南京大學

研 究 生 毕 业 论 文
（申请工程硕士学位）

论 文 题 目 基于旋律相似性的音乐比对分析系统的设计与实现

作 者 姓 名 王若竹

学科、专业名称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 陈振宇 教授

2021 年 **5** 月 **22** 日

学　　号：**MF1932177**

论文答辩日期：**2021** 年 **5** 月 **20** 日

指 导 教 师：　　（签字）

The Design and Implementation of Music Comparison Analysis System Based on Melody Similarity

by

Ruozhu Wang

Supervised by

Professor Zhenyu Chen

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

Software Institute
Nanjing University

May 22, 2021

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于旋律相似性的音乐比对分析系统的设计与实现

工程硕士（软件工程领域）专业 2019 级硕士生姓名：王若竹

指导教师（姓名、职称）：陈振宇 教授

摘　要

随着音乐产业与互联网信息技术的日益发展与结合，音乐市场在容量迅速扩增的同时，也伴随着愈来愈多的音乐侵权纠纷事件，音乐知识产权的保护也愈来愈引起政府、法律界以及人民群众的重视。但是目前法院在判定音乐是否侵权这类问题中并没有具体的量化标准，法官更多的是依靠自身对音乐侵权的认知以及对音乐作品的主观感受进行判定。因此，借助全面客观的音乐比对分析数据辅助音乐侵权相关案件的判定是很有必要的。此外，音乐专业学生作品查重、AI 作曲结果质量鉴定以及音乐爱好者的日常识别比对等场景也需要音乐比对分析系统的客观数据提供支持。

本文设计并实现了基于旋律相似性的音乐比对分析系统，该系统具有音乐上传、音乐比对、结果下载、案例查看等功能。其中，音乐比对主要针对音乐中旋律音程特征和节奏特征的详细相似程度进行分析比较，并提供客观、易懂、全面的音乐比对分析数据。用户上传原作品和疑似相似作品后，该音乐比对分析系统通过构建好的音乐比对分析模型，将音乐比对的结果进行多维度展示，包括整体相似百分比、调式调性比对、相似片段 top3、具体相似片段位置与时长等。在构建音乐比对分析模型时，本文通过比较多种字符串精准匹配算法的效率性能，最终选择效率更高的 Sunday 改进算法作为最终模型中的匹配算法，进一步提升了音乐比对效率。

本文的音乐比对分析系统采用 Vue 框架和 Spring Boot 技术完成了前端网页和后端框架的实现，并通过 MySQL 数据库存储数据信息。通过实验分析，该比对算法效率有明显提升，与其他匹配算法相比，时间可缩短 40% 至 95%。此外，调查问卷结果表明，本文中的比对结果具有更高的可读性。经过一系列功能和性能测试，该系统已满足功能性与非功能性需求，符合预期设想。

关键词：音乐比对，旋律相似性，字符串匹配，音乐分析，音乐侵权

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 Music Comparison Analysis System Based on Melody Similarity

SPECIALIZATION: Software Engineering

POSTGRADUATE: Ruozhu Wang

MENTOR: Professor Zhenyu Chen

Abstract

With the increasing development and integration of the music industry and Internet information technology, the music market is rapidly expanding its capacity, which also accompanied by an increasing number of music infringement incidents. The protection of music intellectual property rights has also attracted more and more attention from the government and the legal profession. But now, there is no specific quantitative criteria for the court to determine whether music infringes or not. The judges rely more on their own cognition of music infringement and subjective feelings of music works to judge. Therefore, it is necessary to provide comprehensive and objective music comparison analysis data to assist in the judgment of music infringement cases. In addition, scenes such as the music works rechecking of music majors, the music quality identification of AI composition results, and the daily recognition and comparison of music lovers also need the support of objective data of music comparison analysis system.

This thesis designs and implements a music comparison analysis system, which has the functions of music uploading, music comparing, result downloading, case viewing and so on. Among them, music comparing mainly analyzes and compares the melody interval and rhythm of two music parts, gives the detailed similarity and provides objective, understandable and comprehensive music comparison analysis data. People upload original works and suspected similar works, the music comparison analysis system with a constructed music comparison model will display the results of music comparison in multiple dimensions, including overall similarity percentage, the comparison of mode and tonality, top3 of similar segments, specific similar segment position and duration, etc. In the construction of the music comparison model, we finally

choose the improved Sunday algorithm as the matching algorithm in the final model by comparing a variety of string matching algorithms, which can further improve the efficiency of music comparison.

The music comparison analysis system in this thesis adopts Vue framework and Spring Boot technology to complete the implementation of front-end web page and back-end framework, and stores data information through MySQL database. Through experimental analysis, the efficiency of this comparison algorithm is significantly improved, compared with other matching algorithms ， time can be shortened by 40% to 95%. In addition, the results of questionnaire show that the comparison result of music comparison analysis in this thesis is more readable.After a series of functional tests and performance tests, the system has realized functional requirements and non-functional requirements, in line with the expected assumption.

keywords: Music Comparison, Melody Similarity, String Matching, Music Analysis, Music Infringement

目　　录

插图清单

附表清单

第一章　引言

1.1　研究背景和意义

随着音乐产业与信息技术的日益发展和结合，先进的信息技术在一定程度上提高了音乐产业资本的利用率，同时也提高了音乐产业中人才的创造力和创作水平，加快了优秀音乐作品的流动速度，使全球音乐产业达到全新高度 [1]。根据国际唱片协会（IFPI）发布的《2020 年全球音乐产业报告》显示，2019 年全球的音乐产业市场收入已达到 202 亿美元，较去年同比增长了 8.0%。截至 2020 年上半年，我国数字音乐的用户规模已近 6.5 亿人，数量已经超过我国全部网民的七成。由此可见，音乐如今已逐渐成为人们精神文化生活不可缺少的一部分。

但是，音乐产业中的音乐作品借助互联网信息技术实现广泛快速传播的同时，也伴随着越来越多音乐侵权事件的发生。据统计，2019 年中国共受理超过 1.6 万起音乐知识产权纠纷案件，从案由分布情况来看，音乐知识产权权属问题和侵权纠纷案件数量占所有案件的 98% 以上。由此可见，当前音乐市场上针对音乐知识产权的侵权现象越来越普遍，由于以上数据仅包括创作者知情且引起较大纠纷而申请法律程序的案件，所以不难想象如今音乐产业中音乐侵权行为的数量规模。音乐作品属于知识产权的一种，音乐侵权的行为意味着音乐创作者所付出的努力与劳动都付诸东流，这会严重打击原创者对于音乐创作的积极性，进一步对音乐产业的发展产生消极影响。所以，结合当今的背景，保护音乐知识产权以及音乐创作者的权益是很迫切的。

从政府相关政策以及法律的角度来看，自 2017 年起，国家与版权相关的机构相继发布了有关版权工作规划以及知识产权审判的多个文件 [2]。截至 2019 年，我国正在以每年平均增加 3.5 家的速度建设知识产权保护相关的机构，累计建设了 20 家知识产权法庭以及 3 家知识产权法院。第七届中国国际音乐产业大会于 2019 年 12 月举办，大会上正式发布了中国音乐家版权保护与服务平台。由此可见，政府对于知识产权的重视程度越来越高，提供了更多的支持，从而为音乐产业的知识产权保护带来了更多便利。

从我国音乐市场的容量和用户趋势来看，截至 2019 年，中国的数字音乐产业收入规模已达 753.4 亿元，同比增长 23.02%。另一方面，众多音乐人开始加强对自身音乐作品的保护，中国音乐著作权协会（简称音著协）2020 年的许可费收益突破 4 亿元，但仍有 88.7% 的音乐创作者仍不是其会员，目前每年有超过 30% 的音乐创作者陆续加入该协会。此外，随着知识产权意识越来越深入人心，如今消费者们也产生了为优秀音乐作品买单的意识。在数字音乐用户中，进行过音乐内容付费行为的用户比例占 82.5%。盗版音乐逐渐成为人们憎恶和反感的对象，比如网易云音乐屡次侵权事件就触碰到了很多音乐爱好者的底线。网易云音乐曾在 2018 年 4 月不顾版权方的下架要求，在版权到期后将周杰伦的两百首热门歌曲以每首两元的价格进行打包售卖，试图以此来降低损失，但是这个行为已严重违反版权的相关规定。即使在版权授权期间，音乐平台都必须经过版权方的授权与同意。虽然后期网易云音乐一方针对这次事件做出了道歉和赔偿，但是这个事件对于很多网易云音乐的粉丝来说也是无法接受的，网易云音乐也因此被大量用户贴上了“吃相难看”这类的标签，此次版权事件也严重削弱了网易云音乐这个平台的口碑，导致了大量用户的流失。由此可见，音乐消费者已经开始重视音乐著作权的归属问题，并且支持维权行为。

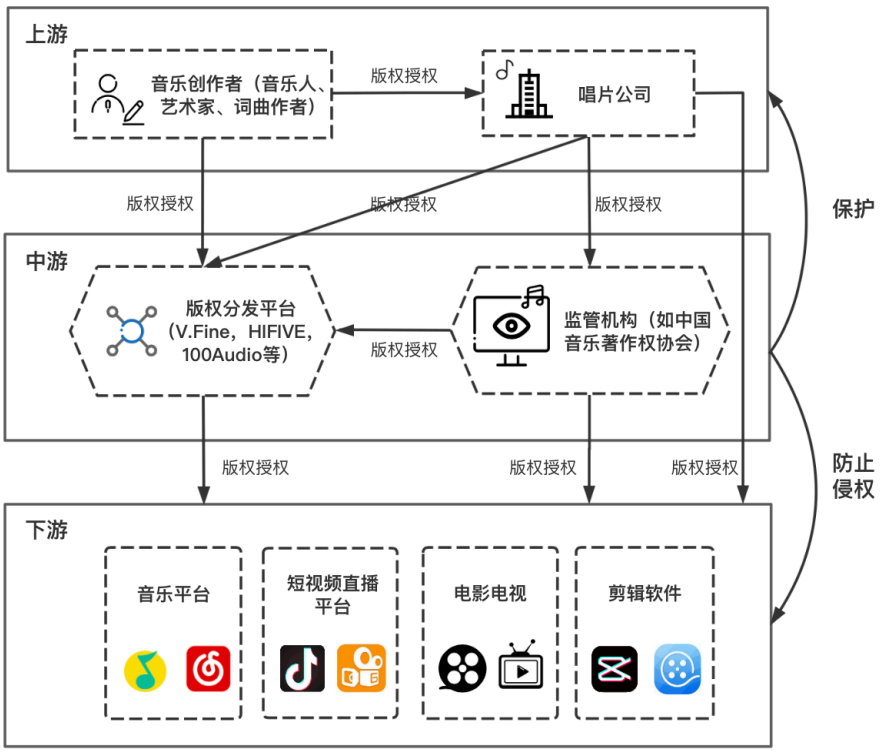


图 1-1: 音乐市场产业链

目前音乐市场的产业链大致分为上、中、下三个部分，如图1-1所示。产业链上游为音乐创作者以及唱片公司，是音乐版权的所有者，其中音乐创作者包括独立音乐人、艺术家以及词曲作者等；产业链中游为版权分发平台以及监管机构；产业链下游为音乐版权的使用者，主要包括音乐平台、短视频直播平台、电影电视、剪辑软件等。现有的音乐版权服务产品多由版权分发平台提供，但是目前旨在为原创的音乐作品提供版权保护手段、服务于监管机构和知识产权法院、用以辅助判定音乐作品是否侵权的产品却很少。

关于音乐侵权的判定，音乐产业中有一个普遍流传的说法，即“若音乐中存在 8 个小节雷同，那么就可以判定音乐作品存在抄袭侵权行为”。关于这个说法，有相关律师曾明确表明是完全不属实的。我国并没有关于音乐产业中音乐侵权的认定规则和相应的版权法，而我国关于知识产权的《著作权法》中也根本不支持该说法。对于法律程序中认定音乐侵权的相关规定，我国通常采取“接触加实质性相似”作为音乐侵权的判定标准，即被告侵权的一方在创作该音乐作品的过程中存在接触到原告一方音乐作品的可能性，并且疑似侵权的音乐作品与原音乐作品之间存在着实质性相似。美国司法界在判定音乐侵权时同样规定了接触原则和实质性相似原则 [3]。其中的“实质性相似”泛指疑似侵权的音乐作品与原音乐作品间具有较高的相似性，对实质性的相似判断缺乏具体检测的方法和标准，是一个很模糊的概念。针对音乐是否具备实质性相似，不同国家有各自的认定标准。

由此可见，目前音乐侵权的判定并不像商标和专利侵权那样有具体的量化标准，法官在处理音乐侵权的案件时更多的是依靠法官自身对音乐侵权的认知以及对音乐作品的主观听觉感受来判定。所以，在当前的背景下，借助客观且全面的音乐比对分析数据来辅助法院及法官进行较为公平且准确的判断是很有必要的。

综上所述，在音乐市场日渐繁荣以及音乐侵权事件增多的背景下，本文将构建一个基于旋律相似性的音乐比对分析系统，该系统可根据用户上传的音乐音频文件快速生成一系列客观的比对分析数据。该系统具有较强的实用意义，除应用到法院音乐侵权案件的判定外，也可用于音乐专业学生作品的查重、AI 作曲的结果鉴定以及音乐爱好者的日常识别比对等场景，有助于保护音乐创作者的劳动成果及声誉，对保护音乐知识产权、启发音乐创作灵感具有积极作用，从而促进音乐市场的持续发展。

1.2 国内外研究现状及分析

本文中基于旋律相似性的音乐比对分析系统的相关研究，主要针对以下两个方面进行展开，一是选择判定音乐作品是否侵权的指标，二是研究音乐旋律识别与比对技术。

1.2.1 判定音乐侵权的指标选择

判断音乐作品是否存在侵权行为之前，首先要了解音乐作品的含义以及侵权的常见方式。音乐作品从本质上来说是将多种声音通过旋律等元素创造性地组合在一起，具体通过音高、调性、调式、节奏、音色、组织形式以及其他构成元素来表达 [4]。目前针对音乐作品的侵权方式有很多种，主要包括音乐剽窃、盗版音乐贩卖、数字音乐侵权、背景音乐侵权、影视作品音乐侵权、表演音乐侵权等手段，其中音乐剽窃是最常见也是最容易引起激烈争论的一种侵权方式。本文中的音乐侵权研究均针对音乐剽窃侵权来展开。

在进行音乐作品比对时，需要观察音乐作品的诸多方面，而一般受保护的方面即是判定音乐作品是否具有实质性相似的方面，这个方面就是音乐作品中的旋律信息。旋律是构成音乐作品最主要的因素，主要由旋律线和节奏组成，节奏构成了音乐的大致框架，而旋律线则在节奏构成的框架中对其进行填充，使音乐饱满完整 [5]。在音乐版权的相关案件中，法院讨论最多的是音乐作品的旋律，旋律信息已成为决定法官判断的最主要因素 [6]。更准确地说，在两个音乐作品的旋律完全不同的情况下，无论音乐中其他要素的相似程度如何，都不会构成音乐抄袭的嫌疑。以我国歌曲《吉祥三宝》与法国歌曲《蝴蝶》为例，这两首音乐作品虽然节奏节拍、演唱方式、填词方式相似程度都很高，但是两者的旋律却完全不同，因此《吉祥三宝》并不构成对《蝴蝶》的抄袭行为。尽管人们会对音乐侵权不自觉地生成主观判断，但法院法官针对该类案件的实际判决其实是基于对音乐作品更为复杂更为全面的考虑。

综上所述，法律界及音乐界中的众多研究表明，判定音乐剽窃的首要因素在于音乐中的旋律信息。法院法官以及相关学者通过研究学习历来案件最终总结出了这样的结论，音乐作品中只有其旋律信息才具有可版权性，其他元素如歌词、演唱方式、表演形式等均不具备单独受保护的专有权利。所以本文中我们将旋律作为音乐比对分析的主要因素，针对音乐旋律的多个方面来进行客观的数据比对与展示。

1.2.2 音乐旋律识别与比对技术

从上一小节中了解到音乐旋律是判定音乐是否侵权的首要指标，下面将针对国内外目前有关音乐旋律的研究进展展开调研与分析。针对音乐旋律的研究主要包括两大方面，一是旋律的识别匹配技术，该技术主要对音乐旋律进行轮廓比对，多用于音乐检索；二是旋律的详细比对技术，该技术主要对音乐旋律进行精准比对与分析。

旋律相似性是音乐学、音乐心理学、音乐分析学等领域的一个关键概念，目前已经在音乐分析 [7]、音乐创作 [8][9]、版权检测 [10]、音乐信息提取 [11]、音乐治疗 [12][13] 等领域中展开了探讨与研究。

目前国内外针对旋律识别匹配的研究较多，主要应用于歌曲的检索与分类、哼唱歌曲及翻唱歌曲的识别等场景。在国外的相关研究中，Ghias[14] 等人创新性地采用音乐分割的方式研究出了首个哼唱系统，该系统对音高变化趋势进行字符串模糊匹配，检索出音乐旋律相对应的歌曲信息。此后，该领域的许多研究人员在该哼唱系统的基础上不断进行改进与优化。McNab[15] 等人增添了音乐节奏作为特征信息来辅助歌曲的匹配与检索，进一步提升音乐检索的准确率。Rho[16] 等人结合了系统使用者的相关反馈，并使用遗传算法实现了效率更高的哼唱检索系统。Pardo[17] 等人针对旋律相似度的计算创新性地提出了两种方法，进一步提升系统性能。Justin Salamon[18] 等人增加了识别翻唱歌曲的功能，并通过 Qmax 算法进行旋律匹配来提升系统性能。Savage[19] 等人根据自动序列比对算法，提出了一种旋律一致性百分比 (PMI) 方法来量化旋律演化。Naziba Mostafa[20] 等人利用深度神经网络技术实现了一个既能检索歌曲又能将嗓音与之相似的歌手的歌曲推荐给用户的音乐检索推荐系统。

国内在音乐旋律识别领域也展开了很多研究，金毅 [21] 等人研究并实现了基于旋律特征的音乐匹配检索。在中科院研究的音乐检索系统 [22] 中，首次将 EMD 算法与 DTW 算法相结合，进一步提升系统的检索性能。李鹏 [23] 等人提出一种改进的自相关函数方法进行基音检测，有助于进行音频分析，完成基于内容的音乐检索。姚光超 [24] 等人采用 SPRING 算法显著降低了子序列匹配的复杂度，并结合 GPU 大大缩短了匹配时间。周利娟 [25] 等人提出了基于 TLDA 和 SVSM 的音乐检索模型，实现音乐个性化的分类管理，并提高音乐信息检索系统的性能。王培培 [26] 等人提出了基于得分矩阵的音乐哼唱快速检索技术，实现音乐的快速检索。

类似哼唱和听歌识曲这种应用场景偏于生活化和娱乐化，但是将旋律识别

匹配技术应用到判定音乐侵权这类场景显然专业度是不够的。现有的相似性度量方法往往是将音乐之间的距离概括为一个单一的相似数值用于歌曲检索，整体相似度这个度量指标的参考价值会因整首歌曲的时长而降低，这会掩盖音乐具体的相似细节。在评估两个音乐作品的相似性时，能够准确提取出相似的部分以及具体相似的时长是非常重要的。此外，相似的音乐旋律片段在整个音乐作品中的重要程度同样会影响实质性相似的判定。如果相似的旋律片段是原音乐作品中的副歌这类核心部分，即使时长不长也存在构成音乐侵权的可能性，反之如果相似旋律为非核心部分，则需要根据相似片段的时长进行进一步的综合判断。但是，在某些音乐侵权案例中，即使两个音乐作品在核心部分上具有几乎相似的片段，也可以通过对音乐作品的调性调式进行改编、增加引子和尾声等手段来降低歌曲的整体相似程度，逃避音乐侵权的嫌疑 [27]。

综上所述，针对较为严肃的应用场景，我们应该对音乐作品中具体的相似片段进行详细的描述与展示，以便于根据其时长以及重要程度来做出相应的判定。

针对音乐旋律相似性的精准定量研究也已得到了一定程度的关注，人们往往比较重视具体的相似片段，而且精准的比对分析结果更具说服力。旋律的精准匹配主要是借助搜索字符串匹配算法，经典的字符串匹配算法主要包括 KMP 算法、BM 算法以及 Sunday 算法等，针对这些算法业内研究人员也进行了改进研究。韩光辉 [28] 等人针对 BM 算法进行了改进，通过在预处理阶段对扩展模式串建立好后缀规则来提高匹配效率。鲁宏伟 [29] 等人在 KMP 算法的基础上进行了改进，该方法根据当前匹配字符的特征值计算出移动距离，在字符串存在重复字符的情况下能有效地提高效率。徐珊 [30] 等人在 Sunday 算法的基础上进行改进，该算法在每一次匹配前先进行判断，以减少无效的匹配从而提高匹配效率。此外，针对音乐旋律比对结果的展现形式，刘雨心 [31] 等人通过在匹配前将重复首字符缩短为一个字符的方式来提高匹配执行的速度。朱永强 [32] 等人结合 Unicode 中文单元的内部关联性，优化了 Sunday 算法的匹配规则，提高了匹配效率。Park[33] 等人提出了 cross-scape，是一种可视化展示旋律相似性的方法，展示两段 MIDI 的相似度以及热力图，包括旋律整体相似性与节奏整体相似性，提高了比对结果的可读性，采用编辑距离结合 Wagner-Fischer 动态规划算法来完成音乐比对。

目前音乐旋律精准比对相关研究主要存在以下问题。首先，比对时间较长。不同的算法比对性能相差较大，需要在尽可能短的时间内得到比对分析结

果；其次，结果的可理解性较低。音乐比对分析结果不够直观会造成用户理解上的困难。所以针对以上问题，本文中的音乐比对分析系统将会呈现出清晰易懂的客观比对分析结果，同时提高比对效率。

1.3 本文主要研究工作

音乐侵权现象的不断增多在一定程度上阻碍了音乐产业的发展与进步，本文设计并实现了基于旋律相似性的音乐比对分析系统，该系统的核心服务为音乐比对分析服务，针对目前旋律比对研究的不足进行改进，将更高效地进行多维度的比对分析，主要研究分为以下几个部分：

（1）基于音乐侵权的现状调研和前期分析，并结合多种应用场景，拆解出了基于旋律相似性的音乐比对分析系统的功能需求，设计和实现了包括数据交互模块、音乐旋律特征提取模块、音乐旋律相似性比对模块以及用户数据管理模块等多个模块在内的服务，满足了用户进行音乐比对和熟悉音乐侵权案例的需求。

（2）设计并实现了数据交互模块，支持用户上传 MIDI 格式或 MusicXML 格式的两段音乐音频文件，系统会将文件存储到后端服务器以便用户后续的使用查看。对于生成的音乐比对结果，用户可以选择下载前端可视化报告页面的 pdf 文件，便于用户后续对于比对分析结果的查看使用。

（3）设计并实现了音乐旋律特征提取模块，根据以上介绍的音乐知识对上传的音乐音频文件进行一系列的规范化预处理操作，再对音乐的旋律音程信息、节奏信息、调性调式信息进行旋律特征的提取操作，用于后续的旋律相似性比对工作。

（4）设计并实现了音乐旋律相似性比对模块，对提取好的旋律特征进行分割，并尝试多种字符串匹配算法完成旋律精确匹配，得出详细的相似片段数据，最终通过可视化的方法将结果进行多维度展示，构建出音乐比对的模型，该模型拥有较高的比对效率并且能够呈现出可读性较高的结果。音乐比对结果包括整体相似百分比、调式调性比对、相似片段 top3 的五线谱比对及音频试听、相似旋律片段位置与时长等信息。

（5）设计并实现了用户数据管理模块，用户可通过该系统对上传的音乐音频文件、音乐比对结果等信息进行管理操作，具体包括查看、下载、删除等操作。

结合具体设计与实现方案，使用 Vue 框架和 Spring Boot 框架对系统进行搭建与开发，使用 MySQL 数据库来存储用户上传的音乐音频文件、音乐比对信息以及侵权案例信息，并通过 RabbitMQ 技术实现异步化调用。此外，对系统进行系统测试与实验分析，验证了该系统的可行性和效率性能。

1.4 本文组织结构

本文设计并实现了基于旋律相似性的音乐比对分析系统，以网页的形式作为载体与用户进行交互。本文共分为六个章节，组织结构如下：

第一章为引言部分，主要介绍了基于旋律相似性的音乐比对项目的背景及意义，并从判定音乐侵权的指标以及旋律识别与比对技术两个方面分析了国内外相关研究现状。

第二章为相关概念与技术综述部分，主要介绍了本文需要学习了解的音乐知识，并针对项目中涉及到的核心技术框架和算法做进一步的介绍和分析，其中主要包括音乐理论知识、音乐旋律的特征提取方法、多种字符串匹配算法、Vue 框架、消息队列等。

第三章为音乐比对分析算法设计与分析部分，提出了一种基于旋律相似性的音乐比对分析模型，详细介绍了比对流程中的具体步骤，并通过实验分析和调查问卷验证了音乐比对分析模型的比对效率和比对结果的可读性。

第四章为音乐比对分析系统搭建部分，详细描述了基于旋律相似性的音乐比对分析系统的搭建与实现，主要包括系统需求分析、系统概要设计、系统数据库设计、系统详细设计与实现等部分。

第五章为音乐比对分析系统测试部分，验证了该系统的基本功能与效率性能。对系统的功能测试和性能测试进行详细设计说明，结果均符合预期。

第六章为总结与展望部分，总结本文主要工作，分析音乐比对下一步工作的主要方向，并对其未来前景进行展望。

第二章　相关概念与技术综述

2.1　相关音乐基础知识

2.1.1　旋律特征选择

音乐的旋律由一系列音符组成，这些音符可以充分表现音乐内容的特征并能描绘出音乐的主题。音乐的各种旋律形态是音符与音符在音乐行进过程中空间与时间的差异性所形成的。音乐旋律所包含的特征要素主要包括 [34]:

（1）音程

音程是用来形容两个音之间的高低关系的，可分为旋律音程与和声音程两类，旋律音程表示的是先后发声的两个音之间的高低关系，而和声音程表示的是同时发声的两个音的高低关系 [35]。旋律具体的起伏形态取决于旋律中的音与音之间的行进方向和距离，也就是音程的距离。旋律音程中的旋律线可以平缓起伏也可以剧烈起伏，不同的旋律线形态会为音乐增添不同的情感色彩。

（2）节奏、节拍

节奏是旋律形态的脉搏，各种音通过节奏型进行组合，创造出完全不同的旋律形态，从而赋予音乐完全不同的风格情感。节奏是由音长和节拍两部分信息构成的。音符按照其时值可分为全音符、二分音符、四分音符、八分音符和十六分音符等多种类型 [35]。节拍是指音乐旋律中各种具有规律的强弱组合，对交替出现的强弱规律进行记录描述。节拍具有多种样式，比如三拍子乐曲的强弱节拍一般为 “强-弱-弱”，四拍子乐曲的强弱节拍一般为 “强-弱-次强-弱”。

（3）调式、调性

调式是指若干乐音围绕某一个音（即主音或中心音）按照相对应的关系（高低关系、稳定性等）联结在一起的有机组织。调性是指调式所具备的特性，主要包含主音高度和调式类别。“大调性” 表示大调式所具备的特性，“小调性” 则表示小调式所具有的特性。

（4）音高、音色

音高，代表音的高或低，音高的差异将整个音域划分成了不同音区（高音

区、中音区、低音区），不同音区对应着不同的音色，不同音高的旋律会存在很大的差异性。音高相对高的音乐旋律会带给人们高亢、激昂的感觉；而音高相对较低的音乐旋律往往会呈现出沉稳、低沉的感觉。

（5）速度、力度

速度表示音乐旋律行进的快慢，可以大致划分成慢速、中速和快速三大类。力度表示音乐旋律整体的强弱类别，可以大致划分为中弱、弱、强、中强等类别。

综上所述，音乐的旋律包含诸多特征，本文将音乐旋律中的旋律音程和节奏作为旋律相似度分析的主要的比对特征。旋律音程信息描绘音高的前后行进走向，可通过前后音高的数值计算得出，可消除调性调式的不同而导致的差异；而节奏信息则是描绘旋律中各种节奏的快慢、长短、停顿等信息。

2.1.2 音乐音频的数字化存储

音乐音频目前有模拟音频和数字音频这两种主流的形式。其中，模拟音频主要是通过机械录音形成的连续的音频；而数字音频主要是通过二进制数字信号经量化、编码等一系列处理后形成的断续的音频。随着计算机的快速发展，数字音频带来了越来越多的便利，在音乐行业也开始广泛应用了数字音频的存储形式。但是，由于生成设备的不同，可能会导致数字音频文件格式的不同或不兼容，这给音乐作品的良好交流造成了一定的阻碍。

为了使优秀的音乐作品能够在不同的环境和平台下展现与使用，音乐技术人员研究并设计出可以用来复用并且顺畅交换乐谱信息的数字音频存储格式。目前在较为专业的音乐领域上，音乐作品主要存储在 MIDI 和 MusicXML 这两种格式中，增强了不同记谱软件设备间的协作与兼容 [36]。下面将对这两种存储形式分别进行介绍。

MIDI(Musical Instrument Digital Interface) 是国际上一种用于存储数字音乐和电子合成乐器的统一标准 [37]，又称乐器数字接口。通过计算机与音乐领域的逐步结合，MIDI 为多种设备提供了实现音乐作品交流的方式，规定了不同设备之间传输音乐数据的相关协议。MIDI 文件曾是完成音乐曲谱交换的重要格式，可模拟多种乐器的声音，但是 MIDI 存储方式对于记谱相关的信息也存在一些弊端，其存储的是音乐中简易化的关键指令，人们需要将包含关键内容的指令发送给音源，转换成声音后才可以听到音乐具体的样子。MIDI 文件都很小，因为本身存储的是一个个指令，对于多声部音乐、乐谱的显示方式、表

示具体的演奏法以及一些较为重要的符号等信息，MIDI 文件都无法有效地呈现，这也引出了音乐工作者目前使用较多的 MusicXML 存储格式。

MusicXML(Music Extensible Markup Language) 和 MIDI 类似，是一种用于电子乐谱信息交换与分享的文件存储格式，又称音乐扩展标记语言 [38]。该存储格式中的音乐信息可以被记谱软件、音乐数据库、演奏软件、音乐教育软件等大多数软件识别并进行复用。相比于 MIDI 存储格式，MusicXML 存储格式可以解决掉上文提到的 MIDI 存储格式的弊端，在进行音乐作品存储的过程中，该存储格式会提供更加丰富全面的音乐信息。MusicXML 通过 XML 格式的文件将音乐作品中的整体乐曲元素、属性等信息进行存储，结构更加简单且兼容性更高。目前的音乐行业开始广泛接触并使用 MusicXML 这一存储格式，它也渐渐成为了音乐作品曲谱文件分享、交流的默认标准格式，便于音乐工作者发布交互式的曲谱，方便地交换音乐数据，进行无障碍地音乐交流，进一步促进音乐产业的发展。

本文的音乐比对分析系统目前支持上传 MIDI 和 MusicXML 存储形式的音乐音频文件。虽然 MIDI 格式缺少一些信息，但是主要指令的展现模式为音乐旋律的研究提供了方便。我们处理包含较多信息的 MusicXML 格式，会先将其转化成 MIDI 格式，然后再对 MIDI 格式的音乐音频文件进行旋律相似性比对等工作。

2.1.3 装饰音、休止符

装饰音是对音乐旋律起到装饰作用的临时音符的总称。通过联结音与音的方式，从而使旋律活跃起来，并赋予其韵律和特色。装饰音使音乐曲调更加丰富饱满，同时塑造音乐的特征形态 [39]。音乐作品中经常使用的装饰音有很多，包括颤音、倚音、滑音、琶音、回音、波音等 [35]。不管是在我国民间音乐作品中，还是在早些的西方音乐作品中，都存在着大量的装饰音。音乐旋律中装饰音的灵活运用，能为平庸无奇的音乐作品增色，表达音乐创作者或激昂、或忧伤、或喜悦的情感。

休止符可用来标记音乐旋律中的静止或停顿，根据时长可将其划分为全休止符、二分休止符以及四分休止符等等。在整体基调较为活跃的音乐作品中运用休止符将会使音乐作品更为利落、干脆，在整体基调较为深沉的乐曲中运用休止符将会使音乐作品更为宁静、悠远。由此可见，音乐作品中休止符的灵活应用可以借助音乐传递不同的情感。

装饰音、休止符这类表现形式在音乐旋律中属于锦上添花的作用，更多的是情感上的表达，对于音乐主旋律来说起不到决定性的作用，所以本文在对音乐旋律进行比对分析时，会对旋律中的装饰音进行简化预处理、去除休止符等操作，进而提高音乐比对分析的效率。

2.2 音乐旋律特征提取

本文针对 MIDI 和 MusicXML 两种存储方式的音乐音频文件进行比对分析，其中，MusicXML 文件会生成对应的 MIDI 文件后再进行统一分析，所以需要针对 MIDI 音乐音频文件中的基本信息和特征信息进行提取操作，比如是否为 MIDI 文件、MIDI 文件的格式类型以及音符、时值、音高、节奏等旋律特征信息，用于音乐旋律的表示和相似性的比对。

首先，在对旋律特征进行提取操作之前需要对 MIDI 文件的基本整体结构进行了解，MIDI 文件主要包含头块和音轨块两个部分 [40]。其中，头块部分对整个 MIDI 文件的基本信息进行了描述记录，音轨块部分对其中包含着的多个 MIDI 数据事件流进行描述记录。MIDI 文件的基本结构如图 2-1所示。

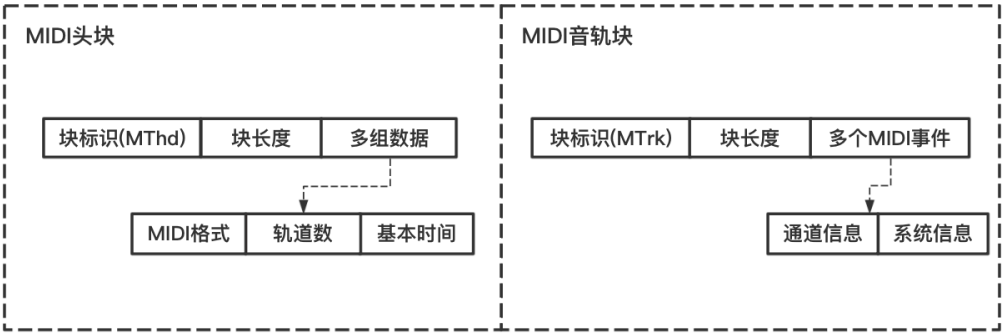


图 2-1: MIDI 文件基本构成

MIDI 文件的头块部分位于文件开头，由 14 个字节构成，主要分为三个部分，分别为类型、长度以及数据 [41]。块标识部分占据 4 个字节，表示该音乐音频文件的格式类型，其中 ASCII 字符 “MThd” 可以判定该音乐文件是否为 MIDI 格式的文件。块长度部分占据 4 个字节，表示该文件头描述部分的字节数长度，该数值固定为 6。数据部分占据 6 个字节，描述的是音乐音频文件的数据主体部分，包括 MIDI 格式信息、轨道数信息和基本时间信息，具体可以识别出单音轨、多音轨、文件中音轨块数等信息。

MIDI 文件的音轨块部分位于头块之后，其头部的构造与头块的头部构造很相似，均包含着块标识和块长度，音轨块的块标识为 ASCII 字符 “MTrk”。音轨块的数据区主要包含多个 MIDI 事件，这些 MIDI 事件可分为通道信息和系统信息。其中，通道信息包含通道模式信息和通道声音信息，记录其模式和声音的相关信息；系统信息包含系统一般信息、系统实用信息以及系统专用信息。

MIDI 文件具备多个轨道，这些轨道可以记录某种声音、某种乐器等任何声音。音乐工作者们在完成创作存储音乐作品时，会习惯性地将音乐作品的主旋律单独存放在 MIDI 文件 16 个通道中的其中一个通道中，不同的通道中音色各不相同。由于音乐创作者创作习惯的差异性，音乐作品中存放着主旋律的通道也具备一定的不确定性 [42]，所以需要寻找到记录主旋律的通道，再从中进行旋律相关特征的提取工作。

本文将利用 Music21 来进行特征提取工作。Music21 是一个由 MIT 开发的计算音乐学分析 Python 库，是一个用于分析、搜索和转换符号音乐数据的通用开源工具包 [43]，功能强大。该工具包中的模块化方法方便音乐家和研究人员快速使用，提供集成了复杂音乐知识的强大软件工具。

Music21 包含很多对象，包括音高、和弦、时值、音程、乐器和标准装饰物等。对于 MIDI 格式的音乐音频文件来说，每一类的特征都是用 0-127 之内的数字进行描述或计算的 [44]，大部分基本特征如音高、时值、节奏等信息均可以直接进行提取。Music21 的特征模块集成了其他工具包提供的标准特征提取工具，而且还增加了新的工具，允许研究人员快速编写强大的特征提取方法，利用系统的内置功能来解析不同数据格式并处理复杂的乐谱。

2.3 编辑距离

编辑距离有多种定义，主要差异取决于对字符串处理种类的不同。当今提到的编辑距离大多数指的是莱文斯坦距离（Levenshtein 距离），Levenshtein 距离是由著名数学家 Vladimir Levenshtein 于 1965 年提出的一种度量字符串之间差异程度的方法 [45]。

字符串之间的编辑距离，通过字符串转换为另一个字符串所需的最少操作次数计算得出，从而判断其相似情况。编辑距离的操作包括插入一个字符、删除一个字符和替换一个字符。例如，“MUSIC” 字符串和 “MASTER” 字符串，“MUSIC” 依次经过三次替换操作和一次插入操作变为 “MASTER”，所以这两个

字符串之间的编辑距离为 4。通过编辑距离，得出的是两个字符串的整体差异情况，经常应用于字符串近似匹配的场景下。通过编辑距离也可得出详细的字符串匹配结果，需要按照递归的方式从编辑距离矩阵的右下角开始寻找该单元格值的出处，共存在三种情况，分别为它的上方单元格、左上方单元格或上方单元格，按照此方法一直寻找，从而得知两个字符串的详细匹配结果。此外，可结合动态规划的思想完成编辑距离的计算，对其进行优化。

得出 Levenshtein 距离后，两个字符串的相似度可按照公式 2-1进行计算：

$$Similarity = 1 - \frac{Levenshtein_{MN}}{Max(L_M, L_N)} \tag{2-1}$$

其中，M、N 为进行比对的两个字符串，$Levenshtein_{MN}$ 表示两个字符串之间的编辑距离，L_M 表示字符串 M 的长度，L_N 表示字符串 N 的长度。通过该公式计算得到两者之间的相似性，该数值的值域为 [0,1]。该数值越大，代表这两个字符串越相似，反之则代表越不相似，从而进一步评估出两个字符串的相似程度。

2.4 字符串精准匹配算法

字符串匹配算法目前已经在多个领域被广泛应用，比如自然语言处理中的拼写检查、生物学中判断两段 DNA 的相似程度、文本编辑比对等。根据匹配的精确度可将其划分为精确匹配和近似模糊匹配两种类型。字符串精准匹配不同于字符串近似匹配，计算机领域的学者们根据字符串匹配的不同种类已经研究出多种经典且高效的字符串匹配算法。在本文的音乐比对分析系统中，需要利用字符串的精准匹配算法将分割后的旋律与目标旋律进行匹配，完成音乐旋律特征的的详细比对分析。下面分别针对 KMP 算法、BM 算法、Sunday 算法以及 Sunday 的改进算法这几种字符串精准匹配算法展开详细介绍。

2.4.1 KMP 算法

KMP(Knuth-Morris-Pratt) 算法，是由 D．E．Knuth 与 V．R．Pratt 和 J．H．Morris 这三位于 1977 年联合推出的算法 [46]，并最终以三人的名字共同命名。不同于 BF(Brute-Force) 暴力算法，KMP 算法可以减少发生不匹配时字符串的移动距离，从而降低了匹配的时间复杂度。

KMP 算法在进行匹配操作之前，会先针对模式串中的每一个字符计算出其对应的失效函数，进而计算出模式串中每个字符对应的 next 数值，构成 next 数组。该数组记录了各字符前的字符串中具有相同前缀的最大长度，辅助在字符匹配失败时模式串的跳转。当模式串和主串的对应字符发生不匹配时，无需像 BF 算法那样均回溯到开头位置，而是直接将模式串移动 next 数组对应数值的位置即可。

在字符串匹配的过程中可能存在着大量的失效字符，使用 KMP 算法时，当模式串与主串发生不匹配时，可以利用先前匹配得到的模式串部分匹配结果来忽略掉失效字符，通过失效函数计算出应该移动的距离，将窗口尽可能地向后移动再继续比较，从而提升匹配效率。KMP 算法在一定程度上提升了字符串的匹配效率，但过程相对复杂且理解相对困难。

2.4.2 BM 算法

BM(Boyer-Moore) 算法，是由 Robert S.Boyer 教授和 J Strother Moore 教授于 1977 年研究提出的一种构思更加巧妙的字符串匹配算法 [47]，相比于 KMP 算法来说，BM 算法的效率更高且更易于理解，具有亚线性的平均时间复杂度。与 KMP 算法一样，BM 算法在匹配前也需要对字符串进行预处理操作。在预处理阶段，需要计算出两个关键的规则相关函数，分别为坏字符规则 (Bad Character Rule) 和好后缀规则 (Good Suffix Shift)。

坏字符规则：将主串中与模式串中匹配失败的字符称为坏字符。通过计算坏字符在模式串中的位置与坏字符在模式串中最右端出现的位置之差来表示模式串需要向后进行移动的具体位数。当主串和模式串中对应字符匹配失败时，需要观察坏字符存在的位置，若坏字符存在主串中，则直接将匹配窗口移动到坏字符所在位置下一个字符后的位置，再继续进行比较；若坏字符存在在模式串中，则将与主串不匹配的字符与对应的模式串字符对齐后，再继续比较。

好后缀规则：将主串中与模式串中由后至前匹配成功的字符串称为好后缀。通过计算好后缀在模式串中的位置以及在模式串中上一次出现的位置来确定模式串需要移动的具体位数。根据好后缀规则进行移动主要分为以下三种情况。第一种情况，与 KMP 算法类似，当对应字符匹配失败时，若模式串中存在与好后缀匹配的子串，则将模式串向后进行移动，使模式串中的好后缀与其对应主串的好后缀对齐；第二种情况，当对应字符匹配失败且模式串中没有与好后缀匹配的子串，可寻找一个与好后缀匹配的最长前缀，将模式串向后移动

使之与其对应的好后缀子串对齐；第三种情况，当匹配失败且模式串没有一个直接与好后缀匹配的子串的情况下，且没有找到一个与好后缀匹配的最长前缀时，可直接将模式串移动到好后缀的下一个字符。

BM 算法使用坏字符规则或好后缀规则来尽可能避免无效的字符匹配，从而完成模式串的跳转，可分别对好后缀和坏字符跳转的距离进行计算，将两者中较大的数值作为模式串移动的距离。这类跳跃式的匹配将会大大提高了匹配的效率，使得 BM 算法相比其他算法更加高效。BM 算法在一般情况下相比于 KMP 算法的匹配速度至少提升 3 至 4 倍。但使用 BM 算法丢失了可以从上一次失败匹配中获取的信息，因此导致了每次匹配时都要对模式串的字符重新进行逐一匹配，存在重复性操作。而 KMP 匹配时使用了上次的匹配信息所以无需每次都从首个字符开始匹配。

2.4.3 Sunday 算法及改进

Sunday 算法是 Daniel M．Sunday 于 1990 年提出的一个基于 BM 算法的字符串匹配算法 [48]。Sunday 算法与 BM 算法进行字符串匹配的核心思路很接近且具有更高的匹配效率。其通过减少匹配次数来提高字符串匹配算法的执行效率，主要思想是：当对应字符匹配失败时，模式串的移动距离决定于位于模式串最后一个字符对应主串中字符的下一个字符。若该字符存在于模式串中，则将模式串中最右端的该字符与主串中的该字符对齐，即移动位数 = 模式串中最右端的该字符到末尾的距离 +1；若该字符不存在于模式串中，则直接跳过该字符，将模式串移动至此字符的下一个字符的位置，即移动位数 = 模式串长度 +1。在对应字符匹配失败的情况下，Sunday 算法能够跳过尽可能多的失效字符来缩短匹配时间，每一步的移动量都比较大，从而会提高匹配效率。

Sunday 算法已成为以上经典字符串精确匹配算法中执行效率最高的方法，但是 Sunday 算法每次匹配都是从模式串的首字符开始依次进行匹配，直到发生不匹配时才按 next 数组进行跳转，进入下一轮匹配。结合以上问题，可对 Sunday 算法进行了改进，其核心思想是在每一轮匹配开始之前，都要检验模式串的最后一个字符对应在主串位置上的字符是否存在于模式串中。若存在，则按照 Sunday 的匹配方法进行匹配；若不存在，模式串直接移动模式串长度的距离，不对其进行匹配，这样就避免了部分字符的无效匹配，提高了执行效率。实际上，改进的 Sunday 算法就是在 Sunday 算法的基础上对其中一种无效匹配情况进行排查。

2.5 Vue 框架

Vue 框架是由尤雨溪 (Evan You) 于 2014 年发布的一个开源的前端渐进式 JavaScript 框架，并拥有完整的生态平台。早期 Vue 框架的开发借鉴了 Angular 框架和 React 框架，Vue 框架在其基础上进行构建与改进，更加轻量且更加高效。本文将采用 Vue 框架完成该音乐比对分析系统的前端网页实现工作，实现前后端分离，提高代码的复用率。

Vue 生态系统由 Vue.js、VueRouter、Vueify、Vuex 和 VueCLI 等核心插件构成，开发人员可以根据自己的需要自由地选择是否使用某插件。其中，VueRouter 是官方支持的 Vue.js 路由库，主要用于构建单页面的 Vue 应用，还可提供导航守卫功能。Vueify 是 Vue.js 应用率很高的组件库，包含了工具和插件、大型社区、专业服务等丰富的生态系统。Vuex 是 Vue.js 的应用状态管理模式，通过该插件可以方便地共享组件状态。VueCLI 是支持 Vue.js 的脚手架工具，它基于著名的前端打包工具 webpack 进行构建，并针对 Vue 进行定制化，具有功能丰富、易于拓展等优势。

目前，Vue.js 已经被广泛应用在各种项目实践中，具备很多优势与特性。下面分别对 Vue.js 的轻量级框架、双向数据绑定以及组件化特性进行简要介绍。

第一，轻量级框架。Vue 是一个基于 MVVM 模式的 JavaScript 库，解决了 MVC 模式的模型代码少、控制器代码多、性能测试难度大等问题。但是 Vue 并不是传统的 MVVM 框架，Vue 框架的核心只有 State 和 View，更加专注于 View 层。Vue 框架更为轻量，其采用了自底向上增量式开发的设计方式，可实现对模版和计算属性的自动追踪。对于相关的开发者来说易于学习并使用，与项目的其他结构整合起来也较为容易，从而提升开发效率。

第二，实现双向数据绑定。保留了 Angular 框架的特点，提高了在数据操作方面的操作效率。使用中间的监控者对两侧的数据和操作进行监控，任意一方一旦发生改变，就会通知相应的另一方修改更新数据，从而实现双向数据绑定。

第三，实现组件化。Vue 将网页划分为出一个个独立且可复用的组件，通过对其进行合理的抽象，从而进行组织和管理。保留了 React 框架的特点，能够将前端开发语言 JS、HTML 和 CSS 写在一个文件里，实现了对 HTML 代码的封装和重用，提高了可读性与可维护性。

2.6 Docker 容器技术

在传统软件开发过程中，由于软件开发和运行环境的不同，相关技术人员很可能会产生分歧，从而影响开发效率。但是，容器技术的出现解决了这类问题。容器技术是将软件项目代码及其运行环境的标准化单元完成打包操作。使用容器技术后，开发人员只需专注于软件业务逻辑的实现，而运维人员只需专注于代码的打包和部署，不用担心环境是否兼容，因为容器中的代码是可以完全脱离容器所处的特定环境而独立运行的，并且可以实现相对应的功能，十分便捷且高效地使其从一个计算环境快速可靠地转移到另一个计算环境。

在容器技术出现之前，人们经常使用虚拟机技术，是另一种较为成熟的虚拟化技术。虚拟机技术是虚拟出一套硬件，运行一个完整的操作系统。在此基础上再进行软件的安装和运行，实现资源的隔离与分配。但是传统的虚拟机技术存在着资源占用多、步骤冗余以及软件启动速度慢这类的问题，技术人员在使用过程中很不方便，效率极有可能并没有得到提升。然而，不同于虚拟机，容器内的应用是可以直接运行在宿主机内核上的，所以软件会更加轻便，且具备更快的启动速度。此外，容器技术会提供开放接口给相关的技术人员进行使用，从而达到方便高效地进行合作的效果。

Docker 是一个通过 Go 语言完成开发的容器应用引擎 [49]，其相当于将程序置于一个可移动的容器中，提供了一种实现自动化部署的系统化方法。通过使用 Docker 技术，我们可以提升在交付和部署应用或软件时的效率，舍弃掉帮助文档和安装程序的形式，Docker 更加快速便捷地进行打包镜像并发布测试，最后一键运行即可。此外，使用 Docker 技术还可以更简便地完成升级、扩容以及缩容等操作，从而降低系统运维工作的复杂度。目前 Docker 已经具备良好的开发者生态及社区，有助于实现持续集成和持续交付，是较为成熟的容器技术并且广泛应用于各大互联网公司的产品中。本文中的音乐比对分析系统将选用 Docker 技术对音乐比对分析模型进行容器化处理。

2.7 RabbitMQ 消息队列技术

消息队列 (Message Queue)，简称 MQ，是一个在消息传输过程中用于存放任务消息的容器，常用于分布式系统之间进行完成通信工作。分布式系统通信的方式主要有两种，一种为直接通信，另一种为借助第三方完成通信，而消息

队列属于后者。消息队列技术通过异步处理来提升系统的性能，当系统需要获取某个消息时，可以从消息队列中直接获得相关消息，减少等待延迟的时间，提升用户的使用体验和系统的吞吐量。同时，消息队列技术还可以降低系统的耦合性，可提高系统的可维护性和容错性。此外，还可以通过削峰填谷来提升系统的稳定性。目前使用较多的消息队列技术包括 RabbitMQ、ActiveMQ、Kafka、RocketMQ 等。本文的音乐比对分析系统中使用了 RabbitMQ 消息队列技术，下面针对 RabbitMQ 技术展开详细说明。

RabbitMQ(Rabbit Message Queue) 是由 Rabbit 科技有限公司通过 Erlang 语言编写的一个轻量级开源消息队列中间件，支持多种客户端和操作系统，实现了 AMQP 高级消息队列协议架构设计。AMQP 是针对面向消息中间件设计的一个应用层开发标准协议 [50]，全称为 Advanced Message Queuing Protocol。AMQP 协议通常被划分为三层，分别为模型层 (Model Layer)、会话层 (Session Layer) 以及传输层 (Transport Layer)。其中，模型层负责业务逻辑的实现；会话层负责异步信号传输的实现，为客户端和服务器端的传输提供可靠保障；传输层负责数据传送，与会话层保持交互。三层协议互不干扰、共同协作，实现异步消息的可靠传输。

RabbitMQ 支持多种消息分发模式、提供多种确认机制以及持久化策略 [51]。消息分发模式目前共四种类型，包括 direct、fanout、topic 以及 headers，使用者可结合业务场景和技术方案采用相应的消息分发模式。RabbitMQ 提供了消息发送和消息接收两种确认机制。持久化策略使其在牺牲性能的情况下，同时拥有故障恢复的能力。通过日志记录每一条未被消费的消息，从而降低数据丢失的风险。在本文的音乐比对分析系统中，主要通过 RabbitMQ 完成音乐比对服务的异步调用，更好地提升音乐比对分析系统的效率性能及用户体验。

2.8 本章小结

本章主要介绍了项目中涉及到的音乐基础知识以及相关技术综述。首先，介绍了该音乐比对分析系统所需要了解的音乐基础理论知识。然后，详细介绍了进行音乐旋律特征提取的方法，采用 Music21 库作为提取工具，作为后续对旋律特征进行分析的基础。其次，详细介绍了旋律特征比对的相关算法，包括编辑距离、KMP 算法、BM 算法、Sunday 算法及其改进算法，方便后续对多种匹配算法进行比较从而选择效率更高的算法。随后分别介绍了前端 Vue 框架以

及后端 Docker 容器技术和 RabbitMQ 消息队列技术。

第三章　音乐比对分析算法设计

3.1　音乐比对分析算法整体概述

本文构建了一个基于旋律相似性的音乐比对分析模型，实现音乐比对功能。音乐比对是该音乐比对分析系统中最核心的功能。高效的计算方法会在软件产品开发的流程中起到非常重要的作用，所以为提高该音乐比对分析系统的比对效率和比对结果可读性，设计了以下针对旋律相似性的音乐比对分析算法。

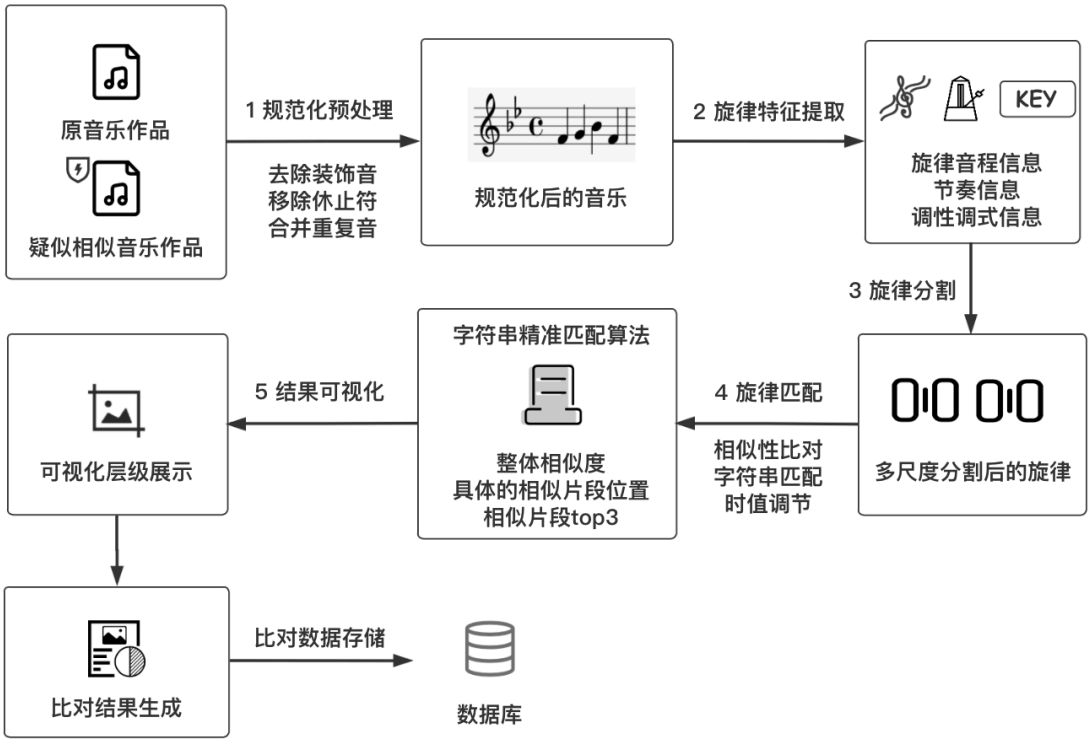


图 3-1: 音乐比对算法整体设计

该系统进行音乐比对的主要流程如图3-1所示。首先，对上传的音乐音频文件进行规范化预处理操作，主要保留两个音乐音频文件的主要部分，便于对后续对音乐旋律的特征提取与相似性比对工作；第二步，对音乐音频文件进

行旋律特征的提取，主要针对音乐的旋律音程信息、节奏信息、调性调式信息进行提取，为后续比对工作做准备；第三步，旋律分割，对音乐旋律进行多尺度分割操作，分割成长度不同的旋律特征序列子串；第四步，进行旋律精准匹配，通过特定算法进行旋律多个特征信息的相似性比对，并根据时值调节器进行音乐旋律长度复原，便于计算旋律音程和节奏的整体相似度；最后，将结果进行可视化显示，将结果进行多维度的呈现，同时，将比对结果保存至数据库中。

该音乐比对分析系统的算法设计主要根据音乐旋律特征来设计一系列音乐旋律比对流程，减少不必要比对，并通过改进的精准匹配算法提高相似性比对效率，最终展示多方面的比对结果，增强比对结果的可读性。下面我们将对以上几个步骤分别展开详细的设计说明。

3.1.1 旋律规范化预处理

首先，需要对上传的音乐音频文件进行规范化预处理操作。规范化预处理包括音乐音频文件格式转换、去除装饰音、移除休止符以及合并重复音等步骤。旋律规范化预处理主要目的是去除掉音乐音频旋律中锦上添花的音符，这些音符的使用更多的是辅助情感上的表达，对于主旋律来说起不到决定性的作用。通过将这些音符进行预处理操作，实现旋律主要部分的保留精炼，辅助后续的相似性比对工作。

音乐音频文件的格式转换主要是利用 Music21 库将文件都转化成 MIDI 格式的文件，由于 MusicXML 文件中包含较多信息，所以需要将用户上传的 MusicXML 格式的音乐音频文件转换成 MIDI 格式，再针对转换成 MIDI 格式的音乐音频文件进行后续的处理。

算法 3.1通过 convert2mid() 方法实现了音乐音频文件的格式转换，首先导入 Music21 库，利用 Music21 库中的 converter 方法对格式为.xml 和.musicxml 格式的音乐音频文件进行读取，再将其转换成统一的.mid 格式的音乐音频文件，返回转换后的 MIDI 文件相对应的路径位置。实现了对音乐音频文件的格式统一，以便于后续的旋律特征提取操作。

上传的音乐音频文件经上述格式转换操作变成.mid 格式后，其已经将旋律中的装饰音按演奏法的形式进行记录，我们通过对音符时值小于十六分音符的音符进行检测识别装饰音，并完成简化装饰音步骤。简化装饰音的步骤示例如图 3-2所示，举例说明了从带有装饰音的曲谱，到生成对应演奏法的 MIDI 格

算法 3.1 音乐音频格式转换

Input:
song_path
Output:
转换为.mid 后的文件

```
xml_format ← xml,musicxml
dir_name ← os.path.dirname(song_path)
song_name ← os.path.basename(song_path).split('.')
if song_name[1] in xml_format then
    score_stream ← music21.converter.parse(song_path)
    score_stream.write('midi', fp=os.path.join(dir_name, f'song_name[0].mid'))
    return f'dir_name/song_name[0].mid'
end if
return song_path
```

式曲谱，到生成最终简化完成的曲谱的整个规范化过程。普通旋律片段规范化的示例如图3-3所示，这三个旋律片段经过规范化预处理后均为相同的音高序列，这些片段经过合并重复音等规范化操作后的音高序列均可规范化为 "66, 67, 70, 66" 这个序列。

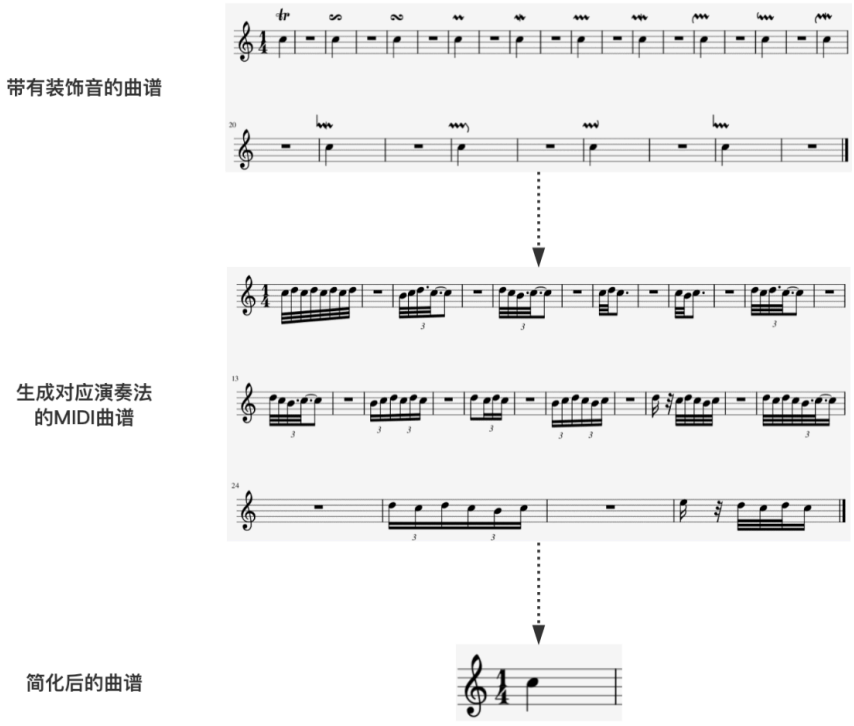


图 3-2: 装饰音简化示例

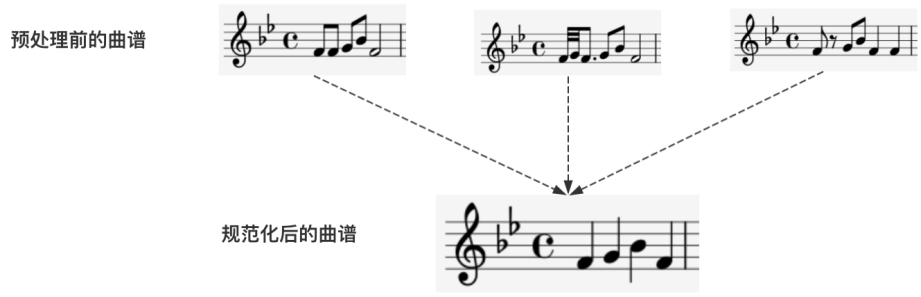


图 3-3: 旋律规范化示例

音乐音频的规范化预处理实现如算法3.2所示。预处理主要包括装饰音的识别与简化、休止符去除以及相邻重复音符的归并等操作，针对音乐旋律 *melody* 进行规范化，进而得到预处理后的音乐旋律 *melody_norm*。

算法 3.2 规范化预处理

Input:
音乐旋律 *melody*
Output:
规范化后的音乐旋律 *melody_norm*

melody_norm ← M
检测识别 *melody_norm* 中的装饰音，包括颤音、波音、回音、滑音
if *melody_norm* 中音符时值 duration< 十六分音符的时值 **then**
    检测识别为装饰音，进行简化合并
end if
将所有音符时值设置为四分音符
移除 *melody_norm* 中的休止符
合并 *melody_norm* 中重复音符
return *melody_norm*

3.1.2 旋律特征提取

完成对音乐音频文件的规范化预处理操作后，我们将对音乐的旋律音程、节奏信息以及调性调式特征进行提取。旋律音程信息描绘在连续的音乐旋律中前后发声的音高走向；节奏信息描绘各种音乐节奏的快慢以及长短等表现；调性调式信息描绘音乐旋律的整体信息用来辅助判断。

提取音符、时值、音高、节奏等基本信息可直接使用 Music21 库中相对应的函数。其中，音符时值的数值对应关系如表3-1所示。在规范化预处理步骤

中我们对音符时值小于 1/16 音符检测识别为装饰音进行简化处理，所以音符时值最小的单位为 16 分音符对应的数值 1。音乐旋律中绝对音高的数值对应关系如表 3-2所示，其中音高通过 0 至 127 的一个数字表示，代表音符的高低。通过实践证明数值 60 代表的音高为 C4。依此类推，每升高半个音，对应的数值+1，同理，每降低半个音，对应的数值-1。

表 3-1: 音符时值对应表

音符	数值
全音符	16
二分音符	8
四分音符	4
八分音符	2
十六分音符	1

表 3-2: 音高数值对应表

	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

本文将使用旋律音程序列来表示旋律的音高特征，音符时值序列来表示旋律的节奏特征。一首音乐音频的旋律声部为 M，定义 $\alpha(M)$ 为 M 旋律的音高特征序列信息，$\theta(M)$ 为 M 旋律的音程特征序列信息，$\beta(M)$ 为 M 旋律的绝对音符时值特征序列信息，$\gamma(M)$ 为 M 旋律的相对音符时值特征序列信息。从前文

介绍的音乐基础知识可知，旋律音程表示的是音乐旋律前后音高的走向，旋律音程可以通过计算音高的数值来得出，其展现了旋律的主要轮廓特征信息。旋律片段被转换成了一系列的旋律音程和节奏特征信息的方式如图3-4所示，可以分别得到 $\alpha(M)$、$\theta(M)$、$\beta(M)$ 及 $\gamma(M)$，方便后续分别对这两个旋律特征进行相似性分析。

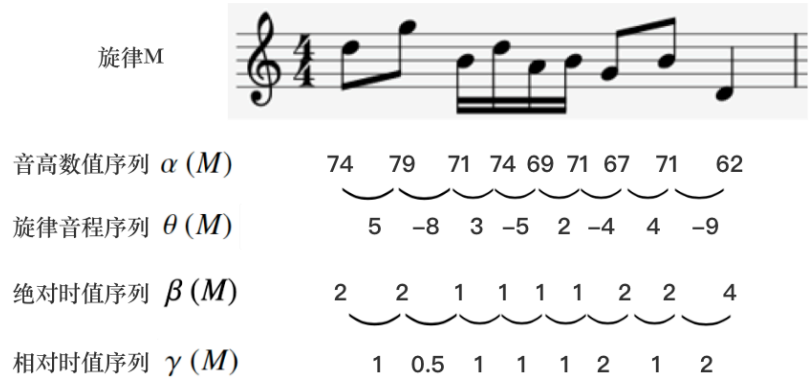


图 3-4: 旋律特征提取示例

Music21 库还易于编写新的 fem，继承公共超类特征提取器后，可以创建新的 fems 并与现有的 fems 一起使用。核心功能是在一个名为 midi_process() 的私有方法中实现的，该方法设置内部存储的特征对象的向量值。featureExtractor 超类提供了对各种乐谱表示的自动访问，可以扁平表示（.flat 属性）通常请求的音乐特征（如音高或音符时值的直方图）。

除了提取旋律音程和节奏相似度的特征外，我们还会提取音乐的调性调式作为整体比对数据之一，提供更加全面的音乐比对参考。本文使用 Music21 库中提供的调性分析函数，该函数通过 Krumhansl-Schmuckler 调性分析算法完成了对音乐旋律调性调式信息的提取操作，该算法通过计算当前音乐的音符总时长与 24 种不同调性特征权重的皮尔逊积矩相关系数，其中包括 12 个主音，并且每个主音对应两种调式，分别为大调和小调。通过与这 24 种情况进行比对，得出数值最大的相关系数，从而得到该音乐音频旋律对应的调性调式信息。

3.1.3 旋律分割

对上述操作后的音乐音频旋律进行分割操作，为后续方便寻找旋律之间精准匹配片段做准备。本文将采用多尺度分割的方法将音乐的旋律特征序列信息分割成不同长度的子串，并通过将这些子串作为搜索单元，分别与目标旋律特

征序列做比对的方式完成字符串匹配操作，以确定旋律中完全匹配的片段。即将多尺度分割出的子串当作字符串匹配过程中的模式串，将目标旋律特征序列当作字符串匹配过程中的主串，从而通过对其进行精准匹配完成后续的相似性比对。

假定旋律特征序列 $S = (m_1, m_2, m_3, ..., m_L)$, 其中 L 为旋律特征序列长度。$S_k^n$ 代表特征序列 S 中长度为 n 的第 k 个子串，即 $S_k^n = (m_k, m_{k+1}, m_{k+2}, ..., m_{k+n-1})$，其中，$1 \leq k \leq L - n + 1$。$S_k^n$ 子串用于计算两首乐曲旋律的局部相似性。以 ABCD 序列为例，展示了将旋律序列进行由最小到整个序列的多尺度分割示例如图 3-5所示。可见 ABCD 序列可划分为 10 个片段，最短片段长度为 1，最长片段长度为 4，即序列的总长度。旋律 S_k^n 表示的示例如图 3-6所示，S_2^2 表示子串 BC，S_1^3 表示子串 ABC。

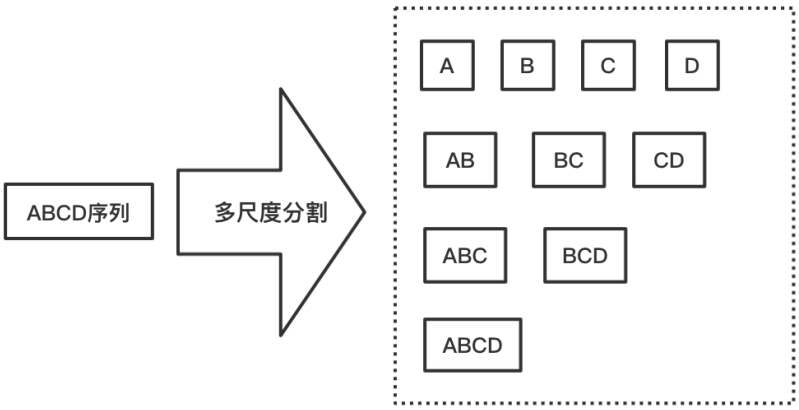


图 3-5: 多尺度分割示例

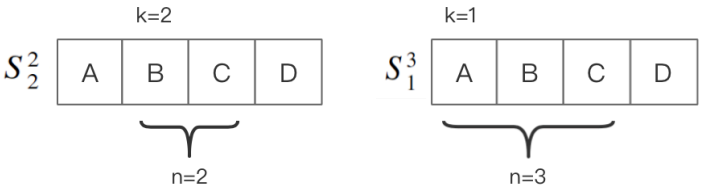
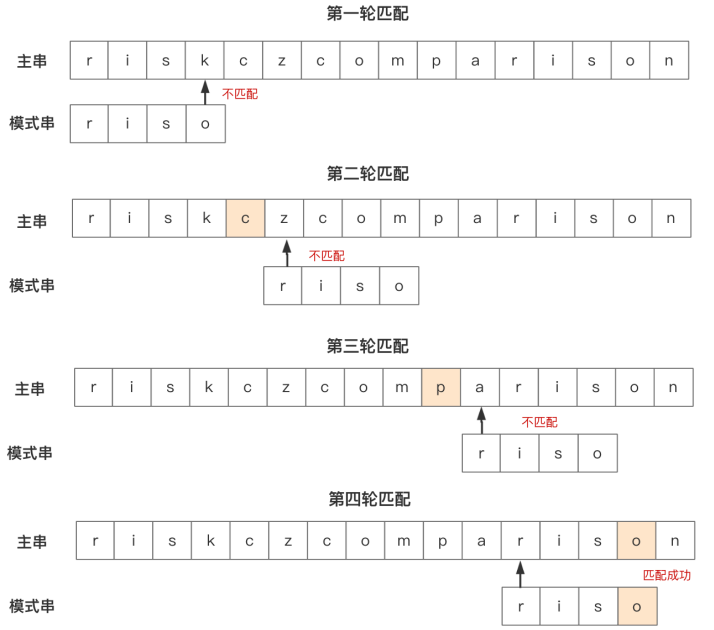


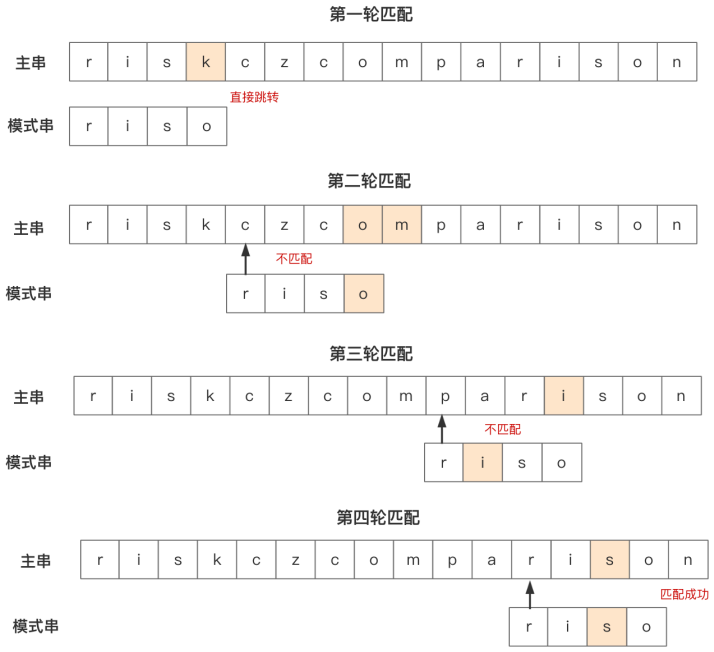
图 3-6: 旋律表示示例

3.1.4 旋律特征相似性比对

在旋律相似性比对的操作过程中，需要利用分割好的旋律特征序列进行字符串精准匹配操作，从而寻找旋律之间的匹配片段。



(a) Sunday 算法匹配过程



(b) 改进的 Sunday 算法匹配过程

图 3-7: 字符串精准匹配过程示例

通过尝试对比多种字符串匹配算法，最终选择一种改进的 Sunday 算法作为该音乐比对模型中寻找相似片段的方法。在字符发生不匹配时，该算法通过跳过尽可能多的失效字符来减少匹配的次数，从而减少字符串匹配的时间，所以每一轮匹配的移动距量都比较大，提高了旋律匹配的效率。本文中的改进算法在执行 Sunday 算法之前，先做一个判断，判断模式串的最后一个字符对应在主串位置上的字符是否存在于模式串中，若不存在，模式串直接移动模式串长度的距离，可以跳过更多的无效字符。

以主串为“riskczcomparison”，模式串为“riso”为例，使用 Sunday 算法和改进的 Sunday 算法进行字符串匹配的流程如图 3-7所示。可以看出，使用 Sunday 算法共进行了 10 次匹配，使用改进的 Sunday 算法进行了 6 次匹配，减少了匹配次数，从而提升了匹配效率。

3.1.5 旋律比对结果生成

通过输入的音乐音频文件以及经过多尺度分割后的音乐片段来寻找匹配的旋律片段，同时定义了特征序列匹配函数，该函数描述了 $S^{n}_{k(b)}$ 与 S_a 所包含的所有长度为 n 的子串的匹配情况。S_a 中所有匹配的子串都会保持在匹配结果列表中，会存放在一个二维匹配子串堆栈中。该二维匹配子串堆栈对于相似性分析至关重要，同时它也可以直观地展示匹配子串片段的具体分布情况。

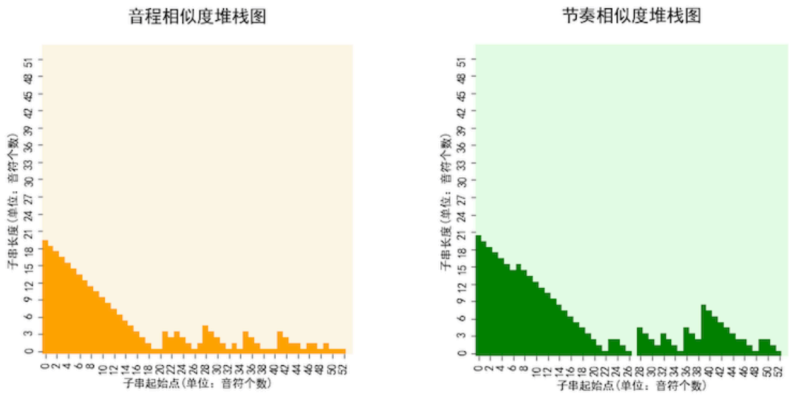


图 3-8: 匹配堆栈图示例

将二维匹配子串堆栈绘制出来，如图 3-8所示。其中横轴表示乐曲 1 中的各子串起始点位置，纵轴表示各子串的长度。如前文所述，我们提取旋律音程和节奏作为旋律比对的主要特征，分别使用橙色和绿色表示旋律音程特征比对

结果和节奏特征比对结果，以区分这两个特征。对应到子串匹配堆栈图，即高亮区域顶点所在的位置。此外，通过该图也可以得出所有相似旋律片段位置以及长度，以图3-8为例，由于图中的数值是相对的音高和节奏数值，且初始值为0，相对数值最小长度设置为2，所以根据左图中的第一个顶点可以看出，从乐曲2中能找到若干旋律片段，使其与乐曲1中第1个元素开始，序列长度为24的子串完全匹配。

子串匹配堆栈图的顶点搜索算法的实现思路如算法3.3所示。根据上述流程中得到的堆栈，通过遍历寻找到堆栈图中所有相似的波峰位置，以及波峰位置所对应的长度，并将其进行存储。分别对旋律音程信息和节奏信息进行顶点搜索操作，从而得到两个音乐音频文件中旋律音程相似片段和节奏相似片段的具体位置和长度。将得到的顶点集按长度从大到小进行排序，选取其中的top3进行详细相似信息的展示。

算法 3.3 顶点搜索算法

Input:
堆栈 *matrix*
Output:
所有顶点 *peak_infos*

```
peaks ← []
values ← []
if index_matrix[0] > index_matrix[1] then
    peaks.append((0 , index_matrix[0]))
    values.append(index_matrix[0])
end if
// 全部遍历一遍，找到所有的波峰位置，并存储相应的值
for idx in (1 , len(index_matrix)-1) do
    if index_matrix[idx - 1]<=index_matrix[idx]>index_matrix[idx+1] then
        peaks.append ((idx, index_matrix[idx]))
        values.append(index_matrix[idx])
    end if
end for
// 在 values 中找到波峰
max_num_index = self.getListMaxNumIndex(values)
peak_infos ← []
for max_num_idx in max_num_index do
    peak_infos.append(peaks[max_num_idx])
end for
return peak_infos
```

在计算旋律的整体相似性数值时，需要借助时值调节器，将匹配的旋律音程序列以时值作为权重，对其进行扩展与复原操作，即将相似的片段按照音符时值的序列信息进行长度的复原。时值调节器的原理如图3-9所示。

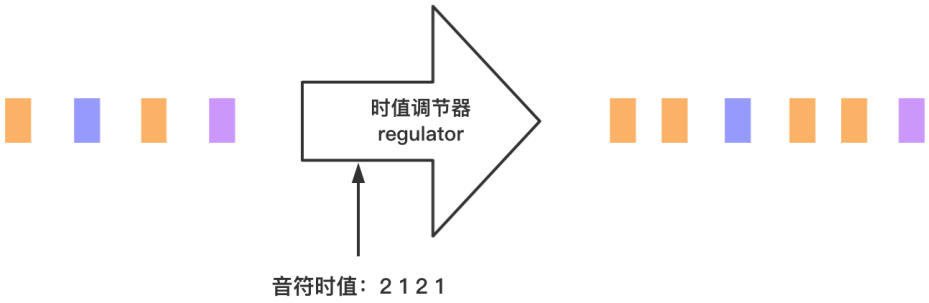


图 3-9: 时值调节示例

根据时值调节器调节后的旋律以及匹配堆栈的顶点进行旋律音程和节奏相似度的计算，其中，旋律音程整体相似度通过公式3-1计算，节奏整体相似度通过公式3-2计算。

$$Pitch\_Similarity\_Score = \frac{\sum_{(k,n)\in peaks_{\theta}} n}{\sum \beta(M)} \tag{3-1}$$

$$Rhythm\_Similarity\_Score = \frac{\sum_{(k,n)\in peaks_{\gamma}} n}{\sum \beta(M)} \tag{3-2}$$

最终该音乐比对分析系统将以两种形式进行结果展示，分别为整体比对情况和详细比对情况。整体比对结果包括整体旋律音程和节奏的相似性、相似性热力图以及详细的相似片段位置与时长；详细比对结果包括调性调式、相似片段 top3 的音乐曲谱片段并且提供相应音频，并将其写入 json 文件中。

3.2 音乐比对分析模型实验分析

音乐比对为该系统的核心功能，旨在对音乐进行客观的旋律比对分析。对于音乐相似的准确性判定，不同人有不同的标准，而且在各个场景下关于音乐是否侵权的判定还需要针对其片段的相似程度和重要程度等方面进行更加全面的考量。所以，本文实验分析部分主要针对该系统的音乐比对效率和比对结果

的可读性进行实验展开，进而评估音乐比对分析模型的效果。

但其实在该系统的音乐侵权案例功能中，通过该音乐比对分析模型针对侵权案例得出的分析结果可以看出，分析结果与法院审判结果相符，即对于判定结果为侵权的案件中，比对结果中明显存在多处相似且时长较长；对于判定结果为非侵权的案件中，比对结果中的相似片段较少且时长较短，并处于歌曲的非核心部分。这从侧面印证了该音乐比对分析模型的结果能够辅助法院法官进行判定，从而提高判定效率。

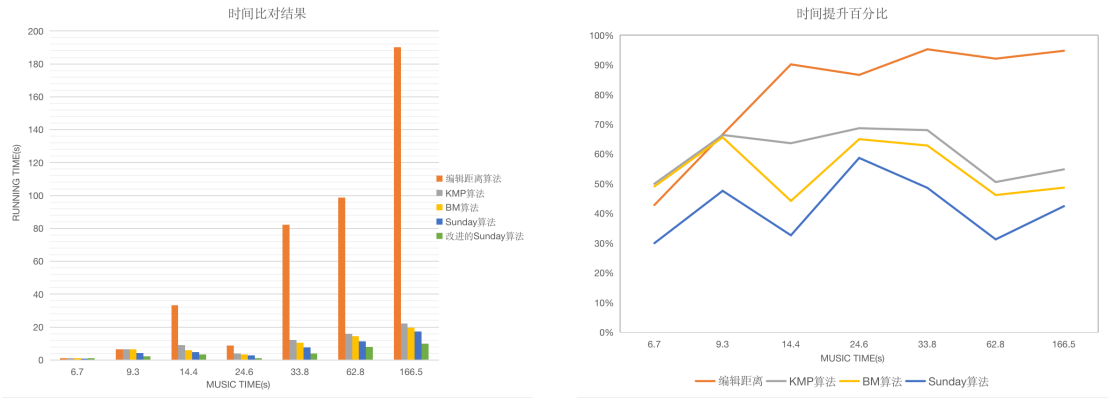
3.2.1 音乐比对效率

本文在网络上的寻找疑似相似的音乐作品，截取不同长度不同片段的音乐音频文件进行音乐比对，本文针对音乐比对效率实验的音乐文件数据集信息如表3-3所示。实验主要从两方面展开比对，一是比较多种匹配算法的效率，从中找到效率性能更佳的算法，应用到音乐比对分析系统的音乐比对模型中；二是将该音乐比对分析模型与目前已有的音乐比对工具 Park 模型 [33] 进行比对效率的比较。

表 3-3: 音乐比对数据集

编号	原作品	原作品时长 (s)	疑似相似作品	疑似相似作品时长 (s)	平均时长 (s)
1	狩猎的哥哥回来了	6.7	乌苏里船歌	6.7	6.7
2	No Scrubs	10.3	Shape Of You	8.2	9.3
3	oh why	16.4	Shape Of You	12.3	14.4
4	creep	32.8	Get free	16.4	24.6
5	Sogno Nostalgico	26.5	Girl In The Dark	41	33.8
6	我为祖国献石油	57	大海航行靠舵手	68.5	62.8
7	十月里响起一声春雷	175	我爱你中国	158	166.5

首先，本文在音乐比对的旋律匹配步骤中，通过替换采用多种字符串匹配算法实现比对算法效率的比较。本文比较了编辑距离算法、KMP 算法、BM 算法、Sunday 算法以及改进的 Sunday 算法，针对不同大小、不同时长的音乐文件采用以上几种算法的比对所需时间如图3-10所示。图中的音乐时长数据为两个比对音乐片段时长的平均值。



(a) 多种算法比对时间 (b) Sunday 改进算法比对时间提升百分比

图 3-10: 多种算法效率比对结果

由图 3-10(a) 可知，改进的 Sunday 算法相对于编辑距离算法、KMP 算法、BM 算法以及原始的 Sunday 算法的比对效率都有明显提高，而且随着音乐片段长度的增长，改进的 Sunday 算法比其他算法在音乐比对时间的优势更加突出。图 3-10(b) 展示了改进 Sunday 算法相比其他四种算法针对不同时长数据集的比对时间提升的百分比情况，改进的 Sunday 算法照比编辑距离算法比对时间平均提升 81.19%，照比 KMP 算法比对时间平均提升 60.31%，照比 BM 算法比对时间平均提升 54.52%，照比原始的 Sunday 算法比对时间平均提升 41.61%。本文的音乐比对模型选择改进的 Sunday 算法作为字符串搜索匹配的算法，具备较高的比对效率。

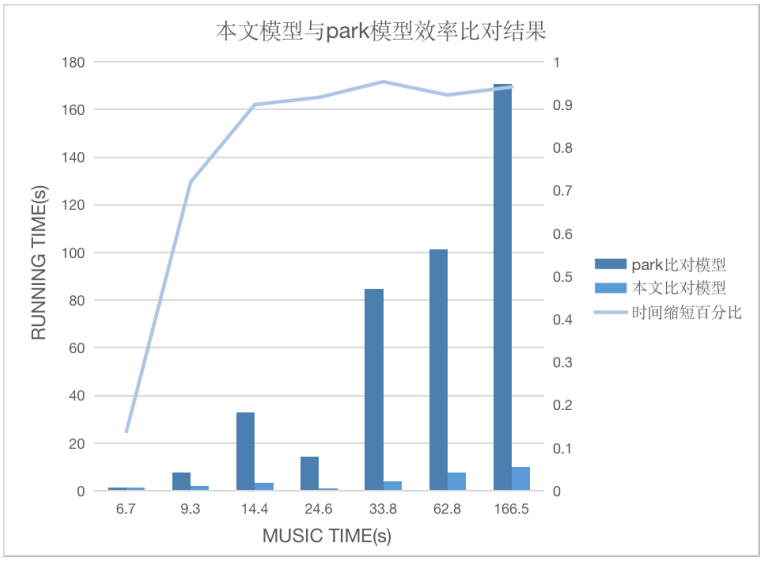


图 3-11: 与 Park 模型效率比对结果

再将本文中使用的音乐比对算法与目前已有的 Park 等人于 2019 年提出的音乐比对算法针对音乐比对功能进行效率上的比较，Park 模型中的比对算法主要是采用编辑距离结合 Wagner-Fischer 动态规划算法。针对上述数据集的比对效率对比如图 3-11所示，可以看出，本文中使用的音乐比对算法的效率要明显高于 Park 提出的音乐比对算法效率，随着音乐比对片段时长的增加，该音乐比对算法的比对时间提升更为明显，时间缩短已达 90% 以上。

通过针对以上两个方面进行对比的结果来看，本文中音乐比对分析系统使用的音乐比对方法具备较高的比对效率，能够快速给出给定音乐音频片段的比对结果。

3.2.2 音乐比对结果可读性

目前关于音乐详细比对的研究较少，本文将本文音乐比对分析模型、Park 音乐比对模型以及 LostinMusic 网站中的音乐比对分析结果进行可读性的直观比对与调查问卷分析。

前文中 Park 提出的音乐比对模型中在结果展示上有进一步的提升，如图 3-12所示，Park 提出的音乐比对模型将整体的比对结果进行可视化展示，但是该方法虽然将比对结果通过相似性热力图的形式进行了展示，但是并没有详细相似片段的解读和说明，用户难以理解两个音乐片段中相似片段的具体位置与长度，无其他可理解度较高的指标说明，理解难度大，且可能导致说服力较低的问题。

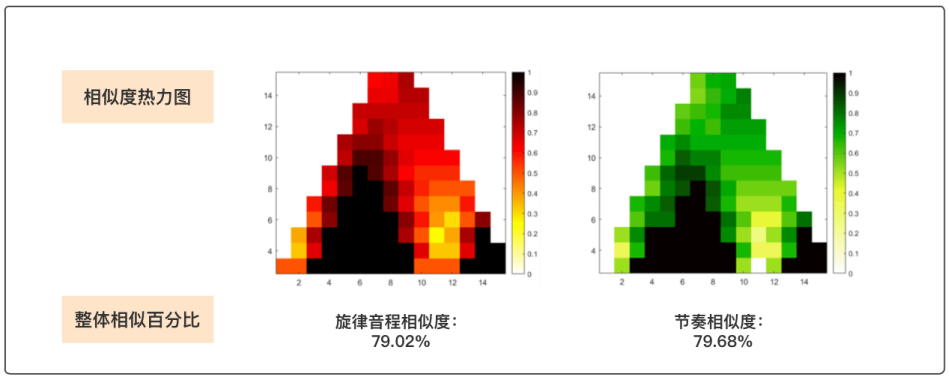


图 3-12: Park 模型比对分析结果

除 Park 等人提出的音乐比对模型外，国外有一个与音乐侵权相关的 LostinMusic 网站，其更倾向于普及类的网站，但其中的音乐证据模块也提供了

音乐比对的结果展现，如图3-13所示。可以看出，该网站的音乐证据模块中，只是罗列出两首音乐文件的五线谱和整体的音频文件，以及十分简短粗略的说明。

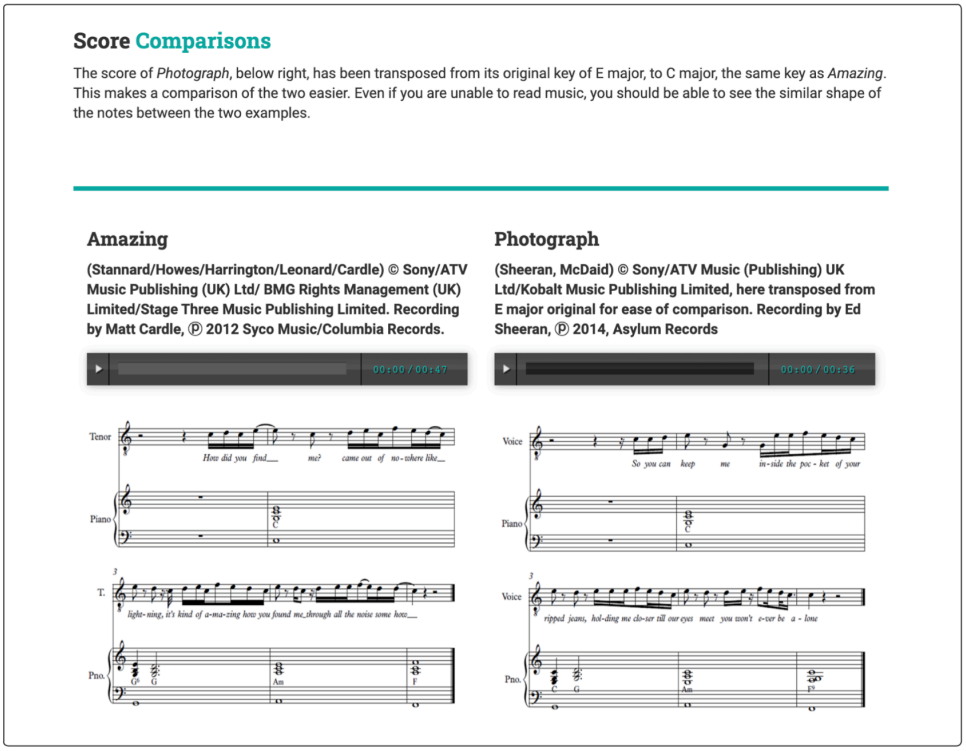


图 3-13: LostinMusic 比对分析结果

本文中的音乐比对分析系统结果如图3-14所示，可以看出结果更加直观易懂且更有说服力，比对角度较多，将通过整体比对结果和详细比对结果两大方面进行结果的分析与展示。整体比对结果中包括旋律音程和节奏的整体相似度百分比、相似片段具体位置与长度说明；详细比对结果包括调性调式信息以及相似片段 top3 信息。其中，相似片段 top3 将展示相似片段的五线谱比对以及时长比对图，并且可供用户进行音频试听。可以看出，该音乐比对分析系统的比对结果可读性和可理解性较高，从而提高用户在比对过程中的比对效率并且有更加直观的比对感受。

为验证本文的音乐比对分析结果具有更高的可读性和可理解性这一观点，我们邀请了 25 位典型用户作为参与者，他们分别有音乐学院的教授、音乐专业的学生、艺术院校的学生、具备音乐知识的音乐爱好者以及不具备音乐知识的音乐爱好者等等，来进行音乐比对结果可读性的实验。

图 3-14: 本文模型比对分析结果

我们要求每位参与者针对相同的音乐分别对填写一份评价音乐比对

结果的调查问卷。我们为本文的音乐比对模型、Park 的音乐比对模型以及 LostinMusic 音乐网站生成的三种音乐比对结果报告设计了一系列用户体验（User Experience）相关问题。我们针对性地设计了七个用户体验问题，并使用五点评分李克特量表对参与者的想法进行统计。李克特量表是由美国著名社会心理学家李克特提出的，该量表中的每一种陈述都将会给用户五种回答选项，根据认同程度的由高至低，分别对应着 5 分至 1 分，该量表目前已经在调查问卷中被广泛使用。该实验中每位参与者对每一个问题对于以上三种比对结果都需要进行回答。此外，对于 LostinMusic 网站中的比对结果已经翻译为对应的中文，以去除由于语言不同可能会产生的用户体验差异。

表 3-4: 比对分析可读性调查问卷结果

编号	用户体验问题	本文	Park	LostinMusic
1	我认为该比对结果很难理解	**1.56**	3.8	4.28
2	我认为该比对结果易于理解	**4.28**	2.48	2.44
3	我认为该比对结果有很多无用的信息	1.76	**1.32**	2.56
4	我认为该比对结果信息不全过于简单	**1.28**	4	3.92
5	我认为该比对结果有助于进行音乐比对	**4.48**	3.08	3.2
6	我认为该比对结果使我更加了解音乐	**3.8**	2.16	2.88
7	我愿意使用该比对结果的系统进行比对	**4.08**	2.88	3.08

25 位参与者在问卷中对于我们设计的七个用户体验问题进行五点式打分，三种比对结果的用户体验问题得分的均值如表3-4所示。我们通过对比三种音乐比对结果在每一个问题的得分来对问卷结果进行分析。由问题 1 和问题 2 的结果可以看出，从整体理解的难易程度上来说，本文的音乐比对结果的呈现方式相较于另外两种比对结果的呈现方法来说更加易于理解。由问题 3 和问题 4 可以看出，本文的音乐比对结果信息更加丰富且无用的信息较少，Park 模型的音乐比对结果内容过于简单，而 LostinMusic 网站上的比对形式较为简单而且存在着无用的信息，比如整段的音乐音频和整段的乐谱。由问题 5、问题 6 和问题 7 可以看出，本文的音乐比对结果可以较好地满足用户进行音乐比对的需求，通过学习音乐比对结果从而更加了解音乐相关的内容，并且有意愿使用该音乐比对结果的音乐比对系统进行比对。

根据三种音乐比对结果的直观比对以及对 25 位参与者进行调查问卷的结果显示，本文中音乐比对分析系统生成的音乐比对结果较为全面且具有较高的可

读性，便于用户理解。

3.3 本章小结

本章对音乐比对分析系统的核心功能音乐比对进行了算法流程的设计与实现说明，主要描述了规范化预处理、旋律特征提取、旋律分割、旋律特征匹配以及结果生成部分，并通过实验针对比对效率以及比对结果可读性两方面展开了设计与分析，通过选取多组不同时长的音乐片段进行比对时间的比较，通过进行调查问卷比较结果可读性。得出结论，该音乐比对分析模型能够以更高的效率生成高可读的音乐比对分析结果。

第四章　音乐比对分析系统搭建

本章将从基于旋律相似性的音乐比对分析系统的整体概述出发，分析目标用户需求，将其转化为该音乐比对分析系统的产品需求，进行详细的功能性和非功能性需求分析，对该音乐比对分析系统展开全面且多角度的概要设计说明，并对该系统的核心模块进行详细设计与实现。

4.1　音乐比对分析系统的整体概述

音乐侵权事件随着计算机技术与音乐的日益结合与发展正在悄然增多，借助全面且客观的音乐比对分析数据辅助音乐侵权相关案件的判定是很有必要的。针对这一背景，我们设计并实现了基于旋律相似性的音乐比对分析系统，旨在高效提供可读性高的客观音乐比对分析数据，便于应用在音乐侵权相关的各种场景。基于旋律相似性的音乐比对分析系统具体业务工作流程如图4-1所示。

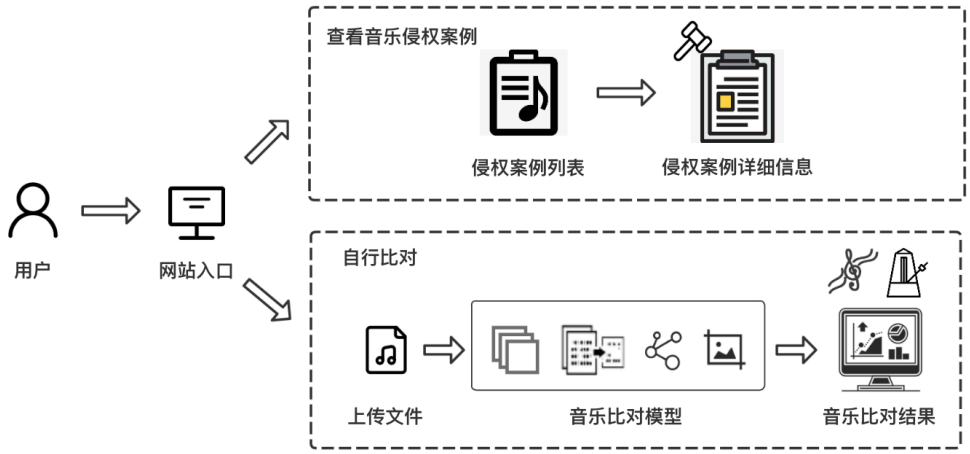


图 4-1: 系统工作流程

从具体业务交互流程上来看，用户通过浏览器进入音乐比对分析系统首页，首页中对该音乐比对分析系统进行简要介绍，并提供了音乐比对和音乐侵权案例的入口。

音乐比对界面中，用户通过数据交互功能上传两个音乐音频文件进行音乐比对，系统将对用户上传的文件进行分析，判定其是否符合该比对分析系统的比对要求，并将上传结果返回给用户。用户上传成功后，可以开始进行音乐比对。在比对的过程中，用户可等待比对结果，也可退出使用该分析系统的其他功能。系统首先将对用户上传的两个音乐音频文件进行规范化预处理、旋律特征提取、旋律分割操作，方便后续对旋律特征进行相似性比对分析，系统通过特定算法完成音乐旋律相似性比对后，将结果展现给用户，并保存至该用户的个人中心记录中。

音乐侵权案例界面中，用户可通过查看音乐侵权案例熟悉了解该系统的基本功能以及通过该系统进行音乐比对生成的客观比对分析结果。音乐侵权案例选取的均为音乐史上较为著名的音乐侵权案件，用户可对音乐历史上关于音乐侵权的相关案件知识有所了解。

此外，用户可通过个人中心中的比对记录来查看已上传的音乐音频文件、上传时间、比对状态、比对分析结果等信息，用户可根据自己的需要对其进行查看、下载保存、删除操作。通过全面且多维度的客观比对分析结果，辅助音乐侵权事件的判定。

4.2 音乐比对分析系统需求分析

需求分析是软件开发生命周期中核心的步骤之一，该步骤需要明确目标用户，并针对该用户需求进行分析与记录，方便验证软件产品是否达到用户预期。下面本文将从用户分析、功能性需求、非功能性需求、用例图和用例描述五个部分对该系统的需求展开多方面的设计说明。

4.2.1 用户分析

基于旋律相似性的音乐比对分析系统的用户主要为音乐比对者，主要的用户角色如表4-1所示，其中包含了不同目标用户对应的详细用户特征，以及可能使用该音乐比对分析系统的具体使用场景。音乐比对者具体可细分为法院法官、音乐创作者、音乐相关专业的老师、AI 作曲研究人员以及音乐爱好者这五类。

表 4-1: 用户角色与使用场景说明

用户角色	用户特征	使用场景
法院法官	具备音乐侵权相关的法律知识，但不具备音乐相关的基础知识，不精通音乐知识	判决音乐侵权案件的过程中提供客观、准确、易懂的比对数据，辅助判决
音乐创作者	具备专业的音乐知识，生活中需要创作音乐，可能是歌手、作曲人、音乐专业的学生等等，完成工作或考试	检测自己完成的音乐是否存在侵权、抄袭嫌疑，或者及时检测是否有抄袭自己作品的现象，及时保护自己的音乐知识产权
音乐相关专业的老师	具备专业的音乐知识，有很多学生，音乐作业太多，没有过多精力分辨学生的作业质量	根据系统提供的客观比对数据自行判断学生是否存在抄袭行为，提升教学效果
AI 作曲研究人员	具备计算机和音乐的专业知识，但是无法判断 AI 技术创作出的作品的音乐质量	检测 AI 作曲质量水平，根据结果对 AI 作曲的结果进一步迭代，提升创作质量
音乐爱好者	爱好音乐，但不精通音乐的专业知识	有怀疑侵权的音乐作品想自行验证，或想了解音乐侵权的著名案件及相关知识

4.2.2 功能性需求分析

功能性需求往往是一个软件产品的核心，其在一定程度上决定了该软件产品未来的应用价值。本文根据项目背景和用户分析的结果，对基于旋律相似性的音乐比对分析系统的功能性需求进行归纳，包括数据交互、音乐侵权案例、音乐比对和用户数据管理等，将以上的主要功能进行细化并总结。该音乐比对分析系统功能性需求列表如表4-2所示。

4.2.3 非功能性需求分析

非功能性需求往往是系统的隐性需求，决定了系统是否能够持续稳定并高效地提供服务，对软件系统同样起到不可或缺的作用。为了证明本文音乐比对分析系统的良好运行，本节将从易使用性、可用性、安全性和可扩展性这四个方面对系统的非功能性需求进行设计与说明。

易使用性。在设计系统界面的交互和流程时，应该从用户角度和系统角度进行分析，尽可能保证系统操作界面友好且简单易懂，符合用户的操作习惯，尽量减少冗余复杂的操作。并且，系统应当快速响应用户的操作请求，减少等待时间。此外，由于该音乐比对分析系统存在涉及法律案件以及个人隐私的可

表 4-2: 功能性需求列表

需求 ID	需求名称	需求描述
RQ1	上传音乐音频文件	系统应该允许用户上传指定格式的音乐音频文件，并根据用户上传的音乐音频文件的具体情况及时为用户反馈上传状态与相应的提示信息，如：格式错误、文件不存在等。
RQ2	查看音乐侵权案例列表	系统应该允许用户查看音乐侵权案例模块，将案例以列表的形式展示，列表中包含音乐侵权案例的基本信息，包括：案例涉及的音乐作品、案例简要说明等，并提供查看案例详情的入口。
RQ3	查看音乐侵权案例详情	系统应该允许用户查看音乐侵权案例的详情，包括案例涉及的音乐名称、案例发生的详细说明、法院审判结果以及通过该音乐比对分析系统生成的比对详情。
RQ4	下载音乐侵权案例详情	系统应该允许用户选择自己感兴趣的音乐侵权案例，查看该案件的详情信息后可以进行下载操作，保存至本地，供用户学习或使用。
RQ5	开始进行音乐比对	系统应该允许用户开启音乐比对功能，在用户将想要比对的音乐音频文件上传成功后，可以通过开始比对按钮来进行比对操作，系统会将两个文件进行比对操作，并将结果返回给用户。
RQ6	查看音乐比对详情	系统应该允许用户对自己上传的音乐音频文件的比对结果进行访问查看，系统将对音乐文件进行多角度的比对，给出整体的比对结果和详细的比对结果。
RQ7	下载比对详情文件	系统应该允许用户对音乐的比对结果进行下载，保存至本地。
RQ8	查看上传和比对记录	系统应该允许用户通过个人中心对自己上传的音乐音频文件和比对结果文件进行访问查看。
RQ9	删除上传和比对记录	系统应该允许用户通过个人中心对针对自己上传的音乐音频文件和比对结果文件进行删除操作。

能性，本系统也将会针对这种情况，提示用户 “数据将会受保护、不会对外泄露” 等友好信息。此外，针对可能出现的操作错误情况，系统需要给出相应的错误提示并指引用户修正。

可用性。系统用户通过电脑 Web 网页对系统进行操作，因此系统应当支持多种常用的操作系统和浏览器。该音乐比对分析系统应运行流畅，需要保持稳定的网络环境，保证用户在 99% 的时间内完全可用。此外，对于音乐时长较长的音乐文件在比对的过程中需要较大的运算量，因此需要避免数据量过大导致的系统延时现象，需要对其采取一定措施。

安全性。系统应该对用户的个人信息数据、上传文件数据、音乐比对数据

进行存储，并保证系统数据的安全，确保数据存储环境稳定安全地运行，以便于根据记录做好数据恢复的工作。

可扩展性。系统应该具备良好的可扩展性和可维护性，避免对系统的模块进行迭代操作时，造成大量重复性的开发工作，导致开发成本与资源的浪费。系统应该对扩展开放，对修改关闭。因此，应该提高系统架构与代码的质量，对系统模块的业务逻辑进行分解和完善，同时也可以提高系统的性能。

4.2.4 系统用例图

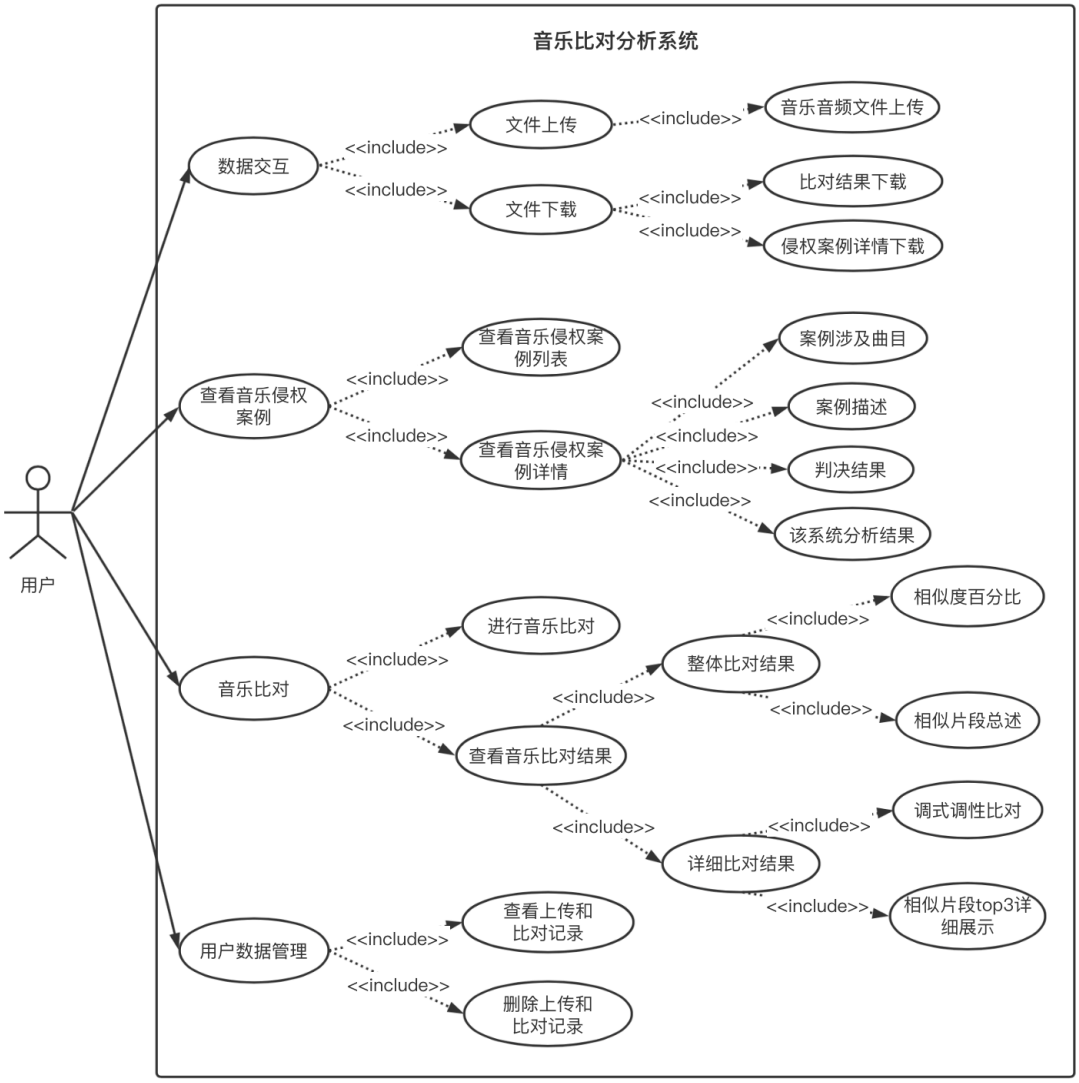


图 4-2: 系统用例图

依据系统功能性需求分析，基于旋律相似性的音乐比对分析系统用例图如图4-2所示，该音乐比对分析系统的主要用例包括数据交互、音乐侵权案例、音乐比对以及用户数据管理。用户使用该音乐比对分析系统，通过完成音乐音频文件的上传，来进行音乐比对，同时用户可以查看以往的音乐侵权案例，系统将以列表和基本信息详情的形式进行展示。此外，用户可以通过个人中心对上传记录以及比对结果记录进行数据管理操作。

4.2.5 系统用例描述

下面针对数据交互、查看音乐侵权案例、音乐比对以及用户数据管理这四大功能进行用例拆分，分别对文件上传、文件下载、查看音乐侵权案例列表、查看音乐侵权案例详情、音乐比对和用户数据管理这 6 个用例进行详细的功能性需求分析说明，包括用例名称、参与者、优先级、用例描述、前置条件、后置条件、正常流程以及扩展流程等信息。该音乐比对分析系统的用例与功能性需求对应关系如表4-3所示。

表 4-3: 系统用例列表

用例 ID	用例名称	包含的需求 ID
UC1	文件上传	RQ1
UC2	文件下载	RQ4、RQ7
UC3	查看音乐侵权案例列表	RQ2
UC4	查看音乐侵权案例详情	RQ3
UC5	音乐比对	RQ5、RQ6
UC6	用户数据管理	RQ8、RQ9

UC1 和 UC2 分别为文件上传和文件下载功能，均属于数据交互功能。数据交互功能是比对结果等数据生成的基础，数据交互模块需要支持固定格式的音乐音频文件的上传，并支持对生成的音乐比对分析结果进行查看与下载操作。系统用户进行文件上传的用例描述如表4-4所示，可以上传的音乐音频文件格式为 MIDI 格式或 MusicXML 格式。音乐音频文件只有通过数据交互上传成功才能进行后续的比对步骤。系统用户使用该音乐比对分析系统进行文件下载的用例描述如表4-5所示，主要针对侵权案例详情信息和比对分析结果进行下载操作，保存至本地，并对下载结果进行返回。

表 4-4: 文件上传用例描述

描述项	具体说明
用例编号	UC1
用例名称	文件上传
参与者	用户
优先级	高
前置条件	用户打开比对分析系统并进入比对界面，http 可提供正常服务
后置条件	文件成功上传，用户可以查看到当前文件的状态
正常流程	1. 用户按照系统要求的格式整理待上传文件； 2. 调用系统的数据上传接口，分别点击上传原作品音频文件以及疑似相似作品音频文件； 3. 系统识别文件是否符合要求，并给出相应状态提示； 4. 系统成功接收上传的文件，将文件写入 MySQL 数据库。
扩展流程	3a. 如有音频文件不存在，提示 “文件不存在” 并让用户重新上传； 3b. 如音频文件格式不符，提示 “格式错误” 并让用户重新上传； 3c. 如音频不是单音轨，提示 “不符合要求” 并让用户重新上传。

表 4-5: 文件下载用例描述

描述项	具体说明
用例编号	UC2
用例名称	文件下载
参与者	用户
优先级	高
前置条件	1. 用户打开比对分析系统，http 可提供正常服务； 2. 系统成功调取音乐侵权案例的详细资料； 3. 系统顺利完成比对任务，并将结果返回。
后置条件	文件成功下载，用户可以查看到当前文件的状态
正常流程	1. 当比对结果数据生成后，用户可以点击下载至本地按钮，保存生成的比对文件； 2. 当用户查看音乐侵权案例详情时，系统显示音乐侵权案例相关信息以及比对详情，用户点击下载按钮，保存侵权案例的相关文件； 3. 系统给出下载的相应提示。
扩展流程	3a. 如文件下载失败，提示 “下载失败” 并让用户重新下载。

UC3 和 UC4 分别为查看音乐侵权案例列表和查看音乐侵权案例详情用例，均属于查看音乐侵权案例功能。音乐侵权案例功能借助音乐侵权的历史中较为著名的音乐侵权案例进行介绍，供用户查看著名音乐侵权案例并且在一定程度上熟悉该音乐比对分析系统的功能。用户查看音乐侵权案例列表的用例描述如表4-6所示，系统将展示音乐侵权案例的基本缩略信息，用户可根据显示的简要案例信息进行选择与查看。用户查看音乐侵权案例详情的用例描述如表4-7所示，详情主要包括案例涉及的原音乐作品（原告）以及疑似侵权的音乐作品（被告）、案例的引发缘由、最终的法院审判结果等等，并且通过本系统对其进行音乐比对分析，并将详细的比对结果进行展示。

表 4-6: 查看音乐侵权案例列表用例描述

描述项	具体说明
用例编号	UC3
用例名称	查看音乐侵权案例列表
参与者	用户
优先级	高
前置条件	1. 用户打开分析系统，http 可提供正常服务； 2. 用户点击选择并进入查看音乐侵权案例界面。
后置条件	用户可成功查看案例列表信息
正常流程	1. 用户打开首页后，点击查看音乐侵权案例按钮，进入该模块； 2. 系统显示音乐侵权案例列表以及查看详情的入口，列表中展示部分缩略信息，部分缩略信息包括案例涉及的曲目名称、简要的案例描述。
扩展流程	无

UC5 为音乐比对功能。音乐比对模块是本音乐比对分析系统的核心模块，供用户直接进行疑似相似音乐的比对，得到客观的音乐比对数据。用户成功上传指定格式的两个音乐文件后，通过音乐比对模型基于旋律音程、节奏、调性调式等旋律特征进行提取并通过字符串匹配算法完成比对工作，最终将多维度的详细客观音乐比对分析数据返回给用户。用户进行音乐比对的用例描述如表4-8所示。经过该音乐比对分析系统得出针对两个音乐的整体比对结果以及详细比对结果报告，其中包括调性调式、相似片段 top3 的详细展示比对、旋律音程相似度百分比、节奏相似度百分比以及详细相似片段位置与长度等信息。

表 4-7: 查看音乐侵权案例详情用例描述

描述项	具体说明
用例编号	UC4
用例名称	查看音乐侵权案例详情
参与者	用户
优先级	高
前置条件	1. 用户打开分析系统，http 可提供正常服务； 2. 用户点击选择并进入查看音乐侵权案例界面。
后置条件	用户可成功查看案例详情
正常流程	1. 用户在音乐侵权案例列表页面选择案例，点击查看音乐侵权案例详情按钮； 2. 系统显示音乐侵权案例的详细信息，详细信息包括涉及的音乐作品名称、案例简要描述、最终判决结果以及经过该音乐比对分析系统比对的详细比对结果； 3. 用户点击下载，系统提示用户将文件下载保存至本地。
扩展流程	无

表 4-8: 音乐比对用例描述

描述项	具体说明
用例编号	UC5
用例名称	音乐比对
参与者	用户
优先级	高
前置条件	1. 用户打开分析系统，http 可提供正常服务； 2. 用户点击选择并进入音乐比对界面且成功上传文件。
后置条件	系统返回针对上传的音乐的详细比对结果
正常流程	1. 用户按照要求的格式成功上传原音乐作品以及疑似相似音乐作品文件，系统显示上传成功的状态； 2. 用户点击开始比对按钮，系统开始进行比对； 3. 比对结束后，系统给出针对上传音乐的比对结果，包括整体比对结果和详细比对结果； 4. 整体比对结果包括上传音乐的旋律音程整体相似度百分比、节奏整体相似度百分比、相似片段位置与长度总述，详细比对结果包括调性调式、旋律音程 top3 和节奏相似片段 top3 的详细五线谱以及音频比对； 5. 将音乐作品信息与比对结果存储至数据库中。
扩展流程	2a. 如文件比对失败，提示 “比对失败” 并让用户重新进行比对。

UC6 为用户数据管理功能。用户数据管理模块供用户在上传音乐音频文件或进行音乐比对后方便对其进行查看、下载、删除等操作。当任务在后台完成上传、分析等任务后，系统会更新当前任务状态，并将结果显示给用户。用户数据管理功能的用例描述如表4-9所示。

表 4-9: 用户数据管理用例描述

描述项	具体说明
用例编号	UC6
用例名称	用户数据管理
参与者	用户
优先级	高
前置条件	1. 用户打开音乐比对分析系统，http 可提供正常服务； 2. 用户点击选择并进入个人中心界面。
后置条件	无
正常流程	1. 用户点击查看个人中心； 2. 系统显示个人中心中的上传记录以及比对记录，并提供查看、下载、删除的操作选项； 3. 用户点击查看详情，可以查看到用户上传的音乐音频记录以及进行音乐比对的记录。
扩展流程	无

4.3 音乐比对分析系统概要设计

4.3.1 整体架构设计

基于旋律相似性的音乐比对分析系统的整体架构设计如图4-3所示，下面将从前端展示交互、服务端业务和数据服务三个角度针对该系统的整体构造进行描述说明。

前端展示交互层，在权衡目前较为主流的前端框架后，该音乐比对分析系统采用了 Vue 前端框架，用户在前端完成操作，前端界面在系统中向后端发送用户对其进行的 Ajax 请求操作，Ajax 在浏览器和 web 服务器之间实现异步数据传输，从而减少网页从服务器请求的信息量。具体通过 Axios 组件处理 Ajax 异步请求，实现延迟加载和局部更新，从而避免用户在发起请求后可能出现页

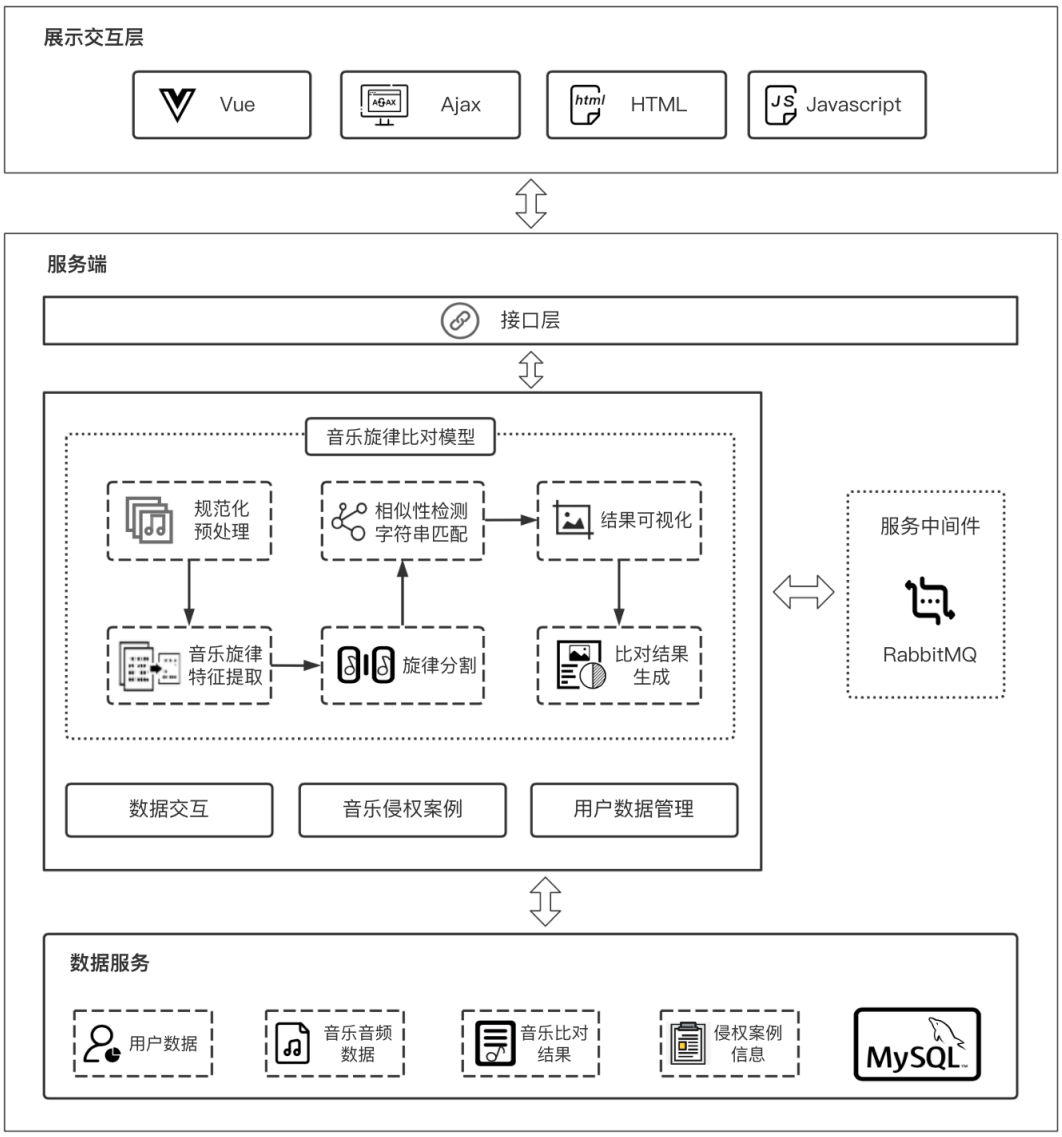


图 4-3: 整体架构图

面卡顿等问题。前端界面获得用户上传指定格式的音乐音频文件后，通过前端框架向后端发送。在后端服务完成音乐比对服务后，将比对结果传输给前端界面，进行展示操作。

后端服务层，使用 Spring Boot 框架进行搭建及开发，服务端部分向外提供 RestfulAPI，提高客户端的便捷性和服务端的可伸缩性，降低客户端与服务端的耦合性。服务端主要负责与系统用户之间的业务交互，包括数据交互、音乐比对、音乐侵权案例查看、用户数据管理。数据交互需要支持用户对文件数据的上传和下载操作，具体包括音乐音频文件的上传、比对结果文件的下载以及

侵权案例详情文件的下载。其中，音乐比对模型是该音乐比对分析系统的核心功能，主要包括旋律预处理模块、旋律特征提取模块、旋律分割模块、相似性比对模块、结果可视化及结果生成模块。其中，旋律特征提取模块内部包括提取旋律的旋律音程信息、节奏信息和调性调式信息，相似性比对模块则主要是基于改进的 Sunday 算法的进行音乐旋律特征的字符串匹配。可视化及结果生成模块主要进行可视化结果展示和具体的比对结果输出，主要包括整体的比对结果和详细的比对结果。此外，也支持用户查看音乐侵权案例和用户数据管理等业务功能。采用以 RabbitMQ 消息队列作为服务中间件，提高该音乐比对分析系统的性能，保证服务的正常执行。

数据服务层，该音乐比对分析系统使用 MySQL 关系型数据库来存储相关信息，信息主要包括用户的基本信息数据、用户的上传音乐音频数据、音乐比对数据以及侵权案例相关数据等，便于用户对系统功能的使用以及对自身数据进行管理操作。

4.3.2 4+1 视图

“4+1” 视图模型是一种使用多个并发视图的软件架构描述模型。从系统的多个涉众角度出发，将这些视图结合起来可以形成完整的系统设计。下面将分别从逻辑视图、开发视图、进程视图以及物理视图对本文的音乐比对分析系统进行多角度的设计与说明。

（1）逻辑视图

逻辑视角描述了系统的抽象结构，基于旋律相似性的音乐比对分析系统的逻辑视图如图 4-4所示。采用分层混合风格展开描述该音乐比对分析系统的逻辑设计，分别从展示层（UI 层）、业务逻辑层（Logic 层）、数据层（Data 层）三个层面进行设计与说明。其中，展示层可以实现与用户的交互工作，主要为用户使用该音乐比对分析系统时的系统界面，包括音乐比对分析系统的主界面、上传音乐音频文件界面、查看音乐侵权案例界面、比对详情界面、个人中心界面等等。业务逻辑层可以实现该音乐比对分析系统的相关业务逻辑，主要包括音乐旋律比对服务、数据交互服务、资源查看与管理服务等等。数据层可以实现对相关数据的存储与访问，主要包括用户数据、音乐侵权案例资源数据、音乐比对结果数据等等。

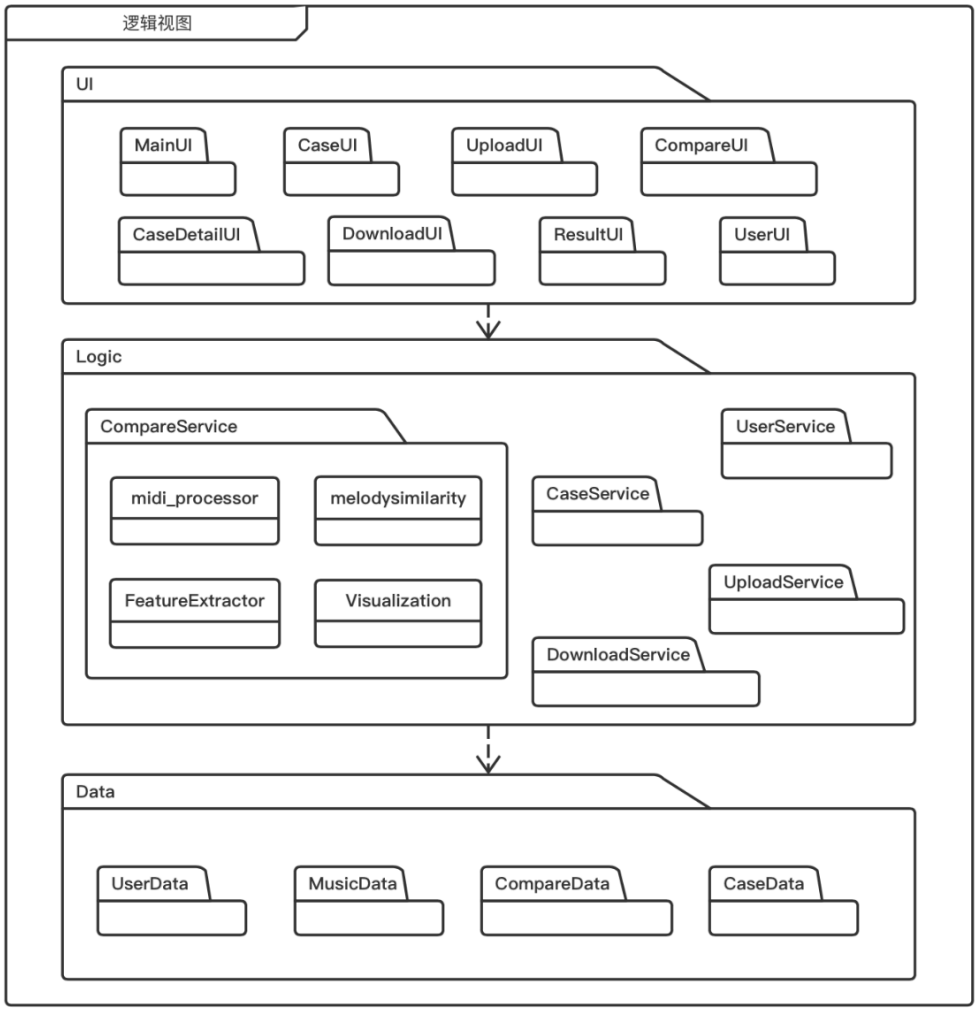


图 4-4: 逻辑视图

（2）开发视图

开发视图描述了开发系统需要完成的任务，即为面向开发者的视图。该音乐比对分析系统的开发视图如图4-5所示，系统可以分为前端 UI、服务业务逻辑和技术服务三部分。其中，前端 UI 部分主要由为页面、组件、静态资源部分组成。组件包 (components) 包含前端页面组件，静态资源包 (assets) 包括图片、音乐音频文件等资源，CSS 目录存放的是系统前端的样式文件。服务逻辑主要根据系统模块以及其实现的业务逻辑功能划分，主要包括 Controller 包、Logic 包、Service 包、Util 包和 Model 包。Controller 包负责接收系统前端的请求并进行路由分发；Logic 包负责处理音乐比对分析系统的业务逻辑并对 Controller 层提供业务逻辑支撑；Service 包负责实现具体的业务逻辑，主要包括音乐旋律

比对、侵权案例查看、数据交互、比对结果可视化等业务；Model 包包含了音乐旋律相似性比对的模型，主要包括规范化预处理、特征提取、旋律分割、旋律相似性比对、可视化等模块；Util 包包含音频识别、生成图像等工具类。技术服务部分主要是系统使用的基础服务，包括 Docker 技术、RabbitMQ 技术、MySQL 技术等等。

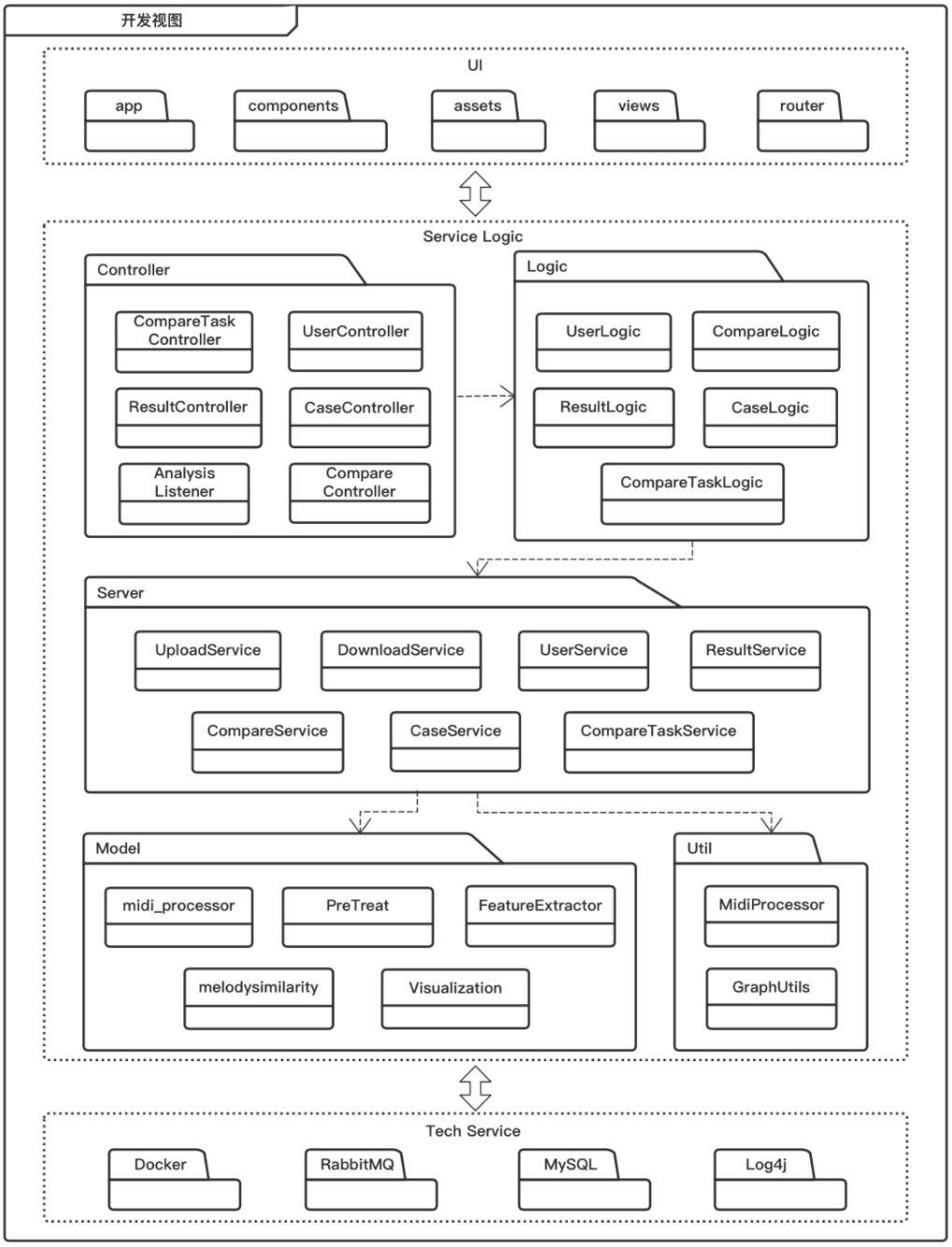


图 4-5: 开发视图

（3）进程视图

进程视图又叫处理视图，表述的是系统内部相关进程与线程实体及其通信过程，从动态的角度对系统进行描述说明。下面将从用户视角和系统视角两方面分别对系统的进程处理进行设计说明。

用户使用该音乐比对分析系统的流程图如图4-6所示，从用户视角对音乐比对分析系统进行整体流程的梳理。首先，在用户视角中，可以自行选择进行音乐比对或者查看音乐侵权案例功能。若用户进行音乐比对，选择并上传规定格式的音乐文件后，即可对其进行音乐比对，得到音乐比对详细结果后可选择将文件下载并保存至本地；若用户查看音乐侵权案例的分析，可以点击查看以往关于音乐侵权著名案件的具体资料，包括原音乐作品和疑似侵权音乐作品、侵权案件相关描述、案件最终审判结果以及通过本系统对其进行音乐比对的详细信息。此外，用户可进入用户中心对个人数据进行管理操作，完成对音乐音频文件以及比对记录的查看、删除、下载等操作。

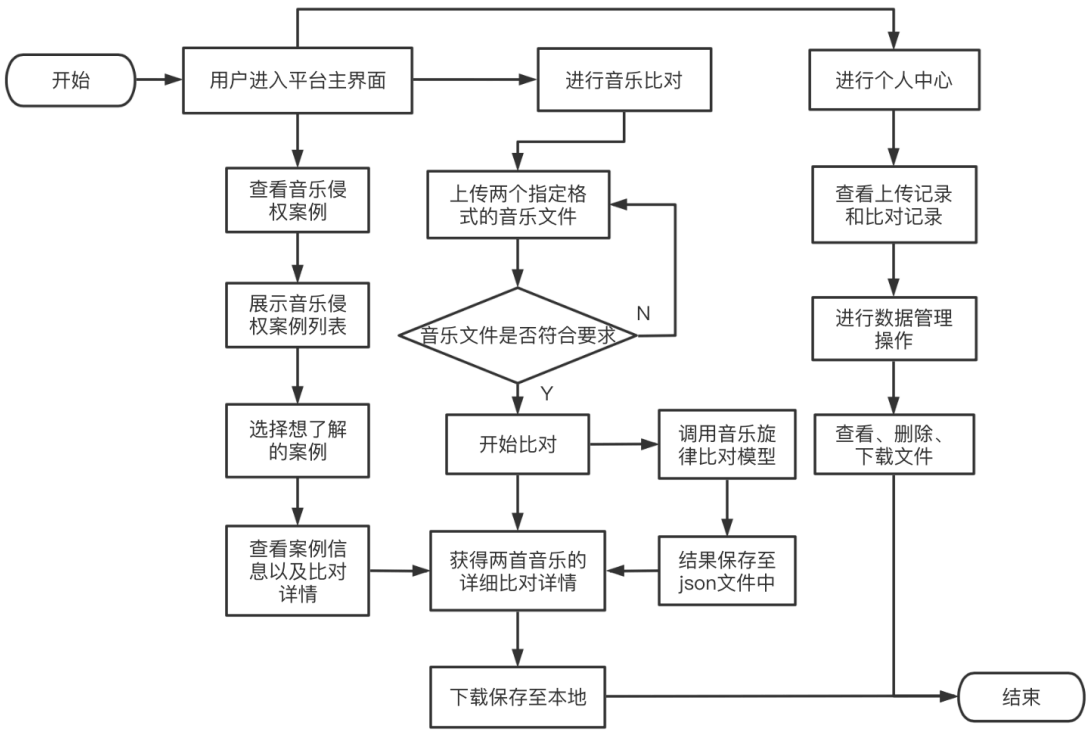


图 4-6: 用户使用流程图

从系统视角进行操作的进程视图如图4-7所示。进程视图主要面向系统设计和集成人员，主要描述系统进程和线程等主要部分之间的通信过程。执行服

务进程会通过消息队列发送不同的操作消息，进而实现音乐旋律信息比对分析服务的调用。音乐比对服务主进程响应相应的 RabbitMQ 消息，调用自身的音乐旋律比对对应的具体方法，对用户成功上传的原音乐作品和疑似相似音乐作品针对其多个旋律特征信息进行多维度的比对分析，最终得出音乐旋律比对结果，并将结果数据存入 MySQL 数据库中，以供后续展示和查询需要。在用户发出查看音乐侵权案例请求时，系统会调取存储在数据库中的音乐侵权相关信息展示在前端页面中，供用户查看。

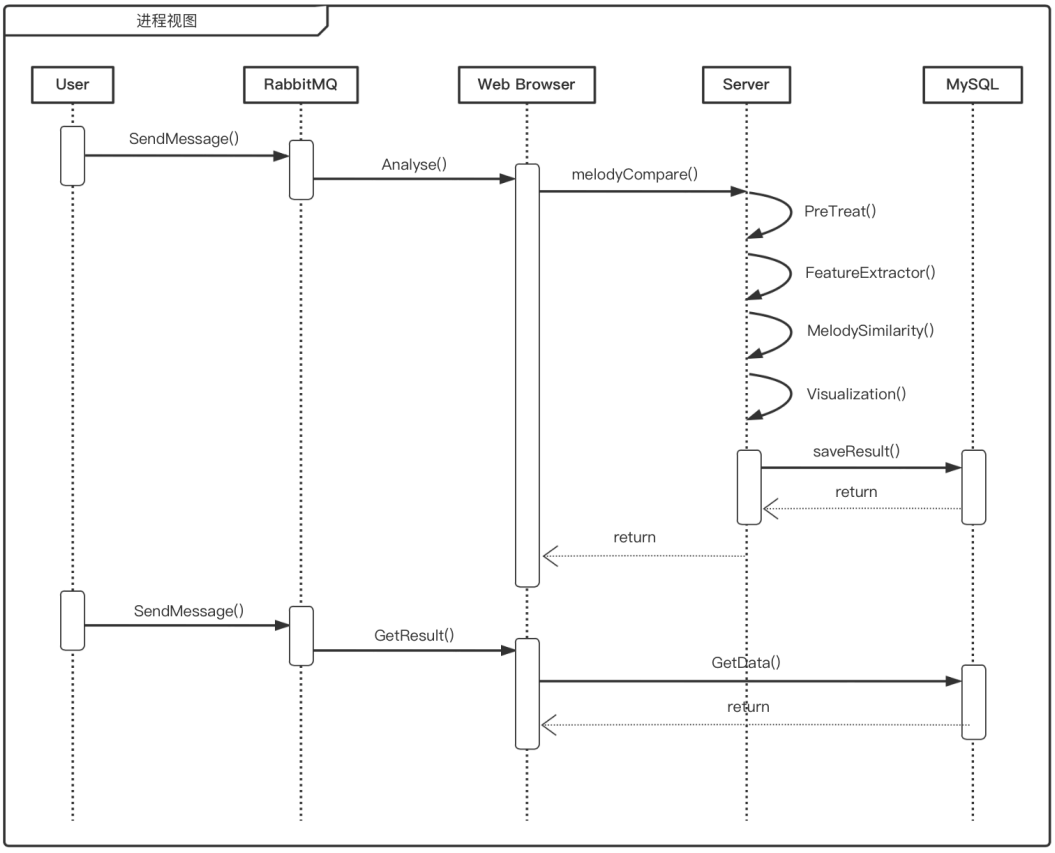


图 4-7: 进程视图

（4）物理视图

物理视图主要面向运维人员，表述了系统物理节点之间的部署设计和通信情况。本系统的全局部署视图如图4-8所示，用户通过电脑浏览器向 Web 服务器发送 HTTP 请求，该音乐比对分析系统的前端服务器成功接收请求后，发送 HTTP 请求申请访问系统后端。音乐比对分析任务通过 RabbitMQ 消息队列服务器，建立异步流水线调用机制。客户端发送请求后，先在 RabbitMQ 消息队

列中间件进行排队流程，并按照先到先服务的原则将任务分发给后端服务器。服务端分为业务请求和底层数据存储两部分，业务请求负责请求响应及业务逻辑计算，主要包括音乐比对、音乐侵权案例以及用户数据管理等业务，并使用 Docker 容器技术针对音乐比对进行容器化处理，利用音乐比对模型对上传的音乐音频文件进行旋律比对分析并返回比对结果。数据存储服务器负责音乐侵权案例、用户上传以及比对资料、音乐比对任务等资源的存储。

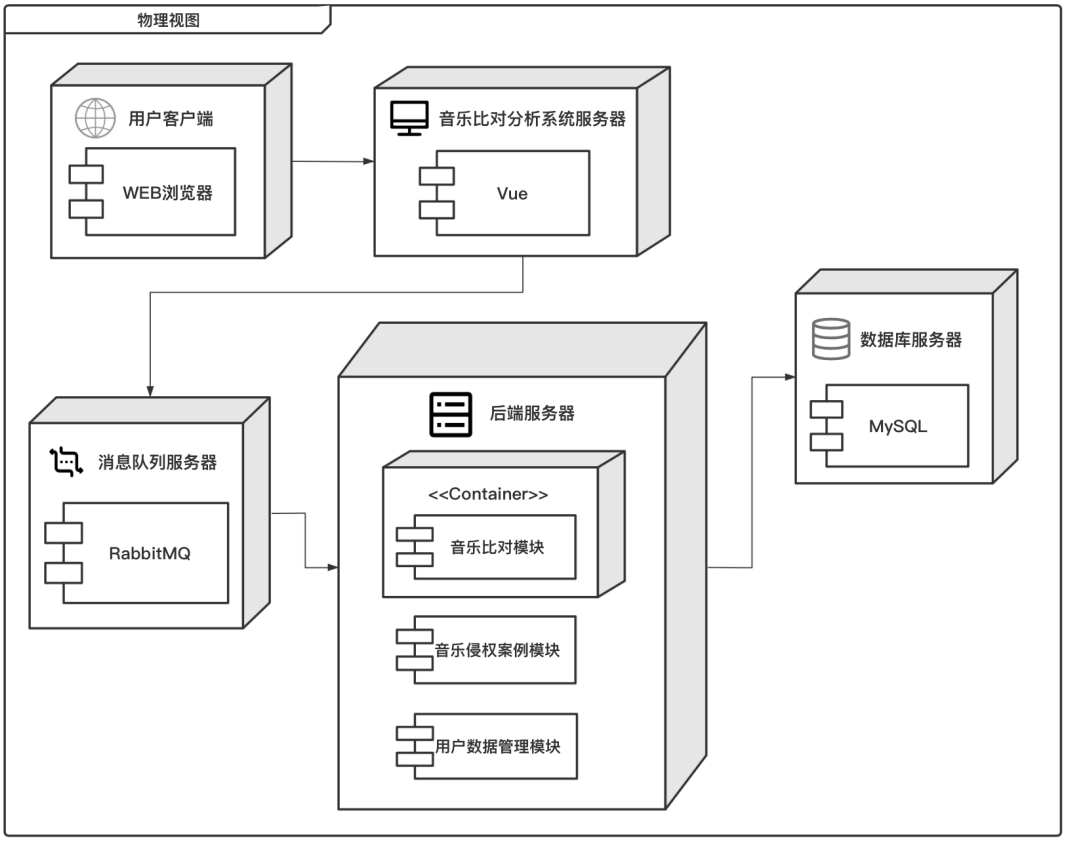


图 4-8: 物理视图

4.4 音乐比对分析系统数据库设计

4.4.1 E-R 图

该音乐比对分析系统选用关系型数据库 MySQL 来对系统中需要存储的数据进行存储工作。在设计数据库时，需要真正把握系统的在具体应用环境下的业务流程，以及在该业务流程中所涉及的各个客观实体对象以及这些实体对象

之间的对应关系以及发生的活动关系。ER 模型可以帮助人们将现实世界中的数据内容映射成模型，根据该模型转化为具体的数据库的表设计，辅助相关人员更加条理化和系统化地设计并实现系统。对存储在数据库中的业务对象分别进行分析和枚举，为了表示各实体之间的关系，该音乐比对分析系统的局部 E-R 图如图 4-9所示。

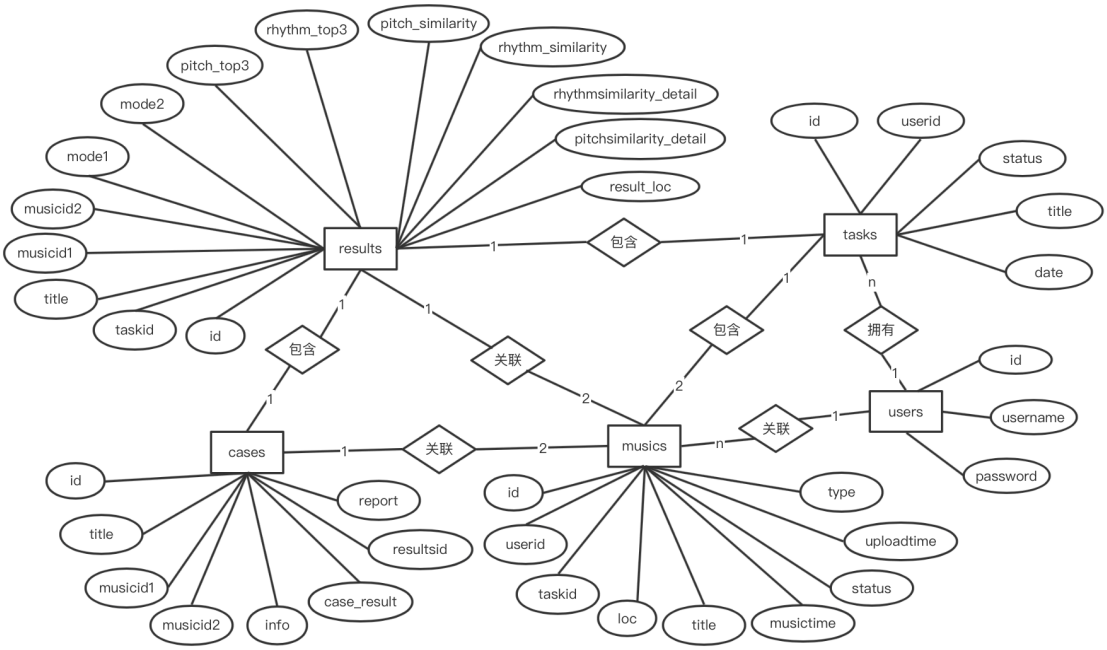


图 4-9: 系统 E-R 图

该音乐比对分析系统的设计围绕音乐比对文件和比对结果展开，数据库中主要包括用户上传以及侵权案例包含的音乐文件信息 (musics)、音乐侵权案例信息 (cases)、音乐比对结果信息 (results)、比对任务信息 (tasks) 以及用户个人中心信息 (users)。

其中，musics 表中包含音乐音频文件的基本信息，cases 表中包含音乐侵权案例的基本信息，results 表包含音乐比对结果的基本信息，tasks 表包含音乐比对任务的基本信息，user 表中包含用户的基本信息。user 表和 tasks 表是 1:n 的拥有关系，一个用户可以创建拥有多个音乐比对任务。tasks 表和 musics 表为 1:2 的包含关系，一个音乐比对任务中包含着两个需要进行比对的音乐文件。tasks 表和 results 表为 1:1 的包含关系，一个音乐比对任务中包含着一个比对结果。musics 表和 results 表为 2:1 的关联关系，两个音乐音频文件对应着一个音乐比对结果。cases 表和 musics 表为 1:2 的关联关系，一个音乐侵权案例包含着

原音乐作品和疑似相似音乐作品两个音乐文件。cases 表和 results 表为 1:1 的包含关系，音乐侵权案例除了对自身案例的相关描述之外，还包含经过该音乐比对分析系统生成的详细的音乐比对结果。

4.4.2 数据库表设计

根据该音乐比对分析系统的功能需求和系统 E-R 图，分别针对音乐音频文件信息 (musics)、音乐侵权案例信息 (cases)、音乐比对结果信息 (results)、比对任务信息 (tasks) 以及用户个人中心信息 (users) 在 MySQL 数据库中构建的数据表进行描述：

（1）音乐音频文件信息表 (musics)

音乐音频文件信息表存储了音乐音频文件相关的所有结构化信息，包括系统信息和内容信息两部分，上传文件后数据库会存储音乐文件唯一标识 id，音乐文件的格式 type，音乐文件在服务器中的存储地址 loc，上传时间 uploadtime，上传时的文件名称 title，上传音乐文件的用户标识 userid，音乐文件时长 musictime，音乐文件的状态 status 以及存储地址 loc。音乐音频文件信息表的主要字段以及相关信息如表 4-10所示。

表 4-10: 音乐音频文件信息表

字段	类型	主键	允许为空	说明
id	int	是	否	音乐文件编号
userid	int	否	否	上传文件的用户标识
taskid	int	否	是	比对任务编号
title	varchar	否	否	音乐文件名称
type	varchar	否	否	音乐文件格式
uploadtime	datetime	否	否	音乐文件上传时间
musictime	time	否	否	音乐文件时长
status	varchar	否	否	音乐文件当前状态
loc	varchar	否	否	音乐文件存储地址

（2）比对结果信息表 (results)

比对结果信息表中存储了音乐侵权案件的相关信息，包括音乐比对结果唯一标识 id，上传文件名称 title，原音乐作品编号 musicid1，疑似相似音乐作品

编号 musicid2，原作品与疑似相似作品的调式调性信息 mode1、mode2，相似片段 top3，旋律音程和节奏的整体相似度百分比以及详细的相似片段位置和时长。比对结果信息表的主要字段以及相关信息如表 4-11所示。

表 4-11: 比对结果信息表

字段	类型	主键	允许为空	说明
id	int	是	否	比对结果编号
taskid	int	否	是	比对任务编号
title	varchar	否	否	比对结果文件名称
musicid1	int	否	否	原音乐作品的编号
musicid2	int	否	否	疑似相似音乐的编号
mode1	varchar	否	否	原音乐作品调性调式
mode2	varchar	否	否	疑似相似音乐作品调性调式
pitch_top3	varchar	否	是	旋律音程相似片段 top3
rhythm_top3	varchar	否	是	节奏相似片段 top3
pitch_similarity	double	否	否	旋律音程整体相似度
rhythm_similarity	double	否	否	节奏整体相似度
pitchsimilarity_detail	text	否	是	旋律音程详细相似片段陈列
rhythmsimilarity_detail	text	否	是	节奏详细相似片段陈列
report_loc	varchar	否	否	比对结果存储地址

（3）音乐侵权案例信息表 (cases)

音乐侵权案例信息表中存储了音乐侵权案例的相关信息，包括音乐侵权案例唯一标识 id，原音乐作品编号 musicid1，疑似相似音乐作品编号 musicid2，关于该案例的相关说明 info，以及最终法院的审判结果 case_result，以及利用本系统对其进行音乐比对生成的详细比对结果 id 和 report。音乐侵权案例信息表的主要字段以及相关信息如表 4-12所示。

（4）比对任务表 (tasks)

用户上传音乐文件后，创建音乐比对任务，保存到比对任务表中。其属性包含音乐比对任务的唯一标识 id，创建比对任务的用户 userid，比对任务标题 title，用户创建比对任务的时间 date 以及该音乐比对任务当前的比对状态 status。音乐比对任务表的主要字段以及相关信息如表 4-13所示。

表 4-12: 音乐侵权案例信息表

字段	类型	主键	允许为空	说明
id	int	是	否	案例标识
title	varchar	否	否	比对结果文件名称
musicid1	int	否	否	原音乐作品的编号
musicid2	int	否	否	疑似相似音乐作品的编号
info	text	否	否	音乐侵权案例相关说明
case_result	text	否	否	审判结果
resultsid	int	否	否	对应的音乐比对结果编号
report	text	否	否	使用该系统的比对结果报告

表 4-13: 比对任务信息表

字段	类型	主键	允许为空	说明
id	int	是	否	比对任务标识
userid	int	否	否	创建该任务的用户 id
title	varchar	否	否	比对结果文件名称
status	varchar	否	否	音乐比对状态
date	datetime	否	否	音乐比对任务创建时间

（5）用户表 (users)

用户表主要记录登录该音乐比对分析系统的用户的基本信息，主要包括用户的唯一标识 id、用户登录需要的用户名 username 以及密码 password。用户信息表的主要字段以及相关信息如表 4-14所示。

表 4-14: 用户信息表

字段	类型	主键	允许为空	说明
id	int	是	否	用户标识
username	varchar	否	否	用户名
password	varchar	否	否	用户密码

4.5 音乐比对分析系统详细设计与实现

本节对该音乐比对分析系统的主要模块进行了详细设计与实现，包括数据交互模块、音乐旋律特征提取模块、音乐旋律相似性比对模块以及用户数据管理模块，其中音乐旋律特征提取和相似性比对为该系统中最核心的功能模块。

4.5.1 数据交互模块

（1）数据交互模块概述

数据交互模块主要实现音乐音频文件以及结果数据的交互，该模块连接用户端、音乐比对分析系统以及 MySQL 数据库系统，主要负责实现音乐音频文件的上传、音乐音频文件的存储、音乐比对分析数据的展示、音乐比对分析结果的下载以及音乐侵权案例报告的下载等功能。

（2）数据交互模块设计与实现

文件上传和文件下载的数据交互流程如图4-10所示。根据该音乐比对分析系统的设计要求，对于用户上传的音乐音频文件主要针对以下三种情况进行判断，分别判断文件是否存在、文件格式是否为 MIDI 文件格式或 MusicXML 文件格式以及是否为单声部音乐文件。

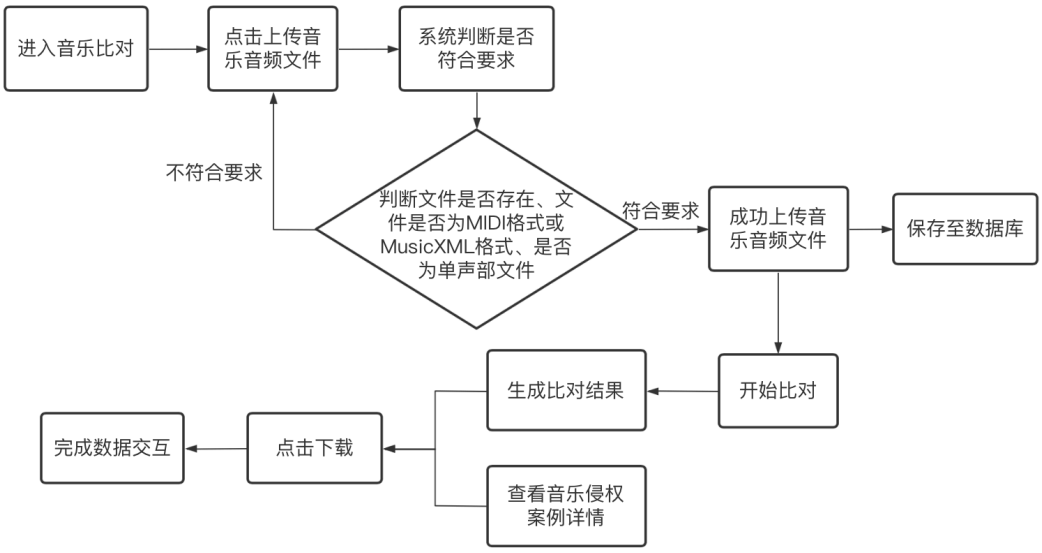


图 4-10: 数据交互流程

数据交互的实现逻辑如图4-11所示。其中，UploadFile 组件通过用户电脑浏览器的 HTTPRequest 对象获取用户上传的音乐音频文件，校验上传音乐音频文件是否与系统的文件要求相符，并返回其对应的结果。在多种文件数据进行交互的情况下，将信息传输至 DataService 组件中，并存储至 MySQL 数据库中。DownloadFile 组件获取所需下载文件在数据库中存储的键值，传入 DataService 组件并获取相应文件在 MySQL 数据库中的存储位置，将文件传输至用户浏览器。DataService 组件是 Web 与 MySQL 数据库系统传输数据和文件的传输介质，负责将用户传输的文件传递给 MySQL 数据库，也负责将文件传递到业务逻辑代码端。在系统生成比对分析结果时，DataService 组件负责将生成的 json 结果文件传递给前端，前端将内容按要求展示在界面中。FileInteraction 组件为系统与数据库系统交互的接口。使用 Docker 镜像，上传文件存于/melody-similarity/upload 中，识别结果存于/melody-similarity/results 中，日志存于/melody-similarity/log 中。

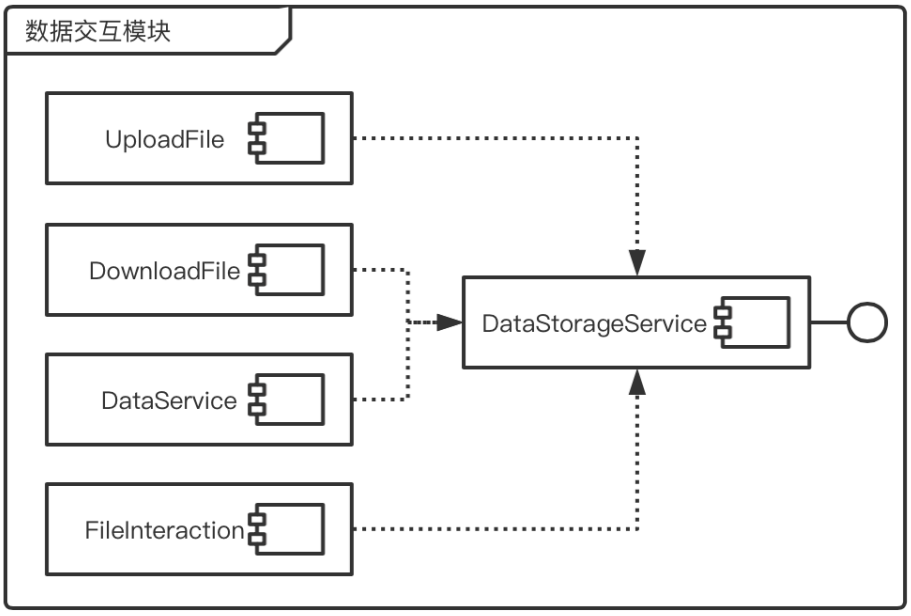


图 4-11: 数据交互实现逻辑

音乐音频文件上传的界面截图如图4-12所示，用户进入到音乐比对页面后，即可进入音乐音频文件上传页面，支持点击上传和拖拽上传两种方式。点击相位置，浏览本地文件目录选择原音乐作品和疑似相似音乐作品文件进行上传。此外，针对上传文件的安全性做了信息提示说明，保护用户上传文件的安全。上传成功后，交由后端完成音乐音频文件存储、音乐比对分析等工作。

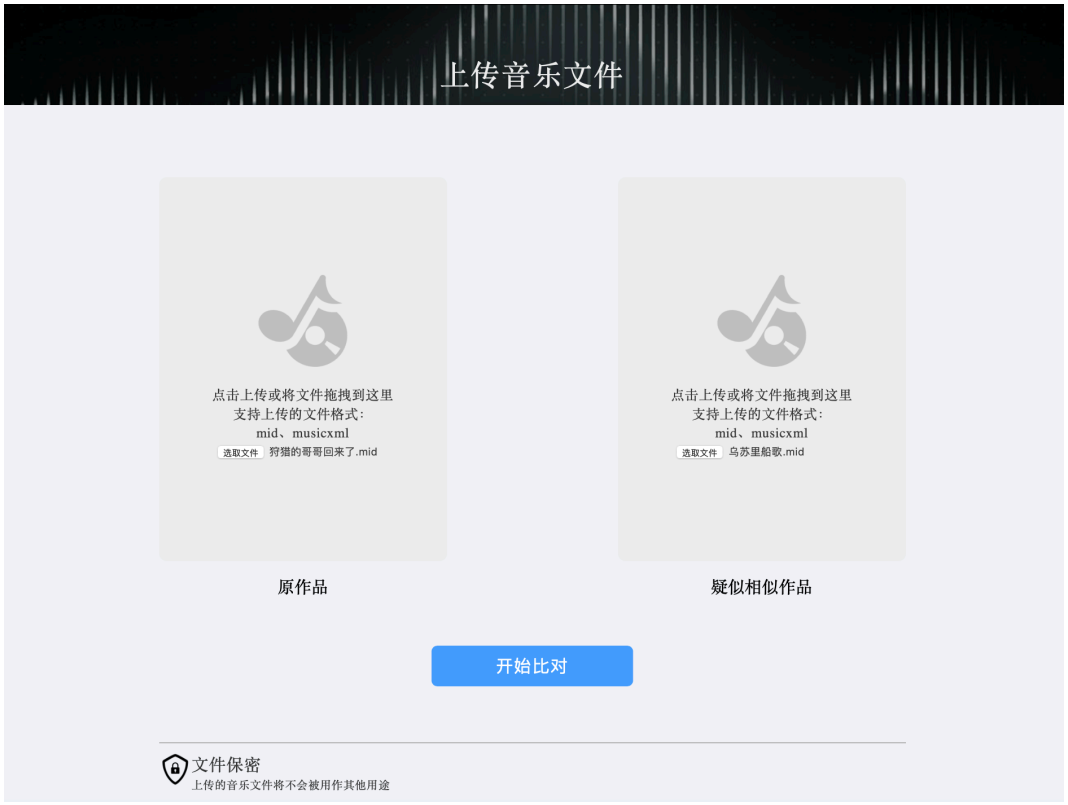


图 4-12: 音乐音频文件上传界面图

4.5.2 音乐旋律特征提取模块

（1）音乐旋律特征提取模块概述

音乐旋律特征提取模块在用户成功上传音乐音频文件后，在进行音乐比对分析任务的过程中需要对其进行音乐旋律多个特征的提取操作，针对音乐音频文件的旋律音程、节奏以及调性调式信息进行提取，以便后续完成音乐的相似性比对操作。

（2）音乐旋律特征提取模块设计与实现

音乐旋律特征提取的核心类图如图 4-13所示。其中，类 FeatureExtractor 是旋律特征提取的核心类，将音乐音频文件 Music 和提取过程 action 作为参数。方法 featureExtractor() 将会根据需要逐个调用各个方法。根据前文的算法设计说明，对旋律特征提取前会对音乐文件进行规范化预处理 (Pretreat) 操作，其中包括文件格式转换 (Convertor) 操作以及规范化 (Normalization) 操作。然后，再

对规范化预处理后的音乐音频文件进行特征提取 (MidiProcessor) 操作，主要提取音乐音频的音高、音符、乐谱、音符时值、调性、调式等基本特征信息，并将其保存在 features 对象中，方便后续对旋律特征信息的操作。

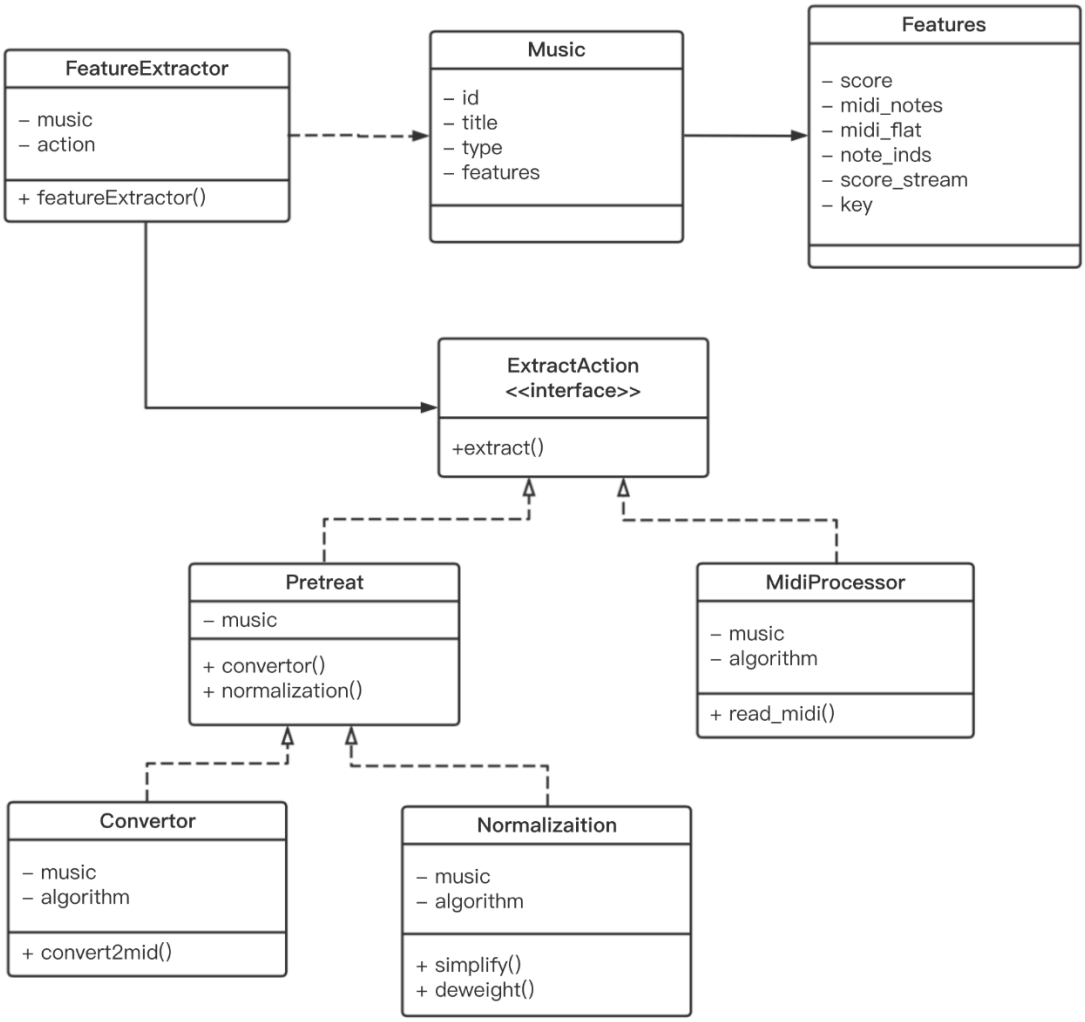


图 4–13: 旋律特征提取类图

针对规范化预处理后的音乐音频文件进行旋律特征提取的操作，根据以上 MIDI 文件的旋律特征表示，来分别提取旋律音高、音符时值以及调性调式的特征信息。其中，旋律音程信息由音高表示，节奏信息由音符时值表示。通过 read_midi() 方法来提取出相应特征信息序列。音乐音频旋律特征提取的核心代码如图4–14所示。

```
def read_midi(self, midi_path):
    # 转换成 MIDI 格式的音乐文件
    midi= music21.converter.parse(midi_path)
    # 省略提取 midiflat、midi_notes、note_inds、score_stream 信息

    # 提取旋律音程和节奏特征信息
    pitches = [int(n.pitch.midi) for n in midi_notes if n.duration.quarterlength != 0]
    rhythms = [int(4 * n.duration.quarterlength) for n in midi_notes if n.duration.quarterlength !=0]
    res = np.array([pitches, rhythms])

    # 提取调性调式信息
    key = score_stream.analyze('key')
```

图 4-14: 旋律特征提取关键代码

4.5.3 音乐旋律相似性比对模块

（1）音乐旋律相似性比对模块概述

该音乐比对分析系统在完成音乐旋律特征提取后，将根据提取到的旋律特征信息进行相似性匹配比对分析，首先对音乐音频旋律特征进行多尺度分割，通过字符串精准匹配算法完成旋律特征序列之间的匹配，得出具体相似的旋律片段起始点和长度信息，并计算整体的相似度，最终将比对分析结果进行多维度展示。

（2）音乐旋律相似性比对模块设计与实现

音乐旋律相似性比对的核心类图如图4-15所示。其中，类 MelodySimilarity 是旋律相似性比对的核心类，将音乐音频文件 Music、旋律特征 Features 和比对分析过程 action 作为参数。方法 melodysimilarity() 将会根据需要逐个调用各个方法。前文中提到会先对音乐旋律进行多尺度分割 (Muti-Segmentation) 操作，再进行旋律音程和节奏两方面的相似性比对 (MelodyCompare) 操作，其间需要通过时值调节器 (LengthRegulator) 来根据音符时值对旋律进行复原操作，最终分析得出旋律音程相似度和具体的相似信息 (PitchSimilarity)、节奏相似度和具体的相似信息 (RhythmSimilarity)，通过可视化 (Visualization) 操作将相似信息进行展示，并将其保存在比对结果 result 对象中，方便后续对两个音乐音频文件的旋律相似性比对结果进行展示。

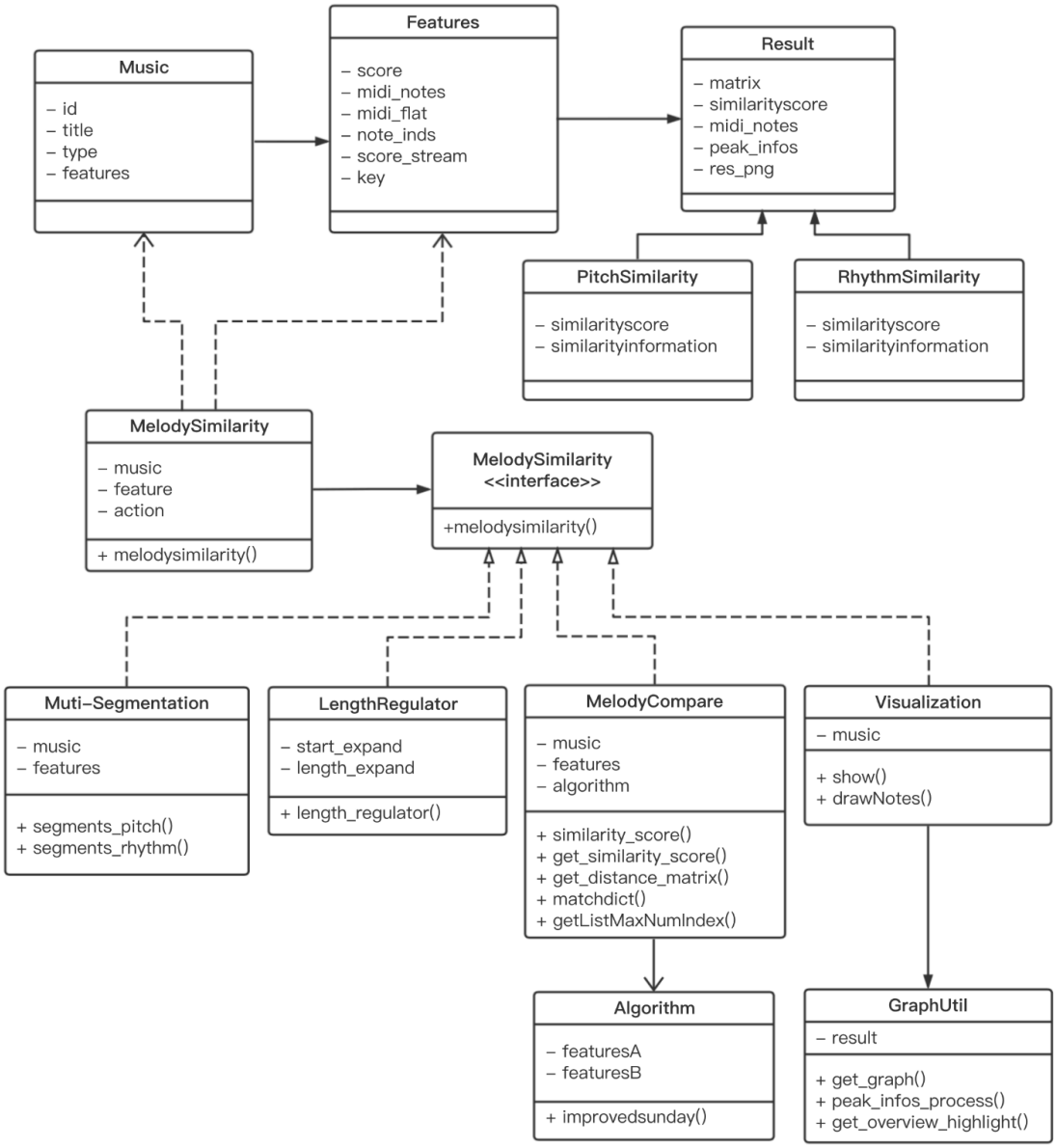


图 4-15: 旋律相似性比对类图

在对旋律特征信息通过 segments_pitch() 方法和 segments_rhythm() 方法进行多尺度分割后，使用 Sunday 改进算法，本文中的音乐旋律进行精准相似性匹配的实现如算法 4.1所示。本文使用的字符串精准匹配算法在 Sunday 匹配算法的基础上进行改进，即在每一次进行 Sunday 算法匹配前，都先找出模式串 list2 中的最后一个字符对应的 list1 中的字符，看其是否存在在模式串中，若存在，则开始使用 Sunday 算法进行匹配。若不存在，则直接移动模式串 list2 的长度距离，即 len2。按照上述步骤重复进行，直到匹配结束。利用该精准匹配算法来得出音乐音频旋律特征序列中详细的相似片段。

算法 4.1 改进的 Sunday 算法匹配

Input:
歌曲的旋律特征序列 *list*1 和 *list*2
Output:
匹配结果 *pAppear*

```
1: len1, len2 ← len(list1),len(list2)
2: pAppear ← []
3: moveDict ← self.matchDict(list2)
4: indexStr1 ← 0
5: for indexStr1 in (0,len1-len2+1) do
6:     indexStr2 ← 0
7:     if indexStr2 ==len2 then
8:         pAppear.append(indexStr1)
9:         indexStr1 += len2
10:        continue
11:    end if
12:    // 判断 list2 的尾字符对应的 list1 的字符是否出现在 list2 中
13:    if matchDict[list2[indexStr1 + len2 - 1] <len2 then
14:        indexStr1 ← matchDict[list2[indexStr1 + len2 - 1]
15:        // Sunday 算法匹配
16:        if list1[indexStr1+len2] not in moveDict.keys() then
17:            indexStr1 += len2+1
18:        else
19:            indexStr1 += moveDict[list1[indexStr1 + len2]]
20:        end if
21:    else
22:        直接进行 Sunday 算法匹配过程
23:    end if
24: end for
25: return pAppear
```

根据多尺度分割以及特征匹配后会得到匹配堆栈图，旋律音程和节奏信息分别用橙色和绿色表示。分别对旋律音程和节奏进行顶点搜索，通过遍历寻找到堆栈图中所有相似的波峰位置，以及波峰位置所对应的长度，并将其进行存储，从而得到两个音乐音频文件中旋律音程和节奏相似片段的具体位置和长度。

通过时值调节器模块，以音符时值作为权重将匹配的旋律音程序列进行扩展，可以求得扩展后的音程序列子串匹配顶点序列，并根据其计算整体相似度百分比。相似度数值计算的实现如算法 4.2所示。再结合匹配堆栈的顶点信息进

行旋律音程整体相似度和整体节奏相似度的计算。

算法 4.2 相似度数值计算

Input:

顶点 *peaks*

音乐文件 *song_a*，音乐文件 *song_b*

Output:

相似性数值 *similarity_score*

```
// 完成时值调节
sim_highlight_a.append(self.length_regulator(song_a , pair))
sim_highlight_b.append([self.length_regulator(song_b , i) for i in value])
total_duration ← sum(song[1])
similarity_array ← [0]*total_duration
for peak in peaks do
    similarity_array[peak[0] : peak[0] +peak[1]] ← [1]*peak[1]
end for
similarity_duration ← sum(similarity_array)
// 计算旋律相似度数值
similarity_score ← similarity_duration/total_duration
return similarity_score
```

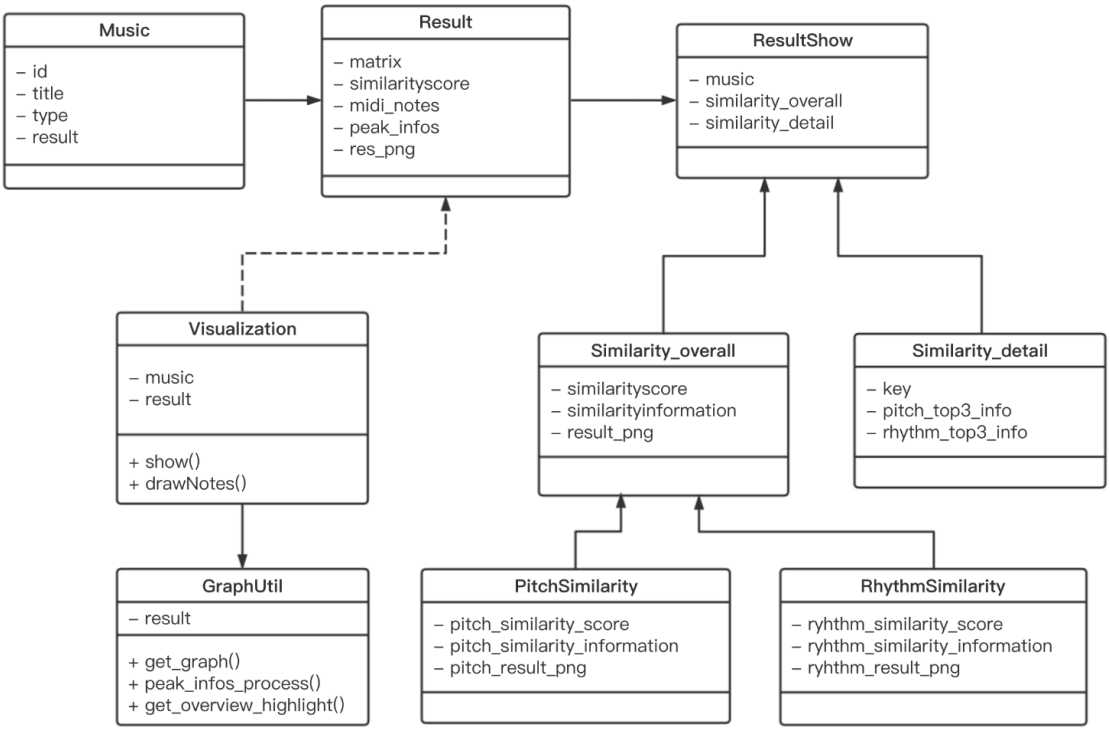


图 4-16: 旋律比对结果类图

音乐旋律比对结果展示的核心类图如图4-16所示。完成音乐旋律比对分析后得出的结果通过 GraphUtil 工具类进行多层次多维度的可视化 (Visualization) 展示，比对结果 (ResultShow) 主要包括涉及的音乐音频文件、整体比对结果 (Similarity_overall) 和详细比对结果 (Similarity_detail)。整体比对结果包括整体旋律音程和节奏的相似性、相似性热力图以及详细的相似片段位置与时长；详细比对结果包括调性调式、相似片段 top3 的音乐曲谱片段并且提供相应音频。将比对分析结果写入 json 文件中，前端通过接口获取提取生成的 json 文件中的相关内容，系统根据文件键值找到 json 文件，将 json 文件的结果传递到比对结果页面并进行展示。

在音乐比对结果可视化的过程中，通过 get_graph() 方法来构造可视化的图片，通过导入 Seaborn 包来实现数据的可视化，Seaborn 是基于 matplotlib 的图形可视化 python 包。本系统中的音乐比对功能主要针对两个音乐音频文件的旋律音程和节奏两个特征信息生成相似性比对图片。比对分析结果可视化的核心代码如图4-17所示。

```
def get_graph(D, pitch_or_rhythm, result_root):
    # generate a graph
    mask = np.zeros_like(D_mss_inversed)

    for i in range(len(D_mss_inversed)):
        for j in range(len(D_mss_inversed[i])):
            if D_mss_inversed[i][j]<0:
                mask[i][j] = True
                D_mss_inversed[i][j] = 0
    # 旋律音程相似性图片
    if pitch_or_rhythm == 1:
        plt.figure()
        # 规定旋律音程相似性图片样式
        plt.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['axes.unicode_minus'] = False
        plt.title('音程相似度堆栈图',y = 1.05,size = 15)
        cmap = sns.light_palette("orange")    # , reverse=True)
        # 省略规定旋律音程相似性图片横纵轴样式及名称
        ax.invert_yaxis()
        plt.savefig(result_root + 'pitch_similarity.png')
    # 节奏相似性图片
    elif pitch_or_rhythm == 2:
        plt.figure()
        # 省略节奏相似性图片样式，与旋律音程相似性图片样式类似
        plt.savefig(result_root + 'rhythm_similarity.png')

    return D_mss
```

图 4-17: 比对结果可视化核心代码

4.5.4 用户数据管理模块

（1）用户数据管理模块概述

用户数据管理模块为用户提供了上传记录及音乐比对分析结果的管理功能。主要内容包括：(1) 用户可以在列表中查看所有已上传的音乐音频文件的基本信息；(2) 用户可查看当前状态，音乐音频文件共有四种可能状态，分别为尚未比对、等待比对、比对中和比对完成，用户可以刷新页面来更新并查看当前音乐比对的状态；(3) 用户可对上传的文件记录以及比对详细数据进行查看、下载以及删除操作。

（2）用户数据管理模块设计与实现

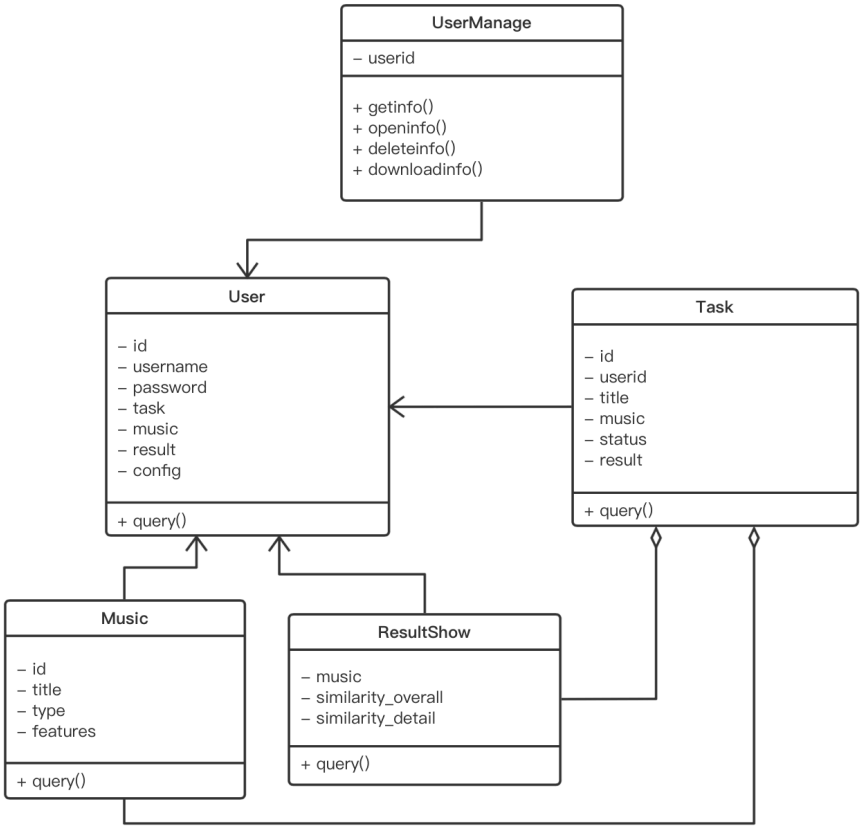


图 4-18: 用户数据管理类图

用户数据管理的核心类图如图4-18所示。用户数据管理涉及 User、UserManage、Music、Result 及 Task 类。当前用户登录时，系统会根据用户 id 获

得用户对象，用户除了包含当前用户的 id、username、password 等基本信息外，还包括用户上传的音乐音频数据 (music)，任务列表 (task)，比对结果文件 (result) 等属性。音乐比对分析系统获得当前用户所有的用户管理数据对象后，用户即可在这些数据的基础上完成对数据的管理操作。用户数据管理可以支持数据的离线生成，用户只需要在数据生成结束后进入个人中心对生成文件进行查看、下载等操作。若用户选择删除文件，系统将删除 MySQL 数据库中存储的文件信息。图 4-19为用户管理个人音乐比对数据的界面截图。用户可查看到自己的音乐比对记录，包括原作品文件、疑似相似作品文件、上传时间、比对状态以及比对结果信息，并可对其进行管理操作。

图 4-19: 用户数据管理界面图

4.6 本章小结

本章详细介绍了基于旋律相似性的音乐比对分析系统的搭建流程。首先对音乐比对分析系统进行了整体业务流程分析以及用户分析。结合系统功能的用例图、用例表以及用例详情对系统功能需求进行了详细的设计说明，并提出了该系统的非功能性需求。此外，介绍了该系统的技术架构设计，并从逻辑视角、开发视角、进程视角和物理视角进行了多角度的描述。本章还介绍了该音乐比对分析系统的数据库设计。最后，对数据交互模块、音乐比对功能中的音乐旋律特征提取和旋律相似性比对模块以及用户数据管理模块进行了详细设计说明与实现，并以类图、伪代码、界面截图等形式进行了说明。

第五章　音乐比对分析系统测试

5.1　测试准备

5.1.1　测试目的

软件测试贯穿于系统研发的整个过程，是软件研发过程的重要环节。本章针对该音乐比对分析系统进行系统测试，验证系统是否满足功能需求以及非功能需求。在软件测试的过程中若发现存在的问题，便于及时采取合理有效的措施对系统进行完善。

5.1.2　测试环境

基于旋律相似性的音乐比对分析系统的测试环境如表5-1所示，针对用户计算机、前端展示服务器、后台应用服务器以及数据库服务器分别从硬件环境和软件环境两个方面对环境进行阐述说明。

表 5-1: 测试环境

	硬件环境	软件环境
用户计算机	CPU 4 核，内存 8GB	Windows Chrome 89.0
前端展示服务器	CPU 4 核，内存 8GB	Ubuntu 16.04.4 LTS，node.js v15.11.0
后台应用服务器	CPU 6 核，内存 8GB	Ubuntu 16.04.4 LTS，Python 3.7，Docker 18.09.2，RabbitMQ 3.7.4
数据库服务器	CPU 4 核，内存 8GB	Ubuntu 16.04.4 LTS，MySQL 5.7

用户端通过个人电脑设备的浏览器来访问该音乐比对分析系统；前端展示服务器负责该系统各个页面的展示以及与用户的交互；后台应用服务器负责响应前端发起的请求，并且负责音乐音频文件的比对分析工作，比对分析算法模型通过 Python 语言实现，并采用 RabbitMQ 消息队列技术作为消息中间件，保证该音乐比对分析系统的高可用性和稳定性，同时使用 Docker 技术进行容器化

处理；数据库服务器采用 MySQL 数据库来存储音乐音频文件信息、音乐侵权案例的详细信息、音乐比对分析结果信息以及用户等相关信息。

5.2 系统测试

5.2.1 功能测试

根据第四章中该音乐比对分析系统的需求分析及设计内容，本小节从用户的角度出发，主要针对数据交互、查看音乐侵权案例、音乐比对以及用户数据管理这四个主要功能对该音乐比对分析系统进行功能测试。根据需求分析中的功能说明设计相对应的测试用例，详细测试用例描述表及测试内容说明如下。

表 5-2: 数据交互测试用例

测试编号	TC1
测试目标	用户可上传所需要比对的音频文件，系统检测所传入的文件格式等基本信息，如文件符合比对条件，则成功传入系统，以便于后续进行比对处理操作；系统返回音乐比对结果后，用户可根据自己的需求对其进行下载保存至本地的操作。
前置条件	1. 用户打开系统，进入比对界面； 2. 用户成功上传音频文件音乐比对结果已经生成完毕； 3. 系统成功显示音乐侵权案例的详情信息。
正常流程	1. 用户按照系统要求的格式整理待上传文件； 2. 调用系统的数据上传接口，分别点击上传原作品音频文件以及疑似相似作品音频文件； 3. 系统接收上传的文件，将文件写入 MySQL 数据库； 4. 当比对结果数据生成后，用户可以点击下载生成的文件； 5. 当查看音乐侵权案例详情时，用户可以点击下载生成的文件。
预期结果	用户上传文件后，成功进行格式等信息检测并返回结果，如没有异常，则成功传入系统。音乐比对结果生成后可以正常进行下载保存至本地。 上传结果情况如下： 1. 如有音频文件格式不符，提示格式错误并让用户重新上传； 2. 如有音频文件不存在，提示文件不存在并让用户重新上传； 3. 如有音频不是单音轨文件，提示不符合要求并让用户重新上传； 4. 如两个音频文件均符合要求，文件上传成功。
实际结果	与预期结果相符

数据交互测试用例如表5-2所示，是根据 UC1 和 UC2 对应的功能需求描述展开的测试用例设计。数据交互模块主要包含数据上传及文件下载两个测试用例。上传需要进行比对的音乐音频文件，系统将传入的数据进行识别并返回相应结果，特别是在用户输入不符合规范的时候，系统能否及时给予相应的错误提醒和指示。音乐比对成功后进行比对数据生成，可以下载保存获取的音乐比对结果文件。在查看侵权案例详情模块，用户也可对其进行下载。结果表明，测试结果与预期结果相符。

查看音乐侵权案例测试用例如表5-3所示，是根据 UC3 和 UC4 对应的功能需求描述展开的测试用例设计。查看音乐侵权案例主要包含音乐侵权案例相关信息的展示，并支持用户进行文件的下载。用户选择查看音乐侵权案例列表并选择感兴趣的案例，系统展示该案例详细的信息，包括案例涉及的音乐作品、案件描述、判决结果以及详细比对结果，其中，详细比对结果为本音乐比对分析系统生成的比对结果。结果表明，测试结果与预期结果相符。

表 5-3: 音乐侵权案例查看测试用例

测试编号	**TC2**
测试目标	用户可选择并查看音乐侵权案例详情并可对其进行下载
前置条件	用户打开系统并选择相应功能
正常流程	1. 用户点击查看音乐侵权案例； 2. 系统显示音乐侵权案例列表以及部分缩略信息； 3. 用户点击查看详情，可以查看到音乐侵权案例的详细信息，包括涉及的音乐作品、案例描述、审判结果以及详细的比对结果。详细的比对结果为本音乐比对分析系统的比对结果。
预期结果	用户可查看音乐侵权案例的列表信息和案例对应的详细信息，并可以将其下载至本地。
实际结果	与预期结果相符

音乐比对测试用例如表5-4所示，是根据 UC5 对应的功能需求描述展开的测试用例设计。音乐比对功能主要供用户自行进行音乐比对。用户通过数据交互将要比对的音乐音频文件成功上传后，等待系统进行比对，系统返回用户详细的比对结果，并可对其进行下载操作。详细比对结果包括整体比对结果以及详细比对结果，具体包含旋律音程和节奏的整体相似度百分比、调性调式信息、相似片段 top3 详细信息、详细的相似片段信息。结果表明，测试结果与预期结果相符。

表 5-4: 音乐比对测试用例

测试编号	**TC3**
测试目标	用户可自行上传疑似相似的文件进行比对，系统可完成比对并返回结果
前置条件	用户打开分析系统进入音乐比对界面
正常流程	1. 用户按照要求的格式成功上传原音乐作品以及疑似相似音乐作品文件，系统显示上传成功的状态； 2. 用户点击开始比对，系统开始进行比对； 3. 比对结束后，系统给出针对上传音乐的比对结果，包括整体比对结果和详细比对结果。整体比对结果包括音乐的调性调式、相似片段 top3、旋律音程和节奏整体相似性，详细比对结果包括所有相似片段的具体位置与长度； 4. 将音乐作品信息与比对结果存储至数据库中。
预期结果	用户可查看到自行上传的音乐音乐文件的详细比对结果，并可对其进行下载。
实际结果	与预期结果相符

用户数据管理测试用例如表 5-5所示，是根据 UC6 对应的功能需求描述展开的测试用例设计。用户数据管理即个人中心模块，主要供用户自行对自己上传的音乐音频文件以及进行比对的记录进行查看、删除、下载等操作。结果表明，测试结果与预期结果相符。

表 5-5: 用户数据管理测试用例

测试编号	**TC4**
测试目标	用户可对数据进行管理工作，包括查看、下载以及删除操作。
前置条件	用户打开个人中心模块
正常流程	1. 用户点击个人中心，进入界面； 2. 系统为用户展示该用户上传的音乐音频文件记录以及进行音乐比对的记录； 3. 用户可以对其进行查看操作； 4. 用户可以对其进行下载操作； 5. 用户可以对其进行删除操作。
实际结果	与预期结果相符

综上所述，该音乐比对分析系统针对数据交互、查看音乐侵权案例、音乐比对以及用户数据管理的功能测试结果表明，均可以实现设计的功能，满足用

户的基本需求，测试结果为通过。

5.2.2 性能测试

我们将对该音乐比对分析系统进行性能测试，以保证系统的可用性。本文分别从前端和后端两个角度，主要针对前端页面的响应时间和后端接口的响应时间两方面来展开性能测试的设计与实现。

针对前端页面响应时间的测试，本文通过使用 Fiddler 工具，分别对用户登录、音乐音频文件上传、开始比对、音乐侵权案例列表查看、音乐侵权案例详情查看、比对结果报告查看、比对结果下载各主要功能界面进行了大量的界面响应操作，通过观察响应时间来测试该音乐比对分析系统的性能。

表 5-6: 前端页面响应结果

界面	最大时间 (ms)	最小时间 (ms)	平均时间 (ms)	测试结果
用户登录	1189.3	528.9	701.9	通过
音乐音频文件上传	1892.3	1009.2	1290.2	通过
开始比对	1989.2	1028.6	1489.2	通过
音乐侵权案例列表查看	1259.2	724.5	892.4	通过
音乐侵权案例详情查看	1322.9	698.2	987.2	通过
比对结果报告查看	1576.8	1098.2	1198.6	通过
比对结果下载	1766.2	988.3	1298.4	通过

该音乐比对分析系统针对以上主要功能进行 100 次测试后对应的最大响应时间、最小响应时间、平均响应时间以及测试结果如表 5-6所示。由统计结果可以看出，该音乐比对分析系统前端界面响应时间均在 2s 以内，均在合理范围内，测试结果为通过。

针对后端接口响应时间的测试，本文通过使用 JMeter 工具来模拟高并发的情况，进行后台接口响应速度测试。JMeter 是 Apache 组织的一款性能测试工具，它目前已经成为性能测试的首选工具。通过 JMeter 可以同时启动多个线程向指定的接口发起请求，以此来模拟多用户高并发的真实场景，实现 web 的压力测试 [52]。使用 JMeter 对本文该音乐比对分析系统进行性能测试的具体实施步骤如下：

首先，确定该音乐比对分析系统待测的功能点接口。根据该音乐分析系统的核心功能以及经常使用到的接口进行分析选择，确定待测接口为音乐音频文件上传接口、开始比对接口以及比对结果查询接口，分别对以上三个功能点接口进行压力测试。

然后，设计并执行测试计划。针对以上确定的功能点接口，根据需求设置相关参数。虚拟用户数设置为 500，线程启动时间设置为 2，循环次数设置为 15。即每 2 秒模拟 500 个用户请求接口，并重复进行 15 次以尽可能避免单次测试可能造成的误差。设计好测试方案后，使用 JMeter 工具开始执行测试计划，添加线程组和 CVS 数据文件相关设置，添加 HTTP 请求并设置请求方式，并在 HTTP 请求中添加监听器，以记录测试的数据。

最后，获取测试结果并分析。分析监听的测试数据并进行分析，根据市场调研，接口请求响应时间的合理范围为 5s 以内。按照此压力测试的需求，分析该音乐比对分析系统的测试结果，若响应时间 <5s，则请求异常数率为 0.00%，即满足测试要求。若不满足测试要求，则分析原因，再次进行压力测试。

表 5-7: 后端接口响应结果

接口	平均值 (ms)	中位数 (ms)	99% 百分位 (ms)	最大值 (ms)	最小值 (ms)	异常率
音乐音频文件上传	208	207	210	221	202	0.00%
开始比对	229	219	232	258	201	0.00%
比对结果查询	244	234	337	345	200	0.00%

针对三个功能点接口的压力测试结果如表 5-7所示。针对以上三个功能，在 2 秒内共进行 500 次并发访问请求，对于音乐音频文件上传接口，平均响应时间为 208ms，响应时间最大值为 221 秒，其中 99% 的用户请求都能在 210ms 内得到系统响应；对于开始比对接口，平均响应时间为 229ms，响应时间最大值为 258 秒，其中 99% 的用户请求都能在 232ms 内得到系统响应；对于比对结果查询接口，平均响应时间为 244ms，响应时间最大值为 345 秒，其中 99% 的用户请求都能在 337ms 内得到系统响应。根据结果可知，所有请求均得到正常响应，异常率均为 0.00%，无错误发生。因此，本次后端接口性能测试的结果均在合理范围内，测试结果为通过。

其中，比对结果查询接口的响应时间如图 5-1所示，可以看出测试出的比对结果查询接口的响应时间有些许波动，但整体较为平稳，响应时间均在

200ms 至 350ms 的区间内。可以得出，该音乐比对分析系统在较高并发的实际场景中也能满足用户的需求。

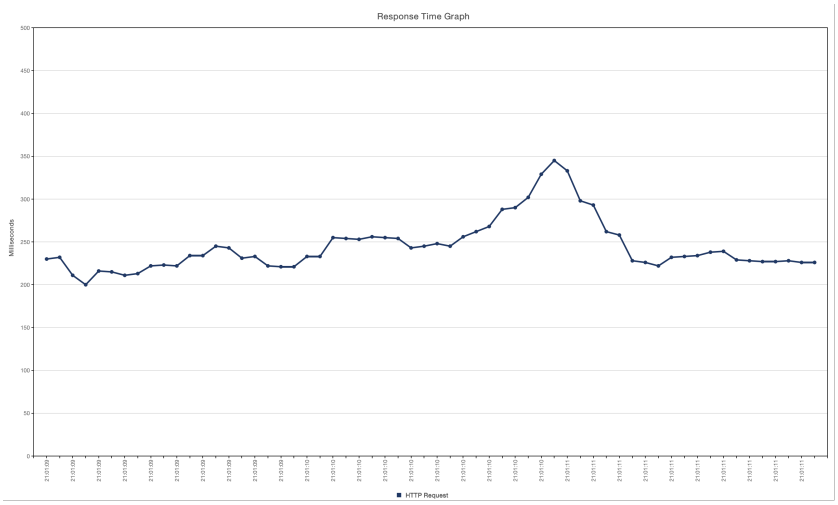


图 5-1: 比对结果查询接口响应情况图

综上所述，系统的前端页面均能及时地对用户的请求进行响应，响应时间均在合理范围内，没有明显的卡顿情况发生，为用户提供了良好的使用体验。模拟高并发情况下的接口响应时间也均在合理范围内，系统依然能够维持正常的业务流程。所以，该音乐比对分析系统的性能测试结果为通过。

5.3 本章小结

本章面向基于旋律相似性的音乐比对分析系统进行了系统测试。系统测试包括功能测试和性能测试。其中，功能测试主要基于功能需求和用例场景设计了多组测试用例，测试结果均与预期相符。性能测试主要借助工具针对前端页面响应时间和后端接口响应时间进行测试，测试结果也与预期相符。

第六章　总结与展望

6.1　总结

随着互联网技术与音乐市场的蓬勃发展，音乐侵权行为变得屡见不鲜，在网络上经常会看到指控他人抄袭音乐作品的消息。为了能够解决目前判定音乐作品是否侵权仅凭主观感觉而无客观比对数据支撑的问题，本文设计并实现了基于旋律相似性的音乐比对分析系统。该系统以音乐比对功能为核心，详细阐述了音乐比对分析的方法和维度，可供用户上传音乐音频文件进行音乐比对，并且可供用户查看以往音乐侵权案件的相关详情信息。

本文的主要工作包括：

首先，从政府、法律、音乐市场三个角度对音乐侵权现象展开调研分析，并且对音乐旋律识别与比对的相关研究现状进行了研究分析，明确了提供客观公正比对结果的音乐比对分析系统的意义和主要的应用场景，为实现该音乐比对分析系统提供了可靠的技术方案。此外，针对本文中需要了解的音乐基础理论知识以及该分析系统涉及的技术进行了概述说明，为后续该系统的设计与实现打好基础。

其次，构建基于旋律相似性的音乐比对分析模型，使用 Python 语言进行算法实现。根据以上介绍的音乐知识对用户上传的音乐音频文件进行一系列的规范化预处理操作，对音乐旋律的旋律音程、节奏以及调性调式等旋律特征信息进行特征提取操作，并对旋律进行多尺度分割，尝试多种字符串匹配算法搜索旋律特征中精确匹配片段，最终通过实验数据表明，Sunday 的改进算法具备更佳的比对性能，此外通过可视化的方法将比对分析结果进行多维度的展示，从而完成音乐比对分析模型的构建。通过实验分析和调查问卷比对了多种算法与方法的比对效率和结果可读性，验证了该模型拥有较高的比对效率并且能够呈现出可读性较高的结果。

最后，完成基于旋律相似性的音乐比对分析系统的整体概要设计、详细设计、实现以及测试工作。借助用例图、用例描述等形式对该系统进行了详细设计，明确了功能性需求和非功能性需求。并通过系统架构图、4+1 视图等形式

对该音乐比对分析系统进行整体且多角度的展现。实现部分通过流程图、类图、伪代码及系统实际页面截图等形式展现该分析系统的实现过程和实现结果。测试部分针对该分析系统进行了系统测试和实验分析。系统测试分为功能测试和性能测试，验证了该音乐比对分析系统的基本功能以及可用性、可靠性等需求。

6.2 展望

基于旋律相似性的音乐比对分析系统可在法院判定、AI 作曲质量检测、作曲专业作品质量检测以及日常比对等场景中进行使用，旨在为音乐产业中疑似侵权案件涉及的音乐作品提供客观、全面且易懂的音乐比对分析数据。但该系统仍存在不足，未来在音乐比对分析方面可以扩展的工作有如下几点:

第一，该音乐比对分析系统目前支持的音乐音频文件格式为 MIDI 格式和 MusicXML 格式，虽然这两个格式目前已经被音乐行业内的作曲人员熟知并广泛使用，但是面对非音乐领域内的用户来说，这两个格式并不属于经常接触的音乐文件格式，普通用户目前常接触的音乐文件格式大多是 mp3、WAV 等等，所以在未来的迭代优化中需要满足更加普遍的音乐文件格式需求，以扩大该音乐比对分析系统的受众范围。

第二，该音乐比对分析系统目前仅支持对单声部的音乐音频进行分析比对，为满足更加普遍的需求，未来需要支持多声部的音乐比对需求。针对音乐音频中人声、旋律和节奏的分离研究是一个难以攻克的问题，在未来的迭代中可针对该问题展开扩展与研究。

第三，该音乐比对分析系统中音乐比对分析的时间效率仍有待提升。该系统在比对较长时长的音乐音频文件时，所需的比对时间较长，在以后音乐比对的研究中可以针对比对分析算法进行研究优化，缩短比对时间，进而提高音乐比对分析系统的效率，提升用户体验。

第四，构建海量的音乐数据库，目前该音乐比对分析系统的比对功能仅支持用户已经明确地找到疑似相似的音乐作品时才可以使用，在未来的迭代中可以直接从海量的音乐音频数据库中匹配到与指定音乐旋律相似性高的歌曲列表，即可以查询到与该音乐相似的全部歌曲，再按照用户需求对其进行详细的音乐比对分析，更有助于保护音乐创作者的权益。

致　　谢

在论文完成之际，我发自内心地向所有在我研究生学习及生活期间给予我帮助的人致谢。

首先，感谢南京大学 iSE 实验室以及我的研究生导师陈振宇老师，陈老师外冷内热，真心为实验室以及每一位学生好，极具责任心，经过多次讨论最终确定了研究方向和毕设选题，陈老师会在百忙之中督促项目和论文进度。今生有幸能加入南京大学 iSE 实验室并且成为陈老师的学生。感谢实验室其他老师两年以来给予我的悉心指导和关心，他们认真负责，在研究生成长道路上起到了正确的指引作用。

其次，感谢我在平安科技实习的导师蒋慧军老师，我经常与蒋老师探讨技术问题并最终找到了合适的毕设选题，感谢蒋老师的指导。此外，还要感谢实习期间遇到的各个岗位的同事们，很幸运在人生初入职场时就遇到他们，谢谢各位老师在技术上给予我的指导与帮助，以及为我提出的宝贵建议。

感谢南京大学提供给我们的美丽校园和优美的生活环境，感谢软件学院所有任课老师的教导，感谢所有辅导员老师的默默支持。我永远以南京大学为豪，诚朴雄伟，励学敦行。

感谢父母和家人们一路以来对我的理解与支持，家就是无论你身在何方，内心都知道永远会有一盏灯为你亮着，我永远爱你们。感谢朋友们的陪伴，愿前程似锦，来日方长。

最后，感谢评阅本论文及论文答辩委员会的各位专家老师们，向你们的敬业和付出致敬。

参考文献

[1] 赵志安，胡舒雅薇. 新图景与新路径：科技创新与融合时代的中国音乐产业——2020 第七届音乐产业高端论坛综述 [J]. 人民音乐, 2021(02) : 82 – 85.

[2] 曹军军. 从“互联网 +”到“音乐 +”的格局变迁观察 [J]. 传媒, 2021(04) : 81 – 83.

[3] 孙新强, 于改之. 美国版权法 [M]. [S.l.] : 美国版权法, 2002.

[4] GHERMAN S. Harmony and its Functionality: A Gloss on the Substantial Similarity Test in Music Copyrights.[J]. Fordham Intellectual Property Media & Entertainment Law Journal, 2009, xix(2) : 483.

[5] 田联韬. 评《乌苏里船歌》与赫哲族民歌的著作权诉讼 [J]. 人民音乐: 评论版, 2003, 000(003) : 15 – 20.

[6] AREWA O B. From JC Bach to hip hop: Musical borrowing, copyright and cultural context[J]. NCL Rev., 2005, 84 : 547.

[7] CAMBOUROPOULOS E. Towards a general computational theory of musical structure[J], 1998.

[8] 冯寅, 周昌乐. 算法作曲的研究进展 [J]. 软件学报, 2006, 17(2) : 209 – 215.

[9] WU J, LIU X, HU X, et al. PopMNet: Generating structured pop music melodies using neural networks[J/OL]. Artif. Intell., 2020, 286 : 103303. https://doi.org/10.1016/j.artint.2020.103303.

[10] MÜLLENSIEFEN D, PENDZICH M. Court decisions on music plagiarism and the predictive value of similarity algorithms[J]. Musicae Scientiae, 2009, 13(1_suppl) : 257 – 295.

[11] BOHAK C, MAROLT M. Calculating Similarity of Folk Song Variants with Melody-based Features[C/OL] // HIRATA K, TZANETAKIS G, YOSHII K. Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009. [S.l.] : International Society for Music Information Retrieval, 2009 : 597 – 602.
http://ismir2009.ismir.net/proceedings/PS4-4.pdf.

[12] SHIN H S, KIM J H. Music therapy on anxiety, stress and maternal-fetal attachment in pregnant women during transvaginal ultrasound[J]. Asian nursing research, 2011, 5(1) : 19 – 27.

[13] KORHAN E A, KHORSHID L, UYAR M. The effect of music therapy on physiological signs of anxiety in patients receiving mechanical ventilatory support[J]. Journal of clinical nursing, 2011, 20(7-8) : 1026 – 1034.

[14] GHIAS A, LOGAN J, CHAMBERLIN D, et al. Query by Humming: Musical Information Retrieval in an Audio Database[C/OL] // ZELLWEGER P. Proceedings of the Third ACM International Conference on Multimedia '95, San Francisco, CA, USA, November 5-9, 1995. [S.l.] : ACM Press, 1995 : 231 – 236.
https://doi.org/10.1145/217279.215273.

[15] BLACKBURN S, ROURE D D. A Tool for Content Based Navigation of Music[C/OL] // EFFELSBERG W, SMITH B C. Proceedings of the 6th ACM International Conference on Multimedia '98, Bristol, England, September 12-16, 1998. [S.l.] : ACM, 1998 : 361 – 368.
https://doi.org/10.1145/290747.290802.

[16] RHO S, HWANG E, KIM M. Music Information Retrieval Using a GA-based Relevance Feedback[C/OL] // 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE 2007), 26-28 April 2007, Seoul, Korea. [S.l.] : IEEE Computer Society, 2007 : 739 – 744.
https://doi.org/10.1109/MUE.2007.161.

[17] PARDO B, SHIFRIN J, BIRMINGHAM W P. Name that tune: A pilot study in finding a melody from a sung query[J/OL]. J. Assoc. Inf. Sci. Technol., 2004,

55(4) : 283 – 300.
https://doi.org/10.1002/asi.10373.

[18] SALAMON J, SERRÀ J, GÓMEZ E. Tonal representations for music retrieval: from version identification to query-by-humming[J/OL]. Int. J. Multim. Inf. Retr., 2013, 2(1) : 45 – 58.
https://doi.org/10.1007/s13735-012-0026-0.

[19] SAVAGE P E, ATKINSON Q D. Automatic Tune Family Identification by Musical Sequence Alignment[C/OL] // MÜLLER M, WIERING F. Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015. : 162 – 168.
http://ismir2015.uma.es/articles/76_Paper.pdf.

[20] MOSTAFA N, WAN Y, AMITABH U, et al. A Machine Learning based Music Retrieval and Recommendation System[C/OL] // CALZOLARI N, CHOUKRI K, DECLERCK T, et al. Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016. [S.l.] : European Language Resources Association (ELRA), 2016.
http://www.lrec-conf.org/proceedings/lrec2016/summaries/1132.html.

[21] 金毅, 黄敏. 基于旋律的音乐检索研究———旋律特征的匹配检索 [J]. 现代图书情报技术, 2003(S2) : 57 – 59.

[22] HUANG S, WANG L, HU S, et al. Query by humming via multiscale transportation distance in random query occurrence context[C/OL] // Proceedings of the 2008 IEEE International Conference on Multimedia and Expo, ICME 2008, June 23-26 2008, Hannover, Germany. [S.l.] : IEEE Computer Society, 2008 : 1225 – 1228.
https://doi.org/10.1109/ICME.2008.4607662.

[23] 李鹏, 周明全, 夏小亮, et al. 改进的基音检测方法及在音乐检索中的应用 [J]. 计算机工程与应用, 2011, 47(006) : 127 – 130.

[24] 姚光超, 郑尧, 肖利民, et al. 基于 MPI+GPU 的哼唱检索系统加速 [J]. 计算机工程与科学, 2013, 035(011) : 168 – 174.

[25] 周利娟, 林鸿飞, 闫俊. 基于 TLDA 和 SVSM 的音乐信息检索模型 [J]. 计算机科学, 2014, 41(002): 174 – 178.

[26] 王培培, 王斌. 支持得分矩阵的音乐检索技术 [J]. 计算机工程与应用, 2017, 053(005): 17 – 23.

[27] ANON. 饶河县四排赫哲族乡政府诉郭颂等侵犯民间文学艺术作品著作权纠纷案 [J]. 中华人民共和国最高人民法院公报, 2004, 000(007): 26 – 32.

[28] 韩光辉, 曾诚. Boyer-Moore 串匹配算法的改进 [J]. 计算机应用, 2014, 34(003): 865 – 868.

[29] 鲁宏伟, 魏凯, 孔华锋. 一种改进的 KMP 高效模式匹配算法 [J]. 华中科技大学学报 (自然科学版), 2006(10): 46 – 48.

[30] 徐珊, 袁小坊, 王东, et al. Sunday 字符串匹配算法的效率改进 [J]. 计算机工程与应用, 2011, 47(029): 96 – 98.

[31] 刘雨心, 孟亮, 窦银科. 一种改进的 Sunday 字符串匹配算法 [J]. 太原理工大学学报, 2013(05): 604 – 607.

[32] 朱永强, 秦志光, 江雪, et al. 基于 Sunday 算法的改良单模式匹配算法 [J]. 计算机应用, 2014, 34(1): 208 – 212.

[33] PARK S, KWON T, LEE J, et al. A Cross-Scape Plot Representation for Visualizing Symbolic Melodic Similarity[C/OL] // FLEXER A, PEETERS G, URBANO J, et al. Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019. 2019: 423 – 430.
http://archives.ismir.net/ismir2019/paper/000050.pdf.

[34] 丁岚, 李蜀果. 乐理教学中音乐作品的旋律分析 [J]. 音乐时空, 2014(1): 138 – 139.

[35] 李重光. 基本乐理通用教材 [M]. [S.l.]: 基本乐理通用教材, 2004.

[36] 杨万钧. 从 MIDI 到 MusicXML——计算机乐谱信息交换格式的发展 [J]. 演艺科技, 2014(7): 45 – 49.

[37] LOY G. Musicians Make a Standard: The MIDI Phenomenon[J]. Computer Music Journal, 1985.

[38] GOOD, MICHAEL. MusicXML: An Internet-Friendly Format for Sheet Music[C] // XML Conference and Expo. 2001.

[39] 吴铁英. 钢琴演奏中的装饰音问题 [J]. 中国音乐, 1995(02): 37–39.

[40] 范建军. MIDI 消息和标准 MIDI 文件格式剖析 [J]. 咸宁学院学报, 2002.

[41] 杨军. MIDI 消息和标准 MIDI 文件格式剖析及应用 [J]. 中南民族大学学报(自然科学版), 2003, 000(0S1): 62–64.

[42] 蓝天, 李扬, 钟婷, et al. 旋律提取技术研究综述 [J]. 计算机应用研究, 2011, 5.

[43] CUTHBERT M S, ARIZA C. Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data[C/OL] // DOWNIE J S, VELTKAMP R C. Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010. [S.l.]: International Society for Music Information Retrieval, 2010: 637–642. http://ismir2010.ismir.net/proceedings/ismir2010-108.pdf.

[44] 刘勇, 林景栋, 穆伟力. 基于 H-K 算法的 MIDI 音乐主旋律提取 [J]. 计算机技术, 2011, 21(6).

[45] LEVENSHTEIN V I. Binary codes capable of correcting deletions, insertions, and reversals[C] // Soviet physics doklady: Vol 10. 1966: 707–710.

[46] KNUTH D E, JR. J H M, PRATT V R. Fast Pattern Matching in Strings[J/OL]. SIAM J. Comput., 1977, 6(2): 323–350. https://doi.org/10.1137/0206024.

[47] BOYER R S, MOORE J S. A Fast String Searching Algorithm[J/OL]. Commun. ACM, 1977, 20(10): 762–772. https://doi.org/10.1145/359842.359859.

[48] SUNDAY D. A Very Fast Substring Search Algorithm[J/OL]. Commun. ACM, 1990, 33(8): 132–142. https://doi.org/10.1145/79173.79184.

[49] ANDERSON C. Docker[J/OL]. IEEE Softw., 2015, 32(3) : 102.
https://doi.org/10.1109/MS.2015.62.

[50] VINOSKI S. Advanced Message Queuing Protocol[J/OL]. IEEE Internet Comput., 2006, 10(6) : 87 – 89.
https://doi.org/10.1109/MIC.2006.116.

[51] ROSTANSKI M, GROCHLA K, SEMAN A. Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ[C/OL] // GANZHA M, MACIASZEK L A, PAPRZYCKI M. Annals of Computer Science and Information Systems, Vol 2 : Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Warsaw, Poland, September 7-10, 2014. 2014 : 879 – 884.
https://doi.org/10.15439/2014F48.

[52] 佘青. 利用 Apache Jmeter 进行 Web 性能测试的研究 [J]. 智能计算机与应用, 2012(02) : 55 – 57.

简历与科研成果

基本信息

王若竹，女，汉族，1997 年 2 月出生，辽宁鞍山人。

教育背景

2019 年 9 月 — 2021 年 7 月 南京大学软件学院 硕士

2015 年 9 月 — 2019 年 7 月 南京邮电大学计算机学院 本科

攻读工程硕士学位期间完成的学术成果

1. 蒋慧军，**王若竹**，“音频修正方法及相关装置”，申请号：2020111756874，已受理。
2. 蒋慧军，**王若竹**，“佛乐旋律相似度的分析方法、装置、设备及存储介质”，申请号：2020111692834，已受理。