语音分离项目报告 V1.2

北京邮电大学

泛网无线通信实验室短距离通信中心

2019/4/9

版本历史 Record

版本/状态 Version/Status	作者 Author	参与者 Participant	起止日期 Date	备注 Memo
V1.0	余嘉诚	无	2019/4/9	1.初版

目录

1. 引言

1.1 背景

语音增强算法在最近的两百年里飞速发展，从最开始的谱减法到现在的深度学习，语音一直都是受到广泛关注的科研项目。大部分语音增强算法主要目标都是通过分离语音与噪声来减轻噪声对人的体感，或者语音识别任务的干扰。语音分离的工作目标就是找到主要声源进行定位，并对残余声源进行抑制。语音分离任务设计的环境就是鸡尾酒会场景，多个未定位的声源，从中提取主要成分，尽可能地获取纯净语音。主要的项目目标在于实现快速地精准地对语音进行硬件到软件的处理并进行异步输出，同时也需要实现对电脑中已经保存的语音文件进行高精度的降噪或者分离处理。

1.2 术语

术语	**解释**
噪声	环境噪声，是指在工业生产、建筑施工、交通运输和社会生活中所产生的干扰周围生活环境的声音。
鸡尾酒会问题	“ 鸡尾酒会问题”（cocktail party problem）是在计算机语音识别领域的一个问题，当前语音识别技术已经可以以较高精度识别一个人所讲的话，但是当说话的人数为两人或者多人时，语音识别率就会极大的降低，这一难题被称为鸡尾酒会问题。
声源定位	声音定位（sound localization）是指动物利用环境中的声音刺激确定声源方向和距离的行为。用于觅食，寻找幼仔、父母，躲避捕食者等。取决于到达两耳的声音的物理特性变化，包括频率、强度和持续时间上的差别。
Socket	网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个 socket。

2. 系统综述

2.1　功能要求

我们的目标产品的主要功能是接收前端录音，后台人为选择算法进行语音分离，同时后台还有有图形界面的操作环境，可以进行功能选择，前端信息获取，以及文件管理

系统架构

系统组成架构如图 1 所示：

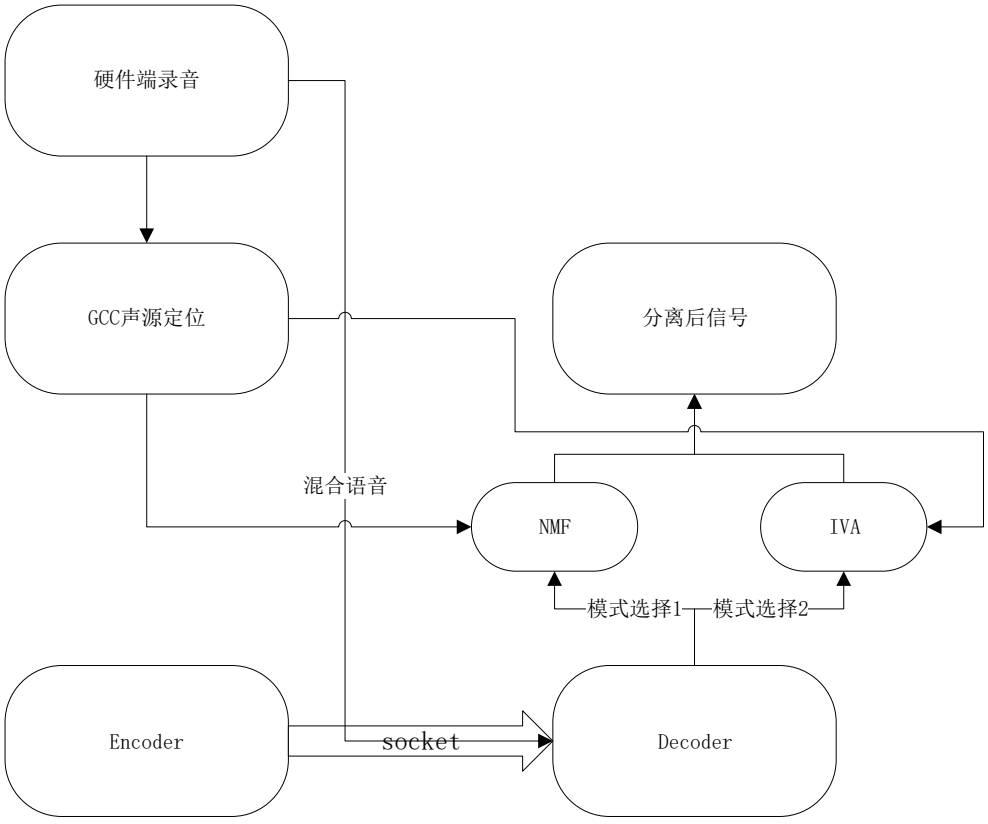

图2-1 系统组成架构图

如上图所示，整个系统

2.2 开发环境

本算法后台开发分为两个部分，
后台中需要使用 python 库：
Numpy: NumPy 系统是 Python 的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比 Python 自身的嵌套列表（nested list structure)结构要高效的多（该结构也可以用来表示矩阵（matrix））。

Librosa：NumPy 系统是 Python 的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比 Python 自身的嵌套列表（nested list structure)结构要高效的多（该结构也可以用来表示矩阵（matrix））。

Tensorflow：TensorFlow 是一个基于数据流编程的符号数学系统，被广泛应用于各类机器学习算法的编程实现，其前身是谷歌的神经网络算法库。

Scikitlearn：sklearn 同样是一种机器学习库，但是它并不基于是使用的张量数据，而是使用了 numpy 的嵌套列表结构，因此在超大规模，乃至分布式时数学计算因为无并行结构，速率稍次于 tensorflow 但是在小规模数据处理时 Scikitlearn 因为不需要封装数据因此也能很快地计算，是本次项目计算实现主体。

re: 用于文件名称，字符串的格式控制，防止错误类别的文件被读取或者写入

matplotlib：matplotlib 中的 pyplot 是 python 中常见的画图工具，我们的算法虽然是对语音进行计算，但是我们也需要展示语谱图。

Winsound: python 播放语音的库文件

Socket: GU socket 通常也称作"套接字"，用于描述 IP 地址和端口，是一个通信链的句柄，应用程序通常通过"套接字"向网络发出请求或者应答网络请求。此处我们用来和前端进行通信。作为数据处理以及 Socket 传输文件的主体。

图形界面：
Tkinter : kinter 模块是 Python 的标准 Tk GUI 工具包的接口.Tk 和 Tkinter 可以在大多数的 Unix 平台下使用,同样可以应用在 Windows 和 Macintosh 系统里.Tk8.0 的后续版本可以实现本地窗口风格,并良好地运行在绝大多数平台中。通过调用内部的回调函数实现页面的跳转以及函数的执行。

3. 前端（客户端）设计

3.1 前端系统功能结构

系统功能模块结构如图 3-1 所示：

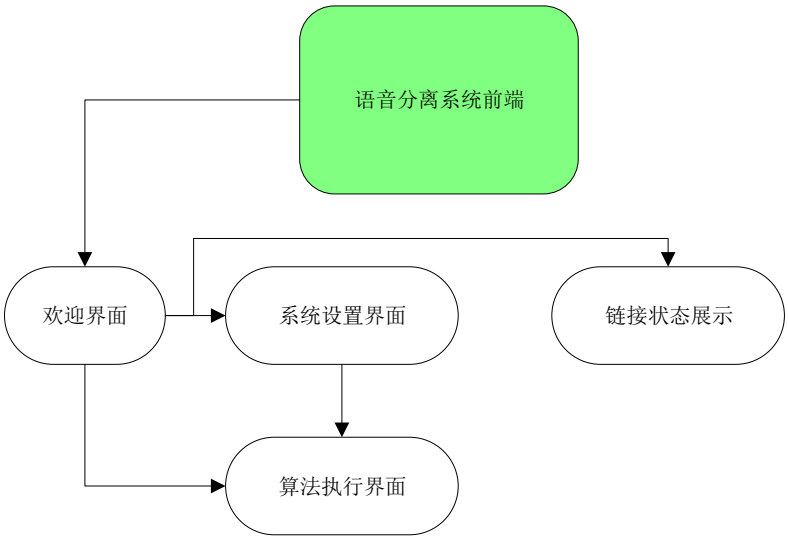

图3-1 系统功能结构图

系统包括欢迎界面，系统设置界面，算法选择界面，算法使用方法介绍界面。其中欢迎界面包含跳转到系统设置，算法选择及简介。欢迎界面主要包含了跳转到其它页面的按钮，以及整个系统的描述文本，系统设置界面包括了算法使用的超参数设置以及运行模式，采样频率等设置。算法选择界面中会给出此算法的适用场景以及优劣势。两种算法都包含了同步输出和从文件读取并处理的工作模式。前端链接状态显示是显示出前端的数据传输状态和 socket 链接的丢包率传输速率等信息，用户应当根据当前状态选择适当的采样频率。

3.2 功能模块详细设计

本节将介绍“语音分离平台”前端功能模块的详细设计。

3.2.1 欢迎界面

3.2.1.1 模块功能说明

欢迎界面是用来显示用户友好的欢迎图片，以及软件操作说明的进入界面，此界面可以跳转到所有功能界面。

3.2.1.2 工作流程

名称	输入	处理	输出
按钮选择	1、 系统设置 2、 算法选择及介绍 3、 链接状态显示	根据用户的按钮选择跳转到对应的界面	下一界面

3.2.1.3 流程图

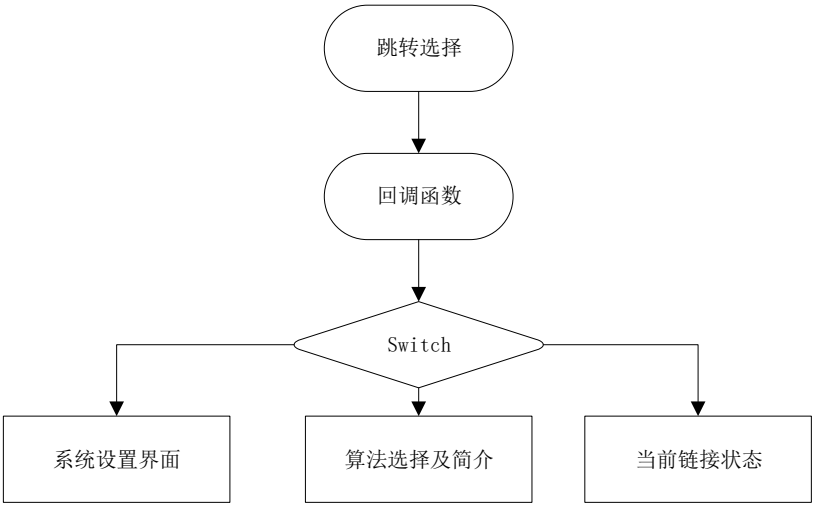

图3-2 欢迎界面示意图

3.2.2 系统设置模块

3.2.2.1 模块功能说明

功能设置模块会将当前的设置信息保存为 json 文件，如果文件存在，那么此界面会直接读取界面中的参数，并显示在当前界面的 text 框体中。

3.2.2.2 工作流程

名称	输入	处理	输出
搜索 json 文件是否存在	None	读取 json 文件储存为键值对	在界面的 text 文本中输出对应的文本
修改设置	在界面中的 text 区域输入对应的参数，按下保存按钮	遍历所有的 text 文本框，更新程序中的字典	将字典保存为 json
返回欢迎界面	点击<-按钮	回调函数关闭当前界面打开欢迎界面	None

3.2.2.3 流程图

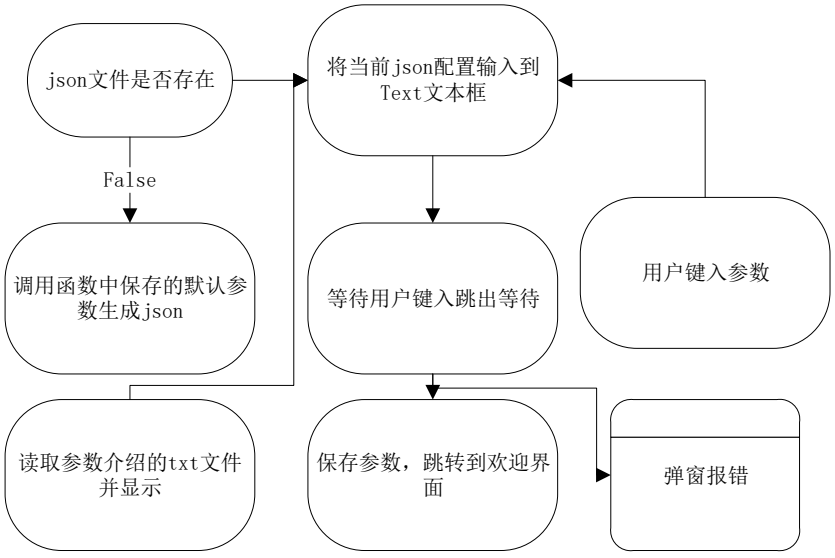

图3-3 个人资料模块流程示意图

3.2.3 算法执行界面

3.2.3.1

算法执行界面包含了参数的读取，后端的算法调用直接读取 json 文件，填充算法函数的输入超参数，进行语音信号分解计算，并保存到文件。或者通过 socket 链接进行语音信号交互，异步处理语音信号并播放，之后保存为文件。
本模块中计算方法分为两类：

（1） 非负矩阵分解，他是一种低秩逼近算法，它的工作性能会随着矩阵的低秩特性损耗而损耗但是任然具备优秀的抗干扰能力，它采用了简单的乘性迭代规则有计算速度快，迭代次数少的性质，在硬件使用 GCC 算法之后我们可以最优语音进行分解达到对非负矩阵分解算法的一种优化。

（2） 独立向量分析算法 IVA 算法，此算法通过对向量化得语音进行不动点迭代获取语音信号的非高斯性极大化的分解向量但是这种算法对于噪声的干扰比较敏感并且运行速率比较慢，所以适合在对于低噪语音文件进行高精度处理。本项目旨在实现一种前端定位-后台再处理的工作模式的语音增强硬件-软件的工作平台。主要的项目目标在于实现快速地精准地对语音进行硬件到软件的处理并进行异步输出，同时也需要实现对电脑中已经保存的语音文件进行高精度的降噪或者分离处理。

名称	输入	处理	输出
模式开关	按钮选择文件读取并处理 按钮选择实时语音信号处理	判断是哪种执行方式调用对应的执行函数	None
非负矩阵分解	模式开关中的传入参数 Json 文件中的超参数 是否实时传输	使用 scikitlearn 中的 NMF 算法对输入信号进行处理	Json 文件中给定声源个数作为数组第一维的二维数组
IVA 算法	模式开关中的传入参数 Json 文件中的超参数 是否实时传输	使用 scikitlearn 中的 IVA 算法对输入信号进行处理	Json 文件中给定声源个数作为数组第一维的二维数组
STFT	语音信号的数组 Json 文件中对应的变换参数	使用 librosa 进行短时傅里叶变化	短时傅里叶变换之后的参数矩阵

文件 I/O	（1） 文件名 （2） 数组+文件名	根据 re.match 函数进行正则匹配如果匹配成功就读取文件如果失败则删除文件，弹窗给报出文件名异常。之后用 librosa.load 函数进行读取	（1） 语音数组 （2） 保存语音文件格式为 T_i.wav
Socket	Json 文件中配置的 IP 地址以及端口号。调用	与前端进行通信，并将文件按照固定大小储存为 buffer 文件，buffer 文件大小从 json 文件中读取	按照 buffer_i 格式保存的语音文件或 numpy 数组格式的文件
播放	语音数组	None	使用 Winsound 播放语音

3.2.3.2 流程图

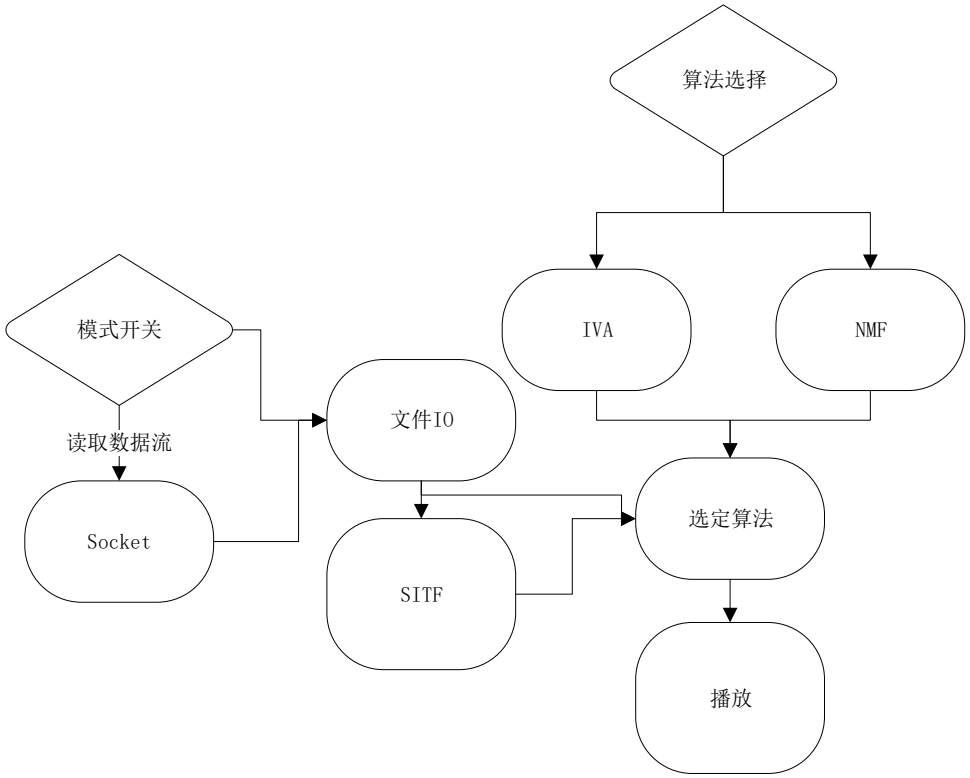

图3-4 算法执行模块流程图

3.2.4 链接检测界面

3.2.4.1 模块功能说明

当用户进入此界面时会执行一次链接测试程序，输出当前的 ping 和

packetlossrate，如果无法链接则弹窗报错。

3.2.4.2 工作流程

名称	输入	处理	输出
检测按钮	用户键入	进行链接测试	丢包率，网络延时
恢复默认参数	用户键入	使用默认参数尝试恢复	修改网络部分的 json 文件

3.2.4.3 流程图

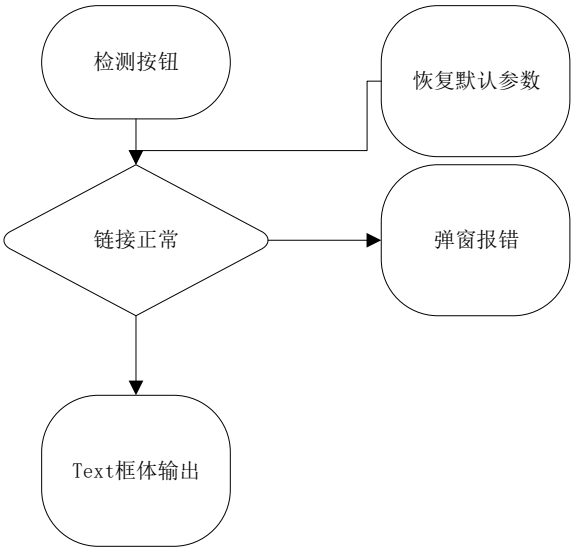

图3-5 链接检测界面

4. 后台管理设计

4.1 非负矩阵分解 NMF 算法描述：

非负矩阵分解，他是一种低秩逼近算法，它的工作性能会随着矩阵的低秩特性损耗而损耗但是任然具备优秀的抗干扰能力，它采用了简单的乘性迭代规则有计算速度快，迭代次数少的性质，在硬件使用 GCC 算法之后我们可以最优语音进行分解达到对非负矩阵分解算法的一种优化。

4.2 独立向量分解 IVA 算法描述

独立向量分析算法 IVA 算法，此算法通过对向量化得语音进行不动点迭代获取语音信号的非高斯性极大化的分解向量但是这种算法对于噪声的干扰比较敏感并且运行速率比较慢，所以适合在对于低噪语音文件进行高精度处理。

4.3 系统功能结构

系统功能模块结构如图 4-1 所示：

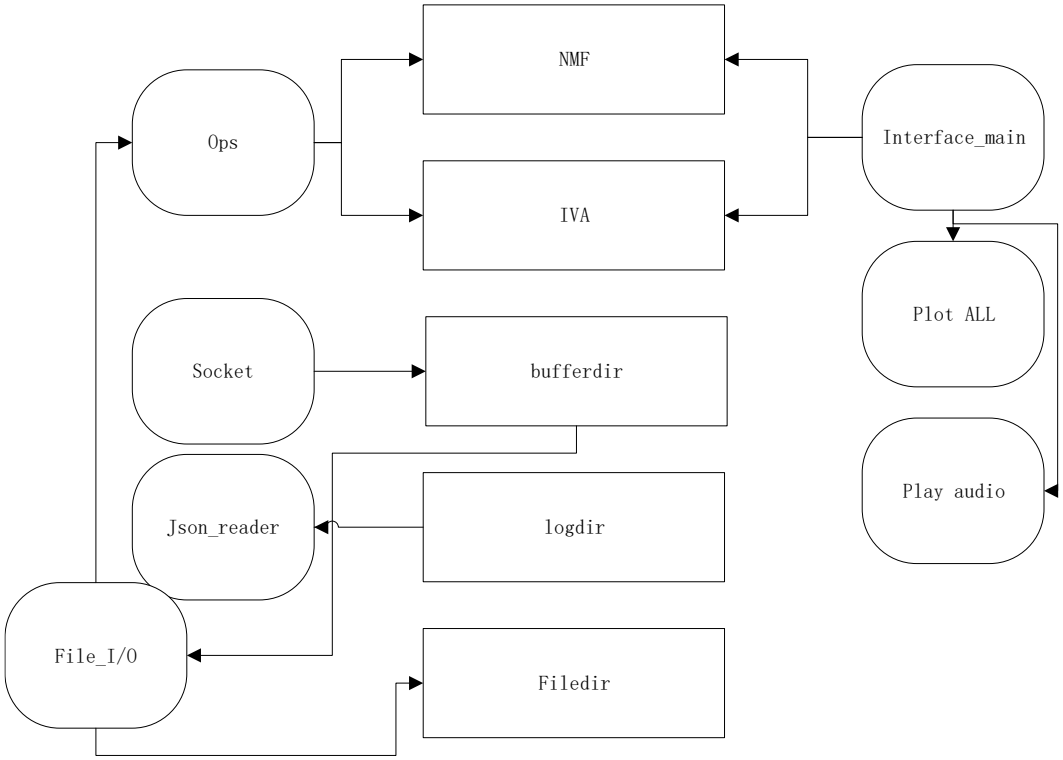

图4-1 系统功能结构图

后台部分的主函数是 Interface_main 获取前端操作信息对后台函数进行调用，如果同时也会接收后台报错反馈到前端进行报错。OPS 类中包含了数据预处理：量子化，数据分段，数据归一化，数据反归一化，STFT,声明 IVA 和声明 NMF 等计算方法，Json_reader 文件中包含了从 logdir 中读取配置参数并还原为字典的方法和读取字典返回参数的方法以及参数范围设定，如果参数异常则会将报错反馈给 Interfacemain 函数。Socket 函数中包含了建立 Socket 链接，测试 Socket 链接的函数，以及将 Socket 传输的文件转换为数组的方法，调用 File I/O 将文件保存到 bufferdir 中。FileI/O 中包含了将数组保存到文件和从文件中读取语音的方法。所有的数据输入输出格式都是使用 np.array 格式 NMF 和 IVA 算法将在之后进行介绍。

4.4 后台函数细节描述

4.4.1.1 模块功能说明

OPS 算数控制模块是在大部分类中都会用到的函数类别，将它们写到一起方便调用。其中含了数据前处理，反前处理过程中的所有方法。
Ops 会被几乎所有函数都调用，因为其中包含了所有预处理以及数值计算过程但是其内部并不会实际地进行计算，只负责被调用。

4.4.1.2 模块功能说明

归一化给出极小极大归一化，预白化的方式，并将归一化系数保留等待还原，粒子化是将语音转化为整数进行计算，切片是使用静默检测将语音的完全静默的两端去掉之后分解为适合计算的长度。之后将语音分段。

4.4.1.3 工作流程

名称	输入	处理	输出
init（self）	None	调用 json_reader 中的 read 函数赋值给本地参数	None
归一化	Np.float32 类型的矩阵	按行遍历数组减去数组中最小值之后除以数组中最大最小值值差返回 0-1 区间的数据	Np.float32 类型的矩阵所有元素都处于 0-1 区间内。 最小值 最大值

反归一化	Np.float32 类型的矩阵 最小值 最大值	按行遍历数组，每行先乘以最小值最大值只差然后加上最小值	Np.float32 类型的矩阵
预白化	Np.float32 类型的矩阵	按行遍历每行减去该行均值并处以该行方差	Np.float32 类型的矩阵 均值数组 方差数组
反白化	Np.float32 类型的矩阵 均值 方差	按行遍历每行乘以方差，并加上均值	Np.float32 类型的矩阵
粒子化	Np.float32 类型的矩阵 采样通道数目	按行遍历，将归一化之后的数据搬移到-1,1 区间，之后使用 mu-law 变化将数据划分为采样通道数目量级的整数	Np.int8 类型的矩阵
反粒子化	忽视数据类型，但是要求是经过粒子化后的数据	进行反 mu-law 变换	Np.float32 的矩阵
STFT	Np.float32 类别的数组	使用 libroas 中的 STFT 进行变换	Np.complex128 的矩阵
ISTFT	任意类别数组但一定要经过 STFT	如果是实数则加入随机噪声的角度，如果本来为复数则直接使用 librosa 中的 ISTFT	Np.float 类型的数组
切片（无反变化）	给定阈值、np.float32 类别数组	使用 librosa 进行切分，使用 librosa.features.pow 得到分帧之后的能量大小，如果末尾能量小于阈值则直接截断	Np.float32 类别数组
Time2batch	Np.float32 类别数组，切分数量，模式旗标	如果模式旗标为 0 则运行不重合切分，如果为 1 则运行 0.5 重合窗长切分	Np.float32 类别矩阵
NMF	Np.complex128 类别矩阵	首先求出模值，然后使用 scikitlearn 进行计算	Mask 矩阵 0-1 二值分布
IVA	Np.complex128 类别矩阵	首先求出模值，然后使用 scikitlearn 进行计算	Mask 矩阵 0-1 二值分布

4.4.1.4 NMF，IVA 实现方式

调用 scikitlearn 中的库文件使用.fit 方法进行计算获取.lable 得到分离掩模然后使用掩模与原数据相乘得到还原语音。这里只负责声明，之后调用时会返回函数的指针，根据函数指针调用函数并传参即可

4.4.1.5 完成数据预处理时的调用方法

此函数负责将数据转化为适合两种算法的不同输入矩阵，在其他函数中调用之后再输入近NMF 或者 IVA 中。在数据处理速度不够好时将函数指针用线程封装起来进行多线程计算。

4.4.2 Interface_main

4.4.2.1 模块功能说明

此模块是整个后端的主函数，负责链接前端和后端，并负责调用其它后端函数主要功能是获取前端想要的操作，执行对应的操作。

4.4.2.2 工作流程

名称	输入	处理	输出
init（self）	None	调用 json_reader 中的 read 函数赋值给本地参数	None
读取文件	前端的返回值	调用 File I/O 函数并传入文件名在执行 O 时也会传入数据	对 File_I/O 的控制 在执行 I 时会返回数组
测试链接	前端的返回值	执行 Socket 类中的 testlink 函数	Linktest 的返回值，包含了 ping 和 packetlossrate
NMF-A	前端返回值	执行 Socket 类中的 getFile 并保存为文件，同时执行 File_IO 中的 readbuffer 函数传入 OPS 中的 NMF 算法	播放语音，同时调用 file_I/O 中的 savefile 所有的恢复语音。
NMF-F	前端返回值	执行 file_I/O 中的 readfile 函数，同时执行 OPS 中非 NMF 算法	播放语音，同时调用 file_I/O 中的 savefile 所有的恢复语音。
IVA-A	前端返回值	执行 Socket 类中的 getFile 并保存为文件，同时执行 File_IO 中的 readbuffer 函数传入 OPS 中的 IVA 算法	播放语音，同时调用 file_I/O 中的 savefile 所有的恢复语音
IVA-F	前端返回值	执行 file_I/O 中的 readfile 函数，同时执行 OPS 中非 IVA 算法	播放语音，同时调用 file_I/O 中的 savefile 所有的恢复语音。
Getjson	前端返回值	执行 json_reader 中的 read 函数	包含了所有配置信息的字典

4.4.2.3 流程图

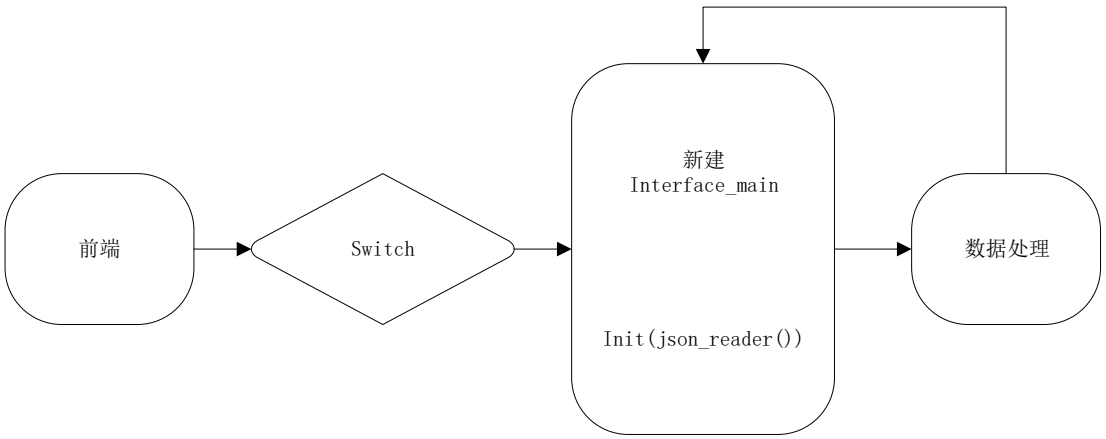

图4-4 interface_main函数

4.4.3 Json_reader 模块

4.4.3.1 模块功能说明

Json_reader 是用来读取配置文件的，后端所有类初始化时都需要调用该模块

4.4.3.2 工作流程

名称	输入	处理	输出
读取	Json 文件位置	使用 json.loads 读取字典	包含所有配置信息的字典
写入	Json 文件位置，参数字典	使用 json.dumps 写入字典	Json 配置文件

4.4.4 Socket 模块

4.4.4.1 模块功能说明

此模块主要是用来实现 PC 端与单片机端实现数据通信，完成 buffer 文件流

的生成。之后调用 File_I/O 进行保存为 buffer，buffer 文件在每次生成之后会自己覆盖自己尽可能不占用储存空间。

4.4.4.2 工作流程

名称	输入	处理	输出
Init()	Json_reader 中的返回值	选取自己需要的字典数值并保存在类中参数包含硬件 ip 地址端口号等信息	None
建立连接	None	通过 python 的 Socket 的 bind 方法进行链接	返回这个链接
传输文件	Socket 输入流	将 IO 流储存为定长大小数组，然后调用 File_io 进行保存，如果传输结束，会将残余数据根据长度进行抛弃，如果比较场则保存为文件继续处理理。注：这里的 Socket 在结束标出现前或者暂停前是不会停止的。但它和其它所有程序都不占用同一个线程，通过文件相互独立的工作。	储存文件
测试链接	类内的 ip 地址	使用 fping 方法获取链接信息	链接信息字典

4.4.4.3 流程图

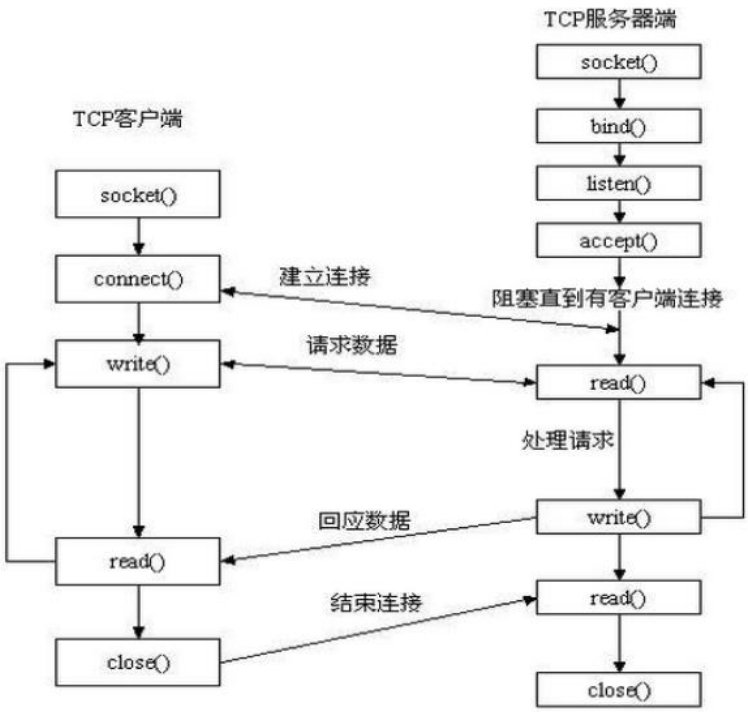

图4-6 层次结构建模流程示意图

4.4.5 File_io 模块

4.4.5.1 模块功能说明

此模块包含五个函数，由外部调用负责储存 buffer 文件，读取 buffer 文件，写入语音文件，读取语音文件，判断文件大小是否能够用于处理，只用来给后台调用。

4.4.5.2 工作流程

名称	输入	处理	输出
init(self)	Json_reader 中返回的字典	选取文件路径/大小等参数进行读取	None
大小判断	此函数仅限于 buffer 文件使用，输入的是 buffer.wav	使用 librosa 读取语音文件，在语音文件不够长的时候返回 false，如果文件足够长返回 True	文件长度是否能够进行处理
文件读取	buffer 和 wav 文件都通过这个函数进行读取	使用 librosa.load 函数将语音文件读取为 numpy 数组	Numpy 数组
Buffer 读取	如果文件储存为 wav 格式不够快的备用方案。读取 npy 文件	使用 numpy.load 直接将流文件转换为数组	Numpy 数组
文件写入	Numpy 数组	使用 librosa.write 函数将 nunpy 数组保存为 wav 文件	Wav 文件
buffer 写入	在文件处理速度不够快的时候的备选方案 。IO 流	将 IO 流直接转换为数组，储存为 numpy 文件并保存	.npy 文件

5. 算法实现流程

5.1 初步实现：

本次产品开发我们准备使用敏捷开发方式，我们首先会构建一个从文件系统读取，执行算法，播放语音的测试模型，考察异步实时播放的可能性以及是否优化算法或者降低采样频率。最初的原型产品测试无误后进入下一步的开发。此时实现的算法有 GCC-NMF GCC-IVA，SEGAN 但不包含传输中的预降噪压缩模块。

5.2 图形界面优化：

之后实现图形界面跳转，功能选择，参数获取等功能在图形界面实现之后进入下一步开发。

5.3 网络通信协议及其进一步优化：

此次开发实现目标为 Udp 协议下的 Sockit 通信将数据直接传输到电脑，进行异步实时处理，根据处理时间长度适当调整采样频率。与硬件结合测试大致产品效果。如果传输连接稳定，能够实现异步传输跳转到下一步。如果不能则在这一步加入预压缩/降噪模块，如果还不能实现异步传输则降低采样频率，使用多线程并行的方式解决问题。完成之后跳转到下一步

5.4 功能及界面的完善：

4.如果步骤 3 没有加入预压缩、降噪模块则这一步加入预压缩降噪模，之后对整个原型产品进行外壳设计，图形界面美化，代码转化为 exe 文件。之后根据用户体验再加入其他模块。

6. 总结：

整个系统的构架为分布式设计，小型计算预处理在单片机上执行，复杂的矩阵计算，语音播放，系统控制分布在计算机端，充分利用计算机的计算能力，并且在前端预先压缩加快传输速率。并且整个算法实现调用外部库文件，系统稳定性强，计算速率快。在之后的项目时间中，我们会将整个系统转向纯神经网络计算，现在神经网络算法高速发展在可并行前提下，神经网络算法不仅在精度还是速度上其实都远远超过了传统算法。而且神经网络是一种分布的近似关系，能够普遍地适应不同环境的噪声和声源。分布式布局这种架构也能够完美地应用在纯神经网络算法中，并且由于神经网络中的参数是无法获取的，这致使通信的数据加密也有保障。之后可能会使用前端进行特征提取以及浅层自动编码，后端并行接受之后还原数据，在多个服务器进行处理实现云计算模式。这种条件下，纯神经网络的优势就会体现出来，可以同时为多个单位进行高性能数据处理，现在 tensorflow，keras，pytorch，caffe 等各种深度学习库都包含了分布式并行计算方法，并且很多大型神经网络项目也是使用了这种分布式或者云计算模式以达到高效计算。