南京大學

研 究 生 毕 业 论 文
（申请工程硕士学位）

论 文 题 目 面向移动应用功能点测试的知识图谱构建技术

作 者 姓 名 王旭

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，房春荣 助理研究员

2022 年 **5** 月 **23** 日

学　　号：**MF20320166**

论文答辩日期：**2022** 年 **05** 月 **20** 日

指 导 教 师：　　（签字）

Knowledge Graph Construction Technology with Mobile Application Function Point Testing

by

Xu Wang

Supervised by

Professor Zhenyu Chen, Assistant Professor Chunrong Fang

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 23, 2022

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：面向移动应用功能点测试的知识图谱构建技术

工程硕士（软件工程领域）专业 2020 级硕士生姓名：王旭

指导教师（姓名、职称）：陈振宇 教授，房春荣 助理研究员

摘　要

由于移动应用的快速迭代，移动应用自动化测试也迅速发展起来。主流的移动应用自动化测试框架已经可以通过同一份脚本对碎片化的移动应用设备做富含逻辑的自动化测试。然而，在面对成千上万的移动应用时，我们仍需根据应用的特性编写特定的脚本。在研究中我们发现看似不同的移动应用之间其实存在着千丝万缕的联系。例如几乎所有移动应用都具备登录的功能。我们将这种宏观的通用功能概念称之为移动应用的功能点。如果我们可以建立移动应用功能点的知识联系，则可以进一步降低测试成本。因此我们提供了一种面向移动应用功能点测试的知识图谱构建技术。1）首先，我们设定了一种生成知识图谱的原始数据输入格式，该数据中包括针对某一功能点的多个移动应用图文众包测试报告。2）之后，我们通过知识图谱的特征提取模块对原始报告进行复现步骤的文本提取、屏幕截图的图片分析。我们对文本进行分词处理、词性分析，对屏幕截图进行文本提取、控件提取。3）随后，我们通过知识图谱的关系提取模块将特征提取中的离散的文本、图片知识信息进行关系的绑定和连接。3）最后，我们通过知识图谱的共指消解模块对关系提取中生成的知识信息进行筛查、去重、补充，并将关系提取中得到的知识信息上传至图数据库中。并对数据库中的实体及实体间的关系进行分析。我们从全国软件测试大赛的移动应用众包测试报告中筛选出有效的测试报告作为数据输入，成功构建出了登录、注册、添加购物车、发送邮件、查询机票五个功能点的知识图谱。此外，我们还通过知识图谱的智能查询模块提供了一种通过既往操作流程快速查找后续流程的搜索方法，加速用户的查询与使用。

最后，我们对知识图谱构建系统的关系提取成功率以及结果知识图谱的查询成功率进行了评估，实验表明在 76 款应用的 761 个独立步骤中，系统关系提取的成功率可以达到 97.24%；对结果知识图谱进行的 83 次查询过程中，查询

结果相较于全局搜索减免了 83.94% 的无用操作，同时预测的精确率和召回率也分别达到了 93.66% 和 91.28%。

关键词： 移动应用、自动化测试、功能测试、知识图谱

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Knowledge Graph Construction Technology with Mobile Application Function Point Testing

SPECIALIZATION: Software Engineering

POSTGRADUATE: Xu Wang

MENTOR: Professor Zhenyu Chen, Assistant Professor Chunrong Fang

Abstract

Due to the rapid iteration of mobile applications, mobile application automation testing has also developed rapidly. The mainstream mobile application automation testing framework has been able to do logic rich automation testing on fragmented mobile application devices through the same script. However, in the face of thousands of mobile applications, we still need to write specific scripts according to the characteristics of the application. In the research, we found that there are countless connections between seemingly different mobile applications. For example, almost all mobile applications have the function of login. We call this macro general function concept the function point of mobile application. If we can establish the knowledge connection of mobile application function points, we can further reduce the test cost. Therefore, we provide a knowledge graph construction technology for mobile application function point testing. 1) Firstly, we set up an original data input format for generating knowledge graph, which includes multiple crowdsourcing test reports with texts and graphs of mobile applications for a function point. 2) After that, we use the feature extraction module of knowledge graph to extract the text of the original report and analyze the pictures of screenshots. We conduct word segmentation and part of speech analysis on the text, and extract the text and widget from the screenshot. 3) Then, we bind and connect the discrete text and graph knowledge information in feature extraction through the relationship extraction module of knowledge graph. 3) Finally, we filter, de-duplicate and supplement the knowledge information generated in relationship extraction through the coreference resolution module of knowledge graph, and upload the knowledge information obtained in relationship extraction to the graph database. The entities in

the database and the relationship between entities are analyzed. We selected effective test reports from the crowdsourcing test reports of mobile applications in the national software test competition as data input, and successfully constructed the knowledge graph of five function points: login, registration, adding shopping cart, sending email and querying air tickets. In addition, we also provide a search method to quickly find the subsequent process through the previous operation process by the intelligent query module of knowledge graph, so as to speed up the query and use of users.

Finally, we evaluate the relationship extraction success rate of the knowledge graph construction system and the query success rate of the result knowledge graph. The experiment shows that in 761 independent steps of 76 applications, the success rate of system relationship extraction can reach 97.24 %; During the 83 queries on the result knowledge graph, the query results reduced the useless operations by 83.94 % compared with the global search, and the prediction accuracy and recall rate reached 93.66 % and 91.28 % respectively.

keywords: Mobile Application, Automated Testing, Function Testing, Knowledge Graph

目　　录

插图清单

附表清单

第一章　引言

1.1　选题的背景和意义

随着移动设备的大量普及，移动应用已进入飞速发展时期。在快速的迭代更新过程中，为了尽可能快速准确的保证移动应用的可靠性，移动应用自动化测试 [1] 应用而生。当前移动应用自动化测试主要以 Monkey[2] 和 Appium[3] 框架为主。Money 即一种随机测试的模式，该框架会对移动应用的页面进行盲目的随机点击测试，以达到测试应用稳定性与可靠性的目的。而 Appium 则允许工作人员编写逻辑脚本，按照一定的顺序来操作移动应用以达到测试功能的目的。

尽管当前主流的移动应用框架相较于之前的纯手工测试已经做到了人力成本的大幅度降低，但仍然存在一些问题。Monkey 框架可以直接应用于所有的移动应用产品，但 Money 的操作是随机的，也就是说在整个的测试中不包含逻辑。在现实生活中，涉及到逻辑的故障不论是出现的频率还是危急的程度都占比较高，因此仅通过 Money 对移动应用进行自动化测试是不够的。相较之下，Appium 可以设定指定的逻辑从而测试移动应用的功能。但是在实操的过程中，我们发现 Appium 的脚本非常敏感。同样的一款移动应用，在不同的移动应用设备上可能会出现细微的差异，这可能导致原来的脚本在其他设备上不再适用，尽管测试的是同一款移动应用；同时我们发现不同的移动应用之间的某些脚本高度类似，我们只是因为移动应用的不同、移动应用版本、设备、页面布局等因素的不同，就需要编写多份定制化的移动应用自动化测试脚本，这无疑也增加了人力成本，放缓了测试的进度。

我们发现人力成本仍旧冗余的主要原因在于我们没有对于移动应用的整体认识。仅仅因为移动应用的布局等细小问题就不得不编写新的功能测试脚本。如果我们能提供一种自动化识别这些差异的方式，就可以让功能测试适应于所有移动应用产品。同时，在对不同的移动应用产品进行测试的过程中，我们发现移动应用的功能之间存在某种联系。例如所有的移动应用产品都具备登录的功能，所有购物类的移动应用都具有添加至购物车的功能。我们将类似登录、

添加至购物车这样的通用宏观逻辑称之为移动应用的功能点。如果我们将不同的移动应用按照功能点组装起来，并进行内部的分析连接，那我们就可以组建移动应用功能点自动化测试的知识，从而帮助 Appium 进行适用于所有移动应用的自动化测试。

因此，我们提出了一种基于移动应用功能点测试的知识图谱 [4] 构建技术。通过分析涵盖文本与图片的众包测试报告，将功能点信息进行整合与去重，并添加至知识图谱的方式构，建出一个强大的移动应用自动化测试知识库，以帮助现有的移动应用自动化测试框架更高效、便捷的测试移动应用。

1.2 国内外研究现状及分析

近年来，由于移动应用设备的大量普及和应用，移动应用产品开始飞速发展，这也伴随着移动应用自动化测试的快速改革迭代。为了能够适应移动应用的快速开发与上线，移动应用自动化测试已经成为当前的主流研究课题。

当前比较主流的移动应用测试主要分为三类，分别是随机的、基于模型的或录制回放的 [5]。随机测试通过对移动应用的页面进行随机的点击来测试移动应用。典型的代表有 Monkey、Dynodroid 等。随机测试 [6] 具有测试效率高、适合压力测试等优点。然而随机测试很难生成特定的输入，同时也会创建大量的重复测试来影响测试效率。并且我们也很难规定出一个停止测试的标准。这些因素都影响了随机测试的有效性和实用性。基于模型的测试 [7] 则较为常见，他会通过一些提前训练或总结的模型或知识对移动应用的自动化测试提供辅助信息以帮助测试的自动执行。这样的测试方法可以减少冗余的测试，同时更具效率性。典型的代表有 MAMBA、SSDA、SwiftHand 等。基于录制回放的自动化测试 [8] 则更多的应用于压力和回归测试。典型的代表有 RERAN 等。

为了在功能和逻辑上对移动应用产品进行自动化测试，随机的移动应用自动化测试与基于录制回放的移动应用自动化测试显然不符合最终目的，因此移动应用自动化功能测试的研究人员主要将研究重点放在了基于模型的移动应用自动化测试的方法上。Anbunathan R[9] 等人提出的基于数据驱动框架的移动安卓自动化测试生成方法，通过建立特定模型来帮助自动化生成测试用例。但该方法是针对单元级别的测试用例自动化生成，无法满足以用户角度对移动应用进行自动化测试。YuanChun Li[10] 等人提出的一种基于深度学习方法的安卓应用自动化黑盒测试工具 Humanoid。通过分析人对移动应用页面的理解，研究人

员学习到页面中会有特定的元素与人交互，所以他们通过分析众包测试报告并识别图片中的有关数据信息来推算当前页面哪些组件被交互的可能性最大。通过这种方式来帮助移动应用自动化测试。这种方式是从 GUI 层面对移动应用进行测试，同时相较于以往自顶向下的遍历测试而言，也在一定程度上对没有必要的测试进行了规避。然而对于测试的内容并没有一个确定的测试逻辑，也即意味着测试的过程并不能针对应用的功能而进行针对性的测试。

无论是哪种基于模型的移动应用自动化测试，当前主流的做法都是对原有的自顶向下的遍历测试进行尽可能的剪枝操作。然而这些剪枝仅仅是为了降低测试的冗余和非必要性，这些测试之间并没有宏观逻辑的联系。这种联系应该作为一种知识保存起来帮助移动应用以带有逻辑的形式对移动应用进行自动化的功能测试。当下主流的保存知识的方式是使用知识图谱 [4]。知识图谱是谷歌在 2012 年提出的一种新概念，它通过将数据有机的整合联系起来形成知识帮助搜索引擎完成相关内容的检索与分析。该技术也已经逐步成熟起来。主要应用与金融、医疗等领域。例如 Xiaoyi Fu[11] 等人通过知识图谱来对市场进行分析，Boya Cheng[12] 等人则通过知识图谱帮助整理了中医药相关知识，从而辅助医生进行诊断与用药。知识图谱也被广泛应用于信息科学领域，例如 Yue Zhao[13] 等人通过知识图谱来提前预测网络安全等级为用户提供更可靠的安全保障。也有一些知识图谱特定的应用于测试领域，例如 Vaibhav Kesri[14] 等人构建了一张知识图谱来帮助软件进行测试。Wenjun Ke[15] 等人还构建了一种不仅能推荐测试数据还能识别故障类型的知识图谱。但是这两张图谱均负责从单元级别对应用进行测试，且并非针对移动应用构建，放在移动应用领域使用结果并不理想。

1.3 本文的工作

从国内外的研究现状看来，基于模型的移动应用自动化测试已经有很多先进的实例，然而仍然存在以下问题：1）这些自动化测试框架只是为了对自顶向下的冗余遍历测试进行剪枝，并没有考虑到测试步骤之间的逻辑关系，自然也没有办法检测出功能逻辑上的故障。2）没有移动应用领域的知识联系。移动应用的知识是没有被提取到的。同时在当前一些已经构建出来的知识图谱当中，也并没有发现能够将移动应用的知识进行专业级别汇总的图谱。因此，本文通过整合梳理移动应用的相关数据，构建出一张全面完整的移动应用知识图

谱，从而在保证覆盖功能逻辑的基础上，帮助移动应用进行自动化测试。

本文设计并实现了基于移动应用功能点测试的知识图谱构建技术。针对移动应用信息繁杂重复、联系隐蔽复杂等特点，将移动应用的相关信息整合起来。同时，基于功能点的前提，将移动应用 GUI 显示的文本、图片、组件、组件之间的关系等进行有机的整合排列，形成一张具有前后继关系的移动应用自动化测试的知识图谱。首先我们对收集的可信众包测试报告进行图文的并行分析；其次，我们采用文本相似度比较算法、图片相似度比较算法和图片控件与文本提取技术将分析出来的文本与图片信息进行整合连接；之后，我们通过分析整合信息来出去冗余内容生成基于功能点测试的移动应用知识图谱；最后，我们提供统一的对外接口帮助外界与知识图谱进行细节交互。

本文主要从功能测试的角度分析并整合外界冗余繁杂的移动应用相关信息，构建出富含操作逻辑的基于功能点测试的移动应用知识图谱。为此，本文在设计和实现系统时从以下两个方面进行了改进与优化：

1）针对性更强，适应性更广。通过将移动应用测试从测试大类中分离出来构建图谱的方式，使得知识图谱的针对性更强。这钟方式可以发现移动应用测试特有而其他类别测试弱相关的信息并将其永久保存起来。因为特定于移动应用进行构建和开发，所以图谱在移动应用测试中的适应性会更强，从而帮助移动应用自动化的进行功能测试。

2）富含测试逻辑，智能指导测试。由于整张知识图谱都考虑到了移动应用的功能测试逻辑，所以知识图谱之间的关系是存在功能逻辑的。这样的知识不再是对遍历测试的简单剪枝，而是对整个移动应用的宏观理解，从而帮助测试发现特定问题。这样的知识图谱就像人类的大脑，指挥着自动化测试按照逻辑逐步执行。

通过自动化的构建基于功能点的移动应用知识图谱，我们可以帮助包括移动应用自动化功能测试在内的多种技术理解移动应用，更好的执行相关操作。

1.4 本文的组织结构

本文主要分为六个章节，组织结构如下：

第一章引言。本章主要介绍了基于功能点测试的移动应用知识图谱构建技术的选题背景和意义、国内外研究现状，并对本文的主要研究工作进行了整体阐述。

第二章系统技术概述。本章首先对移动应用自动化测试、功能测试、功能点测试、场景、知识图谱等相关概念进行了阐述。然后对项目所涉及到的技术和框架进行了介绍。其中包括文本分析的工具与算法（JieBa、Word2Vec、synonyms），图片理解的工具与算法（Canny、CNN、SIFT）以及知识图谱的存储框架（Neo4j）。

第三章系统需求分析与概要设计。本章首先对项目整体背景进行描述，分析和总结出系统的功能性与非功能性需求，并结合用例图和用例表对需求进行阐述。给出了系统总体架构与设计思路，对系统涉及的知识图谱特征提取模块、知识图谱关系提取模块、知识图谱共指消解模块、知识图谱智能查询模块的设计方案进行说明。最后对系统数据库设计进行了阐述。

第四章系统详细设计与实现。在第三章中对系统的需求分析及概要设计的基础上，通过展示调用顺序图、类图及关键代码分别对知识图谱特征提取模块、知识图谱关系提取模块、知识图谱共指消解模块和知识图谱智能查询模块的主要功能实现细节进行阐述。同时也对系统的前端页面进行了展示与讲解。

第五章系统测试与运行效果分析。通过对知识图谱关系提取结果的分析统计了知识图谱构建的正确率，通过知识图谱智能查询的结果统计了知识图谱预测的精确率与召回率。展示了面向功能点测试的知识图谱构建技术的可行性。

第六章总结与展望。本章总结了论文完成期间所做的工作，并就基于功能点测试的移动应用知识图谱构建技术的未来扩展提出了进一步的展望。

第二章　系统技术概述

2.1　移动应用自动化测试

软件的自动化测试 [16] 是根据测试脚本，将人工驱动的测试行为转化为机器自动测试的过程，即通过工具执行程序语言编写的测试脚本，模拟手工测试步骤，实现软件的自动测试。移动应用自动化测试即为针对移动应用产品而执行的软件自动化测试。在软件开发过程中，自动化测试逐渐取代纯手工测试的主要原因是软件开发的开发周期越来越短，这也意味着自动化测试在成本、时间还是质量上都比手工测试更具优势。例如 1）自动化测试可以更便于回归测试的执行；2）同时由于脚本编写的原因，自动化测试还可以保证更好的一致性与复用性；3）除此以外，自动化测试可以帮助更好的配置人力资源，将原来繁杂的工作从人力成本中剥离出来；4）不仅如此，自动化测试还可以帮助执行人工难以完成的复杂测试；5）而且自动化测试还可以帮助增强软件的可信度。这些优点使得自动化测试逐渐盛行。

2.2　功能测试与功能点测试

软件测试可以从不同角度被归类为不同的测试。比较常见的是根据是否对内部代码进行审核来将软件测试区分为黑盒测试和白盒测试。也有通过对待测软件的测试需求出发将软件测试分为功能测试、性能测试、安全测试、内容测试等。功能测试 [17]，顾名思义，是一种测试软件产品功能的测试方法。该方法是一种上层的黑盒测试，它无需了解底层代码的实现细节，只需要保证待测软件能正确实现设计文档中提到的功能即可。

在众多软件产品中存在着各种各样的功能需要通过功能测试的方式进行测试。而在众多的软件产品中，总是存在一些通用的，富含一定知识逻辑的功能测试。例如几乎所有的软件产品都涉及登录的功能，所有涉及到购物的软件都涉及添加购物车的功能。我们将这种多个软件产品中所通用的某个功能称作功能点。因此，功能点测试即为对于不同的软件产品，针对某个通用的功能做

测试的测试方式。例如可以执行登录功能点的测试、添加购物车功能点的测试等。

2.3 功能点测试场景

针对任意移动应用功能点来说，从初始状态到目标状态之间存在多条操作路径，我们将这些不同的操作路径称为移动应用功能点测试的测试场景。例如对于登录功能点来说，可能会存在包括账号密码登录、手机验证码登录、扫描二维码登录、刷脸登录四个不同的功能点测试场景。通过确定功能点测试的场景，可以深入的理解移动应用与功能点测试，并帮助构建完整的知识图谱。

2.4 知识图谱

知识图谱 [18–24]，是谷歌公司于 2012 年发布的一种流行的知识表示形式，简称 KG。其初衷是为了优化搜索引擎返回的结果，增强用户搜索质量及体验。知识图谱是一种典型的多边关系图，由节点（实体）和边（实体之间的关系）组成，本质上是一种语义网络，用于揭示万物之间关系。知识图谱旨在从多种类型的复杂数据出发，抽取其中的概念、实体和关系，是事物关系的可计算模型。知识图谱按照知识的覆盖范围和不同的领域，整体可以划分为通用性知识图谱和领域性知识图谱。随着科技的不断发展，知识图谱在 NLP 领域应用广泛，如语义搜索、智能问答、辅助决策等领域，知识图谱已经成为了人工智能发展的重要动力和核心领域。

定义：知识图谱是结构化的语义知识库，用于以符号形式描述物理世界中的概念及其相互关系。其基本组成单位是“实体－关系－实体”三元组，以及实体及其相关属性－值对，实体间通过关系相互联结，构成网状的知识结构。

知识图谱的一种通用三元组表示形式，即

$$G = (Entity_{head}, Relation, Entity_{tail})$$

$Entity_{head}$ 是三元组 G 中的头实体，$Entity_{tail}$ 为尾实体，$Relation$ 为两个实体之间的关系，其中 $Entity = [Entity_1, Entity_2, ..., Entity_n]$ 表示实体的集合，包含了 n 种实体的概念;$Relation = [Relation_1, Relation_2, ..., Relation_n]$ 表述实体之间的关系集合，包含了 n 种不同的关系。

知识图谱的架构包括括知识图谱自身的逻辑结构以及构建知识图谱所采用的技术（体系）架构。

知识图谱的自身逻辑结构包括 2 个层次：数据层和模式层。在知识图谱的数据层，知识以实体为单位存储在图数据库中。模式层在数据层之上，是知识图谱的核心。在模式层存储的是经过提炼的知识，通常采用本体库来管理知识图谱的模式层，借助本体库对公理、规则和约束条件的支持能力来规范实体、关系以及实体的类型和属性等对象之间的联系。本体库在知识图谱中的地位相当于知识库的模具，拥有本体库的知识库冗余知识较少。

构建知识图谱所采用的技术架构分为三部分：信息抽取 [25–27]、知识融合 [28] 以及知识加工 [29]。信息抽取即如何在各个类型的数据中获取有用的资源信息。知识融合则用于关联多数据源的知识，扩大知识范围。知识加工即通过知识应用将知识图谱与特定的领域或者业务相结合，提高业务效率。

信息抽取部分包括实体的抽取 [30]、关系的抽取 [31] 以及属性的抽取。知识融合 [28] 部分主要是通过实体连接与知识合并来实现的，包括实体消歧、共指消解、合并外部知识库、合并关系数据库等。知识加工 [29] 则通过本体构建、知识推理、质量评估等部分组成。

知识图谱在现实世界中已经得到了广泛的应用。知识图谱通过将互联网中海量的信息进行总结和归纳，将信息转变为知识以帮助人类更好的利用。知识图谱目前主要应用于智能语义搜索、移动个人助理、深度问答系统等领域。由于其高效的可解释性，其不仅可以提升深度学习的可解释性，也可以用于指导深度学习的进程，助力于深度学习的高速发展。

然而知识图谱在实际应用中仍然存在着不少挑战。例如对于信息提取环节，面向互联网开发内容的信息提取技术还处于起步阶段，普遍存在算法召回率低、限制条件多等问题。对于知识统合环节，如何实现准去的实体连接也是不小的挑战。在知识加工、知识更新等环节也存在着各式各样的困难等待着研究人员进一步解决。

知识图谱主要存储在图数据库中。图数据库也是一种非关系型数据库，图数据库是以点、边为基础存储单元，以高效存储、查询图数据为设计原理的数据管理系统。在 DB-Engines[①]提供的图数据实时排名页面中，截至 2022 年 3 月，排在前十名的分别是 Neo4j、TigerGraph、Dgraph、Giraph、Nebula Graph、InfiniteGraph、Memgraph、FlockDB、HyperGraphDB、HugeGraph。

[①]https://db-engines.com/en/ranking/graph+dbms

开发者们通常会根据文档的丰富程度、支持的编程语言、软件的系统版本、使用的费用支出的因素综合考虑并选择某款产品使用。

2.5 事件知识图谱

事件知识图谱 [32, 33] 是知识图谱之后对知识图谱进行细分递进的产物。普通的知识图谱通过实体与关系构建起整张知识图谱的网络，而事件知识图谱则需要通过将实体与事件绑定在一起形成最终的知识网络。事件知识图谱用实体表示事件的触发器与参数，用边表示事件之间的关系。这种关系可以是时间关系、因果关系等等。

从实例视图的角度来说，事件知识图谱事件的获取主要通过四个步骤来完成。分别是事件提取、事件关系的提取、事件共指消解以及事件的参数完成。其中，事件的提取作为构建 EKG 的首要步骤，旨在从文本中提取结构化事件信息，包括具有类型的事件触发器和具有角色的参数。根据是否存在预定义的模式，事件提取可分为基于模式和无模式的事件提取，包括句子级事件抽取、文档级事件抽取以及开放域事件抽取。事件关系提取则主要根据事件提取中的实体内容提取其间时间、因果等方面的关系。事件的共指消解即对于许多文本描述相同的事件，在只是提取之后有必要将参考相同真实世界事件的事件分组到相同的簇中，以便更好地构建 EKG。事件参数完成则为了解决原始文本中的信息的不完整性，避免事件提取过程中不可避免地会出现一些缺失，事件参数完成则旨在完成现有事件，通常形式化为推断和填充事件中缺少的参数或参数角色。

2.6 jieba 分词

jieba 分词 [34] 是 Python 中最好用的中文分词之一，在 GitHub 上开源[①]。其根据前缀词典来对词图进行高效的扫描，从而生成句子中可能出现的所有词汇排列而成的有向无环图。同时，通过动态规划的方式找到最大概率的路径并找出基于词频的最大切合分组。在整个实现的流程中，还加入了 HMM 模型与 Viterbi 算法来处理未存储的新词。

jieba 分词支持四种分词模式，分别为精确模式、全模式、搜索引擎模式与

[①] https://github.com/fxsjy/jieba

paddle 模式。这四种模式分别从精确性、高效性、高召回率、深度学习四个层面优化分词的流程与步骤。同时，jieba 分词还支持繁体分词、自定义字典，同时具有 MIT 的授权协议。

jieba 分词主要有分词、添加自定义字典、关键词提取、词性标注、并行分词、命令行分词等八项功能。其中，分词顾名思义即 jieba 可以将一句完整的中文句子根据词语进行句子的分词操作；添加自定义字典即意味着用户可以自定义词库的字典，将词库中不存在的但领域内存在的词汇添加进词库中帮助词库做决策；关键词提取则通过 TF-IDF 算法对句子中的关键词进行提取；词性标注即通过 jieba 可以标注出句子中的动词、名词等语法结构，其中采用了 ictclas 的标注方法；并行分词即可以对多行文本加快分词的进度；命令行分词将命令行中输出的内容进行分词整理。

jieba 不仅有 Python 的实现方式，还有多种其他语言的实现，包括 Java、C++、Rust、Node.js、Erlang、等多种语言。同时，jieba 分词的效率也非常高。在全模式下，jieba 的分词效率可以达到每秒 1.5MB，在默认模式下则可以达到每秒 400KB。

2.7 Word2Vec 算法

Word2Vec[35] 是 2013 年谷歌向公众开源的一款词向量计算工具。该工具可以在百万数量级的词典和上亿的数据集上进行高效的训练，同时该工具生成的词向量还可以很好的解释词与词之间的相关性。Word2Vec 算法是一个浅层的神经网络。我们通常所说的 Word2Vec 算法其实是 Word2Vec 模型中的 CBOW 模型和 skip-gram 模型提供的。Word2Vec 算法已然成为当今社会中 NLP 的主要分析算法，在自然语言处理领域有着举足轻重的地位。

在 Word2Vec 问世之前，很多 NLP 系统都将单词视为独立的概念，他们认为单词只是不同的索引联系，而 Word2Vec 则改变了这一惯性思维，将句子中的单词视作相互联系的个体，从而进一步分析词汇的作用。Word2Vec 主要依赖于两个模型：CBOW 模型和 skip-gram 模型。CBOW 模型的全称，即 Continuous Bag-of-Words 模型。简单的说，该模型可以通过上下文来预测当前值。在该模型下，词语或文件这样的内容会像被装入袋子一般，不考虑文法以及词的顺序。skip-gram 模型即用当前词来预测上下文。在 NLP 的研究领域中，语料的选取是一个很重要的环节。首先，语料必须充分，其次，语料必须

准确。然而并不是将语料选择充分并保证准确就可以保证最终结果的正确性的，很多时候，处理方法的选择也会导致最终结果正确性的问题。在 n 元模型中，因为窗口大小的限制，导致超出窗口范围的词语与当前词之间的关系不能被正确地反映到模型之中，如果单纯扩大窗口大小又会增加训练的复杂度。Skip-gram 模型的提出很好地解决了这些问题。Skip-gram 即通过跳过某些符号，来帮助正确的训练。

Word2Vec 相较于之前的词向量分析模型来说，由于会考虑文本的上下文，所以使用效果更好。同时，其法维度的降低也让训练的速度得到快速的提升。并且 Word2Vec 的通用性很强，可以用在各种 NLP 的研究当中。然而 Word2Vec 也存在一些问题，例如由于词和向量是一对一的关系，所以对于多义词来说会出现问题。同时，Word2Vec 是一种静态方法，在某些特定任务下无法做动态的优化。基于 Word2Vec 的优缺点，研究人员需要斟酌使用。

2.8 synonyms 中文近义词库

synonyms 是一款高效且准确的中文近义词库，在 GitHub 上开源[①]。是众多中文近义词库中的佼佼者。该词库是通过 Word2Vec 为算法模型，对维基百科内的所有文本进行中文词汇训练的方法构建的。该词库可以提供近义词的查找、词语句子相似度分析等功能，且安装配置简单，并于操作。

2.9 Canny 算法

Canny 算法 [36] 即 Canny 边缘检测算法, 是由 John F. Canny 提出的一种边缘检测算法。图像的边缘即指在一张图片中，图像局部区域亮度变化显著的部分，这块区域的灰度剖面一般可以作为一个阶跃既从一个灰度值在很小的缓冲区域内急剧变化到另一个灰度相差较大的灰度值。Canny 算法通过对集中了图像大部分信息的图象的边缘部分进行分析，将图像的边缘提取出来，确定图像的边界与布局。边缘检测主要是通过图象的灰度变化的度量、检测和定位来实现的。

边缘检测算法一般需要通过滤波、检测、增强这三步来实现。边缘检测算法中主要都是通过基于图像强度的一阶和二阶导数来实现的，然而导数在存在

[①]https://github.com/chatopera/Synonyms

噪声的情况下会变的非常敏感，因此我们需要采用滤波器来改观这种因噪声而产生问题的边缘检测性能。同时我们还需要对边缘部分进行增强。这一步的基础是需要确定图像各个点领域强度的变化值。增强算法可以把需要增强的点凸显出来。在具体实践过程中，可以通过梯度幅度值来衡量该值。除此以外，还需要进行检测操作。经过增强的图像，在一定程度上邻域内会出现很多点的梯度值较大，而这些点在很大程度上并不是我们需要找到的边缘点，因此需要采用某种检测的方式来对这些点进行筛选和取舍操作。在具体实现的过程中，一般会采用设定阈值的方法来实现。

Canny 边缘检测算法首先对图像进行灰度化，之后通过高斯滤波对图像进行滤波操作。在高斯滤波的过程中，可以通过两次一维高斯核或一个二维高斯核来实现。以下是离散化的一维高斯函数，在确定参数后就可以得到一维的核向量。

$$K = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{x \cdot x}{2\sigma \cdot \sigma}} \tag{2-1}$$

以下是离散化的二维高斯函数，确定参数后就可以得到二维核向量。

$$K = \frac{1}{2\pi\sigma \cdot \sigma} e^{\frac{x \cdot x + y \cdot y}{2\sigma \cdot \sigma}} \tag{2-2}$$

之后，再通过一阶偏导的有限差分计算梯度的幅值和方向，再对梯度幅值进行非极大值抑制，最后通过双阈值算法检测并连接边缘，完成 Canny 边缘检测算法的全过程。

2.10 用于控件识别的 CNN 模型

CNN (Convolutional Neural Networks)[37–39] 即卷积神经网络，是机器学习模型中非常重要的一种。CNN 是一类特别设计用来处理二维数据的多层神经网络。该模型被认为是第一个真正成功的采用多层层次结构网络的具有鲁棒性的深度学习方法。CNN 通过挖掘数据中的空间上的相关性，来减少网络中的可训练参数的数量，达到改进前向传播网络的反向传播算法效率，因为 CNN 需要非常少的数据预处理工作，所以也被认为是一种深度学习的方法。在 CNN 中，图像中的局部感知区域被当做层次结构中的底层的输入数据，信息通过前向传播经过网络中的各个层，在每一层中都由过滤器构成，以便能够获得观测

数据的一些显著特征。因为局部感知区域能够获得一些基础的特征，比如图像中的边界和角落等，这种方法能够提供一定程度对位移、拉伸和旋转的相对不变性。

我们通过获取安卓应用市场中可见移动应用的所有控件截图作为输入，经过 CNN 学习出用于控件类型识别的 CNN 模型。该模型可以帮助系统识别某张图片中控件的控件类型，例如是 Button 或 EditText 等。

2.11 SIFT 算法

SIFT(Scale-Invariant Feature Transform) 算法 [40–43]，即尺度不变特征转换算法，是一种电脑视觉的算法用来检测与描述图片中的局部特征，它在空间尺度中寻找极值点，并提取出其位置、尺度、旋转不变量。该算法的应用范围包括物体识别、机器人地图感知与导航、3D 建模等。

SIFT 算法可以被分解为四个步骤，首先是尺度空间极值检测。该步骤会搜索所有尺度上的图像位置。通过高斯差分函数来识别潜在的对于尺度和旋转不变的关键点。关键点定位：在每个候选的位置上，通过一个拟合精细的模型来确定位置和尺度。关键点的选择依据于它们的稳定程度。然后是关键点方向确定。基于图像局部的梯度方向，分配给每个关键点位置一个或多个方向。所有后面的对图像数据的操作都相对于关键点的方向、尺度和位置进行变换，从而保证了对于这些变换的不变性。最后是关键点的描述。在每个关键点周围的邻域内，在选定的尺度上测量图像局部的梯度。这些梯度作为关键点的描述符，它允许比较大的局部形状的变形或光照变化。

SIFT 算法具有较好的稳定性和不变性，能够适应旋转、尺度缩放、亮度的变化，能在一定程度上不受视角变化、仿射变换、噪声的干扰。同时 SIFT 算法还可以区分性好，能够在海量特征数据库中进行快速准确的区分信息进行匹配。SIFT 算法也具备多量性，就算只有单个物体，也能产生大量特征向量。同时 SIFT 算法还具备高速性的特点，能够快速的进行特征向量匹配。SIFT 算法也是可扩展的，能够与其它形式的特征向量进行联合。这些优势都帮助 SIFT 算法在工业界被广泛认可和使用。

2.12 Neo4j 图数据库

Neo4j 图数据库 [44, 45] 是一款高性能的, 非关系型，图形数据库。该数据库将数据以图的形式存储到数据库中。它是一个嵌入式的、基于磁盘的、具备完全的事务特性的 Java 持久化引擎。Neo4j 也可以被看作是一个高性能的图引擎，该引擎具有成熟数据库的所有特性。

Neo4j 是当前最受开发者喜爱的图数据库之一。该数据库通过 Java 进行后台实现。由于使用者众多，在互联网上存在大量的使用即配置文档，可以帮助开发者快速的进行学习并上手。同时，其高效的性能可以支持存储亿万级的数据。该图数据库可以支持多种语言的使用，例如 Python 即可通过 py2neo 包来对数据库进行读写操作。同时该数据库在工业界已经有比较多成功的案例，这也从侧面显示出该数据库的稳定性。同时，该数据库支持索引、事物等关系型数据库中支持的方法来快速的搜索指定内容，加快搜索进程。相较于其他的图数据库而言，Neo4j 长期占据主导地位。支持多样的操作系统且支持集群，是一款非常优秀的图数据库。

2.13 本章小结

本章通过对移动应用自动化测试、功能测试与功能点测试、功能点测试场景、知识图谱、事件知识图谱、jieba 分词、Word2Vec 算法、synonyms 中文近义词库、Canny 算法、用于控件识别的 CNN 模型、SIFT 算法、Neo4j 图数据库十一个概念和技术的阐释，将基于移动应用功能点测试的知识图谱构建技术中所涉及到的关键技术逐一进行了罗列与说明。

第三章　系统需求分析与概要设计

3.1　项目整体概述

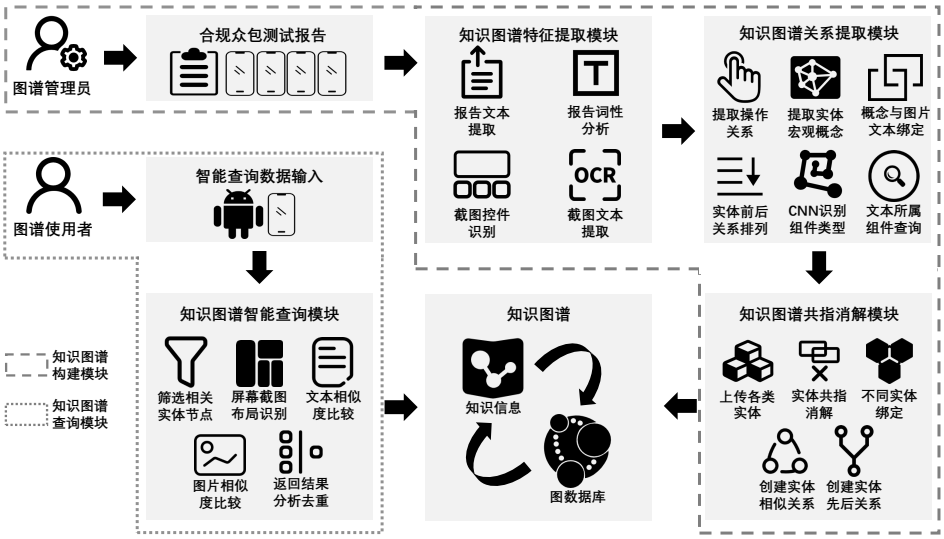


图 3-1: 项目整体流程图

当前，随着移动应用的快速发展，移动应用测试也进入到飞速发展期。为了更快更准的测试移动应用产品，移动应用自动化测试逐渐成为主流的移动应用测试方法。然而当前的移动应用自动化测试方法因为移动应用的不同、移动应用版本的不同等因素导致自动化测试受到局限，灵活性不够。移动应用的自动化测试人员不得不针对不同的移动应用产品编写不同的测试脚本，哪怕是非常简单的差异。这无疑会大量的消耗人力成本，消耗时间。

同时，移动应用领域的相关知识大多都掌握在移动应用领域的专家手中。在移动应用测试的过程中我们需要不断的寻求移动应用领域专家的建议，这种稀缺的访问方式无疑会加大成本同时消耗时间。现阶段还没有移动应用信息的整理组织内容，这也在一定程度上拖慢了移动应用自动化测试的进程。

因此本文提供了一种面向移动应用功能点测试的知识图谱构建技术。以移动应用领域为背景，以自动化测试为目的，以功能点测试作为图谱的构建思路，搭建出一张帮助移动应用领域内的所有参与者了解移动应用的知识图谱，

项目的整体流程如图 3-1所示。我们首先从以往收集到的含有复现步骤的合规移动应用众包测试报告中获取移动应用的操作信息，这些信息中包括原始的操作记录（即复现步骤），同时还包括每步操作的移动应用屏幕截图。我们将这些报告根据功能点分类，有机的组织起来，通过知识图谱特征提取模块对其中的文本、图片做细粒度的拆解，包括文本中有用信息的提取，图片中文本的识别、控件的识别。然后通过知识图谱关系提取模块对控件类型、页面布局进行分析并将拆解的特征结果进行绑定。之后将关系提取出来的数据进行数据的共指消解操作，删去冗余的信息，扩充不足的内容，将数据整理成知识进行存储。在存储的过程中，我们将关系提取数据传递至图数据库，与图数据库中的已有节点进行比较，对相似节点进行补充，对缺失节点进行新增，对重复节点进行删减做到知识的生成，然后构建出最终的知识图谱。在完成知识图谱的构建之后，图谱的使用者即可通过任意移动应用的屏幕截图访问知识图谱的智能查询模块，知识图谱通过筛选相关节点、截图文本控件提取、截图屏幕分析等方式找到需要操作的内容列表作为结果返回给用户，至此完成知识图谱的构建与查询工作。

本文主要设计实现了面向移动应用功能点测试的知识图谱构建技术，通过对原始的众包测试报告进行特征提取，转而对特征数据进行关系提取，最后对关系数据进行共指消解并生成图谱的方式自动化的构建出最终的移动应用功能点知识图谱。该图谱不仅可以帮助移动应用自动化测试进一步降低人力、时间成本，同时也可以在移动应用领域提供一份参考的专业知识模型，帮助移动应用的其他领域专家做决策。

3.2 系统需求分析

3.2.1 功能需求

本文是面向移动应用功能点测试的知识图谱构建技术，因此本文通过分析移动应用功能点测试、构建知识图谱两个层面，从用户与系统的角度分析出了多项功能需求如表 3-1所示。表中主要对列出功能需求的具体内容与优先级进行了描述。

3.2.2 非功能需求

面向移动应用功能点测试的知识图谱构建技术需要满足一定的非功能需求。包括易用性，帮助使用该系统的用户快速掌握相关技术并操作。同时还需要包括可扩展性，有助于后续的科研工作。表 3-2列出了各项具体的非功能需求，并对其进行了优先级的描述。

表 3-1: 面向移动应用功能点测试知识图谱构建技术功能需求

需求编号	需求名称	需求内容	优先级
R1	新建图谱	用户将图文相关的众包测试报告上传至系统，系统将通过拆解、组织、联系等步骤将测试报告转化为特定信息保存到新的知识数据库中	高
R2	新增知识	用户将新的到的众包测是报告上传至系统，系统将通过拆解、组织、联系等步骤将测试报告转化为特定的信息，并与只是数据库中的信息进行比对去重，将新增信息添加至原来的只是数据库中	低
R3	添加配置	用户希望在不同的功能点图谱中添加特定的配置，包括文本的近义词、文本的相关词、特定格式的文本、特定布局的图片、特定样式的图片等。仅通过简单的配置即可帮助用户自动化的生成不同功能点的知识图谱	中
R4	查询图谱	用户通过上传应用当前截图到系统的方式，查询当前页面的信息，包括当前页面的控件、控件类型、控件位置、下一步可能需要操作的控件、下一步的逻辑含义等信息。	中
R5	报告拆解	系统可以对众包测试报告中文本的动宾进行拆解；同时对众包测试报告中图片的文本提取、控件提取、控件识别以及布局分析。将拆解出来的数据保存至指定位置供用户使用	高
R6	信息组织	系统将文本拆解信息与图片拆解信息的相关内容绑定起来并将绑定数据保存至指定位置供用户使用	高
R7	建立关系	系统将所有信息按照操作顺序进行排列和整理，建立组织好信息之间的相互联系，并将数据保存至指定位置供用户使用	高
R8	排查冗余	系统可以对当前新增的节点与知识图谱中的节点做冗余分析，排除冗余内容	高

3.2.3 系统用例图及用例描述

本文实现的是面向移动应用功能点测试的知识图谱构建技术。图 3-2则为该系统的系统用例图。

表 3-2: 面向移动应用功能点测试知识图谱构建技术非功能需求

需求编号	需求名称	需求内容	优先级
R9	易用性	接口调用简单，用户可以在 5 分钟内通读接口定义并着手操作使用	中
R10	可扩展性	系统可以在后续方便的引入更细致的拆解解析步骤，让知识图谱的维度更加丰富	中

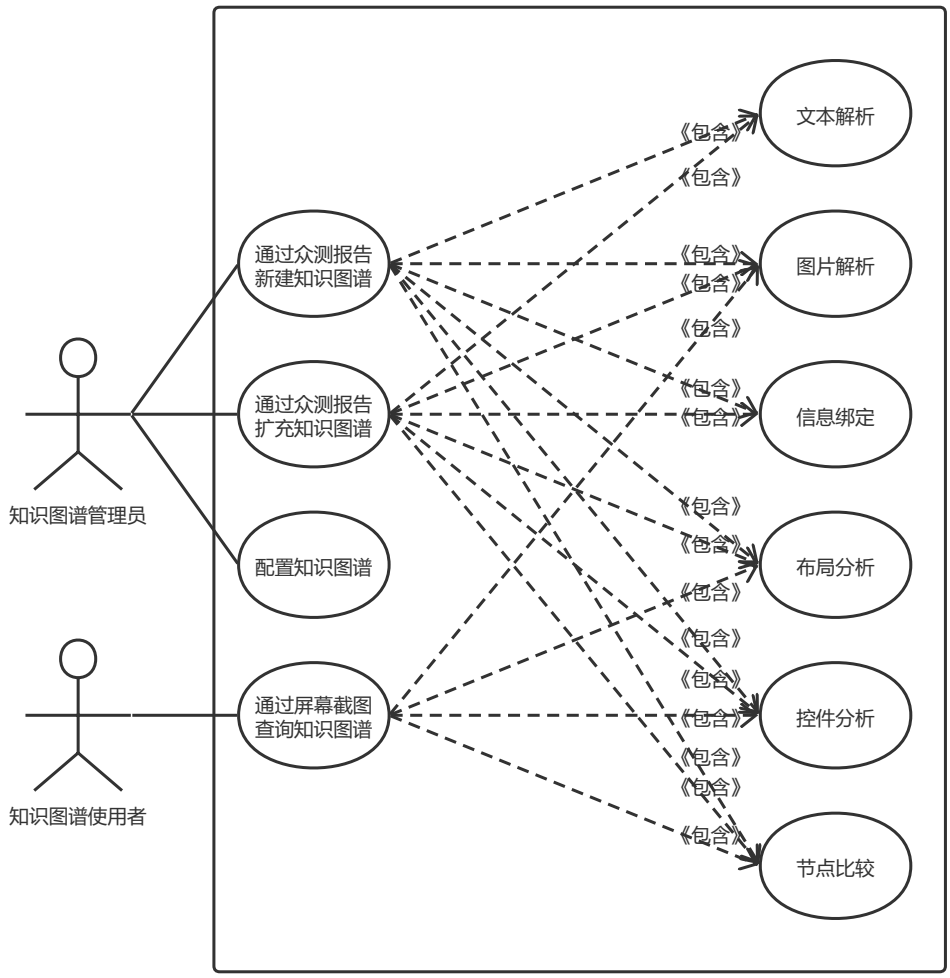


图 3-2: 面向移动应用功能点测试的知识图谱构建技术系统用例图

系统主要分为两个角色，分别为知识图谱管理员及知识图谱使用者。知识图谱管理员即创建、维护知识图谱的管理者。它可以通过众包测是报告新建知识图谱，也可以通过众包测试报告对已有知识图谱进行扩充，同时还可以对不同功能点的知识图谱进行对应的配置。新建和扩充知识图谱的过程还涉及到文本解析、图片解析、信息绑定、布局分析、控件分析、节点比较六个子用例。

知识图谱使用者即其他需要使用知识图谱进行查询访问的用户或系统。其可以通过移动应用的截图进行知识图谱的查询。在知识图谱的查询过程中，还涉及到图片分析、布局分析、控件识别、节点比较四个用例。

如表 3-3所示，系统主要涉及十个系统用例。分别为文本解析、图片解析、信息绑定、布局分析、控件分析、节点比较、通过众测报告新建知识图谱、通过众测报告扩充知识图谱、配置知识图谱以及通过屏幕截图查询中知识图谱。表中对各用例进行编号，同时给出用例与功能需求之间的关系，表的最后一列给出对应的功能需求编号。

表 3-3: 面向移动应用功能点测试知识图谱构建技术系统用例表

用例编号	用例名称	功能需求编号
UC1	文本解析	R1、R2、R5
UC2	图片解析	R1、R2、R5
UC3	信息绑定	R1、R2、R6
UC4	布局分析	R1、R2、R5
UC5	控件分析	R1、R2、R5
UC6	节点比较	R1、R2、R7、R8
UC7	通过众测报告新建知识图谱	R1
UC8	通过众测报告扩充知识图谱	R2
UC9	配置知识图谱	R3
UC10	通过屏幕截图查询知识图谱	R4

下面，本文将通过用例表格对十个用例进行具体的描述。

文本解析是让用户通过提供原始测试报告文本的方式来获取文本的拆解信息，具体信息如表 3-4所示。该用例是为了帮助用户从细节上来控制整个知识图谱的拆解分析流程，并于后续的改进与调整。这里的用户指知识图谱的管理员。用户通过将拥有特定格式的带有图文信息的移动应用众包测试报告存放至指定的文件夹中，然后告知系统原始测试报告的存放文件夹，并设置结果集的存放文件夹，随后调用接口进行分析。系统将对文本进行解析，提取有用信息，分析动宾结构，将结果集存放到指定的文件夹中。

图片解析是让用户通过提供原始测试报告图片的方式来获取图片的拆解信息，具体信息如表 3-5所示。该用例不仅帮助知识图谱管理者从细节上来控制整个知识图谱的构建，同时也帮助了知识图谱的使用者对提供的屏幕截图进行细节分析操作。用户通过将拥有特定格式的带有图文信息的移动应用众包测试报告存放至指定的文件夹中，然后告知系统原始测试报告的存放文件夹，并设

表 3-4: 文本解析需求规格说明

描述项	说明
用例编号	UC1
用例名称	文本解析
参与者	知识图谱管理员
用例描述	用户可以通过提供原始测试报告文本的方式获取文本的拆解信息。包括复现步骤及每一步的操作与控件
前置条件	用户提供符合格式规定的众包测试报告
正常流程	1. 用户将按照规定格式组织的原始众包测试报告（文本内容）放置到特定位置 2. 用户配置原始众包测试报告的存放路径及文本分析结果的存放路径 3. 用户调用接口开始文本分析 4. 系统在结果存放路径中生成文本解析结果
后置条件	文本分析结果存放到指定路径中
异常流程	无

表 3-5: 图片解析需求规格说明

描述项	说明
用例编号	UC2
用例名称	图片解析
参与者	知识图谱管理员、知识图谱使用者
用例描述	用户可以通过提供原始测试报告文本的方式获取图片的拆解信息。包括图片中的文本信息提取、图片中的控件识别
前置条件	用户提供符合格式规定的众包测试报告
正常流程	1. 用户将按照规定格式组织的原始众包测试报告（图片内容）放置到特定位置 2. 用户配置原始众包测试报告的存放路径及图片分析结果的存放路径 3. 用户调用接口开始图片分析 4. 系统在结果存放路径中生成图片解析结果
后置条件	图片分析结果存放到指定路径中
异常流程	无

置结果集的存放文件夹，随后调用接口进行分析。系统将对图片进行解析，包括对图片中文本的提取、图片中组件的识别，然后将结果存放到指定的结果集文件夹中。

信息绑定使用户通过提供 UC1、UC2 提供的众包测试报告拆解信息的数据，对图文离散的相关信息进行绑定，充实信息的内容和维度，如表 3-6所示。信息绑定也是为了帮助用户从细节上来控制知识图谱生成的。该用例的参

表 3-6: 信息绑定需求规格说明

描述项	说明
用例编号	UC3
用例名称	信息绑定
参与者	知识图谱管理员
用例描述	UC1 与 UC2 所拆解的信息是零散的，文本与图片并不相关，信息绑定则将图文之间相关的信息组织绑定起来，充实拆解信息的维度
前置条件	用户已生成文本、图片的拆解信息
正常流程	1. 用户将拆解好的文本与图片信息放置到特定文件夹中 2. 用户配置原始拆解信息的存放路径，并配置绑定后结果集的存放路径 3. 系统进行原始拆解信息的信息绑定，并将结果集存放到指定的存放路径中
后置条件	信息绑定结果存放到指定路径中
异常流程	无

与者为知识图谱管理员。用户先将拆解好的图文数据信息按照特定方式组织并放置到特定的文件夹中。然后用户配置待绑定数据的路径及绑定结果的写出路径并调用接口进行信息绑定。结果将以文件的形式存放到配置的指定文件夹中。

表 3-7: 布局分析需求规格说明

描述项	说明
用例编号	UC4
用例名称	布局分析
参与者	知识图谱管理员、知识图谱使用者
用例描述	通过对截图的分析，提供当前截图的页面组件布局
前置条件	用户提供符合格式规定的移动应用屏幕截图
正常流程	1. 用户将符合规定的移动应用屏幕截图放置到指定的文件夹中 2. 用户配置屏幕截图的存放路径及分析结果的存放结果路径 3. 用户调用接口进行移动应用屏幕截图的页面布局分析 4. 系统分析页面并将结果存放到指定的存放结果路径中
后置条件	页面布局的分析结果存放到指定的路径中
异常流程	无

页面布局是让用户通过提供移动应用屏幕截图的方式获取到当前页面的组件布局信息，具体如表 3-7所示。该用例也是为了帮助用户从细节上掌握知识图谱的构建流程，同时也帮助用户在查询知识图谱中做到结构分析。用户将符

合规定的移动应用屏幕截图放到指定位置并配置存放的路径及结果集的写入路径，并调用分析接口。系统将对页面的布局进行分析，按照行列行列递进的方式将页面划分为特定的布局，将结果写入到指定的路径中完成分析。

表 3-8: 控件分析需求规格说明

描述项	说明
用例编号	UC5
用例名称	控件分析
参与者	知识图谱管理员、知识图谱使用者
用例描述	系统帮助用户识别图片中的控件具体类型
前置条件	提供有效的控件截图
正常流程	1. 用户将控件截图放到特定路径中 2. 用户配置原始图片存放的路径 3. 用户调用分析接口 4. 系统将分析结果写入文件并返回分析结果
后置条件	控件分析结果返回给用户
异常流程	无

控件分析是让用户通过提供控件截图的方式分析出控件的具体类型，例如是按钮还是文本框等，具体信息如表 3-8所示。该用例不仅帮助知识图谱管理员掌控知识图谱的构建流程，还帮助知识图谱的使用者分析自己传递的屏幕截图信息。用户将控件放置到特定路径中，然后配置存放路径并调用分析接口，系统将通过卷积神经网络对图片进行预测并返回组件类型的预测结果。

表 3-9: 节点比较需求规格说明

描述项	说明
用例编号	UC6
用例名称	节点比较
参与者	知识图谱管理员、知识图谱使用者
用例描述	帮助判断两组信息的相似度，避免冗余，完成共指消解的目标
前置条件	用户提供两组特定格式的信息，该信息可以是 UC3 生成的绑定信息，也可以是知识图谱中存放的节点信息
正常流程	1. 用户挑选出两组需要比较的信息，以 json 的方式组织 2. 用户调用接口，传入两组信息 3. 系统分析两组信息的相似度并将结果返回
后置条件	相似度信息返回给用户
异常流程	无

节点比较帮助用户对两组信息的相似度进行判断，具体信息如表 3-9所示。该用例不仅帮助知识图谱管理员掌控构建的细节信息，也帮助了知识图谱的使用者进一步了解自身截图的信息。用户首先挑选出特定的两组信息准备比较，这两组信息需要以 json 的格式进行整理，且内容为 UC3 生成的绑定信息或知识图谱中存放的节点信息。然后用户调用相似度比较接口，系统根据两组信息的文本、图片、控件、布局等一系列方法对两组信息进行比较，并将结果返回给用户使用。

表 3-10: 通过众测报告新建知识图谱需求规格说明

描述项	说明
用例编号	UC7
用例名称	通过众测报告新建知识图谱
参与者	知识图谱管理员
用例描述	知识图谱管理员通过提供原始众包测试报告新建知识图谱
前置条件	提供特定格式的含有图文信息及复现步骤的众包测试报告，用户已成功启动 neo4j 数据库
正常流程	1. 用户将原始测试报告存放到特定路径中，并配置结果路径 2. 用户调用接口生成知识图谱 3. 系统对数据进行拆解、绑定、共指消解，并将结果上传至图数据库中
后置条件	新建图数据库
异常流程	无

通过众测报告新建知识图谱即让用户通过提供合规的众包测试报告的方式来将信息进行拆解、分析、组合绑定、共指消解并上传至图数据库形成知识图谱，具体信息如表 3-10所示。用户只需要将原始的众包测试报告放置特定文件夹中并配置文件夹路径，系统将对文件夹内的所有信息进行拆解、分析、组合绑定、共指消解的操作，并将结果进行编纂并写入到图数据库中，完成知识图谱的创建工作。

通过众测报告扩充知识图谱即让用户在已有知识图谱的情况下，仍能通过新的众包测试报告数据来扩充、完善原来的知识图谱，具体的信息如表 3-11所示。用户需要提供合规的众包测试报告，同时需要打开图数据库并保证数据库中存在知识图谱。将新的众包测试报告放到特定的文件夹下并配置文件的存放路径，然后调用接口，系统将对新的数据进行拆解、分析、信息绑定、共指消解的操作，并于当前知识图谱进行冗余排查，最后将新的节点存入知识图谱当中。

表 3-11: 通过众测报告扩充知识图谱需求规格说明

描述项	说明
用例编号	UC8
用例名称	通过众测报告扩充知识图谱
参与者	知识图谱管理员
用例描述	再已有知识图谱的情况下，仍可以通过新的数据对原有图谱进行扩充
前置条件	提供特定格式的含有图文信息及复现步骤的众包测试报告，用户已成功启动 neo4j 数据库，系统已有知识图谱
正常流程	1. 用户将原始测试报告存放到特定路径中，并配置结果路径 2. 用户调用接口生成知识图谱 3. 系统对数据进行拆解、绑定、共指消解，并将结果上传至图数据库中
后置条件	图数据库得到扩充
异常流程	无

表 3-12: 配置知识图谱需求规格说明

描述项	说明
用例编号	UC9
用例名称	配置知识图谱
参与者	知识图谱管理员
用例描述	用户可以通过配置文件的方式针对不同功能点创建各具特定的知识图谱
前置条件	无
正常流程	1. 用户提编辑 json 格式填写的配置信息 2. 用户调用接口并提供 json 编辑信息 3. 系统修改配置信息
后置条件	系统完成配置信息的修改
异常流程	无

配置图谱用户用户根据不同功能点对知识图谱做个性化的配置工作，具体信息如表 3-12所示。例如对于登录功能点来说我们可能认为“中心”与“我”是近义词，乍一看并不相似，但对于大多数涉及到用户信息的内容都在这两组内容标识的控件标识，这是登录功能点独有的，因此应该单独进行配置。用户可以通过调用接口并提供 json 格式的配置信息来完成配置信息的修改工作。

通过屏幕截图查询知识图谱及通过提供移动应用屏幕截图的方式来获取页面的所有信息，并获知下一步应该对页面做哪些操作，具体信息如表 3-13所示。用户只需要调用接口并提供移动应用的屏幕截图即可获知当前截图总的所有信息，包括图片中的文本信息、组件信息、组件类型信息、页面布局信息、

表 3-13: 通过屏幕截图查询知识图谱需求规格说明

描述项	说明
用例编号	UC10
用例名称	通过屏幕截图查询知识图谱
参与者	知识图谱使用者
用例描述	用户通过提供移动应用的屏幕截图来查询当前页面的信息及下一步可能要操作的控件
前置条件	用户提供移动应用屏幕截图
正常流程	1. 用户调用接口，传入移动应用的屏幕截图 2. 系统分析截图内容并给出下一步可能操作的控件信息列表
后置条件	返回查询信息
异常流程	无

下一步可能需要操作的控件列表信息以及下一步该做什么的逻辑信息。

3.2.4 数据库设计

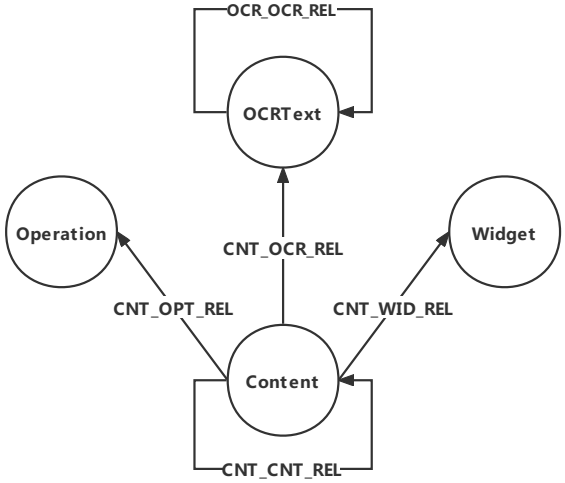


图 3-3: 面向移动应用功能点测试的知识图谱构建技术数据库本体模型图

面向移动应用功能点测试的知识图谱构建技术的数据库设计如图 3-3所示。由于该数据库为非关系型数据库且使用图作为底层存储方式，所以采用本体模型来表示数据库的底层设计结构。可以看到数据库中总共涉及到四个不同的类，分别为 OCRText、Content、Widget 以及 Operation。同时还涉及到五种关系类型，分别为 OCR_ OCR_REL、CNT_OPT_REL、CNT_OCR_REL、

表 3-14: 知识图谱本体模型描述表

类/关系	属性	说明
OCRText	id, name	该类标识针对某个逻辑组件 (即 Content) 而言，它在移动应用中可能出现的所有文本形式。其中 id 唯一标识 OCRText，name 则存放着现实应用中的真实文本内容。
Content	id, name, head, tail	该类标识某个逻辑组件。他表示一个通用的高层概念。其中 id 唯一标识 Content，name 则存放着高层逻辑概念的内容，head 标识着该逻辑概念能不能作为功能点测试的起始步骤，而 tail 则标志着该逻辑概念能不能作为功能点测试的结束步骤。
Widget	id, name, english	该类标识针对某个逻辑组件 (即 Content) 而言，它在移动应用中可能出现的组件类型。其中 id 唯一标识 Widget，name 则存放着该组件类型的中文名称，english 则存放着该组件类型的英文名称。
Operation	id, name, english	该类别标识针对某个逻辑组件 (即 Content) 而言，它在移动应用中可能出现的操作方式。其中 id 唯一标识 Operation，name 则存放着该操作的中文名称，而 english 则存放着该操作的英文名称。
OCR_OCR_REL	relation	该关系标识着两个 OCRText 之间的关系，relation 可以标识为相似，即表示两个 OCRText 被认为相似。
CNT_OPT_REL	relation	该关系标识着 Content 与 Operation 之间的关系，relation 可以标识为操作，即表示对于该概念类型应该执行这种操作。
CNT_OCR_REL	relation	该关系标识着 Content 与 OCRText 之间的关系，relation 可以标识为文本，即表示对于该概念类型存在一种现实的文本表现形式。
CNT_WID_REL	relation	该关系标识着 Content 与 Widget 之间的关系，relation 可以标识为控件，即表示对于该概念类型所依赖的组件的组件类型。
CNT_CNT_REL	relation	该关系标识着两个 Content 之间的关系，relation 可以标识为相似，即表示两个 Content 被认为相似。

CNT_WID_REL、CNT_CNT_REL。表 3-14对这四个数据类及五个关系类进行了详细的描述。

3.3 系统概要设计

3.3.1 系统总体设计

面向移动应用功能点测试的知识图谱构建技术的系统结构如图 3-4所示。整个系统由前端服务器、后端服务器、数据文件系统三部分组成。

前端服务器中主要包含三个任务，分别是构建知识图谱模块、知识图谱智

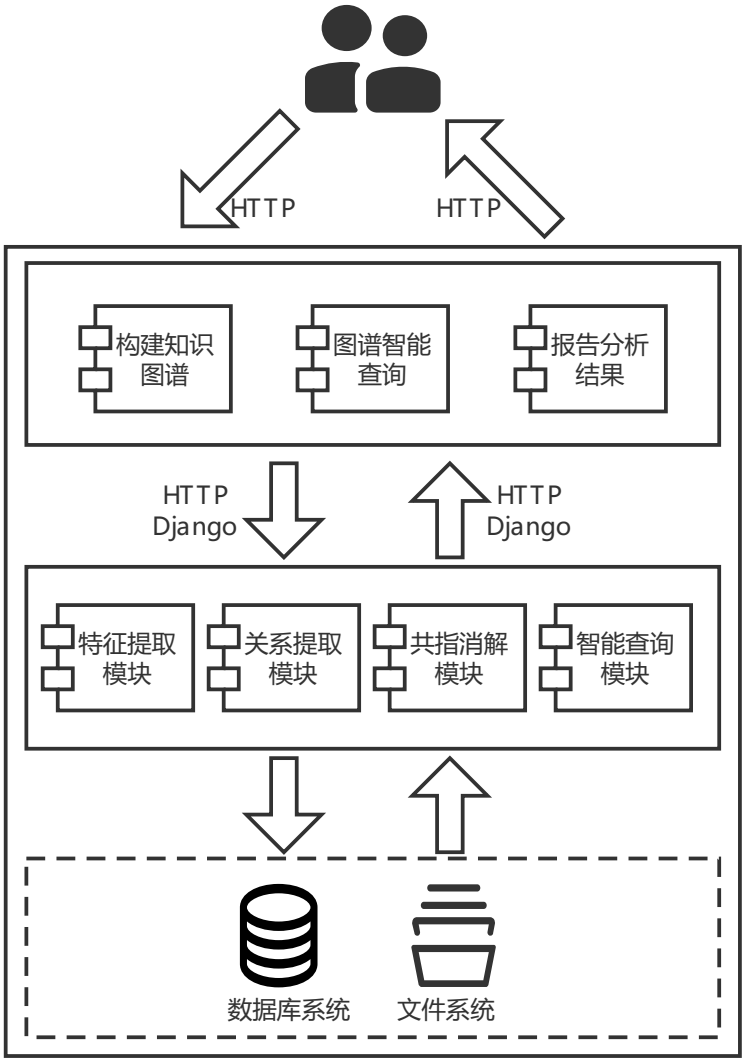


图 3-4: 面向移动应用功能点测试的知识图谱构建技术系统结构图

能查询模块以及报告分析结果查看模块。通过这三个模块的实现帮助用户构建并查阅知识图谱。

后端服务器中主要包含四个任务，分别是知识图谱特征提取模块、知识图谱关系提取模块、知识图谱共指消解模块以及知识图谱智能查询模块。通过这四大模块将知识图谱的增删改查以细粒度的方式供用户选择使用，帮助用户进一步掌控知识图谱的生成细节。

后端服务器通过以图为存储方式的图形数据库 Neo4j 作为数据存储中心，帮助存储知识图谱，同时使用文件系统存储创建知识图谱的中间文件，从而保证了系统的正确运行。

图 3-5从高维度对面向移动应用功能点测试的知识图谱构建技术的系统部

署情况进行了具体说明。系统通过三大模块结合起来，分别为系统前端服务器、系统后端服务器和数据库服务器。其中系统前端服务器通过 Vue 实现，通过 iView 组件库帮助系统的搭建，并通过 HTTP Django 对系统后端服务器做数据获取操作。系统后端服务器通过 Python3 进行实现，并用 Django 框架作为容器运行代码。数据库服务器则通过 Neo4j 作为存储数据库，这种图数据库很好的保证了知识图谱的正确存储与高效查询。同时兼具文件系统帮助系统进行逻辑决策。后端服务器与数据库服务器之间通过 TCP/IP 的方式进行通信，以完成整个系统的工作。

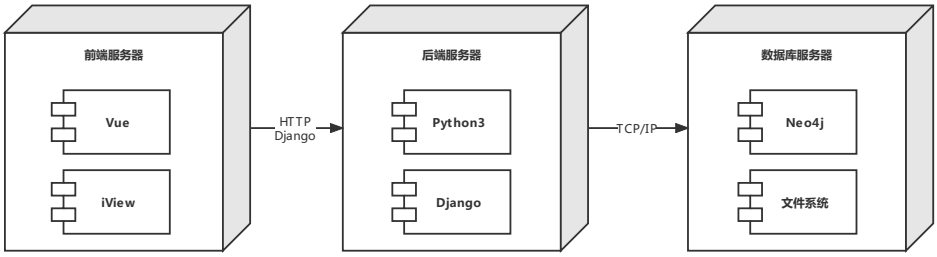


图 3-5: 面向移动应用功能点测试的知识图谱构建技术系统部署视图

根据系统架构图（图3-4）与系统部署视图 (图3-5) 我们可以清楚的看到整个后端系统主要分为四个模块，分别为知识图谱的特征提取模块、知识图谱的关系提取模块、知识图谱的共指消解模块与知识图谱的智能查询模块。下面，我们将对这四个模块进行具体的模块设计说明。

3.3.2 知识图谱特征提取模块设计

知识图谱特征提取模块是为了创建知识图谱做的前期准备工作。该模块主要负责将原始大量的带有复现步骤的移动应用众包测是报告进行有效信息的提取和拆解工作，即一种特征提取的过程。通过将众包测试中的复现步骤文本描述与移动应用故障截图进行分析的方式来帮助知识图谱的创建与执行。

知识图谱特征提取模块需要以功能点为基础，提供图文相关的、带有复现步骤的移动应用众包测试报告，一个完整的众包测试报告如图 3-6所示。可以看到众包测试报告中包含两个大模块，一个是对于故障产生时的文本描述，一个是对应于复现步骤操作的屏幕截图。最左侧的文本报告中的四条复现步骤分别对应于自左起展示的四张屏幕截图。这里是一份有关登录的众包测是报告，

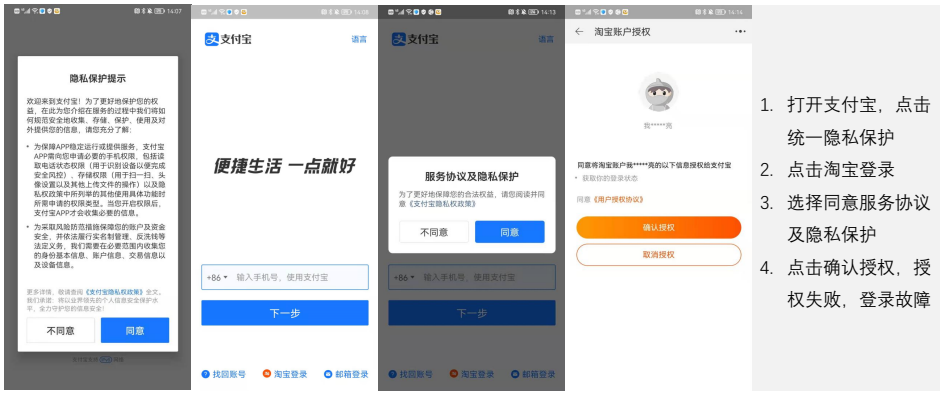


图 3-6: 输入知识图谱特征提取模块的有效众包测试报告样例

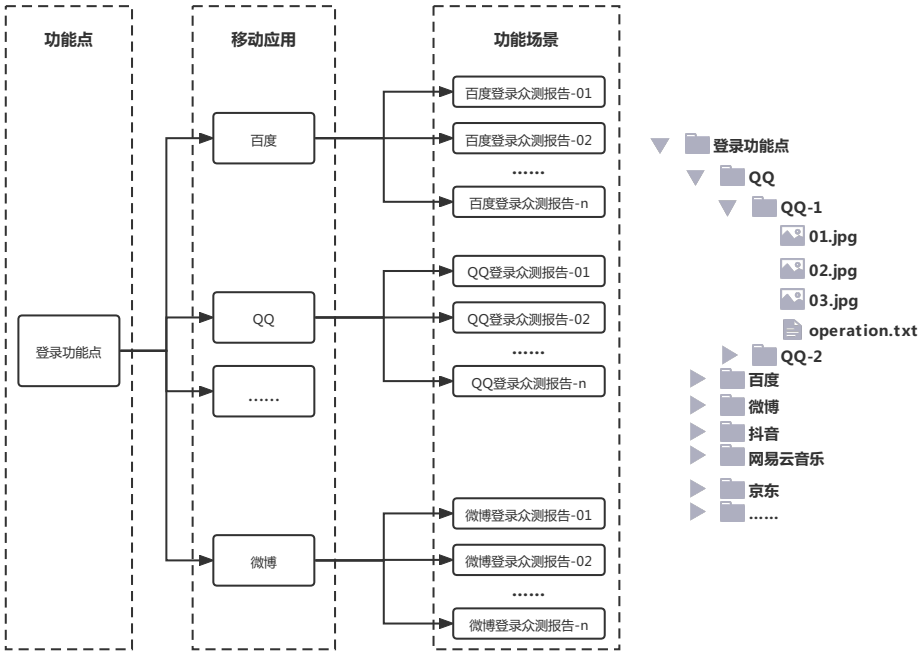


图 3-7: 知识图谱特征提取模块输入数据格式

即可以作为登录功能点知识图谱的原始数据使用。我们通过收集大量的有效的众包测试报告数据，将数据按照树形结构以功能点、移动应用、功能场景进行组织整理，提供给知识图谱特征提取模块，这种组织结构如图 3-7所示。

图 3-7左侧部分展示了原始众包测试报告的组织结构。例如我们现在需要构建登录功能点的知识图谱，我们首先需要创建一个文件夹用于存储所有登录功能点数据，即图中的功能点模块。然后我们在该文件夹下根据不同的应用创建其相应的文件夹，例如图中展示的百度、QQ 等应用，该模块为移动应用模

块。之后则需要再细粒度的将场景放置再移动应用模块的文件夹中。例如对于百度来说，总共收集到了 *n* 份登录的众包测试报告，则在登录功能点/百度路径下创建百度登录众测报告-01、百度登录众测报告-02、……边度登录众测报告-n，并将 *n* 份测试报告分别放置于这 *n* 个文件夹中。每个众包测试报告的文件夹中将复现步骤的文本保存在 operation.txt 中，每个步骤对应的屏幕截图则按照需要 1 到 *n* 进行排列命名。图 3-7右侧部分则给出了一个原始众包测试报告组织结构的具体实例。

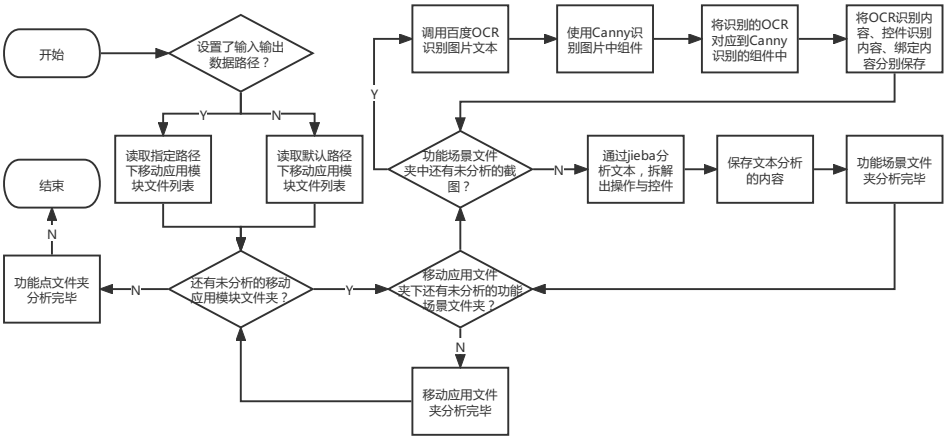


图 3-8: 知识图谱特征提取模块流程图

图 3-8展示了知识图谱特征提取模块的流程图。首先用户将整理好的存放着有效的带有复现步骤的图文众包测试报告原始数据存放到指定文件夹或默认文件夹，然后调用系统进行数据分析。数据分析分为三个层次依次分析，依次为移动应用层面、功能场景层面以及图片文本层面。首先从用户指定的或默认的路径中读取到所有的移动应用模块文件夹，然后循环对每个文件夹进行分析操作。对于每个移动应用模块文件夹，读取其内所有的功能场景文件夹，并循环分析功能场景文件夹。对于每个功能场景文件夹，则首先通过百度 OCR、Canny 算法、Merge 方法将文件夹中的所有屏幕截图进行分析处理，接着通过 jieba 分析文本内容，并将图片与文本的分析结构都进行保存。在循环都结束之后即完成原始数据的拆解分析工作。具体内容如图 3-8所示。

3.3.3 知识图谱关系提取模块设计

知识图谱关系提取模块是为了将知识图谱特征提取模块中分析出来的数据进行整合。由于知识图谱特征提取模块仅通过各种算法和方式分别将文本与图片进行了特征的提取，但并没有将文本与图片间的内容联系起来，因此通过知识图谱关系提取模块的设计来将相关的图文部分整合起来。这样可以让信息转化为知识，并帮助后续知识图谱的形成。

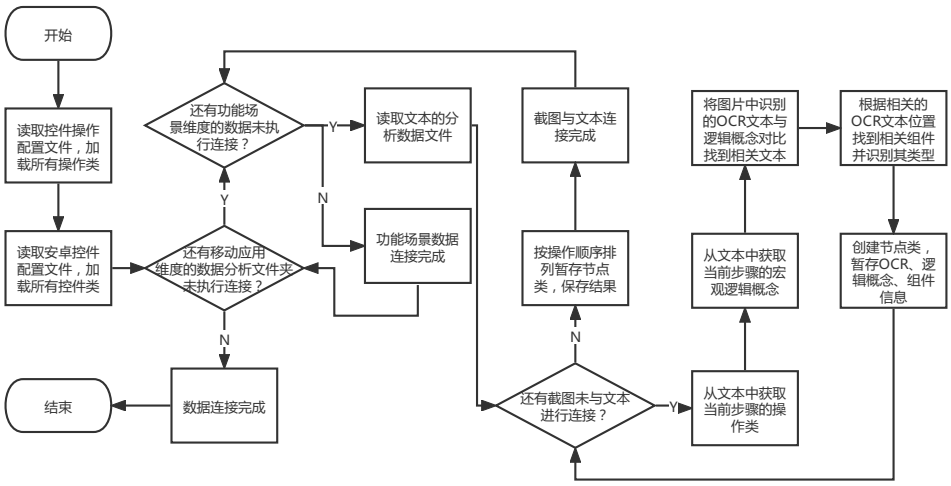


图 3-9: 知识图谱关系提取模块流程图

图 3-9为知识图谱关系提取模块的流程图。系统仍然按照三个维度来连接信息。首先，由于用户操作即控件类型有特定的数目，因此这些内容会当作配置文件分别记录在一个 CSV 文件中，系统首先加载这两个文件中的内容，将操作类及控件类都加载进内存。然后系统将针对该功能点下的每一个移动应用模块的文件夹进行连接操作。循环连接每个移动应用文件夹中的所有功能场景文件夹。针对每个功能场景文件夹，首先通过识别文本提取信息将操作类、宏观逻辑概念提取出来，接着循环连接每一张图片与文本内容：先从图片中的文本中找到与宏观概念相似的截图中的文本作为 OCR 信息，并找到跟 OCR 相似的所有其他 OCR，然后根据 OCR 信息找到文本的出现位置进而找到对应组件信息，最后通过 CNN 对组件惊醒类型识别确定组件类，最后创建节点类将操作、逻辑、OCR、组件、组件类别信息均存入其中，并按照操作顺序依次排列，最终写入文件保存到指定文件夹中。具体内容如图 3-9所示。

3.3.4 知识图谱共指消解模块设计

知识图谱共指消解模块是为了将知识图谱特征提取模块与知识图谱关系提取模块整理好的数据上传到指定的 Neo4j 图数据库中。知识图谱共指消解模块会根据数据的不同类型创建不同的节点，并创建节点之间的联系。同时，在生成或扩充知识图谱的过程当中，还需要对新增的节点与知识图谱内的原有节点进行比较与分析，完成共指消解的操作，保证知识图谱无冗余且可靠。

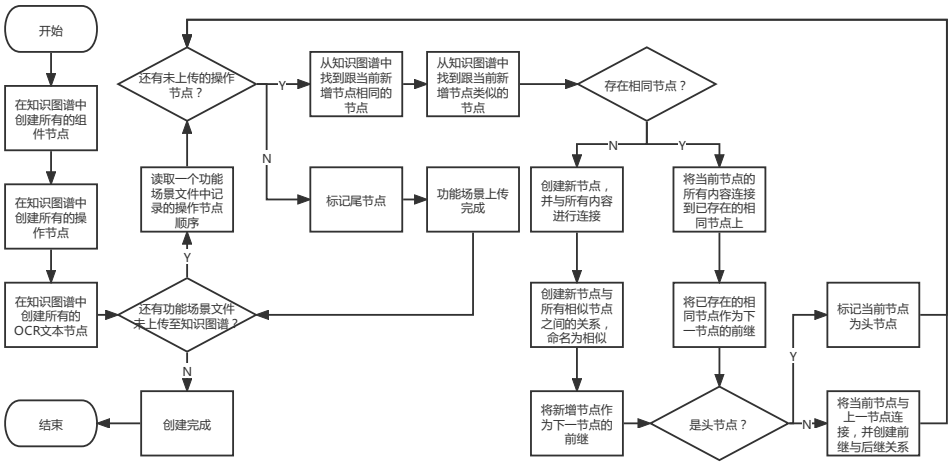


图 3-10: 知识图谱共指消解模块流程图

图 3-10即为知识图谱共指消解模块的流程图。用户将知识图谱关系提取模块产生的数据放到指定位置，系统则首先通过知识图谱关系提取模块产生的操作节点类、控件节点类、OCR 文本节点类将数据上传至知识图谱。其中，OCR 文本节点类之间存在相似或者相关的关系，在知识图谱关系提取模块中都有明确的标识，知识图谱共指消解模块需要在这些相似的 OCR 文本之间添加关系，意为相似。之后系统则读取知识图谱关系提取模块生成的所有功能场景维度的文件，其中记录了多个节点之间的线性知识。循环根据每一个功能场景维度的文件进行上传创建的工作。针对每个功能场景维度文件，系统会按照顺序上传文件中顺序记录的节点。在上传的过程中，系统会将节点之间的前后继顺序添加至图谱中，同时还会寻找节点的相同节点排除冗余，找到节点的相似节点建立联系称之为相似，并标记每一个功能场景中的头节点与尾节点，将获取到的知识信息最大程度的保留在知识图谱当中。具体内容如图 3-10所示。

3.3.5 知识图谱智能查询模块设计

知识图谱智能查询模块是为了帮助使用者快速了解知识图谱，并帮助移动应用自动化测试人员提供自动化功能点测试下一步该做什么的指引帮助。用户可以选定功能点并提供移动应用屏幕截图，系统将通过查询系统分解图片中的内容并告知用户下一步能做哪些逻辑操作，这些操作所设计到的控件，如果操作这些控件等细节描述。

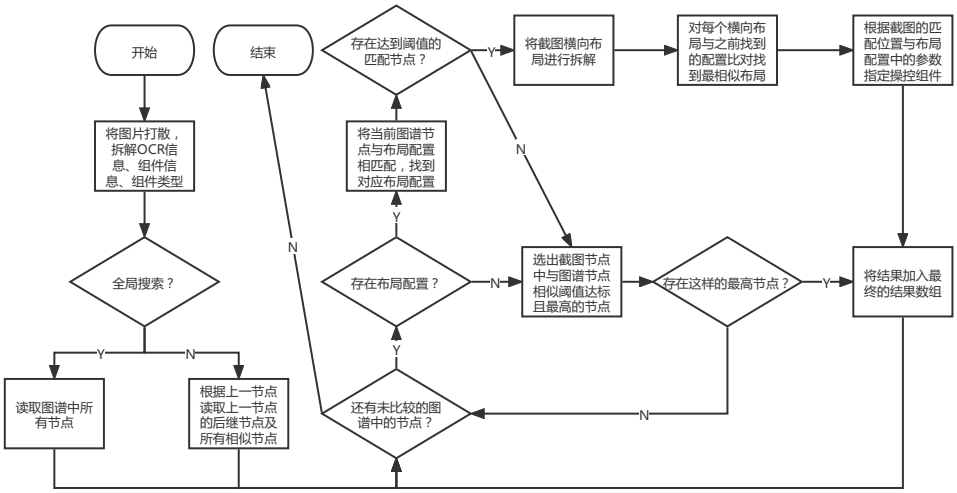


图 3-11: 知识图谱智能查询模块流程图

图 3-11即为知识图谱智能查询模块的流程图。知识图谱的查询分为两种模式，一种是全局搜索模式，该模式会将图谱中的所有节点都进行遍历查询，范围广但时效差；另一种方式是头部搜索，通过给定某个特征来帮助知识图谱缩小比较范围，加快查找速度。在根据选择的搜索方法获取到相应的图谱节点之后，系统将循环判断每个知识图谱节点是否适用。针对每个知识图谱节点，系统将先根据布局配置文件查询该逻辑概念下的节点是否需要按照布局来进行配置，如果需要则首先将截图按照横向布局进行分解，然后将分解出来的每个部分与配置文件中表明的特征进行比对，选择最相似的部分作为选中的布局模块，并按照配置的位置与操作信息将该布局内的特定组件作为结果记录。如果没有布局配置的限制或者在比较中没有发现达到阈值的布局配置，则该知识图谱节点将与查询截图中的所有组件节点进行比较，找到最相似的节点作为结果记录。

3.4 本章小结

本章首先对面向移动应用功能点测试的知识图谱构建技术进行了项目的概述，并分析了其相应的功能需求与非功能需求。我们提供了系统的用例图，并将用例与需求进行了匹配，且针对各用例进行了详尽的规格说明。之后我们提供了系统的数据本体模型来详细说明系统的存储方式，并提供了系统的整体架构图、部署视图来进一步了解整个系统。我们通过将整个系统划分为知识图谱特征提取模块、知识图谱关系提取模块、知识图谱共指消解模块和知识图谱智能查询模块的方式来对整个系统的后端服务进行细节阐述。其中，为各模块提供了流程图以便理解整个开发流程。

第四章　系统详细设计与实现

4.1　知识图谱特征提取模块

知识图谱特征提取模块作为知识图谱构建的第一个模块，在整个知识图谱的构建过程中起到引领的作用。知识图谱特征提取模块会对有效的含有复现步骤的移动应用众包测试报告数据进行拆解工作，并从中提取有用的特征信息，保存到指定位置供后续模块使用。知识图谱特征提取模块的顺序图如图 4-1所示。

4.1.1　知识图谱特征提取模块详细设计

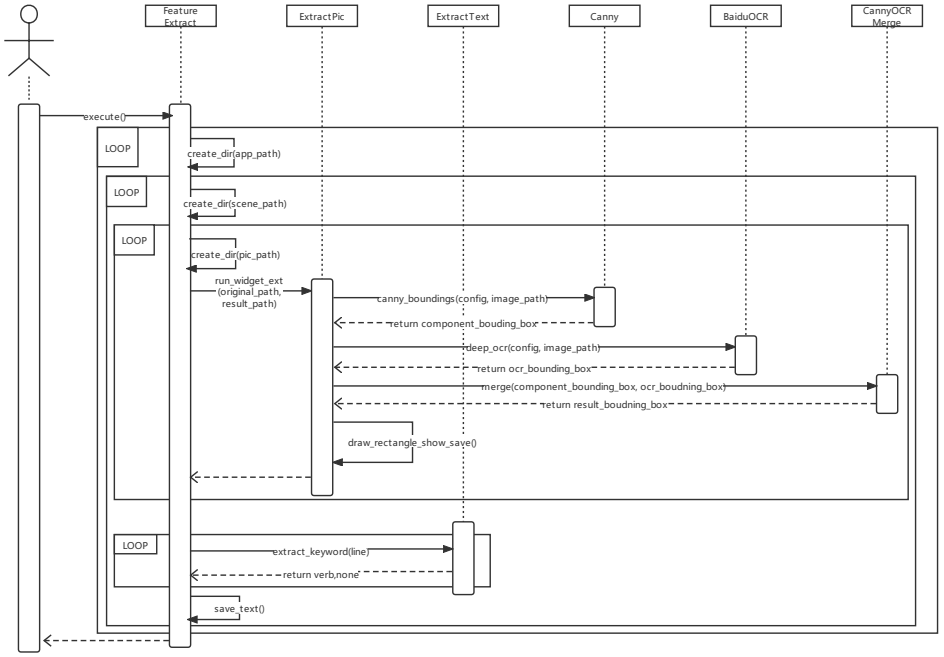


图 4-1: 知识图谱特征提取模块顺序图

知识图谱特征提取模块的顺序图中总共涉及到用户及六个类的调用操作，这六个类的具体信息如表 4-1所示。表中所列出的六个类分别是 FeatureExtract、ExtractPic、ExtractText、Canny、BaiduOCR 以及 CannyOCRMerge。其中

FeatureExtract 是与用户交互的类。该类中为用户提供了操作知识图谱特征提取模块的上层入口，用户仅需将符合规定的众包测试报告放置到指定的文件夹中并对其放置位置与导出结果位置进行配置，便可以调用该类的方法进行知识图谱特征提取的整个步骤。ExtractPic 与 ExtractText 则为两个独立进行特征提取的组件。ExtractPic 用于对图片进行特征提取操作。该类可以对图片中的文本与组件进行识别，将图片信息转化为文本信息保存。ExtractText 则可以对文本信息进行特征提取，该类可以分析文本信息的语法结构，从文本中提取出动词、名词等信息，并将其与测试过程中的操作、内容相对应。该类主要通过 jieba 词库进行设计与实现。除此以外，还涉及到 Canny、BaiduOCR、CannyOCRMerge 三个类，这三个类都是为了 ExtractPic 而生的。ExtractPic 中的组件识别技术依赖于 Canny 来表示图片中的边框信息从而识别出组件。ExtractPic 中的文本识别技术则依赖于 BaiduOCR 提供的百度 OCR 接口从而识别图片中的文本信息。CannyOCRMerge 则首先通过 Canny 与 BaiduOCR 进行组件与文本的识别，然后根据组件与文本的坐标进行二者交集的计算，从而将识别出来的文本对应匹配到识别出来的组件当中，帮助后续模块继续使用特征提取的信息。

表 4-1: 知识图谱特征提取模块操作类功能说明

编号	类名	功能
001	FeatureExtract	知识图谱特征提取模块的启动类，用户可以通过调用该类中的接口对指定文件夹中的移动应用众包测试报告数据做特征提取工作，并将结果保存到指定位置。
002	ExtractPic	该类用于对图片进行特征提取工作。特征提取过程中可以识别图片中的所有组件，并识别出图片中的所有文本信息。
003	ExtractText	该类用于对文本进行特征提取工作。特征提取过程中可以识别一段文字中的动词、宾语，并将其按照操作、内容的方式进行返回。
004	Canny	该类用于对图片中的组件进行识别标记操作。该类可以对图片进行学习和分析，并识别出哪些线条、样式表示的是同一个组件，并标明组件的坐标。
005	BaiduOCR	该类用于对图片中的文本进行提取分析。通过百度 OCR 接口进行识别，返回结果。
006	CannyOCRMerge	该类将 Canny 识别的组件坐标信息与百度 OCR 识别的图片文本信息进行组织绑定，将文本添加到对应的组件坐标信息中。

图 4-1中即通过上述阐明的六个类与用户的相互协作完成了整个知识图谱特征提取模块的操作。首先用户需要对整个操作流程进行配置，包括配置移动应用众包测试报告位置，结果导出的位置，使用的是哪个功能点的知识图谱，

以及一些相似度相关的参数配置。之后用户可以开始调用 FeatureExtract 中的 execute 方法，该方法为知识图谱特征提取模块的总入口，用户在调用该方法之后只需等待结果生成至指定文件夹即可。

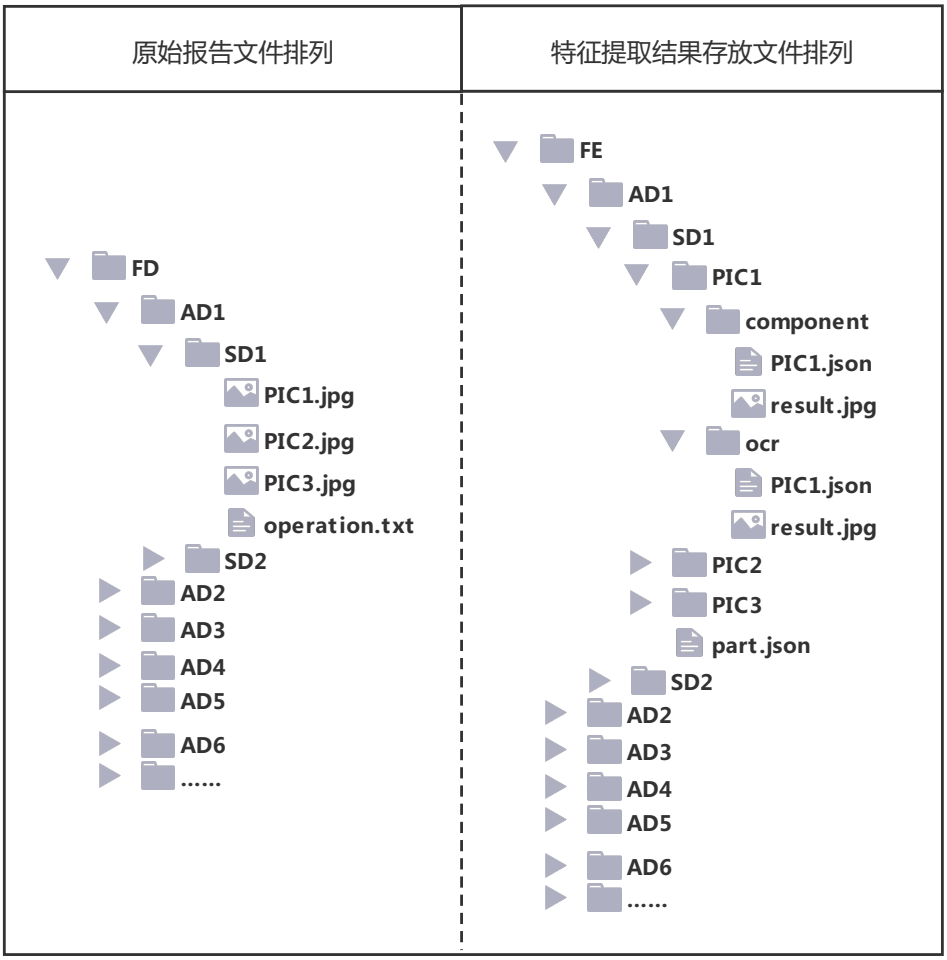
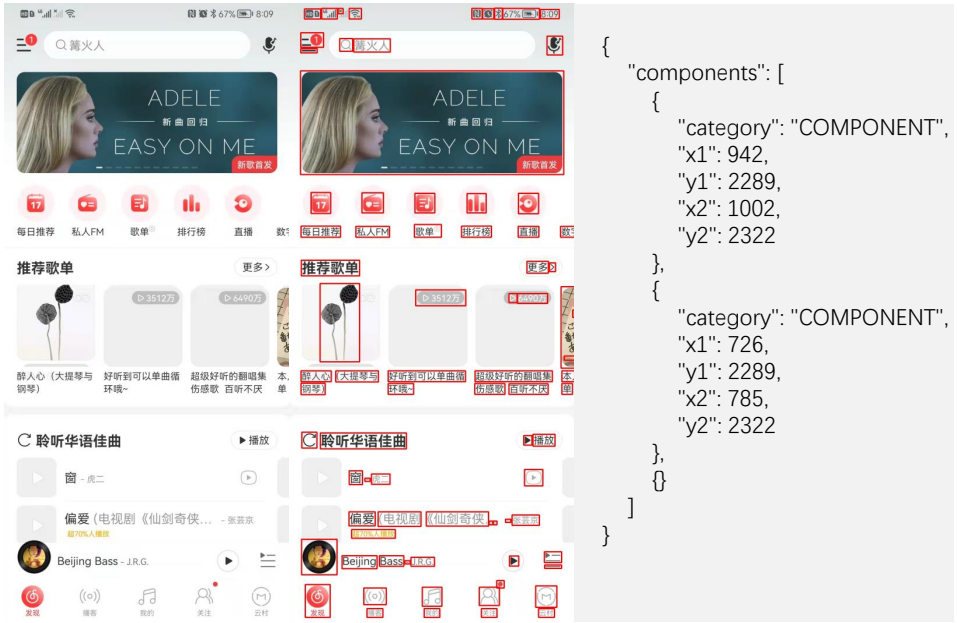


图 4-2: 知识图谱特征提取模块原始文件与结果文件排列方式

在用户调用过 execute 方法之后，FeatureExtract 会根据众包测试报告的文件排列新建文件夹。在第三章中，我们对众包测试报告的组织形式进行了说明，首先会按照功能点创建文件夹 (记作 FD)，之后会在各功能点文件夹中根据移动应用名称创建文件夹 (记作 AD)，接下来会在移动应用名称的文件夹下根据不同的场景创建文件夹 (SD)，在场景文件夹下则存放着移动应用众包测试报告的具体内容。假设我们将原始测试报告存放在 FD1 文件夹中，该文件夹中存放着若干个移动应用文件夹，记作 AD1, AD2, ... , ADn，每个移动应用文件夹 AD 中又存放着多个场景文件夹记作 SD1, SD2, ... , SDn；同时我们希望

特征提取之后的内容存放在 FE 文件夹中。而 FeatureExtract 的工作则是循环的对文件夹中的内容进行提取操作。首先 FeatureExtract 将读取 FD/AD1 文件夹，并在 FE 文件夹中创建一个新的 AD1 文件夹以存放特征提取到的信息，即创建 FE/AD1。接着，FeatureExtract 会读取 FD/AD1/SD1，并在 FE/AD1 中新建 SD1 文件夹以存放对于场景文件夹 FD/AD1/SD1 中的众包测试报告特征提取之后的信息，即创建 FE/AD1/SD1。在创建好文件夹之后，则可以开始具体的特征提取流程，并将结果保存到指定的特征提取文件夹中。循环创建、提取，即可将所有原始众包测试报告中的信息提取出来。图 4-2展示了原始测试报告文件排列方式与对应的特征提取结果文件的存放方式。

在创建好具体的特征提取信息存放文件夹之后，则进入到了特征提取的主要环节。首先对于某个场景文件夹，例如 FD/AD1/SD1 文件夹中，会存放着一个 operation.txt 文件，用于存放该众包测试报告的复现步骤的文本描述。如果 operation.txt 文件中存在 n 条复现步骤，则在 FD/AD1/SD1 文件夹中则会存放着 n 张对应复现步骤的屏幕截图。因此，FeatureExtract 即会提取众包测试报告中图片的信息，也会提取众包测试报告中包含复现步骤的文本信息。



原始屏幕截图 PIC1 **标注组件的屏幕截图 PIC1/component/result.jpg** **截图的组件坐标信息 PIC1/component/PIC1.json**

图 4-3: 知识图谱特征提取模块图片组件提取截图与信息展示

对于 FD/AD1/SD1 中的某张图片 PIC1，系统将首先在 FE/AD/SD1 文件夹中创建 PIC1 文件夹及创建 FE/AD1/SD1/PIC1 文件夹，用于存放对 PIC1 特征

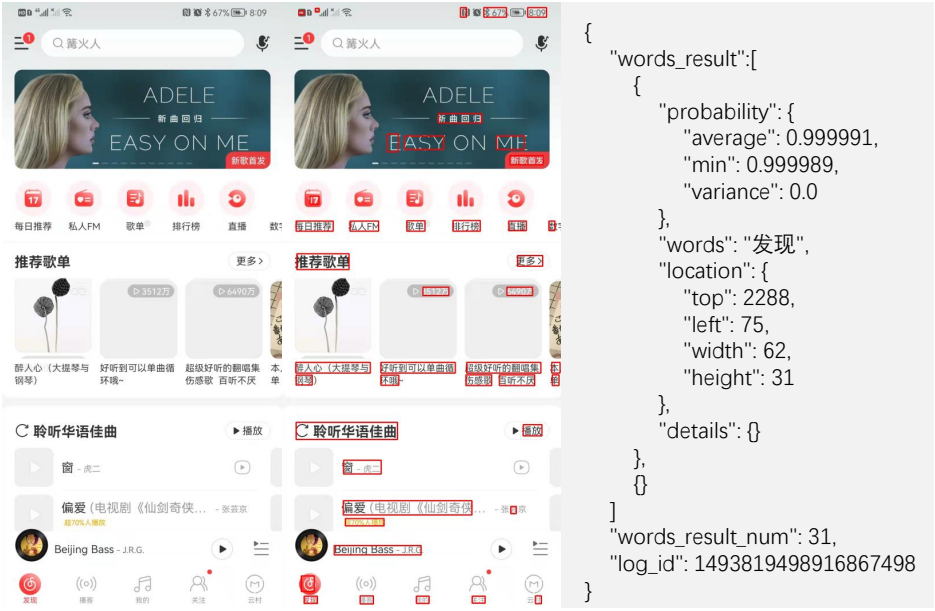


图 4-4: 知识图谱特征提取模块图片文本提取截图与信息展示

提取到的信息。在创建文件夹之后，FeatureExtract 会首先调用 ExtractPic 中的 run_widget_ext 方法，在告知 ExtractPic 原始图片路径 (FD/AD1/SD1/PIC1) 和结果存放路径 (FE/AD1/SD1/PIC1/) 之后，ExtractPic 则开始对图片进行特征提取工作。图片的特征提取过程主要囊括三个步骤。第一步，ExtractPic 调用 Canny 中的 canny_boundings 方法，将图片的原始路径 (FD/AD1/SD1/PIC1) 传递给 Canny，Canny 通过 canny 算法识别出图片中的组件的边框信息，从而标记出来图片中的组件坐标，并将识别出来的截图中的所有组件坐标信息返回个 ExtractPic，该结果记作 component_bounding_box。第二步，ExtractPic 则调用 BaiduOCR 的 deep_ocr 方法，将图片的原始路径 (FD/AD1/SD1/PIC1) 传递给 BaiduOCR，BaiduOCR 则调用百度 OCR 接口将图片中的文本信息全部识别出来，同时标注了识别出来的准去率，识别出文本的坐标位置等信息，将结果返回给 ExtractPic，记作 ocr_bounding_box。第三步，ExtractPic 将得到的组件坐标结果信息 component_bounding_box 及 ocr 坐标信息 ocr_bounding_box 传递给 CannyOCRMerge，调用 merge 方法，将组件与 OCR 文本匹配起来，即找出这些 OCR 文本本来存放在哪个组件当中，将他们联系配对，并将结果返回给 ExtractPic。至此，ExtractPic 获得了图片 PIC1 的所有特征提取信息，他调用自己的 draw_rectangle_show_save 方法，首先

在结果文件夹 FE/AD1/SD1/PIC1/中创建 component 与 ocr 文件夹，即创建 FE/AD1/SD1/PIC1/component 与 FE/AD1/SD1/PIC1/ocr 文件夹，并将组件坐标信息 component_bounding_box 以 json 的格式保存到 FE/AD1/SD1/PIC1/component/PIC1.json 中，将 ocr 文本提取信息 ocr_bounding_box 信息以 json 的格式保存到 FE/AD1/SD1/PIC1/ocr/PIC1.json 中。同时，按照组件坐标的信息将 PIC1 中的组件用红线标注并将图片结果保存到 FE/AD1/SD1/PIC1/component/result.jpg 中，按照 ocr 文本的坐标信息将 PIC1 中的 ocr 文本用红线标注并将图片结果保存到 FE/AD1/SD1/PIC1/ocr/result.jpg 中。图 4-3则展示了 ExtractPic 在组件特征提取之后生成的文件与截图，图4-4则展示了 ExtractPic 在文本特征提取之后生成的文件与截图。在将该场景 FD/AD1/SD1 中的所有图片都特征提取并将结果保存之后，即可进入文本的特征提取环节。

```
1. 点击 三横挂件
2. 点击 “立即登录”
3. 点击勾选 “同意《用户协议》《隐私协议》《儿童隐私政策》《中国移动认证服务协议》”
4. 点击 “一键登录”
```

operation.txt

```
[
  ["点击", "横挂件"],
  ["点击", "立即登录"],
  ["点击勾选", "同意《用户协议》《隐私协议》《儿童隐私政策》《中国移动认证服务协议》"],
  ["点击", "一键登录"]
]
```

part.json

图 4-5: 知识图谱特征提取模块文本提取文件展示

在文本的特征提取环节中，FeatureExtract 首先会读取 operation.txt 中的复现步骤，然后针对复现步骤中的每条步骤，循环进行特征提取操作。针对某一条复现步骤，FeatureExtract 会调用 ExtractText 中的 extract_keyword 方法，将该条复现步骤传递给 ExtractText 中，ExtractText 则通过 jieba 词库的解析分析出文本中的动词与名词，并将动词与名词对应到操作、操作内容，并将结果返回。在完成对整个 operation.txt 中的复现步骤进行特征提取操作后，FeatureExtract 将结果以 json 形式组织起来并保存在 FE/AD1/SD1/part.json 文件中。图 4-5展示了某一 operation.txt 文件经过文本特征提取操作之后生成的 part.json 文件样例。至此图片和文本的特征提取工作就全部结束了。

在对所有的移动应用、每个移动应用的所有场景、每个场景内的所有图片、文本进行特征提取操作之后，特征提取模块的工作就正式结束了。最终的提取结果则按照图 4-2的方式组织存储。

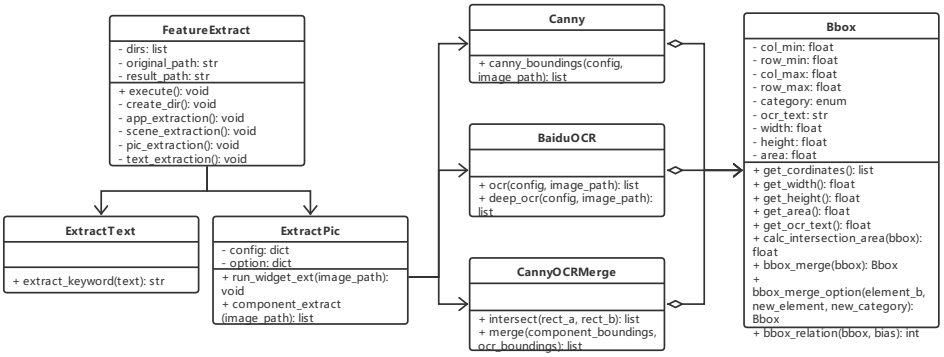


图 4-6: 知识图谱特征提取模块类图

图 4-6则展示了知识图谱特征提取模块的类图设计与实现。该类图中主要涵盖七个类，分别是 FeatureExtract、ExtractText、ExtractPic、Canny、BaiduOCR、CannyOCRMerge、Bbox。其中，前六个类为操作类，具体信息见表 4-1, 其他类为数据类，具体信息如表 4-2所示。

表 4-2: 知识图谱特征提取模块数据类详细说明

编号	类名	功能
007	Bbox	该类用于标记识别出来的组件信息，包括组件的坐标信息，组件的面积信息，组件中所包含的 OCR 文本信息等。

在类图 4-6中我们可以看到 FeatureExtract 类持有知识图谱特征提取模块的原始报告路径、结果存放路径的属性信息，同时它可以通过调用 ExtractPic 与 ExtractText 来实现对于图片及文本的特征提取。ExtractText 通过 jieba 词库对文本信息做特征提取，ExtractPic 则通过 Canny 对图片进行组件的识别，通过 BaiduOCR 对图片进行文本的识别，再通过 CannyOCRMerge 对获得的组件与 OCR 文本信息进行融合。其中，Canny、BaiduOCR、CannyOCRMerge 的返回数据集均为 Bbox 的列表，他们与 Bbox 属于聚合关系，通过将数据整理成 Bbox 列表的形式整理结果返回给上级，完成知识图谱特征提取的工作。

4.1.2 知识图谱特征提取模块代码实现

图 4-7则展示了知识图谱特征提取模块 FeatureExtract 主类中 execute 启动入口。用户仅需调用该入口即可启动特征提取操作。该入口会读取文件夹下移动应用信息，并对每个移动应用进行分析，完成特征提取。

```
# 知识图谱特征提取模块启动入口
def execute(self):
    # 遍历所有涉及到的移动应用文件夹，并逐一对其进行特征提取操作
    for app in self.__dirs:
        self.__app_extraction(app)
```

图 4-7: 知识图谱特征提取模块 FeatureExtract 类启动入口

```
# 对某一移动应用文件夹内的所有内容做特征提取
def __app_extraction(self, app_name):
    # 1.创建用于存放结果的移动应用级别文件夹
    self.__create_dir(self.__result_path + '/' + app_name)

    # 2.针对移动应用文件夹中的所有场景文件夹做特征提取操作
    scenes = os.listdir(self.__original_path + '/' + app_name)
    for scene in scenes:
        self.__scene_extraction(app_name + '/' + scene)
    return
```

图 4-8: 知识图谱特征提取模块 FeatureExtract 类应用分析

图 4-8则展示了 execute 入口方法需要调用的 __app_extraction 方法。该方法会为每个移动应用创建一个用于存放结果的新文件夹，并读取原始移动应用文件夹下的所有场景文件夹，交由 __scene_extraction 方法进行分析。

```
# 对某一场景文件夹内的所有内容做特征提取
def __scene_extraction(self, path):
    # 1.创建用于存放结果的场景级别文件夹
    self.__create_dir(self.__result_path + '/' + path)

    # 2.针对场景文件夹中的所有屏幕截图做特征提取
    pics_path = os.listdir(self.__original_path + '/' + path)
    for pic_path in pics_path:
        self.__pic_extraction(self.__original_path + '/' + path + "/" +
                                                    pic_path)
```

图 4-9则展示场景特征提取的实现步骤。对场景中的每张图片做特征提取，并对场景中的报告文本做特征提取，提取的结果均保存在新创建的文件夹中。

图 4-10则展示了对于屏幕截图进行特征提取的代码实现。该方法主要调用图片的分析入口对图片的控件、文本进行提取分析。

图 4-11则展示了对于报告文本的提取方法。该方法通过调用 jieba 集成的文本词性分析方法对文本进行拆解和识别操作。

```
        # 3.对每一步复现步骤做文本的特征提取，并将结果写入part.json文件
        operation_file = open(self.__original_path + '//' + path + '/
                                                operation.txt', encoding='utf-
                                                8',mode='r')
        line = operation_file.readline()
        text = []
        while line is not None and len(line) > 0:
            text.append(self.__text_extraction(line))
            line = operation_file.readline()
        operation_file.close()
        part_file = open(self.__result_path + '/' + path + '/part.json',
                                                encoding='utf-8',mode='w')
        part_file.write(json.dumps(text,ensure_ascii=False))
        return
```

图 4-9: 知识图谱特征提取模块 FeatureExtract 类场景分析

```
    # 对某一图片做特征提取
    def __pic_extraction(self, path):
      # 1.判断，只有图片才可以进行图片特征提取
        if not isPicture(path):
          return
        # 2.创建用于存储图片特征提取信息的文件夹
        self.__create_dir(path)
        # 3.调用ExtractPic对图片进行特征提取
        original_path = path
        result_path = str(path).replace(self.__original_path, self.
                                                __result_path)
        C = Config()
        C.update_path(dir, dir + '/component', dir + '/ocr', dir + '/merge')
        option = Config_enum()
        main = ExtractPic(C, option)
        main.run_widget_ext(original_path, result_path)
        return
```

图 4-10: 知识图谱特征提取模块 FeatureExtract 类图片提取实现

```
    # 对某一行文本（复现步骤）做特征提取
    def __text_extraction(self, line) -> list:
        verb, object = extract_keyword(line)
        return [verb, object]
```

图 4-11: 知识图谱特征提取模块 FeatureExtract 类文本提取实现

4.2 知识图谱关系提取模块

知识图谱关系提取模块作为知识图谱构建的第二个模块，在整个知识图谱的构建过程中起到承上启下的作用。知识图谱关系提取模块会对知识图谱特征提取模块分析出的图片、文本数据进行分析和整理，将相关的图文数据进行结合，找到其中的图文关系。

4.2.1 知识图谱关系提取模块详细设计

表 4-3: 知识图谱关系提取模块操作类详细说明

编号	类名	功能
001	RelationExtract	知识图谱关系提取模块的启动类，用户通过配置知识图谱特征提取模块提取出来的信息的存储位置，并配置知识图谱关系提取模块结果集的存放位置，然后调用该类的 execute 方法执行知识图谱的关系提取。
002	TextPicBound	该类用于对同一份众包测试报告中的截图与复现步骤进行绑定。该类可以将知识图谱特征提取模块提取出来的图片组件、文本信息，复现步骤操作、操作对象信息，复现步骤的前继、后继顺序进行整理绑定。
003	MessageCompare	该类用于对文本做相似度的比对。该类不仅通过 synonyms 包对文本进行相似度的比对，同时加入了针对不同功能点配置的同义词库帮助进行文本相似度的比对。
004	WidAnalysis	该类用于对某一组件的图片进行分析操作，并识别出该组件的类型，例如是 Button 还是 EditText 等。

知识图谱关系提取模块的顺序图如图 4-12所示。知识图谱关系提取模块的顺序图中总共涉及到用户以及四个类的调用操作，这四个类的具体信息如表 4-3所示。表中所列出的四个类分别是 RelationExtract、TextPicBound、MessageCompare 以及 WidAnalysis。其中 RelationExtract 是与用户交互的类，也是知识图谱关系提取模块的启动类。用户在完成知识图谱特征提取的操作之后，将特征提取的结果存放在指定的文件夹中，将该文件夹配置到 RelationExtract 文件中，并配置好关系提取模块的结果存放路径，然后调用 RelationExtract 的 execute 方法即可开启知识图谱关系提取的整个流程。TextPicBound 则为帮助 RelationExtract 做关系绑定的类。该类通过将同一份众包测试报告中的图片拆解信息与文本拆解信息进行整合和绑定，从而帮助知识图谱关系提取模块建立知识之间的关系。MessageCompare 则用来帮助知识图谱进行文本的相似度比对工作。这里的文本相似度不是简单的判断两个词是否为生活中的近义词，他还需要判断多种模式，例如在特定功能点中是否为近义词，在特定情况下某些格式的内容是否为近义词等。通过特定的逻辑编写保证了文本相似度比较的客观性。另外，WidAnalysis 则用来对一张组件的图片进行分类。该类可以通过对组件的图片进行 CNN 学习，识别出图片中的组件属于 Android 组件库的哪种组件，例如是属于 Button 还是 EditText 等，帮助知识图谱关系提取模块做更细致的知识提取工作。

图 4-12中即通过上述阐明的四个类与用户的相互协作完成了整个知识图谱

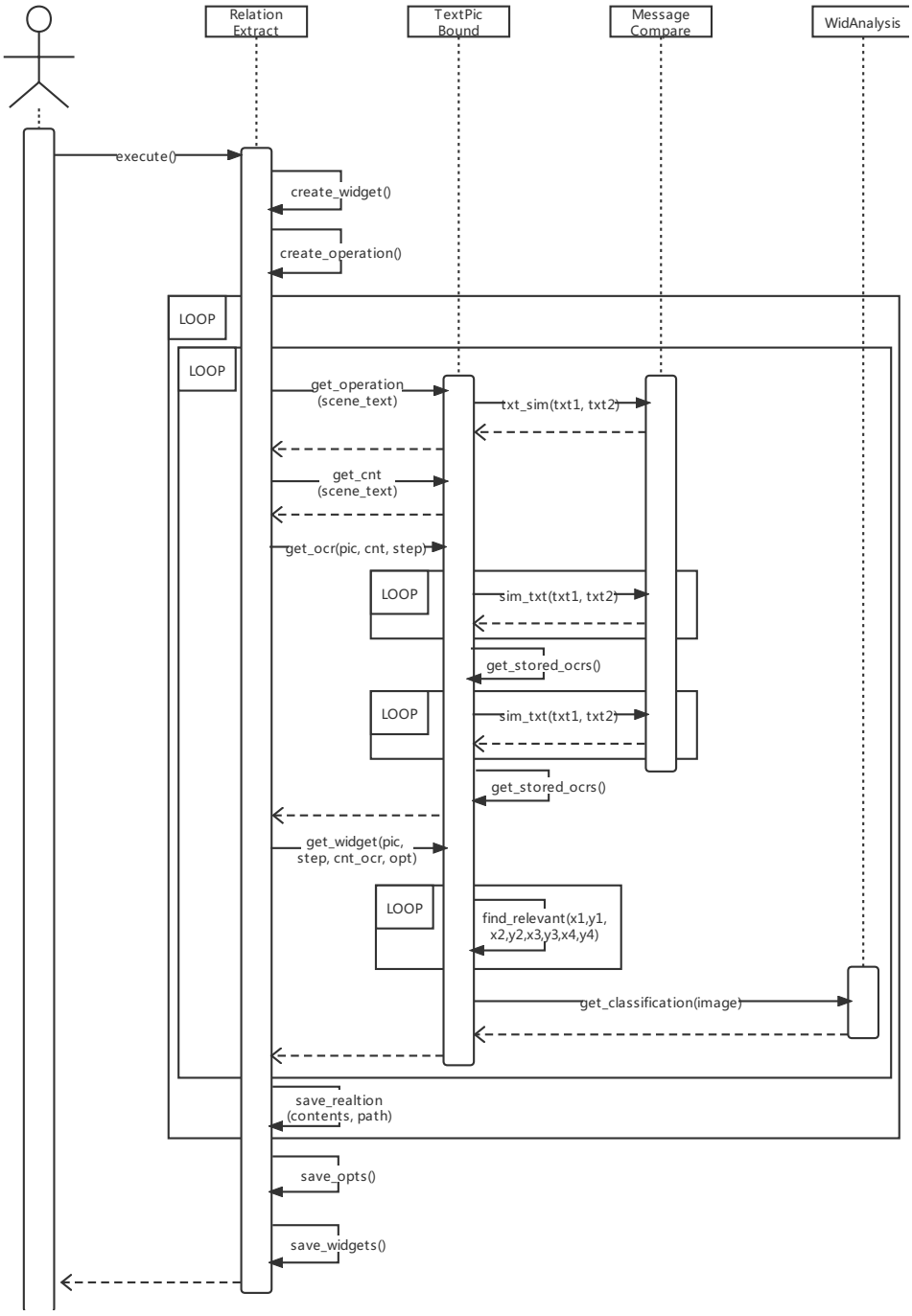


图 4-12: 知识图谱关系提取模块顺序图

关系提取模块的操作。首先，用户需要对整个操作流程进行配置，包括特征提取模块输出数据的保存位置，关系提取模块最终结果集的存放位置等。之后用户可以开始调用 RelationExtract 中的 execute 方法，该方法为知识图谱关系提取模块的总入口，用户在调用该方法之后只需等待结果生成至指定文件夹即可。

在调用了 RelationExtract 文件的 execute 方法后，系统首先会读取组件和操作配置文件，加载常见的组件、操作进内存。因为安卓组件的数目是固定的，操作的方式也是比较少的，所以通过配置文件进行配置，然后 RelationExtract 通过读取的方式将组件、操作信息都加载之内存当中。

然后 RelationExtract 将根据 4.1 节中提前到的知识图谱关系提取的数据输出格式来读取数据，并创建新的文件夹将结果保存到指定的文件夹中。按照 4.1 节中的方式，假设知识图谱的特征提取数据存放在 FE 文件夹中，其中包括移动应用级别的特征提取信息文件夹 AD1, AD2, ..., ADn。移动应用级别的特征提取信息文件夹下存放着场景级别的特征提取信息文件夹 SD1, SD2, ..., SDn。在场景级别的特征提取信息文件夹下则存放着图片的特征提取信息文件夹 PIC1, PIC2, ..., PICn 以及文本特征提取信息文件 part.json。RelationExtract 将根据该组织方式进行读取和关系提取操作。具体的排列方式和关系提取之后的文件排列方式如图 4-13所示。

首先 RelationExtract 将先创建一个以当前时间为文件名的文件夹 (RR) 放置到结果文件路径下。然后将所有移动应用级别下的场景文件夹 (SD) 提取出来，逐一进行关系提取工作。

针对某一个场景文件夹 SD1，我们首先读取 SD1 中的 part.json 文件，将复现步骤加载到内存中。然后我们取出 part.json 文件中的第一条复现步骤 STEP1，并找到与该复现步骤对应的图片特征提取信息 SD1/PIC1 文件夹，将这些信息分多次传递给 TextPicBound 进行图文数据的关系绑定。首先，将 STEP1 传递给 TextPicBound，调用 get_operation 方法，从中提取出该步骤用户对待测应用的操作。然后再将 STEP1 传递给 TextPicBound，调用 get_cnt 方法，从中提取出该步骤用户对待测应用的哪个逻辑概念执行了操作。之后通过图片特征提取信息 (SD1/PIC1/)、上一步获得的逻辑概念信息以及操作的步骤序号来执行图片 OCR 的搜过工作。该工作会用逻辑概念与图片中的所有 OCR 信息进行文本相似度比较，找对最相符的 OCR 信息，同时再将找到的 OCR 信息与系统中所有已知的 OCR 信息进行比较，将所有与该 OCR 信息相似的 OCR 信息标记出来，然后将结果返回给 TextPicBound。每发现一个新的 OCR 信息，就将新的 OCR 信息保存到 RR/ocrs.json 文件中。之后 RelationExtract 再将特征提取的图片信息 (SD1/PIC1/)、操作的步骤序号、与逻辑概念绑定的 OCR 信息以及操作信息传递给 TextPicBound 并调用 get_widget 方法来获取组件类型信息。TextPicBound 首先会在图片组件提取信息中找到循环找到与逻辑概念 OCR 相

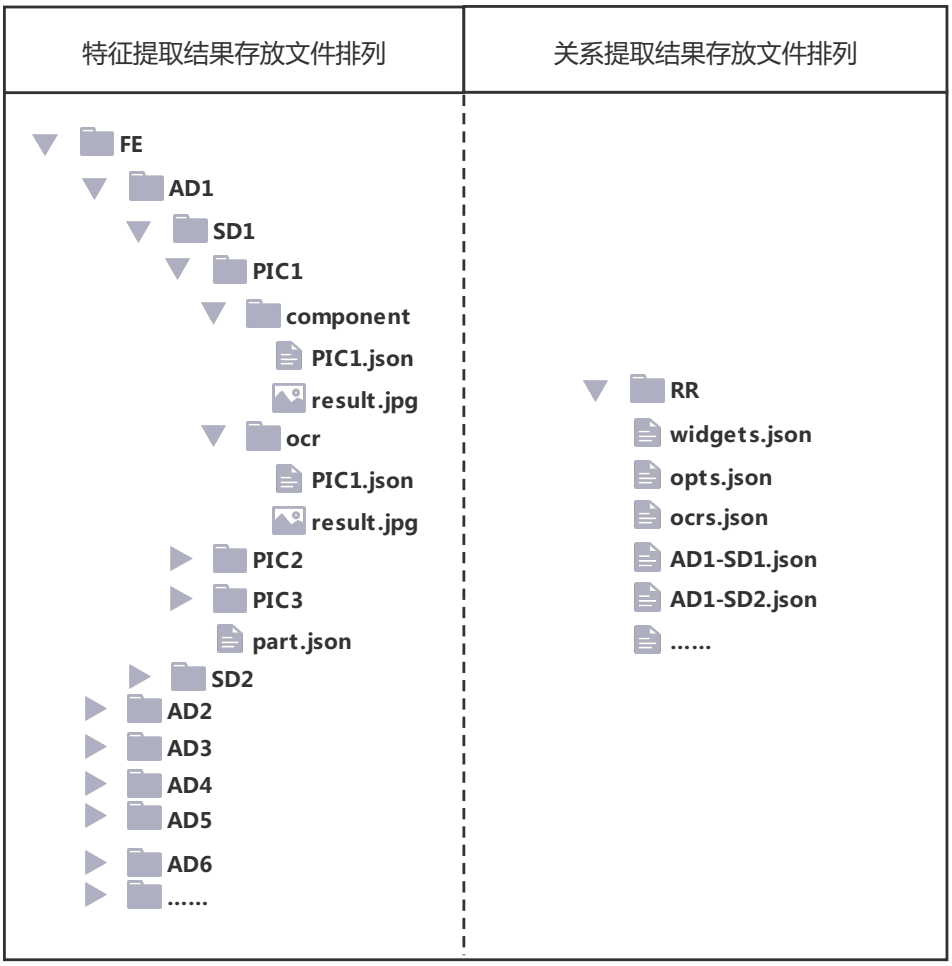


图 4-13: 知识图谱关系提取模块特征提取与关系提取文件排列方式

交面积最大的组件作为该操作的组件，并将该组件从原来的屏幕截图中扣取出来，并调用 WidAnalysis 的 get_classification 方法来获取组件的分类信息，并将结果返回给 RelationExtract。至此，对于一张图片与一个复现步骤的关系提取流程完成。

再针对某一场景文件夹 SD1 中的所有复现步骤与图片特征提取信息进行关系提取之后，会将提取出来的信息按照操作步骤的顺序按顺序排列，即形成前继后继关系，并将形成的前后继结果以 json 文件的形式保存到 RR 文件夹中，以场景名称命名文件。图 4-14为一个以偿命名称命名的关系提取结果文件样例。

在完成所有场景文件夹的关系提取之后，RelationExtract 会将提前加载的控件、操作内容与后续关系提取过程中新出来的操作内容相合并，并将结果分别以 json 形式保存到 RR/widgets.json 文件与 RR/opts.json 文件中。图 4-15即为

```
[
 {
  "id": "73",
  “name”: “三横挂件",
  "english": "",
  "similar": [],
  "opt": {"id": "1","name": "点击","english": "click"},
  "widget": {"id": "5","name": "图像视图","english": "ImageView"},
  “ocr”: {“id”: “1”,“name”: “三横线","similar": []}
 },
 {
  "id": "74",
  "name": "立即登录",
  "english": "",
  "similar": [],
  "opt": {"id": "1","name": "点击","english": "click"},
  "widget": {"id": "5","name": "图像视图","english": "ImageView"},
  "ocr": {"id": "2","name": "立即登录","similar": []}
 },
 {}
]
```


RR/AD1-SD1.json

图 4-14: 知识图谱关系提取模块场景关系提取保存文件样例


```
[
 {
  "id": "1",
  "name": "普通按钮",
  "english": "Button"
 },
 {
  "id": "2",
  "name": "复选框",
  "english": "CheckBox"
 },
 {
  "id": "3",
  "name": "文本框",
  "english": "TextView"
 },
 {
  "id": "4",
  "name": "编辑框",
  "english": "EditText"
 },
 {}
]
```


RR/widgets.json


```
[
 {
  "id": "1",
  "name": "点击",
  "english": "click"
 },
 {
  "id": "2",
  "name": "输入",
  "english": "input"
 },
 {
  "id": "3",
  "name": "滑动",
  "english": "move"
 },
 {
  "id": "4",
  "name": "点击勾选",
  "english": ""
 }
]
```


RR/opts.json


```
[
 {
  "id": "1",
  “name”: “三横线",
  "similar": []
 },
 {
  "id": "2",
  "name": "立即登录",
  "similar": []
 },
 {
  "id": "3",
  "name": "同意《用户协议》《隐私政策》《儿童隐私政策》",
  "similar": []
 },
 {
  "id": "4",
  "name": "一键登录",
  "similar": []
 }
]
```


RR/ocrs.json

图 4-15: 知识图谱关系提取模块控件、操作、OCR 文本保存格式文件样例

widgets.json、opts.json 以及 ocrs.json 的样例文件。

在完成以上所有任务之后，知识图谱关系提取模块的所有流程就执行完毕。

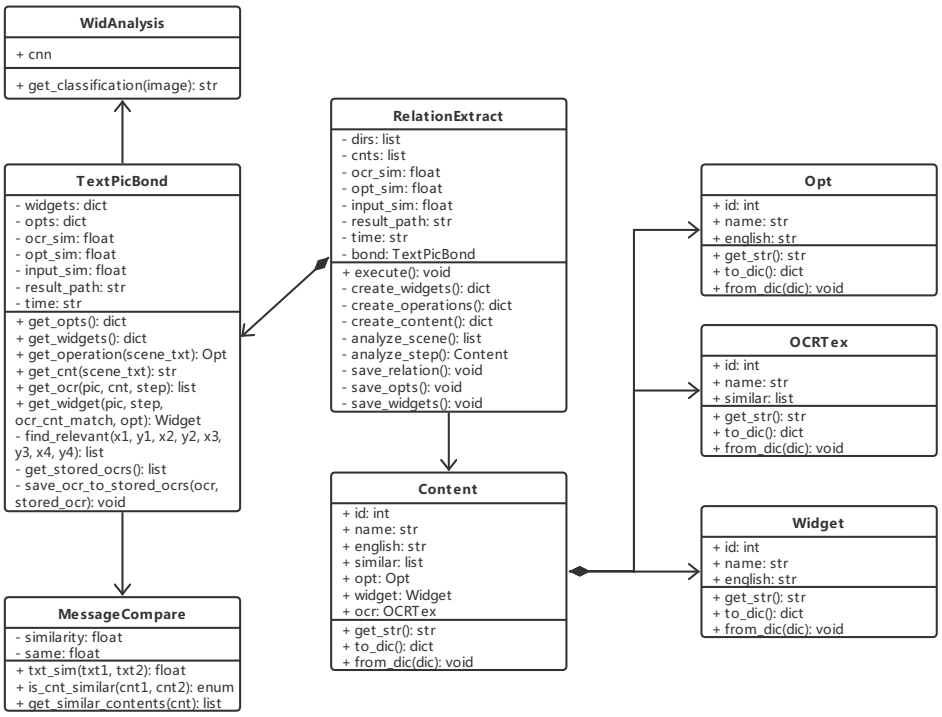


图 4-16: 知识图谱关系提取模块类图

图 4-16则展示了知识图谱关系提取模块的类图设计与实现。该类图中主要涵盖七个类，分别是 RelationExtract、TextPicBound、MessageCompare、WidAnalysis、Content、Opt、OCRTex 以及 Widget。其中，前四个类为操作类，具体信息见表 4-3, 其他类为数据类，具体信息如表 4-4所示。

在类图 4-16中我们可以看到 RelationExtract 与 TextPicBound 为聚合关系，RelationExtract 通过调用 TextPicBound 中的方法分别获得操作信息记录到 Opt 类中、图片文本信息记录到 OCRTex 类中、组件类型信息记录到 Widget 类中，并将逻辑概念、操作、图片文本、组件类型等所有信息均记录到 Content 类中，将 Content 类中的内容进行整理保存，完成知识图谱关系提取模块的整体流程。

表 4-4: 知识图谱关系提取模块数据类详细说明

编号	类名	功能
005	Content	该类用于存放最终放入知识图谱中的某一完整节点信息。该信息包括以逻辑概念为主体，连接操作、控件、图像文本等信息组织的知识图谱节点信息。该节点中还保留着与之相似的知识图谱节点信息。
006	Opt	该类用于存放对待测应用的某一操作。包括操作的名称、操作的英文注释等信息。
007	OCRTex	该类用于存放与逻辑概念匹配的某一 OCR 文本信息，包括 OCR 文本以及该文本的相似文本等。
008	Widget	该类用于存放待测应用的某一组件信息，包括组件的中英文名称、组件的编号等。

```
# 知识图谱关系提取模块的启动入口
def execute(self):
    # 1.加载配置文件中的控件、操作信息进内存，并初始化TextPicBound
    widgets = self.__create_widget()
    opts = self.__create_operation()
    self.__bound = TextPicBound()

    # 2.提取所有场景中的关系，并将结果保存
    self.__create_content()

    # 3.将更新后的控件、操作信息进行保存
    self.__save_opts()
    self.__save_widgets()
    return
```

图 4-17: 知识图谱关系提取模块 RelationExtract 类启动入口

4.2.2 知识图谱关系提取模块代码实现

图 4-17即为知识图谱关系提取模块的启动入口。用户只需要将特征提取出的文件放置到指定位置即可调用该方法进行关系提取操作。该方法将首先通过配置信息创建控件、操作实体，然后再通过分析特征提取出来的文件创建内容、图片文本实体，并建立联系。

图 4-18即为创建内容实体的总入口。系统将根据特征提取模块中输出的文件，逐一进行读取并进行分析，识别其中的关系场景，并将关系记录下来。

对每个内容实体进行创建的过程中，涉及到对每个场景执行流程的分析。图 4-19则展示了通过已经特征提取好的场景数据生成关系绑定数据的代码逻辑。代码读取场景文件中的文本内容与图片内容，将每条文本内容与每张图片内容做对应，然后对每一步进行具体分析。

```
    # 根据特征提取的场景文件夹进行关系提取
    def __create_content(self):
      # 对每个场景文件夹进行关系提取，并保存结果
        for cnt_dir in self.__dirs:
            relation = self.__analyze_scene(cnt_dir)
            self.__save_relation(relation, cnt_dir)
        return
```

图 4-18: 知识图谱关系提取模块 RelationExtract 类创建内容实体

```
    # 对场景文件夹进行关系提取
    def __analyze_scene(self, scene):
        steps_size = int(len(os.listdir(scene)) / 2)
        file = open(scene + '/part.json', 'r', encoding='utf-8')
        texts = json.loads(file.read())
        file.close()
        contents = []
        for i in range(0, steps_size):
            pic_dir_name = str(i + 1).zfill(2)
            contents.append(self.__analyze_step(scene + '/' + pic_dir_name,
                                                texts[i], pic_dir_name))
        return contents
```

图 4-19: 知识图谱关系提取模块 RelationExtract 类分析场景

图 4-20则展示了知识图谱关系提取模块中分析每个步骤并获取结果的方法实现。该方法通过从特征提取的文本中获取操作对象、用户输入对象和布局对象，并通过报告文本与截图文本做匹配找到匹配控件，并识别其类型作为关系提取的结果返回。

```
    # 对每一复现步骤进行关系提取
    def __analyze_step(self, scene_pic, scene_txt, step):
        # 1.获取操作对象
        opt = self.__bound.get_operation(scene_txt)
        # 2.获取用户输入对象
        cnt = self.__bound.get_cnt(scene_txt)
        # 3.获取页面布局对象
        layout = self.__bound.get_layout()
        # 4.获取OCR
        cnt_ocr_match, ocr = self.__bound.get_ocr(scene_pic, cnt, step)
        # 5.获取控件类型
        widget = self.__bound.get_widget(scene_pic, step, cnt_ocr_match, opt
                                         )
        # 6.创建content对象
        content = Content(self.__cnts + 1, cnt, '', [], opt, widget, layout,
                                           ocr)
        self.__cnts = self.__cnts + 1
```

图 4-20: 知识图谱关系提取模块 RelationExtract 类代码实现

4.3 知识图谱共指消解模块

知识图谱共指消解模块作为知识图谱构建的第三个模块，也是知识图谱构建过程的最后一个模块。知识图谱共指消解模块会对知识图谱关系提取模块提取出来的关系进行分析，查询重复内容并去除冗余知识，最终将知识信息上传至 Neo4j 图数据库形成基于移动应用功能点测试的知识图谱。

4.3.1 知识图谱共指消解模块详细设计

知识图谱共指消解模块的顺序图如图 4-21所示。知识图谱共指消解模块的顺序图中总共涉及到用户以及四个类的调用操作，这四个类的具体信息如表 4-5所示。

表 4-5: 知识图谱特征提取模块操作类功能说明

编号	类名	功能
001	GenerateGraph	知识图谱共指消解模块的启动入口。该类帮助用户将知识图谱关系提取模块输出的数据进行分析转化，去除冗余信息，并将结果上传至 Neo4j 图数据库中完成知识图谱的构建。
002	MessageCompare	该类主要用于相似度的比较，包括上述章节中涉及到的特定功能点领域的文本相似度的比较，同时也可以提供知识图谱中信息节点的比较，例如找寻某个节点的所有相似节点等功能。
003	GraphAdd	该类主要用于与 Neo4j 图数据库进行交互，向数据库中添加节点或关系的操作类。
004	GraphSearch	该类主要用于与 Neo4j 图数据库进行交互，从数据库中读取节点或关系的操作类。

表 4-5中列出的四个类分别为GenerateGraph、MessageCompare、GraphAdd以及 GraphSearch。其中 GenerateGraph 为知识图谱共指消解模块的总入口，用户通过提前将知识图谱关系提取模块输出的数据放置到指定的位置，并将关系提取数据的位置配置到知识图谱共指消解模块中，调用 execute 方法即可开始知识图谱的共指消解并将结果上传至 Neo4j 图数据库作为基于移动应用功能点的知识图谱。在整个共指消解的过程中，需要用到 MessageCompare 类来做辅助，这其中包括在特定功能点领域内的共指消解，例如在查询机票的功能点中的出发地涉及到很多地域，例如北京、上海、南京等，MessageCompare 可以根据设定的功能点类型来判断在该领域下北京与上海是相似的。同时 MessageCompare 还可以获取知识图谱中的相似节点，从而助力于知识图谱的构

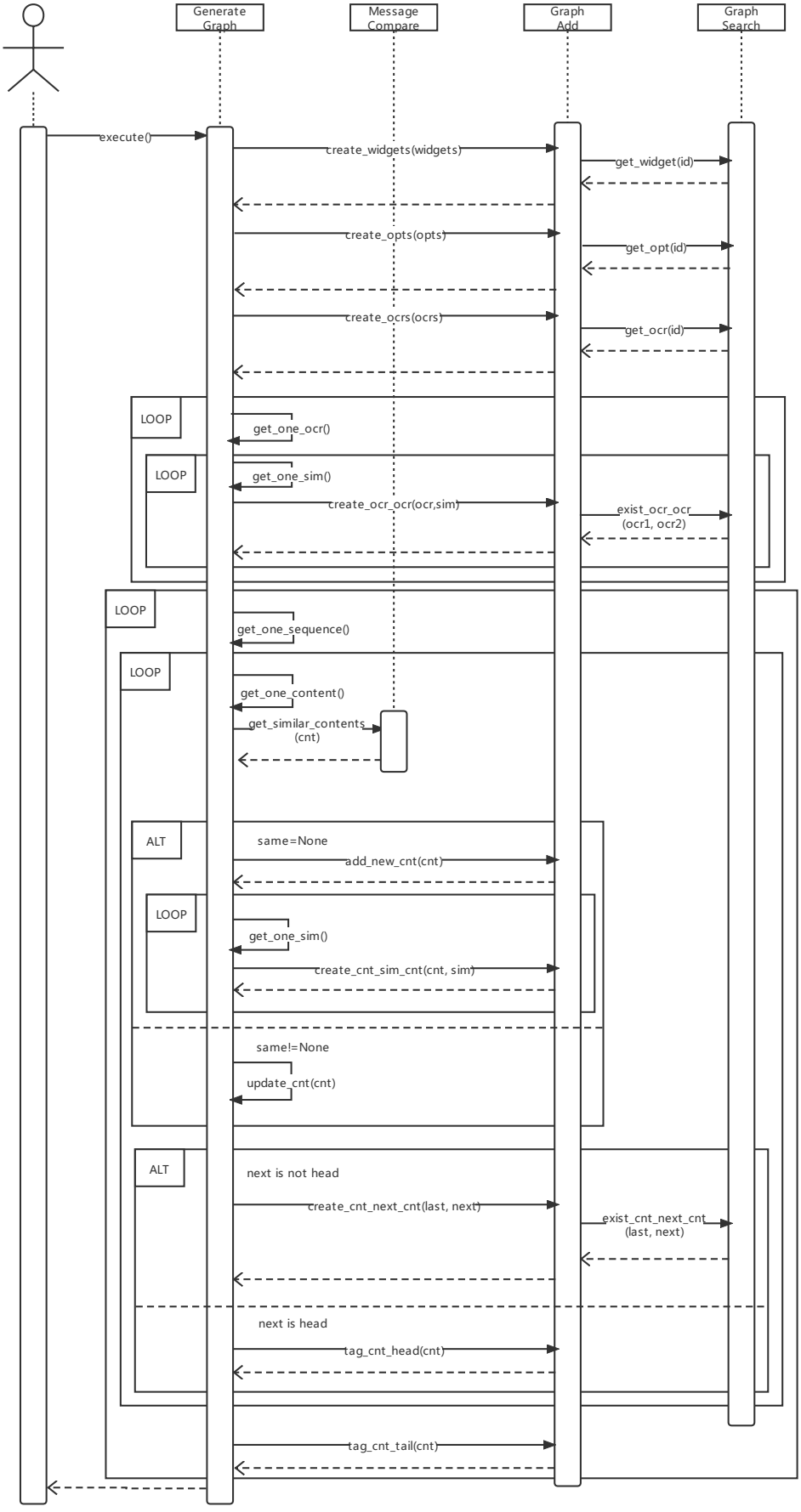


图 4–21: 知识图谱共指消解模块顺序图

建。GraphAdd 与 GraphSearch 即为直接与 Neo4j 图数据库进行交互的类，一个负责向图数据库中写数据，一个负责从图数据库中读数据。这两个类中封装了多样的接口供其他类选择与使用。

图 4-21中即通过上述阐明的四个类与用户的相互协作完成了整个知识图谱共指消解模块的操作。首先，用户通过将知识图谱关系提取模块的输出信息配置到指定位置以调用 GenrateGraph 的 execute 方法启动知识图谱的共指消解过程。首先，GenerateGraph 通过读取 widgets.json 文件、opts.json 文件以及 ocrs.json 文件将控件、操作以及 OCR 文本信息都传递到 Neo4j 图数据库中。在上传的过程中，涉及到调用 GraphAdd 的三个 create 方法，这三个方法在创建之前还会通过 GraphSearch 的 get 方法来确认这些控件、操作、OCR 文本是第一次上传，从而避免信息的重复。对于 OCR 文本的上传，不光要上传 OCR 文本实体，还需要将 OCR 文本之间的关系上传，即哪些 OCR 文本之间在该功能点领域下属于相似实体，则在二者之间新建关系相似，这个操作要依赖于 GraphAdd 的 create_ocr_ocr 来实现，同时，create_ocr_ocr 也会通过 GraphSearch 的 exist_ocr_ocr 来判断是不是第一次上传该相似关系，以避免重复。

在完成控件、操作、OCR 文本的上传工作之后，就到了场景内容文件的共指消解部分了。场景内容文件即 4.2 节中提到的关系提取最终生成的场景关系提取文件，例如 RR/AD1-SD1.json 文件。针对每一个场景关系提取文件，知识图谱共指消解模块会逐一对其进行分析并将结果上传至知识 Neo4j 图数据库来构建知识图谱。

对于每个场景关系提取文件来说，其中包含着多个节点信息，这些信息之间是存在先后顺序的。因此，知识图谱共指消解模块会首先提取出某个场景关系提取文件中的第一个节点信息，并调用 MessageCompare 的 get_similar_contents 方法来获取图数据库中所有与该节点相同、相似的节点。如果数据库中存在相同的节点，则只需要将该数据库中的相同节点里未保存完整的信息通过当前节点进行补充即可；如果数据库中不存在相同的节点，则上传该节点的信息至图数据库中。在上传完成之后，再对当前节点与其所有相似节点之间建立相似关系，通过循环的调用 GraphAdd 的 create_cnt_sim_cnt 方法来实现。在完成节点与节点关系的上传之后，知识图谱共指消解模块会分析当前节点的类型，如果是一个场景文件中的第一个节点信息，则通过 GraphAdd 的 tag_cnt_head 方法将该节点在数据库中标记为头节点，如果该节点不是场景文件中的第一个节点信息，则在数据库中建立该节点与场景文件中的上一节点

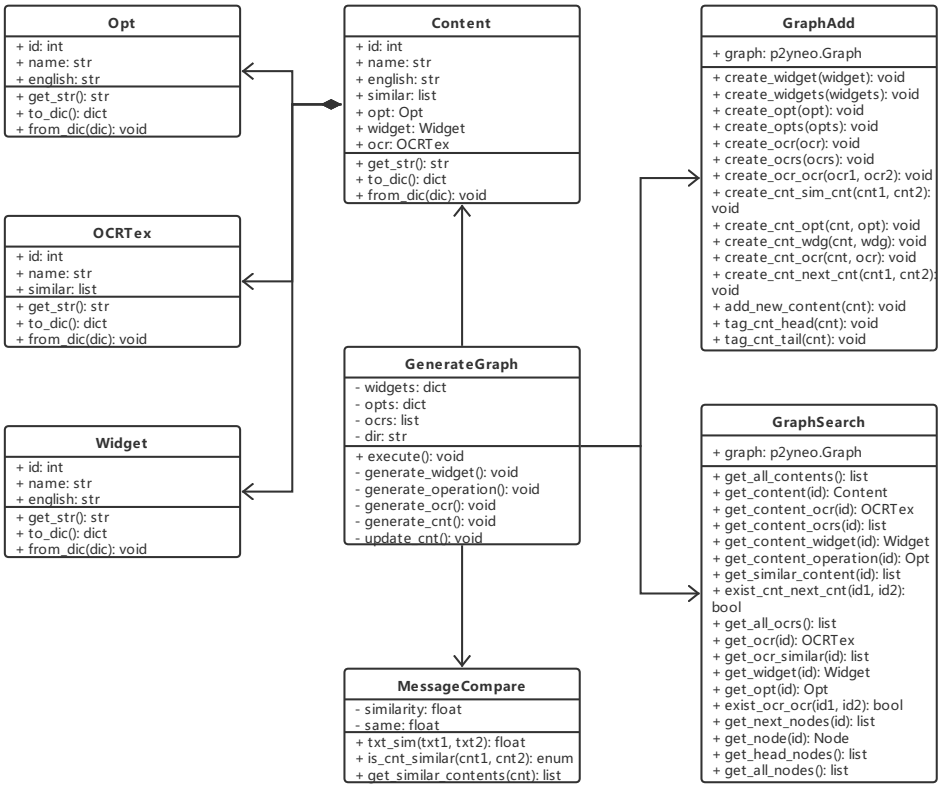


图 4-22: 知识图谱共指消解模块类图

之间的前后继关系。在完成对当前场景关系文件所有节点的分析上传工作之后，GenerateGraph 再通过调用 GraphAdd 的 tag_cnt_tail 的方式标记场景关系提取文件中的最后一个节点为尾节点。至此，一个场景关系提取文件即完成了其共指消解的全过程。

图 4-22则展示了知识图谱共指消解模块的类图设计与实现。该类图主要涵盖八个类，分别是 GenerateGraph、MessageCompare、GraphAdd、GraphSearch、Content、Opt、OCRTex 以及 Widget。其中，前四个类为操作类，具体信息见表 4-5, 其他类为数据类，这些数据类在 4.2 节中同样有涉及，具体信息见表 4-4。

在类图 4-22中我们可以看到 GenerateGraph 为总启动类，他通过使用 MessageCompare、GraphAdd、GraphSearch 来完成知识图谱的共指消解操作并将结果上传至 Neo4j 图数据库。同时通过使用 Content、Opt、OCRTex、Widget 来承载节点数据帮助知识图谱的构建。

4.3.2 知识图谱共指消解模块代码实现

图 4-23即为知识图谱共指消解模块代码的总入口。该方法会通过读取关系提取模块生成的数据分别上传控件实体、操作实体、图片文本实体以及内容实体，并排除冗余建立联系，最终生成面向移动应用功能点测试的知识图谱。

```
    # 知识图谱共指消解模块的启动入口
    def execute(self):
        # 将widget.json文件中的信息上传至图数据库
        self.__generate_widget()

        # 将opts.json文件中的信息上传至图数据库
        self.__generate_operation()

        # 将ocrs.json文件中的信息上传至图数据库
        self.__generate_ocr()

        # 将场景关系提取文件中的信息上传至图数据库
        self.__generate_cnt()
        return
```

图 4-23: 知识图谱共指消解模块 GenerateGraph 类启动入口

图 4-24则展示了共指消解模块生成内容实体时的代码实现。该部分为共指消解模块的中心方法。首先通过读取关系提取模块提供的数据将场景文件以内容实体列表的形式读取到 cnt_seq 中，然后针对每一个列表，系统对列表中的内容实体进行共指消解操作。系统首先判断当前图谱中是否存在相同或相似的节点，如果有相同的节点，则只需要将当前内容实体相连的关系添加到图谱中已存在的节点上即可，如果不存在相同节点而存在多个相似节点，则创建新内容节点，并在新内容节点与相似节点间添加相似关系。同时，在上传列表中节点的过程中，系统还会添加内容节点间的前后继关系，内容节点与操作节点之间的操作关系，内容节点与控件节点之间的控件关系，内容节点与截图文本节点之间的 OCr 文本关系。在最后，还需要对每个列表内容实体的头节点和尾节点分别进行标记。至此循环往复完成内容节点的创建与更新工作。

4.4 知识图谱智能查询模块

知识图谱智能查询模块主要为其他使用基于移动应用功能点测试知识图谱的用户提供了一个查询的入口。用户只需要将某移动应用的屏幕截图传递给智

```
    # 将场景关系提取文件中的信息上传至图数据库
    def __generate_cnt(self):
        # 1.读取所有场景关系提取文件
        scenes = os.listdir(self.__dir)
        cnt_seq = []
        for scene in scenes:
            if scene not in ['widgets.json', 'opts.json', 'ocrs.json']:
                #将文件内容转化为Content类的列表存储到cnt_seq中

        # 2.对每个场景关系提取文件进行共指消解操作并上传节点信息
        deduplicate = MessageCompare()
        for seq in cnt_seq:
            last = None
            # 2.1 分析每个场景提取文件中的每个节点
            for c in seq:
              # 2.1.1 获取当前节点在数据库中已存在的相同、相似节点
                same, similar = deduplicate.get_similar_cnts(c)
                next = None
                # 2.1.2 如果不存在相同节点，则上传当前节点并连接相似节点
                if same is None:
                    GraphAdd.add_new_content(c)
                    for sim in similar:
                        GraphAdd.create_cnt_sim_cnt(c, sim)
                    next = c
                # 2.1.3 如果存在相同节点，则通过当前节点来更行图库中的相同节
                                                        点
                else:
                    self.__update_cnt(c,same)
                    next = same
                # 2.1.4 标记头节点或连接两节点
                if last is not None:
                    GraphAdd.create_cnt_next_cnt(last, next)
                else:
                    GraphAdd.tag_cnt_head(next)
                last = next
            # 2.2 标记尾节点
            GraphAdd.tag_cnt_tail(last)
        return
```

图 4-24: 知识图谱共指消解模块 GenerateGraph 类生成内容实体

能查询模块并确定测试的功能点，知识图谱智能查询模块将通过该功能点的知识图谱分析图片信息并返回可操作的控件序列即逻辑含义，帮助用户执行后续操作。

4.4.1 知识图谱智能查询模块详细设计

知识图谱的智能查询模块的顺序图如图 4-25所示。知识图谱的智能查询模块总共涉及到用户以及六个类的调用操作，这六个类的具体信息如表 4-6所示。表中列出的六个类分别为 NextStep、GraphSearch、LayoutMain、

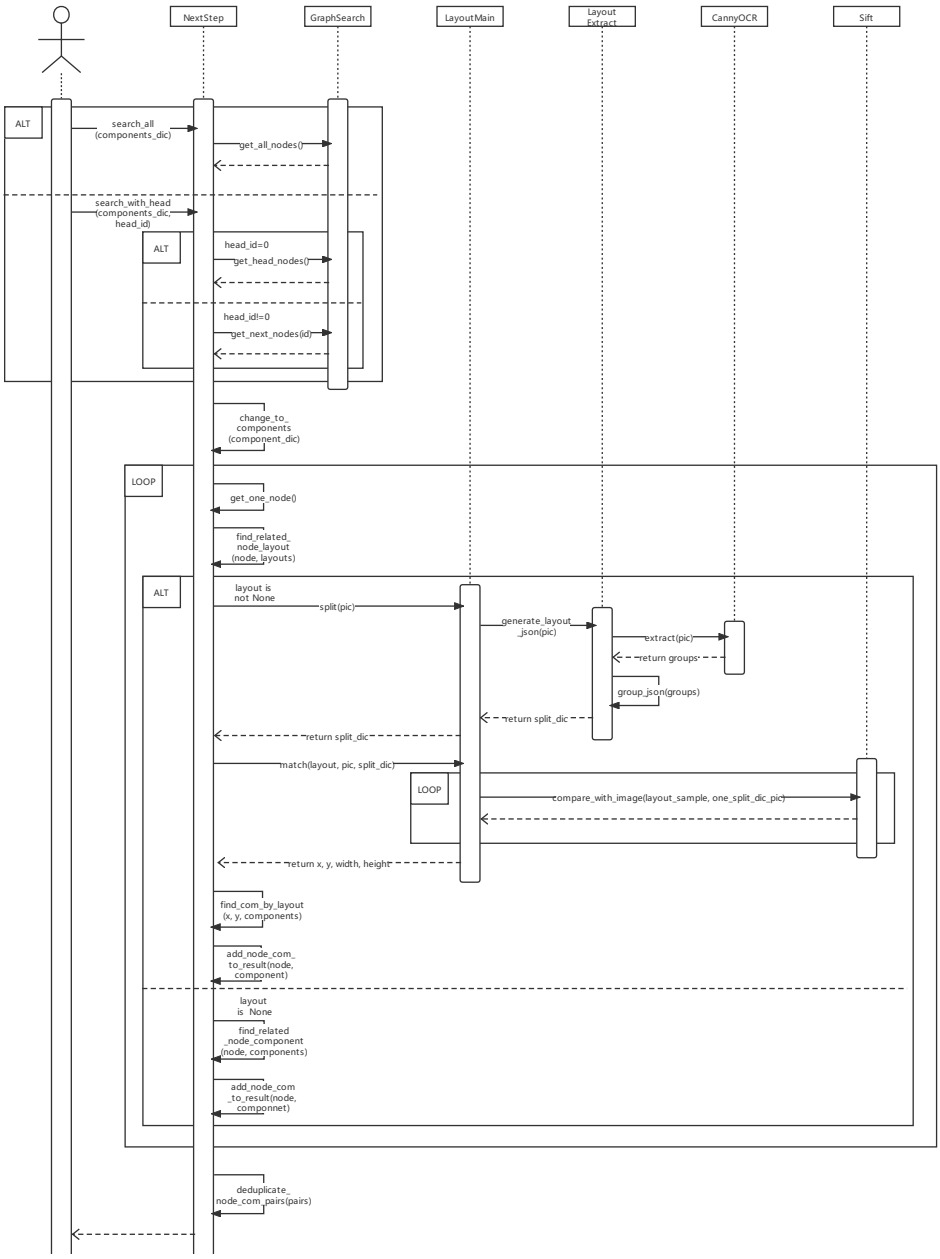


图 4–25: 知识图谱智能查询模块顺序图

LayoutExtract、CannyOCR 以及 Sift。其中，NextStep 为知识图谱智能查询模块的启动入口类，用户只需要调用其中的方法即可对屏幕截图进行分析，并获取下一步可能需要执行的操作与逻辑概念列表。GraphSearch 即为与 Neo4j 图数据进行交互的用于读取数据库信息的接口类。LayoutMain 则主要为智能查询模块提供对屏幕截图做布局分析的能力。该类主要依赖于 LayoutExtract、CannyOCR、Sift 三个类实现。LayoutExtract 依赖与 CannyOCR 对屏幕截图中的

表 4-6: 知识图谱特征提取模块操作类功能说明

编号	类名	功能
001	NextStep	该类为知识图谱智能查询模块的启动入口，用户只需要调用该类中的方法即可对某一张屏幕截图进行数据分析，并返回一个操作与逻辑概念的列表，告知用户针对当前功能点，可以对哪些组件进行操作。
002	GraphSearch	该类主要用于与 Neo4j 图数据库进行交互，从数据库中读取节点或关系的操作类。
003	LayoutMain	该类主要用于帮助 NextStep 识别屏幕截图的布局信息。该类可以将屏幕截图以水平方式划分屏幕截图，并挑选出与当前操作功能最相近的截图部分作为结果返回。
004	LayoutExtract	该类主要用于帮助 LayoutMain 做截图布局的理解。通过标记截图中的组件、文本信息，识别出图片中的横向与纵向布局，根据屏幕截图生成树状 DOM 树。
005	CannyOCR	该类主要用于识别屏幕截图中的组件、OCR 文本信息。
006	Sift	该类通过 SIFT 算法来比较两个不同大小图片的相似程度。

控件与文本信息进行坐标的识别与标记，再通过分析将组件与文本按照行、列的方式进行整合，从而识别出整张截图的布局信息。然后 LayoutExtract 会通过识别出来的布局信息将结果转化为一个 DOM 树的形式返回给 LayoutMain。同时 LayoutMain 还会通过使用 Sift 来做图片相似度的比较。例如对于某个功能点的某个操作步骤，需要在某片相似的区域内完成，LayoutMain 首先通过 LayoutExtract 将图片分解为多个横向模块，然后针对每一个横向模块与配置好的理论模块图片通过 Sift 做图片的相似度比对，从而找到最终需要被操作的截图模块返回给 NextStep 执行后续操作。

图 4-25即通过上述阐明的六个类与用户的相互协作完成了整个知识图谱智能查询模块的操作。首先，用户需要选定功能点，提供屏幕截图。此使，用户可以选择调用 NextStep 的 search_all 方法或者 search_with_head 方法。前者会针对知识图谱进行全局搜索，而后者则通过用户告知知识图谱上一步进行了什么操作的方式来缩小查询搜索范围，提高效率。在调用过之后，NextStep 会根据调用的方式从知识图谱数据库中读取出相应的数据节点信息。需要注意的是，如果通过 search_with_head 来调用，如果 head 不为 0，则按照正常逻辑所搜，如果 head 为 0，则意味着这是该功能点的起始步骤，需要从知识图谱数据库中搜索所有带有 head 属性的节点予以返回。

在获取到知识图谱中的相关数据之后，NextStep 会通过 change_to_components 方法将获取到的节点信息转化为后续使用的组件信息格式。然后针对每一个返

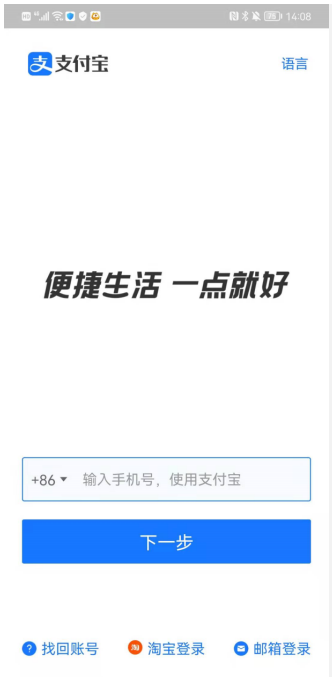
回的节点信息，进行分析，取出符合要求的节点作为结果返回给用户。

对每一个节点的分析主要涉及到两个层面，一个是布局分析的层面，一个是逻辑分析的层面。其中，布局分析的优先级高于逻辑分析。也即意味着如果存在针对该节点的布局配置，则按照布局分析层面执行分析而不按照逻辑分析层面执行分析。

图 4-26: 知识图谱智能查询模块布局分析样例

对于布局层面的分析来说，首先知识图谱智能查询模块先通过 NextStep 调用 LayoutMain 中的 split 方法将图片打碎。该过程需要嗲用 LayoutExtract 的 generate_layout 方法，同时 generate_layout 方法还需要调用 CannyOCR 的 extract 方法并调用自己的 group_json 方法将整张图片以横向的布局维度分解为多个模块。在 NextStep 拿到横向分解的多个模块之后，将该内容与原图交给 LayoutMain 并调用 match 模块，识别出这些分解的横向模块中哪块才是最后需要的模块。该过程需要 Sift 的参与。在智能查询开始之前，我们已经针对某些知识图谱节点的逻辑概念进行了布局信息的配置，该配置中写明了对于哪个功能点的哪个逻辑概念采用布局配置，该布局配置的类似截图以及对于该类似截图需要如何操作。Sift 则通过将分解出来的多个横向模块截图与配置好的布局截图进行图片相似度的分析，找到最相似的横向截图模块作为结果模块，然后通过配置文件中的配置信息找到结果模块截图中需要被操作的截图坐标，将坐标返回给 NextStep。NextStep 在拿到坐标之后，通过调用 find_com_by_layout 从图片的所有控件中找到该坐标属于的控件信息，并将找到的控件记录为需要操作的控件。这样布局层面的分析过程就结束了。图 4-26展示了布局分析时使

用的配置文件样例、以及最终找到指定操作控件的样例。



用户输入屏幕截图


```
{
 'status': 'success',
 'data': [
  {
   'category': 'TextView',
   'ocr': '淘宝登录',
   'operation': '点击',
   'cnt': '淘宝登录',
   'cnt_id': 68,
   'x1': 421,
   'y1': 2114,
   'x2': 674,
   'y2': 2173
  },
  {
   'category': 'TextView',
   'ocr': '邮箱登录',
   'operation': '点击',
   'cnt': '邮箱登录',
   'cnt_id': 35,
   'x1': 829,
   'y1': 2114,
   'x2': 1024,
   'y2': 2173
  },
  {
   'category': 'TextView',
   'ocr': '输入手机号,使用支付宝',
   'operation': '点击',
   'cnt': '手机号登录',
   'cnt_id': 2,
   'x1': 265,
   'y1': 1555,
   'x2': 792,
   'y2': 1610
  }
 ]
}
```


知识图谱智能查询系统返回结果

图 4-27: 知识图谱智能查询模块输入输出数据样例

对于逻辑层面的分析来说，首先知识图谱智能查询系统会对屏幕截图中的所有控件进行遍历，找到与当前节点信息相似度最高且超过设定阈值的组件，即调用 find_related_node_component 方法来实现。之后将结果记录并返回即可。

在对每一个节点都完成分析之后，我们会得到一个列表，这个列表中存储着节点组件对，即意味着列表中的每一项记录了应该按照存储节点中记录的方式来操作存储组件。然而在最后的列表中，我们可能会发现对于某一个控件来说，可能匹配了多个节点信息，我们需要找到相似度最高的节点信息保存，删去其他的节点信息，即调用 deduplicate_node_com_pair 来实现。例如我们针对登录功能点来说，有一个登录的按钮，该按钮为组件。同时知识图谱中有两个节点信息与登录按钮组件匹配，一个节点的逻辑概念为一键登录，另一个节点的逻辑概念为立即登录。在分析之后系统认为立即登录和登录的匹配度更高，因此在结果集中我们加入 (组件：登录按钮，节点：立即登录) 数据对而删除 (组件：登录按钮，节点：一键登录) 数据对，从而完成去重的操作。最后将结果返回给用户即可。图 4-27展示了一个输入输出的结果样例。

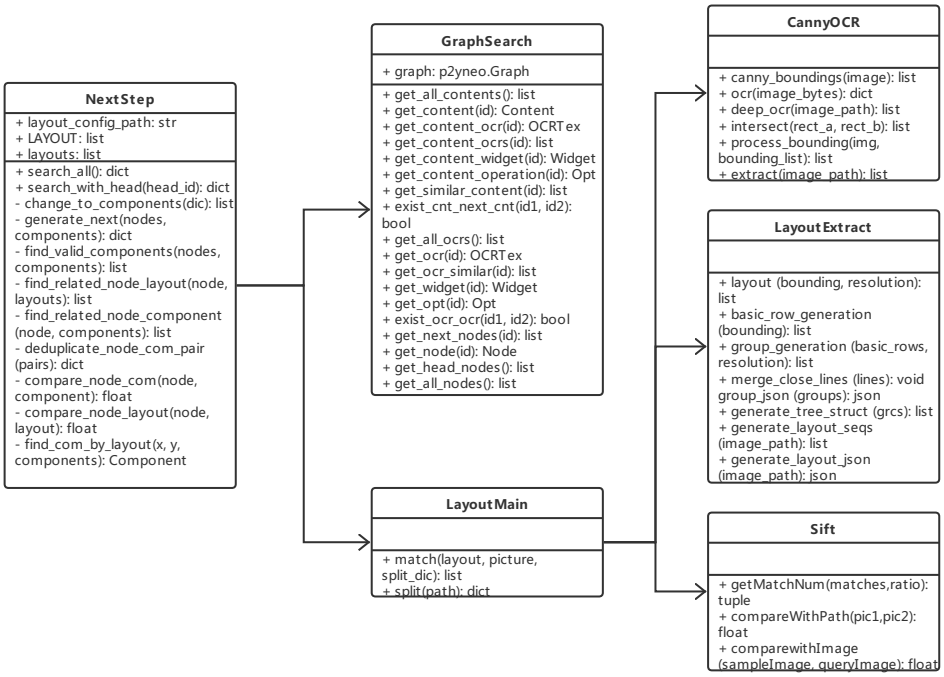


图 4-28: 知识图谱智能查询模块类图

图 4-28则展示了知识图谱智能查询模块的类图设计与实现。该类图主要涵盖六个类，分别为 NextStep、GraphSearch、LayoutMain、LayoutExtract、CannyOCR 以及 Sift。这六个类均为操作类，具体信息见表 4-6。在类图中我们可以看到 NextStep 为智能查询模块的入口启动类，用户通过调用该类中的方法，通过 GraphSearch、LayoutMain 的协助操作知识图谱的查询和布局模式的匹配。其中，LayoutMain 通过 CannyOCR、LayoutExtract、Sfit 来对布局信息做分析，帮助完成知识图谱的智能查询。

4.4.2 知识图谱智能查询模块代码实现

图 4-29则展示了知识图谱只能查询模块关键的启动入口之一。用户通过传入需要查询屏幕截图的控件、文本识别数据以及上一步的操作内容来查询下一步应该执行的操作。其中，当 head_id=0 时认为是其实操作，系统会去图谱中找到可以作为头节点的内容节点返回；若不为 0 则系统会找到该内容节点及其相似节点的所有后继节点进行返回。之后调用 generate_next 方法对返回的节点逐一进行分析比较，选出下一步可能需要操作的节点列表。

```
# 从获取的节点信息中找到匹配的节点、组件对，即（节点，组件）
```

```
def search_with_head(self, components_dic, head_id):
    # 1. 获取需要比对的图中节点
    if head_id == 0:
        nodes = GraphSearch.get_head_nodes()
    else:
        nodes = GraphSearch.get_next_nodes(head_id)

    # 2. 将json转化为component对象
    components = self.__change_to_components(components_dic)

    # 3. 比较
    result = self.__generate_next(nodes, components)
    return result
```

图 4-29: 知识图谱智能查询模块 NextStep 类启动入口

```
def __generate_next(self, nodes, components):
    # 1. 若操作变量为None, 则显示失败, 返回结果
    if nodes is None or components is None or len(nodes) == 0 or len(
                                            components) == 0:
        return {'status': 'failed','data': None}

    # 2. 寻找所有（节点, 组件）对
    pairs = self.__find_valid_components(nodes, components)

    # 3. 若组件对应多个节点, 只保留相似度最高的（节点, 组件）对
```

```
    result_dic = self.__deduplicate_node_com_pair(pairs)
    result = []
    for pair_key in result_dic:
        c: Component = result_dic[pair_key].component
        n: Node = result_dic[pair_key].node
        c.operation = n.opt.name
        c.cnt = n.name
        c.cnt_id = n.id
        result.append(c.to_dic())

    # 4. 返回结果
    return {'status': 'success','data': result}
```

图 4-30: 知识图谱智能查询模块 NextStep 类查找后继

图 4-30则展示了通过图谱查询节点与截图组件数据进行匹配，找到需要操作的组件和操作原因的方法。系统会通过将图谱节点与截图组件传入寻找有效组件的方法获得匹配到的节点、组件数据对，然后对获取到的数据对进行去重操作，保证一个组件只会对应一个最相似的节点，并将结果返回完成查询任务。

```
    # 寻找所有（节点，组件）对
    def __find_valid_components(self, nodes, components):
        result = []
        # 1.针对每一个节点，若有布局配置则进行布局分析，否则进行逻辑分析
        for n in nodes:
            # 1.1 布局分析
            # 1.1.1 先从layout配置文件里面找是否有匹配的布局信息
            related_l, related_l_similarity = self.
                                                __find_related_node_layout
                                                (n, NextStep.layouts)
            if related_l is not None:
                # 找到图片中的哪个模块对应于布局配置
                x, y, w, h = match(related_l, self.picture, self.split)
                # 根据布局配置找到图片中需要被操作（点击）的点
                center_x = w * related_l.x + x
                center_y = h * related_l.y + y
                c = self.__find_com_by_layout(center_x, center_y, components
                                                    )
                if c is not None:
                    temp = NodeComponent(n, c, related_l_similarity)
                    result.append(temp)
```

```
                    continue

            # 1.2 逻辑分析
            # 1.2.1 从components中通过文本等信息去匹配可以返回的图节点
            related_c, related_c_similarity = self.
                                                __find_related_node_component
                                                (n, components)
            if related_c is not None:
                temp = NodeComponent(n, related_c, related_c_similarity)
                result.append(temp)
        return result
```

图 4-31: 知识图谱智能查询模块 NextStep 类寻找有效组件

图 4-31展示了智能查询模块寻找有效组件的代码实现。该模块主要有布局分析和文本分析组成。系统首先对组件进行布局分析，若发现相似模块则根据布局配置将结果添加到返回结果集中。之后系统在对截图中的文本进行相似匹配，将达到相似度阈值的结果也加入到返回结果集中，完成智能查询。

4.5 知识图谱自动化构建系统实例展示

图 4-32: 知识图谱自动化构建系统首页

图 4-32展示了面向移动应用功能点测试的知识图谱构建系统的系统首页。系统总共有三个主功能，分别为构建图谱、查看结果、智能查询。下面，我们将对这三个模块分别进行页面展示与操作说明。

4.5.1 构建图谱模块实例展示

在用户选择进入构建图谱模块之后，用户便进入到构建图谱的上传文件页面，如图 4-33所示。该页面需要用户首先上传满足规范的知识图谱构建原始数据，该数据格式如图 4-2所示。在用户上传了规范的用于构建面向功能点测试的知识图谱原始众包测试报告数据之后，系统将进入到知识图谱的特征提取页面，如图 4-34所示。

图 4-34所示页面展示了正在加载的特征提取模块的主页面。该页面正在加载移动应用 alipay 第二个登录场景的报告文本和屏幕截图。页面中心为特征提取过程的中心板块，页面下方则罗列了当前移动应用的当前场景下文本与截图的特征提取状态。加载完成之后，系统将首先对场景中的报告文本做特征提取操作，如图 4-35所示。

可以从图 4-35中看到报告文本的特征提取操作分为三个模块，首先将原始

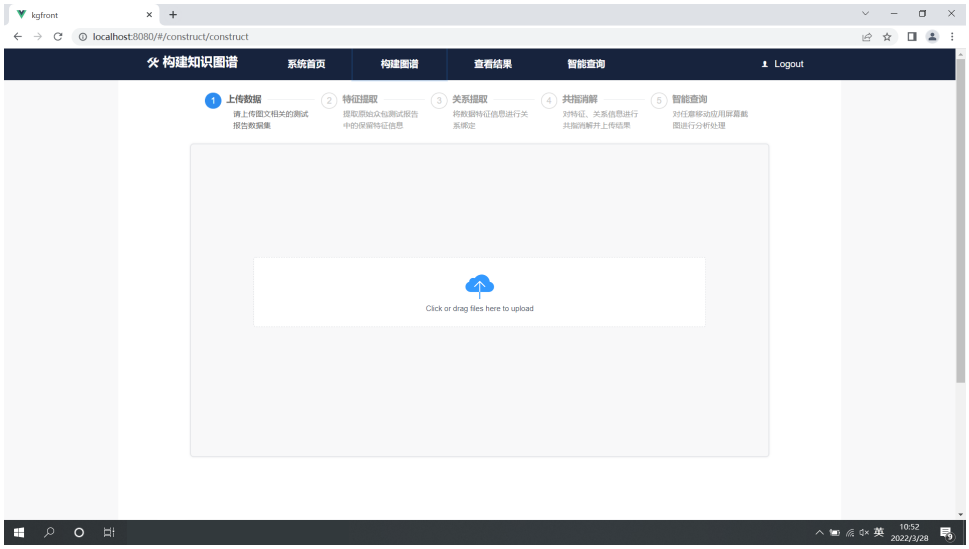


图 4-33: 系统构建图谱模块上传文件页面

图 4-34: 系统构建图谱模块特征提取加载页面

报告文本拆解为复现步骤的排列，然后针对每条步骤进行分词处理，最后通过词性分析识别步骤中的谓语与宾语。

在完成了场景报告文本的特征提取之后，并进入到对原始报告中截图的提取流程中。图 4-36则展示了特征提取屏幕截图时的页面实例。可以看到页面下方的等待队列状态已经更新，同时主面板中主要包括五块内容，分别为原始截图、正在标记中的组件识别截图、正在标记中的文本识别截图、正在标记中的组件控制台输出框、正在标记中的文本控制台输出框。将场景中的截图全部特征提取完成之后，即完成了一个场景的原始报告的特征提取工作。

图 4-35: 系统构建图谱模块特征提取报告提取页面

图 4-36: 系统构建图谱模块特征提取截图提取页面

在完成了原始报告的特征提取工作之后，就进入到了系统构建图谱模块关系提取的加载页，如图 4-37所示。系统在完成对特征提取数据的加载之后即开始关系提取的操作。

关系提取的主面板主要分为五个模块，包括原始报告文本步骤及特征提取结果、原始截图控件提取结果、原始截图文本提取结果、提取过程命令行输出、关系提取结果命令行输出。系统会动态的展示从识别操作、概念到找到匹配文本、控件并识别控件类型的全过程。图 4-38则展示了在完成对移动应用 alipay 场景一的第二个步骤关系提取的结果。

图 4-37: 系统构建图谱模块关系提取加载页面

图 4-38: 系统构建图谱模块关系提取结果展示页面

在对多有特征提取的结果做完关系提取之后，就进入到知识图谱的共指消解页面，如图 4-39所示。页面主要有三个模块组成，分别为知识图谱的图形化展示、构建知识图谱使用的原始报告、知识图谱动态构建的操作输出。

在完成知识图谱的特征提取、关系提取、共指消解之后，并可以通过构建的知识图谱做任意屏幕截图的智能查询。图 4-40展示了通过刚才新建的知识图谱对任意屏幕截图进行知识图谱智能查询并标记关键控件的结果截图。用户首先需要上传需要查询的截图，然后等待知识图谱对截图进行控件、文本、布局的提取与分析，并查阅知识图谱打印最终查询结果。

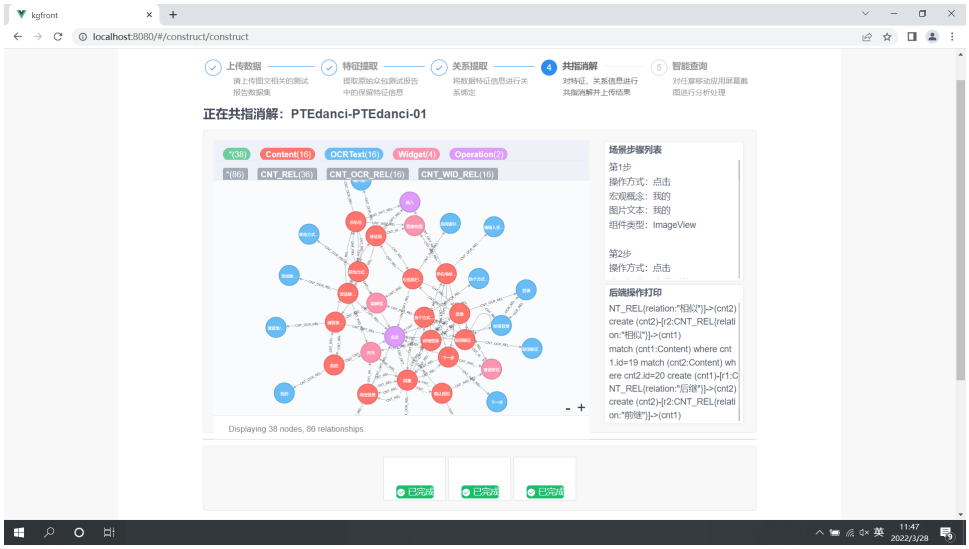


图 4-39: 系统构建图谱模块共指消解结果展示页面

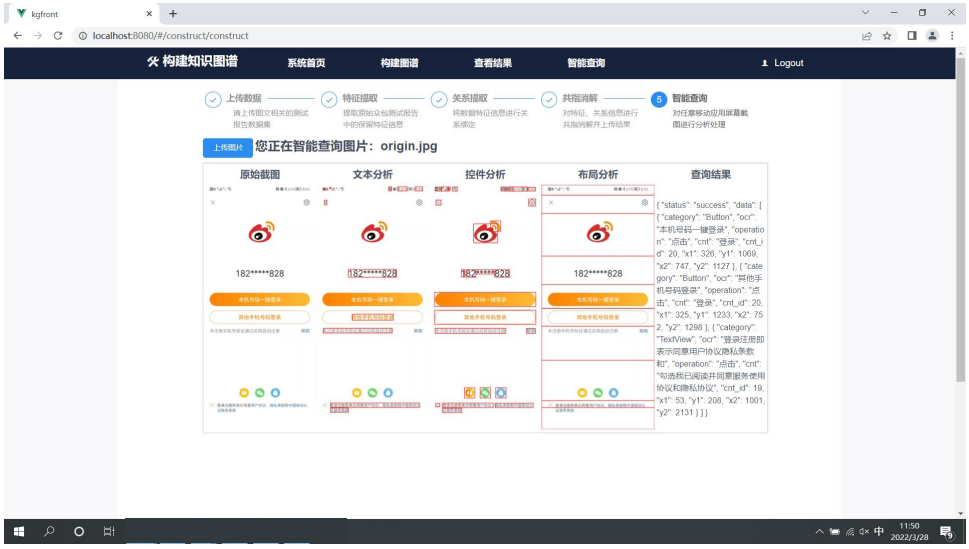


图 4-40: 系统构建图谱模块智能查询结果展示页面

4.5.2 查看结果模块实例展示

用户可以通过查看结果模块查看以往上传的数据以及数据分析、图谱构建的结果。用户可以通过选择功能点、移动应用、场景的方式查看该场景下的原始测试报告、特征提取结果、关系提取结果、知识图谱生成结果。图 4-41则展示了对于登录功能点测试的 alipay 移动应用场景 2 的原始报告及分析、图谱结果。

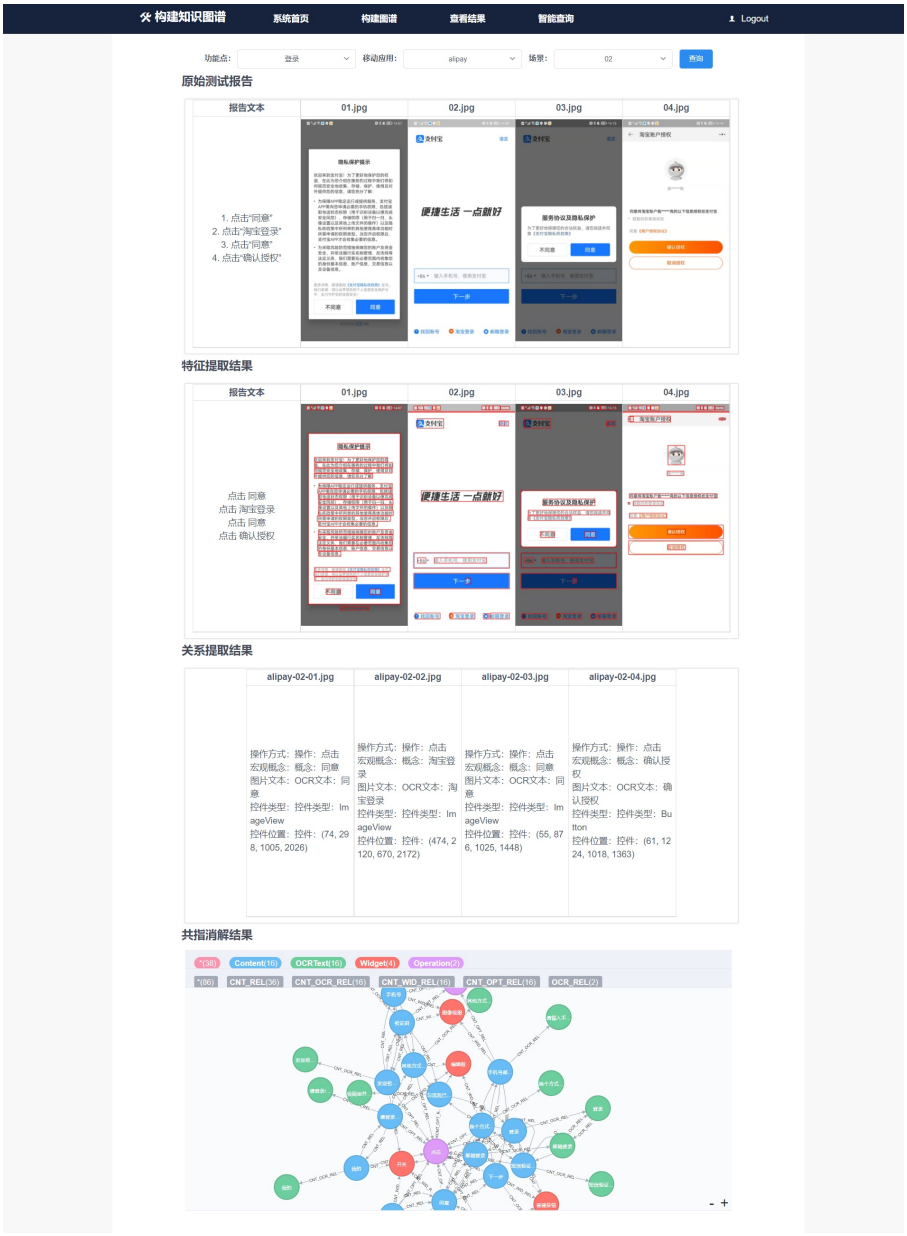


图 4-41: 系统查看结果模块结果展示页面

4.5.3 智能查询模块实例展示

智能查询模块允许用户在选定图谱之后上传任意一张移动应用屏幕截图，并通过查询知识图谱标记出需要被执行的操作。图 4-42则展示了用户在上传查询到屏幕截图之后，系统对图片进行控件、文本、布局的分析并智能查询知识图谱返回结果。

图 4-42: 系统查询图谱模块结果展示页面

4.6 本章小结

本章通过对知识图谱特征提取模块、知识图谱关系提取模块、知识图谱共指消解模块、知识图谱智能查询模块以顺序图、类图、实现代码等方式进行细节描述，阐述了基于移动应用功能点测试的知识图谱构建技术的详细设计实现细节。

第五章　系统测试与运行效果分析

5.1　实验一：知识图谱关系提取成功率

5.1.1　实验目的

本实验旨在对知识图谱自动化构建过程中关系提取的成功率进行统计与分析。在关系提取的过程中，自动化构建过程涉及到对原始文本报告的文本提取，报告提取文本与截图文本的绑定，以及截图文本与所属控件的绑定。通过统计自动化识别与绑定的成功率来分析知识图谱自动化构建技术的可靠性。本实验通过统计计算自动化构建过程的成功率来回答以下研究问题：

RQ1：知识图谱自动化构建过程是否可靠？

5.1.2　实验设置

表 5-1: 知识图谱自动构建涉及应用、场景与步骤数统计表

图谱编号	功能点	构建应用数	构建场景数	构建步骤数
01	登录	25	37	245
02	注册	20	59	343
03	发送邮件	10	10	71
04	查询机票	11	11	49
05	搜索添加购物车	10	10	53
	总计	76	127	761

我们事先自动化构建了 5 个不同的移动应用功能点知识图谱，并在构建的过程中从关系提取模块收集到每一个操作步骤对应的操作绑定情况，并对其进行人工审核，排查关系提取的成功率。五个根据功能点自动构建的知识图谱分别为登录知识图谱图、注册知识图谱、发送邮件知识图谱、查询机票知识图谱以及搜索添加购物车知识图谱。登录知识图谱的构建过程总共涉及到 25 个移动应用中 37 个场景共计 245 次关系提取操作；注册知识图谱的构建过程总共涉及

到 20 个移动应用中 59 个场景共计 343 次关系提取操作；发送邮件知识图谱的构建过程总共涉及到 10 个移动应用中 10 个场景共计 71 次关系提取操作；查询机票知识图谱的构建过程总共涉及到 11 个移动应用中 11 个场景共计 49 次关系提取操作；搜索添加购物车知识图谱的构建过程总共涉及到 10 个移动应用中 10 个场景共计 53 次关系提取操作。五个知识图谱自动化构建时的供涉及 76 款移动应用、127 个场景、761 次关系提取步骤，其具体统计信息如表 5-1所示。

表 5-2: 知识图谱构建过程涉及到的移动应用

图谱编号	功能点	构建涉及应用
01	登录	支付宝、腾讯视频、长龙航空、疯狂背单词、网易云音乐、微博、QQ、海南航空、沪江网校、扇贝听力口语、吉祥航空、乐词新东方背单词、流利说英语、流利说阅读、南方航空、PTE 单词、扇贝单词英语版、中国联合航空、扇贝阅读、深圳航空、拓词、祥鹏航空、中国国航、京东、春秋航空、东方航空、
02	注册	必剪、倒数日、得到、丁香医生、豆果美食、货拉拉、IKEA 宜家家居、金山词霸、孔夫子旧书网、酷我音乐、漫画人、猫眼、美丽修行、Q 房网、人民日报、瑞幸咖啡、扫描全能王、央视频、医鹿、转转
03	发送邮件	139 邮箱、189 邮箱、2980 邮箱、Gmail、Outlook、QQ 邮箱、搜狐邮箱、网易邮箱大师、完美邮箱、新浪邮箱
04	查询机票	长龙航空、春秋航空、东方航空、海南航空、吉祥航空、南方航空、深圳航空、香港航空、祥鹏航空、中国国航、中国联合航空
05	搜索添加购物车	当当、京东、京喜、孔夫子旧书网、苏宁易购、淘宝、淘特、天猫、网易严选、一淘

表 5-2则展示了在自动化构建知识图谱的过程中，5 款功能点知识图谱的构建涉及到的移动应用。这些应用涵盖了影音、社交、教育、音乐等多个应用类别，保证了知识图谱形成的普遍性和客观性。

表 5-3: 知识图谱关系提取过程中报告文本拆解情况统计表

编号	图谱功能点	构建步骤总数	报告文本提取		提取成功率
			提取成功	提取失败	
01	登录	245	242	3	98.78%
02	注册	343	340	3	99.13%
03	发送邮件	71	71	0	100%
04	查询机票	49	49	0	100%
05	搜索添加购物车	53	53	0	100%
	总计	761	755	6	99.21%

在对表 5-2中所有应用的每个场景进行自动化的构建过程后，我们统计了每个步骤的报告文本的拆解内容以及图片文本的匹配内容。我们对于统计并生成的步骤细节进行人工的审核，并标记出正确的报告文本拆解步骤、正确的图片文本匹配步骤。表 5-3则展示了对于上述 5 个功能点 76 个移动应用的 761 个独立步骤报告文本正确拆解的统计结果。在提取报告文本的过程中，我们认为可以将句子中符合情景的谓语、宾语识别出来即表示识别成功。可以看到在对 761 份独立的文本进行拆解分析之后，仅有 6 份步骤拆解的过程出现了问题，报告文本提取的准确率可以达到 99.21%。

表 5-4: 知识图谱关系提取过程中报告、截图文本匹配结果统计表

编号	图谱功能点	构建步骤总数	报告截图文本匹配			匹配成功率
			文本匹配	布局匹配	匹配失败	
01	登录	245	204	29	12	95.10%
02	注册	343	287	52	4	98.83%
03	发送邮件	71	39	32	0	100%
04	查询机票	49	17	28	4	91.84%
05	搜索添加购物车	53	21	31	1	98.11%
	总计	761	568	172	21	97.24%

同时我们也对 76 个移动应用的 761 个独立步骤报告、截图文本匹配情况进行了统计。表 5-4则展示了对于上述 5 个功能点所有独立步骤的报告、截图文本匹配绑定情况。可以看到在对报告截图进行文本匹配的过程中，主要通过两种方式进行匹配，一种是通过报告文本与图片文本的相似度进行匹配，另一种则是通过报告文本配置指定的布局文件进行匹配，整体的匹配成功率达到了 97.24%。

5.1.3 结果分析

我们首先对知识图谱关系提取过程中报告文本拆解情况进行了分析，具体结果统计如表 5-3所示。在表格中我们可以看到无论是每个单项图谱亦或是整体，报告文本提取的成功率都相对较高，整体的提取成功率已达到 99% 以上。这样的结果主要依赖于大部分的复现步骤都是按照既定的理解进行编排的，即句子中包括需要执行的操作及被操作的内容。较高的报告文本提取成功率也提前为后续报告截图文本的匹配成功率打下基础。

表 5-5: 知识图谱关系提取过程中报告、截图文本匹配结果统计表

编号	移动应用	场景	原始文本	拆解操作	拆解内容
01	QQ	02	滑动验证框	验证	框
02	腾讯视频	03	滑动完成验证拼图	完成	验证拼图
03	扇贝听力口语	01	同意协议并登录	登录	同意协议并
04	得到	03	点击微信图标	点击	图标
05	豆果美食	02	点击微信图标	点击	图标
06	Q 房网	01	滑动拼图	拼图	滑动

在得到统计数据之后，我们又对其中 6 份提取失败的报告文本进行了细节的分析，表 5-5展示了这 6 条文本的原始文本信息以及其提取之后的结果。在看到这 6 条报告提取失败的文本之后，我们将问题主要归结为两类：句子分词错误、定语省略问题。其中，编号为 01-03、06 这 4 条文本都是因为句子分词错误导致的。例如失败案例中的编号为 01 的案例，jieba 词库会将文本分解为“滑动”、“验证”、”框” 三个模块，我们提取到宾语为“框”，而离宾语最近的谓语动词即为“验证”，所以该文本就会被拆解为验证与框。由于分词的原因，同时也由于汉语言词语组合排列语法规则多样性的原因，我们不能百分百肯定在对文本进行的分词是符合愿意的，我们也很难保证在分解出来的文本中最后一个谓语和宾语一定是我们需要的操作与内容，所以导致问题的出现。而编号 04、05 这两条文本则由于定于省略的问题而提取失败。我们看到在识别的过程中，我们确实提取到了谓语“点击”与宾语“图标”。然而“图标”与“微信图标”还是有明显差异的。这是在文本的分析过程中过度省略定语而出现的问题。在一般情况下，省略定语是适用的，如果这条文本变为“点击绿色的微信按钮”，那么“绿色”这个定语自然应该被舍弃掉，但是在这里舍弃的过多导致了结果出现问题，这也从侧面反映出在对于特定领域的词语进行分词和解析之前，应该对分词的词库进行特定领域的文本更新，这样才能更高的提高准确性。然而这个问题在本次的知识图谱生成过程中并没有造成确实场景的问题，因为收集到的文本也有直接以“点击微信”为文本的报告，这样就避免了过度省略定语的问题，将结果正确的存入到知识图谱之中了。

报告文本特征提取的高准确率在很大程度上保证了报告、截图文本匹配绑定的准确率。报告、截图文本匹配绑定的统计结果如表 5-4所示。图 5-1则展示了不同功能点在报告、截图文本匹配的过程中通过文本、布局进行识别从而达到准确识别的比率。

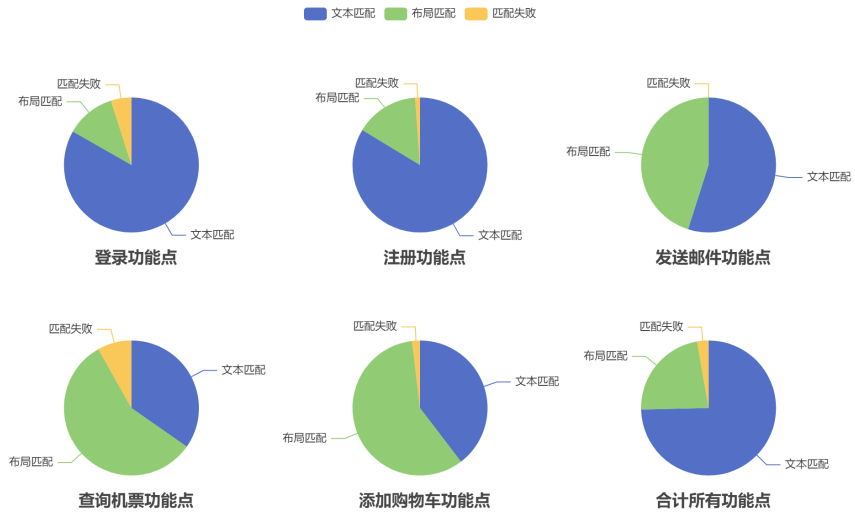


图 5-1: 知识图谱报告、截图文本匹配成功率饼图

首先我们观察到在 5 副不同功能点的饼图中，黄色扇形区域代表的匹配失败的比率均较低，表示在自动化构建的过程中，正确识别报告文本并匹配特定控件的比例是较高的。绿色和蓝色的部分组成了匹配成功的模块，无论是从各功能点来看还是从整体合计功能点来看，都能表现较好效果，整体的匹配成功率也达到了 97.24%。我们将这些未能正确匹配的 21 条文本进行了搜集整理，并将其总结为三类问题，分别是：报告文本拆解出错导致匹配失败、报告文本有误导致匹配失败以及报告文本指向截图中非文本部分。

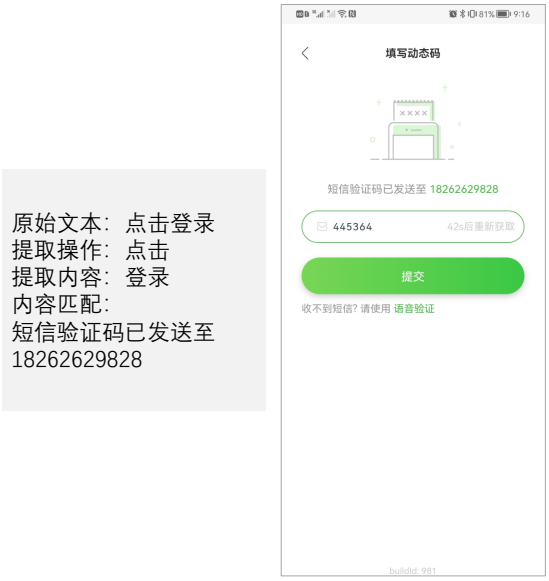


图 5-2: 测试报告错误导致匹配失败样例图

报告文本提取出错导致的匹配失败是指在对报告文本中的文本步骤进行特征提取的过程中，提取时将谓语与宾语弄错，导致后续通过错误的宾语去匹配截图中的控件也开始报错。

另一种问题则是由于报告文本有误导致匹配失败。我们收集到的原始众包测试报告尽管满足了特定的规格版本，但我们并不能确认文本与图片之间存在强相关关系，也就是我们暂时无法判断其是否为图文相关的报告。不乏有少量的报告由于众包测试报告填写人员的忽略导致错误，例如图 5-2中显示的实例，我们可以看到用户填写的报告中，在该页面是点击“登录”按钮并完成登录，可是纵观整张截图可知，真正需要点击的其实是“提交按钮”。由于众包测试报告的错误而导致了最终匹配结果的错误。

图 5-3: 通过布局配置完成报告文本与控件绑定样例图

第三种问题则是由于报告文本指向截图中的非文本部分所导致的匹配失败。我们通过报告文本的特征提取，找到特定的逻辑概念，并从报告截图中根据截图中的文字和布局匹配找到特定的控件作为匹配的结果。对于文本不能匹配的内容来说，一般可以通过配置布局来实现。例如图 5-4所示的步骤，我们可以通过配置使得“微信登录”自动匹配到图片中的微信图标即可。这类报告是有规律可以归纳的。然而对于有些报告则无法通过统一配置进行总结归纳帮助完成匹配任务。例如图所示的完成人机验证任务，这本就是用来排除机器直接执行的操作，在这里自动化构建就会显得有些无能为力。由于自动化验证的图片会变化，自动化验证的内容也可能变成其他非图片的形式，这就导致不能

通过统一的布局配置来对匹配结果进行优化，所以导致了错误的产生。

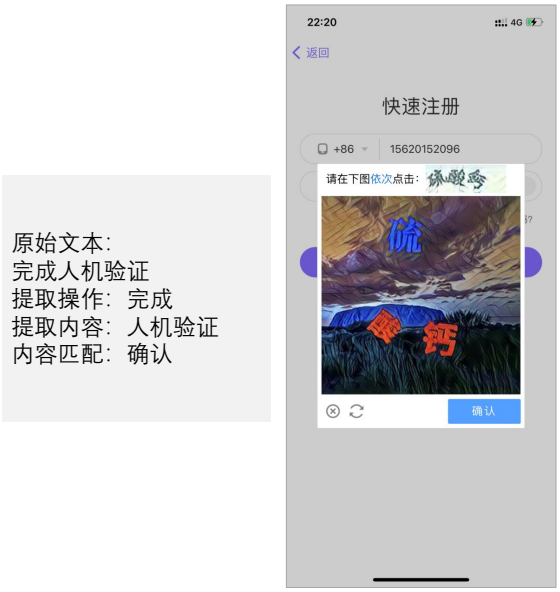


图 5-4: 无法通过文本与布局对控件进行匹配的样例图

在完成对无法匹配的步骤进行分析之后，我们对统计数据中通过文本匹配、布局匹配而锁定控件的步骤数量进行了简单的分析和总结。我们看到在不同的功能点中，尽管最终能够完成匹配的比率基本稳定在九成以上，但是在完成匹配的步骤中，文本匹配与布局匹配的占比却有着比较大的差异。如图 5-1所示，登录和注册功能点中以文本匹配为主，发送邮件功能点文本与布局匹配的能力相差无几，而查询机票功能点、添加购物车功能点则主要以布局配置为主。可以看到，对于不同的功能点来说，文本匹配与布局匹配的重要性是不同的。这也预示着在构建知识图谱之前，需要进行简单的研判，判断该功能点是依赖文本更多还是依赖布局更多，从而增强相关方面的配置，以保证知识图谱的正确构建。

从实验一中我们可以看到，在知识图谱的构建过程中，关键的关系提取模块的提取、匹配成功率均较高，前者可以达到 99.21% 的成功率，而后者也可以达到 97.24% 的成功率，这证明通过文本与布局同时对报告进行分析并从事关系提取工作是可行的。

5.2 实验二：面向功能点的屏幕截图预测能力

5.2.1 实验目的

本实验旨在对使用系统自动化构建生成的知识图谱进行面向功能点的屏幕截图预测，并对其预测能力进行分析。在选定某个功能点之后，知识图谱可以通过智能查询模块对涉及该功能点操作的屏幕截图进行图片的分析，找出在该截图中可能需要被操作的控件，并将为何操作，如何操作，操作坐标信息打印返回，从而对自动化构建的面向移动应用功能点测试的知识图谱进行预测能力的评估。实验二主要研究以下问题：

RQ2：自动化构建的知识图谱预测能力是否达标？

5.2.2 实验设置

在实验过程中，我们主要对登录功能点自动化生成的知识图谱进行了截图预测能力的测试。主要有以下几个原因：首先，形成登录功能点知识图谱的原始数据更多，保证了图谱的完整性，同时也保证了图谱的客观性；第二，登录知识图谱的构建过程中即涉及到文本与控件的匹配也包含布局与控件的匹配，测试与实验的范围更全；第三，登录是上一节中提到的五个功能点中较复杂的一个，涉及的场景更多，预测的流程更难，实验的结果更可性。因此，我们选择登录功能点的知识图谱作为面向功能点的屏幕截图预测能力实验的知识图谱。

表 5-6: 知识图谱关系提取过程中报告、截图文本匹配结果统计表

编号	移动应用	场景数	截图数
01	58 同城	10	15
02	赶集直招	4	9
03	小红书	10	20
04	抖音	6	16
05	soul	3	9
06	得物	6	14
	总计	39	83

我们挑选了 10 款未用来生成知识图谱的应用作为测试集来测试登录功能点

的知识图谱。我们通过对这 10 款移动应用进行人工的操作，将 10 款移动应用中的场景分别记录下来，并把每个移动应用在各场景下执行操作的所有截图均进行保存，然后通过已有的登录功能点知识图谱对这些截图进行预测，并计算其精确率、召回率，并标记其 F 值。表 5-6罗列了测试过程中用到的 10 款移动应用名称，并统计了每款应用在登录时涉及到的场景数以及场景涉及到的屏幕截图数。

我们的评估方式涉及到精确率与召回率。其中准确率的计算公式如下：

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5-1}$$

其中 ACC 即为精确率，TP 表示被知识图谱预测为正类的正样本，TN 表示被知识图谱预测为负类的负样本，FP 表示被知识图谱预测为正类的负样本，FN 则表示被知识图谱预测为负类的正样本。其中正类即表示某个组件被预测为需要进行操作，负类则表示某个组件被预测为不需要被进行操作；正样本即意味着该组件确实需要被操作，而负样本则意味着该组件确实不需要被操作。精确率可以显示出识别成功的准确率，该值越大即意味着预测正确的准确率越高。

召回率的计算公式如下：

$$R = \frac{TP}{TP + FN} \tag{5-2}$$

其中 TP、FN 的定义与准确率相同。该评价指标指明了对于所有应该被操作的控件来说，被知识图谱正确预测出来的比率。

5.2.3 结果分析

在对数据进行审核之前，我们统计了 6 款应用涉及到的 83 张屏幕截图中可操作的控件总数以及推荐进行操作的控件总数，具体信息如表 5-7所示。从表中我们可以看到，在 83 张屏幕截图中，总共涉及到 3722 个控件待操作。如果移动应用自动化测试时使用全局测试注一点击的话工作量将会非常大。这 3722 个不同的操作肯能会生成更多的后续操作，指数级的上涨模式使得测试的深度非常有限。然而通过知识图谱进行预测之后我们可以从 3722 个控件中预测出其中 378 个为在当前功能点下应该进行操作的控件，可以看到从 3722 到 378，我们将操作控件的效率提高了 89.84%。这可以在很大程度上加大自动化测试的力度，同时帮助自动化测试进入到更深的层次。

表 5-7: 屏幕截图中的控件总数与预测控件总数统计表

编号	移动应用	截图数	控件数	预测数	优化率
01	58 同城	15	803	75	92.52%
02	赶集直招	9	481	36	90.66%
03	小红书	20	1002	94	84.17%
04	抖音	16	595	58	87.38%
05	soul	9	278	44	90.25%
06	得物	14	563	71	90.61%
	总计	83	3722	378	89.84%

我们已经通过知识图谱将需要操作的控件数量降低了九成，如果我们提取出来需要操作的一成控件可以覆盖大部分需要覆盖的控件，则表示我们的知识图谱可以有效的预测并极大程度的简化了复杂度。表 5-8则展示了知识图谱在对 6 款应用进行预测之后其预测为正类的正样本（TP）、预测为负类的负样本（TN）、预测为正类的负样本（FP）以及预测为负类的正样本（FN）的数量，并计算了其精确率（ACC）和召回率（R）。

表 5-8: 知识图谱预测结果精确率与召回率统计表

编号	移动应用	TP	TN	FP	FN	ACC	R
01	58 同城	33	727	42	1	94.64%	97.06%
02	赶集直招	14	443	22	2	95.01%	87.50%
03	小红书	40	903	54	5	94.11%	88.89%
04	抖音	25	534	33	3	93.95%	89.29%
05	soul	18	2338	26	1	94.74%	94.74%
06	得物	27	489	44	3	91.65%	90.00%
	总计	157	3329	221	15	93.66%	91.28%

在表 5-8中我们可以看到，对于六款不同的移动应用，每款应用预测的精确率均在 91% 以上，其整体的精确率达到了 93.66%。这样高的精确率即意味着对于预测出来的控件来说，有极高的精确性，预测正确的概率已经达到了 93.66%。同时这 6 款移动应用的召回率也表现很好，其平均召回率也达到了 91.28%。这表明对于所有应该操作的控件，我们可以成功从中预测出 91.28% 的控件进行测试操作。从表格数据的情况来看，表现整体较好。

较高的优化率表明在整体预测的过程中知识图谱可以极大程度的简化操作的流程和范围，进而加速对移动应用的自动化测试，同时加深移动应用自动化

测试的层次；较高的准确率和召回率则表明对于预测出来的操作控件来说，可以较高程度的代表所有需要被操作的控件，覆盖率较高。

5.3 本章小结

本章对面向移动应用功能点测试的知识图谱构建技术进行了实验与评估。本章一共设置了两个实验：知识图谱关系提取成功率和面向功能点的屏幕截图预测能力。这两个实验分别回答了两个问题：知识图谱自动化构建过程是否可靠？自动化构建的知识图谱预测能力是否达标。最终我们发现在关系提取过程中系统可以完成 97.24% 的正确匹配，在截图预测的过程中知识图谱可以简化 89.84% 的冗余控件并达到 93.66% 的精确率和 91.28% 的召回率。即表明自动化构建的面向移动应用功能点的知识图谱不仅构架可靠，同时预测能力也达标。

第六章　总结与展望

6.1　总结

随着移动应用设备的大量普及和移动应用产品的快速迭代，市面上对移动应用自动化测试的要求越来越高。然而主流的移动应用自动化测试框架却因其逻辑覆盖不全或迭代兼容过差而加大了人力成本与时间成本。在对市面上主流的移动应用产品进行统计分析后我们发现移动应用产品之间尽管因版本、用途等原因存在差异，但他们之间却存在着某些逻辑层面相似的部分，我们将这些相似的部分称为移动应用自动化测试的功能点。同时，我们发现在这些功能点的内部存在着一些特定的知识联系。例如操作组件的顺序、操作组件的类型等。如果我们可以把这些内在联系整合起来，并将其用于现有的移动应用自动化测试脚本框架中，即可帮助移动应用进行兼容性更高的，消耗更小的移动应用自动化测试。

基于这些问题和背景，本文实现了面向移动应用功能点测试的知识图谱构建技术。本文通过从高质量的众包测试报告中获取原始的移动应用众包测试数据，包括针对某一功能点的复现步骤以及操作的屏幕截图来生成面向移动应用功能点测试的知识图谱。系统首先通过知识图谱的特征提取模块，使用百度OCR、Canny 算法、jieba 词库将原始报告数据进行拆解，获得截图中的文本、控件及文本中的动名词关系。之后我们将特征提取模块生成的数据交由知识图谱关系提取模块，将文本中的动名词与截图中的文本、控件进行绑定，识别出对于当前报告而言整体操作的流程，操作过程前后继顺序，操作节点的相似性关系，操作实体的首尾特性等关系作为结果进行保存。之后，将关系提取模块保存下来的结果交由知识图谱共指消解模块对数据中的冗余进行删减排查，并将数据上传至 Neo4j 图数据库，完成知识图谱的构建操作。随后，用户既可以通过上传任意屏幕截图的方式对特定功能点的知识图谱进行智能查询操作。系统将根据以往的操作信息从知识图谱中截取到相关实体节点，并对上传的屏幕截图进行文本、控件、布局的分析，在通过布局与文本的双重匹配之后，找到上传截图与知识图谱中的匹配节点，以列表的形式返回给用户，以此来帮助移

动应用自动化测试的顺利进行。

本文对系统的构建过程以及图谱的构建质量进行了实验与评估。在对 76 个移动应用、127 个构建场景、761 个构建步骤构建出来的 5 个功能点知识图谱当中，其绑定的准确率达到了 99.21%。同时，我们对 6 款移动应用、39 个应用场景、83 张屏幕截图进行了知识图谱的预测分析，其相对于遍历测试点击数目的优化率达到了 89.84%，且整体的精确率和召回率也分别达到了 93.66% 和 91.28%。这足以说明面向移动应用功能点测试的知识图谱构建技术的稳定性与可用性。

6.2 展望

目前，本系统的自动化构建过程已经确定，然而在现在的构建和查询过程中仍然存在一些问题。

(1) 知识图谱的查询过程需要对知识图谱的布局进行配置。在配置过程中用户需要对截图进行布局分析并人工配置布局信息。在确定功能点的情况下，这样的配置只会占用较小的时间成本，但是当原始数据数量庞大同时功能点选择丰富多样时便会加大人力成本开销。如何将布局配置转化为全自动化的方式是我们后续需要去继续挖掘的方向之一。

(2) 知识图谱的智能查询模块仍存在一定的效率问题。由于需要对上传的查询截图进行文本与控件的提取，在对控件类型进行分析的过程中会占用较多的时间，这会拖慢整个搜索查询知识图谱的进程。因此，后续改进控件类型分析的 CNN 模型，或提供一种减少 CNN 模型预测次数的机制是必要的。

(3) 知识图谱构建的原始数据集格式限定了数据集的数量，这也会影响后续知识图谱构建的完整性。因此，提供一种可以从海量数据中结构化提取信息的方法也是未来改进的方向之一。

致　　谢

感谢在实验室度过的两年时光，尤其感谢房春荣老师在研究领域的细节指导，让我研究生阶段在学术方面有了更深刻的认识。还要在这里感谢陈振宇老师，和您开会确实很高效，既有压力又有动力，感觉每次你提的意见都是直中要害，这确实在我后来的科研生涯中起到了非常大的作用。除此以外，还要感谢实验室的小伙伴吧，因为氛围好所以研究也没那么枯燥了。接着想感谢一下我的父母，今年是他们供我读书的第 24 年了。谢谢你们一路的支持，也谢谢你们一向尊重我的想法，这篇文章通过后，我就真的毕业了。谢谢这些年来大家的支持，希望我们都有美好的未来。

参考文献

[1] MUSTHAFA F N, MANSUR S, WIBAWANTO A. Automated Software Testing on Mobile Applications: A Review with Special Focus on Android Platform[C/OL] // 2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer). 2020 : 292 – 293. http://dx.doi.org/10.1109/ICTer51097.2020.9325445.

[2] WETZLMAIER T, RAMLER R, PUTSCHöGL W. A Framework for Monkey GUI Testing[C/OL] // 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST). 2016 : 416 – 423. http://dx.doi.org/10.1109/ICST.2016.51.

[3] ZUN D, QI T, CHEN L. Research on automated testing framework for multi-platform mobile applications[C/OL] // 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS). 2016 : 82 – 87. http://dx.doi.org/10.1109/CCIS.2016.7790229.

[4] DUAN Y, SHAO L, HU G, et al. Specifying architecture of knowledge graph with data graph, information graph, knowledge graph and wisdom graph[C/OL] // 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA). 2017 : 327 – 332. http://dx.doi.org/10.1109/SERA.2017.7965747.

[5] MUSTHAFA F N, MANSUR S, WIBAWANTO A. Automated Software Testing on Mobile Applications: A Review with Special Focus on Android Platform[C/OL] // 2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer). 2020 : 292 – 293. http://dx.doi.org/10.1109/ICTer51097.2020.9325445.

[6] CHOUDHARY S R, GORLA A, ORSO A. Automated Test Input Generation for Android: Are We There Yet? (E)[C/OL] // 2015 30th IEEE/ACM International

Conference on Automated Software Engineering (ASE). 2015 : 429 – 440. http://dx.doi.org/10.1109/ASE.2015.89.

[7] MACHIRY A, TAHILIANI R, NAIK M. Dynodroid: an input generation system for Android apps[J], 2013.

[8] GOMEZ L, NEAMTIU I, AZIM T, et al. RERAN: Timing- and touch-sensitive record and replay for Android[C/OL] // 2013 35th International Conference on Software Engineering (ICSE). 2013 : 72 – 81. http://dx.doi.org/10.1109/ICSE.2013.6606553.

[9] ANBUNATHAN R, BASU A. Data driven architecture based automated test generation for Android mobile[C/OL] // 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). 2015 : 1 – 5. http://dx.doi.org/10.1109/ICCIC.2015.7435772.

[10] LI Y, YANG Z, GUO Y, et al. Humanoid: A Deep Learning-Based Approach to Automated Black-box Android App Testing[C/OL] // 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2019 : 1070 – 1073. http://dx.doi.org/10.1109/ASE.2019.00104.

[11] FU X, REN X, MENGSHOEL O J, et al. Stochastic Optimization for Market Return Prediction Using Financial Knowledge Graph[C/OL] // 2018 IEEE International Conference on Big Knowledge (ICBK). 2018 : 25 – 32. http://dx.doi.org/10.1109/ICBK.2018.00012.

[12] CHENG B, ZHANG Y, CAI D, et al. Construction of traditional Chinese medicine Knowledge Graph using Data Mining and Expert Knowledge[C/OL] // 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC). 2018 : 209 – 213. http://dx.doi.org/10.1109/ICNIDC.2018.8525665.

[13] ZHAO Y, YANG Y, TIAN B, et al. An Invocation Chain Test and Evaluation Method Based on Knowledge Graph[C/OL] // 2020 International Conference on

Networking and Network Applications (NaNA). 2020 : 277 – 281.
http://dx.doi.org/10.1109/NaNA51271.2020.00054.

[14] KESRI V, NAYAK A, PONNALAGU K. AutoKG - An Automotive Domain Knowledge Graph for Software Testing: A position paper[C/OL] // 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). 2021 : 234 – 238.
http://dx.doi.org/10.1109/ICSTW52544.2021.00047.

[15] KE W, WU C, FU X, et al. Interpretable Test Case Recommendation based on Knowledge Graph[C/OL] // 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS). 2020 : 489 – 496.
http://dx.doi.org/10.1109/QRS51102.2020.00068.

[16] WHITTAKER J A. What Is Software Testing? And Why Is It So Hard? Practice Tutorial[J]. IEEE SOFTWARE, 2000.

[17] VAJAK D, GRBIć R, VRANJEš M, et al. Environment for Automated Functional Testing of Mobile Applications[C/OL] // 2018 International Conference on Smart Systems and Technologies (SST). 2018 : 125 – 130.
http://dx.doi.org/10.1109/SST.2018.8564626.

[18] 张吉祥, 张祥森, 武长旭, et al. 知识图谱构建技术综述 [J]. 计算机工程, 2022, 48(03) : 23 – 37.

[19] 刘峤, 李杨, 段宏, et al. 知识图谱构建技术综述 [J]. 计算机研究与发展, 2016, 53(03) : 582 – 600.

[20] JI S, PAN S, CAMBRIA E, et al. A Survey on Knowledge Graphs: Representation, Acquisition and Applications[J/OL]. CoRR, 2020, abs/2002.00388.
https://arxiv.org/abs/2002.00388.

[21] 段宏. 知识图谱构建技术综述 [J]. 计算机研究与发展, 2016, 53(3) : 19.

[22] 黄恒琪, 于娟, 廖晓, et al. 知识图谱研究综述 [J]. 计算机系统应用, 2019, 28(06) : 1 – 12.

[23] 漆桂林, 高桓, 吴天星. 知识图谱研究进展 [J]. 情报工程, 2017, 3(01) : 4 – 25.

[24] 徐增林, 盛泳潘, 贺丽荣, et al. 知识图谱技术综述 [J]. 电子科技大学学报, 2016, 45(04) : 589 – 606.

[25] FAN J, KALYANPUR A, GONDEK D, et al. Automatic knowledge extraction from documents[J/OL]. IBM J. Res. Dev., 2012, 56(3) : 5. https://doi.org/10.1147/JRD.2012.2186519.

[26] ETZIONI O, CAFARELLA M J, DOWNEY D, et al. Web-scale information extraction in knowitall: (preliminary results)[C/OL] // FELDMAN S I, URETSKY M, NAJORK M, et al. Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004. [S.l.] : ACM, 2004 : 100 – 110. https://doi.org/10.1145/988672.988687.

[27] FADER A, SODERLAND S, ETZIONI O. Identifying Relations for Open Information Extraction[C/OL] // Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL. [S.l.] : ACL, 2011 : 1535 – 1545. https://aclanthology.org/D11-1142/.

[28] NOY N F, MUSEN M A. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment[C/OL] // KAUTZ H A, PORTER B W. Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA. [S.l.] : AAAI Press / The MIT Press, 2000 : 450 – 455. http://www.aaai.org/Library/AAAI/2000/aaai00-069.php.

[29] ZHENG Z, LIU Y, ZHANG Y, et al. TCMKG: A Deep Learning Based Traditional Chinese Medicine Knowledge Graph Platform[C/OL] // 2020 IEEE International Conference on Knowledge Graph (ICKG). 2020 : 560 – 564. http://dx.doi.org/10.1109/ICBK50248.2020.00084.

[30] SHAALAN K. A Survey of Arabic Named Entity Recognition and Classification[J/OL]. Comput. Linguistics, 2014, 40(2): 469–510. https://doi.org/10.1162/COLI_a_00178.

[31] MIWA M, BANSAL M. End-to-end Relation Extraction using LSTMs on Sequences and Tree Structures[J/OL]. CoRR, 2016, abs/1601.00770. http://arxiv.org/abs/1601.00770.

[32] GUAN S, CHENG X, BAI L, et al. What is Event Knowledge Graph: A Survey[J/OL]. CoRR, 2021, abs/2112.15280. https://arxiv.org/abs/2112.15280.

[33] 项威. 事件知识图谱构建技术与应用综述 [J]. 计算机与现代化, 2020(1): 7.

[34] DING Y, TENG F, ZHANG P, et al. Research on Text Information Mining Technology of Substation Inspection Based on Improved Jieba[C/OL] // 2021 International Conference on Wireless Communications and Smart Grid (ICWCSG). 2021: 561–564. http://dx.doi.org/10.1109/ICWCSG53609.2021.00119.

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

[36] CANNY J F. A Variational Approach to Edge Detection[C/OL] // GENESERETH M R. Proceedings of the National Conference on Artificial Intelligence, Washington, D.C., USA, August 22-26, 1983. [S.l.]: AAAI Press, 1983: 54–58. http://www.aaai.org/Library/AAAI/1983/aaai83-030.php.

[37] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks[C/OL] // BARTLETT P L, PEREIRA F C N, BURGES C J C, et al. Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. 2012: 1106–1114. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[38] ZEILER M D, FERGUS R. Visualizing and Understanding Convolutional Networks[J/OL]. CoRR, 2013, abs/1311.2901.
http://arxiv.org/abs/1311.2901.

[39] SERMANET P, EIGEN D, ZHANG X, et al. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks[C/OL] // BENGIO Y, LECUN Y. 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings. 2014.
http://arxiv.org/abs/1312.6229.

[40] LOWE D G. Object recognition from local scale-invariant features[C] // Proc of IEEE International Conference on Computer Vision. 1999.

[41] BROWN M, LOWE D G. Invariant Features from Interest Point Groups[C] // Proceedings of the British Machine Vision Conference 2002, BMVC 2002, Cardiff, UK, 2-5 September 2002. 2002.

[42] LOWE D. Distinctive image features from scaleinvariant keypoints[J]. International Journal of Computer Vision, 2004, 60.

[43] PANORAMAS R, BROWN M, LOWE D G. Recognising panoramas[C] // Proceedings Ninth IEEE International Conference on Computer Vision. 2008.

[44] LU H, HONG Z, SHI M. Analysis of film data based on Neo4j[C/OL] // 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS). 2017 : 675 – 677.
http://dx.doi.org/10.1109/ICIS.2017.7960078.

[45] 许鑫, 岳金钊, 赵锦鹏, et al. 小麦品种知识图谱构建与可视化研究 [J]. 计算机系统应用, 2021, 30(06) : 286 – 292.

简历与科研成果

基本信息

王旭，男，汉族，1998 年 02 月出生，宁夏银川人。

教育背景

2020 年 9 月 — 2022 年 6 月 南京大学软件学院 硕士

2016 年 9 月 — 2020 年 6 月 南京大学软件学院 本科

攻读工程硕士学位期间完成的学术成果

1. Yu, S., Fang, C., Cao, Z., Wang, X., Chen, Z. . (2021). Prioritize crowdsourced test reports via deep screenshot understanding.
2. Cao, Z., Wang, X., Yu, S., Yun, Y., Fang, C.. (2020). STIFA: crowdsourced mobile testing report selection based on text and image fusion analysis. ASE '20: 35th IEEE/ACM International Conference on Automated Software Engineering. ACM.