



泉州师范学院
QUANZHOU NORMAL UNIVERSITY

毕业论文（设计）

题 目 _____ 基于 Whisper 模型的
_____ 南音叫字智能化纠音研究 _____

学 院	数学与计算机科学学院
专 业	软件工程
学生姓名	王涛
学 号	231308115
指导教师	王鸿伟，魏云川

2025 年 12 月

基于 Whisper 模型的南音叫字智能化纠音研究

摘要

作为“中国音乐历史活化石”的南音，其传承长期受限于闽南语泉州音韵门槛，加之“口传心授”的传统教学模式难以规模化，其传承面临着受众萎缩与代际断层的挑战。如何在数字化浪潮中为非遗保护寻找新的突破口，成为当前亟待解决的课题。本研究尝试引入人工智能技术，构建了一套专注于南音发音矫正的深度学习评估系统，旨在通过计算机辅助手段破解语言隔阂与师资短缺的难题。

在系统架构层面，本研究设计了前后端分离的解决方案：后端依托 Spring Boot 框架保障业务逻辑的稳健性，前端则借助 Uni-app 实现多终端的无缝交互。针对南音作为低资源方言导致的数据匮乏问题，构建了专用语音数据集，并采用迁移学习策略对 Whisper 模型进行微调。实测数据显示，经微调后的 Whisper-large-v3-turbo 模型表现抢眼，将字错率（Word Error Rate, WER）成功压降至 0.83%，在南音识别领域展现出极高的应用价值。此外，为应对高并发场景下的实时反馈需求，系统通过 RabbitMQ 与 Redis 的深度整合，构建了异步分布式处理机制。该系统最终实现了一个涵盖“学习引导-模拟练习-智能评估”的完整闭环，不仅突破了传统时空限制，也为语言类非物质文化遗产的活态传承提供了可复制的数字化模式。

关键词：Whisper 模型；南音叫字；发音测评；深度学习；分布式系统

Research on Intelligent Pronunciation Correction of Nanyin "Jiaozi" Based on the Whisper Model

Abstract

As the "living fossil of Chinese music history," the inheritance of Nanyin has long been constrained by the steep phonological threshold of the Quanzhou Minnan dialect. Compounded by the difficulty of scaling the traditional "master-to-apprentice" oral teaching model, its transmission faces the dual challenges of a shrinking audience and a generational gap. Consequently, exploring new breakthroughs for the protection of Intangible Cultural Heritage (ICH) amidst the digital wave has become an urgent issue. This study introduces artificial intelligence technology to construct a deep learning assessment system dedicated to Nanyin pronunciation correction, aiming to overcome language barriers and teacher shortages through computer-assisted approaches.

At the system architecture level, this research designs a decoupled front-end and back-end solution. The back-end relies on the Spring Boot framework to ensure the robustness of business logic, while the front-end utilizes Uni-app to achieve seamless multi-terminal interaction. To address the data scarcity issue inherent in Nanyin as a low-resource dialect, a dedicated speech dataset was constructed, and a transfer learning strategy was employed to fine-tune the Whisper model. Empirical data indicates that the fine-tuned Whisper-large-v3-turbo model delivers exceptional performance, successfully reducing the Word Error Rate (WER) to 0.83%, which demonstrates immense application value in Nanyin speech recognition. Furthermore, to meet the demand for real-time feedback in high-concurrency scenarios, the system establishes an asynchronous distributed processing mechanism through the deep integration of RabbitMQ and Redis. Ultimately, the system realizes a complete closed loop encompassing "learning guidance, simulation practice, and intelligent assessment". This not only breaks through traditional spatiotemporal limitations but also provides a replicable digital paradigm for the living transmission of linguistic intangible cultural heritage.

Keywords: Whisper Model; Nanyin; Pronunciation Assessment; Deep Learning; Distributed System

目录

摘要	I
Abstract	II
1 绪论	1
1.1 研究背景与意义	1
1.2 国内外研究现状	2
1.2.1 语音识别技术与测评技术现状	2
1.2.2 微服务架构与 Web 开发技术现状	3
1.3 课题目的与意义	3
1.4 本文组织结构	4
2 相关技术简介	6
2.1 后端开发技术	6
2.1.1 Spring Boot 框架	6
2.2 前端开发技术	6
2.2.1 Uni-app 跨平台框架	6
2.2.2 Vue.js 渐进式框架	6
2.3 人工智能与算法技术	7
2.3.1 Transformer 架构	7
2.3.2 Whisper 语音识别模型	7
2.3.3 动态规划算法	7
2.4 中间件与数据库技术	7
2.4.1 RabbitMQ 消息队列	7
2.4.2 PostgreSQL 和 Citus	7
3 系统分析	8
3.1 需求分析	8
3.1.1 功能需求分析	8
3.1.2 核心交互流程	8
3.1.3 数据建模	11
3.2 非功能需求	11
3.2.1 安全要求	11
3.2.2 可用性与可靠性要求	11
4 系统设计	13
4.1 体系结构设计	13
4.2 模块功能设计	14
4.2.1 用户管理模块	14
4.2.2 练习测评模块	14
4.2.3 专家采集模块	14
4.2.4 后台管理模块	14
4.3 数据库设计	15
4.3.1 实体映射	15
4.3.2 数据库表结构	15
5 核心算法与模型优化	18
5.1 数据集的创建与处理	18
5.1.1 数据采集与分布	18
5.1.2 数据标注与格式化	19

- 5.2 模型选择与训练策略19
 - 5.2.1 基座模型架构对比 19
 - 5.2.2 模型微调与训练 20
- 5.3 算法实现22
 - 5.3.1 评价指标定义22
 - 5.3.2 动态规划序列对齐算法22
- 5.4 实验结果与分析23
 - 5.4.1 模型性能评估23
 - 5.4.2 典型误识别示例分析 24
- 6 系统实现 25
 - 6.1 开发环境与部署配置25
 - 6.2 用户管理模块25
 - 6.2.1 用户登录25
 - 6.2.2 用户注册27
 - 6.3 练习测评模块28
 - 6.3.1 歌曲选择页面28
 - 6.3.2 练习界面29
 - 6.3.3 完成练习界面31
 - 6.4 学习记录 33
 - 6.5 专家采集模块 34
 - 6.5.1 采集歌曲选择页面 34
 - 6.5.2 歌曲录音采集页面 35
 - 6.6 后台管理模块36
 - 6.6.1 管理员用户记录模型评分功能37
 - 6.6.2 标准音频审核39
- 7 总结与展望 42
 - 7.1 总结 42
 - 7.2 展望 42
- 参考文献43

1 绪论

1.1 研究背景与意义

中华优秀传统文化为中华民族的精神命脉，非物质文化遗产属于其重要组成部分，蕴含着独特的民族记忆与艺术价值，南音又称为“弦管”，源于福建闽南地区，被称作中国音乐历史的“活化石”。而且已被列入联合国教科文组织人类非物质文化遗产代表作名录。党和国家高度重视文化遗产的保护与传承工作，习近平总书记曾深刻指出：中华文化是我们提高国家文化软实力最深厚的源泉，是我们提高国家文化软实力的重要途径。要使中华民族最基本的文化基因与当代文化相适应、与现代社会相协调，以人们喜闻乐见、具有广泛参与性的方式推广开来，把跨越时空、超越国度、富有永恒魅力、具有当代价值的文化精神弘扬起来，把继承传统优秀文化又弘扬时代精神、立足本国又面向世界的当代中国文化创新成果传播出去。要系统梳理传统文化资源，让收藏在故宫里的文物、陈列在广阔大地上的遗产、书写在古籍里的文字都活起来^[1]。这一重要论述为南音的保护与传承指明了方向，即要将传统文化资源与现代生活紧密结合，赋予其新的时代生命力。

然而，在具体的传承实践中，方言文化的保护面临着诸多挑战。王莉宁等指出：方言文化既是非物质文化遗产的重要组成部分，也是中华优秀语言文化的精粹。其认为：方言文化艺术是最能激发社会大众，尤其是青少年方言文化保护意识的一种方式，是语保社会化最有效的方式之一；也能转换为具有市场价值和经济效益的重要资源，助推文化产业发展。尽管如此，在现代化和城镇化极速发展进程中，传统方言的使用场景不断缩减，依靠方言存活的地方戏曲和曲艺正遭遇着生存考验^[2]。这种生存危机不仅体现在受众的萎缩上，更体现在传承机制的困境中。林琰等的研究进一步指出：非物质文化遗产通过技艺、文学、艺术等形式，成为了中华文明绵延传承的生动见证；在文化强国建设中，非物质文化遗产通过群众参与、合作、交流等形式，丰富了人民群众的精神生活，成为了涵养社会主义核心价值观、坚定民族文化自信的重要源泉；但同时也强调，当前非遗保护面临商业化开发过度与传承人“后继乏人”并存的矛盾，亟需探索高质量的活化路径^[3]。

具体到南音，“活态传承”的核心难点与切入点在于其独特的“叫字”传统。所谓“叫字”，是南音演唱艺术中特有的音韵与咬字规范，也是决定南音唱腔是否地道、是否具备传统神韵的关键所在。南音“叫字”要求演唱者遵循泉州古方言，以鲤城区口音为标准。在南音“口传心授”的传承体系中，“叫字”是衡量演唱者艺术造诣的首要标准，它要求每一个字的发音既要符合古汉语八音的严谨结构，又要兼顾南音旋律润腔的艺术美感。因此，南音叫字具有极高的音韵学价值与精细的声学特征。

针对这一现状，非遗保护的政策导向也在发生转变。黄永林等梳理了我国非遗保护政策的动态演变，指出其历程：新世纪以来，从我国非遗保护政策动态演变看，由“十五”期间“抢救保护”到“十一五”

时期“重点保护”的“保护为主”阶段，发展到从“十二五”时期“生产性保护”到“十三五”时期鼓励“活化利用”的“合理利用”阶段。体现了国家对非遗“文化价值”的重视到对“经济价值”的发现的认知逻辑，凸显了从强调“抢救第一”到加强“重点保护”再到鼓励“合理利用”的政策发展逻辑。其在分析中国非遗保护政策时强调，新时代的非遗工作应从单纯的“抢救性保护”向“活化利用”与“融合发展”转变，让非遗全面融入人民生活^[4]。

具体到南音这一艺术形式，其传承的核心难点在于语言。南音演唱需严格按照闽南语泉州音韵来演唱，其特有的咬字，发声方法包含了南音艺术主要的审美特点。然而，随着普通话不断普及，人口流动愈发频繁，年轻一辈以及非闽南语出身的学习者在学唱南音的时候，常常会陷入到严重的“语音障碍”。作为闽南文化的瑰宝，泉州南音凝聚着千年的艺术积淀和文化记忆。作为闽南文化的瑰宝，泉州南音凝聚着千年的艺术积淀和文化记忆，如何突破语言瓶颈，成为传承的关键。对此，李娇等学者指出：进入数字化时代，这一古老的曲种面临着传承模式转型的紧迫任务。传统的师徒口传心授方式难以适应现代传播需求，而新媒体技术的发展为南音的活态传承提供了新的可能^[5]。

数字化不仅仅是一种传播手段，更是一种生存方式。杨红等的研究表明：非遗项目的活态存续仍旧是非遗保护的首要目的，通过数字化手段进行抢救性保存有利于传承实践的恢复、文化资源的留存、人类记忆的延续。她进一步指出，非遗数字化传播正呈现出“视频化与体验感”及“数字化生存”的新趋势，技术革新为非遗在当代的存续提供了内源动力^[6]。

由此可见，利用人工智能技术针对濒危语言或者方言实施语音识别与评价，已经成为当下学术界和产业界瞩目的焦点。基于此，本研究旨在创建起一个可以帮助学习者纠正南音发音的智能系统，这对于南音艺术的活态传承来说有着非常关键而且迫切的实际意义。

1.2 国内外研究现状

1.2.1 语音识别技术与测评技术现状

在语音识别领域，技术的发展正经历着从传统模型向深度学习模型的深刻变革。鱼昆等在分析行业趋势时指出：当前深度语音识别研究的主要趋势就是不断深入研究端到端模型及其各种优化方法，同时，探索用于迁移学习的工具和数据策略，使得语音识别也能像机器视觉领域那样遍地开花，大大提高整个社会的人工智能化水平。具体而言，自动语音识别（Automatic Speech Recognition, ASR）技术近年来已从早期的高斯混合模型-隐马尔可夫模型（Gaussian Mixture Model-Hidden Markov Model, GMM-HMM）发展到深度神经网络（Deep Neural Network, DNN），而目前端到端（End-to-End）语音识别架构逐渐成为主流^[7]。

在端到端架构的演进中，流式识别和 Transformer 架构是两个重要的里程碑。王澳回等在综述中指出：端到端流式语音识别模型因其能一边输入语音一边输出结果，大幅减少了交互延迟，已成为语音识别领

域的研究热点^[8]。而在模型结构方面，其中谷歌团队 2017 年提出的 Transformer 架构彻底革新了序列模型的构建方式：该架构独特的自注意力机制显著提升了长时语音数据的捕捉能力^[9]。基于此架构，OpenAI 推出的 Whisper 模型采用了该架构模型通过大规模弱监督数据集训练，展现出卓越的多语言识别能力与高鲁棒性^[10]。特别值得一提的是，其对环境噪声干扰，口语差异的抗干扰能力突出，为资源有限的方言语音识别提供了极具价值的创新技术方案。

在应用层面，语音技术与教育的结合催生了计算机辅助发音训练（Computer-Assisted Pronunciation Training, CAPT）系统，该技术在第二语言习得中已展现出成熟的应用价值^[11]。然而，随着应用场景的拓展，测评标准也面临新的挑战。李玮等的研究显示，随着智能产品层出不穷，面向语音交互技术的测评标准日益重要，涵盖了从语音识别到自然语言理解的多个关键环节^[12]。反观目前的 CAPT 研究多聚焦于英语、标准汉语等高资源语种，针对中国戏曲方言及其复杂声调的测评研究尚显匮乏，存在明显的学术空白。与此同时，评价算法也在经历技术迭代：不同于传统算法侧重于后验概率与对数似然比的计算，当前基于深度学习技术的字错率（WER）及音素错误率（Phoneme Error Rate, PER）指标，凭借其更精细的颗粒度，正逐渐成为语音质量评估领域的新主流标准。

1.2.2 微服务架构与 Web 开发技术现状

随着互联网应用生态变得庞大之际，软件架构正在从传统的单体式朝着微服务架构转变。这种转变不仅仅是架构的调整，更是开发模式的革新。冯志勇等指出：微服务具有独立进程、轻量级通信和独立部署环境等显著的特性；将系统整体功能分解到单个微服务中以实现系统的解耦；这种方式不仅可以降低系统的耦合性，提升系统的内聚性，而且减少服务交互的成本，提供更加灵活的服务支持^[13]。

在 Web 开发实践中，技术栈的选择也日益标准化。“前后端分离”已经成为整个行业普遍认可的开发准则，在后端开发时，Spring Boot 框架凭借自身创建独立应用速度快的特点被当作第一选择，而且如果再加上 MyBatis-Plus 框架，就可以极大地减轻持久层的操作压力。对于这种架构优势，霍福华等人经研究显示：微服务是一种全新的软件开发框架，使得可以很好的让较为庞大的软件产品，分为诸多较为零散的小型应用，这样就可以让软件在得到开发的过程中，降低其高耦合的问题。其次，这样精细化的划分模式下，也相应的保障对产品进行耦合程度的降低，让开发人员更加高效率的进行开发，以此降低了研发的经费投入量^[14]。而就前端交互而言，运用 Vue.js 结合 Uni-app 框架，可以适配多种终端，这样就能较好地满足用户在各类设备之间流畅测评的需求。

1.3 课题目的与意义

综上所述，本研究尝试构建起基于深度学习技术的南音发音智能测评系统，旨在解决南音传承中的“语言难关”与“师资短缺”问题。其主要思路在于以微调过的 Whisper 模型当作声学识别引擎，并利用动态规划算法来执行音素层级的发音对比纠错。系统并非简单由一些单一功能模块拼凑而成，而是深

度融合了诸如语音收集，自动化评判，学习路径分析以及专家数据录入等功能，目的在于塑造起一个具备“学-练-评-采-审”关联性的闭合生态系统，从而给南音教学供应一种有力的支撑手段。

从文化遗产的宏观角度而言，本课题属于人工智能技术助力非物质文化遗产保护的革新实践。利用数字化手段减轻南音学习里最繁杂的方言阻碍，有益于考察传统艺术教育的现代化转型路径，可以有效地促使年轻人群经由数字途径认识和掌握南音，这对改善当下南音传承人缺失，受众范围缩小的实际情况有着积极而悠远的影响。

从应用实践的角度来看，闽南语作为典型的地域性语言，其资源相对匮乏，相关语音数据集规模较小。本研究为此专门构建了南音语音数据集，并将迁移学习与微调技术相结合，应用于大型预训练模型，从而验证了 Whisper 模型在地域性语言应用中的潜力。同时，该系统采用了异步解耦架构设计，为涉及大模型推理的 Web 应用开发提供了可参考的工程实践方案。

1.4 本文组织结构

本文围绕基于 Whisper 模型的南音叫字智能化纠音系统的设计与实现展开深入研究，全书共分为七个章节，各章节的具体内容组织如下：

第一章：绪论。本章首先阐述了人工智能助力非物质文化遗产南音数字化保护的研究背景与现实意义。随后，梳理了国内外在自动语音识别（ASR）技术、计算机辅助发音训练（CAPT）系统以及微服务与 Web 开发领域的现状与技术演进，分析了当前低资源戏曲方言评测体系的学术空白。最后，明确了课题的研究目的及实际应用价值，并概述了本文的整体行文结构。

第二章：相关技术简介。本章对构建南音智能测评系统所涉及的核心技术与算法理论进行了系统阐述。在后端层面介绍了 Spring Boot 框架的自动配置与统一调度机制；前端层面分析了 Uni-app 跨平台框架与 Vue.js 的响应式视图状态管理；人工智能层面深入解析了 Transformer 核心架构、Whisper 语音识别模型的预训练优势以及编辑距离动态规划算法；最后介绍了 RabbitMQ 消息队列、PostgreSQL 数据库与 Citus 分布式扩展插件等中间件技术，为后续的系统分析与设计奠定技术理论基础。

第三章：系统分析。本章对南音智能纠音系统进行了全方位的需求解构。首先，针对“学习者、专家、系统管理员”三类核心用户，梳理并确立了系统功能需求清单；其次，详细设计了以“学生沉浸式练习”为核心的异步推理全流程交互逻辑；最后，通过构建核心组件的 E-R 图完成了系统的静态数据建模，并从安全凭证加密（JSON Web Token, JWT）以及高并发“削峰填谷”两方面提出了系统的非功能性需求。

第四章：系统设计。本章完成了系统由业务需求向工程实现的结构转化。首先，基于前后端分离与微服务的混合架构理念，将系统划分为接入层、业务逻辑层、智能服务层和数据持久层四个逻辑层级，实现了计算与业务的松耦合；其次，细化了用户管理、练习测评、专家采集和后台管理四大模块的功能

设计；最后，规范了底层数据库的物理结构设计，详细定义了用户表、歌曲表、歌词表、练习会话表及练习结果详情表等核心表结构。

第五章：核心算法与模型优化。本章是本文算法创新的核心所在。针对南音作为低资源方言导致的“数据孤岛”难题，详细描述了通过专家众包采集并精细标注的南音专属语音数据集的构建与分布情况。同时，阐述了对 Whisper-large-v3-turbo 模型进行迁移学习与有监督微调的训练策略。此外，本章深入推导了基于动态规划的序列软对齐算法，定义了字错率（WER）与音素错误率（PER）的量化评估标准，并对独立测试集上的实验结果及典型误识别示例的成因展开了详细的对比与分析。

第六章：系统实现。本章是对南音叫字智能测评系统的整体工程落地表现。首先给出了跨平台服务集群的开发环境与容器化部署配置；随后，结合具体的系统界面截图，详细展示了用户管理、歌曲浏览与练习测评、专家范读音频采集、管理员后台内容审核及评分人工干预等核心业务模块的前后端联调细节，并给出了基于 Vue.js 的异步请求表单校验、Java 后端 JWT 令牌分发、RabbitMQ 任务轮询以及 StreamAPI 数据聚合计算的核心代码实现。

第七章：总结与展望。本章对全文的研究内容、工程实践及算法优化成果进行了总结，分析了微调大模型在地域性方言应用中的有效性。在此基础上，指出了系统在冷门曲牌泛化能力、高并发推理时延以及多模态视觉纠错等层面的客观局限性，并探讨了未来利用大模型端侧轻量化部署以及“音-画”双模态交互技术实现南音数字教育生态深层次发展的演进方向。

2 相关技术简介

2.1 后端开发技术

2.1.1 Spring Boot 框架

Spring Boot 框架是 Java 生态系统的核心部分，该框架遵循“约定优于配置”这一设计原则。该框架采用智能自动配置策略，极大地简化了传统 Spring 应用里复杂的 XML 设置流程，而且依靠自身包含的 Tomcat, Jetty 等 Servlet 容器，做到了把 Web 应用压缩成轻量级的独立 JAR 包。霍福华等也指出：在基于 SpringBoot 框架体系下，采用的 MVC 设计方式，可以很好的利用模型的方式，将其在传统的框架基础上，进行简化处理。在项目配置工作中，采用的自动化配置方式，可以让 SpringBoot 框架体系中，实现高效开发的效果^[14]。在微服务架构体系当中，这个框架既供应了稳定的依赖管理以及事务调控功能，又能够有效地支持 RESTful API，从而顺应不断变化的业务需求，本系统中，后端就是依靠 Spring Boot 搭建起来的，构建了用户认证，数据保存以及消息队列交互这些关键业务模块的统一调度。

2.2 前端开发技术

2.2.1 Uni-app 跨平台框架

Uni-app 是一款基于 Vue 框架开发的跨平台前端框架。开发者只需编写相同代码，编译器便会自动将其转换为 iOS、Android 及微信小程序等平台的原生代码。通过封装原生功能、提供丰富组件库和完整应用程序编程接口（Application Programming Interface，API），实现跨设备的统一用户体验。系统采用 Uni-app 框架开发界面，确保在移动设备上运行，支持录音、音频播放等功能，前后端交互如图 2-1 所示。

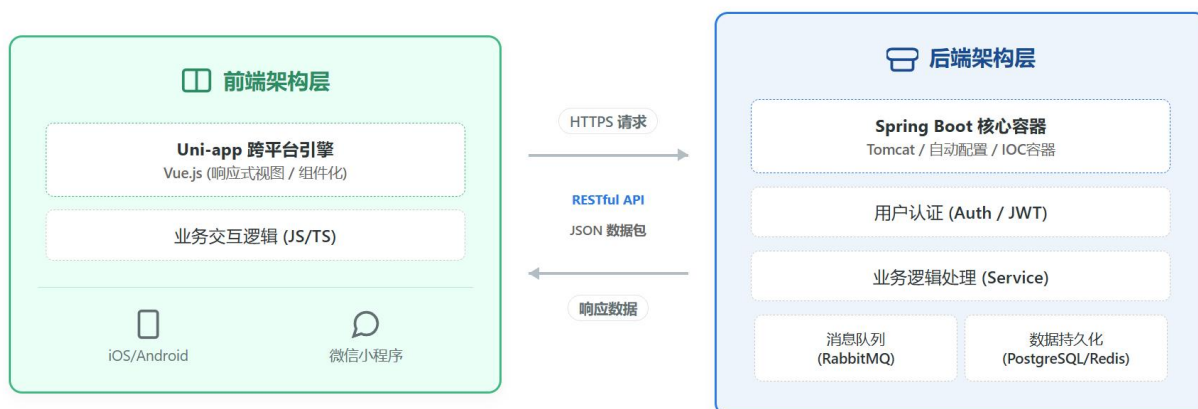


图 2-1 前端与后端架构的数据流示意图

2.2.2 Vue.js 渐进式框架

Vue.js 是一款先进的 JavaScript 框架，专为构建用户界面而设计。其核心库专注于视图层，采用自下而上的开发模式，通过双向数据绑定和组件化开发，能够高效管理数据状态并直接复用 UI 组件。在 Uni-app 架构中，Vue.js 负责响应式视图更新和组件生命周期管理，从而支持客户端的业务交互逻辑。

2.3 人工智能与算法技术

2.3.1 Transformer 架构

Transformer 架构是当今自然语言处理和语音识别技术的核心基石。它摒弃了传统的循环神经网络（Recurrent Neural Network, RNN）和卷积神经网络（Convolutional Neural Network, CNN）结构，转而全面运用自注意力机制，精准捕捉输入序列中的长距离关联。该架构由编码器和解码器组成，支持并行计算，显著提升了模型训练和推理的效率。本研究采用的语音识别模型底层基于 Transformer 架构设计。

2.3.2 Whisper 语音识别模型

Whisper 是 OpenAI 开发的通用语音识别模型，通过涵盖多语言、多场景的监督数据进行训练，总训练时长超过 68 万小时，具备卓越的健壮性和适应性，可应对多种环境挑战^[10]。除支持多语言语音转文字外，还集成了语音翻译与语种识别功能。即便在嘈杂环境、不同口音或专业术语场景下，其识别准确率仍显著优于传统自动语音识别（ASR）模型。本研究针对 Whisper 模型进行专项优化，可精准识别闽南语发音并生成国际音标（International Phonetic Alphabet, IPA）。

2.3.3 动态规划算法

动态规划是一种算法思想，通过将复杂问题分解为子问题来解决。在语音评估中，系统采用动态规划的编辑距离算法（Levenshtein Distance），通过比较标准语音序列与识别语音序列，实现精准匹配。通过构建二维矩阵和反向寻优路径，系统能精确计算替换、删除和插入的次数，从而得出可量化的语音质量指标——字错误率（WER）和音素错误率（PER）。

2.4 中间件与数据库技术

当前，微服务技术已成为构建复杂应用系统的主流选择，其发展正朝着标准化、轻量化和智能化的方向演进^[13]。本系统引入了以下关键中间件以支撑微服务架构的高效运行。

2.4.1 RabbitMQ 消息队列

RabbitMQ 是一款开源消息队列，支持高级消息队列协议（Advanced Message Queuing Protocol, AMQP），兼容点对点、发布/订阅等多种通信模式。它能有效实现分布式应用的异步通信与功能解耦，作为 Java 后端与 Python 人工智能服务之间的桥梁。当需要处理耗时较长的语音评估任务时，系统会将消息序列化后发送至 AI 服务，实现异步处理。该设计方案能有效避免高并发场景下的请求阻塞，显著提升系统处理能力。

2.4.2 PostgreSQL 和 Citus

PostgreSQL 是一款开源高性能对象数据库，以卓越的稳定性和高效处理复杂查询著称。其扩展功能 Citus 可将单服务器 PostgreSQL 数据库升级为分布式数据库，实现分布式存储与并行查询处理。

3 系统分析

3.1 需求分析

3.1.1 功能需求分析

该系统以南音发音的“学-练-评-采-审”这五个核心环节为框架，针对不同身份的用户群体，系统预设了各自对应的功能模块。大致可以把用户划分为三类，即一般的学习者也就是学习者，专业的指导者或者研究者即专家，以及负责系统整体运作的人员即系统管理员。

学习者是该系统的主要用户群体。李娇等学者曾表示，数字媒体技术的参与给非物质文化遗产的“活态传承”增添了新动力^[5]。所以系统核心功能需求集中在自主学习和发音训练上。Rogerson-Revell 的研究表明，计算机辅助语音训练（CAPT）在解决发音矫正的实时反馈方面具有显著优势^[1]。同时，针对传统南音教学中“重技艺轻理论、重口传轻辅助”的问题，陈恩慧提出应创新教学模式，利用现代技术辅助南音演唱教学^[15]。因此，系统需提供训练界面配备标准发音播放功能，并支持用户录制个人发音。系统需自动评估录音质量，直观显示准确率及元音标注正确率等关键指标。此外学习者还需能查看自己的学习进度，随时掌握学习进展和成果。

专家的主要职责是为系统提供高质量的标准语音数据库。专家登录系统后，将进入个人录音界面，界面会显示每首歌曲或歌词的文本内容。专家需按照标准准确录制语音，并根据显示内容录制清晰的示例音。录音完成后，语音数据将上传至后台系统的审核队列，作为后续开发和评估模型的参考数据。

系统管理员的主要职责是维护系统并审核内容。可查看专家提交的音频文件，听取音频样本后决定是否批准。通过审核的音频文件将被纳入系统标准音库。此外，系统管理员还能管理用户账户、歌曲信息等基础数据。

表 3-1 系统功能需求清单

需求序号	使用用户	所属模块	功能描述
1	学习者	学习与发音训练	访问南音曲库，选择曲目进行逐句跟读练习。
2	学习者	学习与发音训练	播放标准发音，录制个人发音并获取自动测评。
3	学习者	学习与发音训练	查看个人学习进度。
4	专家	标准音频采集	登录系统，进入录音界面。
5	专家	标准音频采集	准确录制高质量的南音标准发音示例。
6	专家	标准音频采集	将录制的音频数据上传至后台审核队列。
7	系统管理员	内容审核	查看并试听专家提交的音频文件，审核是否标准。

3.1.2 核心交互流程

系统交互逻辑以学习者的“沉浸式练习”为核心，一旦用户选取了南音曲目，系统就会立即启动训练会话，同时加载歌词文本以及国际音标，而且给予标准读音播放的功能。当用户激活录音控件来执行跟唱之后，音频流会自动被封装起来并上传到服务器端。

后端服务器接收音频数据之后，并不是立即展开计算，而是采用异步处理策略。系统将带有音频路径，用户标识以及曲目元数据封装成消息对象，推送进消息队列缓冲区。该队列由 AI 推理服务来消费，AI 推理服务会从队列当中取出任务，并使用 Whisper 模型去执行声学特征对比。经过计算得出的评分结果会采取双重保存策略，其一保存到 PostgreSQL 数据库里面，其二写入到 Redis 缓存之中从而优化前获取速度。前端依靠轮询机制随时取得反馈信息，把原先单调的评分转变成可视化的字音正确与否标识以及综合得分。用户按照此信息决定是否重新录制或者向前推动学习进度，直到系统产生出最终的练习分析报告为止，这样就构成了一个完整的学习循环，具体流程如图 3-1 所示。

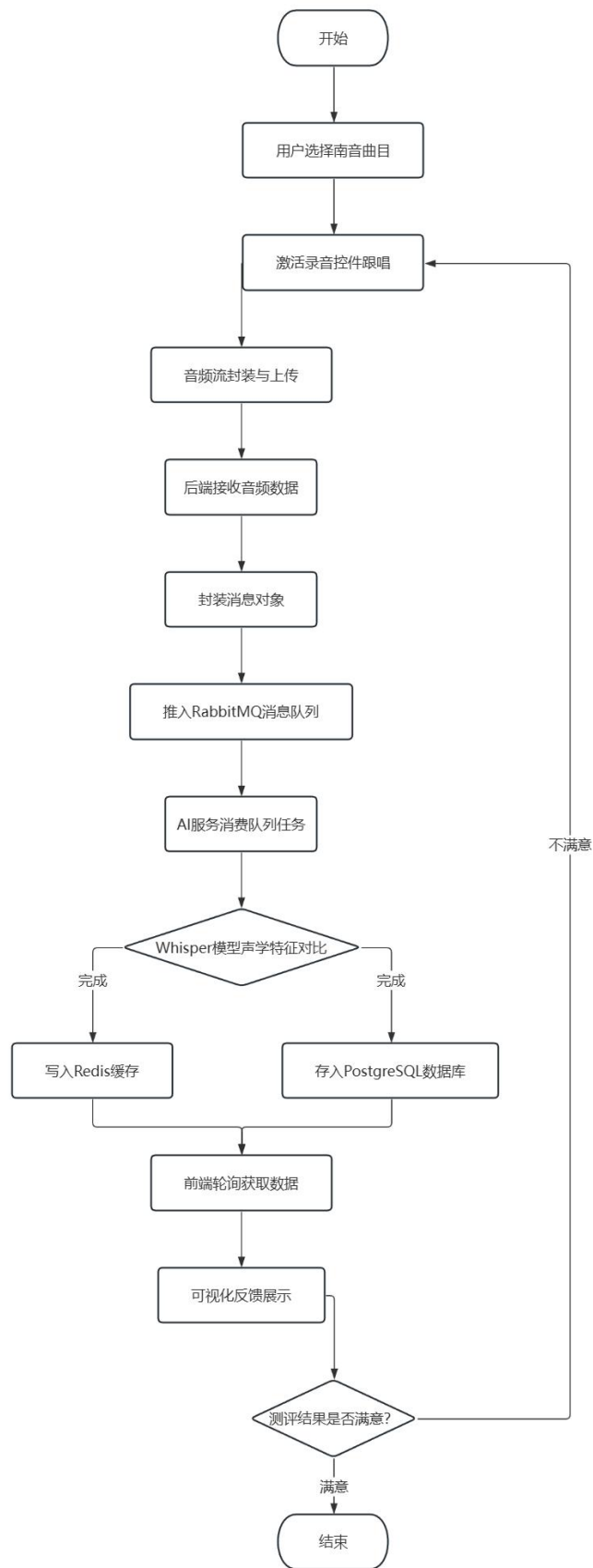


图 3-1 学习者发音评估流程示意图

3.1.3 数据建模

数据建模旨在构建系统的核心组件及其关联要素。从数据维度来看，系统核心组件包括用户、歌曲、歌词、练习、练习结果以及标准录音等。

对象间的关联关系错综复杂且相互重叠。例如，用户可能参与多次训练，每次仅限于一首歌曲，而同一首歌可能包含多个句子。每次训练会产生多个结果项，每个结果项代表对各句子的评分。专家（特殊用户）可录制多套标准录音，每套录音对应特定句子。这些对象间的关联关系构成了系统的静态数据结构，具体数据模型如图 3-2 所示。

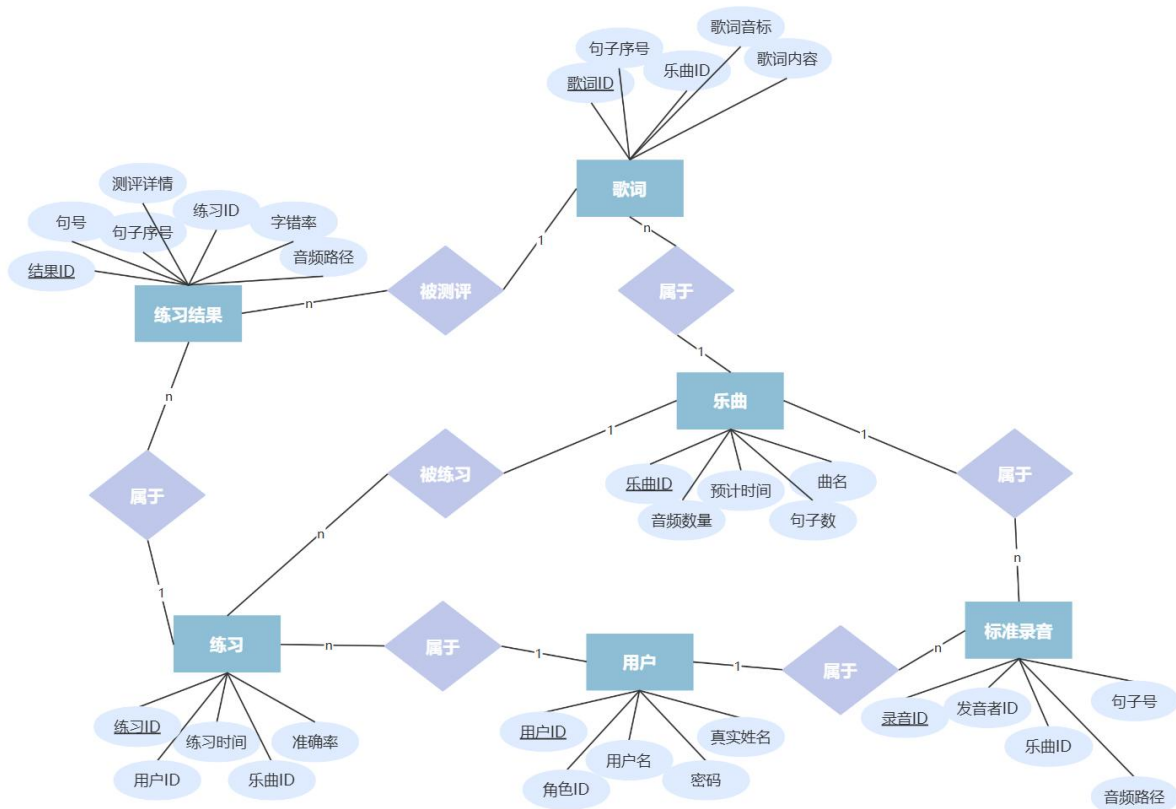


图 3-2 系统核心组件的 E-R 图

3.2 非功能需求

3.2.1 安全要求

数据安全是系统的生命线，用户敏感凭证（譬如密码）存放在数据库时需采用加盐哈希形式加密存储，张墨涵等学者曾就此展开剖析，微服务架构虽优化了灵活性，但也暴露出服务间通信截获之类的新安全风险^[16]。依托此，本系统塑造起依靠 JSON Web Token（JWT）的无状态认证体系，并要求所有受保护 API 执行令牌有效性的校验，而且，运用角色划分的访问控制（Role-Based Access Control，RBAC）模型，对学员，专家以及管理职员执行严格的权限分离，从根本上防止越权访问漏洞的发生。

3.2.2 可用性与可靠性要求

在架构设计方面，采用核心业务和计算服务物理隔离的办法，AI 服务作为一个单独的组件来运行。

就算因为模型推断产生堵塞或者短暂停止，也不会妨碍主要业务流程，比如录音上传，各种任务经由消息队列执行持久化缓冲，做到“削峰填谷”，而且，数据库和缓存层应当可扩展为高可用集群，规避单点故障的风险。

4 系统设计

4.1 体系结构设计

该系统依照现代分布式软件工程理念，舍弃传统单体架构，采取前后端分离并结合微服务的混合架构，利用异步消息中间件创建起松耦合，高内聚的分布式应用生态。倪小璐等人研究表明，在微服务体系当中，合理划分服务边界以及编排机制对于系统稳定性十分关键^[17]。在此基础上，系统整体架构被划分为四个逻辑层级：接入层，业务逻辑层，智能服务层和数据持久层。

（1）接入层（Access Layer）：依靠 Uni-app 跨平台框架创建起来的移动端门户，此层直接针对终端用户，凭借 HTTPS 加密协议与后端实施交互。Uni-app 具有“一次开发，多端部署”的特点，这抹去了 Android，iOS 和小程序这些平台在底层存在的差别，保障了多端用户得到一致的体验。

（2）业务逻辑层（Business Logic Layer）：是由 Spring Boot 框架支持的系统控制核心，此层经由 RESTful API 向外界展示服务，集中处理包含用户认证，会话守护，元数据保存等大量交互的请求。其关键设计思路是“轻重分离”，即业务层并不直接承担耗时较长的语音识别计算任务，而是起到调度器的作用，负责执行权限验证并分解任务，再把计算任务转交给中间件，以此来维持高并发情况下的吞吐能力。

（3）智能服务层（AI Service Layer）：依靠 Python 环境单独部署的算法引擎，此层包含经由领域适应性微调过的 Whisper 模型，该模型充当消息队列的消费者独自运行，它的任务是从 RabbitMQ 获取待处理的音频，展开 ASR 识别及音素对齐操作，凭借动态规划算法创建评分报告。借助 RabbitMQ 做到异步解耦，从而该层能够依照负载状况实施扩缩容，给系统带来了弹性扩展能力。

（4）数据持久层（Data Persistence Layer）：是系统的数据中枢与底层保障，采用“多模态存储与读写分离”的设计策略。针对结构化的业务数据（如用户信息、歌曲/曲目元数据、唱词结构及练习记录），系统部署了 PostgreSQL 结合 Citus 的分布式数据库集群，通过水平分片与多节点协同，攻克了高并发场景下关系型数据的吞吐瓶颈；此外，为缓解底层数据库的并发读取压力，系统引入 Redis 作为高速缓存介质，对高频访问的词句数据及热点元数据实施缓存守护，从而确保了整个分布式生态在数据访问层面的高可靠与低延迟，具体系统概览如图 4-1 所示。

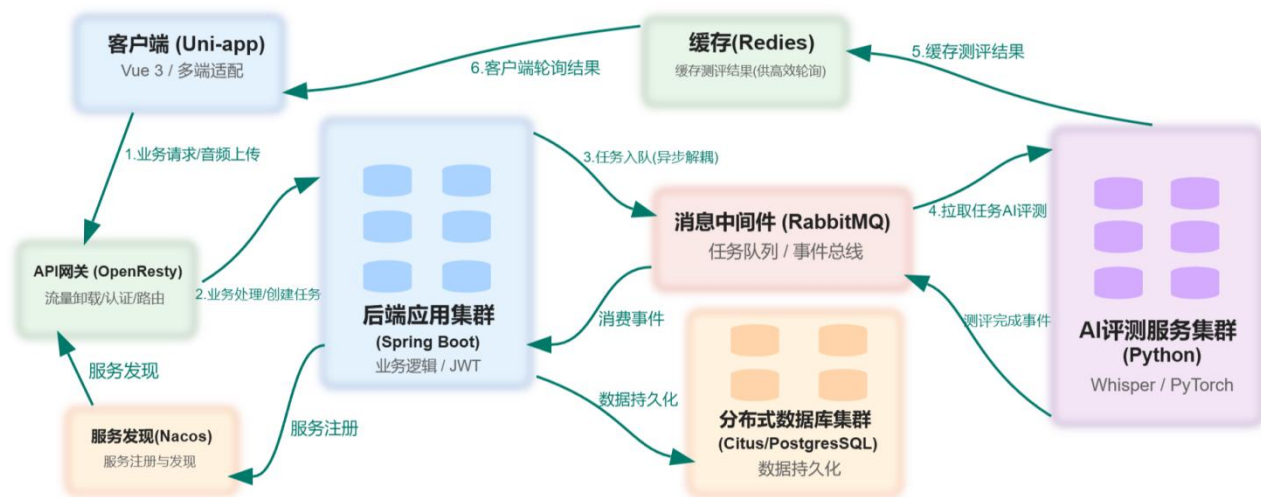


图 4-1 系统概览图

4.2 模块功能设计

4.2.1 用户管理模块

用户管理模块全权负责用户身份及权限的全过程，其中关乎注册，登录以及角色管理等关键功能。注册功能可使新用户凭借用户名和密码创建账户，系统利用密码哈希技术把密码转换为安全存储的哈希值来存储，登录时依靠 JWT 机制，用户经过身份认证之后，服务器会生成含有用户 ID 的 Token，而且在后续的所有请求当中都要用这个 Token 来做身份验证。权限管理采取三级分级机制，即普通学习者，南音专家和系统管理员。经由过滤机制自动拦截权限不符合的请求，从而保证用户只能访问其所被授权范围之内的资源。

4.2.2 练习测评模块

训练评估模块是学习者的主要学习模块，包含四大核心功能：“歌曲浏览”、“录音试听”、“实时评分”和“结果展示”。歌曲浏览功能可从数据库中调取带歌词的完整曲目，并支持分页搜索。录音试听功能通过调用移动设备的原声录音 API，实现用户声音的采集与上传。实时评分功能作为模块核心，采用异步处理技术进行数据处理。客户端完成语音录音后，系统将切换轮询模式。服务器端会将任务推送到消息队列，待 AI 服务处理完毕后，结果会被写入缓存。最终，客户端将获取到包含字错误率（WER）和详细语音对比的评估报告，并进行展示。

4.2.3 专家采集模块

专家录音模块是专为打造高品质标准音数据集而设计的专业服务。当专家用户使用时，系统会显示可录制的音乐和文本。该模块具备三大核心功能：录音、试听播放以及提交审核。与学员练习时不同，专家提交的音频文件不会立即进入评估环节，而是会以“待审核”状态存入数据库，等待管理员确认。该模块设计的核心目标在于确保录音过程高效顺畅，同时保证上传音频文件符合标准化质量要求。

4.2.4 后台管理模块

管理控制台模块是系统管理员用于管理系统运行和内容审核的核心工具，主要具备两大功能：语音

审核与基础数据管理。在语音审核界面，系统会实时显示专家提交的音频样本，管理员可在线试听后选择“通过”或“拒绝”。通过审核的音频将加入模型训练集。基础数据管理模块支持歌曲数据的增删改查功能。歌词内容与用户账户信息，以确保系统基础数据的准确性和时效性，系统整体模块结构如图 4-2 所示。

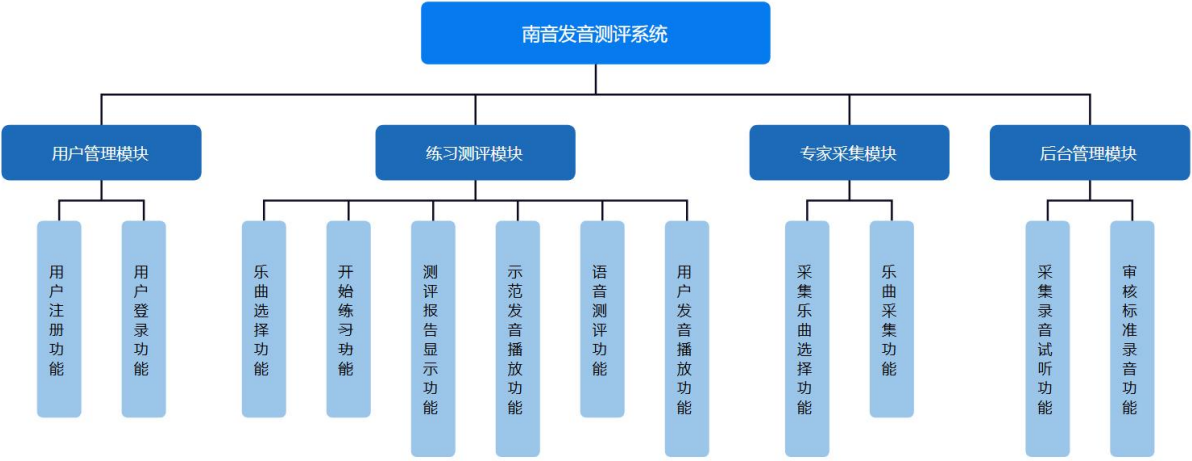


图 4-2 系统功能模块结构示意图

4.3 数据库设计

4.3.1 实体映射

系统根据需求分析阶段设定的数据模型，设计了多个核心组件及其映射关系。用户实体被映射至“User”表，用于存储基础数据和角色信息。对于歌曲和歌词，采用一对多的映射关系，分别映射至“Song”表和“Lyric”表。歌词表同时包含中文字符和对应的国际音标。练习组件则映射至“Practice”表，用于记录每次练习数据。练习结果（PracticeResul）映射为结果表，用存储每一句的评测详情。标准录音实体（Standard Recording）映射为录音表，关联专家用户与特定歌词。

4.3.2 数据库表结构

用户表包含以下字段：主键为 Uuid 类型用户 ID，包含用户名、加密密码、真实姓名及角色 ID，具体如表 4-1 所示。

表 4-1 Users（用户表）详细定义

类型	字段名	用途
Uuid(16)	id	用户 ID
varchar(255)	username	用户名
varchar(255)	role_id	角色
varchar(50)	real_name	真实姓名

歌曲表包含以下字段：主键为歌曲 ID，中文标题为 title_cn，英文标题为 title_en，歌词数量为 lyric_count，时长为 duration，具体如表 4-2 所示。

表 4-2 Songs（歌曲表）详细定义

类型	字段名	用途
----	-----	----

int8(8)	id	歌曲 ID
varchar(255)	duration	预计练习花费时间
int4(4)	lyric_count	歌曲句子数
int4(4)	pronunciator_count	标准示例发音数量
varchar(255)	title_en	英文歌名
varchar(255)	title_cn	中文歌名

歌词表包含以下字段：歌词 ID、外键（歌曲 ID）、句子编号、中文文本及拼音，具体如表 4-3 所示。

表 4-3 Lyrics（歌词表）详细定义

类型	字段名	用途
Bigserial(8)	id	歌词 ID
varchar(512)	hanzi	汉字
varchar(1024)	standard_ipa	标准音标
int4(4)	sentence_number	句号（第几句）
int8(8)	song_id	所属歌曲 ID

练习表包含主键字段（id、user id、song id、start time、end time 和 average accuracy），通过复合键或关联表结构，可高效检索用户使用记录，具体如表 4-4 所示。

表 4-4 Practices（练习表）详细定义

类型	字段名	用途
Uuid(16)	id	练习 ID
Uuid(16)	user_id	用户 ID
int8(8)	song_id	练习的歌曲 ID
float8(8)	average_accuracy	平均准确度
int4(4)	status	状态
timestamp(6)	start_time	开始时间
timestamp(6)	end_time	结束时间

练习结果表（practice_results）包含以下字段：主键为结果 ID，练习编号（practice_id）、句子编号（sentence_number）、字错误率（wer）、准确率（accuracy）、练习结果详情（JSONresult_details）以及语音文件路径（voice_file_path），具体如表 4-5 所示。

表 4-5 Practice_Results（练习结果表）详细定义

类型	字段名	用途
Uuid(16)	id	结果 ID
Uuid(16)	user_id	用户 ID
varchar(512)	voice_file_path	用户录音地址
int4(4)	sentence_number	对应歌词句号
float8(8)	wer	字错率(Word Error Rate)
float8(8)	accuracy	准确率
Uuid(16)	practice_id	对应的练习 ID
Jsonb(变长)	result_details	详细测评信息

标准记录表的标准记录字段包括：录音 ID、曲目 ID、文档 ID、专家用户 ID、音频文件路径、提交批次 ID 以及审核状态，具体如表 4-6 所示。

表 4-6 Standard_Recordings（标准记录表）详细定义

类型	字段名	用途
Bigserial(8)	id	录音 ID
int8(8)	song_id	所属歌曲 ID
Uuid(16)	expert_user_id	专家用户 ID
int4(4)	sentence_number	对应歌词句号
varchar(255)	audio_file_path	标准音频路径

5 核心算法与模型优化

本系统的核心竞争力在于对闽南语方言发音的精准识别与量化评价能力。本章将深入剖析支撑该功能的关键算法逻辑及模型调优策略，力求通过一系列技术支撑，为南音教学提供具有鲁棒性与可解释性的智能反馈机制，真正服务于数字化教学应用。

在模型选型阶段，本项目对比了传统的语音识别模型（如 GMM-HMM、DNN）。虽然传统模型在特定领域表现尚可，但在南音这一低资源方言场景下，却面临着难以逾越的障碍：首先是标注困难，南音独特的发音需要极高专业度的音素级标注，人工成本极高；其次是词典构建困难，闽南语方言缺乏标准化的发音词典，传统模型难以建立有效的声学-语言映射；最后是数据量严重不足，传统模型往往需要海量数据进行从头训练，这对于现存南音语料而言是不现实的。

鉴于此，本研究选择了基于 Transformer 架构的 Whisper 模型，并采用迁移学习的微调策略来解决上述难题。该策略主要包含三个核心步骤：首先，构建专用语料库，建立南音音频到国际音标（IPA）的精准映射，解决数据稀缺问题；其次，进行词表适配，扩展模型词表以包含南音特有的音素符号，使得模型能够正确输出标准的音标序列；最后，实施监督微调，通过冻结模型编码器的大部分参数，仅微调解码器及部分适配层的方式，高效地让模型建立起南音声学特征到国际音标的正确映射，从而在有限数据下实现高性能的识别效果。此外，本章还将详细介绍依靠动态规划实现的音素级比对算法，并对模型在检测集上的性能表现及误差成因进行深入分析。

5.1 数据集的创建与处理

5.1.1 数据采集与分布

南音属于非物质文化遗产，具备很强的地域特征，公开可得的标准发音数据十分缺少，为打破“数据孤岛”难题，本研究采用“专家众包”形式，请经验丰富的南音传承人录下了涵盖 1792 个高质量音频样本的专属数据集。鱼昆等学者曾剖析过，即便端到端 ASR 技术精简了建模步骤，但它在某个特定垂直领域的表现很依赖高质量标注数据的支持^[7]。所以，本研究严格按照“人工标注-交叉验证”的数据清理流程来保障标注准确无误，而且，为使评价结果具备统计学价值，数据集被仔细划分成训练集和测试集，这两者在声学样本上维持着物理上的分离，从而客观检验模型的泛化能力。

该训练数据集共包含 1,643 条音频数据，涵盖南音标准曲集中的多个经典曲目，例如《茶靡架》(cmj)、《共君断约》(gjdj)、《一间草厝》(yjcc)、《一身》(ys)、《元宵十五》(yxsw) 等。以《直入花园》“zrhy”为例，各子集样本量存在差异：其中“ys”子集样本量最多达 405 个，而“zrhy”子集仅 89 个。测试集包含 149 个样本，覆盖上述所有子集，具体数据分布如图 5-1 所示，用于验证模型对未见样本的适应能力。所有音频数据均经过预处理，设置 16,000 赫兹的采样率，并根据 Whisper 模型的要求转换为

单声道格式。

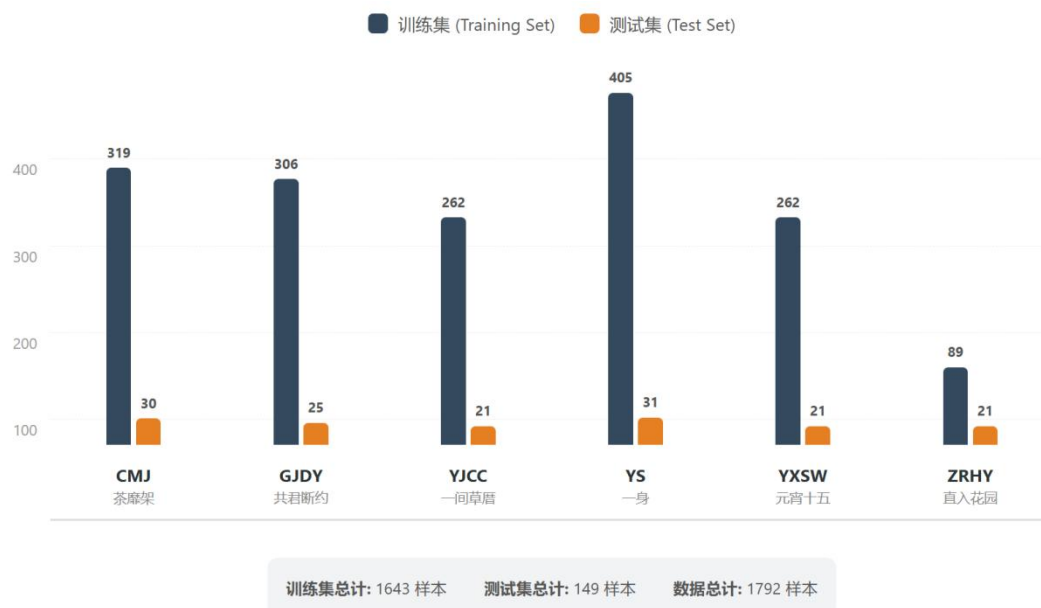


图 5-1 训练集与测试集的歌曲样本数量分布

5.1.2 数据标注与格式化

有监督学习的效果上限由标注数据的质量决定，本研究把国际音标（IPA）系统当作标注基准，以此精确把握闽南语里复杂的元音流转和声调变化。原始数据以结构化的 CSV 格式保存，创建起音频物理路径和 IPA 序列之间的对应关系，在数据加载的时候，用 Pandas 库执行向量化读取，再经由自定义的预处理流程达成路径的绝对化对应，由于模型输入维度存在限制，脚本自带了自动分帧逻辑，对于时长超 30 秒的长音频实施智能截断或者静音分割，从而保证所有输入张量都符合模型训练的维度要求。

5.2 模型选择与训练策略

5.2.1 基座模型架构对比

本研究以 OpenAI 开源的 Whisper 模型为基座模型，该模型采用 Transformer 架构，并通过大规模弱监督数据进行预训练，展现出良好的多语言特征提取能力。为验证不同规模模型在特定方言任务中的差异，对比“large-v3”与“large-v3-turbo”版本。其中 large-v3 参数量更大，理论上具备更强的适应能力，而 large-v3-turbo 则是经过蒸馏优化的版本，旨在保持高性能的同时提升推理速度，具体基座模型对比如图 5-2 所示。

参数指标 (Metrics)	Whisper Large-v3 (基准模型)	Whisper Large-v3-turbo (优化模型)
总参数量 (Parameters)	1550 M	809 M
编码器层数 (Encoder Layers)	32 层	32 层 (无损)
解码器层数 (Decoder Layers)	32 层	4 层 (显著精简)
推理速度 (Inference Speed)	1x (基准)	~6x - 8x
显存占用 (VRAM Usage)	~10 GB	~6 GB
识别精度 (Accuracy/WER)	极高	极高 (仅下降 <1%)



图 5-2 Whisper Large-v3 与 Whisper Large-v3-Turbo 结构参数对比

5. 2. 2 模型微调与训练

模型调优基于 Hugging Face 的 Transformers 框架，采用 Seq2SeqTrainer 控制器训练。核心参数设置为学习率 $1e^{-5}$ ，并通过 200 步预热策略确保初始稳定性。为应对显存限制并提升训练效率，采用梯度累积技术，将累积步数设为 16 步，使批量处理规模符合调优要求。整个训练过程总步数设置为 2000 步，每 50 个步骤会进行一次模型验证并保存。训练期间启用了 FP16 混合训练模式，该模式不仅大幅降低了显存占用，还显著提升了运算速度，模型调优过程损失函数和监控数据如图 5-3 与图 5-4 所示。

训练完成后，系统会自动从验证集加载训练效果最佳的模型权重参数，并最终作为公开版本部署。

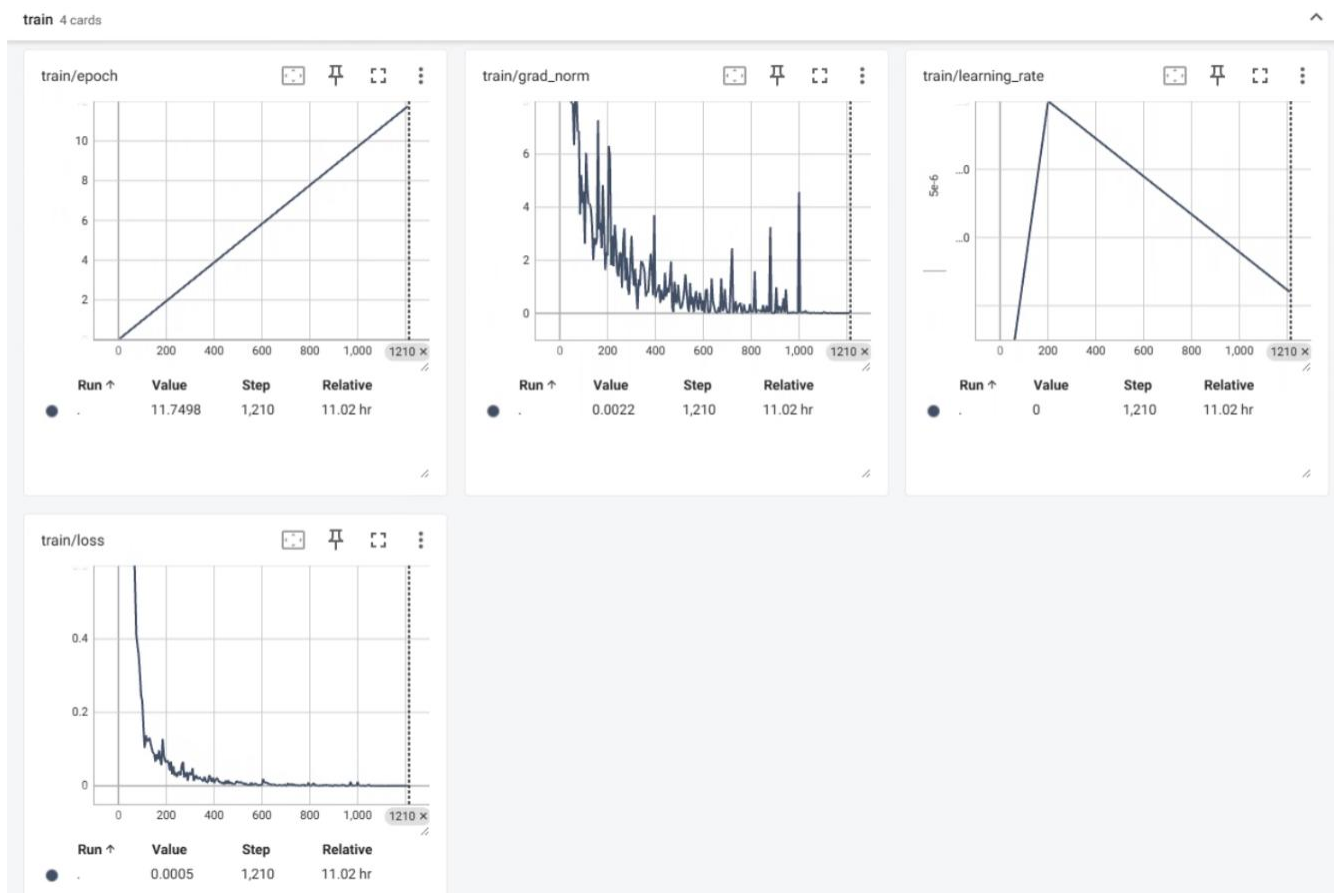


图 5-3 模型训练期间训练集损失函数的下降曲线

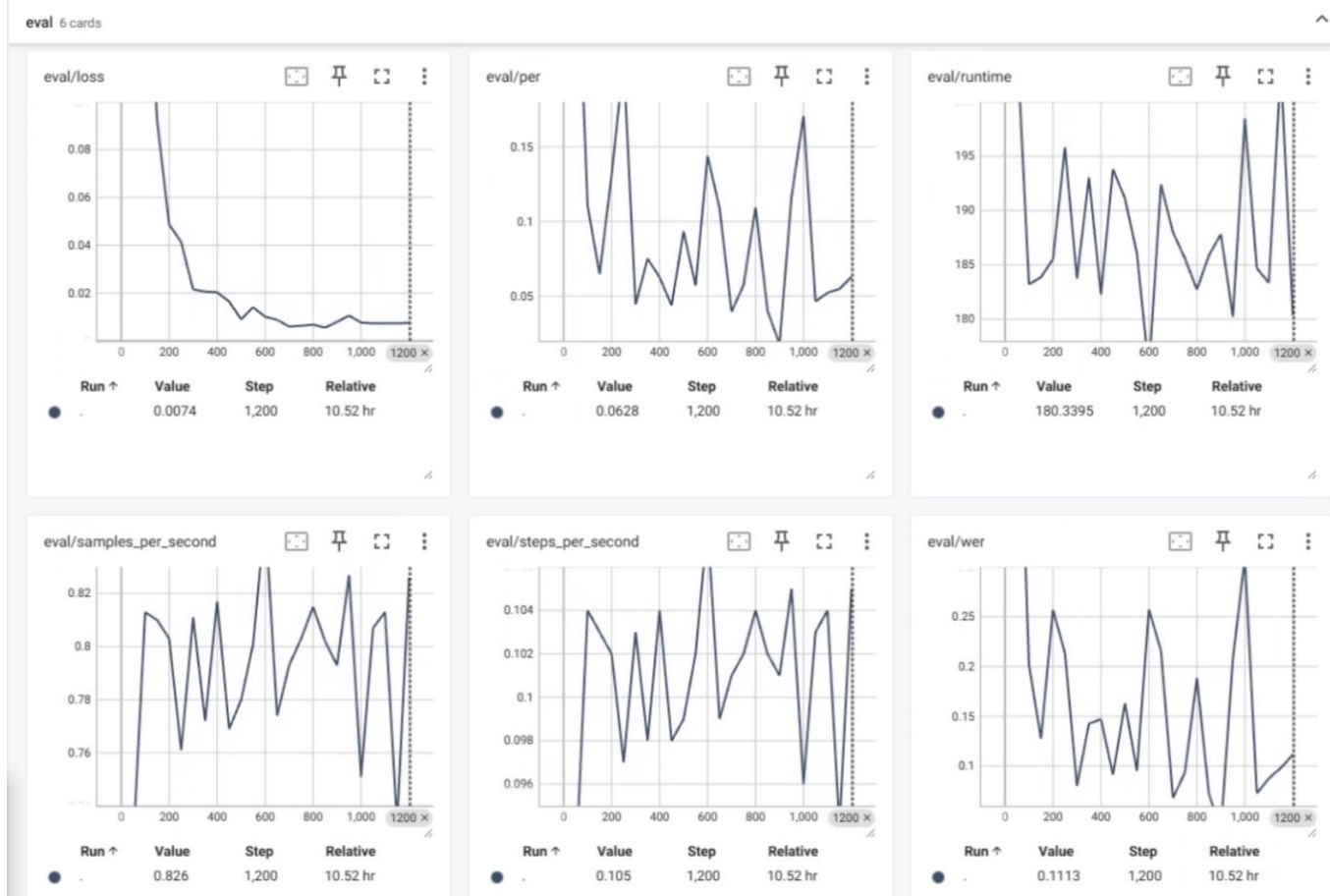


图 5-4 模型训练期间验证集损失函数的下降曲线

5.3 算法实现

5.3.1 评价指标定义

为定量评估发音准确性，本系统采用两项核心指标：字错误率（WER）如公式 5-1 所示，和音素错误率（PER）如公式 5-2 所示。李强等通过实验分析指出：字错率（WER）是自动语音识别测评中最核心的指标之一，能够客观反映识别结果与标准文本之间的差异^[18]。需要特别说明的是，虽然 WER 与 PER 在数学形式上均基于动态规划的编辑距离算法，计算逻辑一致，但二者的基本计算单元（Token）存在本质区别：WER 衡量整体词级准确度，侧重完整语音的识别，以“字”或“词”为单位进行比对；而 PER 则分析字内部细节，深入到音素内部，将语音拆解为更细微的“音素”序列（如声母、韵母、声调等），关注微小成分的偏差。

两者的通用计算公式如下：错误率等于替换错误（S）、删除错误（D）与插入错误（I）之和除以参考序列的总长度（N）。该值越低，表明模型得出的结果越接近标准发音，换句话说，发音质量就越好。

$$WER = \frac{S+D+I}{N} \quad (\text{公式 5-1}) \quad (\text{注：此处 S、D、I、N 计算单元均为字})$$

$$PER = \frac{S+D+I}{N} \quad (\text{公式 5-2}) \quad (\text{注：此处 S、D、I、N 计算单元均为音素})$$

5.3.2 动态规划序列对齐算法

该系统采用基于动态规划的修正距离算法来计算 WER（字错误率）和 PER（音素错误率）。该算法的核心思想是将复杂的最优对齐问题分解为子问题，接收标准语音序列（Reference）和用户发音测评序列（Hypothesis）作为输入，构建一个二维代价矩阵。矩阵中的每个元素（i, j）代表了标准序列前 i 个元素与测评序列前 j 个元素之间的最小编辑距离。

算法从矩阵左上角（初始状态）开始，通过状态转移方程迭代计算插入（Insertion）、删除（Deletion）和替换（Substitution）操作的累积代价，逐步填充整个矩阵。当计算至矩阵右下角（终止状态）时，便得到了两个完整序列的最小总编辑距离。

为了定位具体的错误位置，算法将从右下角向左上角进行回溯（Backtracking），重构出一条全局最优匹配路径。该路径不仅量化了总误差，还能明确标注出错误的具体类型及位置。基于此，系统可生成详细的评估报告，告知用户哪些音节存在误读、遗漏或多读的情况。这种基于动态规划的软校对逻辑，保证了评测结果的数学严谨性与可解释性。

引入该算法的关键意义在于解决了序列错位问题。在缺乏弹性对齐机制的传统比对中，若用户发音序列中出现插入错误（如多读了一个字），会导致后续所有字符与标准序列发生刚性错位（Misalignment）。这种“一步错，步步错”的逻辑会错误地将后续原本正确的发音全部判定为错误，导致最终评分严重失真。而动态规划算法通过动态引入插入或删除代价，能够自动“吸收”这些局部错误，迅速矫正序列偏

移，重新对齐后续字符，从而确保评分的鲁棒性与客观性。具体序列对齐过程如图 5-4 所示。



图 5-4 动态规划距离矩阵计算示意图

5.4 实验结果与分析

5.4.1 模型性能评估

独立测试集评估结果显示，经微调后的 large-v3-turbo 模型表现优异：在全部测试数据中，其字错误率（WER）仅为 0.83%，平均音素错误率（PER）为 0.56%，展现出较高的准确度。相比之下，参数量更大的 large-v3 模型表现欠佳，其 WER 值高达 3.53% 的样本中，turbo 模型仅出现 8 例样本识别偏差，其余 141 例样本均实现 100% 完美匹配。测试结果表明，经过微调后的 turbo 模型在特定小样本方言数据集上展现出良好的鲁棒性和广泛适用性，能有效避免过拟合问题，具体性能对比如图 5-5 所示。

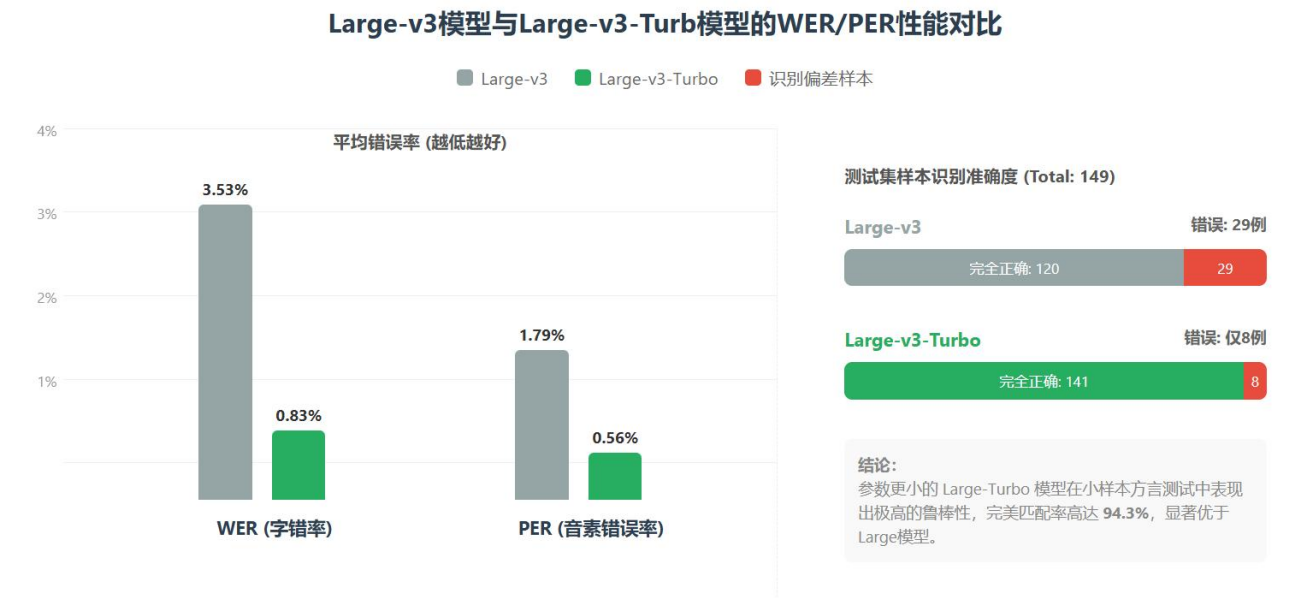


图 5-5 微调后的 WER/PER 性能对比图

5. 4. 2 典型误识别示例分析

通过对 8 个错误样本进行分析, 发现大部分错误集中在“zrhy”子集上。该子集仅包含 89 个训练样本, 数据量远少于其他歌曲, 说明模型未能充分捕捉该歌曲的发音特征, 例如部分汉字在古汉语和现代汉语中的发音差异。以“zrhy/2005”为例, 其中“岭”字的发音被错误地判定为[liaʔ], 而非标准的[nĩã]。在“gjdy”和“yjcc”这些语音数据中, 多数错误源于鼻音与边音的混淆, 例如将[mũã]误判为[mbua], 详细分析如表 5-1 所示。这些错误表明, 即便在资源极为有限的情况下, 模型仍有提升空间来捕捉更精细的发音特征, 同时也指明了通过增加特定类别数据来优化模型性能的方向。

表 5-1 测试数据集中常见错误示例的详细分析表

文件名	预测结果 (IPA)	参考标准 (IPA)	错误类型分析
zrhy/2005.wav	... liaʔ tsĩũ...	... nĩã tsĩũ...	发音特征未捕捉: 将“岭”字的标准音[nĩã]误识别为[liaʔ], 反映出闽南语发音的细微差异未被完全覆盖。
gjdy/2014.wav	tɔ si mbua kə...	tɔ si mũã kə...	鼻音/塞音混淆: 将鼻化元音[mũã]误识别为带前鼻音的塞音[mbua], 属音系近似错误。
yjcc/2005.wav	sui liũ tã...	sui nĩũ tã...	声母混淆(L/N): 典型的边音[l]与鼻音[n]混淆, 这是闽南语学习中常见的难点, 模型在此处表现出类似的犹豫。
zrhy/2017.wav	lian laʔ liu...	lian a³ liu...	插入/多字错误: 预测结果多出了一个音节[laʔ], 可能是对音频中拖音或背景噪音的误读。
zrhy/2010.wav	...kau tsʰĩ tsĩũ...	...kau tsʰi tsĩũ...	鼻化音冗余: 预测结果额外增加了鼻化符号[~], 属于极其细微的音准偏差。

6 系统实现

在系统的前端设计与实现过程中，始终秉持“古朴审美与现代交互交融”的设计原则。界面整体采用现代化的卡片式 UI 布局，力求在视觉上呈现出结构清晰、层级分明的交互体验。正如俞康在探讨南音历史形态时所指出的，南音依靠严谨的“指、谱、曲”体系，塑造了古朴雅正、意蕴深远的传统音乐美学^[19]。这种“结构严谨而气韵生动”的传统古典美学，恰好与现代前端设计中强调的“信息模块化与视觉留白”理念高度契合。

同时，黄月萍等学者在剖析南音出圈现象时强调，非遗的传承不能固守刻板的形式，而应积极探索与现代科技、现代审美的融合路径，从而实现传统文化的“创造性转化”与“当代表达”^[20]。受此启发，本系统在视觉风格上摒弃了繁杂的修饰，将南音古雅的文化特质“转译”为简约的数字设计语言。通过恰当的界面留白、雅致的排版以及结构化的卡片展示，既在数字终端恰如其分地还原了南音内容的韵律之美，又顺应了当代年轻用户的交互习惯，从而为学习者打造出一个既高效实用又赏心悦目的沉浸式非遗学习环境。

6.1 开发环境与部署配置

系统基于跨平台生态构建，目的在于协调服务的适配性及其运行的稳定性。后端运用 Java 17 LTS 版本，并凭借 Spring Boot 框架形成服务集群。而基础设施层，利用 Docker 容器化技术来封装部署 RabbitMQ 消息中间件以及 Redis 缓存服务，经由端口映射达成服务之间的互联互通。AI 算力层以 CUDA 环境为根基，执行 Python 深度学习脚本，最大限度发挥 GPU 在大模型推断时并行计算方面的独特优势。

客户端开发借助 HBuilderX 和 Uni-app 框架，做到了 Android 端和微信小程序的同构发布。数据持久层选择 PostgreSQL 数据库，并整合 Citus 插件，从而为日后的分布式扩展做好准备，Nginx 属于反向代理网关，负责分发静态资源。

6.2 用户管理模块

用户管理模块是系统的门户，承载着身份验证、权限分配和会话管理等核心职责。该模块采用基于 JWT 的无状态认证机制，前端通过 Vue.js 实现响应式表单交互，后端则依托 Spring Security 框架构建坚固的安全防线。

6.2.1 用户登录

登录功能是用户进入系统的第一步。前端设计了简洁直观的登录界面，包含账号、密码输入框及图形验证码组件，登录界面如图所示 6-1 所示。



图 6-1 用户登录页面

登录前端实现逻辑：在用户点击“登录”按钮时，前端会首先进行表单校验，确保用户名和密码非空。随后，调用封装好的 `api.login` 方法，将账号、密码以及验证码标识（UUID）发送至后端 `/api/auth/login` 接口，具体实现代码如代码 6-1 所示。

代码 6-1：用户登录前端表单校验与异步请求实现

```
const handleLogin = async () => {
  if (!form.value.username || !form.value.password || !form.value.captchaCode) {
    uni.showToast({ title: '请填写完整信息', icon: 'none' });
    return;
  }
  uni.showLoading({ title: '登录中...' });
  try {
    const data = await api.login({
      username: form.value.username,
      password: form.value.password,
      captchaCode: form.value.captchaCode,
      captchaKey: form.value.captchaKey
    });
    uni.hideLoading();
    if (data.token) {
      uni.showToast({ title: '登录成功!', icon: 'success' });
      uni.setStorageSync('token', data.token);
      uni.setStorageSync('username', form.value.username);
      uni.reLaunch({ url: '/pages/index/index' });
    }
  } catch (error) {
    uni.hideLoading();
    fetchCaptcha();
    console.error('登录流程出错:', error);
  }
};
```

后端实现逻辑：后端 `AuthController` 接收到请求后，首先校验验证码的有效性。验证通过后，使用 `AuthenticationManager` 进行用户凭证的核对。若认证成功，系统将生成一个包含用户 ID、角色（Role）

及过期时间的 JWT 令牌，并返回给客户端，具体实现代码如代码 6-2 所示。

代码 6-2：后端 AuthController 验证码核对与 JWT 令牌生成逻辑

```
@PostMapping("/login")
public ResponseEntity<?> authenticateUser(@RequestBody LoginRequest loginRequest) {
    // 1. 校验验证码
    if (!validateCaptcha(loginRequest.getCaptchaKey(), loginRequest.getCaptchaCode())) {
        return ResponseEntity.badRequest().body(Map.of("message", "Error: Verification code is incorrect or expired!"));
    }
    Authentication authentication = authenticationManager.authenticate(
        new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword()));
    SecurityContextHolder.getContext().setAuthentication(authentication);
    String jwt = jwtUtil.generateToken(authentication);
    log.info("User {} logged in successfully.", loginRequest.getUsername());
    return ResponseEntity.ok(new JwtResponse(jwt));
}
```

6.2.2 用户注册

注册功能允许新用户创建账户。为了防止恶意批量注册，系统在注册环节同样引入了图形验证码机制，并对密码进行了强度校验，用户注册页面如图 6-2 所示。

A mobile app registration page titled '注册账号' (Register Account) and '创建账号' (Create Account) with the subtitle 'Join Us Today'. The page contains four input fields: '账号' (Account) with placeholder '请输入用户名' (Please enter username), '密码' (Password) with placeholder '请输入密码' (Please enter password), '确认密码' (Confirm Password) with placeholder '请再次输入密码' (Please re-enter password), and '验证码' (Verification Code) with placeholder '输入验证码' (Enter verification code). To the right of the verification code field is a CAPTCHA image showing the characters 'zmR'. At the bottom is a large green button labeled '立即注册' (Register Now) and a link '已有账号? 去登录' (Already have an account? Go to login).

图 6-2 用户注册页面

前端实现逻辑：注册页面包含账号、密码、确认密码及验证码输入框。前端代码重点处理了“两次密码输入一致性”的校验逻辑。注册成功后，系统会自动跳转回登录页，引导用户进行首次登录，具体实现代码如代码 6-3 所示。

代码 6-3：用户注册前端密码一致性校验与提交逻辑

```
const handleRegister = async () => {
    const { username, password, confirmPassword, captchaCode } = form.value;

    if (!username || !password || !confirmPassword || !captchaCode) {
        uni.showToast({ title: '请填写所有字段', icon: 'none' });
    }
}
```

```

    return;
  }
  if (password !== confirmPassword) {
    uni.showToast({ title: '两次输入的密码不一致', icon: 'none' });
    return;
  }
  uni.showLoading({ title: '注册中...' });
  try {
    await api.register({
      username: username,
      password: password,
      captchaCode: captchaCode,
      captchaKey: form.value.captchaKey
    });
    uni.hideLoading();
    uni.showToast({ title: '注册成功', icon: 'success' });
    setTimeout(() => {
      uni.navigateBack();
    }, 1500);
  } catch (error) {
    uni.hideLoading();
    fetchCaptcha();
  }
}
};

```

后端实现逻辑：后端接收到注册请求后，首先检查用户名是否已存在。若用户名可用，系统将使用 BCryptPasswordEncoder 对用户密码进行哈希加密，随后将用户信息（包括默认角色 Role_Student）写入数据库，具体实现代码如代码 6-4 所示。

代码 6-4：后端用户注册密码加密与持久化存储实现

```

@PostMapping("/register")
public ResponseEntity<?> registerUser(@RequestBody LoginRequest registerRequest) {
    // 1. 校验验证码
    if (!validateCaptcha(registerRequest.getCaptchaKey(), registerRequest.getCaptchaCode())) {
        return ResponseEntity.badRequest().body(Map.of("message", "Error: Verification code is incorrect or expired!"));
    }
    if (userRepository.findByUsername(registerRequest.getUsername()).isPresent()) {
        return ResponseEntity.badRequest().body(Map.of("message", "Error: Username is already taken!"));
    }
    User user = new User(registerRequest.getUsername(),
        passwordEncoder.encode(registerRequest.getPassword()), "默认用户", 1L);
    userRepository.save(user);
    log.info("User registered successfully: {}", registerRequest.getUsername());
    return ResponseEntity.ok(Map.of("message", "User registered successfully!"));
}

```

6.3 练习测评模块

6.3.1 歌曲选择页面

歌曲选择页面是用户进入练习流程的入口，展示可供练习的南音曲目资源。歌曲选择页面如图 6-3 所示

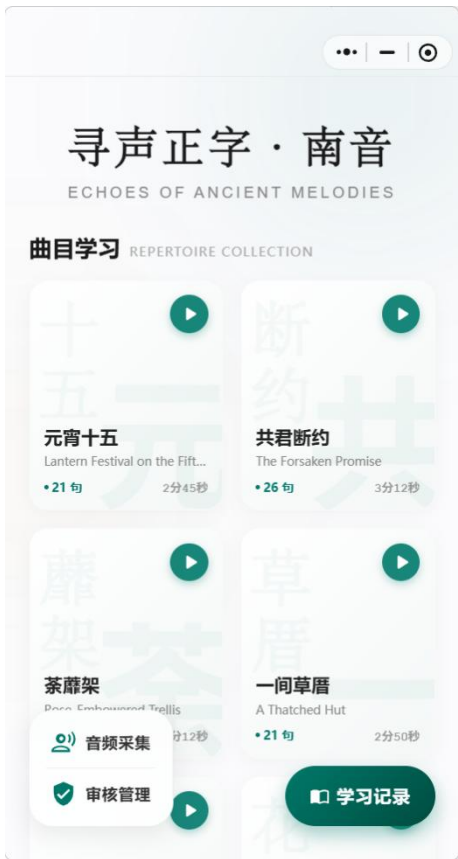


图 6-3 歌曲选择页面

前端采用卡片式布局（Card Layout）展示曲目列表，每张卡片包含曲目标题、时长、歌词句数等关键信息。点击卡片后，系统会调用后端接口创建一个新的“练习会话”（Practice Session），并跳转至具体的练习页面。启动练习会话具体实现代码如代码 6-5 所示。

代码 6-5：歌曲选择界面创建练习会话的前端触发逻辑

```
const goToLearningPage = async (song) => {
  uni.showLoading({ title: '正在创建练习...' });
  try {
    const response = await api.startPractice({ songId: song.id });
    if (response.success && response.practiceId) {
      const practiceId = response.practiceId;
      uni.hideLoading();
      uni.navigateTo({
        url:
          '/pages/practices/practices?songId=${song.id}&practiceId=${practiceId}&lyricCount=${song.lyricCount}&pronunciatorCount=${song.pronunciatorCount}`
      });
    } else {
      throw new Error(response.message || '创建练习失败');
    }
  } catch (error) {
    uni.hideLoading();
    console.error("创建练习失败:", error);
    uni.showToast({
      title: error.message || '创建练习失败，请重试',
      icon: 'none'
    });
  }
}
```

6.3.2 练习界面

练习界面是系统交互最复杂的页面，集成了歌词展示、标准音播放、录音采集和实时评测四大功能。

歌词与音标同步展示：界面核心区域展示当前句子的汉字与闽南语音标。前端通过 Vue 的响应式数据绑定，实时更新 `currentSentenceId` 对应的内容。标准音频播放器支持“发音人切换”功能。

录音与上传逻辑：录音功能基于 `uni.getRecorderManager` 实现。用户长按按钮开始录音，松开结束。录音完成后，音频文件会被直接上传至 Nginx 服务器，随后由后端触发异步评测任务，录音上传与轮询结果开始逻辑具体实现代码如代码 6-6 所示。

代码 6-6：练习界面录音文件上传与 ASR 评测任务轮询前端实现

```
const processRecording = async (filePath) => {
  if (!filePath) return;
  isLoading.value = true;
  try {
    evaluationStatusText.value = '创建任务...';
    const taskData = await api.createTask();
    if (!taskData.success) throw new Error(taskData.message || '创建任务失败');
    taskId.value = taskData.taskId;
    evaluationStatusText.value = '上传并校验...';
    const uploadResponse = await api.uploadVoice({
      filePath: filePath,
      formData: {
        taskId: taskId.value,
        practiceId: practiceId.value,
        Song_id: String(currentSongId.value),
        Sentence_id: String(currentSentenceId.value)
      }
    });

    if (!uploadResponse.success) {
      uni.showModal({
        title: '录音提示',
        content: uploadResponse.message || '录音校验未通过，请重试',
        showCancel: false,
        confirmText: '我知道了',
        success: () => {
          isLoading.value = false;
          evaluationStatusText.value = '';
        }
      });
      return;
    }

    // 如果上传成功且校验通过，开始轮询
    startPolling(taskId.value);

  } catch (error) {
    console.error('评测全流程出错:', error);
    isLoading.value = false;
    evaluationStatusText.value = '';
    uni.showToast({
      title: '网络请求失败，请重试',
      icon: 'none'
    });
  }
};
```

评测结果可视化：当轮询获取到成功状态后，前端会解析 JSON 格式的评测报告。系统使用不同颜色高亮显示发音状态：绿色代表正确 (Correct)，红色代表错误 (Substitution)，删除线代表漏读 (Deletion)。同时，界面上方会显示本次发音的字错率 (WER) 和准确度评分，如图 6-4 所示。



图 6-4 评估反馈界面截图

6.3.3 完成练习界面

当用户完成一首曲目所有句子的练习后,系统会引导用户进入结算流程。后端接口/api/practices/finish 负责计算本次练习的平均准确率,并将练习状态标记为“已完成”,练习完成界面如图 6-5 所示。



图 6-5 歌曲练习完成数据统计

结算逻辑:前端在最后一页点击“完成练习”时触发提交。后端接收到请求后,会校验所有句子的完成情况,防止用户跳过练习直接提交,具体实现代码如代码 6-7 所示。

代码 6-7: 后端结束练习会话及平均准确率聚合结算逻辑

```
@PostMapping("/practices/finish")
@Transactional
public ResponseEntity<?> finishPractice(@RequestBody Map<String, String> payload) {
    String username = SecurityContextHolder.getContext().getAuthentication().getName();
    User user = userRepository.findByUsername(username)
        .orElseThrow(() -> new RuntimeException("User not found"));
    UUID practiceId = UUID.fromString(payload.get("practiceId"));
    PracticeId compId = new PracticeId(user.getId(), practiceId);
    Practice practice = practiceRepository.findById(compId)
        .orElseThrow(() -> new RuntimeException("Practice not found with composite id: " + compId));
    if (!practice.getUserObject().getId().equals(user.getId())) {
        return ResponseEntity.status(403).body(Map.of("success", false, "message", "无权操作此练习"));
    }
    int lyricCount = practice.getSong().getLyricCount();
    long evaluatedCount = practice.getResults().stream().count();
    if (evaluatedCount < lyricCount) {
        log.warn("用户 '{0}' 尝试完成练习 '{0}', 但句子数量不足 ({0} < {0})", username, practiceId, evaluatedCount, lyricCount);
        return ResponseEntity.ok().body(Map.of("success", false, "message", "还有" + (lyricCount - evaluatedCount) + "句没完成,请完成所有测评再提交"));
    }
    double avgAccuracy = practice.getResults().stream()
        .mapToDouble(PracticeResult::getAccuracy)
        .average()
    }
```

```

        .orElse(0.0);
practice.setStatus(1); // 状态置为已完成
practice.setEndTime(LocalDateTime.now());
practice.setAverageAccuracy(avgAccuracy);
practiceRepository.save(practice);
log.info("用户 '{}' 成功完成练习 '{}', 平均准确率: {:.2f}%", username, practiceId, avgAccuracy);
return ResponseEntity.ok(Map.of("success", true, "message", "练习已完成"));
}

```

6.4 学习记录

学习记录模块为用户提供了个人学习数据的可视化展示，帮助用户回顾历史表现并制定学习计划。数据可视化该页面包含三个核心部分：概览卡片用于展示累计学习时长、练习曲目数及平均准确率。热力图日历用于通过颜色深浅展示用户每日的活跃程度，直观反映学习频率。最近练习列表用于按时间倒序列出最近完成的练习记录，点击可查看详细报告，学习记录页面如图 6-6 所示。



图 6-6 歌曲练习完成数据统计

后端通过 JPA 和 Stream API 对用户的练习数据进行聚合统计。例如，热力图数据是通过按日期分组

统计完成次数得出的，具体实现代码如代码 6-8 所示。

代码 6-8：后端基于 Stream API 的用户每日活跃度分组统计实现

```
public Map<LocalDate, Long> getCalendarActivity(User user) {  
    List<Practice> completedPractices = practiceRepository.findByUserAndStatus(user.getId(), 1);  
    return completedPractices.stream()  
        .filter(p -> p.getEndTime() != null)  
        .collect(Collectors.groupingBy(  
            p -> p.getEndTime().toLocalDate(),  
            Collectors.counting()  
        ));  
}
```

6.5 专家采集模块

6.5.1 采集歌曲选择页面

该页面是专家开始工作的起点。为了提高采集效率，界面设计上突出了“任务导向”。任务列表展示：页面展示了所有待采集的曲目。权限校验：由于采集功能仅对特定用户开放，页面加载时会预先进行权限检查。如果普通用户尝试访问，会被后端拦截或前端路由守卫重定向，采集歌曲页面图 6-7 所示。



图 6-7 音频采集首页界面截图

6.5.2 歌曲录音采集页面

这是专家进行高强度录音工作的核心界面。为了确保数据质量，系统在该页面引入了“强制试听”和“批次管理”机制，录音与质量预检专家按下录音键时，前端调用 `recorderManager` 进行录制。录制结束后，系统强制要求专家点击“试听”按钮，确认无误后“提交”按钮才会亮起，防止误操作提交劣质音频，采集页面如图 6-8 所示。



图 6-8 音频采集界面截图

为了便于后期管理，每次进入录音页面，系统会在前端生成一个唯一的 UUID 作为 `batchId`。该批次号将贯穿本次录音会话的所有文件，确保同一专家的同一次录制行为可被整体追溯，具体实现代码如代码 6-9 所示。

代码 6-9：专家录音采集会话唯一批次号（UUID）生成与加载前端实现

```
onLoad((options) => {
  if (options.songId) {
    songId.value = options.songId;
    totalSentences.value = Number(options.lyricCount);
    songTitle.value = options.title || '录音';
    batchId.value = generateUUID();
    loadLyric(1);
  } else {
    console.error("严重错误：未接收到 songId");
    uni.showToast({ title: "参数缺失", icon: "none" });
  }
  console.log("页面接收到的参数:", songId.value);
});
```

专家模式上传点击提交后，前端会调用后端专属的 `/api/expert/prepare-upload` 接口获取上传凭证。后端会在 Redis 中标记该任务为“专家任务”，从而绕过常规的 ASR 评测流程，转而进入质量检测流程

（如检测信噪比、静音片段等）具体实现代码如代码 6-10 所示。

代码 6-10：后端处理专家音频上传与 Python 算力层质量检测调度逻辑

```
private ResponseEntity<?> handleExpertRecording(String taskId, String filePath, String songIdStr, String sentenceIdStr,
String pid) {
    try {
        String username = SecurityContextHolder.getContext().getAuthentication().getName();
        Long songId = Long.parseLong(songIdStr);
        int sentenceNumber = Integer.parseInt(sentenceIdStr);
        Optional<Map<String, String>> lyricOpt = lyricService.getLyric(songId, sentenceNumber);
        if (lyricOpt.isEmpty()) {
            throw new RuntimeException("未找到对应歌词");
        }
        RestTemplate restTemplate = new RestTemplate();
        // 构建请求体
        Map<String, Object> requestBody = new HashMap<>();
        requestBody.put("filePath", filePath);
        requestBody.put("expectedCount", expectedCount);

        try {
            ResponseEntity<Map> response = restTemplate.postForEntity(PYTHON_SERVICE_URL,
requestBody, Map.class);
            Map<String, Object> result = response.getBody();
            boolean isSuccess = (Boolean) result.get("success");
            int actualCount = Integer.parseInt(result.get("actualCount").toString());
            String message = (String) result.get("message");
            if (!isSuccess) {
                log.warn("校验失败: {}", message);
                return ResponseEntity.ok(Map.of(
                    "success", false,
                    "status", "RETRY",
                    "message", message
                ));
            }
            log.info("Python 服务校验通过: {}", message);
        } catch (Exception e) {
            log.error("Python 服务调用异常", e);
            return ResponseEntity.status(500).body(Map.of(
                "success", false,
                "message", "AI 音频服务不可用, 请联系管理员启动 Python 服务"
            ));
        }
        String batchId = pid == null ? "UNKNOWN" : pid;
        StandardRecording recording = new StandardRecording(song, sentenceNumber, expert, filePath, batchId);
        standardRecordingRepository.save(recording);
        stringRedisTemplate.delete(ExpertController.EXPERT_TASK_PREFIX + taskId);
        return ResponseEntity.ok(Map.of(
            "success", true,
            "message", "采集成功, 已自动切分为 " + expectedCount + " 个片段"
        ));
    } catch (Exception e) {
        log.error("保存专家音频失败", e);
        return ResponseEntity.status(500).body(Map.of("success", false, "message", "保存失败: " +
e.getMessage()));
    }
}
```

6.6 后台管理模块

后台管理模块是保障系统数据质量的关键防线。管理员通过该模块不仅能审核专家提交的标准语料，还能对模型的自动评分进行人工复核，从而为后续模型微调积累高质量的反馈数据。

6.6.1 管理员用户记录模型评分功能

为了持续优化 Whisper 模型的评测准确性，系统设计了“评分校验”功能。管理员可以查看所有学习者的练习记录，并对模型的打分进行人工干预，页面如图 6-9 所示。



图 6-9 管理员查看练习记录页面

全局练习记录查看与学习者端仅能查看个人记录不同，管理员账号登录后，调用 `/api/admin/batches/pending` 接口可获取全平台最近的练习列表，练习列表如图所示。后端通过 Spring Security 获取当前用户角色，若是管理员，则返回包含用户名的全局数据流，具体实现代码如代码 6-11 所示。

代码 6-11：管理员端全平台历史已完成练习记录条件排序与聚合检索实现

```
public List<RecentPracticeDTO> getAllRecentPractices() {
    PageRequest pageRequest = PageRequest.of(0, 9999, Sort.by("endTime").descending());
    List<Practice> allPractices = practiceRepository.findAll(pageRequest).getContent();

    return allPractices.stream()
        .filter(p -> p.getStatus() == 1) // 必须是已完成的练习
        .map(p -> {
            String userName = p.getUserObject() != null ? p.getUserObject().getRealName() : "未知用户";
            int totalCount = p.getSong().getLyricCount();
            // 统计已审数量
            long reviewedCount = p.getResults().stream()
```

```

        .filter(PracticeResult::isActive)
        .filter(r -> Boolean.TRUE.equals(r.getAdminReviewed()))
        .count();
    return new RecentPracticeDTO(
        p.getId().toString(),
        "[" + userName + "]" + p.getSong().getTitleCn(),
        p.getEndTime(),
        p.getAverageAccuracy(),
        p.getSong().getId(),
        totalCount,
        reviewedCount, // 已审
        totalCount      // 总数
    );
})
.sorted((dto1, dto2) -> {
    boolean isFinished1 = dto1.getReviewedCount() >= dto1.getTotalCount();
    boolean isFinished2 = dto2.getReviewedCount() >= dto2.getTotalCount();
    // 规则 1: 未完成的排前面 (isFinished=false < isFinished=true)
    if (isFinished1 != isFinished2) {
        return isFinished1 ? 1 : -1;
    }
    // 规则 2: 同状态下, 时间倒序 (新的在前)
    return dto2.getPracticeDate().compareTo(dto1.getPracticeDate());
})
.limit(9999)
.collect(Collectors.toList());
}

```

评分人工干预：管理员进入具体的练习详情页后，界面会在自动评分下方显示“专家评分校验”面板。合理性判断：管理员首先听音，判断模型的评分是否合理。分数修正：若认为模型评分偏差过大（如模型判错但实际发音准确），管理员可选择“不准”，并通过滑块给出新的人工评分（1-100 分）。数据闭环：提交后，数据被标记为 `admin_reviewed=true`，这些带有偏差标记的数据将作为负样本用于后续模型的训练，具体页面如图 6-10 所示。

识别结果

准确率: 87.5% 正确: 7 错误: 1 删除: 0 插入: 0

标准:	tit	lip	hue	hŋ	si	hue	mbi	pʰaŋ
识别:	tit	lip	hue	hŋ	si	hue	mĩ	pʰaŋ

专家评分校验
(已评价)
模型打分是否合理? ☐ 合理 ☒ 不准

请点击下方识别错误的字:

tit	lip	hue	hŋ	si
hue	mĩ	pʰaŋ		

提交并下一句

重新录制 下一个例句

图 6-10 管理员评分页面

6.6.2 标准音频审核

该功能用于处理专家采集模块提交的音频数据。只有通过审核的音频，才能正式进入系统的标准音库（Standard Corpus）。

批次化审核列表:系统以“提交批次”为单位展示待审核任务。列表项清晰展示了提交人（专家姓名）、曲目名称、提交时间以及该批次包含的句子数量，审批列表如图 6-11 所示。



图 6-11 管理员审核首页界面截图

逐句听审与操作：进入批次详情后，管理员可以像播放列表一样逐句试听。界面设计了高效的“通过/驳回”操作区，审批页面如图 6-12 所示。



图 6-12 管理员审核界面截图

管理员可以选择或者驳回，通过：音频文件路径被正式写入 StandardRecording 表，状态更为 APPROVED。驳回：该条记录被标记为 REJECTED 音频文件将被舍弃，具体实现代码如代码 6-12 所示。

代码 6-12：管理员端专家采集单条音频审核状态流转后端实现

```
@PostMapping("/audit/single")
@Transactional
public ResponseEntity<?> auditSingleRecording(@RequestBody Map<String, Object> payload) {
    requireAdmin();
    Long recordingId = ((Number) payload.get("recordingId")).longValue();
    Integer status = (Integer) payload.get("status");

    StandardRecording recording = standardRecordingRepository.findById(recordingId)
        .orElseThrow(() -> new RuntimeException("Recording not found"));

    recording.setStatus(status);
    standardRecordingRepository.save(recording);

    return ResponseEntity.ok(Map.of("success", true));
}
```

7 总结与展望

7.1 总结

本研究着眼于非物质文化遗产南音的数字化抢救及活态传承需求，创建起一个结合发音习得，智能评定与语料抢救性采集的辅助教学系统。在工程架构方面，系统采用了“前后端分离+微服务”这种技术形式：后端凭借 Spring Boot 形成了高内聚的业务逻辑核心，前端利用 Uni-app 达到了全终端覆盖的效果，冲破了设备之间的隔阂。从核心算法来讲，本研究对方言低资源这个难点，塑造了南音专属的语音数据集，而且更新性地用迁移学习技术对 OpenAI 的 Whisper 模型执行了领域适应性的微调。实验证据表明，经过改良后的 Whisper-large-v3-turbo 模型在独立的测试集里把字错率（WER）降低到 0.83%，这很好地证实了这条技术路径在方言识别任务当中具备稳定性和高效性。

系统还创建起规范化的语料生产流水线，经由专家端的数据收集以及管理端的质量监督，达成了南音标准音数据的不断汇集。在最后的教学反馈部分，依靠动态规划的音素级比较算法，给学习者赋予了精准细致的订正引导，进而切实创建起从“听辨”一直到“训练”再到“评价”的完整流程。

7.2 展望

尽管本系统已达成预期的核心功能，但在实际部署及深度应用时仍存在一定的客观局限。基于系统现状，后续研究将首先拓展数据生态以克服方言的长尾效应。当前专用语料库虽保证了数据质量，但相比于海量的南音古谱而言其覆盖面仍较为单薄，导致模型应对未收录的冷门曲牌或复杂唱腔时泛化能力受限。未来计划引入“众包采集”机制，深度链接南音传承人与高校社团，从而快速扩大数据集规模并增添其丰富性，着重抢救和保护那些珍贵的孤本曲目数据。

其次，算力成本与响应延迟的矛盾需要解决。依靠 Transformer 架构的大模型推理在物理层面仍是计算密集型任务，面对高并发流量时易造成消息队列积压，导致物理时延上升并影响交互连贯性。针对这一难点，后续的技术攻关将聚焦于模型轻量化与底层架构演进。在算法端，计划探究 INT8 量化与剪枝技术，结合 faster-whisper 等推理加速库，使模型摆脱对高端 GPU 显存的重度依赖，长远目标是实现模型的端侧部署。在工程服务端，未来将考虑逐步把核心异步处理模块向性能更优的 Go 语言体系迁移，并探索使用 Redis Streams 替代现有的 RabbitMQ 进行任务管理，进一步提升高并发场景下的系统吞吐量。

最后，在教学反馈维度的拓展上。南音演唱极重“口法”与“归韵”，当前的纯声学特征反馈难以直接引导使用者的口型开闭程度以及气息运用。未来的研究将尝试引入计算机视觉技术，构建“音-画”双模态智能评定体系。通过摄像头动态获取并剖析用户发音时的面部肌肉与口型变动情况，再融合声学特性执行综合评分。系统将能提供更具直观性的可视化纠错指引，进而打造出包含“听觉识别、视觉校正、数据回应”的南音数字教育生态。

参考文献

- [1] 习近平. 加强文化遗产保护传承弘扬中华优秀传统文化[J]. 中国新闻发布(实务版), 2024, (05):3-7.
- [2] 王莉宁, 康健侨. 中国方言文化保护的现状与思考[J]. 语言战略研究, 2022, 7(04):76-85. DOI:10.19689/j.cnki.cn10-1361/h.20220406.
- [3] 林琰, 李惠芬. 非物质文化遗产的保护机制与活化路径[J]. 南京社会科学, 2023, (03):151-160. DOI:10.15937/j.cnki.issn1001-8263.2023.03.016.
- [4] 黄永林, 李媛媛. 新世纪以来中国非遗保护政策发展逻辑及未来取向[J]. 民俗研究, 2023, (01):5-17+156. DOI:10.13370/j.cnki.fs.2023.01.005.
- [5] 李娇, 谢珍珍. 新媒体时代非物质文化遗产的活化传承措施研究——以泉州南音为例[J]. 文化学刊, 2025, (06):143-146.
- [6] 杨红. 非物质文化遗产数字化传播的意义更新与趋势分析[J]. 中国非物质文化遗产, 2023, (05):102-107.
- [7] 鱼昆, 张绍阳, 侯佳正, 等. 语音识别及端到端技术现状及展望[J]. 计算机系统应用, 2021, 30(03):14-23. DOI:10.15888/j.cnki.csa.007852.
- [8] 王澳回, 张琰, 宋文字, 等. 端到端流式语音识别研究综述[J]. 计算机工程与应用, 2023, 59(02):22-33.
- [9] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.
- [10] Radford A, Kim J W, Xu T, et al. Robust speech recognition via large-scale weak supervision[C]//International conference on machine learning. PMLR, 2023:28492-28518.
- [11] Rogerson-Revell P M. Computer-assisted pronunciation training (CAPT):Current issues and future directions[J]. Relc Journal, 2021, 52(1):189-205.
- [12] 李玮, 朱岩. AI 语音交互技术及测评研究[J]. 信息通信技术与政策, 2019, (12):83-87.
- [13] 冯志勇, 徐砚伟, 薛霄, 等. 微服务技术发展的现状与展望[J]. 计算机研究与发展, 2020, 57(05):1103-1122.
- [14] 霍福华, 韩慧. 基于 SpringBoot 微服务架构下前后端分离的 MVVM 模型[J]. 电子技术与软件工程, 2022, (01):73-76. DOI:10.20109/j.cnki.ets.2022.01.018.
- [15] 陈恩慧. 承南音之精髓育时代之新人——泉州师范学院“南音演唱”课程教学创新方案[J]. 中国音乐, 2022, (02):200-207. DOI:10.13812/j.cnki.cn11-1379/j.2022.02.025.
- [16] 张墨涵, 王雪英, 沈学东, 等. 微服务架构技术与挑战[J]. 网络安全技术与应用, 2023, (02):3-4.

- [17] 倪小璐,王旭英,边俊凯,等.微服务软件架构设计模式及其应用[J].杭州师范大学学报(自然科学版),2021,20(04):442-448.DOI:10.19926/j.cnki.issn.1674-232X.2021.04.016.
- [18] 李强,金陈潇帅,许一骅.浅谈自动语音识别测评指标字错率和句错率的应用[J].现代传输,2020,(01):61-64.
- [19] 俞康.南音的音乐史形态及传承创新策略[J].四川戏剧,2024,(12):114-118.
- [20] 黄月萍,陈恩慧,陈静愉.南音的创造性转化及其当代表达——以2023年央视春晚《百鸟归巢》为例[J].泉州师范学院学报,2025,43(03):57-62.DOI:10.16125/j.cnki.1009-8224.2025.03.008.