

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 |  | 2021-11-10 | 创建文件 |
|  |  |  |  |  |

目录

[1.概述 3](#_Toc20760)

[1.1简介 3](#_Toc1184)

[1.2 读者对象 4](#_Toc4644)

[1.3术语定义 4](#_Toc30902)

[2.总体设计 5](#_Toc16722)

[2.1 总体需求 5](#_Toc13682)

[2.2 运行环境 6](#_Toc26214)

[2.2.1 软件环境 6](#_Toc1536)

[2.2.2 硬件环境 6](#_Toc16516)

[2.3系统总体结构设计 7](#_Toc29425)

[2.3.1 系统结构说明 7](#_Toc27635)

[2.3.2系统总体时序流程 7](#_Toc28035)

[3.能力接口模块设计 9](#_Toc2751)

[4.数据设计 13](#_Toc22065)

[4.1 WebSocket接口请求 13](#_Toc12182)

[4.1.1 开始帧 13](#_Toc16555)

[4.1.2 音频数据帧 14](#_Toc1103)

[4.1.3 结束帧 14](#_Toc10408)

[4.1.4 接收帧 15](#_Toc12803)

[5.配置及测试设计 17](#_Toc9774)

[5.1 配置文件设计 17](#_Toc32421)

[5.2 测试功能点 18](#_Toc25930)

[6.可靠性设计 20](#_Toc29804)

[6.1 健壮性设计 20](#_Toc1098)

[6.2 故障恢复 21](#_Toc18837)

[7.扩展性设计 22](#_Toc21746)

[8.安全性设计 24](#_Toc13135)

[9.维护性设计 25](#_Toc8270)

[10.易用性设计 26](#_Toc25794)

# 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

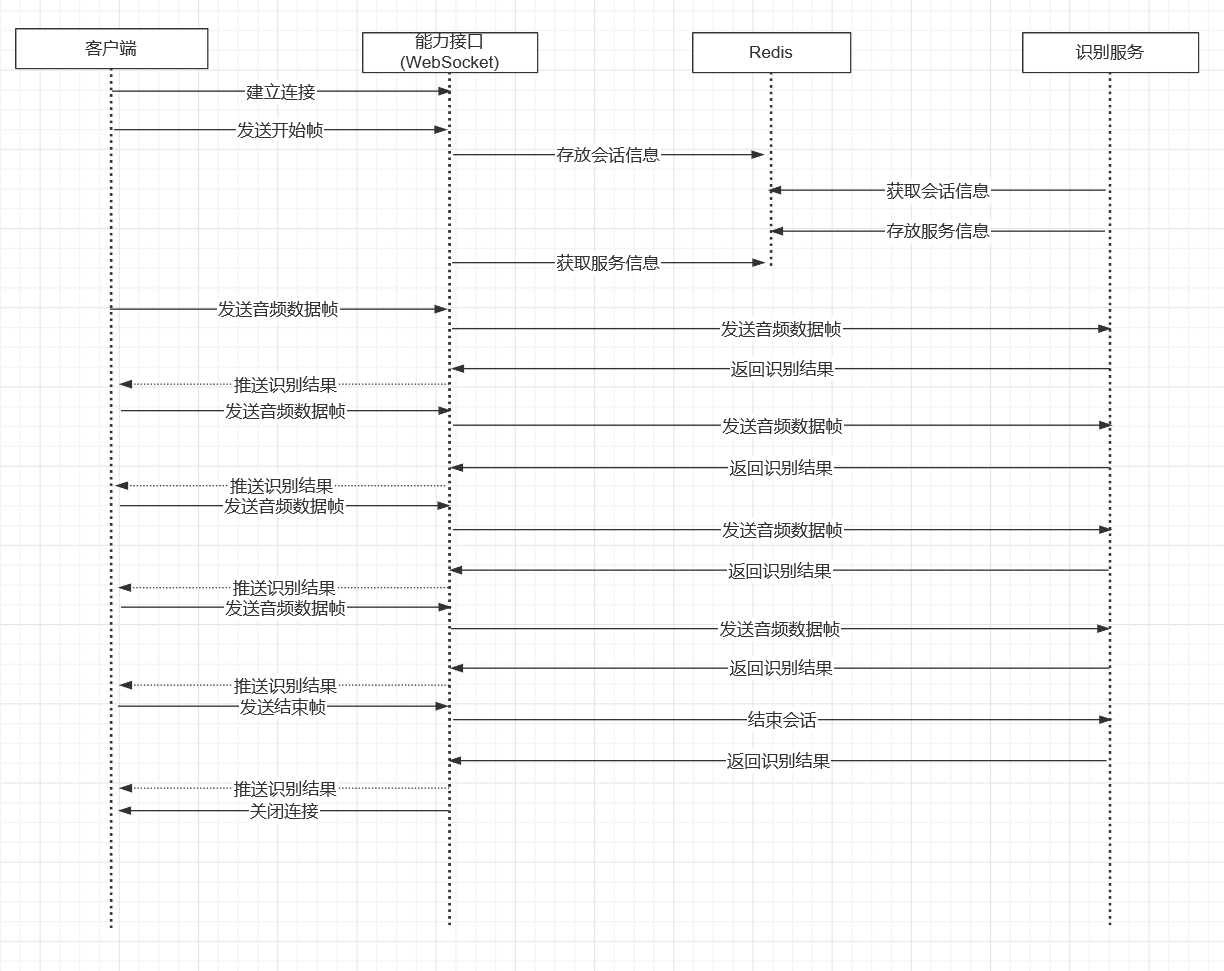
### 2.3.1 系统结构说明

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：

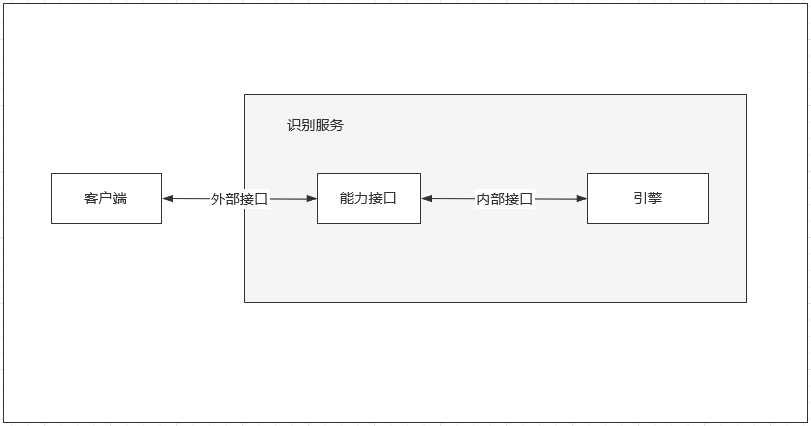
 从上面时序图上看出，能力接口将新的会话信息存放在redis，空闲的识别服务获取会话信息后，将本身信息写入redis，能力接口根据会话Id从redis中读取识别服务信息，之后能力接口和识别服务才建立一对一的连接。该设计方式，使识别服务的扩容更加便利。

# 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：

该接口主要通过WebSocket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，并校验引擎是否返回识别结果，当有识别结果时，将结果推送给客户端。

**public** **void** **onOpen** (Session session);

建立连接：客户端与服务端进行握手操作。

**public** **void** **onClose**(Session session,String sid);

关闭连接：服务端主动与客户端断开连接，一般发生在结束帧之后。

**public** **void** **onMessage**(**byte**[] bytes,Session session);

音频数据帧：客户端发送音频数据至服务端，服务端通过内部接口(TSocket)与引擎交互，获取结果，校验组装后通过外部接口(sendMessage)至客户端。

**public** **void** **onMessage**(String msg,Session session);

开始帧：客户端发送开始帧到服务端，服务端通过内部接口(Redis)与引擎交互，建立会话。

结束帧：客户端发送结束帧到服务端，服务端通过内部接口(TSocket)与引擎交互，并调用外部接口(onClose)主动与客户端断开连接。

**public** **void** **onError**(Session session, Throwable error);

异常：若连接发生异常，将通过外部接口(onError)通知服务端。

**public** **void** **sendMessage**(String message,Session session,String sid);

推送数据：服务端通过内部接口(TSocket)与引擎交互后，若有识别结果，通过此接口将数据推送至客户端。

内部接口相关介绍：

能力接口将客户端发送的请求数据经过处理，发送至在线识别引擎,并解析在线识别结果后，组装成相应的格式返回给客户端。

Redis接口：

建立会话，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 |
| request | JSON字符串 |  |
| 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 |

TSocket接口：

负责与引擎交互，数据帧发送数据/获取结果，结束帧关闭会话。

|  |  |  |
| --- | --- | --- |
| **数据帧请求** | | |
| 接口参数 | 数据格式 | 参数说明 |
| request | JSON字符串 |  |
| cmd | 字符串 | 命令类型 |
| syncid | 字符串 | 同步标识 |
| sid | 字符串 | 请求的唯一标识号 |
| data | 字符串 | 音频数据 |

|  |  |  |
| --- | --- | --- |
| **结束帧请求** | | |
| 接口参数 | 数据格式 | 参数说明 |
| request | JSON字符串 |  |
| cmd | 字符串 | 命令类型 |
| sid | 字符串 | 请求的唯一标识号 |

# 4.数据设计

4

## 4.1 WebSocket接口请求

4.1.1 开始帧

WebSocket连接成功后，发送开始帧开始语音识别，帧的类型（Opcode） 是Text， 使用json序列化。

开始帧参数如下：

| 字段名 | 数据类型 | 必选项 | 描述 |
| --- | --- | --- | --- |
| action | 字符串 | 是 | START，开始帧的类型 |
| type | 字符串 | 是 | 识别语音类型  1:普通话 |
| auf | 字符串 | 是 | 音频采样，支持如下格式：  audio/L16;rate=8000  audio/L16;rate=16000 |

4.1.2 音频数据帧

开始帧发送完成后，发送音频数据帧，帧的类型（Opcode）是Binary，内容是二进制的音频内容。除最后一个音频数据帧，每个帧的音频数 据长度建议为100ms，音频数据采用网络字节序发送。

注意：

1.建议音频流每100ms发送1600（8k16bit）/3200(16k16bit)字节；

4.1.3 结束帧

最后一个音频数据帧发送结束后，发送结束帧， 帧的类型（Opcode） 是Text， 使用json序列化。

结束帧参数如下：

| 字段名 | 数据类型 | 必选项 | 描述 |
| --- | --- | --- | --- |
| action | 字符串 | 是 | STOP，结束帧的类型 |

4.1.4 接收帧

语音识别服务对音频数据帧进行实时识别，返回文本识别结果，帧的类型（Opcode）是Text， 使用json序列化。

接收结果帧参数如下（成功）：

| **参数** | **类型** | **说明** |
| --- | --- | --- |
| action | string | 结果标识，started:开始帧，result:结果，error:异常 |
| code | string | 结果码(具体见[错误码](https://www.xfyun.cn/doc/asr/rtasr/API.html" \l "%E9%94%99%E8%AF%AF%E7%A0%81)) |
| data | string | 结果数据 |
| desc | string | 描述 |
| sid | string | 会话ID |

其中sid字段主要用于DEBUG追查问题，如果出现问题，可以提供sid帮助确认问题。

结果数据（data）

| **字段** | **含义** | **描述** |
| --- | --- | --- |
| bg | 句子开始时间 | 前端点帧数 |
| ed | 句子结束时间 | 后端点帧数 |
| engine\_name | 备用字段 | 备用字段 |
| epStatus | 引擎端点状态 | 1：表示已找到前端点2：表示已找到后端点 |
| pags | 识别结果状态 | 0：表示中间结果1：表示最终结果 |
| phoneme | 分词 | 分词 |
| time | 分词时间 | 分词时间 |
| recStatus | 引擎识别结果状态 | 0:中间识别结果；5-最终识别结果 |
| result | 识别结果 | 识别结果 |
| status | 引擎状态备用字段 | 目前同recstatus |
| syncid | 同步标识 | 包序 |

# 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 |
| Server.port | WebSocket端口 |
| Asr.redisSetList | Redis中会话队列key值 |
| Asr.redisIsCluster | Redis是否使用集群 |
| Asr.tScoketTimeOut | 能力接口与引擎交互超时时间 |
| Asr.retAuwResult | 是否返回中间结果 |
| Spring.redis.cache.nodes | Redis集群信息 |
| Spring.redis.cache.host | Redis单节点信息 |
| Spring.redis.cache.password | Redis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 |
| 200 | 成功 |
| 32012 | 从redis获取数据失败 |
| 32013 | 连接引擎失败 |
| 32014 | 获取引擎ip失败 |
| 32015 | 引擎返回值为空 |
| 32016 | 识别结果转换异常 |
| 32017 | Auw解析xml失败 |
| 32018 | Grs解析xml失败 |
| 32019 | 引擎返回结果格式异常 |
| 32020 | Type参数不是START或者STOP,请检查 |

## 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。