南京大學

NANJING UNIVERSITY

研 究 生 毕 业 论 文

（申请硕士专业学位）

论 文 题 目　面向表达力评测的题目建设与分析系统设计与实现

作 者 姓 名　汤大业

专 业 名 称　软件工程

研 究 方 向　软件工程领域

指 导 教 师　刘嘉 副教授

2020 年 5 月 1 日

学　　号：**MF1832144**

论文答辩日期：**2020 年 5 月 23 日**

指 导 教 师：　　**（签字）**

The Design and Implementation of Problem Construction and Analysis System for Expression Ability Evaluation

By

Daye Tang

Supervised by

Associate Professor **Jia Liu**

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

Software Institute

May 2020

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：面向表达力评测的题目建设与分析系统设计与实现

软件工程 专业 2018 级硕士生姓名：汤大业

指导教师（姓名、职称）：刘嘉 副教授

摘 要

随着社会经济的发展，各行各业逐渐摒弃了以学历、分数作为唯一指标的人才招录方式，转而选择以综合素质评价为基础的人才评价体系。表达能力包括沟通能力，概括能力，领导能力，交际能力等，作为人才的基础能力，在人才评价体系中处于十分重要的位置。然而，当前对表达能力的评价仍然停留在面试官决断这一初级层次，需要大量人力和时间成本，并且无法量化，只能靠经验划分层次。表达力评测项目针对这一问题，提出了一套新型表达力评测方案，通过参与评测，用户可以量化自己的表达能力水平。

题目建设与分析系统是表达力评测项目的重要部分，它关注了评测中“题目”这一核心要素从生成初始化到投入使用再到分析优化的整个流程。首先，为了保证题目主题来源多样性与题目数量，系统需要定时爬取网络资源，并初始化为备用问题；其次，在用户参与评测过程中，系统将根据用户提交的音频，使用音频分析和文本分析手段分析出用户的表达能力水平，并提供简要的评语；随后，系统还需要利用用户提交的答案数据，使用统计学习方法来优化优化题目的参数，提升模型的稳定性；此外，系统也会提供必要的支持，使得专业人员可以介入题目的生命周期，从备用语料库中生成新的题目，或者人为地修改题目参数。

本系统采用了前后端分离的技术，为了兼顾开发效率和开发稳定性，前后端均采用较为主流的框架。前端方面，微信小程序端采用了小程序官方的 MVVC 框架，网页端采用 Vue 框架。后端方面使用 Python 作为编程语言，采用 Flask 作为开发框架，并采用 Celery 作为分布式任务队列框架。数据存储方面，系统缓存数据存储在 Redis 中，结构化数据存储在 MongoDB 中，非结构化的文件数据存储在百度对象存储 BOS 中。

本文对系统进行了功能测试，并使用 Jmeter 对系统服务端进行了压力测试。系统已经上线并经历了两次实验运营。实验结果表明，系统的评分结果的与专家评分差距在 5 分以内的样本数量超过 75%，说明该系统能够在一定程度上量化评测者的表达能力。

关键词：表达力评测，音频处理，文本分析，统计学习方法

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of Problem Construction and Analysis System for Expression Ability Evaluation

SPECIALIZATION: Software Engineering

POSTGRADUATE: Daye Tang

MENTOR: Associate Professor Jia Liu

Abstract

With the development of social economy, all walks of life have gradually abandoned the talent recruitment method which takes the academic record and the score as the only index, and chose the talent evaluation system which is based on the comprehensive quality evaluation. The ability of expression includes the ability of communication, the ability of generalization, the ability of leadership, the ability of communication and so on. As the basic ability of talents, it plays a very important role in the talent evaluation system. However, the current evaluation of the expression ability still remains at the primary level based on the interviewer's decision, which requires a large amount of labor and time costs. This kind of evaluation method cannot be quantified and can only be divided into levels by experience. Aiming at this problem, the expression ability evaluation project puts forward a new program of expression ability evaluation. By participating in the evaluation, users can quantify their own level of expression ability.

The problem construction and analysis system is the part of the expression ability evaluation project. It focuses on the entire process of the core element of the "problem kit" in the evaluation, from generation and initialization to use and analysis and optimization. First, in order to ensure the number and diversity of problems, the system needs to regularly crawl network resources and initialize it as a backup question. Second, during the user participation in the evaluation process, the system will use audio analysis and text analysis methods to analyze the user's expression level and provide brief comments. Third, the system also needs to use the answer data submitted by the user and use statistical learning method to optimize the parameters of the problem kit and improve the stability of the model. In addition, the system will also provide the

necessary support so that professionals can intervene in the problem life cycle, generate new problems from the alternate corpus, or modify the problem parameters artificially.

This system adopts the technology of separating the front and back ends. In order to give consideration to the development efficiency and development stability, the front and back ends adopt the mainstream framework. On the aspect of front end, WeChat mini-program client uses applet official MVVC framework while website client uses Vue framework. The backend uses Python as a programming language, Flask as development framework, and Celery as a distributed task queue framework. In terms of data storage, the system cache data is stored in Redis, structured data is stored in MongoDB, and unstructured file data is stored in baidu object storage BOS.

In this paper, the system is functionally tested, and Jmeter is used to stress test the system server. The system has been online and has experienced two experimental operations. The experimental results show that the number of samples within 5 points between the system's scoring results and the expert's score exceeds 75%, indicating that the system can quantify the evaluator's expressive ability to some extent.

Keywords: Expressiveness Evaluation, Audio Processing, Text Analysis, Statistical Learning Method

目录

表 目 录

图 目 录

第一章　引言

1.1　项目背景及意义

随着社会经济的发展，各行各业逐渐摒弃了以学历、分数作为唯一指标的人才招录方式，转而选择以综合素质评价为基础的人才评价体系 [1]。表达能力包括了领导能力、沟通能力、概括能力等，在人力资源管理、销售顾问、项目经理等岗位，表达能力是最重要的人才素质。目前来说，表达能力的评测方式局限于线下观察，最典型的就是在面试过程中，由专业人员（面试官）考察，这需要耗费较大的人力和时间成本，并且无法量化，仅能依靠面试官的印象与经验来划分人才层次。此外，由于无法获悉自身表达能力所处层次，人才自身也很难有的放矢地进行提升。

近些年来，随着网络技术不断发展，一些线上测评应用应运而生，其中有偏向基础知识的各类考试系统，例如猿测评，牛客测评，也有偏向编码能力的各类 OJ 系统，例如 leetcode OJ，浙大 OJ 等。如图1.1所示，传统的线上评测系统由于评分逻辑固化，往往只能提供选择题，填空题等客观题，最终的评测结果也只有每道题的对错与总分，而对于一些主观题，例如语文作文，英语翻译等，这些系统往往只能依赖于后台大量专业老师阅卷判分。

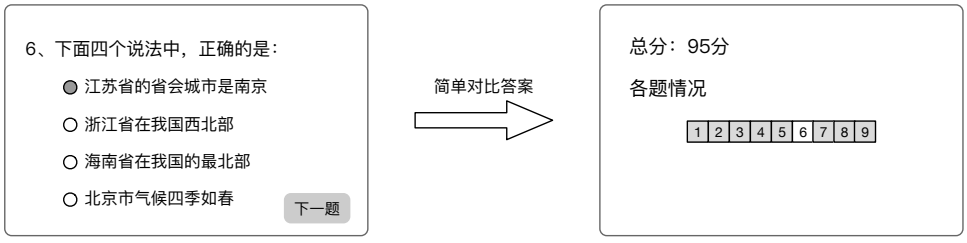

图 1.1: 传统线上评测示例

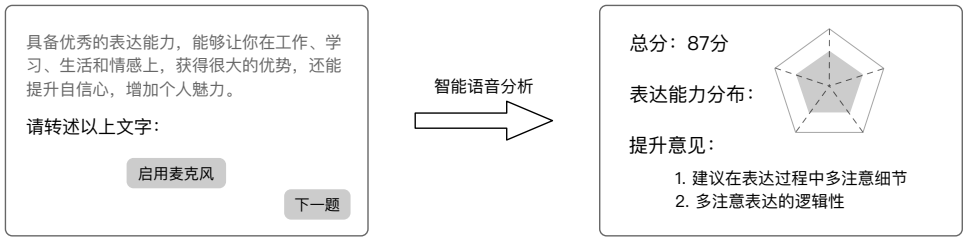

图 1.2: 表达能力评测示例

表达能力是一种非常主观的能力，评测者的回答没有“对错”之分，对于表达能力测试题来说，甚至没有“正确答案”。因此，在表达力评测这一领域中，存在的主要矛盾是：线下评测效率低成本高，而线上评测系统又很难准确评价。如图1.2所示，表达力评测解决方案针对这一矛盾，提出了一种模拟人机对话的表达力评测过程：用户使用麦克风回答系统给出的问题，全部回答完成之后，系统将使用智能语音分析程序分析用户的音频，给出用户的表达力评测分数和能力维度分布蛛网图，并给出简要评语。

“题目套件”指评测中一道题目的内容及其相关的所有参数和回答数据，在表达力评测解决方案中，题目建设与分析系统控制了题目套件从生成到分析再到优化的全部过程，是解决上述矛盾的关键所在。系统使用题目参数和用户音频对用户进行判分，并利用用户数据优化这些参数，使得判分结果更加精确。

1.2　国内外研究现状及分析

根据本文的研究问题，下面从线上评测系统、语音分析与语音评测、表达能力评价体系这三个方面来论述国内外的研究现状。

1.2.1　线上评测系统

在线评测系统（Online Judge System，简称 OJ）一般指在编程竞赛中用于练习算法题的在线系统，著名评测系统包括西班牙的 UVaOJ[1]，俄罗斯的 Codeforces[2]，中国的北大 OJ[3]等。随着互联网的普及，许多公司的招聘和晋升流程中也加入了在线评测内容，比较有名的包括美国的 Leetcode[4]，印度的 GeeksforGeeks[5]等。OJ 的概念与技术逐渐普及之后，开源 OJ 系统出现在大家的视野中，其中以 HustOJ[2] 最广为人知，HustOJ 可以借助 Doker 等虚拟化容器概念实现快速部署，甚至可以使用某些 Linux 系统的包管理工具一键部署。

这类 OJ 系统以服务专业人士为目标，用户往往具有较好的计算机使用能力，因此系统并不重视用户体验，界面风格也较为简略。此外，这些系统的评测题型局限于编程题这一种，分析过程采用“黑盒测试”策略，即事先准备好测试数据，用户提交了程序之后，服务器会编译（解释）用户源文件得到可执行文件，并使用测试用例的输入进行测试，将用户输出与标准输出进行比对 [3]。为

[1]https://onlinejudge.org/

[2]https://codeforces.com/

[3]http://poj.org/

[4]https://leetcode.com/

[5]https://www.geeksforgeeks.org/

了防止源代码抄袭和恶意攻击等现象，有些 OJ 系统会对用户输入的程序进行过滤，并放入沙盒中进行隔离 [4]。

传统 OJ 系统的题目和测试数据大多来自于管理员上传或社区贡献，并保存在自己的服务器中。虚拟评测（Virtual Judge）是一种较为特殊的 OJ 系统，它并不会保存题目或测试数据，而是采用爬虫的方式从各个其他 OJ 系统上获取题目，并提交用户回答到对应 OJ 系统上获取反馈，最终汇集整理之后展示给用户，虚拟评测在其中扮演了代理角色 [5]。

近些年来，陆续出现了一些服务非专业人士的在线评测系统，例如服务于普通企业招聘的北森测评[6]，服务于中小学生教育的猿测评[7]等。北森测评是面向 HR 面试的测评工具，帮助企业确定人才的职位适配程度，从而科学识别人才，题目形式以选择题和判断题为主，而题型包括了逻辑算术题、心理测试题等。猿测评是面向中小学生自我学习的测评工具，题目形式上紧跟升学考试或高考的题型，在判分逻辑上，猿测评遵循了客观题机器判分，主观题专家线上判分的模式，组织招聘了一大批专业老师线上阅卷。

1.2.2　语音分析与语音评测

语音分析技术包括了语音识别、语音合成、语义理解，语音评测等 [6]。以语音识别为例，语音识别（Automatic Speech Recognition，简称 ASR）技术是一项较为成熟的技术，其设想早在电子计算机被发明之前就已经出现，声码器可以作为语音识别的第一次尝试。计算机发明之后，AT&T 实验室第一次在计算机上实现了基于共振峰跟踪的数字 1-10 识别应用，但此时语音识别距离实际应用还有很远的距离 [7]。上世纪 60 年代，人工智能第一波浪潮兴起，在隐马尔可夫链理论（HMM）出现后 [8]，基于 Rabiner 等人的研究，李开复博士在卡耐基梅隆大学发布了第一个大词汇量语音识别系统 Sphinx，从此以后几乎所有语音识别都以此模型为基础 [9]。近些年来，语音识别功能随着智能手机和智能家居走入千家万户，逐渐成为为语音分析技术中最广为人知的一项技术。国外的 Nuance 公司、谷歌公司，国内的科大讯飞、百度、腾讯等公司均有较为成熟的语音识别解决方案。

语音评测也是语音分析技术的一种，相比较语音识别而言，语音评测更偏重于基于一个标准发音模板对使用者的发音水平进行评价，并指出其错误。语音评测技术在智慧教育和医疗康复这两个领域中有着重要的作用。

在智慧教育领域，语音评测技术在考试场景和日常教育场景中均有较为广

[6]https://www.beisen.com/

[7]http://ytk.yuanfudao.com/cp

泛的应用。全国普通话水平测试（PSC）和大学英语四六级考试口语考试（CET-SET）均使用了人机对话形式，考生阅读完材料或回答完问题后，机器将在专家辅助下对考生的回答进行自动判分。在日常教育场景下，语音评测技术广泛应用在 K12 母语学习、职业外语学习等应用中，学生可以使用这些应用评价自己的口语水平，并获得专业分析及改进意见 [10]。国内有多家公司都在语音评测 + 教育的领域中发力，其中得到广泛应用的包括科大讯飞的语音评测解决方案[8]、腾讯云的英语语音评测[9]等。

在医疗领域，医学研究表明，许多疾病包括听力受损、语言发育迟缓以及阿尔茨海默病（AD）都会不同程度地影响人的说话和表达能力 [11]。在这些疾病康复过程中，除了专业大脑相关区域测量之外，还需要专业人员对其说话语音进行评测，关注其是否达到正常标准。语音评测技术目前主要应用语言能力相关疾病的康复过程中，通过搭载语音评测技术，失语症康复治疗仪等专业仪器可以帮助专业康复医师更好地工作。

1.2.3　表达能力评价体系

表达能力可以从医学和社会学两个角度来评价。从医学角度来讲，表达能力和说话能力、听力类似，对应了大脑中某些皮层。现代医学证明，大脑中，脑前额叶主要影响了人的个性表达能力、决策能力和与社会沟通能力。此外大脑中的颞叶和顶叶这两个区域相互作用，也会对上述能力产生影响 [12]。因此，对以上区域的专业医学观察可以在一定程度上解释并区分人的表达能力。

在日常生活中，人的表达能力评价往往从其社会学角度出发，即通过与其交流，从不同的维度评价其表达能力。在不同场景下，学者给表达能力定义了不同的评价标准。在托福雅思等语言考试场景下，考官关注点集中在发音是否流畅，语速是否均衡，观点是否明确，词汇是否丰富这四个维度 [13]；在高校演讲教育场景下，评价标准集中在语音标准、语用规范、表达内容、表达技巧、形象风度、应变能力、表达效果这七个维度 [14]；在人才评价场景下，表达能力的评价维度更偏向于概括能力、还原能力、沟通能力这三个方面。总结起来，在日常交流与沟通中，针对性、清晰性、逻辑性、情感性、个性化是口语表达中最需要注意的几个方面。

[8]https://www.xfyun.cn/services/ise

[9]https://cloud.tencent.com/product/soe-e

1.3　本文主要工作

本文所设计与实现的题目建设与分析系统是表达力评测解决方案的核心系统，其需要完成的目标主要有四个。首先，系统需要针对主观题无法自动评分的痛点，解决题目的分析问题；其次，系统需要通过优化参数来提升题目评分的准确性；接着，系统需要解决题目来源多样性的问题，即生成题目；最后，系统需要为题目提供增删改查业务逻辑。本系统将围绕这四个目标来展开设计，分为四个主要的模块：

1）分析模块：用户使用麦克风回答题目之后，系统将使用题目参数与用户音频文件来分析用户的回答情况，给出其表达能力的分数与维度分布情况。同时，系统也会针对用户回答过程中的情况指出其不足之处，并提出简要建议，帮助用户提升自己的表达能力。

2）优化模块：优化模块与分析模块是相辅相成的，分析模块使用题目参数分析用户的表达能力情况，得到用户数据，而优化模块将使用用户数据优化题目的参数。在收集了一定量的用户数据之后，系统会参照题目本身的情况，使用不同的算法对题目参数进行优化与重排，以保证分析过程的准确性和稳定性。

3）语料模块：与传统的评测系统不同，表达力评测是一个全新的领域内的评测项目，并没有现成的语料可供使用，专家在新增题目时需要一定语料作为思路来源。为了使题目数量维持在较高水平避免用户的恶意刷题现象，系统需要定期从网络资源中爬取合适的文本，整理之后形成原始的语料库，并挖掘语料的关键词、标签等语义信息，为语料初始化参数。

4）业务模块：题目建设与分析系统包含了一系列对题目的增删改查操作，这些操作被统一收敛在业务模块中，以 Flask Web app 的形式对外发布 http 接口。客户端在权限允许的情况下可以访问业务接口，对题目进行业务操作。

此外，本文还叙述了系统的持续集成情况和测试情况。系统使用了 Gitlab+jenkins 作为 CI/CD 工具，保证了软件工程过程中的持续性。在系统测试环节中，除了功能测试和性能测试之外，还包括对两次实际公测情况的简要说明。

1.4　本文组织结构

本文的组织架构如下：

第一章：引言部分。这一章概述了表达力评测项目的背景、意义和前景，总结了国内外对线上评测、语音分析和表达能力评价体系的相关研究以及本文的主要工作，指出了题目建设与分析系统在表达力评测项目中的重要地位。

第二章：相关理论和技术部分。这一章叙述了本系统使用的相关理论与技术，包括数据挖掘模型、统计学习方法、自然语言处理技术、Scrapy 框架、Flask 框架等。

第三章：需求分析与概要设计部分。这一章从项目总体规划开始，介绍了系统的涉众分析、用例分析、功能与非功能需求，并以 4+1 视图的形式介绍了系统的架构和模块划分，最后介绍了系统的数据库设计。

第四章：详细设计与实现部分。这一章从四个模块的角度，以类图、顺序图、关键代码和界面截图等形式介绍了系统的详细设计与实现。

第五章：持续部署与测试部分。这一章从软件工程的角度介绍了系统如何进行持续集成与持续部署，并详细介绍了系统的测试情况，包括功能测试、性能测试与实验评估。

第六章：总结与展望部分。这一章对系统的工作进行了总结，指出了目前存在的问题和不足，展望了系统未来的发展方向。

第二章　相关理论与技术

2.1　数据挖掘

本质上，表达力评测的分析模型是一个数据挖掘的过程，即以用户提交的音频文件作为数据来源，从中挖掘出有用的信息，进而分析出用户的表达能力水平。数据挖掘 [15]（Data Mining），又称知识发现（Knowledge Discovering），指的是从大量的数据中提取出有用的知识，这一过程类似于砂砾淘金的过程，因此使用了“挖掘”一词 [16]。近些年来，随着大数据、云计算等概念进入主流视野，无论是国家还是企业都越来越意识到数据挖掘这一技术的重要性。

2.1.1　数据挖掘过程模型

广义上的数据挖掘模型涵盖了从用户数据收集到有用知识展示的所有过程，这其中，关键的步骤包括数据收集、数据清洗、数据集成、数据挖掘（狭义）、模式评估、知识展示这六个主要步骤 [17]。

数据收集步骤将从包括各类智能终端、用户使用记录在内的地方收集数据，这一步骤实际上包含了部分隐私与伦理问题 [18]。数据清洗 [19] 指在正式进行数据挖掘之前，对数据进行的预处理工作，通常包括去除噪声、纠正不一致、统一维度、补齐缺失数据等操作，数据清洗使得数据源更加整齐，避免数据挖掘过程被干扰产生难以使人信服的结果。数据集成 [20] 指的是将不同数据源的数据集成到一起，并对数据进行规约，常见的操作例如主成分分析、小波变换等，数据集成大大减小了数据源的体积和数据挖掘的难度。狭义的数据挖掘 [21] 步骤指从比较整齐、完整的数据源中获取知识的过程，常见的操作包括分类、回归、异常分析等。模式评估步骤将评估上一步中获取的知识是否是有价值的。知识展示步骤 [22] 将以各种可视化手段，把数据挖掘过程中获取到的知识生动、形象地展示给用户。

2.1.2　数据属性与数据向异性度量

数据往往是一系列具有相似结构的实体集合，而这些实体包含了多个维度，又称数据的属性或数据的特征，实体可以表示为各个属性组成的向量 [23]。数据的属性种类包括二元属性、标称属性、序数属性和数值属性。二元属性又称布尔属性，只有两个取值，通常被标记为 1 和 0，例如疑似患者患病与否、某场事故的受害人生还与否等。标称属性又称枚举属性，指取值在一定枚举范围内的属

性，例如头发的颜色、人的职业等。序数属性和标称属性类似，也是取值在一定范围内的属性，但序数属性往往有某些有意义的顺序，例如人的学历，军队的军衔等。数值属性是定量的，是可以度量并直接计算距离的量，例如人的年龄，学生的成绩等。

数据相异性指两个实体之间的距离，通常由这两个实体的属性计算得到，不同类型的属性对应的计算方式不同 [24]。枚举属性和二元属性当且仅当两个实体对应相同时为 $\frac{1}{size_f}$，否则为 0，其中 $size_f$ 是所有枚举属性与二元属性的总个数 X。序数属性之间的距离可以公式 $|\frac{r_{if}-1}{M_f-1}-\frac{r_{jf}-1}{M_f-1}|$ 表示，其中 r_{if} 和 r_{jf} 表示两个实体属性对应的排位，M_f 表示这个属性全部的排位数量。数值属性的距离计算公式为 $\frac{x_{if}-x_{jf}}{max_f-min_f}$，其中 x_{if} 和 x_{jf} 是这两个对象的数值属性，max_f 和 min_f 是这个属性的最大值和最小值。

2.2　统计学习

在优化模块中，系统利用用户数据，通过一定的统计分析对题目的参数进行优化，使得分析模型更加稳定。统计学习（Statistical Learning）是计算机使用数据构建模型并利用模型对数据进行预测和分析的一门技术，又称为统计机器学习 [25]。统计学习是人工智能的一个分支，在上世纪五六十年代，以符号主义为核心的统计学习是人工智能的主要发展方向。二十世纪九十年代，以神经网络为基础的连接主义成为人工智能的主流，并随着计算机算力的不断增加，神经网络理论迅速发展延伸出深度神经网络理论 [26]。连接主义的缺点在于，学习过程是黑箱不可见的，即缺乏解释性，而与之相对的统计学习过程是白盒的，具备可解释性 [27]。

2.2.1　回归分析

回归分析 [28] 是统计学中经常使用的一种分析数据的方法。回归分析需要建立自变量 X 和因变量 Y 之间存在的关系模型，即寻找参数 β 和函数 f，使得 $Y \approx f(X,\beta)$，从图形角度触发，回归分析即寻找一条最合适的曲线拟合数据点。回归分析对数据的数值型属性的预测具有很重要的意义，被广泛应用在经济学、生物化学、医学等领域中。

回归模型分为线性回归、对数几率回归。线性回归（Linear Regression）有两种基本的模型，简单线性回归和复变量线性回归。简单线性回归解决了单一自变量与因变量的对应关系，预测的函数 f 限定为线性函数，需要学习的变量是参数 β。复变量线性回归将简单线性回归中的单一自变量升级为多变量，即预测向量 $\vec{x}$ 与因变量 y 线性关系中的参数 [29]。对数几率回归又称逻辑斯蒂回归

（Logistic Regression）[30] 是最常用的一种预测模型，用于预测实体属于某类别的几率，它采用了函数模型 $f(x) = \frac{1}{1+e^{-(\omega^T x+b)}}$，需要学习的参数是 ω 和 β。

2.2.2　梯度下降法

在回归分析和其他的一些分析理论中，经常出现已知函数类型 f, 并且有一批数据（自变量 x）和对应标签（因变量 y），需要确定参数 θ 的情形。在这种场景下，我们可以建立损失函数 $J(\theta) = dist(f(\theta, x), y)$，寻求合适的 θ 使得上面的函数值最小。在某些特定的模型，例如线性回归模型中，θ 的学习过程可以使用最小二乘法解决 [31]。但在某些复杂场景中，最小二乘法无法使用，这时可以使用梯度下降法 [32]（Gradient Descent）。

梯度下降法的主要原理是逐步调整参数，使得损失函数沿着当前位置下降最快的方向，即损失函数的切线方向逐渐下降。梯度下降法的过程由以公式2.1指定，其中 α 为给定的学习率：

$$\theta_i = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i} \quad i = 1, 2, 3, \ldots, n \tag{2.1}$$

梯度下降的过程是一个迭代的过程，上述过程需要重复数次，直到连续两次的预测结果差值在给定的较小数值 ϵ 以内或达到最大迭代次数为止。

2.3　自然语言处理技术

分析模型在提取用户回答的文本特征时，应用了大量自然语言处理技术。自然语言处理（Natural Language Processing，NLP）是人工智能和语言学的交叉学科，包括了语法解析、语义认知、关键词提取，语句生成，情感分析等范畴 [33]。自然语言处理起源于著名的图灵测试，上世纪五六十年代，乔治城实验成功实现了机器翻译，可以将少数俄语自动翻译为英文 [34]，机器翻译作为自然语言处理的重要分支流传至今。上世纪七十年代，研究人员开始设计“概念本体论”程序，这一时期出现了许多智能聊天机器人，演化成今天的各种智能助手应用。随着深度神经网络发展和大型语料库的构建，自然语言处理的前景十分广大。

2.3.1　语法分词

语法分词是自然语言处理，尤其是中文、日文等不断词语言自然语言处理的基本步骤 [35]。和英文不同的是，中文的两个词语之间并没有明显的间隔，因此分词是中文文本信息分析、自动标注和机器翻译等技术的重要基础。

中文的语法分词技术有三种主要类型 [36]。一是基于字符串匹配的分词方法，这种方法以词典匹配为核心，实质上一种专家规则的切分。分词器将待切

分的文本进行基本的预处理之后，按照词典中的词语进行逐步匹配，遇到歧义的文本时，往往参照最长匹配或最优匹配进行决策。二是基于理解的分词方法，这种方法模拟人的理解，以深度神经网络方法为核心，训练时间较长，需要大量标注好的数据，且往往存在过拟合现象。三是基于统计学习的分词方法，其内核可以是隐马尔可夫模型、最大熵模型、条件随机场模型等，谷歌搜索的中文分词即使用了隐马尔可夫链模型。总的来说，基于字符串匹配和统计学习的分词方法速度较快，分词成本比较低，但是仍然存在不同领域内分词效果差距较大，算法复杂度较高的问题。

本文使用的 jieba 分词[1]综合使用了字符串匹配分词方法和统计学习分词方法，并支持用户自定义词典。分词器使用前缀匹配模式扫描文本，生成词语所有可能情况的有向无环图。接着，分词器会使用动态规划查找最大概率匹配路径。对于词典以外的汉字，分词器会采用隐马尔可夫模型预测词语构成。

2.3.2　词语向量化与近义词获取

词语向量化（Word2Vec）指将词语（短文本）转化为数字向量的过程，词语向量化有两种基本的思路。

第一种思路基于单字击中的想法，每个字用一个单独的向量维度表示，某个词语的向量是词语包含的字对应向量的集合。举个例子，如果“花”、“钱”、“朵”的向量表示分别为 $[1,0,0,\ldots]$、$[0,1,0,\ldots]$、$[0,0,1,\ldots]$，那么“花钱”和“花朵”的对应向量分别是 $[1,1,0,\ldots]$ 和 $[1,0,1,\ldots]$。这种思路的好处在于计算速度较快，且易于理解和反向分析，而缺点在于缺乏语义性，实际上无论是中文还是英文，相同的字在不同的地方往往有含义很大的差别，而有些字完全不同的词语可能表达相同的含义。

第二种思路基于分布假设，即上下文一致的词，其语义相似，因此它们的向量表示应该类似。神经网络语言模型采用的就是文本分布式表示。词向量是训练这个神经网络时的隐藏层参数 [37]。

近义词获取是在词语向量化的基础上进行的。本系统采用的近义词工具 synonyms[2]是 Python 的一个中文近义词库，它包含近义词匹配、句子相似度计算、语义偏移等功能。它使用了分布假设思路对维基百科的中文语料库进行了训练，生成词向量之后获得近义词表。

[1]https://github.com/fxsjy/jieba

[2]https://github.com/huyingxi/Synonyms/

2.3.3　关键词提取

关键词提取指从成段文本中获取关键词的信息检索技术，是自然语言处理技术的重要分支。目前来说，提取关键词有两种较为通用的算法 TF-IDF 和 TextRank。

TF-IDF（词频-逆文档频率）算法的基本思想是，如果一个词在本文档中反复出现，但在整个文档库中出现频率较低，说明这个词在本文档中起到关键作用。某个词语的关键词比重计算方法由公式2.2决定，其中 tf 表示词语 i 在文档 D 中的词频，而 idf 表示在文档集合中，这个关键词出现文档的个数。TF-IDF 的优点在于计算简单迅速，在特定场景下十分有效，而缺点在于其效果很大程度上取决于文档库的质量：如果文档库有偏向性，可能导致许多关键词比重计算出现偏差 [38]。

$$\omega_{t_i D} = tf_i \cdot ln(idf_i) \tag{2.2}$$

TextRank（文本排名）算法基于谷歌公司给搜索网页排序的 PageRank 算法，其基本思想是：如果一个词语出现在许多不同的词语后面，说明这个词 TextRank 值较高；而在一个 TextRank 值较高的词语后面的词语相应的 TextRank 值也会较高，最后 TextRank 值较高的词是关键词 [39]。这种思路的优点在于它考虑了不同词语之间的关联性，因此语义性上更好，而缺点在于它开始时假定所有词语权重一致，即仅依赖于文档本身，不考虑整个文档库的情况。

2.4　Vue 简介

Vue.js[3]（简称 Vue）是在美中国留学生尤雨溪（Evan You）于 2014 年发布开源的一款渐进式 JavaScript 框架。最早的 Vue 是作者从 AngularJS 中提取了部分想法构建的，因此在设计理念上，Vue 与 AngularJS 有许多相似之处，但更加轻量，运行也更加高效。从狭义上说，Vue 是一套前端开发框架，但是实际上，Vue 有自己完整的生态平台，并催生了许多基于 Vue 的前端解决方案，本系统采用的前端开发模板 D2Admin 即是基于 Vue 的脚手架模板。

2.4.1　Vue 渐进式框架

Vue 框架是一个渐进式的框架，这表明它在概念上是迭代递进的，最内层是核心而最外层靠近开发者。Vue 从核心向外分别是视图模板引擎、组件系统、客户端路由、大规模状态管理和构建工具。

[3]https://cn.vuejs.org/

Vue 并不是传统的 MVVM 框架，它只有状态 state 和模板 view 两个核心概念，而它的核心视图模板引擎即是 state 与 view 之间的互相映射。用户交互会改变状态，而状态的改变会带来模板的重新渲染。Vue 的组件系统和 React 的自定义组件类似，它支持用户自定义广义的 Html 标签并形成复用。客户端路由又称前端路由，开发者可以使用不同的路由规则映射不同的页面。Vue 的大规模状态管理职能由 Vuex 实现，它解决了多组件共享状态时，可能会出现的数据流、控制流重叠的问题。此外，Vue 为广大开发者提供了大量的构建工具和脚手架，这些工具是的开发者可以快速搭建并部署 Vue 应用。

2.4.2　Vue 核心插件

Vue 的核心插件包括 vue-router、vuex、vue-loader、vueify 和 vue-cli 等，这些插件是 Vue 官方支持的，但并不包含在 Vue 核心库中，开发者可以选择使用或不使用这些插件。众多功能丰富的插件为 Vue 良好生态平台提供了有力的支撑，这也是 Vue 开源的重要原因。

Vue Router 是官方支持的路由库，它往往是大部分单页面 vue 应用的首选路由库。但实际上，Vue 也支持许多著名的第三方路由框架，如 Page.js 和 Director 等。Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式，它使用了集中式存储来进行状态组件管理，以方便不同组件之间共享状态。Vue Loader 是一个 webpack 的 loader，它允许你以一种名为单文件组件的格式撰写 Vue 组件，并支持单组件 css 和热重载等特性，为 Vue 应用开发提供强大的工作流支持。Vueify 顾名思义，指的是使用.vue 格式的文件定义组件，一个.vue 文件就是一个组件，在其中可以定义样式和用户操作响应，这使得定义组件的代码变得更加简洁。Vue CLI 是 Vue 官方支持的用来创建应用的脚手架工具，它基于著名的前端打包工具 webpack，并针对 Vue 进行了定制化 [40]。

2.5　Flask 简介

Flask[4]是一个用 Python 语言编写的 web 应用框架，基于 WSGI 工具和 Jinja2 模板引擎。系统的业务模块使用 Flask 对外提供 web 服务。作为和 Django 几乎同等程度受欢迎的 python 后端框架，Flask 提供了更轻量的开发体验，更少的配置选项和更小的应用体积。在保持较为弹性的框架前提下，Flask 提供了许多独具特色的核心内容。Flask 的特色功能包括：

1）Flask 是 100%WSGI1.0 兼容的。由于 Python 是单线程的，多个请求同时到达服务器时需要排队执行，而 uwsgi 作为 flask 与 nginx 之间的 web 服务器，可

[4]https://palletsprojects.com/p/flask/

以有效的唤起多个 Flask 处理线程并发处理请求，大大增加了服务器吞吐量。

2）Flask 提供了完整的请求拦截与分发，并对 Restful 风格请求有额外的支持，Flask 的请求路径可以包含各类参数，还可以使用 app.before_request 注解对请求进行预处理，实现统一的请求过滤功能。

3）Flask 提供了内置的开发用服务器，这使得开发人员模拟用户请求调试程序时更加便利。

4）Flask 基于 Unicode，这对汉语、日语和韩语等非拉丁字母的语言提供了无比的便利性，开发人员不必额外考虑令人头痛的字符转换问题，可以更好地专注于核心逻辑的开发。

5）Flask 集成了 python 自带的单元测试框架 unittest，这使得开发者可以更好的对代码进行单元测试，在一些第三方可视化工具的帮助下，开发者可以看到代码的覆盖率、复杂度等可视化信息。

除了上面提到的特色功能以外，Flask 还提供了支持安全 cookies、Google App Engine 兼容等功能 [41]。

2.6　Redis 简介

Redis[5]全称 Remote Dictionary Server 即远程字典式服务，是目前主流的缓存中间件，它是一个基于内存的，高性能的 key-value 数据库，经常用来存储 session 信息，登录状态等需要经常读取的信息。本系统使用 Redis 存储分析任务元数据、用户信息等。

相比较其他内存数据库和主流的关系型/非关系型数据库，Redis 主要具有以下几个优势 [42]：

1）Redis 是完全基于内存、单进程、单线程的，所有的指令都是顺序执行，不用考虑多线程访问时线程切换而消耗 CPU。因此 Redis 更适合高并发的读写请求，在处理并发读写时性能极高。

2）Redis 提供了分布式集群和高可用机制的实现。在 CAP 原理取舍上，Redis 采取了较为通用的 BASE 理论，即基本可用、软状态、最终一致性。

3）尽管 Redis 是一个基于内存的数据库，Redis 也支持数据持久化。Redis 的数据持久化使用了快照 + 日志形式。快照是一次全量的备份，能够将内存中的 Redis 数据全部保存在磁盘中；日志是增量式的，它会保存上一次快照之后系统所有的读写操作。Redis 在重启之后会读取最新的快照，并按照日志重新执行一遍指令，以恢复到之前的状态。

[5]https://redis.io/

4）除了基本的 key-value 数据存储之外，Redis 还提供了其他类型数据结构的存储，包括有序列表（list）、无序集合（set）、排序集合（zset）、哈希列表（hash）等。

5）Redis 所有单个操作都是原子性的，即要么成功执行要么完全不执行，对于组合操作，Redis 可以使用 MULTI 和 EXEC 指令支持事务性。

6）Redis 还支持发布订阅设计模式、key 过期策略制定等丰富特性。

2.7　Scrapy 简介

Scrapy[6]框架是一个 Python 编写的网络爬虫框架，被广泛应用在数据挖掘，网站监测、自动化测试等领域中。在开源爬虫框架里，Scrapy 框架在 Github 上的标星数比其他 2-6 名框架标星数的总和还要多，足可以看出 Scrapy 的受欢迎程度。系统的语料模块使用了 Scrapy 框架从网络资源中获取文本素材，经过初始化后形成了原始语料。

2.7.1　Scrapy 架构简介

架构上，Scrapy 可以分为引擎、调度器、下载器、爬虫、管道这几个部分。引擎是 Scrapy 框架的核心，负责调度器、下载器、爬虫、管道等组件的通信、信号传递、数据传输等工作，也是框架的指挥中心，负责生成请求。调度器实际上是请求的任务队列，它从引擎处获得请求，按照一定的方式进行整理排列之后，将其加入调度队列中；下载器从调度器中获得请求实体，发送请求并获得响应实体，再讲响应发还给引擎；爬虫需要从响应中获取数据字段组成实体，一般来说，爬虫处理的是网页数据，即 html 文件数据，但是爬虫也可以处理直接的 API 数据；管道负责处理爬虫获得的实体对象，并交由后续指定的分析步骤使用。

除了上面所说的核心部件之外，Scrapy 还可以自定义下载中间件和爬虫中间件。下载中间件可以在下载器中自定额外的下载功能，对请求和响应进行预处理，爬虫中间件可以在爬虫中自定义数据字段提取方式，实现更复杂的实体提取目标。

2.7.2　Scrapy 特点

Scrapy 是一个快速的、高层次的 web 爬虫框架，其主要特点包括可扩展性强、构建项目快、性能高、文本处理方法多等。

Scrapy 的可扩展性主要体现在流水线的作业形式上，它允许开发者自定义管道组件实现自己想要的功能，当业务需要扩展时，只需要添加额外的管道即

[6]https://scrapy.org/

可实现。不同于其他爬虫工具，Scrapy 是一个框架，它提供了丰富的预设功能与代码，这使得 Scrapy 能够快速构建大型的网络抓取项目，而具有类似功能的 Pyspider 等仅是一个工具，只实现了部分 Scrapy 功能。Scrapy 支持多线程操作，可以同时向不同的网页发起抓取请求，并且配置起来十分容易，这大大提高了抓取性能。此外，Scrapy 提供了丰富的 Html 处理方法，能够提取任意标签下的数据内容，还支持不完整 Html 文档分析，大大降低了开发者数据处理的难度 [43]。

2.8　MongoDB 简介

MongoDB[7]是一种面向文档的数据库管理系统，属于非关系型数据库，最早由 10gen 团队于 2007 年 10 月开发，2009 年 2 月首度推出。为了解决 Web 应用数据存储难以扩展的问题，MongoDB 将每一个数据存储为一个文档，数据结构由嵌套的 key-value 组成，不同的数据文档结构可以相同也可以不同，这为数据实体的扩展性提供了便利。调查显示，在 2019 年，非关系型数据库占到了 40% 的市场份额，而 MongoDB 在非关系型数据库中占据近六成，足见其受欢迎程度。本系统所有的数据实体均采用了 MongoDB 进行存储。

MongoDB 是非关系型的，这意味着 MongoDB 并不完全遵循范式（normalization）。MongoDB 是可以反范式（denormalization）的，即每个实体拥有的其他实体都是一份完整的数据副本，而不是一个链接，这意味着在读取数据时无需链接数据库更加容易，但在更新数据时需要更新多个副本。这为程序员开发提供了很大的便利，即程序员可以决定何时嵌入文档副本，何时嵌入引用，这完全取决于内嵌文档的大小和读写频率。

MongoDB 拥有类似面向对象实体查询的搜索方式，这帮助 MongoDB 赢得了许多习惯使用面向对象语言的程序员的青睐。MongoDB 的查询语法很完善，原生地支持包括正则搜索在内的几乎所有的搜索方式，并且性能极高。

MongoDB 在底层使用 Bson 和 GridFS 作为文件存储结构。BSON 是一种类 json 的一种二进制形式的存储格式，它是 MongoDB 默认的数据存储格式，用于存储 16MB 以下的文档。GridFS 用于存储 16MB 以上的“大文件”，例如图片，音频等。GridFS 会将大文件对象分割成多个小的文件片段，每个文件片段将作为 MongoDB 的一个文档存储，为了方便程序员操作，分片的过程对外部是完全透明的。

此外，MongoDB 是支持索引的，这对互联网应用来说十分重要，因为大部分的互联网应用数据条目数较大，专注于某几个字段的搜索，而且对性能要求较

[7]https://www.mongodb.com/

高，如果没有索引则需要对文档库进行遍历搜索，这是不可接受的。MongoDB 默认在 _id 字段上添加索引，支持自定义单字段索引、多字段索引、复合索引、文本索引（暂时不支持中文文本索引）等。和大部分关系型数据库相同，MongoDB 的索引原理是 B+ 树索引 [44]。

2.9　本章小结

本章主要介绍了本系统涉及的相关理论与技术。理论层面，本章首先介绍了分析模块使用的数据挖掘技术，包括数据挖掘模型和数据属性简介；接着介绍了优化模块使用的统计学习方法，重点介绍了回归分析和梯度下降法；最后介绍了语料模块使用的自然语言处理技术，包括语法分词、词语向量化等。技术层面，本章介绍了前端使用的 Vue 生态，后端使用的 Flask 框架、Redis 技术、Scrapy 框架 MongoDB 数据库。

第三章　系统需求分析与概要设计

本系统是表达力评测项目的核心系统，负责整个解决方案中题目的生命周期建设与分析过程。本章将从表达力评测项目的总体规划出发，分析目标用户（系统）的需求，从不同角度对系统进行架构设计，并对系统的持久化模型进行概要设计。

3.1　表达力评测项目总体规划

表达能力在日渐丰富的社会人才评价体系中处于重要的位置，在某些管理职位显得尤为关键。不同于分数、学历、智商等硬性指标，表达能力是一种“软实力”，目前并没有很好的量化手段。表达力评测项目针对这一问题，提出了一种类似于传统考试的表达力评测过程：用户逐题回答系统给出的问题，全部回答完成之后，系统会给出评测者的分数，并提供简要评语。

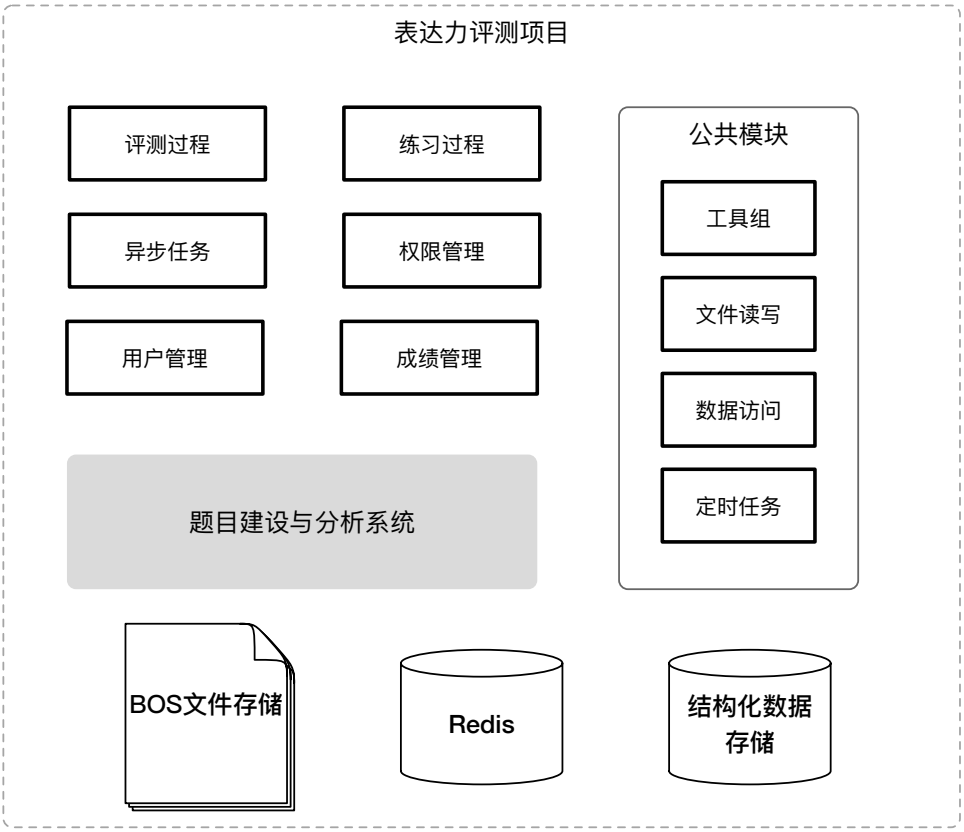

图 3.1: 表达力评测项目架构

如图3.1所示，表达力评测项目包含了评测流程、权限管理、异步任务管理等模块，而题目建设与分析系统是表达力评测项目的核心部分，它主要提供以下三个功能：一方面，在评测流程中，系统需要根据用户的回答分析用户的表达能力水平；另一方面，系统需要定期优化题目参数，提升评分的稳定性和准确性；最后，为了保证题目源源不断，避免出现所谓的“刷题”现象，系统需要提供语料来源来辅助管理员新增题目。

3.2　涉众分析

表达力评测项目的涉众分析如表3.1所示，主要包括评测者和管理员这两类。评测者是表达力评测应用的主要用户，为了对自己的表达能力有一个清晰的认知，或者有需求提升自己的表达能力，他们愿意参加表达力评测。这类人通常具有较高的文化素质，职业包括高等院校学生、白领等，也有一定的设备操作能力，但他们可能并不精通计算机操作，也不一定有条件使用桌面端软件。管理员是解决方案购买方的工作人员，他们通常具有较好的计算机操作水平，工作环境固定为桌面端，比较熟悉 office 等应用软件。这类人经过培训后，对表达能力的概念和内涵有一定了解，但实际上不清楚系统运行原理，尤其数学水平、分析水平较差。

表 3.1: 用户角色说明

用户角色	用户特征
评测者	评测者是需要评测自己表达能力的用户，一般来说具备一定的设备使用能力，但并不精通计算机操作，且有很大的移动设备使用需求，包括手机，平板电脑等，使用系统的环境并不固定。
管理员	管理员是系统所有方的工作人员，使用环境比较正式，一般为桌面端，具备计算机操作能力，对人的表达能力构成有一定了解，但不了解系统内部运行原理，使用时需要较多系统提示。

3.3　系统需求分析

3.3.1　系统用例图

根据项目总体规划中系统需要实现的功能及本项目的涉众特征，得出系统用例如图3.2所示。其中直接面向管理员的用例两个，分别是题目管理与备用语料管理，题目管理包括对题目的增改查操作，备用语料管理包括备用语料删除操作；通过评测模块间接面向评测者的用例有两个，分别是评测题组选取和成绩分析；此外，系统还有两个通过定时任务管理器控制的辅助功能，分别是备用语料新增和题目优化。

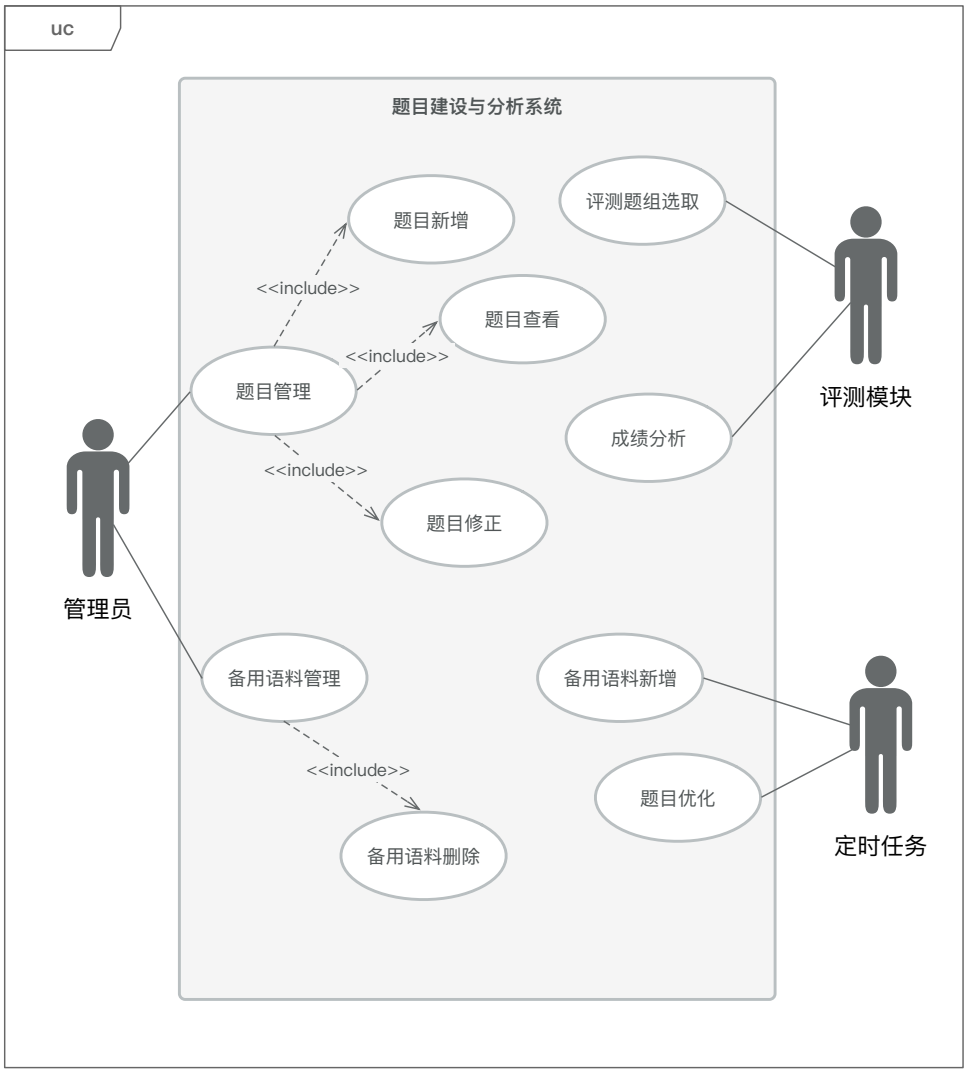

图 3.2: 系统用例

3.3.2　系统用例描述

题目新增对于整个表达力评测项目来说至关重要，如果一个评测应用的题目是有限的不能新增，评测应用将很快被恶意刷题者全部摸清摸透。新增题目需要支持单独新增和批量新增，其用例描述如表3.2所示。

表 3.2: 题目新增用例描述

描述项	**说明**
名称	题目新增
参与者	管理员
优先级	中
触发条件	管理员需要新增题目
前置条件	管理员已通过系统验证，有该操作权限
后置条件	新增题目入库
正常流程	1. 管理员进入题目新增界面，选择新增题目类型 2. 如果是语音题型，系统需要从备用语料中给出提示，帮助管理员快速建立内容 3. 如果是其他题型，系统需要管理员手动录入题目的所有内容，包括题目主体，题目标签，题目难度，正确答案等 4. 管理员确认录入，系统新增该题
扩展流程	3a. 其他题型需要支持批量录入 1. 用户下载示例文件，用表格的形式大量录入相同类型题目，确认提交 2. 系统给出进度条提示有多少已录入
特殊需求	1. 单道题目录入系统响应时间不超过 1s

管理员可以查看到所有的题目清单，对其常见属性，例如难度、标签等需要支持筛选和排序。此外，有些题目模型在自动调优多次之后仍然不够稳定，这些题目风险较高，比较需要人为介入修改，风险程度是题目清单的默认排序依据。题目列表查看的用例描述如表3.3所示。

表 3.3: 题目列表查看用例描述

描述项	**说明**
名称	题目列表查看
参与者	管理员
优先级	高
触发条件	管理员需要查看题目列表以及题目详情
前置条件	管理员已通过系统验证，有该操作权限
后置条件	无
正常流程	1. 管理员进入题目列表页面，查看题目清单 2. 系统按照风险程度从高到低排序，分页展示题目 3. 管理员选择查看某个具体题目详情 4. 系统展示题目的详细内容，包括内容，词库等
扩展流程	无
特殊需求	1. 题目清单支持按照难度、标签、类别等依据对题目列表进行筛选 2. 列出题目列表的系统响应时间不超过 5s，且需要有等待提示

在查看了题目清单之后，管理员可以选择一个题目，对其内容进行修订，系

统会提供必要的帮助来提示管理员，例如这道题的平均分等。管理员可以修改的内容包括题目的文本内容，题目描述，词库，难度，正确答案等。题目修正的用例描述如表3.4所示。

表 3.4: 题目修正用例描述

描述项	**说明**
名称	题目修正
参与者	管理员
优先级	中
触发条件	管理员需要修正题目的相关内容
前置条件	管理员已通过系统验证，有该操作权限
后置条件	修正后的题目入库
正常流程	1. 管理员选定某道题进行修正 2. 系统给出这道题的作答情况，并给出修正意见，例如降低难度，修改词库，删除题目等 3. 管理员修正题目，提交修正 4. 系统显示提交成功，修正该题目
扩展流程	无
特殊需求	1. 删除题目时需要给出二次提示 2. 修正每道题的系统响应时间不超过 1s

当管理员认为备用语料库中有一些素材不适合作为正式题目使用时，管理员可以将其从备用语料库中手动删除。备用语料删除的用例描述如表3.5所示。

表 3.5: 备用语料删除用例描述

描述项	**说明**
名称	备用语料删除
参与者	管理员
优先级	低
触发条件	管理员需要删除备用语料库中的内容
前置条件	管理员已通过系统验证，有该操作权限
后置条件	对应备用语料从库中被删除
正常流程	1. 管理员进入备用语料库页面 2. 系统按照时间顺序给列出备用语料清单 3. 管理员选择一个备用语料库进行删除 4. 系统显示提交成功，删除该备用语料
扩展流程	无
特殊需求	1. 删除备用语料时需要给出二次提示 2. 删除每条语料的系统响应时间不超过 1s

评测者开始评测之前，评测系统会请求题目系统生成试卷，系统将会综合

评测者的个人用户画像、偏好选择、评测历史以及题库自身的情况选取题目组合成试卷并返回。评测题组选取的用例描述如表3.6所示。

表 3.6: 评测题组选取用例描述

描述项	**说明**
名称	评测题组选取
参与者	评测系统
优先级	高
触发条件	评测者开始了一次表达力评测过程，评测系统正常请求生成题组
前置条件	评测者已经通过了系统验证，符合当前系统的评测条件，包括剩余评测次数和会员到期时间等
后置条件	对应试题入库
正常流程	1. 评测系统以用户信息和用户偏好选择作为参数，发送评测题组选取请求 2. 系统综合分析之后，将选取好的评测题组返回
扩展流程	无
特殊需求	1. 选取题组的系统响应时间不超过 1s

在评测者回答了一道题，提交了自己的答案或音频文件之后，评测系统会提交一个分析任务到任务队列中，系统需要根据用户的提交分析用户的表达能力，并将成绩与分析结果一同入库。成绩分析的用例描述如表3.7所示。

表 3.7: 成绩分析用例描述

描述项	**说明**
名称	成绩分析
参与者	评测系统
优先级	高
触发条件	在表达力评测过程中，评测者提交了某道题的回答
前置条件	音频文件已归档，评测者的回答的元信息已经入库
后置条件	分析结果入库
正常流程	1. 用户在评测过程回答了一个问题，提交该问题之后，评测系统保存回答元信息，发送任务分析请求 2. 系统分析用户回答，返回成绩和分析结果。
扩展流程	2a. 分析过程中出现了无法分析等错误 1. 系统重试数次相同操作 2. 如果重试失败，系统需要返回对应的错误码和错误信息
特殊需求	1. 分析过程的系统响应时间不超过 5s

备用语料是题目生命周期建设的起点，也是绝大多数题目的源头。在定时任务模块的帮助下，系统可以定期执行网络任务，从网络上爬取资源，经过简单的初始化后生成备用语料。备用语料新增的用例描述如表3.8所示。

表 3.8: 备用语料新增用例描述

描述项	说明
名称	备用语料新增
参与者	定时任务模块
优先级	中
触发条件	在规定时间内，定时任务模块触发任务
前置条件	无
后置条件	新增备用语料入库
正常流程	1. 定时任务模块启动备用语料新增任务 2. 系统从配置中心读取任务参数，执行爬虫任务并将原始素材做简单初始化，形成备用语料 3. 系统将新生成的备用语料入库
扩展流程	无
特殊需求	无

随着使用次数增多，题目会积累用户数据。题目优化过程利用这些用户的作答数据，执行一定的算法任务优化题目的相关参数，提升题目的稳定性。题目优化的用例描述如表3.9所示。

表 3.9: 题目优化用例描述

描述项	说明
名称	题目优化
参与者	定时任务模块
优先级	高
触发条件	在规定时间内，定时任务模块触发任务
前置条件	无
后置条件	优化后的题目入库
正常流程	1. 定时任务模块启动题目优化任务 2. 系统将遍历题目列表，按照类型的不同执行不同算法优化题目 3. 系统将优化后的题目入库
扩展流程	无
特殊需求	无

3.3.3　功能需求

根据系统的用例分析，可进一步进行功能需求分析。本系统的主要功能需求如表3.10所示。

表 3.10: 功能需求列表

ID	需求名称	需求描述
R1	成绩分析	评测者完成评测之后，系统需要根据题目和评测者提交的回答，分析并计算出用户的表达能力水平
R2	题目优化	为了保证题目模型稳定性，提升题目分析的准确度，系统需要定时优化题目的参数
R3	语料新增	为了保证题目数量和题目来源多样性，系统需要定时从网络中爬取合适资源，形成备用语料
R4	评测题组选取	评测者在参与评测之前，系统需要从题库中选取合适的题目组成试题提供给评测者使用
R5	语料删除	当管理员认为一个备用语料不适合作为题目时，管理员可以删除它
R6	题目新增	管理员可以从备用语料中选取素材，新增题目到正式题库中，也可以直接新增题目
R7	题目列表查看	管理员可以查看题库中的所有题目，这些题目需要能够按照风险程度排序，以便帮助管理员做出决策
R8	题目修正	管理员可以选定题目，修改其文本内容，难度等参数

3.3.4　非功能需求

表达力评测解决方案是面向商用的正式应用系统，本系统作为解决方案的核心部分，系统的可用性、性能、可扩展性等质量属性须达到较高要求。结合用例分析一节，系统关注的质量属性及其要求如表3.11所示。

表 3.11: 非功能需求列表

质量属性	需求描述
可用性	1. 系统需要保证 99% 的时间内完全可用 2. 系统数据库及文件库需要做好主从备份，并且能够及时切换备份，保证数据访问 3. 更新系统时要保证服务不中断
易用性	1. 针对可能出现的所有错误，系统需要分别给出错误提示并指引用户修正 2. 所有表格若无特殊说明，均需提供筛选，分页等功能 3. 需要等待超过 2s 的页面均需页面提示，必要时需要提供进度百分比提示
可扩展性	对于可能出现的新功能，例如上线新的活动场景等，系统需要新增的工作量不超过 10 人日
可伸缩性	系统的所有模块需要支持服务器弹性扩容以应对突发的流量高峰
可修改性	对于系统原有功能的修改，例如取消邀请，全面公测等，系统需要修改的工作量不超过 5 人日
性能	1. 所有页面若无特殊说明，系统响应时间不超过 1s 2. 分析子系统分析问题的吞吐量需要达到 10 个/s，并支持扩容

3.4　系统概要设计

本节将从结构方面对系统进行设计。首先，本节会介绍系统的架构和模块划分，说明每个模块需要完成的主要任务以及模块之间如何协作。随后，本节会以“4+1”视图模型对系统从不同角度进行体系结构设计。最后，本节简要介绍了系统的数据库实体设计。

3.4.1　系统架构设计

题目系统的系统架构图如图3.3所示。系统的客户端包括微信小程序端和网页端，其静态文件由 CDN 进行缓存，以提高客户端下载速度。后端是去中心化的服务器节点群，其中分析模块单独部署在计算节点上。下面对各个模块进行详细描述。

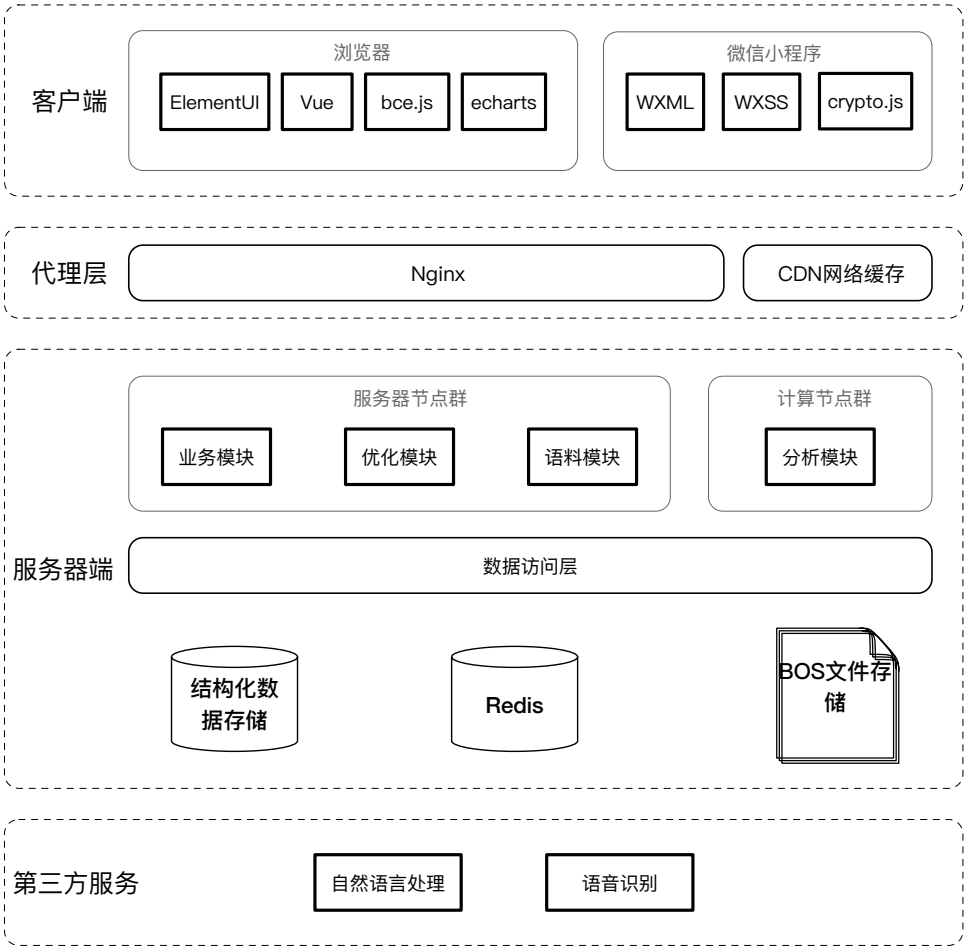

图 3.3: 题目建设与分析系统架构

系统网页端页面模板使用 D2Admin，这是一个基于 Vue 的 MVVM 模板方

案，使用了最新的前端技术栈，能够完成相对复杂的前端逻辑。UI 组件库选择了较为成熟的 ElementUI，它提供了预定义好的样式，以便快速完成设计与开发。前端的静态请求均由内容分发网络提供了缓存，提升了资源的下载速度，而动态请求均应用 Axios 组件进行，可以有效降低页面的延迟，避免重复刷新。数据可视化方面，前端采用了较为成熟的 Echarts 解决方案，它可以进行深度定制以实现复杂的图形展板。

为了便于应用科学计算库，系统的服务端全部使用 Python 语言进行开发，并采用 Flask 框架对外提供 Http API，与前端进行格式化数据交互。对于一般的请求，业务模块会通过数据访问层底层的数据库进行交互，完成请求并返回。对于评测系统发出的分析请求，分析模块会异步地处理这个任务，处理完毕之后，分析模块会将结果存储在 MongoDB 中，并修改 Redis 中的任务状态，等待评测系统轮询。

前端与服务端之间采用 Nginx 的反向代理功能进行路由转发，Nginx 的负载均衡功能可以让服务端实现较好的水平扩展，能够保证在服务高峰期动态添加服务器资源。服务端采用 Redis 作为缓存存储，用于存储用户 session 和任务数据。Redis 缓存可以让数据在多个系统，多个节点之间共享，并且不需要频繁的数据库连接过程。持久化存储方面，系统将结构化数据存储在 MongoDB 数据库中，非结构化数据存储在对象存储服务器中。

3.4.2　系统模块划分与模块职责

从图3.3中可以看出，系统主要分为业务模块、优化模块、语料模块以及分析模块这四个部分。在一个题目的生命周期中，这四个模块所扮演的角色如图3.4所示。语料模块为题目提供了原始素材，根据配置中心下发的任务参数，定时从网络资源中爬取语料，经过简单的分析整理之后，形成了供管理员借鉴的文本素材。分析模块对外输出的主要功能是成绩分析功能，在评测完成之后，分析模块将根据题目和评测者的回答分析出用户的表达能力水平。优化模块将使用用户的回答数据来优化题目的参数，回答数据越多，优化地越准确。业务模块为题目和备用语料提供了业务访问操作，评测题组选取，题目列表查看，手动修改题目参数等业务功能均为业务模块的职责。

从图3.4中可以看出，语料模块从网络上爬取并初始化题目，随后，在不断的用户评测，回答的过程中，用户数据不断进入系统，题目将经历分析-优化-再分析的循环过程，其中管理员可以通过业务模块监控系统行为，进行人工修正。在这一闭合模型中，题目模型的稳定程度会越来越高。

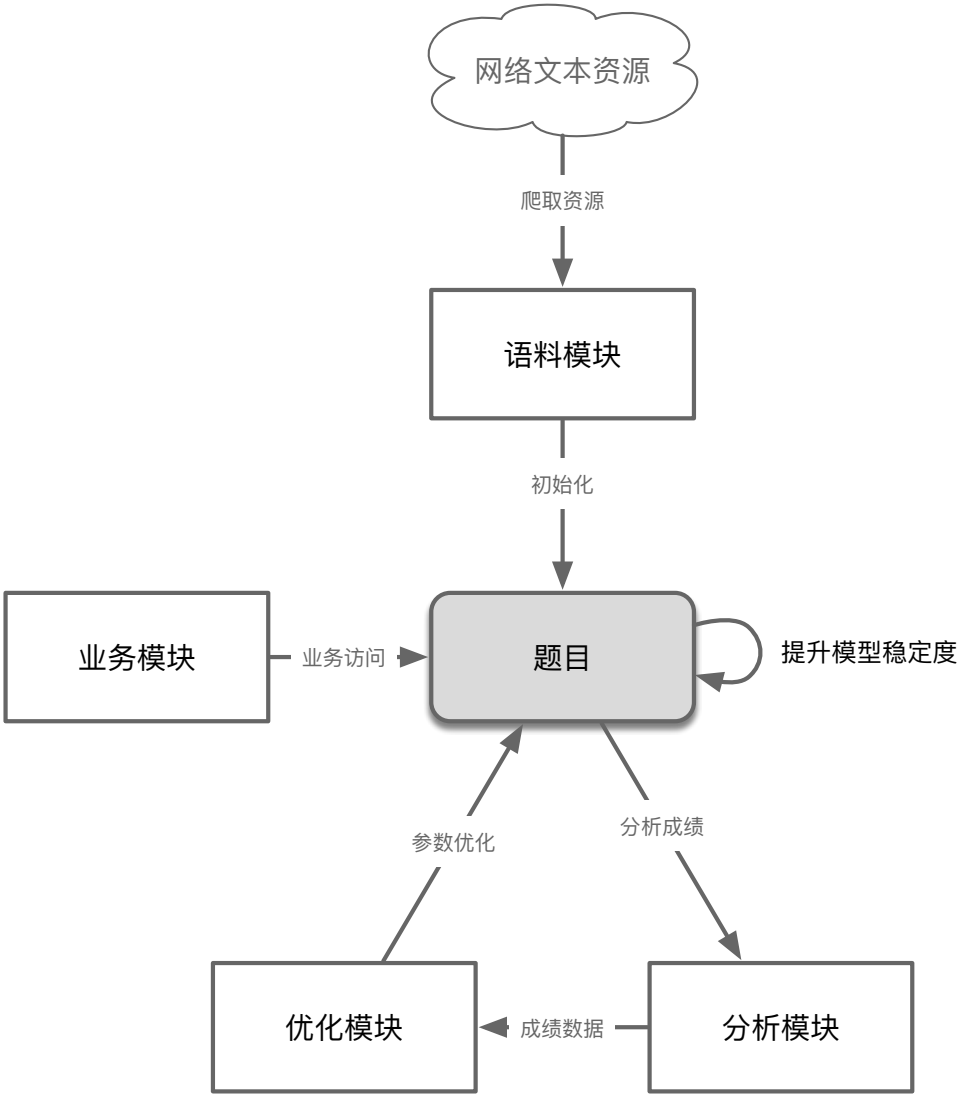

图 3.4: 模块的职责分配

3.4.3　4+1 视图

“4+1”视图设计是软件架构领域广泛应用的一种设计方式，它最早由 Philippe Kruchten 于 1995 年提出 [45]。“4+1” 视图的核心 “1” 指的是用户使用场景，即系统用例，这一部分在3.3.2节中已经详细叙述过，其余的 “4” 代表逻辑视图、开发视图、进程视图以及物理视图。本节将从这四个视图出发，从不同角度阐述系统的体系结构。

逻辑视图需要从功能性上描述系统，即面向用户的视图。如图3.5所示，系统的核心由六个包组成。controller 包提供了与前端交互的 REST 风格的 API 接口，这个包基于 Flask 的路由转发实现。process 包是定时任务包，它由解决方案公共的定时任务模块控制，定时启动某些任务。task 包将会从分布式任务队列中

获取任务，完成任务之后将结果存储到数据库中。service 包将提供各个实体类具体的业务逻辑，例如查询、新增、排序等。algorithms 包封装了其他包所需要的算法功能，包括第三方算法和自实现算法。data 包封装了底层数据的访问操作，包括结构数据库与非结构数据存储。

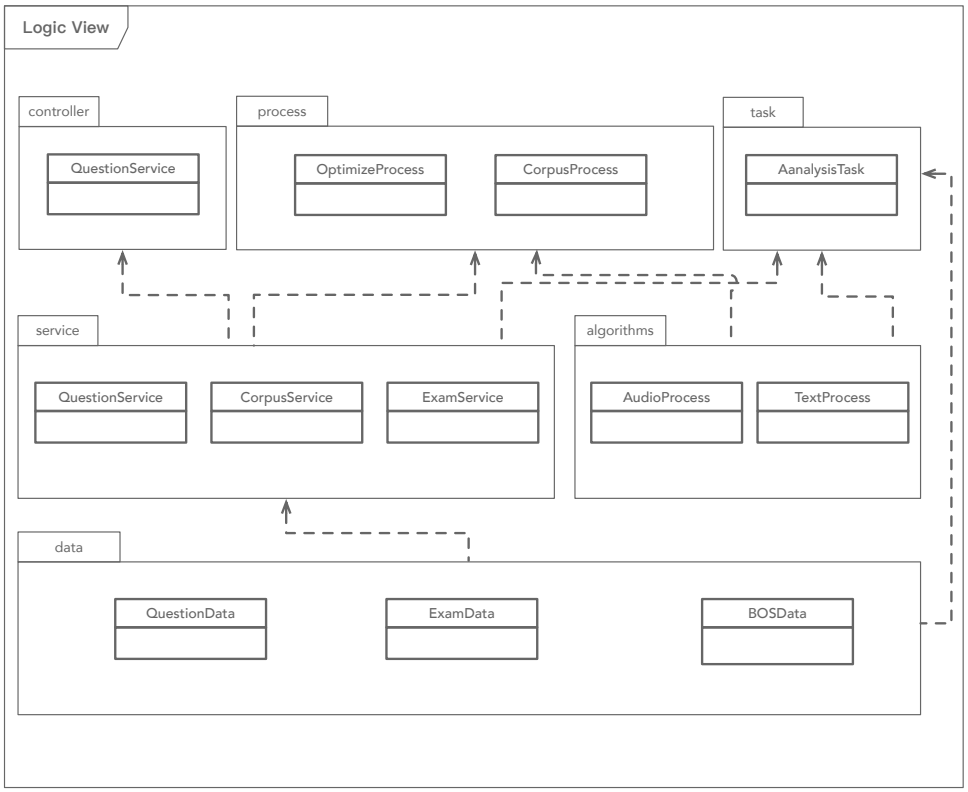

图 3.5: 逻辑视图

开发视图描述了开发系统需要完成哪些任务，即面向开发者的视图。如图3.6所示，系统主要分为客户端和后端两个部分。系统客户端主要包括题目列表查看，题目新增和题目修正这三个页面，以及部分的错误提示页和等待页。系统后端分为四个主要的模块，即业务模块，优化模块，语料模块和分析模块，其中分析模块又分为特征提取、分数计算、总分核算和处理算法几个子模块。此外，系统还包含了共用的工具组，包括文件读写和时间操作等。存储方面，Redis 缓存数据，MongoDB 数据库和 BOS 对象存储均需要对应的访问接口，以向上屏蔽底层细节。

图 3.6: 开发视图

进程视图表述了系统内部相关进程与线程实体及其通信过程，是从动态角度描述系统的视图。如图3.7所示，用户评测完成之后，评测系统会向任务队列中添加一个任务，而分析模块进程会获取这个任务并开始分析。首先分析模块进程会从数据库进程中获取此次评测的实体，并从中获取音频文件地址。之后，分析模块会从对象存储服务进程中下载音频文件并进行分析。随后，分析进程将分析结果存储到数据库中，等待评测系统轮询获取。图中还画出了优化的过程，定时任务进程会定期启动优化任务，任务完成之后，该进程会把优化结果存储到数据库中。

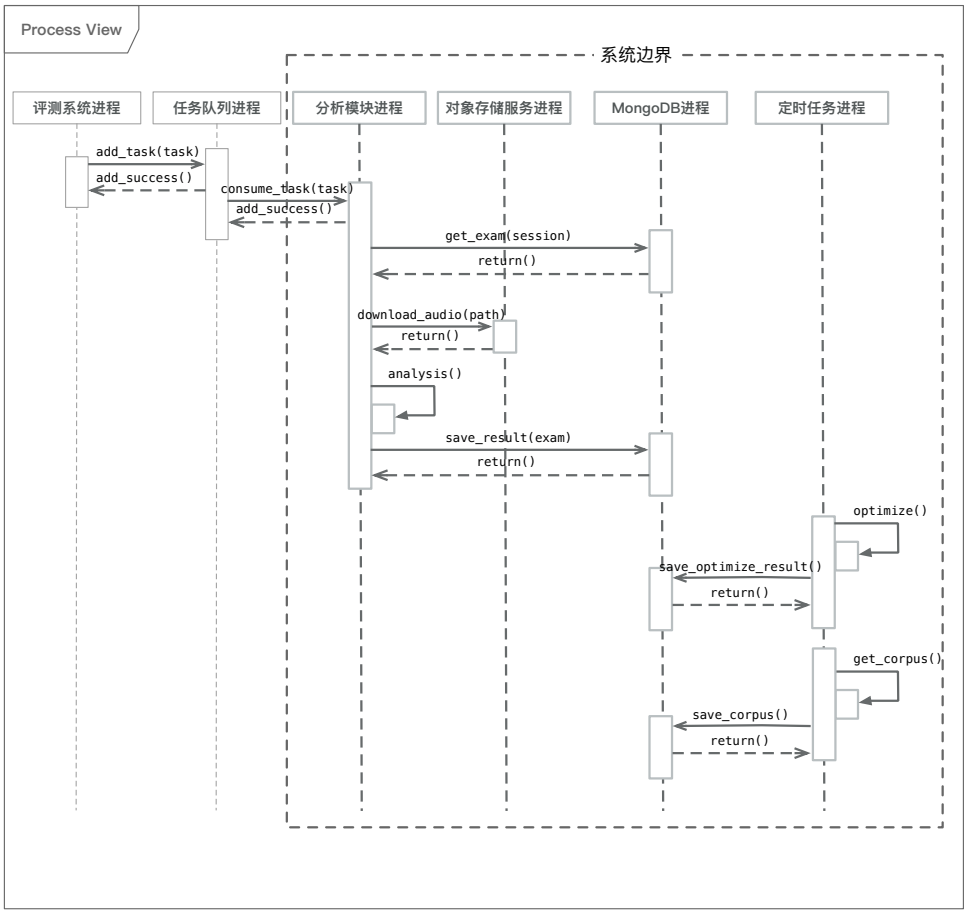

图 3.7: 进程视图

物理视图面向运维人员视角，主要表述了系统物理节点之间的通信和部署情况。如图3.8所示，前端 app 部署在专门的前端服务器上，并使用 CDN 缓存加速访问。后端的部署遵循两条原则：一是对外同步服务与对内异步任务分离，因此，后端将业务模块、语料模块和优化模块部署在服务器节点上，而分析模块单独放在计算节点上，为了保证系统的可伸缩性，服务器节点和计算节点均可以弹性扩展；二是大规模的数据存储与系统计算分离，因此系统将 MongoDB 数据库单独部署在服务器上，并购买了百度云的对象存储服务，用于存储非结构化的文件。系统各个模块的自动集成与自动部署详见5.1节。

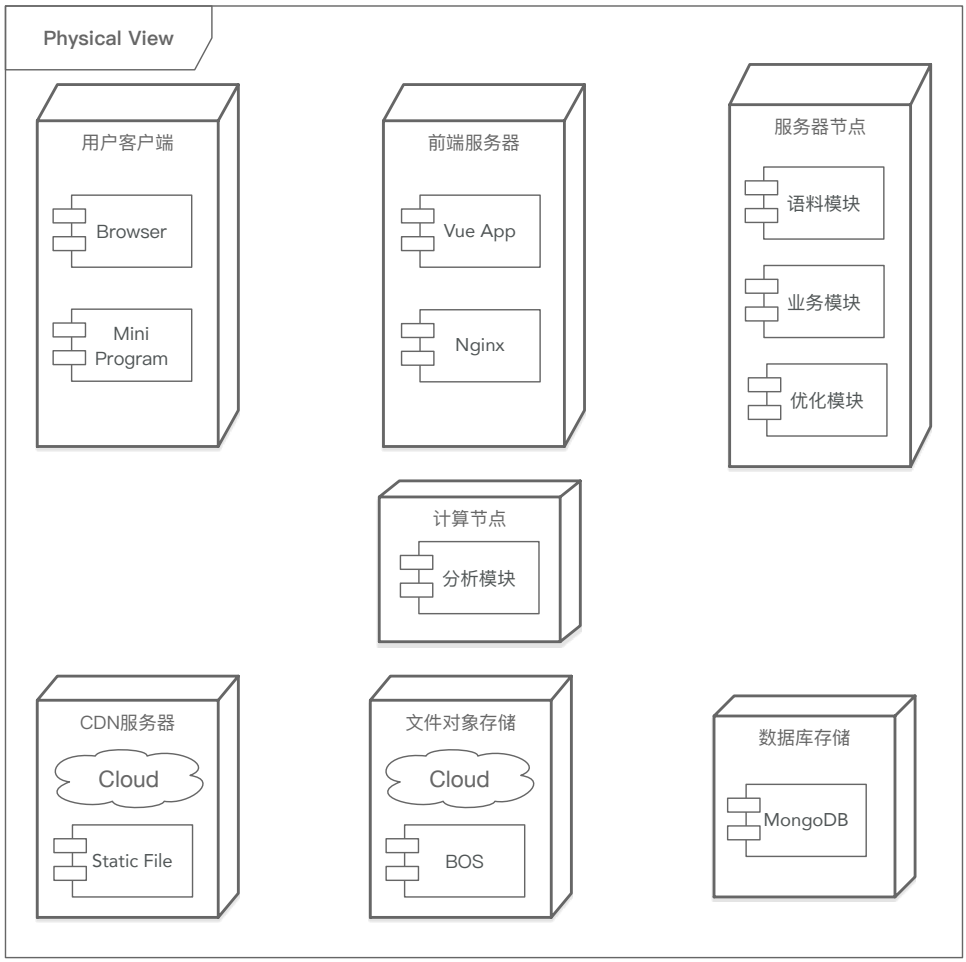

图 3.8: 物理视图

3.5　数据库设计

表达力评测项目的结构化存储采用了非关系型数据库 MongoDB。MongoDB 是一种文档型的非关系型数据库，相比较传统的关系型数据库，NoSQL 的好处在于不需要满足关系型数据库繁琐的关系需求，存储的数据类型更加灵活，能够存储的数据上限也更多。

非关系型数据库的的设计通常以星星模型为思路展开，即首先确定核心对象，从核心出发，思考其内部对象以及核心对象与内部对象之间的对应关系，随后以内部对象为核心对象迭代上述过程，直至完成整个数据库 schema 的设计。本系统的数据库设计如图3.9所示。其中深色边框为 MongoDB 中的文档（document），浅色边框的是内部对象，用枚举（Enum）标记的对象是枚举类型，只能在有限范围内取值。

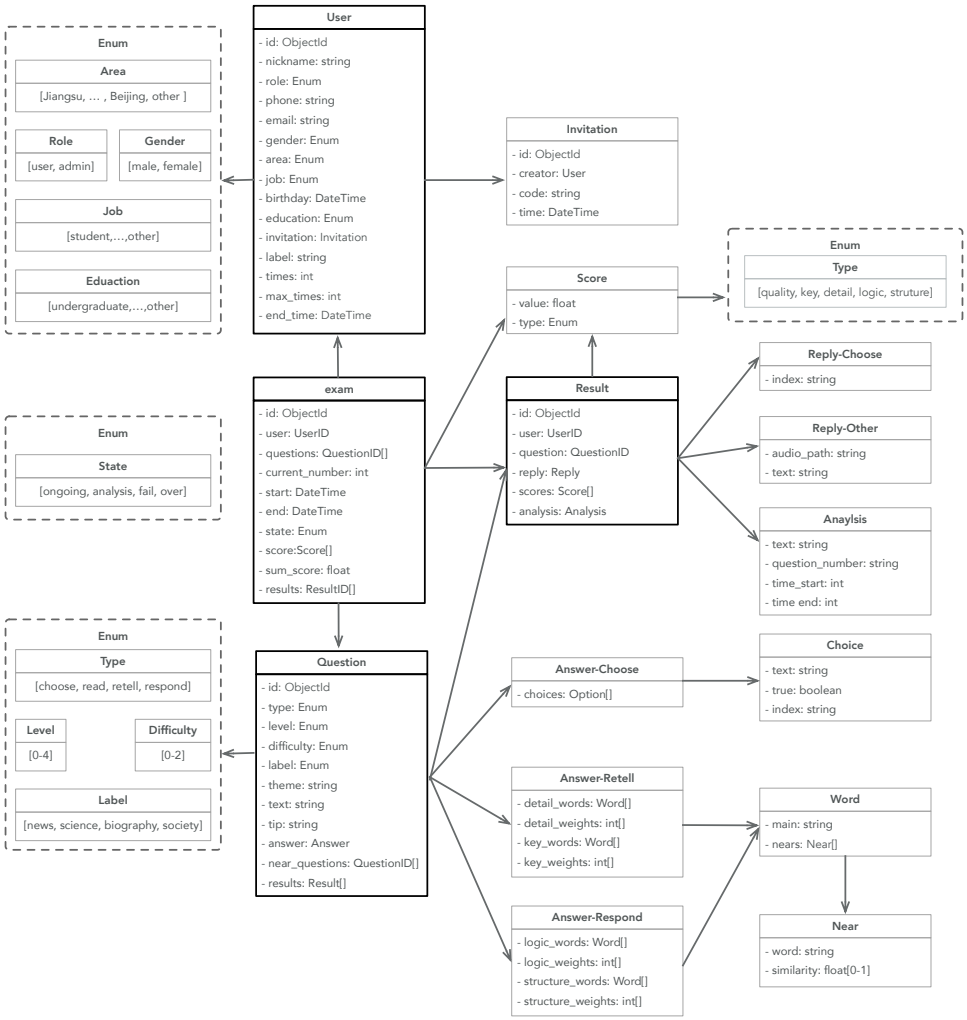

图 3.9: 数据库设计

从图3.9可以看出，核心实体包括题目（Question）、考试（Exam）、用户（User）、分析结果（Result）这四个，下面详细叙述这四个核心实体及其关键属性。

题目是解决方案的核心数据资产，也是系统主要管理的对象。题目的关键在于其繁杂的参数，这些参数也是分析模块和优化模块的主要关注点。题目实体的关键属性如表3.12所示，值得注意的是，备用语料是特殊的题目，它的级别属性为 0。

表 3.12: Question 实体的主要字段

字段	字段类型字段含义	
type	Enum	题目类型，目前有朗读题，复述题和问答题
level	Enum	题目级别,目前范围从 0-4,表示题目模型的稳定程度,详见4.1.3节
difficulty	Enum	题目难度，目前范围从 0-2，详见4.1.3节
label	Enum	题目标签，目前有新闻类，科普类，传记类和社会类
text/tip	String	题目文本内容和提示
answer	Object	题目答案，选择题答案即正确的选项列表，复述题和问答题答案主要由词库组成，详见4.1.2节
nears	QuestionID[]	邻近的问题集，其更新详见4.2.1节
results	ResultID[]	本题所有的分析结果集

结果实体是系统关注的重点对象，它代表了用户对某个问题的一次回答以及系统给出的成绩分析，优化模块将主要使用题目的用户数据来实现题目的优化。结果实体的关键属性如表3.13所示。

表 3.13: Result 实体的主要字段

字段	字段类型字段含义	
user	userID	成绩对应的用户
question	QuestionID	成绩对应的题目
reply	Object	用户对应的用户回复，选择题的回复是简单的选项，语音题的回复包括了音频文件地址和识别的文本等
scores	Object[]	最终分数，分数分为音质，主旨，细节，逻辑，结构五种
analysis	object	成绩对应的分析，包括了简要评语，对应题号和时间段

考试实体表示一次实际的评测过程，是评测系统关注的主要对象，本系统将主要关注考试的分数属性是如何计算获得的。考试实体的关键属性如表3.14所示。

表 3.14: Exam 实体的主要字段

字段	字段类型字段含义	
user	userID	考试对应的用户
questions	QuestionID[]	评测对应的题组
cur_num	int	考试当前进行到题号
start/end	DateTime	考试的开始时间和结束时间
state	Enum	考试的状态，包括正在进行，正在分析，失败，完结这四种
scores	Object[]	考试对应各个维度的分数
sum_score	float	评测的总分
results	Object[]	评测对应的各个题目的分析结果

用户实体表示系统的一个使用者，通常来说是一个评测者，是评测系统关注的主要对象。在本系统中，以用户属性为基础的用户画像将部分参与评测题组的选取。用户实体的关键属性如表3.15所示。

表 3.15: User 实体的主要字段

字段	字段类型字段含义	
gender	Enum	性别，目前有男，女，保密
area	Enum	地区，目前包括国内的省级行政区和其他地区
job	Enum	职业，包括了学生，老师，记者等主要职业
education	Enum	学历，目前包括本科，研究生，博士，其他
label	String	标签，用于表示用户组，同一个用户组的用户通常具有某种联系，详见4.4.1节

3.6　本章小结

本章首先使用架构图的方式叙述了表达力评测项目的总体规划，指出了题目系统在项目中所处的位置。接着，本章进行了涉众分析和需求分析，通过用例说明的形式分析了系统的需求，列出了系统的功能性需求和非功能需求。然后，本章介绍了系统的总体设计，将系统划分为分析模块、优化模块、语料模块和业务模块，并详细叙述了这些模块的职责。随后，本章以 4+1 视图的形式，详细介绍了系统的体系结构设计。此外，本章还介绍了系统的数据库设计，列出了各个实体的核心字段。

第四章　系统详细设计与实现

本章将在系统概要设计的基础上，采用自上而下分解的方式，将系统划分为模块和子模块（流程），并对它们做出详细设计与实现。

4.1　分析模块详细设计与实现

分析模块是题目建设与分析系统的核心部分，也是整个表达力评测项目区分于其他类似评测系统的核心功能。在评测流程中，用户每完成一道题，评测模块都会添加一个任务到 Celery[1]分布式任务队列中，任务信息包括了本次评测对应的考试信息，题目信息以及用户回答这三个重要信息。之后，部署在多个计算节点上的分析模块会竞争式处理这个任务。

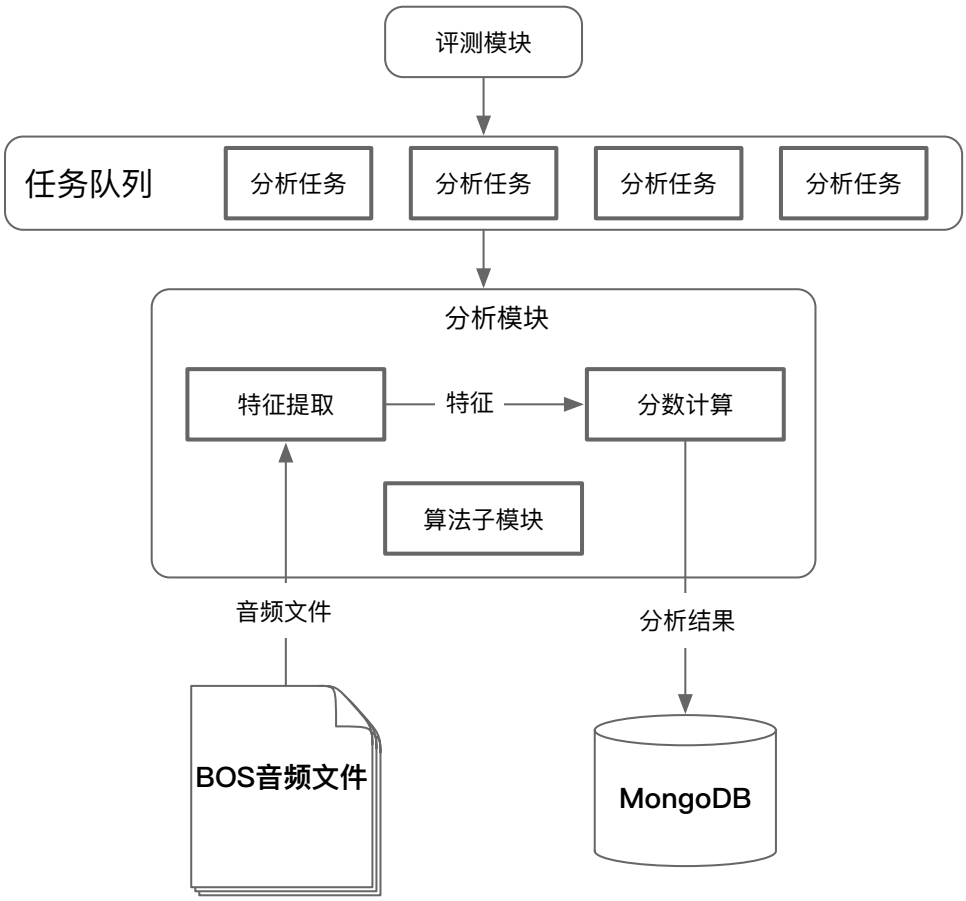

图 4.1: 分析模块分解

[1]https://github.com/celery/celery/

题目分析的主要流程是：首先，分析模块会根据用户回答中的音频下载地址，从 BOS 文件存储中下载音频文件到本地；接着，分析模块会对音频文件进行简要的清洗剪辑，再进行特征提取；其次，分析模块会调用分数计算函数。对于不同类型的题目，分数计算函数会采用不同的模型来分析本次回答，并将结果记录至数据库中；最后，分析模块会判断整个评测过程的所有题目是否已经分析完毕，并调用总分核算子模块来对整个评测进行总分的计算。

从上面的论述中可以看出，分析模块有三个主要的子流程，即特征提取、分数计算和总分核算。此外，这三个流程中涉及的文本和语音算法被单独封装在了处理算法子模块中。图4.1表述了分析模块的分解以及它在整个系统中的位置，4.1.1节，4.1.2节，4.1.3节和4.1.4节将详细叙述这些子模块。

4.1.1　特征提取

用户回答问题之后，客户端会收集用户的音频文件，并上传到指定位置。特征提取子模块将以音频文件为数据源，尽可能多的提取有用的特征。音频文件是没有经过压缩的 wav 文件，实际上是一长串的数字。特征提取的过程如图4.2所示。

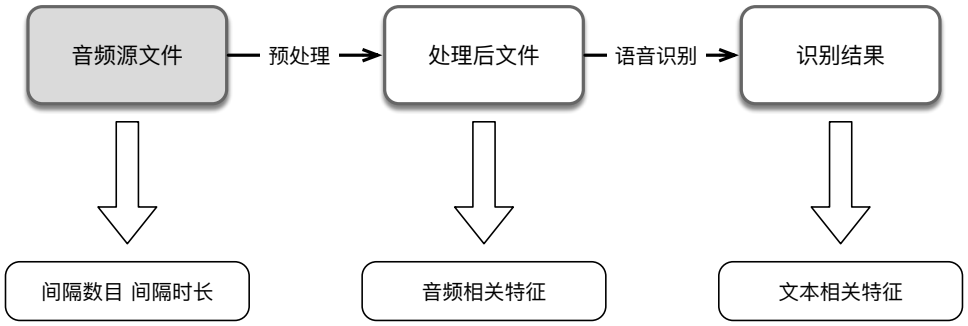

图 4.2: 特征提取流程

从图4.2中可以看出，特征提取主要涉及的过程有四个：去除前后空白和中间间隔，这一步的主要目的是清洗音频数据以便后续步骤，并且可以计算做题过程中的停顿数，进而反应评测者的音质；提取音频相关特征，包括时长，频谱，音量，情感波动等，这些特征可以直接用于分数计算；语音识别文本采用了商用接口方案，这一步需要将音频文件转化为文本；提取文本相关特征，包括识别结果，语速趋势，词性分布，助词频率，清晰度，词语击中向量等。

值得一提的特征是词语击中向量，在转述题和问答题任务中，系统将提取这一特征。词语击中的基本想法是考察评测者的回答文本中是否包含某些特定的词语，包含时为 1，不包含时为 0。而针对一组词语（词库），评测者的回答可

以抽取出一组词语向量。例如评测者回答“高铁是一个现代化的交通工具”，如果词库是【高铁，交通，汽车】，该评测者的回答的词语击中向量将是【1，1，0】。词语击中向量特征的使用详见4.1.2节。由于商用接口技术和评测者环境的限制，评测者音频的文本识别可能并不会十分准确，即出现可能的“谐音”现象。例如评测者的回答可能会被识别成“糕点是一个现代化的搅动工具”，在转述题的上下文语境下，系统将使用谐音击中的方式，认为该回答仍然击中了高铁和交通这两个词语，最终得到的词语击中向量仍然是【1，1，0】，这样的设计可以尽可能的减少商用接口错误识别引发的问题。谐音击中的实现详见4.1.4节。

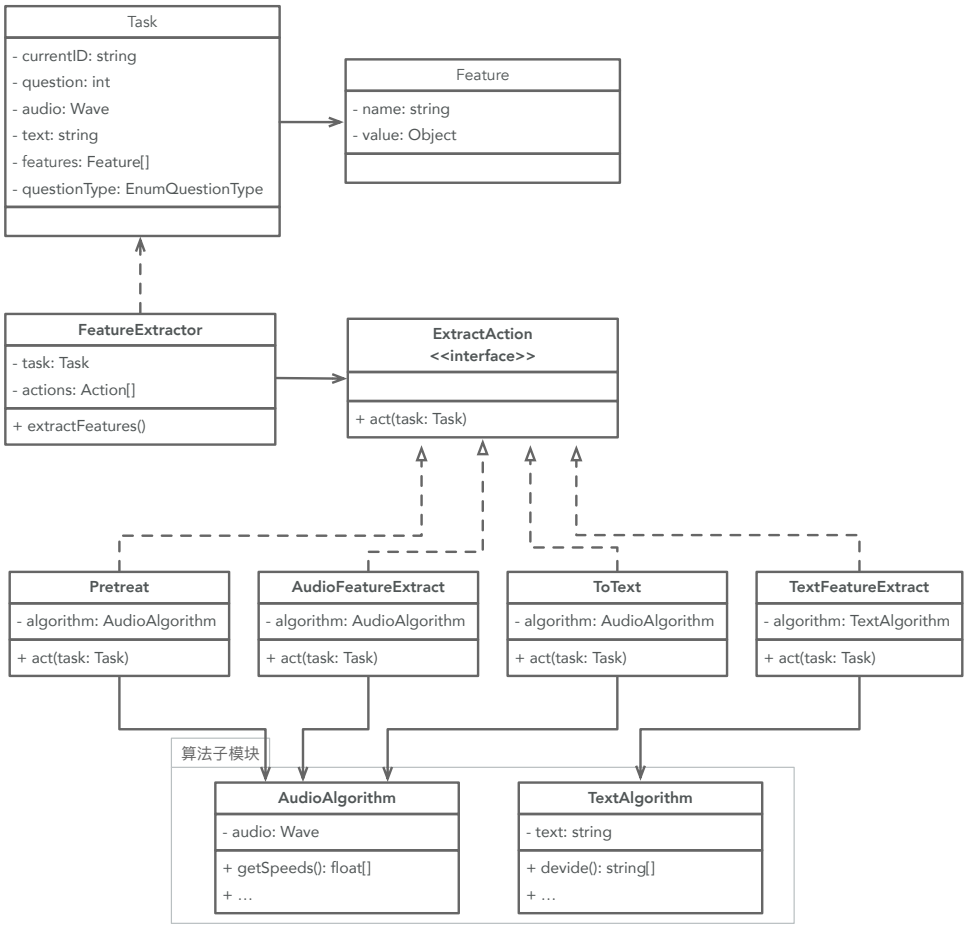

图 4.3: 特征提取类图

图4.3是特征提取子模块的核心类图。类 FeatureExtractor 是特征提取的核心类，它接受任务 Task 和提取过程数组 actions 作为构造参数，方法 extractFeatures() 逐个调用各个提取过程，并统一上报可能出现的错误。特征提取的具体工作交给了 ExtractAction 及其子类，分别是预处理（Pretreat）负责剪辑音频文件中的前后空白，并去除停顿；音频特征提取（AudioFeatureExtract）负责提取音频

特征；语音文本识别（ToText) 负责将音频识别为文本；文本特征提取（TextFeatureExtract）负责将提取文本特征。在这些子类中使用的具体的音频算法和文本算法详见4.1.4节。

提取特征的核心代码如图4.4所示，略去了错误信息处理的细节。FeaturesExtractor 类获取任务信息之后，会初始化特征提取步骤，并依次调用处理步骤的 act 方法。特征提取完毕后，具体的工作类会将特征存储在 task 的 features 对象中，以便后续工作使用。

```
class Pretreat(ExtractAction):
    def act(self, task):
        pass # 此处省略预处理

class ToText(ExtractAction):
    def act(self, task):
        rcg_text = ""
        # 此处省略将task中的音频转化为文本
        task.features.append(Feature("rcg_text", rcg_text))

class FeatureExtractor:
    def __init__(self, task):
        self.task = task
        self.actions = []
        self.actions.append(Pretreat())
        self.actions.append(ToText()) #省略添加其余处理类

    def extract_features(self):
        for action in self.actions:
            action.act(task=self.task)
```

图 4.4: 特征提取核心代码

4.1.2　分数计算

分数计算子模块将使用特征提取子模块得到的特征计算本题的分数。对于不同题目类型，模型关注的特征也不同，其最终考察的表达能力维度也不同。系统的题目类型分为朗读题、转述题、问答题这三种，并将表达力的评价维度量化到音质分、主旨分、细节分、逻辑分、结构分这五种分数上。朗读题即朗读屏幕上的文字，要求字正腔圆，含有感情，主要为了评价评测者的音质是否饱满，是否悦耳动听，语速是否均衡；转述题将一段文字展示给评测者看，较短时间后，这段文字会消失，并要求评测者复述之前的文字，这种题型主要为了评价评测者的主旨把握能力和细节还原能力；问答题将抛出一个开放式的问题要求评测者回答自己的看法，主要为了考察评测者在表达时能否灵活运用各种逻辑技巧，是否具有较为严谨的表达结构。

对于朗读题来说，系统更关注它的音频相关特征，例如语速，频率分布等，

因为这些特征更能准确表达评测者的声音质量；而对于转述题和问答题，系统更关注它的文本相关特征，例如词语击中向量、虚词比例等，因为这些特征能够反映评测者是否准确地点出了某个概念。下面将分题目类型叙述其分数计算模型。

朗读题的判分逻辑采用专家规则来判断，原因主要有三点：一是朗读题题量不需要很多，因此不需要针对每道题使用不同的模型；二是朗读题评分应用的特征数目较少，并且特征比较浅显，不需要深度挖掘特征与结果之间的关系，例如清晰度越高，音质分相应越高；三是朗读题题目难度比较容易判断，基本不需要进行动态调整。表4.1叙述了朗读题分数计算的相关逻辑。

表 4.1: 朗读题分数计算规则

核算步骤	**核算逻辑**
初始化	初始化为满分 100
清晰度核算	清晰度越低，扣分越多，有上限
无效表达率核算	无效表达率越高，扣分越多，有上限
时长核算	在一定区间以外的时长均需要扣分
音量核算	在一定区间以外的时长均需要扣分，需要记录音量作为分析项
语速核算	语速列表的方差表示语速的变化，方差超过一定值的需要扣分，需要记录语速作为分析项
停顿核算	停顿数目越多，扣分越多，有上限，需要记录停顿做为简要分析项
频谱核算	频率分布在一定区间之外的需要扣分，需要记录频谱作为分析项
情感波动核算	情感波动超出一定阈值的需要扣分，需要记录情感波动作为简要分析项
完成度核算	完成度表示评测者对整个文本朗读的完成情况，最后需要将剩余分数乘上完成度
兜底核算	低于 0 分的核算为 0 分

转述题的考察重点在于评测者是否能够准确的转述事情，考察评测者在转述过程中能否把握素材的主旨和细节。转述题也考虑了部分音质问题，即在转述过程中，是否有较多的停顿等。与朗读题不同的是，转述题的题目中包含了词库参数和词语权重向量参数，其中词库参数用于在特征提取子模块中提取出词语击中向量特征，而词语权重参数和题库参数对应，表示某个词语击中之后，应该获得多少对应的分数。词语权重向量参数作为题目的一个重要参数，其优化算法将在4.2节中叙述。

主旨分和细节分的评价主要参考了文本相关特征中的主旨词语击中向量，这个特征较好地反映了用户是否提到了某些概念，从而反映评测者对原素材的主旨理解和细节还原是否到位。具体的做法如公式4.1所示，其中 H 表示击中向量，即评测者说到了那些关键词，W 表示权重向量，即这些关键词对应的分数

是多少。

$$Score = 100 \times \frac{H \cdot W^T}{\sum_i^n W_i} \tag{4.1}$$

图4.5是一个典型的主旨分计算过程，从图中可以看到，用户文本和题目的词库参数决定了用户的击中向量特征，而击中向量和题目的权重向量参数决定了用户最后的得分。

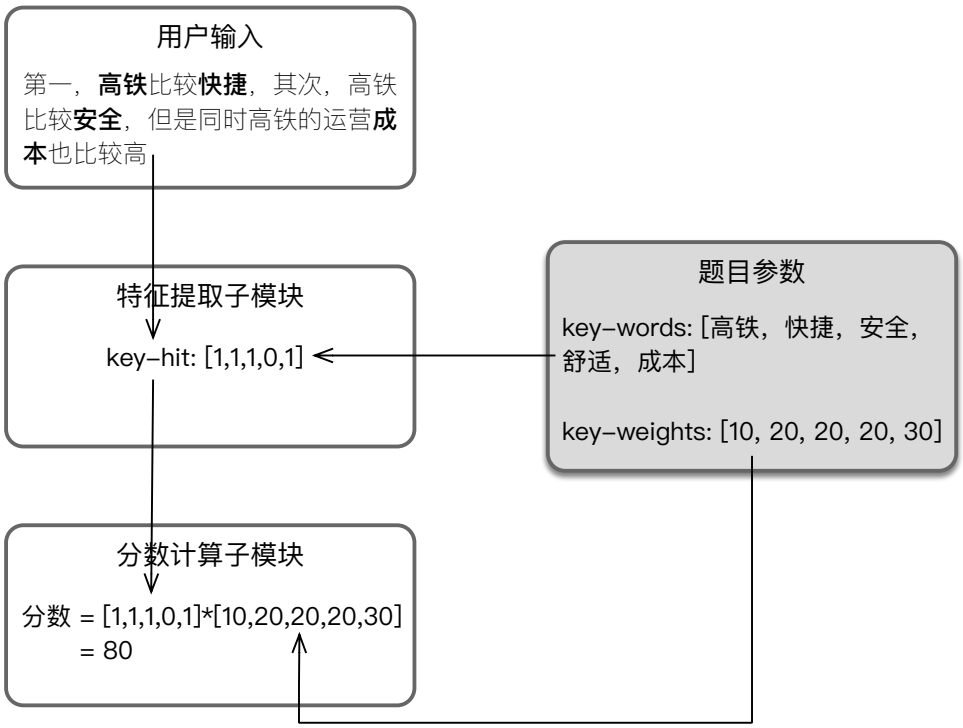

图 4.5: 分数计算过程实例

问答题的考察重点在于评测者是否能够较好的表达自己的想法，在表达过程中是否注重使用某些表达技巧，例如总分总式的表达顺序，能否用一二三四来分点论述，或者能够举出一些示例来辅助表达。问答题的分数计算逻辑与转述题类似，这里不再赘述。

由于不同题目类型的任务采取的分数计算策略并不相同，该子模块的设计采用策略模式，其核心类图如图4.6所示。ScoreComputer 类是分数计算子模块的核心类，它接受一个 Task 和一个 AnalysisStrategy（分析策略接口）作为构造参数，对外提供 scoreCompute 方法。在该方法中，系统会调用成员变量 AnalysisStrategy 的分析方法，依据具体的 AnalysisStrategy 实例，系统采取不同的分析策略，返回对应的结果。

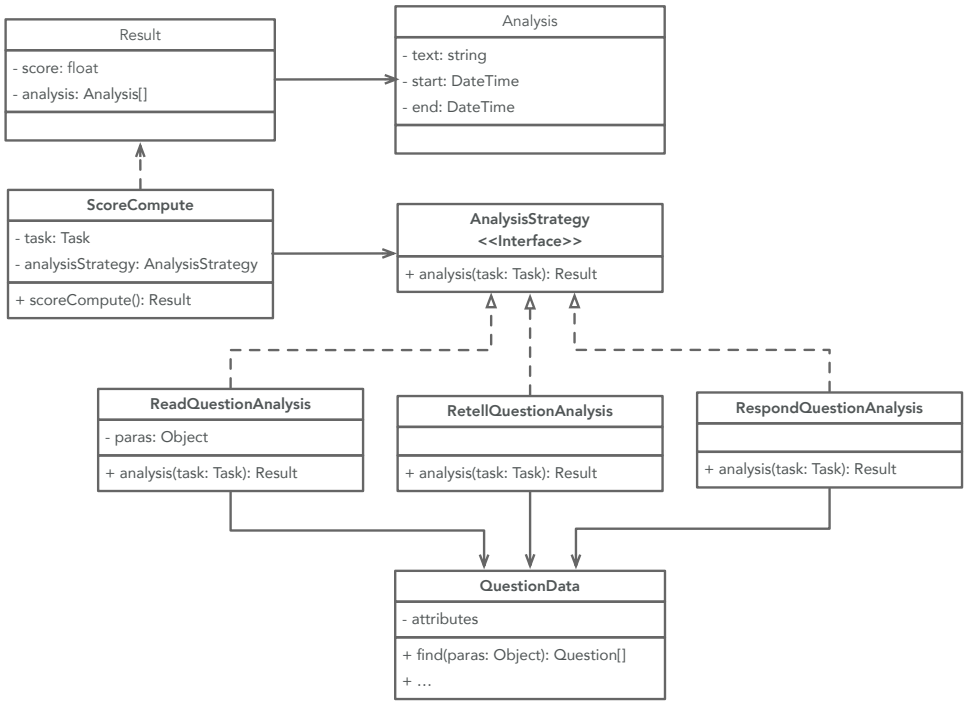

图 4.6: 分数计算类图

4.1.3　总分核算

在评测过程中，当分析模块处理完用户最后一个问题回答之后，分析模块将调用总分核算子模块来对整次评测的总分进行核算。如图4.7所示，在相同的判分规则下，不同题目的分数分布有着较大区别。由于大部分题目都是由机器自动生成，其词库参数，权重参数等不够准确，这影响了题目的稳定性。

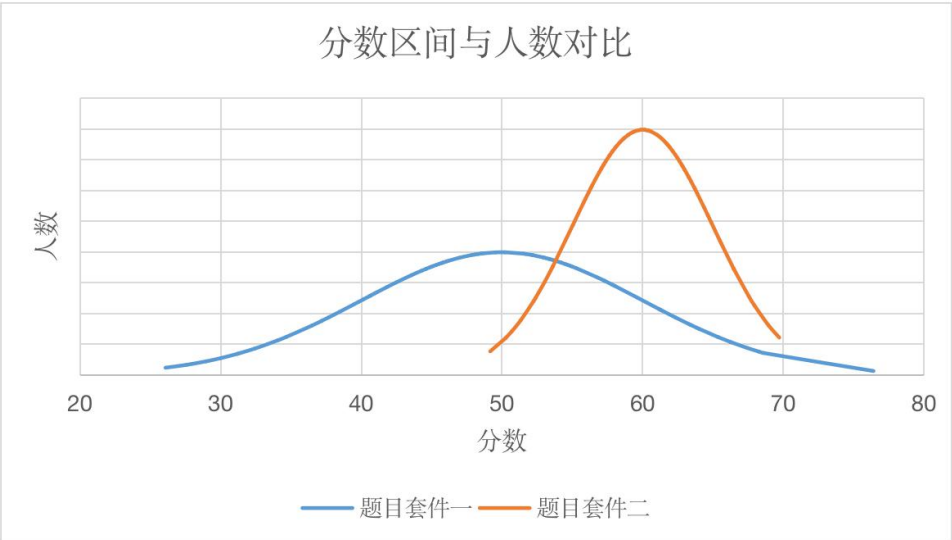

图 4.7: 不同题目的分数分布

事实上，在一次评测的题组中，不同题目之间并不“平等”，即这些题目的分数并不会“平均地”纳入最后的总分核算中。因此需要引入额外的参数来表示这种“平等性”，并进行额外的总分核算步骤。

为此，系统引入额外的题目参数级别。级别参数表示一道题的模型稳定性，其初始化为 0，表示极不稳定，不能使用，需要人为介入。题目级别越高，表示此题使用得较多且表现良好，模型较稳定，即越“值得信赖”，最高为 4。例如，级别为 1 的题目不参与计分，仅作为提升使用，其对应的系数为 0，级别为 4 的题目级别系数为 1。此外，不同题目难度差距会很大，按照同一标准计分会有所偏颇，系统引入题目的难度参数，难度最低为 0，最高为 2。

图4.8表示总分核算子模块的核心逻辑，即级别核算过程和难度核算过程，其中级别系数和难度系数由配置文件指定。从代码中可以看出，在一套题组中，每道题的分数会先以级别系数加权平均，再使用难度系数进行调和。

```
# 级别系数和难度系数
level_co = get_config('level_co') #[0, 0.5, 0.75, 1]
difficulty_co = get_config('difficulty_co') #[0.8, 1, 1.2]

class ScoreSum:
    def __init__(self, task):
        self.task = task

    def sum_up(self):
        sum_result = 0
        # 计算出这套试卷中所有问题的级别系数之和
        level_cos = []
        for q in self.task.questions:
            level_cos.append(level_co[q.level])
        level_co_sum = sum(level_cos)
        for question in self.task.questions:
            score = question.score
            level, difficulty = question.level, question.diffculty
            # 级别核算
            this_score = score * level_co[level] / level_co_sum
            # 难度核算
            this_score *= difficulty_co[difficulty]
            sum_result += this_score
        return 100 if sum_result > 100 else sum_result
```

图 4.8: 总分核算核心代码

4.1.4　处理算法

处理算法子模块对外提供了分数计算过程中需要的算法函数，主要包含音频处理算法，文本处理算法，谐音算法等，该模块部分函数需要异步调用，均提供回调接口。处理算法子模块集成了大量的第三方库和外部商用服务接口，也有部分自己实现的算法，为上层的分析过程提供了统一的调用方式和错误处理

方式。4.2表列出了处理算法子模块对外提供的功能列表。

表 4.2: 处理算法子模块函数列表

功能	参数	主要实现方式	返回值/回调函数参数
获取并去除间隔	音频文件	扫描音频文件，以 2 帧作为窗口，调用第三方库 webrtcvad 确认其是否为人声，如果不是人声，记录间隔开始时间与结束时间，最后复制出没有间隔的、新的音频	返回间隔列表与去除间隔后的音频文件
频率分布	音频文件	使用快速傅里叶变换（fft）计算频率分布	成功回调参数是频率分布结果，失败参数是失败原因
语音识别	音频文件	分段调用百度语音接口识别，每段识别会自动重试 3 次，如果全部成功则调用成功回调，否则调用失败回调	成功回调参数是分段的识别结果，失败回调参数是失败原因
情感倾向识别	音频文件	调用百度情感识别接口，会自动重试 3 次，如果成功则调用成功回调，否则调用失败回调	成功回调参数是情感识别结果，失败回调参数是失败原因
音量识别	音频文件	分段测算音量，音量使用音频文件中各取样数字绝对值的算数平均来表示	返回音量列表
清晰度计算	用户文本、标准文本	清晰度表示用户文本与标准文本之间的距离，使用 Levenshtein 距离 (编辑距离) 进行计算	返回清晰度（0-1 浮点数）
无效表达率计算	用户文本、标准文本	无效表达率表示用户文本有多少标准文本没有的内容，算法同“清晰度计算”	返回无效表达率（0-1 浮点数）
完成度计算	用户文本、标准文本	完成度表示用户文本完成了多少标准文本的内容，算法同“清晰度计算”	返回完成度（0-1 浮点数）
词性分布计算	用户文本	调用第三方库 jieba 分词作为分词方法	分词结果、各类实词词性比例、虚词比例
谐音击中测算	用户文本、词语	使用谐音击中算法，比较用户文本是否击中某个词语	击中时返回 1，否则返回 0

以谐音击中算法为例，4.1.1节中已经叙述过谐音击中的概念。谐音击中与直接击中相对，直接击中表示文本是否包含了某词语，而谐音击中表示文本的音节中是否相似地包含某个词语的音节。以文本“糕点是一个现代化的搅动工具”是否击中关键词“交通”作为一个典型例子，该算法处理这一示例的过程如图4.9所示。

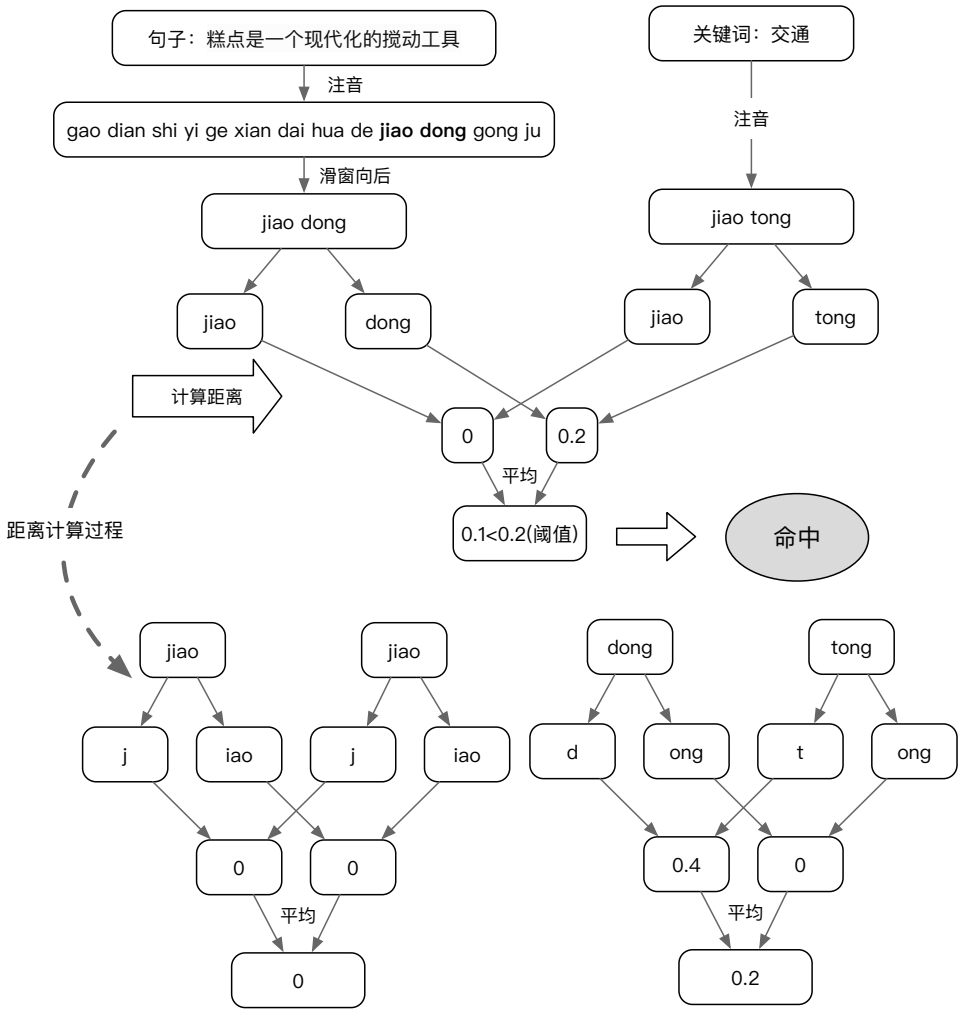

图 4.9: 谐音算法过程实例

图4.10是这个函数实现的核心代码，省略了错误处理相关代码。首先，函数会利用第三方库 pypinyin[2]将文本和关键词分别注音；接着，函数将以关键词长度为滑窗大小，从前向后检验是滑窗内的词语是否与目标关键词谐音。判断两个词语是否谐音需要分别计算对应每个拼音的距离取平均，如果这个平均距离小于指定的阈值，则判定两个词语谐音相似。而两个拼音的距离计算则参考了汉语的发音规律 [46]，以声母韵母相似度加权平均的方式获得。

[2]https://github.com/mozillazg/python-pinyin

```
def in_pronunciation(word, sentence):
    sentence_pinyin = change_into_pinyin(sentence)  # 文本的注音
    word_pinyin = change_into_pinyin(word)  # 关键词的注音
    len_sentence, len_word = len(sentence_pinyin), len(word_pinyin)
    if len_sentence < len_word:
        return False
    for i in range(len_sentence - len_word + 1):  # 滑窗向后依次检验
        temp = sentence_pinyin[i:i + len_word]
        if similar_pronunciation(word_pinyin, temp):  # 计算两个词语拼音的距离
            return True
    return False


def similar_pronunciation(word1, word2):
    if not len(word1) == len(word2):
        return False
    sim = 0
    for i in range(len(word1)):
        temp = destination(word1[i], word2[i])  # 计算每个拼音的距离
        sim = sim + temp
    return sim / len(word1) <= similar_threshold  # 如果小于指定的阈值则返回谐音
相似


def destination(pinyin1, pinyin2):
    cons1, vowe1 = cons_and_vowe(pinyin1)  # 获取拼音的声母和韵母
    cons2, vowe2 = cons_and_vowe(pinyin2)
    dest1 = dest_table1[cons.index(cons1)][cons.index(cons1)]  # 查表获得声母
距离
    dest2 = dest_table2[vows.index(vowe1)][vows.index(vowe2)]  # 查表获得韵母
距离
    return (alpha * dest1 + beta * dest2) / (alpha + beta)  # 加权平均
```

图 4.10: 谐音算法核心代码

4.2　优化模块详细设计与实现

题目参数优化是保证模型稳定性的重要步骤，也是整个模型生态走向闭环的最后一步。在优化模块中，系统会利用用户数据呈现的统计学规律，调整题目的参数，使得题目呈现更好的数据分布情况，从而提升以后用户回答分析的准确率。本节将先从介绍优化算法的设计思路，再介绍如何让算法在代码中落地实现。

4.2.1　优化算法设计

不同的参数将采用不同的优化算法，朗读题由于模型固定，需要调整的参数只有难度参数；转述题和问答题需要优化词语权重向量参数和级别参数。此外，优化模块也会更新题目实体的 nears 字段和 results 字段。

难度参数的优化过程将由公式4.2决定，其中 D_i 表示第 i 题的难度，$Avgs$ 是

所有该类型题的平均分数组，函数 *avg* 和 *sd* 分别表示数组的平均值和标准差，函数 *near* 将计算结果就近划分到 $[-1, 0, 1]$ 中。该算法的理由是，当某道题平均分在所有题目的平均分中处于较高位置时，可以认为这道题较为简单，反之这道题比较难。

$$D_i = 1 - near(\frac{Avgs_i - avg(Avgs)}{sd(Avgs)}) \tag{4.2}$$

权重向量参数的优化思路来自于对用户数据分数的正态拟合。对于一道转述题（问答题来说），理想的分数分布应该是符合高斯分布的，即高分和低分较少，中间的分数较多，因此可以通过调整权重向量参数，拟合用户分数与正态分布曲线。权重向量参数的优化过程包括以下几个重要步骤：

（1）按照得分的升序排列，提取这道题所有样本的击中向量，以行向量形式组成矩阵 *hits*，并提取该题的权重向量 *weights*，分数数组 *scores*，由公式4.1可得 $scores^T = hits \cdot weights^T$。

（2）以 *scores* 的平均数和标准差作为参数，随机生成相同样本容量的符合高斯分布数组 $scores_{gaus}$

（3）现在有了输入 (属性)*hits* 和标准数据 $scores_{gaus}$，需要通过统计学习的方式优化权重 *weights*

优化过程的损失函数由公式4.3给出，其中 L_2 表示向量到 0 向量的欧几里得距离，$weight^2$ 项是学习过程中的惩罚项。由于实际数据 *scores* 与输入 *hits* 之间的线性关系，此优化过程有两种方式可以选择。第一种是梯度下降法，此方法的优势在于其通用性，并且可以添加惩罚项，劣势在于计算过程较慢，可能需要较多次回归，这也是系统目前采用的方法。第二种是最小二乘法，此方法仅限于线性关系的最优解，不能添加惩罚项，但是速度较快。

$$J(weights) = L_2(hits \cdot weights - scores_{gaus}) + weights^2 \tag{4.3}$$

级别参数将主要由两点决定。一是题目的已有的样本数量，这是一个硬性指标，样本数量低于某特定值的题目级别也必须较低，例如样本容量低于 40 的题目级别只能为 1。在样本数量超过一定数目之后，需要假设已有样本分布符合标准正态分布，并计算出拒绝该假设的 p 值，p 值越小，说明已有样本 “越符合” 标准正态分布，即该题的级别越高。该检验过程采用卡方检验，p 值的计算过程如公式4.4所示，其中 n 表示样本容量，k 是划分区间个数，此处取 20，X_i 为在此区间划分下第 i 个区间中样本频数，p_i 为正态分布下该区间的理论频率。

$$p = 1 - \chi^2 = 1 - \sum_{i=1}^{k} \frac{(X_i - np_i)^2}{np_i} \tag{4.4}$$

题目的 nears 属性表示题目的邻近其他题目列表。寻找邻近实体是数据分析中的常见操作，尤其是在 k-means 等聚类算法中，其基本做法是依次计算实体属性之间的向异性，再统一计算实体之间的距离。计算完题目之间的距离之后，系统会使用距离较小的题目更新 nears 列表。题目需要考虑的属性包括标称属性级别和难度、枚举属性标签、数值属性平均分这几种属性。具体的距离计算过程可以参考2.1.2节“数据属性与数据向异性度量”。

4.2.2　优化算法实现

优化模块的类图如图4.11所示，优化模块的核心类 OptimizeTask 继承自 ScheduleTask，表示优化过程是一个定时任务。OptimizeTask 将先从底层数据库中获取数据，并调用 OptimizeRunner 实际执行优化任务。

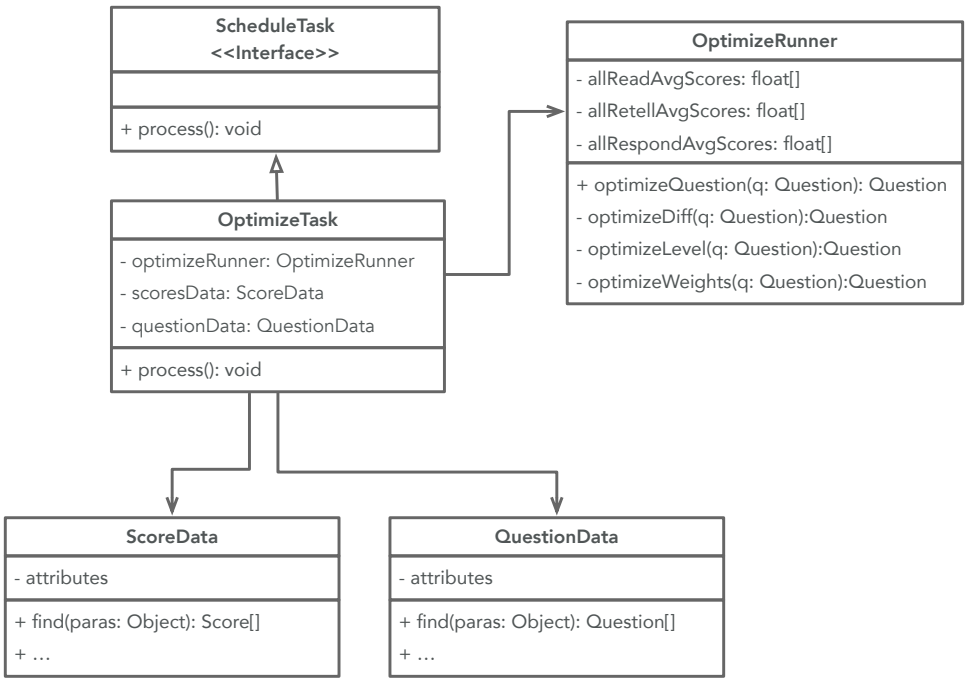

图 4.11: 优化模块类图

图4.12是整个优化过程的代码框架，在 process() 方法中，OptimizeTask 循环题目列表，逐题调用 OptimizeRunner 的优化函数 optimize_question()，并逐个优化难度参数，权重向量参数和级别参数。

```
# 优化任务，继承自定时任务
class OptimizeTask(ScheduleTask):
    def __init__(self):
        self.score_data = ScoreData()
        self.question_data = QuestionData()
        read_scores = self.score_data.get_read_scores()
        retell_scores = self.score_data.get_retell_scores()
        respond_scores = self.score_data.get_respond_scores()
        # 获取各种类型题目的分数列表，并将其传入OptimizeRunner中
        self.optimize_runner = OptimizeRunner(read_scores, retell_scores,
respond_scores)

    # 定时任务固定方法
    def process(self):
        questions = self.question_data.get_questions()
        for q in questions:
            # 逐个优化
            self.optimize_runner.optimize_question(q)

class OptimizeRunner:
    def __init__(self, all_read_avgs, all_retell_avgs, all_respond_avgs):
        self.all_read_avgs = all_read_avgs
        self.all_retell_avgs = all_retell_avgs
        self.all_respond_avgs = all_respond_avgs

    def optimize_question(self, q):
        # 根据题目类型调用不同的优化方法
        self._optimize_diff(q)
        if q.type == QuestionType.RETELL or q.type == QuestionType.RESPOND:
            self._optimize_weights(q)
            self._optimize_level(q)

    def _optimize_diff(self, q):
        pass
    def _optimize_weights(self, q):
        pass
    def _optimize_level(self, q):
        pass
```

图 4.12: 优化过程框架

以权重参数的优化过程为例，图4.13是该优化过程的核心代码。首先，函数会从题目中获取权重向量 weights 和所有的样本 samples；接着，函数会整理样本数据生成击中矩阵 hits 和分数向量 scores(列向量)；然后，函数提取出分数向量的相关参数，并用这些参数生成标准正态分布 score_gauss；最后，函数调用梯度下降算法优化权重向量。

```
    def _optimize_weights(self, q):
        weights = np.array(q.get_weights())
        samples = q.get_samples()  # 提取问题所有样本
        # 整理成二维数组并按照score排序
        hits_scores = [sample['hit'] + [sample['score']] for sample in
samples]
        hits_scores.sort(key=lambda t: t[-1])
        hits_scores = np.array(hits_scores)
        # 拆分hits和scores
        hits, scores = hits_scores[:, :-1], hits_scores[:, -1:]
        # 生成标准正态分布
        n, mu, sigma = len(scores), scores.mean(), scores.std()
        scores_gauss = np.random.normal(mu, sigma, n).sort()
        # 调用梯度下降法优化参数weights
        GradientDescent.optimize_param(weights, hits, scores, scores_gauss)
        q.set_weights(weights)
```

图 4.13: 权重向量参数优化过程

4.3　语料模块详细设计与实现

语料模块并不对外提供服务，它在系统中扮演了一个辅助性质的角色。评测类应用的一个特殊性在于其需要规模较大且不断更新的题库，否则评测者很容易刷遍整个题库，使得应用失去评测效果，因此系统设定了语料模块。作为题目文本的来源，语料模块为管理员添加题目时提供思路支持。语料模块可以从互联网世界中爬取合适的资源，自动分析整理形成原始素材存入库中。管理员在添加题目时，只需要对初始化参数进行少量修改，便可以直接使用这些素材。即使有些语料不能直接使用，管理员也可以从中获得思路上的启发。

利用 python 制作爬虫获取网络资源是 python 语言主要的应用方向之一。语料模块采用了较为成熟的 scrapy 框架来进行文本段落爬取步骤，这个框架封装了底层的网络连接等细微操作，并提供了大量可以直接使用的网页处理方法。结合 scrapy 的流水线框架，图4.14展示了语料模块的控制流程。首先，系统会从特定的网络地址中爬取网络资源，做简单清洗，包括去除空白字符，去除特殊符号等，形成结构化文本段落；接着，系统会对文本进行简单的分词分句梳理，初始化其参数，最终形成一道级别为 0 的题目；最后，系统需要将生成的题目保存录入库中。4.3.1节和4.3.2节会详细叙述文本段落爬取子模块和题目参数初始化子模块，入库的步骤比较简单，这里略过。

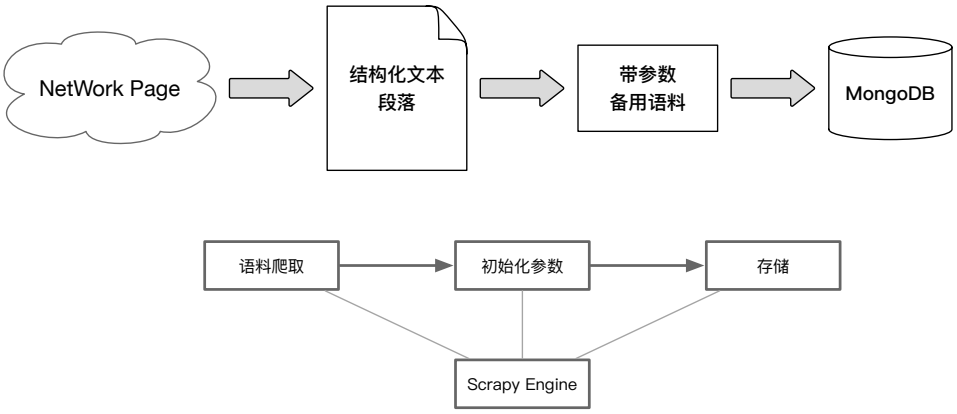

图 4.14: 语料模块流程

4.3.1　文本段落爬取

语料模块是一个由定时任务控制的模块，默认的执行周期为每天一次。因此，为了保证每次爬取的文本不同于之前的文本，模块采取了“二次爬取”的策略，即首先从一个地址上爬取主题，再从另一个地址上爬取真正的资源。本系统采用了“百度热词榜 + 百度百科”的组合模式，图4.15是文本段落爬取子模块的顺序图。

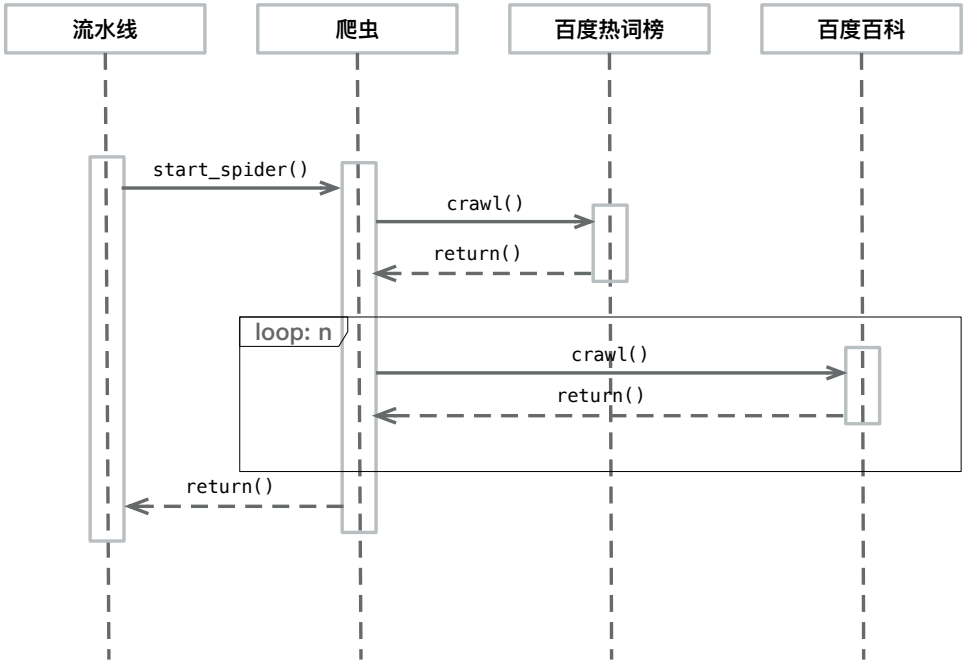

图 4.15: 文本段落爬取顺序图

确定了上述爬取策略之后，可以得到该子模块的类图，图4.16是该子模块的主要类图，BaikePipeline 类是子模块的入口，BaikeSpider 是具体爬取任务的执

行者，在设定了开始 url 之后，爬虫会自动访问网络资源，并将 response 作为参数传入回调函数 parse 中，爬虫需要解析这个 response，生成一个 item 供后面的流水线步骤使用。

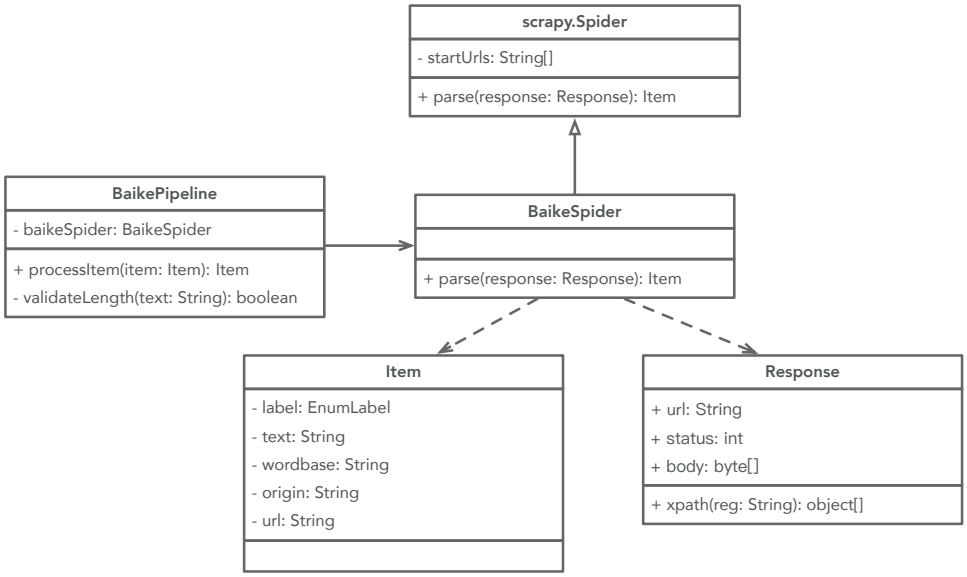

图 4.16: 文本段落爬取类图

```
class BaikeSpider(scrapy.Spider):
    def parse(self, response):
        # xpath选择器可以从response中选择符合条件的文本组，这里选取热词
        hots = response.xpath(selector['hot_word']).extract()
        # 遍历所有热点，等待下一次请求过程，并传入回调函数
        for hot in hots:
            yield Request(self._baike_url + hot, callback=self.parse_word)

    def parse_word(self, response):
        if len(response.xpath(selector['title']).extract()) == 0:
            # 非多页义项页面，将内容连成一段话
            content_list = response.xpath(selector['content']).extract()
            content = "".join(content_list).replace('\n', '')
            item = Item(url=response.url, content=content)
            yield item
        else:
            # 多义项页面，选取第一个链接继续爬取
            word = response.xpath(selector['item']).extract()[0]
            yield Request(self._baike_url + word, callback=self.parse_word)
```

图 4.17: 文本段落爬取核心代码

图4.17展示了文本段落爬取的核心代码，其中 parse 函数是请求热词榜单的回调，这个回调函数中，系统利用 xpath 选择器选择出热词榜单，抛出（yield）

了对单个热词的二次请求，指定 parse_word 函数为新的回调。在 parse_word 函数中，系统拼接网页中的文本形成段落，组装 item 并返回。

4.3.2　题目参数初始化

爬取好文本段落以后，流水线的下一步是初始化题目的参数，包括难度、级别、词库等，其中最主要的就是词库的初始化。对于转述题来说，能否准确地初始化关键词将直接影响到分析模块中词语击中向量特征能否正确提取。

本模块采用了 python 第三方库 jieba 来实现关键词的提取，提取过程中采用了 TF-IDF 算法。如图4.18所示，词库的生成需要几个主要的步骤。首先，系统会依据首句和分句的规则，分离主旨句和细节句；然后，系统将会使用 jieba 提供的方法，使用词性作为额外筛选条件，提取主旨词（细节词）；接着，系统需要对每个词语进行同义词的扩充，原因是一些书面化的词语并不适用于口语表达，例如“美利坚合众国”，大家一般都会直接说它的同义词“美国”，因此需要进行同义词的扩充，将“美国”也纳入词库中；最后系统会去除部分词库中重复的词语。对于细节词来说，需要进行的额外步骤是数量词扩充，即将数量词对应的阿拉伯数字纳入词库中。

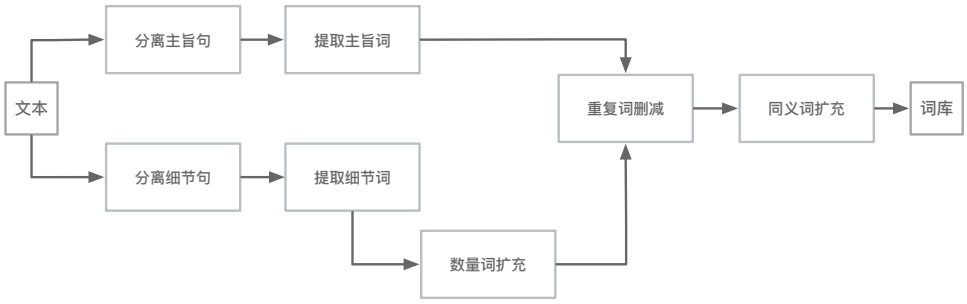

图 4.18: 题目参数初始化流程

图4.19展示了题目参数初始化的核心代码，其中 generate_wordbase 用于生成词库。在这个函数中，系统首先按照配置的分隔符进行分句，然后对主旨句提取主旨词，对细节句提取细节词。主旨词的词性限定在名词，动词等实词里，而细节词的词性则限定在形容词，副词，数量词等修饰词里，接着系统会调用 expand_synonyms 对词库进行同义词扩充，最后系统将提取好的词库返回。

```
# 主旨词词性、细节词词性、句子分割符
key_pos = get_config('key_pos')
detail_pos = get_config('detail_pos')
separator = get_config('separator')

class WordbaseGenerator:
    @classmethod
    def generate_wordbase(cls, content):
        sentences = re.split(separator, content)[:-1]  # 分离句子
        key_sentence = sentences[0]  # 分离主旨句
        # 提取主旨词并去重
        keywords = cls._duplication_remove(cls._get_keys(key_sentence))
        detail_sentences = sentences[1:]  # 分离细节句
        detailwords = []
        # 对每个细节句提取细节词并去重
        for sentence in detail_sentences:
            temp = cls._duplication_remove(cls._get_details(sentence))
            temp = cls._expand_numeric(temp)  # 数量词扩充
            detailwords.append(temp)
        wordbase = {'keywords': keywords, 'detailwords': detailwords}
        return cls.__expand_synonyms(wordbase)

    def _expand_synonyms(wordbase):
        # 省略同义词扩充
        pass
```

图 4.19: 题目参数初始化核心代码

4.4　业务模块详细设计与实现

业务模块是用户可以通过客户端直接访问的模块，实现的需求包括评测题组选取，题目的增改查等。业务模块封装在一个 flask app 之中，并通 uwsgi 网关提供的多线程支持，对外提供 http 服务。

4.4.1　评测题组选取

评测题组选取是业务模块提供的一项主要功能，用户在评测开始时，客户端会像题目模块请求此次评测使用的题组。图4.20是评测题组选取时的状态图，从图中可以看出，选取过程需要考虑的因素包括两个大的方面，即用户方面和题库方面。

用户方面包括用户属性和用户可能进行的偏好选择。系统会以用户属性为基础生成用户画像，并采用表驱动的专家规则算法分析用户的文化水平和可能的兴趣偏好，进而生成针对性的题目标签和题目难度。例如学生可能对科普类更感兴趣，而记者可能更需要新闻类题目的训练，同时，他们对题目难度的需求也各不相同。当然，如果用户手动选择了标签和难度偏好，用户的选择将会覆盖系统的默认参数。

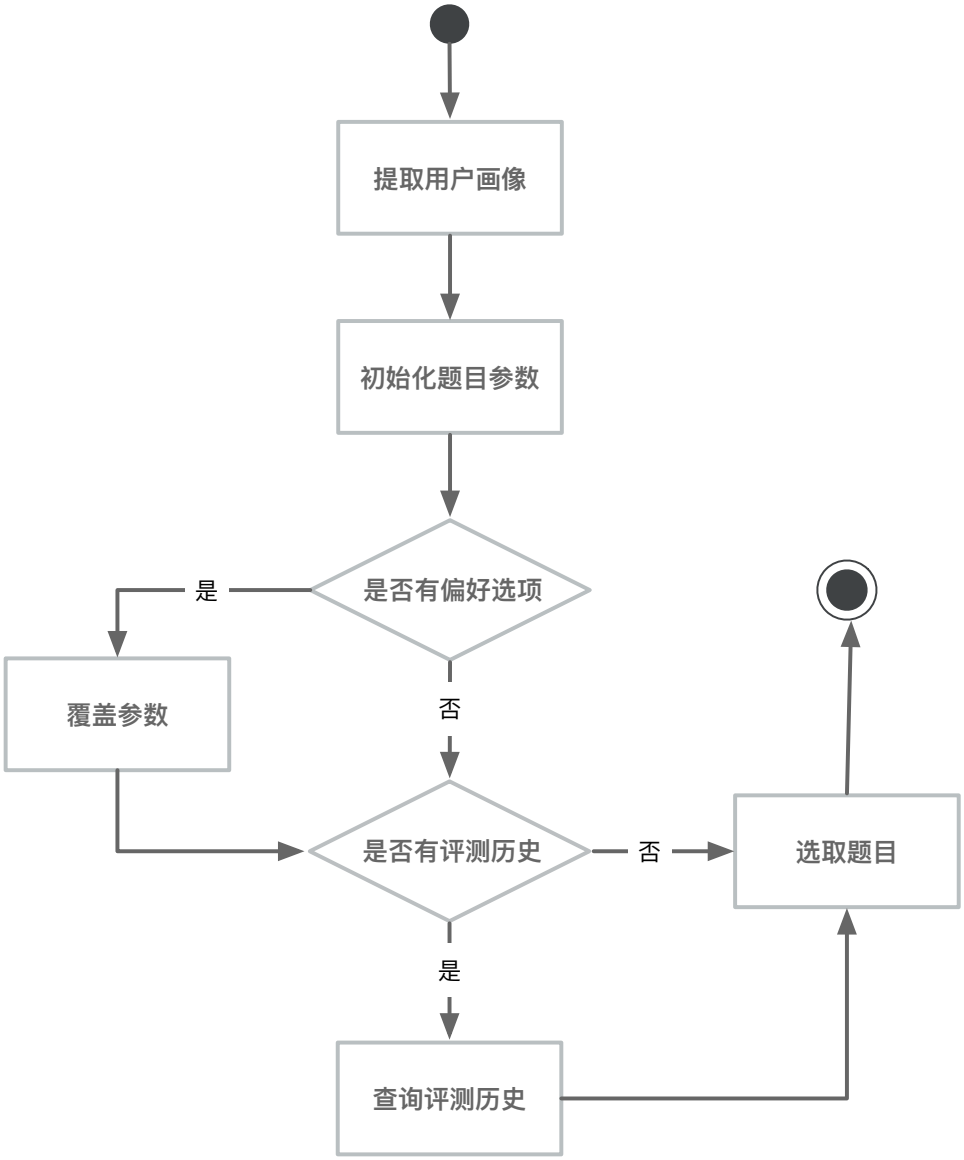

图 4.20: 评测题组选取状态图

题库方面包括用户历史评测题目。为了保证同一个（组）用户评测问题不重复，系统会分析用户（组）的评测历史，并从题库中寻找与用户（组）评测历史相近但不相同的题目，从而更好的强化评测者的表达能力。题目的邻近题目组参见4.2.1节。

图4.21展示了评测题组选取过程的类图，核心类 PaperGenerator 提供试卷生成的功能，UserFilter 和 QuestionFilter 分别从用户角度和问题角度进行过滤。

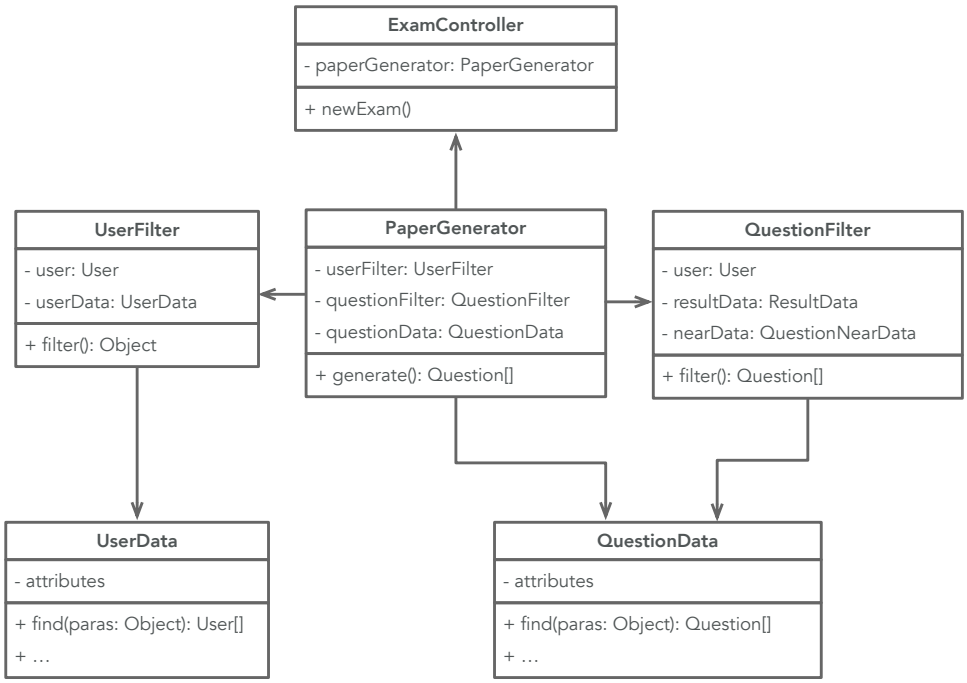

图 4.21: 评测题组选取类图

4.4.2　业务处理示例

业务模块还有题目的增删改等业务处理操作。业务模块是典型的分层架构，由前端的可视化层，后端的控制器层、业务逻辑层、数据层这四层构成，上下层之间存在依赖关系。

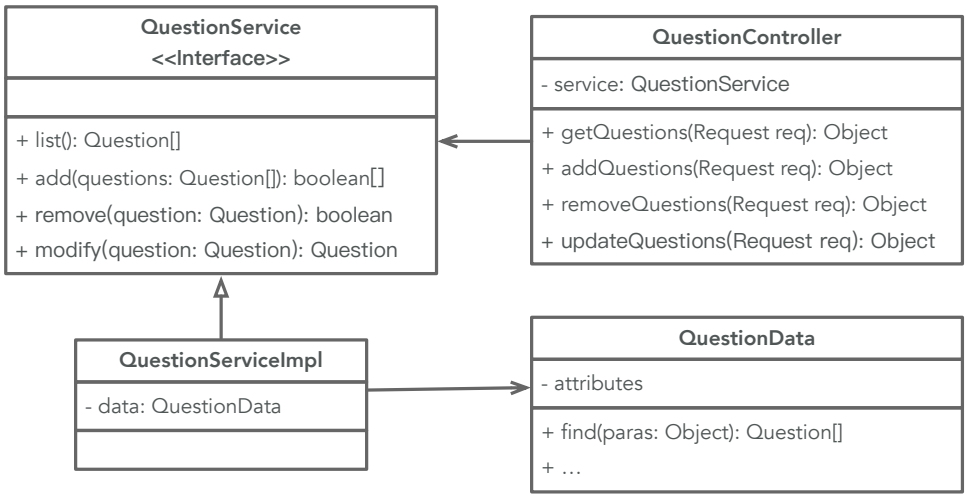

图 4.22: 题目列表查看类图

如图4.22所示，QuestionController 是 flask 的控制器，每个方法对应一个 http

请求；QuestionService 是服务层，承接了业务逻辑功能，例如排序、过滤、合并请求条件等；QuestionData 封装了 MongoDB 的数据库读写操作。

这里以题目列表展示为例介绍业务操作的实现，图4.23是展示题目列表的后端核心代码。controller 从 request 的参数列表中提取出请求参数之后，service 会根据业务逻辑添加筛选条件和默认的排序规则，随后调用 data 层的数据库查询方法。

```
class QuestionController():
    @admin.route('/questions', methods=['GET'])
    def get_all_questions(self, request):
        # 从request中获取页码，页大小，筛选条件，排序依据参数
        page, size, conditions, sort = self._get_args(request,
            ['page','size','conditions','sort'])
        return jsonify(resp.success(service.list(page, size, conditions)))

    def _get_args(self, request, keys):
        # 省略获取参数方法
        pass


class QuestionServiceImpl(QuestionService):
    def list(self, conditions=None, page=1, size=10, sort=None):
        # 查询的时候排除level为0的题目套件，因为level为0的是备用语料
        conditions = join_condition(conditions, {"level": {"$ne": 0}})
        if not sort:
            # 默认的排序是级别最低 && 使用次数最多
            sort = {"level": 1, "used_times": -1}
        return self.data.get_questions(conditions, page, size, sort)


class QuestionData:
    def __init__(self):
        client = pymongo.MongoClient(MongoConfig.connect)  # 创建连接
        mdb = client[MongoConfig.db]  # 获取db
        self.questions = mdb[MongoConfig.questions]  # 获取question文档指针

    def get_questions(self, conditions=None, page=1, size=10, sort=None):
        skip, limit = (page - 1) * size, size  # 计算获得页码
        sort = {} if sort = None else sort  # 获得排序依据
        # 根据查询结果返回列表
        return self.questions.find(conditions)
            .sort(sort).skip(skip).limit(limit)
```

图 4.23: 题目列表展示核心代码

图4.24展示了题目列表展示的页面截图，从图中可以看到，题目按照修改级别排序，并支持筛选、分页等操作。

图 4.24: 题目列表前端截图

4.5　本章小结

本章主要叙述了分析模块、优化模块、语料模块和业务模块的详细设计，通过类图、流程图、顺序图、算法表述等形式叙述了这些模块的静态模型和动态交互过程，并使用核心代码、页面截图等形式给出了模块重要逻辑的实现过程和结果。

第五章　持续集成与系统测试

本章将从软件工程角度，阐述系统在集成部署方面和测试方面的工作。在集成部署方面，系统采用了业内通用的 Gitlab[1]+Jenkins[2]作为 CI/CD 工具链，实现代码推送与系统自动更新。在测试方面，除了基本的功能测试与性能测试以外，本章也简要叙述了两次试运营测试的实施及效果。

5.1　持续集成与持续部署

敏捷理论认为，持续集成与持续部署是软件工程过程中最优先采用的敏捷实践，也是效果最明显的。参照系统物理视图图3.8，本系统的前端部分、服务端部分和分析模块均使用 Gitlab+Jenkins 进行持续集成与部署。三个部分的任务截图如图5.1所示。

图 5.1: Jenkins 任务列表

以前端部分为例，如图5.2所示，Jenkins 连接了远端的 Gitlab 仓库并设置了 Credentials，在 Gitlab 端设置 push 后钩子函数（webhook）之后，系统会在每次有 push 请求之后触发构建任务。

[1]https://www.gitlab.com/

[2]https://jenkins.io/zh/

图 5.2: 前端构建 Jenkins 配置

前端的构建任务由一系列脚本任务组成。如图5.3所示，脚本首先定义目标文件夹路径变量；随后脚本会删除原有目标文件夹，并根据最新前端源码编译出最新包；最后，脚本会移动目标文件夹，使得路径与原路径匹配。

```
dist_target_folder='/var/www/expression-vue/dist'
echo "=============Start building static files============="
if [ -d dist ]; then
    ls -ld dist/
    rm -rf dist/
fi
npm install
npm run build --fix
echo "=============Start moving dist folder============="
mkdir -p $dist_target_folder
rm -rf $dist_target_folder
mv dist $dist_target_folder
ls -ld $dist_target_folder
du -hs $dist_target_folder
```

图 5.3: 前端构建脚本

5.2　系统测试

根据3.3.3节和3.3.4节，系统需要实现一系列功能需求，并保证可用性和高并发下的性能等非功能需求。本节对系统个各项功能在边界情况下进行了功能测试，并进行了系统各个接口的压力测试。

5.2.1　测试环境准备

根据图3.8描述的物理视图，测试环境包括运行 Chrome 浏览器和微信小程序的客户端、运行 Vue App 和 Nginx 的前端服务器、运行 Flask App 的服务器节点、运行分析模块代码的计算节点和运行数据库的存储节点，并购买了 CDN 服务和文件对象存储服务。除了用户客户端以外，所有服务器均采用独立的百度

云服务服务器，相关客户端与服务器配置如表5.1，其中设备参数中服务器的参数分别是带宽、内存、磁盘和服务器类型。

表 5.1: 测试环境说明

设备名称	设备参数	运行程序	程序版本
用户计算机	MacOS 10.15.3	Chrome 浏览器	v80.0
用户手机	Android 9	微信小程序	v7.0.12
前端服务器	4M 8GB 40GB 通用型	Vue App Nginx	Python3.6 v1.14.0 Ubuntu
后端服务器	4M 8GB 40GB 通用型	Flask App	Python3.6
计算服务器	4M 4GB 40GB 密集计算型	Python Process	Python3.6
存储服务器	4M 8GB 40GB 通用型	MongoDB	v3.6.3

5.2.2　功能测试

功能测试将对3.3.3节叙述的所有功能逐一进行验证，采用黑盒测试的手段，即仅考虑系统的功能表现，并不关注系统内部结构与实现情况。由于系统分为分析模块、优化模块、语料模块和业务模块，不同模块之间差异较大，不能采用同一种方式进行测试，需要单独设计测试方案验证各个模块的功能是否符合预期。下面将从模块划分的角度介绍测试方案。

(1) 分析模块

分析模块主要涉及表3.10中功能需求 R1，即根据题目元信息和用户输入的音频数据分析题目，给出用户的表达能力水平情况。这部分涉及到了较多边界情况和异常情况，例如空白音频、音频时间极短或极长、音频质量极差等。以第一种题型朗读题为例，表5.2给出了这一题型的测试用例，从表中可以看出，正常情况、边界情况和异常情况均有所覆盖。

表 5.2: 分析模块测试方案设计

编号	输入音频情况	预期输出
TC1-TC10	正常环境，字正腔圆，情感充沛，语速均衡	大于 90 分
TC11-TC20	正常环境，有少数错字和停顿，语速略快，较为紧张	80-90 分
TC21-TC30	正常环境，错字和停顿较多，语速正常，有方言口音，有额外的习惯助词用语	60-70 分
TC31-TC40	正常环境，语速极快，难以分辨，毫无情感	40-50 分
TC41-TC50	正常环境，语速极慢，有大段空白	20-40 分
TC51-TC60	嘈杂环境，时长正常，声音含糊不清	提示更换环境重新作答
TC61	空白音频	0 分

编号 TC1 到 TC50 的五十个测试用例由表达力领域专家录制而成，打乱后

经由另一组专家判分。这些专家来自演讲家、教师等多个行业，除了必要的录制要求之外对系统没有了解。系统收集了这些测试用例的专家判分与系统判分数据对比，对于每一个测试用例，如果系统判分与专家判分之差在 5 分以内则可以认为系统判分较为符合预期。图5.4展示了这一模块测试情况，图 1-5 展示了测试用例 TC1-TC50 的专家判分结果与系统判分结果的对比，可以看出专家结果与机器结果基本一致，有一定的相似性；图 6 是所有测试用例专家结果与机器结果差值，大部分差值均在 5 分可接受范围内，说明分析模块的功能满足预期。

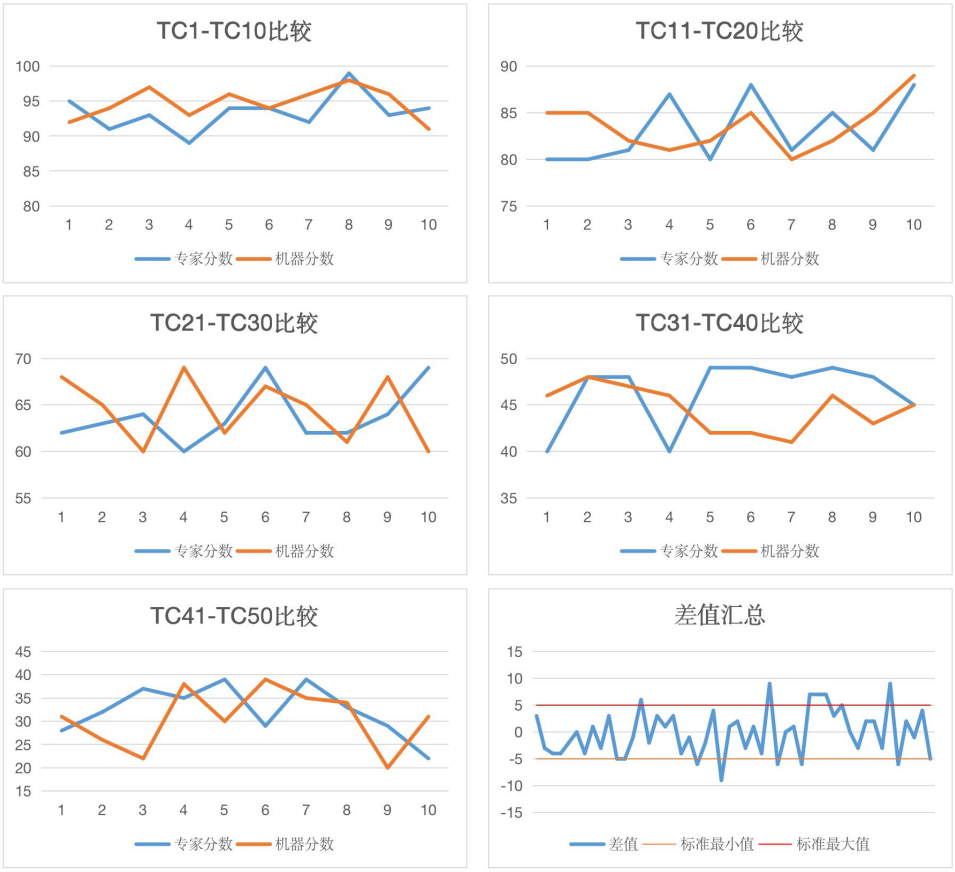

图 5.4: 分析模块功能测试结果

(2) 优化模块

优化模块主要涉及表3.10中功能需求 R2，即系统需要能够根据题目的用户数据参数来优化题目的参数，使得题目模型更加稳定。这部分的测试难点在于判断某道题是否被优化，是否能够更好地达到评测的效果，针对这一测试难点，表5.3给出了优化模块的测试方案设计。

表 5.3: 优化模块测试方案设计

#	说明
1	随机从题库中选择一个题目，并从普通用户中收集 20 个样本
2	专家对步骤 1 中采集到的样本逐一判分，收集数据
3	使用优化前的模型对步骤 1 中采集的样本逐一判分，收集数据
4	运行优化模块
5	使用优化后的模型对步骤 1 中采集的样本逐一判分，收集数据
6	合并步骤 2 步骤 3 和步骤 5 收集到的数据，按照步骤 2 数据升序排列
7	重复步骤 1-6 以消除随机误差

测试方案5.3通过对比专家判分数据（被认为是标准数据）、优化前模型判分数据、优化后模型判分数据这三组数据来验证优化模型的有效性。从直观感受来看，如果优化后数据比优化前数据更接近专家数据，即可证明优化模块功能符合预期。图5.5给出了某题目的测试结果，从图中可以看出，相比起优化前数据，优化后数据与专家数据更加接近。除了使用图形直观感受之外，系统还在统计学的意义上，通过计算样本点差距，同样得出了相同的结论。

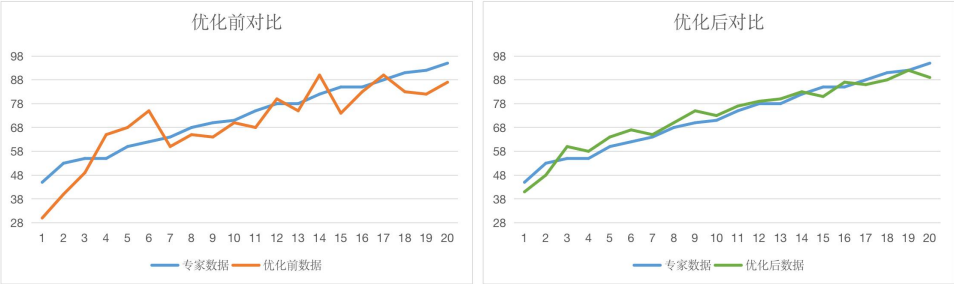

图 5.5: 优化模块功能测试结果

(3) 语料模块

语料模块主要涉及表3.10中功能需求 R3，即系统需要定时从互联网中爬取合适资源，形成备用语料，以保证题目来源的多样性。这部分的测试任务需要确认模块是否能够正确的抓取文本、提取词库、获取同义词等。结合语料模块“二次抓取”的特征，本模块的测试方案中将测试用例分为了两个部分，第一部分测试模块能否正确地抓取热词，第二部分测试模块针对某个热词能否正确的形成原始语料。

第一部分需要测试语料模块从网络中获取热词的能力。以 2020 年 3 月 16 日的测试情况举例，图5.6给出了这一天的测试结果日志截图，从图中可以看出，系统能够获取到这一天百度风云榜中搜索热度较高的短语，并能够提取出热词。

```
------- Corpus Crawler Start -------
DateTime: 2020-03-20 03:30
Request URL: https://top.baidu.com/buzz?b=1
Requestion Success
GET hotspot: 塞尔维亚紧急状态，施乐暂停收购惠普，迈阿密市长确诊，
             美联储利率降至零，泰国新增32例确诊，江苏确诊病例清零，
             英女王迁离伦敦，黄书豪出家，宁夏确诊病例清零，几内亚首例确诊
GET words: (塞尔维亚紧急状态) Get word: 塞尔，维亚，塞尔维，塞尔维亚，紧急，状态，紧急状态
GET words: (施乐暂停收购惠普) Get word: 施乐，暂停，收购，惠普
GET words: (迈阿密市长确诊) Get word: 迈阿密，市长，确诊
GET words: (美联储利率降至零) Get word: 联储，美联储，利率，降至，零
GET words: (泰国新增32例确诊) Get word: 泰国，新增，32，例，确诊
GET words: (江苏确诊病例清零) Get word: 江苏，确诊，病例，清零
GET words: (英女王迁离伦敦) Get word: 女王，英女王，迁离，伦敦
GET words: (黄书豪出家) Get word: 黄，书豪，黄书豪，出家
GET words: (宁夏确诊病例清零) Get word: 宁夏，确诊，病例，清零
GET words: (几内亚首例确诊) Get word: 几内亚，首例，确诊
```

图 5.6: 语料模块提取热词功能测试结果

第二部分需要测试语料模块搜索热词获取文本素材并初始化题目参数的能力。以词语“迈阿密”这个词为例，图5.7给出了这个词生成原始语料的日志截图，从图中可以看出，系统能够成功地搜索百度百科形成文本素材，并能够提取关键词、扩展同义词等。

```
------- SEARCH WORD(迈阿密) -------

Request URL: https://baike.baidu.com/item/%E8%BF%88%E9%98%BF%E5%AF%86

TEXT: 迈阿密（Miami）是美国佛罗里达州第二大城市，位于佛罗里达半岛比斯坎湾
。迈阿密是国际性的大都市，在金融、商业、媒体、娱乐、艺术和国际贸易等方面
拥有重要的地位，也是许多公司、银行和电视台的总部所在。2008年，迈阿密被《
福布斯》杂志评为“美国最干净的城市”。

WORDBASE: { "keywords":[ [ "坎湾" ], [ "比斯" ], [ "迈阿密", "洛杉矶", "
新奥尔良", "西雅图" ], [ "佛罗里达州", "德克萨斯州", "加利福尼亚州", "北
卡罗莱纳州" ] ], "detailwords":[ [ [ "大都市", "大城市" ], [ "国际性", "
世界性", "全球性" ], [ "国际贸易", "对外贸易" ], [ "电视台", "电视频道",
"广播公司", "电台" ] ], [ [ "2008","二零零八" ], [ "福布斯", "富豪榜", "
彭博" ], [ "杂志", "周刊", "月刊", "时尚杂志" ] ] ] }

LEVEL: 0

DIFF: 0

LABLE: science
```

图 5.7: 语料模块初始化题目测试结果

(4) 业务模块

业务模块涉及表3.10中 R4-R8 这五个需求，包括评测题组选取、新增题目、删除备用语料、查看题目列表等。这些需求需要测试各个具体的业务层接口方法，因此这一模块的测试可以采用传统单元测试的方法。表5.4给出了业务模块测试方案的设计，方法名表示需要测试的函数，说明表示测试用例的需要测试的功能，输入和预期输出给出了单元测试的桩数据。

表 5.4: 业务模块测试方案设计

方法名	测试说明	输入	预期输出
question.addone	新增题目	题目文本、主题、参数	题目新增成功，返回其数据库 ID
question.addall	批量新增题目	题目实体集合	新增成功，返回一组数据库 ID
question.getforexam	评测题组选取	用户参数、考试参数	返回一组题目实体
question.getforshow	题目列表展示	筛选条件，排序条件，分页参数	按照条件返回题目列表
question.getcorpus	获取所有备用语料	分页参数	按照条件返回被用语料列表
question.delcorpus	删除备用语料	备用语料 ID	删除成功
question.modify	修改题目	题目 ID，修改项	题目修改成功，返回题目实体

特务模块的功能测试使用了 python 自带单元测试框架 unittest，以 question.addone 方法为例，图5.8展示了该方法的测试日志输出，从图中可以看出，这个方法完成了新增题目，并返回题目 ID 的功能。

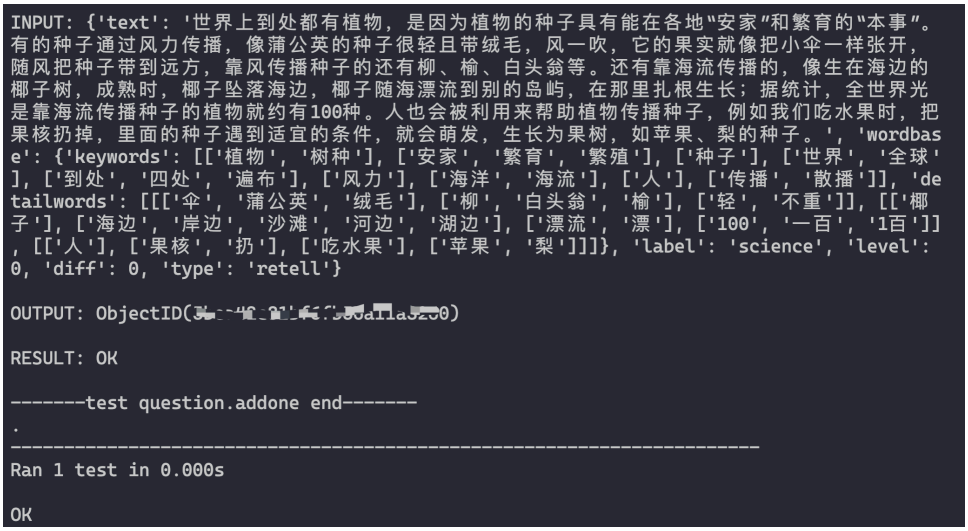

图 5.8: 业务模块功能测试结果

5.2.3　性能测试

本部分将使用自动化测试工具 Jmeter 对系统的分析模块和业务模块进行性能测试，而优化模块和语料模块属于定时任务性质的模块，无需进行性能测试，不在小节叙述范围之内。

(1) 分析模块

分析模块执行任务时会从 Clery 中提取任务，分析之后将结果存入数据库中。这个模块的性能测试方案有两个主要步骤，第一步启动脚本，将较大数量的任务添加进任务队列中；第二步使用 Jmeter[3]模拟 Http 请求轮询访问获取结果的接口，查看有多少请求正确返回。性能的指标主要有两个，第一个是正确分析的任务数量占所有任务的比重，第二个是分析任务平均每秒处理数量。

本模块的性能测试一共进行了五轮，任务数量分别为 50，100、200，500、1000，测试结果如图5.9所示，从左图中可以看出，随着任务数量增加，正确分析的任务比重有所降低，但仍然维持在 95% 以上，这部分的原因应该是商用接口调用 qps 到达了上限；从右图中可以看出，分析任务平均每秒处理数量维持在 10 个左右，这一数字可以随着硬件的扩大而扩大。

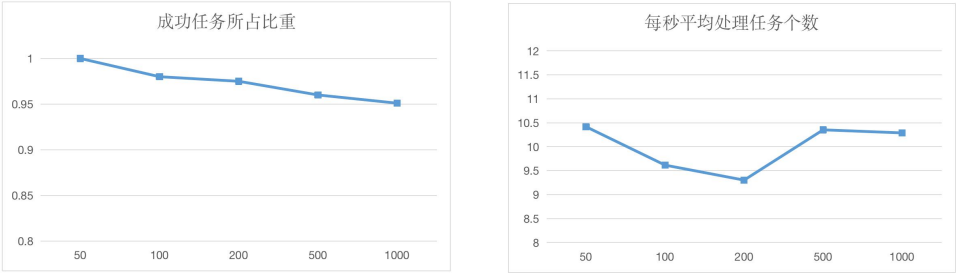

图 5.9: 分析模块性能测试结果

(2) 业务模块

业务模块的性能测试和一般的性能测试相同，系统使用 Jmeter 模拟大量的客户端线程访问后端的 Http 接口，查看其请求成功率与请求 QPS。如图5.10所示，Jmeter 的线程组（用户组）设置了 100 个线程，在 2 秒内发出全部 http 请求，并重复两次，这些线程会依次访问三个 http 请求。

图 5.10: 业务模块性能测试 Jmeter 参数

[3]https://jmeter.apache.org/

Jmeter 提供了聚合报告和响应时间图等多种方式来监听性能测试结果。性能系统聚合结果树表明，在整个 600 个请求中，响应的平均值是 128ms，中位数是 94ms，99% 的请求在 306ms 之内完成，异常率为 0。图5.11给出了测试过程中，请求成功数量与请求时间的关系图，可以看出除开始部分之外，请求的响应时间较为平稳。

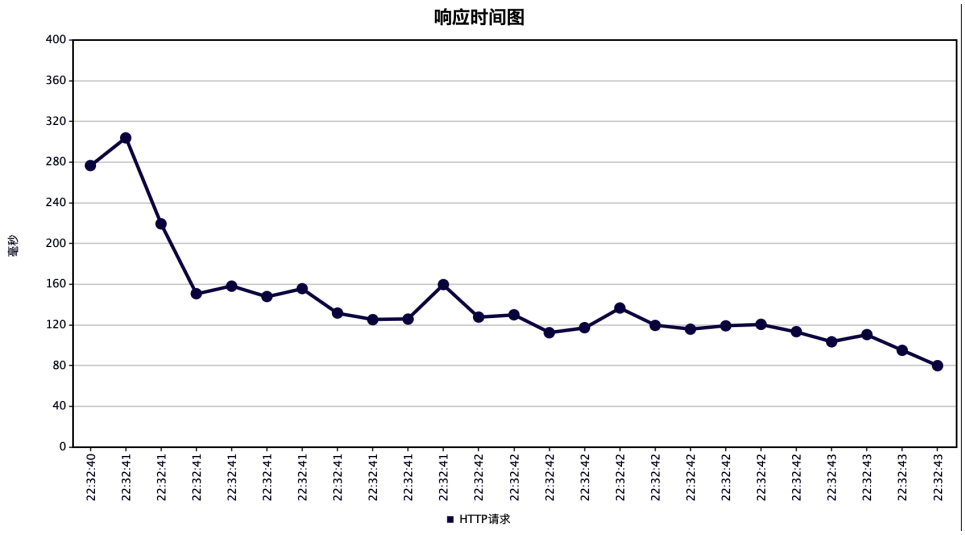

图 5.11: 业务模块性能测试响应时间图

5.3　试运营测试

在对系统进行了功能测试和非功能测试之后，我们对系统进行了试运营测试。试运营测试共有两轮，其中 A 轮是内部人员测试，B 轮是有外部人员参与的公测。本小节将简要介绍这两轮试运营测试的实施情况和数据情况。

5.3.1　A 轮内测

A 轮试运营测试在 2019 年 11 月 23 日开始至 11 月 25 日结束，主要由内部人员参与，共发放内测码 600 个。这一轮试运营实施时，系统的微信小程序端尚未完成，并不能够支持内测，因此所有的内测人员均使用浏览器客户端完成了测试。A 轮内测的测试步骤如下所示：

1）系统发放内测码给受邀人员，受邀人员注册表达力评测项目

2）内测人员参与评测，分析模块给出成绩

3）专家抽取容量为 50 的样本进行专家评测，给出成绩

4）比较并分析系统数据和专家数据

A 轮试运营测试的结果如表5.5所示，表中可以看出，内测人员的参与度为88.6%，系统可用性为 100%，服务器占用情况良好。五个维度的分数分布中，音质分和细节分在人群中表现出了较好的正态分布性，主旨分分数偏高，总体来看，机器判分与专家判分具有一定的相似性。

表 5.5: A 轮试运营测试结果展示

项目	试运营测试结果
参与人次	测试总次数 532 次
可用性情况	成功结束 431 人次，中途主动结束 88 人次，失败并提示重新评测 13 人次，无系统崩溃
服务器节点情况	处理器占用最高 8%，内存占用率最高 40%，入/出带宽占用最高 30/100kBs
分析节点情况	处理器占用最高 4%，内存占用率最高 56%，入/出带宽占用最高 100/170kBs
音质分分布情况	平均分 74.3，标准差 15.2，与专家判分相比平均差距 4.7 分
主旨分分布情况	平均分 80.5，标准差 6.9，与专家判分相比平均差距 2.1 分
细节分分布情况	平均分 55.3，标准差 11.0，与专家判分相比平均差距 3.2 分
逻辑分分布情况	平均分 63.5，标准差 15.3，与专家判分相比平均差距 8.0 分
结构分分布情况	平均分 66.3，标准差 14.2，与专家判分相比平均差距 6.5 分

5.3.2　B 轮公测

B 轮试运营测试在 2019 年 12 月 23 日开始至 12 月 28 日基本结束。这一轮试运营将测试范围扩大到了外部人员，使用微信小程序作为推广工具，并加入了圣诞节的活动主题。这一轮试运营以扩展用户窗口为目的，如图5.12所示，小程序在五天时间内收获了 1559 人的访问和约 2200 人次的测试，公测效果良好。

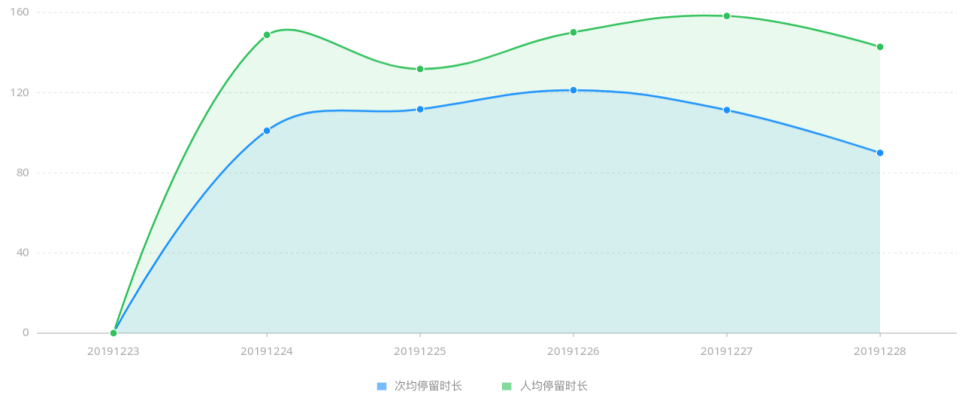

图 5.12: B 轮试运营测试结果展示

5.4　本章小结

本章主要叙述了系统的持续集成持续部署情况和测试情况。持续集成方面，本章介绍了系统的集成工具链 Gitlab+Jenkins 并以前端构建为例，介绍了系统的自动化编译和自动化部署过程。测试方面，本章以图表的形式介绍了系统四个模块的功能测试，并使用 Jmeter 对系统进行了性能测试。最后，本章简要介绍了系统的两次实验评估情况。

第六章　总结与展望

6.1　总结

随着人才管理、人才评价体系日益丰满，人才的表达能力成为综合素质重要组成部分，在招聘、晋升等环节中起到了至关重要的作用。为了解决现有表达能力评价手段集中于线下、时间与人力成本高、无法量化、无法横向比较等问题，本文设计并实现了基于表达能力评测的题目建设与分析系统。系统以表达力评测使用的题目为核心，详细叙述了其生命周期中的关键步骤，并指出了步骤之间的关联，有效地支撑了表达能力评测项目的运行。本系统从分析、优化、来源和日常管理这四个方面叙述了题目的生命周期建设。

题目的分析过程是表达力评测系统的核心过程，在本系统中，分析模块会综合分析用户提交的音频数据和题目元信息，从音频中提取特征，按照一定算法模型计算分数，并使用总分核算步骤对一次评测所有的题目进行总分累计。除了为用户提供表达能力分数与简要分析参考之外，本模块还为题目积累了大量的用户数据。

题目的优化过程会根据分析模块积累的用户数据修正题目的参数。在本系统中，优化模块将定时扫描题库，使用这些用户数据，针对不同的参数使用不同的优化模型。这一部分的主要目的是反哺分析模块，提高分析模型的稳定性和准确性。

题目的来源并不是凭空想象、任意捏造的，完全由专家出题会增加许多专家工作量，提高系统的成本。为了解决题目素材问题，语料模块将定时从互联网中爬取下载资源，在简要的整理、清晰之后，自动地初始化其参数，使其成为题目的思路来源。这一模块的主要目的是减少专家出题的工作量，也使得题目的来源更广泛，适应面更广。

除了上面三个主要的、自动的生命周期流程之外，题目还需要人工进行日常管理。业务模块使用了传统的 web 应用模式，采用前后端分离的架构，提供了修改题目，查看题目列表等功能。

在技术实现上，系统使用 Vue 作为前端开发框架，Flask 作为后端开发框架，使用 Nginx 实现请求转发与负载均衡，使用 Redis 作为缓存数据库，并使用 MongoDB 数据库主从备份提高数据库容灾能力。主流技术与框架的使用不仅提高了开发效率，也减少了开发过程中出现的问题。在系统测试上，各个模块单独

设计并实施了功能测试和性能测试方案，完成了既定需求。在实验评估上，以分析模块为核心的表达能力评测模型取得了较好的效果，系统评分与专家评分绝对值在 5 分以内的比例达到了 78%，说明表达力评测模型评分可以一定程度上代替专家评分。

6.2 展望

本系统是表达能力数字化的一次尝试，也是综合素质评价量化的一次尝试，无论是在系统设计还是在系统实现上都存在一定的不足和改进空间。未来可以从以下四个方面出发进一步改善系统，并对其他综合素质，例如创造力、想象力等量化过程做出启发。

第一，在分析模块中，目前提取特征局限于较浅层次的音频特征和文本特征，大量能够反映表达能力的特征没有被注意到，例如音频的波形特征可以反映表达过程是否韵律感、节奏感，文本的语义理解可以反映表达内容的情感倾向，文本的语法分析可以反映说话时的逻辑性与结构性等等。目前这些特征暂时出于研究与证实阶段，在未来的迭代中可以逐步加入分析模型中。

第二，优化模块目前聚焦于级别、词库、难度等参数的修正，是一个无监督的学习过程。无监督学习的优点是无需事先准备好的样本和标签，但是也有学习效果较差的缺点。在未来规划中，优化模块可以接入专家标注或者众包标注，为一部分样本人工打标签，这样可以结合无监督学习和监督学习的优点，更好的优化题目的参数。

第三，语料模块目前存在提示较少、文本素材质量较低的问题，往往四到五个文本素材才能出现一个质量较高，足够专家出题的素材来源，这部分可以改进的点集中在语料来源和参数初始化两方面。语料来源方面，未来可以接入众包系统，使用人工输入的方式而不是网络爬取的方式获取；参数初始化方面，目前已经在研究更优化、更适合口语的的关键词提取方法。

第四，由于开发算法落地相关系统的经验不足，系统实现过程中还有一些地方存在复杂度较高、逻辑混乱、代码冗余的现象，在以后的迭代中，我们需要慢慢解决这些问题。

参考文献

[1] T. Davis, M. Cutt, N. Flynn, Talent assessment: A new strategy for talent management, Gower Publishing, Ltd., 2007.

[2] 李博, 孟成博, 对 hustoj 在线评测系统的若干优化与创新, 现代计算机 (专业版) (35) 49–52+58.

[3] A. Kurnia, A. Lim, B. Cheang, Online judge, Computers & Education 36 (4) (2001) 299–315.

[4] 李定才, 瞿绍军, 胡争, 段兵, 成幸毅, 唐强, Research on the safety of online judge system based on windows 计算机技术与发展 021 (9) 204–207.

[5] C. G. H. Suarez, V. A. Bucheli, F. Restrepo-Calle, F. A. Gonzalez, A strategy based on technological maps for the identification of the state-of-the-art techniques in software development projects: Virtual judge projects as a case study, in: Colombian Conference on Computing, Springer, 2018, pp. 338–354.

[6] J. L. Flanagan, Speech Analysis Synthesis and Perception, 1965.

[7] C. Forgie, M. L. Groves, F. C. Frick, Automatic recognition of spoken digits, Journal of the Acoustical Society of America 30 (7) 669–669.

[8] M. J. Beal, Z. Ghahramani, C. E. Rasmussen, The infinite hidden markov model, in: Advances in neural information processing systems, 2002, pp. 577–584.

[9] K.-F. Lee, Automatic speech recognition: the development of the SPHINX system, Vol. 62, Springer Science & Business Media, 1988.

[10] 梁迎丽, 梁英豪, 基于语音评测的英语口语智能导师系统研究, 现代教育技术 (11) 84–87.

[11] G. A. Young, D. H. Killen, Receptive and expressive language skills of children with five years of experience using a cochlear implant, Annals of Otology Rhinology & Laryngology 111 (9) 802–810.

[12] Z. Shirzadi, B. Stefanovic, H. J. Mutsaerts, M. Masellis, B. J. MacIntosh, A. D. N. Initiative, Classifying cognitive impairment based on the spatial heterogeneity of cerebral blood flow images, Journal of Magnetic Resonance Imaging 50 (3) (2019) 858–867.

[13] Butler, A. Frances, Eignor, Daniel, Jones, Stan, McNamara, Tim, K. Barbara, Toefl 2000 speaking framework: A working paper.

[14] 韦星, 高职学生口头表达能力评价体系初探, 时代教育 (23) 234–234.

[15] D. J. Hand, Principles of data mining, Drug safety 30 (7) (2007) 621–622.

[16] J. Han, K. Micheline, Data mining: concepts and techniques 5 (4) (2006) 1 – 18.

[17] R. Wirth, J. Hipp, Crisp-dm: Towards a standard process model for data mining, in: Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining, Springer-Verlag London, UK, 2000, pp. 29–39.

[18] L. Van Wel, L. Royakkers, Ethical issues in web data mining, Ethics and Information Technology 6 (2) (2004) 129–140.

[19] F. Kamiran, T. Calders, Data preprocessing techniques for classification without discrimination, Knowledge and Information Systems 33 (1) (2012) 1–33.

[20] C. Clifton, M. Kantarcioğlu, A. Doan, G. Schadow, J. Vaidya, A. Elmagarmid, D. Suciu, Privacy-preserving data integration and sharing, in: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, 2004, pp. 19–26.

[21] D. L. Olson, D. Delen, Advanced data mining techniques, Springer Science & Business Media, 2008.

[22] Y. Lee, R. Kennedy, Visual presentation technique for data mining software, uS Patent 6,269,325 (Jul. 31 2001).

[23] H. Liu, H. Motoda, Feature selection for knowledge discovery and data mining, Vol. 454, Springer Science & Business Media, 2012.

[24] S. Boriah, V. Chandola, V. Kumar, Similarity measures for categorical data: A comparative evaluation, in: Proceedings of the 2008 SIAM international conference on data mining, SIAM, 2008, pp. 243–254.

[25] Z. Xuegong, Introduction to statistical learning theory and support vector machines, Acta Automatica Sinica 26 (1) (2000) 32–42.

[26] E. Alpaydin, Introduction to machine learning, MIT press, 2020.

[27] P. Smolensky, Connectionist modeling: Neural computation/mental connections.

[28] I. T. Jolliffe, Principal components in regression analysis, in: Principal component analysis, Springer, 1986, pp. 129–155.

[29] G. A. Seber, A. J. Lee, Linear regression analysis, Vol. 329, John Wiley & Sons, 2012.

[30] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, M. Klein, Logistic regression, Springer, 2002.

[31] K. Gauss, Theoria motus corporum coelestium in sectionibus conicis solem ambientum (theory of motion of the celestial bodies moving in conic sections around the sun) (1963).

[32] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747.

[33] C. D. Manning, C. D. Manning, H. Schütze, Foundations of statistical natural language processing, MIT press, 1999.

[34] J. Hutchins, The history of machine translation in a nutshell, Retrieved December 20 (2009) (2005) 1–1.

[35] 黄昌宁, 赵海, 中文分词十年回顾, Ph.D. thesis (2007).

[36] 龙树全, 赵正文, 唐华, 中文分词算法概述, Ph.D. thesis (2009).

[37] D. Zhang, H. Xu, Z. Su, Y. Xu, Chinese comments sentiment classification based on word2vec and svmperf, Expert Systems with Applications 42 (4) (2015) 1857–1863.

[38] J. Ramos, et al., Using tf-idf to determine word relevance in document queries, in: Proceedings of the first instructional conference on machine learning, Vol. 242, Piscataway, NJ, 2003, pp. 133–142.

[39] R. Mihalcea, P. Tarau, Textrank: Bringing order into text, in: Proceedings of the 2004 conference on empirical methods in natural language processing, 2004, pp. 404–411.

[40] O. Filipova, Learning Vue. js 2, Packt Publishing Ltd, 2016.

[41] M. Grinberg, Flask web development: developing web applications with python, " O'Reilly Media, Inc.", 2018.

[42] J. L. Carlson, Redis in action, Manning Publications Co., 2013.

[43] D. Myers, J. W. McGuffee, Choosing scrapy, Journal of Computing Sciences in Colleges 31 (1) (2015) 83–89.

[44] K. Banker, MongoDB in action, Manning Publications Co., 2011.

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

[46] 王理嘉, 《汉语拼音方案》与世界汉语语音教学, 世界汉语教学 (2) (2005) 5–11.

简历与科研成果

基本情况 汤大业，男，汉族，1996 年 11 月出生，江苏省宿迁市人。

教育背景

2018.9 ~ 2020.6 南京大学软件学院 硕士

2014.9 ~ 2018.6 南京大学软件学院 本科

这里是读研期间的成果

1. 马徐骏，刘嘉，詹晨，孟磊，王浩宇，褚东宇，**汤大业**，王磊，“一种动态优化表达力评测题目的方法”，申请号：202010254255.6，已受理。
2. 马徐骏，褚东宇，**汤大业**，刘嘉，詹晨，孟磊，王浩宇，王磊，“一种用于快速表达力测试的智能化题目分类及推荐方法”，申请号：202010287246.7，已受理。
3. 马徐骏，刘嘉，詹晨，孟磊，王浩宇，褚东宇，**汤大业**，王磊，“一种为快速表达力测试生成题目的方法”，申请号：202010254294.6，已受理。

致　　谢

论文接近尾声之际，我首先想感谢刘嘉老师和陈振宇老师在项目开发和论文写作过程中给我的意见与指导。在项目开发过程中，两位老师会在百忙之中定期督促项目进度，并及时询问开发中遇到的挑战与问题，从专业角度给予我们帮助。在论文写作过程中，两位老师提出了许多建设性的意见为我答疑解惑，帮助我提升论文写作水平。感谢两位老师研究生期间对我的悉心指导，他们严谨的学术态度、开阔的产品视野和极强的行动能力使我受益终身。

我要感谢和我共同完成项目的褚东宇同学。他比我稍微年长一些，项目经验更加丰富，在开发过程中始终认真负责，在许多方面，尤其是软件工程方面给予了我很多帮助。

我要感谢马徐骏先生对我的帮助。马老师既是项目的发起人，又是表达能力领域的专家，马老师团队在表达能力专业领域给我们许多专业性、指导性的意见，帮助我们深入理解业务概念，更好地完成项目。

最后，我要感谢我的父母家人在论文写作期间给予我的关爱与呵护，在疫情期间伴我良多。同时，我也要感谢南京大学软件学院对我的培养，使我成长了许多。

版权与原创性说明

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

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

作者签名：汤小业

日期：2020 年 5 月 27 日