

中科院声学所离线识别引擎

代码模块接口设计说明

目录

[1.概述 5](#_Toc90302732)

[1.1简介 5](#_Toc90302733)

[1.2 读者对象 7](#_Toc90302734)

[1.3术语定义 7](#_Toc90302735)

[2.总体设计 8](#_Toc90302736)

[3.接口设计 11](#_Toc90302737)

[3.1 引擎接口设计 11](#_Toc90302738)

[3.1.1 引擎初始化接口 11](#_Toc90302739)

[3.1.2 引擎退出接口 12](#_Toc90302740)

[3. 1.3 引擎开启一路识别 12](#_Toc90302741)

[3. 1.4 引擎开启一路识别 13](#_Toc90302742)

[3. 1.5该路识别设置事件回调函数 14](#_Toc90302743)

[3. 1.6该路识别设置结果回调函数 14](#_Toc90302744)

[3. 1.7该路会话结果回调函数设计 15](#_Toc90302745)

[3. 1.8该路会话进行语音数据识别函数设计 16](#_Toc90302746)

[3. 1.9 该路会话停止送任务函数设计 17](#_Toc90302747)

[3. 1.10 该路会话参数参数设置函数 17](#_Toc90302748)

[3. 1.11该路会话设置热词列表 18](#_Toc90302749)

[3. 1.12该路会话释放热词列表资源 19](#_Toc90302750)

[3.2 引擎接口流程 20](#_Toc90302751)

[3.3 引擎数据在线处理流程 21](#_Toc90302752)

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| 1.0 | 陈向东 | 龚云波  黎塔 | 2021/04/28 |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 1.概述

1

1.1简介

“好记性不如烂笔头”，反映出人们对记录的重视程度。从传统的  
纸文化进步到计算机时代，除了手写识别，当信息量更大的时候，人们  
会选择音频和视频。虽然人们不用担心存储容量不够用，但是今后如何  
有效的利用这些多媒体的记录，成为一个突出问题。

很多人为此倾向花高价邀请速录员。一来需要成本，二来在专业领  
域，培训和保密又成为新的问题。现在，用户有了新的选择-声学所识别引擎。该识别引擎不仅能保存语音，还能把语音转化成文字与之对应保存下来。有了它，今后无论何时用户想要调出所需的记录，只要通过查找相应文字并简单点击，相关的语音片断就找到了。同时，本引擎还提供了直接识别以往录音的功能，把以前宝贵的资料统统转化成文本，给用户的媒体档案库将来查询时提供最大便利。

中科院声学所连续语音识别引擎，针对连续音频流（即来自说话人直接录入的语音，或者广播电台、电视台或其他领域的音频信号）进行识别，将音频信息自动转化成文字。覆盖汉语中绝大多数词语，适用于说普通话的任何人群。输出的结果都是汉字，兼容数字。

在输入的声音中，检测出可靠的语音，排除噪声音乐等，实时送入语音识别解码器进行识别。识别引擎把音视频中提取出的语音分成 25 毫秒一帧，提取有用特征，然后识别出一些类似拼音的结果（声学模型），再根据汉语字词句之间的搭配概率（语言模型），综合考虑。当然，考虑的越多（beam路径越大），识别准确率相对提高，同时消耗的时间就增长了；所以我们有优化策略，及时排除不可能的结果，避免系统过慢。用户可以通过调节这些参数来平衡识别质量和速度，以满足实际的需要。

最后返回的识别结果，软件以汉语中基本词语为单位给出了包含对应时间点的词序列。

需要说明的是，对识别正确率以及识别速度来说，输入语音（普通话）的质量（录音过程）是很重要的。录音时应尽可能的排除噪声和音乐、增大语音，识别效果就会好得多，因此建议用户在录音的时候选用质量较好的麦克风。

另外，系统在检测是否有语音信号时可能会带来一定的时间滞后，用户在实时录入时在每句话结束后，需要略有停顿，这样可以提高系统语音检测的速度和精度。语音检测的越好。识别引擎的识别效果也就越好。

## 1.2 读者对象

本文档是对识别引擎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
* 了解语音识别过程和原理
* 了解声学所识别引擎实现原理

## 1.3术语定义

TBNR，Thinkit-Broadcast-News-Recongnize：中科信利识别引擎

CN，Confuse-Network：混淆网络

VAD，Speech Activity Detection：语音活动检测

WFST, Weighted Finite-State Transducer：加权有限状态转录即

ITN,Inverse –Text-Normalization：逆文本标准化

# 2.总体设计

2

声学所在线识别引擎的大体流程可以分为五步：

第一步：进行识别环境的初始化，构建WFST解码网络。

第二步：将输入的语音数据送入到声学所的识别引擎中进行分段处理，得到有效的分段语音数据。

第三步：将步骤二中得到的分段语音数据进行特征提取，得到分段语音对应的特征数据。

第四步：将步骤中的特征数据进行解码，得到多个候选的识别结果。

第五步：识别结果进行相应的后处理。并通过回调函数进行返回。

识别引擎总体设计流程如图2.1所示：



图2.1 识别引擎总体设计流程

如上图所示，首先对输入的音频数据进行端点检测，获取逐句的有效语音数据，然后将其送入特征提取模块，接着对于提取的声学特征进行解码，解码过程采用transformer的decoder输出进行字符同步的束搜索算法，在搜索空间内寻找匹配概率最大的最优路径，得到识别结果。

# 3.接口设计

3

## 3.1 引擎接口设计

## 3.1.1 引擎初始化接口

|  |  |
| --- | --- |
| 函数原型 | **TBNR\_Init(const char\* systemDir, char\* configFile, int NumberOfSession = -1);** |
| 功能 | 读取配置文件，加载识别模型构建解码网络，创建解码线程，初始化数据处理资源，初始化识别结果后处理资源。 |
| 参数说明 | char \* systemDir:引擎模型位置 |
| char\* configFile 引擎配置文件 |
| int NumberOfSession 引擎并发量 |
| 返回值 | 成功返回0,失败返回<0;; |
| 说明 |  |

## 3.1.2 引擎退出接口

|  |  |
| --- | --- |
| 函数原型 | **TBNR\_Exit();** |
| 功能 | 释放解码解码网络，回收解码线程，释放数据处理资源和识别结果后处理资源 |
| 参数说明 |  |
| 返回值 | 成功返回0,失败返回<0;; |
| 说明 |  |

## 3. 1.3 引擎开启一路识别

|  |  |
| --- | --- |
| 函数原型 | **TBNR\_Start(int sessionId = 0);** |
| 功能 | 输入要开启识别的会话ID，当该会话已经被占用时返回处于空闲状态的识别会话ID，并初始化该路会话识别资源 |
| 参数说明 | int sessionId 请求开启识别的会话ID |
| 返回值 | 成功返回处于空闲状态的识别会话ID，失败返回<0 |
| 说明 | 当所有会话(**NumberOfSession**)都被占用时返回值<0，表示没有多余的会话ID可被使用 |

## 3. 1.4 引擎开启一路识别

|  |  |
| --- | --- |
| 函数原型 | **TBNR\_Stop(int sessionId);** |
| 功能 | 关闭一路sessionId识别会话,释放该路会话资源 |
| 参数说明 | int sessionId 请求关闭识别的会话ID |
| 返回值 | 成功返回0,失败返回<0 |
| 说明 |  |

## 3. 1.5该路识别设置事件回调函数

|  |  |
| --- | --- |
| 函数原型 | **TBNR\_SetEventCallbackFunc(TBNR\_ProcessEventCallback pFunc, int sessionId = 0);** |
| 功能 | 设置该路识别会话的事件回调函数,通过时间回调函数实时返回该路识别的状态 |
| 参数说明 | TBNR\_ProcessEventCallback pFunc 事件回调函数地址 |
| int sessionId 该路识别会话ID |
| 返回值 | 成功返回0,失败返回<0 |
| 说明 | 引擎通过事件函数返回引擎工作状态,以便通知接口层引擎识别完成,可以停止当前识别会话以复用； |

## 3. 1.6该路识别设置结果回调函数

|  |  |
| --- | --- |
| 函数原型 | **TBNR\_SetResultCallbackFunc (TBNR\_ResultCallback pFunc, int sessionId = 0);** |
| 功能 | 设置该路识别会话的事件结果函数,通过结果回调函数实时返回该路的识别结果 |
| 参数说明 | TBNR\_ResultCallback pFunc 结果回调函数地址 |
| int sessionId 该路识别会话ID |
| 返回值 | 成功返回0,失败返回<0 |
| 说明 |  |

## 3. 1.7该路会话结果回调函数设计

|  |  |
| --- | --- |
| 函数原型 | **typedef void (\*TBNR\_ResultCallback) (const Task \*pResultArray, int numberOfTasks, int sessionId);** |
| 功能 | 获取识别结果 |
| 参数说明 | Task \*pResultArray 分段结果数组头地址 |
| int numberOfTasks 分段结果数据大小 |
| int sessionId 会话ID |
| 返回值 |  |
| 说明 | 每出发一次回调函数可能返回会个分段的识别结果 |

## 3. 1.8该路会话进行语音数据识别函数设计

|  |  |
| --- | --- |
| 函数原型 | **TBNR\_SendData(char \*buf, int length, int sessionId = 0, SPEECHTYPE type = TSR\_RAW\_16, long timestamp = -1);** |
| 功能 | 送入语音数据进行识别 |
| 参数说明 | char \*buf 语音数据 |
| int length 语音数据大小(字节) |
| int sessionId 会话ID |
| SPEECHTYPE type 语音格式 |
| long timestamp 更新时间戳 |
| 返回值 | 成功返回0,失败返回<0 |
| 说明 |  |

## 3. 1.9 该路会话停止送任务函数设计

|  |  |
| --- | --- |
| 函数原型 | **TBNR\_StopRecording(int sessionId = 0);** |
| 功能 | 通知引擎停止送语音数据 |
| 参数说明 | int sessionId 会话ID |
| 返回值 | 成功返回0,失败返回<0 |
| 说明 | 该函数主要通知引擎送任务结束，对最后一个分段进行强制识别。 |

## 3. 1.10 该路会话参数参数设置函数

|  |  |
| --- | --- |
| 函数原型 | **int TBNR\_SetParam(char \*paramName,** **char \*paramValue, int sessionId = 0)** |
| 功能 | 设置当前识别会话资源的某些参数值 |
| 参数说明 | char \*paramName 参数名称 |
| char \*paramValue 参数值 |
| int sessionId 会话ID |
| 返回值 | 成功返回0,失败返回<0 |
| 说明 | 设置参数具体有“isSaveData”是否保存语音数据，“isSaveFeature”是否保存特征，“printPinyin”是否输出拼音，“printPunctuation”是否输出标点，“doITN”是否做ITNdoITN，“incompletetimeout”VAD参数调节 |

## 3. 1.11该路会话设置热词列表

|  |  |
| --- | --- |
| 函数原型 | **int TBNR\_LoadHotWord(HotWord\* pHotWordList,int nLen,int sessionId);** |
| 功能 | 设置当前识别会话的热词列表 |
| 参数说明 | HotWord\* pHotWordList热词信息数组首地址 |
| int nLen 热词信息数组大小 |
| int sessionId 会话ID |
| 返回值 | 成功返回0,失败返回<0 |
| 说明 | 热词为GBK编码格式 |

## 3. 1.12该路会话释放热词列表资源

|  |  |
| --- | --- |
| 函数原型 | **Int TBNR\_FreeHotWord (int sessionId);** |
| 功能 | 释放当前识别会话的热词资源 |
| 参数说明 | int sessionId 会话ID |
| 返回值 | 成功返回0,失败返回<0 |
| 说明 |  |

## 3.2 引擎接口流程



## 3.3 引擎数据在线处理流程



图5.3.1 在线模式数据处理函数调用流程

如图5.3.1所示，外层通过TBNR\_SendData()接口函数完成送入语音任务，内层对进来的语音任务依次进行高通滤波，EPD端点检测，特征提取，然后将提取的特征压入各自对应任务池中，解码线程依次从任务池中依次获取特征任务，完成解码并将解码后的结果各自的结果池中，并通过结果回调返回结果。

说明1：外层的会话数和内层的解码线程数是相等，并且会话ID(SessionID)与解码ID(DecodeID)是一一对应的，也就是说外层的会话与内层的解码进行了绑定。

说明2：在EPD端点检测时，检测到起点时，将起点之后的每一小包任务(1024个采样点)送入特征提取模块进行特征提取，将提取的特征压入与之会话ID对应的任务池中等待解码；当解码器解码到尾点包时，获取解码结果。也就说在线识别环境是在检测到起点之后开始进行解码。