科大讯飞
iFLYTEK

智能化技术训练营
一期题目及相关技术培训

2014.05.10

写在前面的话

- 对象
  - 具备基础编程能力的，对语音及相关技术感兴趣的同学
- 定位
  - 基本信号处理，基础数据结构
- 范围
  - 语音分析，数据挖掘
- 预备知识
  - 了解数字信号处理概念
  - 初步的高等数学知识
  - 计算机编程基础知识

写在前面的话

- 语音是人类沟通和获取信息最自然便捷的手段和方式，也是文化的基础和民族的象征。
- 而智能语音及语言交互技术，可以应用在社会生活的方方面面，拥有广阔的产业化前景，尤其在军事、教育、汉语国际推广等重要战略领域，都有广泛应用和重大推广意义。

例子


下午2:46
我好无聊哇
有我陪你就不会无聊的，因为我懂好多知识呢。你可以说"历史上的今天"或者"古诗词"，我就可以说给你听啦。
我爱你
终于向我表白了，哈哈！
点击说话

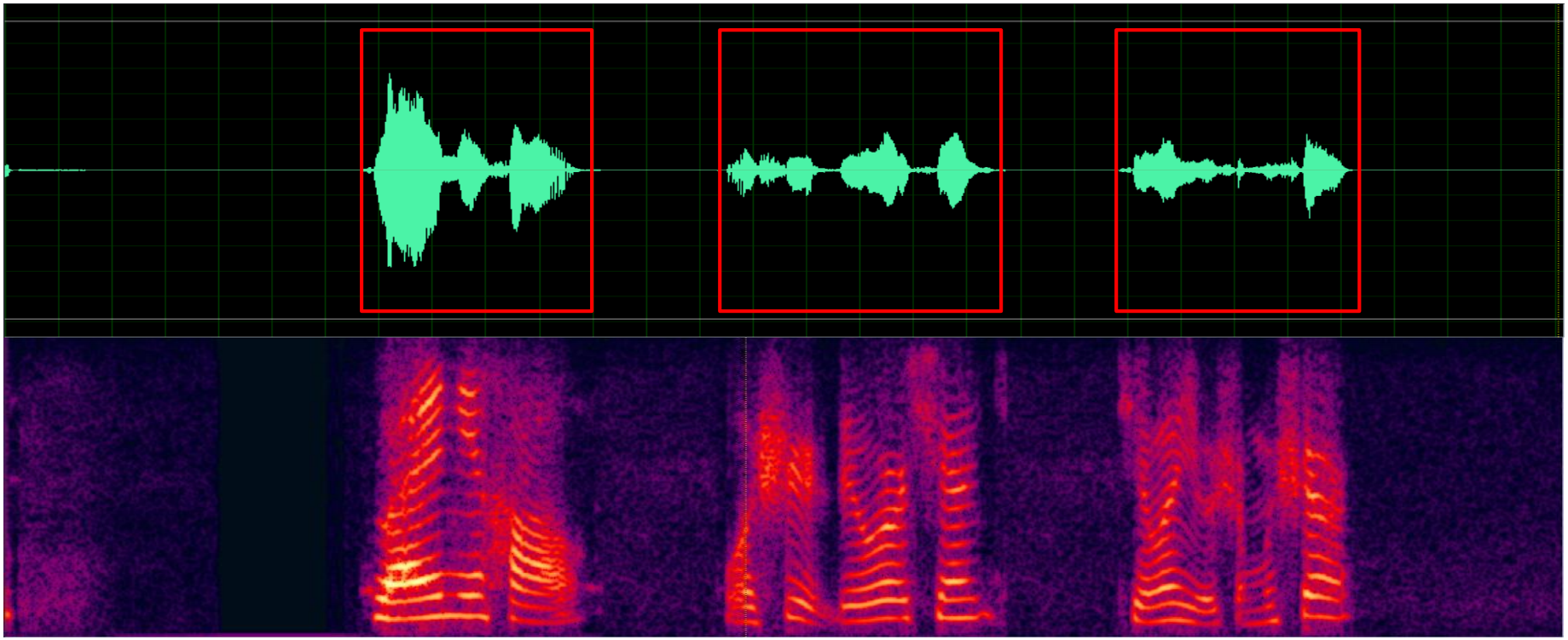

“我的高跟鞋在哪里？”
高跟鞋在壁橱中间第二层。
苹果资讯
apple.178.COM

提纲

- 题目一：有效音频段区分
- 题目二：数据打包及检索
- 考核规则
- 相关知识的普及

有效音频段区分——背景

- 语音识别中，只有真正语音部分才是我们研究的主体，无效音频段会对我们识别的效果产生较大的负面影响。



有效音频段区分——题目

- 编写程序，对一段音频进行分析，划分出有效音频的区间。

- 例：

输入：音频的路径

输出：{[40,80],[230,560]...}

头文件

```
//+-------------------------------------------------------------------------+
//+ 初始化
//+ 对需要的数据结构进行初始化操作
//+-------------------------------------------------------------------------+
//+ return         - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int Initialize();
typedef int (*Proc_Initialize)();

//+-------------------------------------------------------------------------+
//+ 有效音频段区分
//+ 对输入的音频进行有效段区分，输出结果
//+-------------------------------------------------------------------------+
//+ szWavPath      - 音频输入路径（8k16bit）
//+ szResultString     - 输出结果，10毫秒分为1帧，单位为帧，格式：{[20,400],[450,952],...}
//+ return         - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int ProcessVad(const char *szWavPath, std::string &szResultString);
typedef int (*Proc_ProcessVad)(const char *szWavPath, std::string &szResultString);


//+-------------------------------------------------------------------------+
//+ 逆初始化
//+ 对需要的数据结构进行析构
//+-------------------------------------------------------------------------+
//+ return         - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int unInitialize();
typedef int (*Proc_unInitialize)();
```

有效音频段区分——要求

- 使用c/c++
- 使用单线程
- 衡量指标
  - 端点检测准确性（端点偏差需小于0.2秒）
  - 处理速度

有效音频段区分——可能涉及的技术点

- 算法类：
  - 信号的时域分析
  - 信号的频域分析
  - 机器学习
  - 。 。 。
- 工程类
  - **SIMD（SSE, AVX...）**
  - 。 。 。

提纲

- 题目一：有效音频段区分
- 题目二：数据打包及检索
- 考核规则
- 相关知识的普及

数据打包及检索——背景

- 随着网络的发展，数据爆炸式的增长让我们在海量信息中迷失。故我们需要更高效的检索方案来获取对我们有意义的数据。

数据打包及检索——题目

- 将指定的歌手列表编译为一个资源包。
- 加载资源包，对输入的歌手名，在资源包中进行检索，给出对应的热度信息。

- 例：
- 输入：杨幂
- 输出：{[杨幂,27952601]...}

name	hot
何炅	44874956
谢娜	41690032
舒淇	31247766
孙燕姿	29073493
杨幂	27952601
林俊杰	22778981
范玮琪	21361021
杨丞琳	20741610
郭富城	20403410
李维嘉	18909095

头文件-打包

```
//+-------------------------------------------------------------------------+
//+ 初始化
//+ 对需要的数据结构进行初始化操作
//+-------------------------------------------------------------------------+
//+ return          - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int Initialize();
typedef int (*Proc_Initialize)();

//+-------------------------------------------------------------------------+
//+ 数据打包
//+ 对给出的歌单进行打包，输出资源文件
//+-------------------------------------------------------------------------+
//+ szListPath      - 歌单输入路径
//+ szResPath       - 资源输出路径
//+ return          - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int ProcessBuildRes(const char *szListPath, const char *szResPath);
typedef int (*Proc_ProcessBuildRes)(const char *szListPath, const char *szResPath);

//+-------------------------------------------------------------------------+
//+ 逆初始化
//+ 对需要的数据结构进行析构
//+-------------------------------------------------------------------------+
//+ return          - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int unInitialize();
typedef int (*Proc_unInitialize)();
```

头文件-检索

```
//+-------------------------------------------------------------------------+
//+ 初始化
//+ 对需要的数据结构进行初始化操作
//+-------------------------------------------------------------------------+
//+ szResPath      - 资源输入路径
//+ return         - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int Initialize(const char *szResPath);
typedef int (*Proc_Initialize)(const char *szResPath);

//+-------------------------------------------------------------------------+
//+ 资源检索
//+ 加载编译好的资源包，对输入的歌手名进行检索，并输出对应的热度信息
//+-------------------------------------------------------------------------+
//+ szSingerName     - 歌手名,格式：杨幂
//+ szResultString     - 输出结果，格式：{[杨幂,27952601],...}
//+ return         - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int ProcessSelelct(const char *szSingerName, std::string &szResultString);
typedef int (*Proc_ProcessSelelct)(const char *szSingerName, std::string &szResultString);

//+-------------------------------------------------------------------------+
//+ 逆初始化
//+ 对需要的数据结构进行析构
//+-------------------------------------------------------------------------+
//+ return         - 返回码
//+-------------------------------------------------------------------------+
extern "C" __EXPORT int unInitialize();
typedef int (*Proc_unInitialize)();
```

数据打包及检索——要求

- 使用c/c++
- 使用单线程
- 衡量指标
  - 资源包尽量小
  - 资源包加载速度尽量快
  - 检索速度尽量快

数据打包及检索——可能涉及的技术点

- 算法类
  - 数据压缩算法
  - 数据检索算法
  - 。。。
- 工程类
  - SIMD（SSE，AVX...）
  - 。。。

提纲

- 题目一：有效音频段区分
- 题目二：数据打包及检索
- 考核规则
- 相关知识的普及

提交形式

- 提供以windows动态链接库及源码
- 入口为指定API

➢ Init(…);

➢ Process(…);

➢ unInit(…);

➢ …

测试集

- 测试集按照难度分为A、B、C三个集合。
- 难度： A > B > C

数据集评分加权	权值
A	0.5
B	0.3
C	0.2

衡量标准

- 效果分 = Σ（数据集效果得分 * 数据集评分加权）
- 效果满分80分

- 依据代码的编程风格，稳定性等一些指标，给出附加分。
- 附加满分20分

提纲

- 题目一：有效音频段区分
- 题目二：数据打包及检索
- 考核规则
- 相关知识的普及

什么是优秀的程序？

- 可读性
- 稳定性
- 扩展性
- 高效性

动态链接库

- 动态链接库英文为DLL，是Dynamic Link Library 的缩写形式，DLL是一个包含可由多个程序同时使用的代码和数据的库。
- 优点

➢ 1、扩展了应用程序的特性；
➢ 2、可以用许多种编程语言来编写；
➢ 3、简化了软件项目的管理；
➢ 4、有助于资源共享；
➢ 5、有助于应用程序的本地化；

Debug和Release的区别

- Debug 通常称为调试版本，它包含调试信息，并且不作任何优化，便于程序员调试程序。
- Release 称为发布版本，它往往是进行了各种优化，使得程序在代码大小和运行速度上都是最优的，以便用户很好地使用。
- Debug 和 Release 的真正区别，在于一组编译选项。

DEBUG

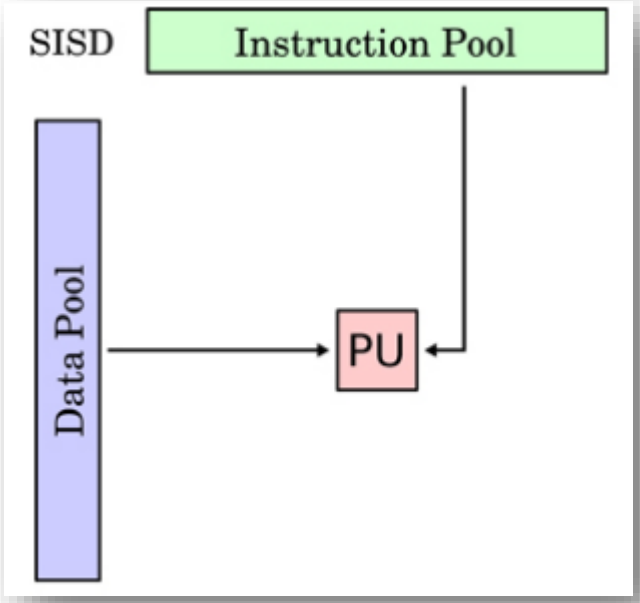
- /Od 关闭优化开关
- /D "_DEBUG" 相当于 #define _DEBUG,打开编译调试代码开关(主要针对assert函数)
- /ZI 创建 Edit and continue(编辑继续)数据库，这样在调试过程中如果修改了源代码不需重新编译
- GZ 可以帮助捕获内存错误

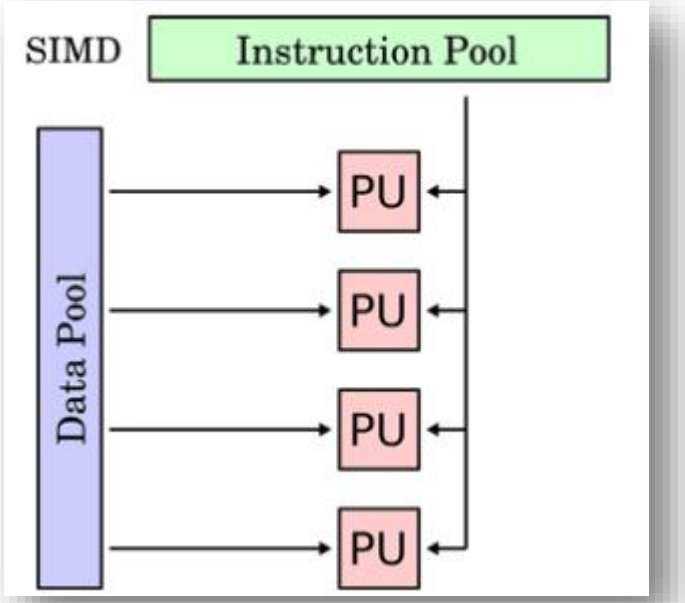
RELEASE

- /O1 或 /O2 优化开关，使程序最小或最快
- /D "NDEBUG" 关闭条件编译调试代码开关(即不编译assert函数)

SIMD

- Single Instruction Multiple Data，单指令多数据流，能够复制多个操作数，并把它们打包在大型寄存器的一组指令集，例：3DNow!、SSE。





- 加油！