

中科院声学所离线识别引擎

XML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](#_Toc19847)

[1.1简介 3](#_Toc31191)

[1.2 读者对象 3](#_Toc2084)

[1.3术语定义 4](#_Toc5686)

[2.总体设计 5](#_Toc10500)

[2.1 总体需求 5](#_Toc16828)

[2.2 运行环境 6](#_Toc11162)

[2.2.1 软件环境 6](#_Toc7067)

[2.2.2 硬件环境 6](#_Toc9653)

[2.3系统总体结构设计 7](#_Toc4164)

[3.模块设计 9](#_Toc12402)

[3.1任务处理 10](#_Toc260)

[3.1.1构造CResult2XML对象 10](#_Toc18926)

[3.2.2 中间结果文件整理 12](#_Toc5503)

[3.2.3 构造xml 结构体 16](#_Toc32076)

[3.2.4 生成xml 文件 17](#_Toc28186)

[3.2 结果发送 18](#_Toc19456)

[3.3 任务接收 20](#_Toc26255)

[4.配置及测试设计 21](#_Toc27277)

[4.1 配置文件设计 21](#_Toc21055)

[4.2 测试功能点 22](#_Toc13333)

[5.可靠性设计 24](#_Toc10241)

[6.扩展性设计 25](#_Toc4962)

[7.维护性设计 26](#_Toc383)

[8.易用性设计 27](#_Toc21851)

# 1.概述

1

**1.1简介**

XML服务端主要功能，基于之前的识别、场景分割、叠音服务端、ITN、标点服务端产生的中间结果文件进行整合处理，并进行静音检测、叠音角色判断、语速计算、能量计算、语速计算、关键字检索等功能，最终生成可被查询、检索、分析结构化的xml文件。

## 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：根据各模块中间文件组装xml文件；

性能需求：组装xml内容不崩溃；

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

Xml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，经过生成CResult2XML对象、中间文件数据整理、构造XML相关结构体、生成xml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构造CResult2XML对象

CResult2XML构造函数完成成员变量赋值。

CResult2XML::CResult2XML(TaskInfo\* pTaskInfo, Result2XMLPara\* pStResult2XMLPara, StPunctuationRule\* pStPunctuationRule)

{

string audioName =pTaskInfo->audioname();

string audioUrl =pTaskInfo->audiourl();

string resultfilePath =pTaskInfo->resultfilepath();

string XMLfilePath =pTaskInfo->xmlfilepath();

channels = pTaskInfo->channels();

LOG\_PRINT\_INFO("audioName: %s",audioName.c\_str());

LOG\_PRINT\_INFO("audioUrl: %s",audioUrl.c\_str());

LOG\_PRINT\_INFO("resultfilePath: %s",resultfilePath.c\_str());

LOG\_PRINT\_INFO("XMLfilePath: %s",XMLfilePath.c\_str());

string tempName = audioName;

size\_t dotpos = tempName.rfind(".");

string basename = tempName.substr(0,dotpos);

wavFullName = audioUrl+"/"+audioName;

sentFile = resultfilePath+"/"+basename+"\_sent.txt";

sentBakFile = resultfilePath+"/"+basename+"\_sent\_bak.txt";

roleResultFile = resultfilePath+"/"+basename+"\_text\_role\_result.txt";

interResultFile = resultfilePath+"/"+basename+"\_inter\_result.txt";

emotionResultFile= resultfilePath+"/"+basename+"\_emotion\_result.txt";

itnFile = resultfilePath+"/"+basename+"\_itn.txt";

puncFile = resultfilePath+"/"+basename+"\_punc.txt";

emotionResultItnPuncFile=resultfilePath+"/"+basename+"\_emotionItnPunc\_result.txt";

sentResultItnPuncFile=resultfilePath+"/"+basename+"\_sentItnPunc\_result.txt";

gidResultFile = resultfilePath+"/"+basename+"\_gender\_result.txt";

gidResult1 = resultfilePath+"/"+basename+"\_1.txt";

gidResult2 = resultfilePath+"/"+basename+"\_2.txt";

XMLfile = XMLfilePath+"/"+basename+".xml";

pResult2XMLPara = pStResult2XMLPara;

pPunctuationRule = pStPunctuationRule;

}

### 3.2.2 中间结果文件整理

中间结果数据整理包括整合sent、int、pun文件到\_sentItnPunc\_result.txt，整合role、int、pun文件到\_emotionItnPunc\_result.txt，解析\_emotionItnPunc\_result.txt文件，分句能量计算，解析\_sentItnPunc\_result.txt，角色重确认（双声道语音），解析叠音文件，静音计算：



整理中间结果文件流程图

主要函数设计

函数原型：

int Integrate\_File(const char\* itnFile,const char\* puncFile,const char\* emotionResultFile, const char\* emotionResultItnPuncFile, const char\* sentResultItnPuncFile);

参数说明：

const char\* itnFile 阿拉伯中间文件

const char\* puncFile 标点中间文件

const char\* emotionResultFile 角色中间文件

const char\* emotionResultItnPuncFile 输出整合itn，标点后的角色文件

const char\* sentResultItnPuncFile 输出整合itn，标点后的sent文件

返回值：

int 类型，0：成功，<0：失败。

功能：

整合标点、数字到新的sent、role文件。

函数原型：

int int Parser\_Emotion(const char\* szFile, vector<StSentenceItem>& vecStSentence); //解析场景分割结果;

参数说明：

const char\* szFile 场景分割中间结果文件

vector<StSentenceItem>& vecStSentence 全局结果结构体

返回值：

int 类型，0：成功，<0：失败。

功能：

解析场景分割文件。

函数原型：

int CalcEnergy(const char\* szFile, vector<StSentenceItem>& vecStSentence,int& nDuration); //计算能量;

参数说明：

const char\* szFile 语音文件名

vector<StSentenceItem>& vecStSentence 全局结果结构体

int& nDuration 语音时长

返回值：

int 类型，0：成功，<0：失败。

功能：

计算每个分局的能量到vecStSentence全局结构体中。

函数原型：

int Parser\_Sent(const char\* szFile, vector<StSentenceItem>& vecStSentence); //解析识别结果;

参数说明：

const char\* szFile sent文件名

vector<StSentenceItem>& vecStSentence 全局结果结构体

返回值：

int 类型，0：成功，<0：失败。

功能：

解析sent中间结果到vecStSentence全局结构体中。

函数原型：

int Reconfirm\_Role(const char\* szFile, bool role, vector<StSentenceItem>& vecStSentence);//重新确认角色;

参数说明：

const char\* szFile 备份左右声道识别sent文件

bool role 左右声道角色是否固定

vector<StSentenceItem>& vecStSentence 全局结果结构体

返回值：

int 类型，0：成功，<0：失败。

功能：

当双声道语音左声道固定角色时，角色重确认。

函数原型：

int GenerateInterruptedFile(int channels, const char \*interruptedFile, const char\* szFile, const char \*wavName, vector<StSentenceItem>& vecStSentence);//生成叠音文件

参数说明：

int channels 声道数

char \*interruptedFile 叠印文件名

const char\* szFile sent文件

const char \*wavName 语音名

vector<StSentenceItem>& vecStSentence 全局结果结构体

返回值：

int 类型，0：成功，<0：失败。

功能：

当双声道并且叠音文件不存在时，生产叠音文件。

函数原型：

int Parser\_Interrupted(const char\* szFile, vector<StInterruptedItem>& vecStInterrupted,const int interDuration); //解析叠音结果;

参数说明：

const char\* szFile 叠音文件名

vector<StInterruptedItem>& vecStInterrupted 保存叠音结果结构体

const int interDuration 叠音阈值

返回值：

int 类型，0：成功，<0：失败。

功能：

解析叠音中间结果文件，结果存入vecStInterrupted全局接结构体中。

函数原型：

int CResult2XML::CalcAllSilence(); //计算所有角色静音

参数说明：

返回值：

int 类型，0：成功，<0：失败。

功能：

根据静音阈值计算出所有角色满足阈值条件的时间点。

### 3.2.3 构造xml 结构体

构造XML结构体包括初始化xml结构，生成xml结构中的speaker-separation内容部分，生成xml结构中的search内容部分：



构造xml 结构体流程图

主要函数设计

函数原型：

int CResult2XML::InitXMLServerInput();

参数说明：

返回值：

int 类型，0：成功，<0：失败。

功能：

初始化xml结构，包括声道数、语音名、语音时长、比特率等。

函数原型：

int CResult2XML::ParseSpkDiarizationResult();

参数说明：

返回值：

int 类型，0：成功，<0：失败。

功能：

根据解析的中间结果文件内容，生成speaker-separation部分每个分句开始时间 、结束时间、分句能量、语速、时长、角色。

函数原型：

int CResult2XML::ParseSearchResult();

参数说明：

返回值：

int 类型，0：成功，<0：失败。

功能：

根据解析的中间结果文件内容，生成search部分内容，包括坐席、客户每个分词及分词对应开始结束时间点、分词分词个数、检索出来的关键词及关键词对应时间点、静音部分及对应时间点、叠音部分及对应时间点、高能量部分及对应时间点。

### 3.2.4 生成xml 文件

主要函数设计

函数原型：

void XMLServer::XMLS\_GenXML(const XMLServerInputData &input, string &xmlStr);

参数说明：

const XMLServerInputData &input 整理对应xml结构数据

string &xmlStr 字符串形式输出xml结果

返回值：

功能：

成成xml文件中每个节点，并且给节点赋值，最终组成标准xml文件。

## 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，xml模块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 |
| LongSilenceDuration | int | 静音阈值，单位：ms |
| InterruptedDuration | int | 叠音阈值，单位：ms |
| EngergyThreshold | int | 高能量阈值，单位：dB |
| LogFile | 字符串 | 日志文件 |
| LogLevel | 0-6 | 输出日志级别 |
| LogFileSize | int | 日志文件最大大小，单位：M |
| LogFileNum | int | 最多保存日志数量 |
| role\_channel\_fixed | int | 固定角色开关,1:固定,0:不固定 |
| role\_left | int | 固定左声道为,0:坐席,1:客户 |
| keyword | 字符串 | 关键字文件名 |
| OnOffkeyword | false:true | 关键字开关 |
| SpeedDuration | int | 平均语速阈值，用来计算年龄 |
| roleNums | int | 支持角色人数[2,3]: 2:客户，坐席 3:客户，坐席，系统播报 |

## 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。 |
| 静音阈值 | 设置不同阈值，输出正确结果。 |
| 叠音阈值 | 设置不同阈值，输出正确结果。 |
| 高能量阈值 | 设置不同阈值，输出正确结果。 |
| 关键词开关 | 打开关键词开关，正确输出关键词 |
| 双声道固定角色 | 双声道角色固定，xml正确输出角色内容 |
| 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 |

# 5.可靠性设计

5

离线系统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、中间结果文件完整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；同时支持叠音、静音、能量阈值参数配置，以适应不同的应用场景；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。

XML模块可配置configure\_xml.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：

LongSilenceDuration 静音阈值，单位：ms

InterruptedDuration 叠音阈值，单位：ms

EngergyThreshold 高能量阈值，单位：dB

# 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。