个公共服务。

计算过程中和计算结束后，都需要将中间结果和最终结果存入数据库，方便展示和分析。除此之外，还需要向 EventBus 发送处理结束的事件，触发后续的文件清理和任务状态变更等操作。

3.7.2 流程设计

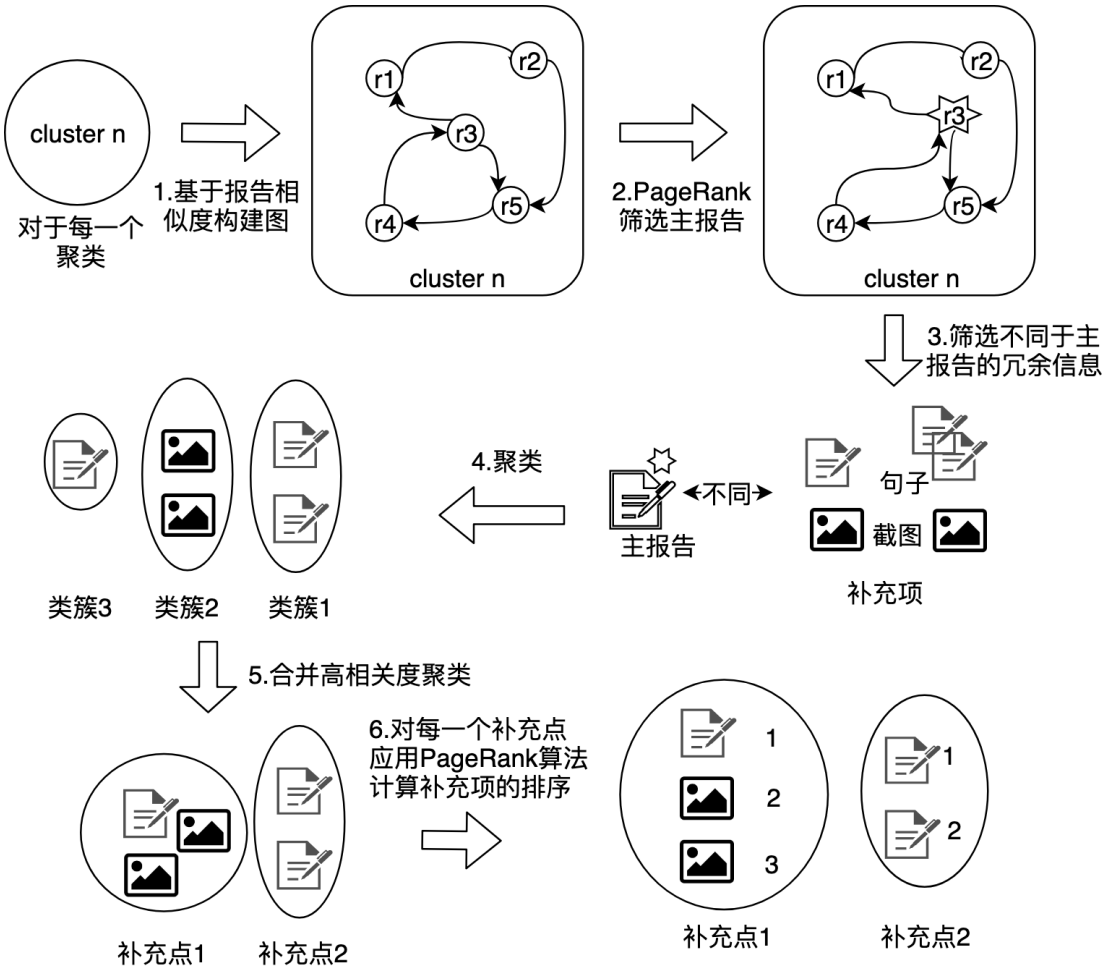

图 3.17 融合模块流程设计

融合模块的流程设计如图 3.17 所示。融合模块的处理流程主要参考 Hao 等人提出方法[Hao et al., 2019]，并结合实际业务系统自行实现。下面是融合阶段的主要步骤：

(1) 融合模块会把同一个聚类里的相似报告抽象成带权图，节点是报告，边的权重是两个报告的相似度（基于聚合模块生成的距离矩阵）。

(2) 进一步利用 PageRank 算法计算节点的权重，以此度量报告的重要性。最高权重的节点作为“主报告”放到“聚合报告”中。

(3) 在同一个聚类里，分别比较“主报告”和其他报告的句子、截图的相似度，并把不同的句子、截图标记出来作为“候选项”。

(4) 考虑到“候选项”之间也存在重复，通过对这些“候选项”聚类，将相似的“候选项”聚合到一起，每个聚类代表一个子主题。

(5) 对相关度较高的不同聚类进行合并，度量方法是用 Jaccard 相关系数度量子主题包含的相同缺陷报告比例，超过一定阈值（目前是 50%）就进行合并。

(6) 对各个子主题使用 PageRank 算法，对每个子主题内部的信息项进行排序，排名越靠前的信息项越能代表该子主题的内容。

3.7.3 核心类图

融合模块的核心类图如图 3.18 所示。和聚合模块一样，融合模块的计算也由 Aggregator 类的 aggregate 方法触发。

MasterReportService 主要负责主报告相关的业务逻辑，包括主报告选取、检测报告是否已聚合、获取所有主报告等功能。MasterReportService 的计算主要依赖于 GraphService，因为主报告选取的算法采用的是 PageRank 算法，需要将报告构建为图结构。

GraphService 提供的 buildDirectedGraph 方法能将报告构建为图结构，提供的 pageRank 方法能在图结构的基础上来计算节点权重。PageRank 类主要提供算法实现和计算结果的查询。

DiffTextService 和 DiffImgService 类主要负责识别同一类簇中，其他报告相对于主报告的差异点。具体方法是将每份报告拆解为句子和单张图片，依次与主

报告进行比较。这个过程中会用到文本和图片处理模块的相关功能，如分句、相似度计算等。除此之外，这两个类还要对差异点进行聚类，以形成子主题。

Aggregator 的 combineCluster 方法会对相关度较高的类簇进行合并。合并方法是计算两个类簇之间的相同原始报告数所占的比例，目前当该比例超过 50% 时，认为两个类簇描述的内容相近，会合并为同一类簇。SupplementItemDao 主要用于存储合并后的结果和补充项类簇的排序结果。

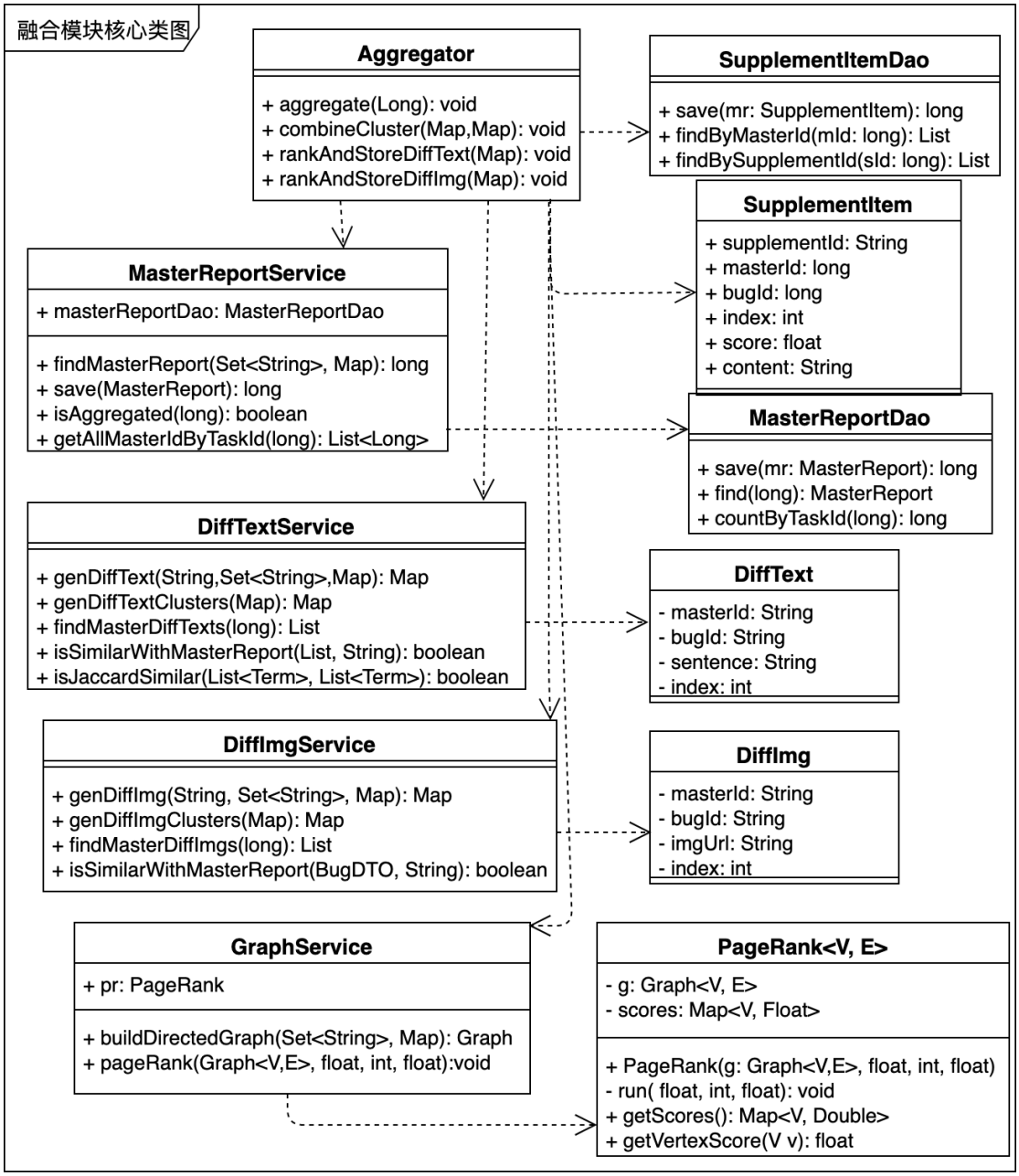

图 3.18 融合模块核心类图

3.7.4 数据库设计

融合模块数据库 ER 图如图 3.19 所示。master_report 表示主报告，diff_txt 和 diff_img 分别表示差异文本和差异图片，supplement_item 表示最终的补充项数据。cluster 与 master_report 的关系是一对一，每个类簇对应一个主报告；master_report 与 diff_txt 和 diff_img 的关系都是一对多，每个 master_report 对应产生一个到多个差异文本和图片；supplement_item 与 diff_txt 和 diff_img 的关系都是一对一，为计算方便，diff_txt 和 diff_img 保存计算过程的中间结果，supplement_item 保存类簇合并后的最终结果。

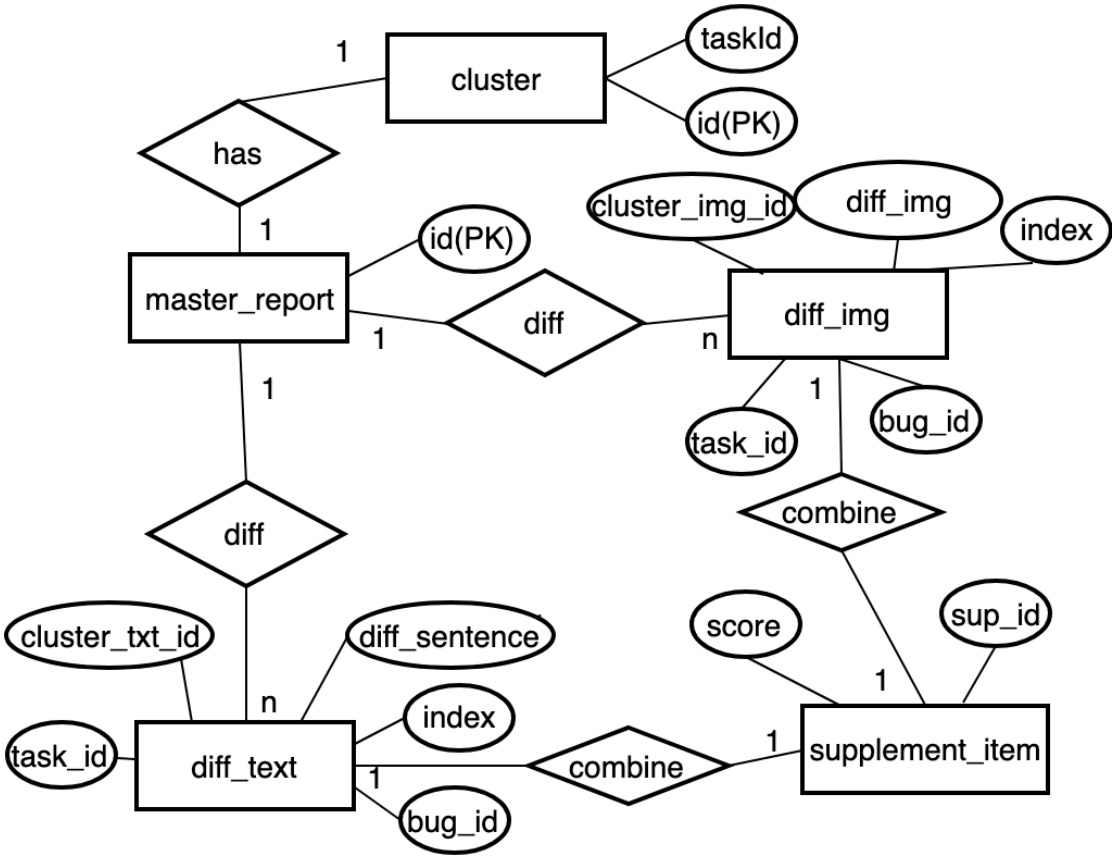

图 3.19 融合模块 ER 图

表 3.12 所示为 cluster 表的数据库表设计及相关字段说明。主要用于存储聚合模块的聚类结果和主报告信息以及所属的众包任务。

表 3.12 cluster 表

字段名	类型	描述
cluster_id	BIGINT	类簇 id，用于聚类唯一标识
master_id	BIGINT	主报告 id，格式与 bug id 相同
task_id	BIGINT	任务 id，众包任务的唯一标识

表 3.13 所示为 diff_text 表的数据库表设计及相关字段说明。主要用于记录差异文本所属的聚类、原始报告、在原始报告中的序号、句子内容以及句子聚类结果。

表 3.13 diff_text 表

字段名	类型	描述
cluster_id	BIGINT	类簇 id，用于聚类唯一标识
master_id	BIGINT	主报告 id，格式与 bug id 相同
bug_id	BIGINT	bug 报告 id，所属原始报告 id
cluster_txt_id	VARCHAR	句子类簇 id，对 diff_text 进行聚类得到的类簇 id
index	INT	序号，句子在描述段落里的序号
diff_sentence	VARCHAR	句子，与主报告差异较大的句子

表 3.14 所示为 diff_img 表的数据库表设计及相关字段说明。主要用于记录差异图片所属的聚类、原始报告、在原始报告的序号、图片地址以及图片聚类的结果。

表 3.14 diff_img 表

字段名	类型	描述
cluster_id	BIGINT	类簇 id，用于聚类唯一标识
master_id	BIGINT	主报告 id，格式与 bug id 相同
bug_id	BIGINT	bug 报告 id，所属原始报告 id
cluster_img_id	VARCHAR	图片类簇 id，对 diff_img 进行聚类得到的类簇 id
index	INT	序号，图片在描述段落里的序号
diff_img	VARCHAR	图片，与主报告差异较大的图片

表 3.15 所示为 supplement_item 表的数据库表设计及相关字段说明。主要用于保存最终的“候选项”类簇合并结果和合并后 PageRank 计算结果，即最终聚合报告的补充点数据。

表 3.15 supplement_item 表

字段名	类型	描述
cluster_id	BIGINT	类簇 id，用于聚类唯一标识
master_id	BIGINT	主报告 id，格式与 bug id 相同
bug_id	BIGINT	bug 报告 id，所属原始报告 id
sup_id	VARCHAR	合并类簇 id，对 diff_text 和 diff_img 进行合并之后的最终结果，被合并的类簇 id 会发生改变
index	INT	序号，候选项在报告里的序号，如句子在描述段落里的序号，或者图片的序号
content	VARCHAR	候选项，可能是句子或者图片地址
pr_score	FLOAT	PageRank 分数，每一个补充点聚类里各项补充点的排序结果

3.8 审核交付模块设计

3.8.1 架构设计

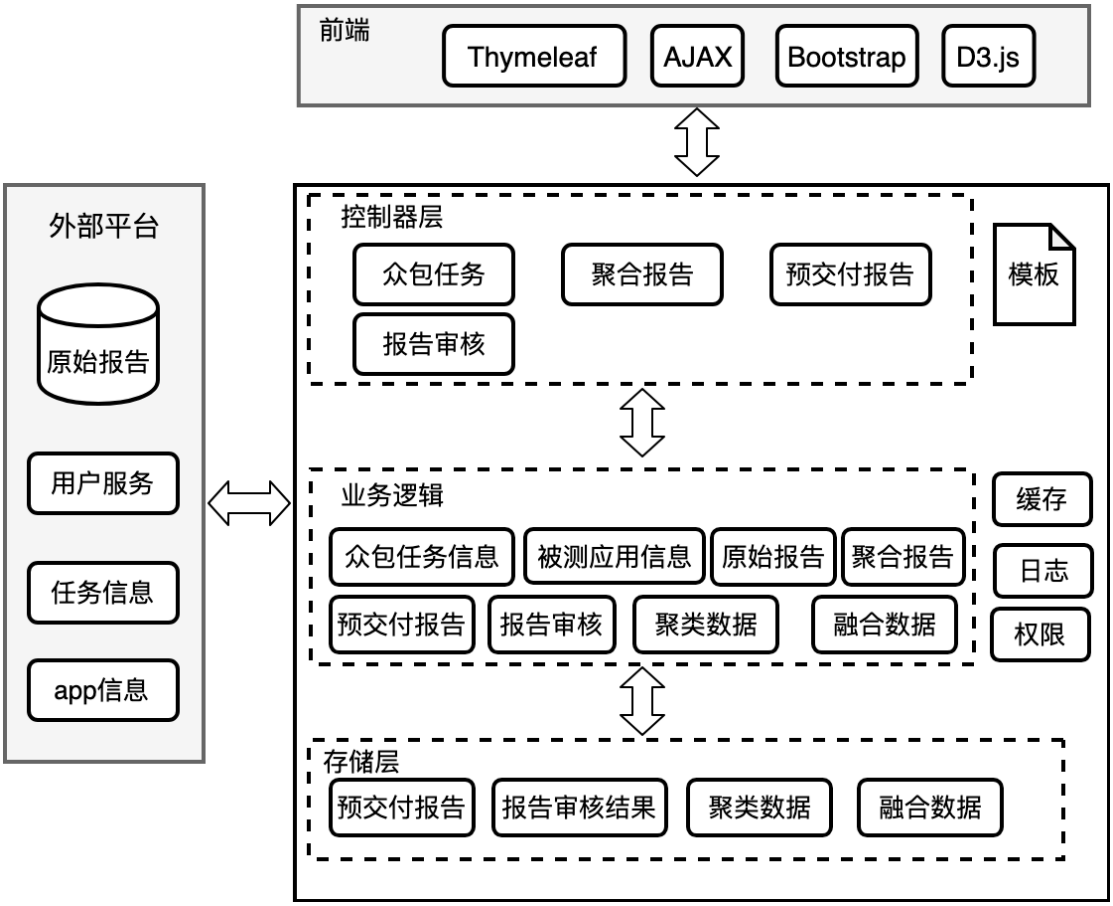

图 3.20 审核交付模块架构图

审核交付模块架构设计如图 3.20 所示。该模块的主要功能是承担审核交付业务的相关业务逻辑。因为本系统是 B/S 架构，因此总体设计按照 MVC 的模式开发。控制器层主要处理来自外部的请求，决定返回的数据或页面。模板控制前端页面展示逻辑的变化和用户交互逻辑的变化。业务逻辑层则负责面向更细粒度的业务规则和细节处理，包括业务数据的多样化查询、数据的拆解和封装、业务规则的执行等。存储层则面向数据库进行数据的存取。

前端渲染需要的模板文件由控制层管理。业务逻辑层除执行既定的业务规则外，还要完成日志收集、权限校验，以及对业务数据的缓存。权限验证主要基于外部系统提供用户认证服务。日志收集内容包括用户操作记录、错误日志和异常日志。错误日志数将被实时监控，超过阈值后监控系统将会报警。

3.8.2 流程设计

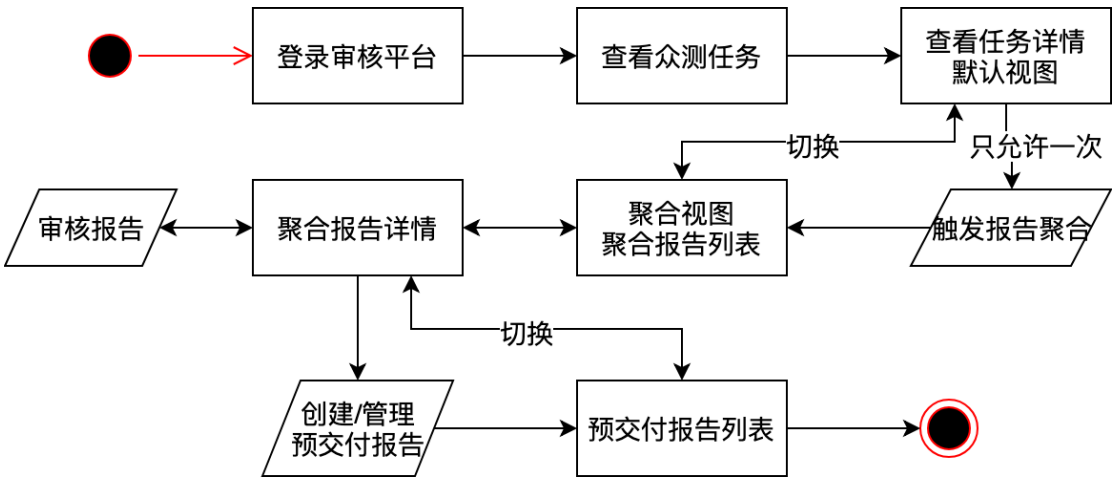

图 3.21 审核交付模块流程设计

审核交付模块的流程设计如图 3.21 所示，主要从审核人员的角度描述审核的大致流程。

(1) 登录审核平台。

(2) 审核人员查看待审核的众测任务。

(3) 选中任务后查看任务详情，显示审核进度和原始报告。

(4) 触发报告聚合，完成后能打开聚合视图，查看聚合报告列表。

(5) 选中聚合报告进行审核，基于该报告创建最终要交付的报告，完成后确认审核，报告状态改变。

(6) 可以查看预交付报告列表，对不满意的可以进行修改或删除。

重复步骤 5 和 6 直至完成所有审核工作。

3.8.3 核心类图

审核交付模块核心类图如图 3.22 所示，主要承担审核交付业务的实现。

TaskController 接收并处理众包任务列表和详情页面相关的请求，包括获取任务列表、获取任务列表详情、报告审核进度等。TaskService 获取任务列表的基本信息和应用的基本信息，逻辑实现是先查询本地是否已经获取过相关信息，获取过就直接返回，否则访问外部服务进行获取。

AggController 接收并处理聚合报告列表和详情页面相关的请求，包括获取聚合报告列表、获取聚合报告详情、报告审核等。ClusterService 获取相关聚类数据。BugRreviewService 主要提供报告审核相关业务逻辑的实现，包括状态查询和标记报告为已审核状态等。BugReportService 获取原始报告的列表，同样是先查询本地，没有则访问外部服务获取。SupplementService 负责补充点数据的查询和封装。聚合关系的展示由 GraphController 控制，GraphService 负责构建前端图形可视化所需数据。预交付报告由 FinalController 控制，FinalService 负责交付报告的增删改查。

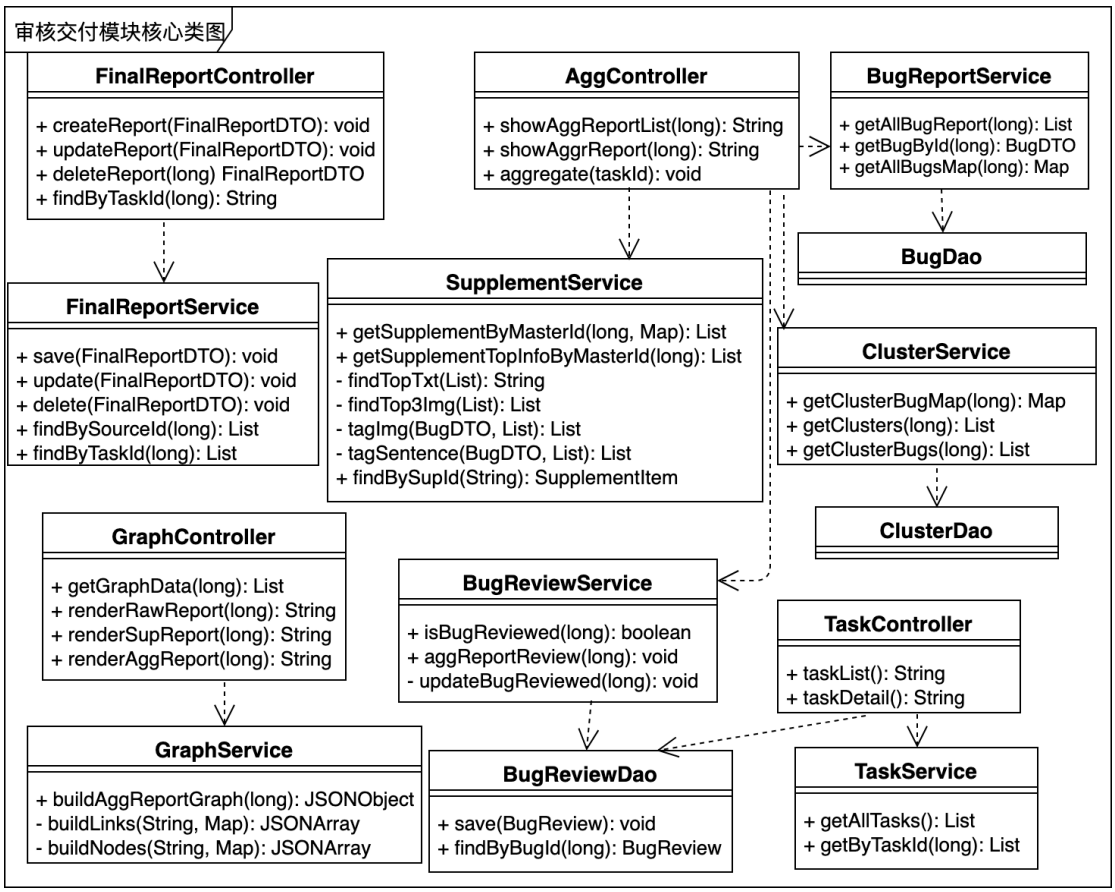

图 3.22 审核交付模块核心类图

3.8.4 数据库设计

审核交付模块的 ER 图如图 3.23 所示。final_report 表示预交付报告，bug_review 表示报告审核,user 表示用户信息，user 表存储在外部平台，task 表表示众测任务。final_report 与 task 的关系是多对一的关系，即一个任务会产生

多份预交付报告；cluster 和 final_report 是一对多关系，一个 cluster 会产生多份 final_report；bug_review 和 bug 是一对一的关系。

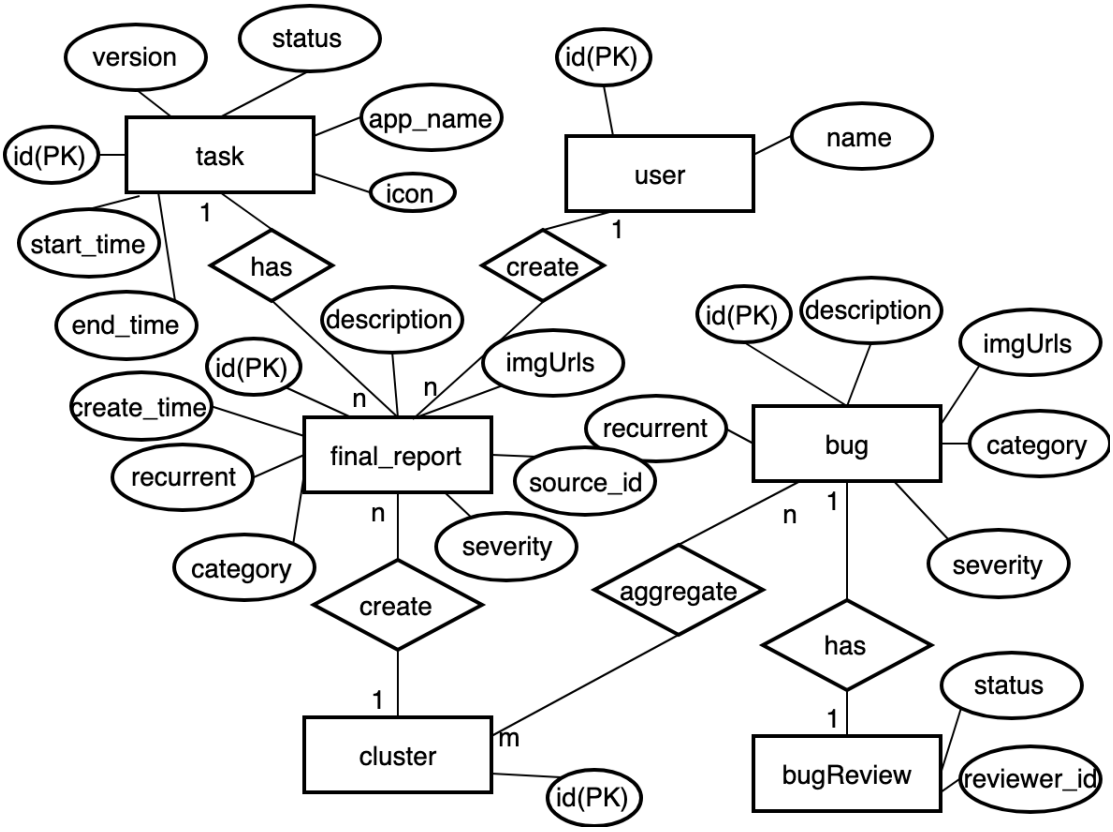

图 3.23 审核交付模块 ER 图

如表 3.16 所示是 final_report 的数据库表设计和字段说明，主要负责记录预交付报告的相关信息。为方便溯源，记录了任务 id、来源于哪个聚合报告、创建者 id 等。除此以外是报告的主体内容，包括描述、图片地址、严重性等。

表 3.16 final_report 表

字段名	类型	描述
id	BIGINT	预交付报告唯一 id
task_id	BIGINT	任务 id，对应众包任务的唯一标识
source_id	BIGINT	类簇 id，标明这份交付报告是由哪个聚合报告产生
creator_id	BIGINT	创建者 id，记录这份交付报告是由哪个用户创建
description	VARCHAR	bug 描述，报告中对 bug 的描述
img_urls	VARCHAR	图片地址，报告上传的图片存储地址，多张图片地址以逗号间隔
category	INT	bug 所属类型，具体类型与 bug 表相同
severity	INT	bug 严重性，具体类型与 bug 表相同
recurrent	INT	bug 可复现程度，具体类型与 bug 表相同

如表 3.17 所示是 bug_review 的数据库表设计和字段说明。主要负责 bug 报告审核状态的存储，因为考虑到 bug 表本身数据量大，表宽度较大，并且审核

状态单独查询的频率较高，因此把 bug_review 表单独拆分出来。通过单独拆分，系统降低了对 bug 表的查询频率，提升读写性能。

表 3.17 bug_review 表

字段名	类型	描述
bug_id	BIGINT	bug 唯一 id，bug 表主键
status	INT	报告审核状态
reviewer_id	BIGINT	审核人员 id

3.9 本章小结

本章首先对众包审核交付平台的功能、非功能需求及用户场景进行分析；其次分析系统整体架构和技术选型，包括模块划分和 4+1 视图；接着对测试报告自动化处理的总体设计进行概述；最后分别对报告处理的四个核心模块（文本处理模块、图片处理模块、聚合模块和融合模块）以及审核交付模块的架构、流程设计、数据库设计和核心类图进行详述。通过分析，为方便集成，系统采用服务化的方式部署；为提高松耦合特性，系统采用异步化、事件化的设计。为提高数据读取效率，系统合理利用缓存，并通过对数据库表的垂直拆分来降低数据库访问压力。

第四章 众包审核交付系统的实现

本章在第三章系统需求和设计的基础上，将对各个模块的实现细节进行描述，以顺序图来描述关键组件的调用过程，并展示系统模块实现部分的关键代码，最后展示系统的实现界面。

4.1 文本处理模块的实现

4.1.1 顺序图

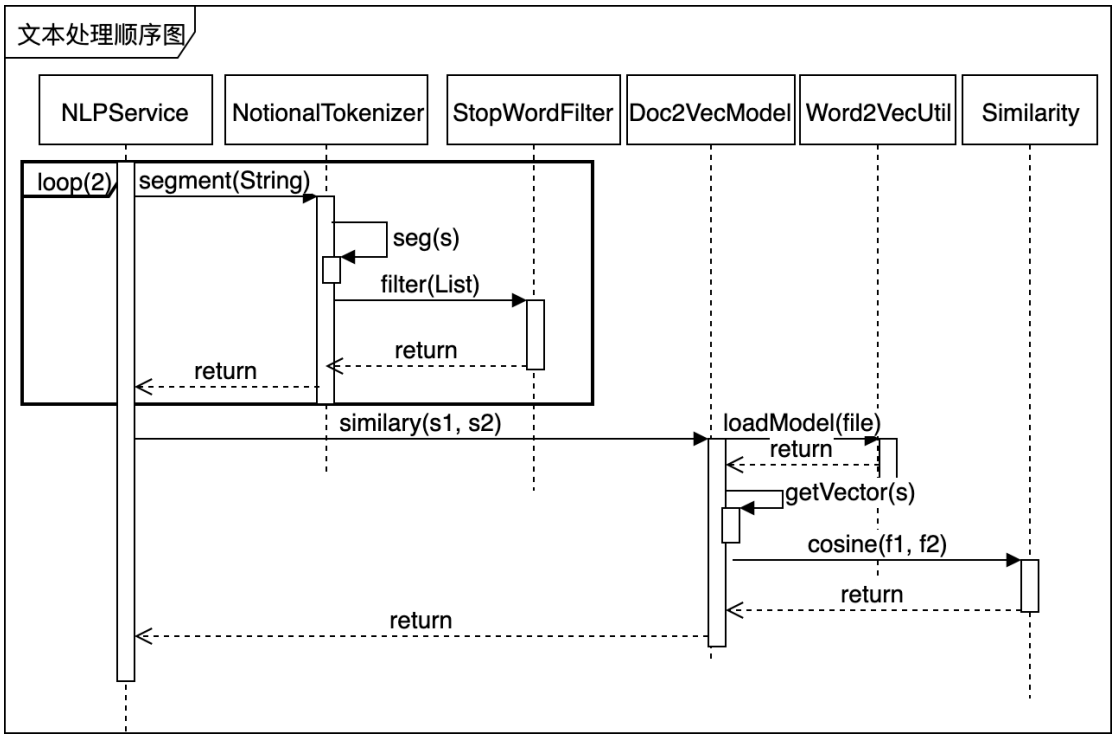

图 4.1 文本相似度计算顺序图

如图 4.1 所示是文本处理模块在计算两个文本相似度时的调用顺序，也是本模块最为核心的处理过程。在接收到两个需要计算相似度的文本时，NLPService 会调用 NotionalTonkenizer 类进行分词处理，分词完成后会调用 StopWordFilter 类的 filter 方法，从磁盘加载或从内存读取停用词表，过滤分词结果中的停用词，并将最终结果返回。

接下来会调用 Doc2VecModel 的 similary 方法计算相似度，具体步骤是 Doc2VecModel 会调用 Word2VecUtil 加载 Word2VecModel（加载文件只会在

第一次发生，第一次加载成功后每次都是从内存读取），然后获取文本的向量，调用 Similarity 计算两个向量的余弦相似度并返回结果。

4.1.2 关键代码

如图 4.2 的代码所示，主要展示图 4.1 中 NLPService 的 similar 方法，描述的是文本相似度的主要计算过程。

```
public float similar(String what, String with) {
        Vector a = query(what); //获取句 a 的特征向量
        if (a == null) return -1f;
        Vector b = query(with); //获取句 b 的特征向量
        if (b == null) return -1f;
        return Similarity.cosine(a. getElementArray(), b. getElementArray());
}
private Vector query(String s) {
        WordVectorModel wordVectorModel = Word2VecUtil.loadModel();
        if (content == null || content.length() == 0) return null;
        List<Term> termList = NotionalTokenizer.segment(content);
        //创建 Vector 对象保存词向量，维度要与 model 的维度一致
        Vector result = new Vector(wordVectorModel.dimension());
        int n = 0;
        for (Term term : termList) {
            Vector vector = wordVectorModel.vector(term.word);
            if (vector == null) {
                continue;
            }
            ++n;
            result.addToSelf(vector); //对每个词的词向量相加取平均，最终构成句向量
        }
        if (n == 0) {
            return null;
        }
        result.normalize();//向量归一化
        return result;
}
```

图 4.2 文本处理模块代码

NLPService 的 similar 方法会调用 query 方法分别获取两个文本的特征向量，如果获取失败，会返回一个默认值-1，表示计算失败。query 会通过 Word2VecUtil 加载词向量模型，如果加载失败，返回空值。加载成功后，会先进

行分词，并获取词数组中每个词的词向量。为获得句向量，会将所有词向量相加取平均，并进行归一化处理，最终结果作为文本特征向量。

4.2 图片处理模块的实现

4.2.1 顺序图

如图 4.3 所示是图片处理模块进行图片下载和特征抽取、保存的调用流程，也是本模块的核心，图片相似度的计算就是基于此而完成的。下面分析图中调用过程。ImgProcessService 调用 ImgDownload 进行图片下载，ImgDownload 通过 BugReportService 获取所有报告，从而获得原始图片的地址。然后通过 downloadImg 方法下载所有图片，下载完成后通过 FeatureExtractor 调用 CEDD，使用 LIRE[7] 的算子对图片流进行特征抽取，结果利用 FileUtil 保存到本地。

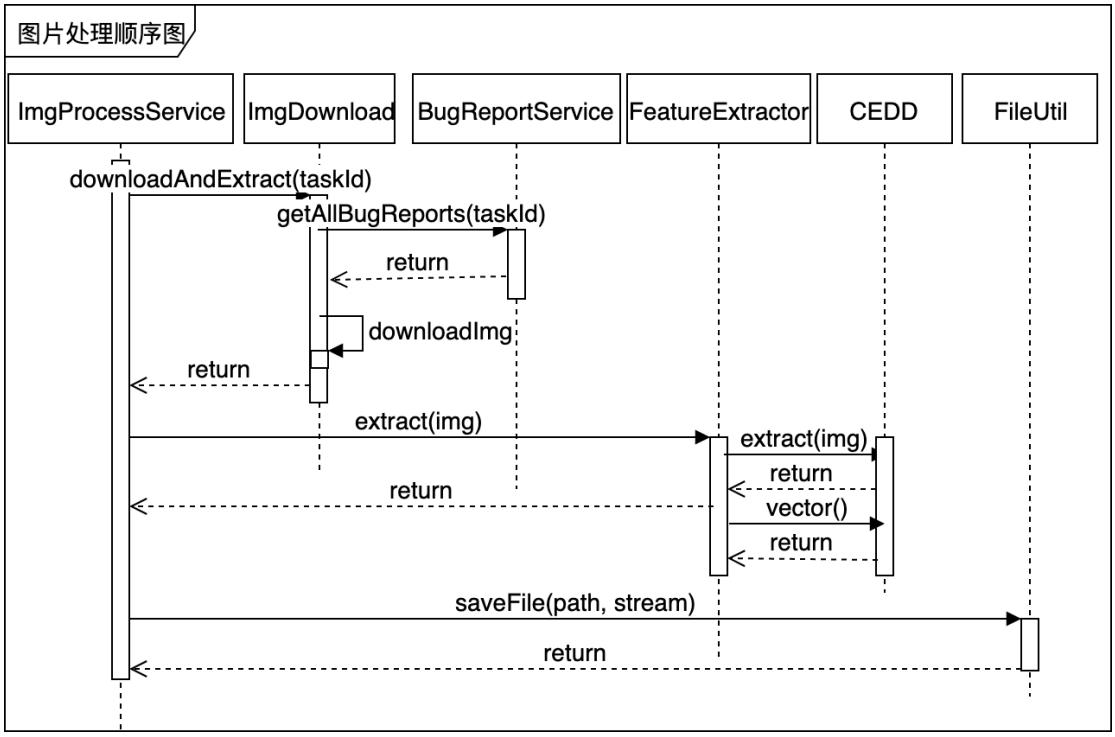

图 4.3 图片下载和特征抽取顺序图

4.2.2 关键代码

如图 4.4 所示，主要展示图 4.3 中描述的图片地址获取、图片下载、特征抽取的业务逻辑，为不影响阅读省略了一些非关键代码。图片下载是通过 URL 对

象获取图片流，并将图片流传给特征描述符对象。特征抽取的算法是通过 CEDD 等全局特征的子类来实现的，各个子类具体通过 LIRE[7] 框架提供的算子来实现。

```
public void downImg(long taskId) {
        … // 获取所有 bug 报告代码省略
        bugs.forEach(bug -> {
            if (bug.getImgUrls() != null && bug.getImgUrls().length > 0) {
                String[] imgUrls = bug.getImgUrls();
                … // 校验代码省略
                String fileName = bug.getId() + "_" + i;
                ImageDownload.createImage(imgUrls[i], fileName);
            }
        });
}
public static void createImage(String imgurl, String filePath) throws Exception {
        URL url = new URL(encode(imgurl, "utf-8"));
        BufferedImage image = ImageIO.read(url); // 下载图片
        GlobalFeature f = new CEDD();
        f1.extract(img); // 抽取特征
        float[] v = f.getFeatureVector(); //获取特征向量
        …//存储特征代码省略
}
```

图 4.4 图片下载和特征抽取代码

4.3 聚合模块的实现

4.3.1 顺序图

如图 4.5 所描述的是聚合模块的主要调用过程。聚合模块入口是 Aggregator 的 aggregate 方法，通过 BugReportService 类获取所有原始报告，具体使用 RestTemplate 类模拟 HTTP 请求访问公司平台接口。

Aggregator 通过调用 DistanceMatrix 类的 getHybridDistMatrix 方法分别触发文本处理模块和图片处理模块的预处理和相似度计算。具体地，通过调用 NLPService 对报告文本描述分词、清洗和提取特征，并计算相似度；通过调用 ImageProcessService 进行图片下载、特征抽取、相似度计算；最后，通过调用 DistanceMatrix 类对两种计算结果以取平均的方式进行合并。在实际执行过程中，为节省内存空间，会返回相似矩阵的下三角矩阵。

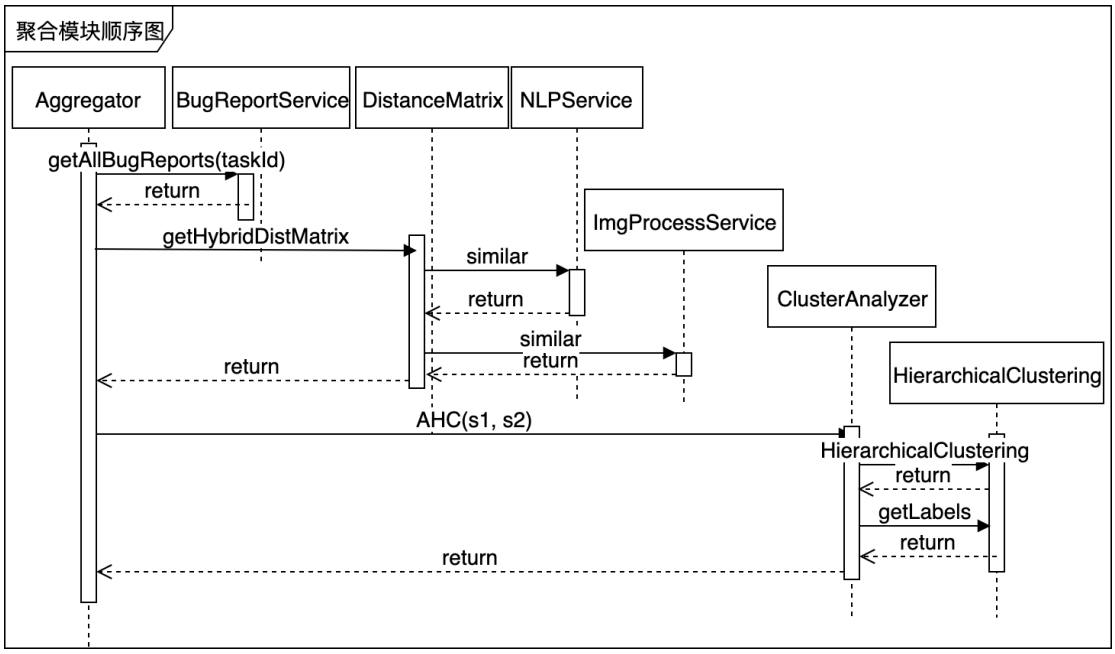

图 4.5 聚合模块顺序图

获取到相似矩阵的下三角矩阵之后，调用 ClusterAnalyzer 进行聚类。具体步骤是调用 HierachricalClustering 进行计算，HierachricalClustering 会调用 WardLinkage 类作为类簇合并的算法。最终，HierachricalClustering 会返回聚类结果，进行处理之后会存入到数据库中。

4.3.2 关键代码

如图 4.6 所示是聚合模块的核心调用流程，主要描述聚合模块主要的代码执行过程。首先获取所有报告及所有报告 ID。然后通过 DistMatrix 获取报告相似度矩阵，结果是下三角矩阵压缩的一维数组。最后调用 ClusterAnalyzer 对象进行聚类。

```
public void aggregate(long taskId) {
        … // 获取所有 bug 报告代码省略
        ClusterAnalyzer<String> analyzer = new ClusterAnalyzer<>();
        List<String> bugIds = bugs.stream().map(BugDTO::getId).collect(Collec-
                tors.toList());
        // 获取距离矩阵，返回值是压缩成一维的下三角矩阵
        float[] distMatrix = DistMatrix.genHybridDist(bugs);
        // 聚类，参数分别是距离矩阵、报告 ID 和聚类树的高度
        List<Set<String>> clusters = analyzer.AHC(distMatrix, bugIds, 0.23);
}       // 存储聚类结果部分代码省略
```

图 4.6 聚合模块核心调用代码

如图 4.7 所示是凝聚式层次聚类算法的核心调用流程，主要描述凝聚式层次聚类算法主要的代码执行过程，参数是抽象类 Linkage，实际参数传入具体使用的类簇间距离度量算法，如 WardLinkage。

```
public HierarchicalClustering(Linkage linkage) {
        int n = linkage.size();
        // 初始化
        merge = new int[n - 1][2];     // 合并过程数组
        int[] id = new int[n];          // 类簇 id 编号
        height = new double[n - 1];   // 层次聚类高度记录
        int[] points = new int[n];       //当前集合中的所有点
        for (int i = 0; i < n; i++) {       //初始化数组的值
            points[i] = i;
            id[i] = i;
        }
        FastPair fp = new FastPair(points, linkage); // 用于构建邻近度矩阵
        for (int i = 0; i < n - 1; i++) {
            height[i] = fp.getNearestPair(merge[i]);
            linkage.merge(merge[i][0], merge[i][1]);       // 合并最近邻的类簇
            fp.remove(merge[i][1]);                 // 去除被合并的那个类簇
            fp.updatePoint(merge[i][0]);         // 更新邻近度矩阵，因为合并后类簇之间的邻近度关系发生了改变
            int p = merge[i][0];   //记录合并的类簇 id
            int q = merge[i][1];
            merge[i][0] = Math.min(id[p], id[q]);   //将较小 id 放在前
            merge[i][1] = Math.max(id[p], id[q]);
            id[p] = n + i;
        }
        …//后续处理省略
}
```

图 4.7 凝聚式层次聚类算法核心代码

4.4 融合模块的实现

4.4.1 顺序图

如图 4.8 所示是融合模块的核心调用顺序图，描述了融合模块和各个类的核心交互过程。

由图中可以看出，融合模块依旧由 Aggregator 触发，在聚合模块计算结束后，Aggregator 会调用 MasterReportService 计算选出主报告。具体地，

MasterReportService 调用 GraphService，通过 buildDirectedGraph 方法构建图，节点是报告，边的权重则是报告之间的相似度，相似度数据来自于聚合模块给出相似矩阵，这里使用 JGraph 组件实现。接着 GraphService 会调用 PageRank 类计算得到每个节点的权重，权重最高者是主报告。MasterReportService 会将相关结果存入数据库。

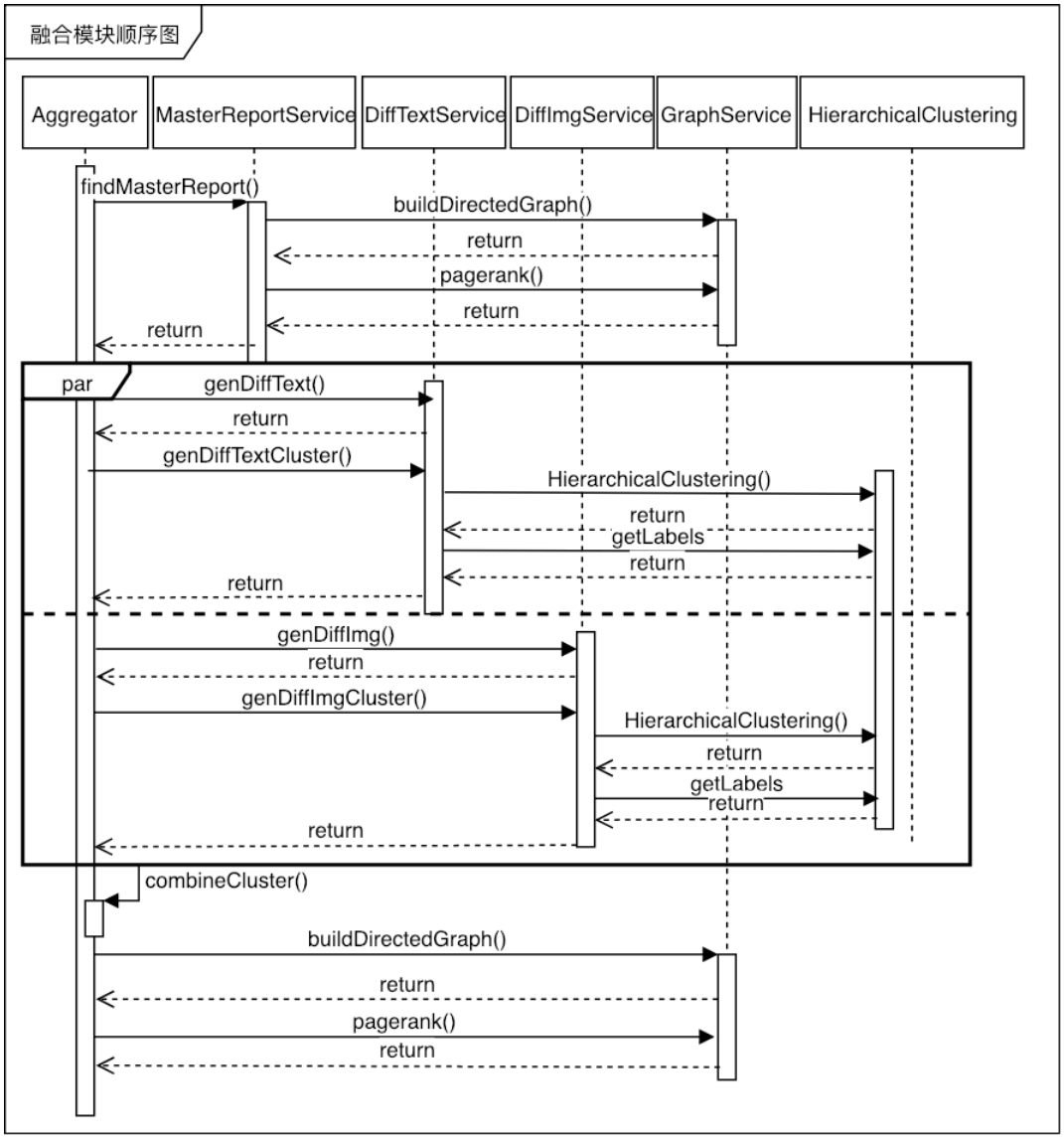

图 4.8 融合模块顺序图

接着 Aggregator 会并行调用 DiffTextService 和 DiffImgService 对主报告和同一个类簇的其他报告的句子、图片进行依次比较，找出差异较大的。这里会用到文本处理模块和图片处理模块进行分句和相似度计算。考虑到拆分的句子和图片会有一定的重复，得到结果后会调用聚合模块进行聚类，得到句子和图片的类簇。Aggregator 会调用 combineCluster 方法对相关度较高的类簇进行合并。

最后会再次调用 GraphService 分别对各个类簇进行 PageRank 的计算，对各个“补充项”进行排序，以找出比较有代表性的“补充项”。结果存入数据库。

4.4.2 关键代码

```
public void aggregate(long taskId) {
        … // 非融合模块部分的代码省略
        for (Set<String> cluster : clusters) {
            // 计算类簇的主报告
            String masterReport = masterReportService.findMasterReport(cluster, bug-
Map);
            // 保存主报告
            masterReportService.saveMasterReport(masterReport, cluster);
            //缓存主报告
            masterClusterMap.put(masterReport, cluster);
        }
        // 计算比较主报告与类簇的其他报告的差异文本，并存储
        Map<String, List<DiffText>> masterDiffTextMap = diffTextService.genMaster-
            DiffTextMap(masterClusterMap, bugMap);
        // 计算比较主报告与类簇的其他报告的差异图片，并存储
        Map<String, List<DiffImg>> masterDiffImgMap = diffImgService.genMaster-
            DiffImgMap(masterClusterMap, bugMap);
        // 对差异文本进行聚类，并存储结果
        Map<String, List<Group<String, DiffText>>> masterDiffTextClustersMap = dif-
            fTextService.genDiffTextClusters(masterDiffTextMap);
        // 对差异图片进行聚类，并存储结果
        Map<String, List<Group<String, DiffImg>>> masterDiffImgClustersMap =
            diffImgService.genDiffImgClusters(masterDiffImgMap);
        //合并差异文本和差异图片的类簇
        combineCluster(masterDiffTextClustersMap, masterDiffImgClustersMap);
        //对最终类簇项的差异文本进行排序
        supplementService.rankAndStoreDiffText(masterDiffTextClustersMap);
       //对最终类簇项的差异图片进行排序
       supplementService.rankAndStoreDiffImg(masterDiffImgClustersMap);
}
```

图 4.9 融合模块核心流程代码

如图 4.9 是融合模块的核心调用流程，是 aggregate 方法中涉及到融合模块计算流程的部分，具体描述的是图 4.8 中展示的调用过程。MasterReportService 通过循环计算每个 cluster 对应的主报告，并及时将结果存储到数据库中。DiffTextService 和 DiffImgService 会计算每个 cluster 里主报告的差异点，包括

句子和图片，并对差异点进行聚合。接下来会对文本和图片的差异点类簇进行合并，形成补充点，如 combineCluster 方法所示。最后 SupplementService 会将结果进行保存。

4.5 审核交付模块的实现

4.5.1 查看众包任务的实现

查看众包任务主要包括查看众包任务列表和查看众包任务详情两个用例，涉及到两个界面。如图 4.10 所示是查看众包任务列表的界面，主要展示的是各个任务的时间，待测应用的图标和名字以及报告审核进度。未完成的审核任务将会被放置在列表的顶部，方便审核人员快速查找到。

图 4.10 众包任务列表界面

如图 4.11 所示是查看众包任务详情的界面，页面上方主要是报告审核进度的 Dashboard，蓝色代表报告总数，绿色代表已审核数，红色代表未审核数，并有相应进度条展示。下方主要是原始报告列表信息。报告列表除了基本信息外，还有审核状态和审核人，如果触发了报告聚类，还会显示原始报告所属的聚类报告。为了便于查看，列表采用分页展示，并且支持多级别分页。左侧是用户状态和快捷导航，方便用户快速浏览预交付报告和已审核、未审核的报告。

图 4.11 众包任务详情界面

4.5.2 异步触发的实现

本节主要介绍异步机制的总体实现，异步机制基于事件监听和回调的方式来处理任务，不同于同步调用，可以实现代码解耦，提高可修改性和可扩展性，并在一定程度上提高效率。代码中 EventUtil 是操作 EventBus 的工具类，提供注册监听器和触发事件的功能。事件相关的数据通过事件对象来进行传递。具体细节如图 4.12 所示。

```
@Component
public class EventUtil {
    private static Logger logger = LoggerFactory.getLogger(EventUtil.class);
    private EventBus eventBus;
    public EventUtil() { //初始化异步线程池
        eventBus = new AsyncEventBus(Executors.newCachedThreadPool());
        eventBus.register(this);
    }
    //注册事件回调函数
    public void register(Object listener) { eventBus.register(listener);}
    public void post(Event event) { //用于业务代码触发事件
        logger.info("event fired: {}", event.getDescription());
        eventBus.post(event);
    }…//省略其他代码
}
```

图 4.12 异步工具类实现代码

如图 4.13 所示是事件监听器的具体实现。EventListener 类是一个是事件监听器的实现，通过实现 InitializingBean 的生命周期函数来注册进 EventBus，通过注解@Subscribe 来标记回调函数。

```
@Component
public class EventListener implements InitializingBean {
    @Autowired
    private EventUtil eventUtil;
    @Override
    public void afterPropertiesSet() throws Exception {
        eventUtil.register(this); //将监听器注册到 EventBus
    }
    // 订阅自动化报告处理结束事件，事件回调函数如下
    @Subscribe
    public void updateTaskResult(AggregateDoneEvent e)
    throws Exception {
        // 省略具体的业务实现
    }
}
```

图 4.13 异步事件监听实现代码

4.5.3 查看和审核聚合报告的实现

图 4.14 聚合报告列表界面

聚合列表界面如图 4.14 所示，每个方块代表一个聚合报告，右上角显示有审核状态，点击按钮可以查看被聚合的报告列表。方块下方显示的是主报告的内

容，方便审核人员快速了解各个报告的主题内容。左侧是快速导航和用户状态，可以快速过滤不同审核状态的报告。

聚合报告详情页面如图 4.15 所示。界面中间从上至下依次是报告的严重性、报告分类等基本信息。中间是主报告的描述信息和图片，点击图片可放大查看。下方是补充点信息框，展开可以看到原始报告，和被标红的补充点。界面左侧是快速导航，能查看聚合视图、默认视图 和预交付报告。右侧上方是一个词云图，主要提取该聚合报告的文本信息依据词频生成，作用是帮助用户快速理解报告主题。右方是基于此聚合报告创建的预交付报告列表。

图 4.15 聚合报告详情界面

图 4.16 是聚合报告详情界面的一部分，展示内容是聚合关系的可视化，目的是为直观地展示聚合报告的层次关系，使审核人员理解原始报告经过怎样的处理得到的聚合报告。底部黄色圆点代表原始报告，可以看到 R73 同时出现在补充点 1 和 2，原因是一份原始报告的不同部分体现了不同的内容，因此被聚合到两个补充点。中间蓝色圆代表补充点，顶部橙色大圆代表聚合报告，连线表示聚合关系。其中每个节点点击，右侧都会有对应的内容描述生成，并且描述框顶部有对应的颜色，使其对应关系更加明显。

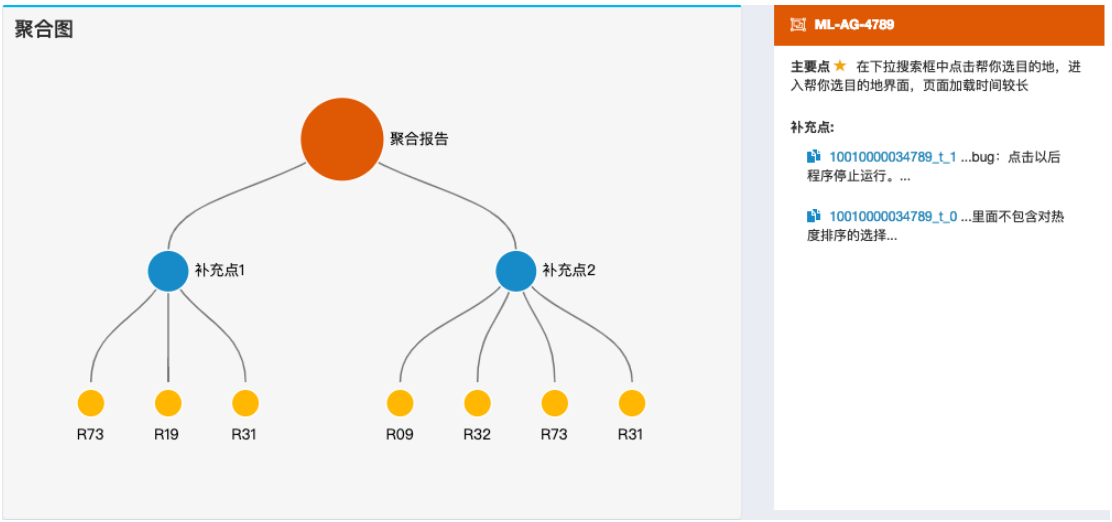

图 4.16 聚合关系可视化界面

4.5.4 预交付报告管理的实现

图 4.17 预交付报告列表界面

如图 4.17 所示是预交付报告列表界面。列表展示了基本的报告信息，除此之外还有每份报告所属的聚合报告和创建人，点击报告 ID 可以直接打开对应聚合报告详情页面。右侧是操作栏，可以修改和删除不满意的报告。

报告编辑界面如图 4.18 所示，可以填写 Bug 分类、严重性、可复现程度、描述和截图，截图可以通过拖拽来增加和删除，点击可放大查看图片细节。编辑框右上角是保存和取消按钮，点击保存会提示结果并刷新列表，点击取消则恢复原状。

图 4.18 预交付报告编辑界面

4.6 本章小结

本章主要对众包审核交付系统的实现细节进行详述。首先，详述文本处理模块的相似度计算调用顺序图和对应代码实现。其次，详述图片处理模块的下载和特征抽取顺序图和对应代码实现。然后，在聚合模块中对计算顺序和聚类算法的主要实现都进行了详述。接着，对融合模块的调用顺序图和主要流程的代码实现进行详述。最后，详述审核交付模块在各个用例需求场景下的界面和每个界面对应的设计思路，以及异步触发的代码实现。

第五章 众包审核交付系统的测试

本章将对系统的功能实现、服务器水平扩展能力和数据库主备切换能力进行测试。首先描述测试准备，包括本次测试目标和测试环境，然后分别展示功能测试、水平扩展测试和数据库主备测试的用例设计和执行结果。

5.1 测试准备

5.1.1 测试目标

本系统的软件测试主要针对两方面来进行，一方面是验证软件是否满足需求，进行有效性测试；另一方面是验证软件质量是否过关，尽可能减少缺陷，进行缺陷测试。

根据第三章对众包审核交付系统的需求分析和架构设计，本章的系统测试将包括以下三个部分：

(1) 功能测试。这部分主要检查系统运行是否满足业务需求，确保最终产品符合规格，能够较好支持审核人员完成众包审核交付任务。

(2) 水平扩展性测试。这部分主要检查系统增加服务器时，负载是否会线性下降，以证明系统具备一定的水平扩展能力。

(3) 数据库主备切换测试。这部分主要检查数据库主备切换的能力，以应对数据库宕机的情况。

5.1.2 测试环境

表 5.1 硬件和系统环境

服务	硬件环境
Web 服务	阿里云服务器，型号 ecs.sn1ne.xlarge，Ubuntu16.04，2 台
数据库服务	阿里云服务器，型号 ecs.c1.large，Ubuntu16.04，2 台
负载均衡服务	阿里云服务器，型号 ecs.c1.large，Ubuntu16.04，1 台
缓存服务	阿里云服务器，型号 ecs.s3.large，Ubuntu16.04，1 台

如表 5.1 所示是部署系统各个服务需要的硬件资源和操作系统版本。本系统部署于阿里云服务器，使用 Linux 作为运行环境。其中，Web 服务使用两台服务

器是为了测试系统的负载均衡和水平扩展性；数据库服务使用两台服务器是为了测试主备切换的效果。

表 5.2 展示了各个服务具体应该部署的软件及版本。Web 服务采用较为稳定的 Tomcat 作为底层运行容器，并使用 Maven 管理依赖和打包部署。数据库需要部署 Keepalived 软件来实现主备自动切换。

表 5.2 部署软件

服务	部署软件
Web 服务	JDK 1.8.0_66、Tomcat 8.0.53、Maven 3.3.9
数据库服务	MySQL 5.7.16、Keepalived
负载均衡服务	Nginx 1.10.2
缓存服务	Redis 5.0.3

图 5.1 展示了 Nginx 关于负载均衡部分的配置内容，其含义是监听来自 80 端口的请求，并将请求转发给虚拟上游服务器 crowd_review，然后通过轮询来把请求发送给两台具体的服务器。

```
http {
    # ... 省略其它配置
    upstream crowd_review {
        server 192.168.0.100:8080; # 可替换真实 ip 和端口
        server 192.168.0.101:8080;
    }
    server {
        listen 80; # 监听 80 端口请求
        location / { # 匹配所有请求
            # 转发给 crowd_review
            proxy_pass http://crowd_review;
        }
    }
    # ... 省略其它配置
}
```

图 5.1 Nginx 负载均衡配置

图 5.2 以 YAML 格式展示了数据库连接池的相关配置，主要配置了连接池的初始值、最大活跃连接数、最小空闲连接数、最大连接等待时间等。配置连接池的目的一方面是让数据库连接不会每次使用完后被销毁，而是重复利用，极大减少了资源开销，提高了效率；另一方面也可以对数据库连接进行监控，发现慢查询，及时优化数据库查询。

```
spring:
  datasource: //数据源
    driver-class-name: com.mysql.jdbc.Driver //驱动名称
    druid:
      initial-size: 10 //初始大小
      max-active: 80 //最大活跃连接数
      minIdle: 10
      maxWait: 60000 #配置获取连接等待超时的时间
      #配置检测连接是否活跃的间隔时间，单位是毫秒
      timeBetweenEvictionRunsMillis: 60000
      validationQuery: SELECT 'x'
      poolPreparedStatements: true
      #配置连接的最小生存时间，单位是毫秒
      minEvictableIdleTimeMillis: 300000
      maxPoolPreparedStatementPerConnectionSize: 20

      # 配置 druid 监控统计，'wall'用于防火墙
      filters: stat,wall,log4j
      #打开 mergeSql 功能和慢 SQL 记录
      connectionProperties: druid.stat.mergeSql=true;druid.stat.slowSqlMillis=5000
# ... 省略其它配置
```

图 5.2 数据库连接池配置

图 5.3 展示的是数据库主要配置，配置内容包括端口、socket 文件地址、用户名、相关日志、数据的目录等。除了 server-id 每台数据库不同外，其他配置均相同。

```
[client]
port                = 3306
socket              = /data/3306/mysql.sock
[mysqld]
user      = mysql       #根据真实情况替换
port      = 3306
socket    = /data/3306/mysql.sock
basedir = /app/mysql
datadir = /data/3306/data
log-bin = /data/3306/mysql-bin # binlog 目录
server-id = 6             # server id  不能相同
skip_name_resolve = 0 # 跳过域名解析参数
[mysqld_safe]
log-error=/data/3306/mysql_3306.err # 错误日志
pid-file=/data/3306/mysqld.pid        # 进程 id 存放地址
```

图 5.3 数据库部分配置

5.2 功能测试

5.2.1 测试设计

本小节依据第三章需求分析得到的功能列表进行测试用例设计，面向功能测试，尽量保证覆盖各项功能需求，目标是确保最终产品符合规格，系统行为符合产品预期，能够较好地支撑众包审核交付业务。

表 5.3 展示了查看众包任务列表的测试过程和预期结果，主要针对用例描述 UC1 的功能进行设计，确保覆盖任务列表的各项功能细节。

表 5.3 查看众包任务列表测试用例

测试 ID	TC1
测试名称	查看众包任务列表
测试功能	审核人员可查看、搜索、排序所有众包任务，待审核任务置顶，显示任务基本信息和审核进度。
测试步骤	1. 审核人员登录； 2. 查看未审核结束的任务是否置顶； 3. 点击表头排序按钮两次，观察是否依次升序、降序； 4. 点击搜索框搜索“途牛”； 5. 鼠标移到进度条； 6. 点击进度条。
预期结果	1. 未审核任务置顶； 2. 依次升序、降序； 3. 搜索结果显示为“途牛旅游”的所有应用信息； 4. 显示未审核的报告数； 5. 调整任务详情页； 6. 跳转众包任务详情页。

表 5.4 展示了查看众包任务详情的测试过程和预期结果，测试的是系统展示众包任务详情的细致程度，主要关注点是显示的正确性和列表的基础查询功能（排序、检索等）。

表 5.4 查看众包任务详情测试用例

测试 ID	TC2
测试名称	查看众包任务详情
测试功能	审核人员可查看审核进度，可查看、检索、排序所有原始报告，显示报告的基本信息。
测试步骤	1. 查看 Dashboard 是否正确显示审核进度； 2. 点击表头排序按钮两次，观察是否依次升序、降序； 3. 点击描述列，是否可以查看描述详情。

预期结果	1. 正确显示审核进度； 2. 依次升序、降序； 3. 能展开查看详情。

表 5.5 展示触发报告聚合测试用例设计，测试的是系统是否能正确触发测试报告自动化处理，主要关注点是系统处理过程中是否会禁止重复点击，计算完成后是否会及时更新等细节。

表 5.5 触发报告聚合测试用例

测试 ID	TC3
测试名称	触发报告聚合
测试功能	审核人员可触发报告聚合，处理结束后显示聚合视图入口。
测试步骤	1. 点击“触发报告聚合”按钮，是否灰显，并提示“聚合中…”； 2. 计算完成后是否显示“聚合视图”按钮，是否隐藏“触发报告聚合”按钮； 3. 原始报告“所属聚合报告”列是否显示聚合报告 ID； 4. 点击聚合报告 ID 是否可跳转对应聚合报告页面。
预期结果	1. 按钮灰显，并显示“聚合中…”； 2. 显示“聚合视图”按钮； 3. 显示聚合报告 ID； 4. 能跳转对应聚合报告页面。

表 5.6 展示聚合报告列表的测试用例设计，测试的是系统除了展示聚合列表之外，是否具备简单过滤功能和一些细节展示（摘要等），帮助审核人员在查看时能获得更多的帮助，提高效率。

表 5.6 查看聚合报告列表测试用例

测试 ID	TC4
测试名称	查看聚合报告列表
测试功能	审核人员可查看聚合报告列表，查看、筛选报告是否已审核，查看聚合报告的基本信息。
测试步骤	1. 查看聚合报告方块是否显示报告 ID、聚合报告数、报告摘要内容； 2. 点击图片是否能够放大查看； 3. 点击列表按钮是否能查看被聚合的原始报告列表； 4. 点击左侧“已审核报告”，是否只显示已审核的报告； 5. 点击左侧“未审核报告”，是否只显示未审核的报告； 6. 点击报告 ID 是否能跳转对应聚合报告详情页面。
预期结果	1. 正确显示报告 ID、聚合报告数、报告摘要内容； 2. 图片能够放大查看； 3. 能查看被聚合的原始报告列表； 4. 只显示已审核的报告； 5. 只显示未审核的报告； 6. 能跳转对应聚合报告详情页面。

表 5.7 展示聚合报告详情的测试用例设计，测试的是系统对聚合报告信息展示的丰富程度，主要关注系统对主要点和补充点的展示是否足够细致，对聚合关系的展示是否足够直观。

表 5.7 查看聚合报告详情测试用例

测试 ID	TC5
测试名称	查看聚合报告详情
测试功能	审核人员可查看聚合报告详情，查看聚合报告的基本信息，查看主要点、补充点，查看聚合关系。
测试步骤	1. 查看报告时间、严重性、分类是否正确显示； 2. 查看主要点显示是否正确、点击图片是否可以放大查看； 3. 点击补充点展开按钮，是否展开显示详情； 4. 查看补充点是否对“差异点”（句子、图片）标红，点击图片是否可以放大查看； 5. 查看聚合关系是否显示正确，点击节点是否可以查看细节； 6. 查看右上角的词云是否显示正确； 7. “已创建报告”没有报告时，是否显示“暂无”，有报告时是否显示预交付报告列表。
预期结果	1. 正确显示； 2. 显示正确，图片能够放大查看； 3. 可以展开； 4. 差异句子和差异图片均标红显示； 5. 显示正确，点击节点可查看细节信息； 6. 显示正确，按照词频显示不同大小； 7. 没有报告时显示“暂无”，有报告时显示报告列表。

表 5.8 展示了审核聚合报告的测试用例设计，由于功能较为简单，因此设计也比较简单，主要关注系统提示是否正确，状态更新是否及时、正确。

表 5.8 审核聚合报告测试用例

测试 ID	TC6
测试名称	审核聚合报告
测试功能	审核人员确认审核完毕后，可点击“已审核”按钮修改报告状态。
测试步骤	审核人员在任意聚合报告页面点击“已审核”按钮，系统是否提示成功，并修改状态。
预期结果	系统提示结果，并更新页面状态。

表 5.9 展示了预交付报告的测试用例设计，主要测试系统对预交付报告增删改查的便捷性，主要关注点是预交付报告在创建时操作是否简单，编辑、删除报告是否便捷，提示是否合理。

表 5.9 管理预交付报告测试用例

测试 ID	TC7
测试名称	管理预交付报告
测试功能	审核人员可在聚合报告页面和预交付报告列表页面，对预交付报告报告进行查询和修改（增删改）。
测试步骤	1. 在聚合报告详情页面点击“创建预交付报告”按钮，是否显示报告编辑页面； 2. 编辑页面是否显示 Bug 分类、严重性、复现程度，并且可以选择； 3. 编辑页面是否可以编辑 Bug 描述信息； 4. 编辑页面添加、删除图片是否可以通过拖拽完成； 5. 重复添加图片是否会去重； 6. 点击“保存”按钮是否成功； 7. 点击“取消”是否会恢复； 8. 在聚合报告详情页面或预交付报告列表页面，点击删除报告； 9. 在聚合报告详情页面或预交付报告列表页面，点击编辑报告； 10. 预交付报告列表页面，点击聚合报告 ID。
预期结果	1. 显示编辑页面； 2. 显示正确； 3. 可以编辑； 4. 通过拖拽即可增删图片； 5. 重复添加图片自动去重； 6. 提示成功，刷新列表； 7. 恢复原状； 8. 提示删除成功，刷新列表； 9. 跳转对应聚合报告详情页，进入编辑模式，并自动填充报告信息； 10. 跳转对应聚合报告页面。

5.2.2 测试执行

表 5.10 功能测试用例执行结果

测试用例 ID	对应用例描述	测试结果
TC1	UC1	通过
TC2	UC2	通过
TC3	UC3	通过
TC4	UC4	通过
TC5	UC5	通过
TC6	UC6	通过
TC7	UC7	通过

测试人员严格按照测试用例描述步骤，依次执行所有测试用例，并记录结果到表 5.10 中。可以看出执行结果全部通过，表明系统完成了需求分析里提炼的系统功能需求，被测系统行为符合产品功能设计。

5.3 水平扩展性测试

本小节的测试目标是检测 Web 服务器的水平扩展性，即随着服务器增加，每台服务器的平均负载能接近线性下降，以证明系统设计具备符合企业要求的性能和稳定性。

5.3.1 测试设计

本次测试的测试环境与功能测试相同。本次测试的基本思路是通过对比在一定并发量下，一台服务器和两台服务器的性能指标（CPU、内存等），来测试服务器是否具备一定的水平扩展性。为了数值更加明显，故选择有一定计算量的读接口进行测试，本次测试选择的是获取聚合报告详情的接口，该接口需要构建图数据，计算词云，给补充点做标记等。为了便于对比，本次测试要求两台服务器型号一样。具体测试步骤如表 5.11 所示。

表 5.11 Web 服务可扩展性测试用例

测试 ID	TC8
测试接口	获取聚合报告列表
测试工具	JMeter、nmon
测试方案	1. JMeter 设置 1000 请求数，30 秒循环一次，持续 10 分钟； 2. Nginx 配置两台服务器地址； 3. 先启动服务器 Server1，2 分钟后，再启动另一台服务器 Server2； 4. 5 分钟后关闭服务器 Server2； 5. 使用 nmon 观察并记录两台服务器数值变化。

5.3.2 测试执行

如图 5.4 所示，测试执行时间是 15:06:00 到 15:16:00 分。其中 15:08:00 启动 Server2，15:14:00 分时关闭 Server2。可以观察到 Server2 启动之前，Server1 的 CPU 负载时 40%左右；Server2 启动之后，两台服务器的负载维持在 24%左

右，下降超过 40%，接近线性下降。不过可以注意到两台服务器同时启动时，负载总和是超过单台服务器（大约 50%），其原因是每台服务器的操作系统需要消耗一定的 CPU。

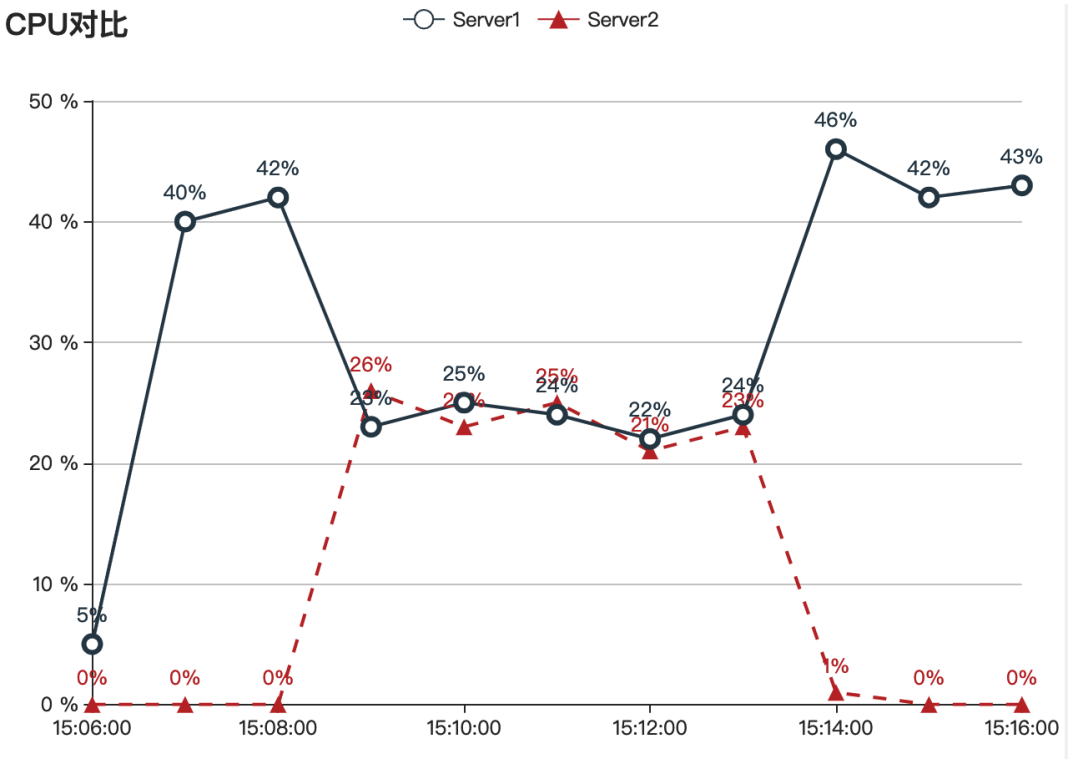

图 5.4 CPU 对比

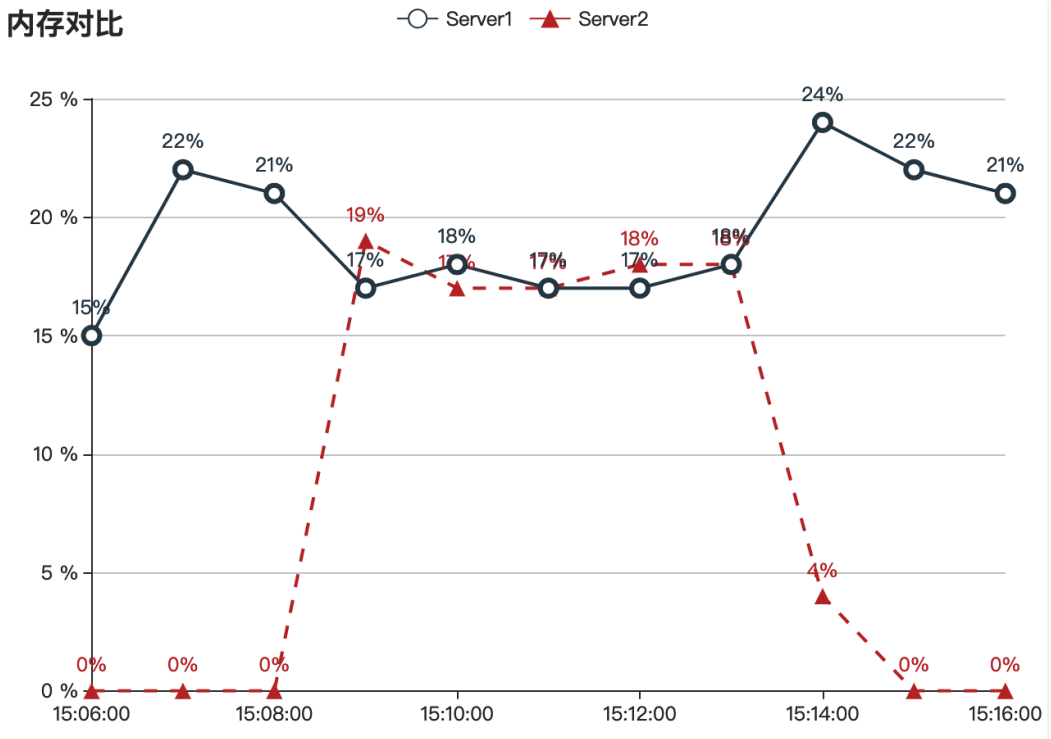

图 5.5 内存对比

图 5.5 的测试时间与图 5.4 相同，主要对比的是两台服务器的内存使用情况。服务器使用的是 4 核 8G 内存。从图中可以看出，在请求到达前，服务器约消耗了 15%的内存，原因是操作系统内核和 Java 进程占用了一定资源。在 Server2 启动前，单台服务器内存消耗约 22%。Server2 启动之后，两台服务器的内存消耗均为 18%左右。表面上看可能没有线性下降，但计算时去除掉操作系统内核和 Java 进程启动时占用的资源，并发请求带来的内存消耗是从 7%降低到 4%，约下降了 42%。Server2 关闭之后，Server1 又基本恢复之前的内存消耗。

经过上述性能指标的对比，可以观察到在 Server2 启动后，Server1 因为并发请求带来的 CPU 和内存消耗均下降了约 40%，接近线性下降，满足非功能需求中提到的水平可扩展性。同时考虑到目前系统主要提供企业内部使用，不具备高并发场景，目前的架构设计已足够应对系统的使用场景。

5.4 数据库主备切换测试

本小节的测试目标是检测系统中的数据库主备切换，即数据库发生宕机时（如进程被杀、服务器被重启），数据库访问能否切换到没有问题的节点上。

5.4.1 Keepalived 设置

Keepalived 用于在 Master 和 Slave 之间建立长连接，监控数据库状态。当 Master 失去响应时能及时执行相关脚本，依据权重将 Slave 转换成新的 Master，同时执行关闭旧 Master 等收尾工作。Keepalived 是实现主备自动切换的关键。图 5.6 展示的是 Master 节点的 Keepalived 的配置，图中对各项配置项含义进行了详细说明。

```
global_defs {
    router_id MYSQL-1 //表示运行 keepalived 服务器的一个标识
}
vrrp_script check_mysql {//  定义自动核对 mysql 的脚本
     script "/etc/keepalived/bin/check_mysql.sh"
     interval 22
     weight 2
}
vrrp_instance VI_1 {
     state BACKUP //指定角色,此处配置的角色是是 BACKUP，BACKUP 将根据优先级
```

```
    决定主或从
    interface eth1 //配置所要监测的网络接口
    virtual_router_id 10 /配置虚拟路由标识，如果是同一个 vrrp 实例必须使用相同的标
    识，而不同集群必须不同。
    priority 100    //优先级，用于选举 master，取值范围是 1-255
    advert_int 1 //发 VRRP 包的时间间隔，即多久进行一次 master 选举
    nopreempt //不抢占，即允许一个 priority 比较低的节点作为 master
    track_script {
        check_mysql //指定核对的脚本，check_mysql 是上述自定义的
    }
}
virtual_server 192.168.1.66 3306 {
    protocol TCP //指定转发协议类型，有 TCP 和 UDP 两种
    delay_loop 2   //配置系统状况的检查时间，单位是秒
    lb_kind DR //配置 LVS 的负载均衡机制，可选项有 NAT、TUN、DR
    lb_algo rr    //配置调度算法，rr 表示轮询算法
    persistence_timeout 50 //会话最大可保持的时间。
    //省略转发请求部分配置
}
```

图 5.6 Master 节点 Keepalived 部分配置

5.4.2 测试设计

数据库主备切换的测试思路比较直接，只需要对比主库和从库的 QPS 变化即可。具体地，一开始先模拟请求访问服务器，此时服务器会请求主库数据，然后关闭 MySQL 进程，观察 MYSQLMTOP 里主库和从库 QPS，主库应该逐渐下降为 0，从库会逐渐上升。表 5.12 展示了测试用例设计的细节。

表 5.12 数据库主备切换测试用例

测试 ID	TC9
测试接口	获取聚合报告列表
测试工具	JMeter、MYSQLMTOP
测试方案	1. JMeter 设置 1000 请求数，30 秒循环一次，持续 5 分钟； 2. 发起访问 2 分钟后，关闭主库进程； 3. 使用 MYSQLMTOP 观察数据库 QPS 变化；

5.4.3 测试执行

图 5.7 展示了测试过程中主库和从库的 QPS 变化。测试开始时间是 11:22:00，结束时间是 11:27:00 共 5 分钟，数据点取 30 秒的平均值，QPS 基本维持在 100

左右，原因是每次请求该接口平均会产生 3 次数据库查询，QPS 每 30 秒会产生约 3000 次查询，平均每秒约 100 次。关闭主库进程的时间是 11:24:00，可以看到此时 Master 的 QPS 逐渐下降，Slave 的 QPS 逐渐上升，说明主备成功切换。

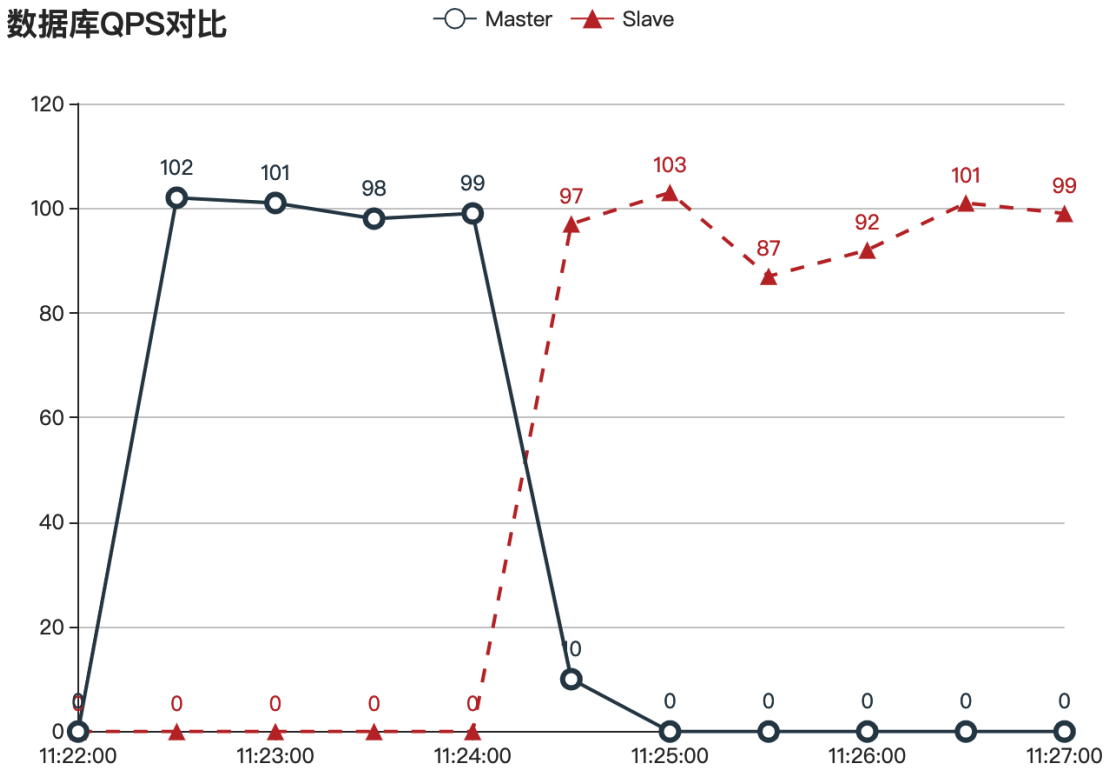

图 5.7 数据库 QPS 对比

本次测试模拟的情况是主库因为宕机导致无法响应时，从库能及时提供服务。经过测试，主从之间能做到及时切换。不过这种方案不保证能应对所有情况，当主库是因请求量过大导致宕机时，从库依然会面临同样的情况，不过本系统面向企业内部服务，目前不会出现高并发场景，系统设计能满足当前需求。

5.5 本章小结

本章主要对众包审核交付系统进行功能性和非功能性的测试。功能性测试主要针对第三章需求分析中总结的用户场景依次进行测试，验证系统是否满足产品规格要求。经测试，系统已实现用户的各项功能需求。非功能性测试针对服务器水平扩展的能力和数据库主备切换的能力进行测试。经过测试服务器具备一定的水平扩展能力，同时，数据库主备切换也能很好完成。最终，经过测试可以确认系统不仅能满足产品功能需求，同时具备一定的稳定性和可靠性。

第六章 总结与展望

6.1 总结

众包测试报告数量大、重复率高、质量相对较低的特点，给报告审核工作带来工作量大、重复工作多、报告阅读困难等问题。为解决该问题，本文设计开发了基于测试报告融合的众包审核交付系统。

该系统借鉴报告融合的思想，在传统众包测试报告审核流程中，创新地引入了基于报告融合的测试报告处理机制。该机制包括聚合和融合两个阶段。在聚合阶段，系统基于文本和截图的特征，对原始报告进行聚类，从而使相似报告被聚合到统一类簇中。在融合阶段，首先，系统基于 PageRank 算法，计算每个类簇中报告的权重，选取最高权重者作为主报告；其次，系统将每个类簇中的其他报告拆分为句子和截图，并和主报告对比，筛选出差异点；最后，系统对差异点聚类，每个类簇作为该主报告的一个补充点。最终主报告与补充点共同构成了一份聚合报告。

通过聚合阶段的处理，审核人员可以集中处理相似报告，极大降低了需要审核的报告数，从而减少了工作量和重复工作。通过融合阶段的处理，多份相似报告的冗余信息按照主报告和补充点的形式进行组织和展示，提高了报告的可读性，降低了因报告质量低导致的阅读困难。

本文主要工作如下：

首先，综述相似报告检测和测试报告摘要领域的研究现状。本文对相似报告检测领域和测试报告摘要领域的研究工作进行了较为深入的整理和分析。在该过程中，受到 Hao 等人[Hao et al., 2019]研究工作的启发，将基于报告融合的测试报告处理机制引入测试报告审核流程中。

其次，完成众包审核交付系统的需求分析。本文对测试报告审核交付的业务流程和困难进行深入分析，并重新设计了围绕报告融合的审核业务流程。

第三，完成众包审核交付系统的工程设计，并引入报告融合的测试报告处理机制。本文对系统功能模块划分为文本处理模块、图片处理模块、聚合模块、融合模块和审核交付模块。为降低系统耦合，增强可扩展性，本系统充分利用

EventBus 进行事件化和异步化的设计。为保证数据的可靠性，数据库采用主从同步的部署方案。

最后，完成众包审核交付系统的实现，并进行了系统测试。本文依照工程设计对系统进行了实现，并对系统功能、水平扩展性和数据库主备切换能力进行了测试。结果显示系统均通过测试。

目前本系统已在南京慕测信息科技有限公司的众包测试平台上线运行。该系统通过聚合相似报告、合理组织和展示相似报告的冗余信息以及友好的用户体验，降低了审核人员的工作量，减少了重复工作，提高了报告可读性。该系统已成功帮助公司完成数十次众包任务的审核工作，帮助公司更好地将测试产物交付给任务发布方，提高了客户满意度，并为企业降低了人力成本，提高了经济效益。

6.2 进一步工作展望

当前系统存在下列可以后续改进的方面。

(1) 可交付报告自动化生成。在现阶段的审核流程中，系统为保证可交付报告的质量，依然需要人工介入进行整理。未来本系统将在人工智能相关领域继续探索，结合机器学习技术来自动生成可交付的众包测试报告，进一步减少人力成本。

(2) 测试报告推荐。现阶段审核人员随机选择聚合报告进行审核。未来系统将记录审核人员的审核历史数据，并基于该数据对审核人员应该审核的报告进行推荐，通过增强匹配度来提高审核效率。例如一名审核人员审核金融类应用的测试报告比较多，可能更善于理解该类问题。

(3) 聚合报告置信度计算。未来系统可以评估每个聚合报告的效果，并进行打分。效果好的聚合报告有较高的置信度，审核人员可以不用进行审核，直接将其转换成交付报告，进而降低审核人员工作量。

综上，本文所设计的众包审核交付系统还存在一些可继续提升的地方。通过可交付报告自动化生成和聚合报告置信度计算，系统可以进一步降低审核工作量。通过测试报告推荐，可以提高报告审核的精确度。最终，系统将会更高效、准确地协助审核人员完成审查工作，帮助企业进一步降低人力成本，提高经济效益。

参 考 文 献

[Alipour et al., 2013] A. Alipour, A. Hindle, and E. Stroulia, A Contextual Approach Towards More Accurate Duplicate Bug Report Detection, In *Proceedings of Mining Software Repositories (MSR'2013)*, pages 183–192, 2013.

[Banerjee et al., 2013] S. Banerjee, Z. Syed, J. Helmick, and B. Cukic, A Fusion Approach for Classifying Duplicate Problem Reports, In *Proceedings of International Symposium on Software Reliability Engineering (ISSRE'2013)*, pages 208–217, 2013.

[Bettenburg et al., 2008] N. Bettenburg, R. Premraj, and T. Zimmermann, Duplicate Bug Reports Considered Harmful ... Really?, In *Proceedings of IEEE International Conference on Software Maintenance (ICSME'2008)*, pages 337-345, 2008.

[Bostock et al., 2011] M. Bostock, V. Ogievetsky, and J. Heer. D^3 Data-driven Documents, *IEEE Transactions on Visualization and Computer Graphics*, 2011, 17(12): 2301-2309.

[Carlson, 2012] J. L. Carlson, *Redis in Action*, New York: Manning Publications, 2012.

[Chatzichristofis et al., 2008] S. A. Chatzichristofis, and Y. S. Boutalis. CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval, In *Proceedings of International Conference on Computer Vision Systems (ICVS'2008)*, pages 312-322, 2008.

[Dang et al., 2012] Y. Dang, R. Wu, H. Zhang, D. Zhang, and P. Nobel, Rebucket: A Method for Clustering Duplicate Crash Reports

Based on Call Stack Similarity, In *Proceedings of International Conference on Software Engineering (ICSE'2012)*, pages 1084–1093, 2012.

[Erkan et al., 2004] G. Erkan, and D. R. Radev, Lexrank: Graph-based Lexical Centrality as Salience in Text Summarization, *Journal of Artificial Intelligence Research*, 2004, 22: 457–479.

[Eugster et al., 2003] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec, The Many Faces of Publish/Subscribe, *ACM Computing Surveys*, 2003, 35(2): 114-131.

[Feng et al., 2016] Y. Feng, J. A. Jones, Z. Chen, and C. Fang, Multi-objective Test Report Prioritization Using Image Understanding, In *Proceedings of Automated Software Engineering (ASE'2016)*, pages 202–213, 2016.

[Ferreira et al., 2016] I. Ferreira, E. Cirilo, V. Vieira, and F. Mourao, Bug Report Summarization: An Evaluation of Ranking Techniques. In *2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS'2016)*, pages 101-110, 2016.

[Hao et al., 2019] R. Hao, Y. Feng, J. A. Jones, Y. Li, and Z. Chen, CTRAS: Crowdsourced Test Report Aggregation and Summarization, In *Proceedings of International Conference on Software Engineering (ICSE'2019)*, 2019.

[Heule et al., 2013] S. Heule, M. Nunkesser, and A. Hall, HyperLogLog in Practice: Algorithmic Engineering of A State of The Art Cardinality Estimation Algorithm, In *Proceedings of International Conference on Extending Database Technology (EDBT'2013)*, pages 683-692, 2013.

[Hindle et al., 2016] A. Hindle, A. Alipour, and E. Stroulia, A Contextual Approach Towards More Accurate Duplicate Bug Report Detection and Ranking, *Empirical Software Engineering*,

2016, 21(2): 368–410.

[Huang, 2008] A. Huang, Similarity Measures for Text Document Clustering, In *Proceedings of New Zealand Computer Science Research Student Conference (NZCSRSC'2008)*, pages 9-56, 2008.

[Huffman, 1952] D. A. Huffman, A Method for The Construction of Minimum-Redundancy Codes, In *Proceedings of the Institute of Radio Engineers (IRE'1952)*, pages 1098-1101, 1952.

[Jiang et al., 2018] H. Jiang, X. Chen, T. He, Z. Chen, and X. Li, Fuzzy Clustering of Crowdsourced Test Reports for Apps, *ACM Transactions on Internet Technology*, 2018, 18(2): 18.

[Jiang et al., 2017a] H. Jiang, J. Zhang, H. Ma, N. Nazar, and Z. Ren, Mining Authorship Characteristics in Bug Repositories, *Science China Information Sciences*, 2017, 60(1): 012107.

[Jiang et al., 2017b] H. Jiang, N. Nazar, J. Zhang, T. Zhang, and Z. Ren, Prst: A Pagerank-based Summarization Technique for Summarizing Bug Reports with Duplicates, *International Journal of Software Engineering and Knowledge Engineering*, 2017, 27(6): 869-896.

[Klein et al., 2014] N. Klein, C. S. Corley, and N. A. Kraft, New Features for Duplicate Bug Detection, In *Proceedings of International Conference on Mining Software Repositories (MSR'2014)*, pages 324–327, 2014.

[Lazar et al., 2014] A. Lazar, S. Ritchey, and B. Sharif, Improving the Accuracy of Duplicate Bug Report Detection Using Textual Similarity Measures, In *Proceedings of the 11th Working Conference on Mining Software Repositories. ACM (MSR'2014)*, pages 308–311, 2014.

[Lotufo et al., 2012] R. Lotufo, Z. Malik, and K. Czarnecki, Modelling the Hurried Bug Report Reading Process to Summarize Bug Reports, In *Proceedings of International Conference on Software Maintenance (ICSM'2012)*, pages 430-439, 2012.

[Mao et al., 2017] K. Mao, L. Capra, M. Harman, and Y. Jia, A Survey of The Use of Crowdsourcing in Software Engineering, *Journal of Systems and Software*, 2017, 126: 57-84.

[Mikolov et al., 2013] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Cutting Away the Confusion from Crowdtesting. In *Advances in Neural Information Processing Systems (NIPS'2013)*, pages 3111-3119, 2013.

[Nguyen et al., 2012] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun, Duplicate Bug Report Detection with A Combination of Information Retrieval and Topic Modeling, In *Proceedings of Automated Software Engineering (ASE'2012)*, pages 70–79, 2012.

[Page et al., 1999] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Stanford InfoLab, SIDL-WP-1999-0120, 1999.

[Patel et al., 2015] S. Patel, S. Sihmar, and A. Jatain, A Study of Hierarchical Clustering Algorithms, In *2nd International Conference on Computing for Sustainable Global Development (INDIACom'2015)*. pages 537-541, 2015.

[Rocha et al., 2016] H. Rocha, M. T. Valente, H. Marques-Neto, and G. C. Murphy, An Empirical Study on Recommendations of Similar Bugs, In *Proceedings of International Conference on Software Analysis, Evolution and Reengineering (SANER'2016)*, pages 46–56, 2016.

[Runeson et al., 2007] P. Runeson, M. Alexandersson, and O. Nyholm, Detection of Duplicate Defect Reports Using Natural Language Processing, In *Proceedings of International Conference on Software Engineering (ICSE'2007)*, pages 499–510, 2007.

[Sun et al., 2010] C. Sun, D. Lo, X. Wang, J. Jiang, and S. Khoo, A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval, In *Proceedings of International Conference on Software Engineering (ICSE'2010)*, pages 45–54, 2010.

[Sun et al., 2011] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, Towards More Accurate Retrieval of Duplicate Bug Reports, In *Proceedings of Automated Software Engineering (ASE'2011)*, pages 253–262, 2011.

[Sureka et al., 2010] A. Sureka, and P. Jalote, Detecting Duplicate Bug Report Using Character N-gram-based Features, In *Proceedings of Asia Pacific Software Engineering Conference IEEE (APSEC'2010)*, pages 366–374, 2010.

[Tian et al., 2012] Y. Tian, C. Sun, and D. Lo, Improved Duplicate Bug Report Identification, In *Proceedings of Conference on Software Maintenance and Reengineering (CSMR'2012)*, pages 385–390, 2012.

[Wang et al., 2008] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, An Approach to Detecting Duplicate Bug Reports Using Natural Language and Execution Information, In *Proceedings of International Conference on Software engineering (ICSE'2008)*, pages 461–470, 2008.

[Wang et al., 2018] J. Wang, M. Li, S. Wang, T. Menzies, and Q. Wang, Cutting Away the Confusion from Crowdtesting. *arXiv preprint*, 2018, arXiv:1805.02763.

[Yang et al., 2016] X. Yang, D. Lo, X. Xia, L. Bao, and J. Sun, Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports, In *Proceedings of International Symposium on Software Reliability Engineering (ISSRE'2016)*, pages 127–137, 2016.

[Yeasmin et al., 2014] S. Yeasmin, C. K. Roy, and K. A. Schneider, Interactive Visualization of Bug Reports Using Topic Evolution and Extractive Summaries, In *Proceedings of International Conference on Software Maintenance and Evolution (ICSME'2014)*, pages 421–425, 2014.

[Yeasmin et al., 2015] S. Yeasmin, C. K. Roy, and K. A. Schneider, How Should We Read and Analyze Bug Reports: An Interactive Visualization Using Extractive Summaries and Topic Evolution, In *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering (CASCON'2015)*, pages 171–180, 2015.

致　　谢

论文接近尾声之际，我首先想感谢我的导师刘嘉老师以及陈振宇老师和房春荣老师在我硕士期间给予的指导和帮助。在毕业设计阶段，感谢三位老师全程认真负责的指导。在项目开发阶段，感谢三位老师持续给予我建设性的指导意见，并帮助我不断完善项目。在毕业论文的写作阶段，感谢几位老师持续的关心和答疑解惑，本篇论文的完成离不开老师们的帮助，我要向三位老师表达最崇高的敬意。

其次，我要感谢实验室的郝蕊学姐、李玉莹学姐。感谢他们在我整个项目开发以及撰写毕业论文过程中提供的建议和帮助，耐心地替我解答所有困惑，让我受益匪浅。同时我还要感谢周子懿、陈笑智、张双江、赵文远、李沛源、周顺祥同学，感谢他们与我一同讨论和解决遇到的问题，正是他们的帮助才促使我完成最终的项目，同时他们也让我学到很多优秀的品质。

最后，我要感谢大学期间软件学院对我的培养，让我从一个对计算机编程、对软件开发一窍不通的人，成长为一个具有相对完整软件开发知识和能力的硕士毕业生。感谢软院的所有老师为教学工作付出的辛勤努力，也感谢软件学院的其他优秀的同学们，让我有一起共同努力的伙伴，有这样一个积极向上良好的学习环境。

版权及论文原创性说明

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

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

作者签名：

日期： 年 月 日