

中科院声学所在线识别引擎

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 |  | 2021-11-02 | 创建文件 |
|  |  |  |  |  |

目录

[1.概述 3](#_Toc21806)

[1.1简介 3](#_Toc2556)

[1.2 读者对象 4](#_Toc31105)

[1.3术语定义 4](#_Toc5971)

[2.总体设计 5](#_Toc30252)

[2.1 总体需求 5](#_Toc30821)

[2.2 运行环境 6](#_Toc32347)

[2.2.1 软件环境 6](#_Toc19403)

[2.2.2 硬件环境 6](#_Toc30197)

[2.3系统总体结构设计 7](#_Toc28340)

[2.3.1 系统结构说明 7](#_Toc21171)

[2.3.2系统总体时序流程 7](#_Toc24629)

[3.能力接口模块设计 10](#_Toc24228)

[4.数据设计 11](#_Toc13348)

[4.1 HTTP接口请求 12](#_Toc12327)

[4.3 HTTP接口返回 14](#_Toc14791)

[5.配置及测试设计 23](#_Toc2657)

[5.1 配置文件设计 23](#_Toc23761)

[5.2 测试功能点 24](#_Toc12985)

[6.可靠性设计 26](#_Toc15243)

[6.1 健壮性设计 26](#_Toc27363)

[6.2 故障恢复 28](#_Toc804)

[7.扩展性设计 28](#_Toc31862)

[8.安全性设计 30](#_Toc27501)

[9.维护性设计 31](#_Toc27186)

[10.易用性设计 32](#_Toc13714)

# 1.概述

1

**1.1简介**

在线语音识别系统（Automatic Speech Recognition）是以语音为处理对象，通过语音信号处理技术，让机器把语音转换为相应的文本的系统。实现系统具备识别率高、性能稳定、接入方便等特点；支持一句话识别和实时识别两种服务，能够满足不同业务类型需求。

实时识别是对音频流进行识别，适用于实时性要求的场景。

一句话识别是对20s内的短语音进行识别，适用于短语音转写的场景。

语言和方言：目前支持中文普通话和带有一定方言的中文普通话；

音频属性：支持wav、pcm引擎格式，支持8k、16k采样率的单声道语音；

音频数据长度：实时识别建议每个数据包为1600字节，一句话识别建议每包大小不超过100k。

## 1.2 读者对象

本文档的读者对象为在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。

本说明给出在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，并将处理结果返回至客户端；

性能需求：语音识别出对应文本内容延时不超过200ms；

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，校验数据有效性及封装成识别服务需要的数据格式；

### 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：

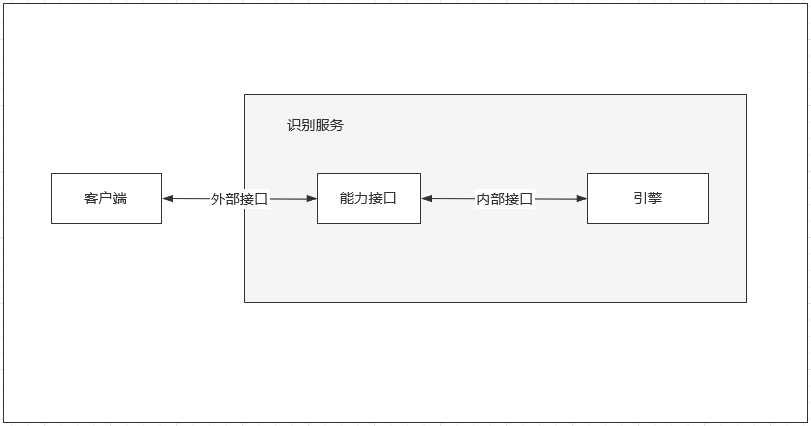
 从上面时序图上看出，能力接口将新的会话信息存放在redis，空闲的识别服务获取会话信息后，将本身信息写入redis，能力接口根据会话Id从redis中读取识别服务信息，之后能力接口和识别服务才建立一对一的连接。该设计方式，使识别服务的扩容更加便利。

# 3.能力接口模块设计

3

在线识别能力接口通过外部接口与客户端交互，通过内部接口与引擎进行交互。

模块间关系如下：



**外部接口相关函数介绍：**

**public** Object **deal\_request**(String type, String cmd, String auf, String aue, String sid, String data, String syncid, String machinfo, String threadnum, String hotword,String user,String token)

函数功能：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。

Tsokcet接口：与引擎交互，auw发送语音数据，grs获取识别结果，sse结束会话。

# 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“deal\_request” |
| id | 整数 | JsonRpc规定参数，暂未使用 |
| params | JSON对象 | 方法参数 |
| type | 字符串 | 在线语音识别使用引擎类型 |
| cmd | 字符串 | 命令类型 |
| auf | 字符串 | 音频采样 |
| aue | 字符串 | 音频格式 |
| sid | 字符串 | 请求的唯一标识号 |
| data | 字符串 | 音频数据 |
| syncid | 字符串 | 同步标识 |
| machinfo | 字符串 | 引擎地址 |
| threadnum | 字符串 | 要设置的线程数 |
| hotword | 字符串 | 热词数据 |
| user | 字符串 | 鉴权用户名 |
| token | 字符串 | 鉴权密钥 |

例如：

|  |
| --- |
| Session begin请求  {  "jsonrpc": "2.0",  "method": "deal\_request",  "id": 1,  "params": {  "type": "01",  "cmd": "ssb",  "auf": "audio/L16;rate=8000",  "aue": "raw",  "user": "xxxx",  "token": "xxxx",  }  } |
| Audio write请求  {  "jsonrpc": "2.0",  "method": "deal\_request",  "id": 1,  "params": {  "cmd": "auw",  "sid": "4d5660000000",  "data": "base64编码的音频数据",  "hotword": "base64编码的热词数据",  "syncid": "1"  }  } |
| Get result请求  {  "jsonrpc": "2.0",  "method": "deal\_request",  "id": 1,  "params": {  "cmd": "grs",  "sid": "4d5660000000",  "syncid":"n"  }  } |
| Session end请求  {  "jsonrpc": "2.0",  "method": "deal\_request",  "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 |
| 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 | 引擎识别结果状态  0 , 中间识别结果  2：正在获取识别结果 |
| result | JSON数组 | 识别结果列表:(中间识别结果，无结果时为空)  text:识别结果  phoneme:分词  segtime:分词时间  score:置信度 |
| syncid | 字符串 | n，对应请求包的同步标识 |
| engine\_name | 字符串 | 引擎IP |
| bg | 整数 | 起点标识(0:未检测到起点,1:起点) |
| ed | 整数 | 尾点标识(0:未检测到尾点,1:尾点) |
| 返回示例:  {  "jsonrpc": "2.0",  "id": 1,  "result": {  "ret":0,  "sid":"4d5660000000",  "recStatus":2,  "result":[{  "text": "天气很好啊。",  "phoneme": "天 气 很 好 啊。",  "segtime": "2.12 2.20 2.20 2.37 2.37 2.57 2.57 2.72 2.72 2.88",  "score": 93.02638244628906  }, {  "text": "天气很好啊。",  "phoneme": "天 气 很 好 啊。",  "segtime": "2.12 2.20 2.20 2.37 2.37 2.57 2.57 2.72 2.72 2.88",  "score": 93.02638244628906  }],  "syncid": "n",  "engine\_name":"127.0.0.1",  "bg":1,  "ed":0  }  } | | |

|  |  |  |
| --- | --- | --- |
| **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 | 在线语音识别结果状态:  2：正在获取识别结果  5：最终识别结果 |
| result | JSON数组 | 在线语音识别结果:(无结果时为空数组)  text:识别结果  phoneme:分词  segtime:分词时间  score:置信度 |
| syncid | 字符串 | n，对应请求包的同步标识 |
| engine\_name | 字符串 | 引擎IP |
| bg | 整数 | 起点标识(0:未检测到起点,1:起点) |
| ed | 整数 | 尾点标识(0:未检测到尾点,1:尾点) |
| 返回示例:  {  "jsonrpc": "2.0",  "id": 1,  "result": {  "ret": 0,  "sid": "4d5660000000",  "recStatus": 5,  "result": [{  "text": "天气很好啊。",  "phoneme": "天 气 很 好 啊。",  "segtime": "2.12 2.20 2.20 2.37 2.37 2.57 2.57 2.72 2.72 2.88",  "score": 93.02638244628906  }, {  "text": "天气很好啊。",  "phoneme": "天 气 很 好 啊。",  "segtime": "2.12 2.20 2.20 2.37 2.37 2.57 2.57 2.72 2.72 2.88",  "score": 93.02638244628906  }],  "syncid": "n",  "engine\_name":"127.0.0.1",  "bg":1,  "ed":0  }  } | | |

|  |  |  |
| --- | --- | --- |
| **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 |
| ASR\_REDIS\_SINGLE | Redis单节点信息 |
| ASR\_REDISSETLIST | Redis中会话队列key值 |
| ASR\_REDISADD | Redis集群信息 |
| ASR\_INCLUDEPASSWORD | Redis是否设置密码 |
| ASR\_REDISPASSWORD | Redis密码 |
| ASR\_CLUSTER | Redis是否使用集群 |
| ASR\_TSOCKETTIM | 能力接口与引擎交互超时时间 |
| ASR\_HTTPHEADSWITCH | 会话信息是否存储数据库 |
| ASR\_ISREALTIME | 是否为实时识别 |
| log4j.rootLogger | 日志输出级别 |
| log4j.appender.MyRfile1.File | 日志输出文件路径 |
| url | 数据库地址 |
| username | 数据库用户名 |
| password | 数据库密码 |

## 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。 |
| 8k16bit pcm语音 | 参数传递正确情况下，输出识别结果。 |
| 16k16bit pcm语音 | 参数传递正确情况下，输出识别结果。 |
| 保存语音 | 配置保存语音参数，保存语音正常。 |
| 关闭保存语音 | 配置不保存语音参数，不保存语音。 |
| 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 |
| 检查数据包序号 | 配置检查数据包序号，数据包乱序返回错误信息 |
| 不检查数据包序号 | 配置不检查数据包序号，数据包乱序，不返回错误信息 |
| 单句识别 | 参数传递正确情况下，输出识别结果。 |
| 实时识别 | 参数传递正确情况下，输出识别结果。 |
| 实时识别，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 |

## 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。