Project 2：Kaldi 的安装和基本使用

520030910334 张涵雪

1. Kaldi 的安装

1.1. Kaldi 源码编译过程

首先将代码 git clone 到本地，然后按照 install 中的提示一步步运行。

首先运行：

```
cd kaldi/tools/extras
sh check_dependencies.sh
```

得到缺少包的信息，并按照提示一步步安装，没有什么 brew 什么即可，如：

```
#以提示“sox is not installed.”为例
brew install sox
```

其中 MKL 和 libtoolize 会给出两个网址，libtoolize 可通过 wget 下载后解压缩并配置编译安装，MKL 只有在需要 python3.7 的情况下才需要安装，linux 系统可直接使用指令：

```
sh install_mkl.sh
```

此后的安装过程都比较简单：

```
cd ../
make    #这一步会很久
cd ../src
./configure
make depend     #这一步也会很久
make            #这一步也会很久
make install
```

测试是否安装完成：

```
cd kaldi/egs/yesno/s5
sh run.sh
```

显示得到：

```
local/score.sh: scoring with word insertion penalty=0.0,0.5,1.0
%WER 0.00 [ 0 / 232, 0 in , 0 del, 0  ub ] exp/mono0a/decode_te t_ye no/wer_10_0
.0
```

表示测试 232 个词结果全部正确。

1.2. 遇到的困难

在该过程中，由于对于配置环境比较熟悉，基本没有遇到什么困难。配置环境大概是一旦熟了就不会出什么问题的过程，关键是要熟悉 linux 操作。

1.3. Kaldi 的基本结构

在 Kaldi 的一级主目录中包括：egs、misc、scripts、src、tools、windows。

- egs: Kaldi 的实例目录，其中例子包含语音识别、语种识别、声纹识别、关键字识别等。
- misc： 此目录包含了一些相关 paper 的汇总、以及相关 docker、htk 等资源。
- tools：存放 Kaldi 依赖库安装脚本。
- src： Kaldi 的源码目录，主要保存了包括 GMM、HMM 等在内的大部分 Kaldi 语音项目源代码，其中有两类文件夹，一类是算法原目录，一类为算法组合生成的可执行程序目录。
- windows：在 Windows 平台运行所必须的脚本以及相关的执行程序。
- scripts：用来存放 Rnnlm，以及相应的运行脚本。

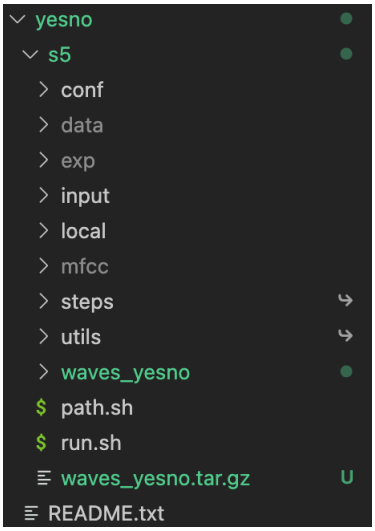
其中在我们这次的项目中最重要的就是 egs 和 src 目录，相当于对 egs 中例子的黑箱调用，具体 egs 的运行过程及目录结构会在后面详细说明。

2. Kaldi 的基本使用

在 task2 中我本来想跑 thchs30 识别项目，但是由于数据量太大且时间限制遂放弃；又准备跑 an4 样例，但数据下载的网址失效了。最终还是选择了最简单的 yesno 样例。

2.1. 示例程序的基本结构

yesno 下的 s5 目录如下：



其中 conf 文件是该项目的参数设置文件；data 文件夹用来存放语言模型、发音字典和音素信息等等；waves_yesno 是原始音频目录；steps 和 utils 分别包含训练中某些步骤和需要使用的工具的脚本；local 中包含了该实例一些特需的脚本；exp 文件夹包含相关 log 文件，模型训练完后，声学模型被存放在 exp/mono0a 里；解码测试输出到 exp/mono0a/decode_test_yes；path.sh 通过设置路径来设置环境变量；mfcc 文件夹用于存储提取到的音频特征；run.sh 是对于项目的黑盒封装，是使用项目的接口。

下面进行解读 run.sh 中的代码：

- 下载数据并准备需要工具和路径。
- 数据预处理
  - prepare_data.sh：划分数据集并创建 data 文件夹，把划分好的音频文件放入 train_yesno 和 test_yesno 并转换成 Kaldi 能处理的格式（Text、wav.scp、utt2spk、spk2utt）
  - prepare_dict.sh：建立词典，当前项目的词典中有三个词，Ken(yes) 、Lo（no）和 SIL，该脚本将 input 下的文件 cp 到词典目录下并添加 SIL。
  - prepare_lang.sh：依据我们建立的词典构建字典 FST 脚本，其中有多个选项，包括是否所有静音共享一个 pdf、是否将 phone 更详细拆分、是否存在同音等，生成语言模型。
  - prepare_lm.sh：将语言模型转换成 G.fst 格式并保存在 data/lang_test_tg 目录下。
- 特征提取
  - make_mfcc.sh：提取梅尔倒谱系数
  - compute_cmvn_stats.sh：倒谱均值方差归一化，提取声学特征以后，将声学特征从一个空间转变成另一个空间，使得在这个空间下更特征参数更符合某种概率分布，压缩了特征参数值域的动态范围，减少了训练和测试环境的不匹配等。
  - fix_data_dir.sh：确保 spk、utt、text 等对应。
- 单音素模型训练：train_mono.sh
  - 构建决策树和初始 GMM，构造初始模型。
  - 构建文本状态图。
  - 强制对齐，把各帧匹配到状态图上。
  - 用 Viterbi 更新模型（高斯核数目），使用 EM 算法迭代更新 GMM 参数。
  - 训练至模型收敛，生成.mdl 声学模型文件
- 创建解码器并保存：mkgraph.sh
- 解码和测试：使用 decode.sh 对测试集进行解码并且计算 Word error rate（WER）

2.2. 运行结果展示

```
local/score.sh: scoring with word insertion penalty=0.0,0.5,1.0
%WER 0.00 [ 0 / 232, 0 in , 0 del, 0  ub ] exp/mono0a/decode_te t_ye no/wer_10_0
.0
```

说明字错误率为 0，即准确率为 100 测试集一共 29 条音频，每条音频有 8 个单因素字，一共 232 个字。

3. Project2 总结

在此次任务中，我认为主要的收获是学会了 kaldi 的黑箱调用，这一部分难度不大，只需掌握一些 linux 指令即可明白脚本中每一步在做什么，但是如果深入去了解更底层脚本的细节那么就需要一些知识的摄入，包括但不限于各种算法及一些数据结构的实现。目前对 kaldi 的感觉就是一个语音工具箱，可以提供一些语音处理的工具和模板，当然我们也可以搭建自己的语音识别系统，甚至可以通过更改源码实现新的模型架构，这需要进一步的掌握与熟练。