南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目	面向协作式众包测试推荐系统的设计与实现
作 者 姓 名	韩鹏飞
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 4 月 11 日

学　　号：**MF1732039**

论文答辩日期：**2019 年 5 月 13 日**

指 导 教 师：　　（签字）

The Design and Implementation of Recommendation System for Collaborative Crowdsourcing Testing

By

Pengfei Han

Supervised by

Associate Professor **Jia Liu**

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

Software Institute

April 2019

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：面向协作式众包测试推荐系统的设计与实现

工程硕士（软件工程领域） 专业 2017 级硕士生姓名：韩鹏飞

指导教师（姓名、职称）：刘嘉 副教授

摘 要

众包测试作为一种新兴软件测试方法，能够在短时间内召集大量众包工人同时进行在线测试。相较于传统软件测试能够适应移动互联网时代快速迭代软件产品的测试需求，因而在学术和工业界得到了广泛关注。当前对于众包测式的研究，由于缺少众包测试平台的支持，因而多采用模型仿真或数据分析的方法，这种方法缺少真实场景的验证。现阶段，众测平台采取用户间相互竞争的方式，导致产生了大量重复报告，且报告质量参差不齐。同时，由于长尾效应的存在，部分测试需求得不到完全覆盖。

本文针对当前众包测试存在的问题，提出了面向协作式众包测试的推荐系统。用户在整个测试过程中，既承担测试任务，也承担对于已提交报告的审核任务。系统在用户填写报告时进行实时相似报告推荐，引导用户对已提交报告进行审核，从而避免重复报告的提交。系统在用户提交报告后，会对其进行审核报告和测试页面的任务分配。审核报告推荐根据用户协作数据和历史提交记录为用户提供待审核的报告列表，测试页面推荐以可视化的方式为用户推荐待测页面。对于推荐报告，用户可以进行点赞和点踩操作，利用用户间交叉审核，验证报告描述问题的普遍性。同时，用户亦可以选择对推荐报告进行修改后提交，利用多人共同编辑，提升报告的文本描述质量。

本系统使用主流框架进行实现，采用Angular2作为前端框架，Spring Boot作为后端框架，Redis作为查询缓存，MongoDB作为数据库。相似报告推荐采用Word2Vec 与WMD算法实现。审核报告推荐采用基于模型的协同过滤方法实现。测试页面推荐采用基于用户历史提交记录的多源最短路径方法实现。

本文对系统进行了功能测试，并使用Jmeter对系统进行了压力测试。系统在众包测试平台部署运行，与传统竞争式报告提交系统的A/B测试结果表明，该系统能够有效减少众包测试中的重复报告，提高报告文本描述质量，加快测试页面全覆盖的速度。

关键词： 众包协作，众包测试，协同过滤，任务分配

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of Recommendation System for Collaborative Crowdsourcing Testing

SPECIALIZATION: Software Engineering

POSTGRADUATE: Pengfei Han

MENTOR: Associate Professor Jia Liu

Abstract

As an emerging software testing method, crowdsourcing testing can call a large number of workers and conduct online testing in a short time. Therefore it has received extensive attention in academic and industrial circles. Compared to traditional software testing, it can adapt to the testing needs of fast iterative software products in the mobile Internet era. At present, due to the lack of support from the crowdsourcing test platform, the research on crowdsourcing testing take action in model simulation or data analysis. These methods lack verification of the real scene. At this stage, crowdsourcing test platforms adopt a competitive approach, resulting in a large number of duplicate and uneven reports. At the same time, due to the long tail effect, some testing requirements are not fully covered.

In view of the problems existing in current crowdsourcing testing, this thesis proposes a recommendation system for collaborative crowdsourcing testing. During the entire testing process, users undertake both the test task and the audit task. The system performs real-time similar report recommendation when users filling in reports, guiding users to review the submitted report, thereby avoiding the submission of duplicate reports. After submitting reports, this system provides users with task assignment. The audit report recommendation provides users with a list of reports to be reviewed based on users' collaboration data and historical submission records. The test page recommendation visually suggests pages to be tested for users. For the recommendation report, users can perform likes or dislikes. This system uses cross-audit between users to verify the generality of problems in reports. At the same time, this system provides users with choices to modify the recommendation report, using multiple people editing together to improve the quality of the description in the report.

This system is realized by mainstream frameworks, with Angular2 as the front-end framework, Spring Boot as the back-end framework, Redis as the query cache, and MongoDB as the database. Similar report recommendation is implemented by Word2Vec and WMD algorithm. The audit task recommendation is implemented using a model-based collaborative filtering approach. The test page recommendation is implemented using the multi-source shortest path method based on users' histories.

This system is tested for function. Besides, it uses Jmeter to complete the stress test. The system is deployed and run on the crowdsourcing test platform. Competition with independent system showed that the system can effectively reduce the repeated reports. It also improves the quality of report text descriptions and reduces the time required for the full coverage of test pages.

Keywords: Collaborative Crowdsourcing, Crowdsourcing Testing, Collaborative Filtering, Task Assignment

目　录

表　目　录

图　目　录

第一章　引言

1.1　项目背景

移动互联网快速发展的时代，软件产品的生命迭代周期被急剧压缩。传统软件测试测试周期长，且大多使用自动化脚本进行测试遍历，无法模拟用户的真实使用场景。如何快速得到大量用户的真实反馈，以较低成本并且高效地完成应用测试任务，成为了一个难题。

众包技术可以在一定程度上解决这一难题。众包是一种分布式问题解决模式 [1]，企业借助互联网平台发布任务，以自由自愿的形式通过经济或经验激励众包参与人员（以下简称用户）分享时间、创意和知识 [2]。众包概念自提出以来，已经成功应用于机器学习标注、自然语言处理、人机交互等领域。众包软件工程作为众包领域下的一个分支，由于其工作的专业性，致使任务发布方很难将软件工程完整拆分后交由不同的用户来完成。以软件工程的需求分析，详细设计，代码实现到软件测试的工作流程为例，其要求严格按照先后顺序来完成，每一步之间有着强关联，很难通过众包平台 [3]来完成用户之间的协作。因此，大部分众包软件工程任务仍采用整体的形式进行发布，以国内较为出众的“猪八戒”众包平台为例，其平台软件业务大部分都是整体众包，个人或组织通过竞拍方式接单后，给出最终的解决方案。

众包测试作为众包软件工程 [4]下的一个分支，相较于众包软件工程而言，其任务之间关联不大，可并发执行。众包测试可以同时召集大量用户在线参加测试任务，对应用测试所需的真实场景和用户操作进行完整模拟，测试周期短且成本低。因而众包测试也是整个众包软件工程中发展最为迅速的一个分支，与此同时，涌现出许多众包测试平台（以下简称众测平台）。

众包测试的构成主要包括任务发起者、众测平台与用户 [5]。其工作流程主要包含以下三个阶段：

1）任务准备：任务发起者向众测平台提交测试需求，众测平台发布任务。

2）任务执行：用户选择并执行任务，在平台提交测试报告。

3）报告整合：平台对报告有效性进行判定，并整合形成交付报告。

在传统的众包测试中，用户从开始任务就持续单向对众测平台输出信息。众测平台在收集到大量用户历史行为数据与测试任务数据的情况下，却未能及时向用户形成反馈意见。与此同时，众测平台仍采用个体竞争的方式 [6]。对于

测试的用户来说，他们异地参与、水平不一，这就导致以下问题的产生：。

1）大量重复报告 [7]：用户对于系统中已提交的报告内容无感知，导致重复报告的提交。

2）报告内容不完善：单个用户很难对于一个问题进行完整深入描述。

3）平台验证问题普遍性困难：由于移动应用的运行情况受硬件型号、系统版本、网络状况等诸多方面的影响。平台验证问题是否存在普遍性困难。

4）测试页面力度分布不均 [8]：在整个测试过程中，用户处于一种盲人摸象的视角，在选择待测模块及需求时由于长尾效应而导致测试力度分布不均匀，部分测试页面很难完成覆盖。

如何利用用户间协作，减少重复报告，提升报告质量，完成测试需求的覆盖成为了一个难题。本文针对众包测试的任务执行阶段，提出了一种面向协作式众包测试的推荐系统，利用信息共享和任务分配 [9]，解决上述问题。

1.2　国内外研究现状

根据本文所研究的问题，下面将分别从众测平台 [10]、协作式众包以及众包推荐系统三个方面来论述国内外的研究现状。

1.2.1　众测平台现状

国内众测平台有百度众测[1]、腾讯众测[2]、Testin众测[3]等; 国外主流的测试平台有Google Cloud Test Cloud[4]、Amazon Mechanical Turk[5]等。上述众测平台虽然用户群体在稳定增长，但在平台设计上仍然处于初级阶段，靠庞大的用户群来完成任务。用户之间仍保持相对独立状态，并未真正实现人员协作，限制了群体效能的发挥 [11]。

以国内某较早开展众包测试的平台之一为例，其拥有较多的用户群体和较大的月活量。所开展的移动应用测试包含自动化测试和人工测试两种形式。其中，人工测试主要采取众包的方式来进行，其众测报告填写需要用户选择Bug属性、Bug所处页面、上传测试截图并进行文字描述。用户在整个提交过程中，没有任何提示和反馈。导致测试结束后，平台审核人员进行报告检查工作异常繁琐。平台收集到的众包测试报告中含有较多的重复报告，甚至会出现

[1]test.baidu.com
[2]task.qq.com
[3]www.testin.cn
[4]firebase.google.com
[5]www.mturk.com

同一个用户提交多份重复报告的情况。相较于自动化脚本测试而言，众包测试费时费力，却没有得到预期的效果。

1.2.2　协作式众包研究现状

国内外也有很多研究人员在进行协作式众包相关的研究。其中，Wu [12]等人提出协作作为众包的本质特点将会受到越来越多企业的关注，但想实现从多人合作到群体协作，从1+1=2 到1+1>2，任重而道远。To [13]等人研究了众包平台应当如何根据用户历史记录进行用户调度和任务分配，从而满足系统运行过程中实时变化的需求。Pan [14] 等人权衡了在协作式众包系统中，质量、时间和成本，提出了一种帮助系统确定如何结合可用人员的专业知识以最大化预期结果质量同时最小化时间消耗的方法。

陈 [15]等人将科研众包任务匹配模型分为了以下三种，非涌现型、量变式与质变式。非涌现型众包中用户个体彼此独立，相互竞争，发包方希望从中选最有方案。量变式以大量样本为基础，任务简单且同质化，需要对样本进行统计分析得到最终结果。最为典型的例子是机器学习中的众包标注。相较于前面两种模型，质变式具有任务异质化较强，更希望样本聚合后产生质的飞跃，而不仅仅是数量上的汇聚统计的特点。其对应采用的科研众包模式为维基协作式。所谓维基协作式 [16]，即通过多人对于同一个问题的共同编辑，不断完善对于该问题的描述。

上述对于协作式众包的研究，仍处于起步阶段，部分研究提出了协作方法的设想并进行了模型仿真，但缺少系统的实现与验证。

1.2.3　众包推荐系统研究现状

众包推荐系统，即从全局与用户的角度出发，通过推荐，使得任务能够充分发挥出用户的优势，以达到更快任务收敛与更高任务质量的系统。Yuen [17]等人提出用户在选择任务上花费的时间与完成任务所花费的时间相当，并第一次提出在众包系统中使用矩阵分解来完成任务推荐。通过对亚马逊(MTurk)平台搜集的结果表明，用户的历史可以反映用户对于众包任务的偏好。任务分配可以帮助用户快速找到合适的任务，并使其产生更高质量的输出。Li [18]等人提出对于空间众包任务，任务推荐可以从用户的技能、付款和位置三个因素出发计算匹配度，其实验验证的数据集基于Epinion的用户信任关系网络。

Geiger [19]等人提出在众包系统中引入个性化的任务推荐机制，并且通过对相应的学术文献进行系统评价，为该机制的设计提供概念基础。他强调了众包系统需要通过大规模在线实验，与主流推荐系统研究的改进相结合，并强调

最为重要的是实证的结果。Aldhahri [20]等人提出众包系统成功与否的关键所在集中在以下三个目标，使用户与其兴趣所在的任务相匹配，降低众包任务发布方的成本和时间，提高任务接受率。根据以上目标，他们强调现有众包推荐系统必须为此做出改进。

上述对于众包推荐系统的研究，由于缺少众包平台的支持，因而多采用公开数据集进行数据分析。这种研究方式的实验结果与真实环境下的推荐结果存在较大的差异性。同时，推荐系统主要应用于任务准备阶段，而非任务执行阶段，导致单个任务质量的提升并不明显。

1.3　本文主要工作

针对以上的研究现状和众测平台所存在的问题，本文将从众测平台的角度出发，在任务执行阶段针对众包测试用户，开发一个面向协作式众包测试的推荐系统。该系统将完成整个测试任务进行过程中的信息共享与任务分配。其中，本系统所提及的众包测试主要面向移动应用及Web的功能性测试 [21]。平台用户在整个过程中，既是提交者也是审阅者，充分利用用户之间的协作以及平台推荐引导来提升报告质量。

1) 协作方式：用户对于推荐报告可以选择点赞、点踩，众测平台在报告整合阶段，可以参照该信息，对于报告质量做出初步评价。同时，用户亦可以选择共同编辑（Fork），对推荐报告进行修改，修改记录将会以多叉树的形式进行存储。原始报告为其根节点，叶子节点将相较于根节点对于问题的描述更加准确。同时，用户协作数据将构成用户-报告的评分矩阵，作用于审核报告推荐。

2) 信息共享：用户填写测试报告时，系统根据用户填写内容提供实时相似报告推荐，引导用户对相似问题进行审核，以减少重复报告的提交。

3) 任务分配：用户提交报告后，在执行下一个测试任务前，系统将根据用户的历史记录以及当前提交情况，进行任务分配，以加快测试进程。任务分配主要分为审核报告推荐与测试页面推荐。审核报告推荐为用户提供可能感兴趣的报告进行审核。测试页面推荐可视化当前测试页面覆盖情况，根据用户提交历史，为用户推荐待测页面。

本系统作用于众包测试的任务执行阶段，因此需要完成基础的报告提交系统，其中包括了Bug属性的选择，Bug描述的填写，截图的上传与标注等功能。同时，为了验证推荐结果的有效性，需要记录推荐结果与用户反馈以计算推荐系统的召回率。综上，系统主要分为报告提交模块、测试页面推荐模块、审核报告推荐模块以及用户行为记录模块。

为保证系统的可用性，需要对其进行功能测试和压力测试。同时，该系统需要与传统竞争式的众包测试系统进行A/B对比测试，以验证协作式众包测试相较于竞争式众包测试的有效性。

1.4　本文组织结构

国内目前的众包平台正处于发展阶段，且大部分平台仍采用个体竞争，未能充分发挥群体之间协作的优势。本文正是在这样的背景下，首先分析了在众包测试中协作方法和推荐系统的必要性及其意义。接下来，介绍了完成本系统所需要的相关技术和推荐算法。分析了系统的需求，用多种UML方法并进行了概要设计。接着，对于本系统中重要模块的具体实现进行了介绍。最后，对本系统进行了测试和实验评估，并对本文进行了总结，对相关技术的发展做出了展望。本文的组织结构如下：

第一章，引言部分。对众包测试，协作式众包的概念以及推荐系统在众包中的应用做了一个简单的介绍。

第二章，相关技术综述。对于文中所需要使用到的高可用、高并发技术框架以及推荐系统所需要的算法进行了简单介绍。

第三章，推荐系统的需求分析与概要设计。对系统的需求进行了分析，根据系统需求对系统的整体结构、各个模块以及数据库进行了设计。

第四章，推荐系统的详细设计与实现。对报告提交模块、审核报告推荐模块、测试页面推荐模块、用户行为记录模块的具体实现进行了详细描述。

第五章，推荐系统的测试和实验评估。对系统进行功能和压力测试。并将系统与竞争式的众包测式系统进行了A/B 对比测试。

第六章，总结和展望。对论文所完成的工作进行了总结，对协作式众包测试推荐系统的未来技术发展方向作了展望。

第二章　相关技术综述

2.1　高可用相关技术

系统高可用的特性，要求其采用主流的可扩展框架进行开发。因而，本系统选择使用以下三种技术进行开发。采用Spring Boot作为系统的后端框架，Angular2作为系统的前端框架，Docker作为系统的运行环境。下面将简单介绍本系统技术选型的优势。

2.1.1　Spring Boot

Spring Boot是在2013年推出用于简化Spring项目开发、配置、测试、部署的开发框架 [22]。它本质上是一些库的集合，这些库中包含了在开发中必须的大量实用基础框架。如内嵌容器(Tomcat)、JMS框架、日志框架(Log4j)、测试框架(JUnit)、NoSQL数据库(MongoDB)、缓存框架(Redis)等 [23]。

Spring Boot本质上依旧基于Spring框架，即用以管理Java组件的生命周期，将事物组合到一起的框架。但是，相较于传统Spring项目而言，它无需部署WAR文件，内嵌Tocamt，可以直接打包成可运行的jar文件。同时，它简化了Maven的配置，当需要同时引入多个依赖包时，会自行选择可配合运行的依赖包，减少依赖包的版本冲突。因而使用Spring Boot会使得原本的配置简化，仅需在Maven加入所需要的依赖包。由于本系统以微服务 [24]的形式为平台提供服务，微服务要求应用轻量可扩展，因此采用该框架作为系统的后端服务。

2.1.2　Angular2

AngularJS [25]是一款由Google维护的开源JavaScript库，用以协助单一页面应用程序运行的All in One框架。它的目标是通过MVC功能增强基于浏览器的应用，使得开发和测试变得更加简单。在第二个版本中全面使用TypeScript且基本不向下兼容，同时改名为Angular2。

Angular2使用MVC模式，将页面(DOM)、前端业务逻辑以及数据解绑，后端减轻了负担，使得Web应用更加轻便。Angular2使用声明式编程，框架扩展了HTML，通过双向数据绑定来展示动态内容。开发者可以将想要展示的DOM和数据进行双向绑定，一旦双向绑定完成，用户在页面上的输入会由Angular2保存到变量中，并且会自动更新所有应用了该变量的内容中去，从而在效果上达到即时的数据同步。

Angular2应用程序采用模块化的方式，即将代码组织成独立的块或桶，每个桶提供独立的功能，这使得应用程序的测试、升级和维护都更加简单高效。模块化也使得工作容易拆分，可以将庞大的前端应用拆分后由多个团队共同协作完成，从而加快开发进度。由于本系统中包含大量动态数据，因此，采用Angular2作为前端框架。

2.1.3　Docker

Docker [26]是由Paas供应商dotCloud于2013 年推出的一款开源应用容器引擎，它可以让开发者将自己的应用及运行环境打包到一个可移植的容器中，然后将该容器发布到任意一台Linux机器上来运行。容器采用沙箱机制，互相间不会有任何接口。相较于传统的VM(Virtual Machine，虚拟机)而言，它占用内存和CPU资源更少。一个完整的Docker运行环境由以下四部分组成：Docker Client、Docker Daemon、Docker Image以及Docker Container。

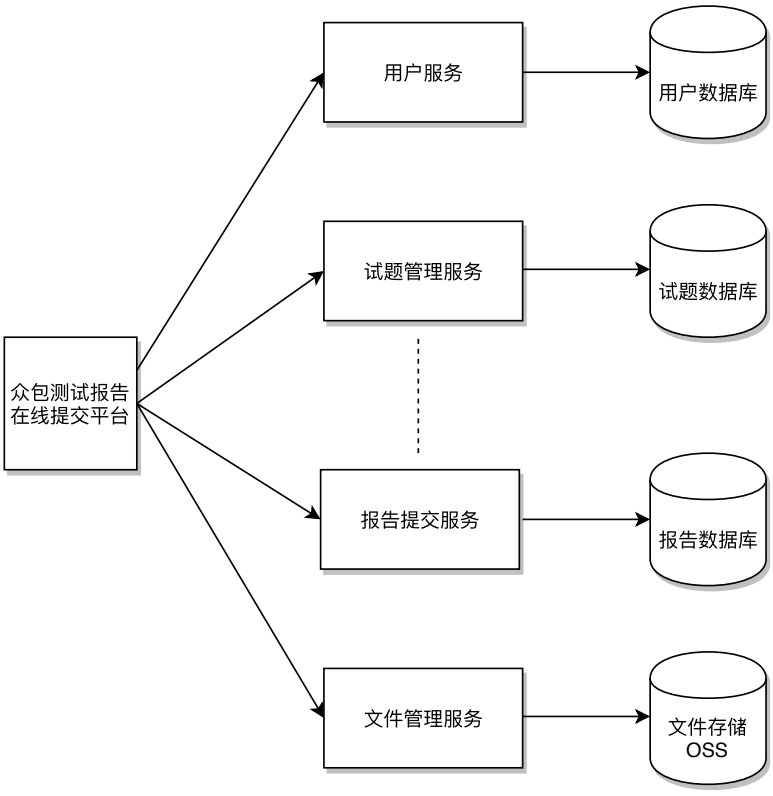

图 2.1: 众测报告提交平台业务逻辑拆分图

Docker是一个容器工具，提供应用所需的虚拟运行环境，它与微服务是密切相关的。如图 2.1所示，以众测平台的部分功能为例，其业务逻辑会拆分成

用户服务、试题服务、报告服务等等。实际上，一个庞大的软件系统就是一系列容器的集合：前端容器，业务逻辑容器、数据库容器、队列容器等。这样可以做到各个模块间的解耦，便于模块复用和维护。使用Docker就可以将系统拆分成多个标准化容器，在一台计算机上可以同时运行多个容器，他们之间采用RESTful的API 进行信息的传递。这样，即便某项微服务失败，系统也不会崩溃，且可以做到基本可用和快速恢复。由于Docker 是微服务最为常见的载体，因而使用其作为项目的运行环境。

2.2　高并发相关技术

鉴于系统需要支撑众测平台较大的用户量，同时，该系统还需要应对众包任务发布方要求在限定时间内的高并发访问，要求在设计该系统时需要考虑并发性。因而，本系统选择了MongoDB 作为系统的数据库，选择Redis作为系统的缓存。下面将概述这两种技术。

2.2.1　MongoDB

MongoDB [27]是一个面向文档的数据库，它具有高性能、易使用的特点。

NoSQL存储方法大约可以分为四种：列存储、键值存储、图形存储、文档存储。MongoDB使用JSON的变种BSON进行文档存储。针对MongoDB的操作都使用JSON，客户端提交或接收的数据都以JSON或者JSON数组的形式来展现。Java中gson等工具库已经对JSON进行了完整支持，可以快速将Java类转化成JSON数据。MongoDB支持嵌入子文档。一个数据库可以有多个集合，每个集合都由多个文档组成。集合和文档并不与传统数据库的表和列完全对等。而且，MongoDB并不需要事先定义集合，它随时可以创建。每个集合中可以包含有不同样式的文档记录。

相对于传统关系型数据库如MySQL而言，在一个关系型数据库中，一份Bug报告一定需要包含完整的信息，如所处页面，Bug属性、Bug描述、截图URL等。同时，Bug描述的内容可能会非常复杂。而MongoDB中每一条数据为一个独立JSON，因此可以支持存储不完整的数据与复杂数据，从而避免空间浪费。MongoDB对非索引字段的查询速度全面胜过传统数据库，建议使用delete加insert取代update操作。Spring Boot框架可以快速引入MongoDB，需要在Maven中引入spring-data-mongodb 依赖。由于本系统中读操作远高于写操作，且对于数据安全性要求高，MongoDB便于主从备份，因而采用其作为数据库。

2.2.2　Redis

Redis是一款开源的高性能Key-Value数据库 [28]，是当前最热门的NoSql数据之一。由于其会将所有数据保存在内存中，所以其相对于硬盘存储具有最明显的优势就是快速。相较于Memcached等Key-Value存储系统，Redis还支持将内存中的数据周期性写入磁盘。如图 2.2所示，其特性决定了对于结果不频繁变动的SQL，适合将其运行结果放入Redis缓存中，以减轻数据库的压力，同时加快响应速度。

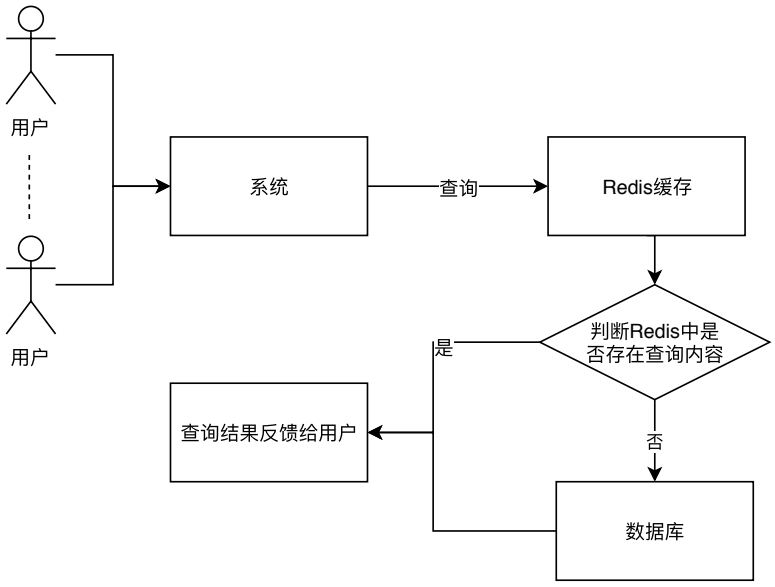

图 2.2: Redis应用示意图

Redis与其他的非关系型数据库大多只支持字符串类型不同 [29]，其value支持的数据结构类型极为丰富，包含了string，hash，list，set以及sorted set。Spring Boot框架可以快速引入Redis，需要在pom.xml文件中引入spring-boot-starter-data-redis依赖。由于本系统中存在大量查询操作，且部分数据变化不频繁，因此采用Redis作为查询缓存数据库。

2.3　推荐方式与目标

推荐系统的目标是为用户推荐一系列其潜在感兴趣的商品或信息 [30]。根据推荐方式的不同，主要分为以下四种 [31]:

1) 热门推荐：热门推荐相对简单，主要以排行榜的形式实现，在商品售卖中较为常用。

2) 人工推荐：人工推荐方式成本过高，但效果最为明显。多用于新闻中。

3) 相关推荐：根据关联规则挖掘或用户浏览记录，推荐用户目标商品。

4) 个性化推荐：基于用户历史行为，推荐用户潜在的喜好。

众包测试推荐与商品推荐并不完全相同，相较于从用户角度出发，更注重于测试的过程和质量。其目标主要集中在以下三点：

1) 减少重复报告：众包测试往往产生大量测试报告，传统个体竞争的方式，用户并不清楚当前系统有哪些已提交的报告，因而会形成大量重复报告，增加了平台审核的工作。

2) 提升报告质量：提升报告质量除了提升报告的描述质量，还包括利用用户交叉验证报告所描述问题的普遍性。

3) 提高页面覆盖率：众包测试所需时间越长，则任务发布方所消耗的成本也越高。因此，需要快速完成测试需求中的页面覆盖。但由于长尾效应的存在，很多页面难以得到测试，用户大多会集中在易于发现问题的页面。

基于上述三个目标，本系统采用相关推荐与个性化推荐相结合的方法。具体实现方式即相似报告推荐以及基于用户历史的任务分配两种方式。

2.4　文本相似度计算

在信息检索中，相似度的计算反应了用户输入内容与系统推荐结果的相关程度。在本系统中，用户提交报告的输入主要分为标签选择，文本输入和图片输入三个部分。鉴于系统实时性的要求，图片相似的计算过于缓慢，所以本系统中的相似度计算主要依靠文本相似度计算与标签匹配共同完成。

文本相似度 [32]计算大致分为两步：构建文本矩阵和计算文本矩阵相似度。构建文本矩阵主流方法有词袋模型、词向量等，但由于词袋模型不会考虑词语之间的关联，因此本系统选用词向量来完成。文本矩阵相似度的常见计算方法有余弦相似度、曼哈顿距离等，这些计算方法只考虑词频，不考虑上下文，但在Bug描述中，会有操作顺序、反馈结果等上下文联系，因此本系统选用单词最短移动距离来计算两个文本间的相似度。

下面将介绍本系统所采用的文本矩阵构建算法Word2Vec以及文本相似度计算的WMD算法。

2.4.1　Word2Vec

词向量并非Word2Vec提出，很早就有了基于词汇表的词向量，用0或1代表一段文本中是否包含词汇表中的某个词语。也有根据词频形成的词向量，但这样的模型都不能反映文本中词语上下文的联系。

CBOW(Continuous Bag-of-Words)和Skip-Gram模型用于神经网络语言模型。CBOW通过给定上下文，来预测下一个input word。Skip-Gram与之相反，通过input word 来预测上下文。Word2Vec(Word to Vector，即词向量)是使用前馈神经网络来将词语转化成向量的模型 [33]。在形成的向量空间中，语义相似的单词其向量间的距离会更短。

对于英文文本，通常根据标点符号和空格即可完成对句子的分割得到单词。但是，在中文里，提取中文词语是一个难题 [34]。中文开源分词工具已经相对成熟，其中较为流行的有结巴分词，Ansj等。这些中文开源分词工具多采用前缀词典以实现更为高效的词图扫描。根据扫描结果生成目标句子中所有成词可能所形成的有向无环图。最后采用动态规划查找最大概率路径，生成基于词频的最大切分组合。

2.4.2　WMD算法

WMD(Word Mover’s Distance，单词最短移动距离)是由Kusner [35] 等人提出的测量两个文本文档之间相似度的算法。传统的计算文本向量算法如曼哈顿距离和余弦相似度，无法从整体上来度量文档间的相似度。其仅仅是基于文本的用词，对于词序并不敏感，这就会造成严重的信息缺失。而WMD算法计算作为一个文档的嵌入词需要“移动”以到达另一个文档的嵌入词的最小距离量。其相对前面的计算方法而言，能够更好地表示句子中局部共现的单词。

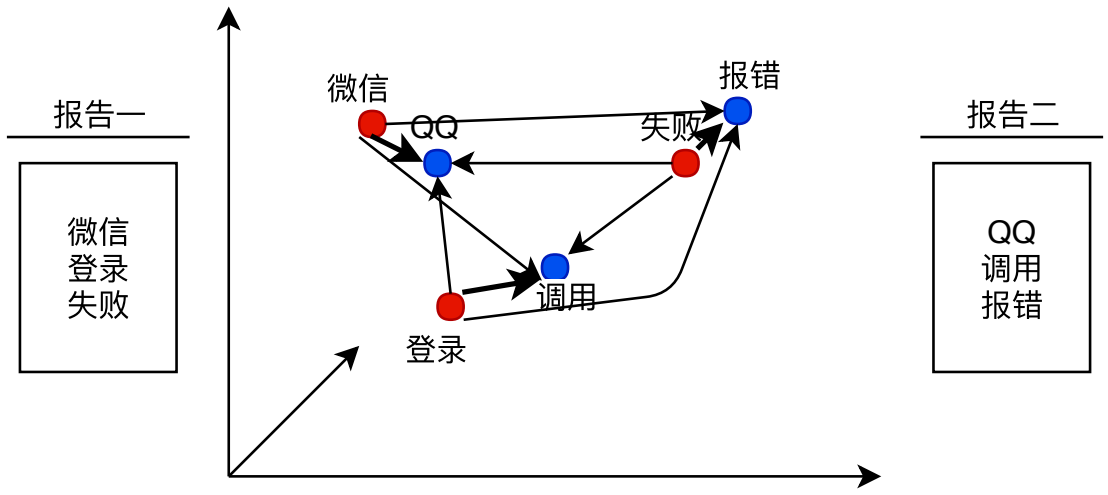

图 2.3: WMD算法示意图

以图 2.3为例，在进行分词之后，两份报告均剩下3个词，使用这3个词来比较两份报告的相似度。首先将这些词使用Word2Vec映射到词向量空间中，然后

分别计算报告一的每个词到报告二的每个词的距离，由于微信与QQ在语义上更为相近，所以他们在向量空间中的距离也最短。在此例子中，报告一与报告二的文本相似度等于报告一中所有单词转移到报告二中所有单词最短距离的和，即D(微信，QQ) + D(登录，调用) + D(失败，报错)。

2.5　任务分配计算

本推荐系统除2.4小节介绍的实时相似度推荐，还要进行测试任务的分配。由于用户在整个测试过程中，既要承担测试任务，也要承担审核任务。因而对于一个用户，需要根据用户历史记录 [36]来同时形成测试页面推荐与审核报告推荐。本系统将根据其历史提交的报告和当前所有用户提交报告所在页面，页面间的跳转关系生成图结构，使用多源最短路径进行测试页面推荐。主要使用基于模型的协同过滤，利用矩阵分解来进行审核报告推荐。

2.5.1　最短路径算法

最短路径是图论中的一个经典算法 [37]。它主要是解决在图中，从某一节点出发，沿图的边到另一节点所经过的路径中，边权值之和最小的路径。以图中共存在N个节点，M条边为例，常见的问题有：

1) 已知起点的最短路径：确定起点，求单源、无负权值边的最短路径，最常采用的算法为Dijkstra算法。算法复杂度为$O(N^2)$。

2) 已知终点的最短路径：在无向图中，其本质上依旧为求已知起点的最短路径。在有向图中，该问题可以采用将所有的边反转方向，转化为求已知起点的最短路径。

3) 起始点均确定的最短路径：确定起始点，求两点间的最短路径。此类问题较为简单，可以采用深度或广度优先的遍历方式。

4) 全局最短路径：无确定节点，求多源、无负权值边的最短路径。常使用Floyd算法，利用矩阵记录图结构。算法复杂度较高，为$O(N^3)$。

本系统需要解决的问题可以抽象为，在有向图中，所有边均有为正权值。已知多个起始点，且图中有部分节点已经被占领，求与所有起始节点最近的三个节点。因此，采用Flord算法，并针对问题的特殊性进行了优化。

2.5.2　基于模型的协同过滤

Adomavicius [38]等人将主流的推荐系统分为如下三类：基于协同过滤的推荐算法、基于内容的推荐算法和混合推荐算法。基于内容的推荐算法根据物品

的属性，向用户推荐相似的商品信息。基于协同过滤的推荐算法主要分为三类：基于用户的协同过滤、基于物品的协同过滤和基于模型的协同过滤。基于用户的协同过滤的假设前提是如果两个用户的历史行为相似，则他们在未来的行为选择也会相似。例如：用户A喜欢物品B，用户C与用户A相似，则将物品B推荐给用户C。在本系统中，用户水平不一，且由于进行测试的环境和习惯也不尽相同，故不适用基于用户的协同过滤算法。

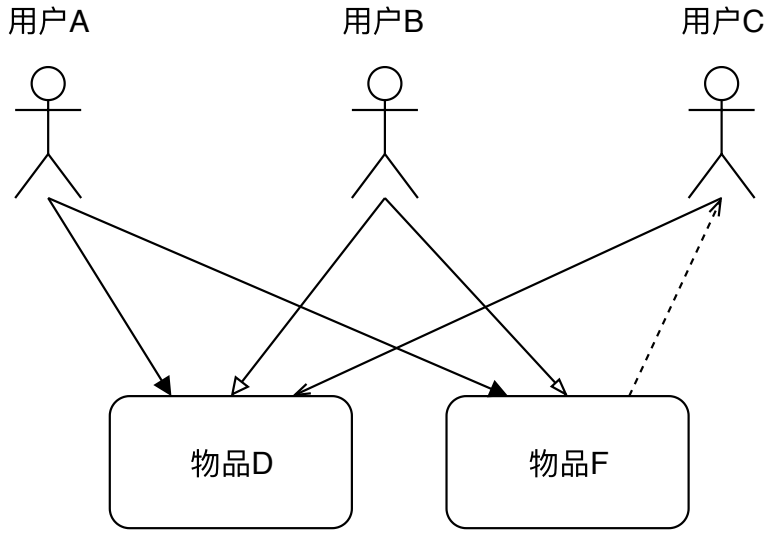

图 2.4: 基于物品的协同过滤

基于物品的协同过滤则是根据所有用户对物品的评价，发现物品间的相关度，最后根据用户的历史记录来进行推荐。如图 2.4所示：用户A和用户B 均喜欢物品D 和物品F，则任务物品物品D和物品F有关联，所以当用户C喜欢物品D时，则将物品F推荐给用户C。但由于该方法需要大量用户评价数据，但在测试任务开始时，此类数据较少，因此会遇到冷启动问题，故不采用该方法。

基于模型的协同过滤则是通过降维的手段在保留主要信息的同时，降低信息量的规模。比如，一份测试报告可以用所在页面，问题类别，严重程度，这样一个三维隐向量来进行表示。在降维之后，推荐问题就会简化为计算向量相关度的问题。

2.5.3　矩阵分解算法PMF

矩阵分解 [39]是基于模型的协同过滤中运用最为广泛的方法。矩阵分解会形成三个矩阵，用户隐特征矩阵和物品隐特征矩阵用于表示用户和物品，同时，还会形成用户对于物品评分的二维矩阵矩阵，该矩阵其中一个维度表示所有用

户，另一个维度表示物品，矩阵中的值表示用户对于物品的评分信息。

PMF (Probabilistic Matrix Factorization)算法 [40]假设用户U和项目V特征矩阵均服从高斯分布，通过评分矩阵已知值得到U和V的特征矩阵，然后用特征矩阵去预测评分矩阵中的未知值。影响一个用户是否喜欢某一个物品的因素有X个，评分矩阵可以分解为两个低维矩阵的乘积$R = U^T V$，其中$D \times N$矩阵U描述N个用户的对于这X 个因素的关注情况，$D \times M$矩阵V描述M个物品这X个因素的值。

2.6　本章小结

本章主要介绍了面向协作式众包测试推荐系统的相关技术框架和推荐算法。首先，介绍了系统可用性相关的前端框架Augular2、后端框架Spring Boot与运行环境Docker。然后介绍了与系统并发相关的数据库MongoDB及缓存技术Redis。最后，介绍了推荐系统中相似度推荐相关的Word2Vec与WMD算法，以及任务分配所需要的基于模型的协同过滤与最短路径算法。本章的主要目的是为使系统有强有力的理论依据和实现基础。

第三章　推荐系统的需求分析与概要设计

本系统主要针对众包测试的参与人员进行设计，在其参与众包测试整个过程中，为其提供信息共享和任务匹配。在本系统中，一份众包测试报告就代表了用户对于一个Bug的描述记录。在一次测试任务中，用户可以上传多份报告。本章将分析系统目标用户的需求以及对系统的前后端架构，实体类、关键算法进行概要设计。

3.1　系统需求分析

众包测试从整体上来看，其服务的对象主要有两类，一类为发布题目的众包任务发布方，一类为参与众包任务的用户。对于众包任务发布方而言，他们希望能够使用更少的人力资源，在更短的时间内完成预期的测试目标。这也是本系统的设计目标：减少重复报告，提升报告质量，缩短测试时长，提高页面覆盖率。在本系统中，任务发布方需要提供待测app、测试页面EXCEL表格以及测试需求说明。该部分由外部平台系统负责收集此类信息。

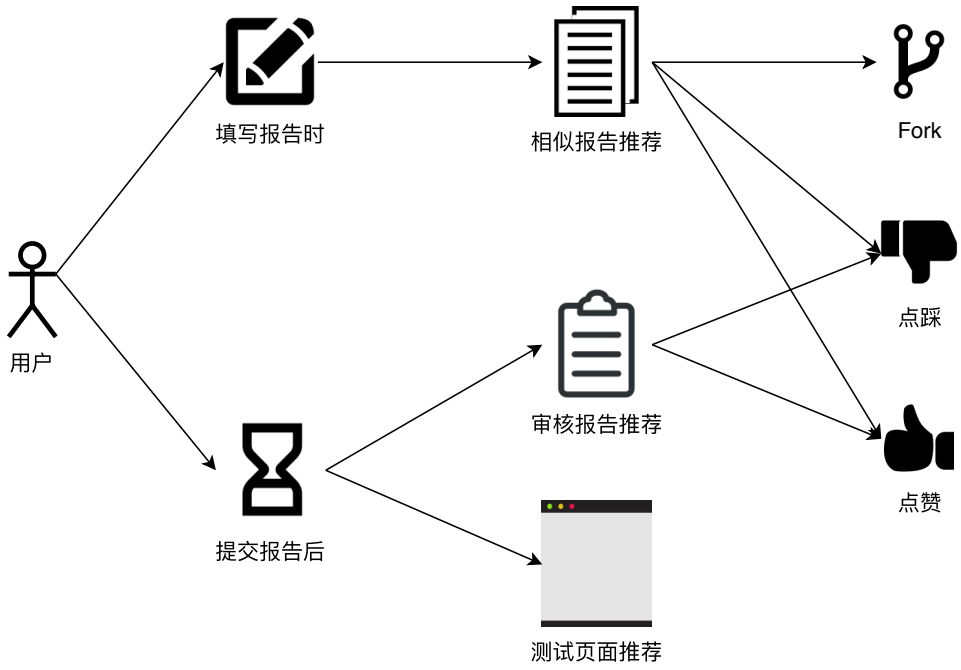

图 3.1: 系统功能示意图

用户由于其异地参与，水平不一，因而要求系统的设计不能过于复杂，采

用最为自然地交互方式为用户提供服务。本系统的使用人员为用户，如图 3.1所示，该系统仅面向众测平台用户进行设计。系统主要在其任务执行阶段对其进行推荐，推荐系统也主要分为两个大的模块，实时相似报告推荐和任务分配。其中，任务分配又分为审核报告推荐和测试页面推荐，二者是平行的关系，用户可以根据个人意愿选择相应的推荐结果。

1) 用户在填写报告时，系统需要根据用户的选择和输入，实时提示当前已提交的报告中与该用户提交内容可能相似的报告。对于推荐的报告，用户可以对其进行点赞、点踩和Fork这三种审核操作。从而避免用户提交重复的报告，增加任务发布方的审核成本。同时，审核的结果也可以辅助任务发布方进行结果的评判。

2) 用户在提交完当前报告后，系统需要根据用户的历史行为信息，如点赞、点踩、Fork、报告提交记录等进行推荐。对于当前操作用户，需要为其展示全局测试进度与用户个人测试进度的对比。根据用户提交历史为其推荐测试页面。同时，需要根据用户协作数据以及提交历史，来判断其可能还会对哪些报告感兴趣，从而为用户推荐待审核测试报告。

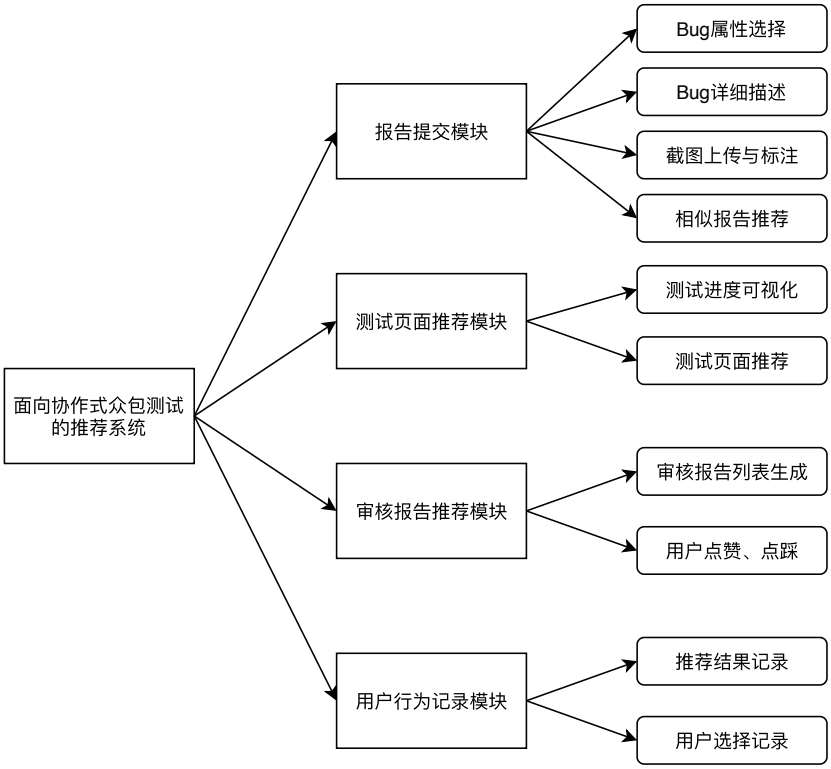

图 3.2: 系统模块划分

如图 3.2所示，从系统角度出发，系统主要分为四大模块，报告提交模块、

测试页面推荐模块、审核报告推荐模块与用户行为记录模块。其中用户行为记录模块采用异步的方式进行，即该模块与其它三个模块为并行关系，在系统运行期间将持续运行。对于每次推荐的结果，系统都会进行记录。同时，还需要记录用户在得到推荐结果之后的行为决策，从而计算推荐系统的召回率，便于对系统做出评价。该模块的具体记录内容会在算法设计小节进行描述，下面将介绍除用户行为记录模块外，另外三个与用户进行交互模块的需求。

3.1.1　报告提交模块

报告提交模块是本系统中最为复杂的核心模块。该模块的用例如图 3.3 所示，用户在该模块中进行报告填写。一份完整的众包测试报告填写内容主要包含以下4个方面：

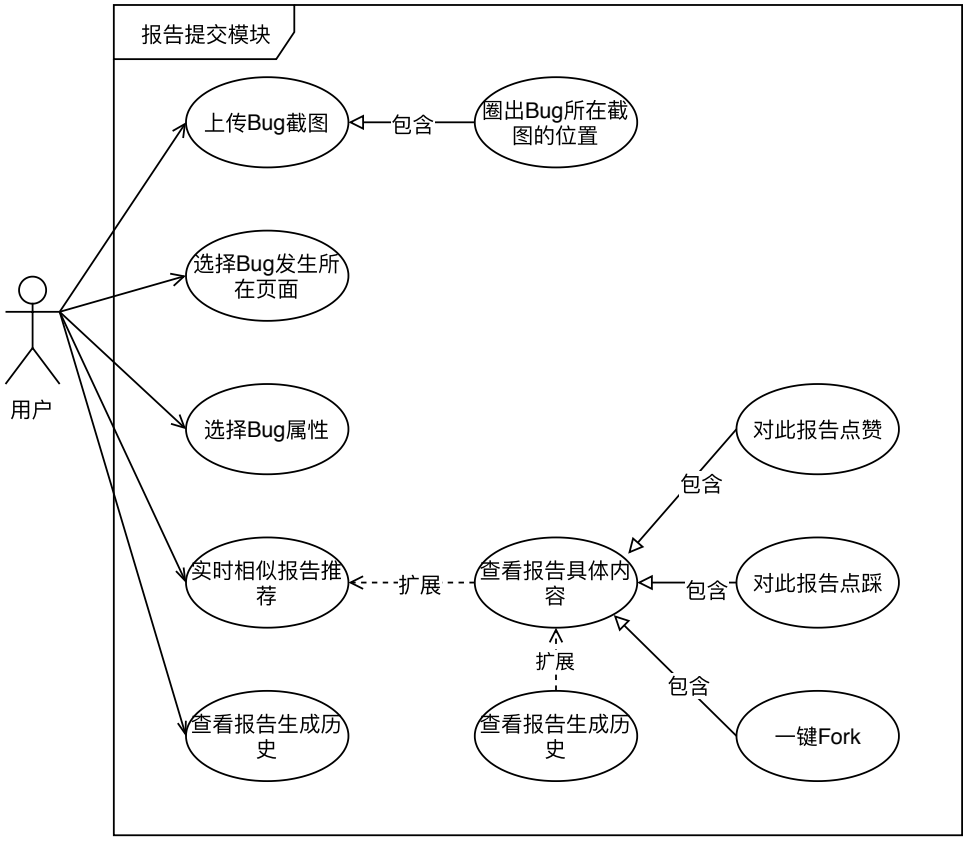

图 3.3: 报告提交模块用例图

1) 选择Bug所处的页面，由于任务发布方已经提供了页面间的跳转逻辑，所以用户仅需按照测试时的点击路径进行页面选择。这样可以防止用户对于页面描述的差异性导致推荐结果不准确的情况发生。

2) 填写Bug属性，基于Bugzilla [41]中提交Bug所需要填写的内容，结合移动应用及Web应用功能测试的需求，本系统中的Bug包含以下三种属性：复现程度、严重等级以及漏洞分类。复现程度有无规律复现、小概率复现、大概率复现、必现和其他五类。严重等级有待定、较轻、一般、严重和紧急五类。漏洞分类包括不正常退出、功能不完整、用户体验、页面布局缺陷、性能、安全和其他七类。

3) 填写Bug的详细描述，用户需要对于该Bug进行一句话概括和发生前置条件和产生结果的详细描述。

4) 上传Bug截图，用户需要上传当前提交报告中所描述的Bug产生时的屏幕截图，系统可接收的图片格式有jpg、png等。用户在上传图片完成后需要对图片中Bug所处的具体位置使用鼠标进行标注。从而能够让任务发布方快速定位问题所在的具体位置。

表 3.1: 报告提交模块用例表

ID	UC_01
用例名称	报告提交
用例目标	用户填写发现的Bug并提交
参与者	用户
前置条件	1. 接受众包任务 2. 填写运行环境信息 3. 点击创建
成功结果	提示上传成功
正常流程	1. 上传Bug截图并进行标记 2. 选择Bug所处页面 3. 选择Bug属性 4. 填写Bug详细描述 5. 查看相似报告推荐列表 6. 提交报告
扩展流程	5a. 发现相同问题描述 1. 进行点赞、点踩操作 5b. 发现相同问题但描述不够清晰 1. 一键Fork至当前填写内容 2. 对报告进行修改

鉴于系统实时性的要求，除用户上传截图的操作外，用户的每一次选择和进行Bug 描述的文字填写都需要进行实时的相似报告推荐，生成相似报告列表。用户在得到报告推荐列表后，可以查看每条报告的具体描述和截图，同时可以查看该报告是原始报告，还是由别人的报告Fork修改后得到，即将该报告的历史提交记录可视化。如果用户认同此报告的内容，则可以对报告进行点赞操作，无需自己再提交相同的报告。如果用户不认同此报告的内容，则可以对报告进行点踩操作，然后继续填写用户个人报告。如果用户部分认同该报告的内容，则可以选择一键Fork至个人当前填写的报告。然后，在此报告的基础上修改后进行提交。

根据该模块用例的用户需求分析得到的用例如表 3.1所示。其中要注意的是，前置条件中的接受众包任务和填写运行环境信息是在外部平台上进行的。按照正常流程，用户点击创建应该都会完成一次提交，但是，如果用户进行点踩或者点赞操作，则用户有可能终止其提交操作。

3.1.2　测试页面推荐模块

表 3.2: 旅游app测试页面需求

一级页面	二级页面	三级页面
搜索	推荐目的地	地区选择
		查看更多
	热门搜索	品类
		周边
		主题
		换一换
	帮你选目的地	热度排序
		区域
		主题
		全部
	搜索30030554	邀请好友
		进行砍价
我的	推荐有礼	推荐方式一

测试页面推荐模块是用户在提交完一份测试报告后出现的模块。以某平台的一次移动应用众包测试需求为例。该任务需要对制定版本的某旅游app进行功

能测试，测试所需要覆盖的页面如表 3.2所示。一级模块代表用户在进入app 后即可看到的模块，在此页面基础上，点击进入相应的二级模块，三级模块以此类推。

如图 3.4所示，该模块主要完成了以下四种信息的可视化展示：

1) 测试页面路径信息：将众包任务发布方提供的测试页面需求EXCEL表格转化为图结构，点为页面，边表示页面间存在跳转关系。

2) 当前测试总体情况信息：标注当前测试已覆盖的页面及已覆盖页面用户提交的报告数量。

3) 用户提交报告页面覆盖情况：标注当前用户对于此次众包任务已经发现Bug的页面。

4) 测试页面推荐信息：在图表中显著标注推荐用户进行下一步测试的页面所在，当用户完成全覆盖时，该信息失效。

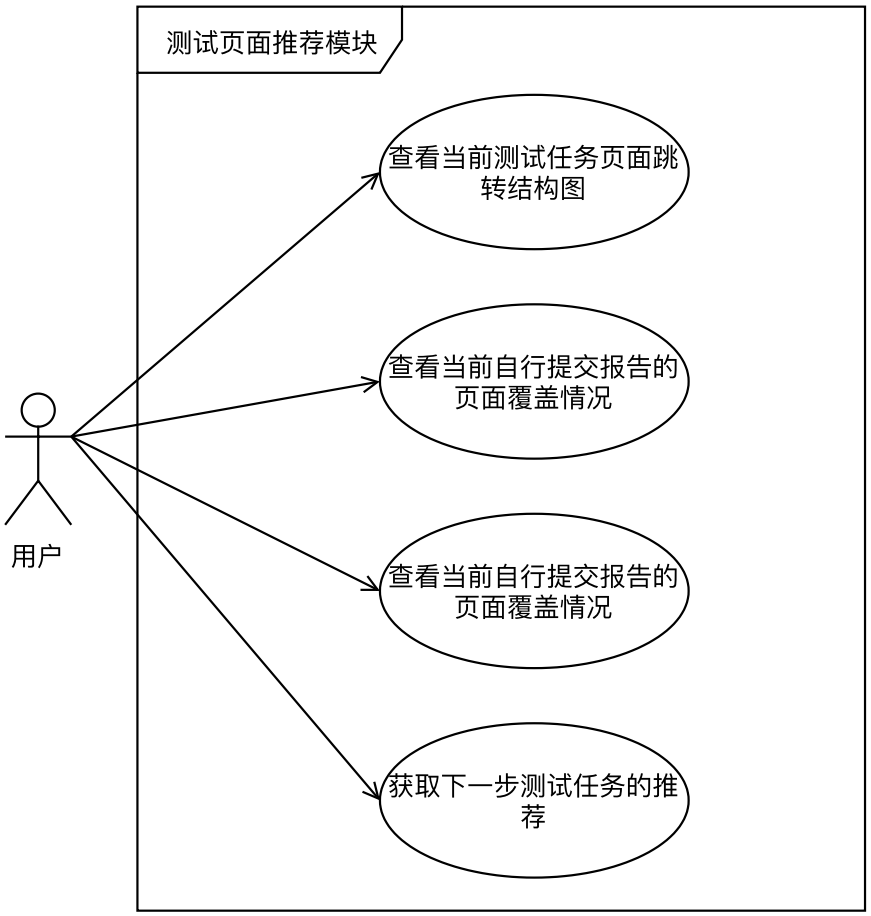

图 3.4: 测试页面推荐模块用例图

其中，第一种信息构成图表，另外三种信息会在该图表中以不同图例的形

式进行标注提示。作为一种可视化信息的展示，要求图表必须简单、直观，表现出页面间跳转的逻辑关系。

根据该模块用例的用户需求分析得到的用例如表 5.7所示。用户会在提交报告后，或自行点击查看推荐按钮时进入此模块。模块没有扩展流程，其主要是为了向用户展示当前的测试情况并根据情况结合用户历史进行测试工作推荐，但最终的下一步工作的决定权仍然在用户手中。

表 3.3: 测试页面推荐模块用例表

ID	UC_02
用例名称	测试页面推荐
用例目标	使用户了解当前测试进展 推荐下一步的测试页面
参与者	用户
前置条件	1a. 提交报告完成 1b. 点击查看推荐按钮
成功结果	用户根据情况，自行决定下一步测试计划
正常流程	1. 展示当前整体测试情况 2. 展示用户当前完成的测试情况 3. 推荐用户下一步需要测试的页面 4. 用户点击查看该页面下已提交的报告 5. 关闭推荐页面
扩展流程	无

3.1.3　审核报告推荐模块

审核报告推荐模块会在用户提交测试报告后与测试页面推荐在同一个弹窗中显示。如图 3.5所示，用户在该模块会得到类似于相似度推荐的审核任务列表。该列表的生成需要根据用户的历史提交记录、用户点赞、点踩记录以及Fork 记录生成，列表中不会存在用户已产生以上交互记录的报告。根据该报告被审核的次数进行排序，以解决因长尾效应导致用户交叉审核时部分报告难以得到审核的问题。对于推荐列表中的内容，用户同样可以在查看详细描述后，对其进行点踩或点赞操作。

根据该模块用例的用户需求分析得到的用例如表 3.4所示。需要注意的是，与上一小节类似，该模块也是有两种前置条件，一种是用户提交报告后自动弹

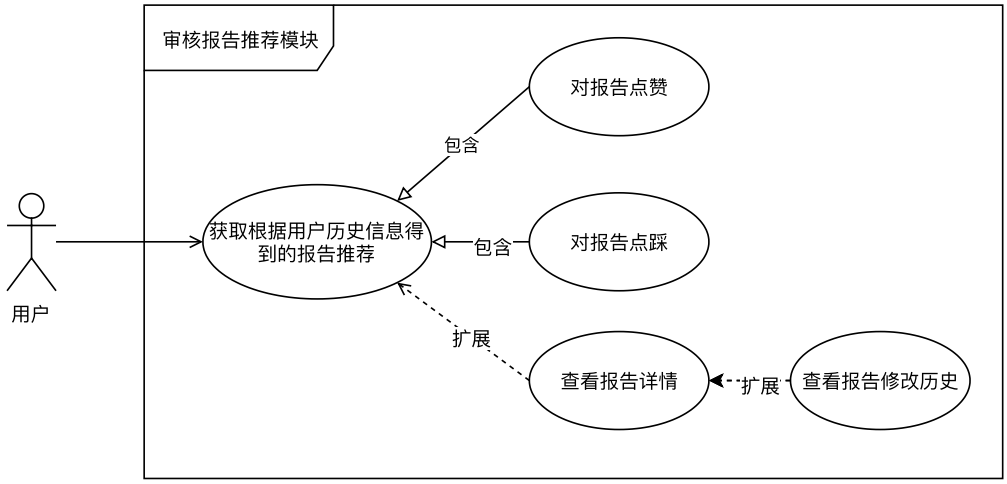

图 3.5: 审核报告推荐模块用例图

出，另一种是用户自行点击查看推荐按钮弹出。

表 3.4: 审核报告推荐模块用例表

ID	UC_03
用例名称	审核报告推荐
用例目标	为用户推荐待审核的报告
参与者	用户
前置条件	1a. 提交报告完成 1b. 点击查看推荐按钮
成功结果	对推荐报告完成审核
正常流程	1. 点击查看推荐报告详情 2. 按照推荐报告自行进行测试 3. 对测试报告进行点赞或点踩 4. 关闭推荐弹窗
扩展流程	无

3.1.4　系统非功能性需求

系统非功能性需求是整个系统需求的重要组成部分。其决定了系统中的架构设计，算法设计和工具选择。本系统作为一个实时推荐系统，主要的非功能性需求需求主要集中在以下几点：

1) 响应时间：在进行实时相似度推荐时，推荐列表要在0.4s内完成刷新，

否则推荐结果将失去时效性。对于任务分配，需要在0.5s内完成结果计算和图表绘制显示。

2) 并发操作：以某众测平台的一次众包测试任务为例，测试时长为4小时，同时有80 个账号同时登录进行众包测试。由于用户在进行报告填写时，每两秒就会向后端发送一次查询请求，故系统至少能够在一秒内满足100个用户同时进行查询操作。除此之外，系统还需满足100个用户同时进行提交报告操作。

3) 可靠性：对于用户的提交或者点击要有提示，对于用户的输入数据以及上传截图要进行检查，防止出现数据异常。对于用户的提交数据，需要进行备份，以防丢失。

4) 可扩展性：系统中的各类推荐算法以及结果排序算法要能够轻松进行替换和扩展。

3.2　系统概要设计

这一部分将从实现层面来对系统进行设计。首先会介绍系统的整体架构，说明各部分之间如何进行协作。然后以“4+1”视图模型对系统从不同角度进行概要设计，其设计目的是满足3.1小节的系统需求。

3.2.1　系统架构设计

本系统主要后端主要以微服务的形式对外提供RESTful的接口。除了与系统业务所需要的前端框架进行通信之外，还需要与外部平台进行通信，获取众包测试任务的信息和用户的数据信息。

如图 3.6所示，系统主要分为以下三个部分：

1) 采用Angular2框架的前端，该部分将会以独立前端的形式单独部署，打包上线。

2) 采用Spring Boot框架的后端，该部分采取MVC的视图模式。Controller层用于拦截前端发送的HTTP请求，并将对应的请求转发至对应的Service层进行处理。Service 层与外部平台的交互采用RESTful API的形式来进行，交互主要有两个方面：获取题目信息并存储到Redis之中，获取用户信息并存储到MongoDB中。这两种获取都会在Service 层在本地进行搜索后发现缺失相关信息的情况下才会进行，并且只会进行一次。

3) 采用Redis作为缓存，MongoDB作为数据库的存储部分。后端的Service层会根据当前数据的存储情况，选择是从Redis中读取查询缓存数据，亦或通

过MongoOperations从MongoDB中读取数据。最后，Service层会将数据库操作结果返回给Controller，由Controller决定返回给用户的数据形式。

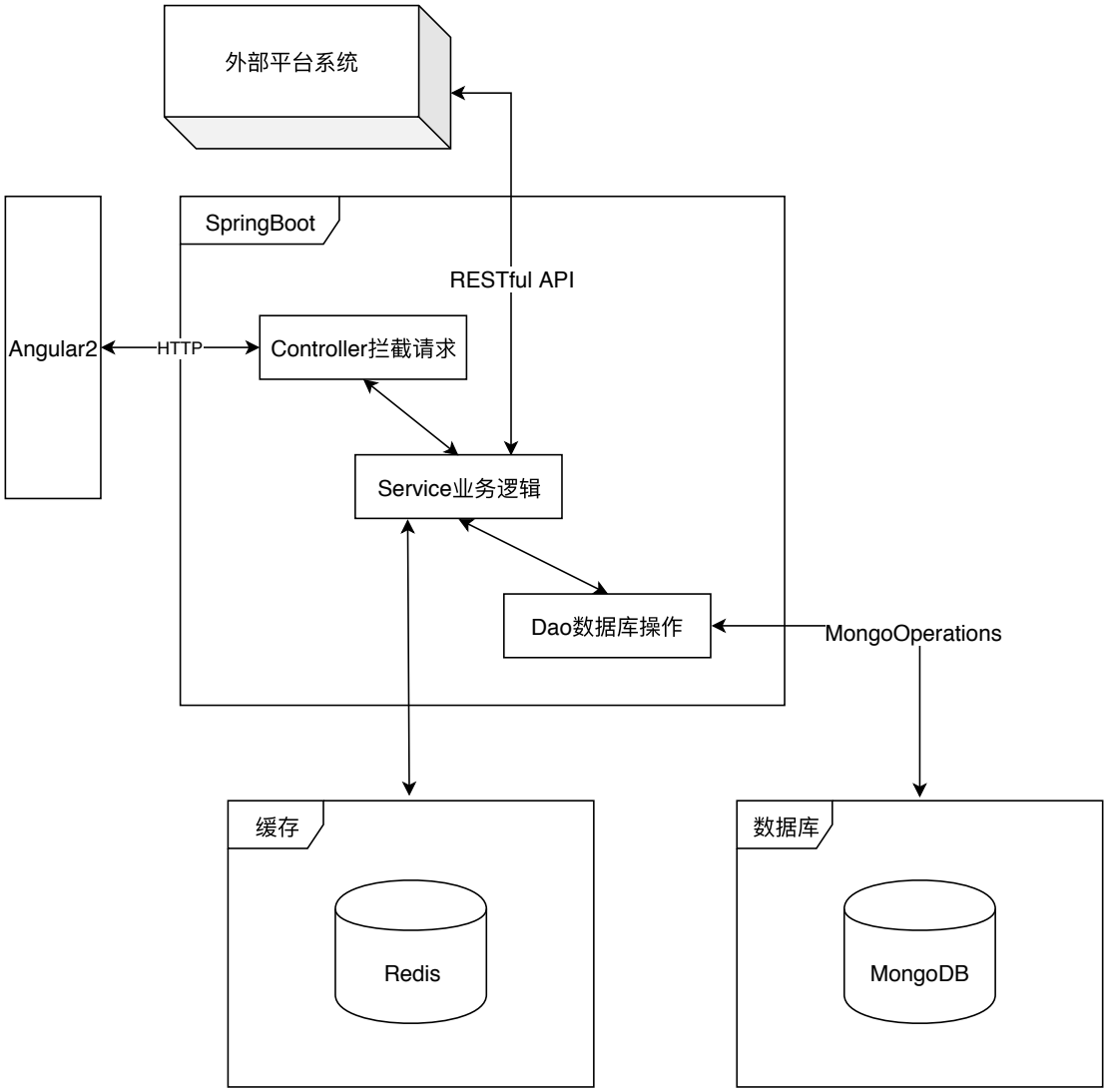

图 3.6: 系统整体架构设计

3.2.2　4+1视图模型

软件架构由元素、形式和关系三部分组成，用以处理软件高层次结构的设计与实施。“4+1”视图模型由以下五个主要视图组成：

1）逻辑视图：从用户服务视角出发，设计对象模型，在本文中对应3.2.3小节的实体类设计图。

2）过程视图：从集成人员视角出发，描述软件设计的并发与同步特征，在本文中对应第四章系统实现中各模块的时序图。

3）物理视图：从系统运维人员视角出发，描述软件到硬件的映射，反映了系统分布式的特征。在本文中对应接下来介绍的系统部署图。

4）开发视图：从程序开发人员视角出发，描述开发环境中软件的静态组织结构。在本文中主要分为前后端逻辑视图，即接下来介绍的系统前端工作示意图与后端包图。

5）场景视图：从系统用户视角出发，描述用户的使用场景，在本文中对应3.1小节中各模块的用例图。

系统整体部署视图如图 3.7所示，其中Web服务器、数据库服务器以及后端服务器均部署在阿里云上。用户通过Chrome等浏览器通过HTTP请求访问网站，由Web服务器做出页面响应。Web服务器中运行Angular2应用程序，其通过HTTP 请求访问后端微服务。后端服务器运行Redis缓存应用程序以及Spring Boot应用程序。其中，Spring Boot通过数据库连接访问数据库服务器。数据库服务器中运行MongoDB程序，将数据持久化。

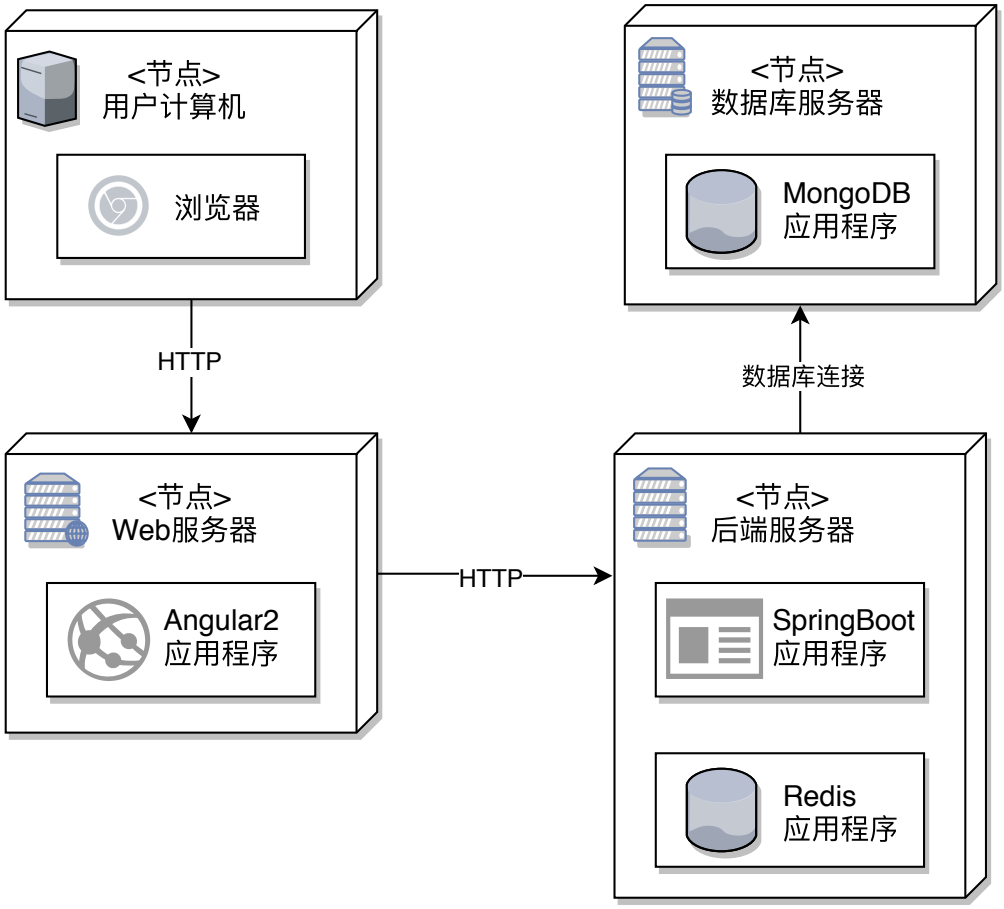

图 3.7: 系统物理部署视图

本系统的前端采用Angular2架构，该部分主要由以下八个部分组成：模块(Modules)、组件(Components)、模版(Templetes)、元数据(Metadata)、数据绑定(DataBingding)、指令(Directives)、服务(Services)、依赖注入(Dependency Injection)。如图 3.8所示，组件相当于MVC模式中的Controller层，模版相当于View层，依赖注入中的服务相当于Model层。在本系统中，模版采用HTML，其与组件通过元数据进行双向绑定。例如，当用户点击查看推荐时，由于模版与组件进行了事件绑定，所以会触发对应的事件。而组件又将请求转发给对应的服务，由服务负责与后端进行HTTP请求。同理，在服务获得推荐列表数据之后，再由对应的组件通过属性绑定，在模版上显示出最终的结果。

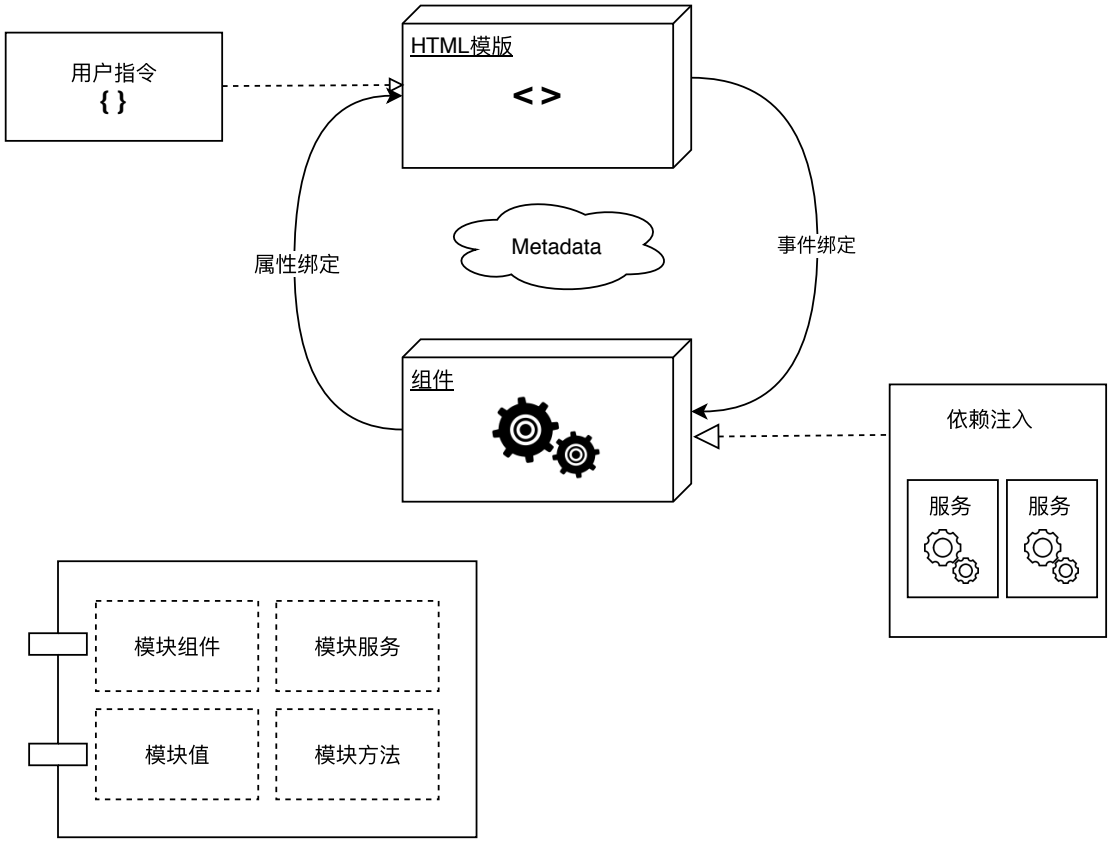

图 3.8: 系统前端工作示意图

本系统后端的包结构如图 3.9所示，其实现严格遵照MVC模式。

1）Resources包中包含了MongoDB连接池、Redis连接、Bean信息等本系统所需要的配置信息，信息以xml文件的形式进行存储。

2）Test包主要采用Junit4来进行单元测试和系统测试。它会在项目每次打包时自动运行，在测试全部通过后才会打包成功。

3）Controller、Service、Dao包的分别完成了HTTP请求的拦截与转发，业务逻辑以及MongoDB数据库的操作。这三个包中类的依赖关系通过Spring Boot提供的注解@Autowired方法进行依赖注入。Controller中注入了Service，Service中注入了Dao。

4）Entities包中有本系统所有的实体信息，每一个类对应了MongoDB数据库中的一张表。

5）Util包由Service中的类按需进行调用，其中包含了一些通用算法，如排序、随机ID的生成以及语料库的生成等。

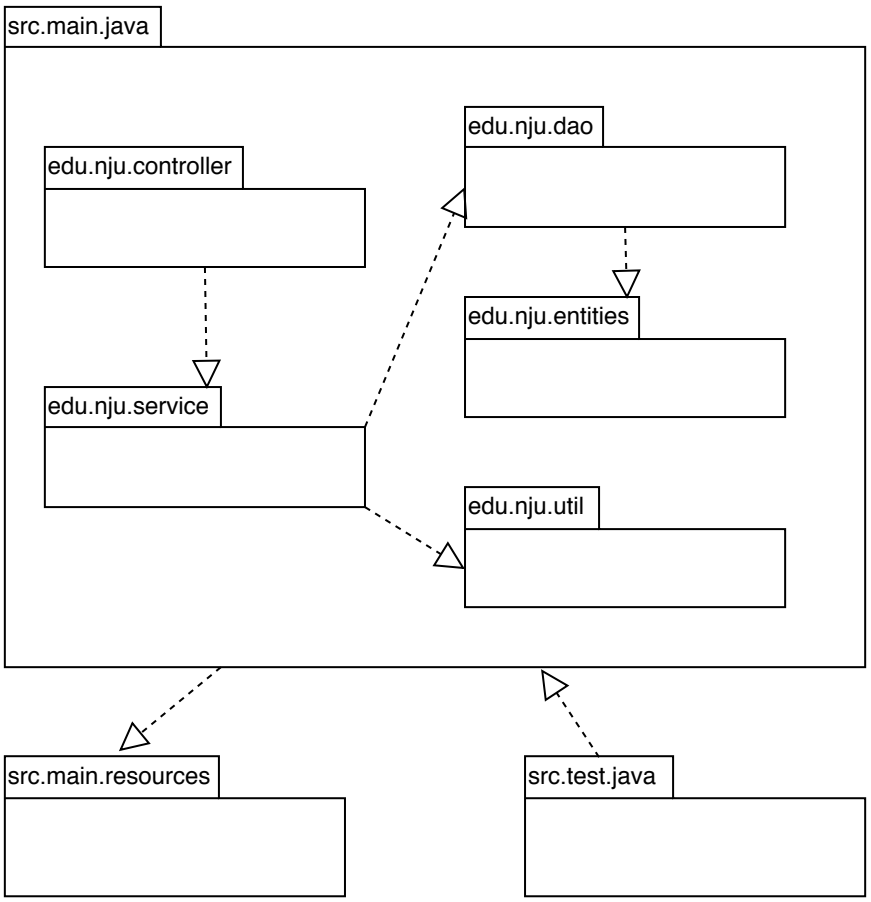

图 3.9: 系统后端包图

3.2.3　实体类设计

本系统的实体类如图 3.10所示，其中最重要的类就是Bug类。该类的属性与外部平台的Bug报告所包含信息保持一致，其中包含了Bug的所属案例标号、复现程度、严重等级、漏洞分类、详细描述、创建时间以及图片URL。同时，

每个Bug 类还包含了以下一些类的引用：

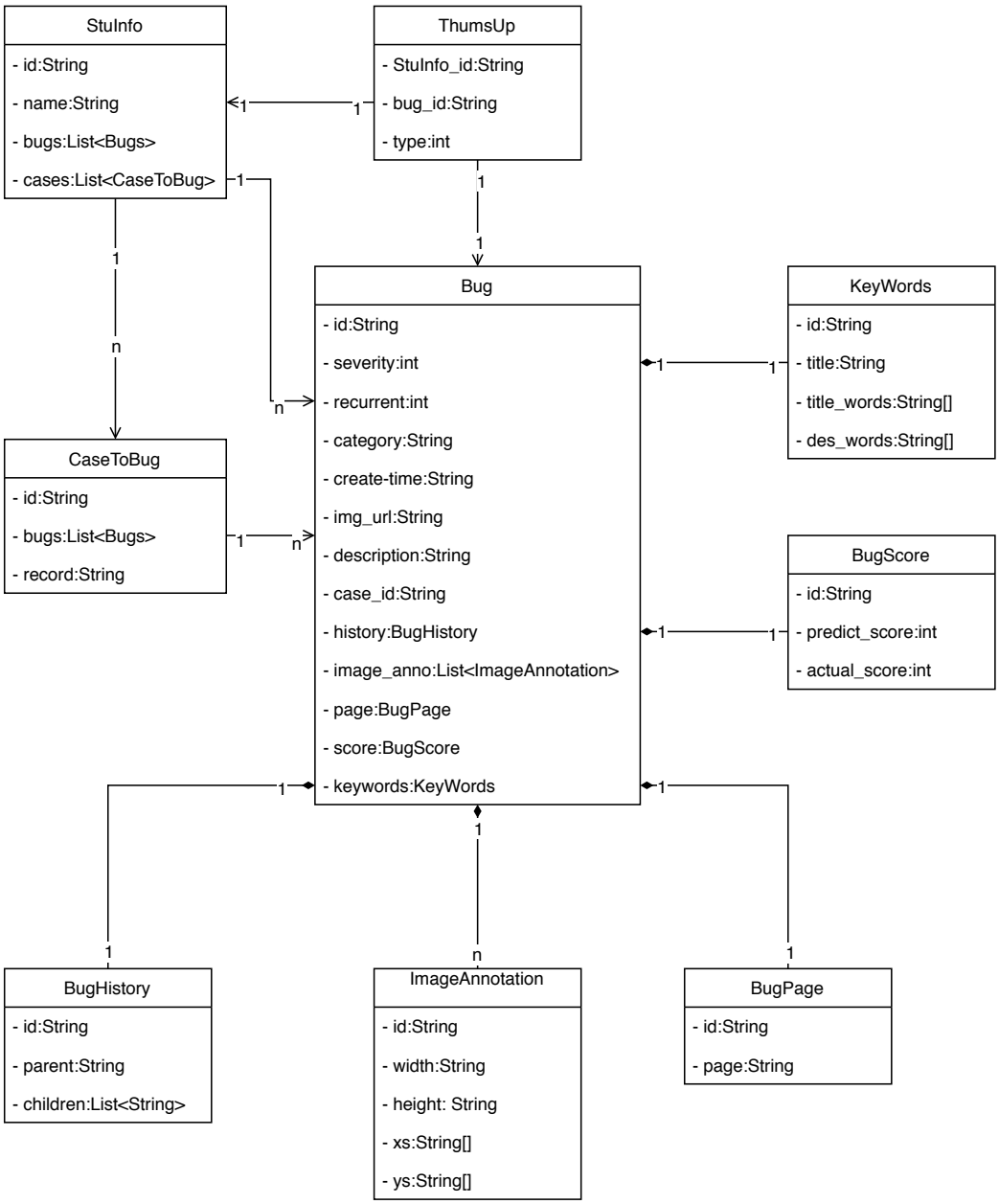

图 3.10: 系统实体类图

1) KeyWords类：该类记录了用户对于Bug的文字描述信息。首先，title属性保存了用户的对于此问题的概要描述，用于在推荐列表中使用户能够快速了解报告的内容，从而决定是否点开继续查看详细信息。由于系统实时性的要求，如果在用户填写时才进行分词，会大大影响运行速度，因而该类保存了用户的描述信息的分词结果，以String数组的形式进行存储。

2) BugHistory类：该类记录了用户所提交报告的历史版本信息，相当于树形结构中的一个节点。因为有Fork关系存在，每一个Bug报告都可能存在有唯一的前驱与诸多的后继。

3) 0至n个ImageAnnotation类：Bug类中对于该类的引用个数取决于用户一共上传的Bug截图个数。每张截图对应了一个ImageAnnotation类。该类记录了上传图片的宽高信息以及用户的绘图信息，以坐标对的形式进行存储。

4) BugPage类：该类记录了Bug发生的页面信息。单独将其保存是因为并非所有测试任务都有页面信息，同时，单独存储也可以加快检索速度。

5) BugScore类：该类用于记录任务发布方和系统自动化对于一份报告的评分信息。

系统会从外部平台获取用户的信息和测试任务信息，将用户信息保存在StuInfo中，其主要保存了用户姓名，同时包含了n个CaseToBug的实例以及m个Bug的实例，分别代表了用户参与了n次众包测试任务以及提交了m份报告。测试任务信息保存在CaseToBug类中，该类还包含了z个Bug的实例，代表在该测试任务下所有提交的报告。ThumsUp记录了用户点赞和点踩的信息，其中包含一个StuInfo的实例和一个Bug的实例，其代表了某位用户对某份报告进行了点赞或点踩操作。

3.3　算法设计

作为一个面向协作式众包测试的推荐系统，除了工程设计，算法设计也是很重要的一个方面。

3.3.1　协作方法设计

对于一份推荐报告，用户可以对其进行点赞、点踩与共同编辑三种处理方式。在初始时刻，每个参与者按照自己的测试情况提交测试报告。在测试任务进行的过程中，用户之间通过推荐的内容产生关联，又通过对于一份Bug报告采用以上的处理方式产生协作。系统将实时的提交情况通过推荐反馈给群体中的每一个体。个体根据当前的推荐结果，不断更新对于某一个Bug的描述。如图 3.11所示，上述过程一直持续下去，因为有共同编辑的关系存在，每一个Bug报告都可能存在有唯一的前驱与诸多的后继，最终呈现出一个树状的结构，该树状结构的叶子结点，就越趋近于对于这个Bug最为完整的描述。同时，一个报告也可能产生多个分支，从不同角度来完善对于一个问题的描述。

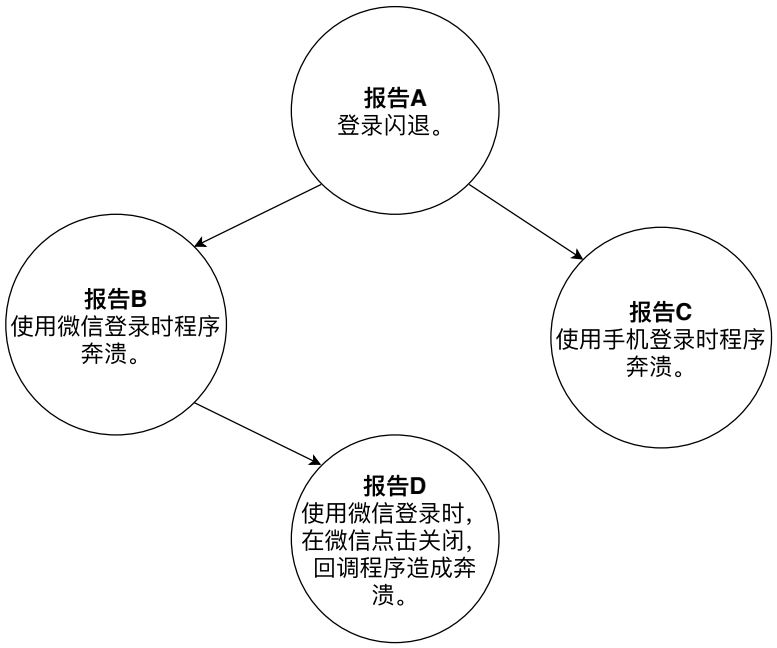

图 3.11: 协作方法设计

3.3.2　相似报告推荐算法设计

用户在提交个人Bug报告时，实时提示是否有相似的报告，以及具体相似度的计算值，避免用户提交重复报告。系统对于Bug的描述主要有两种方式，第一种是属性的选择，其中包括了页面路径、复现程度、所属类别以及严重等级。这些可选择的属性可以直接进行数据库索引匹配，筛选出的数据，再进行文本相似的计算，从而减少文本相似度的计算量。另外还有一些为自然语言描述，这是实现推荐的重点与难点所在。本系统中对其处理过程如图 3.12所示，其中，在构建词向量时，需要借助公开的语料库以及当前众测平台的众包测试数据来构建Word2Vec 的模型。两份报告的最终相似度由所处页面、Bug属性以及Bug 描述共同构成。

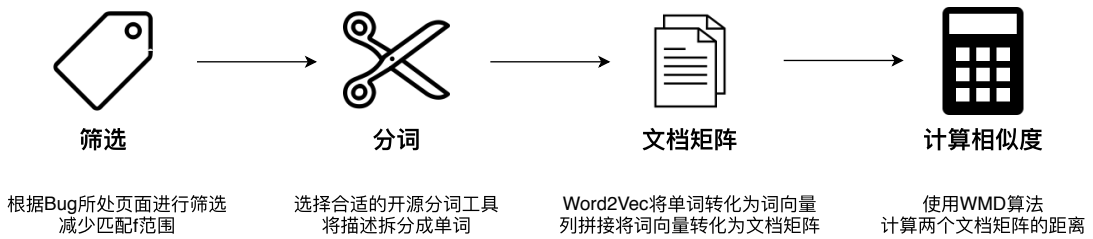

图 3.12: 相似报告推荐步骤

3.3.3　测试页面推荐算法设计

本部分主要需要将当前测试的实时情况展现给用户，当前测试情况的可视化包含了测试需求中的页面跳转结构，所有用户及当前用户已覆盖的页面。同时，根据用户历史提交记录来判断用户接下来在哪些页面进行测试会花费更少的页面跳转代价。

首先需要将测试需求转化为图结构。图中每个节点代表一个页面，有向边代表两个页面可以互相跳转的关系，权值代表两个页面间跳转的代价，即所经过的页面数，在前端完成页面跳转逻辑的可视化。计算待测推荐页面时，后端需要将图结构转化为邻接矩阵。以图 3.13为例，白色节点代表尚未有用户提交的页面，灰色代表非当前用户提交的页面，黑色节点代表当前用户提交过的页面。则用户覆盖了{2,5}，系统当前所有提交的报告覆盖了其中的{2,3,5}。推荐时，从节点2出发，从邻接矩阵中查找距离该节点最近的节点，为5节点，由于该节点用户已提交，则结束。开始向下遍历，由于3节点已有用户提交，故跳过，接着将4节点加入结果数组。向上遍历，将1节点加入结果数组。至此，2节点遍历结束。同样，再从5节点开始遍历，最终结果数组中有{4,1,6,7}，根据跳转距离排序后取前三，则最终推荐结果为{4,6,7}。

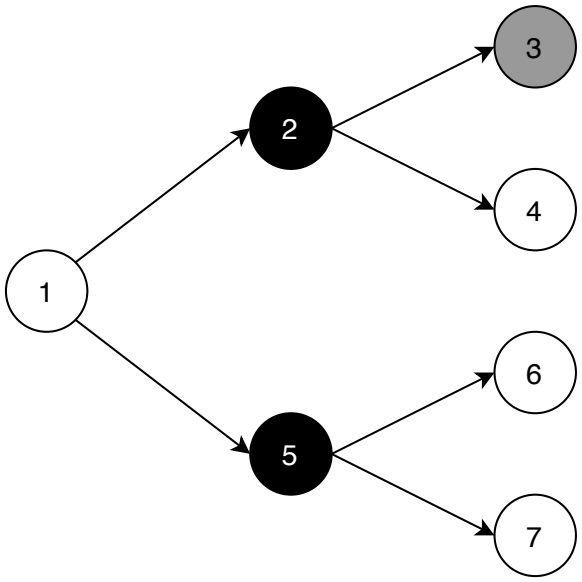

图 3.13: 测试页面覆盖情况示意图

3.3.4　审核报告推荐算法设计

任务分配主要是根据用户的历史行为记录以及当前报告被审核次数，推测用户可能还会关注某些问题。从而引导用户去测试验证相关的问题或对已测过

的报告进行评价，也可以使得因长尾效应难以被审核的报告引起用户的更多关注。该推荐将采用PMF 算法来根据用户已产生评价记录的报告特征来推断用户可能会感兴趣的其他报告。其中，本系统中用户历史行为的正反馈评分有点赞记录、提交历史与修改记录，负反馈评分有差评记录。

如图 3.14所示，影响一个用户是否喜欢某一份报告的因素有K个，评分矩阵可以分解为两个低维矩阵的乘积$R = U^T V$，其中$K \times N$矩阵U 描述N 个用户对于这K个因素的关注情况，$K \times M$矩阵V描述M份报告这K个因素的值。通过矩阵分解得到两个隐特征矩阵后，再利用其乘积得到所有用户对所有报告的评分矩阵。在本系统中，将报告被审核次数作为最后形成列表的排序依据，从而增加冷门报告被审核的几率。

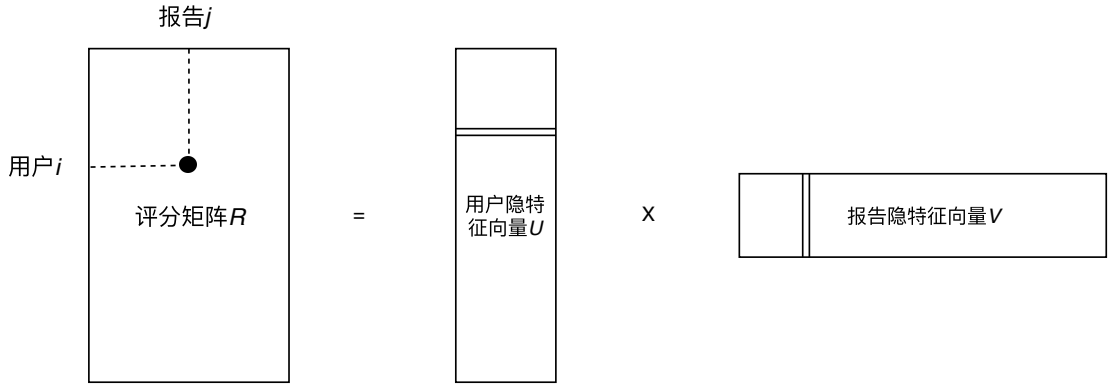

图 3.14: PMF算法示意图

3.3.5　推荐召回率算法设计

推荐系统的评价通常使用准确率和召回率，但由于众包测试任务的特殊性，无法预测推荐列表，导致系统很难从准确率上进行评价。因此，本系统主要记录用户对于系统推荐所做出的决策，从而便于从召回率上对本推荐系统进行评价。同时，系统可以以此作为指标，进行算法的调整。三种推荐方式分别需要记录如下信息：

1) 相似报告推荐，记录用户点击推荐的相似报告，与用户是否对其进行了点赞或者点踩。得出相似报告推荐召回率：

$$S_1 = \frac{\sum \text{用户对此报告点踩、点赞或Fork次数}}{\sum \text{用户点击报告查看详情次数}} \tag{3.1}$$

2) 审核报告推荐，每次进行推荐，记录推荐列表中的所有报告，用户每点击一份报告记录一次。得出审核报告推荐召回率：

$$S_2 = \frac{\sum 用户点击推荐报告次数}{\sum 总推荐报告数目} \tag{3.2}$$

3) 测试页面推荐，记录页面的推荐信息，与用户下一次的提交数据进行对比。得出测试页面推荐召回率：

$$S_3 = \frac{\sum 用户在推荐后的提交页面与推荐页面相同次数}{\sum 总推荐页面数} \tag{3.3}$$

在用户行为记录的前端方面，由于JavaScript I/O非阻塞的特性，所有行为记录都对原有业务逻辑的实现不会产生影响，与业务逻辑的实现并发执行。而后端实现方面，新建UserRecord表，表中属性及其含义如表 3.5所示。主要记录了某个用户在某个时间对某个内容所做出的行动。

表 3.5: UserRecord表属性定义

属性	含义
user_id	做出此行为的用户ID
target_id	用户所做出行为的对象，如报告ID
time	用户操作发生的时间
action	用户所作出的行为
remarks	行为备注信息

3.4　本章小结

本章主要首先从系统目标用户的需求出发，通过用例图和用例表等方法对系统报告提交模块、测试页面推荐模块和审核报告推荐模块的功能性需求进行了分析。接下来，从系统的实现层面，使用架构图，类图等方法对系统进行了整体概要设计。最后，对推荐系统中三种推荐方式所采用的算法以及推荐系统召回率计算进行了设计。本章为下一章系统的具体实现打下了坚实的基础。

第四章　推荐系统的详细设计与实现

本章将在第三章分析与设计的基础上，从细节上论述各个部分主要的实现方法。每个模块将首先结合具体页面实现截图说明此部分所实现的功能，然后会介绍业务逻辑的实现。最后，会介绍本部分所采用算法的具体实现。

图 4.1: 系统初始页面

由于系统是基于Angular2的应用，所以除在对页面截图进行圈注时会进行页面跳转外，所有的功能都会在页面中利用Angular2的Module决定页面所需要展示的内容，不会有任何的页面跳转发生。用户在接受众包测试任务并填写运行环境信息后和用例设计后进入到如图 4.1所示的本系统页面中。

此页面是系统的基础页面，用户在第一次进入本系统，或提交完报告后都会进入此页面。页面展示了用户所有已提交的报告，同时，通过点击按钮，可以进入以下三种子页面：

1) 点击右上角的创建Bug按钮，报告填写模块出现。

2) 点击任务分配按钮，测试页面推荐模块和审核报告推荐模块以模态框的形式出现。

3) 点击Bug的查看按钮，查看自己已提交报告的详情，此部分复用报告填写模块代码，但对输入框进行了限制，无法对已提交内容进行修改。（为防止用户根据其它用户对于自己报告的修改，再次修改个人报告，干扰平台对于用户贡献度的计算。用户可以选择Fork自己的报告后，再修改）

4.1　报告提交模块详细设计与实现

关于本模块所需要完成的功能已在3.1.1小节进行了用例描述。用户在点击创建报告后，触发对应的Angular服务，在HTML模版中显示如图 4.2所示的报告填写页面。本模块主要分为四个区域，左上角，根据已提交的Bug ID，输入后一键填充到当前填写页面；左侧，填写Bug的详细信息；左下角，选择需要上传的图片；右侧，实时推荐列表。其中，左侧的页面数据通过访问外部平台的OSS拿到如表3.4所示的EXCEL表后，通过js进行解析后，使用级联选择器来完成显示。

图 4.2: 报告提交模块页面

4.1.1　报告填写的实现

用户在图 4.2页面的左侧，可以进行Bug的所属页面，漏洞分类，严重等级，复现程度，所属用例的选择，然后对Bug进行概要描述，最后对测试过程和结果进行详细描述。在这些选择和文字输入的过程中，右侧的推荐列表都会随着输入内容的不断完善，而不断精确推荐最相似的报告。

系统会将用户对于页面的选择顺序和所得到的结果记录到Redis中。由于采用级联选择器，当用户重新选择了上级页面时，或关闭创建报告页面时，缓存

自动失效。

此处以用户点击所属页面为例，来说明推荐列表如何根据用户的所选内容进行动态变化。至于在用户进行文字描述时，如何进行文本相似度的计算，会在4.1.3 小节中进行介绍。如图 4.3所示，用户在选择一级页面后，会触发前端的choice_service，由其负责向后端发送HTTP GET请求，recommendController在接收到请求后，调用recommendService中的ByPage()方法。该Service会先查看Redis中是否有该用户的选择结果信息，如果存在，则直接在这个结果之中进行更加精确的分类。该时序图中展示的是当Redis缓存中不存在该用户的记录时，则会调用bugDao的findByPage()方法，由其使用MongoOperations对MongoDB数据库进行查找后，返回用户选择的该页面下所有的Bug。Service在接收到结果后，会将该结果连同用户此次的选择信息保存至Redis，以便下次用户再次选择时加快查询速度。由于查询的结果是无序的，所以Service还会调用sort方法对其根据提交时间进行排序后，将排序完成的结果返回给Controller。Controller会把结果数组转化为JSONArray返回给前端服务，前端服务再利用属性绑定在对应的HTML模版上显示出推荐的列表。

Redis中的页面筛选缓存结果会在用户关闭创建Bug、提交Bug或者重新选择不同的一级页面时失效。

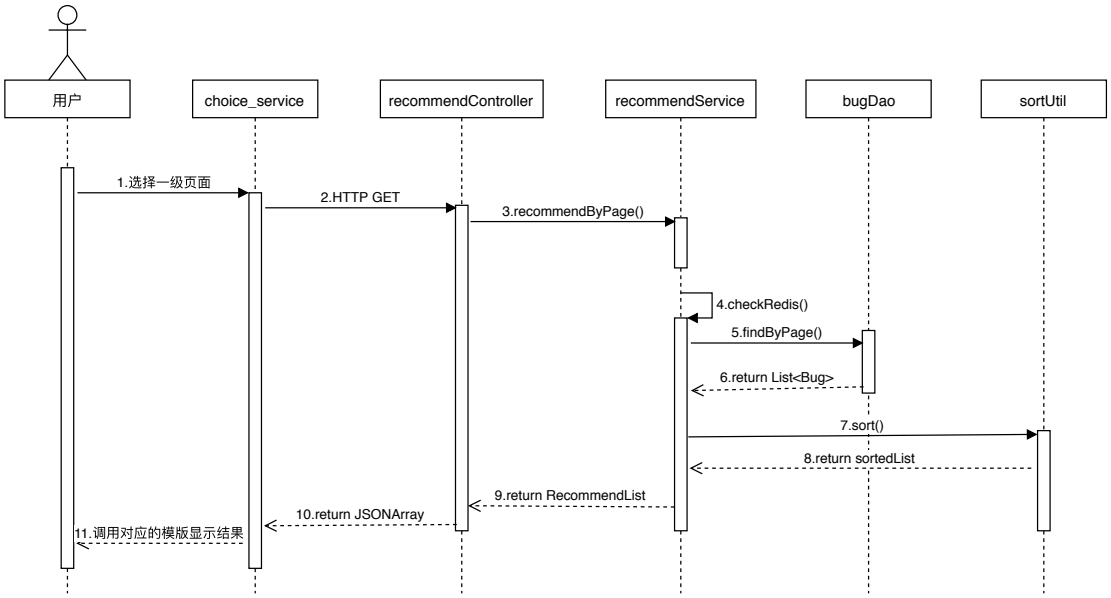

图 4.3: 页面选择时序图

4.1.2　用户协作的实现

为了使用户快速交互，减少输入，因此用户首次进入创建Bug页面，详细

描述与图片上传默认不显示，用户可以点击填写时再次出现如图 4.1所示的详情填写。用户在点击推荐列表中的任意一条推荐信息后，通过Angular2的事件绑定，加载出该报告的详细信息。点击之后的页面如图 4.4所示。左上方为Bug的详细描述，右上方为报告的截图信息，左下方为用户的交互部分，右下方为该报告的Fork历史的可视化展示。在本页面，用户可以对历史信息中的节点进行点击，点击之后，该页面会显示用户当前所点击节点的报告。用户点击Fork按钮，则会将该报告的所有内容自动填充到左侧正在填写的报告中。通过这种方式，用户无需再填写相同的内容。以图4.4所示，记用户所Fork的报告为A，其已经拥有2个版本的Fork记录。用户Fork报告A，修改并提交后的报告为B。则报告A的children数组扩大至3。报告B的parent属性为报告A的ID，其children数组为空。

图 4.4: 查看报告详情及用户协作页面

如图 4.5所示，用户在点击推荐列表中的任一报告后，会触发对应的服务向后端发送请求。后端在接收到请求之后，主要分为三个步骤来进行：

1) 使用BugDao获取报告的详细描述。

2) 判断用户是否对于该报告已经点赞或点踩。

3) 获取报告的历史信息，即有哪些报告是通过Fork它修改后得到的，以及它又是由哪份报告修改得到的。

在完成这三部分之后，将结果整合进JSONObject返回给前端，前端得到数据后，再将其中的Bug JSONObject转化为前端对应的对象，从而从中获取到该报告的image_url信息，再去访问oss，进行显示。

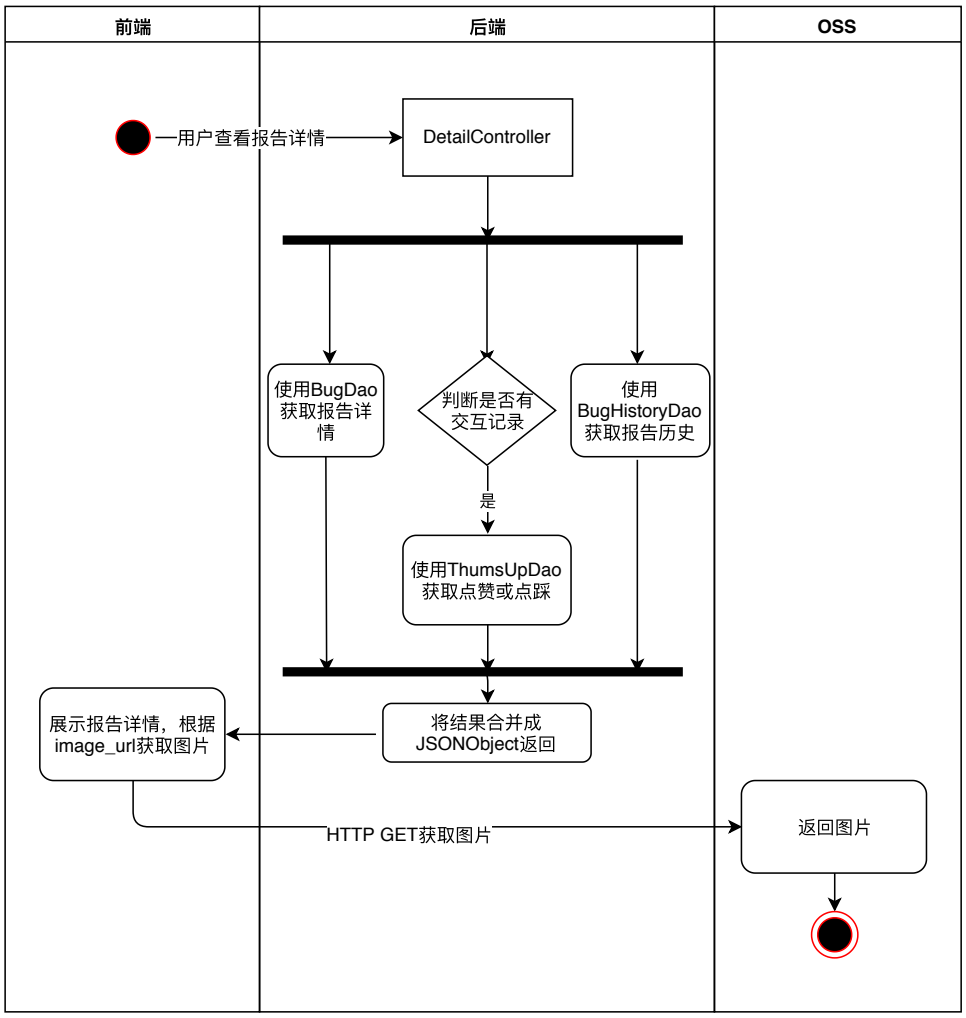

图 4.5: 用户查看报告详情活动图

报告历史记录的可视化生成，后端通过BugHistory类来实现。该类记录了Bug的历史信息，每条BugHistory都记录了这份报告的唯一parent（父报告）、多个children（子报告），唯一所属root（根结点）。因此，当用户点击某份报告时，由于该报告的BugHistory记录了其所属根节点，因而可以直接从根节点开始使用递归算法向下进行遍历。先获取该报告的子节点数组，然后对数组中的所有子节点进行深度遍历。使用该方法无需由该报告根据其存储的父节点，向上进行遍历，大大加快了查找速度。具体的实现代码如图 4.6所示，dfs为递归方法，getDepth为获取报告的历史，返回的是报告ID 的List⟨*String*⟩ 的数组，其

```
public List<List<String>> getDepth(String id) {
    BugHistory root = historydao.findByid(id);
    List<List<String>> result = new ArrayList<List<String>>();
    if(root == null) { return result; }
    List<String> list = new ArrayList<String>();
    list.add(root.getId());
    dfs(root, result, list);
    return result;
}
private void dfs(BugHistory root, List<List<String>> result, List<String> list) {
    List<String> children = root.getChildren();
    if(children.size() != 0) {
        for(String child : children) {
            list.add(child);
            dfs(historydao.findByid(child), result, list);
            list.remove(child);
        }
    } else { result.add(new ArrayList<String>(list)); }
}
```

图 4.6: 获取报告历史记录代码

中的每一个数组代表树结构中从根节点出发，到其中一个页节点的路径。前端再使用Echarts的Tree将该数组转化为可视化的图结构。

用户在点击Fork按钮之后，并未向后端发送任何请求，前端会将该报告的类属性直接赋值给填写报告的类属性，从而触发属性绑定，在页面上显示出内容。同时，前端还会记录下用户正在填写的报告Fork自该推荐报告，并且在用户点击提交时将该信息上传。

用户点踩或点赞会提示交互成功，记录一条ThumsUp数据，在用户再次查看详情时，只显示点赞或点踩的图标。再次点击图标，可取消点赞或点踩，删除数据库中的记录。

4.1.3　相似报告计算的实现

在进行相似度报告推荐之前，先要利用语料库训练出词向量。本系统采用搜狗文本语料库与移动应用众包测试历史数据训练生成。预料格式为一个txt文本文件，每行是一个句子分词后的单词列表，词与词之间用空格进行分割。众包测试数据利用Ansj进行分词，生成文本文件后，使用Hanlp中的Word2Vec包进行词向量的训练，训练过程如图 4.7中的代码所示。TRAIN_FILE_NAME为语料库文件，MODEL_FILE_NAME为训练完成的词向量文件。在指定完这两个参数之后，利用hanlp中Word2VecTrainer进行训练，每个单词会生成100维的向量。

在完成从语料库到词向量的转化之后，程序启动前的数据准备阶段结束。

```
static WordVectorModel trainOrLoadModel() throws IOException {
    {
        if (!IOUtil.isFileExisted(MODEL_FILE_NAME))
        {
            if (!IOUtil.isFileExisted(TRAIN_FILE_NAME))
            {
                System.err.println("语料不存在");
                System.exit(1);
            }
            Word2VecTrainer trainerBuilder = new Word2VecTrainer();
            return trainerBuilder.train(TRAIN_FILE_NAME, MODEL_FILE_NAME);
        }
        return loadModel();
    }
}
```

图 4.7: 语料库生成词向量模型代码

该词向量文件会在系统开始运行时，自动加载进内存。

整个推荐过程如图 4.8所示，用户在题目或描述中进行文字输入时，触发文本相似度计算：

1) 前端每隔两秒检测当前输入框中的内容与用户上次的输入是否一致，如果有改变，则向后端发送一次查询请求。

2) 系统会先判断用户是否有选择页面信息，如果进行过页面选择，则会在Redis 中获取通过页面筛选后的Bug数组。如果没有选择页面，则会从MonogoDB 中获取该测试任务下所有报告，生成Bug数组。

3) 为了加快系统响应速度，达到3.1.4小节所描述非功能性需求中的响应时间的需求，系统在用户在用户提交Bug时就已经对描述进行了分词，并将分词结果保存在Keywords类中。所以，需要遍历Bug数组中的Keywords，同时，对用户的输入结果进行分词，分别构建文档矩阵。

4) 利用WMD算法，计算用户输入与所有Bug的文本相似度，结合所在页面与Bug 属性计算最终的相似度分数。利用sortUtil对结果按分数高低进行排序。

相似度的最终计算不仅仅由Bug描述的文本相似度一项构成，还由Bug标题的文本相似度、Bug所在页面，Bug属性这四项共同构成。最终相似度的计算公式如下所示：

$$S_{nm} = \frac{\sum_{i=1}^{l} k_i R_i}{\sum_{i=1}^{l} k_i} \times 100\% \tag{4.1}$$

其中，S_{nm}代表第n份报告与第m份报告的相似度。l代表计算相似度时共计l个因素。由于部分报告的所在页面停留在一级页面，部分则停留在二级或

三级页面，导致进行比较的报告页面属性数量不完全相同。因此，l需要取缔n份报告和第m份报告所包含因素的最大值，即$l = Max(l_n, l_m)$。i代表第i个因素，k_i代表第i个因素在计算最终相似度时所占比重，为常数。

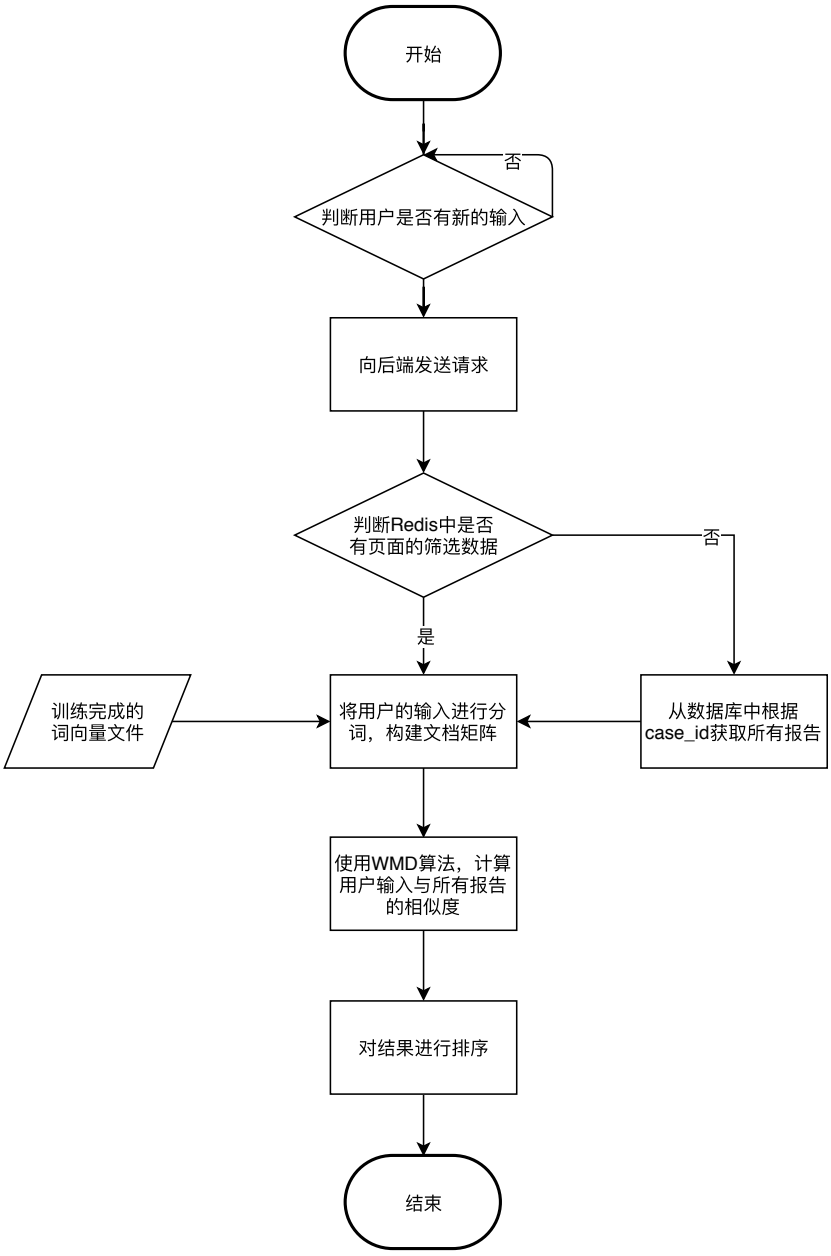

图 4.8: 相似报告推荐流程图

R_i代表报告在第i个因素上的得分，得分采用百分制。k_i取值如表 4.1所示。

如图 4.9所示，当用户所提交的报告与推荐报告的相似度超过90% 时，右侧推荐列表会使用黄色进行显著标注，提示用户可能会提交了重复报告。并且

表 4.1: k_i取值表

i范围	R_i取值	k_i取值
1~3	报告所属页面相同为100，不同为0	0.05
4~6	报告属性相同为100，不同为0	0.05
7	报告题目相似度(0~100)	0.3
8	报告描述相似度(0~100)	0.4

在用户尝试强制提交时，给予弹出页面提示，警告用户这样可能会产生重复报告，是否进行提交。

图 4.9: 相似度过高提示页面

举例说明: 报告A = {“一级页面”:”搜索”，“漏洞分类”:”页面布局缺陷”,”严重等级”:”紧急”，“复现程度”:”必现”，“题目”:”搜索结果页面出现大面积留白”，“描述”:”点击搜索后，结果页面出现大面积留白”}

报告B = {“一级页面”:”搜索”，”二级页面”:”搜索30030554”，“漏洞分类”:”页面布局缺陷”，”严重等级”:”紧急”，“复现程度”:”必现”，“题目”:”搜索结果页面出现大面积留白”，“描述”:”按要求输入30030554，点击搜索后，结果页面出现大面积留白”}

报告A与报告B，一级页面相同，二级页面不同，三级页面均不存在，Bug属性全部相同，题目相似度100%，描述相似度93.25%，如式4.2所示的相似度计算过程为:

$$S_{AB} = \frac{0.05 \times 100 \times 4 + 0.3 \times 100 + 0.4 \times 93.25}{0.05 \times 5 + 0.3 + 0.4} \times 100\% \approx 91.89\% \tag{4.2}$$

4.1.4　截图标记与上传的实现

用户在点击选择图片文件后，点击上传截图完成图片的上传。图 4.10展示了用户在点击图片上传之后前端的调度时序图。用户点击上传截图会触发Angular2 HTML模版对应的事件绑定，从而调用upload_pic服务。图片并不会保存在系统中，而是会集中存储在外部平台所提供的OSS中，系统仅会保存图片的URL信息。OSS保存成功后会返回图片的URL，URL命名方式为IP+创建时间+图片名称，这样可以确保URL的唯一性。下面就是一个图片URL的示例：http://oss.com/app/154245475/201811171938.png。图片上传模块在通知用户图片上传完毕后，还会调用缩略图显示的组件，通过属性绑定，在对应的HTML模版中显示用户上传图片的缩略图。

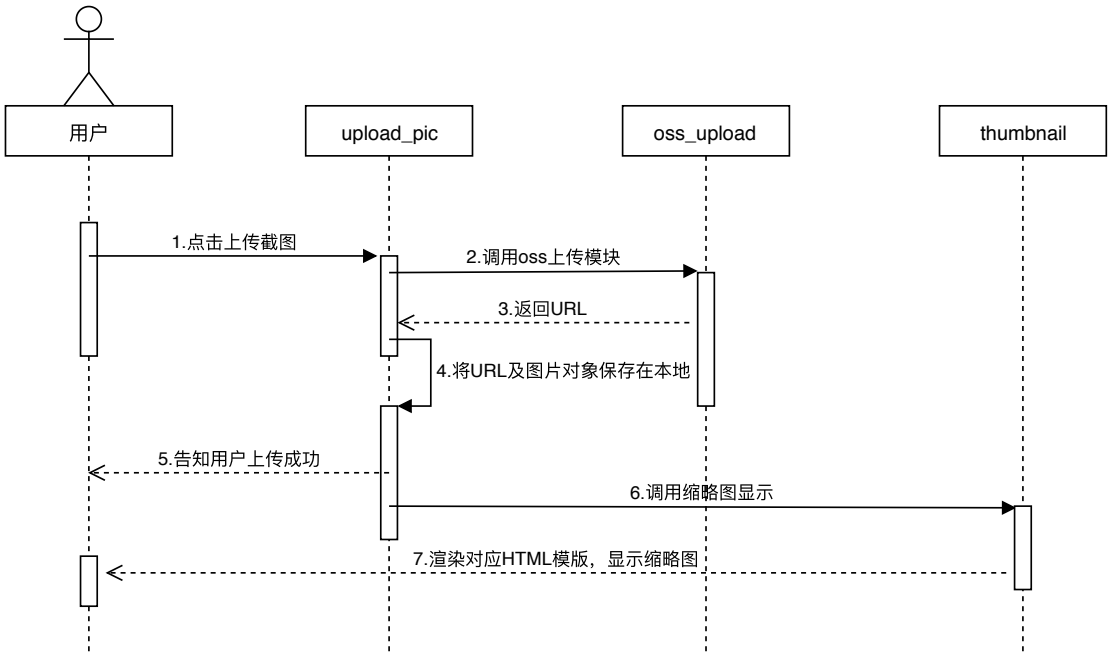

图 4.10: 图片上传模块时序图

在页面成功显示缩略图后，可以继续点击已上传的图片进行图片信息的标注，如图 4.11所示。显示出来的连续鼠标痕迹实际是由n个点的位置信息，进行连线后形成的轨迹。

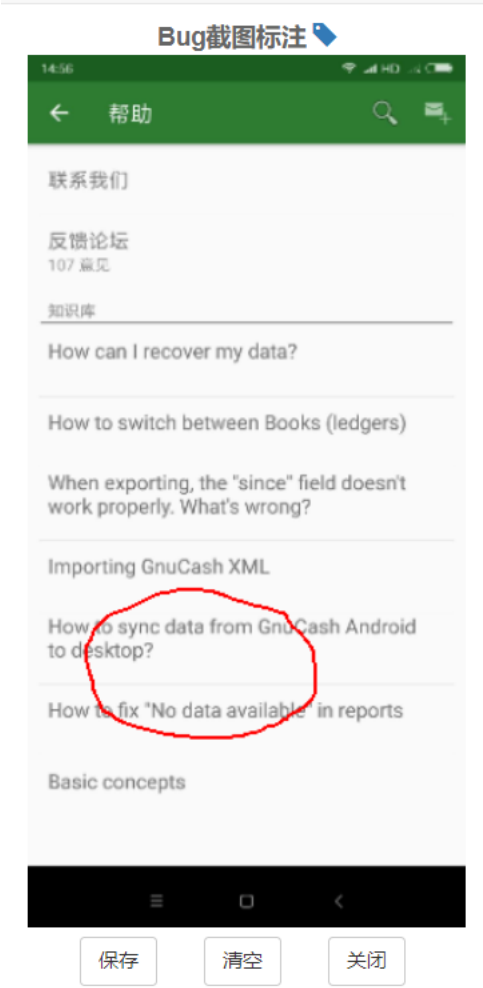

图 4.11: 截图标记与上传页面

点击标注之后，会调用OpenNewWindow方法，在浏览器中再打开一个新的页面进行图片标注。系统会根据上传图片的宽高，创建一个同样大小的2D canvas画布对象。首先将图片填充进画布，然后，当用户在图片上进行点击时，触发canvas 的onmousedown事件，记录当前点在画布上的的x, y坐标对起始位置信息。鼠标开始移动时，触发onmousemove事件，调用如图 4.12所示的draw方法，首先将记录的用户上次鼠标所在的点与此时监测到鼠标所在的位置两点连成一条直线，然后将当前鼠标位置信息存入xs和ys坐标数组。记录的坐标信息需要减去canvas的起始坐标，得到坐标的偏移量，记录偏移信息。鼠标松开时，触发onmouseup事件，停止记录鼠标移动的信息。当用户点击保存时，上传图片URL，图片长宽高，以及xs和ys 点的集合数组。在用户再次点击图片时，会将图片与点共同显示在canvas画布上，并且将所有点按照存储的顺序进行连线，形成标注的记录。

```
function draw(ev) {
        if (!ev) var ev = window.event;
        if (mouse_be_down == 1) {
            drawing = 1;
            the_canvas_context.lineTo(ev.pageX-the_canvas.offsetLeft,
ev.pageY-the_canvas.offsetTop);
            the_canvas_context.stroke();
            xs.push(ev.pageX-the_canvas.offsetLeft);
            ys.push(ev.pageY-the_canvas.offsetTop);
        }
    }
```

图 4.12: onMouseMove响应事件代码

4.2　测试页面推荐模块详细设计与实现

本模块主要用于在众包测试的任务执行阶段的整个过程中，系统为用户提供测试页面覆盖情况，使用户能够清楚了解测试页面的覆盖情况。用户还需要根据系统推荐，对下一步的测试做出计划。其实现主要分为两部分，首先需要可视化测试页面覆盖信息，同时，根据用户已提交的信息，为用户推荐下一步的测试页面。

4.2.1　测试实况可视化的实现

```
option = {
    title: { text: 'Graph 简单示例' },
    series : [{
            type: 'tree',
            label: { normal: { show: true }},
            data: [{name: '节点1', children:[{name: '节点2'}]}]
    }]
};
myChart.setOption(option);
```

图 4.13: Echarts Option配置代码示例

本部分使用Echarts中的Tree完成实现。Echarts中数据输入通过配置项形式使用setOption()方法完成，且支持动态数据渲染，即，当数据变化时，再次调用setOption()方法，只会将此次数据中与上次数据对比的不同之处重新进行渲染。而Tree相较于普通Graph而言，无需指定点在图中的位置，位置信息会自动计算。其中，图的数据输入，即配置项信息如图 4.13所示，输入数据为一种嵌

套格式，当前节点唯一标识为”name”的值，children记录了当前节点下的所有子节点。最后通过echarts对象将该option设置进去，Tree会自动刷新并显示。

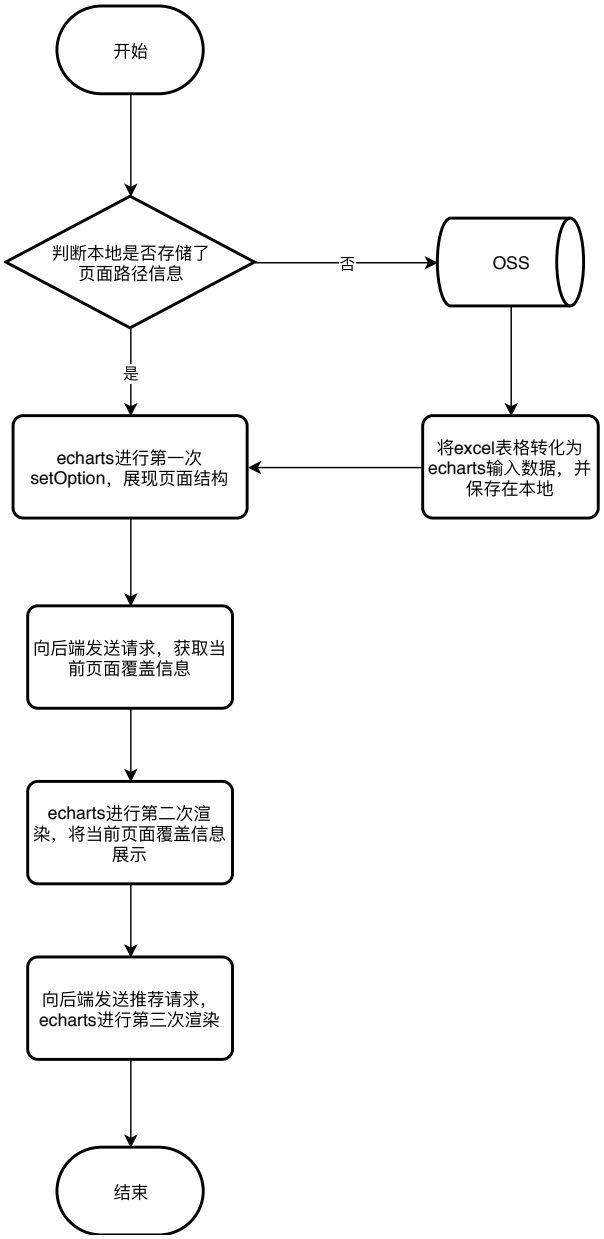

图 4.14: 可视化形成流程图

如图 4.14所示，整个推荐图的生成一共需要渲染三次，即需要获取三次数据，并将其装载到option的data中。

1) 页面路径结构信息：这是图最基本的信息，需要判断前端本地是否已经存储了该信息，如果没有存储，则需要访问外部平台的OSS获取如表3.4所示的测试需要EXCEL，然后利用JavaScript中的xlsx第三方库完成EXCEL的解析，生

成Echarts中的data数据，并进行第一次的图渲染。

2) 当前页面覆盖情况：前端向后端发送请求，获取当前全部用户和目标用户的覆盖情况，后端返回如下所示的JSON数据，将覆盖统计信息加入data数据，并进行第二次渲染。

{”all”:{”热门攻略”:1,”品质团”:3,” 养生温泉”:4,”牛人新品”:25},”self”:{” 热门攻略”:1, ”养生温泉”:1,”牛人新品”:1}}

3) 目标用户推荐结果：前端再次向后端发送请求，获取目标用户的推荐测试页面信息，将推荐的文字描述赋值给Echarts对象的title，页面数组加入data数据，并进行最终渲染。

4.2.2　测试页面推荐的实现

表 4.2: 页面结构样例

一级页面	二级页面	三级页面
搜索	推荐目的地	地区选择
		查看更多

测试任务的推荐主要使用邻接矩阵的存储方式与最短路径算法共同完成。首先，后端从前端获取到页面结构JSON数据后，构建页面与页面之间跳转复杂度的邻接矩阵。以表 5.10中的页面结构为例，会形成如下所示4×4 的邻接矩阵S。

$$S = \begin{Bmatrix} 0 & 2 & 3 & 3 \\ 2 & 0 & 2 & 2 \\ 3 & 2 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{Bmatrix} \tag{4.3}$$

其中，横纵坐标x =｛搜索，搜索，推荐目的地，地区选择，查看更多｝= y^T。S_{ij}代表页面i到页面j所需要的跳转次数。以“地区选择”页面为例，除与自身距离为0外，属于同一级页面下的“查看更多”所需跳转次数最少，其值为1。与上一级页面“推荐目的地”距离为2，上上级页面“搜索”距离为3。在进行多源最短路径计算时，从用户已提交的节点出发，分三部分进行深度搜索。首先是同级页面，即距离为1的节点。然后向上深度搜索，直到到达一级页面。最后进行向下深度搜索，直到到达三级页面。将结果保存至数组，最后进行排序后，取距离最短的前三页面。

假设推荐目标用户已提交的报告所覆盖的页面集合为$my(x)$，所有页面的集合为$target(x)$，所有用户已提交报告所覆盖的页面集合为$all(x)$。这三个集合的从属关系如图 4.15所示。其中，$my \subseteq all \subseteq target$，$R_1 = target - all$，$R_2 = all - my$。

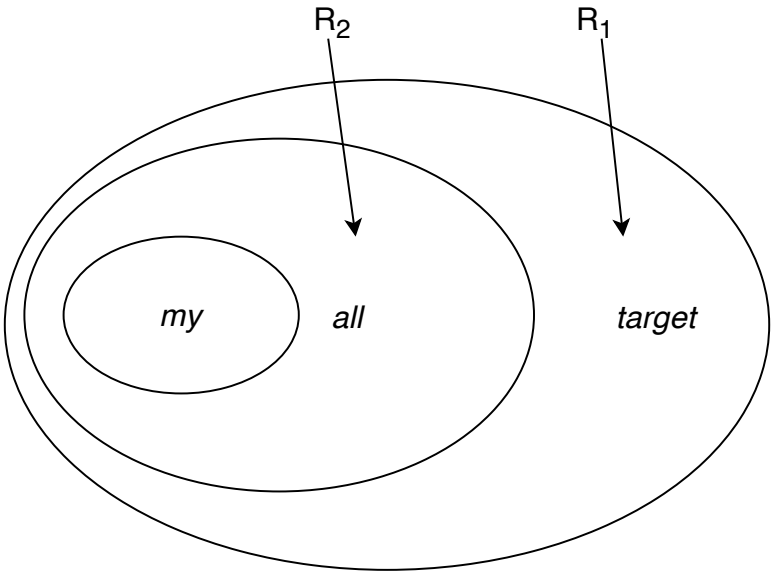

图 4.15: 页面集合示意图

在进行推荐时，优先推荐集合R_1与集合my中按用户提交历史计算所得出跳转耗费最少的页面。当$all = target$ 时，再推荐集合R_2与集合my中多源最短路径计算所得结果。这样，可以首先利用用户群体，完成整个测试任务需求中的页面全覆盖。然后，引导用户个人完成所有待测页面全覆盖。

4.3　审核报告推荐模块详细设计与实现

本模块主要介绍待审核列表的生成过程。该模块利用用户在测试过程中产生的协作数据，推测用户行为所代表的潜在隐特征向量，系统根据隐特征向量，为用户未产生交集的测试报告进行评分，根据评分排序后为其推荐待审核的测试报告。对于推荐的审核任务，用户需要在自行验证后，对其进行点赞或点踩操作，至此完成对于推荐报告的审核。本模块主要介绍利用用户在测试过程中产生的行为数据，推测用户行为所代表的潜在隐特征向量，系统根据隐特征向量，为用户未产生交集的测试报告进行评分，根据评分排序后为其推荐待审核的测试报告。对于推荐的审核任务，用户需要在自行验证后，对其进行点赞或点踩操作，至此完成对于推荐报告的审核。

4.3.1　用户评分矩阵构建

概率矩阵分解是基于模型的协同过滤的一种实现方式。在本系统中，算法输入主要由以下六部分组成：

1) N：用户数目

2) M：报告数目

3) K：报告的特征数目，在本系统中，报告主要由所处页面，报告属性和文本描述3 部分组成，故$K = 3$为常数。

4) $K \times N$用户隐矩阵U：该矩阵中u_{ik}代表用户i对于因素k的喜爱程度。假设用户隐特征向量服从正态先验分布，如式4.4所示，其中$\sigma_U^2 I$为用户向量的协方差矩阵。本系统利用*java.util.Random*中的*nextGaussian*()方法，采取$Math.sqrt(b) * random.nextGaussian() + a$生成均值为$a$，方差为$b$的随机数，利用这些随机数初始化隐特征矩阵。

5) $K \times M$报告隐特征矩阵V：该矩阵中v_{kj}代表报告j，在因素k 上的得分。假设报告隐特征向量同样服从正态先验分布，如式4.5所示，其中$\sigma_V^2 I$为报告向量的协方差矩阵。其初始化方法与用户隐特征向量相同。

6) $N \times M$用户评分矩阵$R = U^T V$：该矩阵中r_{ij}代表用户i对于报告j的评分。

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} N(u_i|0, \sigma_U^2 I) \tag{4.4}$$

$$p(U|\sigma_V^2) = \prod_{i=1}^{N} N(v_i|0, \sigma_V^2 I) \tag{4.5}$$

其中，最为重要的就是构建用户的评分矩阵。在本系统中，用户对于一个报告的评分主要由提交报告，Fork报告，点赞报告与点踩报告四部分组成。将用户对于这四种报告的评分映射到(-1,1)的区间，其具体数值，如表 4.3 所示。

利用以上评分表，即可得到所有用户对于所有Bug的评分矩阵，其中，推荐目标用户x 其评价过的报告集合记为$R(x)$，则：

$$R(x) = sub(x) \cup good(x) \cup Fork(x) \cup none(x) \cup diss(x) \tag{4.6}$$

表 4.3: 用户报告评分表

报告类别	评分取值	目标用户x对于此类报告的集合表示
提交	1	$sub(x)$
点赞	0.7	$good(x)$
Fork	0.4	$fork(x)$
用户无评分	0	$none(x)$
点踩	-0.7	$diss(x)$

4.3.2　推荐结果的生成

假设评分矩阵R_{ij}服从高斯分布，计作$R_{ij} \sim N(U_i^T V_j, \sigma^2)$。利用已有的观测矩阵$R$估算出参数$U, V$，主要使用极大似然估计和最大后验概率来进行计算。其推理过程在此处不予介绍。最终的能量转化函数为：

$$E(U,V) = \frac{1}{2}(R - U^T V)^2 + \frac{\lambda_U}{2} U^T U^2 + \frac{\lambda_V}{2} V^T V^2 \tag{4.7}$$

其中，$\lambda_U = \sigma^2/\sigma_U^2$，$\lambda_V = \sigma^2/\sigma_V^2$。$E(U,V)$模型在训练过程中的损失函数。

```
int steps = 6000;
double alpha = 0.0002;
double beta = 0.02;
for(int step = 0 ; step < steps ; ++step) {
    for(int i = 0 ; i < N ; ++i) {
        for(int j = 0 ; j < M ; ++j) {
            if(R[i][j] > 0) {
                double eij = R[i][j];
                for(int k = 0 ; k < K ; ++k) {
                   eij -= U[i][k] * V[k][j];
                }
                for(int k = 0 ; k < K ; ++k) {
                    U[i][k] += alpha * (2 * eij * V[k][j] - beta * U[i][k]);
                    V[k][j] += alpha * (2 * eij * U[i][k] - beta * V[k][j]);
                }
            }
        }
    }
    double loss = 0;
    for(int i = 0 ; i < N ; ++i) {
        for(int j = 0 ; j < M ; ++j) {
            if(R[i][j] > 0) {
                double eij = 0;
                for(int k = 0; k < K; ++k) {
                    eij += U[i][k] * V[k][j];
                }
                loss += Math.pow(R[i][j] - eij,2);
                for(int k = 0; k < K; ++k) {
                    loss += (beta / 2) * (Math.pow(U[i][k], 2) + Math.pow(V[k][j], 2));
                }
            }
        }
    }
    if(loss < 0.001) { break; }
}
```

图 4.16: 矩阵分解实现代码

如图 4.16所示，为矩阵分解的实现代码。使用梯度下降来求解U, V 中的每一个元素，设置最大步长为6000。在误差loss 低于0.001时，循环结束，此时可以得到用户的新评分矩阵R'。对于推荐目标用户x，其对所有报告的评分即为向量$R'(x)$，其值的集合记为$R'(x)$。此时，$R(x)$中原本用户无评分的集合$none(x)$已经有了新的评分。将该集合中，分数大于0 的报告取出，则得到了用户可能感兴趣的报告。对于这部分报告，再根据报告获得的点赞与点踩数的和，从小到大进行排序，将排序结果的前十推荐给用户。这样可以使得大部分被提交的报告都得到用户之间的交叉审核。交叉审核结果可以辅助平台对报告做出评价。

4.3.3　模块实现效果

由于审核报告推荐与测试页面推荐的结果在同一个模态框中实现，故将两个模块的实现效果此进行统一展示。该页面会在用户点击查看推荐按钮，或者完成一次提交后弹出。

如图 4.17所示，左侧为测试任务的推荐，蓝色节点为暂未有Bug提交的页面，黄色节点为已有用户提交的页面，红色节点为用户自己提交的报告。对于希望用户进行测试的页面，会用绿色动态闪烁的形式提醒用户。右侧为审核任务的推荐列表，与相似报告推荐类似，用户可以点击每份报告进行详细查看，并且可以进行点赞和点踩的操作。

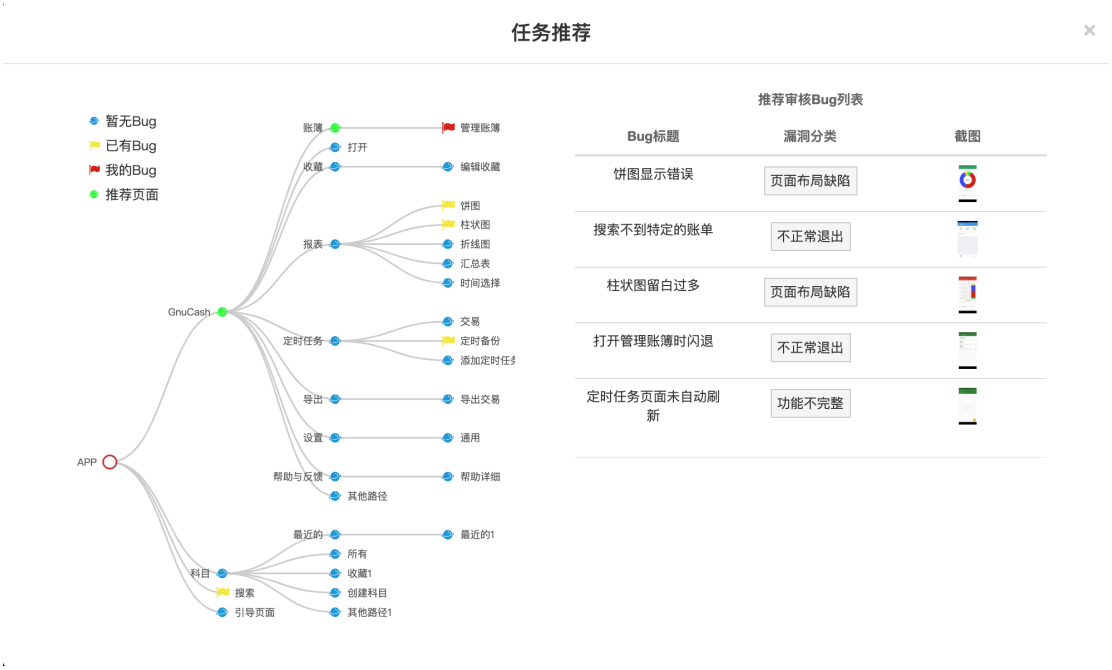

图 4.17: 测试页面推荐实现效果图

4.4　用户行为记录模块详细设计与实现

本模块主要用于记录推荐结果以及用户对于该结果所做出的响应，从而利用召回率对推荐系统做出评估。

4.4.1　实现类图

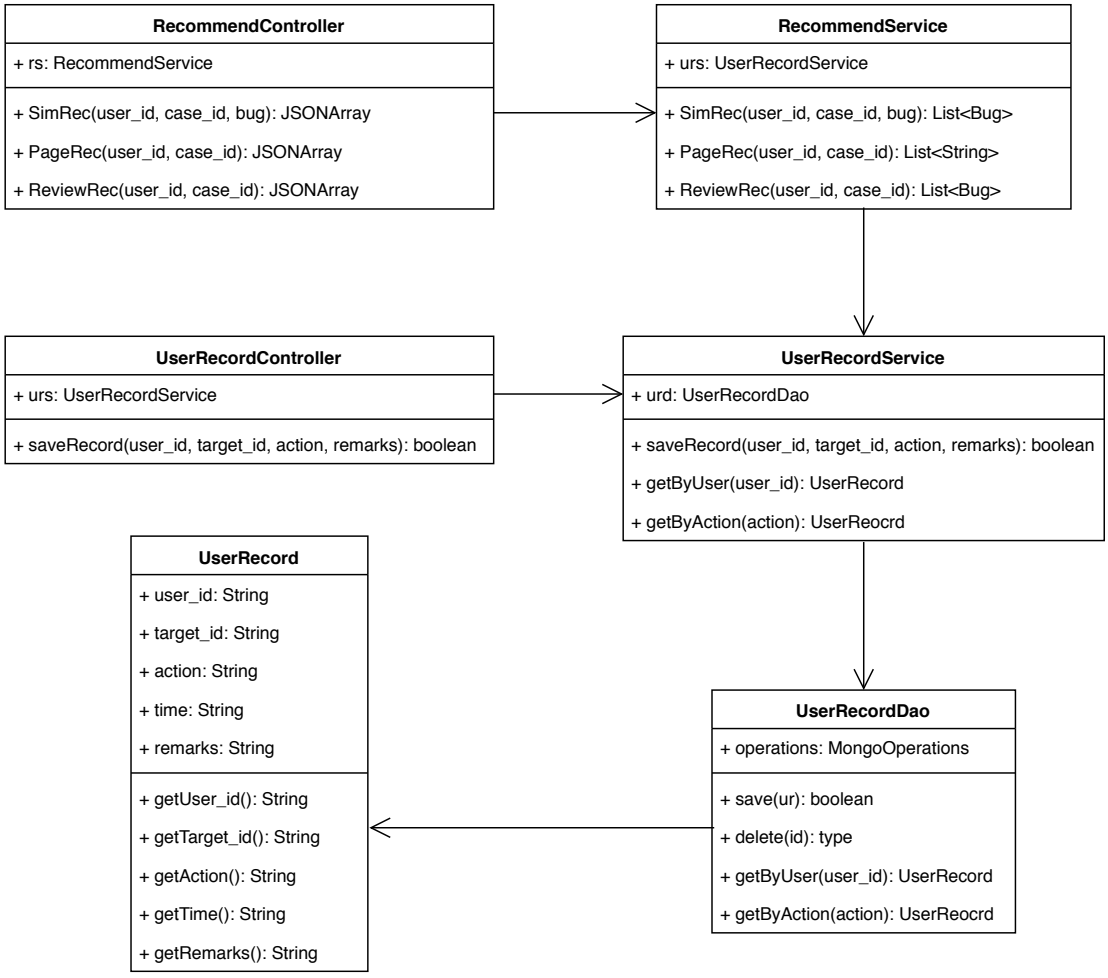

图 4.18: 用户行为记录实现类图

如图 4.18所示，该模块的实现主要与六个类有关。其中，UserReord类的字段定义已在3.3.5小节中进行过介绍。RecommendController在接收到前端通过HTTP发送的推荐请求时，调用RecommendService进行业务逻辑处理。该Service中PageRec()与ReviewRec()推荐方法函数会在计算得到推荐结果后，调用UserRecordService中的saveRecord()方法，记录推荐结果。

用户行为的记录主要通过UserRecordController完成。前端在用户点击如保存报告按钮后，会利用JavaScript非阻塞I/O的特性，并行向后端同时发送

保存报告和用户行为记录的请求。UserRecordController在接收到请求后，会调用UserRecordService进行处理，其负责获取当前系统时间，并将数据组装成UserRecord类，调用UserRecordDao，利用其中的MongoOperations将数据保存至MongoDB数组库，将数据持久化。

4.4.2　数据记录

本模块在数据库中所记录的推荐结果数据如表 4.4所示，主要记录了测试页面推荐的页面结果数组和审核报告推荐的推荐结果列表ID。action字段记录了推荐的结果类型。由于相似报告推荐的召回率分母为用户点击某份报告查看详情的记录，因此无需记录该推荐的结果数组。

表 4.4: 推荐结果数据

action	user_id	taget_id
PageRec	测试页面推荐用户	推荐页面结果数组
ReviewRec	审核报告推荐用户	审核报告推荐结果报告ID数组

本模块所记录的用户行为数据如表 4.5所示，对于相似报告推荐，主要记录了用户查看某份报告详情并对其进行了点赞、点踩或Fork的操作。对于测试页面推荐，主要记录了用户提交报告的操作，并记录此报告描述Bug所在页面，以便与推荐结果进行对比。对于审核报告推荐，主要记录了用户对于推荐报告进行点赞或点踩的操作。

表 4.5: 用户行为数据

action	user_id	taget_id	remarks
SimRec	相似报告推荐用户	查看详情的报告ID	
SimRecConf	相似报告推荐用户	查看详情的报告ID	点赞、点踩、Fork
PageRecConf	测试页面推荐用户	提交报告ID	Bug 所在页面
ReviewRecConf	审核报告推荐用户	查看详情的报告ID	

如上述两表所示，系统完成推荐结果与用户行为的记录后，则可利用Confirm记录作为分子，Recommend记录作为分母，分别进行推荐召回率的计算。以相似报告推荐为例，系统相似报告推荐的召回率计算如式4.8所示：

$$S_1 = \frac{\sum SimRecConf}{\sum SimRec} \tag{4.8}$$

4.5　本章小结

本章主要分模块，利用时序图、状态图、流程图等多种方式介绍了系统中报告提交模块、测试页面推荐模块、审核报告推荐模块和用户行为记录模块的具体实现。对于较为关键的业务逻辑和推荐算法，结合具体实际代码，根据实现逻辑进行了解释。最后，对于所有的实现，都结合项目实际运行的页面截图进行了描述。对于系统实现结果的测试和实验评估，将在下一章中进行介绍。

第五章　推荐系统的测试与实验评估

5.1　系统测试

根据第三章功能需求与非功能性需求分析，系统需要保证可用性和并发性。因此，本节将对系统的基础功能进行功能测试，对边界输入情况进行边界测试，最后会对系统的查询和提交接口进行压力测试。

5.1.1　测试环境准备

根据图3.7所描述的物理部署视图，如表 5.1所示，系统测试环境的准备包括运行chrome浏览器的用户计算机，运行Angular2程序的前端服务器，运行Spring Boot Docker以及Redis Docker的后端服务器以及运行MongoDB应用程序的数据库服务器。所有服务均独立部署在阿里云服务器中，根据所运行应用的不同，选择了不同版本的Linux服务器。除数据库连接外，应用间均采用RESTful接口进行通信。

表 5.1: 测试环境准备

设备名称	运行程序	程序版本信息
用户计算机 (Mac OS 10.14)	Chrome浏览器	Chrome 72.0(64bit)
ECS服务器 (Ubuntu 18.04)	Angular 2	Angular 2.4
ECS服务器 (4G内存，50M带宽)	Spring Boot Docker Redis Docker	Docker 17.09
		Spring Boot 2.0.2
		Redis 4.0.7
ECS服务器(Debian 8)	MongoDB	MongoDB 3.2.16

5.1.2　功能测试

功能测试是为了对系统的各项功能进行逐一验证，又被称为黑盒测试，或数据驱动测试。其只需要考虑系统的功能，不需要考虑软件的内部结构和实现。根据系统的特性，操作方法以及用户需求，通过设计功能测试用例，检查系统是否达到预期目标。

本小节的功能测试将与3.1小节的系统需求保持一致。从用户视角出发，举例模拟用户在系统中的正常使用流程，对报告填写模块、测试页面推荐模块和审核报告推荐模块进行功能测试。其中报告填写模块需要模拟用户正常上传报告与Fork 上传报告两种方式。

如表 5.2所示，在进行该测试之前，需要在”搜索”页面下提交部分与本次测试所提交内容不同的报告，用以在用户填写过程中验证相似报告推荐的有效性。该测试用例模拟了用户填写选择Bug所处页面、选择Bug属性、填写概要描述、填写详细描述以及上传Bug截图并进行标注的全过程。预期情况下，在用户填写Bug过程中，相似报告推荐列表不断刷新。但由于推荐内容与用户当前填写内容相似度较低，因此用户选择直接上传该报告。实际测试结果与预期结果相符。

表 5.2: 报告提交测试用例

用例编号	TC-RS-01
测试用户ID	10000
测试目标	用户填写并提交所发现的Bug，并且填写过程中，有相似报告推荐
前置条件	系统中已在”搜索”页面下有部分报告提交，且该部分提交报告与当前测试所提交内容不同
正常流程	1. 点击创建Bug按钮 2. 选择Bug所处页面(搜索) 3. 选择Bug属性(必现、严重、不正常退出) 4. 选择Bug题目(输入小数点，程序闪退) 5. 填写Bug详细描述(点击搜索，用键盘输入小数点时，程序崩溃，闪退。) 6. 上传Bug截图，并进行标注 7. 点击提交
预期结果	在用户选择所处页面、Bug属性，填写Bug描述时，右侧推荐列表不断刷新相似报告。系统提示用户提交成功
实际结果	与预期结果相符

如表 5.3所示，该测试在TC-RS-01测试用例完成后进行。测试用例模拟了系统中另一位用户填写过程中，在预期情况下，当其完成与TC-RS-01测试用例

相同的页面选择，属性选择以及题目输入后，通过相似报告推荐发现了系统中已提交的相似报告。其不再自行提交，而是选择Fork该推荐报告后进行提交的过程。实际测试结果与预期结果相符。

表 5.3: Fork报告测试用例

用例编号	TC-RS-02
测试用户ID	10001
测试目标	用户在填写过程中发现有相似报告，从而选择Fork修改后进行提交
前置条件	TC-RS-01测试用例完成
正常流程	1. 点击创建Bug 2. 选择Bug所处页面(搜索) 3. 选择Bug属性(必现、严重、不正常退出) 4. 选择Bug题目(输入小数点，程序闪退) 5. 发现推荐列表出现目标报告，点击查看详请 6. 点击Fork按钮后，修改Bug详细描述(点击搜索，键盘输入小数点，百分号或者波浪号时程序闪退) 7. 点击提交
预期结果	在用户选择所处页面，填写Bug描述时，右侧推荐列表不断刷新相似报告。用户点击相似度较高的报告，右侧页面显示该报告详情，点击Fork按钮后，左侧填写区自动填充目标报告内容。点击提交按钮，系统提示用户提交成功
实际结果	与预期结果相符

如表 5.4所示，该测试在TC-RS-02测试用例后进行。测试用例模拟了用户在提交了上述报告后，自行点击任务分配按钮。在预期情况下，系统应在模态框中显示当前测试进展的树形结构图，其中包含了当前已完成测试的页面，该用户已完成的页面，并提示用户下一步应该进行测试的页面。实际测试结果与预期结果相符。

如表 5.5所示，由于该推荐结果与测试页面推荐结果在同一页面中显示，故该测试同样需要在TC-RS-02 测试用例完成后进行。测试用例模拟了用户提交报告后，自行点击任务分配按钮，在预期情况下，系统应显示待审核的测试报告列表。用户在对推荐的第一个报告进行验证后，点击查看详情，并对其进行了

点赞操作。然后，关闭详情页面，返回至待审核任务列表。在预期情况下，列表应刷新，重新计算待审核报告结果，且该结果中不包含用户之前点赞的报告。实际测试结果与预期结果相符。

表 5.4: 测试页面推荐测试用例

用例编号	TC-PR-01
测试用户ID	10000
测试目标	可视化当前测试需求和进展，根据用户提交历史，推荐其进行下一步测试的页面
前置条件	TC-RS-02测试用例完成
正常流程	1. 点击任务分配按钮 2. 查看测试进展 3. 鼠标放置于测试页面图标上 4. 查看该页面所提交报告数量 2. 点击关闭按钮
预期结果	1. 模态框左侧出现测试任务页面路径可视化图 2. 在”推荐目的地”、”热门搜索”等页面上使用黄色图标进行标注，表示该页面已有用户提交报告 3. 在”搜索”页面上用红色图标进行标记，表示用户已在此页面提交报告 4. 在”我的”、”帮你选目的地”和”搜索联想”三个页面用绿色图标进行闪烁标注，表示推荐用户进行以上三个页面的测试 5. 当用户鼠标放置于某个测试页面图标时，Label显示当前页面提交报告数量
实际结果	与预期结果相符

本部分的功能测试主要模拟了两位不同用户，通过面向协作式众包测试的推荐系统进行报告提交的过程。第一位用户在系统中没有相应报告的情况下，进行了报告提交，并利用系统的测试页面推荐，进行下一步的测试。第二位用户在提交与第一位用户相同问题时，通过相似报告推荐进行Fork操作，补充描述该问题，并利用审核报告推荐对其他报告进行审核。

表 5.5: 审核报告推荐测试用例

用例编号	TC-RR-01
测试用户ID	10001
测试目标	用户填写并提交所发现的Bug，并且填写过程中，有相似报告推荐
前置条件	TC-RS-02测试用例完成
正常流程	1. 点击第一条推荐报告，查看详情 2. 在本机进行验证后，点击点赞按钮 3. 点击详情关闭按钮，返回推荐列表
预期结果	模态框右侧出现待审核任务列表，用户点击第一条推荐报告，显示该报告的详情。点击点赞按钮，显示点赞成功图标。点击关闭，刷新推荐列表，已进行点赞操作的报告不会再出现在列表中
实际结果	与预期结果相符

5.1.3　边界测试

本部分将分别对三个推荐模块进行边界测试，其目的是保证系统在极端输入或条件下，依旧可以正确做出响应。

报告填写模块的测试用例，如表 5.6所示。其边界条件主要是用户进行报告填写时可能不提供完整的描述信息。

表 5.6: 相似报告推荐边界测试用例

用例ID	用户操作	预期结果	实际结果
1	用户未选择Bug所在页面，直接进行填写描述	在所有报告中进行检索，返回推荐列表	与预期相符
2	用户描述与当前所提交的报告无匹配结果	推荐列表显示当前无相似报告	与预期相符
3	当前系统无报告提交	显示当前无相似报告	与预期相符
4	用户未选择Bug属性信息进行提交	提示用户报告内容缺失，不予提交	与预期相符

测试页面推荐模块的测试用例，如表 5.7所示。其边界条件主要是当前系统无提交或用户无提交的情况。

表 5.7: 测试页面推荐边界测试用例

用例ID	用户操作	预期结果	实际结果
1	当前无用户提交报告， 用户直接点击查看推荐	推荐所有一级页面	与预期相符
2	用户无提交报告， 但系统有报告提交	推荐系统中未有报告 提交的页面	与预期相符

审核报告推荐模块的测试用例，如表 5.8所示。其边界条件主要是当前用户无提交或系统无提交的情况。

表 5.8: 审核报告推荐模块测试用例

用例ID	用户操作	预期结果	实际结果
1	用户未提交任何报告， 没有历史的情况下点击推荐	返回最新提交的 多份报告	与预期相符
2	当前没有用户提交报告， 用户直接点击推荐	显示无推荐	与预期相符

5.1.4　压力测试

本部分将使用Jmeter对系统的查询接口和报告上传接口进行压力测试。整个Jmeter 的运行过程如下所示：

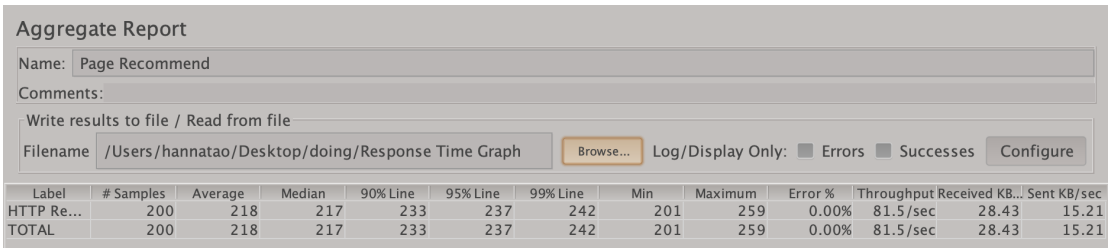

图 5.1: 测试页面推荐压力测试结果

1) 在开始运行Jmeter.bat之后，首先在Test Plan下新建Tread Group线程组。线程组中每个线程都可以看成一个虚拟用户，其线程数量在整个执行过程中不会发生改变。

2) 设置线程组中Nuber of Treads为80，Ramp-Up Period为2，LoopCount为2。该设置模拟了80个用户，在两秒内同时对服务器发送请求，并且会循环请求两

次。

3) 在线程组中添加三组HTTP GET请求，分别对三种推荐算法进行压力测试。另外，创建一组HTTP POST请求，对报告提交进行测试。

4) 在线程组中再添加一个用户监听结果的聚合报告。这里以测试页面推荐为例。具体测试结果如图 5.1所示。在四秒内，共进行了200次访问请求，平均响应时间为218ms，响应时间中位数为217ms，其中99%的用户请求都在242ms内得到了相应，没有错误请求的发生，系统通过该压力测试。如图5.2所示，以10ms为间隔记录了响应时间，接口响应所需时间较为平稳。

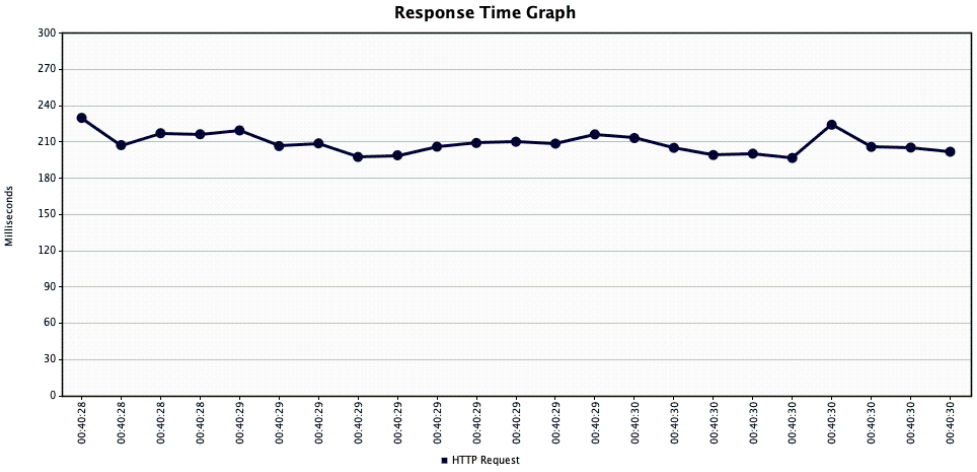

图 5.2: 接口响应所需时间图

5.2　实验评估

系统评估首先将介绍实验的设计与测试目标，然后将就实验结果，从多个角度来进行个体竞争与协作式众包测试推荐方式对比。同时，会就面向协作式众包测试推荐系统得到的结果进行独立分析。

5.2.1　A / B测试

本系统的A/B测试将40个用户分成两组，其中B组采用1.2.1小节所提到的个体竞争式众包测试报告提交系统上传测试报告，A组采用本系统进行报告提交。两组人员对同一个移动应用进行功能测试，实验时间为2个小时，对比用户所提交测试报告的重复率、报告的质量以及最终报告页面覆盖的情况。报告质量的评价由该移动应用的开发专家完成。每份报告根据提供截图，描述质量，得分分布在$[0, 10]$ 的区间内，得分大于0分的即被认定为有效Bug。重复提交的报告不得分。

5.2.2　实验结果

表 5.9: 对比实验结果

	A组	**B组**
总提交数	201	251
Bug覆盖种类	142	135
有效Bug数	182	196
有效比率	90.5%	78.2%
重复报告数	4	14
页面全覆盖所需时间	78分钟	未完成

如表 5.9所示，系统在2个小时的测试时间内，A组共收到201 份报告，覆盖了142种不同Bug。其中由专家判定为有效Bug报告的有182个，有效比率高达90.5%。有两组报告判定重复。其中一组原因为两位用户对于该Bug描述产生在不同页面上，但实际情况为该Bug仅存在某一固定页面，因而导致在进行页面筛选时，不会出现相似报告的推荐。另一组两位用户提交时间相近，因而在提交时没有能够发现重复。测试页面共有24个，本组报告在测试进行至第78分钟时，完成了页面全覆盖。

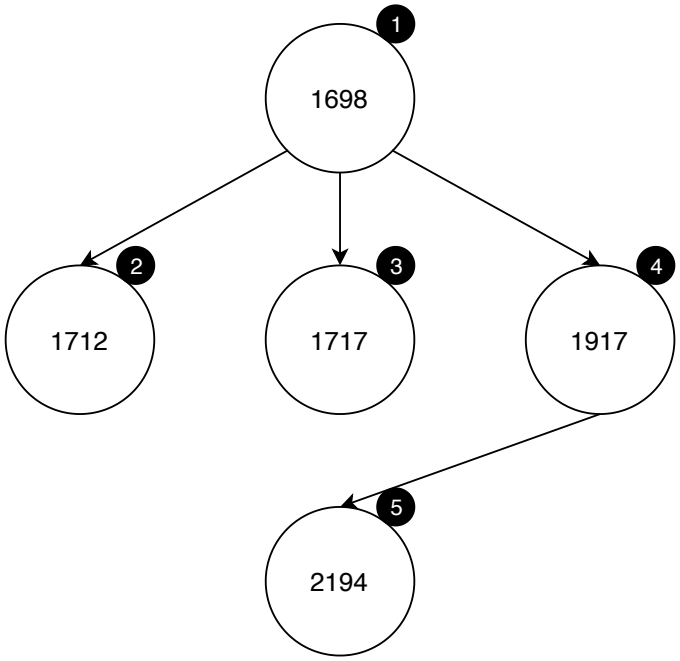

图 5.3: Bug树生成过程示例图

B组虽提交数目达到251份，比A组更多，但其仅覆盖了135种不同Bug。专家认定的有效Bug数为196，有效比率为78.2%。其中重复报告有14组，这14组中有2组为同一个人提交了两条完全一致的Bug。直至测试结束，该组仍未完成测试页面的全覆盖，覆盖页面达到19个。仍有5个页面由于长尾效应的存在未能完成覆盖。

系统协作效果显著，以如图 5.3所示的一棵Bug树的生成过程为例，展示了用户之间互相协作，对于一个Bug的描述层层深入的过程。其中，白色圆圈内的数字表示Bug 序号，黑色圆圈数字表示其提交顺序。直线起点表示原始报告，指向节点表示由此报告Fork之后修改形成的报告。

如表5.10所示，这5份报告由4位用户提交。4位用户通过本系统的相似报告推荐，利用Fork操作，对查看更多目的地时app出现的问题不断进行深入描述，从而提升了Bug的描述质量。

表 5.10: Bug树相关报告列表

报告ID	创建时间	Bug 描述	增益
1698	8:05	点击查看更多目的地时app停止运行	根节点，发现Bug
1712	8:07	1.点击查看更多 2.进行查看更多页面，app停止运行	描述步骤化
1717	8:09	当启动app第一次点击查看更多时，app会出现崩溃，重启错误不再发生	只有第一次进入时才会崩溃
1917	9:03	1.推荐目的地查看更多时，可能造成app停止运行或崩溃 2.推荐目的地查看更多时，可能造成app需要加载时间变长	第一次提出页面加载时间变长
2194	9:46	打开应用进入搜索页面，点击推荐目的地查看更多，一定出现以下情况中的其中一种： 1.app崩溃，停止运行 2. 页面卡顿，长时间空白后才进入正确的查看更多页面	总结上述情况，描述更加完整

报告被点赞数与专家最终评分结果如图 5.4所示，其中去除了无效报告。每个点表示一份或多份报告的重合结果，曲线表示被点赞数与专家评分的拟合结果。横坐标为被点赞数，纵坐标为专家评分。从结果来看，仍有少部分报告未

得到审核。随着点赞数越多，其得分的下限越高。报告的专家评分随着被点赞数的增多，呈现上升趋势。

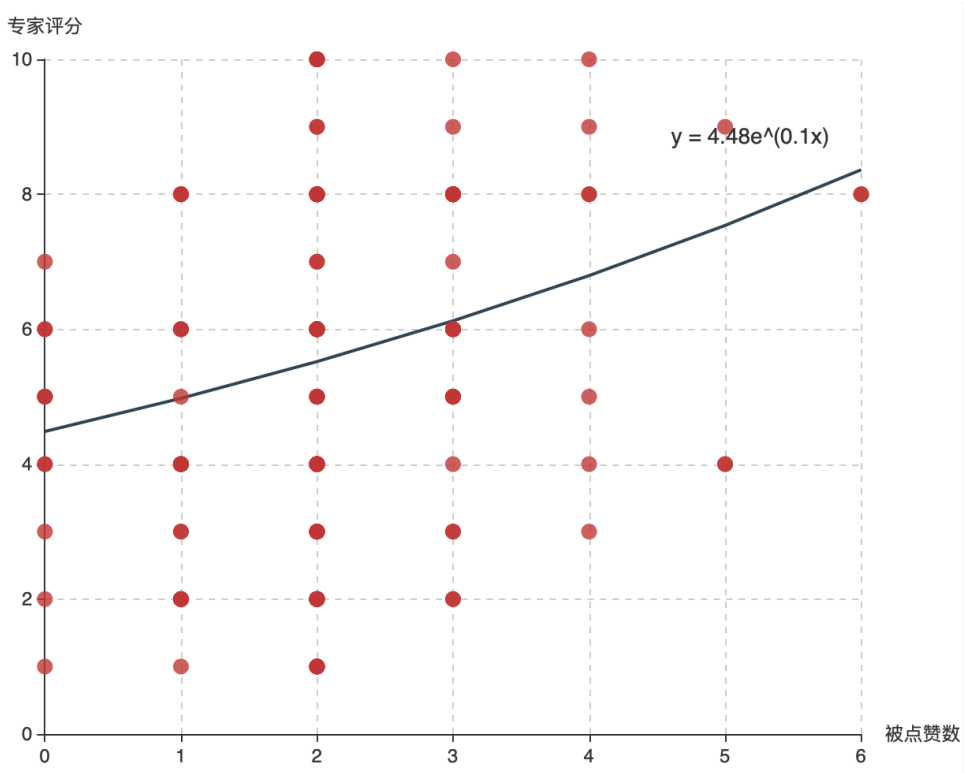

图 5.4: 报告被点赞数与专家评分关系图

综上所述，本系统通过信息共享与任务分配，在以下三个方面相较于传统独立竞争式众包测试系统取得了更好的结果：

1) 减少重复报告：利用实时相似报告推荐，在报告填写过程中，有效阻止了重复报告的提交。

2) 提升报告质量：利用多人协作，不断完善对于同一个Bug的描述，提升报告质量。

3) 加快页面覆盖：测试页面推荐中，首先，有效可视化了测试需求。同时，根据用户的历史提交记录，使得用户以最小的代价，进行尚未提交报告页面的测试。

同时，用户间相互进行审核的点赞、点踩结果也能够更好地辅助专家对报告进行评分。根据3.3.5小节设计的用户行为记录，其中，相似报告推荐的召回率为85.7%，测试页面推荐的召回率为78.4%，审核报告推荐的召回率为62.3%。推荐系统的召回率也基本达到预期。

5.3　本章小结

本章根据需求，对系统的基础功能进行了测试，同时，设计了边界输入用例对系统进行了边界测试，然后利用Jmeter对系统进行了并发压力测试。在真实的实验条件下，将本系统与个体竞争式的众包测试系统进行了A/B测试，验证了系统的有效性。

第六章　总结和展望

6.1　总结

当前众包测试的研究多集中于任务准备阶段与报告整合阶段，任务执行阶段由于缺少众包平台的支持，研究只能通过模型仿真来完成。同时，众测平台所采用的竞争式报告提交系统未能有效发挥众包优势。本文针对众包测试的任务执行阶段，基于众测平台开发出了面向协作式众包测试的推荐系统，在以下方面进行创新，以解决众测平台当前报告质量低下，平台验证困难等问题：

1）协作方法设计：使用多叉树记录报告修改历史，利用多人共同编辑完成对问题的补充描述。同时，平台可以通过报告被点赞和点踩的数量判定报告所描述问题的有效性。

2）相似报告推荐：从Bug所处页面，Bug属性，Bug问题描述等多维度出发，在用户填写报告时实时推荐与用户当前所提交内容相似的报告，引导用户对他人描述进行审核，避免重复报告的提交。

3）测试页面推荐：根据测试需求中的页面结构，可视化测试任务与测试进展，采用邻接矩阵记录页面间跳转耗费，利用多源最短路径计算待测页面，完成测试页面推荐。用户能够根据推荐页面快速决定下一阶段的测试任务。

4）审核报告推荐：利用用户协作数据和提交历史构建用户评分矩阵，使用概率矩阵分解完成基于模型的协同过滤，根据报告审核次数排序后进行审核报告推荐，利用用户间的交叉审核，帮助平台验证报告的普遍性。

本系统使用Angular2作为系统的前端架构，以完成页面内容的多样变化。Echarts 作为前端数据可视化方法，快速在图表中刷新动态数据。使用最为主流且轻量的微服务框架Spring Boot作为系统后端架构。Redis作为缓存减轻数据库压力，MongoDB 作为数据库，便于主从备份和提高查询速度。系统最终以RESTful接口形式对外提供服务。

本文对系统的各个模块进行了功能测试，根据并发需求，进行了系统压力测试，测试结果表明，系统能够在有效时间内完成用户请求响应。同时，系统记录了推荐结果与用户决策，利用召回率验证了推荐系统的有效性。采用A/B测试与个体竞争式的众包测试系统进行对比，结果表明该系统相较于个体竞争式的系统，有效报告比率增加12.3%，重复报告比率减少4%，并且完成了测试需求页面的全覆盖。

6.2　工作展望

本系统主要应用于众包测试任务中的任务执行阶段，该阶段需要用户在系统中填写测试报告，系统应做出实时反馈。因此，部分耗时较长的技术需要在优化其时间复杂度后再进行应用，这些技术的应用主要集中在以下三点：

1）图像处理技术：相似报告推荐可以结合用户所上传的图像信息融合文本信息后进行更为精确的相似度计算。

2）自动化脚本技术：用户所描述产生Bug的操作过程，系统自动生成可执行脚本并在对应机型上进行验证。

3）语义分析技术：审核报告推荐可以结合用户所提交报告的文本描述语义，使得概率矩阵分解的结果更为准确。

同时，随着系统所支持众测平台用户数的不断增长，参与众包测试的用户数也逐渐增多，因而对于系统的稳定性和并发性提出了新的要求。系统未来会采用分布式的方式进行部署以满足并发需求。

参考文献

[1] J. Howe, The rise of crowdsourcing, Wired magazine 14 (6) (2006) 1–4.

[2] H. Shen, Z. Li, J. Liu, J. E. Grant, Knowledge sharing in the online social network of yahoo! answers and its implications, IEEE Trans. Computers 64 (6) (2015) 1715–1728.

[3] M. Vukovic, Crowdsourcing for enterprises, in: 2009 IEEE Congress on Services, Part I, SERVICES I 2009, Los Angeles, CA, USA, July 6-10, 2009, 2009, pp. 686–692.

[4] K. Mao, L. Capra, M. Harman, Y. Jia, A survey of the use of crowdsourcing in software engineering, Journal of Systems and Software 126 (2017) 57–84.

[5] 章晓芳, 冯洋, 刘峘, 陈振宇, 徐宝文, 众包软件测试技术研究进展, 软件学报29 (1) (2018) 69–88.

[6] 冯剑红, 李国良, 冯建华, et al., 众包技术研究综述, 计算机学报38 (9) (2015) 1713–1726.

[7] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. M. Nezhad, E. Bertino, S. Dustdar, Quality control in crowdsourcing systems: Issues and directions, IEEE Internet Computing 17 (2) (2013) 76–81.

[8] X. Zhang, Z. Chen, C. Fang, Z. Liu, Guiding the crowds for android testing, in: Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume, 2016, pp. 752–753.

[9] P. Cheng, X. Lian, L. Chen, J. Han, J. Zhao, Task assignment on multi-skill oriented spatial crowdsourcing, IEEE Trans. Knowl. Data Eng. 28 (8) (2016) 2201–2215.

[10] P. Narula, P. Gutheim, D. Rolnitzky, A. Kulkarni, B. Hartmann, Mobileworks: A mobile crowdsourcing platform for workers at the bottom of the pyramid, in:

Human Computation, Papers from the 2011 AAAI Workshop, San Francisco, California, USA, August 8, 2011, 2011.

[11] 张志强, 逄居升, 谢晓芹, 周永, 众包质量控制策略及评估算法研究, 计算机学报36 (8) (2013) 1636–1649.

[12] W. Wu, W.-T. Tsai, Z. Hu, Y. Wu, Towards a game theoretical model for software crowdsourcing processes, in: Crowdsourcing, Springer, 2015, pp. 143–161.

[13] L. Tran, H. To, L. Fan, C. Shahabi, A real-time framework for task assignment in hyperlocal spatial crowdsourcing, ACM TIST 9 (3) (2018) 37:1–37:26.

[14] Z. Pan, H. Yu, C. Miao, C. Leung, Efficient collaborative crowdsourcing, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA., 2016, pp. 4248–4249.

[15] 陈英奇, 赵宇翔, 朱庆华, 科研众包视角下公众科学项目的任务匹配模型研究, 图书情报知识No.183 (3) (2018) 6–17.

[16] J. Stuckman, J. Purtilo, Analyzing the wikisphere: Methodology and data to support quantitative wiki research, JASIST 62 (8) (2011) 1564–1576.

[17] M. Yuen, I. King, K. Leung, Taskrec: Probabilistic matrix factorization in task recommendation in crowdsourcing systems, in: Neural Information Processing - 19th International Conference, ICONIP 2012, Doha, Qatar, November 12-15, 2012, Proceedings, Part II, 2012, pp. 516–525.

[18] M. Li, X. Sun, X. Jin, L. Tian, Worker recommendation with high acceptance rates in collaborative crowdsourcing systems, in: Collaborative Computing: Networking, Applications and Worksharing - 14th EAI International Conference, CollaborateCom 2018, Shanghai, China, December 1-3, 2018, Proceedings, 2018, pp. 53–74.

[19] D. Geiger, M. Schader, Personalized task recommendation in crowdsourcing information systems - current state of the art, Decision Support Systems 65 (2014) 3–16.

[20] E. Aldhahri, V. Shandilya, S. G. Shiva, Towards an effective crowdsourcing recommendation system: A survey of the state-of-the-art, in: 2015 IEEE Symposium

on Service-Oriented System Engineering, SOSE 2015, San Francisco Bay, CA, USA, March 30 - April 3, 2015, 2015, pp. 372–377.

[21] Y. Tung, S. Tseng, A novel approach to collaborative testing in a crowdsourcing environment, Journal of Systems and Software 86 (8) (2013) 2143–2153.

[22] X. Larrucea, I. Santamaria, R. C. Palacios, C. Ebert, Microservices, IEEE Software 35 (3) (2018) 96–100.

[23] F. Gutierrez, Spring Boot Messaging, Messaging APIs for Enterprise and Integration Solutions, Springer, 2017.

[24] A. Balalaie, A. Heydarnoori, P. Jamshidi, Microservices architecture enables devops: Migration to a cloud-native architecture, IEEE Software 33 (3) (2016) 42–52.

[25] F. Rysavy, T. Cerný, M. Tomásek, Aspect-oriented user interfaces design integration to angular 2 framework, in: 6th International Conference on IT Convergence and Security, ICITCS 2016, Prague, Czech Republic, September 26, 2016, 2016, pp. 1–3.

[26] D. Bernstein, Containers and cloud: From LXC to docker to kubernetes, IEEE Cloud Computing 1 (3) (2014) 81–84.

[27] A. Rahartomo, R. F. Aji, Y. Ruldeviyani, The application of big data using mongodb: Case study with scele fasilkom UI forum data, in: International Workshop on Big Data and Information Security, IWBIS 2016, Jakarta, Indonesia, October 18-19, 2016, 2016, pp. 51–56.

[28] Y. Wang, L. Zhang, J. Tan, M. Li, Y. Gao, X. Guerin, X. Meng, S. Meng, Hydradb: a resilient rdma-driven key-value middleware for in-memory cluster computing, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, Austin, TX, USA, November 15-20, 2015, 2015, pp. 22:1–22:11.

[29] W. Cao, S. Sahin, L. Liu, X. Bao, Evaluation and analysis of in-memory key-value systems, in: 2016 IEEE International Congress on Big Data, San Francisco, CA, USA, June 27 - July 2, 2016, 2016, pp. 26–33.

[30] 许海玲, 吴潇, 李晓东, 阎保平, 互联网推荐系统比较研究, 软件学报20 (2) (2009) 350–362.

[31] G. Adomavicius, Y. Kwon, Improving aggregate recommendation diversity using ranking-based techniques, IEEE Trans. Knowl. Data Eng. 24 (5) (2012) 896–911.

[32] 刘美玲, 郑德权, 赵铁军, 于洋, 动态多文档文摘模型, 软件学报23 (2) (2012) 289–298.

[33] L. Ma, Y. Zhang, Using word2vec to process big text data, in: 2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29 - November 1, 2015, 2015, pp. 2895–2897.

[34] D. Zhang, H. Xu, Z. Su, Y. Xu, Chinese comments sentiment classification based on word2vec and svm^{perf}, Expert Syst. Appl. 42 (4) (2015) 1857–1863.

[35] M. J. Kusner, Y. Sun, N. I. Kolkin, K. Q. Weinberger, From word embeddings to document distances, in: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, 2015, pp. 957–966.

[36] 吴丽花, 刘鲁, 个性化推荐系统用户建模技术综述, 情报学报25 (1) (2006) 55–62.

[37] J. Hershberger, M. Maxel, S. Suri, Finding the k shortest simple paths: A new algorithm and its implementation, ACM Trans. Algorithms 3 (4) (2007) 45.

[38] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.

[39] 李乐, 章毓晋, 非负矩阵分解算法综述, 电子学报36 (4) (2008) 737–743.

[40] J. Stuckman, J. Purtilo, Analyzing the wikisphere: Methodology and data to support quantitative wiki research, JASIST 62 (8) (2011) 1564–1576.

[41] H. Rocha, G. de Oliveira, H. Marques-Neto, M. T. Valente, Nextbug: a bugzilla extension for recommending similar bugs, J. Software Eng. R&D 3 (2015) 3.

简历与科研成果

基本情况 韩鹏飞，男，汉族，1995 年 2 月出生，江苏省东台市人。

教育背景

2017.9～2019.6 南京大学软件学院 硕士

2013.9～2017.6 东北大学软件学院 本科

读研期间的成果

1. **韩鹏飞**，宋少行，“一种面向众包测试平台的协作方法”，申请号：201910269760.5，已受理。

致　　谢

首先感谢我的导师刘嘉老师，在硕士期间给予我学业上和工作上的帮助。刘嘉老师幽默风趣，与他相处毫无距离感。同时，还需要对实验室的陈振宇老师、房春荣老师表示感谢，从毕业设计的选题到项目的实现，他们都给予了耐心指导。

感谢李玉莹学姐、杨乙霖学姐、宋少行同学参与我毕设的全过程，并提出了很多宝贵的意见。

感谢我的父母，负担我在硕士研究生期间的生活，给予我精神上的鼓励与支持。

最后，感谢母校南京大学，感谢软件学院，让我渡过了充实的研究生时光。