南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目	基于深度语义匹配的测试用例复用系统
作 者 姓 名	刘智彪
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

2022 年 5 月 20 日

学　　　　号：MF20320098

论文答辩日期：**2022 年 05 月 20 日**

指　导　教　师：　　　　　（**签字**）

Test Case Reuse System Based on Deep Semantic Matching

By

Zhibiao Liu

Supervised by

Professor Zhenyu Chen

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

Software Institute

May 20, 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：基于深度语义匹配的测试用例复用系统

工程硕士（软件工程领域）专业 2020 级硕士生姓名：刘智彪

指导教师（姓名、职称）：陈振宇 教授

摘 要

软件测试是软件质量和可靠性的重要保障手段，是软件开发过程中不可或缺的环节。测试用例的设计是软件测试的关键过程。设计良好的测试用例是开展软件测试的前提，测试用例的质量决定软件测试的质量和效果。随着软件版本持续迭代，在软件测试的过程中会积累丰富的历史测试用例。随着系统的功能点持续增加，所需测试用例数量也在不断增加，这造成了测试人员工作效率降低以及测试周期延长。测试用例复用是使用累积的历史测试用例等资料来进行新测试的方法。如何利用好历史测试用例，做好新的测试任务，提高测试人员工作效率，这是一个有实际意义的问题。

针对上述问题，本文设计并实现了基于深度语义匹配的测试用例复用系统。系统提供测试用例标准模板，帮助用户组织并上传历史测试用例，并提供测试用例管理服务。系统对历史测试用例进行规范化预处理，运用自然语言处理技术提取文本信息，对测试用例关键字段进行向量化处理，为其建立可快速检索的索引。用户输入测试功能点描述后，系统使用深度语义匹配技术为其推荐合适的历史测试用例，用户可根据实际需求选择相应的历史测试用例进行收藏、复制或修改以完成新的测试任务，达到复用目的。

本系统使用 Spring Boot 作为后端框架，Vue 作为前端框架，支撑与保障系统的基本运行流程。使用 MySQL 和 Elasticsearch 作为数据库存储测试用例基础数据和测试用例分析数据，使用 Redis 作为缓存技术，提高了系统的可用性。

本系统提供了测试用例分析、管理和复用功能，通过推荐可复用的历史测试用例，辅助测试人员编写新测试用例。系统的效果测试选取了文本匹配领域的 3 个测试数据集，评价指标选用斯皮尔曼系数，匹配效果优于其他基线算法，实验结果还经过人工评测，验证了系统的使用效果，能在真实使用场景中提高测试人员的工作效率。

关键词：软件测试，测试用例复用，深度语义匹配

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Test Case Reuse System Based on Deep Semantic Matching

SPECIALIZATION: Software Engineering

POSTGRADUATE: Zhibiao Liu

MENTOR: Professor Zhenyu Chen

Abstract

Software testing is an important means of guaranteeing software quality and reliability, and an indispensable link in the software development process. The design of test cases is a key process of software testing. Well-designed test cases are the premise of software testing, and the quality of test cases determines the quality and effect of software testing. With the continuous iteration of software versions, a wealth of historical test cases will be accumulated in the process of software testing. As the function points of the system continue to increase, the number of required test cases is also increasing, which reduces the tester's work efficiency and prolongs the test cycle. Test case reuse is a method of using accumulated historical test cases and other data to conduct new tests. How to make good use of historical test cases, do new test tasks well, and improve the work efficiency of testers, this is a practical problem.

Aiming at the above problems, this paper designs and implements a test case reuse system based on deep semantic matching. The system provides a test case standard template to help users organize and upload historical test cases, and provides test case management services. The system normalizes and preprocesses historical test cases, uses natural language processing technology to extract text information, performs vectorization processing on key fields of test cases, and establishes indexes that can be retrieved quickly. After the user enters the description of the test function point, the system uses the deep semantic matching technology to recommend appropriate historical test cases for it. Users can select corresponding historical test cases to collect, copy or modify according to actual needs to complete new test tasks.

The system uses Spring Boot as the back-end framework and Vue as the front-end framework to support and guarantee the basic operation process of the system. It uses

MySQL and Elasticsearch to store test case basic data and test case analysis data. It uses Redis as the cache technology to improve the availability of the system.

The system provides test case analysis, test case management and test case reuse functions, and assists testers to write new test cases by recommending reusable historical test cases. The effect test of the system selects three test data sets in the field of text matching and the evaluation index uses the Spearman coefficient. Experiments show that the matching effect is better than that of other baseline algorithms. The experimental results have also been manually evaluated to verify the use of the system, which can improve the work efficiency of testers in real use scenarios.

Keywords: Software Testing, Test Case Reuse, Deep Semantic Matching

目录

表 目 录 ········ **viii**

图 目 录 ········ **x**

第一章 引言 ········ **1**

1.1 项目背景和意义 ········ 1

1.2 国内外研究现状 ········ 2

1.2.1 测试用例复用研究现状 ········ 2

1.2.2 深度文本匹配研究现状 ········ 3

1.3 本文主要工作 ········ 3

1.4 本文组织结构 ········ 4

第二章 相关技术概述 ········ **6**

2.1 NLP 技术分析 ········ 6

2.1.1 词向量模型 ········ 6

2.1.2 Word2Vec ········ 6

2.2 文本匹配 ········ 7

2.3 深度语义匹配 ········ 8

2.4 相关工程技术 ········ 9

2.4.1 Elasticsearch ········ 9

2.4.2 Redis ········ 10

2.5 本章小结 ········ 10

第三章 基于深度语义匹配的测试用例复用系统的需求分析与设计 ········ **11**

3.1 系统整体概述 ········ 11

3.2 系统需求分析 ········ 11

3.2.1 测试用例分析模块需求分析 ········ 11

3.2.2 测试任务管理模块需求分析 ········ 13

3.2.3 测试用例匹配模块需求分析 · · · · 14
3.2.4 非功能性需求分析 · · · · 15
3.2.5 系统用例描述 · · · · 16
3.3 系统概要设计 · · · · 22
3.3.1 系统架构设计 · · · · 22
3.3.2 系统模块划分 · · · · 24
3.3.3 系统视图模型 · · · · 24
3.4 基于深度语义匹配的测试用例复用技术 · · · · 27
3.4.1 测试用例定义 · · · · 27
3.4.2 负采样技术 · · · · 29
3.4.3 测试用例匹配 · · · · 30
3.5 测试用例分析模块设计 · · · · 33
3.5.1 架构设计 · · · · 33
3.5.2 详细设计 · · · · 33
3.5.3 数据库设计 · · · · 35
3.6 测试任务管理模块设计 · · · · 37
3.6.1 架构设计 · · · · 37
3.6.2 详细设计 · · · · 38
3.6.3 数据库设计 · · · · 38
3.7 测试用例匹配模块设计 · · · · 40
3.7.1 架构设计 · · · · 40
3.7.2 详细设计 · · · · 41
3.7.3 数据库设计 · · · · 42
3.8 本章小结 · · · · 43

第四章 基于深度语义匹配的测试用例复用系统的实现 · · · · 44
4.1 测试用例分析模块的实现 · · · · 44
4.1.1 文件传输实现 · · · · 44
4.1.2 测试用例分析流程 · · · · 47
4.1.3 关键代码 · · · · 50
4.2 测试任务管理模块的实现 · · · · 52

4.2.1 测试任务管理流程 …… 52
4.2.2 关键代码 …… 55
4.3 测试用例匹配模块的实现 …… 56
4.3.1 测试用例匹配流程 …… 56
4.3.2 关键代码 …… 57
4.4 本章小结 …… 58

第五章 基于深度语义匹配的测试用例复用系统的测试与实验分析 …… 60
5.1 测试准备 …… 60
5.1.1 测试目标 …… 60
5.1.2 测试环境 …… 60
5.2 功能测试 …… 60
5.3 性能测试 …… 64
5.3.1 测试设计 …… 64
5.3.2 测试结果 …… 65
5.4 可用性测试 …… 66
5.4.1 测试设计 …… 66
5.4.2 测试结果 …… 67
5.5 效果测试 …… 67
5.5.1 文本匹配效果测试 …… 67
5.5.2 测试用例复用效果测试 …… 68
5.6 本章小结 …… 70

第六章 总结与展望 …… 71
6.1 总结 …… 71
6.2 展望 …… 72

参考文献 …… 73

简历与科研成果 …… 76

致谢 …… 77

表 目 录

3.1 测试用例分析模块功能性需求列表 …… 13
3.2 测试任务管理模块功能性需求列表 …… 14
3.3 测试用例匹配模块功能性需求列表 …… 15
3.4 下载数据集模板用例描述 …… 16
3.5 上传测试用例集用例描述 …… 17
3.6 查看测试用例集列表用例描述 …… 17
3.7 测试用例集检索用例描述 …… 18
3.8 查看测试用例详情用例描述 …… 18
3.9 测试用例检索用例描述 …… 19
3.10 创建测试任务用例描述 …… 19
3.11 上传测试需求用例描述 …… 20
3.12 查看测试任务详情用例描述 …… 20
3.13 测试任务检索用例描述 …… 21
3.14 测试功能点描述匹配推荐用例描述 …… 21
3.15 测试任务匹配推荐用例描述 …… 22
3.16 测试用例内容复用用例描述 …… 22
3.17 标准测试用例示例 …… 28
3.18 User 表 …… 36
3.19 Dataset 表 …… 36
3.20 TestCase 表 …… 36
3.21 VectorData 表 …… 37
3.22 Task 表 …… 39
3.23 TriggerTask 表 …… 40
3.24 TaskResult 表 …… 40
3.25 DLModel 表 …… 42
3.26 OriginalText 表 …… 43

4.1 Dataset 类详情列表 46
4.2 TestCase 类详情列表 46
4.3 测试用例分析主要类列表 48
4.4 AnalysisTask 类详情列表 49
4.5 AnalysisResult 类详情列表 49

5.1 系统测试环境 60
5.2 下载数据集模板测试用例 61
5.3 上传测试用例集测试用例 61
5.4 查看测试用例集列表测试用例 62
5.5 查看测试用例详情测试用例 62
5.6 创建测试任务测试用例 63
5.7 查看测试任务测试用例 63
5.8 测试任务检索测试用例 64
5.9 性能测试接口列表 64
5.10 性能测试结果 66
5.11 文本匹配效果测试结果 68
5.12 用例复用效果 A 组测试结果 69
5.13 用例复用效果 B 组测试结果 69
5.14 测试用例设计评审得分列表 69

图 目 录

2.1 ES 节点架构图 ········· 9

3.1 测试用例分析模块用例图 ········· 12
3.2 测试任务管理模块用例图 ········· 14
3.3 测试用例匹配模块用例图 ········· 15
3.4 系统架构图 ········· 23
3.5 逻辑视图 ········· 25
3.6 开发视图 ········· 25
3.7 进程视图 ········· 26
3.8 物理视图 ········· 27
3.9 测试用例匹配算法流程图 ········· 30
3.10 ESIM 模型结构图 ········· 31
3.11 测试用例分析模块架构图 ········· 34
3.12 测试用例分析模块核心类图 ········· 34
3.13 测试用例分析模块 ER 图 ········· 35
3.14 测试任务管理模块架构图 ········· 37
3.15 测试任务管理模块核心类图 ········· 38
3.16 测试任务管理模块 ER 图 ········· 39
3.17 测试用例匹配模块架构图 ········· 40
3.18 测试用例匹配模块核心类图 ········· 41
3.19 测试用例匹配模块 ER 图 ········· 42

4.1 测试用例集模板下载流程图 ········· 44
4.2 测试用例集上传流程图 ········· 45
4.3 上传测试用例集截图 ········· 47
4.4 测试用例分析时序图 ········· 47
4.5 读取上传文件代码 ········· 50
4.6 关键词提取代码 ········· 51

4.7 测试用例向量化代码 …… 52
4.8 测试任务管理时序图 …… 53
4.9 创建测试任务截图 …… 53
4.10 测试任务管理列表截图 …… 54
4.11 查看测试任务详情截图 …… 54
4.12 创建测试任务代码 …… 55
4.13 测试用例匹配时序图 …… 56
4.14 ESIM 模型构建代码 …… 57
4.15 测试用例综合排序代码 …… 58

5.1 JMeter 线程组配置截图 …… 65
5.2 HTTP 请求配置截图 …… 65
5.3 可用性测试记录 …… 66

第一章　引言

1.1　项目背景和意义

随着软件行业的蓬勃发展，软件融入了我们日常的社交、办公、学习、娱乐等各个场景之中。质量和使用安全性对于软件是至关重要的特性，而这些通常需要软件测试来保障。软件测试是软件质量和可靠性的重要保障手段，是软件开发过程中不可或缺的环节。随着用户对软件质量的要求不断提高，软件测试在软件开发过程中的重要程度越来越突出。据统计，在部分软件开发项目中，软件测试在整个软件生命周期的开销占比可高达 50%[1]，更值得一提的是，对于某些高可靠性要求或生命攸关的软件，如银行交易系统、航空航天软件、军事武器控制软件等，其软件测试的开销远超其他开发阶段开销的总和 [2]。

测试用例的设计是软件测试的关键过程，设计良好的测试用例是开展软件测试的前提。设计测试用例的目的在于发现软件缺陷，因此这项工作较为依赖对软件的熟悉程度和测试工程师的经验，若测试人员有所变动或设计习惯不同，软件测试的维护将会变得更为复杂。因此，完全依靠人工设计测试用例的局限性日益凸显，难以满足软件测试的需要。在设计测试用例时，有几个因素对软件测试的质量和效率有显著影响。一是测试用例的规范性，测试用例的结构和内容形式满足测试标准和用例规范要求。二是测试用例的有效性，测试用例的内容能够识别软件缺陷，满足测试要求。三是测试用例设计的效率，设计工作尽可能在可控的测试周期内完成，避免影响软件的交付。这也是当前软件测试要解决的重要问题 [3]。

测试用例复用是使用累积的历史测试用例等资料来进行新测试的方法 [4]，复用的对象主要是历史测试用例，也包括累积的其他测试相关资料和经验。在面对相似的软件测试需求或软件迭代时，通过复用旧测试用例，完成新测试用例的设计，开展新的测试 [5]。

当前的测试用例复用技术主要有基于本体的描述方法、形式化建模的方法和基于检索的方法。基于本体的描述方法要求构建本体模型 [6]，将测试用例中的概念与本体中的概念、关系和实例作映射，基于领域的本体模型构建的全过程要求与领域专家密切配合。形式化建模的方法是通过 UML、Petri 网 [7] 等形式化语言对需求建模，判断需求相似性是基于模型结构来进行的，作为用例复用的衡量标准。基于检索的方法是通过字面检索的方式来进行测试用例的检索和复用，要求测试工程师有较强的业务抽象能力。

建模方法需要深入软件的功能细节，当软件系统较为庞大时，文档数量过多，形式化建模的复杂度过高。另外，基于字面检索的方法只考察了文本间的字面匹配和相似程度，对于文本的语义没有深入对比，容易检索出字面相似但语义差异较大的用例，而真正的语义相似的用例反而难以被推荐复用。

针对上述问题，本文提出了基于深度语义匹配的测试用例复用技术，开发了测试用例复用系统。系统提供测试用例数据模板，帮助用户上传规范化的测试用例数据集。在用户上传测试用例之后，通过 NLP 技术分析测试用例数据，存储测试用例向量化信息。最终，用户在设计新测试用例时，可以创建测试任务，输入测试功能点描述，系统将通过深度语义匹配模型，为每个测试功能点描述匹配历史测试用例，结合搜索引擎相似性得分等因素为用户推荐历史测试用例，用户可对其进行编辑、复制等操作，完成测试用例复用。综上所述，本系统通过挖掘历史测试用例的价值，帮助测试人员设计新测试用例，开展新的测试，为提高测试效率作出贡献。

1.2 国内外研究现状

1.2.1 测试用例复用研究现状

测试用例复用是使用历史版本软件和相似软件的测试用例、测试方法、测试执行文档等经验沉淀的产物，在软件迭代或开展新的软件测试时作为指导，帮助提高测试效率。目前测试用例复用往往局限在一个组织内部或项目内部，例如测试用例在回归测试时的复用，因为测试用例设计形式和描述较为统一，但缺点显而易见，即项目依赖较深，可移植性不强。

国内外已经有一些测试用例复用理论研究和技术探索。Alain Vouffo-Feudjio 和 Ina Schieferdecker 提出了基于 TTCN 的测试模式 [8]，通过测试模式的概念，阐述了测试模式的特征，并描述了测试模式在测试开发过程中的使用，通过数表组合来对测试用例建立模型，这种模式可支持在不同软件和相同软件不同版本之间实现测试用例复用；Forbes Gibb 等提出了基于 XML 的构件描述方法 [9]，通过构件来抽象化描述软件功能点，使用 XML 语言来规范描述构件，实现基于构件的测试用例复用方法；A. von Mayrhauser 等提出了基于领域模型的测试用例复用方法 [10]，这是一种基于软件重用、领域分析和领域建模的测试用例复用方法，在测试用例复用领域引入领域分析和领域方法概念，定义测试复用阶段，拆解复用过程，并提供了相应的工具。张娟等人 [11] 使用 Z 语言的形式，以测试功能点和测试项为基础，定义了复用度量公式，设计了质量评价模型。

也有研究人员从软件需求角度，研究基于需求的测试用例复用。赵鄂提出用形式化方法描述自然语言形式的软件需求，用形式化语言进行需求建模，通

过对比各个模型优缺点，最终选择 UML 方式来描述软件需求。根据软件需求设计对应的用例模型，特征模型与用例模型优缺点有所互补，能够较直观地表达领域需求。为解决自然语言形式软件需求的二义性，范斌等人 [12] 提出了需求分析方法：首先确定需求描述的关键词，在需求生成阶段严格规范化需求文档；然后以 SPARDL 建模语言为主，建立需求模型，解析前面所规范化的需求文档。这种基于模型的方法能够应用到测试用例复用领域。

有研究人员还提出了通过检索构件资产库的方式来进行测试用例复用。Frakes 等 [13] 基于构件描述，将构件的描述与检索方法分为三类：人工智能方法、超文本方法和信息科学方法。Mili[14] 基于检索复杂度，把构件检索方法分为基于文本的、基于规约的和基于词法描述子的编码和检索。信息科学方法在构件复用中较为成功，最常见的是属性、关键词和正文检索集中编目方法。

1.2.2 深度文本匹配研究现状

文本匹配是自然语言理解领域的一个重要问题，在智能问答、机器翻译、信息检索等领域应用非常广泛。

传统的文本匹配模型需要大量特征抽取和人工定义，针对特定任务设计，可学习参数少，泛化能力不强。Debasis Ganguly[15] 融合了文档模型和 LDA 模型，专注于使用词嵌入来提高检索效率，构建了一个广义语言模型，其中一对单词之间的相互独立不再成立，利用单词的向量嵌入来推导单词之间的转换概率。张伟男等 [16] 提出了一种基于主题推理的问题检索翻译模型，该模型通过利用主题信息，合理控制翻译噪声，进而优化问题检索结果。

近年来，自然语言处理成为深度学习的应用热点领域之一，文本匹配领域的模型逐渐以深度文本匹配模型为主。DSSM 模型 [17] 是文本匹配领域较早的深度学习模型，其结构是 Siamese 网络结构，对文档和查询的文本匹配度建模，通过计算向量的余弦相似度来衡量文本相似度，相比传统文本匹配模型效果显著提升。为解决基于卷积神经网络不能捕捉句子长距离的依存关系的问题，Palangi 等人 [18] 提出基于长短时记忆的文本匹配模型 LSTM-RNN。查询项和文档分别由长短时记忆的循环神经网络表示为一个向量，然后计算两个向量表达的余弦距离作为相似度的度量，输出最终的匹配值。该模型对于网页搜索的在线日志数据的效果较为显著。

1.3 本文主要工作

针对测试用例复用的研究和应用现状，以及测试人员在设计测试用例阶段面临的效率问题，本文提出了基于深度语义匹配的测试用例复用技术，开发了

测试用例复用系统，实现历史测试用例的上传、分析、管理和推荐复用。本文的工作主要为以下几个方面：

（1）提出了基于深度语义匹配的测试用例复用方法。本文引入文本匹配领域的深度语义匹配技术，利用测试用例文本内容中的语义关系，预测其语义匹配度，结合搜索引擎的字面检索能力和用户对测试用例的复用行为记录，设计了测试用例复用匹配度的计算流程，为历史测试用例的匹配提供了一定的客观标准。

（2）提供了历史测试用例的分析和管理功能。通过提供标准测试用例集模板，帮助用户重新组织历史测试用例的结构，使之兼容系统要求。使用自然语言处理技术对用户上传的测试用例进行分析，并连同分析结果保存至 Elasticsearch，便于用户检索。

（3）提供了历史测试用例的复用功能。通过创建测试任务的方式，获取测试功能点描述等关键输入，基于深度语义匹配，为用户推荐匹配度最高的历史测试用例，并提供复制、收藏等功能，提升复用行为的便捷性。

系统主体部分使用 Java 语言基于 Sring Boot 框架进行开发，前端部分使用 Javascript 语言基于 Vue 框架开发，测试用例分析服务与匹配服务使用 Python 语言基于 Flask 框架开发，对外提供基于 HTTP 的通信接口，并使用 Redis 作为缓存，数据库模块采用数据备份方式来保证系统的可靠性。

1.4 本文组织结构

本文共六个章节，组织结构如下：

第一章为引言，主要介绍了基于深度语义匹配的测试用例复用系统的项目背景和相关技术的国内外研究现状，说明了测试用例复用系统的研发目标和意义所在。

第二章为相关技术概述，主要介绍了开发过程中用到的相关技术，包括相关 NLP 技术，文本匹配技术和相关工程技术。

第三章为系统的需求分析与设计，通过对系统需求的分析，设计了系统的总体架构和各个模块的功能拆解，系统需求分为功能性需求和非功能性需求，架构设计主要描述了系统总体架构图和 4+1 视图，各模块的设计主要描述了模块的架构图，类图和数据库设计。

第四章为系统的实现，通过顺序图对各个模块的执行流程进行详细阐述，并对关键部分通过代码展示与解析的方式进行描述。

第五章为系统的测试与实验分析，介绍了系统的各项测试工作和为验证系统效果而进行的实验，测试包括功能测试、性能测试、可用性测试和效果测试。

第六章为总结与展望，对本文和系统的所做工作进行了总结，并指出了研发过程中存在的问题，提出了改进方向。

第二章　相关技术概述

2.1　NLP 技术分析

2.1.1　词向量模型

自然语言文本是非结构化的和不可计算的，在许多自然语言处理的任务中需要其转化为可计算的数据形式。词向量模型的思路是，先把文本分割为词序列，让词成为文本的最小单位。一个单词对应一个索引，也可以是对应位置为 1，其余位置为 0 的 one-hot 编码向量 [19]。这样的表示方法简单且有效，缺点是词与词之间没有相似性和语义关联，向量不表达语义信息。one-hot 编码向量有显而易见的缺陷，即在词汇量很多时，会制造维度灾难，模型的计算量复杂度剧增。

为解决这一问题，1986 年 Hinton[20] 提出了词的分布式表示，即词向量。词向量每个维度都是实数，而非大多位数为 0，是一个维度相对较低的稠密向量。随后 Bengio[21] 提出了一种 N-Gram 神经概率语言模型，在训练模型的过程中生成了词向量。此外，Mikolov 等人 [22] 提出了词向量生成模型 Skip-Gram 和 CBOW，Skip-Gram 模型把上下文当作输出，中心词当作输入，CBOW 模型恰好相反。相比之前的模型，Mikolov 剔除了前馈式神经网络的隐含层，大幅优化了模型的计算复杂度，并使得词向量的语义可解释性更强。Mikolov 等人还提出了子采样与负采样方法 [23]，其负采样方法可以用更快速的方式训练高频单词，获取更优的词向量，还证明了对高频词汇的子采样可以在训练过程中加快收敛速度，提高低频单词表示的准确度。

除了上述的模型以外，Collobert 等人 [24] 也提出了一种词向量生成模型。在这个模型当中输入是一个窗口大小为 N 的若干词汇，它们被映射矩阵转化成词向量，也使用前馈神经网络进行学习，使用语料中的窗口上下文为正例，替换掉中心词的上下文为负例，训练生成词向量。

2.1.2　Word2Vec

2013 年，Google 公司发布了词向量工具 Word2Vec[25]。Word2Vec 的核心原理是前文所提到的两个模型 Skip-Gram 和 CBOW。Skip-Gram 模型使用一个中心词作为输入，用于预测该词的上下文，CBOW 模型刚好相反，以中心词的上下文为输入来预测该词本身。

Word2Vec 的训练模型是只有一个隐含层的神经元网络，其输入是词汇对应的 one-hot 编码向量，输出也是 one-hot 编码向量。通过语料中采集的样本来训练这个神经元网络，并且采用层次 softmax 和负采样方法来加速训练，层次 softmax[26] 使用哈夫曼树而非二分类方法，负采样方法 [27] 忽略了出现概率低的词语，以此达到减小词袋规模的目的，只需预测总体类别的一个子集。模型收敛之后，从输入层到隐含层的权重参数，即为每一个词的词向量表示。

Word2Vec 自问世之后被广泛使用，因为自然语言处理任务中词汇级别的处理非常常见，此外，它也可以用作句子和文档级别的自然语言处理任务，作为问题的语料预处理方式。工业界中，机器翻译、相似商品推荐和社交网络推荐等领域也常常见到以 Word2Vec 为基础的应用。本文在分析测试用例时需要对测试用例文本内容进行文本向量化等预处理，因此选用该工具。

2.2 文本匹配

文本匹配是自然语言理解中的一个核心问题，在机器翻译、问答系统和信息检索等场景中，它们的核心任务都可以通过某些方式抽象成文本匹配问题。在机器翻译中，可以理解为两种语言之间的文本匹配任务；在问答系统中，核心任务可以抽象成问题和答案候选集之间的匹配；在信息检索时，核心任务是查询和文档之间的文本匹配任务。

对于文本匹配中大多数的实际场景，可以将其抽象成一个排序问题。给定一段文本 T 和一个文本列表 $\{t_1, t_2, \ldots, t_n\}$，需要从该文本列表中选出与给定文本 T 相匹配的文本。文本匹配模型通常会计算列表中每段文本的匹配程度 s_i，s_i 排序越靠前的文本其匹配程度越高。

评价一个排序结果好坏的评价指标主要有：$P@k$，$R@k$，MAP 和 MRR。设定前 k 个真实排序文本中，匹配文本有 M_k 个，而前 k 个预测排序文本中，所匹配文本是 N_k 个，则评价指数 $P@k$(Precision at k) 和 $R@k$(Recall at k) 的计算公式定义如下：

$$P@k = \frac{N_k}{k},\ R@k = \frac{N_k}{M_k}.$$

在预测排序中，假设真实匹配文本的排序位置分别为 $\{l_1, l_2, \ldots, l_n\}$，其中 n 为列表中所有匹配文本的数量，则 MAP(Mean Average Precision) 的计算公式定义如下：

$$MAP = \frac{\sum_{i=1}^{n} P@k_i}{n}$$

同样的，若考察排序最前面的 k_1 个真实匹配文本，就可以定义 MRR(Mean Reciprocal Rank) 的计算公式：

$$MRR = P@k_1$$

传统的文本匹配模型和方法存在着一些问题。首先是需要人工进行特征选择，耗时费力。其次，文本的语义相似程度难以通过基于主题模型的隐空间模型来建模表达。这些都是传统文本匹配方法的瓶颈所在。

2.3 深度语义匹配

深度学习的特征表示学习能力十分优异，研究人员针对文本匹配任务提出了许多深度文本匹配模型。这些深度学习模型可以分为以下 3 类：基于单语义文档表达的、基于多语义文档表达的和直接建模匹配模式的。

DSSM(Deep Structured Semantic Model)[17] 是较早的深度文本匹配模型之一，采用 Siamese 网络结构。DSSM 在文本向量化时，先划分文本的最小单位，可以是单词或若干字母的片段，将其映射成一个向量，然后通过 3 层的全连接层表达句子主题向量，进行正负样本的训练。基于单语义文档表达的深度学习模型有一定的优势，其匹配速度较快，文本映射的结果便于保存，还可以用无监督数据来预训练。但是这类方法也有其不足，例如只关注了与匹配有用的部分信息。

与单语义文档表达不同，基于多语义文档表达的方法综合考虑了文本的局部性表达和全局性表达。Socher 等人 [28] 提出了可伸缩递归自动编码器 uRAE。该方法首先对两个待匹配文本构建句法树，然后使用无监督数据进行预训练，获得一个编码器，然后就可以通过它对词、句子和文本进行向量化。多语义文档表达的特点在于考虑了不同粒度的文本，这是它相对单语义文档表达方法的优势。

直接建模匹配模式的思路是直接捕捉文本匹配的各个特征。DeepMatch[29] 就是基于直接建模匹配的模型，它分为局部匹配层和综合层，输入的文本对经过局部匹配层后产生局部匹配的结果，综合层是一个可以综合多个局部匹配结果的神经网络。2016 年的 MatchPyramid 模型 [30] 则提出用匹配矩阵的概念描述两段文本的交互方式，基于这种方式并进行二维卷积，提取得到匹配空间的模式，经过全连接网络计算文本之间的相似度。

由于深度语义匹配模型在文本匹配领域的优异表现，本文将深度语义匹配技术与测试用例复用相结合，通过测试功能点等关键字段的文本匹配来为用户提供可复用的测试用例。

2.4 相关工程技术

2.4.1 Elasticsearch

Elasticsearch 是一个底层基于 Lucene 实现的分布式搜索引擎 [31]，具有分布式特性，通过 RESTful API 对外提供服务。Elasticsearch 以其易用性而著名，屏蔽了 Lucene 的底层细节，在日志数据分析、网站搜索等方面有广泛的使用场景。

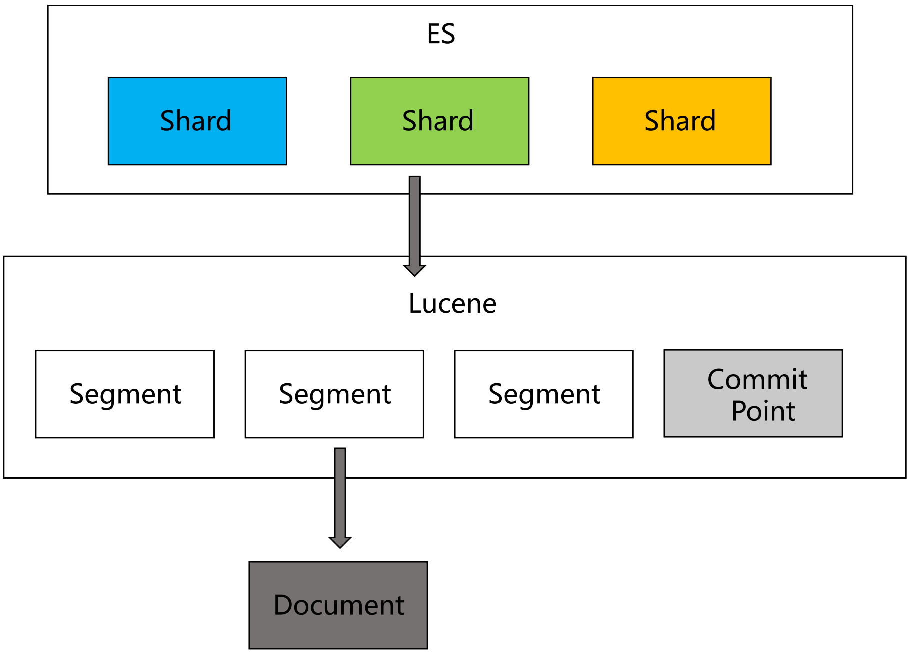

图 2.1: ES 节点架构图

Elasticsearch 可以为不同的数据类型提供高效的搜索服务和分析服务，包括结构化和非结构化的文本和数据。Elasticsearch 高效地实现了存储和索引数据，然后提供对这些数据的快速检索服务。Elasticsearch 还有极佳的可伸缩性，随着数据量的增长，其分布式特点为系统的部署和扩容提供了可能性。

Elasticsearch 的节点架构如图2.1所示。每一个 Shard 可以看作一个 Lucene Index，其中有一个 Commit Point 文件和多个 Segment 文件。集群运行中在 ES 配置好索引后无法动态调整分片配置。因此，只能新建索引对数据进行重新索引来调整分片数量，该操作比较耗费性能。

Elasticsearch 写入数据的过程如下。客户端选择一个节点发送请求，称之为协调节点。协调节点对文档进行路由，将请求转发给文档所在的对应节点。对应节点上的主分片处理请求，然后将数据同步到从节点。最后，协调节点如果发现主节点和所有的从节点都操作成功后，就会把返回消息发送给客户端。

Elasticsearch 在工业界应用已较为广泛。例如，软件项目中可用来存储和分析日志等监控数据；作为存储引擎，可以使用 Elasticsearch 来自动化业务工作流；还有人使用 Elasticsearch 来管理和分析地理空间数据、生物基因数据等。因为处理数据的速度和灵活性方面的优势，本系统使用 Elasticsearch 来存储测试用例文本向量化后的数据，也用它来执行测试用例匹阶段的向量检索等操作。

2.4.2 Redis

Redis 是一种基于内存键值的非关系型数据库 [32]，常被用作分布式的高速缓存。与 Mysql 等数据库相比，Redis 的优势是存取数据非常高效快速，因为其数据读写是在内存中进行的。相比较其他的内存键值存储系统，Redis 支持的数据结构更加丰富多样，功能更加完善，使用体验更佳。Redis 支持周期性持久化数据，启动后加载持久化数据进行数据恢复，降低了因系统故障导致数据丢失的概率。Redis 还支持多种不同语言的客户端，采用的架构为 CS 架构。

当使用 Redis 查询数据时，系统首先会去 Redis 缓存中根据键值查询，如果返回值为空，说明数据不存在，再转向数据库中查询数据，查询到的结果会放进 Redis 缓存里，这样做的原因是便于下次查询可以直接从缓存里获取数据。如果一个数据属于不经常变动的热点数据，或者需要进行大量计算，那么很适合使用 Redis 缓存来进行存储，这样能帮助加快系统的查询速度，减轻系统的计算压力和数据库的查询压力。本系统使用 Redis 来进行测试用例匹配结果数据的缓存，这样便于实现匹配结果的“一次计算，多次查询”，减轻系统的计算压力和数据库访问压力。

2.5 本章小结

本章介绍了本系统使用到的相关理论与技术。首先是测试用例分析所使用的相关自然语言处理技术，包括词向量技术和 Word2Vec 工具。然后对文本匹配的发展和相关理论进行了阐述，由此引出了该领域较为前沿的深度语义匹配技术。最后对用到的工程技术进行了介绍，包括搜索引擎工具 Elasticsearch 和非关系型数据库 Redis。

第三章　基于深度语义匹配的测试用例复用系统的需求分析与设计

3.1　系统整体概述

本系统在为软件测试人员在设计测试用例阶段提供测试用例复用的功能，并且提供管理历史测试用例数据集的功能。目的是帮助测试人员管理并利用好历史测试用例数据，便于测试人员检索历史测试用例，推荐可复用的测试用例以减少编写新测试用例的时间，缩短测试周期。

在软件开发工作中，企业或项目组会累积相当数量的测试用例。在面对新测试需求的时候，软件测试人员在设计新测试用例时可使用本系统来辅助设计，通过系统推荐可复用的历史测试用例，挑选测试用例进行复制、修改，完成新测试用例的设计工作。本章针对基于深度语义匹配的测试用例复用技术进行需求分析，对系统所使用的前后端工程架构，数据库存储，深度语义匹配算法等关键部分进行概要设计，对各个功能模块进行详细设计。

首先，系统对已有的历史测试用例需要进行预处理，提取测试功能点描述等文本信息，通过词向量模型对其向量化，使用 Elasticsearch 存储向量数据并构建检索索引。使用时，系统使用人员输入测试功能点描述并创建测试任务，系统使用自然语言处理技术对测试功能点描述的原始文本输入进行处理，主要的处理有分词、词性标注和实体识别。然后先使用 Elasticsearch 的检索功能对测试用例进行检索，得到候选数据集，再对所得候选数据集进行深度语义匹配计算，结合 ES 相关性分值和用例复用记录，最终得到推荐的可复用测试用例。

3.2　系统需求分析

3.2.1　测试用例分析模块需求分析

如图3.1所示是测试用例分析模块用例图。测试用例分析模块的主要功能如下。

用户下载测试用例数据集模板。测试用例数据集模板规定了系统可识别的测试用例的格式规范，包括字段定义、字段中英文名称、字段描述和备注信息等。

用户上传测试用例数据集。挖掘历史测试用例的价值是系统的核心目标之

一，用户将其按模板的规范要求组织成测试用例集文件后，可以将其上传至系统中，上传步骤也是分析历史测试用例的前提。

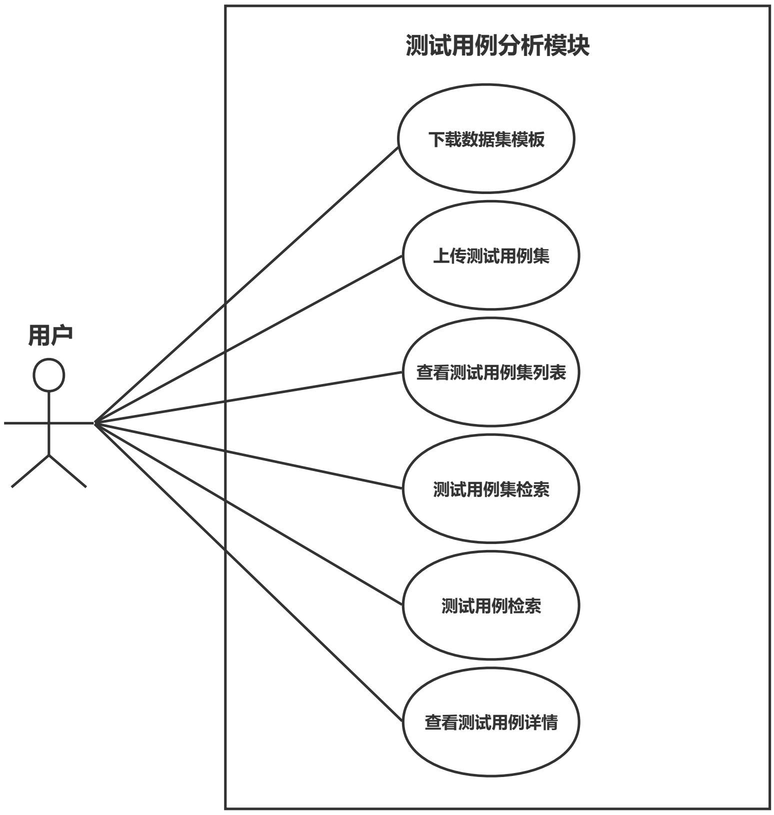

图 3.1: 测试用例分析模块用例图

用户查看测试用例集列表。系统提供了测试用例集管理功能，用户在上传测试用例集之后，可以在测试用例集列表页面看到所上传的所有测试用例集。

用户检索测试用例集。当测试用例集过多，会导致测试用例集列表长度过长。用户可以在搜索框输入关键词直接搜索测试用例集，可以根据创建时间、测试用例数量等条件对测试用例集列表进行排序，便于查看所要查找的测试用例集。

用户检索测试用例。在测试用例列表页面，用户可以浏览一个测试用例集中的各条测试用例的简略信息，包括测试用例名称、测试用例类型和创建时间等。通常一个测试用例集中会有多条测试用例，列表形式不便查找特定的某个测试用例。用户可以在搜索框输入关键词来搜索特定的测试用例，也可以根据条件对其进行排序。

用户查看测试用例详情。测试用例列表页面只展示测试用例的简略信息。用

户可以点击任意一个测试用例，进入测试用例详情页面，查看该测试用例的全部信息。

表 3.1: 测试用例分析模块功能性需求列表

需求 ID	需求名称	需求描述
R1	下载数据集模板文件	用户能够下载系统提供的测试用例文档标准模板
R2	上传测试用例集	用户能够选择本地文件，上传基于标准模板的历史测试用例数据集并存储在系统中
R3	查看测试用例集列表	用户可以查看自己上传的测试用例集列表
R4	测试用例集检索	用户可以在测试用例集列表页面的搜索框搜索自己上传的测试用例集
R5	查看测试用例详情	用户可以查看测试用例集中的一条历史测试用例的详细信息
R6	测试用例检索	用户可以在测试用例列表页面的搜索框搜索其中的测试用例

根据上文的分析，可得测试用例分析模块的功能性需求列表，如表3.1所示。

3.2.2 测试任务管理模块需求分析

如图3.2所示是测试任务管理模块用例图。测试任务管理模块的主要功能如下。

用户创建测试任务。面对新的测试任务，测试人员在设计测试用例阶段可以使用本系统作为辅助设计工具。用户首先应创建测试任务，填写测试任务名称等信息，选择测试类型，通过文本框输入或文件上传的形式输入测试需求。

上传测试需求。新的测试需求是新测试阶段的测试用例编写工作所围绕的核心。用户可以通过文本框输入和上传本地测试需求文件两种方式来上传测试需求。

查看测试任务详情。在测试任务列表页面，用户可以浏览自己所创建的测试任务，但是列表只展示了简略信息。用户可以点击任意一个测试任务，进入测试任务详情页查看该任务的所有信息。

测试任务检索。一个用户可以创建多个测试任务，当任务较多时，查询某一特定测试任务会变得较为困难。用户可以在测试任务列表页面的搜索框中输入关键词来搜索某一测试任务，也可以通过测试类型、创建时间等条件来排序筛选。

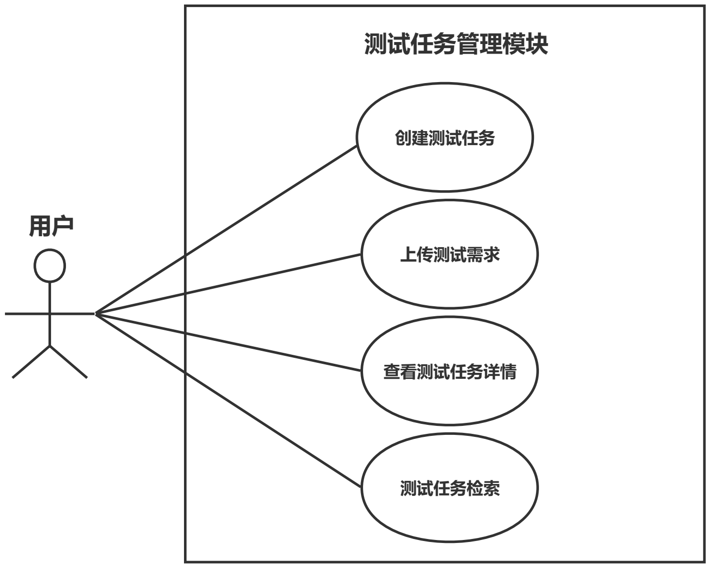

图 3.2: 测试任务管理模块用例图

根据上文的分析，可得测试任务管理模块的功能性需求列表，如表3.2所示。

表 3.2: 测试任务管理模块功能性需求列表

需求 ID	需求名称	需求描述
R7	创建测试任务	用户可以填写任务信息和测试需求，创建一个测试任务
R8	上传测试需求	用户能够在创建测试任务页面的文本框输入测试需求，或者选择本地测试需求文件来上传测试需求
R9	查看测试任务详情	用户可点击进入自己创建的测试任务，可以查看系统为每一条测试功能点描述推荐的历史测试用例并复制到剪贴板
R10	测试任务检索	用户可以根据条件对自己的所有历史测试任务进行筛选排序，完成测试任务的检索

3.2.3 测试用例匹配模块需求分析

如图3.3所示是测试用例匹配模块用例图。测试用例匹配模块的主要功能如下。

测试功能点描述匹配推荐。一个测试任务中可以包含多条测试功能点描述，用户可以单独查看某一条测试功能点描述的测试用例匹配结果。

测试任务匹配推荐。测试任务完成测试用例匹配后，用户可以查看整个任务的历史测试用例匹配结果及其统计信息。

测试用例内容复用。系统为测试任务完成历史测试用例匹配后，用户可以查看系统的推荐历史测试用例，还可以收藏某个历史测试用例，复制其内容至剪贴板来进行复用。

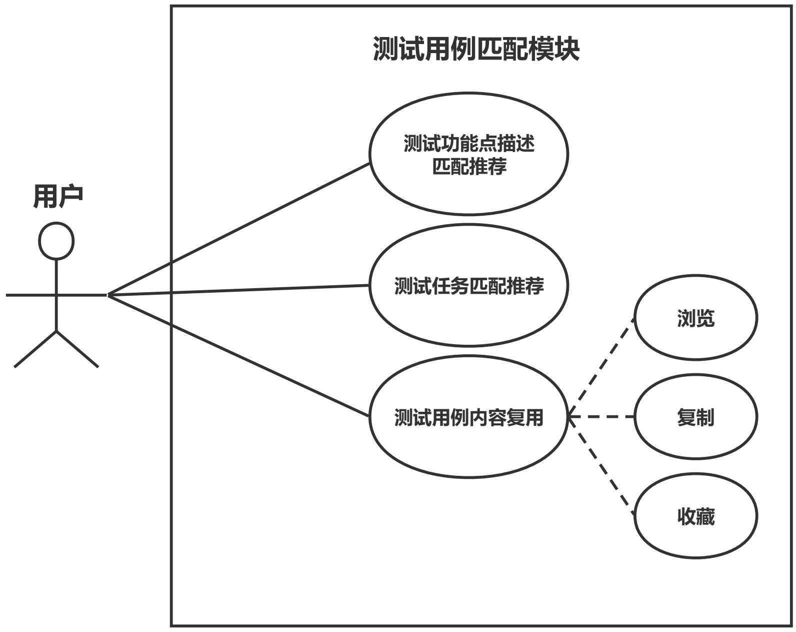

图 3.3: 测试用例匹配模块用例图

根据上文的分析，可得测试用例匹配模块的功能性需求列表，如表3.3所示。

表 3.3: 测试用例匹配模块功能性需求列表

需求 ID	需求名称	需求描述
R11	测试功能点描述匹配推荐	用户可以查看任意一条测试功能点描述的匹配推荐结果
R12	测试任务匹配推荐	用户用户可以查看一个测试任务的所有测试功能点描述的匹配推荐结果
R13	测试用例内容复用	用户可以浏览、收藏系统推荐的历史测试用例，并将其内容复制到剪贴板

3.2.4 非功能性需求分析

测试用例复用系统的主要目标是帮助测试人员针对新测试需求来设计新测试用例，通过测试人员的使用反馈和工作效率变化来评估系统的效果。在满足基本使用的前提下，还有可用性，易用性，可维护性和可扩展性的非功能性需求。

可用性：系统应有较好的可用性，作为工具型应用，在 99% 的时间内保持正常运行，对数据进行备份，在遇到网络波动、进程崩溃等问题时能及时重启和数据恢复。

易用性：系统在使用体验方面应该做到用户体验良好，提供设计合理与美感兼具的人机交互界面，在界面切换时给出提示，在用户等待时提供过渡动画。系统应该易于学习和使用，界面上提供合理的文字说明，复用测试用例的操作流程做到简洁明了。

可维护性：系统应有较好的可维护性。出现故障后，能较为快速地进行故障定位和修复故障，提供完整且详细的日志记录，关键代码部分应有清晰的注释。

可扩展性：系统应有良好的设计风格，功能模块之间相互解耦，各个功能设计具备高层次抽象设计，便于后续的模块功能的升级和扩展，例如深度语义匹配算法的升级。

3.2.5 系统用例描述

根据前文的功能性需求分析，可知系统用例包括：下载数据集模板、上传测试用例集、查看测试用例集列表、测试用例集检索、测试用例检索、查看测试用例详情、创建测试任务、上传测试需求、查看测试任务详情、测试任务检索、测试功能点描述匹配推荐、测试任务匹配推荐和测试用例任务复用，共 13 个系统用例。

表 3.4: 下载数据集模板用例描述

ID	**UC1**
名称	下载数据集模板
参与者	用户
用例描述	用户在系统中下载数据集模板文件到本地
前置条件	用户登录并进入系统的数据集模板下载页面
后置条件	无
正常流程	1. 用户点击左侧边栏的上传测试用例集按钮，进入上传测试用例集页面 2. 用户点击下载数据集模板文件蓝色链接 3. 用户选择文件保存路径并下载文件至本地
异常流程	3a. 下载过程中网络出现异常导致下载失败，系统提示“网络异常，请重新下载”

下面将对各个系统用例进行详细描述。

下载数据集模板是规范待上传数据格式的第一步。系统希望通过提供数据模板的方式来对用户上传的测试用例数据的格式作出约束，便于系统提取测试用例关键信息。其用例描述如表3.4所示。

上传测试用例集是复用历史测试用例的前提。用户将积累的丰富数量的测试用例集按数据模板的格式组织起来，将其上传到系统中，形成专属的历史测试用例库。其用例描述如表3.5所示。

表 3.5: 上传测试用例集用例描述

ID	**UC2**
名称	上传测试用例集
参与者	用户
用例描述	用户选择本地测试用例集文件上传到系统
前置条件	用户登录并进入系统的上传测试用例集页面
后置条件	用户可以检索和查看所上传的测试用例集
正常流程	1. 用户点击左侧边栏的上传测试用例集按钮，进入上传测试用例集页面 2. 用户填写测试用例集名称 3. 用户点击文件选择按钮，选择本地测试用例集文件并确认上传 4. 系统提示数据集上传成功
异常流程	2a. 文件名不符合要求，系统提示“文件名不符合要求，请重新命名” 2b. 文件名重复，系统提示“文件名重复，请重新命名” 3a. 上传过程中网络出现异常导致上传失败，系统提示“网络异常，请重新上传” 3b. 文件格式不对，系统提示“文件格式不符合要求，请重新上传”

上传测试用例集后，用户可以查看自己上传的测试用例集列表，，并按条件排序来检索。测试用例集列表展示测试用例集的简略信息，包括测试用例集名称、测试用例数量和创建时间。其中测试用例数量和测试创建时间可用于排序，测试用例名称可用于检索。其用例描述如表3.6所示

表 3.6: 查看测试用例集列表用例描述

ID	**UC3**
名称	查看测试用例集列表
参与者	用户
用例描述	用户在系统中查看自己上传的测试用例集列表
前置条件	用户登录并进入系统的管理测试用例集页面
后置条件	无
正常流程	1. 用户点击左侧边栏的管理测试用例集按钮，进入管理测试用例集页面 2. 用户点击名称按钮或创建时间按钮对测试用例集排序 3. 用户点击翻页按钮进行翻页，浏览测试用例集列表
异常流程	无

为快速查找某一测试用例集，系统在测试用例集列表提供搜索框，用户可

以输入关键词进行检索，也可以按条件排序筛选。其用例描述如表3.7所示。

表 3.7: 测试用例集检索用例描述

ID	**UC4**
名称	测试用例集检索
参与者	用户
用例描述	用户可以在测试用例集列表页面的搜索框搜索自己上传的测试用例集
前置条件	用户登录并进入系统的管理测试用例集页面
后置条件	无
正常流程	1. 用户点击左侧边栏的管理测试用例集按钮，进入管理测试用例集页面 2. 用户在搜索框输入关键词以搜索测试用例集 3. 用户点击翻页按钮进行翻页，直至找到所需的测试用例集
异常流程	2a. 搜索结果为空，页面无测试用例集可展示 3a. 测试用例集信息不足一页，无需翻页

为更好地管理和复用历史测试用例，在每个测试用例集的测试用例列表页面，用户可以查看上传的测试用例的详细信息。其用例描述如表3.8所示。

表 3.8: 查看测试用例详情用例描述

ID	**UC5**
名称	查看测试用例详情列表
参与者	用户
用例描述	用户可以在系统中查看自己上传的测试用例集中的每条测试用例的详情
前置条件	用户登录并进入系统的管理测试用例集页面
后置条件	用户可以点击关闭按钮返回测试用例集页面
正常流程	1. 用户点击左侧边栏的管理测试用例集按钮，进入管理测试用例集页面 2. 用户点击进入一个测试用例集 3. 用户点击详情按钮，查看一条测试用例的详细信息
异常流程	1a. 用户未上传过测试用例集，页面为空 2a. 未成功读取测试用例集信息，系统提示“请刷新页面” 3a. 加载测试用例信息失败，系统提示“加载测试用例信息失败”

系统在测试用例列表提供搜索框，用户可以输入关键词检索特定的测试用例，也可以按条件排序筛选测试用例。其用例描述如表3.9所示。

表 3.9: 测试用例检索用例描述

ID	**UC6**
名称	测试用例检索
参与者	用户
用例描述	用户可以在测试用例列表页面的搜索框搜索其中的测试用例
前置条件	用户登录并进入测试用例列表页面
后置条件	无
正常流程	1. 用户点击左侧边栏的管理测试用例集按钮，进入管理测试用例集页面 2. 用户点击查看详情进入测试用例集列表页面 3. 用户点击某个测试用例集，查看其测试用例 4. 用户在搜索框输入关键词以搜索测试用例 5. 用户点击翻页按钮，直至找到所查找的测试用例
异常流程	无

为复用历史测试用例，用户需要创建一个测试任务。一个测试任务中可包含多条测试功能点描述，即对应多个不同的测试需求。系统将为每条测试功能点描述推荐可复用的历史测试用例，以树状的形式展示在页面上。其用例描述如表3.10所示。

表 3.10: 创建测试任务用例描述

ID	**UC7**
名称	创建测试任务
参与者	用户
用例描述	用户输入测试功能点描述等信息来创建一个测试任务
前置条件	用户登录并进入系统的创建测试任务页面
后置条件	用户可以查看创建的测试任务
正常流程	1. 用户点击左侧边栏按钮，进入创建测试任务页面 2. 用户填写测试任务必要信息，包括任务名称，测试类型和任务描述 3. 用户填写多条测试功能点描述 4. 用户点击创建任务按钮，完成任务创建 5. 系统提示任务创建成功
异常流程	2a. 测试任务必要信息填写不完整，系统提示“请完整填写任务信息” 2b. 测试任务名称不符合要求，系统提示“测试任务名称不符合要求，请重新命名” 3a. 用户忘记填写测试功能点描述，系统提示“请至少填写一条测试功能点描述”

创建测试任务时，系统支持两种测试需求的上传方式，可以通过文本框输入，也可以上传测试需求文件由系统来解析。其用例描述如表3.11所示。

表 3.11: 上传测试需求用例描述

ID	UC8
名称	上传测试需求
参与者	用户
用例描述	用户能够在创建测试任务页面的文本框输入测试需求，或者选择本地测试需求文件来上传测试需求
前置条件	用户登录并进入系统的创建测试任务页面
后置条件	无
正常流程	1. 用户点击左侧边栏按钮，进入创建测试任务页面 2. 用户填写测试功能点描述，可以为空 3. 用户选择本地测试需求文件并上传
异常流程	3a. 文件格式不对，系统提示“文件格式不正确” 3b. 测试需求文件上传失败，系统提示“请重新上传”

测试任务创建成功后，用户可查看测试任务，查看系统为每条测试功能点描述推荐的历史测试用例，并可选中合适的测试用例来导出，也可将其复制到剪贴板。其用例描述如表3.12所示。

表 3.12: 查看测试任务详情用例描述

ID	UC9
名称	查看测试任务
参与者	用户
用例描述	用户查看所创建的测试任务详情
前置条件	用户登录并进入系统的管理测试任务页面
后置条件	无
正常流程	1. 用户点击左侧边栏按钮，进入查看测试任务页面 2. 用户点击一个测试任务，进入任务详情页面 3. 用户点击一条测试功能点描述，查看系统为其推荐的历史测试用例 4. 用户选择一条测试用例，点击复制按钮，将其复制到粘贴板
异常流程	3a. 系统无可推荐的历史测试用例，提示“暂无可推荐的测试用例”

本系统基于测试任务的形式来为用户提供测试用例复用服务，测试任务创建后，其信息将在系统中持久化保存。对于已创建的测试任务，用户可以根据名称，测试类型和创建时间等条件来进行排序和检索，这样便于用户对历史测试任务的管理和查询。其用例描述如表3.13所示。

表 3.13: 测试任务检索用例描述

ID	UC10
名称	测试任务检索
参与者	用户
用例描述	用户按条件筛选和检索所创建的测试任务
前置条件	用户登录并进入系统的管理测试任务页面
后置条件	用户可以点击查看任意一个测试任务的详情
正常流程	1. 用户点击左侧边栏按钮，进入查看测试任务页面 2. 用户输入关键词对测试任务进行搜索 3. 用户点击创建时间排序箭头对测试任务进行排序 4. 用户点击翻页按钮进行翻页 5. 用户找到所要查询的测试任务，点击进入查看任务详情
异常流程	2a. 关键词搜索结果为空，不展示测试任务信息 4a. 测试任务信息不足一页，无需翻页 5a. 加载测试任务信息失败，系统提示“加载测试任务信息失败”

一个测试任务包含多条测试功能点描述，用户可以单独查看系统为每一条测试功能点描述推荐的历史测试用例。其用例描述如表3.14所示。

表 3.14: 测试功能点描述匹配推荐用例描述

ID	UC11
名称	测试功能点描述匹配推荐
参与者	用户
用例描述	用户可以查看任意一条测试功能点描述的匹配推荐结果
前置条件	用户登录并进入一个测试任务的详情页面
后置条件	无
正常流程	1. 用户点击边栏的测试任务管理按钮，进入测试任务管理页面 2. 用户点击进入一个测试任务的详情页面 3. 用户点击测试任务中的一条测试功能点描述 4. 用户选择一条推荐的历史测试用例并点开查看
异常流程	无

创建测试任务后，用户可以查看系统该任务所有测试功能点描述所推荐的历史测试用例，还可以查看系统对该任务的匹配结果统计情况。其用例描述如表3.15所示。

表 3.15: 测试任务匹配推荐用例描述

ID	UC12
名称	测试任务匹配推荐
参与者	用户
用例描述	用户可以查看一个测试任务的所有测试功能点描述的匹配推荐结果
前置条件	用户登录并进入一个测试任务的详情页面
后置条件	无
正常流程	1. 用户点击边栏的测试任务管理按钮，进入测试任务管理页面 2. 用户点击进入一个测试任务的详情页面 3. 用户点击查看结果按钮 4. 用户选择一条推荐的历史测试用例并点开查看
异常流程	无

用户查看系统为测试功能点描述所推荐的历史测试用例时，可以收藏自己所满意的测试用例，还可以直接点击复制，将其内容复制至剪贴板以复用测试用例。其用例描述如表3.16所示。

表 3.16: 测试用例内容复用用例描述

ID	UC13
名称	测试用例内容复用
参与者	用户
用例描述	用户可以浏览、收藏系统推荐的历史测试用例，并将其内容复制到剪贴板
前置条件	用户登录并进入一个测试任务的详情页面
后置条件	无
正常流程	1. 用户点击边栏的测试任务管理按钮，进入测试任务管理页面 2. 用户点击进入一个测试任务的详情页面 3. 用户点击查看结果按钮 4. 用户选择一条推荐的历史测试用例并点开查看 5. 用户点击收藏该历史测试用例 6. 用户点击复制按钮，将其内容复制至剪贴板
异常流程	无

3.3 系统概要设计

3.3.1 系统架构设计

测试用例复用系统的系统架构图如图3.4所示。系统采用分层结构，包括前端展示层，后端服务层，应用框架层和数据存储层，这样的层次结构有利于前后端分离式开发，实现前后端解耦。下面将对各个层次进行具体介绍。

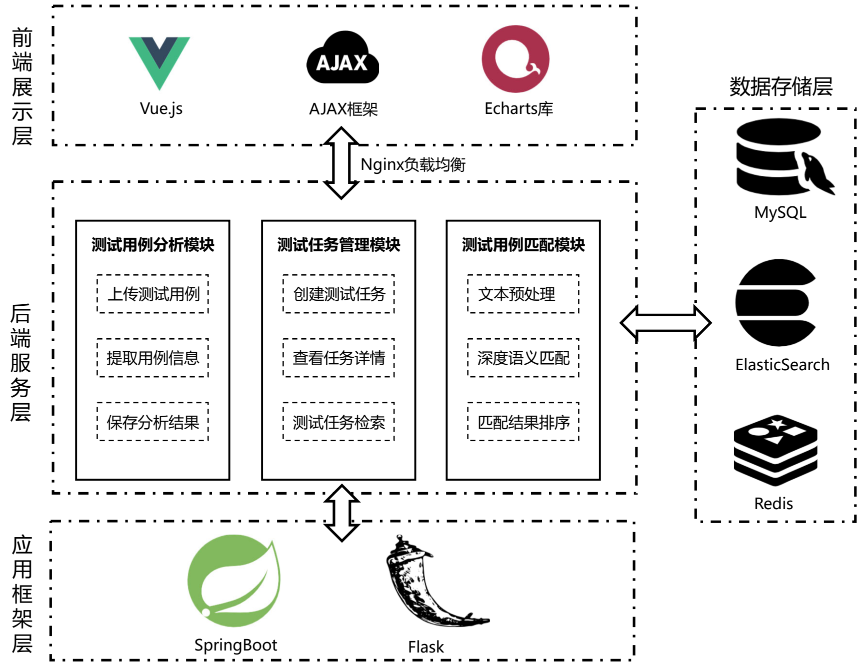

图 3.4: 系统架构图

前端展示层，选择 Vue 框架作为前端开发的基础框架。Vue 是一套轻量级的渐进式用户界面构建框架，且 Vue.js 生态已经非常完善。系统使用 Vue-cli 脚手架进行原型的敏捷开发，并实现项目的配置部署，使用 Vue-router 实现系统页面路由选择。系统的 UI 组件框架选择 ElementUI，风格简洁美观，功能强大。系统使用 AJAX 异步通信框架实现前后端通信。为实现统计数据的直观展示和良好的用户体验，系统使用 Echarts 库实现数据的可视化渲染。为保证系统的可用性和稳定性，系统使用 Nginx 实现系统的负载均衡。

后端服务层，划分为三个模块：测试用例分析模块、测试任务管理模块和测试用例匹配模块。该层负责实现系统的核心功能。用户上传历史测试用例数据集之后，系统提供测试用例分析功能，可实现测试用例关键信息提取和文本分析，并将分析结果持久化保存。对于测试任务，系统提供测试任务管理的功能，以 Restful 接口的方式提供服务，实现对测试任务相关表单的查询访问。对于测试用例匹配，系统通过 python 脚本形式提供文本预处理功能，并调用训练好的深度语义匹配模型进行测试用例的匹配，还提供排序方法将匹配结果排序后输出，最后返回结果到前端页面。

应用框架层，系统使用 Spring Boot 框架作为后端服务的实现框架，简化了搭建和开发过程，也省去了定义样板化配置的步骤。系统使用基于 Python 语言的 Flask 框架搭建服务，提供文本处理和语义匹配相关功能。

数据存储层提供各类型数据的存取服务。用户上传的测试用例集的信息，所创建的测试任务将存储到 MySQL 数据库中，测试用例分析结果将存入 ElasticSearch，针对新测试任务的推荐测试用例结果将存入 Redis。

3.3.2 系统模块划分

本系统可划分为三个模块：测试用例分析模块、测试任务管理模块和测试用例匹配模块。用户上传的测试用例集将被测试用例分析模块进行处理，测试任务管理模块将为用户创建的测试任务服务，测试用例匹配模块针对测试任务中的测试功能点描述进行历史测试用例的推荐匹配。下面将对各个模块展开详细介绍。

测试用例分析模块：用户上传测试用例集，系统将测试用例数据保存至 MySQL 数据库，并执行测试用例分析方法，将文本分词结果和向量数据等信息保存至 ElasticSearch，用于后续的检索，数据召回和向量比对。

测试任务管理模块：用户创建测试任务，填写任务信息，输入测试功能点描述，然后换可在测试任务管理页面查看自己创建的任务详情。若测试任务过多，可通过名称和创建时间等条件来排序和检索特定的测试任务。

测试用例匹配模块：系统通过开源语料数据训练词向量模型和深度语义匹配模型，对于用户输入的测试功能点描述文本信息进行分词等预处理工作，去除停用词后使用 ElasticSearch 检索功能召回部分测试用例数据作为候选数据，然后使用深度语义匹配模型进行结果预测，匹配成功的测试用例结合多项指标排序后输出。

3.3.3 系统视图模型

本节将从逻辑视图、进程视图、开发视图、物理视图和场景视图五种视角来详细介绍系统的总体架构设计。其中场景视图即为系统用例图，在上文中以划分模块用例图的形式呈现，这里不再赘述。

系统的逻辑视图如图3.5所示。逻辑视图从用户的视角对系统职责逐级划分，描述系统所提供的功能。其中与测试用例直接相关的接口有三个，CaseUploadService 负责测试用例集的上传，TemplateDownloadService 负责测试用例模板的下载，DBService 负责测试用例在数据库的存取过程。

TestTaskManageService 提供测试任务管理服务，其中 TaskCreateService 提供测试任务创建相关的功能，TaskDetailService 提供测试任务查看的功能，TaskSearchService 提供测试任务检索的功能。和匹配历史测试用例相关的接口有 3 个，其中 TextProcessService 提供文本预处理的功能，SemanticMatchService 提供深度语

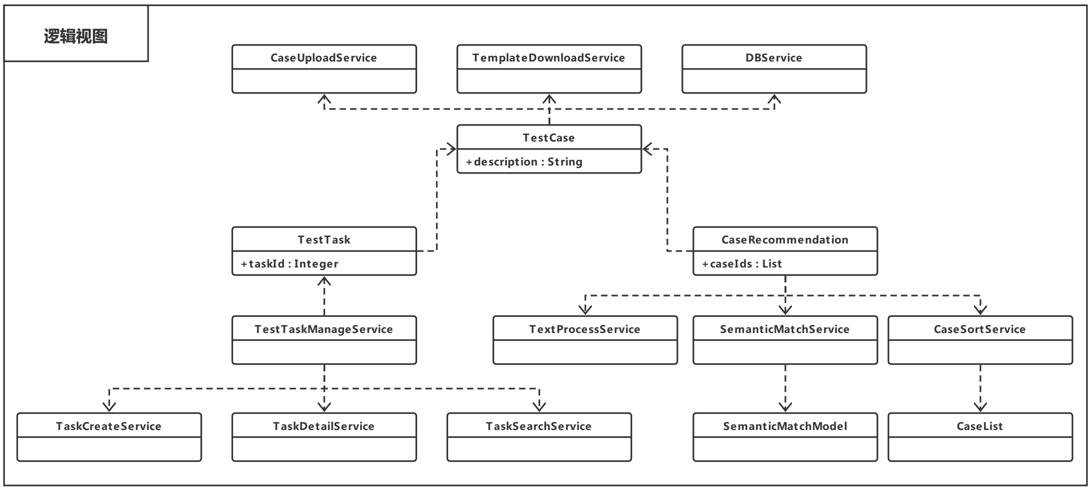

图 3.5: 逻辑视图

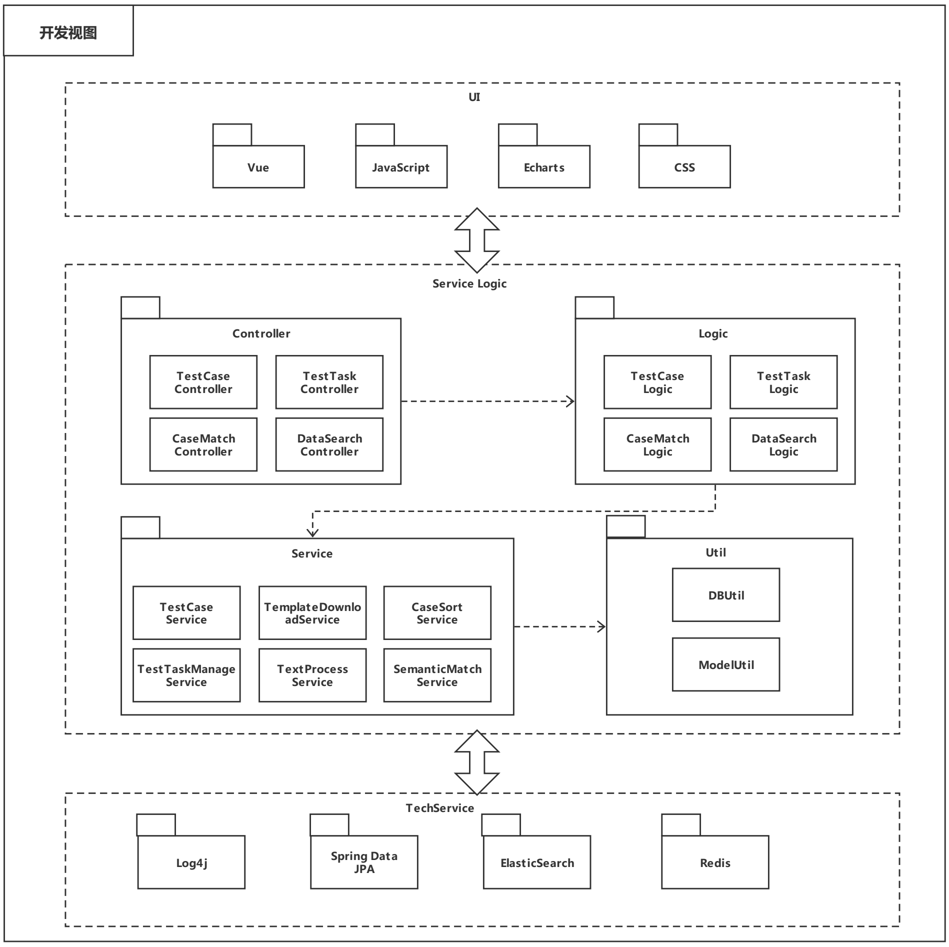

图 3.6: 开发视图

义匹配的功能，CaseSortService 提供测试用例排序的功能。

系统的开发视图如图3.6所示。开发视图是从系统开发人员的视角来描述系统组织架构设计。系统总体分为前端 UI、业务逻辑和技术服务支持。前端 UI 使用 Vue，Echarts 和 Javascript 构建主要页面，使用 CSS 调整组件样式。业务逻辑层分为 Controller，Logic，Service 和 Util，Controller 层负责路由路径匹配，与前端进行数据对接，Logic 层负责实现具体业务逻辑，Service 层负责接口抽象设计和功能提供，Util 层则是一些复用率高的功能的集合，相当于一个工具包。技术服务支持层是系统用到的底层服务，包括 Spring JPA，日志服务 Log4j，搜索引擎 ElasticSearch 和缓存服务 Redis。

系统的进程视图如图3.7所示。进程视图是从系统集成的视角来描述各个服务之间的进程通信。主线程启动并完成注册配置后，向测试用例分析服务发起同步调用，得到分析结果。主线程再发起测试任务创建，触发任务服务访问数据库，完成创建后返回结果。主线程发起对测试任务的访问，任务服务调用匹配服务执行测试用例匹配，匹配完成后将结果写入 Redis，返回匹配结果至主线程。

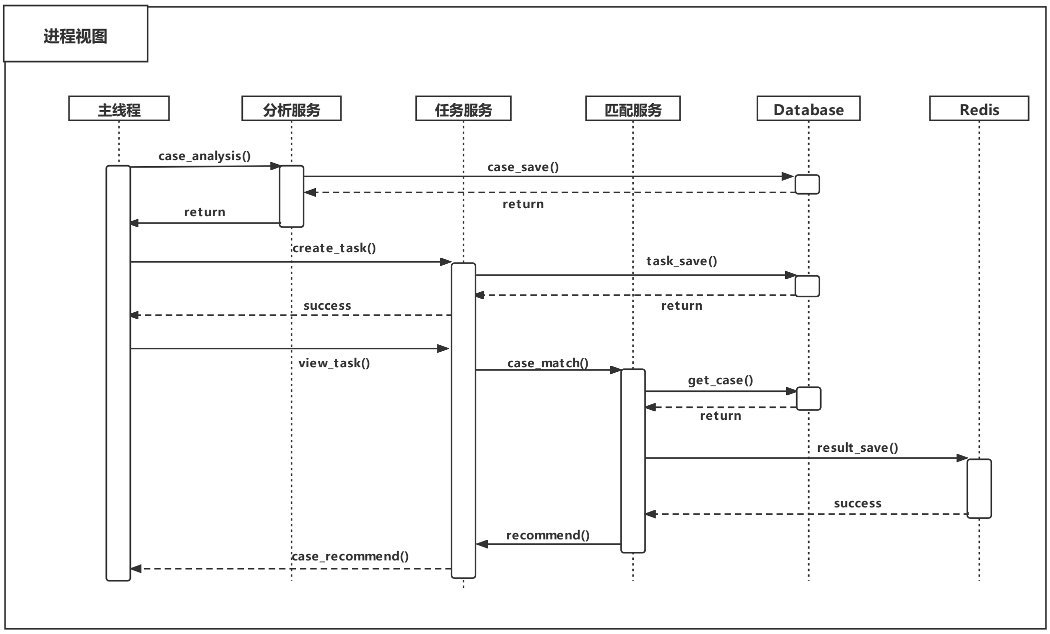

图 3.7: 进程视图

系统的物理视图如图3.8所示。物理视图又称为部署视图，是从运维人员视角对系统的物理节点间的通信方式和部署方式的描述，关注的是软件、硬件的映射和系统的可用性、性能等问题。系统使用 Nginx 解析用户请求后通过负载均衡算法将请求分发给对应的服务器。系统后端项目打包成 jar 包部署在服务器上，通过 RESTful 方式提供测试用例复用服务。系统的主要数据存储在 MySQL

数据库中，并配置了从数据库保证系统的可用性。系统运行过程的缓存数据存储在 Redis 服务器，与应用服务器通过 Redis 协议进行通讯。系统的文本向量数据存储在 ElasticSearch。

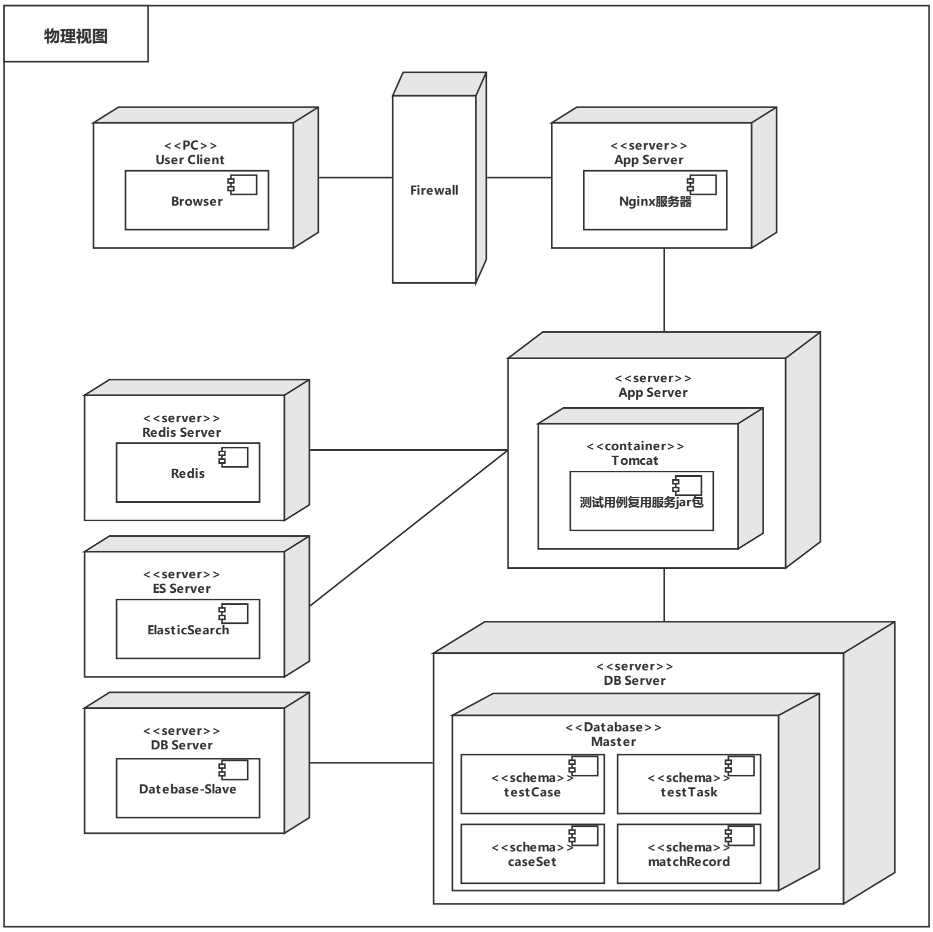

图 3.8: 物理视图

3.4　基于深度语义匹配的测试用例复用技术

3.4.1　测试用例定义

测试用例是软件测试过程的关键成果产出，测试用例的质量很大程度上决定着测试的质量。测试用例是为实现某测试目标而编写的一组测试输入、执行条件和预期结果。一个测试用例应至少具备以下要素：测试目标，测试环境，测试输入，测试步骤，预期结果。

本系统对于一个标准测试用例的各个字段规定如下。

1. 测试用例名称 (Test Case Name)，测试用例的名称，应简洁清晰，对测试用例内容进行概括；

2. 测试类型 (Test Type)，测试用例的类型，如功能测试，性能测试，可用性测试等，因不同的划分方法而异；

3. 测试功能点描述 (Test Description)，对测试用例希望测试的目标的描述，如验证被测对象的某一功能或某种性能；

表 3.17: 标准测试用例示例

字段	内容
测试用例名称	某网站登录功能测试用例
测试类型	功能测试
测试功能点描述	用户在登录页面正确输入账号密码后能够登录系统，进入网站首页
测试环境	操作系统：Windows 10 家庭版 浏览器：Chrome99.0.4844.82（正式版本）64 位
前置条件	无
后置条件	用户登录成功，处于已登录状态
测试步骤	1. 访问网站登录页面 2. 在账号文本框输入用户账号 xxxxxx 3. 在密码文本框输入密码 xxxxxx 4. 点击登录按钮
预期结果	1. 进入系统登录页面 2. 输入正常 3. 输入正常 4. 登录成功，进入用户首页
设计人员	鲁 xx
设计日期	2022 年 3 月 5 日

4. 测试环境 (Test Environment)，测试执行所需要的环境条件，包括软件环境、网络环境和物理环境等；

5. 前置条件 (Precondition)，定义了测试用例能够触发的先决条件，通常是系统状态；

6. 后置条件 (Post Condition)，定义了测试用例完成后的系统所处的状态；

7. 测试步骤 (Test Process)，测试用例的执行步骤，包括测试过程中的数据输入等；

8. 预期结果 (Expected Result)，用于和测试结果对比的标准结果，包括测试所得的数据、性能等系统表现；

9. 设计人员 (Designer)，设计该测试用例的设计人员姓名；

10. 设计日期 (Design Date)，设计该测试用例的日期时间记录。

根据本文对测试用例各字段的规定，如表3.17所示是本系统中的一个标准测试用例示例。

3.4.2 负采样技术

负采样技术最初是出现在 Skip-Gram 模型的训练中，后来广泛应用于自然语言处理和计算机视觉等领域。在深度文本匹配领域，大部分负例是毫不相干的 easy negatives，句子对之间完全不相似，这样难以训练出一个效果良好的模型。对于那些句子对相似但又不同的负例，被称为 hard negatives，这样的负例很大程度上决定着模型的效果，是训练中的关键数据。因此，如何多构造一些这样的 hard negatives，提高训练样本中这类型负例的比例，是负采样技术的关键。

本文使用的第一种负采样方法具体过程如下。首先，使用关键词拼接在原始检索输入的后面以增强原始检索，假设原始检索序列为 $\{p_1, p_2, \ldots, p_m\}$，关键词序列为 $\{q_1, q_2, \ldots, q_n\}$，则拼接后的检索序列为 $\{p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_n\}$。然后需要通过搜索引擎从数据集中挑选负样本，一个最直接的方式是通过搜索引擎所给出的样本的相似分数来判断，相似分数越低的样本往往就是负样本。此外，本文还引入了检索序列 Q 和候选 R 之间的字面重叠分数，目的是一个好的负样本在重叠与不重叠之间有好的平衡。最后，本文的负样本选取规则如下：

$$\begin{cases} 1, \text{ if } \mathrm{F}(Q,R) < \alpha \ and \ \frac{|Q \cup R| - |Q \cap R|}{|Q \cap R|} > \beta \\ 0, \text{ otherwise} \end{cases}$$

其中 $F(Q,R)$ 是搜索引擎工具提供的查询 Q 和结果 R 的相似分数，α 和 β 是用于调整的阈值参数。如果一个搜索出来的样本与查询值差异较大，那么它们的相似分数大概率会小于 α，而由于重叠部分很少，它们的重叠分数也很可能会大于 β。因此，这两个指标都是尽可能保证所得样本为负样本，但又不会让两个样本之间毫无关联，因为这样的样本几乎不会被搜索引擎检索出来。

本文使用的第二种负采样方法是随机实体替换方法。首先识别出句子中的实体词后，使用实体词库中的其他实体进行替换，生成新的负例样本，这样的样本看上去与原输入非常相似但含义大概率不同。例如，“测试教师主界面的新增考试功能”被实体替换后变为“测试教师主界面的新增作业功能”，这样的样本

就是内容相似的负样本。

3.4.3 测试用例匹配

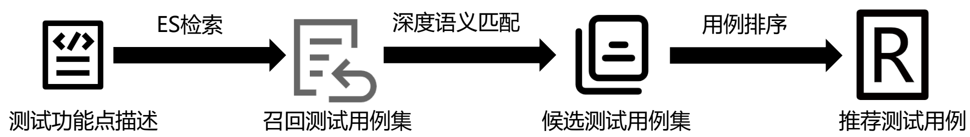

图 3.9: 测试用例匹配算法流程图

如图3.9所示是测试用例匹配算法流程图。首先提取测试功能点描述文本信息到 ElasticSearch 中进行检索，得到召回测试用例集。ElasticSearch 是一个分布式的全文搜索引擎，基于 RESTful web 接口提供服务。一般而言，ElasticSearch 会根据相关性分数 _score 对检索结果进行排序，这个相关性分数可以看作是 ElasticSearch 用来表示每个文档与查询的匹配程度。相关性分数 _score 表现为一个非负浮点数，分数越高表示搜索引擎认为其结果相关性越高。

本文使用的 ElasticSearch7.0 版本中，默认的相似度评分算法是 Okapi BM25 算法。假设一个查询为 Q，它包含关键字 $\{q_1, q_2, \ldots, q_n\}$，则文档 D 的 BM25 分数计算公式为：

$$\mathrm{Score}(Q, D) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\mathrm{avgdl}})} \tag{3.1}$$

其中 $f(q_i, D)$ 表示文档中的词频，即 q_i 在文档 D 中出现的频率。$IDF(q_i)$ 表示逆文档频率，用于衡量这个词提供的信息量多少，其含义是若一个查询词是很罕见的且在某个文档里出现了，则其分数会很高，反之对于常见的词分数就不高。$|D|$ 表示文档的词数量，$avgdl$ 表示文档平均词数量。k_1 和 b 是算法中的两个可调节变量。k_1 控制非线性词频率归一化，默认设置值为 1.2，用于控制词频对于计算得分的重要性，设想这样一个场景，如果一个查询词在文档中出现频率越高，则该文档越接近我们希望检索得到的内容，那么就应该将 k_1 设置得大一些。b 控制文档长度对于分数的惩罚力度，默认设置值为 0.75，它作为文档长度比率的系数，如果较大，则文档长度对计算分数的影响较大，如果设置为 0，那么长度比率将无法影响分数高低，文档的长度将不作为影响因子。

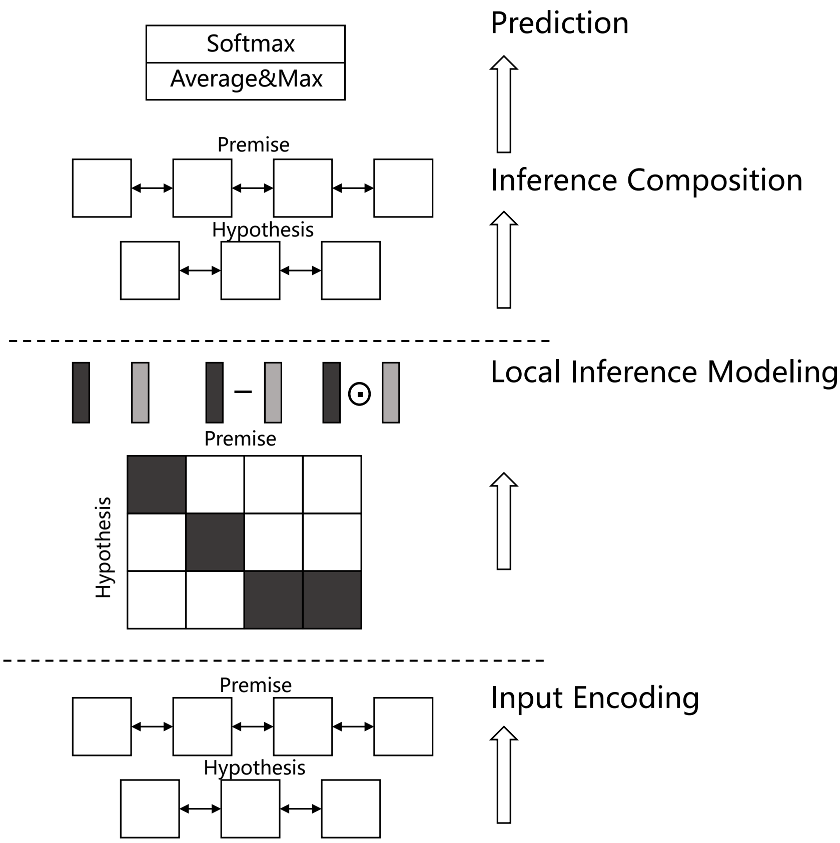

图 3.10: ESIM 模型结构图

ES 检索得到召回测试用例集之后，取相关性分数最高的 K 个测试用例进行深度语义匹配的计算，系统中 K 的大小可通过配置文件调节，默认取值 100。本文使用 ESIM 模型来进行深度语义匹配计算。ESIM 模型是一个综合了 BiLSTM 和注意力机制的模型，在文本匹配方面的效果非常出色。

ESIM 模型结构如图3.10所示。ESIM 模型结构分为四个部分：Input Encoding，Local Reference Modeling，Inference Composition，Prediction。首先是 Input Encoding 层，该层的输入是两个待匹配的句子，首先通过 word embedding 得到句子的向量化表示，然后输入到 BiLSTM 重新编码，得到编码后的向量作为该层的输出。因为这里的 word embedding 是基于 GloVe 预训练得到的，没有反映句子中的上下文关系，所以需要 BiLSTM 重新编码一下，生成新的输入编码向量。编码公式如下：

$$\begin{aligned} \bar{\mathbf{a}}_i &= \text{BiLSTM}(\mathbf{a}, i), \forall i \in [1, \ldots, \ell_a], \\ \bar{\mathbf{b}}_j &= \text{BiLSTM}(\mathbf{b}, j), \forall j \in [1, \ldots, \ell_b]. \end{aligned} \tag{3.2}$$

其中 $\text{BiLSTM}(\mathbf{a}, i)$ 表示对文本 a 的第 i 项进行编码。

在 Local Reference Modeling 层，会对两个句子的向量进行局部推理建模，利用 attention 让两个句子发生交互以分析其联系。首先让之前得到的两个向量相乘：

$$e_{ij} = \bar{\mathbf{a}}_i^T \bar{\mathbf{b}}_j \tag{3.3}$$

如果两个词的联系较深，其向量乘积也会较大。然后再进行如下处理：

$$\begin{aligned}
\tilde{\mathbf{a}}_i &= \sum_{j=1}^{\ell_b} \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{\ell_b} \exp\left(e_{ik}\right)} \bar{\mathbf{b}}_j, \forall i \in [1, \ldots, \ell_a], \\
\tilde{\mathbf{b}}_j &= \sum_{i=1}^{\ell_a} \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{\ell_a} \exp\left(e_{kj}\right)} \bar{\mathbf{a}}_i, \forall j \in [1, \ldots, \ell_b].
\end{aligned} \tag{3.4}$$

上述处理过程的含义是，假设输入的第一句话中有个词为“登录”，先分析该词与第二句话中每个词的联系，计算的 e_{ij} 作为权重值，然后用第二句话中的各个词向量乘以权重去表示“点击”这个词。依照这样的方式逐个分析，得到新的待计算序列。为了判断两个句子的联系大小，需要分析两个序列的差异，ESIM 的处理方式是计算它们的差与点积，并且将向量拼接到一起：

$$\begin{aligned}
\mathbf{m}_a &= [\bar{\mathbf{a}}; \tilde{\mathbf{a}}; \bar{\mathbf{a}} - \tilde{\mathbf{a}}; \bar{\mathbf{a}} \odot \tilde{\mathbf{a}}], \\
\mathbf{m}_b &= [\bar{\mathbf{b}}; \tilde{\mathbf{b}}; \bar{\mathbf{b}} - \tilde{\mathbf{b}}; \bar{\mathbf{b}} \odot \tilde{\mathbf{b}}].
\end{aligned} \tag{3.5}$$

然后是 Inference Composition 层，该层将 Local Inference Modeling 的结果输入 BiLSTM 做一次全局分析，提取上下文信息：

$$\begin{aligned}
\mathbf{v}_{a,t} &= \operatorname{BiLSTM}\left(F\left(m_{a,t}\right), t\right), \\
\mathbf{v}_{b,t} &= \operatorname{BiLSTM}\left(F\left(m_{b,t}\right), t\right).
\end{aligned} \tag{3.6}$$

其中 F 是单层神经网络，用于减少模型参数，避免过拟合。t 表示 BiLSTM 在 t 时刻的输出值。考虑到对于不同的句子向量长度不同，还需要进行池化处理，结果储存到长度固定的向量中。因为求和运算对序列的长度是敏感的，所以 ESIM 同时进行了 Max 池化和 Average 池化，结果放入一个向量：

$$\begin{aligned}
\mathbf{v}_{a,\text{ ave}} &= \sum_{i=1}^{\ell_a} \frac{\mathbf{v}_{a,i}}{\ell_a}, \quad \mathbf{v}_{a,\max} = \max_{i=1}^{\ell_a} \mathbf{v}_{a,i} \\
\mathbf{v}_{b,\text{ ave}} &= \sum_{j=1}^{\ell_b} \frac{\mathbf{v}_{b,j}}{\ell_b}, \quad \mathbf{v}_{b,\max} = \max_{j=1}^{\ell_b} \mathbf{v}_{b,j} \\
\mathbf{v} &= \left[\mathbf{v}_{a,\text{ ave}}; \mathbf{v}_{a,\max}; \mathbf{v}_{b,\text{ ave}}; \mathbf{v}_{b,\max}\right]
\end{aligned} \tag{3.7}$$

最后，在 Prediction 层把向量 v 放入分类器，在输出层使用 softmax 函数，得到输出结果。

经过深度语义匹配计算后，匹配分值达到阈值的测试用例即为候选测试用例，接下来将进入用例排序环节。在本系统中，用例排序将根据搜索引擎得分，语义匹配得分，测试用例推荐频次和测试用例复用频次来计算最终的综合推荐分数。计算公式如下：

$$\text{Score} = S_1 \cdot \alpha + S_2 \cdot \beta - \log \frac{F_r + 1}{F_u + 1} \tag{3.8}$$

其中 S_1 为搜索引擎相关性得分，S_2 为语义匹配得分，F_r 为测试用例被推荐的历史累积频次，F_u 为测试用例被复用的历史累积频次。设想这么一种情况，一个测试用例多次被匹配成功并推荐为可复用测试用例，但是它几乎没有被用户选用过，那么对于这样的测试用例，应该降低它的推荐权重。于是本文在设计得分公式时引入了 F_r 和 F_u，根据它们的比值来调整一个测试用例的“推荐价值”。此外，α 和 β 是两个可调节参数，用于调整搜索引擎相似度分数和语义匹配分数对于测试用例排序得分的影响大小，便于根据不同的使用场景和语料数据来调整测试用例匹配算法的效果。

3.5 测试用例分析模块设计

3.5.1 架构设计

测试用例分析模块架构图如图3.11所示。用户首先下载测试用例模板，了解系统要求的测试用例格式规范。按规范组织好数据后，用户可以上传历史测试用例集，在上传过程中有格式检查和文件校验，若不通过则需重新上传。上传成功的测试用例数据被存储至 MySQL 数据库，这部分数据成为系统的待分析数据并进入后续环节。待分析的测试用例会被执行文本信息提取，提取的主要字段包括测试类型，测试功能点描述，测试步骤等。一方面，这些文本信息将进行分词、去除停用词。另一方面，系统会预先搭建向量模型，通过预先准备的语料去训练模型以得到有效的向量模型，再将待分析测试用例的文本信息转化为向量。最后，系统将测试用例的分词处理结果和对应向量信息包装成统一格式的数据结构，存储至 ElasticSearch，并为其建立索引。

3.5.2 详细设计

测试用例分析模块核心类图如图3.12所示。FileTransfer 是文件传输的基类。TemplateDownloader 类提供 download() 方法，负责数据模板的下载。CaseUploader

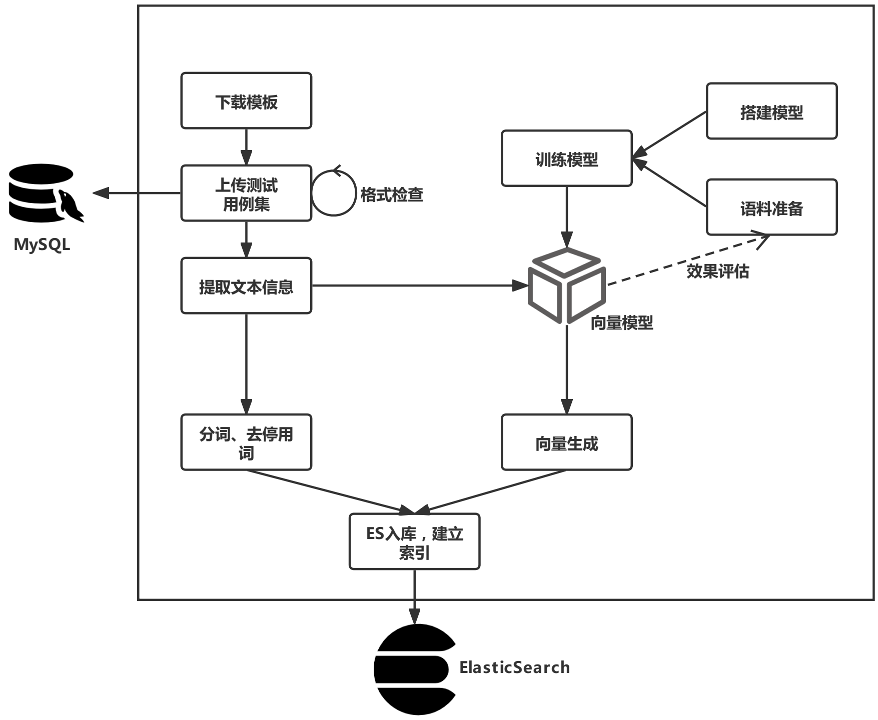

图 3.11: 测试用例分析模块架构图

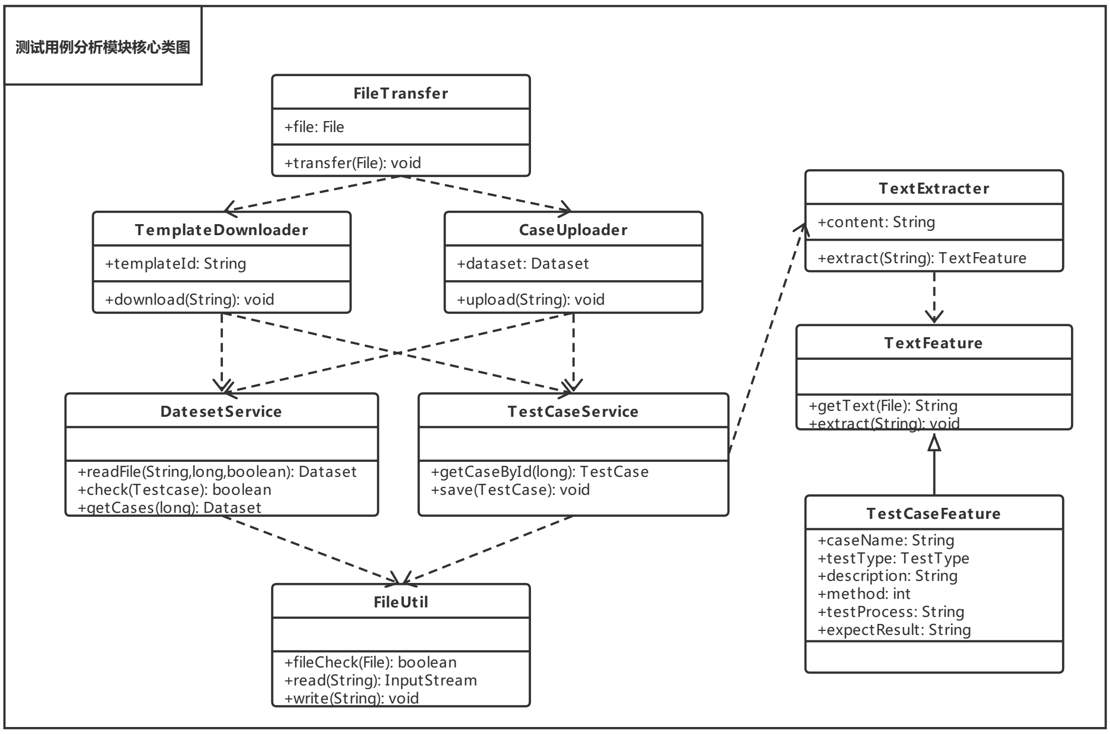

图 3.12: 测试用例分析模块核心类图

类提供 upload() 方法，负责测试用例数据的文件上传。这二者都依赖 DatasetService 和 TestCaseService 提供的功能。DatasetService 是测试用例数据集的功能入口，readFile() 方法实现原始文件读取数据，check() 方法实现文件的格式检查和内容校验。TestCaseService 对外提供测试用例查询和保存的一些方法。FileUtil 是文件工具类，提供文件检查和读写文件的基础方法。

内容分析部分，TextExtracter 是信息提取的基类，提供 extract() 方法执行文本内容的提取和分析。TextFeature 类是文本内容特征基类，它的子类 TestCaseFeature 是测试用例特征类，封装了待分析的测试用例的基础属性，包括用例名称，测试类型，测试步骤等。

3.5.3 数据库设计

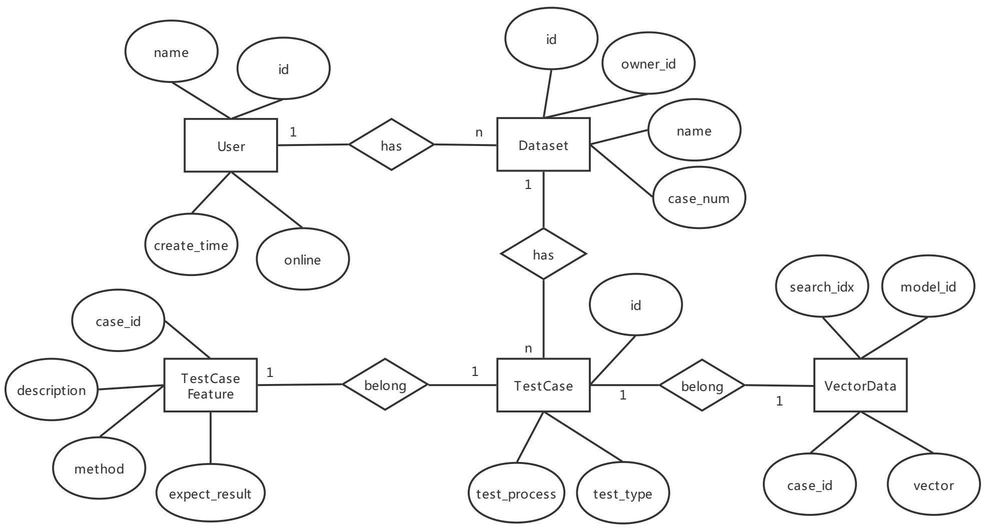

图 3.13: 测试用例分析模块 ER 图

测试用例分析模块 ER 图如图3.13所示。User 表是系统的注册用户，Dataset 表是测试用例集，二者是一对多关系，表示以一个用户可以是多个测试用例数据集的所有者。TestCase 表是系统中的测试用例，它与 Dataset 也是一对多关系，表示一个测试用例集中有多条测试用例。VectorData 表是测试用例对应的向量信息及其索引，TestCaseFeature 表是测试用例对应的测试用例特征，它们与测试用例都是一对一关系。

如表3.18所示是 User 表各个字段的详细说明，该表记录了注册用户的个人信息，主要包括了 id、name、create_time 和 online 等字段。

如表3.19所示是 Dataset 表各个字段的详细说明，该表记录了一个数据集的详细信息，主要包括了 id、name、owner_id、case_num 和 create_time 等字段。

表 3.18: User 表

字段	含义	类型	说明
id	用户 id	BIGINT	id 为表的唯一主键
name	用户名称	VARCHAR	用户名称必须唯一，不可重复
create_time	用户注册时间	TIMESTAMP	无
online	用户在线状态	INT	1 表示在线，0 表示离线
availability	账户是否可用	INT	1 表示可用，0 表示不可用

表 3.19: Dataset 表

字段	含义	类型	说明
id	数据集 id	BIGINT	id 为表的唯一主键
name	数据集名称	VARCHAR	数据名称必须唯一，不可重复
owner_id	所属用户 id	BIGINT	无
case_num	用例数量	INT	无
create_time	数据集上传时间	TIMESTAMP	无
description	数据集描述	TIMESTAMP	无
field	专业领域	TIMESTAMP	指对应软件的所处领域

如表3.20所示是 TestCase 表各个字段的详细说明，该表记录了一条测试用例的详细信息，主要包括了 id、owner_id、test_type 和 test_process 等字段。

表 3.20: TestCase 表

字段	含义	类型	说明
id	用例 id	BIGINT	id 为表的唯一主键
own_id	所属用户 id	BIGINT	无
test_type	测试类型	VARCHAR	无
test_process	测试步骤	VARCHAR	无
test_requirement	测试功能点描述	VARCHAR	无
constraint	前提和约束	VARCHAR	无
expected_result	预期结果	VARCHAR	无
dataset_id	所属数据集 id	BIGINT	无
create_time	上传时间	TIMESTAMP	无
remark	备注	VARCHAR	无

如表3.21所示是 VectorData 表各个字段的详细说明，该表记录了一个测试用例对应的向量，主要包括了 id、model_id 和 case_id 等字段。

表 3.21: VectorData 表

字段	含义	类型	说明
id	数据集 id	BIGINT	id 为表的唯一主键
model_id	模型 id	BIGINT	对应向量模型 id
case_id	用例 id	BIGINT	对应测试用例 id
vector	向量数据	TEXT	无

3.6 测试任务管理模块设计

3.6.1 架构设计

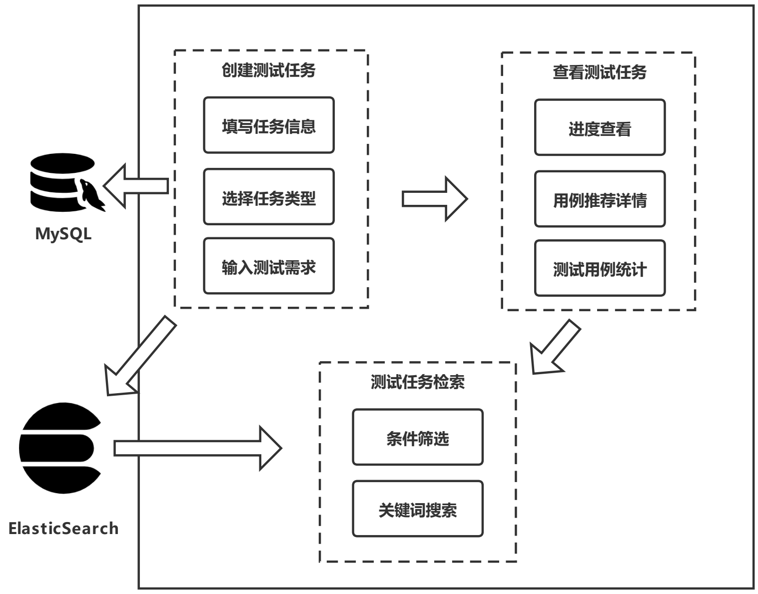

图 3.14: 测试任务管理模块架构图

测试任务管理模块架构图如图3.14所示。本模块从用户创建测试任务开始，创建任务需要的步骤包括填写任务信息 taskInfo，选择测试类型 testType，输入测试需求 testRequirement。用户填写完毕后确定创建，任务表单信息将在经过校验后存入 MySQL 数据库，部分字段信息将被存入 ElasticSearch。创建任务成功后，用户可在测试任务管理界面查看任务。用户可查看任务进度，包括任务运行时长，任务当前状态。已完成的任务，用户可以查看历史测试用例的推荐详情，系统为每一条测试功能点描述都推荐了若干个历史测试用例，并做了测试用例统计情况分析。为方便用户选择测试任务，系统支持通过条件筛选和直接搜索关键词的方式来查找测试任务。

3.6.2 详细设计

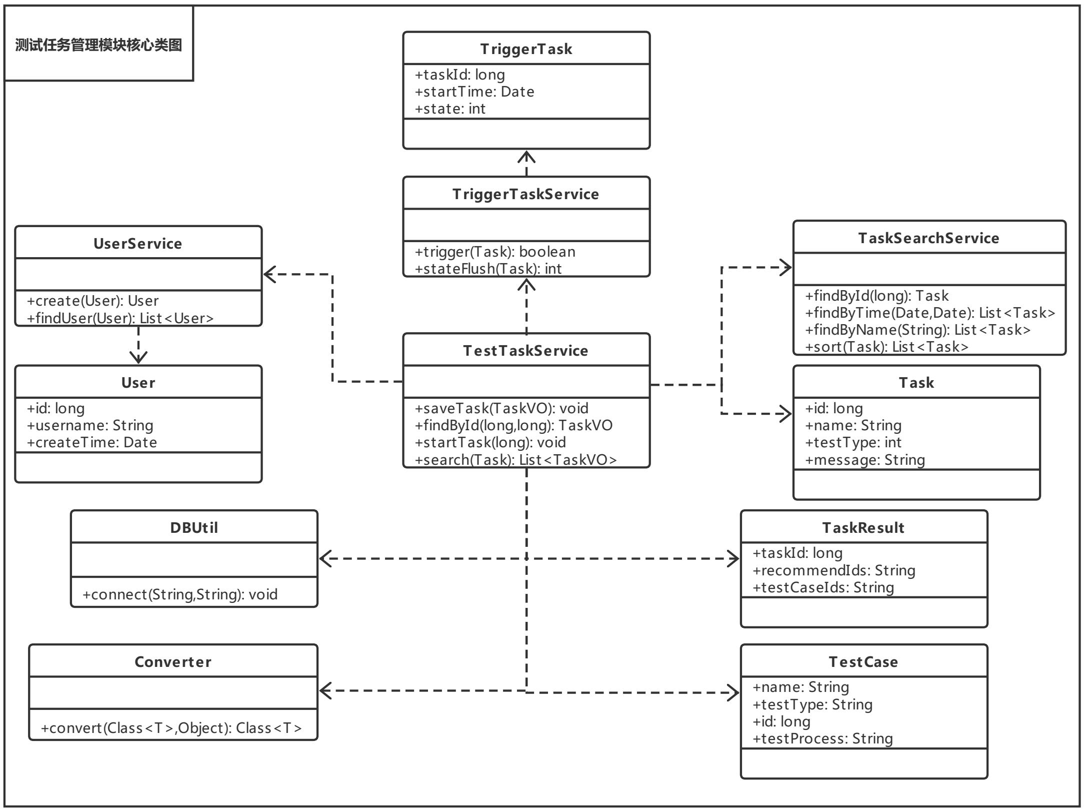

图 3.15: 测试任务管理模块核心类图

测试任务管理模块核心类图如图3.15所示。TestTaskService 是该模块的核心接口类，其 saveTask() 方法提供保存所创建的任务功能，findById() 提供根据 ID 精确查询测试任务功能，startTask() 是触发任务执行的关键方法，search() 方法是根据特定条件检索测试任务的方法。本模块用到了 UserSevice 中提供与用户身份检查和查询用户的功能。TriggerService 实现了测试任务状态刷新和触发执行的功能，通过监听器 TaskListener 监听任务的实时状态，系统根据监听器返回值来设定任务的多种状态，包括运行中、成功和失败。TaskSearchService 是实现了任务检索功能，提供多个检索测试任务的查询方法和排序方法。DBUtil 是数据库相关的工具类，Converter 是对象转换的工具类，在本模块中都被使用到。

3.6.3 数据库设计

测试任务管理模块 ER 图如图3.16所示。用户在使用测试用例复用功能之前需要先创建一个测试任务，该任务会被系统自动执行，用户可以在测试任务管理

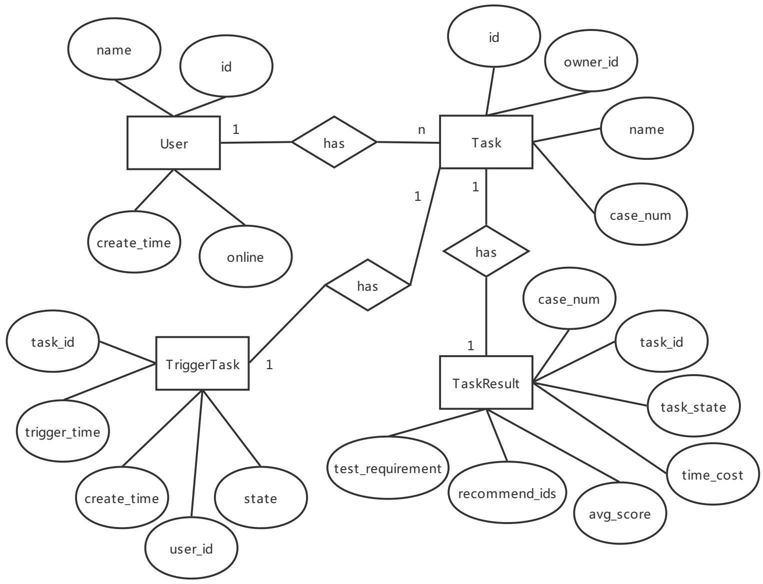

图 3.16: 测试任务管理模块 ER 图

页面查看任务状态。Task 表是系统的测试任务所对应的表，测试任务的相关信息在用户创建测试任务时获得。TaskResult 表是测试任务结果信息，TriggerTask 表是测试任务执行信息，它们与 Task 表都是一对一关系，表示一个任务对应唯一一条测试执行信息记录和唯一一个测试任务结果。

如表3.22所示是 Task 表各个字段的详细说明，该表记录了一个测试任务的详细信息，主要包括了 id、owner_id、name 和 test_type 等字段。

表 3.22: Task 表

字段	含义	类型	说明
id	任务 id	BIGINT	id 为表的唯一主键
owner_id	所属用户 id	VARCHAR	无
name	任务名称	VARCHAR	任务名称必须唯一，不可重复
test_type	测试类型	VARCHAR	只支持单类型任务
case_num	期望用例数量	INT	期望获得的推荐测试用例数量
recommend_ids	推荐用例	INT	推荐测试用例 id 列表
create_time	创建时间	TIMESTAMP	任务创建时间

如表3.23所示是 TriggerTask 表各个字段的详细说明，该表记录了一个测试任务的执行信息，主要包括了 id、user_id、task_id 和 create_time 等字段。

如表3.24所示是 TaskResult 表各个字段的详细说明，该表记录了一个测试任务的匹配结果，主要包括了 id、task_id、和 case_num 等字段。

表 3.23: TriggerTask 表

字段	含义	类型	说明
id	记录 id	BIGINT	id 为表的唯一主键
user_id	所属用户 id	BIGINT	无
task_id	任务 id	BIGINT	任务 id
create_time	创建时间	TIMESTAMP	无
trigger_time	执行时间	TIMESTAMP	任务触发执行的时间
state	任务状态	INT	任务当前执行状态

表 3.24: TaskResult 表

字段	含义	类型	说明
id	记录 id	BIGINT	id 为表的唯一主键
task_id	任务 id	BIGINT	任务 id
case_num	用例数量	INT	推荐用例数量
task_state	任务状态	TIMESTAMP	无
time_cost	任务时长	BINGINT	任务执行的时长
test_requirement	测试功能点描述	VARCHAR	无
recommend_ids	用例 id 列表	VARCHAR	推荐测试用例 id 列表
avg_score	用例平均得分	VARCHAR	推荐测试用例的平均匹配得分

3.7 测试用例匹配模块设计

3.7.1 架构设计

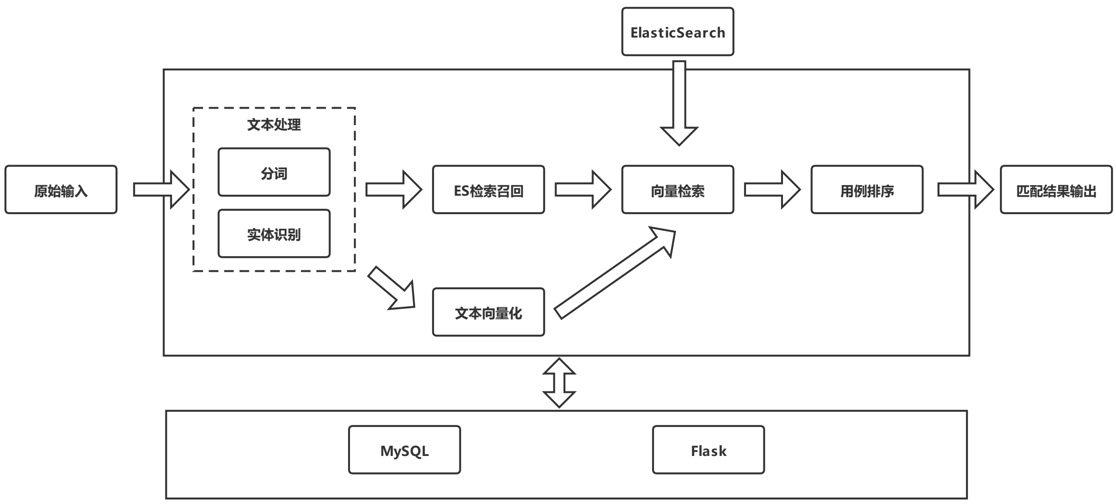

图 3.17: 测试用例匹配模块架构图

测试用例匹配模块架构图如图3.17所示。本模块的原始输入是用户填写的测试功能点描述信息，以自然语言文本的形式作为输入。文本输入首先将经过文本预处理，主要包括文本分词和实体识别，得到的分词结果和实体识别结果将在后面的 ES 检索环节作为检索的部分关键词，也会在用例排序环节对用例的排序得分赋予一定的权重。ElasticSearch 检索召回的测试用例作为候选测试用例集，再通过深度语义匹配模型将用户输入和候选测试用例集进行语义匹配预测，匹配得分达到阈值的用例将被选中，进入用例排序环节。用例排序时，测试用例的语义匹配得分，关键词字面得分和推荐热度得分将被综合加权计算得到推荐得分，最终排序输出结果。

3.7.2 详细设计

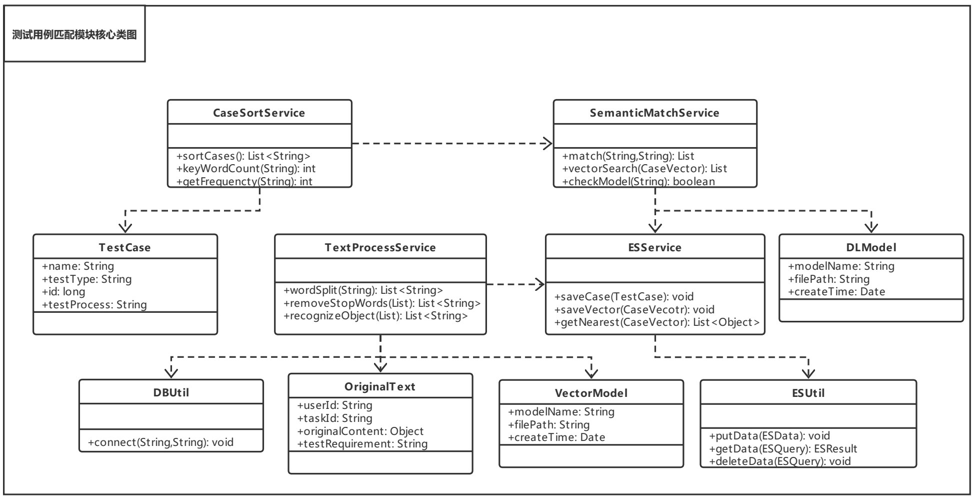

图 3.18: 测试用例匹配模块核心类图

测试用例匹配模块核心类图如图3.18所示。TextProcessService 是文本处理功能接口，wordSplit() 方法提供文本分词功能，removeStopWords() 提供停用词检查与去除功能，recognizeObject() 提供实体识别功能。OriginalText 是原始输入文本信息的封装类。VectorModel 是预先训练好的向量模型的封装类。ESService 提供与 ElasticSearch 交换数据所需要的接口功能，包括存储测试用例结构化信息，存储测试用例文本向量化信息和 ES 库中进行向量检索。DBUtil 是 MySQL 数据库工具类，ESUtil 是 ElasticSearch 工具类，在本模块都被使用到。SemanticMatchService 实现了用例匹配功能，其依赖的 DLModel 是深度语义匹配模型的封装类。CaseSortService 实现了用例排序的功能，主要依据用例的模型匹配得分，字面得分和推荐热度来排序。

3.7.3 数据库设计

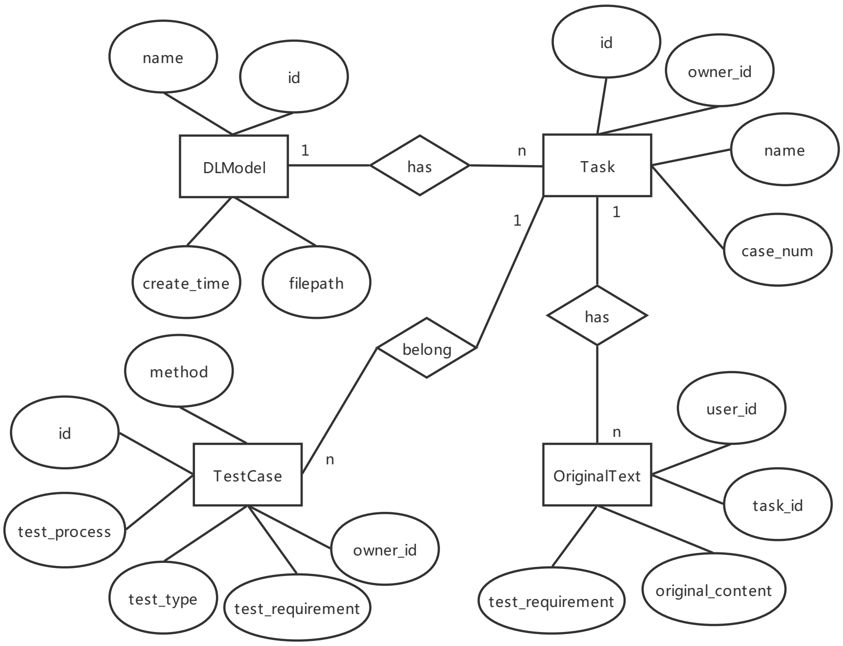

图 3.19: 测试用例匹配模块 ER 图

测试用例匹配模块 ER 图如图3.19所示。DLModel 表是深度语义匹配模型，它对应多个测试任务，表示多个测试任务会调用同一个模型。OriginalText 是测试任务原始输入文本，它和 Task 是一对多关系。

如表3.25所示是 DLModel 表各个字段的详细说明，该表记录了一个测试任务的匹配结果，主要包括了 id、name、和 create_time 等字段。

表 3.25: DLModel 表

字段	含义	类型	说明
id	模型 id	BIGINT	id 为表的唯一主键
name	模型名称	BIGINT	模型名称不可重复
create_time	创建时间	TIMESTAMP	无
filepath	模型文件路径	VARCHAR	模型文件绝对路径
type	模型类型	VARCHAR	无
size	模型文件大小	BIGINT	单位为 MB

如表3.26所示 OriginalText 表各个字段的详细说明，该表记录了一个测试任务的匹配结果，主要包括了 id、user_id、task_id、和 original_content 等字段。

表 3.26: OriginalText 表

字段	含义	类型	说明
id	记录 id	BIGINT	id 为表的唯一主键
user_id	用户 id	BIGINT	无
task_id	任务 id	BIGINT	无
original_content	原始输入	VARCHAR	原始输入文本信息
test_requirement	测试功能点描述	VARCHAR	无

3.8 本章小结

本章对测试用例复用系统的需求进行分析，并阐述了系统的概要设计和各模块详细设计。需求分析包括功能性需求分析、非功能性需求分析和系统用例分析，系统的概要设计主要是系统架构设计图和 4+1 视图，功能模块分为测试用例分析模块、测试任务管理模块和测试用例匹配模块，通过模块架构图、模块核心类图和数据库表设计来介绍各个模块的具体内容。

第四章　基于深度语义匹配的测试用例复用系统的实现

4.1　测试用例分析模块的实现

4.1.1　文件传输实现

本系统是测试用例复用系统，所有的历史测试用例均由用户自主上传。由于不同用户的历史测试用例的格式不完全相同，为避免格式问题给用户带来的困扰，系统提供可下载的测试用例集模板，详细定义了系统所兼容的测试用例数据格式，用户可根据模板对历史测试用例重新组织，然后再上传至系统。上述过程涉及模板下载和测试用例集上传，均属于文件传输问题。本小节将详细讲述本模块文件传输部分的实现。

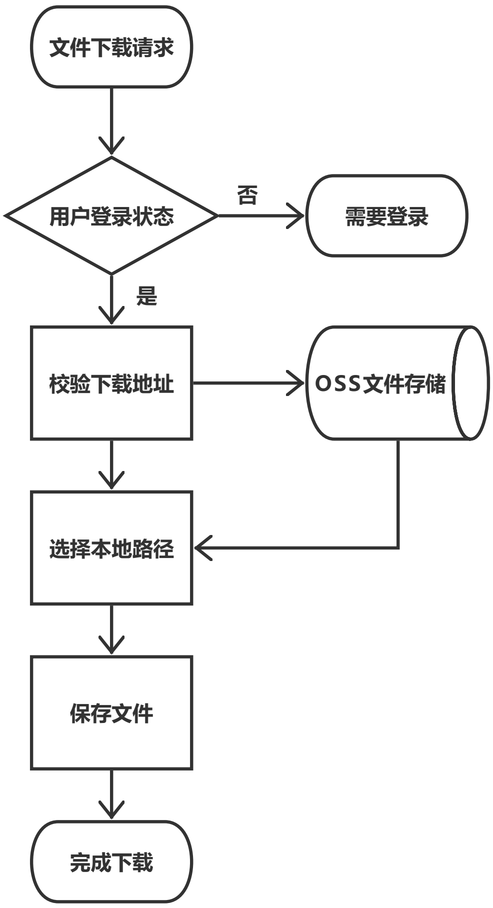

图 4.1: 测试用例集模板下载流程图

如图4.1所示是测试用例集模板下载流程图。用户再下载页面点击链接下载模板时，会发送文件下载请求。接收到前端发送的下载请求后，系统会检查用户的登录状态，若状态不可用，将拒绝该下载请求并返回结果，提示用户请先完成登录。若状态可用，系统将校验文件的 URL 下载地址，若下载地址无误，则发送 token 序列等必要信息给前端。通过后端发来的 token 序列等信息，与远程的 OSS 文件存储服务器建立连接后，用户即可选择本地路径下载模板文件。

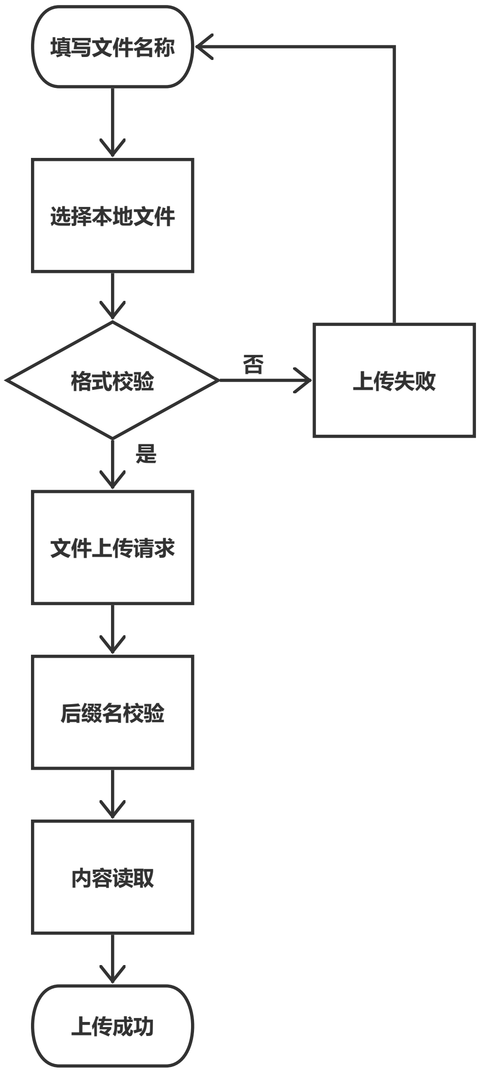

图 4.2: 测试用例集上传流程图

如图4.2所示是测试用例集上传流程图。在上传测试用例集页面，用户首先填写测试用例集名称，该名称必须唯一，将作为后续的测试用例集检索等操作的标识。用户点击上传按钮，打开文件选择窗口，选择本地文件上传。系统将对

上传内容进行校验，例如文件名称唯一性，文件大小不能超过 500M。若格式校验不通过，系统提示“上传失败”，用户将需要重新上传。格式校验通过后，系统将向后端发起文件上传请求，后端接收请求后，会对接收到的各个文件参数进行校验，例如文件后缀名是否在支持范围内，上传用户 ID 是否缺失。参数校验通过后，系统将读取文件内容，解析出其中的字段与对应内容。对应字段的内容封装到实体对象之后，系统将其持久化保存至数据库。

表 4.1: Dataset 类详情列表

字段	含义	类型	说明
id	数据集编号	long	主键，不为空
name	数据集名称	String	检索依据，应唯一
ownerId	创建者 ID	long	测试用例集的创建者，也是其所有者
testCaseNum	测试用例数量	int	所包含测试用例的数量
size	数据集文件大小	double	单位为 MB
caseList	测试用例列表	List<Long>	所包含测试用例的 ID 列表
fileType	文件类型	String	测试用例集文件类型
createTime	创建时间	Timestamp	一般为文件上传时的 UTC 标准时间

表 4.2: TestCase 类详情列表

字段	含义	类型	说明
id	测试用例 ID	long	主键，不为空
caseName	测试用例名称	String	可以不唯一
ownerId	创建者 ID	long	测试用例的上传者，也是其所有者
testType	测试类型	String	测试用例所属类型
testProcess	测试步骤	String	测试用例的测试步骤
testRequirement	测试功能点描述	String	测试用例的测试功能点描述
precondition	前置条件	String	测试用例的前置条件
postcondition	后置条件	String	测试用例的后置条件
expectedResult	预期结果	String	测试用例的预期结果
designer	设计人员	String	测试用例的设计人员
designTime	设计日期	String	测试用例的设计日期
datasetId	测试用例集 ID	long	所属测试用例集的 ID
linkIdList	关联测试用例列表	List<Long>	同一测试用例集的其他测试用例 ID 列表
createTime	创建时间	Timestamp	测试用例上传时 UTC 时间

测试用例集内容持久化的核心类是 Dataset 类和 TestCase 类。Dataset 类是测试用例集的实体类，其详情如表4.1所示。TestCase 类是测试用例的实体类，其

详情如表4.2所示。

4.1.2 测试用例分析流程

图 4.3: 上传测试用例集截图

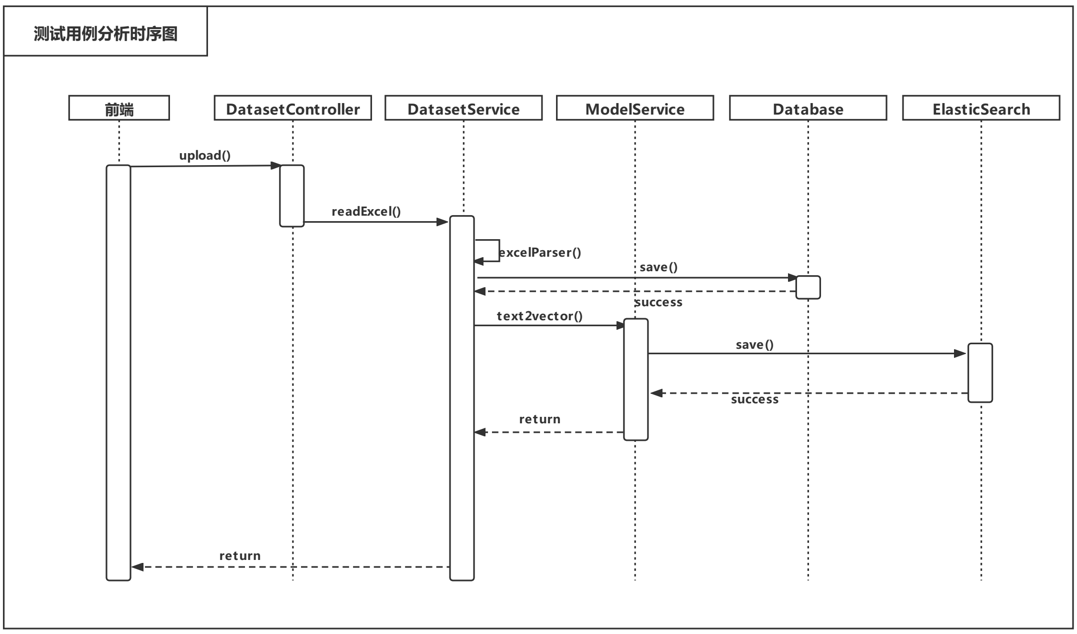

图 4.4: 测试用例分析时序图

在分析测试用例之前，用户需要先将测试用例集上传至系统中，上传测试用例集页面如图4.3所示。系统同时支持.xls 和.xlsx 格式的文件上传，上传时需

要正确命名测试用例集名称，避免和以往的测试用例集名称混淆，也便于用户后续对测试用例集的查询、筛选等操作。若用户对测试用例的格式规范不了解，还可以在当前页面直接下载数据集模板文件参照模板文件的字段来组织填写测试用例集。

测试用例分析模块的时序图如图4.4所示。在上传测试用例集之前，首先调用 downloadFile() 方法通过远程 URL 地址下载数据模板文件。下载成功后，用户根据模板填充待上传数据集。DatasetController 是测试用例分析的入口，用户选择本地文件并上传，系统根据 URL 访问路径将 HTTP 请求分发到指定路由方法 uploadByExcel(), 同时解析 HTTP 请求中的参数，包括数据集名称 name，上传者身份 ownerId 和文件对象 multipartFile。解析出的值作为参数，然后调用 DatasetService 中的 readExcel() 方法，检查文件后缀名并读取 Excel 表中的内容，然后调用 DatasetRepository 的 save() 方法保存到 MySQL 数据库。测试用例入库后，系统将生成测试用例分析的异步任务 AnalysisTask，测试用例数据需经过 ModelService 的向量化处理，调用其 text2Vector() 方法，将测试功能点描述 test_requirement 和测试步骤 test_process 等字段转化为高维向量，然后存储至 ElasticSearch。上述步骤都成功后，系统提示用户上传测试用例集成功。

表 4.3: 测试用例分析主要类列表

分类	包含的类
控制器层	BaseController, DatasetController, FileController, TestCaseController, UserController, AnalysisController
服务层	DatesetService, ModelService, ESService, UserSevice, NLPService, F-ileTransferService
数据访问层	DatasetDAO, TestCaseDAO, AnalysisRecordDAO
数据类	DatasetVO, TestCaseVO, UserVO, AnalysisTaskVO
配置类	ExecutorConfig, RestTemplateConfig, SwaggerConfig

如表4.3所示是测试用例分析模块设计的主要类，包括控制器层、服务层、数据访问层、数据类和配置类。控制器层的类负责接收请求和分发至各处理方法，请求的具体处理过程由服务层提供，数据访问层提供对数据库的访问，主要是增删改查操作，数据类是对实体的封装，配置类则包含系统运行的一些必要配置信息和逻辑。其中 BaseController 抽象类是所有 Controller 类的基类，它的子类 DatasetController 等负责提供对外接口，各自调用了对应的 Service 类，DatasetService 类提供测试用例集的创建、保存等方法，ModelService 类提供测试用例文本向量化相关的方法，ESService 类提供 ElasticSearch 文档创建、保存等方法，UserService 类提供用户登录状态查询、用户权限检查等方法，NLPService

类提供测试用例文本分词、实体识别等方法。模块所涉及的类还包括数据库访问相关的 DAO 类，数据实体 VO 类和一些配置类。

表 4.4: AnalysisTask 类详情列表

字段	含义	类型	说明
id	任务编号	long	主键，不为空
queue	队列名称	String	消息队列的名称
datasetId	测试用例集 ID	long	任务对应的测试用例集 ID
status	任务状态	int	用例分析任务的状态，分为成功、失败和进行中
params	任务参数	Map<String,Object>	执行任务所需的参数，例如向量模型路径
callback	回调参数	Map<String,Object>	任务执行完成后回调方法的参数，例如 ES 文档的 ID，在回调方法中需要将执行结果保存至 ES

用户上传测试用例集后，系统会自动创建测试用例分析任务，一个测试用例集对应一个测试用例分析任务 AnalysisTask，其详细描述如表4.4所示。分析任务主要执行测试用例文本的分词，测试步骤、测试功能点描述等关键字段的向量化。核心执行参数封装在 Map 类型的 params 中，而回调参数封装成 Map 类型的 callback。

每个用例分析任务的结果对应实体类 AnalysisResult，主要属性有分析任务 ID、测试用例集 ID、ES 文档 ID 和任务执行结果。其中核心属性是 Map 类型的 result，它保存了测试用例分析的多种结果，包括分词结果和向量化数据。测试用例文本的分词结果和向量化结果将保存至 ElasticSearch。

表 4.5: AnalysisResult 类详情列表

字段	含义	类型	说明
id	分析结果编号	long	主键，不为空
taskId	分析任务 ID	String	对应的分析任务 ID
datasetId	测试用例集 ID	long	对应的测试用例集 ID
esId	ES 文档 ID	String	对应的 ElasticSearch 文档 ID
result	任务结果	Map<String,Object>	用例分析结果，包括向量化信息，分词结果等

4.1.3 关键代码

```
public DatasetVO readExcel(String name, long ownerId, MultipartFile file, long tagId) throws IOException {
        DatasetVO datasetVO = null;
        //判断文件类型
        boolean type = file.getOriginalFilename().split("\\.")[1].equals("xls");
        datasetVO = readAndSave(name, ownerId, tagId, file, type);
        return datasetVO;
    }
public DatasetVO readAndSave(String name, long ownerId, long tagId, MultipartFile file, boolean isXls) throws IOException {
        InputStream inputStream = file.getInputStream();
        Workbook workbook = null;
        if(isXls) {
            workbook = new HSSFWorkbook(inputStream);
        } else {
            workbook = new XSSFWorkbook(inputStream);
        }
        //选第一张表
        Sheet sheet = workbook.getSheetAt(0);
        Dataset dataset = new Dataset();
        ...
        Dataset result = datasetRepository.save(dataset);
        ...
        datasetRepository.save(dataset);
        ...
    }
```

图 4.5: 读取上传文件代码

如图4.5所示是读取上传文件的代码，首先通过调用文件对象的 getOriginal-Filename() 方法获得文件名称，根据文件名称取得文件后缀名，判断文件类型，依据不同类型传递不同参数值执行 readAndSave() 方法，得到 DatasetVO 类对象作为返回值。在 readAndSave() 方法中，先根据文件对象的 InputStream 和文件类型构造 Workbook 对象，然后读取 Excel 的第一张 Sheet 表，将内容封装到 Dataset 类对象，最后通过 DatasetRepository 存储 Dataset。

上传测试用例后，系统需要对测试用例的文本内容进行处理和分析。本文所指测试用例都是自然语言形式的文本，所以需要使用 NLP 技术对文本进行文本分词和关键词提取操作。

如图4.6所示是关键词提取代码。首先读取停用词文件 stop_words.txt，生成停用词表，停用词表包含了特殊字符、标点符号、助词、介词、连词和副词等多种类型的常见中文停用词类型。去除停用词的目的是减小词序列长度，降低无用信息的干扰，提高关键词提取效率。建立数据库连接后，开始从数据库读取测

试用例的完整内容，使用 jieba 精确模式进行分词，并根据词性进行筛选，提取特定类型的词语作为测试用例的对应关键词。最后保存分词结果和关键词提取结果到对应文件。

```
def word_split(attribute_name, output_path):
# 读取停用词
stop_words = get_stop_words(os.path.dirname(__file__) + '/data/stop_words.txt')
test_case_data = mysql_connect.get_test_cases_from_mysql()
...
for line in test_case_data:
    requirement = line[attribute_name]
    # 精确模式分词
    jieba_list = jieba.cut(requirement, cut_all=False)
    pseg_words = pseg.cut(requirement)
    ...
    for word in jieba_list:
        if word not in stop_words:
            tmp.append(word)
    for word in pseg_words:  # 词性识别，提取实体词
        if word.flag in ["n", "ns", "nt", "nz"] and word.word not in stop_words:
            labels.append(word.word)
...
# 保存分词与关键词提取结果
with open(requirement_split_path, 'w', encoding='utf-8') as f:
    for line in split_result:
        ...
    f.close()
...
return split_result
```

图 4.6: 关键词提取代码

测试用例的向量化是测试用例分析的关键步骤，用例分析结果不仅会用于用例检索，还将用作深度语义匹配模型的输入数据。如图4.7所示是测试用例向量化的关键代码。首先通过连接 MySQL 数据库，读取库中测试用例内容，然后初始化 Tokenizer 并加载 transformer，对于每一段测试需求描述文本和测试步骤文本，均先校验其内容是否可用，然后对其进行 token 编码，编码后的结果转为 numpy 数组，再经过 transformer 模型的 predict() 方法生成文本向量，所得 numpy 向量转为 list 类型后存入 dict，最终存储至数据库。

```
def text2vector():
    test_case_data = mysql_connect.get_test_cases_from_mysql()   #连接数据库
    data = []
    vector_name = 'cls'
    tokenizer = Tokenizer(dict_path, do_lower_case=True)
    model = build_transformer_model(config_path, checkpoint_path)   #加载向量模型
    output = []
    output_req = []
    for line in test_case_data:
        dict = line
        # 校验测试用例的测试需求描述是否可用
        if is_valid_test_requirement(line):
          ...
          token_ids_req, segment_ids_req = tokenizer.encode(line['test_requirement']) #向量编码
            if vector_name == 'cls':
                cls_vector_req=model.predict([np.array([token_ids_req]),
                                np.array([segment_ids_req])])[0][0]
                output_req.append(cls_vector_req)
                # nparray 转 list
                dict['requirement_vector'] = cls_vector_req.tolist()
        # 校验测试用例的测试步骤是否可用
        if is_valid_test_process(line):
            ...
            token_ids, segment_ids = tokenizer.encode(line['test_process'])   #向量编码
            if vector_name == 'cls':
                cls_vector = model.predict([np.array([token_ids]), np.array([segment_ids])])[0][0]
                output.append(cls_vector)
                # nparray 转 list
                dict['process_vector'] = cls_vector.tolist()
        data.append(dict)
    db_connect. save_data(data)
```

图 4.7: 测试用例向量化代码

4.2 测试任务管理模块的实现

4.2.1 测试任务管理流程

如图4.8所示是测试任务管理时序图。用户创建测试任务的 HTTP 请求被分发到 TaskController，系统解析请求的各个字段，包括任务名称 name，测试类型 type，任务创建者 creator_id，测试功能点描述 test_requirement 和任务描述 messsage 等。将各个字段封装成 Task 对象后，再存入 MySQL 数据库。创建任务成功后，前端发起查看任务的请求，路由将其定位到 TaskController 的 getById() 方法。对于任务中一条具体的测试功能点描述，用户可点击查看其测试用例匹配情况，系统调用 TestCaseService 的 testCaseGeneration() 方法，计算与该条测试

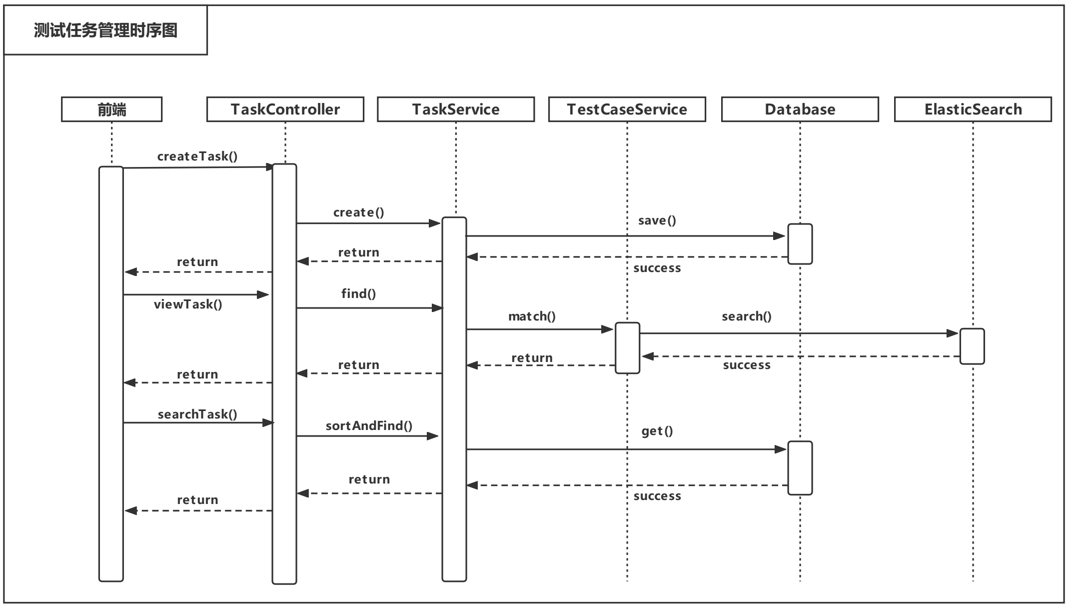

图 4.8: 测试任务管理时序图

功能点描述相匹配的历史测试用例，得到结果后返回。用户还可以在测试任务管理页面直接发起排序请求和关键词搜索请求来筛选测试任务，TaskService 中的 sortAndFind() 方法可对应执行筛选条件并返回检索结果。

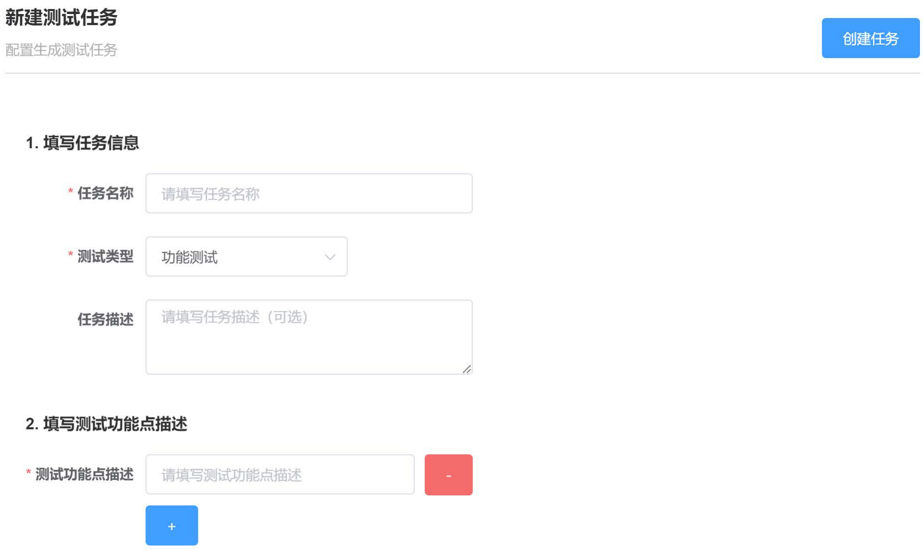

图 4.9: 创建测试任务截图

如图4.9所示是创建测试任务页面截图。用户需要填写任务名称，任务描述，测试选择测试类型，然后完整填写测试功能点描述，最后点击创建任务按钮来

新建一个测试任务。其中任务描述为选填项，其他均为必填项，测试类型和测试功能点描述是一个测试任务的关键信息，在后面的测试用例匹配中会用来作为重要的匹配项，影响系统的用例推荐结果。

图 4.10: 测试任务管理列表截图

图 4.11: 查看测试任务详情截图

测试任务创建完成后，用户可以在系统中管理所创建的测试任务，管理列表如图4.10所示。用户可以在搜索框中输入关键字，对测试任务进行搜索，也可以根据测试类型、用例数量、创建时间对测试任务进行排序筛选。

用户不仅可以管理测试任务，还可以查看测试任务详情。如图4.11所示是查看测试任务详情页面截图。以测试任务“微博系统测试”为例，点击“查看详情”即可进入任务详情页，可以看到详情页展示的任务名称，测试类型为“功能测试”，任务描述为“微博系统搜索框测试”，测试需求描述有三条，分别为“测试系统搜索功能的特殊字符判别”，“测试系统搜索功能的历史搜索展示”和“测试系统分页功能的搜索页面展示”，可以点击下方的“点击查看”按钮来查看系统为这些测试需求所推荐的可复用测试用例。

4.2.2 关键代码

```
public @ResponseBody Response uploadIfExcel() throws Exception {
        TaskVO taskVO = new TaskVO();
        taskVO.setName(name);
        ...
        if (file.isPresent()) {
            InputStream inputStream = file.get().getInputStream();
            Workbook workbook = null;
            if (file.get().getOriginalFilename().split("\\.")[1].equals("xls")) {
                workbook = new HSSFWorkbook(inputStream);
            } else if (file.get().getOriginalFilename().split("\\.")[1].equals("xlsx")) {
                workbook = new XSSFWorkbook(inputStream);
            }
            Sheet sheet = workbook.getSheetAt(0);
            String testRequirementTest = "";
            for (Row row : sheet) {
                if (row.getRowNum() == 0) {
                    continue;
                }
                testRequirementTest = ...
            }
            taskVO.setTestRequirement(testRequirementTest);
        } else {
            taskVO.setTestRequirement(testRequirement);
        }
        TaskPartVO taskPartVO = taskService.saveTask(taskVO);
        return new Response(ResponseCode.OPERATION_SUCCESS, taskPartVO);
}
```

图 4.12: 创建测试任务代码

如图4.12所示是创建测试任务的关键代码。首先创建 TaskVO 对象将解析所得字段都设置成相应的对象属性值，然后判断 file 对象是否为空来确定用户是否在创建时上传了 Excel 文件。若用户是通过上传文档的形式来创建任务，则构

造 Workbook 对象，遍历 sheet 表格读取所有内容，否则直接设置 TaskVO 对象的 testRequirement 属性值。最后调用 TaskService 的 saveTask() 方法，TaskVO 对象作为参数，保存任务至数据库，并返回成功码 OPERATON_SUCCESS。

4.3 测试用例匹配模块的实现

4.3.1 测试用例匹配流程

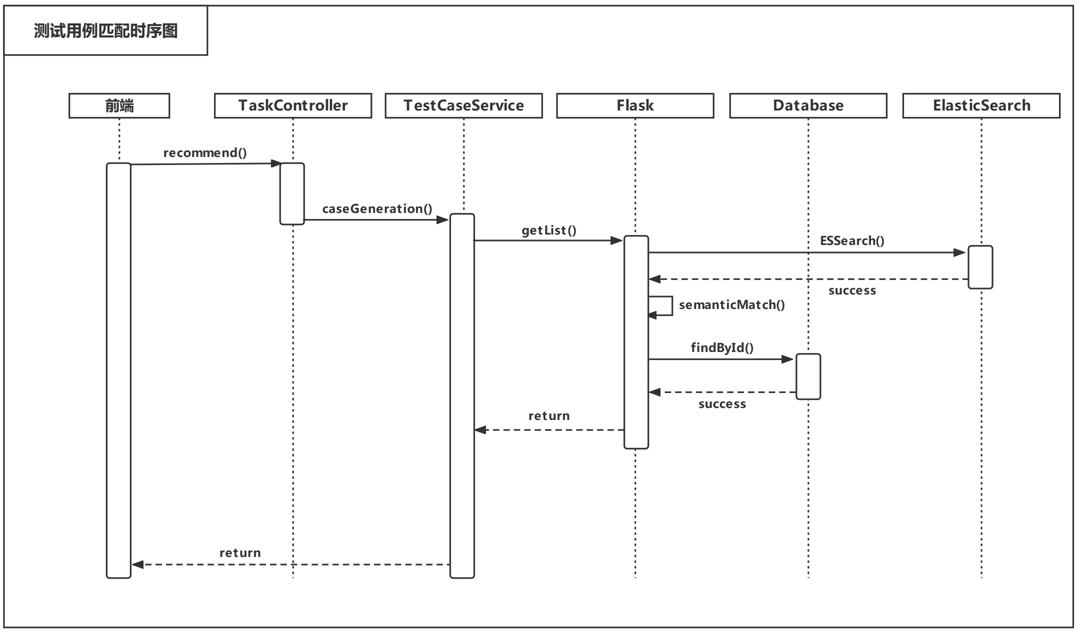

图 4.13: 测试用例匹配时序图

如图4.13所示是测试用例匹配时序图。前端的测试用例匹配 HTTP 请求映射到 TaskController 的 taskRun() 方法,在该方法中又调用 TestCaseService 的 caseGenaration() 方法，caseGeneration() 中通过 RestTemplate 发起 RestAPI 服务调用。运行中的 Flask 服务接收到测试用例匹配的请求后，首先通过 ElasticSearch 进行初步检索，其默认的相似度查询算法为 BM25 算法。通过以测试功能点描述为查询序列，ElasticSearch 的相关性分数 _score 为比较标准，可检索得到若干测试用例，称之为召回测试用例集。然后再对召回测试用例的文本内容和查询序列进行深度语义匹配计算 semanticMatch()，匹配分值达到阈值的测试用例被选中进入候选测试用例集。最后根据本文设计的综合推荐分数公式，对每一个候选测试用例计算其综合推荐分数，计算完成后，取得分最高的若干个测试用例，推荐为该测试功能点描述的可复用测试用例。然后访问 MySQL 数据库，根据测试用例 ID 进行查询，将匹配成功的测试用例内容返回上游调用者，最终到达前端页

面。

4.3.2 关键代码

```
def construct(self):
    a, b = inputs()
    # 输入接 embedding+BiLSTM
    embedded_1 = embedding(a)
    embedded_2 = embedding(b)
    lstm_embedding = keras.layers.Bidirectional(keras.layers.LSTM(
        self._params['lstm_units'],
        return_sequences=True,
        dropout=self._params['dropout_rate']
    ))
    #编码得到新的 embedding
    co1 = lstm_embedding(embedded_1)
    co2 = lstm_embedding(embedded_2)
    soft_1, soft_2 = SoftAttention()([co1, co2])
    ...
    #相似性加权拼接
    t_1 = keras.layers.concatenate([co1, soft_1, sub_soft_1, mul_soft_1])
    t_2 = keras.layers.concatenate([co2, soft_2, sub_soft_2, mul_soft_2])
    #用 BiLSTM 提取上下文信息
    bi_1 = keras.layers.Bidirectional(keras.layers.LSTM(
        self._params['lstm_units'],
        dropout=self._params['dropout_rate']))(t_1)
    bi_2 = ...
    #池化操作
    max_pool_1 = keras.layers.GlobalMaxPooling1D()(bi_1)
    max_pool_2 = keras.layers.GlobalMaxPooling1D()(bi_2)
    pooled = keras.layers.concatenate([max_pool_1, max_pool_2])
    pooled = keras.layers.Dropout(rate=self._params['dropout_rate'])(pooled)
    esim = Model(inputs=[a, b], outputs=output)
    return esim
```

图 4.14: ESIM 模型构建代码

如图4.14所示是深度语义匹配模型 ESIM 构建代码。首先对于输入的两个句子接 embedding 和 BiLSTM，得到所需要的 embedding，使用 BiLSTM 的原因是它可以学习到 word 与上下文的关系，可以认为是在获得 word embedding 之后，在当前语境的再一次重新编码。经过对齐之后，按 ESIM 模型结构将向量进行多次计算和拼接。拼接之后再次使用 BiLSTM 提取上下文信息，进行推理信息融合。最后进行池化操作，经过分类器输出预测结果。

经过深度语义匹配模型计算后，分数达到阈值的测试用例将进入候选测试用例集，系统将对候选测试用例进行综合排序。如图4.15所示是测试用例综合排

序代码。对于得到的候选测试用例集，需要进行最后的综合排序分数计算来确定推荐的可复用测试用例是哪些用例。首先需要获取可变配置参数 alpha 和 beta，它们是综合得分计算公式中的两个控制因子。然后对于每个候选测试用例，获取它的搜索引擎得分，语义匹配得分，累积推荐频次和累积复用频次，然后根据设计好的公式计算综合得分，存储至哈希表中。再调用排序方法获取 topN 的推荐结果，封装成 CaseInfo 并返回结果，同时将推荐结果，用户 ID 和任务 ID 等信息缓存至 Redis，加速该任务的测试用例匹配结果的查询速度。

```
public List<CaseInfo> caseSort(List<String> caseIds, String input, String userId) {
        JSONObject obj = JSONObject.fromObject(input);
        String content = obj.getString("content");
        //获取可变配置参数
        double alpha = CommonConstant.ALPHA;
        double beta = CommonConstant.BETA;
        Map<String, Double> scoreMap = new HashMap<>();
        List<CaseInfo> result = new ArrayList<>();
        try {
            for (String id : caseIds) {
                double esScore = testCaseService.getESScore(id, content);
                double esimScore = testCaseService.getESIMScore(id, content);
                int recommend = testCaseService.getRecommendTime(id);
                int reuse = testCaseService.getReuseTime(id);
                //综合推荐分数计算
                double sum = esimScore * alpha + esimScore * beta -
                        Math.log((double) (recommend + 1) / (recommend + 1));
                scoreMap.put(id, sum);
            }
            //排序，获取 topN
            recommend = getTopNByScore(scoreMap);
            //缓存推荐结果
            saveResultToRedis(recommend, userId, obj.getString("taskId"));
        } catch (Exception e) {
            e.printStackTrace(); //打印堆栈日志
            LogUtil.info("caseSort error: " + e.getMessage());
        }
        return result;
    }
```

图 4.15: 测试用例综合排序代码

4.4 本章小结

本章通过流程图、顺序图、类列表、核心类详情和关键代码结合的方式对系统的测试用例分析模块、测试任务管理模块和测试用例匹配模块的实现进行详细地阐述。测试用例分析模块分为文件上传部分和测试用例分析核心部分，分

别通过流程图、时序图、核心类列表和用例分析代码进行阐述。测试任务管理模块主要展示了测试任务管理页面和测试任务详情页面。测试用例匹配模块主要介绍了测试用例匹配流程和 ESIM 模型实现代码。

第五章　基于深度语义匹配的测试用例复用系统的测试与实验分析

5.1 测试准备

5.1.1 测试目标

本章是对系统的测试部分和实验分析部分。系统的测试包括功能测试、性能测试、可用性测试和效果测试。功能测试主要根据系统的功能性需求对系统所应具备的功能进行检验，验证是否满足功能性需求。性能测试针对系统的性能进行全方位测试，尤其是并发情况下的测试执行结果。可用性测试是为了验证系统在宕机后的恢复情况与恢复耗时是否达标。效果测试是对系统所提供的核心服务的有效性进行检验，主要是用户在使用测试用例复用系统后，设计测试用例的效率是否提高。

5.1.2 测试环境

系统测试环境配置如表5.1所示。系统的前后端项目和测试用例匹配服务均部署在阿里云 ECS 服务器。服务器安装 Ubuntu 操作系统，CPU 为 4 核，内存大小 16G，使用 MySQL 数据库存储系统基础数据，使用 Elasticsearch 存储测试用例分析数据，使用 Redis 缓存机制存储测试用例匹配结果。

表 5.1: 系统测试环境

参数	**详情**
服务器	阿里云 ECS 服务器
操作系统	Ubuntu18.04
内存	16G
CPU	4 核
数据库	MySQL8.0.21, Elasticsearch7.8.0, Redis5.0.3
基础环境	Java8, Python3.8
负载均衡	Nginx1.15.3

5.2 功能测试

表5.2描述了下载数据集模板的测试用例，在测试用例数据集模板下载页面，用户可将模板文件下载至本地文件系统的指定路径。

表 5.2: 下载数据集模板测试用例

ID	**TC1**
测试名称	下载数据集模板
待测功能	用户能在模板下载处将数据集模板下载至本地
测试步骤	1. 用户成功登录系统 2. 用户进入下载数据集模板页面 3. 用户点击下载链接，选择本地文件保存路径 4. 用户点击确定按钮，下载文件至本地
预期结果	1. 用户登录成功 2. 页面展示下载链接 3. 系统弹出文件路径选择框 4. 数据集模板文件下载至指定路径
实际结果	符合预期

表5.3描述了上传测试用例集的测试用例，用户选择符合格式规范要求的本地测试用例集文件，填写测试用例集名称后可将其上传至系统。

表 5.3: 上传测试用例集测试用例

ID	**TC2**
测试名称	上传测试用例集
待测功能	用户在上传测试用例页面，将本地测试用例集上传至系统中
测试步骤	1. 用户进入上传测试用例页面 2. 用户填写测试用例集名称信息 3. 用户点击按钮选择本地文件 4. 用户确定上传文件
预期结果	1. 系统展示上传测试用例页面 2. 系统校验测试用例集名称 3. 系统弹出文件路径选择框 4. 测试用例集文件成功上传至系统
实际结果	符合预期

表5.4描述了查看测试用例集列表的测试用例，用户在测试用例集管理页面能够查看测试用例集列表，并能根据测试用例的创建时间、测试用例数量进行排序，输入关键词搜索测试用例集，查看测试用例集详情。

表 5.4: 查看测试用例集列表测试用例

ID	TC3
测试名称	查看测试用例集列表
待测功能	用户在查看测试用例集列表，并根据条件进行排序和检索
测试步骤	1. 用户进入管理测试用例集页面 2. 用户点击创建时间进行排序 3. 用户点击测试用例数量进行排序 4. 用户在搜索框输入关键词“模板”进行检索 5. 用户点击查看详情按钮
预期结果	1. 系统展示测试用例集列表 2. 测试用例集列表按照创建时间进行排序 3. 测试数据集列表按照测试用例数量进行排序 4. 系统显示名称中含有关键词的测试用例集 5. 系统跳转到测试用例集详情页
实际结果	符合预期

表5.5描述了查看测试用例详情的测试用例，用户在测试用例列表页面能够查看测试用例，并能根据测试用例的创建时间进行排序，根据测试方法、测试类型等进行筛选，输入关键词搜索测试用例，查看测试用例详情。

表 5.5: 查看测试用例详情测试用例

ID	TC4
测试名称	查看测试用例详情
待测功能	用户在测试用例集中查看一个测试用例的详细信息
测试步骤	1. 用户进入测试用例列表页面 2. 用户点击创建时间进行排序 3. 用户点击测试方法进行筛选 4. 用户点击测试类型进行筛选 5. 用户在搜索框输入关键词“测试用例 A” 6. 用户点击查看详情按钮
预期结果	1. 系统展示测试用例列表 2. 测试用例列表按照创建时间进行排序 3. 测试用例根据指定测试方法被筛选出来 4. 测试用例根据指定测试类型被筛选出来 5. 系统显示名称中含有指定关键词的测试用例 6. 系统跳转到测试用例详情页，显示测试用例详情
实际结果	符合预期

表5.6描述了创建测试任务的测试用例，用户可以在创建测试任务页面填写任务名称、任务描述和测试功能点描述，选择测试类型，创建测试任务。

表 5.6: 创建测试任务测试用例

ID	TC5
测试名称	创建测试任务
待测功能	用户在任务创建页面填写相关信息，创建测试任务
测试步骤	1. 用户进入创建测试任务页面 2. 用户填写任务名称 3. 用户点击下拉框选择测试类型 4. 用户填写任务描述信息 5. 用户填写测试功能点描述 6. 用户点击创建任务按钮
预期结果	1. 系统展示创建测试任务页面 2. 系统校验测试任务名称，不能为空 3. 系统校验测试类型，不能为空 4. 系统校验任务描述，可以为空 5. 系统校验测试功能点描述，至少需填写一条 6. 系统创建测试任务成功
实际结果	符合预期

表5.7描述了查看测试任务的测试用例，用户在测试任务管理页面查看测试任务详情，可查看每条测试功能点描述的历史测试用例匹配结果，并可将结果复制至剪切板。

表 5.7: 查看测试任务测试用例

ID	TC6
测试名称	查看测试任务
待测功能	用户在测试任务管理页面查看测试任务，查看测试用例匹配结果
测试步骤	1. 用户进入测试任务管理页面 2. 用户点击一个测试任务 3. 用户点击一条测试功能点描述 4. 用户点击查看结果按钮 5. 用户点击展开一条历史测试用例 6. 用户选择复用该测试用例，将其复制至剪切板
预期结果	1. 系统展示测试任务管理页面 2. 系统跳转测试任务详情页面 3. 系统显示为该测试功能点描述匹配的测试用例 4. 系统显示所有历史测试用例匹配结果 5. 系统显示测试用例详细信息 6. 系统将该测试用例内容复制至剪切板
实际结果	符合预期

表5.8描述了测试任务检索的测试用例，用户可以在测试任务管理页面根据

创建时间等条件对测试任务进行排序，根据测试类型等进行筛选，在搜索框输入关键词搜索，并可以查看一个测试任务的详情。

表 5.8: 测试任务检索测试用例

ID	TC7
测试名称	测试任务检索
待测功能	用户在测试任务管理页面根据条件对测试任务排序，输入关键词检索测试任务
测试步骤	1. 用户进入测试任务管理页面 2. 用户点击创建时间对测试任务进行排序 3. 用户选择测试类型筛选测试任务 4. 用户在搜索框输入关键词搜索测试任务 5. 用户选择一个测试任务，点击查看详情
预期结果	1. 系统展示测试任务管理页面 2. 系统按照创建时间排序测试任务 3. 系统显示属于被选中测试类型的测试任务 4. 系统显示包含关键词的测试任务 5. 系统跳转测试任务详情页
实际结果	符合预期

5.3 性能测试

5.3.1 测试设计

本文性能测试主要是针对系统接口功能的并发测试，验证系统接口在并发访问情况下的吞吐量，判断系统性能是否符合使用要求。如表5.10所示是性能测试接口列表。

表 5.9: 性能测试接口列表

接口编号	接口描述	接口路径
I1	下载数据集模板	/dataset/download
I2	上传测试用例集	/dataset/upload
I3	查看测试用例列表	/testcase/get
I4	查看测试用例详情	/testcase/detail
I5	创建测试任务	/task/create
I6	查看测试任务	/task/get
I7	查看测试匹配结果	/task/match
I8	检索测试任务	/task/search

本文使用测试工具 JMeter 对各个接口进行测试，以接口“创建测试任务”为例，其 JMeter 配置如下：JMeter 线程组配置如图5.1所示，使用 JMeter 创建线程

组，参数 Number of Threads 设置为 400，Ramp-up Period 为设置为 5，Loop Count 为设置为 5，这些参数表示模拟了 400 个用户同时访问接口，一个用户在 5 秒内发送 5 次请求；还需要线程组中创建针对接口的 HTTP 请求配置信息，配置字段有协议名称、IP 地址或主机域名、端口号、访问方法类型、访问路径和请求参数等，配置情况如图5.2所示。

图 5.1: JMeter 线程组配置截图

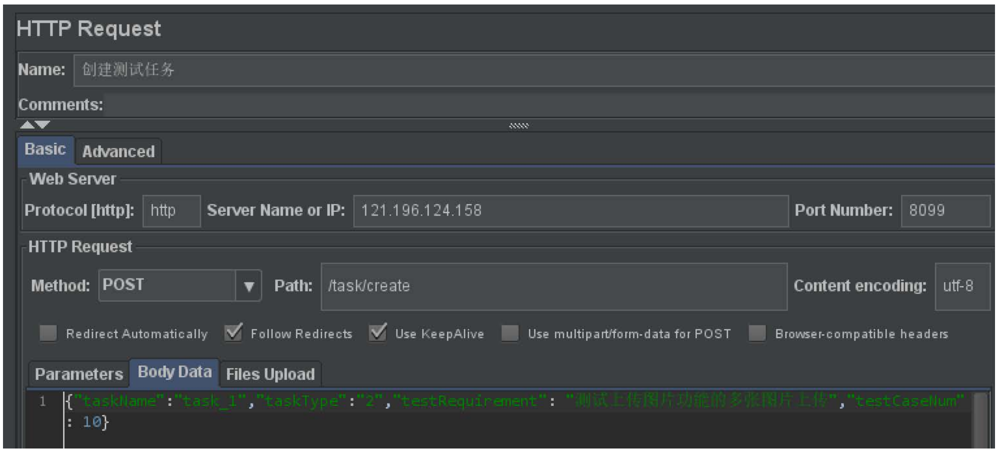

图 5.2: HTTP 请求配置截图

5.3.2 测试结果

系统的性能测试结果如表5.10所示。以接口 I6 “创建测试任务” 为例，这个接口是系统的核心接口之一，使用频率较高，响应时间要求也较高。从测试结果可以看到，400 个线程模拟 400 个用户在 5 秒内各发送了 5 次访问请求，总计 2000 次请求，经统计可知，错误请求数为 0，平均响应时间为 137ms，95% 的请求都在 156ms 内得到响应。

总的来看，在并发访问的情况下，各个接口的平均响应时间均较为理想，且与 95% 响应时间的差距不大，各接口的错误请求数量为 0，说明系统的性能表

表 5.10: 性能测试结果

接口编号	错误请求数量	平均响应时间	95% 响应时间
I1	0	112ms	143ms
I2	0	127ms	155ms
I3	0	106ms	130ms
I4	0	108ms	143ms
I5	0	141ms	169ms
I6	0	137ms	156ms
I7	0	209ms	238ms
I8	0	133ms	151ms

现比较平稳。综上所述，本系统能在并发环境下平稳运行，具备较高性能。

5.4 可用性测试

本节将针对系统的可用性进行测试。因为本系统主要作为工具型软件被软件测试人员使用，所以系统的可用性要求并不非常高。本文对系统的可用性要求是系统宕机恢复时间在 3 分钟以内，并对系统进行可用性测试，以验证系统的可用性符合基本要求。

5.4.1 测试设计

图 5.3: 可用性测试记录

系统的可用性测试的测试方式是在部署成功的服务器上，手动制造服务下线的情况，然后观察系统服务的恢复时间，对系统的观测方式是持续调用系统

接口并记录系统接口的恢复时间点，与系统的下线时间点作比较，得到系统的宕机恢复时长，判断其是否符合设计要求。

5.4.2 测试结果

如图5.3所示是系统的可用性测试记录。在 15:10:30 左右，人为关闭了服务进程，监控服务监测到这一情况后，于 15:10:55 开始重启服务。通过接口响应时间记录可知，在 15:12:00 左右系统已恢复服务，系统不可用的时间不超过 3 分钟，表明系统的可用性符合要求。

5.5 效果测试

5.5.1 文本匹配效果测试

为验证本文提出的基于深度语义匹配的测试用例复用方法的有效性，我们将通过使用文本匹配的一些基本方法与之对比。本文选用的基本方法包括 BM25 算法、VSM 方法和 Doc2Vec 模型。

BM25 算法，是一个由英国的信息检索方向的计算机科学家提出的算法。BM25 算法是搜索算法中的一种权重度量方法，用于计算文档与查询关键词之间的相关性。BM25 算法在计算权重时引入了文档长度与平均文档长度的比率，与 TF-IDF 不同，除了引入长度比率外，BM25 可以控制过高的词频对排序的影响。

VSM，即向量空间模型，通过 TF-IDF 统计词频权重，将文档映射成 n 维特征空间中的一个特征向量，然后通过向量的距离计算来计算向量的相似度。

Doc2Vec 模型，通过在 Word2Vec 模型中增加文档向量维度来将文档映射成一个高维向量，通过计算欧氏距离衡量文档之间的相似度。本文使用 python 中 gensim 库的 doc2vec 工具包来进行实验测试。

本文在 STS18、STS19 和 SICK-R 三个数据集上进行了文本匹配任务的实验。其中 STS18 和 STS19 是 SemEval 评测比赛在 2018 年和 2019 年的赛事数据集，SICK-R 是 SICK 数据集中的子任务，目标是推断两个句子之间的语义相关性。以上数据集的格式均为两段短文本和一个人工标注的介于 0 和 5 之间的分数，该分数表示两段短文本之间的语义匹配程度，5 表示两段文本表达同一个语义，0 表示两段文本的语义完全不相关。

为评价实验结果，本文选用斯皮尔曼相关系数（Spearman’s rank correlation coefficient）作为评价指标。它用于衡量两组数值的相关性，范围在-1 和 1 之间，当两组值完全正相关时值为 1，在这里用于衡量文本匹配方法的预测匹配度和人

工标注的语义相关程度之间的相关性，相关性越高说明文本匹配方法的效果越好。

表 5.11: 文本匹配效果测试结果

匹配方法	Sp-STS18	Sp-STS19	Sp-SICK-R	每百条数据执行时间（单位：s）
ESIM	0.69	0.66	0.61	38.437
BM25	0.51	0.49	0.46	31.892
VSM	0.53	0.50	0.45	40.018
Doc2Vec	0.61	0.63	0.54	43.256

文本匹配效果测试结果如表5.11所示，Sp 表示斯皮尔曼相关系数。从实验结果可知，本文所使用的基于 ESIM 模型的文本匹配方法在 3 个数据集上的文本匹配效果优于其他对比方法，其斯皮尔曼系数均为最高，说明其文本匹配效果与人工标注的结果最为接近。匹配速度方面，除 BM25 方法的文本匹配执行速度较为领先之外，其他几种方法均无明显差别。

5.5.2 测试用例复用效果测试

为验证本系统的实际使用效果，本文使用对比实验的方式来进行效果测试。实验的执行过程如下：

1. 选择多个软件及其对应的若干测试需求共 30 个，将这 30 个测试需求随机分为 3 组，每组 10 个，得到 3 组含测试需求的测试任务，称为任务 1、任务 2 和任务 3。

2. 将参与实验的 12 名测试人员随机且平均分为 2 组，称为 A 组和 B 组。

3. 首先，让 A 组和 B 组的测试人员均在不使用本系统的条件下，每个人完成任务 1 的测试用例设计工作。然后，在相同情况下，让 A 组人员使用本系统和 B 组人员不使用本系统，每个人完成任务 2 的测试用例设计工作。最后，让 A 组人员和 B 组人员均使用本系统，每个人完成任务 3 的测试用例设计工作。

4. 在上述实验过程中，记录每个测试人员完成一组任务所用的时长。任务完成后，参与实验的 5 名评审专家根据为测试人员所设计的每个测试用例进行质量评分，最终分组计算其平均得分。

测试用例复用效果测试结果如表5.12和表5.13所示。12 名测试人员和 5 名评审专家参与了本次实验，每名测试人员均完成了 3 组测试任务，设计了相应的

表 5.12: 用例复用效果 A 组测试结果

人员编号	任务 1 时长	任务 2 时长	任务 3 时长
P1	38.6 分钟	27.7 分钟	34.1 分钟
P2	42.3 分钟	30.2 分钟	31.5 分钟
P3	43.9 分钟	36.8 分钟	36.9 分钟
P4	37.5 分钟	31.7 分钟	30.6 分钟
P5	46.3 分钟	33.4 分钟	36.1 分钟
P6	41.6 分钟	28.8 分钟	33.5 分钟
平均时长	41.7 分钟	31.4 分钟	33.8 分钟

表 5.13: 用例复用效果 B 组测试结果

人员编号	任务 1 时长	任务 2 时长	任务 3 时长
P7	44.3 分钟	38.8 分钟	33.9 分钟
P8	45.9 分钟	42.0 分钟	31.3 分钟
P9	37.1 分钟	35.4 分钟	28.7 分钟
P10	39.8 分钟	34.6 分钟	26.5 分钟
P11	35.2 分钟	33.1 分钟	32.3 分钟
P12	39.0 分钟	38.5 分钟	36.5 分钟
平均时长	40.2 分钟	37.1 分钟	31.5 分钟

测试用例并记录了每个任务所用的时长，5 名评审专家对这些测试用例进行了评分。从表中数据可以看出，对于同一组任务，使用了本系统的测试人员的完成速度高于未使用本系统的测试人员；对于同一组人员，在参与不同组任务时，使用本系统时的完成速度也会高于未使用本系统时的完成速度。

表 5.14: 测试用例设计评审得分列表

评审专家编号	任务 1-A 组	任务 1-B 组	任务 2-A 组	任务 2-B 组	任务 3-A 组	任务 3-B 组
E1	7.1	7.9	8.2	7.7	8.3	8.0
E2	7.4	7.7	7.7	7.3	7.5	7.8
E3	6.7	6.6	7.2	6.7	7.4	7.3
E4	6.5	6.8	7.0	6.3	6.6	7.1
E5	8.1	8.0	8.5	7.6	8.2	8.1
平均得分	7.2	7.4	7.7	7.1	7.6	7.7

此外，通过表5.14可以注意到，A、B 两组测试人员在均使用和均不使用本系统的情况下，所设计的测试用例的平均评审得分差别不大，只有在完成任务 2 时，在 A 组人员使用了本系统且 B 组人员未使用本系统的情况下，A 组人员的得分相对高于 B 组人员的得分，在一定程度上说明系统所推荐的历史测试用例

对测试人员设计的测试用例质量有正向的影响，用例复用有一定的效果。

总结上述，基于深度语义匹配的测试用例复用系统在一定程度上能辅助测试人员设计测试用例，挖掘历史测试用例价值，提高测试人员测试效率，为测试人员的测试工作做出了贡献。

5.6 本章小结

本章对基于深度语义匹配的测试用例复用系统进行了测试。设计了功能测试的测试用例，验证了系统的功能符合设计需求，性能测试执行结果证明了系统的性能符合使用要求，可用性测试证明了系统的宕机恢复时间达标，效果测试的测试结果证明了本文提出的测试用例匹配方法的有效性，也证明了测试用例复用功能能帮助测试人员调高工作效率，具有实用价值。

第六章　总结与展望

6.1　总结

测试用例是软件测试的关键要素，既验证了软件功能的完整性，又能发现软件缺陷。设计良好的测试用例使得软件测试过程能有序进行，确保测试有足够的覆盖率，又有一定的侧重突出性，对重点模块集中测试。高质量的测试用例还有一定的可重复性，在版本迭代过程中，复用一定比例的测试用例能节省测试人员工作量，提高软件测试的测试效率，缩短测试周期。设计测试用例是软件测试的关键步骤，也是决定测试用例质量的核心环节。测试用例质量影响软件测试的执行质量，也间接影响软件质量。在设计测试用例阶段复用历史测试用例，能充分利用历史测试用例的价值，辅助设计新测试用例，提高测试人员工作效率。

本文首先介绍了系统的项目背景和意义，介绍了测试用例复用和深度文本匹配的国内外研究现状。本文还对系统用到的相关技术进行了介绍，主要有用到的 NLP 技术，文本匹配技术和一些工程技术。

本文对系统进行了需求分析和用例描述，需求分析包括功能需求和非功能需求，通过系统用例图和用例描述表对各个用例进行了详细描述。系统的架构设计通过架构设计图和 4+1 视图的形式展示，并将系统划分为三个模块，分别为测试用例分析模块、测试任务管理模块和测试用例匹配模块。测试用例分析部分，系统提供测试用例数据集模板下载，帮助用户上传符合系统要求的测试用例。在用户上传历史测试用例之后，系统使用自然语言处理技术对用例进行内容分析，包括分词、向量化等操作，并将分析结果保存至 Elasticsearch。测试任务管理部分，用户应先创建测试任务，填写任务信息，输入测试功能点描述，选择测试类型。用户可以在测试任务管理列表页面管理自己所创建的测试任务，根据条件进行排序，在搜索框输入关键词进行任务检索。测试用例匹配部分，用户创建任务后，系统将使用基于深度语义匹配的测试用例复用方法，为每一条测试功能点描述推荐与之匹配度最高的若干历史测试用例，匹配过程结束后，用户可以查看任务详情和测试用例匹配结果，可以浏览、收藏系统推荐的历史测试用例，可以直接将一个历史测试用例的内容复制至剪贴板，复用该测试用例。最后，系统的开发工作已完成并通过了各项测试，测试结果表明系统的功能完整，各项性能符合设计要求。

6.2 展望

本系统已线上部署并投入使用，但仍存在不足之处，有的地方尚待进一步改进，主要是以下几个方面：

（1）测试用例复用方式单一。系统目前只支持用户创建测试任务，输入测试功能点描述的方式来获取推荐的历史测试用例。用户对测试用例的复用方式仅限于复制测试用例内容，方式较为单一，未来可以考虑扩展用户输入的方式和复用测试用例的途径，提供多种格式的用例匹配方式，例如提供在线编辑等复用模式，提升用户体验。

（2）测试用例推荐机制有待改进。系统记录了用户的检索行为、匹配和复用记录，但缺乏进一步的行为分析，推荐机制有待优化。后续可以对用户的行为记录做深层次的分析，并将其加入系统的推荐流程中，提高系统的推荐准确率和用户的使用效率。

参考文献

[1] ELLIMS M, BRIDGES J, INCE D C. The Economics of Unit Testing[J]. Empir. Softw. Eng., 2006, 11(1) : 5 – 31.

[2] 郑人杰. 计算机软件测试技术 [M]. [S.l.] : 清华大学出版社, 1992.

[3] HETTIARACHCHI C, DO H, CHOI B. Effective Regression Testing Using Requirements and Risks[C] // Eighth International Conference on Software Security and Reliability, SERE 2014, San Francisco, California, USA, June 30 - July 2, 2014. [S.l.] : IEEE, 2014 : 157 – 166.

[4] POONAWALA M H. Reusable strategies for testing safety-critical systems[M]. [S.l.] : University of Minnesota, 1999.

[5] FREEMAN P. Reusable software engineering: Concepts and research directions[C] // . 1983.

[6] CAI L, TONG W, LIU Z, et al. Test case reuse based on ontology[C] // 2009 15th IEEE Pacific Rim International Symposium on Dependable Computing. 2009 : 103 – 108.

[7] ZHANG W, ZHAO D. Reuse-Oriented Test Case Management Framework[C] // 2013 International Conference on Computer Sciences and Applications. 2013 : 512 – 515.

[8] VOUFFO-FEUDJIO A, SCHIEFERDECKER I. Test Patterns with TTCN-3[C] // GRABOWSKI J, NIELSEN B. Lecture Notes in Computer Science, Vol 3395 : Formal Approaches to Software Testing, 4th International Workshop, FATES 2004, Linz, Austria, September 21, 2004, Revised Selected Papers. [S.l.] : Springer, 2004 : 170 – 179.

[9] GIBB F, MCCARTAN C, O'DONNELL R, et al. The integration of information retrieval techniques within a software reuse environment[J]. J. Inf. Sci., 2000, 26(4) : 211 – 226.

[10] von MAYRHAUSER A, MRAZ R T, WALLS J, et al. Domain Based Testing: Increasing Test Case Reuse[C] // Proceedings 1994 IEEE International Conference on Computer Design: VLSI in Computer & Processors, ICCD ’94, Cambridge, MA, USA, October 10-12, 1994. [S.l.] : IEEE Computer Society, 1994 : 484 – 491.

[11] 张娟. 软件测试中测试用例复用的研究 [D]. [S.l.] : 上海: 上海大学, 2012.

[12] 顾斌, 董云卫, 王政. 面向航天嵌入式软件的形式化建模方法 [J]. Journal of Software, 2015, 26(2) : 321 – 331.

[13] FRAKES W B, POLE T P. An empirical study of representation methods for reusable software components[J]. IEEE transactions on software engineering, 1994, 20(8) : 617 – 630.

[14] MILI H, MILI F, MILI A. Reusing software: Issues and research directions[J]. IEEE transactions on Software Engineering, 1995, 21(6) : 528 – 562.

[15] GANGULY D, ROY D, MITRA M, et al. Word Embedding Based Generalized Language Model for Information Retrieval[C] // . New York, NY, USA : Association for Computing Machinery, 2015.

[16] ZHANG W, ZHANG Y, LIU T. A topic inference based translation model for question retrieval in community-based question answering services[J]. Chinese Journal of Computers, 2015, 38(2) : 313 – 321.

[17] HUANG P-S, HE X, GAO J, et al. Learning deep structured semantic models for web search using clickthrough data[C] // Proceedings of the 22nd ACM international conference on Information & Knowledge Management. 2013 : 2333 – 2338.

[18] PALANGI H, DENG L, SHEN Y, et al. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2016, 24(4) : 694 – 707.

[19] 严红. 词向量发展综述 [J]. 现代计算机 (专业版), 2019.

[20] HINTON G E, OTHERS. Learning distributed representations of concepts[C] // Proceedings of the eighth annual conference of the cognitive science society : Vol 1. 1986 : 12.

[21] BENGIO Y, DUCHARME R, VINCENT P. A neural probabilistic language model[J]. Advances in Neural Information Processing Systems, 2000, 13.

[22] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[23] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality[J]. Advances in neural information processing systems, 2013, 26.

[24] COLLOBERT R, WESTON J, BOTTOU L, et al. Natural language processing (almost) from scratch[J]. Journal of machine learning research, 2011, 12(ARTICLE) : 2493 – 2537.

[25] CHURCH K W. Word2Vec[J]. Natural Language Engineering, 2017, 23(1) : 155 – 162.

[26] PENG H, LI J, SONG Y, et al. Incrementally learning the hierarchical softmax function for neural language models[C] // Proceedings of the AAAI Conference on Artificial Intelligence : Vol 31. 2017.

[27] GOLDBERG Y, LEVY O. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method[J]. arXiv preprint arXiv:1402.3722, 2014.

[28] SOCHER R, HUANG E, PENNIN J, et al. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection[J]. Advances in neural information processing systems, 2011, 24.

[29] LU Z, LI H. A deep architecture for matching short texts[J]. Advances in neural information processing systems, 2013, 26.

[30] PANG L, LAN Y, GUO J, et al. Text matching as image recognition[C] // Proceedings of the AAAI Conference on Artificial Intelligence : Vol 30. 2016.

[31] GHEORGHE R, HINMAN M L, RUSSO R. Elasticsearch in action[M]. [S.l.] : Manning Shelter Island, NY, 2015.

[32] CARLSON J. Redis in action[M]. [S.l.] : Simon and Schuster, 2013.

简历与科研成果

基本情况 刘智彪，男，汉族，1998 年 10 月生，江西省吉安市人

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

致　谢

行文至此，我也即将告别我的研究生学习生涯，乃至整个学生生涯。在南京大学的六年学生生涯，将成为我生命中一段弥足珍贵的经历。时间回到六年前的金秋，第一次站在南京大学校门前，背着行囊的青葱少年满眼憧憬着未来。时光荏苒，转眼已六载，此刻的我坐在南园十一舍的宿舍里写下这些文字。

感谢我的导师陈振宇教授，感谢他在研究生学习期间给予我的指导和帮助，使我得到足够的锻炼和成长。感谢 ISE 实验室的赵源师兄，感谢他从选题阶段起就一直帮助我开展论文的研究和撰写工作。

感谢我的父母和家人，他们是我的求学生涯中的幕后英雄。感谢他们一直支持和鼓励着我，为我创造了良好的条件，让我能安心投入到毕业论文的研究工作中。感谢我的女朋友，感谢一直以来她给予我的关怀和爱意，这些关爱成为了我的动力源泉。

感谢参与论文评审和答辩的老师，感谢你们在百忙之中依然费心尽力。

感谢所有在这六年里遇到的可爱的人，感谢你们的帮助和鼓励。

即将离开校园，走向更广阔的社会，前路漫漫不可知，在这里想祝福自己，愿自己在往后的路途中历经磨练后依旧笑对人生，祝自己在追寻理想的道路上乘风破浪！