# 代码深度研究报告



生成时间：2025-08-19 14:55:04｜本报告由 GitHave AI 生成，仅用于研究目的，不构成任何形式的法律建议或保证，不承担因使用本报告而导致的任何损失或损害。

# 一、前言

* **仓库名称**：chordPrediction
* **仓库描述**：和弦预测
* **仓库分支**：master
* **仓库地址**：<https://github.com/kinglegendzzh/chordPrediction>
* **分析路径**：[/Users/apple/Public/generates-git/chordPrediction/service](file:///Users/apple/Public/generates-git/chordPrediction/service) (子路径: service)
* **项目总结**：

ChordPrediction 是一个基于马尔科夫链的智能音乐创作工具，通过分析历史和当前的和弦序列，实时预测和生成高质量的和弦序列。它支持多种和弦序列训练，提供交互界面进行和弦标注、保存和预览，并采用模块化设计便于扩展和定制。

* **当前分析的service 模块总结**：

该目录 `service` 主要处理音乐相关服务，包括 MIDI 输入管理、和弦序列预测、音阶匹配和音符播放。核心功能涉及 MIDI 设备管理、和弦预测、音阶识别及音符播放控制。关键组件包括 MIDI 输入处理模块、和弦预测模型、音阶匹配算法和音符播放服务。该目录采用面向对象和事件驱动设计，模块化和可扩展性强，支持音乐创作、分析和播放。

# 二、项目描述

# ChordPrediction - 智能音乐创作工具

## 项目背景与目标

ChordPrediction 是一个基于马尔科夫链的和弦预测算法的智能音乐创作工具。它旨在帮助用户生成高质量的和弦序列，提高音乐创作的效率和多样性。该工具适用于音乐创作、音乐教育、音乐游戏等多个领域，特别是对于希望快速生成和弦序列的音乐爱好者和专业人士。

## 核心功能与特色亮点

### 主要功能

1. **和弦预测**：通过分析历史和当前的和弦序列，实时预测和生成和弦。
2. **多种和弦序列训练**：支持多种和弦序列训练，以增强不同音乐风格之间的过渡多样性。
3. **交互界面**：提供一个交互界面，允许用户通过简单的演奏生成所需的和弦，并进行标注、保存、预览和模型训练。

### 关键模块

* **sounds**：管理音频相关的资源和功能，包括音频文件的加载、播放和管理。
* **utils**：提供一系列通用的辅助工具类和函数，简化代码的编写和维护。
* **labels**：定义和管理不同音乐风格和模式的标签和模型。
* **models**：存储与处理压弦键映射相关的数据缓存。
* **records**：存储和管理各种音乐相关的数据和模式。
* **service**：处理音乐相关的服务和功能，包括 MIDI 输入管理、和弦序列预测、音阶匹配等。
* **VirtualKeyboard.py**：创建一个智能化音乐创作工具的用户界面，支持 MIDI 输入设备或键盘映射。
* **MatrixView.py**：创建一个图形用户界面，用于显示概率转移矩阵的和弦分布图。

## 项目价值与作用

ChordPrediction 通过智能化的和弦预测算法，为音乐创作提供了强大的支持。它不仅能够帮助用户快速生成高质量的和弦序列，还能够通过多种和弦序列训练，增强不同音乐风格之间的过渡多样性。此外，交互界面的设计使得用户可以方便地进行和弦标注、保存和预览，进一步提高了音乐创作的效率和质量。

## 项目架构设计

ChordPrediction 采用模块化设计，各个模块协同工作，共同实现音乐创作工具的功能。核心模块包括：

* **sounds**：负责音频资源的加载、播放和管理。
* **utils**：提供通用的辅助工具类和函数，提高代码的复用性和可维护性。
* **labels**：定义和管理音乐风格和模式的标签和模型。
* **models**：存储和处理压弦键映射相关的数据缓存。
* **records**：存储和管理各种音乐相关的数据和模式。
* **service**：处理音乐相关的服务和功能，包括 MIDI 输入管理、和弦序列预测、音阶匹配等。
* **VirtualKeyboard.py**：创建用户界面，支持 MIDI 输入设备或键盘映射。
* **MatrixView.py**：创建图形用户界面，用于显示概率转移矩阵的和弦分布图。

## 技术栈与依赖

ChordPrediction 项目使用了以下主要技术栈和依赖：

* **Python**：作为主要编程语言。
* **Pygame**：用于创建用户界面和处理 MIDI 输入设备。
* **PyInstaller**：用于打包 Python 应用程序，生成可执行文件。
* **MIDI 输入设备**：支持 MIDI 输入，用于实时生成和弦序列。

## 使用方式与扩展性

ChordPrediction 提供了一个交互界面，用户可以通过简单的演奏生成所需的和弦，并进行标注、保存、预览和模型训练。此外，项目采用模块化设计，便于扩展和定制。用户可以根据需要添加新的功能模块或修改现有模块，以满足特定的业务需求。

## 安全性与性能

ChordPrediction 项目在设计时注重安全性与性能。通过模块化设计和优化数据访问机制，确保了应用程序的高效运行。同时，项目遵循 GNU Lesser General Public License（LGPL）许可证，保护用户的自由使用和修改软件库的权利。

## 总结

ChordPrediction 是一个基于马尔科夫链的和弦预测算法的智能音乐创作工具。它通过智能化的和弦预测算法，为音乐创作提供了强大的支持。项目采用模块化设计，各个模块协同工作，共同实现音乐创作工具的功能。通过使用 Pygame 和 PyInstaller 等工具，项目能够生成可执行文件，便于在不同环境中部署和运行。

# 三、分析摘要

## 代码结构概览

service 目录下的代码主要分为以下几个模块：

1. **ChordHMMV2.py**: 实现了一个基于HMM的和弦预测模型，支持情绪标签和风格标签。
2. **MidiInput.py**: 处理MIDI输入，包括键盘映射和MIDI设备管理。
3. **ChordHMM.py**: 实现了一个基于HMM的和弦预测模型，不支持情绪标签和风格标签。
4. **ChordVectorFaiss.py**: 使用Faiss索引和映射进行和弦向量的存储和检索。
5. **ChordVectorAnnoy.py**: 使用Annoy索引和映射进行和弦向量的存储和检索。
6. **MatchingScales.py**: 匹配音阶的类。
7. **soundNoise.py**: 初始化Fluidsynth并加载SoundFont，支持MIDI音乐的播放。
8. **markov.py**: 实现了一个基于马尔科夫链的和弦预测器。
9. **numpyMarkov.py**: 使用NumPy实现马尔科夫链模型。

这些模块通过函数调用和依赖关系紧密相连，共同实现和弦预测和音乐创作的功能。

## 核心模块和函数

### 1. ChordHMMV2.py

* **函数：predict\_next\_chords**
  + **作用**：根据当前和弦序列、情绪标签和风格标签，使用Viterbi算法预测下一个和弦，并返回概率高于阈值的和弦及其匹配概率的列表。
  + **重要性**：高，被引用次数为4次，复杂度为35。

### 2. MidiInput.py

* \*\*函数：\_on\_key\_press\*\*
  + **作用**：处理键盘按下事件，当焦点不在 QLineEdit 控件上时，根据按键字符执行相应的操作。
  + **重要性**：高，被引用次数为1次，复杂度为21。

### 3. ChordHMM.py

* **函数：predict\_chord**
  + **作用**：根据当前和弦序列预测下一个和弦。
  + **重要性**：高，被引用次数为2次，复杂度为22。

### 4. ChordVectorFaiss.py

* **函数：add\_chord**
  + **作用**：向索引中添加和弦及其对应的按键字符串，并将其转换为向量表示。
  + **重要性**：高，被引用次数为4次，复杂度为48。

### 5. ChordVectorAnnoy.py

* **函数：midi\_notes\_to\_vector**
  + **作用**：将 MIDI 音符序列转换为 One-Hot 编码的向量表示。
  + **重要性**：高，被引用次数为5次，复杂度为66。

## 架构依赖关系图

MidiInput.py ──→ ChordVectorFaiss.py  
MidiInput.py ──→ ChordVectorAnnoy.py  
ChordHMMV2.py ──→ ChordVectorFaiss.py  
ChordHMMV2.py ──→ ChordVectorAnnoy.py  
ChordHMM.py ──→ ChordVectorFaiss.py  
ChordHMM.py ──→ ChordVectorAnnoy.py  
MatchingScales.py ──→ ChordVectorFaiss.py  
MatchingScales.py ──→ ChordVectorAnnoy.py  
soundNoise.py ──→ ChordVectorFaiss.py  
soundNoise.py ──→ ChordVectorAnnoy.py  
markov.py ──→ ChordVectorFaiss.py  
markov.py ──→ ChordVectorAnnoy.py  
numpyMarkov.py ──→ ChordVectorFaiss.py  
numpyMarkov.py ──→ ChordVectorAnnoy.py

## 关键路径分析

关键路径包括： 1. **MidiInput.py -> ChordVectorFaiss.py -> add\_chord**: 处理MIDI输入并添加和弦到索引。 2. **ChordHMMV2.py -> predict\_next\_chords**: 使用HMM模型预测下一个和弦。 3. **ChordVectorFaiss.py -> midi\_notes\_to\_vector**: 将MIDI音符序列转换为向量。

这些路径涵盖了系统的主要处理链路，确保了和弦预测和音乐创作的核心功能。

## 架构特点评估

### 优点

1. **模块化设计**: 代码通过模块化设计，便于扩展和定制。
2. **面向对象和事件驱动设计**: 使用面向对象和事件驱动设计，提高了代码的可维护性和可扩展性。

### 改进空间

1. **模块耦合度**: 部分模块之间的耦合度较高，可能需要进一步拆分和解耦。
2. **代码复用**: 部分功能在多个模块中重复实现，可以考虑提取公共函数或模块。

通过以上分析，可以看出service目录下的代码结构清晰，核心模块和函数功能明确，架构依赖关系明确，关键路径分析全面，架构特点评估合理。

# 四、模块明细

以下是chordPrediction项目中service子模块的主要模块信息：

## service

| 模块名称 | 类型 | 完整路径 | 函数数量 |
| --- | --- | --- | --- |
| service | 文件夹 | /Users/apple/Public/generates-git/chordPrediction/service | 77 |

### service

**基本信息：**

* **类型：** 文件夹
* **路径：** service
* **函数数量：** 77
* **初次分析时间：** 2025-08-19T14:40:17.37395+08:00
* **更新时间：** 2025-08-19T14:41:02.320671+08:00

**模块描述：**

该目录 `service` 主要用于处理音乐相关的服务和功能，包括 MIDI 输入管理、和弦序列预测、音阶匹配、音符播放等。核心功能包括 MIDI 输入设备的管理、和弦序列的预测和分析、音阶的匹配和识别，以及 MIDI 音符的播放控制。关键组件包括 MIDI 输入处理模块、和弦预测模型、音阶匹配算法、音符播放服务等。该目录在项目中起到了核心作用，负责提供音乐相关的各种服务和功能，支持音乐创作、分析和播放。设计模式上，该目录采用了面向对象和事件驱动的设计，通过类和方法封装了各个功能模块，实现了模块化和可扩展性。

## service

| 模块名称 | 类型 | 完整路径 | 函数数量 |
| --- | --- | --- | --- |
| ChordVectorFaiss.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py | 8 |
| ChordHMMV2.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py | 16 |
| numpyMarkov.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py | 7 |
| soundNoise.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/soundNoise.py | 4 |
| ChordHMM.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py | 14 |
| ChordVectorAnnoy.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py | 8 |
| markov.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/markov.py | 2 |
| MatchingScales.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/MatchingScales.py | 2 |
| **init**.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/**init**.py | 0 |
| MidiInput.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py | 16 |

### ChordVectorFaiss.py

**基本信息：**

* **类型：** 文件
* **路径：** service/ChordVectorFaiss.py
* **父路径：** service
* **函数数量：** 8
* **初次分析时间：** 2025-08-19T14:40:17.371472+08:00
* **更新时间：** 2025-08-19T14:41:00.18639+08:00

**模块描述：**

该文件 `service/ChordVectorFaiss.py` 主要用于管理一个基于 Faiss 的向量索引，用于存储和检索和弦向量。它实现了向量索引的创建、添加、删除、更新和搜索功能，并提供了将 MIDI 音符序列转换为 One-Hot 编码向量的方法。文件中核心组件包括 Faiss 索引、ID 映射和向量转换逻辑。该文件在项目中起到了数据存储和检索的关键作用，支持和弦数据的高效管理和相似度搜索。设计模式上，该文件采用了面向对象的设计，通过类和方法封装了索引管理的各个功能。

### ChordHMMV2.py

**基本信息：**

* **类型：** 文件
* **路径：** service/ChordHMMV2.py
* **父路径：** service
* **函数数量：** 16
* **初次分析时间：** 2025-08-19T14:40:17.371469+08:00
* **更新时间：** 2025-08-19T14:41:00.18639+08:00

**模块描述：**

该文件`ChordHMMV2.py`主要用于实现和弦序列预测的隐马尔可夫模型（HMM），支持情绪标签和风格标签作为上下文。文件中实现的核心功能包括初始化HMM模型、训练模型参数、预测下一个和弦、生成和弦序列以及可视化发射矩阵。这些功能通过一系列辅助函数和方法实现，包括Baum-Welch算法、Viterbi算法、前向和后向算法等。该文件在项目中扮演着关键角色，负责处理和弦序列预测任务，并提供模型训练和预测的接口。设计模式上，该文件采用面向对象编程，通过类和方法封装了HMM模型的各个组件和操作，使得代码结构清晰，易于维护和扩展。

### numpyMarkov.py

**基本信息：**

* **类型：** 文件
* **路径：** service/numpyMarkov.py
* **父路径：** service
* **函数数量：** 7
* **初次分析时间：** 2025-08-19T14:40:17.371523+08:00
* **更新时间：** 2025-08-19T14:40:50.634274+08:00

**模块描述：**

该文件 `numpyMarkov.py` 主要用于实现和弦序列的预测和分析，通过马尔可夫链模型来预测和弦的转换概率。文件中实现的核心功能包括初始化马尔可夫链模型、构建概率转移矩阵、预测下一个和弦以及可视化转移矩阵。关键组件包括和弦索引字典、概率转移矩阵和马尔可夫链的阶数。该文件在项目中起到了关键作用，特别是在音乐分析和和弦预测领域，通过提供准确的和弦转换概率，支持音乐创作和分析工具的开发。设计模式上，该文件采用了面向对象的设计，通过类和方法封装了马尔可夫链模型的各个功能，使得代码结构清晰，易于维护和扩展。

### soundNoise.py

**基本信息：**

* **类型：** 文件
* **路径：** service/soundNoise.py
* **父路径：** service
* **函数数量：** 4
* **初次分析时间：** 2025-08-19T14:40:17.37155+08:00
* **更新时间：** 2025-08-19T14:40:53.200299+08:00

**模块描述：**

该文件的主要功能是提供一个基于 Fluidsynth 的 MIDI 音乐播放服务，支持通过 MIDI 输入事件控制音符的播放和关闭。文件中实现的核心功能包括初始化 Fluidsynth、加载 SoundFont 文件、设置默认乐器音色以及处理 MIDI 输入事件来控制音符的播放。关键组件包括 Fluidsynth 库、SoundFont 文件和 MIDI 输入事件处理器。该文件在项目中的作用是作为音乐播放服务的核心模块，负责处理 MIDI 输入并控制音符的播放。设计模式上，该文件采用了一种事件驱动的架构，通过处理 MIDI 输入事件来触发音符的播放和关闭操作。

### ChordHMM.py

**基本信息：**

* **类型：** 文件
* **路径：** service/ChordHMM.py
* **父路径：** service
* **函数数量：** 14
* **初次分析时间：** 2025-08-19T14:40:17.371477+08:00
* **更新时间：** 2025-08-19T14:40:55.941905+08:00

**模块描述：**

该文件`ChordHMM.py`主要用于实现和弦序列预测的隐马尔可夫模型（HMM）。它包含了一系列核心功能和方法，用于初始化模型、训练模型参数、预测下一个和弦以及生成和弦序列。文件中的关键组件包括状态转移矩阵、发射矩阵和初始状态概率。通过Baum-Welch算法和Viterbi算法，该文件实现了模型的训练和预测。在项目中，该文件扮演着核心角色，负责处理和弦序列的预测任务。设计模式上，该文件采用了面向对象的设计，通过类和方法封装了HMM模型的各个组成部分和操作。

### ChordVectorAnnoy.py

**基本信息：**

* **类型：** 文件
* **路径：** service/ChordVectorAnnoy.py
* **父路径：** service
* **函数数量：** 8
* **初次分析时间：** 2025-08-19T14:40:17.371471+08:00
* **更新时间：** 2025-08-19T14:40:55.941905+08:00

**模块描述：**

该文件 `service/ChordVectorAnnoy.py` 主要用于管理和操作 Chord 向量的索引和映射。它实现了从头开始创建索引和映射，加载已有的索引和映射文件，以及添加、删除、更新和搜索和弦的功能。核心组件包括 Annoy 索引库用于高效地存储和检索向量，以及一个映射字典用于存储和弦名称、按弦字符串和 MIDI 音符列表之间的关系。该文件在项目中扮演着关键角色，负责处理和弦数据的索引和搜索，从而支持后续的音乐分析和推荐系统。设计模式上，该文件采用了面向对象的设计，通过类和方法封装了索引和映射的管理逻辑，使得代码结构清晰，易于维护和扩展。

### markov.py

**基本信息：**

* **类型：** 文件
* **路径：** service/markov.py
* **父路径：** service
* **函数数量：** 2
* **初次分析时间：** 2025-08-19T14:40:17.371545+08:00
* **更新时间：** 2025-08-19T14:41:00.18639+08:00

**模块描述：**

该文件 `markov.py` 主要用于实现一个和弦预测器，通过构建马尔科夫链模型来预测音乐中的和弦序列。文件中实现的核心功能包括初始化马尔科夫链模型和根据当前和弦序列预测下一个和弦。关键组件包括马尔科夫链模型和状态转移概率。该文件在项目中扮演着和弦预测的核心角色，通过提供准确的和弦预测，支持音乐生成和分析。设计模式上，该文件采用了马尔科夫链模型，通过状态转移概率来预测下一个和弦，体现了马尔科夫过程的应用。

### MatchingScales.py

**基本信息：**

* **类型：** 文件
* **路径：** service/MatchingScales.py
* **父路径：** service
* **函数数量：** 2
* **初次分析时间：** 2025-08-19T14:40:17.371504+08:00
* **更新时间：** 2025-08-19T14:40:53.200299+08:00

**模块描述：**

该文件 `MatchingScales.py` 主要用于实现音阶匹配功能，通过比较输入音符集与预定义音阶的相似度，找到最匹配的音阶。核心功能包括初始化匹配类、将根音转换为 MIDI 音高类编号、合并和去重输入音符集、遍历预定义音阶并计算匹配度，最后按匹配度排序返回匹配的音阶列表。该文件在项目中扮演着音阶识别和匹配的关键角色，通过设计模式实现了音阶的转调和匹配度计算，确保了音阶匹配的准确性和效率。

### **init**.py

**基本信息：**

* **类型：** 文件
* **路径：** service/\_\_init\_\_.py
* **父路径：** service
* **函数数量：** 0
* **初次分析时间：** 2025-08-19T14:40:17.374068+08:00
* **更新时间：** 2025-08-19T14:40:53.200299+08:00

**模块描述：**

该文件是服务模块的初始化文件，主要功能是定义服务模块的入口点，并初始化服务所需的资源和配置。文件中实现的核心功能包括服务的启动、停止以及资源的管理。关键组件包括服务配置、日志记录和异常处理。该文件在项目中起着至关重要的作用，它是服务模块的起点，负责协调其他模块的运行。设计模式上，该文件采用了工厂模式来创建服务实例，确保服务的可扩展性和可维护性。

### MidiInput.py

**基本信息：**

* **类型：** 文件
* **路径：** service/MidiInput.py
* **父路径：** service
* **函数数量：** 16
* **初次分析时间：** 2025-08-19T14:40:17.371474+08:00
* **更新时间：** 2025-08-19T14:40:55.941905+08:00

**模块描述：**

该文件 `MidiInput.py` 主要用于管理MIDI输入设备或键盘映射，支持加载已有的索引和映射文件，或者从头开始创建新的索引和映射。它初始化pygame和pygame.midi，设置MIDI输入设备或启用键盘映射模式，并跟踪按下的键。核心功能包括处理音符的按下和释放事件、初始化MIDI输入设备、启用键盘映射模式以及持续监听MIDI输入事件。该文件在项目中扮演着关键角色，负责处理用户输入的MIDI信号，并将其转换为虚拟键码，以便进一步处理和使用。设计模式上，该文件采用了事件驱动和状态管理的设计，通过处理不同类型的MIDI事件来实现功能。

# 五、函数明细

* **代码文件统计**：py文件10个(100.0%)，

## 重点信息统计

### 前五个重点关注文件（按重要性排序）

1. service/ChordHMMV2.py (重要性得分: 31.380)
2. service/MidiInput.py (重要性得分: 29.209)
3. service/ChordHMM.py (重要性得分: 27.609)
4. service/ChordVectorFaiss.py (重要性得分: 17.612)
5. service/ChordVectorAnnoy.py (重要性得分: 17.559)

### 函数统计维度分析(排名前五)

| 统计维度 | 数值 | 函数名称 | 文件路径 |
| --- | --- | --- | --- |
| 被引用次数 | 5次 | midi\_notes\_to\_vector | service/ChordVectorFaiss.py |
| 被引用次数 | 5次 | midi\_notes\_to\_vector | service/ChordVectorAnnoy.py |
| 被引用次数 | 4次 | add\_chord | service/ChordVectorAnnoy.py |
| 被引用次数 | 4次 | chords\_to\_indices | service/ChordHMMV2.py |
| 被引用次数 | 4次 | add\_chord | service/ChordVectorFaiss.py |
| 扇出数 | 10次 | **init** | service/markov.py |
| 扇出数 | 10次 | **init** | service/soundNoise.py |
| 扇出数 | 10次 | **init** | service/ChordVectorAnnoy.py |
| 扇出数 | 10次 | **init** | service/ChordVectorFaiss.py |
| 扇出数 | 10次 | **init** | service/MidiInput.py |
| 复杂度 | 66 | baum\_welch | service/ChordHMMV2.py |
| 复杂度 | 55 | baum\_welch | service/ChordHMM.py |
| 复杂度 | 48 | add\_chord | service/ChordVectorFaiss.py |
| 复杂度 | 35 | predict\_next\_chords | service/ChordHMMV2.py |
| 复杂度 | 34 | **init** | service/ChordHMMV2.py |
| 调用链深度 | 6层 | run | service/MidiInput.py |
| 调用链深度 | 5层 | \_on\_key\_press | service/MidiInput.py |
| 调用链深度 | 5层 | \_on\_key\_release | service/MidiInput.py |
| 调用链深度 | 5层 | \_process\_midi\_events | service/MidiInput.py |
| 调用链深度 | 4层 | **init** | service/ChordHMMV2.py |

**上述统计中涉及的函数：**

**midi\_notes\_to\_vector** [service/ChordVectorAnnoy.py]: 该函数用于将 MIDI 音符序列转换为 One-Hot 编码的向量表示。它接受一个 MIDI 音符序列作为输入，并根据音符的值在向量中对应的位置设置为1，其余位置为0。

**add\_chord** [service/ChordVectorFaiss.py]: 该函数用于向索引中添加和弦及其对应的按键字符串，并将其转换为向量表示。如果设置了强制插入，则无论是否存在相同记录都会插入；否则，会检查是否存在相同记录，如果存在则跳过插入。

**predict\_next\_chords** [service/ChordHMMV2.py]: 该函数根据当前和弦序列、情绪标签和风格标签，使用Viterbi算法预测下一个和弦，并返回概率高于阈值的和弦及其匹配概率的列表。

\*\*\_on\_key\_release\*\* [service/MidiInput.py]: 该函数用于处理键盘释放事件，当焦点不在文本输入框时，根据按键映射释放相应的音符。

**init** [service/soundNoise.py]: 该函数用于初始化 Fluidsynth 并加载 SoundFont，以支持 MIDI 音乐的播放。它会根据操作系统的不同选择合适的音频驱动，并加载指定的 SoundFont 文件，设置默认的乐器音色。

\*\*\_process\_midi\_events\*\* [service/MidiInput.py]: 该函数用于处理MIDI事件，包括音符按下和释放。它会解析MIDI事件的状态、音符和力度，并根据这些信息调用相应的处理函数。

**add\_chord** [service/ChordVectorAnnoy.py]: 该函数用于将和弦名称和按弦字符串添加到索引中，并将其转换为 MIDI 音符序列的 One-Hot 编码向量。

**chords\_to\_indices** [service/ChordHMM.py]: 该函数将和弦序列转换为对应的索引序列。它通过遍历输入的和弦列表，并使用self.chord\_index字典将每个和弦映射到其对应的索引。如果和弦不在字典中，则返回-1。

**init** [service/markov.py]: 该函数用于初始化一个和弦预测器，通过构建马尔科夫链来预测和弦序列。它接受和弦序列和一个可选的阶数（默认为2），并生成一个马尔科夫链模型，用于后续的和弦预测。

**init** [service/ChordVectorAnnoy.py]: 该类用于管理Chord向量的索引和映射，支持加载已有的索引和映射文件，或者从头开始创建新的索引和映射。

**init** [service/ChordHMMV2.py]: 该函数用于初始化一个HMM模型，该模型用于和弦序列预测，并支持情绪标签和风格标签作为上下文。它处理和弦序列、情绪标签和风格标签，并初始化HMM模型的参数。

**baum\_welch** [service/ChordHMM.py]: 该函数实现了Baum-Welch算法，用于训练隐马尔可夫模型（HMM）的参数，包括状态转移概率矩阵A、观测概率矩阵B和初始状态概率向量pi。该算法通过迭代更新这些参数，以最大化观测序列的概率。

**run** [service/MidiInput.py]: 该函数用于持续监听MIDI输入事件，并在检测到事件时进行处理。

**init** [service/ChordVectorFaiss.py]: 该类用于管理一个基于 Faiss 的向量索引，用于存储和检索和弦向量。它加载已有的索引和映射，或者创建一个新的索引，并提供添加和检索向量的功能。

**baum\_welch** [service/ChordHMMV2.py]: 该函数实现了Baum-Welch算法，用于训练HMM模型的参数，考虑上下文依赖。该算法通过迭代更新模型的转移概率矩阵A和观测概率矩阵B，以及初始状态概率向量pi，以最大化观测序列的似然性。

**midi\_notes\_to\_vector** [service/ChordVectorFaiss.py]: 该函数将 MIDI 音符序列转换为 One-Hot 编码的向量表示，用于后续的向量索引和相似度计算。

**chords\_to\_indices** [service/ChordHMMV2.py]: 该函数将和弦序列转换为对应的索引序列。它遍历输入的和弦列表，并使用self.chord\_index字典将每个和弦映射到其对应的索引。如果和弦不在字典中，则返回-1。

**init** [service/MidiInput.py]: 该类用于管理MIDI输入设备或键盘映射，支持加载已有的索引和映射文件，或者从头开始创建新的索引和映射。它初始化pygame和pygame.midi，设置MIDI输入设备或启用键盘映射模式，并跟踪按下的键。

\*\*\_on\_key\_press\*\* [service/MidiInput.py]: 该函数用于处理键盘按下事件，当焦点不在 QLineEdit 控件上时，根据按键字符执行相应的操作。如果按下反引号（`）或波浪号（~），则切换键盘监听开关。如果监听启用且按键字符在预设的键映射中，则调用\_handle\_note\_press方法处理音符按下事件。

## 函数信息(按重要性排序)

### 1. **init** ([service/ChordVectorFaiss.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：10-30
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：25
* 调用链深度：4
* 功能描述： 该类用于管理一个基于 Faiss 的向量索引，用于存储和检索和弦向量。它加载已有的索引和映射，或者创建一个新的索引，并提供添加和检索向量的功能。
* 实现流程： 初始化时，设置向量维度、索引路径和映射路径。 检查索引和映射文件是否存在，如果存在则加载它们，否则创建一个新的索引。 初始化一个用于存储内部 ID 与和弦信息的映射字典。 如果索引和映射文件存在，从文件中读取索引和映射，并设置下一个可用的 ID。 如果索引和映射文件不存在，创建一个新的 L2 距离的平面索引，并设置下一个可用的 ID 为 0。
* 引入包： os,pickle,faiss,utils.filePath,
* 调用： exists,read\_index,keys,IndexFlatL2,
* 内部依赖描述：

### 2. **init** ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：25-35
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：15
* 调用链深度：4
* 功能描述： 该类用于管理MIDI输入设备或键盘映射，支持加载已有的索引和映射文件，或者从头开始创建新的索引和映射。它初始化pygame和pygame.midi，设置MIDI输入设备或启用键盘映射模式，并跟踪按下的键。
* 实现流程： 初始化pygame和pygame.midi。 调用\_init\_midi\_input函数初始化MIDI输入设备或键盘映射模式。 打印所有可用的MIDI设备信息。 让用户手动选择一个MIDI设备。 如果用户没有选择设备，则启用键盘映射模式。 如果用户选择了设备，则初始化该MIDI设备。 跟踪按下的键，并使用SoundNoise类处理声音噪声。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： **init**,SoundNoise,init,\_initialize\_midi\_input,
* 内部依赖描述：
  + **init**: 该函数用于初始化一个隐马尔可夫模型（HMM），专门用于和弦序列的预测。它接受一个和弦序列列表和隐藏状态数作为输入，并为每个和弦分配一个唯一的索引。初始化过程中，它会计算不同和弦的数量，并建立索引到和弦的映射。此外，它还会随机初始化HMM模型的参数，包括状态转移矩阵A、发射矩阵B和初始状态概率pi。
  + \_initialize\_midi\_input: 该函数用于初始化MIDI输入设备或键盘映射。它首先打印所有可用的MIDI设备信息，然后让用户手动选择一个MIDI设备。如果用户没有选择设备，则启用键盘映射模式。如果用户选择了设备，则初始化该MIDI设备。

### 3. **init** ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：9-27
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：22
* 调用链深度：4
* 功能描述： 该函数用于初始化一个隐马尔可夫模型（HMM），专门用于和弦序列的预测。它接受一个和弦序列列表和隐藏状态数作为输入，并为每个和弦分配一个唯一的索引。初始化过程中，它会计算不同和弦的数量，并建立索引到和弦的映射。此外，它还会随机初始化HMM模型的参数，包括状态转移矩阵A、发射矩阵B和初始状态概率pi。
* 实现流程： 检查输入的和弦序列列表是否为空。如果不为空，则继续初始化；如果为空，则初始化为空列表和空字典。 为和弦序列中的每个唯一和弦分配一个唯一的索引，并建立索引到和弦的映射。 计算不同和弦的数量。 随机初始化HMM模型的参数，包括状态转移矩阵A、发射矩阵B和初始状态概率pi。 如果输入的和弦序列列表为空，则初始化为空列表和空字典，并等待后续加载模型时设置参数A、B和pi。
* 引入包： pickle,utils.filePath,
* 调用： index\_chords,items,initialize\_parameters,
* 内部依赖描述：
  + index\_chords: 该函数用于为和弦序列中的每个唯一和弦分配一个唯一的索引。
  + initialize\_parameters: 该函数用于随机初始化一个隐马尔可夫模型（HMM）的参数，包括状态转移矩阵A、发射矩阵B和初始状态概率pi。

### 4. **init** ([service/MatchingScales.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MatchingScales.py))

* 所属模块/包：MatchingScale
* 行号位置：36-39
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：4
* 调用链深度：4
* 功能描述： 该函数用于初始化一个匹配音阶的类，设置根音、输入音阶和阈值。
* 实现流程： 接收根音、输入音阶和阈值作为参数。 将根音、输入音阶和阈值分别赋值给类的实例变量root\_note、input\_notes和threshold。 初始化完成，为后续的音阶匹配操作做准备。

### 5. **init** ([service/ChordVectorAnnoy.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：9-25
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：21
* 调用链深度：4
* 功能描述： 该类用于管理Chord向量的索引和映射，支持加载已有的索引和映射文件，或者从头开始创建新的索引和映射。
* 实现流程： 初始化时，设置向量维度、索引路径和映射路径。 检查索引路径和映射路径是否存在。 如果存在，加载索引和映射文件，并设置下一个可用的ID为现有ID的最大值加一，然后卸载索引。 如果不存在，初始化下一个可用的ID为0。
* 引入包： os,pickle,annoy,utils.filePath,
* 调用： AnnoyIndex,exists,keys,unbuild,
* 内部依赖描述：

### 6. **init** ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：11-40
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：34
* 调用链深度：4
* 功能描述： 该函数用于初始化一个HMM模型，该模型用于和弦序列预测，并支持情绪标签和风格标签作为上下文。它处理和弦序列、情绪标签和风格标签，并初始化HMM模型的参数。
* 实现流程： 接收和弦序列、情绪标签和风格标签作为输入。 为每个和弦分配一个唯一的索引，并计算不同和弦的数量。 构建上下文，通过将情绪标签列表和风格标签组合成元组，并将这些元组添加到集合中，最终返回一个上下文的列表。 初始化HMM模型的参数，包括状态转移矩阵A、发射矩阵B（上下文依赖）和初始状态概率pi。 如果输入为空，则初始化为空列表和字典，并准备好在加载模型时设置参数。
* 引入包： time,pickle,utils.filePath,
* 调用： index\_chords,items,build\_contexts,initialize\_parameters,
* 内部依赖描述：
  + index\_chords: 该函数用于为和弦序列中的每个唯一和弦分配一个唯一的索引。
  + build\_contexts: 该函数用于构建上下文，通过将情绪标签列表和风格标签组合成元组，并将这些元组添加到集合中，最终返回一个上下文的列表。
  + initialize\_parameters: 该函数用于随机初始化一个隐马尔可夫模型（HMM）的参数，包括状态转移矩阵A、发射矩阵B和初始状态概率pi。

### 7. **init** ([service/soundNoise.py](file:///Users/apple/Public/generates-git/chordPrediction/service/soundNoise.py))

* 所属模块/包：SoundNoise
* 行号位置：10-34
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：32
* 调用链深度：4
* 功能描述： 该函数用于初始化 Fluidsynth 并加载 SoundFont，以支持 MIDI 音乐的播放。它会根据操作系统的不同选择合适的音频驱动，并加载指定的 SoundFont 文件，设置默认的乐器音色。
* 实现流程： 初始化 Fluidsynth 并启动合适的音频驱动。 加载指定的 SoundFont 文件。 设置默认的乐器音色为钢琴音色。
* 引入包： platform,fluidsynth,utils.filePath,logging,
* 调用： **init**,info,Synth,start,filePath,sfload,program\_select,
* 内部依赖描述：
  + **init**: 该函数用于初始化一个隐马尔可夫模型（HMM），专门用于和弦序列的预测。它接受一个和弦序列列表和隐藏状态数作为输入，并为每个和弦分配一个唯一的索引。初始化过程中，它会计算不同和弦的数量，并建立索引到和弦的映射。此外，它还会随机初始化HMM模型的参数，包括状态转移矩阵A、发射矩阵B和初始状态概率pi。
  + filePath: 该函数用于生成指定目录的完整路径。它通过获取当前文件所在目录的上两级目录，然后将传入的文件目录拼接在一起，最终返回完整的路径。

### 8. **init** ([service/markov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/markov.py))

* 所属模块/包：ChordPredictor
* 行号位置：7-29
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：23
* 调用链深度：4
* 功能描述： 该函数用于初始化一个和弦预测器，通过构建马尔科夫链来预测和弦序列。它接受和弦序列和一个可选的阶数（默认为2），并生成一个马尔科夫链模型，用于后续的和弦预测。
* 实现流程： 初始化一个空的和弦列表和一个空的马尔科夫链字典。 将所有输入的和弦序列汇总到一个列表中。 遍历和弦列表，构建马尔科夫链。对于每个和弦序列，根据指定的阶数（order）创建当前状态和下一个状态，并将下一个状态添加到当前状态的列表中。 记录最后一个状态和对应的下一个状态。 打印马尔科夫链的初始化结果。
* 引入包： random,

### 9. **init** ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：6-16
* 重要性得分：3.01111
* 被引用次数：2
* 扇出数：10
* 复杂度：14
* 调用链深度：4
* 功能描述： 该类用于初始化一个马尔可夫链模型，用于预测和弦序列中的下一个和弦。它通过为每个唯一和弦分配一个索引，并构建一个概率转移矩阵来实现这一功能。
* 实现流程： 初始化预测器，设置马尔可夫链的阶数。 构建和弦索引字典，为每个唯一和弦分配一个唯一的整数索引。 计算唯一和弦的数量。 初始化概率转移矩阵，大小为 (n\_chords^order, n\_chords)。 根据输入的和弦序列构建概率转移矩阵，通过马尔可夫链模型来预测和弦的转换概率。
* 调用： index\_chords,zeros,build\_transition\_matrix,
* 内部依赖描述：
  + index\_chords: 该函数用于为和弦序列中的每个唯一和弦分配一个唯一的索引。
  + build\_transition\_matrix: 该函数用于根据输入的和弦序列构建概率转移矩阵，通过马尔可夫链模型来预测和弦的转换概率。

### 10. add\_chord ([service/ChordVectorAnnoy.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：37-47
* 重要性得分：2.75833
* 被引用次数：4
* 扇出数：2
* 复杂度：13
* 调用链深度：2
* 功能描述： 该函数用于将和弦名称和按弦字符串添加到索引中，并将其转换为 MIDI 音符序列的 One-Hot 编码向量。
* 实现流程： 解析按弦字符串，提取根音和音符序列。 将音符序列转换为 MIDI 音符的整数列表。 将 MIDI 音符列表转换为 One-Hot 编码的向量表示。 将向量添加到索引中，并生成一个唯一的 ID。 将和弦名称、按弦字符串和 MIDI 音符列表与该 ID 关联。 递增下一个可用的 ID
* 引入包： os,pickle,annoy,utils.filePath,
* 调用： midi\_notes\_to\_vector,add\_item,
* 内部依赖描述：
  + midi\_notes\_to\_vector: 该函数用于将 MIDI 音符序列转换为 One-Hot 编码的向量表示。它接受一个 MIDI 音符序列作为输入，并根据音符的值在向量中对应的位置设置为1，其余位置为0。

### 11. add\_chord ([service/ChordVectorFaiss.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：42-85
* 重要性得分：2.75833
* 被引用次数：4
* 扇出数：2
* 复杂度：48
* 调用链深度：2
* 功能描述： 该函数用于向索引中添加和弦及其对应的按键字符串，并将其转换为向量表示。如果设置了强制插入，则无论是否存在相同记录都会插入；否则，会检查是否存在相同记录，如果存在则跳过插入。
* 实现流程： 检查是否设置了强制插入，如果未设置则遍历索引中的记录，检查是否存在相同的和弦名称和按键字符串。 如果存在相同记录且未设置强制插入，则跳过插入并返回。 解析按键字符串，提取根音和 MIDI 音符序列。 将 MIDI 音符序列转换为 One-Hot 编码的向量表示。 获取当前索引的总数，作为新向量的 ID。 将新向量添加到索引中。 更新映射，将新向量的 ID、和弦名称、按键字符串和 MIDI 音符序列关联起来。 更新下一个可用的 ID。
* 引入包： os,pickle,faiss,utils.filePath,
* 调用： items,midi\_notes\_to\_vector,add,array,
* 内部依赖描述：
  + midi\_notes\_to\_vector: 该函数用于将 MIDI 音符序列转换为 One-Hot 编码的向量表示。它接受一个 MIDI 音符序列作为输入，并根据音符的值在向量中对应的位置设置为1，其余位置为0。

### 12. baum\_welch ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：141-200
* 重要性得分：2.64829
* 扇出数：7
* 复杂度：66
* 调用链深度：2
* 功能描述： 该函数实现了Baum-Welch算法，用于训练HMM模型的参数，考虑上下文依赖。该算法通过迭代更新模型的转移概率矩阵A和观测概率矩阵B，以及初始状态概率向量pi，以最大化观测序列的似然性。
* 实现流程： 初始化转移概率矩阵A、观测概率矩阵B和初始状态概率向量pi。 将和弦序列转换为对应的索引序列。 对于每个迭代，初始化A\_num、A\_den、B\_num、B\_den和pi\_sum。 对于每个序列，计算前向概率alpha和后向概率beta。 计算xi和gamma，用于更新A和B的值。 累积A和B的更新量。 更新模型参数A、B和pi。 归一化A、B和pi，防止数值问题。 重复上述步骤，直到达到最大迭代次数或收敛
* 引入包： time,pickle,utils.filePath,
* 调用： chords\_to\_indices,zeros,context\_to\_index,forward,backward,nan\_to\_num,
* 内部依赖描述：
  + chords\_to\_indices: 该函数将和弦序列转换为对应的索引序列。它通过遍历输入的和弦列表，并使用self.chord\_index字典将每个和弦映射到其对应的索引。如果和弦不在字典中，则返回-1。
  + context\_to\_index: 该函数用于将情绪标签和风格标签转换为上下文索引。它首先对情绪标签进行排序并转换为元组，然后与风格标签一起组成上下文元组。最后，通过上下文元组在self.context\_index字典中查找对应的索引，如果找到则返回索引，否则返回-1。
  + forward: 该函数实现了前向算法，用于计算HMM模型中的alpha值。alpha值是HMM模型中用于计算观察序列概率的重要参数。
  + backward: 后向算法用于计算HMM模型中的beta值，beta值表示在给定观察序列和状态转移矩阵、发射矩阵的情况下，从当前时间步到序列末尾的路径概率。

### 13. baum\_welch ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：96-146
* 重要性得分：2.64829
* 扇出数：7
* 复杂度：55
* 调用链深度：2
* 功能描述： 该函数实现了Baum-Welch算法，用于训练隐马尔可夫模型（HMM）的参数，包括状态转移概率矩阵A、观测概率矩阵B和初始状态概率向量pi。该算法通过迭代更新这些参数，以最大化观测序列的概率。
* 实现流程： 初始化模型参数A、B和pi。 遍历每个观测序列，计算前向概率alpha和后向概率beta。 计算xi和gamma，用于更新A和B的更新量。 累积A和B的更新量，并更新模型参数A、B和pi。 防止数值问题，对A、B和pi进行归一化。 重复上述步骤，直到达到最大迭代次数或参数收敛。
* 引入包： pickle,utils.filePath,
* 调用： chords\_to\_indices,zeros,forward,backward,
* 内部依赖描述：
  + chords\_to\_indices: 该函数将和弦序列转换为对应的索引序列。它通过遍历输入的和弦列表，并使用self.chord\_index字典将每个和弦映射到其对应的索引。如果和弦不在字典中，则返回-1。
  + forward: 该函数实现了前向算法，用于计算HMM模型中的alpha值。alpha值是HMM模型中用于计算观察序列概率的重要参数。
  + backward: 后向算法用于计算HMM模型中的beta值，beta值表示在给定观察序列和状态转移矩阵、发射矩阵的情况下，从当前时间步到序列末尾的路径概率。

### 14. delete\_chord ([service/ChordVectorFaiss.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：117-143
* 重要性得分：2.53097
* 被引用次数：2
* 扇出数：3
* 复杂度：33
* 调用链深度：2
* 功能描述： 该函数用于从FAISS索引中删除指定的和弦，并重新构建索引。
* 实现流程： 遍历当前的ID映射，将不匹配指定和弦名称的条目及其对应的向量添加到新的向量列表和新的ID映射中。 重置FAISS索引和ID映射，并重新设置下一个可用ID。 如果存在剩余的向量，将它们添加到新的FAISS索引中。 保存新的FAISS索引和ID映射到指定路径。
* 引入包： os,pickle,faiss,utils.filePath,
* 调用： items,midi\_notes\_to\_vector,IndexFlatL2,add,array,save\_index,
* 内部依赖描述：
  + midi\_notes\_to\_vector: 该函数用于将 MIDI 音符序列转换为 One-Hot 编码的向量表示。它接受一个 MIDI 音符序列作为输入，并根据音符的值在向量中对应的位置设置为1，其余位置为0。
  + save\_index: 该函数用于保存Faiss索引和映射文件。它将Faiss索引写入指定路径，并将ID映射序列化后保存到另一个指定路径。

### 15. delete\_chord ([service/ChordVectorAnnoy.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：72-81
* 重要性得分：2.53097
* 被引用次数：2
* 扇出数：3
* 复杂度：14
* 调用链深度：2
* 功能描述： 该函数用于从索引中删除指定的和弦，并更新相关的映射和索引。
* 实现流程： 根据和弦名称查找所有相关的索引ID。 遍历这些索引ID，逐个从索引中删除项，并从映射字典中删除对应的条目。 在删除所有相关项后，重新构建索引并保存更新后的索引和映射文件。
* 引入包： os,pickle,annoy,utils.filePath,
* 调用： items,unbuild,remove\_item,build\_index,
* 内部依赖描述：
  + build\_index: 该函数用于构建索引并保存索引和映射文件。

### 16. predict\_chord ([service/markov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/markov.py))

* 所属模块/包：ChordPredictor
* 行号位置：32-54
* 重要性得分：2.33604
* 扇出数：1
* 复杂度：25
* 调用链深度：2
* 功能描述： 该函数用于预测音乐中的和弦。它根据当前和弦序列和马尔科夫链模型来选择下一个和弦，并返回下一个和弦及其概率。
* 实现流程： 获取当前状态：将当前和弦序列的最后几个和弦（与马尔科夫链状态长度相同）作为当前状态，如果当前和弦序列长度小于马尔科夫链状态长度，则使用当前和弦序列作为当前状态。 检查当前状态是否在马尔科夫链中：如果在，从马尔科夫链中获取下一个和弦的选择列表，并从中随机选择一个和弦作为下一个和弦。计算下一个和弦的概率。 如果当前状态不在马尔科夫链中：从默认的下一个和弦选择列表中随机选择一个和弦作为下一个和弦。如果默认列表为空，则选择和弦序列的第一个和弦。计算下一个和弦的概率。 更新状态：将当前状态和下一个和弦合并为新的状态，并更新马尔科夫链中的下一个状态选择列表。 返回结果：返回下一个和弦及其概率。
* 引入包： random,
* 调用： choice,get,
* 内部依赖描述：

### 17. predict\_chord ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：73-95
* 重要性得分：2.33604
* 扇出数：1
* 复杂度：26
* 调用链深度：2
* 功能描述： 该函数用于根据当前和弦预测下一个和弦，并过滤出概率大于阈值的预测结果。它首先检查当前和弦的数量是否少于马尔科夫链的阶数，然后将当前和弦状态转换为索引，获取该状态的转移概率，并过滤出概率大于阈值的和弦。最后，按概率降序排序并规范化概率，返回预测的和弦及其概率。
* 实现流程： 检查当前和弦的数量是否少于马尔科夫链的阶数，如果少于则抛出异常。 将当前和弦状态转换为对应的索引。 获取该状态的转移概率。 过滤出概率大于阈值的和弦。 按概率降序排序并规范化概率。 返回预测的和弦及其概率。
* 调用： ValueError,state\_to\_index,keys,
* 内部依赖描述：
  + state\_to\_index: 该函数用于将和弦状态转换为对应的索引，以便在马尔可夫模型中进行状态管理。

### 18. predict\_next\_chords ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：228-257
* 重要性得分：2.31751
* 扇出数：3
* 复杂度：35
* 调用链深度：2
* 功能描述： 该函数根据当前和弦序列、情绪标签和风格标签，使用Viterbi算法预测下一个和弦，并返回概率高于阈值的和弦及其匹配概率的列表。
* 实现流程： 将当前和弦序列转换为索引。 检查序列中是否存在未知和弦，如果存在则返回None。 将情绪标签和风格标签转换为上下文索引。 检查上下文索引是否有效，如果无效则返回None。 使用Viterbi算法获取最可能的隐藏状态序列。 获取当前隐藏状态可能的下一个和弦及其概率。 过滤概率高于阈值的和弦。 按概率降序排序并返回预测的和弦及其匹配概率的列表。
* 引入包： time,pickle,utils.filePath,
* 调用： chords\_to\_indices,context\_to\_index,viterbi,where,sort,
* 内部依赖描述：
  + chords\_to\_indices: 该函数将和弦序列转换为对应的索引序列。它通过遍历输入的和弦列表，并使用self.chord\_index字典将每个和弦映射到其对应的索引。如果和弦不在字典中，则返回-1。
  + context\_to\_index: 该函数用于将情绪标签和风格标签转换为上下文索引。它首先对情绪标签进行排序并转换为元组，然后与风格标签一起组成上下文元组。最后，通过上下文元组在self.context\_index字典中查找对应的索引，如果找到则返回索引，否则返回-1。
  + viterbi: 该函数实现了Viterbi算法，用于在给定观察序列的情况下，寻找最可能的隐藏状态序列。

### 19. \_initialize\_midi\_input ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：37-47
* 重要性得分：2.20516
* 被引用次数：1
* 扇出数：4
* 复杂度：15
* 调用链深度：3
* 功能描述： 该函数用于初始化MIDI输入设备或键盘映射。它首先打印所有可用的MIDI设备信息，然后让用户手动选择一个MIDI设备。如果用户没有选择设备，则启用键盘映射模式。如果用户选择了设备，则初始化该MIDI设备。
* 实现流程： 打印所有可用的MIDI设备信息。 让用户手动选择一个MIDI设备。 如果用户没有选择设备，则启用键盘映射模式。 如果用户选择了设备，则初始化该MIDI设备。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： \_print\_midi\_devices,\_choose\_device,\_enable\_keyboard\_mapping,\_initialize\_midi\_device,
* 内部依赖描述：
  + \_print\_midi\_devices: 该函数用于打印所有可用的MIDI设备的信息。
  + \_choose\_device: 该函数用于手动选择MIDI设备。用户可以通过输入设备ID来指定设备，如果不输入则跳过使用键盘映射。
  + \_enable\_keyboard\_mapping: 该函数用于启用键盘映射模式，并启动一个键盘监听器，监听键盘的按下和释放事件，以便在事件发生时调用相应的处理函数。
  + \_initialize\_midi\_device: 该函数用于初始化MIDI输入设备，尝试通过pygame.midi模块打开指定ID的MIDI输入设备，并设置是否使用键盘映射。

### 20. search\_chord ([service/ChordVectorAnnoy.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：55-70
* 重要性得分：2.11958
* 扇出数：2
* 复杂度：18
* 调用链深度：2
* 功能描述： 该函数用于根据给定的 MIDI 音符序列搜索最相似的和弦。它首先将 MIDI 音符序列转换为 One-Hot 编码的向量表示，然后使用 Annoy 索引库查找与该向量最相似的和弦，并返回这些和弦的名称、演奏方式以及与输入向量的距离。
* 实现流程： 将 MIDI 音符序列转换为 One-Hot 编码的向量表示。 使用 Annoy 索引库查找与该向量最相似的和弦。 返回最相似的和弦的名称、演奏方式以及与输入向量的距离。
* 引入包： os,pickle,annoy,utils.filePath,
* 调用： midi\_notes\_to\_vector,get\_nns\_by\_vector,
* 内部依赖描述：
  + midi\_notes\_to\_vector: 该函数用于将 MIDI 音符序列转换为 One-Hot 编码的向量表示。它接受一个 MIDI 音符序列作为输入，并根据音符的值在向量中对应的位置设置为1，其余位置为0。

### 21. search\_chord ([service/ChordVectorFaiss.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：95-115
* 重要性得分：2.11958
* 扇出数：2
* 复杂度：25
* 调用链深度：2
* 功能描述： 该函数用于根据给定的 MIDI 音符序列搜索最相似的和弦。它首先将 MIDI 音符序列转换为 One-Hot 编码的向量表示，然后使用 Faiss 索引搜索最相似的和弦，并返回最相似的和弦名称、按压字符串和距离。
* 实现流程： 将 MIDI 音符序列转换为 One-Hot 编码的向量表示。 将向量表示转换为 NumPy 数组。 使用 Faiss 索引搜索最相似的和弦。 根据搜索结果获取和弦信息，包括和弦名称、按压字符串和距离。 返回最相似的和弦信息列表。
* 引入包： os,pickle,faiss,utils.filePath,
* 调用： midi\_notes\_to\_vector,array,search,get,
* 内部依赖描述：
  + midi\_notes\_to\_vector: 该函数用于将 MIDI 音符序列转换为 One-Hot 编码的向量表示。它接受一个 MIDI 音符序列作为输入，并根据音符的值在向量中对应的位置设置为1，其余位置为0。

### 22. viterbi ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：202-226
* 重要性得分：2.09377
* 被引用次数：2
* 复杂度：27
* 调用链深度：1
* 功能描述： 该函数实现了Viterbi算法，用于在给定观察序列和上下文索引的情况下，寻找最可能的隐藏状态序列。
* 实现流程： 初始化delta和psi数组，用于存储每个时间步和状态下的概率和前向状态索引。 根据初始概率pi和发射概率B，计算第一个时间步的状态概率delta[0]。 从第二个时间步开始，遍历每个时间步，计算每个状态的概率，并记录前向状态索引。 根据计算得到的概率，回溯找到最可能的状态序列。 返回最可能的状态序列。
* 引入包： time,pickle,utils.filePath,
* 调用： zeros,argmax,
* 内部依赖描述：

### 23. viterbi ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：148-171
* 重要性得分：2.09377
* 被引用次数：2
* 复杂度：26
* 调用链深度：1
* 功能描述： 该函数实现了Viterbi算法，用于在给定观察序列的情况下，寻找最可能的隐藏状态序列。
* 实现流程： 初始化delta和psi数组，用于存储每个时间步和状态的最大概率和前驱状态。 设置初始状态概率delta[0]，即初始状态的概率乘以观察序列第一个观测值的概率。 从第二个时间步开始，遍历每个时间步，对于每个状态，计算从上一个时间步到当前状态的最大概率，并记录前驱状态。 在所有时间步结束后，从最后一个时间步开始，根据psi数组回溯，找到最可能的状态序列。 返回找到的最可能的状态序列。
* 引入包： pickle,utils.filePath,
* 调用： zeros,argmax,
* 内部依赖描述：

### 24. \_on\_key\_press ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：79-95
* 重要性得分：2.08305
* 扇出数：1
* 复杂度：21
* 调用链深度：5
* 功能描述： 该函数用于处理键盘按下事件，当焦点不在 QLineEdit 控件上时，根据按键字符执行相应的操作。如果按下反引号（`）或波浪号（~），则切换键盘监听开关。如果监听启用且按键字符在预设的键映射中，则调用\_handle\_note\_press方法处理音符按下事件。
* 实现流程： 获取当前焦点的控件。 检查焦点控件是否为 QLineEdit，如果是，则不触发 MIDI 音符映射。 检查按键字符是否为反引号（`）或波浪号（~），如果是，则切换键盘监听开关，并记录日志。 如果监听启用且按键字符在预设的键映射中，则调用\_handle\_note\_press方法处理音符按下事件。 忽略特殊键（如 Shift 等）的异常情况。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： focusWidget,info,upper,\_handle\_note\_press,
* 内部依赖描述：
  + \_handle\_note\_press: 处理按下音符，将音符添加到活动键列表中，并触发按键按下事件。

### 25. \_process\_midi\_events ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：142-151
* 重要性得分：2.07678
* 被引用次数：1
* 扇出数：2
* 复杂度：13
* 调用链深度：5
* 功能描述： 该函数用于处理MIDI事件，包括音符按下和释放。它会解析MIDI事件的状态、音符和力度，并根据这些信息调用相应的处理函数。
* 实现流程： 遍历传入的MIDI事件列表。 解析每个事件的状态、音符和力度。 记录状态、音符和力度的日志信息。 根据状态和力度判断音符是按下还是释放。 如果是按下音符，调用\_handle\_note\_press函数处理。 如果是释放音符，调用\_handle\_note\_release函数处理。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： info,\_handle\_note\_press,\_handle\_note\_release,
* 内部依赖描述：
  + \_handle\_note\_press: 处理按下音符，将音符添加到活动键列表中，并触发按键按下事件。
  + \_handle\_note\_release: 该函数用于处理音符的释放事件。当接收到音符释放的事件时，首先检查该音符是否在活动音符列表中。如果在，则将其从活动音符列表中移除，并调用内部函数 \_emit\_key\_release 来触发按键释放事件，处理MIDI音符的释放，并记录相关信息。

### 26. build\_transition\_matrix ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：31-55
* 重要性得分：2.05322
* 被引用次数：1
* 扇出数：1
* 复杂度：27
* 调用链深度：2
* 功能描述： 该函数用于根据输入的和弦序列构建概率转移矩阵，通过马尔可夫链模型来预测和弦的转换概率。
* 实现流程： 遍历输入的和弦序列，对于每个和弦序列，计算其马尔可夫链的阶数。 对于每个和弦序列中的和弦组合，将其转换为对应的索引，并记录当前状态和下一个和弦的索引。 更新转移矩阵中对应状态和下一个和弦的计数。 对转移矩阵的每一行进行归一化处理，将其转换为概率。 最终返回构建好的概率转移矩阵。
* 调用： state\_to\_index,get,
* 内部依赖描述：
  + state\_to\_index: 该函数用于将和弦状态转换为对应的索引，以便在马尔可夫模型中进行状态管理。

### 27. predict\_next\_chord ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：173-191
* 重要性得分：2.00719
* 扇出数：2
* 复杂度：22
* 调用链深度：2
* 功能描述： 该函数根据当前和弦序列预测下一个和弦。它首先将和弦转换为索引，然后使用Viterbi算法找到最可能的隐藏状态序列，最后根据这些状态预测下一个和弦。
* 实现流程： 将当前和弦序列转换为索引。 检查序列中是否包含未知和弦，如果包含则返回None。 使用Viterbi算法找到最可能的隐藏状态序列。 根据最后一个隐藏状态预测下一个和弦的概率分布。 选择概率最高的和弦作为预测结果并返回。
* 引入包： pickle,utils.filePath,
* 调用： chords\_to\_indices,viterbi,argmax,
* 内部依赖描述：
  + chords\_to\_indices: 该函数将和弦序列转换为对应的索引序列。它通过遍历输入的和弦列表，并使用self.chord\_index字典将每个和弦映射到其对应的索引。如果和弦不在字典中，则返回-1。
  + viterbi: 该函数实现了Viterbi算法，用于在给定观察序列的情况下，寻找最可能的隐藏状态序列。

### 28. forward ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：61-77
* 重要性得分：1.97319
* 被引用次数：2
* 复杂度：18
* 调用链深度：1
* 功能描述： 该函数实现了前向算法，用于计算HMM模型中的alpha值。alpha值是HMM模型中用于计算观察序列概率的重要参数。
* 实现流程： 初始化Alpha矩阵，其大小为T（观察序列长度）乘以N（状态数）。 设置Alpha矩阵的第一个时间步的值，即初始状态概率乘以发射概率。 从第二个时间步开始，遍历每个时间步，计算每个状态的alpha值，该值由发射概率和前一个时间步的alpha值以及状态转移概率计算得出。 返回计算得到的Alpha矩阵。
* 引入包： pickle,utils.filePath,
* 调用： zeros,
* 内部依赖描述：

### 29. forward ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：101-120
* 重要性得分：1.97319
* 被引用次数：2
* 复杂度：21
* 调用链深度：1
* 功能描述： 该函数实现了前向算法，用于计算HMM模型中的alpha值，alpha值表示在时间t状态下观察到序列O的概率。
* 实现流程： 初始化Alpha矩阵，其大小为T（观察序列长度）x N（状态数）。 根据初始状态概率pi和发射矩阵B，计算时间t=0时的状态概率alpha[0]。 从时间t=1开始，遍历每个时间步，对于每个状态j，计算状态概率alpha[t, j]，该概率由发射概率b和前一时间步所有状态的概率之和乘以状态转移概率A得到。 返回计算得到的Alpha矩阵。
* 引入包： time,pickle,utils.filePath,
* 调用： zeros,
* 内部依赖描述：

### 30. backward ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：79-94
* 重要性得分：1.94993
* 被引用次数：2
* 复杂度：18
* 调用链深度：1
* 功能描述： 后向算法用于计算HMM模型中的beta值，beta值表示在给定观察序列和状态转移矩阵、发射矩阵的情况下，从当前时间步到序列末尾的路径概率。
* 实现流程： 确定观察序列O的长度T和状态转移矩阵A、发射矩阵B的维度N。 初始化beta矩阵，beta[T-1]设为全1，表示在序列末尾时每个状态的概率为1。 从倒数第二个时间步开始，向前遍历到第一个时间步。 对于每个时间步t和每个状态i，计算beta[t, i]，即从状态i出发，经过所有可能的路径到达序列末尾的概率之和。 返回计算得到的beta矩阵。
* 引入包： pickle,utils.filePath,
* 调用： zeros,ones,
* 内部依赖描述：

### 31. backward ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：122-139
* 重要性得分：1.94993
* 被引用次数：2
* 复杂度：20
* 调用链深度：1
* 功能描述： 该函数实现了一个后向算法，用于计算贝塔值（Beta matrix），这是隐马尔可夫模型（HMM）中的一个重要组成部分。贝塔值在HMM中用于计算从当前状态到观测序列末尾的概率。
* 实现流程： 确定观测序列的长度T和状态数N。 初始化贝塔矩阵beta，其中beta[T-1]设置为全1，因为从最后一个观测到序列末尾的概率为1。 从倒数第二个观测开始，逆序遍历观测序列。 对于每个状态i，计算从状态i到序列末尾的概率，这包括状态转移概率A[i, :]和发射概率B[:, O[t + 1], context\_index]的乘积。 将计算结果累加到贝塔矩阵beta[t, i]中。 返回计算得到的贝塔矩阵beta。
* 引入包： time,pickle,utils.filePath,
* 调用： zeros,ones,
* 内部依赖描述：

### 32. save\_model ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：315-333
* 重要性得分：1.93731
* 复杂度：21
* 调用链深度：1
* 功能描述： 该函数用于将模型参数保存到指定的文件路径中。
* 实现流程： 定义一个包含模型参数的字典model\_data。 使用pickle模块将model\_data字典以二进制格式写入到指定的文件路径中。 打印保存成功的消息。
* 引入包： time,pickle,utils.filePath,
* 调用： filePath,dump,
* 内部依赖描述：
  + filePath: 该函数用于生成指定目录的完整路径。它通过获取当前文件所在目录的上两级目录，然后将传入的文件目录拼接在一起，最终返回完整的路径。

### 33. save\_model ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：237-253
* 重要性得分：1.93731
* 复杂度：19
* 调用链深度：1
* 功能描述： 该函数用于将模型参数保存到指定的文件路径中。
* 实现流程： 定义一个包含模型参数的字典model\_data。 使用pickle库将model\_data字典以二进制格式写入到指定的文件路径中。 打印保存成功的消息。
* 引入包： pickle,utils.filePath,
* 调用： filePath,dump,
* 内部依赖描述：
  + filePath: 该函数用于生成指定目录的完整路径。它通过获取当前文件所在目录的上两级目录，然后将传入的文件目录拼接在一起，最终返回完整的路径。

### 34. data\_cut ([service/ChordVectorAnnoy.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py))

* 行号位置：92-103
* 重要性得分：1.90559
* 扇出数：2
* 复杂度：15
* 调用链深度：3
* 功能描述： 该函数用于从缓存文件中读取和处理和弦数据，并将处理后的数据添加到索引中，最后构建索引并保存索引和映射文件。
* 实现流程： 打开指定的缓存文件进行读取。 逐行读取文件内容，去除每行的前后空格。 跳过空行。 将每行内容按’::’分割成和弦名称和按压字符串列表。 将和弦名称和每个按压字符串添加到索引中。 构建索引并保存索引和映射文件。
* 引入包： os,pickle,annoy,utils.filePath,
* 调用： strip,add\_chord,build\_index,
* 内部依赖描述：
  + add\_chord: 该函数用于将和弦名称和按弦字符串添加到索引中，并将其转换为 MIDI 音符序列的 One-Hot 编码向量。
  + build\_index: 该函数用于构建索引并保存索引和映射文件。

### 35. import\_data ([service/ChordVectorFaiss.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py))

* 行号位置：154-165
* 重要性得分：1.90559
* 扇出数：2
* 复杂度：15
* 调用链深度：3
* 功能描述： 该函数用于从缓存文件中导入和处理和弦数据，并将其索引保存到Faiss索引中。
* 实现流程： 打开指定的缓存文件进行读取。 逐行读取文件内容，去除每行的前后空格。 跳过空行。 将每行内容按’::’分割成和弦名称和按压字符串列表。 将和弦名称和每个按压字符串添加到索引中。 保存索引到指定路径。
* 引入包： os,pickle,faiss,utils.filePath,
* 调用： strip,add\_chord,save\_index,
* 内部依赖描述：
  + add\_chord: 该函数用于将和弦名称和按弦字符串添加到索引中，并将其转换为 MIDI 音符序列的 One-Hot 编码向量。
  + save\_index: 该函数用于保存Faiss索引和映射文件。它将Faiss索引写入指定路径，并将ID映射序列化后保存到另一个指定路径。

### 36. generate\_chord\_sequence ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：193-212
* 重要性得分：1.90387
* 扇出数：1
* 复杂度：21
* 调用链深度：2
* 功能描述： 该函数使用训练好的隐马尔可夫模型（HMM）生成指定长度的和弦序列。它首先初始化状态和观测值列表，然后根据初始状态概率和状态转移概率，随机选择下一个状态和对应的和弦。这个过程重复进行，直到生成指定长度的和弦序列。
* 实现流程： 初始化状态和观测值列表。 根据初始状态概率选择第一个状态。 进入循环，直到生成指定长度的和弦序列： 根据当前状态的概率选择下一个状态。 根据当前状态的概率选择一个和弦。 将选择的状态和和弦添加到列表中。 将当前状态更新为选择的下一个状态。 将观测值列表中的和弦转换为对应的和弦名称。 返回生成的和弦序列。
* 引入包： pickle,utils.filePath,
* 调用： choice,
* 内部依赖描述：

### 37. generate\_chord\_sequence ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：259-286
* 重要性得分：1.90387
* 扇出数：1
* 复杂度：30
* 调用链深度：2
* 功能描述： 该函数使用训练好的HMM模型生成指定长度的和弦序列，支持情绪标签和风格标签。
* 实现流程： 接收输入参数：序列长度、情绪标签列表和风格标签。 将情绪标签和风格标签转换为索引，如果索引无效则输出错误信息并返回None。 初始化状态和观察序列列表。 随机选择初始状态。 根据当前状态和上下文索引，计算并归一化每个和弦的概率。 根据和弦概率随机选择一个和弦，并将其添加到观察序列中。 根据当前状态的转移概率，随机选择下一个状态。 重复上述步骤直到达到指定的序列长度。 将观察序列中的索引转换为和弦名称，并返回生成的和弦序列。
* 引入包： time,pickle,utils.filePath,
* 调用： context\_to\_index,choice,
* 内部依赖描述：
  + context\_to\_index: 该函数用于将情绪标签和风格标签转换为上下文索引。它首先对情绪标签进行排序并转换为元组，然后与风格标签一起组成上下文元组。最后，通过上下文元组在self.context\_index字典中查找对应的索引，如果找到则返回索引，否则返回-1。

### 38. \_on\_key\_release ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：97-107
* 重要性得分：1.89155
* 扇出数：1
* 复杂度：14
* 调用链深度：5
* 功能描述： 该函数用于处理键盘释放事件，当焦点不在文本输入框时，根据按键映射释放相应的音符。
* 实现流程： 获取当前焦点的控件。 检查焦点控件是否为QLineEdit类型，如果是则不触发音符释放事件。 尝试获取按键的字符并转换为大写。 检查按键字符是否在键映射中，如果在则调用\_handle\_note\_release方法释放相应的音符。 如果按键字符不在键映射中或发生AttributeError异常，则忽略该事件。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： focusWidget,upper,\_handle\_note\_release,
* 内部依赖描述：
  + \_handle\_note\_release: 该函数用于处理音符的释放事件。当接收到音符释放的事件时，首先检查该音符是否在活动音符列表中。如果在，则将其从活动音符列表中移除，并调用内部函数 \_emit\_key\_release 来触发按键释放事件，处理MIDI音符的释放，并记录相关信息。

### 39. update\_chord ([service/ChordVectorFaiss.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：145-151
* 重要性得分：1.88770
* 扇出数：6
* 复杂度：10
* 调用链深度：3
* 功能描述： 该函数用于更新指定和弦的 MIDI 音符序列。它首先删除指定和弦，然后添加新的 MIDI 音符序列，并最后保存更新后的 Faiss 索引和映射文件。
* 实现流程： 删除指定和弦 添加新的 MIDI 音符序列 保存更新后的 Faiss 索引和映射文件
* 引入包： os,pickle,faiss,utils.filePath,
* 调用： delete\_chord,add\_chord,save\_index,
* 内部依赖描述：
  + delete\_chord: 该函数用于从索引中删除指定的和弦，并更新相关的映射和索引。
  + add\_chord: 该函数用于将和弦名称和按弦字符串添加到索引中，并将其转换为 MIDI 音符序列的 One-Hot 编码向量。
  + save\_index: 该函数用于保存Faiss索引和映射文件。它将Faiss索引写入指定路径，并将ID映射序列化后保存到另一个指定路径。

### 40. update\_chord ([service/ChordVectorAnnoy.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：83-89
* 重要性得分：1.88770
* 扇出数：6
* 复杂度：10
* 调用链深度：3
* 功能描述： 该函数用于更新指定和弦的 MIDI 音符序列。它首先删除指定和弦，然后添加新的音符序列，并最后构建索引并保存索引和映射文件。
* 实现流程： 删除指定和弦 添加新的音符序列 构建索引并保存索引和映射文件
* 引入包： os,pickle,annoy,utils.filePath,
* 调用： delete\_chord,add\_chord,build\_index,
* 内部依赖描述：
  + delete\_chord: 该函数用于从索引中删除指定的和弦，并更新相关的映射和索引。
  + add\_chord: 该函数用于将和弦名称和按弦字符串添加到索引中，并将其转换为 MIDI 音符序列的 One-Hot 编码向量。
  + build\_index: 该函数用于构建索引并保存索引和映射文件。

### 41. \_handle\_note\_release ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：115-119
* 重要性得分：1.85983
* 被引用次数：2
* 扇出数：1
* 复杂度：7
* 调用链深度：4
* 功能描述： 该函数用于处理音符的释放事件。当接收到音符释放的事件时，首先检查该音符是否在活动音符列表中。如果在，则将其从活动音符列表中移除，并调用内部函数 \_emit\_key\_release 来触发按键释放事件，处理MIDI音符的释放，并记录相关信息。
* 实现流程： 检查音符是否在活动音符列表中。 如果音符在活动音符列表中，则将其移除。 调用内部函数 \_emit\_key\_release 来触发按键释放事件，处理MIDI音符的释放，并记录相关信息。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： remove,\_emit\_key\_release,
* 内部依赖描述：
  + remove: 该函数用于从队列中移除指定索引位置的元素。
  + \_emit\_key\_release: 该函数用于触发按键释放事件，处理MIDI音符的释放，并记录相关信息。

### 42. load\_model ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：335-351
* 重要性得分：1.85605
* 复杂度：18
* 调用链深度：1
* 功能描述： 该函数用于从指定路径加载模型参数，包括状态转移矩阵A、发射概率矩阵B、初始概率向量pi、和一些其他相关数据。
* 实现流程： 接收模型文件的路径作为参数。 使用pickle模块以二进制模式打开指定路径下的模型文件。 从文件中加载模型数据，并将其存储在变量model\_data中。 将model\_data中的各个参数（如A、B、pi等）赋值给类的相应属性。 打印加载完成的消息，显示加载的文件路径。
* 引入包： time,pickle,utils.filePath,
* 调用： filePath,
* 内部依赖描述：
  + filePath: 该函数用于生成指定目录的完整路径。它通过获取当前文件所在目录的上两级目录，然后将传入的文件目录拼接在一起，最终返回完整的路径。

### 43. load\_model ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：255-269
* 重要性得分：1.85605
* 复杂度：16
* 调用链深度：1
* 功能描述： 该函数用于从指定路径加载模型参数，并将这些参数赋值给类的实例变量。
* 实现流程： 接收模型文件的路径作为参数。 使用pickle模块以二进制读取模式打开指定路径下的模型文件。 从文件中加载模型数据，并将其存储在变量model\_data中。 从model\_data中提取各个模型参数，并将它们赋值给类的实例变量A、B、pi、chord\_index、n\_states、n\_chords和index\_chord。 打印加载完成的消息，显示模型文件的路径。
* 引入包： pickle,utils.filePath,
* 调用： filePath,
* 内部依赖描述：
  + filePath: 该函数用于生成指定目录的完整路径。它通过获取当前文件所在目录的上两级目录，然后将传入的文件目录拼接在一起，最终返回完整的路径。

### 44. \_emit\_key\_press ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：121-126
* 重要性得分：1.78690
* 被引用次数：1
* 扇出数：1
* 复杂度：9
* 调用链深度：3
* 功能描述： 该函数用于触发按键按下事件，将MIDI音符转换为虚拟键码，并通过信号和日志记录按键按下事件。
* 实现流程： 计算虚拟键码：将传入的MIDI音符减去36，得到虚拟键码。 触发按键按下事件：通过信号v\_key\_pressed发出虚拟键码。 记录日志：使用logging.info记录按键按下事件，包括MIDI音符。 处理MIDI输入：调用self.sn.on\_midi\_input方法，传入MIDI音符和最大值127，处理MIDI输入事件。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： emit,info,on\_midi\_input,
* 内部依赖描述：
  + on\_midi\_input: 该函数用于处理MIDI输入事件，根据输入的音符和速度控制音乐服务中的音符播放或关闭。

### 45. \_emit\_key\_release ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：128-133
* 重要性得分：1.78690
* 被引用次数：1
* 扇出数：1
* 复杂度：9
* 调用链深度：3
* 功能描述： 该函数用于触发按键释放事件，处理MIDI音符的释放，并记录相关信息。
* 实现流程： 计算虚拟按键值，通过将输入的MIDI音符减去36得到。 触发按键释放事件，通过调用self.v\_key\_released.emit(virtual\_key)。 记录按键释放事件的日志，使用logging.info记录MIDI音符的释放信息。 调用self.sn.on\_midi\_input(note, 0, False)方法，通知音量噪声处理模块MIDI音符的释放事件，参数为音符值、0（表示音量为0）和False（表示按键释放）
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： emit,info,on\_midi\_input,
* 内部依赖描述：
  + on\_midi\_input: 该函数用于处理MIDI输入事件，根据输入的音符和速度控制音乐服务中的音符播放或关闭。

### 46. run ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：135-140
* 重要性得分：1.78533
* 扇出数：1
* 复杂度：9
* 调用链深度：6
* 功能描述： 该函数用于持续监听MIDI输入事件，并在检测到事件时进行处理。
* 实现流程： 函数首先进入一个无限循环，直到self.running为False。 在循环中，检查是否启用了键盘映射，如果没有启用并且MIDI输入有事件发生，则调用midi\_input.poll()进行检查。 如果MIDI输入有事件发生，调用midi\_input.read(10)读取最多10个事件，并将这些事件传递给\_process\_midi\_events方法进行处理。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： poll,read,\_process\_midi\_events,
* 内部依赖描述：
  + \_process\_midi\_events: 该函数用于处理MIDI事件，包括音符按下和释放。它会解析MIDI事件的状态、音符和力度，并根据这些信息调用相应的处理函数。

### 47. \_handle\_note\_press ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：109-113
* 重要性得分：1.77874
* 被引用次数：2
* 扇出数：1
* 复杂度：7
* 调用链深度：4
* 功能描述： 处理按下音符，将音符添加到活动键列表中，并触发按键按下事件。
* 实现流程： 检查音符是否在活动键列表中，如果不在，则将其添加到列表中。 调用 \_emit\_key\_press 函数，将 MIDI 音符转换为虚拟键码，并通过信号和日志记录按键按下事件。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： add,\_emit\_key\_press,
* 内部依赖描述：
  + \_emit\_key\_press: 该函数用于触发按键按下事件，将MIDI音符转换为虚拟键码，并通过信号和日志记录按键按下事件。

### 48. visualize\_emission\_matrix ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：214-224
* 重要性得分：1.77569
* 复杂度：18
* 调用链深度：1
* 功能描述： 该函数用于可视化发射矩阵B，通过热图展示发射概率的分布。
* 实现流程： 创建一个大小为10x8的图形窗口。 使用imshow函数显示发射矩阵B，颜色映射为’hot’，插值方法为’nearest’。 添加颜色条以显示颜色与发射概率的对应关系。 设置图形的标题为’Emission Matrix Heatmap’。 设置x轴标签为’Chord Index’，y轴标签为’Hidden State Index’。 显示图形窗口
* 引入包： pickle,utils.filePath,
* 调用： figure,imshow,colorbar,title,xlabel,ylabel,show,
* 内部依赖描述：

### 49. visualize\_emission\_matrix ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：288-302
* 重要性得分：1.77569
* 复杂度：22
* 调用链深度：1
* 功能描述： 该函数用于可视化发射矩阵B（仅针对单个上下文）。它首先检查是否存在上下文数据，如果不存在则打印提示信息并返回。如果存在上下文数据，则使用matplotlib库创建一个图形，并显示发射矩阵B的第一个上下文的热图。热图的颜色映射为’hot’，并添加颜色条、标题和轴标签。
* 实现流程： 检查是否存在上下文数据，如果不存在则打印提示信息并返回。 使用matplotlib库创建一个图形，设置图形大小为10x8。 显示发射矩阵B的第一个上下文的热图，颜色映射为’hot’，并添加颜色条。 设置图形的标题为’Emission Matrix Heatmap (Context 0)‘。 设置x轴标签为’Chord Index’，y轴标签为’Hidden State Index’。 显示图形
* 引入包： time,pickle,utils.filePath,
* 调用： figure,imshow,colorbar,title,xlabel,ylabel,show,
* 内部依赖描述：

### 50. on\_midi\_input ([service/soundNoise.py](file:///Users/apple/Public/generates-git/chordPrediction/service/soundNoise.py))

* 所属模块/包：SoundNoise
* 行号位置：45-49
* 重要性得分：1.74997
* 被引用次数：2
* 扇出数：2
* 复杂度：7
* 调用链深度：2
* 功能描述： 该函数用于处理MIDI输入事件，根据输入的音符和速度控制音乐服务中的音符播放或关闭。
* 实现流程： 检查输入的音符是否为音符按下事件（is\_note\_on为True）。 如果是音符按下事件，调用play\_note\_on函数，传入音符和速度参数，通过fluidsynth库的noteon方法播放音符。 如果不是音符按下事件（即音符关闭事件），调用play\_note\_off函数，传入音符参数，关闭指定音符的播放。
* 引入包： platform,fluidsynth,utils.filePath,logging,
* 调用： play\_note\_on,play\_note\_off,
* 内部依赖描述：
  + play\_note\_on: 该函数用于在音乐服务中播放指定音符。它接受两个参数：音符（note）和速度（velocity，默认为127）。函数通过调用fluidsynth库的noteon方法来实现音符的播放。
  + play\_note\_off: 该函数用于关闭指定音符的播放。

### 51. midi\_notes\_to\_vector ([service/ChordVectorAnnoy.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：27-35
* 重要性得分：1.73990
* 被引用次数：5
* 复杂度：9
* 调用链深度：1
* 功能描述： 该函数用于将 MIDI 音符序列转换为 One-Hot 编码的向量表示。它接受一个 MIDI 音符序列作为输入，并根据音符的值在向量中对应的位置设置为1，其余位置为0。
* 实现流程： 初始化一个长度为 vector\_dim 的全零向量。 遍历输入的 MIDI 音符序列。 对于每个音符，检查其值是否在有效范围内（0 到 vector\_dim-1）。 如果音符值有效，则将向量中对应位置的值设置为1。 返回生成的 One-Hot 编码向量。
* 引入包： os,pickle,annoy,utils.filePath,

### 52. midi\_notes\_to\_vector ([service/ChordVectorFaiss.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：32-40
* 重要性得分：1.73990
* 被引用次数：5
* 复杂度：10
* 调用链深度：1
* 功能描述： 该函数将 MIDI 音符序列转换为 One-Hot 编码的向量表示，用于后续的向量索引和相似度计算。
* 实现流程： 初始化一个长度为 vector\_dim 的全零向量，数据类型为 float32。 遍历输入的 MIDI 音符序列，对于每个音符，如果音符值在有效范围内（0 到 vector\_dim-1），则将该音符位置的向量值设为 1.0。 返回转换后的 One-Hot 编码向量。
* 引入包： os,pickle,faiss,utils.filePath,
* 调用： zeros,
* 内部依赖描述：

### 53. build\_contexts ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：53-66
* 重要性得分：1.73287
* 被引用次数：1
* 复杂度：15
* 调用链深度：1
* 功能描述： 该函数用于构建上下文，通过将情绪标签列表和风格标签组合成元组，并将这些元组添加到集合中，最终返回一个上下文的列表。
* 实现流程： 接收情绪标签列表的列表和风格标签列表作为参数。 遍历情绪标签列表和风格标签列表，使用zip函数将它们一一对应。 将每个情绪标签列表转换为排序后的元组，以便进行哈希操作。 将情绪标签元组和风格标签组合成一个新的元组，作为上下文。 将每个上下文添加到集合中，确保上下文的唯一性。 将集合转换为列表并返回，作为最终的上下文集合。
* 引入包： time,pickle,utils.filePath,
* 调用： add,
* 内部依赖描述：

### 54. index\_chords ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：42-51
* 重要性得分：1.72766
* 被引用次数：3
* 复杂度：11
* 调用链深度：1
* 功能描述： 该函数用于为和弦序列中的每个唯一和弦分配一个唯一的索引。
* 实现流程： 接收一个包含和弦序列的列表作为参数。 创建一个空集合来存储唯一的和弦。 遍历每个和弦序列，将其中的和弦添加到集合中，确保每个和弦只出现一次。 使用enumerate函数为集合中的每个和弦分配一个唯一的索引，并返回一个和弦到索引的映射字典。
* 引入包： time,pickle,utils.filePath,
* 调用： update,
* 内部依赖描述：

### 55. context\_to\_index ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：90-99
* 重要性得分：1.72766
* 被引用次数：3
* 复杂度：11
* 调用链深度：1
* 功能描述： 该函数用于将情绪标签和风格标签转换为上下文索引。它首先对情绪标签进行排序并转换为元组，然后与风格标签一起组成上下文元组。最后，通过上下文元组在self.context\_index字典中查找对应的索引，如果找到则返回索引，否则返回-1。
* 实现流程： 接收情绪标签列表和风格标签作为输入参数。 对情绪标签列表进行排序并转换为元组。 将排序后的情绪元组与风格标签组合成上下文元组。 在self.context\_index字典中查找上下文元组对应的索引。 如果找到索引则返回该索引，否则返回-1。
* 引入包： time,pickle,utils.filePath,
* 调用： get,
* 内部依赖描述：

### 56. index\_chords ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：29-38
* 重要性得分：1.72766
* 被引用次数：3
* 复杂度：11
* 调用链深度：1
* 功能描述： 该函数用于为和弦序列中的每个唯一和弦分配一个唯一的索引。
* 实现流程： 接收一个包含和弦序列的列表作为参数。 初始化一个空集合用于存储唯一的和弦。 遍历每个和弦序列，将其中的和弦更新到唯一和弦集合中。 使用enumerate函数为每个唯一的和弦分配一个索引，并返回一个和弦到索引的映射字典。
* 引入包： pickle,utils.filePath,
* 调用： update,
* 内部依赖描述：

### 57. index\_chords ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：18-29
* 重要性得分：1.72766
* 被引用次数：3
* 复杂度：13
* 调用链深度：1
* 功能描述： 该函数用于为和弦序列中的每个和弦分配一个唯一的索引，并返回一个和弦到索引的映射字典。同时，它还为未知和弦分配了一个独特的索引。
* 实现流程： 初始化一个空集合 unique\_chords 用于存储唯一的和弦。 遍历输入的和弦序列 sequences，将每个和弦添加到 unique\_chords 集合中，确保所有和弦都是唯一的。 计算 unique\_chords 集合的长度，并将其赋值给 unknown\_chord\_index，作为未知和弦的唯一索引。 使用 enumerate 函数遍历 unique\_chords 集合，生成一个和弦到索引的映射字典，并返回该字典。
* 调用： update,
* 内部依赖描述：

### 58. initialize\_parameters ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：40-51
* 重要性得分：1.71843
* 被引用次数：2
* 复杂度：13
* 调用链深度：1
* 功能描述： 该函数用于随机初始化一个隐马尔可夫模型（HMM）的参数，包括状态转移矩阵A、发射矩阵B和初始状态概率pi。
* 实现流程： 生成一个随机的状态转移矩阵A，其维度为n\_states x n\_states。 对矩阵A的每一行进行归一化，确保每一行的和为1。 生成一个随机的发射矩阵B，其维度为n\_states x n\_chords。 对矩阵B的每一行进行归一化，确保每一行的和为1。 生成一个随机的初始状态概率向量pi，其维度为n\_states。 对向量pi进行归一化，确保其和为1。
* 引入包： pickle,utils.filePath,
* 调用： rand,
* 内部依赖描述：

### 59. initialize\_parameters ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：68-80
* 重要性得分：1.71843
* 被引用次数：2
* 复杂度：14
* 调用链深度：1
* 功能描述： 该函数用于随机初始化HMM模型的参数，包括状态转移矩阵A、发射矩阵B（上下文依赖）和初始状态概率pi。
* 实现流程： 生成一个随机的状态转移矩阵A，其维度为(n\_states, n\_states)，并进行行归一化，确保每行的和为1。 生成一个随机的发射矩阵B，其维度为(n\_states, n\_chords, n\_contexts)，并对和弦维度进行归一化，确保每行的和为1。 生成一个随机的初始状态概率pi，其维度为(n\_states)，并进行归一化，确保总和为1。
* 引入包： time,pickle,utils.filePath,
* 调用： rand,
* 内部依赖描述：

### 60. find\_matching\_scales ([service/MatchingScales.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MatchingScales.py))

* 所属模块/包：MatchingScale
* 行号位置：41-64
* 重要性得分：1.71723
* 复杂度：26
* 调用链深度：1
* 功能描述： 该函数用于找到与输入音符集匹配的音阶。它首先将根音转换为 MIDI 音高类编号，然后合并所有和弦的音符并去重。接着，它遍历预定义的音阶，将每个音阶转调到根音，并计算与输入音符集的匹配度。最后，它按匹配度排序并返回匹配的音阶列表。
* 实现流程： 将根音转换为 MIDI 音高类编号。 合并所有和弦的音符并去重，得到输入音符集的音高类集合。 遍历预定义的音阶，将每个音阶转调到根音。 计算每个音阶与输入音符集的匹配度。 按匹配度排序并返回匹配的音阶列表。
* 调用： items,sort,
* 内部依赖描述：

### 61. state\_to\_index ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：57-71
* 重要性得分：1.70564
* 被引用次数：2
* 复杂度：16
* 调用链深度：1
* 功能描述： 该函数用于将和弦状态转换为对应的索引，以便在马尔可夫模型中进行状态管理。
* 实现流程： 接收当前和弦状态作为输入。 初始化索引为0。 遍历和弦状态中的每个和弦，计算其对应的索引。 如果和弦在已知和弦索引中，则使用其索引；否则，使用默认的未知和弦索引。 确保索引在有效范围内。 根据和弦的顺序和位置，计算索引的权重并累加到总索引中。 确保最终索引在有效范围内。 返回计算得到的索引。
* 调用： get,
* 内部依赖描述：

### 62. save\_index ([service/ChordVectorFaiss.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorFaiss.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：87-93
* 重要性得分：1.65881
* 被引用次数：3
* 复杂度：9
* 调用链深度：1
* 功能描述： 该函数用于保存Faiss索引和映射文件。它将Faiss索引写入指定路径，并将ID映射序列化后保存到另一个指定路径。
* 实现流程： 使用faiss库的write\_index方法将索引保存到self.index\_path指定的路径。 打开self.mapping\_path指定的路径，以二进制写模式打开文件。 使用pickle库的dump方法将ID映射self.id\_mapping序列化并写入文件。 完成文件的写入操作后，关闭文件。
* 引入包： os,pickle,faiss,utils.filePath,
* 调用： write\_index,dump,
* 内部依赖描述：

### 63. \_enable\_keyboard\_mapping ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：60-64
* 重要性得分：1.64656
* 被引用次数：1
* 扇出数：1
* 复杂度：7
* 调用链深度：2
* 功能描述： 该函数用于启用键盘映射模式，并启动一个键盘监听器，监听键盘的按下和释放事件，以便在事件发生时调用相应的处理函数。
* 实现流程： 启用键盘映射模式，设置self.use\_keyboard\_mapping为True。 记录日志信息，提示使用键盘映射模式。 调用\_start\_keyboard\_listener函数，启动键盘监听器，监听键盘事件。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： info,\_start\_keyboard\_listener,
* 内部依赖描述：
  + \_start\_keyboard\_listener: 该函数用于启动一个键盘监听器，监听键盘的按下和释放事件，并在事件发生时调用相应的处理函数。

### 64. chords\_to\_indices ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：82-88
* 重要性得分：1.62844
* 被引用次数：4
* 复杂度：8
* 调用链深度：1
* 功能描述： 该函数将和弦序列转换为对应的索引序列。它遍历输入的和弦列表，并使用self.chord\_index字典将每个和弦映射到其对应的索引。如果和弦不在字典中，则返回-1。
* 实现流程： 接收一个和弦列表作为输入。 遍历和弦列表中的每个和弦。 使用self.chord\_index字典查找当前和弦的索引。 如果和弦存在于字典中，则返回其索引；否则返回-1。 将所有和弦的索引组合成一个新的列表并返回。
* 引入包： time,pickle,utils.filePath,
* 调用： get,
* 内部依赖描述：

### 65. chords\_to\_indices ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：53-59
* 重要性得分：1.62844
* 被引用次数：4
* 复杂度：8
* 调用链深度：1
* 功能描述： 该函数将和弦序列转换为对应的索引序列。它通过遍历输入的和弦列表，并使用self.chord\_index字典将每个和弦映射到其对应的索引。如果和弦不在字典中，则返回-1。
* 实现流程： 接收一个和弦列表作为输入。 遍历和弦列表中的每个和弦。 使用self.chord\_index字典查找当前和弦的索引。 如果和弦存在于字典中，则获取其索引；否则，返回-1。 将所有和弦的索引组合成一个新的列表并返回。
* 引入包： pickle,utils.filePath,
* 调用： get,
* 内部依赖描述：

### 66. build\_index ([service/ChordVectorAnnoy.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordVectorAnnoy.py))

* 所属模块/包：ChordVectorIndex
* 行号位置：49-53
* 重要性得分：1.60613
* 被引用次数：3
* 复杂度：8
* 调用链深度：1
* 功能描述： 该函数用于构建索引并保存索引和映射文件。
* 实现流程： 调用self.index.build(n\_trees)方法构建索引，其中n\_trees是树的数量。 调用self.index.save(self.index\_path)方法将构建好的索引保存到指定路径。 使用pickle.dump将id\_mapping对象以二进制格式保存到self.mapping\_path指定的路径。
* 引入包： os,pickle,annoy,utils.filePath,
* 调用： build,save,dump,
* 内部依赖描述：

### 67. \_initialize\_midi\_device ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：71-77
* 重要性得分：1.57896
* 被引用次数：1
* 复杂度：9
* 调用链深度：1
* 功能描述： 该函数用于初始化MIDI输入设备，尝试通过pygame.midi模块打开指定ID的MIDI输入设备，并设置是否使用键盘映射。
* 实现流程： 尝试通过pygame.midi.Input方法初始化MIDI输入设备，传入的参数是device\_id。 如果初始化成功，设置self.midi\_input为初始化后的MIDI输入设备对象，并将self.use\_keyboard\_mapping设置为False。 如果初始化过程中发生pygame.midi.MidiException异常，捕获该异常并抛出一个包含错误信息的ValueError异常。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： Input,ValueError,
* 内部依赖描述：

### 68. visualize\_transition\_matrix ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：97-108
* 重要性得分：1.56718
* 复杂度：19
* 调用链深度：1
* 功能描述： 该函数用于可视化概率转移矩阵，并将其以热图的形式输出为图片。
* 实现流程： 创建一个大小为10x8的图形窗口。 使用热图展示转移矩阵，颜色越深表示概率越大。 添加颜色条以显示颜色与概率的对应关系。 设置图形的标题为’Transition Matrix Heatmap’。 设置X轴标签为’Next Chord Index’，Y轴标签为’Current State Index’。 显示图形窗口以输出热图图片。
* 调用： figure,imshow,colorbar,title,xlabel,ylabel,show,
* 内部依赖描述：

### 69. stop ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：153-159
* 重要性得分：1.52628
* 复杂度：8
* 调用链深度：1
* 功能描述： 停止MIDI输入并清理相关资源。
* 实现流程： 将self.running设置为False，表示停止MIDI输入。 如果self.use\_keyboard\_mapping为False，则关闭MIDI输入并退出pygame。 退出pygame以清理所有pygame相关的资源。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： quit,
* 内部依赖描述：

### 70. \_print\_midi\_devices ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：49-53
* 重要性得分：1.46739
* 被引用次数：1
* 复杂度：7
* 调用链深度：1
* 功能描述： 该函数用于打印所有可用的MIDI设备的信息。
* 实现流程： 遍历所有可用的MIDI设备。 获取每个设备的信息。 打印设备的索引和信息。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： get\_count,get\_device\_info,
* 内部依赖描述：

### 71. \_start\_keyboard\_listener ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：66-69
* 重要性得分：1.40062
* 被引用次数：1
* 复杂度：6
* 调用链深度：1
* 功能描述： 该函数用于启动一个键盘监听器，监听键盘的按下和释放事件，并在事件发生时调用相应的处理函数。
* 实现流程： 导入必要的模块，包括pynput.keyboard用于监听键盘事件。 创建一个键盘监听器对象，指定按下和释放事件的处理函数。 启动监听器，使其开始监听键盘事件。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： Listener,start,
* 内部依赖描述：

### 72. print\_model\_parameters ([service/ChordHMMV2.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMMV2.py))

* 所属模块/包：ChordHMM
* 行号位置：304-313
* 重要性得分：1.37812
* 复杂度：10
* 调用链深度：1
* 功能描述： 该函数用于打印HMM模型的参数，包括状态转移矩阵A、发射矩阵B和初始状态概率pi。
* 实现流程： 打印状态转移矩阵A 打印发射矩阵B 打印初始状态概率pi
* 引入包： time,pickle,utils.filePath,

### 73. print\_model\_parameters ([service/ChordHMM.py](file:///Users/apple/Public/generates-git/chordPrediction/service/ChordHMM.py))

* 所属模块/包：ChordHMM
* 行号位置：226-235
* 重要性得分：1.37812
* 复杂度：10
* 调用链深度：1
* 功能描述： 该函数用于打印HMM模型的参数，包括状态转移矩阵、发射矩阵和初始状态概率。
* 实现流程： 打印状态转移矩阵 (A): 打印发射矩阵 (B): 打印初始状态概率 (pi):
* 引入包： pickle,utils.filePath,

### 74. \_choose\_device ([service/MidiInput.py](file:///Users/apple/Public/generates-git/chordPrediction/service/MidiInput.py))

* 所属模块/包：MidiInput
* 行号位置：55-58
* 重要性得分：1.32355
* 被引用次数：1
* 复杂度：5
* 调用链深度：1
* 功能描述： 该函数用于手动选择MIDI设备。用户可以通过输入设备ID来指定设备，如果不输入则跳过使用键盘映射。
* 实现流程： 提示用户手动指定一个设备ID，或者按回车跳过使用键盘映射。 读取用户输入的设备ID。 如果用户输入了设备ID，则将其转换为整数并返回；如果用户按回车跳过，则返回None。
* 引入包： pygame,pygame.midi,PyQt5.QtCore,PyQt5.QtWidgets,pynput.keyboard,service.soundNoise,pynput,logging,
* 调用： strip,
* 内部依赖描述：

### 75. play\_note\_off ([service/soundNoise.py](file:///Users/apple/Public/generates-git/chordPrediction/service/soundNoise.py))

* 所属模块/包：SoundNoise
* 行号位置：40-42
* 重要性得分：1.17361
* 被引用次数：1
* 复杂度：4
* 调用链深度：1
* 功能描述： 该函数用于关闭指定音符的播放。
* 实现流程： 接收一个音符参数。 调用fluidsynth库的noteoff方法，关闭指定音符的播放。 参数0表示通道号，参数note表示要关闭的音符。
* 引入包： platform,fluidsynth,utils.filePath,logging,
* 调用： noteoff,
* 内部依赖描述：

### 76. play\_note\_on ([service/soundNoise.py](file:///Users/apple/Public/generates-git/chordPrediction/service/soundNoise.py))

* 所属模块/包：SoundNoise
* 行号位置：36-38
* 重要性得分：1.17361
* 被引用次数：1
* 复杂度：4
* 调用链深度：1
* 功能描述： 该函数用于在音乐服务中播放指定音符。它接受两个参数：音符（note）和速度（velocity，默认为127）。函数通过调用fluidsynth库的noteon方法来实现音符的播放。
* 实现流程： 接收音符（note）和速度（velocity）参数。 使用fluidsynth库的noteon方法，传入通道号（0）、音符和速度参数，以播放指定音符。
* 引入包： platform,fluidsynth,utils.filePath,logging,
* 调用： noteon,
* 内部依赖描述：

### 77. transition\_matrix\_to\_string ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：110-116
* 重要性得分：1.16793
* 复杂度：8
* 调用链深度：1
* 功能描述： 该函数将概率转移矩阵转换为字符串形式，便于查看和调试。
* 实现流程： 使用np.array2string方法将概率转移矩阵转换为字符串。 设置precision参数为2，确保输出的小数点后保留两位。 使用separator参数为’, ’，设置元素之间的分隔符为逗号和空格。 返回转换后的字符串形式的转移矩阵。
* 调用： array2string,
* 内部依赖描述：

### 78. **init** ([service/**init**.py](file:///Users/apple/Public/generates-git/chordPrediction/service/__init__.py))