

中科院声学所离线识别引擎

概要设计说明

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V1.0 | 杜一广 |  | 2021-11-05 | 创建文件 |
|  |  |  |  |  |

目录

[1.概述 3](#_Toc30880)

[1.1简介 3](#_Toc29940)

[1.2 读者对象 4](#_Toc14157)

[1.3术语定义 4](#_Toc26105)

[2.总体设计 5](#_Toc26048)

[2.1 总体需求 5](#_Toc15414)

[2.2 运行环境 5](#_Toc3045)

[2.2.1 软件环境 5](#_Toc26196)

[2.2.2 硬件环境 6](#_Toc29311)

[2.3系统总体结构设计 6](#_Toc21445)

[2.3.1 系统结构说明 6](#_Toc22031)

[2.3.2系统总体时序流程 7](#_Toc30353)

[3.能力接口模块设计 8](#_Toc4571)

[4.数据设计 11](#_Toc12477)

[4.1 HTTP接口请求 11](#_Toc3144)

[4.3 HTTP接口返回 14](#_Toc29124)

[5.配置及测试设计 19](#_Toc14052)

[5.1 配置文件设计 20](#_Toc11512)

[5.2 测试功能点 21](#_Toc18383)

[6.可靠性设计 22](#_Toc22469)

[6.1 健壮性设计 22](#_Toc16230)

[6.2 故障恢复 25](#_Toc22312)

[7.扩展性设计 25](#_Toc9315)

[8.安全性设计 27](#_Toc15840)

[9.维护性设计 28](#_Toc29128)

[10.易用性设计 29](#_Toc2338)

# 1.概述

1

**1.1简介**

离线语音识别系统是以语音为处理对象，通过全文转写、场景分割、叠音检测、静音检测、XML服务等技术，让机器把语音转换为相应的文本结果的系统。实现系统具备识别率高、性能稳定、接入方便等特点；离线识别是对音频流进行识别，适用于实时性要求低的场景。

识别引擎：主要是对语音进行连续语音识别也即全文转写，得到连续语音识别的结果（\*\_sent.txt）。一般是作为系统的第一个处理环节。

聚类引擎：将语音识别引擎转写的文本内容进行角色区分，将坐席和客户的内容区别开来，可以给出不同角色的分别时间段，并给出角色的判断。

ITN引擎:主要将语音识别引擎转写的文本内容出现的数字汉字按照一定的规则转换为阿拉伯数字。

标点引擎：主要是语音识别引擎转写的文本内容片段按照一定的规则添加标点。

叠音引擎：主要是检测语音中出现抢插话的具体位置，给出每个叠音段的起止时间点。

语言和方言：目前支持中文普通话；

音频属性：支持较为常见的单/双声道语音；

音频数据长度：离线识别建议每个数据包大小不超过1M。

## 1.2 读者对象

本文档的读者对象为离线识别系统的开发人员、测试人员、系统维护人员及接入识别系统的第三方业务人员，通过本文档能够从总体上了解识别系统的架构形式及数据流向。

本说明给出离线识别系统的设计说明，包括最终实现的系统必须满足的功能、性能、接口、附属测试工具程序及设计约束等。

目的在于：

* 为开发人员提供依据；
* 为代码修改、维护提供条件；

## 1.3术语定义

语音识别（Automatic Speech Recognition）。

# 2.总体设计

2

## 2.1 总体需求

功能需求：满足将客户端语音数据发送至引擎，并将识别结果返回至客户端

接口需求：接口参数简单明了，接口中的每个参数都要有实际意义，保证接口调用流程清晰。

系统验证工具：提供系统完整性，功能正确性的验证工具；

## 2.2 运行环境

### 2.2.1 软件环境

|  |  |  |
| --- | --- | --- |
| 分类 | 名称 | 版本 |
| 操作系统 | Centos | 7.0以上 |
| 数据库 | Redis | 3.2.0 |
| 能力接口 | Tomcat | 9.0.35 |

### 2.2.2 硬件环境

|  |  |  |
| --- | --- | --- |
| 服务器 | 最低配置 | 推荐配置 |
| redis服务器 | CPU：1  内存：256M | CPU：1  内存：1G |
| 识别服务器 | CPU：4  内存：10G  磁盘：20G | CPU：10  内存：15G  磁盘：30G |

## 2.3系统总体结构设计

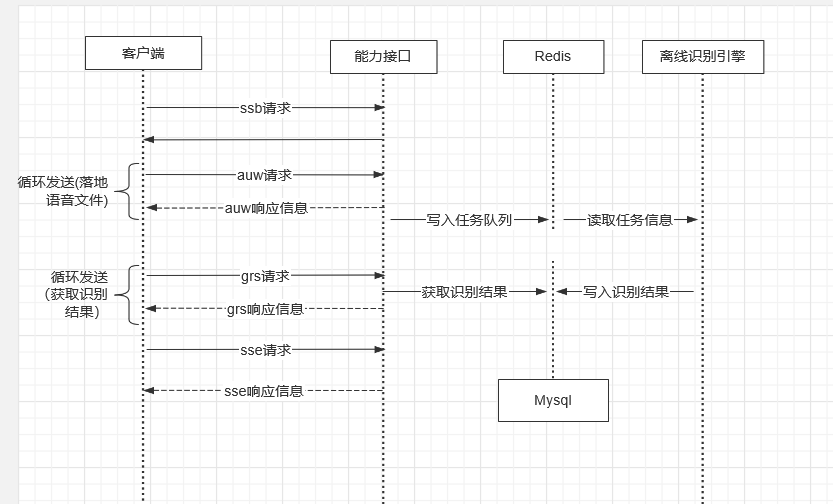
### 2.3.1 系统结构说明

能力接口服务：接收客户端请求，校验数据有效性及落地保存完整语音并将任务信息推送至引擎后，读取识别结果文件并返回至客户端；

### 2.3.2系统总体时序流程

能力接口、redis和识别服务构建生产者-消费者模式，降低了能力接口和识别服务耦合度，极大的提高了识别服务的横向扩展能力。

能力接口时序流程如下所示：

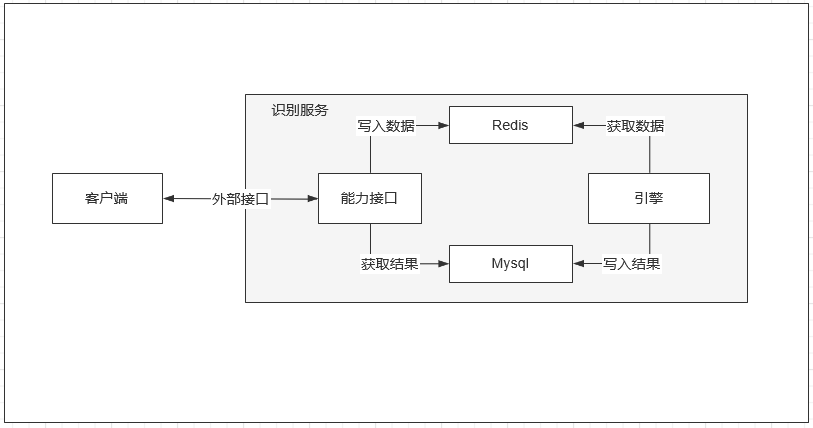
 从上面时序图上看出，能力接口获取语音数据后，落地成完整的语音文件，将新的会话信息存放在redis，空闲的识别服务获取会话信息后，取走相应的语音文件进行识别，识别完成后会将识别结果写入数据库，能力接口通过grs接口查询识别状态并读取识别结果文件返回给客户端。该设计方式，使识别服务的扩容更加便利。

# 3.能力接口模块设计

3

离线识别能力接口通过外部接口与客户端交互，通过Redis与Mysql与引擎进行交互。

模块间关系如下：



**HTTP接口相关函数介绍：**

**public** Object **offline\_stream**(String cmd, String type, String sid, **int** audioStatus, String data, String postfix,String channels,String stereo\_on,String user,String token, **int** urgent)

接口功能：该函数主要通过HTTP协议与客户端进行网络通信的数据交互，接收客户端发送的请求数据，并将识别后的数据返回给客户端。

**内部接口相关介绍：**

Redis接口：将会话信息按照下列格式组装成json字符串，放入redis的List队列，由有空闲线程的引擎取走。

|  |  |  |
| --- | --- | --- |
| 接口参数 | 数据格式 | 参数说明 |
| sid | 字符串 | 请求的唯一标识号 |
| audioName | 字符串 | 保存的语音文件名 |
| audioUrl | 字符串 | 保存语音文件的路径 |
| audioServerIP | 字符串 | 语音文件所在服务器IP |
| channels | 字符串 | 声道数 |
| stereo\_on | 字符串 | 双声道类型 |
| postfix | 字符串 | 语音文件后缀名 |
| isLID | 字符串 | 是否过语种识别引擎 |

Mysql接口：能力接口查询数据库任务会话状态，会话状态为识别完成后，读取对应的识别接口xml文件，通过grs接口返回至客户端，xml格式数据说明如下：



# 4.数据设计

4

## 4.1 HTTP接口请求

|  |  |  |
| --- | --- | --- |
| 接口参数 | 数据格式 | 参数说明 |
| jsonrpc | 字符串 | JsonRpc版本号 固定值字符串“2.0” |
| method | 字符串 | 命令字，固定为“offline\_stream” |
| id | 整数 | JsonRpc规定参数，暂未使用 |
| params | JSON对象 | 方法参数 |
| type | 字符串 | 离线语音识别使用引擎类型 |
| cmd | 字符串 | 命令类型 |
| postfix | 字符串 | 返回结果格式或音频后缀名 |
| user | 字符串 | 鉴权用户名 |
| token | 字符串 | 鉴权密钥 |
| sid | 字符串 | 请求的唯一标识号 |
| data | 字符串 | 音频数据 |
| audioStatus | 整数 | 音频状态 |
| channels | 字符串 | 声道数 |
| stereo\_on | 字符串 | 双声道类型 |
| urgent | 整数 | 任务优先级 |

例如：

|  |
| --- |
| Session begin请求  {  "jsonrpc": "2.0",  "method": "offline\_stream",  "id": 1,  "params": {  "type": "01",  "cmd": "ssb",  "postfix": "sent.txt",  "user": "xxx",  "token": "xxx"  }  } |
| Audio write请求  {  "jsonrpc": "2.0",  "method": "offline\_stream",  "id": 1,  "params": {  "cmd": "auw",  "sid": "4d5660000000",  "type": "1",  "data": "base64编码的音频数据",  "audioStatus": 1,  "postfix": "mp3",  "channels": "0",  "stereo\_on": "4",  "urgent": 1  }  } |
| Get result请求  {  "jsonrpc": "2.0",  "method": "offline\_stream",  "id": 1,  "params": {  "cmd": "grs",  "sid": "4d5660000000",  "type": "1"  }  } |
| Session end请求  {  "jsonrpc": "2.0",  "method": "offline\_stream",  "id": 1,  "params": {  "cmd": "sse",  "sid": "4d5660000000"  }  } |

## 4.3 HTTP接口返回

|  |  |  |
| --- | --- | --- |
| **SSB请求返回** | | |
| 返回参数 | 数据格式 | 参数说明 |
| jsonrpc | 字符串 | JsonRpc版本号 固定值字符串”2.0” |
| id | 整数 | JsonRpc规定参数，暂未使用 |
| result | JSON对象 | 正确返回参数 |
| ret | 整数 | 接口返回值，0表示正常返回 |
| sid | 字符串 | 请求的唯一标识号 |
| 返回示例:  {  "jsonrpc": "2.0",  "id": 1,  "result": {  "ret": 0,  "sid": "4d5660000000"  }  } | | |

|  |  |  |
| --- | --- | --- |
| **AUW请求返回** | | |
| 返回参数 | 数据格式 | 参数说明 |
| jsonrpc | 字符串 | JsonRpc版本号 固定值字符串”2.0” |
| id | 整数 | JsonRpc规定参数，暂未使用 |
| result | JSON对象 | 正确返回参数 |
| msg | 字符串 | 返回消息内容 |
| 返回示例:  {  "jsonrpc": "2.0",  "id": 1,  "result": {  "msg": "data is accepted!"  }  } | | |

|  |  |  |
| --- | --- | --- |
| **GRS请求返回** | | |
| 返回参数 | 数据格式 | 参数说明 |
| jsonrpc | 字符串 | JsonRpc版本号 固定值字符串”2.0” |
| id | 整数 | JsonRpc规定参数，暂未使用 |
| result | JSON对象 | 正确返回参数 |
| ret | 整数 | 接口返回值，0表示正常返回 |
| sid | 字符串 | 请求的唯一标识号 |
| recStatus | 整数 | 离线语音识别结果状态:  2：正在获取识别结果  5：整个会话识别结果获取完成 |
| result | 字符串 | 识别结果 |
| 返回示例:  {  "jsonrpc": "2.0",  "id": 1,  "result": {  "ret": 0,  "sid": "4d5660000000",  "recStatus": 5,  "result": "xml文件内容"  }  } | | |

|  |  |  |
| --- | --- | --- |
| **SSE请求返回** | | |
| 返回参数 | 数据格式 | 参数说明 |
| jsonrpc | 字符串 | JsonRpc版本号 固定值字符串”2.0” |
| id | 整数 | JsonRpc规定参数，暂未使用 |
| result | JSON对象 | 正确返回参数 |
| ret | 整数 | 接口返回值，0表示正常返回 |
| sid | 字符串 | 请求的唯一标识号 |
| 返回示例:  {  "jsonrpc": "2.0",  "id": 1,  "result": {  "ret": 0,  "sid": "4d5660000000"  }  } | | |

|  |  |  |
| --- | --- | --- |
| **错误请求返回** | | |
| 返回参数 | 数据格式 | 参数说明 |
| jsonrpc | 字符串 | JsonRpc版本号 固定值字符串”2.0” |
| id | 整数 | JsonRpc规定参数，暂未使用 |
| error | JSON对象 | 错误返回参数 |
| code | 整数 | jsonRpc错误码 |
| message | 字符串 | jsonRpc 错误消息 |
| data | JSON对象 | 错误返回参数 |
| errno | 整数 | 错误码，见接口文档 |
| errmsg | 字符串 | 错误说明 |
| 返回示例:  {  "jsonrpc": "2.0",  "id": 1,  "error": {  "code": -34000,  "message": "server error",  "data": {  "errno": -32010,  "errmsg": "params cmd must be required!"  }  }  } | | |

# 5.配置及测试设计

5

## 5.1 配置文件设计

设置配置文件，方便使用不同控制逻辑，支持不同的功能，能力接口服务配置文件实现如下目的：

* 通过参数控制支持特殊功能
* 日志输出
* 数据库连接
* 服务地址配置
* 引擎交互时间控制

配置文件主要参数如下表：

|  |  |
| --- | --- |
| 参数 | 说明 |
| redis.queue | Redis中会话队列key值 |
| redisCluster | Redis集群信息 |
| redisSingle | Redis单节点信息 |
| isPassword | Redis是否设置密码 |
| isCluster | Redis是否使用集群 |
| redisType | Redis密码是否加密 |
| redisPassword | Redis密码 |
| redisPassKeys | Redis加密密钥 |
| redisUrgent | 优先级急别 |
| audio.ServerIP | 本机IP |
| grsSleepTime | Grs重复查询间隔时间 |
| voiceTempPath | 语音落地路径 |
| url | 数据库地址 |
| username | 数据库用户名 |
| password | 数据库密码 |
| log4j.rootLogge | 日志输出级别 |
| log4j.appender.logfile.File | 日志输出文件路径 |

## 5.2 测试功能点

根据设计需求，对服务必须满足的功能点进行验证测试。

|  |  |
| --- | --- |
| 测试功能 | 期望结果 |
| 服务启动验证 | 正确配置服务参数，启动服务，CPU、内存满足情况下，服务能够正常启动。 |
| 语音格式 | 参数传递正确情况下，输出识别结果。 |
| 保存语音 | 保存语音正常。 |
| 保存识别结果 | 配置保存识别结果参数，保存识别结果正常。 |
| 关闭保存识别结果 | 配置不保存识别结果参数，不保存识别结果 |
| 配置不同返回结果 | 返回不同的结果形式 |
| 日志级别 | 调整不同的日志级别，日志输出内容不同 |
| 日志文件大小 | 调整日志文件大小，保存日志文件大小不同 |

# 6.可靠性设计

6

## 6.1 健壮性设计

服务运行过程中，客户端发送过来的异常数据及异常会话可能导致输出结果错误、服务崩溃等问题，为避免此类问题软件设计中有如下措施：

* 检查输入数据的数据类型,防止异常数据引起输出结果错误。
* 模块调用时检查参数的合法性,对非法参数请求返回错误信息提示。
* 简化软件的功能复杂性,与服务无关功能禁止加在服务上。
* 网络异常超时，返回对应错误信息。
* 识别超时等内部异常错误，返回错误码信息

如下是部分错误码信息：

|  |  |
| --- | --- |
| 错误码 | 说明 |
| -32010 | 缺少CMD参数或CMD参数为空 |
| -32011 | 缺少CHANNELS参数或CHANNELS参数为空 |
| -32012 | 缺少STEREO\_ON参数或STEREO\_ON参数为空 |
| -32013 | 连接REDIS失败 |
| -32014 | DATA数据为空 |
| -32015 | 缺少SID参数或SID参数为空 |
| -32016 | Redis队列内缓存数量超限 |
| -32017 | SID不存在,请发送SSB获取SID |
| -32018 | AUW已发送结束状态标识，请勿继续发送语音 |
| -32019 | Type参数值异常 |
| -32020 | 获取返回文件类型失败 |
| -32024 | 操作线程数参数异常 |
| -32025 | 获取引擎列表失败 |
| -32026 | 获取引擎信息失败 |
| -32027 | 设置引擎线程数超过最大值 |
| -32028 | 设置引擎线程数失败 |
| -32029 | AUW已经结束 |
| -32030 | USER或者TOKEN参数不存在或者为空 |
| -32031 | USER与TOKEN不匹配 |
| -32032 | USER超过并发限制 |
| -32033 | USER超过通讯量限制 |
| -32034 | USER获取并发量失败 |
| -32035 | USER获取通讯量失败 |
| -32036 | 找不到语音文件 |
| -32037 | 找不到XML文件 |
| -32038 | 语音文件处理失败 |
| -32039 | 无法识别语音处理状态 |
| -32040 | 语音文件推送失败 |
| -32041 | branch或者urgent参数值异常 |
| -32042 | 语音太短 |
| -32043 | 异常语音不能处理 |

## 6.2 故障恢复

服务运行中异常情况导致的服务重启，识别服务需要不在人为干预的情况下自动重启。

# 7.扩展性设计

7

会话通过生产者消费者模式处理，降低能力接口和识别服务的耦合度，能力接口接收到会话后，将会话信息存放redis，识别服务端有空闲线程时，主动从redis队列中获取会话信息，通过redis队列减弱能力接口和识别服务的耦合关系，实现能力接口和识别服务的多对多关系。

如下图所示：



能力接口和识别服务的低耦合度，可以使新增接点的识别服务方便的实现横向扩展，满足后期服务的大并发需求。

# 8.安全性设计

8

从客户端发送的语音数据需要进行base64加密处理，同时从服务端返回的响应数据也进行base64加密，在传输的过程中一定程度上保证数据的安全性。

# 9.维护性设计

9

能力接口和识别服务之间通过redis建立对应关系，能力接口、redis、识别服务三者构成的生产者消费者模式，使新的识别服务接入识别系统时，不需要关闭识别系统，从而实现识别服务的横向扩容。

识别服务的参数配置在启动服务时默认使用最优配置，如果机器环境（如cpu、内存）发送变化时，根据需要停止服务重新配置。

# 10.易用性设计

10

代码编译：在代码工程指定目录下执行编译脚本即可生成可执行文件；将可执行文件替换到部署系统的指定目录下即可实现系统更新；

系统部署：提供完整的文档介绍，用户可以根据文档说明完成系统部署启动；

系统调用：系统提供了调用demo，用户可以先使用demo测试验证系统的正确性，然后可以根据接口文档和demo完成接口集成调用。