Kaldi识别语音英文数字的ASR系统

环境：

本地环境：

操作系统：Windows10 1803

CPU:corei5-4200H

RAM: 4G

GPU:GTX-960M

北邮浪潮服务器环境：

操作系统：Ubuntu 16.04.5 LTS

CPU: Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz

物理CPU： 2个 逻辑CPU：48个

GPU: ASPEED Graphics

运行环境配置：

Kaldi工具包以及相应的库已经安装完成。

数据准备：

该demo数据来源于：<https://hlthu.github.io/public/downloads/numbers-en.tar.gz>

备注：语音包中包含128个语音文件，wav格式的，分别是两个人朗读，其中每个文件只包含三个数字。这 128 文件中 80 （8\*10）个用于训练， 48 (8\*6) 个用于测试。并且训练数据和测试数据都被分成了 8 部分（可以假装成 8 个人），每部分分别 10 个 和 6 个。训练集和测试集的前五个目录是男士朗读，后面三个是女士朗读的。

该数据集：采用16kHz采样，16位量化。如果是8kHz采样，需要在提取特征时指明采样频率，即steps/make\_mfcc.sh里面的computer-mfcc-feats命令加上- -sample-frequency=8000的选项，或者在conf中的mfcc.conf文件中指明。



本文可参考链接：

<http://kaldi-asr.org/doc/kaldi_for_dummies.html>

<https://blog.csdn.net/m0_38055352/article/details/82416633>(此链接后面未成功识别出)

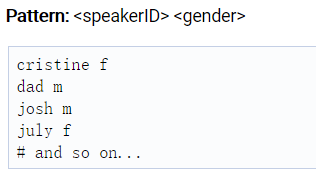
该demo存放地址：

Kaldi项目下的egs/digits地址

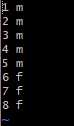
具体流程：

1. 创建一个文件夹data，在里面创建test和train，local子文件夹
2. 准备相关声学数据：
3. **编写spk2gender文件**

备注：This file informs about speakers gender. As we assumed, 'speakerID' is a unique name of each speaker (in this case it is also a 'recordingID' - every speaker has only one audio data folder from one recording session). In my example there are 3 female and 5 male speakers (f = female, m = male).

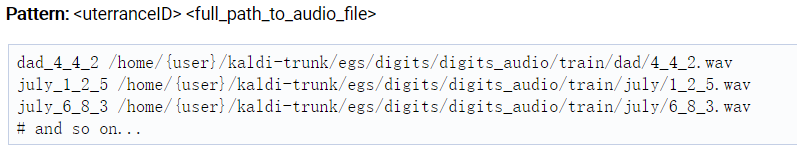


我们在此编写的文件为：



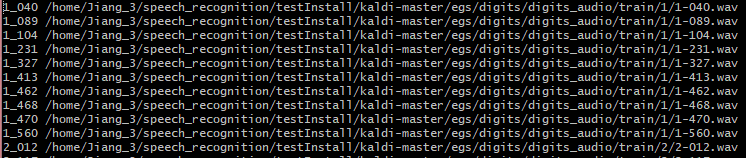
1. **编写wav.scp文件**

备注：This file connects every utterance (sentence said by one person during particular recording session) with an audio file related to this utterance. If you stick to my naming approach, 'utteranceID' is nothing more than 'speakerID' (speaker's folder name) glued with \*.wav file name without '.wav' ending (look for examples below).



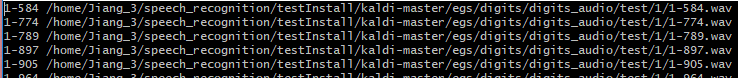
在此编写的文件部分如下所示：

错误示范：



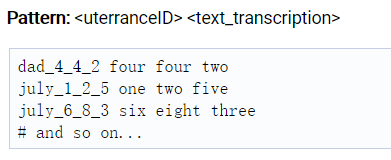
注意：音频的命名一定要都写成“-”连接（尤其是data/test中）。

正确示范：



1. **编写text文件**

备注：This file contains every utterance matched with its text transcription.

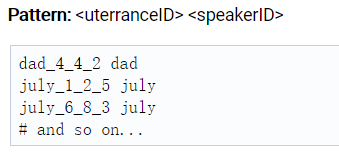


在此编写文件部分如下所示：



1. **编写utt2spk文件**

备注：This file tells the ASR system which utterance belongs to particular speaker.



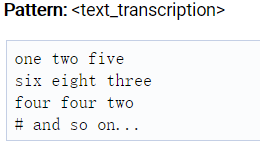
在此编写的文件部分为：



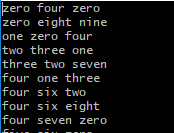
备注：这四个文件同样的道理在test文件夹中也应该准备，并按照相同的格式及规则准备

1. 语料库
2. **编写corpus.txt文件**

备注：In kaldi-master/egs/digits/data/local create a file corpus.txt which should contain every single utterance transcription that can occur in your ASR system (in our case it will be 100 lines from 100 audio files).



在此编写的文件局部所示为：



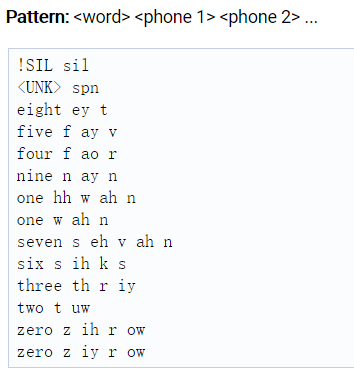
备注：这些声学文件基于我们的数据集不大可以自己手动完成编写，且帮助理解kaldi的执行流程，在以后数据集更大以后可以使用脚本文件自动编写这些声学文件。

1. Language Data准备语言数据

在kaldi-master/data/local文件夹下创建一个dict文件夹，语言数据主要包括 lexicon.txt 、nonsilence\_phones.txt、silence\_phones.txt、 optional\_silence.txt，将这四个文件保存到dict文件夹下。同时因为我们目前做的demo和kaldi教程文档中的数据集的格式是一样的，所以下列这些数据文件可以按照文档中的来编写，也就是下面所展示的截图。（PS: 该截图均来自于Kaldi官方文档）

1. lexicon.txt

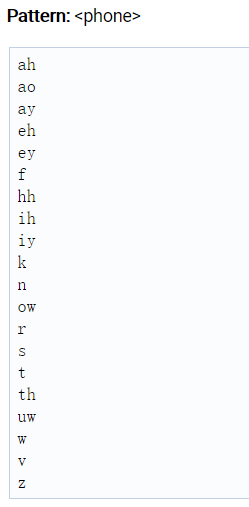
备注：This file contains every word from your dictionary with its 'phone transcriptions’ （这个文件中包括了你语音中出现的所有词的发音，也就是音素表达）



1. nonsilence\_phones.txt

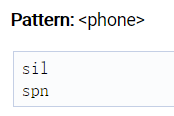
备注：This file lists nonsilence phones that are present in your project.

（这个文件包括了上面所有的非静音音素，同样，下面提供本次所需要的非静音音素。）



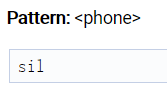
1. silence\_phones.txt

备注：This file lists silence phones.（这个文件包括了上面所有的静音音素）



1. optional\_silence.txt

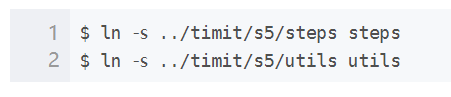
备注：This file lists optional silence phones.（这个文件包含可选的静音音素）



1. Tools attachment准备依赖工具

备注：You need to add necessary Kaldi tools that are widely used in exemplary scripts.

通过创建软连接或者复制其他kaldi项目中的utils和steps文件夹放到根目录下，如：



或者

cp -r ../yesno/s5/utils ./

cp -r ../yesno/s5/steps ./

1. Scoring script

备注：This script will help you to get decoding results.

同理我们可以将kaldi其他例程中的local下的score.sh复制到我们的local文件夹下（PS：该local文件夹就处于根目录下）

cp ../voxforge/s5/local/score.sh ./local/

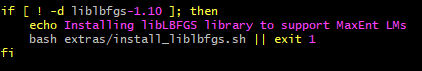
1. SRILM installation

备注：You also need to install language modelling toolkit that is used in my example - SRI Language Modeling Toolkit (SRILM).

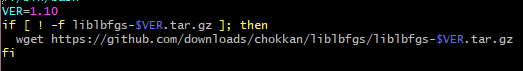
（官方教程中要求安装这个，但是我看其他的教程中有是否安装这个好像问题不大）

在此简述下我的安装流程：

1. 在tools下执行install\_srilm.sh
2. 根据出现的提示查看install\_srilm.sh脚本中是走到哪一步出现了问题
3. 首先可能会走到这一步出现问题

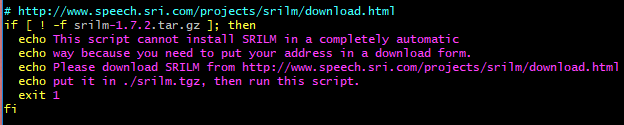


其中的install\_liblibfgs.sh脚本中我们可以看一下



其中的下载链接因为我们的服务器无法访问外网就需要自己手动下载完成后将其上传至tools文件夹，（注意该脚本是在tools文件夹下执行的，所以其下载的liblbfgs-1.10文件必须上传到tools文件夹下。

1. 其次执行到这一步会出错

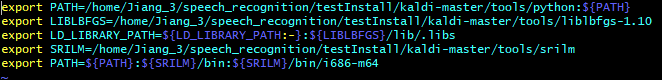


其中的srilm-1.7.2.tar.gz是我自己去官网下的，在此之前这里写的srilm.tgz,所以我们需要去注释所给的网站下载srilm版本文件，并将脚本文件中的srilm.tgz换成我们所下载的文件名

1. 最后执行install\_srilm.sh脚本会显示安装成功



只需要执行一下env.sh即可将其添加路径到path.sh中，可以看下env.sh脚本



就是添加相应的路径

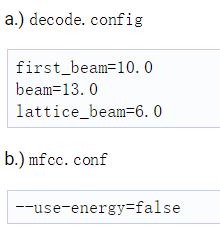
注意：此处最好执行下env.sh脚本，否则会说srilm还是没安装

1. Configuration files配置文件

备注：It is not necessary to create configuration files but it can be a good habit for future.

1. In kaldi-trunk/egs/digits create a folder conf.
2. Inside kaldi-trunk/egs/digits/conf create two files (for some configuration modifications in decoding and mfcc feature extraction processes - taken from /egs/voxforge):

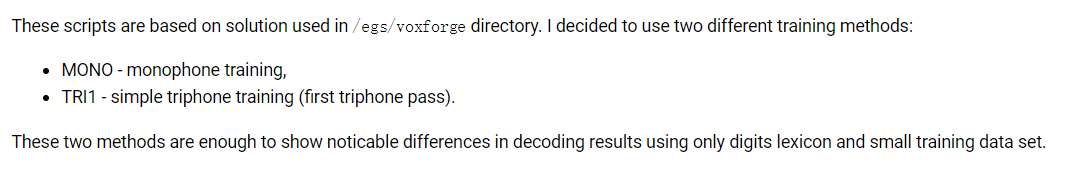
大致内容如下，可以复制其他项目中的也可以自己写



注意文档中的配置文件需要写清楚，之前因为写错了mfcc.conf导致一直没发现错哪了，后来看日志文档才发现错误。

1. Running scripts creation创建运行脚本

备注：根据官方文档教程，知道后面脚本的内容



1. 编写cmd.sh（不使用外部集群）



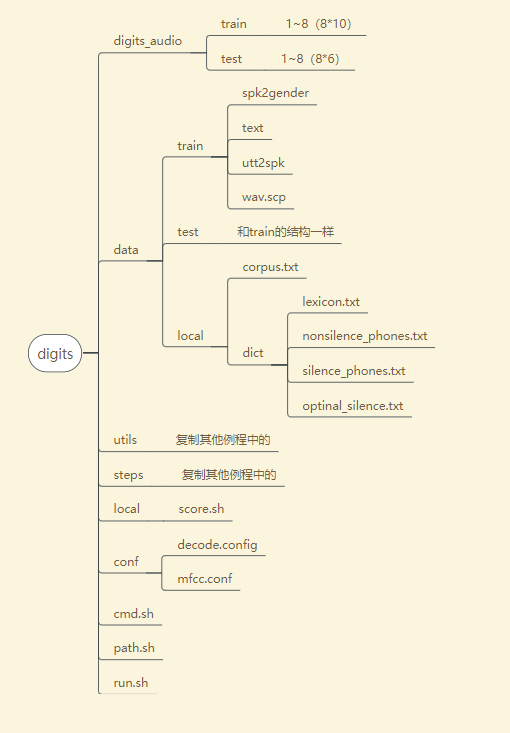
1. 编写path.sh

参考文档上的path.sh的写法，注意修改路径

1. 编写run.sh

参考文档上的run.sh的写法

在完成以上所有步骤后，其文件结构大概如图所示：



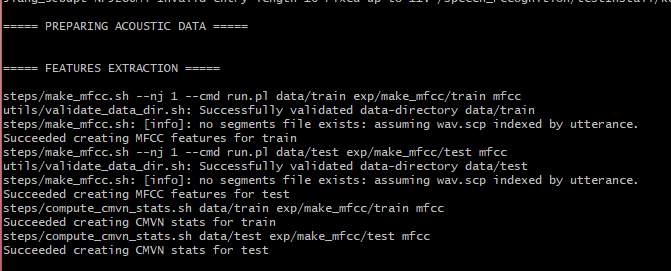
1. 执行run.sh 脚本

sudo bash run.sh

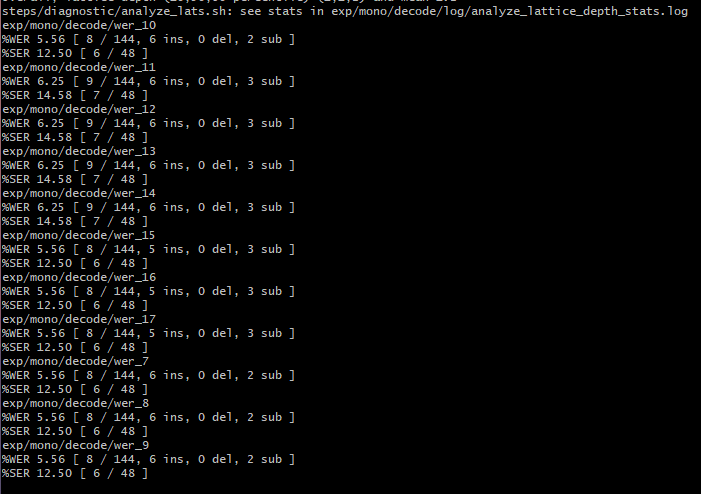
备注：在ubuntu 16.04中为了sh的迭代计算次数有限，故此处需要使用bash 。

可以看到效果如下：

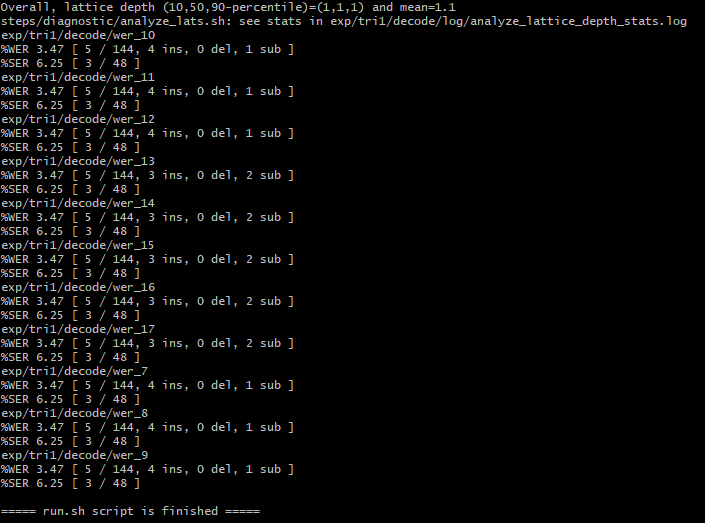
数据准备和特征提取部分：（这部分很重要！！）



MONO模型（单音素模型）识别结果：



TRI1模型（三音素模型）的识别情况：



备注：WER：词识别错误率

SER：句子识别错误率

可见TRI1模型的识别率是好过MONO的。

总结：该文档跟着kaldi官方教程中，如何利用自己的数据集来完成一个实验的，但是其中我使用的两个模型还是来源于kaldi中的。做完了这个demo，对kaldi中的数据准备部分有了更多的认识以及对一些运行中的错误有了更多了解，但是其中涉及到的很多声学名词还需补补。