

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

目录

[1.概述 3](#_Toc8656)

[1.1简介 3](#_Toc17312)

[1.2 读者对象 3](#_Toc27687)

[1.3术语定义 4](#_Toc9753)

[2.总体设计 5](#_Toc1378)

[2.1 总体需求 5](#_Toc2333)

[2.2 运行环境 6](#_Toc32342)

[2.2.1 软件环境 6](#_Toc11744)

[2.2.2 硬件环境 6](#_Toc85)

[2.3系统总体结构设计 7](#_Toc31145)

[3.模块设计 9](#_Toc16323)

[3.1任务处理 9](#_Toc2722)

[3.1.1转写语音 10](#_Toc30764)

[3.2 结果发送 12](#_Toc5129)

[3.3 任务接收 14](#_Toc26160)

[4.配置及测试设计 15](#_Toc17647)

[4.1 配置文件设计 15](#_Toc5577)

[4.2 测试功能点 16](#_Toc21993)

[5.可靠性设计 17](#_Toc20398)

[6.扩展性设计 18](#_Toc32101)

[7.维护性设计 19](#_Toc1761)

[8.易用性设计 20](#_Toc18299)

# 1.概述

1

**1.1简介**

识别模块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）。

## 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），简称（ASR）。

# 2.总体设计

2

## 2.1 总体需求

功能需求：把语音转换为连续的文字存入sent中间结果文件；

性能需求：根据机器性能开不同线数；

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

识别部分主要包括一个任务接收线程，一个结果发送线程，一个任务处理线程组成。



任务接收线程：完成接收总控端发送的任务，存入任务队列。

任务处理线程：从任务队列中取出待处理任务，经过调用TBNR相关函数，把语音转写为连续的文字，按照一定格式写入sent中间结果文件。处理结果存入结果队列。

任务发送线程:从结果队列中取已经处理完的任务，发送到总控端。

# 3.模块设计

3

## 3.1任务处理



### 3.1.1转写语音

* **Recognize**()

函数原型：

int Recognize(int SessNum,char\* line);

参数说明：

int SessNum[in] 识别线程编号

char\* line[in] 待识别的语音的完全路径

返回值：

int 类型，0：成功，<0：失败。具体取值及说明如下：

0：识别成功。

-1：无法打开语音文件

-2：语音文件大小为0。

-3：TBNR\_Start函数返回错误。

功能：

该函数主要调用TBNR引擎的相关接口来完成一条音频文件的识别，并将结果写入相应的结果文件中（\*\_sent.txt）。

函数流程图：



Recoginize()函数流程图

## 3.2 结果发送

结果发送线程主要是从结果队列中获取任务后，发送处理结果给总控端。

结果发送线程对应线程函数为RestaskSendThreadProc(),流程图如下：



RestaskSendThreadProc()函数流程图

## 3.3 任务接收

任务接收线程主要是接收总控端发送的任务，并将接收到的内容放入任务队列。

任务接收线程对应线程函数为TaskInsertThreadProc(),流程图如下：



TaskInsertThreadProc()函数流程图

# 4.配置及测试设计

4

## 4.1 配置文件设计

设置配置文件，方便使用不同控制逻辑，支持不同的功能及结果输出，识别配置文件实现如下目的：

* 通过参数控制支持特殊功能
* 检测异常数据
* 日志输出
* 引擎资源加载

配置文件主要参数如下表：

|  |  |  |
| --- | --- | --- |
| 参数 | 取值 | 说明 |
| isSaveData | true:false | 是否保存vad切分语音 |
| LogFileName | 字符串 | 日志文件名 |

## 4.2 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。 |
| 保存vad切分语音 | 参数配置保存vad切分语音，能正确保存vad 切分语音。 |
| 输出日志 | 打开日志参数，正确输出日志。 |

# 5.可靠性设计

5

离线系统识别部分设计考虑了实际应用场景的复杂性和输入的多样性的情况，内部加入了大部分实际情况中会出现的异常情况的处理方案，socket连接异常断开重连、语音文件有效性检查等异常事件的处理；并且整套系统测试按照严格的7\*24小时压力测试标准进行测试，同时也针对所有异常数据进行异常情况测试，保证引擎的稳定性和可靠性。

# 6.扩展性设计

6

服务端处理引擎支持多路并发，保证了不同cpu数和内存大小的机器拓展需求；系统本身遵循模块化设计，可以随时添加定制化功能，满足不同应用场景的需要。

# 7.维护性设计

7

离线系统为了方便后期的引擎维护，采用标准c/c++函数同时在较低编译器上进行编译。

识别模块可配置WFSTDecoder\_onlyrec.cfg文件部分参数，保证了引擎的可配性和可维护性。常见参数如下：

isSaveData=false 是否保存vad切分语音

LogLevel=0 底层引擎日志参数

key\_wfst=../hotword/TLG.rescore.bin 热词模型

# 8.易用性设计

8

系统安装部署时有一键安装部署脚本，只需要执行脚本即可安装引擎转写系统及所需的中间件（如ffmpeg、sox、redis等），不需要安装人员逐个安装，减少操作量。系统启停均有对应的脚本，实现自动启动、停止服务。