南京大學

研 究 生 毕 业 论 文
（申请工程硕士学位）

论 文 题 目 基于知识图谱的众测任务分配技术的设计与实现

作 者 姓 名 段梦洋

学科、专业名称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 陈振宇教授，冯洋助理研究员

2021 年 **5** 月 **20** 日

学　　号：**MF1932037**

论文答辩日期：**2021** 年 **5** 月 **20** 日

指 导 教 师：　　（签字）

Design and Implementation of the Crowdsourced Testing Task Assignment Techniques Based on Knowledge Graph

by

Mengyang Duan

Supervised by

Professor Zhenyu Chen, Research Associate Yang Feng

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
MASTER OF ENGINEERING
in
Software Engineering

Software Institute
Nanjing University

May 20, 2021

学 位 论 文 原 创 性 声 明

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

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

论文作者签名：_____

日期：　　年　　月　　日

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于知识图谱的众测任务分配技术的设计与实现

工程硕士（软件工程领域）专业 2019 级硕士生姓名：段梦洋

指导教师（姓名、职称）：陈振宇教授，冯洋助理研究员

摘　要

众包测试通过短时间招募大量测试工人对待测软件进行测试，解决了传统测试过程中测试人员组成单一、周期长、成本高的问题。但众包模式下测试人员的非专业和不确定也会导致测试报告质量参差不齐、测试需求覆盖率不达标、重复缺陷报告数量多、效率低的问题。

为此，本文设计了“基于知识图谱的众测任务分配技术”，力图通过个性化任务分配充分发挥众包工人在众测中的个体优势和群体智慧，改善上述问题，提高测试报告质量和测试完成效率。技术具体包括三个模块：1）知识图谱数据获取模块：引入“协作式众包测试”概念，构建完整的众测报告提交平台来获取知识图谱所需数据。众包工人在此平台领取三级页面测试任务填写缺陷报告时可以看到他人的相似缺陷报告并在其基础上进行增改，帮助提高报告质量，减少重复报告数目。众包工人也需要执行审核任务，即通过点赞点踩的方式审阅他人缺陷报告，推动最终可交付报告生成。2）知识图谱特征学习模块：系统根据平台中数据构建众包领域内知识图谱，结合众包工人历史执行任务记录作为输入，利用机器学习模型得到众包工人对特定任务的偏好，综合当前测试任务的三级页面覆盖情况，审核任务的缺陷报告点赞点踩数量情况，个性化分配测试任务和审核任务给众包工人。3）知识图谱任务分配模块：将任务分配结果和当前任务执行情况可视化展示给众包工人，帮助提高测试需求的覆盖率和整体测试效率。

在实现技术上，本系统前端使用 Angular2 框架，后端平台部分使用 Spring Boot 框架经典的 MVC 架构。热点数据用 Redis 缓存在内存中提高响应速度，使用 NoSql 数据库的代表 MongoDB 进行数据持久化，有利于提高系统数据的处理速度和灵活性。特征学习模块使用 Python 脚本构建，利于和存储知识图谱实体关系数据的 Neo4j 图数据库便捷交互。此外，系统使用 Nginx 负载均衡，

使用 Docker 容器化技术对所有模块进行了独立部署，保证系统各部分的健壮性和可移植性。

系统进行了功能测试和性能测试并顺利通过，证明系统满足使用需求。同时通过设计实验的方式比较使用本文技术系统与采用协同过滤技术分配任务的系统在一次众包测试中提交缺陷报告的数量和质量，证明了本文技术有效性。

关键词： 知识图谱，特征学习，众包测试，任务分配

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Design and Implementation of the Crowdsourced Testing Task Assignment Techniques Based on Knowledge Graph

SPECIALIZATION: Software Engineering

POSTGRADUATE: Mengyang Duan

MENTOR: Professor Zhenyu Chen, Research Associate Yang Feng

Abstract

Crowdsourcing testing can recruit a large number of testing workers to test the software in a short time, which solves the problems of single tester composition, long cycle and high cost in the traditional testing process. However, the non professional and uncertain testers in crowdsourcing mode will also lead to the problems of uneven test report quality, substandard test requirement coverage, large number of repeated bug reports and low efficiency.

Therefore, this paper designs a "crowdsourced testing task assignment technique based on knowledge graph", trying to make full use of the individual advantages and group wisdom of crowdsourced workers in crowd testing through personalized task assignment, improve the above problems, and improve the quality of test reports and test completion efficiency. The technique includes three modules: 1) knowledge graph data acquisition module: the concept of "collaborative crowdsourcing test" is introduced, and a complete crowdsourcing report submission platform is built to obtain the required data of knowledge graph. On this platform, crowdsourcing workers can receive three-level page test tasks, and when filling in bug reports, they can see similar bug reports and make changes based on them to help improve the quality of reports and reduce the number of duplicate reports. Crowdsourcing workers also need to perform audit tasks, that is, to review other people's bug reports by means of likes and dislikes, and to promote the generation of final deliverable reports. 2) Knowledge graph feature learning module: the system builds an internal knowledge graph of the crowdsourcing field based on the data in the platform, and combines the historical task records of crowdsourcing workers as input, uses the machine learning model to get the crowd-

sourcing workers' preference for specific tasks, and integrates the three-level page coverage and bug like status, assign test tasks and audit tasks to crowdsourcing workers. 3) Knowledge graph task assignment module: visualizing the task assign results and current task execution to crowdsourcing workers in order to improve the coverage of test requirements and overall test efficiency.

In terms of implementation techniques, the front end of the system uses Angular2 framework, and the back-end platform uses the classic MVC architecture of SpringBoot framework. Hot data is cached in memory with Redis to improve the response speed. MongoDB database, the representative of NoSQL database, is used for data persistence, which is conducive to improving the processing speed and flexibility of system data. The feature learning module is built by Python script, which can easily interact with Neo4j graph database which stores entity relationship data of knowledge graph. Also, the system uses Nginx for load balancing and Docker containerization technique for independent deployment of all modules to ensure the robustness and portability of each part of the system.

The function test and performance test of the system are carried out at the same time, which proves that the system meets the use requirements. Meanwhile, we compare the quantity and quality of bug reports in a crowdsourcing test between the task assignment system based on knowledge graph and the system based on collaborative filtering, which proves the effectiveness of our task assignment technique.

keywords: Knowledge Graph, Self-Taught Learning, Crowd-sourcing Test, Task Assignment

目　　录

插图清单

附表清单

第一章　引言

“基于知识图谱的众测任务分配技术”是在众包测试越来越受到重视并得到广泛应用的背景下提出的，目的是帮助提高整体众包测试的效率。本技术参考了国内外的研究现状，引入领域内知识图谱与联合学习帮助实现个性化高效率的任务分配。在本章对其背景及基本情况进行概述如下。

1.1　项目背景及意义

在当今社会，互联网产业飞速发展，软件产品层出不穷。面对不断增加的需求和快速的迭代，通过软件测试工作保证软件产品质量的重要性也愈发突出。传统的软件测试大多雇佣固定的人员编写测试脚本对软件进行测试，测试周期长成本高，且无法获得真实使用者对产品的反馈。

众包 (Crowdsourcing) 是 Howe Jeff 于 2006 年在美国《连线》杂志上首次提出的一种新型商业模式 [1]。它是由互联网发展催生出来的一种分布式问题解决和生产组织模式，具体指的是公司将之前雇佣员工完成的事情，拆分成小的独立任务招募普通群众去完成 [2]。众包测试作为众包软件工程 [3] 下的一个新分支，可以在短时间内招募大量测试工人对待测软件进行测试，解决了传统测试过程中反馈少的缺点并有效的降低成本。但同时，众包模式也带来了一些新的问题。由于测试过程中可能需要专业的测试方法和知识，非专业的众包测试人员执行测试任务可能出现测试报告质量低的问题。此外，如何降低重复报告数量，实现测试需求的全覆盖，提高整体测试效率也是在众包测试过程中要重点考虑的问题 [4] [5] [6]。

针对上述问题，我们通过引入协作式众包的模式，以及优化众包过程中任务分配的方式来解决。传统的众包采用竞争模式，每个众包工人相互独立的接受并完成任务，容易产生重复且质量参差不齐的测试报告，增加审查难度。协作式众包模式下众包工人可以看到他人的测试报告内容并直接在其基础上进行增改，这样多人协作一份报告的方式有效减少了重复报告的数量，提高了报告的质量。此外，我们要求众包工人对他人缺陷报告进行审查，通过点赞点踩的

方式评价该报告，帮助管理人员筛选出有效的测试报告生成最终交付报告。

在任务分配方式中，传统的众包测试平台多采用人工分配任务或众包工人自行探索的方式。这两种方式虽然简单，但均存在一些问题。人工分配任务的方式没有考虑众包工人实际参与并完成任务的情况，可能会因为部分众包工人的缺席等造成部分众包任务不能正常完成。此外，不同的众包工人和众包任务有其自己的特点，这种直接指定的方式忽略了这种特性而不能根据工人的优势和任务的特性将特定任务分配给最适宜的众包工人。而众包工人自行探索的方式虽然满足了众包工人的偏好，但无法保证测试需求的全覆盖。

少数众包测试平台在分配任务时使用了协同过滤方法，例如 Han 等人 [?] 在某众包测试平台中使用众包工人历史参与任务的记录形成交互矩阵，通过矩阵分解的方式分析任务的特征分配相似特征的新任务给众包工人。这种方式考虑了众包任务的特性，但没有利用众包测试的领域内知识，当众包工人历史参与任务比例较低时由于交互矩阵的稀疏也会造成任务分配的不准确。

为此，我们需要引入辅助信息帮助任务分配。本文设计了一种新型众包测试任务分配技术，它使用知识图谱引入众包工人和众包任务的领域内特性及其语义关系并具象化保存下来，与众包工人历史参与任务记录一起输入机器学习模型中。它以众包工人历史参与任务为基准，沿知识图谱的关系边进行传播，自动学习众包工人对不同关系的偏好。当众包工人需要分配众包任务时，就可利用训练好的模型进行偏好传播，根据不同众包任务在知识图谱中的位置匹配出最适合的任务，提高众包测试过程的效率。

1.2 国内外研究现状

根据本文研究的相关问题，下面将分别从“众包测试平台”、“众包任务分配”、“推荐系统”、“知识图谱用于推荐系统”四个角度来介绍国内外研究的现状，为我们提供研究思路和方法的借鉴。

1.2.1 众包测试平台领域研究现状

一个完整的众测任务主要有任务发布者、众测平台、众包工人三个参与方 [7]。任务发布者在众测平台上根据要求提交测试需求，平台对需求进行细化处理，转换成测试任务发布，众包工人在平台上领取并执行任务，提交具体的测试报告，再由平台对报告进行整理，最终交付给任务发布者。在整个流程

中，众测平台 [8] 对接着任务发布者和众包工人，其质量直接影响整个众包测试的质量。

目前国内外都有很多比较成熟的众包测试平台，如百度众测平台、360 众测平台、Testin 云端众测、CNVD 众测平台等。这些测试平台大多规模较大，通过吸引足够数量的众包工人对任务发布者发布的 APP 或 Web 软件根据文档要求进行独立的探索性测试并提交发现的缺陷报告。这种情况下提交的缺陷报告数量虽多，但重复报告较多，没有发挥群体智能的优势，也给处理这些数量众多的原始缺陷报告生成最终的交付报告的产生造成了很大的困难 [2]。

为此，协作式众包模式也被国内外很多研究人员关注。例如，Alexandre 等人 [9] 提出众包工人在基于竞争的众包软件开发中会面临许多能力、协作、时间管理方面的困难，可能导致他们退出众包任务。Alsayyari 等人 [10] 认为使用众包方式进行软件测试是非常独特且极具价值的。但是目前的众包平台缺少对众测中协作的支持。Mridha 等人 [11] 认为在竞争激烈的众包市场引入协作，能够更加高效地执行任务。Pan 等人 [12] 权衡了协作式众包中质量-成本-时间的矛盾，提出一种新的任务分配方式，通过集合众包工人的专业知识进行分配，减少众包时间，提高众包质量。

然而，这些研究大多仍处在理论研究阶段，并没有在众包平台中得到实践应用。如何在实践中实现从“合作”到“协作”的跨越，达到 1+1>2 的效果,其过程依然任重道远 [13]。

1.2.2 众包任务分配领域研究现状

在大多数众包平台中，众包任务的分配仍采用人工固定分配或由众包工人自行选择的方式。其中人工固定分配的方法人工成本高，容易忽略各个任务以及众包工人之间的特性 [14]。一项以 Mturk [15] 众包市场为模版的调查显示，众包工人在搜索任务时主要关注任务的第一页和第二页。此外，当部分工人未能按时在线执行任务时也不能及时动态调整方案。根据众包工人意愿自行选择任务的方式则容易出现难度较大的众包任务执行人数较少的情况，造成测试任务覆盖率不足的问题。

众包任务分配作为任务发布方和众包工人高效交互的主要因素 [16]，众多研究人员也对其展开了深入的研究。这些研究方法大多采用不同的方式对众包任务和众包工人进行建模完成匹配。例如，Geiger 等人 [17] 提出在众包系统中引入任务推荐机制帮助任务分配，将众包系统与主流个性化推荐系统结合，再

通过大量的实验验证各种算法的优劣。Rahman 等人 [18] 提出了根据众包工人的个人因素（技能和工资）以及基于群体的人为因素（工人间协作的亲和力）来将众包工人进行建模并根据优化目标进行分配。Ho 等人 [19] 提出了一种探索性算法，提前了解工人的技能，根据工人到达的时间以及技能与任务属性的匹程度为工人个性化分配众包任务。Moayedikia 等人 [20] 根据工人之间行为的相似度对众包工人自动聚类并生成相应的自动机。根据聚类结果使用对应自动机为工人进行自动任务分配。Qiao 等人 [21] 设计了一种新的协作式众包任务分配方法，由完成过任务的工人向其他工人提供反馈，根据反馈中的信息构建模型评估工人亲和度和能力值，根据模型的结得到特定任务和工人的匹配度。

1.2.3 推荐系统领域知识图谱研究现状

“众包任务分配”主要是将合适的众包任务分配给在线的众包工人，其核心可以看作是一个推荐系统。目前国内外关于个性化推荐的研究很多，主要可以被分成三类：基于内容的推荐方法、基于协同过滤的推荐方法和混合推荐方法 [22]。Pazzani 等人 [23] 提出的“基于内容的推荐方法”通过对被推荐物品进行特征提取和建模，找到和用户历史感兴趣物品相似的物品进行推荐。基于协同过滤的推荐方法基于用户群体的历史感兴趣物品建立用户行为矩阵，对相似群体的用户推荐相似的物品。这样可以省略基于内容的推荐方法中的物品建模过程实现高效推荐。然而，这种推荐方法也存在着用户行为矩阵稀疏或新加入用户、物品时推荐准确度较低的问题。为此，我们在协同过滤的基础上引入用户或物品的辅助信息来丰富内容，解决上述问题。

在各种辅助信息中，知识图谱虽然提出较晚，但越来越受到重视。它通过“实体-边-实体”的三元组形式有效的将用户和物品的属性信息与行为信息组织起来作为推荐系统的信息来源，极大程度上提高了推荐系统的准确性和多样性，并提供了推荐系统的可解释路径。主要有以下三种分类：基于嵌入的方法、基于路径的方法，以及联合方法。基于嵌入的方法将实体和边映射到低维空间向量中，得到其嵌入表示，利用知识图谱丰富其维度信息。其中又包括基于翻译距离的嵌入，如 Antoine 等人 [24] 提出的 TransE 模型等，以及以 Bordes 等人 [25] 提出的 SME 为例的基于语义匹配的模型。这种方法中实体嵌入学习和推荐模型的学习是依次进行的，容易忽略图中的信息联通方式，一般无法提供推荐结果的解释。基于路径的方法将知识图谱视为异构信息网络，通过用户和推荐对象的联通相似性提升推荐效果。它具有天生的可解释性，但由于对用户

和推荐对象的表示比较简单，准确性有待提高。联合方法结合了前两种方法，利用嵌入传播考虑用户和推荐对象在知识图谱中有多跳邻居的情况，兼具可解释性和准确性，是目前主流的知识图谱推荐方法，包括 RippleNet [26] 等模型。

应用建立好的领域内知识图谱和上述推荐模型，各种个性化推荐服务应运而生。Wang 等人 [27] 建立医学领域知识图谱，将疾病、药物、病人的关系映射到低维空间为病人推荐安全的药物。Lu 等人 [28] 建立了大型的旅游知识图谱，为游客个性化推荐适合的景点。Haussmann 等人 [29] 构建了一个统一的食品图，记录食品的营养成分和产地信息，为健康饮食的消费者提供饮食建议。由此可见，在众包测试领域建立领域内知识图谱并据此进行任务分配也是非常具有前景且可行的。

1.3 本文主要工作

针对上述研究现状，众包测试任务分配面临的困难以及众包测试平台的系统缺陷，本文以高效分配众包测试任务为目的，设计了基于知识图谱的众包测试任务分配系统。参考国内某众包测试平台的移动应用测试测试和 Web 功能测试 [30]，本文所设计众测任务分配系统主要用于协作式众包平台，主要解决当前众包系统存在的人工干预任务分配、任务完成度低、结果质量不可靠等问题。本系统主要由以下三个模块组成:

1）知识图谱数据获取模块：为了获取构建众包测试知识图谱的数据，基于协作式众包的思想构建了一个完整的提交报告平台。众包工人可以在此平台提交测试用例和缺陷报告，且在编辑缺陷报告时系统会实时根据众包工人输入的缺陷属性和描述推荐相似报告，众包工人可以查看相似报告并进行点赞点踩操作，或在其基础上进行增益补充，以此减少重复报告，提高总体数据质量。

2）知识图谱特征学习模块：根据领域内知识构建知识图谱，基于知识图谱和众包工人历史兴趣构建 RippleNet 模型，输出众包工人对新任务的潜在偏好，结合任务覆盖率等要求给出任务分配结果。

3）知识图谱任务分配模块：在众包工人提交缺陷报告后根据特征学习结果分配三级页面测试任务和缺陷报告审核任务，同时以树状结构显示当前三级页面任务的缺陷报告提交情况，帮助众包工人掌握整体测试动态。

为了保证系统达到预期功能，满足可用性要求，我们对系统进行了功能和性能测试。同时，通过对照实验的方法对本系统基于知识图谱的任务分配技术

与其他众包测试任务分配算法进行了对比，验证了基于知识图谱的众包测试任务分配技术可以达到提高测试报告质量及测试需求覆盖率，减少重复报告数量，提高总体测试效率方面的效果。

1.4 本文组织结构

本文共分为六个章节，组织结构如下。

第一章为引言部分。主要介绍了众包测试、众测平台以及众包测试任务分配的基本情况与国内外研究现状，以及当前众包测试过程中任务分配方面的问题，引出了本文的技术及主要工作。

第二章为相关技术概念介绍。对测试任务分配技术以及测试报告提交平台构建过程中使用到的 Angular2、Spring Boot 框架、Mongo 数据库、Neo4j 图数据库、知识图谱、RippleNet 模型等理论和技术进行了介绍。

第三章为本系统的需求分析与概要设计。对系统进行了模块划分并使用 UML 图等对各个模块功能进行阐述。使用各种视图对系统整体架构进行了设计，使用类图和架构图对各模块进行了详细设计。

第四章在第三章需求分析的基础上，介绍了系统的具体实现。主要包括了本系统各个模块的进程图、关键代码以及实现截图。

第五章为测试与实验评估部分，通过功能测试和压力测试对系统各项功能进行测试验收，同时设计实验与其他众包测试任务分配技术进行比较，证明了本文技术的有效性。

第六章为总结与展望。本章节简单总结了本文与本系统的各项工作与贡献，同时针对系统的不足之处进行了分析，提出了今后的改进方向。

第二章　相关技术概述

任务分配的过程其实就是将合适的众包任务分配给参与项目的众包工人的过程。为了减少分配任务的人工成本，提高灵活性，最大限度发挥工人的优势，我们利用历史众包工人参与众包任务的信息构建相关知识图谱，采用RippleNet模型通过联合学习的方式将知识图谱合并到推荐系统中，取得了良好的效果。本章节对使用到的各项技术进行具体介绍。

2.1　任务分配相关技术

2.1.1　知识图谱

传统推荐系统因为只关注用户和物品的交互行为，容易遇到交互信息相对大量的物品记录过于稀疏的问题，造成过拟合问题。或是在新的用户和物品加入时因为没有交互行为无法进行推荐造成冷启动问题。为了解决上述问题，我们需要在用户物品交互信息的基础上引入辅助信息增强我们对用户和物品的认识，弥补交互记录的问题。

在多种多样的辅助信息中，知识图谱逐渐引起越来越多的研究人员的关注。这个概念最早由Google公司提出，来改进其搜索引擎针对语义化问题的搜索结果。它本质上是一种结构化的语义网络，由节点和边组成。节点代表推荐系统中的实体，边代表这些实体间的关系，通过“实体-边-实体”的三元组形式有效的将用户和物品的属性信息以及用户与物品的行为信息组织起来作为推荐系统的信息来源，极大程度上提高了推荐系统的准确性和多样性，并提供了推荐系统的可解释路径。

如图 2-1为电影领域知识图谱的一部分。其中的圆点均代表实体，直线代表实体间的关系。假若一个用户喜欢“霸王别姬”这部电影，那么根据知识图谱，“阿飞正传”与其属于同一个主演，“末代皇帝”与其属于同一种题材，“搜索”与其属于同一个导演。我们可以根据知识图谱中这些实体间的联系向用户推荐相关电影。

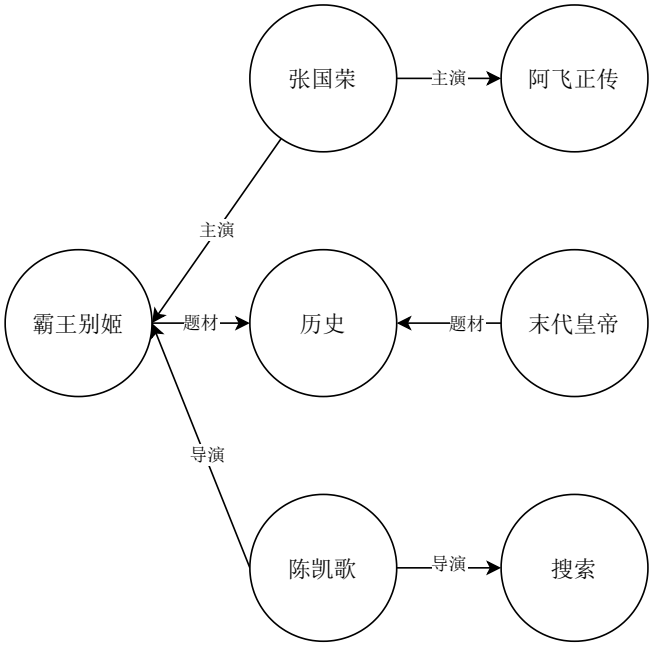


图 2-1: 知识图谱示例

知识图谱的构建需要多种技术的支持。例如我们需要通过知识抽取的技术从开放的各种数据中自动化提取可用的知识，包括实体、属性，以及关系。通过知识融合技术，消除实体及关系定义中出现的一些歧义问题。通过知识推理技术，可以在已有的知识图谱的基础上进一步挖掘隐藏的知识从而丰富图谱数据。目前，国内的知识图谱技术正在飞速发展的进程中，各项具有丰富语义，开放互联的知识图谱在个性化推荐、知识问答等领域中发挥出了巨大的价值，如创作共享的 Freebase 知识库 [31]、语义网 DBpedia [32]、多语言知识库 YAGO [33] 等。除了这些全领域知识图谱，在许多专业化的领域里知识图谱也得到广泛应用，如中药领域知识图谱 HerbNet，乳腺癌领域知识图谱以及数学领域的知识库 WolframAlpha。

2.1.2 RippleNet 模型

RippeleNet 是由 Wang [26] 等人在 2018 年提出的将知识图谱用于推荐系统的模型。不同于之前基于嵌入和基于路径的知识图谱推荐方法，RippleNet 采用端到端的方式，将用户-项目对作为输入并直接输出用户选择特定项目的概率。它主要用到偏好传播的思想，将每个用户的历史选择项目作为种子集，沿着知

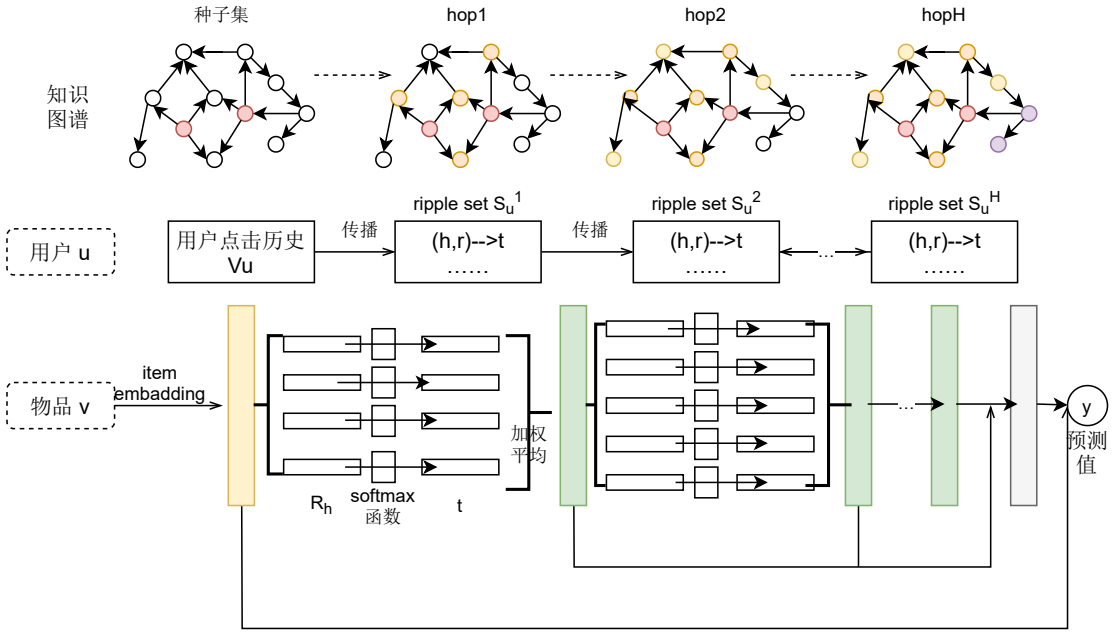


图 2-2: RippleNet 模型示意图

识图谱的关系边向外，以类似水波的形式传递，以此来发现用户的潜在兴趣。

在 RippleNet 模型的输入中除了知识图谱，还包括用户行为矩阵。对于用户集合 $\mathcal{U} = \{u_1, u_2, \ldots\}$，和物品集合 $\mathcal{V} = \{v_1, v_2, \ldots\}$，给定矩阵 $\mathbf{Y} = \{y_{uv} \mid u \in \mathcal{U}, v \in \mathcal{V}\}$。当用户 u 和物品 v 之间存在交互信息时，$y_{uv} = 1$, 否则为 0。对于没有交互信息的用户和物品，其项为空，由此可见该矩阵为稀疏矩阵，我们的目标就是预测那些用户与物品没有交互部分的值。

如图 2-2所示为 RippleNet 的架构。我们定义 Ripple 集合来表示用户的潜在偏好。假设给定用户行为矩阵 Y 和用户 u，那么该用户的历史行为记录为该矩阵中所有 $y_{uv} = 1$ 的物品，计作 $\mathcal{E}_u^0 = \mathcal{V}_u = \{v \mid y_{uv} = 1\}$。将这部分数据集作为种子，作为起点向相邻节点传播，这样每沿着知识图谱关系扩展一次称为一个 hop。在 RippleNet 中，对于给定交互矩阵 Y 和知识图谱 G，用户 U 的 k-hop 相关实体集合表示如公式所示。

$$\epsilon_u^k = \left\{ t | (h, r, t) \in G \text{ and } h \in \epsilon_u^{k-1} \right\}, k = 1, 2, ..., H \tag{2-1}$$

其中，$\epsilon_u^0 = V_u$，即用户历史交互过的物品集合。

对于由 (h,r,t) 三元组组成的知识图谱，用户 u 的 k-hop 波纹集被定义为从

ϵ_u^{k-1} 开始的知识三元组

$$S_u^k = \left\{(h,r,t)|(h,r,t) \in G \ and \ h \in \epsilon_u^{k-1}\right\}, k = 1,2,...,H \tag{2-2}$$

对于物品表示，模型中采用张量分解的方法表示实体的语义关系。将每个物品 v 用嵌入 $v \in R^d$ 表示，d 是嵌入的维度。给定 v 和用户 u 的 1-hop 波纹集合 S_u^1，通过将物品 v 和三元组的头实体 h_i 和关系 r_i 比较，为 S_u^1 中的每个三元组分配如下关联概率：

$$p_i = softmax(v^T R_i h_i) = \frac{exp(v^T R_i h_i)}{\sum_{(h,r,t)\in S_u^1} exp(v^T Rh)} \tag{2-3}$$

在得到关联概率后，我们还需要考虑嵌入矩阵 R_i，因为物品实体对之间通过不同的关系衡量其距离，可能结果不同。因此，我们将 S_u^1 加权中的尾部之和乘以相应的相关概率，并返回向量 o_u^1

$$o_u^1 = \sum_{(h_i,r_i,t_i)\in S_u^1} p_i t_i \tag{2-4}$$

其中 o_u^1 可以被看作为户 u 的点击历史 v_u 中关于物品 v 的 1-hop 反馈。通过累加不同 hop 的反馈和得到：

$$u = o_u^1 + o_u^2 + ... + o_u^H \tag{2-5}$$

最终组合用户嵌入和物品嵌入，利用 sigmod 激活函数得到预测用户点击物品的概率为：

$$\hat{y}_{uv} = \sigma(u^T v) \tag{2-6}$$

2.2 系统架构相关技术

为了保证代码的逻辑清晰、可读性强，以及系统的高可用以及可扩展，我们使用了成熟的系统框架对系统架构进行了设计，将基础工作交给框架本身，实现“高内聚，低耦合”的效果。在前端，我们主要使用了 Angular2 框架，后端使用了 Spring Boot 框架，在部署时使用了 Docker 容器作为运行环境。

2.2.1 Angular2

近年来，框架作为规范和简化软件开发过程，实现基础功能的工具得到广泛关注，各种框架应运而生。Google 于 2010 年 10 月发布了自己的首个 Web 应用框架 AngularJS [34], 也被称作 Angular。它创新性地提出了“双向数据绑定”的理念，得到软件行业从业者的强烈关注和拥护。但是，其本身也存在着适用型不足，开发难度较高等问题。

因此，Google 团队于 2016 年 9 月发布了 Angular2，它虽是 Angular 框架的升级版，却基于 ES6 开发并对其进行了全部的重写。它最大的特点是采用组件式开发，废除了之前 Controller 加 $Scope 的表达，整体更加简洁清晰。在性能上，它较之前版本渲染速度更快。在高可用上，它考虑了移动应用开发以及对未来标准的适配。同时，它采用了依赖注入的机制，再加上整体基于 TypeScript 开发，与 Java 和 C# 语法较为类似，整体开发难度大大降低。

基于以上优势，我们在系统开发的过程中使用 Angular 2 作为前端框架，为用户提供良好的系统交互体验。

2.2.2 Spring Boot

Spring 框架是 Java 程序开发语言中的一种开源框架，为软件开发提供了许多便捷的支持，它主要使用控制反转的关键特性，配合依赖注入的方式，配置管理对象的整个生命周期。利用面向切面编程的特性将事务管理从编程式事务管理改为声明式事务管理，同时使用多样化的持久化技术访问数据，支持多种 WEB 框架简化开发过程等。虽然它的组件是轻量级的，但配置却非常繁琐，需要用大量 XML 文件去定义。此外，许多大型项目往往需要依赖很多包，这些依赖包相互间的依赖和版本间的兼容问题也给开发者带来很多不便。

Spring Boot 框架是由 Pivotal 团队开发的新的 Spring 框架，它继承了 Spring 框架原有的优点，又改进了 Spring 框架开发应用的起始过程。此外，其本身集成的许多框架解决了依赖包版本冲突问题及引用的不稳定等问题。它具有以下特点：创建过程快，集成框架多；通过嵌入 Servlet 容器简化去除应用打包过程，直接运行；starters 管理自动依赖与版本控制；将配置自动化，降低开发成本，也可以在必要时修改默认值；不用管理 XML 配置文件，通过注解直接使用；准生产环境的运行时应用监控；集成云计算技术等等 [35]。

基于以上 Spring Boot 的优秀特性，我们在后端开发的过程中使用 Spring

Boot 框架，保证开发效率，提高系统的可用性。

2.2.3 Docker 容器

在软件开发的过程中，环境配置一直是一项复杂繁琐的内容。我们经常会遇到在自己的机器上正常运行的程序在其他机器上却意外崩溃，这多半是由操作系统的不同、各种库和组件的版本及安装方式不同等环境问题造成的。因此，在软件部署时能复制相同的环境就成为我们关注的问题。

最基础的解决方法是使用虚拟机的方式来安装系统还原原始环境，它在当前系统上安装另外一个系统，却并不为当前系统所感知，只是以普通文件的形式存在。这种方式解决了上述环境问题，但是虚拟机占用的资源很多、启动较慢，还通常会有一些冗余的系统级别操作步骤，因此并不适合实际应用。

Linux 新发展出的另一种虚拟化容器技术可以很好的解决上述问题。它不再是一个完整的操作系统，而只是对进程进行单独隔离，接触虚拟的资源 [36]。这种进程单元相比虚拟机的系统单元启动速度快、占用内存资源少，大大降低了成本。Docker [37] 就是 Paas 供应商 dotCloud 在 2013 年发布的开源应用容器引擎，它是 Linux 容器的一种再次封装，提供接口供开发者方便的创建、启动和管理容器 [38]。Docker 将应用程序本身和其他环境的依赖写入一个文件，称作镜像，运行这个镜像就能生成容器，在镜像中指定的虚拟环境里运行程序。这样做可以方便地提供各种一次性的环境供测试和开发，也可以基于容器随开随关的特性提供弹性的云服务支持。另外，多个容器运行系统的不同服务也便于实现系统的微服务架构。

基于以上优点，我们在系统构建部署时使用一组 Docker 容器，每个容器运行系统的一个服务构成微服务架构便于系统各部分独立开发与扩展。所有的 Docker 容器通过 docker-compose 进行管理。

2.3 数据存储相关技术

为了保证高并发下数据的存储性能和可用性，我们根据不同数据存储形式的特性选择了几种数据存储技术。其中，整个众包测试平台数据采用 MongoDB 数据库存储，知识图谱相关数据采用 Neo4j 图数据库形式存储，读写要求高的数据附加了 Redis 缓存。

2.3.1 MongoDB 数据库

随着互联网 Web2.0 的发展，传统关系型数据库的发展遇到了许多瓶颈，例如对海量数据的高效存储访问，对数据高可用和可扩展的要求。因为有着一致性相关约束，传统关系型数据库在性能上并不能达到期望要求。因此，非关系型数据库收到越来越多关注。NoSQL(Not Only Sql)，即非关系型数据库。它打破了传统关系型数据库的 ACID 理论约束与提前定义好的表结构的存储限制，在大数据的背景下存在着无可比拟的优势，再加上其简单的操作，受到很多开发者的青睐。

MongoDB [39] 是一个用 C++ 开发的，高性能、开源的文档型数据库，是当前 NoSQL 数据库中非常具有代表性的数据库之一。它的数据被分在不同的数据集合中，数据格式以类似 Json 的 BJson 键值对形式存储，可以存储比较复杂，容易变化的数据结构。其查询速度也非常快，可以更加高效地实现像关系型数据库一样的单表查询，也可以构建索引优化查询，成本低且伸缩性强，非常适合于大型网站数据的实时插入，更新和查询。

在众包测试平台系统中，一份缺陷报告会包含多种多样不同形式的信息，例如标题、类型、所属三级页面、描述及截图地址等。这些信息非常复杂且随着系统功能需求变化可能存在变更，因此在 SpringBoot 框架中引入 Mongo DB 依赖，使用 MongoDB 数据库进行存储保证访问的效率。

2.3.2 Neo4j 图数据库

Neo4j 是一个用 Java 语言和 Scala 语言开发的高性能 NoSQL 图数据库。与传统数据库不同，它将数据存储在图中，用节点和关系这两种数据类型来存储。节点通过关系构成的边连接起来，形成关系型网络结构。这种形式非常直观的把各种实体及其间的关联关系具象化地表示出来。当有新的数据实体需要加入的时候也只需要构造新的节点并构造关系与之前的节点关联起来，十分灵活方便。在进行关联性的查询时，相比传统数据库的多表级联查询，Neo4j 底层专门针对图结构进行了优化，每个节点都有指向邻居节点的指针，这样在遍历的性能和查询语句的使用方便性上都有很大优势。

在我们构建众包测试任务相关知识图谱时，同样是构建一个把不同种类信息连接在一起的关系网络，因此我们使用图数据中最常见的 Neo4j 来构建知识图谱。单独构造众包任务、众包工人、缺陷报告相关实体，并用工人参与众包

任务，工人撰写缺陷报告等关系连接起来，这样在我们给不同众包工人分配任务时，可以直接根据他们以前参与过的众包任务和撰写过的缺陷报告的相关属性推荐类似的众包任务。

2.3.3 Redis 缓存组件

由于系统中部分数据需要访问的频率高，并发多，但是频繁操作数据库可能会造成锁表时间长，从而导致系统性能下降。我们可以将这部分数据直接放在缓存中，如图 2-3将硬盘读写操作变成内存获取，那么就可以很大程度上提高速度和并发量。

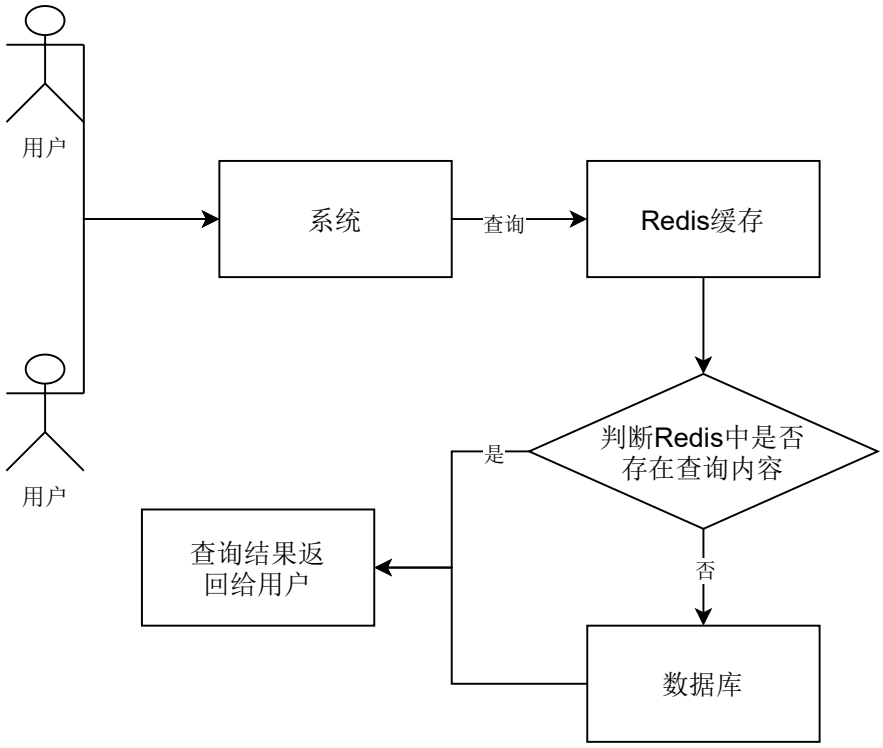


图 2-3: Redis 示意图

Redis 正是满足我们要求的内存数据存储系统。它是由 ANSI C 语言开发的开源 NoSQL 数据库，存储的是 key-value 键值对 [40]。它使用单线程多路 IO 复用模型，读写速度远大于一般的数据库。它支持多种数据结构满足我们不同场景下的存储需求。它的操作都是原子操作，不会出现不一致的问题，且可以将内存中的数据持久化到磁盘上，保证数据不会丢失。因此，它被广泛的应用作缓存数据库，减少底层数据库的压力。

在众包测试任务分配技术中，我们要针对不同众包工人的特点分配适合的众包任务，当同时有多个众包工人加入时，这些众包任务相关数据被大量访问，因此使用 Redis 来缓存这些高频率使用的数据，提高系统性能。

2.4 本章小结

本章主要概述了基于知识图谱的众包测试任务分配技术以及众测平台开发过程中所使用到的算法和相关技术框架。首先介绍了系统架构相关技术前端 Angular2, 后端 Spring Boot, 以及 Docker 容器，其次介绍了数据存过程中使用的 MongoDB 数据库，Neo4j 图数据库，以及 Redis 缓存，最后介绍任务分配技术中使用到的知识图谱和 RippleNet 模型。通过对这些框架及算法的详细了解，为后续开发奠定了坚实的理论基础。

第三章　需求分析与概要设计

本章通过对“基于知识图谱的众包测试任务分配技术”进行整体分析与模块划分，理清了系统的各项需求并进行了架构层面的设计与各模块的详细设计，为具体实现部分奠定了基础。

3.1　系统整体概述

众包测试平台主要服务对象包括众包任务发布者和众包工人。众包任务发布者在平台上按照规定的格式上传待测软件、测试需求、测试任务具体的三级页面信息以及任务时间，参与人数等信息。众包平台在指定时间开发任务，众包工人选择并加入众测任务，根据分配的具体任务在相关页面上进行测试，编写测试用例，提交测试报告。整个功能图见图 3-1。

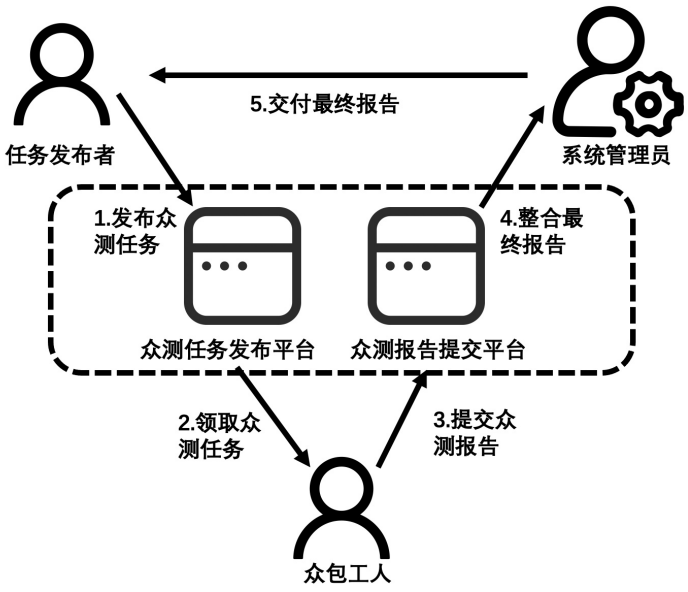


图 3-1: 系统功能图

由于众测任务发布主要在外部平台上实现，我们的系统只关注于众测报告

提交平台功能的实现，供众包工人提交众测报告使用。然而众包工人均为线上报名参与众包任务，其技术能力和信用度参差不齐，因此要求我们的系统在界面上简单易用，在任务分配上力求精准。任务分配主要分成两大部分，一是在用户领取任务填写报告时进行三级页面测试任务的分配，另外则是审核任务的分配。这两种任务分配方法均基于知识图谱技术，采用算法模型训练得到结果。因此将系统分成如图 3-2三个模块：知识图谱数据获取模块、知识图谱特征学习模块以及知识图谱任务分配模块。

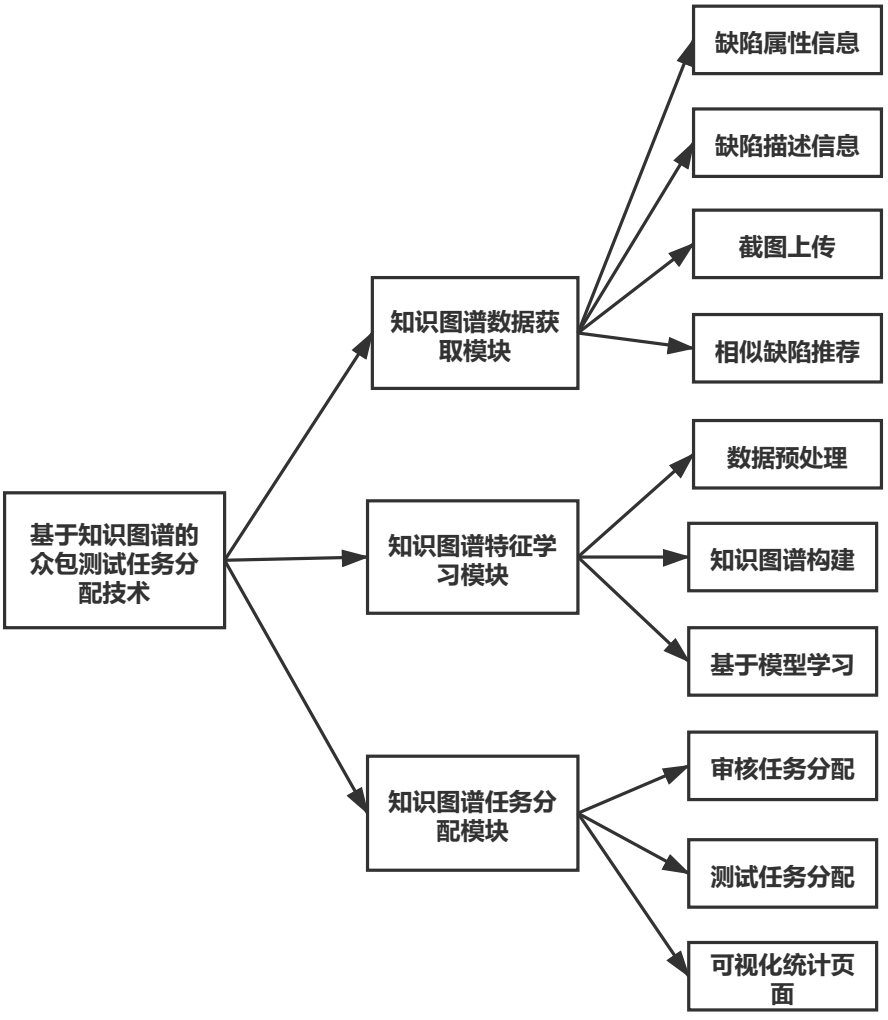
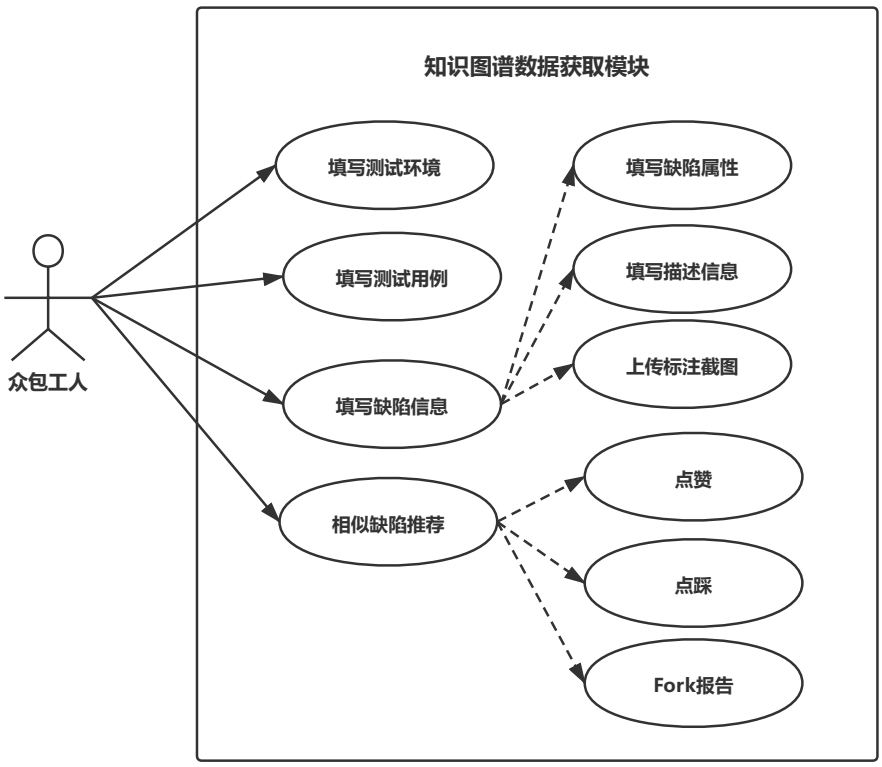


图 3-2: 系统模块划分

3.2 系统需求分析

3.2.1 知识图谱数据获取模块用例分析



]

图 3-3: 知识图谱数据获取模块用例图

知识图谱数据获取模块主要作用于众包测试过程中，根据众包测试流程将提交的测试用例和缺陷报告数据持久化存储，利用这些数据构建领域内知识图谱。其实现的主要功能如图 3-3所示。

1）众包工人填写当前测试环境。在不同的设备及系统版本环境下可能会出现兼容问题导致的缺陷，因此我们首先需要根据要求填写当前测试环境，方便后期对缺陷的复现和修复。

2）众包工人填写测试用例。测试用例有助于我们理清测试思路，保证当前任务下测试的覆盖率。因此我们先参考测试用例编写规范编写测试用例，再据此开展测试工作。

3）众包工人测试找到缺陷并填写缺陷基本信息。根据 [41] 中定义的缺陷具有的基本信息，结合移动测试和 Web 测试的特点，我们定义平台中缺陷基本

信息包括缺陷所属的三级页面、类别、严重程度、复现程度，以及所关联的测试用例，其中三级页面信息由我们接受的众测任务决定。类别主要包括不正常退出、功能不完整、安全、性能、页面布局缺陷、用户体验和其他共七种。严重程度有待定、较轻、一般、严重和紧急五种。复现程度包括无规律复现、小概率复现、大概率复现、必现和其他。

4）众包工人填写缺陷的标题和详细描述。根据规范要求，缺陷描述简洁清晰，按序号排序，复现步骤连贯，描述清楚前置条件和后置条件。

5）众包工人上传缺陷截图。众包工人需要根据描述中的问题上传对应的截图帮助复现问题。同时在平台上可以直接对截图进行标注，突出问题的具体位置或需要注意的问题。

在众包工人填写缺陷报告的同时，基于协作式众包的要求，平台根据工人选择的缺陷属性及对应的描述信息自动推荐相似的缺陷报告，具体算法设计见后文算法设计小节。众包工人可以查看这些推荐的相似报告的具体内容及截图。如果认可报告内容，则直接对该报告进行点赞操作。反之对这份报告点踩，继续提交自己的缺陷报告。另外，众包工人还可以对当前报告执行 fork 操作，这份报告内容会自动同步到当前书写的报告，众包工人只需要基于其进行修改补充内容或增加截图。

根据上述需求，此模块的用例表见表 3-1所示。

3.2.2 知识图谱特征学习模块本体库构建

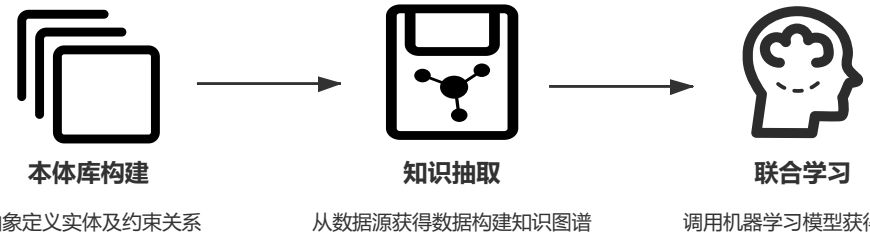


图 3-4: 任务分配步骤

此模块是本任务分配技术的关键，通过知识图谱和深度学习的结合，给出基于众包工人和众包任务特性的个性化分配任务结果，其主要包含三个阶段：本体库构建、知识抽取和联合学习获得分配结果，如图 3-4。本体库构建即对知识图谱中的概念实体进行抽象定义；知识抽取即从不同的数据来源获得满足

表 3-1: 知识图谱数据获取模块用例表

ID	UC1
用例名称	获取众包测试数据
用例目标	众包工人提交众测报告
参与者	众包工人
前置条件	众包工人参与一场众包测试
后置条件	众包工人完成一场众包测试任务
基本操作流	1. 填写当前测试环境 2. 接受测试任务 3. 填写测试用例 4. 填写缺陷属性及内容 5. 提交报告
可选操作流	4a. 查看相似缺陷报告发现描述相同 1. 对相似报告进行点赞 4b. 查看相似缺陷报告发现描述错误 1. 对该报告进行点踩 4c. 查看相似报告发现需要补充 1. 点击“一键 Fork”按钮，原始报告自动复制到当前报告中 2. 在该报告基础上进行编辑

要求的知识数据并在特定的图数据库中进行存储；联合学习利用模型，以用户历史行为记录和知识图谱作为输入进行学习，最终输出特定用户对特定任务的感兴趣程度。由于知识抽取和联合学习过程均涉及到具体的代码设计，故在此章节只完成本体库构建部分设计。

要实现基于知识图谱的任务分配，首先需要构建一个完整、准确的知识图谱，能够尽可能包含领域内的重要关系，为后期的分配奠定语义逻辑基础。而构建知识图谱有自顶向下和自底向上两种方式。自底向上方式从开放领域数据中自动提取实体，选择其中置信度高的实体构建本体库，确定其相关关系。而自顶向下的方式则根据经验首先定义好本体与数据关系，再从数据中匹配出对应的实体构建知识图谱，主要用于知识范围相对固定，概念较为明确的情况。由于本任务分配系统仅用于众包测试任务分配领域，具有较为明确的领域知识，故采用自顶向下的方式。这也要求我们对领域内知识具有详细的了解，准确清晰地定义其中的实体关系、属性值域，以及约束关系。

测试任务分配和审核任务分配虽然同属于众包测试领域，但分配的的实体分别是三级页面测试任务和缺陷报告，影响其分配的要素也存在不同。为了更准确的根据属性对其个性化分配，我们分别分析影响其分配的要素，构建一个

同时包含两种分配方式关键要素的知识图谱。在联合训练阶段再各自筛选出影响其分配的实体和关系作为模型输入得到最终分配结果。

首先考虑测试任务分配，其分配的实体是三级页面测试任务。通过对测试任务相关知识的调研，发现可能影响其分配的知识要素如表3-2所示。其中，众测任务要素提供了其所属的案例、包含的三级页面信息及对应的关键词信息。众包工人要素提供了其姓名、学历等信息。

表 3-2: 测试任务分配领域知识要素描述

序号	知识要素名	描述
1	众测任务	众测任务要素主要包括测试的案例信息、三级页面信息，以及包含的关键词信息。
2	众包工人	众包工人要素包括众包工人的姓名、学历信息。

分析表3-2中确定的知识要素可以发现，不同的众测任务可能有相同的一级页面、二级页面及关键词信息，这些信息决定了每个测试任务的独特性，会在一定程度上影响任务分配的结果，因此我们将其抽象出来形成单独的概念。又因为“一级页面”、“二级页面”、“三级页面”同属于界面，区别仅在于它们所属的层级不同，因此我们将它们均归为“页面”概念。最终，形成“众测任务”、“众包工人”、“页面”、“关键词”四个概念。

表 3-3: 测试任务分配知识图谱的实体属性

实体类 (Class)	名称	属性类别	标识	值域	描述
众测任务 (Task)	一级页面	对象属性	hasPage1	Page	该任务的一级页面信息
	二级页面	对象属性	hasPage2	Page	该任务的二级页面信息
	三级页面	对象属性	hasPage3	Page	该任务的三级页面信息
	参与工人	对象属性	hasWorker	Worker	该任务参与的众包工人
	关键词	对象属性	hasKeyword	Keyword	该任务的关键词信息
页面 (Page)	名称	数据属性	pageName	String	该页面标题名称
	级别	数据属性	layer	Int	该页面所属级别
	所属众测任务	对象属性	ownedBy	Task	此页面在哪一个众测任务中用到
众包工人 (Worker)	姓名	数值属性	name	String	众包工人的姓名
	学历	数值属性	education	String	众包工人的学历信息
	参与任务	对象属性	inTask	Task	众包工人参与的众测任务
关键词 (Keyword)	名称	数据属性	word	String	该关键词名称

接着我们定义每个概念类的属性、值域和约束信息。实体类主要包含两种属性：对象属性和数据属性。数据属性是实体的固有属性，其值为一个基本数

据类型。对象属性描述了两个实体之间的关系，值域是一个对象。例如，众测任务类含有一级页面属性，其值域为 Page 类，代表该任务的一级页面信息。定义页面类的一个数据属性为 layer，代表该页面是第几级页面。此外，实体内的某些属性是互逆关系的，例如众测任务类的一级页面、二级页面、三级页面信息与页面类的所属众测任务就是互逆关系的。

最终创建出来的的任务分配领域实体及属性如表 3-3所示。

审核任务分配构建本体库过程同上，由于审核的主体是缺陷报告，其可能受影响的知识要素如表 3-4所示。其中，缺陷报告要素包含了缺陷的漏洞分类、严重等级、复现程度、父子报告、点赞点踩信息。众测任务要素提供了测试的三级页面信息和任务的关键词信息。众包工人要素提供了众包工人的姓名和学历信息。

表 3-4: 审核任务分配领域知识要素描述

序号	知识要素名	描述
1	缺陷报告	缺陷报告包含了缺陷的漏洞分类、严重等级、复现程度、父子报告、点赞点踩信息。
2	众测任务	众测任务主要包含测试的三级页面信息和任务的关键词信息。
3	众包工人	众包工人主要包括了众包工人的姓名、学历信息。

结合众测知识分析表 3-4中的知识要素，缺陷报告的漏洞分类区分了不同缺陷报告的类型，可能影响审核任务分配的结果，因此将其抽象出来；缺陷报告的父子报告表明了两份报告是对于同一个问题的不同描述，关系密切，也会影响审核任务分配,但其与报告本身同属于报告，区别仅在于其在 Fork 报告树中位置不同，因此合并为“缺陷报告”实体。缺陷报告的历史点赞点踩信息同样可能影响审核任务分配结果，其对应的主体仍是众包工人，因此作为缺陷报告对象属性存在。最终形成“缺陷报告”、“漏洞分类”、“众测任务”、“众包工人”四个概念。对于每个概念确定属性、值域以及约束信息，最终形成的审核任务分类知识图谱的实体类如表 3-5所示。

通过以上步骤，任务分配相关知识图谱的本体库构建完成，在实现阶段我们只需要从数据源获得数据并存储在图数据库中填充知识图谱，就可以作为输入利用模型获得任务分配结果。

表 3-5: 审核任务分配知识图谱的实体属性

实体类 (Class)	名称	属性类别	标识	值域	描述信息
缺陷报告 (Bug)	严重等级	数值属性	severity	Int	缺陷报告的严重等级 (待定/较轻/一般/严重/紧急)
	复现程度	数值属性	recurrent	Int	缺陷报告的复现程度 (其他/无规律复现/小规律复现/必现)
	漏洞分类	对象属性	hasType	BugType	缺陷报告的漏洞分类
	所属任务	对象属性	inTask	Task	缺陷报告所属于的三级页面任务
	所属工人	对象属性	ownedBy	Worker	缺陷报告所属的众包工人
	操作工人	对象属性	thumbsBy	Worker	缺陷报告点赞点踩的众包工人
	父报告	对象属性	hasParent	Bug	缺陷报告的父报告
	子报告	对象属性	hasChild	Bug	缺陷报告的子报告
众测任务 (Task)	三级页面	数值属性	page	String	缺陷报告的三级页面信息
	含有报告	对象属性	hasBug	Bug	众测任务下的缺陷报告
众包工人 (Worker)	姓名	数值属性	name	String	众包工人的姓名
	学历	数值属性	education	String	众包工人的学历信息
	操作报告	对象属性	thumbBug	Bug	众包工人点赞点踩的缺陷报告
	撰写报告	对象属性	hasBug	Bug	众包工人编写的缺陷报告
漏洞分类 (BugType)	分类	数据属性	type	Int	该缺陷报告的漏洞分类

3.2.3 知识图谱任务分配模块用例分析

知识图谱任务分配包括测试任务分配和审核任务分配。测试任务分配是在众包工人第一次或提交缺陷报告后为众包工人分配在指定三级页面寻找缺陷的模块，而审核任务分配是在相同界面上显示推荐用户去审核某份报告，即对其进行点赞或点踩操作。这份审核列表的得出综合考虑了众包工人历史审核缺陷报告的偏好和待审核报告已点赞、点踩的次数，并剔除了该众包工人已审核过的缺陷报告。这样既保证了众包工人审核到自己感兴趣的缺陷报告，又尽量使得每份报告均有足量的点赞、点踩记录便于对其后续评价。

以 M 平台一次 WEB 应用众包测试为例，需要对某失物招领网站的系统功能进行测试，其三级页面信息如表表3-6所示。其中，一级页面即用户访问网站首先看到的页面，二级页面即用户在一级页面发生点击操作后跳转的新页面。三级页面即在二级页面基础上点击跳转出现的新页面。当跳转层数达不到三级时，在表中多出的级数上重复上层页面名称。

该模块实现后，可以为众包工人提供以下统计信息的可视化展示，具体如

表 3-6: 某 WEB 应用功能测试三级页面需求

一级页面	二级页面	三级页面
注册	注册	注册
登陆	邮箱登陆	邮箱登陆
	手机号登陆	手机号登陆
首页	浏览分类	校园卡
		身份证
其他	其他	其他

用例图3-5所示：

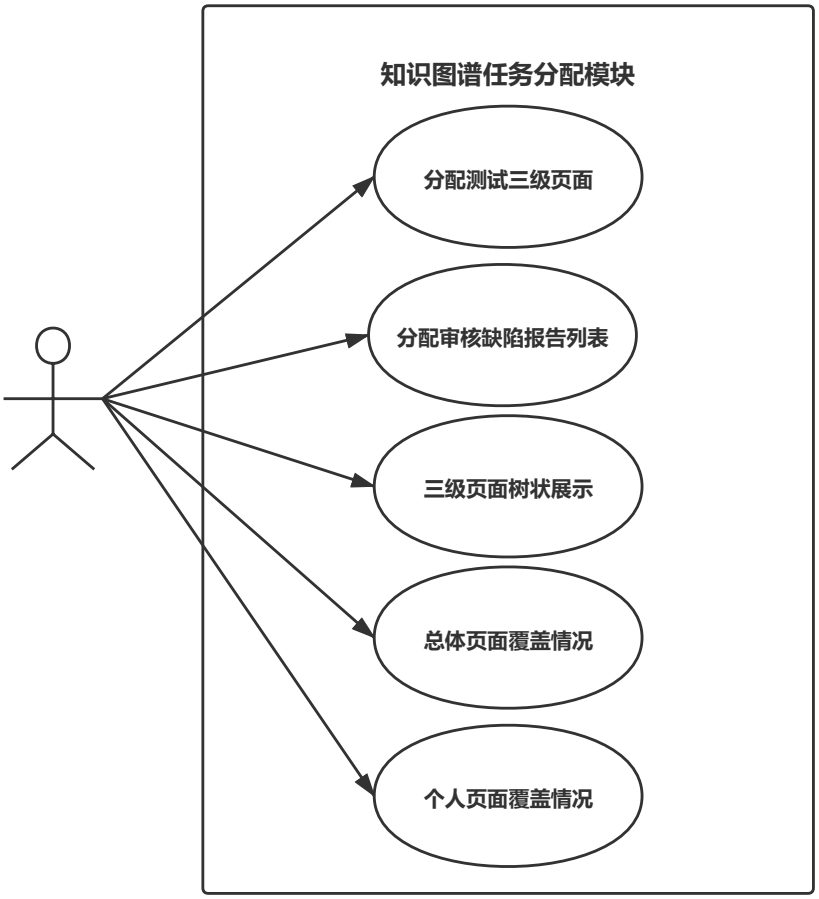


图 3-5: 知识图谱任务分配模块用例图

1）分配去哪个三级页面继续完成众测任务。当众包工人在完成所有三级页面的任务时，提示已完成所有任务。

2）分配去审核的缺陷报告列表。众包工人可以点击具体的缺陷报告并对

其进行点赞点踩评价。

2）三级页面层级结构的树状显示，帮助众包工人了解和选择不同页面下的测试任务。

3）每个三级页面下缺陷报告个数，即总体页面覆盖情况。

4）此众包工人提交的缺陷报告的页面覆盖情况。

其中，分配任务通过文字信息高亮展示。三级页面层级结构通过可视化树状形式展示构成主图。每个节点是一级页面，每条边代表从上到下存在跳转关系。总体和个人页面覆盖情况通过在图中节点中以颜色标识区分，并以图示辅助，力求做到简明扼要。

为了搭建一个能够准确反映系统中主要实体关系的知识图谱帮助后期任务分配系统基于缺陷报告和众包工人特性的任务分配，同时对分配结果进行验证，系统还需要记录众包工人每一次接收或拒绝分配的测试任务或待审核缺陷报告的情况。整个模块的用例表如表3-7所示。

表 3-7: 知识图谱任务分配模块用例表

ID	UC2
用例名称	知识图谱任务分配
用例目标	帮助众包工人高效完成众测任务，提交缺陷报告
参与者	众包工人
前置条件	1a. 众包工人初次进入众测任务 1b. 众包工人提交缺陷报告
后置条件	众包工人继续完成测试用例和缺陷报告
基本操作流	1. 树状展示测试页面层级结构 2. 展示总体页面覆盖情况和个人页面覆盖情况 3. 系统分配众包工人新的测试任务 4. 系统分配众包工人待审核缺陷报告列表 5. 众包工人点击待审核缺陷报告查看并审核 6. 众包工人接受任务并在指定页面完成测试用例和缺陷报告 7. 系统记录众包工人接收任务与否
可选操作流	4a. 众包工人未按照分配情况审核缺陷报告 6a. 众包工人拒绝该测试任务，自行进行探索性测试

3.2.4 非功能需求分析

在满足以上功能需求的同时，系统同样需要满足非功能需求的约束，才能让系统真正可以方便高效地服务于用户。在系统设计过程中，我们提出了如下

非功能需求：

1）在性能方面，系统需要保证众测平台相关服务，包括测试报告的提交、查看等基础操作在响应时间上不超过 1 秒。每次测试任务和审核任务分配与当前任务完成情况的可视化展示应在 5 秒内，以此保证分配结果的时效性。

2）在高可用性方面，对于用户的每次输入操作进行数据的筛查，防止出现数据异常问题导致系统崩溃。数据存储时在本地做好备份，防止数据丢失。在部署时采用主从部署，使得系统崩溃时能够快速恢复。系统应该能承受 200 名用户同时接受众测任务并提交测试报告。每周内系统奔溃次数不超过 1 次，每次时间不长于 10 分钟。

3）在可扩展方面，系统代码的编写应该符合面向对象的规范，所有算法和模块相互独立可以替换。使用 Nginx 负载均衡机制保证当并发量提升时可以通过增加服务器的方式缓解压力。

4）在易用性方面，界面符合人机交互的基本原则，用户的所有操作有相应的提示。同时配备操作文档保证用户快速熟悉系统功能。

3.3 系统总体设计

此模块介绍系统整体设计。先从架构层面介绍系统各模块间的协作方式，再以“4+1 视图”规范对系统各个模块进行概要设计，以达到系统预期的功能及非功能需求。

3.3.1 系统架构设计

考虑整个众测流程，先由任务发布者在外部平台上创建众测任务，众包工人登录外部平台浏览信息参与任务，获得 token 信息，之后通过 URL 跳转到本系统执行具体的众测任务。token 是一个序列化的唯一字符串，包含了众测任务和众包工人相关信息。平台前端通过 HTTP 请求调用知识图谱数据获取模块和知识图谱任务分配模块，任务分配模块调用 Python 脚本编写的知识图谱特征学习模块。外部平台也可通过 Restful API 与本系统进行交互。这些服务通过 MongoDB 数据库、Neo4j 图数据库，以及 OSS 将信息持久化存储。整体架构如图 3-6所示。

其中，系统前端主要包括数据获取视图以及任务分配视图，采用 Angular2 框架，具有组件化开发以及高效渲染的特点。同时使用 Bootstrap 可视化库，

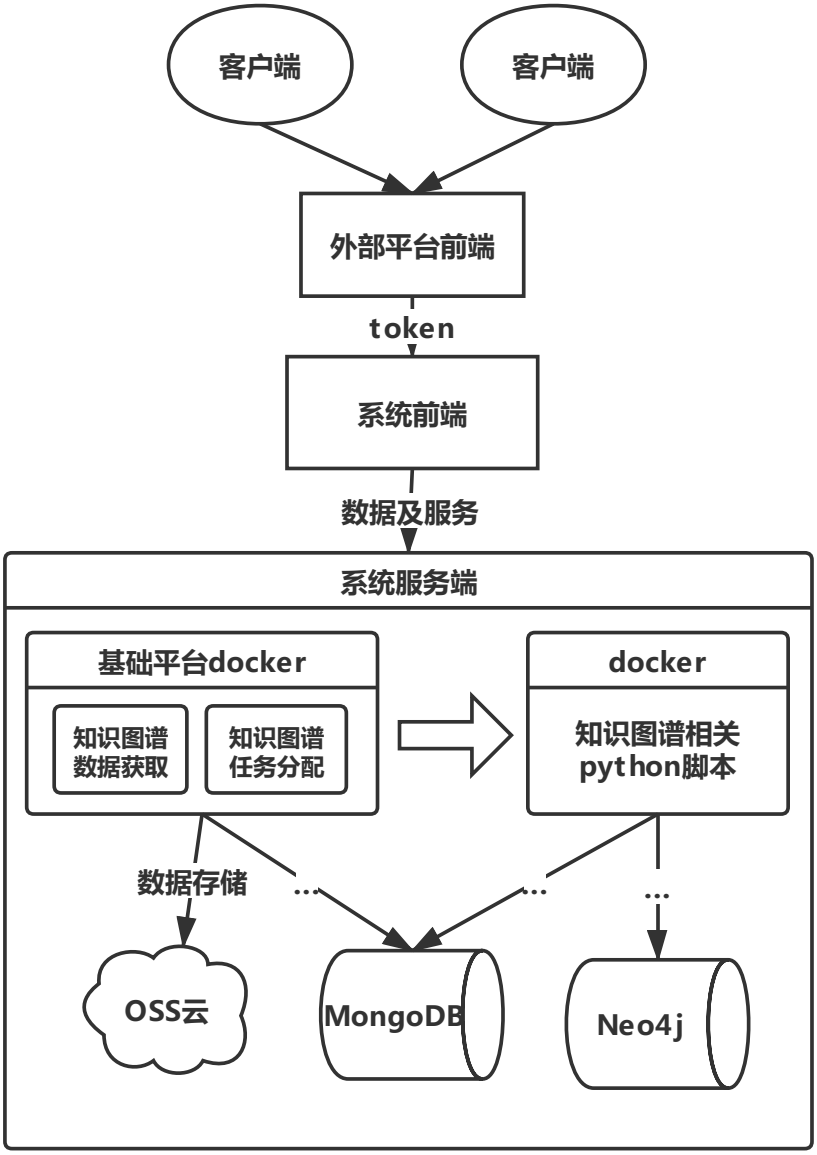


图 3-6: 系统整体架构设计

通过调用其丰富的 UI 组件达到预期效果。在数据生成可视化图表方面选择了 Echarts 库，基于 Canvas 实现各种图表类型，简单高效。采用 jQuery 库进行事件处理，简化了对 DOM 和事件的操作。同时，使用 jNotify 插件实现无阻塞的消息通知，使用灵活。

后端平台部分包括数据获取模块和任务分配模块，使用 Spring Boot 框架，采用经典的 MVC 模式组织代码。Controller 层根据路径接收前端的请求并转发到对应的 Service 层。Service 层负责业务逻辑的处理，并将对数据的增删改查

相关操作转发到 DAO 层处理，由 DAO 层来直接与数据库交互。知识图谱相关任务分配模块使用 Python 脚本进行处理，由 Service 层直接调用 Python 相关代码来获取任务分配结果进行展示。

使用 Redis 作为缓存。虽然数据最终会持久化到数据库中，但由于数据在数据库的查询操作较为费时且并发度不够，对于高频查询的数据先放入 Redis 缓存中，由 Service 层先访问 Redis 读取，减少数据库操作次数，更加高效。

在前端和后端之间使用 Nginx，通过反向代理的方式实现负载均衡，帮助后端服务器在流量增大的情况先实现水平扩展。

3.3.2 系统视图模型

“4+1 视图”是由 Philippe Kruchten [42] 在 1995 年提出的对软件逻辑架构的一种描述方式，从五个不同的角度描述软件体系结构，为程序开发人员广泛使用。其主要结构如图 3-7所示。

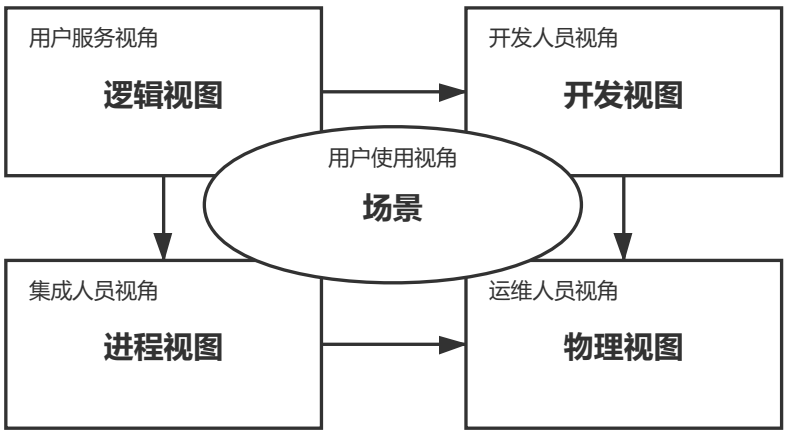


图 3-7: 4+1 视图结构图

本节将从“4+1 视图”的角度介绍本系统整体设计。场景视图为五个视图中的核心，它确定了系统边界、系统用户以及系统功能和场景，常通过 UML 用例图的形式表示。由于 3.1 小节中已分别描述各模块用例图及相关使用场景，故在此不做赘述。

逻辑视图是对系统职责的逐级划分，体现了系统的功能需求，通过系统分解的各个逻辑元素抽象及其间交互关系表示。如图 3-8为众包测试任务分配技术的逻辑视图。TaskAssignService 负责总体的任务分配，在算法上它依赖于 Python 脚本中的 predict() 方法，在这个方法中调用 ratings _final 获得该众包工人历史偏好记录，输入 RippleNet 网络进行神经网络的联合学习获得分配结果。至于知识图谱的构建所需要的数据缺陷报告数据收集来自于 BugService，由于我们要获得缺陷的多维属性，因此其调用 humbsUpService 获得每个缺陷的点赞点踩信息，BugHistoryService 获得缺陷的父子报告和根报告信息。每个任务的信息依赖于 TaskService，它会将三级页面信息拆分成具体的任务等待分配。至于反映任务分配技术效率的用户行为记录信息，由 ActionRecordService 负责，通过 TaskAction 实体和 ReviewBugAction 实体来进行记录。

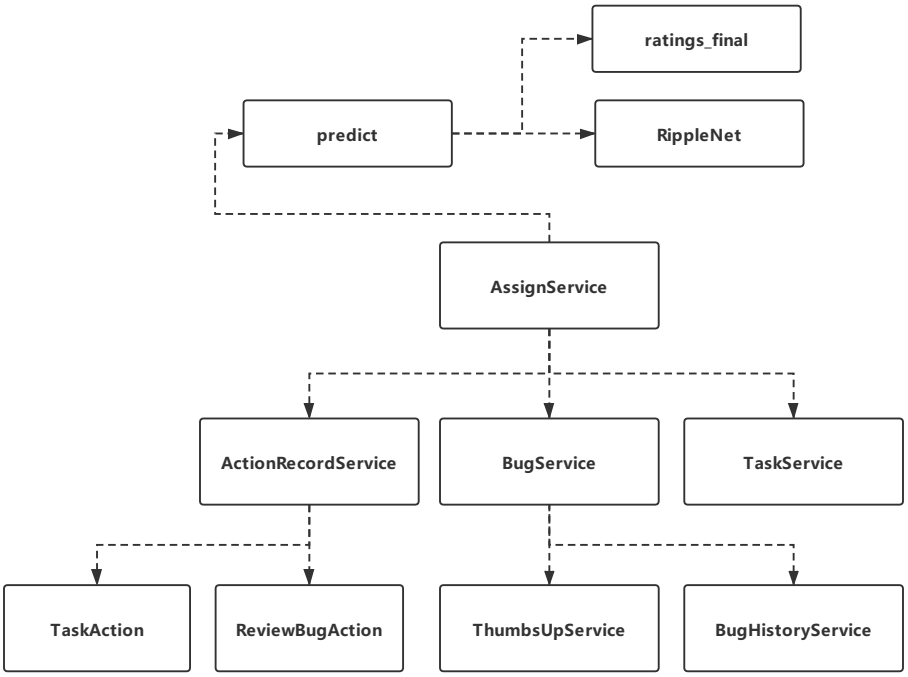


图 3-8: 逻辑视图

开发视图主要用来描述软件中各个模块的组织与管理。如图 3-9为本系统的开发视图。其中前端使用 Angular2 框架中约定的 Component 组件、Model 模型、Asset 静态文件以及负责前后端交互的 Service 构成。后端 Controller 包接受前端请求，并转发到对应的 Service 类。Service 包负责业务逻辑的处理，并将对数据的增删改查相关操作转发到 DAO 层处理，由 DAO 层来直接与数据库交互。他们之间的依赖关系通过 Spring Boot 中的 @Autowired 注解实现。而

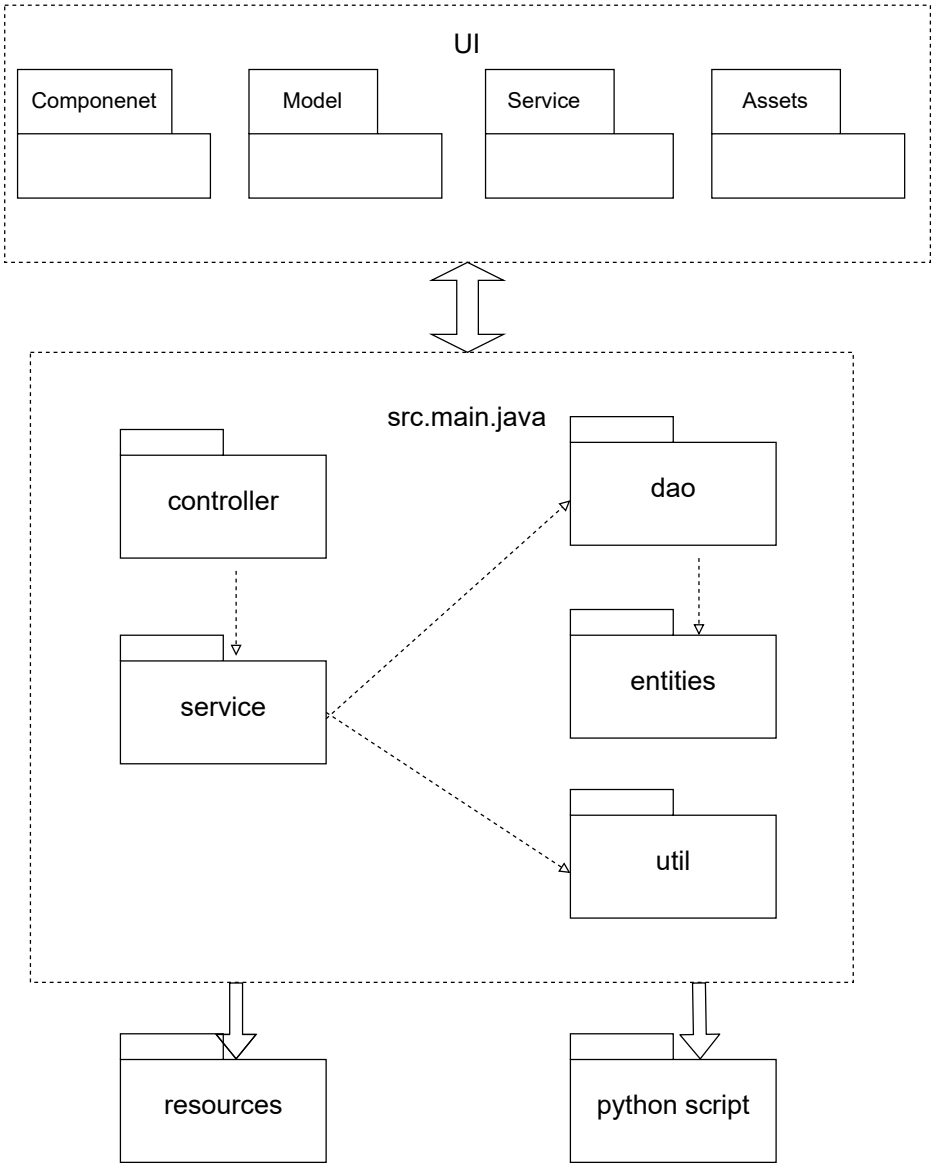


图 3-9: 开发视图

Entities 包保存了系统中的所有实体信息，与数据库中的表一一对应。Util 包封装了一些通用的算法或操作，由 Service 层调用。同时，系统中基于知识图谱的任务分配相关 Python 脚本也经过 Util 包调用获得结果。

进程视图关注系统动态运行特性与逻辑架构之间的交互关系。如图 3-10为本系统的进程视图。先由前端进程发送 HTTP 请求给 Nginx 服务器，Nginx 服务器考虑负载均衡将请求转发给对应的服务端进程。服务端进程调用 MongoDB 获得众包工人以及整个众包任务的数据。分配任务时调用 Python 脚本获得任

务分配结果，Python 脚本访问 Neo4j 图数据库获得该工人的偏好历史数据和知识图谱数据一起输入训练好的模型获得分配结果。同时服务端调用 MongoDB 数据库获得缺陷报告等数据。在众包工人执行任务时，提交的测试用例和缺陷报告数据将被持久化到数据库中。对于热点数据，如推荐缺陷列表等，先从 Redis 缓存中查找，再访问数据库。当存在更新时同时更新 Redis 和数据库记录。当所有流程结束后，服务端返回 HTTP 响应给 Nginx 服务器，Nginx 返回结果给前端进程，在页面上产生相应的效果。

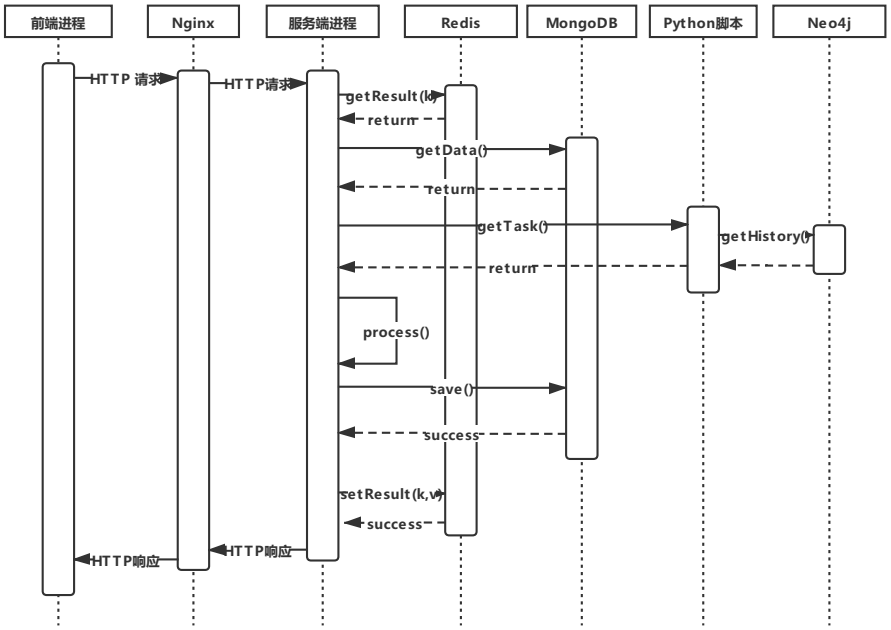


图 3-10: 进程视图

物理视图是对软件交付信息的描述，帮助软件运维人员解决系统的拓扑结构、安装通信等问题，如图 3-11所示。用户通过浏览器发出 HTTP 请求访问 WEB 服务器，经过防火墙过滤后访问 Nginx 服务器进行负载均衡，将请求转发给对应的后端服务器。后端服务器负责相应的业务逻辑以及热点数据的缓存服务，并将数据相关操作传递给数据库服务器。同时后端服务器调用 Python 脚本获得任务分配结果。所有的模块均单独部署，并独立打包为 Docker 容器，具有良好的伸缩性和可移植性。

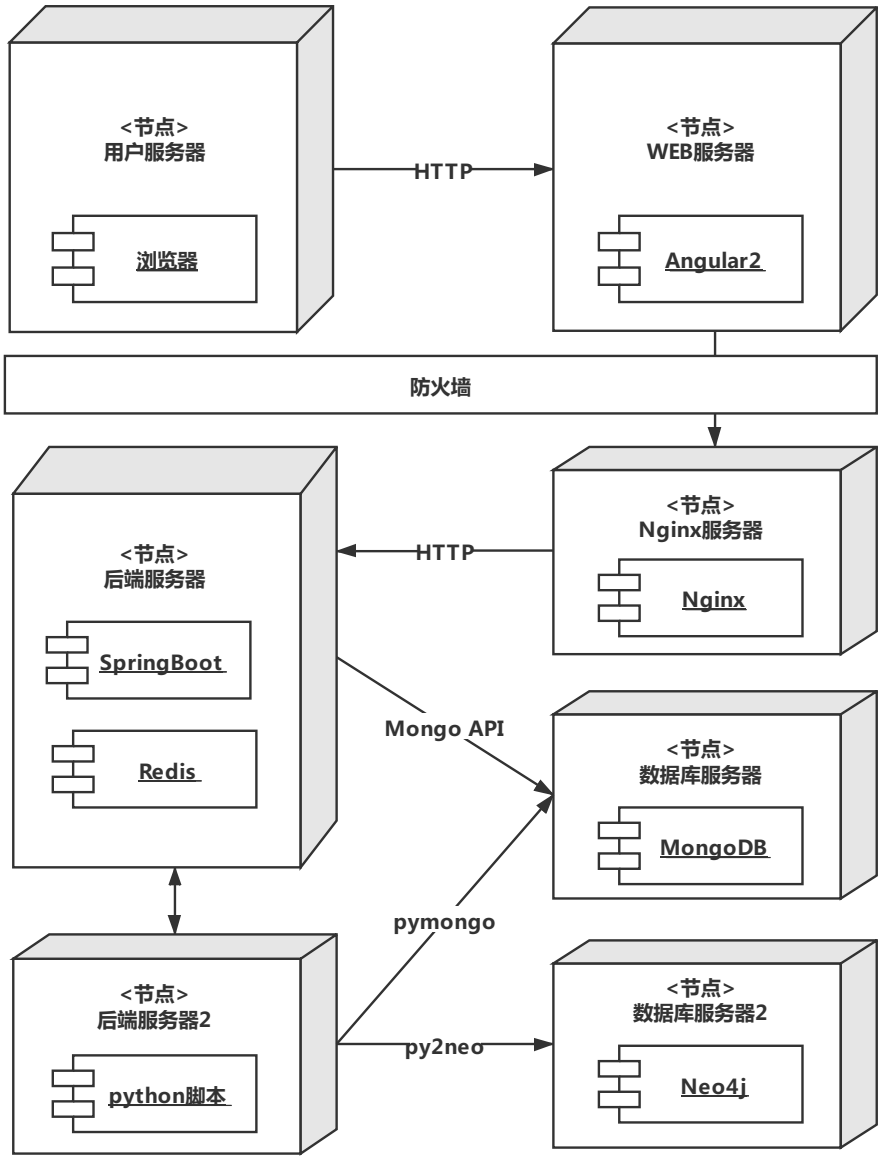


图 3-11: 物理视图

3.3.3 系统实体类设计

本系统中包含的主要实体类如图 3-12所示。下面详细介绍图中各实体类的具体属性及其相互关系。

1）Bug 类是所有实体类的基础，它包含了缺陷的唯一标识 ID、所属 bug _page 的 ID、所属报告 ID、严重程度、复现程度、缺陷所属类别、创建时间、图片在 OSS 存储的 URL 地址、缺陷报告的标题及具体描述信息。每个 Bug 实体也包含了许多引用。

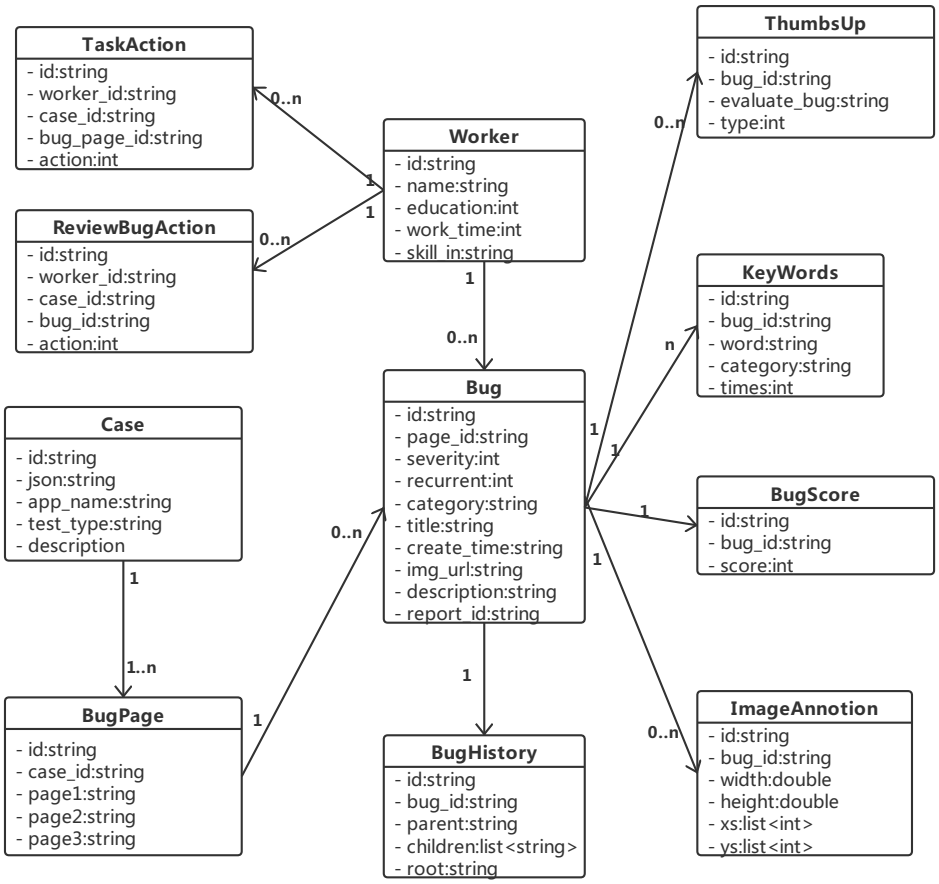


图 3-12: 实体类图

2）BugHistory 记录了缺陷报告经过 Fork 操作后的父子报告信息，parent 是此缺陷报告的唯一父节点，children 则记录了该报告的子节点列表，形成了树状报告结构，而 root 就是该报告树的根节点。

3）ThumbsUp 类记录了缺陷报告的点赞点踩记录。每一条记录中包含原始缺陷报告 ID，被操作的缺陷报告 ID 以及操作类型。

4）KeyWords 类记录了缺陷报告信息的关键词，包括对应的缺陷报告 ID，具体的实体词、所属的分类，以及其出现的次数。通过关键词的构建可以帮助系统快速找到相似缺陷报告，帮助任务分配。

5）BugScore 类记录缺陷报告的人工审核评分，每个条目包含缺陷报告 ID 及对应的分数。通过 BugScore 可以帮助度量每份缺陷报告乃至工人的优劣。

6）ImageAnnotion 类记录了缺陷报告中上传截图的标注信息。每一个条目对应一份截图，记录了其所属的缺陷报告 ID，图片的宽和高，及所有标注点以

左上角为零点的 (x,y) 坐标轴信息组。

7）BugPage 类每一个条目对应系统任务分配中的一个任务。它是由所属的题目 ID 和具体的三级页面组成的。一个 BugPage 可以对应多份缺陷报告，即可以重复分配。

8）Case 类记录每一次众包测试过程中具体的题目，包含了待测软件名称及类别信息，测试描述，需求文档，及根据要求格式化的 json 文件。json 文件中包含三级页面信息，根据它解析出对应的 BugPage。

9）Worker 类记录了众包工人的基本信息，包括工人 ID、姓名、学历、工作时长、以及其擅长的类别。其信息来自于与外部系统的交互。

3.4 其他算法设计

3.4.1 任务分配召回率算法设计

通用的衡量任务分配算法的标准有两种，准确率和召回率。由于分配任务无法得到最优解，因此无法使用准确率衡量。在此，我们基于召回率对本任务分配算法进行评价，并基于此对分配算法进行改进。

对于测试任务分配，众包工人可以选择接受也可以拒绝并自行探索测试。在系统中我们记录每次出现任务分配界面后众包工人选择接受分配任务与拒绝的次数及比率，计算可以得到总体任务分配召回率：

$$S_1 = \frac{\sum 用户接受分配任务次数}{\sum 总分配任务次数} \tag{3-1}$$

为了记录所有众包工人对于分配测试任务的选择，并不影响原始功能逻辑，在数据库中我们新建 TaskAction 表记录，具体属性及含义如表 3-8所示。当众包工人发生选择时调用此表新增记录。

对于审核任务分配，根据系统中记录的众包工人按照分配的审核缺陷列表对缺陷报告进行点赞点踩的次数和总点赞点踩次数的比率，计算得到审核任务分配的召回率：

$$S_2 = \frac{\sum 用户审核分配任务次数}{\sum 用户审核任务总次数} \tag{3-2}$$

表 3-8: TaskAction 表定义

属性	数据类型	含义
id	String	唯一标识 ID
user_id	String	工人 ID
case_id	String	所属的案例 ID
bug_page_id	String	所分配的任务 ID
action	Int	具体行为（0 接受，1 拒绝）

同样为了记录众包工人对于审核缺陷报告的选择，在数据库中新建 ReviewBugAnction 表，具体属性即含义如表 3-9所示，当众包工人对待审核任务进行操作时新增记录。

表 3-9: reviewBugAction 表定义

属性	数据类型	含义
id	String	唯一标识 ID
user_id	String	工人 ID
case_id	String	所属的案例 ID
bug_id	String	分配的审核缺陷报告 ID
action	Int	具体行为（0 接受，1 拒绝）

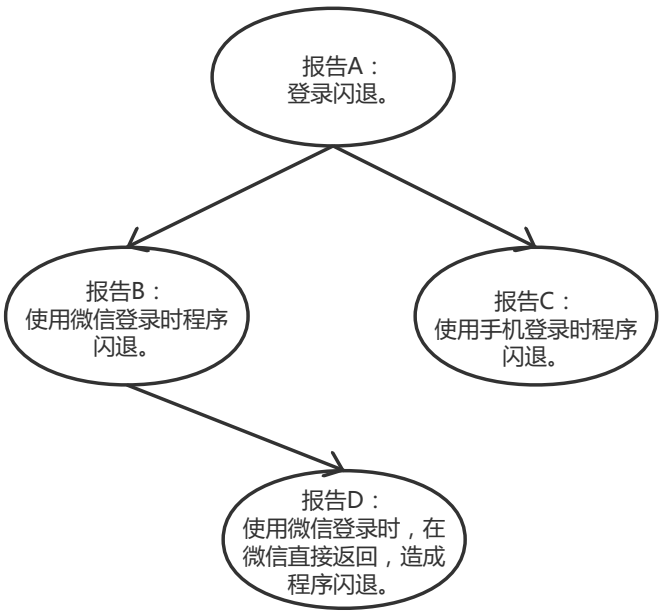


图 3-13: 协作方法设计

3.4.2 协作方法设计

基于协作式众包测试的思想，在测试过程中众包工人可以看到他人的众包报告并采取一系列的交互操作，如点赞、点踩、或 Fork 他人的缺陷报告并继续编辑。在初始阶段，平台只按照当前点赞次数排序推荐质量较高的缺陷报告。当众包工人开始编写自己的缺陷报告并按照要求选择相应属性，系统基于缺陷的属性筛选适合的缺陷报告推荐。当工人开始进行具体描述时，系统实时地根据文本相似度进行详细缺陷报告的推荐，并列出当前与其他报告计算后的相似度大小。当相似度大于 95% 时，当前报告无法提交，通过这样的方式降低重复报告数量。系统鼓励众包工人间进行协作，即对他人报告进行评价或继续编辑。通过 Fork 他人报告，以该报告为父节点向下建立子报告，细化其具体内容。其他工人则可以再次 Fork 本报告，最终使得整体报告呈现树状结构。每一棵树代表一类缺陷，越向下是对其越细化的描述。其不同叶子节点也可能是对此问题从不同角度的细化描述，如图3-13所示。

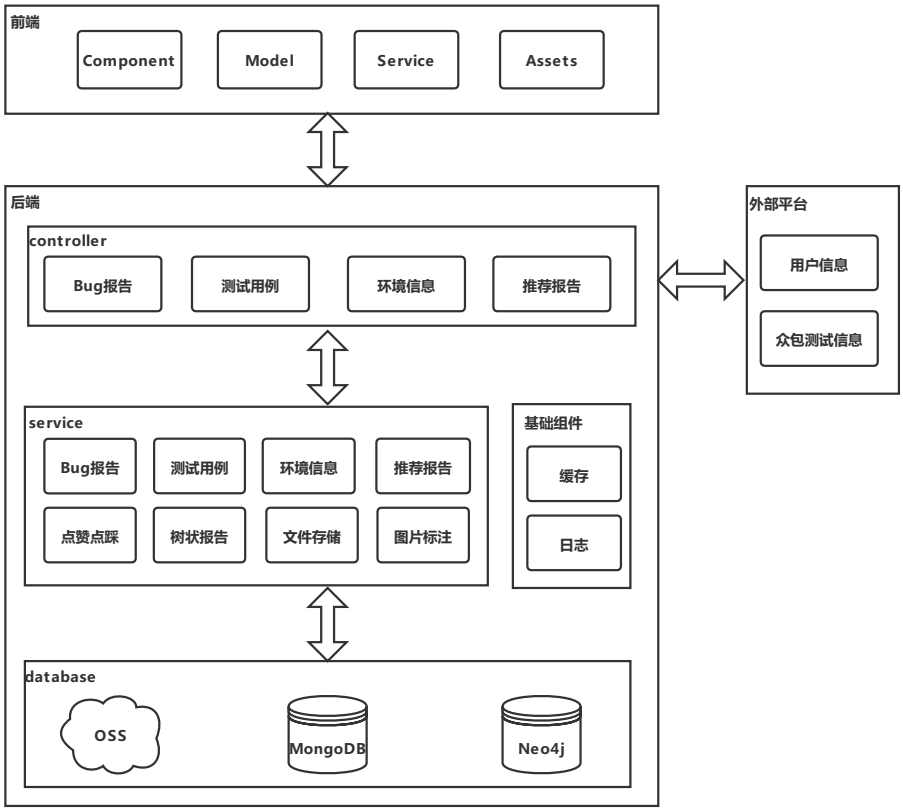


图 3-14: 知识图谱数据获取模块架构设计图

3.5 知识图谱数据获取模块详细设计

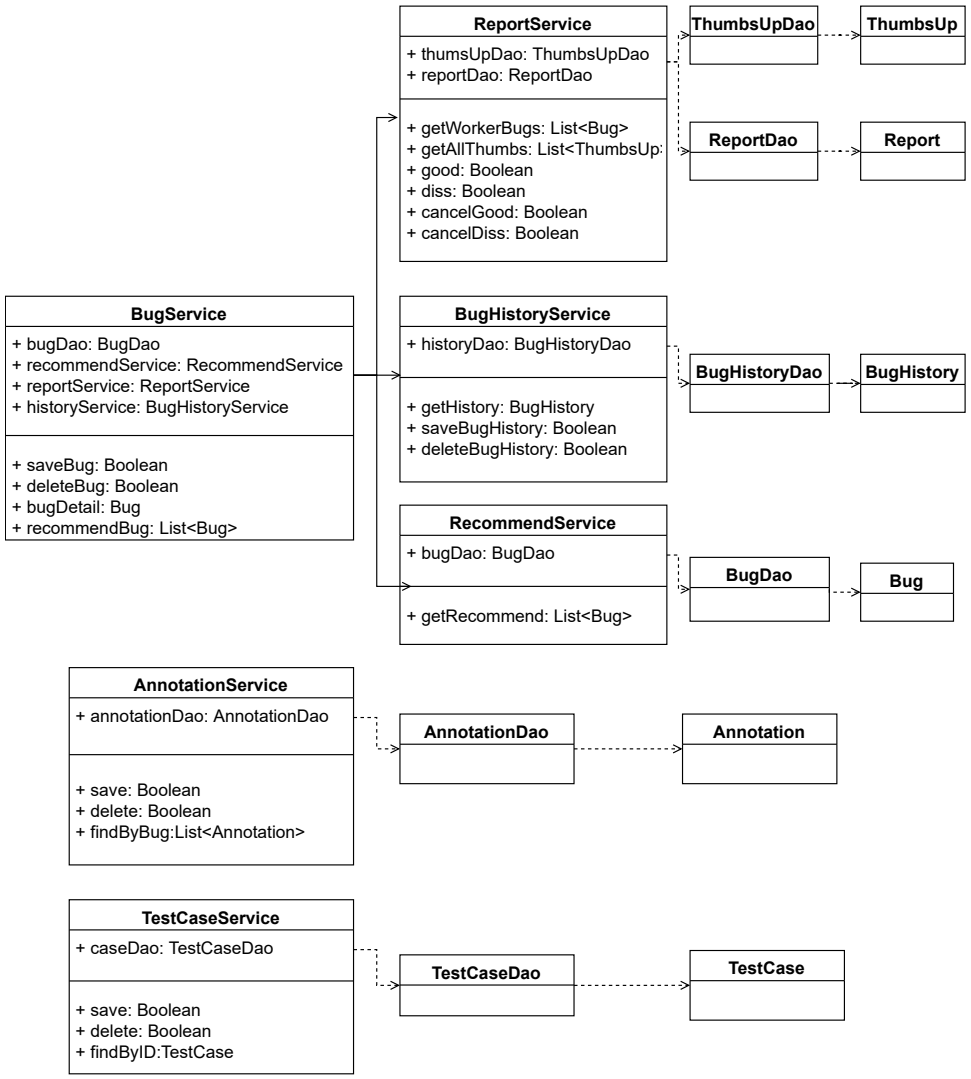


图 3-15: 知识图谱数据获取模块核心类图

3.5.1 架构设计

知识图谱数据获取模块架构设计如图 3-14所示。本模块主要负责实现众包测试报告提交的基本流程，以此获取构建知识图谱所需要的数据。Controller 层既可以获得前端关于报告信息的请求并转发给对应的 Service 服务，还与外部平台交互获得众包工人和众包测试的信息传递给 Service 层。Service 层将报告信

息按照逻辑拆分成对缺陷报告、测试用例、环境信息、推荐报告、点赞点踩、树状报告、文件存储、图片标注相关的处理，分别通过操纵底层的 MongoDB 表结构完成数据操作。对于图片信息的存储放入 OSS 云中，在 MongoDB 中只保存对应的 URL 地址。当缺陷报告新增或被点赞点踩时，还需要在 Neo4j 图数据库中增加相应的实体和关系。为了提高热点数据的访问速度，在系统中加入 Redis 缓存；为了对日志信息进行采集，计划接入独立的日志服务。

3.5.2 核心类设计

知识图谱数据获取模块核心类图如图 3-15所示。其中 BugServicea 负责缺陷报告的编辑与查看以及获得推荐报告。它依赖于 ReportService 获得点赞点踩信息，BugHistoryService 获得缺陷报告的父子报告信息，以及 Recommend-Service 获得推荐报告信息。AnnotationService 负责对图片标注信息的存储，TestCaseService 获得测试用例的保存与编辑。每个 Service 依赖于对应的 DAO 层和数据实体与 MongoDB 交互。

3.6 知识图谱特征学习模块详细设计

3.6.1 架构设计

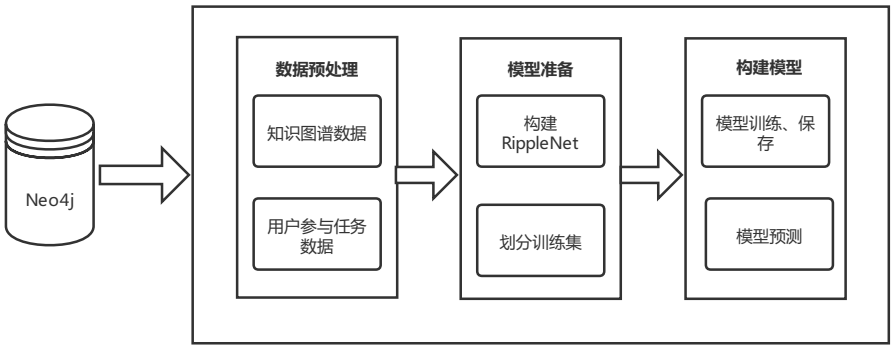


图 3-16: 知识图谱特征学习模块架构设计图

知识图谱特征学习模块架构设计如图 3-16所示。本模块主要负责利用图数据库中数据构建知识图谱，训练模型并得到任务分配结果。其依赖外部 Neo4j 数据库数据进行数据预处理，得到文本化的知识图谱数据和用户历史参与任务

数据，以知识图谱数据为基础构建 RippleNet 网络，以用户历史参与任务数据为基础划分训练集、验证集及测试集。将这两部分数据输入模型中通过梯度下降迭代训练模型并保存。之后便可以使用保存好的模型得到预测众包工人对指定任务的偏好，根据偏好和外部规约得到最终任务分配结果。

3.6.2 核心类设计

图 3-17: 知识图谱特征学习模块核心类图

知识图谱特征学习模块核心类图如图 3-17所示。add _data 在众包工人提

交缺陷报告或对缺陷进行点赞点踩时被调用，实时的向图数据库中增添记录。main() 用于训练并保存模型。它先调用 preprocess.py 对知识图谱数据和历史交互数据保存成文本并处理，接着调用 data _loader.py 划分数据集，构建 RippleNet 网络，最终调用 train.py 训练并保存模型，它依赖于 model.py 中保存的模型架构。在分配测试任务和审核任务时分别调用 rec _page 和 rec _bug 文件，经过数据处理并获得 RippleNet 结构后根据保存好的模型权重获得潜在偏好值，经过筛选后得到最终结果。

3.7 知识图谱任务分配模块详细设计

3.7.1 架构设计

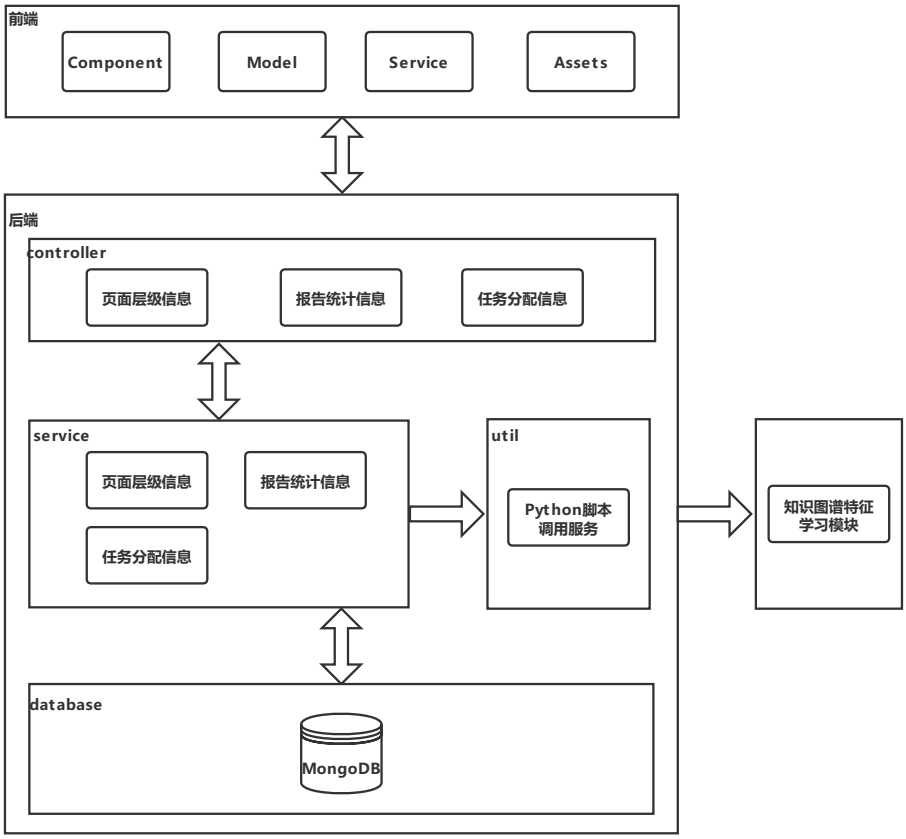


图 3-18: 知识图谱任务分配模块架构设计图

知识图谱任务分配模块架构设计如图 3-18所示。本模块对知识图谱特

征学习模块分配结果进行展示并记录众包工人执行分配任务情况。后端通过 MongoDB 获得众包三级页面任务信息及众包工人总体和个人在各个页面找到缺陷的个数。同时，后端和知识图谱任务分配模块交互获得任务分配结果，返回给前端进行可视化展示。在任务分配后，记录众包工人是否按照分配结果完成众包任务，并将结果记录在数据库中，在之后对任务分配算法进行评估。

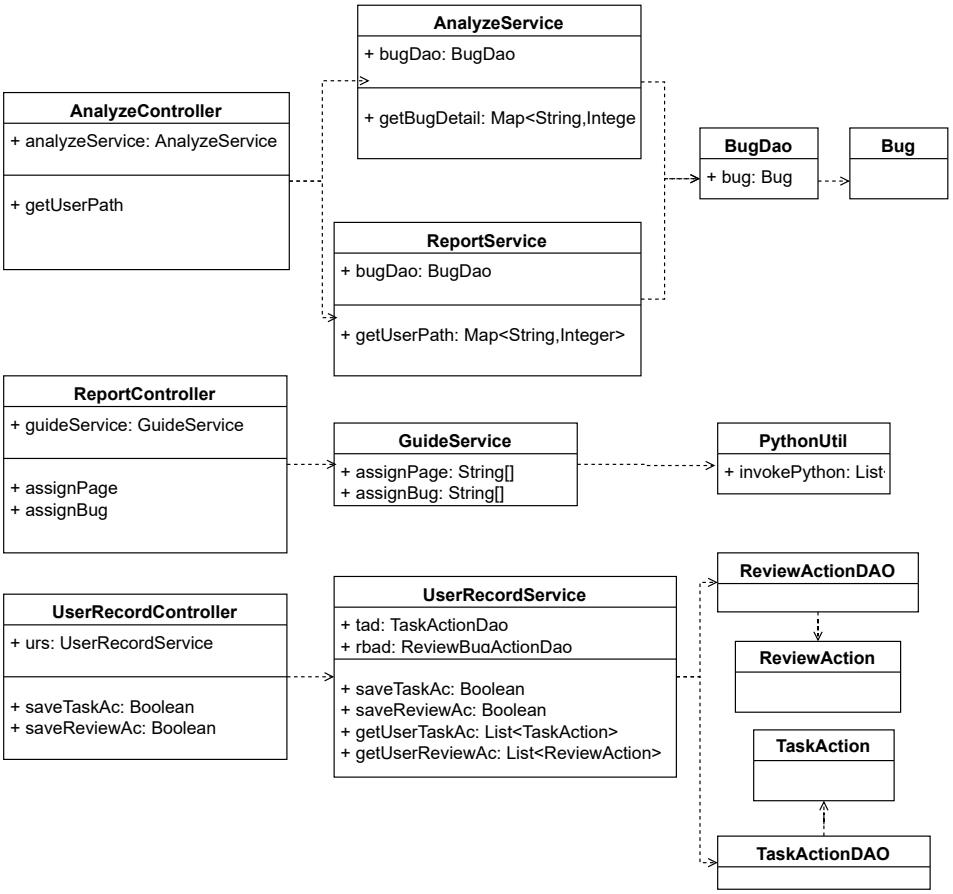


图 3-19: 知识图谱任务分配模块核心类图

3.7.2 核心类设计

知识图谱任务分配模块核心类图如图 3-19所示。AnalyzeController 的 getUserPath() 获得当前三级页面众包任务执行情况。其依赖的 getBugDetail() 获得所有众包工人对不同任务的执行情况，getUserPath() 获得当前众包工人对不同任务的执行情况。getUserPath() 将结果整合为 JSON 形式返回给前端。ReportController 接收前端对于测试任务分配和审核任务分配的请求，转发给

GuideService 的对应方法，通过 PythonUtil 的 invokePython 调用知识图谱特征学习模块代码获得分配结果。UserRecordController 负责记录众包工人对分配结果的执行情况，将结果封装为 ReviewAction 和 TaskAction 实体，进行存储。

3.8 本章小结

本章从系统需求的角度出发，将系统分成了知识图谱数据获取、知识图谱特征学习、知识图谱任务分配三大模块。并使用用例图和用例描述对各个模块进行了详细分析，对知识图谱的本体库进行了构建。接着使用架构图对系统总体架构进行了阐述，使用“4+1 视图”从多个角度对系统架构进行设计，使用类图对系统主要实体类进行了阐述。为后文具体实现打下基础。

第四章　基于知识图谱的众测任务分配技术实现

本章会在第三章详细设计的基础上完成系统的具体实现。通过系统真实页面截图和关键代码讲解，阐述各个模块的构建方式和最终实现效果。

4.1　知识图谱数据获取模块的实现

在 3.1.1 小节中阐述了本模块需要完成的功能和相关用例。众包工人在参与众包任务后，通过 URL 跳转到众包测试平台，填写测试的基本信息，包括测试报告名称、测试设备名称和品牌，以及操作系统信息，如图 4-1所示。

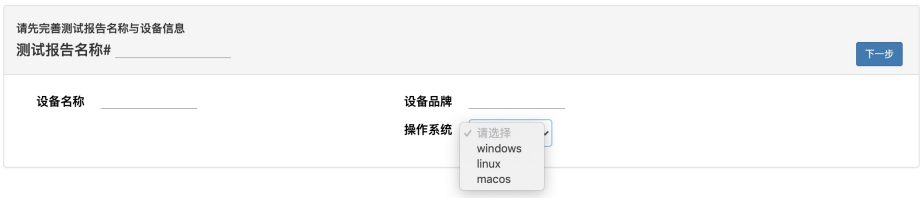


图 4-1: 测试报告基本信息界面

填写完整后点击下一步，会出现整体报告提交界面，主要有三个部分：测试报告基本信息、测试用例填写，以及缺陷填写部分。在测试用例填写和缺陷填写部分右侧分别点击” 创建用例”、“创建 Bug”按钮即可展开用例和缺陷的编辑界面，整体如图 4-2所示。

测试报告基本信息、测试用例、缺陷报告信息填写均通过用户编辑后提交触发前端 Angular 服务，调用后端接口并保存至对应的 MongoDB 数据库中，在此不做赘述。

缺陷填写部分除了缺陷描述信息填写区域外，还包括左上角“请输入 Bug 唯一标识 Id 一键 Fork”的信息框，在 Fork 报告后会自动填充原报告 ID，也可以通过手动输入报告 ID 来进行 Fork 操作，实现协作式众包；左下角上传截图部

图 4-2: 报告提交整体页面

分，支持多截图上传和标注；缺陷推荐列表，根据当前报告与其他提交缺陷的相似度推荐并排序展示。现对这三部分的详细设计实现分别进行阐述。

4.1.1 协作缺陷报告部分设计与实现

当众包工人在编辑缺陷报告时点击推荐列表中的一份推荐缺陷报告时，右侧会显示推荐缺陷报告的详细信息，如图 4-3所示。左上方为对该缺陷报告执行点赞、点踩或 Fork 操作，左下方为该报告的属性及描述信息，右上方为该报告的截图，可以点击查看大图，右下方为该报告的 Fork 情况，通过树的形式可视化显示，其中的每个节点都可以直接点击查看其对应的缺陷报告。以图 4-3为例，该报告没有父报告，有三个子报告。若众包工人点击”Fork” 按钮，则原报告的 Fork 树中儿子节点增加 1，而新报告的父节点则设置为被 Fork 报告节点，同时自动将父报告的属性和描述信息同步到新报告中，方便众包工人在此基础上直接进行修改。

图 4-3: 查看并 Fork 推荐报告

当某份推荐缺陷报告被点击时，前端服务向对应的后端接口发送请求，后端接口调用对应的服务，主要包括以下三部分：获取缺陷的属性及描述信息；获取众包工人对该报告的点赞点踩信息；获取该报告的 Fork 树信息并可视化展示。后端将这三部分结果整合成与前端约定好的大 JSON 对象，由前端再进行拆分展示，具体过程如图 4-4所示。

缺陷报告的 Fork 树信息在后端是通过 BugHistory 类来存储的。每个 BugHosstory 记录对应一个缺陷，记录了这个缺陷的父报告 ID，子报告 ID

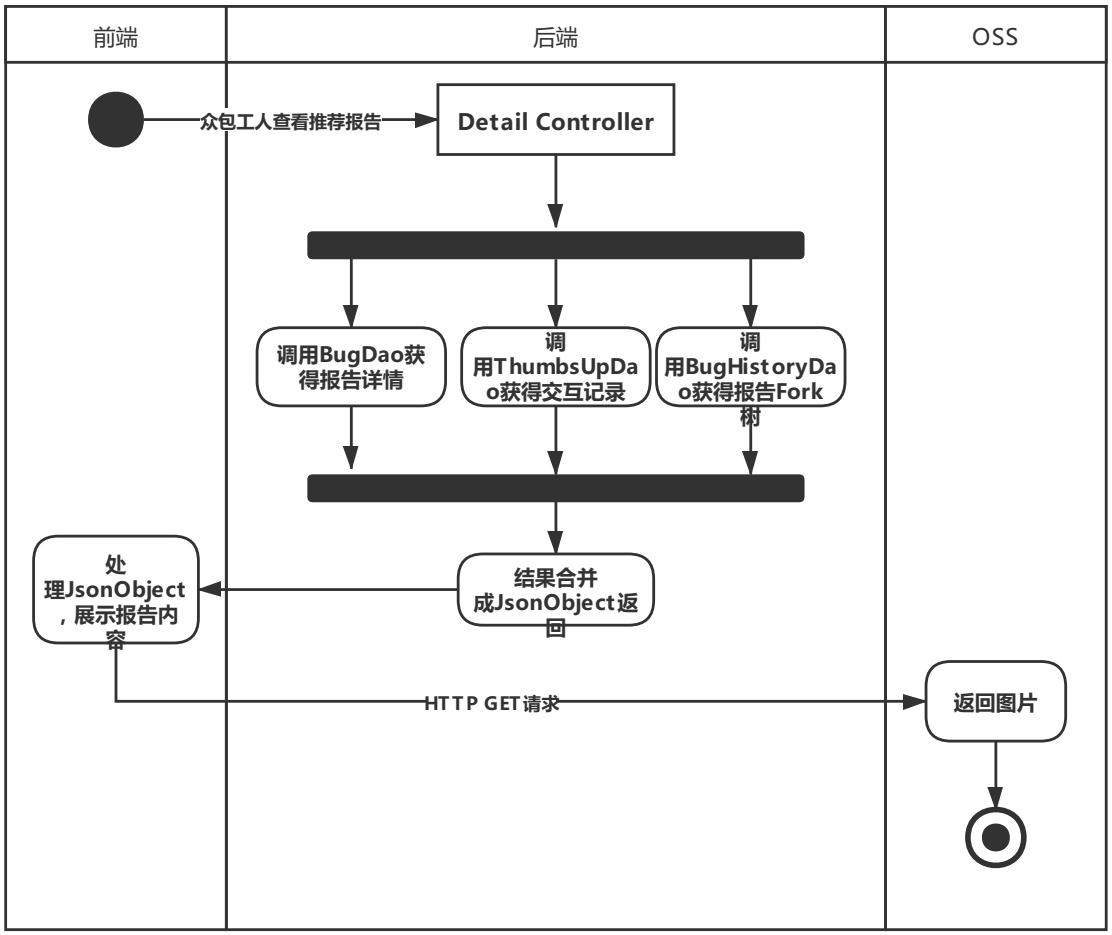


图 4-4: 推荐报告详情活动图

数组，以及报告树的根报告 ID。当众包工人选择某报告时，直接通过 BugHistory 类获得其根报告 ID，然后从根报告开始使用深度优先的方法递归向下遍历获得 Fork 树信息，具体代码如图 4-5所示。GetDepth() 方法先获得报告的根节点信息，再调用 dfs() 深度优先遍历，最终返回 List<List<String» 集合，表示从根节点到叶子节点的所有可能路径。前端使用 Echarts 中的树形结构图将集合转化为树状结构。

当众包工人点击“Fork”按钮时，前端通过属性绑定的方式将原报告属性和描述信息直接复制到当前报告信息中，并记录原报告的 ID，不需要调用后端的接口。当此报告修改提交后再保存新的报告内容和被 Fork 的报告 ID。

当众包工人进行点赞或点踩某份缺陷报告时，会在该报告对应的 ThumbsUp 表中记录当前缺陷报告 ID 信息及对应操作。当再次查看该报告时，会查询是否有点赞点踩记录并在前端有不同的显示，众包工人也可取消点赞或点踩，

在后端删除对应的记录。

```
public List<List<String>> getDepth(String id) {
    BugHistory bugHistory = historydao.findByid(id);
    List<List<String>> result = new ArrayList<List<String>>();
    if(bugHistory == null) { return result;}
    String rootId = bugHistory.getRoot();
    if(rootId == null||rootId.equals(" ")) {return result;}
    BugHistory root = historydao.findByid(rootId);
    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 4-5: 获得 Fork 树信息代码

4.1.2 截图上传标注部分设计与实现

众包工人在点击“上传文件”按钮后，从本地文件夹中选择图片文件，点击“上传截图”进行上传。众包工人的上传操作触发前端 upload _pic 组件的 uploadImgToOss 方法，调用公共接口将图片保存在外部平台的 OSS 中，系统只需要保存图片的 URL 信息。URL 由四部分组成，分别是 IP、存储目录、当前时间戳以及文件名，例如“http://site.oss.aliyuncs.com/app/1613554405786/下载.png”，通过这样拼接的 URL 可以保证其在外部平台存储时的唯一性。当存储成功后，在报告中显示图片的缩略图，并可以再次点击图片显示大图并对其进行标注，如图 4-7所示。

点击标注时，会先将原报告界面置灰，在页面正中根据上传图片的长和宽

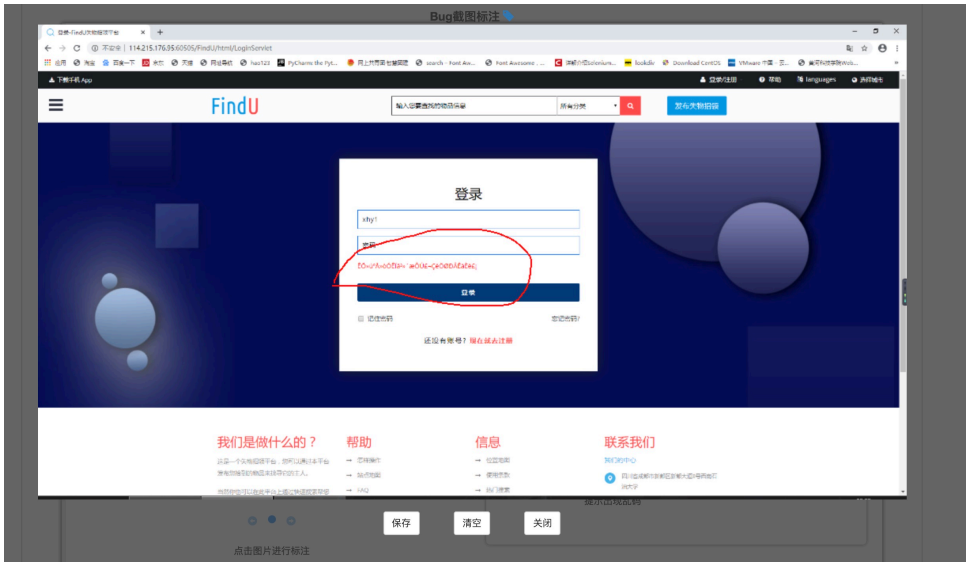


图 4-6: 查看上传图片并标注

创建一个 canvas 对象，并将图片填充在 canvas 画布中。当众包工人在图片上点击鼠标时，会触发 canvas 的 onMouseDown() 方法，从当前坐标位置开始画线，并保存 x,y 坐标信息到对应的数组中，如图 4-7代码所示。这里的坐标信息指的都是当前位置坐标减去 canvas 画布对于父元素的偏移量。当鼠标按住并移动时，onMousemove() 方法同样会保存移动的每个点的坐标信息，并将当前位置与之前的点连接起来，直到鼠标松开，onMouseup() 方法停止记录。通过不断连接鼠标移动过程中的位置点实现对图片标注的轨迹记录。当众包工人保存标注信息时，系统保存坐标数组，并在下次打开时根据位置信息重新形成连线。

4.1.3 缺陷报告推荐部分设计与实现

缺陷报告推荐推荐列表实现逻辑在不同时间有所不同。当用户未编辑缺陷报告时，推荐列表按照当前缺陷报告点赞数量排序，如果数量相同按照编辑时间排序。这样将最典型的缺陷报告排在前面可以对众包工人编辑缺陷报告起到规范指导作用。

当众包工人选择缺陷的三级页面、漏洞分类、严重等级、复现程度后，会按照所选条件筛选缺陷报告进行展示。由于每次条件选择都会从之前筛选过的结果中再次进行筛选，为了保证排序的效率，使用 Redis 缓存上次选择后剩下的缺陷报告结果。当众包工人重新进行选择、提交缺陷报告、放弃编辑此报告或关闭页面时，之前的缓存失效。

```
@HostListener('mousedown', ['$event'])
onMousedown(ev:MouseEvent) {
    this.mouse_be_down = 1;
    if (this.permitEdit) {
    //if (!ev) var ev = new MouseEvent();
        this.the_canvas_context.lineWidth = 1;
        this.the_canvas_context.strokeStyle = "#FF0000";
        this.the_canvas_context.fillStyle = "#FF0000";
        this.the_canvas_context.beginPath();
        this.the_canvas_context.moveTo(ev.pageX-
this.the_canvas.offsetLeft,ev.pageY-this.the_canvas.offsetTop-
$("#canvas_div").offset().top);
        this.xs.push(ev.pageX-this.the_canvas.offsetLeft);
        this.ys.push(ev.pageY-this.the_canvas.offsetTop-
$("#canvas_div").offset().top);
    }
}
```

图 4-7: onMouseDown 方法代码

此处以选择二级页面为例，介绍当众包工人对当前缺陷的基本属性进行选择时，推荐缺陷列表的动态变化过程。如图 4-8所示，当众包工人选择二级页面后，会触发前端 bug _select 组件，通过 HTTP GET 请求调用后端接口，后端 RecommendController 接到请求后，调用对应的 Service 方法 recommendByPage()。在此方法中会先去查看 Redis 缓存是否保存有历史缺陷推荐列表，如果有的话直接在此基础上进行更细粒度的筛选，否则需要通过 BugPageDao 中的方法访问 MongoDB 数据库得到结果。之后将结果记录在 Redis 中以便之后的查询。在返回给前端记录前，Service 方法还会调用排序方法根据相似度对推荐缺陷列表进行排序。Controller 调用对应的模版现实结果。

4.2 知识图谱特征学习模块设计与实现

4.2.1 数据抽取与知识图谱构建

在 3.2.2 小节中已详细描述了根据众包测试领域内知识构建测试任务分配和审核任务分配知识库的过程，在此章节只需要使用数据集实例化其中的实体及属性。在知识图谱数据获取模块中，系统使用 MongoDB 数据库持久化了众包测试相关数据，其中与构建本体相关的实体类如图 4-9所示。考虑到图数据库

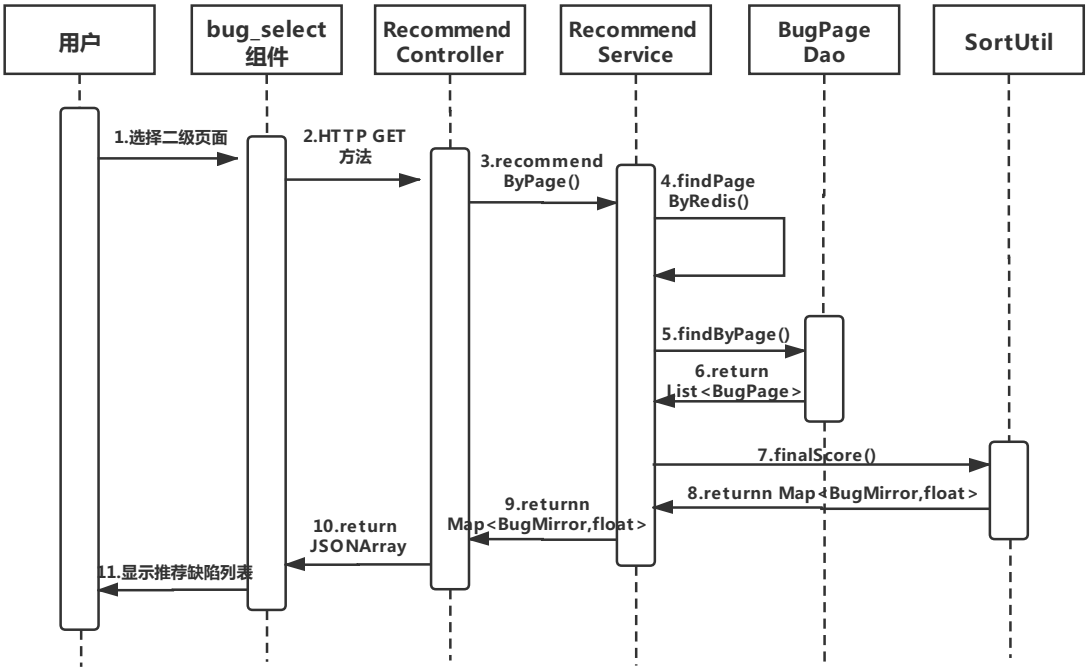


图 4-8: 选择二级页面时序图

可以更好的对知识库中实体和关系进行映射，在系统中选用 Neo4j 图数据库存储处理后的知识数据。

Neo4j 中包含两种类型实体：节点和边。其中节点对应知识图谱中的实体，其包含的属性对应本体库中实体类的数据属性；边对应知识图谱中的关系，也是本体库中实体的对象属性。通过以上方法，我们可以将外部数据库中的数据存储到图数据库中，构建出符合要求的知识图谱。

如图 4-10为使用 Python 脚本从 MongoDB 数据库中获取数据并存储到 Neo4j 数据库中的代码示例。py2neo 包是 Neo4j 数据库的 python 驱动，我们利用它来构建 Cypher 语句对 Neo4j 数据库进行操作。对于从 MongoDB 中获得的数据，先判断其对应的节点在 Neo4j 数据库中是否已创建，如果没有则新创建节点并根据要求赋值属性，之后在两个新创建的节点上创建对应的关系。

经过如上知识抽取过程，最终创建的知识图谱如图 4-11所示。它包含了测试任务分配和审核任务分配相关的各项实体和关系。在之后的过程中需要根据机器学习模型的需求将数据进行处理输入到模型中。

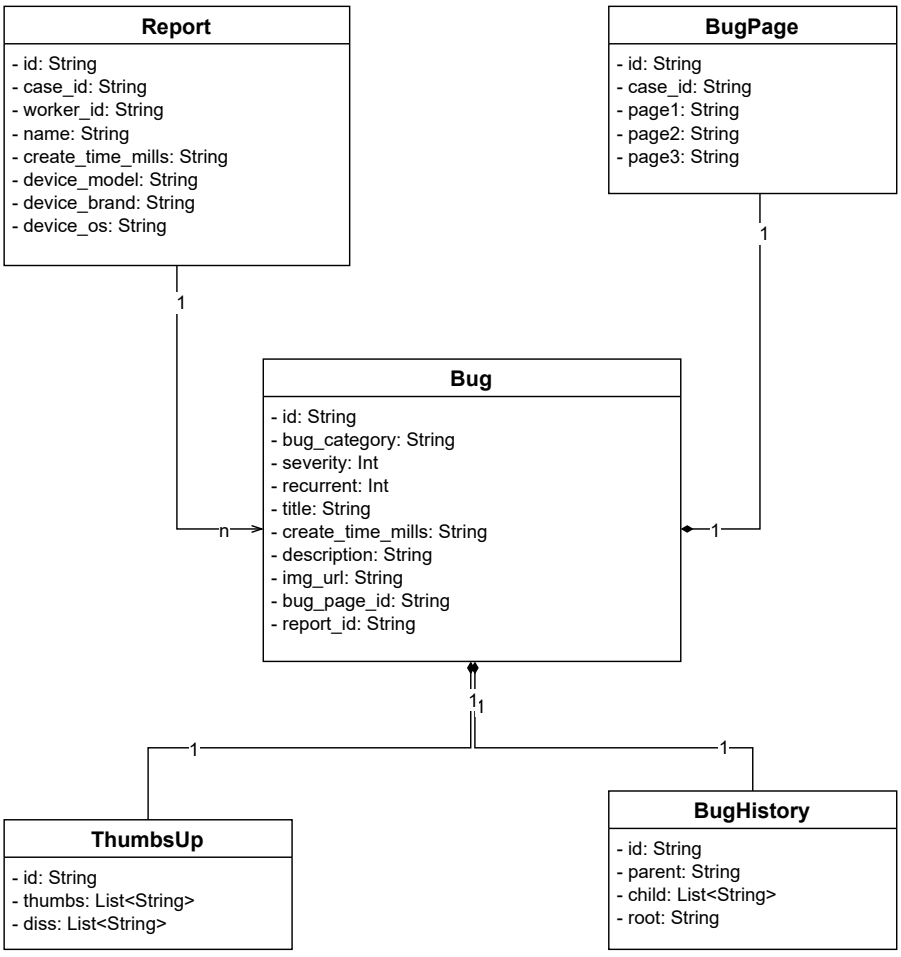


图 4-9: 知识图谱相关实体类图

4.2.2 模型输入

RippleNet 模型的训练数据主要包括:

（1）用户集 U={u1, u2, ...}，项目集 V={v1, v2, ...} 的用户-项目交互矩阵

$$Y = \{y_{uv} \mid u \in U, v \in V\}, \text{其中}$$

$$y_{uv} = \begin{cases} 1, & \text{如果}(u, v) \text{存在交互} \\ 0, & \text{否则} \end{cases} \tag{4-1}$$

（2）知识图谱

$$G = \{(h, r, t) \mid h, t \in E, r \in R\}$$

```
report_list = collection_report.find()
for report in report_list:
    report_id = id_to_ObjectId(report['_id'])
    worker_id = report['worker_id']          # worker 节点
    worker_node = matcher.match('worker').where("_.id='" + str(worker_id) + "'").first()
    if (worker_node == None):
        cursor_worker = collection_student_info.find({'worker_id':str(worker_id)})
        if (cursor_worker.count() > 0):
            worker = cursor_worker[0]
            worker_node = Node('worker', name=worker['name'])
            worker_node['id'] = worker['worker_id']
            graph.create(worker_node)
    bug_by_report = collection_bug.find({'report_id': str(report_id)})
    for bug in bug_by_report:
        bug_node = matcher.match("bug").where("_.id='" + str(bug['_id']) + "'").first()
        if (bug_node == None):
            bug_node = Node('bug', name=bug['title'])
            bug_node['id'] = str(bug['_id'])
            bug_node['create_time'] = millis_trans(str(bug['create_time_millis']))
            # 构造 Bug 节点
            graph.create(bug_node)
            if (worker_node != None):      # bug-worker 关系
                bug_to_worker = Relationship(bug_node, 'ownedBy', worker_node)
                worker_to_bug = Relationship(worker_node, 'hasBug', bug_node)
            graph.create(bug_to_worker)
            graph.create(worker_to_bug)
```

图 4-10: 知识图谱构建关键代码

希望学习到预测函数：

$$\hat{y}_{uv} = F(u, v; \Theta), \quad \hat{y}_{uv} \text{ 表示用户} u \text{ 点击项目} v \text{ 的可能性}$$

我们对训练数据的处理需要经过图 4-12所示流程，现对其进行详细阐述。

为了将训练数据输入模型，需要对数据进行预处理，得到 kg _final 和 rating _final 文件。以审核任务分配为例，其 rating _final 包含两列数据，分别是 worker _id 及其历史点赞点踩的缺陷报告 bug _id，表示众包工人历史对以上报告产生了交互操作。此处需要注意的是，这里的 bug _id 必须是在知识图谱文件 kg _final 中出现过的 bug _id，这是 RippleNet 模型训练的前提。

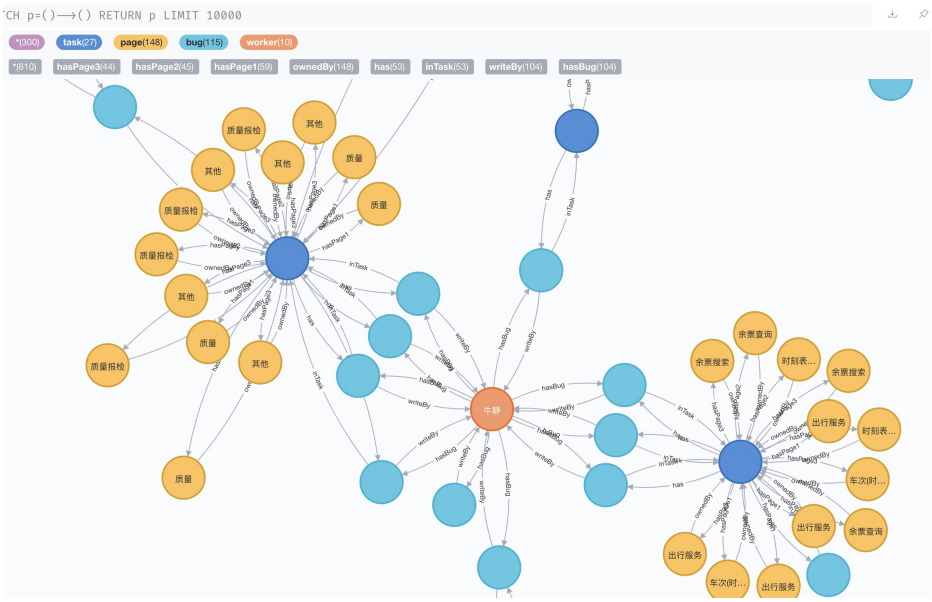


图 4-11: 某众包工人提交缺陷报告相关知识图谱截图

kg _final 包含三列，分别代表三元组关系的头节点，关系，及尾节点。在这里分别是 bug _id, 关系 id 和 bug _id，这部分数据需要对知识图谱数据进行处理。根据表 3-5中定义的 bug 实体类的对象属性，可以对审核任务分配知识图谱关系定义如表 4-1:

表 4-1: kg _final 表关系定义

关系 ID	关系	备注
1	bug.hasType.bug	两个缺陷属于同一个漏洞分类
2	bug.inTask.bug	两个缺陷属于同一个任务
3	bug.writeBy.bug	两个缺陷被同一个众包工人提交
4	bug.hasParent.bug	两个缺陷有同样的父报告
5	bug.thumbs.bug	两个缺陷被同一个众包工人点赞点踩

对于每一种关系，在代码中均需要调用 match _entity() 方法，如图 4-13所示。如“bug.writeBy.bug”关系，其对应的知识图谱三元组为”bug-writeBy-worker”，关系 ID 为 3，因此传入头节点 bug，关系 writeBy，尾节点 worker，对应的图引擎，关系编号 3，以及需要写入的文件路径。在代码中使用 Match 语句找到被同一个众包工人提交的缺陷报告，并将两者对应的 bug _id 及关系编号写入 csv 文件中。

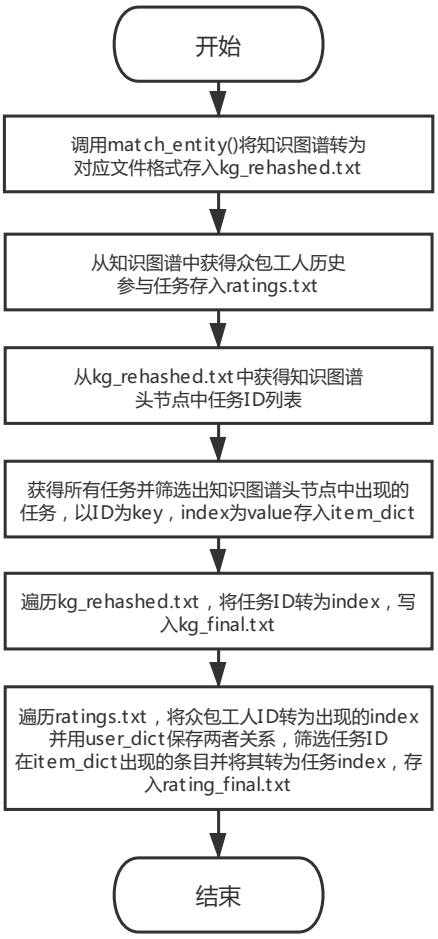


图 4-12: 知识图谱训练输入流程图

此处我们已经得到了满足格式要求的 RippleNet 模型输入，但在 RippleNet 中需要对 user 和 item 进行嵌入层操作，将数据转换为向量，要求我们输入的 user _id 和 item _id 应该小于等于嵌入维度，也就是他们的个数，因此我们需要对输入的 ID 进行转换。通过 Python 中的 dict 保存原始 ID 和新 ID 的关系，之后将新 ID 输入最终的 kg _final 和 rating _final 文件。

至此，所有数据预处理操作均已完成，可以直接将数据集交给 RippleNet 模型进行训练。

4.2.3 特征学习

RippleNet 模型在第二章中有所介绍，算法 4.1描述了其基本架构。在实际训练时，我们首先将数据集分成训练集、验证集和测试集，对训练集的数据采

```
def match_entity(head,relation,tail,graph,relationNo,file):
    result = graph.run("MATCH (b1:"+head+")-[:"+relation+"]->(:"+tail+")<-[:"+relation+"]-
(b2:"+head+")RETURN b1.id, b2.id")
    data = []
    for r in result:
        str = r["b1.id"]+" "+relationNo+" "+r["b2.id"]
        data.append(str)
    append_csv(file,data)

def append_csv(path,data):
    with open(path, "a+", newline='') as file:
        csv_file = csv.writer(file)
        csv_file.writerows(data)
```

图 4-13: 构建知识图谱输入代码

算法 4.1 RippleNet 基本算法

Input: 用户历史交互矩阵 Y，知识图谱 G
Output: 用户 u 选择 item v 的概率

1: 初始化所有参数
2: 对每一个用户 u，计算其 H-hop 波纹集合 $\left\{\mathcal{S}_u^k\right\}_{k=1}^H$
3: **for** i in training iteration **do**
4: 从用户历史交互矩阵 Y 中取样组成 mini-batch
5: 从知识图谱 G 中取样 mini-batch
6: 在 mini-batch 上通过反向传播的方式计算梯度 $\partial\mathcal{L}/\partial\mathbf{V}$,$\partial\mathcal{L}/\partial\mathbf{E}$,$\{\partial\mathcal{L}/\partial\mathbf{R}\}_{r\in\mathcal{R}}$, 和 $\{\partial\mathcal{L}/\partial\alpha_i\}_{i=1}^{\vec{H}}$, 其中 V 和 E 分别是所有项和实体的嵌入矩阵，R 是关系 r 的嵌入矩阵
7: 根据学习率采用梯度下降的方法更新 $\mathbf{V},\mathbf{E},\{\mathbf{R}\}_{\mathbf{r}\in\mathcal{R}}$ 和 $\{\alpha_i\}_{i=1}^H$
8: **end for**

用随机梯度下降算法来迭代地优化损失函数。在每次训练迭代中，使用负采样策略随机从用户历史交互矩阵 Y 和知识图谱 G 中采样组成小批量 minibatch，计算损失 L 相对于模型参数 Θ 的梯度，并基于采样的小批量 minibatch 通过反向传播的方法更新所有参数完成迭代。

如图 4-14为 RippleNet 前向传播的代码。它将实体嵌入矩阵中的索引为样本变量的项取出赋值给样本嵌入矩阵。将空列表赋值给头部实体、关系、尾部实体三个嵌入，随即循环总跳数：在每次循环中分别去实体嵌入矩阵和关系嵌入矩阵中寻找索引为记忆实体和记忆关系列表中该跳的内容对应的矩阵添加到实体嵌入和关系嵌入列表。然后调用 key _addressing() 方法得到总跳数的响应，再调用预测函数算出模型的分数，最终进行 sigmod 归一化。

在训练 RippleNet 模型的时机上，我们也需要结合领域特征进行考虑。对

```
def forward(
    self,
    items: torch.LongTensor,
    labels: torch.LongTensor,
    memories_h: list,
    memories_r: list,
    memories_t: list,
):
    item_embeddings = self.entity_emb(items)    # [batch size, dim]
    h_emb_list = []
    r_emb_list = []
    t_emb_list = []
    for i in range(self.n_hop):
        # [batch size, n_memory, dim]
        h_emb_list.append(self.entity_emb(memories_h[i]))
        r_emb_list.append(          # [batch size, n_memory, dim, dim]
            self.relation_emb(memories_r[i]).view(
                -1, self.n_memory, self.dim, self.dim
            )
        )
        # [batch size, n_memory, dim]
        t_emb_list.append(self.entity_emb(memories_t[i]))
    o_list, item_embeddings = self._key_addressing(
        h_emb_list, r_emb_list, t_emb_list, item_embeddings
    )
    scores = self.predict(item_embeddings, o_list)
    return_dict = self._compute_loss(
        scores, labels, h_emb_list, t_emb_list, r_emb_list
    )
    return_dict["scores"] = scores
    return return_dict
```

图 4-14: Ripplenet 前向传播代码

于测试任务分配，其分配的主体为三级页面测试任务，在整场众包测试过程中其本体库中的要素均未发生变化。因此只要在发布众包测试且未开始前训练好模型就可以在整场考试中使用并得到测试任务分配结果。在众包测试过程中只需要动态增加众包工人交互记录，即众包工人参与某项三级页面任务的记录，对模型本身架构没有影响。

对于审核任务分配，其分配的主体为待审核的缺陷报告，随着众包测试的进行其数量在不断变化，知识图谱的结构也需要随之调整。为了能正常使用知识图谱训练模型并基于模型给出审核任务分配结果，我们对整场众包测试的时

```
@Override
public void configureTasks(ScheduledTaskRegistrar taskRegistrar) {
    this.taskRegistrar = taskRegistrar;
}
private Set<ScheduledFuture<?>> getScheduledFutures() {
    if (scheduledFutures == null) {
        try {
            // spring 版本不同选用不同字段 scheduledFutures
            scheduledFutures = (Set<ScheduledFuture<?>>) BeanUtils.getProperty(taskRegistrar,
"scheduledTasks");
        } catch (NoSuchFieldException e) {
            throw new SchedulingException("not found scheduledFutures field.");
        }
    }
    return scheduledFutures;
}

//添加任务
public void addTriggerTask(String taskId, TriggerTask triggerTask) {
    if (taskFutures.containsKey(taskId)) {
        throw new SchedulingException("the taskId[" + taskId + "] was added.");
    }
    TaskScheduler scheduler = taskRegistrar.getScheduler();
    ScheduledFuture<?> future = scheduler.schedule(triggerTask.getRunnable(),
triggerTask.getTrigger());
    getScheduledFutures().add(future);
    taskFutures.put(taskId, future);
}
```

图 4-15: 定时任务关键代码

间进行划分，默认每 20 分钟（作为参数可调整）更新一次模型并基于上一次训练的模型分配待审核缺陷报告。这样既利用了模型结果结合众包工人和缺陷报告特征合理进行任务分配，也确保了所有缺陷报告都在模型的候选集中不被遗漏。具体代码如图 4-15所示。

此更新任务由 Java 中的 ScheduledTask 定时器实现，它基于 SchedulingConfigurer 接口重写了 configureTasks() 方法，将新增的定时任务内容和触发时间封装成 ScheduledFuture 形式并保存在 scheduledFutures 中。同时通过 Map 形式的 taskFutures 将任务 ID 和任务关联起来。

4.2.4 任务分配结果输出

通过训练好的模型，我们可以获得众包工人对给定页面集合 V 中每一个页面的潜在偏好分数，从而根据偏好的高低决定其测试任务的分配。但在输出最终分配页面时，我们还需要考虑页面覆盖率的因素。

假设众包测试中所有任务集合为 target，它实现了对测试页面的全覆盖。所有众包工人已完成的测试任务集合为 all，当前工人完成的报告为 my。其中，my ⊆ all ⊆ target，且 R1=target-all，R2=all-my，如图 4-16所示。

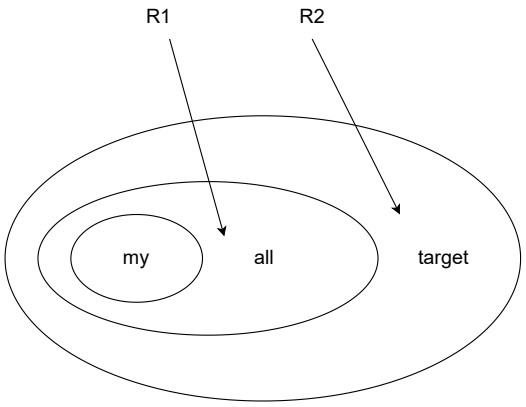


图 4-16: 页面覆盖示意图

```
def split_page(worker_id,case_id):
    # 连接 neo4j 数据库
    graph = Graph('http://localhost:7474', username='neo4j', password='123456')
    cyper = "MATCH (t:task{case_id:'"+case_id+"'}) RETURN id(t)"
    task_list = graph.run(cyper)
    task_list_set = set(task_list)
    cyper = "MATCH (t:task{case_id:'" + case_id + "'})-[:has]->(:bug)<-[:hasBug]-
(w:worker) return distinct id(t)"
    finish_task = graph.run(cyper)
    finish_task_set = set(finish_task)
    left_task = task_list_set-finish_task_set
    if(len(left_task)>0):
        return left_task
    cyper = "MATCH (t:task{case_id:'" + case_id + "'})-[:has]->(:bug)<-[:hasBug]-
(w:worker{id:'"+worker_id+"'}) return id(t)"
    my_task = graph.run(cyper)
    my_task_set = set(my_task)
    rec_task = task_list_set-my_task_set
    return rec_task
```

图 4-17: 筛选页面集合代码

据此，我们需要对页面集合 V 上根据不同情况做一步筛选操作。当所有众包工人尚未实现测试页面的全覆盖，即 $R1 \neq \emptyset$ 时，$V = R1$；当 $R1 = \emptyset \&\& R2 \neq \emptyset$ 时，V=R2; 当 $R1 = R2 = \emptyset$ 时，完成所有测试任务分配，具体代码如图 4-17所示。通过此种方式，我们利用群体智能引导众包工人首先完成群体测试路径的

全覆盖，再完成个人测试路径的全覆盖，可以有效提高页面覆盖率。

对于审核任务的分配，同样需要考虑缺陷的点赞点踩次数，防止长尾效应导致部分缺陷报告没有任何点赞点踩操作，影响后续对缺陷报告的审核评分。因此在选择输入待分配缺陷报告集合时，同样需要经过预处理过程，首先筛选出非本人编写的缺陷报告，按照点赞点踩次数进行排序获得互动次数最少的 20 份报告。将这 20 份缺陷报告作为模型的输入获得众包工人对其的潜在偏好，选择偏好值最高的前 5 份缺陷报告作为审核任务进行分配。

4.3 知识图谱任务分配模块设计与实现

本模块主要是根据众包工人历史完成众测任务情况和知识图谱保存的众测领域内知识，为众包工人个性化分配众包测试任务及缺陷审核任务。帮助其高效参与完成任务，同时减少系统内少重复缺陷报告数量，提高报告质量，利于最终可交付报告的生成。同时在众包工人参与一次众包测试时为其可视化显示当前各三级页面下任务完成情况，帮助众包工人了解当前测试的总体页面覆盖情况与个人完成情况。

4.3.1 分配可视化的实现

为了能准确清晰描述出三级页面的层级信息，在系统前端使用 Echarts 中的树状图来实现。Ecahrts 中的数据和配置是调用 setOption(option) 方法通过配置项的形式传入的，具有动态渲染的优点。具体来说就是当数据发生变化需要重新调用 setOption(option) 方法时，会自动比对两次的数据，只在不同的部分进行渲染。而在 Echarts 中，树状结构想比于其他普通的图结构，也不需要指定各个节点在图中的位置信息，由 Echarts 自动计算得到。

在系统中生成可视化报告树的基本流程如图 4-18所示。大致包含以下几个基本步骤。

（1）获得页面层级信息结构的嵌套 JSON 数据。首先需要判断前端是否已经存储了该数据，如果没有则需要从 OSS 中获得如表所示的 EXCEL 文件，并调用第三方库将 xlsx 格式转为 Echarts 输入的 JSON 格式，进行第一次渲染。

（2）前端调用后端获得页面覆盖情况的接口，获得这次众包测试中所有众包工人和当前众包工人完成众测任务的情况，前端据此用颜色区分自己执行过的三级页面，并设置当鼠标悬停时，显示在该页面下当前用户发现的缺陷数

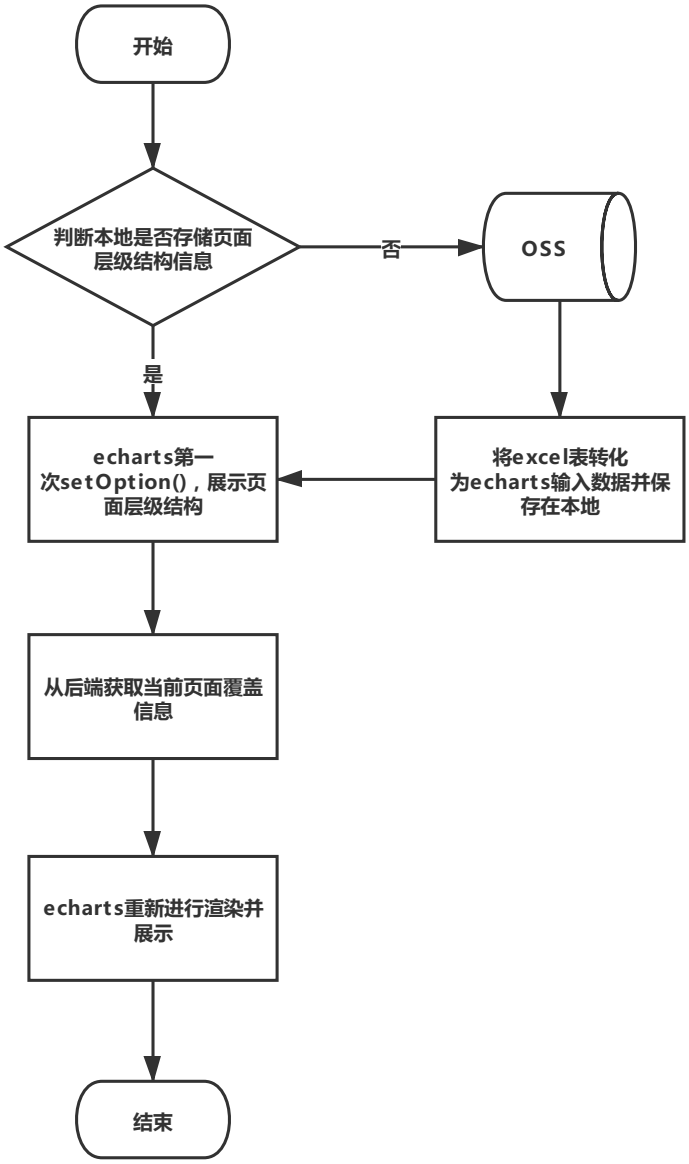


图 4-18: 生成可视化报告树流程图

与该页面下发现的缺陷总数的百分比。后端返回的 JSON 数据形式如下。

{”all”:{” 注册”:1,” 邮箱登录”:3,” 校园卡”:4,” 身份证”:25,” 其他”:5},”self”:{” 注册”:1,” 邮箱登录”:1,” 身份证”:2}}

4.3.2 分配结果记录

为了记录众包工人对于任务分配结果的响应，计算召回率以对分配过程进行评估并改进，其流程如图 4-19所示。

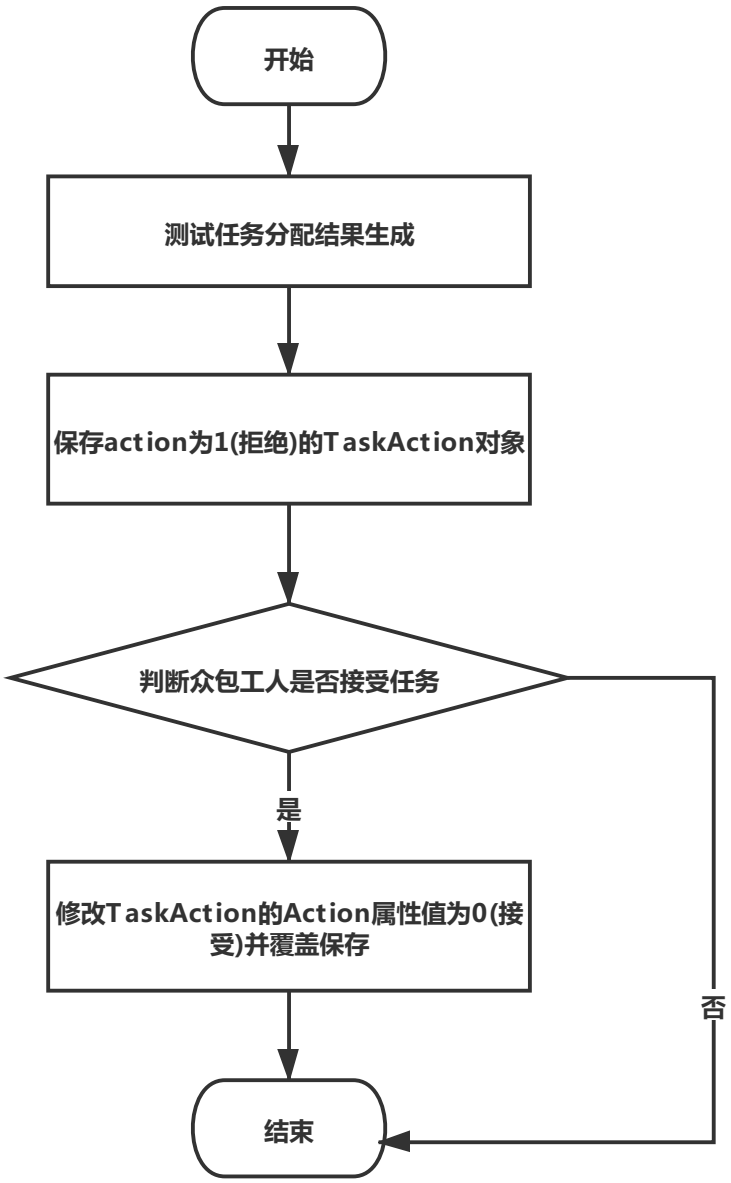


图 4-19: 分配结果记录流程图

测试任务分配和审核任务分配的记录过程类似，以测试任务分配为例，当生成测试任务分配结果时，将结果返回给前端显示的同时会在后端调用 UserRecordService 的 saveTaskAc() 方法。在这个方法中构建 TaskAction 对象，其 action 属性默认为“1 拒绝”，并以此为参数调用 TaskActionDao 的 save() 方法持久化存储。当众包工人选择接受任务时，再由前端调用 UserRecordController 的 saveTaskAc() 方法，此时参数 action 值为“0 接受”，并传递给 UserRecordService。其中 saveTaskAc() 方法检查到已存在对应记录，对其修改后覆盖存储进

MongoDB 数据库中。通过此种方式防止当众包工人直接关闭任务分配模态框时行为数据无法记录的问题。

最终，我们可以利用 TaskActionDao 中的 getByAction 方法获得一场众包测试中的测试任务分配召回率。如果 action 为 0(接受) 的记录数为 AccAction，action 为 1(拒绝) 的记录数为 RefAction，那么测试任务分配的召回率为：

$$S_1 = \frac{\sum AccAction}{\sum RefAction + \sum AccAction} \tag{4-2}$$

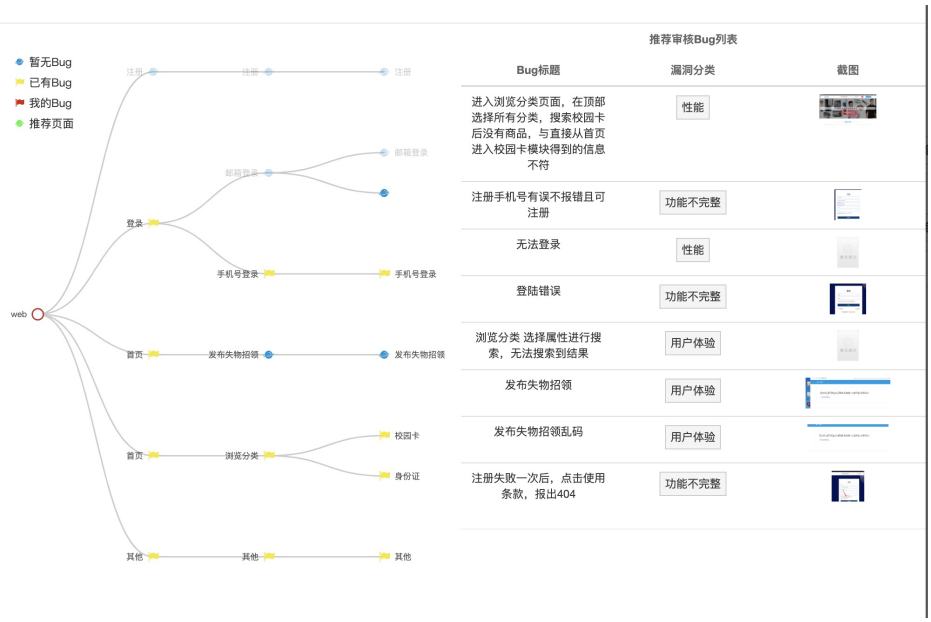


图 4-20: 任务分配截图

4.3.3 模块实现效果

最终任务分配模块实现如图 4-20所示。该页面会在众包工人首次进入众包测试报告提交界面或提缺陷报告后弹出，也可由众包工人自己点击“推荐任务”按钮后弹出。

在任务分配界面中，左侧是测试任务分配结果。用树状结构展示了不同三级页面的任务，并用蓝、黄、红、绿颜色分别标注了暂无缺陷报告、已有缺陷报告、我的缺陷报告的页面节点，并用图例佐以说明。在右侧为审核任务分配

列表，与填写缺陷报告时的相似缺陷报告推荐列表类似，众包工人可以点击该列表中缺陷报告查看详情，并进行点赞点踩操作。

4.4 本章小结

本章利用文字说明、代码示例、时序图、类图、活动图等多种 UML 图结构，以及系统截图的方式对系统中知识图谱数据获取模块、知识图谱特征学习模块和知识图谱任务分配模块的具体实现进行了详细阐述。为下一章节中对系统各个模块的功能、性能测试验收奠定了基础。

第五章　系统测试与实验评估

根据第三章提出的功能需求和非功能需求，本节对系统的基础功能进行功能测试，对边界情况进行边界测试，对系统关键接口进行压力测试，以保证系统的高可用性。

5.1　系统测试

5.1.1　测试环境

根据图3-11中的物理视图准备如表5-1所示测试环境。系统前端运行Angular2.4 程序，依赖 Node.js8.16.0 和 Npm6.4.1 环境。服务端程序分为三个部分，分别构建成 SpringBoot 容器，Redis 容器和 Python 容器，其中 SpringBoot 容器依赖 JDK1.8 和 Tomcat9.0.2 环境。数据库程序构建 MongoDB 容器和 Neo4j 容器，所有服务分别独立部署在阿里云 ECS（Elastic Computer Service，弹性计算服务）服务器设备上。

除此之外，功能测试和性能测试均在用户计算机上完成。功能测试需要使用 Chrome 浏览器访问系统，判断系统的各项功能是否符合测试用例的要求。性能测试使用 JMeter 程序，模拟高并发情况测试系统是否可用。

5.1.2　功能测试

功能测试，又称黑盒测试，是指在不关心系统内部实现的情况下根据预先提出的各项功能需求编写测试用例，对系统进行逐一验证，比较预期结果与实际结果，判断系统是否达到用户使用的要求。现对系统各个模块编写测试用例进行功能测试。

对于知识图谱数据获取模块，首先需要填写缺陷报告的基础信息并编写测试用例，这部分对应的测试用例如表 5-2所示。经测试实验结果与预期相符。

对于缺陷报告的填写，主要存在两种情况，即众包工人正常填写缺陷报告并提交与众包工人 Fork 相似缺陷报告修改后提交。对于第一种情况，其测试用

表 5-1: 测试环境表

组件	设备名称	运行程序	程序版本信息	环境依赖
前端程序	ECS 服务器 (Ubuntu 18.04)	Angular2	Angular2.4	Node.js 8.16.0 Npm 6.4.1
服务端程序	ECS 服务器 (4G 内存，50M 带宽)	SpringBoot Docker Redis Docker Python Docker	Docker 17.09 SpringBoot 2.0.2 Redis 4.0.7 Python 3.6	JDK 1.8 Tomcat 9.0.2
数据库程序	ECS 服务器 (Debian 8)	MongoDB Docker Neo4j Docker	Docker 17.09 MongoDB 3.2.16 Neo4j 3.5.22	无
功能测试程序	用户计算机 (Mac OS 10.14.6)	Chrome 浏览器	Chrome 浏览器 88.0(64 位)	无
性能测试程序	用户计算机 (Mac OS 10.14.6)	JMeter	JMeter5.1.1	JDK1.8

表 5-2: 填写报告基本信息测试用例

用例编号	TC-RS-1
测试功能	众包工人正常填写报告基本信息及测试用例
前置条件	众包工人报名一场众测并进入测试页面
测试步骤	1. 众包工人填写测试基本信息，包括测试报告名称、设备名称、设备品牌、操作系统 2. 点击下一步按钮 3. 点击创建用例 4. 填写用例名称、前置条件、测试步骤、预期结果 5. 点击保存按钮
预期结果	系统保存用户所填信息，并在页面显示用户提交的测试用例
实际结果	与预期结果相符

例如表 5-3所示。

当众包工人填写的缺陷报告与推荐的缺陷报告描述问题一致，但希望对其进行补充完善时，可以直接 Fork 原报告进行修改，其测试用例如表 5-4所示。

当众包工人提交缺陷报告后，或自行点击“任务推荐”按钮后，系统会自动弹出新的模态框，用树状结构显示当前众包测试完成进度与自己已发现缺陷的页面，并个性化为其分配下一步适合的测试任务与审核任务。众包工人可以接受测试任务并去任务指定的界面发现缺陷，具体测试用例如表 5-5所示。测试结果与预期相符。

审核任务的分配和测试任务分配在同一个界面中显示，因此其测试也依赖于 TC-TA-1 用例的正确执行。表 5-6所示的 TC-TA-2 用例模拟了众包工人根据

表 5-3: 提交缺陷报告测试用例

用例编号	TC-RS-2
测试功能	用户可以正常提交缺陷报告，点赞点踩相似缺陷报告，并在填写缺陷报告时有实时相似缺陷报告推荐
前置条件	TC-RS-1 测试用例正确执行，且系统中已有其他缺陷报告提交
测试步骤	1. 众包工人点击“创建 Bug”按钮 2. 众包工人选择缺陷的三级页面信息、漏洞分类、严重程度、复现程度、所属用例 3. 众包工人填写缺陷的标题和详细描述 4. 众包工人点击某推荐缺陷报告查看详情 5. 众包工人对该报告进行点踩操作 6. 众包工人上传缺陷截图并标注 7. 众包工人点击保存按钮
预期结果	系统在众包工人填写缺陷信息时实时推荐相似缺陷报告 点击推荐缺陷报告时显示报告详情 点踩推荐缺陷报告后其点踩按钮置灰 点击保存后提示保存成功并存储数据
实际结果	与预期结果相符

表 5-4: Fork 报告测试用例

用例编号	TC-RS-3
测试功能	用户可以正常提交缺陷报告，点赞点踩相似缺陷报告，并在填写缺陷报告时有实时相似缺陷报告推荐
前置条件	TC-RS-2 测试用例正确执行，且系统中已有其他缺陷报告报告提交
测试步骤	1. 众包工人点击“创建 Bug”按钮 2. 众包工人选择缺陷的三级页面信息、漏洞分类、严重程度、复现程度、所属用例 3. 众包工人填写缺陷的标题和详细描述 4. 众包工人点击某推荐缺陷报告查看详情 5. 众包工人对该报告进行点踩操作 6. 众包工人上传缺陷截图并标注 7. 众包工人点击保存按钮
预期结果	系统在众包工人填写缺陷信息时实时推荐相似缺陷报告 点击推荐缺陷报告时显示报告详情 点踩推荐缺陷报告后其点踩按钮置灰 点击保存后提示保存成功并存储数据
实际结果	与预期结果相符

审核任务分配结果审核缺陷报告的过程，在实际执行过程中与预期结果一致。

对于知识图谱特征学习模块，其预测结果作为输出交给任务分配模块。因此在任务分配模块的功能测试中已经测试了知识图谱特征学习模块预测用户对物品的潜在偏好的功能。在此处，我们只需要验证其知识图谱构建和模型训练

表 5-5: 测试任务分配测试用例

用例编号	TC-TA-1
测试功能	树状结构展示当前总体和个人测试情况，并分配之后的众包测试任务
前置条件	TC-RS-3 测试用例正确执行，系统弹出任务分配模态框
正常流程	1. 查看总体测试情况 2. 鼠标放在树状图节点上查看不同页面的测试进展 3. 查看分配的测试页面并点击接受任务
预期结果	模态框左侧显示页面结的树状可视化图 在已有众包工人提交的页面上用黄色图标进行标记 在当前众包工人提交缺陷报告的页面上用红色图标进行标记 当鼠标放在某个页面节点时，显示该页面当前众包工人提交缺陷数和总发现缺陷数的比率 在右上角显示分配的测试任务界面 用户点击接受后在新缺陷报告中自动填写分配三级页面信息
实际结果	与预期结果相符

表 5-6: 审核任务分配测试用例

用例编号	TC-TA-2
测试功能	系统为众包工人个性化分配审核任务
前置条件	TC-TA-1 测试用例正确执行
正常流程	1. 查看总体分配审核任务列表 2. 点击第一条分配审核的缺陷报告详细查看 3. 重新验证后点击点赞操作
预期结果	模态框右侧显示分配审核缺陷报告列表 众包工人点击某一条缺陷报告时显示详情 点击点赞按钮，点赞按钮高亮显示，提示点赞成功
实际结果	与预期结果相符

保存的功能，如表 5-7中用例所示。在实际执行过程中与预期结果一致。

5.1.3 边界测试

大量的测试经验表明，软件错误的发生往往出现在极端或者边界的情况下。因此，本小节将对系统中的各个模块进行边界测试，保证测试的全面覆盖以确保系统的高可用。

对于知识图谱数据获取模块，其边界测试用例如表 5-8。它的边界条件多发生在用户未能按要求完整填写所有信息。

知识图谱任务分配模块的边界测试用例如表 5-9所示。其边界条件主要发生在系统内还没有缺陷报告，和众包工人完成所有任务的情况下。

表 5-7: 构建知识图谱及模型测试用例

用例编号	TC-KG-1
测试功能	在每一次出新众包任务时处理数据并训练模型
前置条件	系统中出现新的众包任务
正常流程	1. 运行 get_kg() 构建知识图谱中新的节点和关系 2. 数据预处理获得 kg_final 和 ratings_final 文件 3. 数据输入模型训练后保存模型
预期结果	知识图谱中加入新的节点和关系 模型成功训练并保存
实际结果	与预期结果相符

表 5-8: 知识图谱数据获取模块边界测试用例

用例 ID	用户操作	预期结果	实际结果
1	众包工人未填写测试基本信息并点击下一步	系统提示请先填写基本信息	与预期结果相符
2	众包工人未选择缺陷报告某项属性	系统提示报告信息不完整	与预期结果相符
3	系统内还没有已提交的缺陷报告	无相似报告推荐	与预期结果相符
4	众包工人选择的属性没有匹配到相似缺陷报告	无相似报告推荐	与预期结果相符
5	众包工人缺陷报告未关联测试用例	系统提示请先指定测试用例	与预期结果相符

5.1.4 性能测试

性能测试是使用自动化的工具来模拟系统使用期间可能遇到的各种情况，包括正常、峰值或异常负载等，以此判断系统各项性能参数是否满足预期要求，找到性能瓶颈并针对性地进行优化。本小节使用 JMeter 工具对系统各个模

表 5-9: 知识图谱任务分配模块边界测试用例

用例 ID	用户操作	预期结果	实际结果
1	众包工人没有接受分配的任务直接关闭任务分配界面	系统正常显示报告提交界面	与预期结果相符
2	系统内还没有提交的缺陷报告情况下分配审核任务	分配的审核缺陷报告列表为空	与预期结果相符
3	众包工人完成所有页面的测试任务	提示已完成所有测试任务	与预期结果相符
4	众包工人完成所有页面的审核任务	分配的审核缺陷报告列表为空	与预期结果相符

块的关键接口进行压力测试，具体过程如下：

1) 首先在测试计划列表下建立一个线程组。线程组是一组线程的集合，而每一个线程都是对一个用户的模拟，线程组的线程数经过设定后不会改变。

2) 设置线程组中线程数量为 200，总启动时间为 5，循环次数为 2，表示 JMeter 会模拟 200 个用户在 5s 内对服务器分别发送两次请求。

3) 在线程组中添加两组 HTTP GET 请求，分别对测试任务分配和审核任务分配算法进行压力测试。另外，创建一组 HTTP POST 请求，对知识图谱数据获取模块关键接口进行测试。

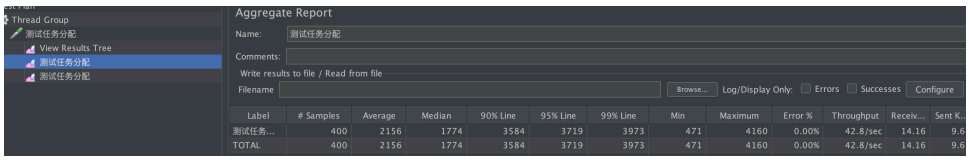


图 5-1: 测试任务分配压测报告

4) 在线程组中再添加一个用户监听结果的聚合报告。这里以测试任务分配为例，其测试报告如图 5-1。根据图中结果，JMeter 在连续的五秒时间对添加的接口进行了 400 次访问请求，这些接口的平均响应时间为 2156ms，中位数为 1774ms，并且 99 % 的请求响应时间都限制在 4 秒内，没有发生错误或崩溃。据此，系统满足了相关性能要求。图 5-2记录了每 100ms 中接口的响应时间变化，整体较为平稳。

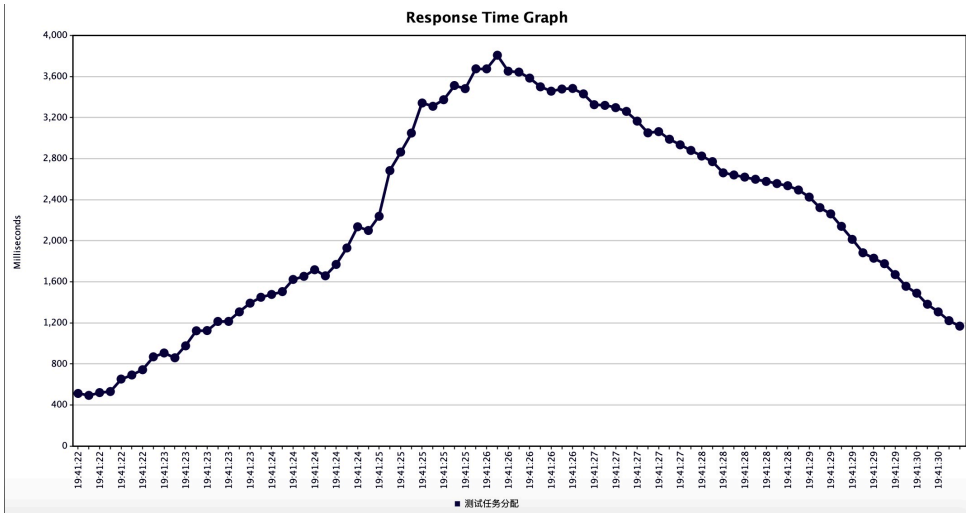


图 5-2: 响应时间图

5.2 实验评估

5.2.1 实验目的

本实验的目的是评估基于知识图谱的众包测试任务分配技术在提高测试报告质量及测试需求覆盖率，减少重复报告数量，提高总体测试效率方面的能力。为了能清晰反映出本技术带来的影响，我们将 Han [?] 提出的众包测试任务分配方法作为对比方法 (为了便于引用，在后文中称为 Han 任务分配方法)。Han 任务分配方法在测试任务分配时基于三级页面的层级结构路径推荐距当前页面路径最短的页面任务，审核任务分配基于用户历史点赞点踩缺陷报告列表运用协同过滤的方法推荐新的缺陷报告。通过对两种分配方法的对比，希望证明在众包测试任务分配过程中引入知识图谱在上述目标中的优越性。

5.2.2 实验设置

首先我们对待测软件进行筛选，从候选列表中找到历史执行过系统的众包测试，确认其各级页面下均有缺陷的测试软件。这样达到要覆盖所有测试需求，需要在各级页面下均找到缺陷的条件。

表 5-10: 测试结果记录

	A 组	B 组
提交缺陷数	234	255
6 分以上缺陷数	171	158
高分比率	73.1%	62.0%
重复缺陷数	24	37
Fork 树个数	48	43
审核缺陷数	178	129
审核比率	77.8%	50.6%
页面全覆盖最小时间	81 分钟	未完成

我们找到 100 名众包工人，这些众包工人均执行过超过 3 次的众包测试任务，发现超过 50 个被后期认定为有效的缺陷，我们认为这些众包工人可以代表普通众包工人的水平。在实验中将其随机分成 AB 两组各 50 人，A 组系统采用本文中提出的“基于知识图谱的众包测试任务分配技术”，B 组系统采用 Han 任务分配方法，同时对待测软件进行了两个小时的众包测试。

在测试完成后我们统计了两组众包工人提交缺陷报告的数量、审核缺陷报告数量、Fork 树个数、页面全覆盖的最小时间，并邀请了领域内专家对这些缺陷报告进行打分和判重，统计了高分缺陷数和重复缺陷数。最后我们根据 3.4.1 节中提出的任务分配召回率指标对两种分配算法进行了定量化评估，验证本技术在提高测试报告质量及测试需求覆盖率，减少重复报告数量，提高总体测试效率方面的能力。

5.2.3 实验结果分析

表 5-11: 众包工人对分配任务执行情况记录

	A 组	B 组
众包工人接受测试任务分配次数	182	161
众包工人拒绝测试任务分配次数	52	94
测试任务分配召回率	77.8%	63.1%
众包工人接受审核任务分配次数	147	95
众包工人拒绝测试任务分配次数	31	34
审核任务分配召回率	82.6%	73.6%

如表 5-10所示，在两个小时的测试时间中，A 组共提交 234 份缺陷报告，其中专家评分 6 分及以上的缺陷数量为 171，比率为 73.1 %。有 24 份缺陷报告存在重复，通过 Fork 操作形成的树状结构供 48 个，审核缺陷报告共 178 份，审核比率为 77.8 %。众包工人在规定时间内发现所有页面的缺陷，完成测试路径的覆盖用时 74 分钟。

表 5-12: 测试任务分配三级页面举例

一级页面	二级页面	三级页面
首页	浏览分类	校园卡
		身份证
个人信息	修改个人信息	校园卡
		身份证

而 B 组提交缺陷报告数量高于 A 组，为 255 份，但其 6 分以上缺陷数为 158，比率仅为 62.0 %，低于 A 组。其中重复缺陷报告数量也较 A 组有所增加，为 37 份。Fork 树个数略低于 A 组，为 43 个。审核缺陷数量则远低于 A 组，仅有 129 份，比率为 50.6 %。因为有众包工人在测试中途离开未能完成自

己的测试任务，导致出现测试页面没有进行测试的情况，最终未完成页面的全覆盖。通过以上数据，我们可以看出基于知识图谱的众包测试任务分配技术相比于普通的协同过滤等技术可以提高众包测试报告质量及测试需求覆盖率，减少重复报告数量，提高总体测试效率方面的能力。

表 5-11中记录了 AB 组众包工人对分配的测试任务和审核任务的执行情况及根据 3.4.1 节中提出的算法计算得到的任务分配召回率。可以看出，众包工人对于基于知识图谱的众包测试任务分配技术更为接受。

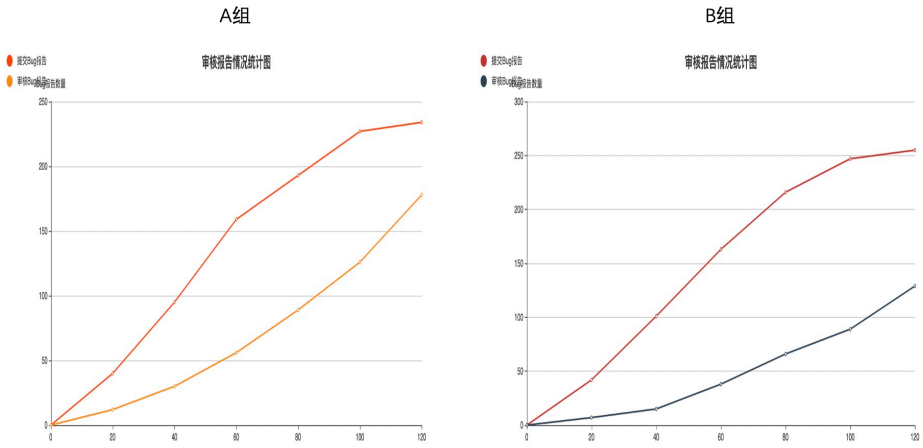


图 5-3: 审核任务执行情况统计图

分析使用 Han 分配方法的 B 组数据，其测试任务分配只考虑了三级页面的跳数层级关系，没有考虑三级页面间内容的相似性。例如对于 5-12中的三级页面任务，“首页-浏览分类-校园卡”与“个人信息-修改个人信息-校园卡”，它们的一二级页面不一致，在树状结构中两个页面节点的跳数也呈最大，但在内容上均属于对校园卡信息的处理，所以本技术对完成其中一个任务的众包工人分配了另外的任务，而 Han 分配方法没有做此分配。结果显示，A 组众包工人均接受了此种类型的分配，证明了基于层级跳数的测试任务分配存在的局限，以及知识图谱运用在测试任务分配上的优越性。

对于审核任务分配，Han 分配方法只考虑了众包工人在本场众包测试中历史点赞点踩缺陷报告的数据，当交互数据少且缺陷数量多的情况下，由于交互矩阵的稀疏造成分配的审核报告集中在固定的缺陷报告范围中，大多数的缺陷报告审核任务不能有效分配。而基于知识图谱的分配技术由于考虑了所有报告间的语义相似度，在交互矩阵稀疏的情况下其分配的缺陷报告审核任务也相对较为均衡。如图 5-3可见，A 组的审核任务提交曲线较 B 组更为平滑，最终审

核缺陷报告的数量与比例也高于 B 组。因此，知识图谱相对于普通协同过滤方法更适用于众包测试审核任务分配。

表 5-13: Fork 报告树举例

报告 ID	创建时间	父 报 告 ID	缺陷描述	补充说明
1341	8:07	/	点击更多新闻时 APP 崩溃	发现缺陷并作为根节点
1687	8:12	1341	1. 点击更多新闻 2. 系统停止运行	使用步骤规范缺陷描述
1689	8:13	1341	第一次点击更多新闻按钮时 APP 崩溃	具体描述问题出现的情况，便于复现
1693	8:34	1689	1. 第一次点击更多新闻 2. 系统停止运行 3. 重启恢复正常	总结 1687 和 1689 两份缺陷报告内容
1705	8:55	1693	1. 第一次点击更多新闻 2. 系统停止运行 3. 重启后点击更多新闻 4. 系统长时间加载后正常	提出当 APP 再次重启时系统加载时间长的问题并完整描述问题

由表 5-10数据可见，使用有效的基于知识图谱的个性化测试任务分配，A 组众包工人协作缺陷报告的意愿也有所增强。如表 5-13为 A 组测试过程中一个缺陷报告树中包含的缺陷报告，记录了众包工人通过协作方式对一个问题层层深入的过程，并最终有效提高了缺陷报告质量。

表 5-10中 A 组的审核报告数显著高于 B 组，对于缺陷报告的质量评估有很大的帮助。如图 5-4为报告被点赞数量与专家评分结果的关系图。其中横坐标为缺陷报告被点赞的数目，纵坐标为专家评分。由图 5-4可见，随着缺陷报告被点赞数量的增多，报告评分也整体上升。因此 A 组报告最终可以通过参考缺陷报告的评分快速评价各个缺陷报告，并利于最终报告的交付。

综上所述，采用基于知识图谱的众包测试任务分配技术相较于 Han 任务分配方法分配众包测试任务，可以鼓励众包工人间相互协作，减少重复报告；帮助实现测试页面的全覆盖；有效的提高缺陷报告的质量，推动缺陷报告的初次评审，有利于最终可交付测试报告的形成。

5.3 本章小结

本章根据第三章提出的需求，对系统的知识图谱数据获取模块、知识图谱特征学习模块和知识图谱任务分配模块进行了功能测试、边界测试和性能测

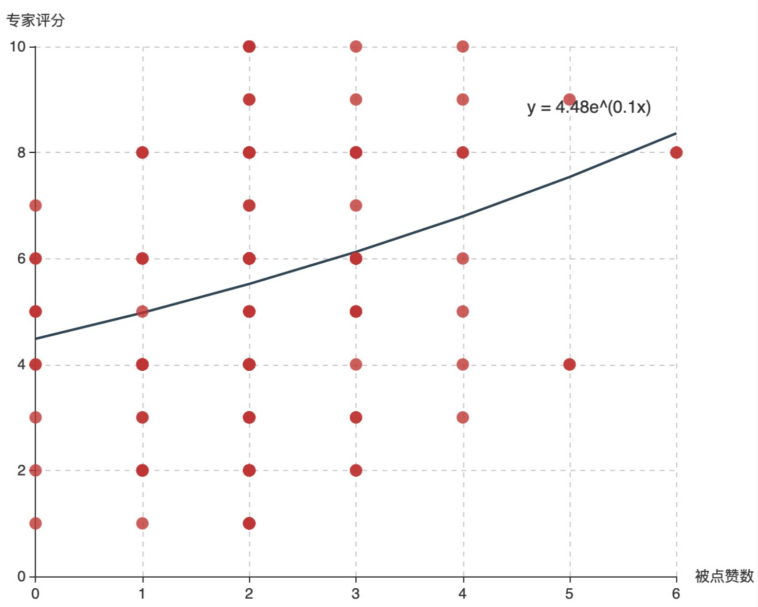


图 5-4: 报告被点赞数和专家评分关系图

试。测试结果表明本系统已经达到了需求阶段提出的目标，且具有较好的可用性。同时，通过对照实验的方法对本系统基于知识图谱的任务分配技术与其他众包测试任务分配算法进行了对比，验证了基于知识图谱的众包测试任务分配技术可以达到提高测试报告质量及测试需求覆盖率，减少重复报告数量，提高总体测试效率方面的效果。

第六章　总结与展望

6.1　总结

众包测试可以在短时间内招募大量测试工人对待测软件进行测试，解决了传统测试过程中测试人员固定、反馈少、成本高的缺点。但同时，众包模式也也可能带来测试人员质量参差不齐、测试效率低、重复缺陷报告数量多的问题。为了解决以上竞争式众包测试中存在的问题，在本文中引入协作式众包测试概念，发挥群体智能的优势对缺陷报告进行不断完善，同时优化任务分配技术，引入知识图谱补充众测领域内知识，通过机器学习为众包工人实现个性化分配任务，提高总体测试效率。具体工作包括如下几个方面：

（1）协作式报告提交：众包工人可以根据测试用例提交缺陷详细报告。在报告提交时，鼓励众包工人使用 Fork 操作对描述过同样问题的历史报告进行完善，使用树状结构保每次存修改记录及其父子节点。同时，通过点赞点踩缺陷报告对他人报告进行评审，为最终交付报告奠定基础。

（2）基于知识图谱的任务分配技术：根据众包测试领域内知识构建测试任务分配和审核任务分配相关本体库，对某平台历史众包测试数据进行知识抽取并搭建知识图谱，使用 RippleNet 模型采用联合学习的方法进行训练，最终输出某个众包工人选择某个三级页面任务和某个缺陷报告的概率。

（3）可视化任务分配展示：将测试需求中三级页面信息用树状结构可视化展示，同时标注出当前各个页面上总体找到缺陷数量与当前众包工人发现缺陷数量，方便掌握总体测试动向。利用任务分配技术得出结果为众包工人分配测试任务和审核任务，引导其发挥优势，提高总体测试效率。

在实现技术上，本系统前端使用 Angular2 框架，用 Echarts 进行可视化界面的快速渲染。后端平台部分使用 Spring Boot 框架，基于知识图谱的任务分配部分使用 Python 脚本实现。数据库部分使用 MongoDB 数据库存储各项用户操作数据，对热点数据使用 Redis 缓存。对于知识图谱相关数据使用 Neo4j 图数据库以三元组形式存储。同时系统使用 Nginx 进行负载均衡，使用 Docker 容器化技术对所有模块独立化部署，保证系统各部分的健壮性和可移植性。

最终对系统进行了功能测试和性能测试并顺利通过，证明系统满足了提出的各项需求。同时系统记录了用户被分配任务后的具体操作，利用召回率证明了任务分配技术的有效性。又采用设计实验的方式比较使用基于知识图谱的任务分配技术系统与采用协同过滤技术分配任务的系统在一次众包测试中提交缺陷报告的数量和质量，再次证明了基于知识图谱的任务分配技术的有效性。

6.2 展望

在目前阶段系统仍然存在许多不足之处，在未来主要可以从以下几个方面进行改进:

（1）测试用例的协作式设计。现阶段测试用例由每名众包工人在提交测试报告时自行撰写，费时费力且会存在很多重复的测试用例，在生成最终交付缺陷报告时也没有起到推动作用。因此，考虑将众测任务分成多个阶段。第一阶段由众包工人协作编写测试用例，保证其完整性，之后再根据共同撰写的测试用例编写缺陷报告。

（2）引入语义分析。目前阶段构建的测试任务分配知识图谱和审核任务分配知识图谱均是根据领域内知识采用自顶向下的方式构建的，没有考虑一些通用语义上的联系，例如“最热景点”和“热门旅行地”可能是两个关联度很高的众测任务，可以推荐给对其中某一个感兴趣的众包工人，未来可以通过引入语义知识图谱比较两者的语义相似度，提高分配系统的准确性。

（3）当系统使用人数增多，并发访问量提高时，为了保证系统的稳定，可以建立数据库集群，并采用主从部署的方式对数据进行备份。

致　　谢

首先我想感谢我的导师陈振宇老师。有幸在读研期间能加入 ISE 智能软件工程实验室，在日常学习生活中给了我很多学习上的指导和帮助。特别是在毕业设计期间，陈老师给我们提供了选题的思路以及许多资料，帮助我们去逐步深入问题。同时，他定期的检查也及时暴露了我不同阶段的问题，让我时刻反省，不再懈怠。

其次我要感谢 ISE 实验室慕测项目组的黄勇老师和所有同学，实验室给我们提供了很多的项目机会让我可以充分的动手实践，提高编程技术。黄老师定期的会议与交流也让我接触到许多新技术。尤其是李文龙、徐佳炜同学，在毕设期间我们也经常一起交流，受益良多。

最后感谢我的父母给我全方位的支持，感谢母校南京大学对我们的悉心培养，希望在今后的人生旅程中我能不忘初心，继续前行。

参考文献

[1] HOWE J. The Rise of Crowdsourcing[J]. Wired magazine, 2006, 14(6): 1–4.

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

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

[4] SHEN H, LI Z, LIU J, et al. Knowledge sharing in the online social network of yahoo! answers and its implications[J]. IEEE transactions on computers, 2014, 64(6): 1715–1728.

[5] CHENG P, LIAN X, CHEN L, et al. Task Assignment on Multi-Skill Oriented Spatial Crowdsourcing[J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(8): 2201–2215.

[6] BOGERS M, AFUAH A, BASTIAN B. Users as Innovators: A Review, Critique, and Future Research Directions[J]. Journal of Management, 2010, 36(4): 857–875.

[7] 章晓芳, 冯洋, 陈振宇, et al. 众包软件测试技术研究进展 [J]. 软件学报, 2018, 29(1): 69–88.

[8] NARULA P, GUTHEIM P, ROLNITZKY D, et al. MobileWorks: A Mobile Crowdsourcing Platform for Workers at the Bottom of the Pyramid.[J]. Human Computation, 2011, 11(11): 45.

[9] ZANATTA A L, MACHADO L, STEINMACHER I. Competence, collaboration, and time management: Barriers and recommendations for crowdworkers[C]// 2018 IEEE/ACM 5th International Workshop on Crowd Sourcing in Software Engineering (CSI-SE). 2018: 9–16.

[10] ALSAYYARI M, ALYAHYA S. Supporting coordination in crowdsourced software testing services[C] // 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). 2018 : 69 – 75.

[11] MRIDHA S K, BHATTACHARYYA M. Introducing Collaboration in Competitive Crowdsourcing Markets:Toward Managing Decomposable Tasks[J]. IEEE intelligent systems, 2019, 34(1) : 23 – 31.

[12] PAN Z, YU H, MIAO C, et al. Efficient collaborative crowdsourcing[C] // Proceedings of the AAAI Conference on Artificial Intelligence : Vol 30. 2016.

[13] WU W, TSAI W-T, HU Z, et al. Towards a game theoretical model for software crowdsourcing processes[G] // Crowdsourcing. China : Springer, 2015 : 143 – 161.

[14] CUI Q, WANG S, WANG J, et al. Multi-Objective Crowd Worker Selection in Crowdsourced Testing.[C] // SEKE : Vol 17. 2017 : 218 – 223.

[15] CHILTON L, HORTON J, MILLER R, et al. Task search in a human computation market. In Proceedings of the ACM SIGKDD workshop on human computation (pp. 1–9)[C] // HCOMP'09: Proceedings of the ACM SIGKDD Workshop on Human Computation. 2009.

[16] 丁婧, 易树平, 杨文彩, et al. "任务-人"匹配对人-信息系统交互效率影响的探讨 [J]. 人類工效學, 2007, 13(3) : 49 – 51.

[17] GEIGER D, SCHADER M. Personalized task recommendation in crowdsourcing information systems—Current state of the art[J]. Decision Support Systems, 2014, 65 : 3 – 16.

[18] RAHMAN H, ROY S B, THIRUMURUGANATHAN S, et al. Task assignment optimization in collaborative crowdsourcing[C] // 2015 IEEE International Conference on Data Mining. 2015 : 949 – 954.

[19] PASCANU R, MIKOLOV T, BENGIO Y. Proceedings of the 30th International Conference on International Conference on Machine Learning-Volume 28[J], 2013.

[20] MOAYEDIKIA A, ONG K-L, BOO Y L, et al. Task assignment in microtask crowdsourcing platforms using learning automata[J]. Engineering Applications of Artificial Intelligence, 2018, 74 : 212 – 225.

[21] QIAO L, TANG F, LIU J. Feedback based high-quality task assignment in collaborative crowdsourcing[C] // 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA). 2018 : 1139 – 1146.

[22] ADOMAVICIUS G, TUZHILIN A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(6) : 734 – 749.

[23] PAZZANI M J, BILLSUS D. Content-based recommendation systems[G] // The adaptive web. USA : Springer, 2007 : 325 – 341.

[24] BORDES A, USUNIER N, GARCIA-DURAN A, et al. Translating embeddings for modeling multi-relational data[C] // Neural Information Processing Systems (NIPS). 2013 : 1 – 9.

[25] BORDES A, GLOROT X, WESTON J, et al. Joint learning of words and meaning representations for open-text semantic parsing[C] // Artificial Intelligence and Statistics. 2012 : 127 – 135.

[26] WANG H, ZHANG F, WANG J, et al. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems[C] // Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018 : 417 – 426.

[27] WANG M, LIU M, LIU J, et al. Safe medicine recommendation via medical knowledge graph embedding[J]. arXiv preprint arXiv:1710.05980, 2017.

[28] LU C, LAUBLET P, STANKOVIC M. Travel attractions recommendation with knowledge graphs[C] // European Knowledge Acquisition Workshop. 2016 : 416 – 431.

[29] HAUSSMANN S, SENEVIRATNE O, CHEN Y, et al. FoodKG: a semantics-driven knowledge graph for food recommendation[C] // International Semantic Web Conference. 2019 : 146 – 162.

[30] TUNG Y H, TSENG S S. A novel approach to collaborative testing in a crowd-sourcing environment[J]. Journal of Systems & Software, 2013, 86(8): 2143 – 2153.

[31] BOLLACKER K, COOK R, TUFTS P. Freebase: A shared database of structured general human knowledge[C] // AAAI: Vol 7. 2007: 1962 – 1963.

[32] BIZER C, LEHMANN J, KOBILAROV G, et al. DBpedia-A crystallization point for the Web of Data[J]. Journal of web semantics, 2009, 7(3): 154 – 165.

[33] SUCHANEK F M, KASNECI G, WEIKUM G. Yago: A large ontology from wikipedia and wordnet[J]. Journal of Web Semantics, 2008, 6(3): 203 – 217.

[34] RYSAVY F, CERNY T, TOMASEK M. Aspect-Oriented User Interfaces Design Integration to Angular 2 Framework[C] // 2016 6th International Conference on IT Convergence and Security (ICITCS). 2016: 1 – 3.

[35] GUTIERREZ F. Spring Boot Messaging[M]. USA: Springer, 2017.

[36] MERKEL D. Docker: lightweight linux containers for consistent development and deployment[J]. Linux journal, 2014, 2014(239): 2.

[37] BERNSTEIN D. Containers and cloud: From lxc to docker to kubernetes[J]. IEEE Cloud Computing, 2014, 1(3): 81 – 84.

[38] BOETTIGER C. An introduction to Docker for reproducible research[J]. ACM SIGOPS Operating Systems Review, 2015, 49(1): 71 – 79.

[39] RAHARTOMO A, AJI R F, RULDEVIYANI Y. The application of big data using MongoDB: Case study with SCeLE Fasilkom UI forum data[C] // 2016 International Workshop on Big Data and Information Security (IWBIS). 2016: 51 – 56.

[40] WANG Y, ZHANG L, TAN J, et al. Hydradb: A resilient rdma-driven key-value middleware for in-memory cluster computing[C] // SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2015: 1 – 11.

[41] ROCHA H, OLIVEIRA G D, MARQUES-NETO H, et al. NextBug: a Bugzilla extension for recommending similar bugs[J]. Journal of Software Engineering Research & Development, 2015, 3(1): 1 – 14.

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

简历与科研成果

基本信息

段梦洋，女，汉族，1997 年 11 月出生，陕西省咸阳市人。

教育背景

2019 年 9 月 — 2021 年 6 月 南京大学软件学院 硕士

2015 年 9 月 — 2019 年 6 月 南京大学软件学院 本科

《学位论文出版授权书》

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

作者签名：_____

_____年_____月_____日

论文题名	基于知识图谱的众测任务分配技术的设计与实现				
研究生学号	MF1932037	所在院系	软件学院	学位年度	2021
论文级别	☐ 硕士　☐ 硕士专业学位 ☐ 博士　☐ 博士专业学位 （请在方框内画勾）				
作者 Email	mf1932037@smail.nju.edu.cn				
导师姓名	陈振宇教授，冯洋助理研究员				

论文涉密情况：

☐ 不保密

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

注：请将该授权书填写后装订在学位论文最后一页（南大封面）。