**SpeechRecognizer 接口说明**

**目录**

综述、用到的第三方库、接口说明、附件资料说明、备注

**一、综述**

这个库采用源码发布形式，没有lib和dll文件。SpeechRecognizer是封装讯飞语音的一个类，目前支持了两个功能：语音识别和语音听写，下面分别简单介绍下。

语音听写，是用来识别用户的语音，比如用户说“攻击”，服务器会返回“供给”之类的中文文字。

语音听写，对于歧义这类情况，可以采用结合语法文件的使用。它的效果相当于约定某个发音只能是匹配到某些中文词汇，比如“gongji”，只能匹配到我们需要的“攻击”。

更详细的描述可以参考讯飞的doc文档下面的资料。

**二、用到的第三方库**

* Boost的线程库
* 讯飞的sdk，分别为win32和win64版本
* Openal音频库，主要用来听写用户输入的语音

**三、SpeechRecognizer类的接口说明**

模块命名空间

* SpeechRecognizer

createSpeechRecognizer

* 函数原型

static SpeechRecognizer\* createSpeechRecognizer();

* 功能

用来实例化一个SpeechRecognizer的工厂函数.

* 参数

无

* 返回值

如果函数调用成功返回实例指针，否则抛出错误。

* 用法示例

SpeechRecognizer\* speech;

//first Instantiate our recognizer

speech = new SpeechRecognizer();

//注意最后删除资源 或者 采用**std::auto\_ptr<SpeRec::SpeechRecognizer>**

delete speech;

SpeechRecognizer

* 函数原型

SpeechRecognizer SpeechRecognizer (void);

* 功能

用来实例化一个SpeechRecognizer.

* 参数

无

* 返回值

如果函数调用成功返回实例指针，否则抛出错误。

* 用法示例

namespace SpeRec = SpeechRecognizer;

SpeechRecognizer\* speech;

std::auto\_ptr<SpeRec::SpeechRecognizer> speech( SpeRec::createSpeechRecognizer());

set\_logger\_params

* 函数原型

bool set\_logger\_params(const char \* log\_path,int32 log\_level);

* 功能

用来设置日志记录信息。必须在init前调用。

* 参数

const char \* log\_path 日志记录路径，请填写绝对地址。默认存储在当前运行位置的目录下，生成"speechRecognizer.log"文件。

int32 log\_level 日志等级，在日志命名空间SpeechRecognizer:: logger下。

//log level

enum LogLevel { INFO=0, WARNING, \_ERROR, FATAL, NONE };

* 返回值

如果函数调用成功返回实例指针，否则抛出std::runtime\_error("Cannot open file");错误。

* 用法示例

namespace SpeRec = SpeechRecognizer;

try{

speech->set\_logger\_params("abc.log",SpeRec::logger::INFO);//path and level

}

catch(const std::runtime\_error& e)

{

//file not exit thrown!

//e.what();

}

init

* 函数原型

bool SpeechRecognizer (void);

* 功能

用来初始化一个SpeechRecognizer实例，包括配置openal参数，初始化内部用的变量等。

* 参数

无

* 返回值

如果函数调用成功返回true，否则返回false。

* 用法示例

//first Instantiate our recognizer

bool fun\_status = false;

fun\_status = speech->init();

if( fun\_status != true )

{

//check log

return -1;

}

startIsrListening

* 函数原型

bool startIsrListening(const char \* appid,const char \* isr\_params);

* 功能

用来开始语音听写, 非阻塞，会在内部启动一个线程。需要结合stopListening一起用，完成整个的调用过程。

* 参数

const char \* appid 我们应用的标识符，这个找舟哥要

const char \* isr\_params 语音听写的参数，具体可以参考讯飞的文档

* 返回值

如果函数调用成功返回true，否则返回false。

* 用法示例

const char \* appid = "appid=5201a7cd";

const char\* recog\_params = "rst=plain,sub=asr,ssm=1,aue=speex-wb;7,auf=audio/L16;rate=16000,grt=abnf";//注意sub=asr,grt=abnf

////when user click or some evt happens,start listening

fun\_status = speech->startIsrListening(appid,isr\_params);//非阻塞，会在内部启动一个线程

if( fun\_status != true )

{

//start listening error

//may need to check log to find out reasons

return -1;

}

startRecogListening

* 函数原型

bool startRecogListening(const char \* appid, const char\* grammar,const char \* recog\_params);

* 功能

用来开始语音听写。

* 参数

const char \* appid 我们应用的标识符，这个找舟哥要

const char\* grammar 语法文件的路径，具体写法参考讯飞文档

const char \* recog\_params 语音听写的参数，具体可以参考讯飞的文档

* 返回值

如果函数调用成功返回true，否则返回false。

* 用法示例

const char \* appid = "appid=5201a7cd";

const char\* recog\_params = "rst=plain,sub=asr,ssm=1,aue=speex-wb;7,auf=audio/L16;rate=16000,grt=abnf";//注意sub=asr,grt=abnf

const char \* grammar = "abnf/alishanlongdan.abnf";

////when user click or some evt happens,start listening

fun\_status = speech->startRecogListening(appid,grammar, recog\_params);//非阻塞，会在内部启动一个线程

if( fun\_status != true )

{

//start listening error

//may need to check log to find out reasons

return -1;

}

getResultLength

* 函数原型

bool getResultLength(int32 &result\_length,int32 &status);

* 功能

查询当前语音识别的结果状态，可以查询识别结果的长度和识别状态

* 参数

int32 &result\_length 内部存放识别结果的buffer的长度

int32 &status 识别状态，在模块命名空间下：

//返回结果的状态码

enum

{

SPEECHRECOGNIZER\_SUCCESS = 1, //结果返回成功

SPEECHRECOGNIZER\_INCOMPLETE = 3, //结果获取当中

SPEECHRECOGNIZER\_FAIL = -1, //出错

};

* 返回值

如果函数调用成功返回true，否则返回false。

* 用法示例

暂无

getResult

* 函数原型

int32 getResult(char \*result,int32 buffer\_length,int32 &status,bool buffer\_clear);

* 功能

获取识别结果。

* 参数

char \*result 客户端的存放结果的buffer指针

int32 buffer\_length 传进来的buffer长度

int32 &status识别状态，同上

bool buffer\_clear 是否对内部的结果buffer清零

* 返回值

返回int32类型的值，>=0时表示拷贝的数据长度，=-1时候表示调用错误。

* 用法示例

基本用法是轮询接口，判断状态和byte\_copy的值

//get result

do{

Sleep(1000); //sleeptime can be changed

byte\_copy = speech->getResult(result,BUFFER\_LENGTH,status,clear\_buffer);

cout<<"API:: "<<"byte buffer.byte\_copy: "<<byte\_copy<<". status is: "<<status<<endl;

if( status == SpeRec::SPEECHRECOGNIZER\_SUCCESS ){

cout<<"API:: "<<"data length in byte(s): "<<byte\_copy<<" || result is: "<<result<<endl;

break;

}else{

if(status == SpeRec::SPEECHRECOGNIZER\_FAIL)

{

byte\_copy = speech->getError(result,BUFFER\_LENGTH);

cout<<"API:: "<<"Error! Length: "<<byte\_copy<<" Result:"<<result<<endl;

break;

}

else if( status == SpeRec::SPEECHRECOGNIZER\_FAIL\_OTHER )

{

//

}

}

}while( max\_try++ < 5);

getError

* 函数原型

int32 getError (char \* error\_str,int32 out\_buffer\_length);

* 功能

获取语音识别过程产生错误的原因。

* 参数

char \* error\_str 客户端传入的内存指针

int32 out\_buffer\_length 内存区大小

* 返回值

返回int32类型的值，>=0时表示拷贝的数据长度，=-1时候表示调用错误。

* 用法示例

暂无

**四、附件资料说明**

SpeechRecognizer文件夹为封装的接口工程文件。

windows\_1.048\_5201a7cd(1).zip 为讯飞提供的win32和win64开发包。

**五、备注**

目前功能还存在些bug，仍在调整当中。