

中科院声学所离线xml

数据结构设计说明

目录

[1.概述 5](#_Toc31695)

[1.1简介 5](#_Toc2580)

[1.2 读者对象 5](#_Toc27449)

[3. 数据结构设计 6](#_Toc16988)

[3.1任务消息：TaskInfo 6](#_Toc12782)

[3.2结果消息：ResultInfo 7](#_Toc5685)

[3.3 Xml模块类定义 8](#_Toc23195)

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V1.0 | 王超锋 |  | 20211102 |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 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

本文档是对离线工程部分的内部代码流程和相关技术点的整体说明，因此读者最好满足如下几点条件后再读此文档：

* 有一定的C和C++语言基础

# 3. 数据结构设计

3

本节主要介绍protobuf消息结构设计和XML模块结构体。

总控端和各服务端之间通信的消息结构的封装采用Google Protocol Buffer( 简称 Protobuf)，类TaskInfo用于表示语音任务，类ResultInfo用于表示处理结果。

### **3.1任务消息：TaskInfo**

message TaskInfo

{

//音频信息

required string serialNum = 1;

required string audioname=2;

required string audiourl = 3;

required int32 channels=4;

required string language=5;

optional string keywordlist = 6;

optional string resultfilePath = 7 ;

optional string xmlfilePath=8;

}

说明：

* serialNum：音频文件的唯一性标识。
* audioname：音频文件的名称，例如：1.V3。
* audiourl ：音频的存放目录，例如：/home/thinkit/tempvoice

根据2,3的拼接得到语音文件的绝对路径。

* channels：音频的声道信息，0：单声道；1:双声道。
* language：音频语种信息，zhn：中文，eng：英语 .......。

若未设置，默认为zhn。

* keywordlist ：关键词列表的绝对路径。

若未设置，则默认使用XML服务端目录下的关键词列表。

* resultfilePath ：结果文本文件存放的绝对目录。

若未设置，则默认与xmlfilePath相同。

* xmlfilePath:最终的xml结果文件存放的绝对路径。

请务必给定该路径，否则任务将处理失败。

总控端从数据库中获取每条语音任务的详细信息后，都将对应生成一个个TaskInfo。当语音任务需要发送给某服务端前，再将TaskInfo序列化为一个字符串，发送给服务端进行处理。

### **3.2结果消息：ResultInfo**

message ResultInfo

{

required string serialNum = 1;

required int32 errorCode = 2;

optional bytes audioname=3;

optional bytes xmlfilePath = 4;

}

说明：

* serialNum：音频文件的唯一性标识。与TaskInfo中serialNum对应。
* errorCode：结果错误码。 0表示正常返回，-1表示分析过程有错误
* audioname：音频文件的名称。与TaskInfo中audioname对应。
* xmlfilePath: xml结果存放路径。与TaskInfo中xmlfilePath对应。

各服务端每处理完一个TaskInfo任务后，将相应构建一个ResultInfo，当需要将结果反馈给总控端前，将ResultInfo序列化为一个字符串，发送给总控端进行后续处理。

### 3.3 Xml模块类定义

本节主要整理中间结果并生成xml结果的类。

class CResult2XML

{

private:

string wavFullName; //原始语音文件

string sentFile; //识别结果文件

string sentBakFile; //识别结果分声道文件

string puncFile; //标点结果文件

string itnFile; //阿拉伯数字结果文件

string roleResultFile; //场景分割结果文件

string interResultFile; //叠音结果文件

string emotionResultFile;

string emotionResultItnPuncFile;//包含阿拉伯数字、标点

string sentResultItnPuncFile;//包含阿拉伯数字、标点

string gidResultFile; //性别识别结果文件

string gidResult1;

string gidResult2;

string XMLfile; //xml结果文件

Result2XMLPara \*pResult2XMLPara; //xml相关参数

StPunctuationRule \*pPunctuationRule; //标点符号规则

int nDuration; //语音总时长，单位：毫秒

int channels;

vector<StSentenceItem> vecStSentence; //识别、场景分割结果

vector<StInterruptedItem> vecStInterrupted; //叠音结果

vector<StSilenceItem> vecn0StSilence; //坐席静音结果

vector<StSilenceItem> vecn1StSilence; //客户静音结果

vector<StSilenceItem> vecn2StSilence; //系统静音结果

vector<StSilenceItem> vecmixStSilence; //坐席和客户都未说话时的静音结果

vector<StGenderItem> vecgender; //性别结果

XMLServerInputData input;

XMLServer server;

//数据准备：解析识别、场景分割、叠音结果，计算静音结果

int ParseData();

//计算静音

int CalcAllSilence();

int CalcSilence(vector<StSentenceItem>& vecSent, vector<StSilenceItem>& vecStSilence);

void CheckSilAndInter(vector<StSilenceItem>& vecStSilence, int start, int end);

//生成XMLServerInputData

int ConvertResult2InputData();

//初始化xml结构

int InitXMLServerInput();

//生成xml结构中的speaker-separation内容部分

int ParseSpkDiarizationResult();

//生成xml结构中的search内容部分

int ParseSearchResult();

//关键字处理

int DealKeyWord(const char\* text,const char\* time,std::string &keyword,std::string &keywordtime);

public:

//构造函数

CResult2XML(TaskInfo\* pTaskInfo, Result2XMLPara\* presult2XMLPara, StPunctuationRule\* pStPunctuationRule);

//读取中间结果->生成xml结果

int DoTask();

//获取结果XML名字

string GetXMLFileFullName();

}

typedef struct Result2XMLPara

{

XmlPhase m\_phase; //xml结果格式

string m\_Encoding; //xml文件编码格式

string m\_standalone; //

string m\_AudioFmt; //语音编码格式

string m\_channel; //语音声道数 单声道："mono",双声道"stereo

int m\_roleNums; //角色个数 2:客户，坐席 3:客户，坐席，系统播报

int m\_sampleRate; //采样率

int m\_bitRate; //位数

int m\_LongSilenceDuration; //静音阈值，单位：ms

int m\_InterruptedDuration; //叠音阈值，单位：ms

int m\_EngergyThreshold; //能量阈值,单位：dB

bool role\_channel\_fixed; //固定左右声道开关

bool role\_left; //固定左声道为"客户"或"坐席"

string m\_Keyword; //关键字

string m\_OnOffKeyword; //关键字开关

string m\_OnOffChannel; //mix角色

float m\_SpeedDuration; //平均语速阈值，用来计算年龄

float m\_EnergyDuration; //平均音量阈值，用来计算年龄

Result2XMLPara(): m\_Encoding("utf-8"),

m\_standalone("no"),

m\_AudioFmt("pcm"),

m\_channel("mono"),

m\_roleNums(2),

m\_sampleRate(8000),

m\_bitRate(16),

m\_LongSilenceDuration(3000),

m\_InterruptedDuration(500),

m\_EngergyThreshold(70),

role\_channel\_fixed(false),

role\_left(1),

m\_Keyword("keyword"),

m\_OnOffKeyword("false"),

m\_OnOffChannel("false"),

m\_SpeedDuration(0.0),

m\_EnergyDuration(0.0){}

}Result2XMLPara;