

中科院声学所离线聚类

数据结构设计说明

目录

[1.概述 5](#_Toc31701)

[1.1简介 5](#_Toc27606)

[1.2 读者对象 5](#_Toc18773)

[3. 数据结构设计 6](#_Toc9027)

[3.1任务消息：TaskInfo 6](#_Toc3434)

[3.2结果消息：ResultInfo 7](#_Toc14735)

[3.3 角色区分模块结构体、类定义 8](#_Toc15873)

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

聚类模块主要是对语音进行聚类，区分出不同说话人的时间片段，并且对说话人角色进行判断（坐席/客户）。

## 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消息结构设计和角色区分模块结构体。

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 角色区分模块结构体、类定义

本节主要整理角色区分模块结构体定义及类。

class OneWav

{

struct RESULT

{

double start;

double end;

std::wstring text;

double confidence;

std::wstring role;

unsigned sentenceId;

};

struct TEXT

{

std::wstring text;

unsigned sentenceId;

};

public:

OneWav(string wavfile, string sentRstFile,string wordsRsttFile,string ABRstFile,string ABRstFileFormat,string delRstSpace,bool flag, bool flag1,double time\_up\_symbol=0.3,double time\_down\_symbol=0.8);

~OneWav();

private:

/\* 用来处理\_sent.txt文件 \*/

void tbnr\_file(std::string);

/\* 用来处理聚类产生的文件 \*/

void cluster\_file(std::string, std::vector<double>&);

/\* 场景分割 \*/

void classify();

std::string wavname; //语音绝对路径

std::string sentfile,wordsResultFile,ABResultFile,ABFormat,delSpace;

double time\_up\_symbol;

double time\_down\_symbol;

bool symbol\_on;

bool format\_text;

std::vector<TEXT> arryText; //\*\_sent.txt中识别结果，以词为单位，记录词语所在句子的编号

std::vector<std::wstring> arryTime; //\*\_sent.txt所有时间

std::vector<double> arryTime\_Start; //\*\_sent.txt每个词开始的时间\*1000

std::vector<double> arryTime\_End; //\*\_sent.txt每个词结束的时间\*1000

std::vector<double> arryConfidence; //\*\_sent.txt每句的置信度\*100

std::vector<double> arryTime\_A; //\*\_1.txt中的时间数值

std::vector<double> arryTime\_B; //\*\_2.txt中的时间数值

std::vector<double> arryTime\_C; //\*\_3.txt中的时间数值

std::vector<RESULT> result\_words;

std::vector<std::wstring> words\_sum;

std::vector<RESULT> result\_text;

};