Project 3：LVCSR 系统搭建

520030910334 张涵雪

1. Baseline (70 分)

1.1. 数据处理及特征提取 (10 分)

官方 recipe 中数据处理与特征提取使用了以下脚本：

- 准备字典相关文件。

```
aishell_prepare_dict.sh
```

- 读取语料库，得到 wav.scp、音标，生成文本 text、wav.scp、uut3pk 及 spk2utt。

```
aishell_data_prep.sh
```

- 生成字典模型和 L.fst。

```
prepare_lang.sh
```

- 生成语言模型和 G.fst。

```
train_lms.sh
format_lm.sh
```

- 生成梅尔倒谱系数并均值方差归一化（提高声学特征对说话人、录音设备、环境、音量等因素的鲁棒性）并修复排序错误，最后移除被指明需要特征数据或标注的找不到被需要的数据的那些发音。

```
make_mfcc_pitch.sh
compute_cmvn_stats.sh
fix_data_dir.sh
```

过程进行得比较顺利，没有遇到什么问题。

1.2. 模型训练 (50 分)

训练过程解释如下：

- 首先训练单音子隐马尔科夫模型（train_mono.sh），然后对其进行测试（使用 mkgraph.sh 建立完全的识别网络，输出一个有限状态转换器，再使用 decode.sh 以语言模型和测试数据为输入计算 CER 和 WER），用训练好的模型将数据进行强制对齐方便以后使用。
- 对上一步的结果进行两轮三音子模型的训练和测试，过程和上述类似，并用训练得到的三音子模型来做强制对齐。
- 对上一步的结果进行特征扩帧，使用 LDA 降维，然后经过多次迭代轮数估计一个对角变换得到新的特征（MLLT），进行三音子训练、解码、测试，对结果进行基于特征空间最大似然线性回归对齐（align_fmllr.sh）。
- 对上一步的结果进行说话人自适应训练（对每个说话人分别估计说话人信息，期望在此基础上能够真正得到说话人无关码本，核心任务是对特征码本做 FMLLR），对自适应模型解码、测试、做 FMLLR 对齐。
- 增加叶节点数和高斯数继续训练，并进行解码、测试、对齐，得到一个更大规模的 SAT + GMM-HMM 模型。

由于该过程解码过慢，我想在服务器配置 kaldi 运行，但是遇到了很多困难：服务器没有 sudo 权限问题、服务器不能连接外网导致无法 git clone 只能从本地 scp 文件的问题，本地下载路径和服务器不同而导致的服务器 mkdir 权限问题等等。

1.3. 实验结果 (10 分)

不同过程中路径如下 (以 cer 为例)：

- train_mono：

```
exp/mono/decode_test/scoring_kaldi/
    cer_details/cer_bootci
```

- train_deltas：

```
exp/tri1 (tril2) /decode_test/scoring_kaldi/
    cer_details/cer_bootci
```

- train_lda_mllt：

```
exp/tri3a/decode_test/scoring_kaldi/
    cer_details/cer_bootci
```

- SAT system：

```
exp/tri4a (tri5a) /decode_test/scoring_kaldi
    /cer_details/cer_bootci
```

绘制表格如下：

表 1: *SAT + GMM-HMM* 各阶段结果表

阶段	CER	WER
mono	46.21	58.70
deltas_1	28.68	43.87
deltas_2	28.71	43.84
LDA+MLLT	26.05	41.48
SAT_{small}	20.80	36.36
SAT_{big}	22.52	38.17

其中，我们展示的 SAT 是说话人相关特征解码的结果，倘若仍然使用 SI 解码，各指标反倒变差，直观上是因为我们希望最终的结果在加入说话人特征时更容易移动到最优点，因此在全局来看离任何一个说话人都不太近，符合我们的预期。

2. 语料数量对模型性能的影响 (15 分)

2.1. 修改方法

local/aishell_data_prep.sh 第十五行需要修改为：

```
aishell_text=$2/train_large.txt
```

更换语料库后所有的制图和解码的步骤都需要重新运行。

2.2. 性能对比

表 2: 不同语料规模性能对比

语料规模	CER	WER
small	20.80	36.36
large	15.28	25.27

3. 训练 DNN-HMM 系统 (15 分)

3.1. 训练流程

流程如下：使用 SAT + GMM-HMM 结果对齐得到的数据作为 DNN 的输入进行训练 (train_dnn.py)，然后解码并测试。

在这一步我遇到了困难，主要是环境的问题，我在使用了助教提供的脚本后 (run_tdnn.sh) 出现了格式错误，在原先的脚本并不会出错，最终在这一步更换超算环境得以解决。

3.2. 性能对比

表 3: *DNN-HMM* 性能对比

模型	CER	WER
GMM-HMM	20.80	36.36
DNN-HMM	18.65	34.10

4. 优化系统 (20 分)

4.1. 方法

在原先 SAT + GMM-HMM 模型的基础上加入 n-gram 语言模型，采用 srilm 工具进行训练、解码、测试，通过调整声学权重和语言权重得到最佳结果。其实这一步有很多优化可以做，因为模型的剪枝合并自由度很高，但是由于时间限制导致没有彻底完成。

4.2. 性能对比

表 4: *n-gram* 语言模型性能对比

模型	CER	WER
GMM-HMM	20.80	36.36
GMM-HMM_{n-gram}	??	??

由于时间和算力限制，这一步还没有跑完，只是一个想法，目前只有代码，没有结果。

5. 最佳性能

CER=15.28%，WER=25.27%