语音识别实验报告

无 63 徐远帆 李明哲 李加捷

一、 背景调研

语音是人类最重要和最有效的交互方式，准确的语音识别技术可以极大地提高人机交互的效率。自动语音识别（Automatic Speech Recognition，ASR）是指利用数学模型，通过计算手段，将语音信号转化为对应文字的过程。语音识别是一个复杂而综合的研究方向，需要结合信号处理、信息论、模式识别与机器学习、自然语言处理、高性能计算等学科的知识加以研究。语音识别涉及大量专业知识和充满挑战的编程，因此我们采用已有的语音识别开源框架 Kaldi 和深度学习工具 Tensorflow 来构建整套可供实际演示语音识别系统。

语音本身是一维变长信号，同时可以包括噪声、人声等大量信息。输入的语音信号首先会被处理系统参数化并抽象、变换为具有一定统计性质的特征，这个过程称为特征提取。为将特征与输出结果对应，需要训练语音识别模型，其中包括声学模型和语言模型。声学模型包含一个发音字典，使读入的文本标注细化为发音序列，计算机在不同的层级训练发音与特征的对应关系得到模型。而语言模型通过上下文内容来判断当前时刻不同词汇出现概率，以确定相似发音的字。本次实验重点关注基于深度学习的声学模型训练和测试，不涉及语言模型的训练。

二、 方法描述

为完成本次实验，首先在 Linux 系统下安装 Kaldi 工具箱和下载数据集 THCHS-30。

1. 特征提取

计算机读入语音音频后，首先对语音进行编码得到语音特征。由于语音是时变信号，需要对语音分帧，每帧长度约为 25ms，帧移为 10ms。再通过傅里叶变换等方法提取声学特征，包括线性预测参数 LPC、感知线性预测系数 PLP、梅尔倒谱系数 MFCC 等。MFCC 是 Kaldi 中最常用特征，在 run.sh 中，steps/make_mfcc.sh 用于生成语音

对应的 MFCC 参数。

2. 语言模型训练

常用的语言模型是 N-gram 模型。N-gram 语言模型通过上文已经出现的 N-1 个词，描述第 N 个词的出现概率。THCHS-30 中提供了 extra_questions.txt，lexicon.txt（发音字典），nonsilence_phones.txt，silence_phones.txt，optional_silence.txt 使 Kaldi 生成语言模型。

3. 声学模型训练：基于高斯混合模型（GMM）和隐马尔可夫模型（HMM）

$$P(\mathbf{O}|\mathbf{w})=\sum_{Q} p(\mathbf{O}|Q)P(Q|\mathbf{w})$$

对于一段语音，其在对应文本标注上的似然概率可以表示为：

为计算这个概率，一般采用 HMM 建模，主要通过转移概率和输出观测概率分布刻画。

基于 GMM 的隐马尔可夫模型，指的就是 HMM 中的观测概率用 GMM 建模。通过利用训练语音的特征和其对应的合成 HMM，更新所有 HMM 参数（状态转移概率，状态对应高斯混合模型的均值、方差以及权重等）。迭代这个过程最终就可以找到该 HMM 的局部最优解。

4. 得到各发音的初始 HMM

经典的做法是采用 flat start，即由大量语音计算均值、方差，所有 HMM 状态都采用由这些均值、方差初始化的高斯分布，并将状态间的转移概率设为相等。经过若干次迭代，HMM 定义将逐渐趋于合理。此时的发音单元都是单一的汉语音素，不考虑上下文的连接关系，因此这种模型被称为单音素（monophone）模型。

单音素建模不能反映相同音素在上下文不同时，其发音的变化。为了解决这个问题，对于每个音素，同时考虑其左边与右边的近邻，构成一个新的发音单元，由此得到的模型称为三音素（triphone）模型。对于三因素模型需要大幅降维，否则单个模型的训练特征会极少。因此需要根据三音素模型中不同 HMM 状态之间的相似程度进行聚类，对类似的状态进行绑定，聚类采用决策树分裂的方法。

在此基础上，在 THCHS-30 对应的脚本中还采用了特征 LDA 降维、最大似然线性变换 MLLT、说话人自适应训练 SAT 等技术。这些变换尽可能减小不同说话人在特征域

与模型域中分布的差别，从而提高对不同人的识别效果。

5. 基于深度神经网络的声学模型训练

将 HMM 结构中的高斯混合模型替换为深度神经网络。对一段输入特征，一个训练良好的神经网络可以将其分类到对应的发音单元上，且网络的输出是每个发音单元的后验概率。训练神经网络时，采用反向传播算法（BP）更新参数，采用常见的交叉熵代价函数。

首先得到每一帧特征对应的发音单元，利用 train 子集训练神经网络，利用 dev 子集生成验证集防止模型过拟合。但是每一帧分类结果最优并不代表整段语音解码最优，MPE 使通过神经网络模型解码后得到的整段语音结果与参考标注之间的音素错误率最小。

三、 实验步骤

1. 配置环境——安装 Kaldi、下载 THCHS30 数据集

从 github 上 clone 下来工程文件，根据文件夹中的 INSTALL 配置环境、安装所需依赖和编译，Kaldi 安装完成后，在 http://www.openslr.org/18/上下载 THCH30 数据集，解压并放在对应的地方，更改脚本中的路径。

2. 预训练——先按照实验指导书做一遍（熟悉流程及相关知识）

按照实验指导书要求，修改 run.sh 脚本，在 shell 中运行，进行预训练，此步骤不求得到正确结果，主要熟悉实验指导书以及训练流程，记录下出现的 error，查找解决方式。

3. 正式训练——结合预训练中出现的问题修改脚本（结合自身情况修改方案和步骤）

根据预训练中出现的问题，对应的修改 run.sh 脚本，比如修改核数和修改数据处理部分，在利用较为完整的时间，运行 run.sh 脚本，此过程重在得到正确的训练模型。

4. 分步训练——保留训练模型，解码验证结果

在正式训练中得到了完整正确的模型，但是解码过程仍因内存不足出现 error，因此在脚本中注释掉相应的训练、数据预处理部分，只留下 decode 解码部分，逐个模型解码并检验是否训练正确，此过程耗时很长，旨在得到解码结果，判断模型是否训练正确。

5. 图形界面制作——利用训练好的模型进行语音识别，利用 Qt 编写可视化界面

首先是根据 run.sh 脚本中的代码写出自己的各个模型的脚本，我们小组为 5 个模型写了 5 个脚本，每个脚本主要是模型，解码地址不同，其他均一致。

采用 qt5 编写图形界面，根据界面上的单选框决定调用哪一个脚本，调用 qt5 里的系统命令 system 来实现外部脚本的运行，最后从输出的 txt 文件里读出内容并显示到文本框。

为了适应 qt5 的脚本执行，对自己编写的脚本做了一些修改，比如加上几条 cd 命令。

四、 问题与解决方案

1. 选取的 CPU 核数

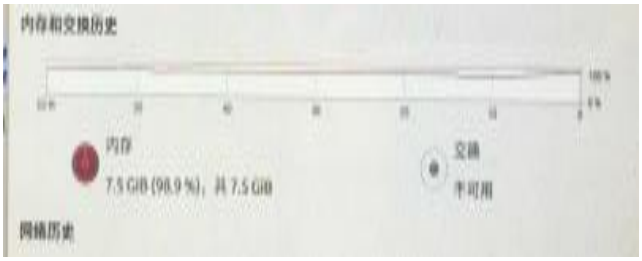
对应于脚本中指定的 -j --nj

安装过程、训练过程中均出现了电脑卡顿、直接卡死等问题。解决方案是减小 CPU 核数，牺牲速度来换取成功率。

2. 内存不足

对应于脚本中的&(后台运行)，ERROR: FstHeader::Read: Bad FST header:

解决方案：删除&，并将训练和解码分步进行



3. 识别结果准确率低

对应于 tri3b、tri4b 模型的解码方式和文件储存位置。解码方式不同、解码后保存的位置不同。

```
#test monophone model
local/thchs-30_decode.sh --mono true --nj $n "steps/decode.sh" exp/mono data/mfcc


#test tri3b model
local/thchs-30_decode.sh --nj $n "steps/decode_fmllr.sh" exp/tri3b data/mfcc
```

五、 成果结论

1. 训练结果

data、mfcc——数据、特征

exp——模型

mono —— 单音素模型

tri1 —— 上下文相关的三音素模型

tri2b —— 加入 LDA 和 MLLT 的三音素模型

tri3b —— 基于特征空间的最大似然线性回归（fMLLR）进行说话人自适应训练的模型

tri4b —— 综合以上所有特征训练得到的模型

final.mdl —— 训练得到的模型

phones.txt —— 音表

words.txt —— 词表

运行 RESULTS 脚本：

%WER 50.75 [41174 / 81139, 557 ins, 2193 del, 38424 sub]

exp/mono/decode_test_word/wer_8_0.0

%WER 36.23 [29398 / 81139, 548 ins, 1074 del, 27776 sub]

exp/tri1/decode_test_word/wer_10_0.0

%WER 32.30 [26205 / 81139, 421 ins, 1023 del, 24761 sub]

exp/tri2b/decode_test_word/wer_10_0.0

%WER 29.38 [23840 / 81139, 373 ins, 859 del, 22608 sub]

exp/tri3b/decode_test_word/wer_10_0.0

%WER 33.59 [27256 / 81139, 435 ins, 1087 del, 25734 sub]

exp/tri3b/decode_test_word.si/wer_9_0.5

%WER 27.92 [22653 / 81139, 416 ins, 764 del, 21473 sub]

exp/tri4b/decode_test_word/wer_10_0.5

%WER 31.22 [25329 / 81139, 530 ins, 841 del, 23958 sub]

exp/tri4b/decode_test_word.si/wer_9_0.0

主要用到的脚本代码参考了 thchs30 文件中给出的 run.sh 脚本

1. local/thchs-30_data_prep.sh //数据准备

2. steps/make_mfcc.sh steps/compute_cmvn_stats.sh

//进行特征提取

3. local/thchs-30_decode.sh

//解码

4. demo.sh

//从.lat 文件解出 txt 文档

2. 自采数据的识别

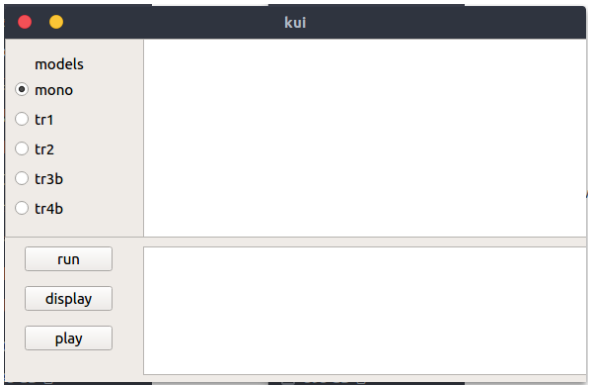
```
cd /home/li/kaldi/egs/thchs30/s5/
. ./cmd.sh
. ./path.sh
H=`pwd`
n=1

thchs=/home/li/kaldi/egs/thchs30/s5/thchs_data;
local/thchs-30_data_prep.sh $H $thchs/data_thchs30 || exit 1;
cd /home/li/kaldi/egs/thchs30/s5/
rm -rf data/mfcc && mkdir -p data/mfcc &&  cp -R /home/li/kaldi/egs/thchs30/s5/data/train /home/li/kaldi/egs/thchs3
for x in  test; do
   #make_mfcc
   steps/make_mfcc.sh --nj $n --cmd "$train_cmd" data/mfcc/$x exp/make_mfcc/$x mfcc/$x || exit 1;
   cd /home/li/kaldi/egs/thchs30/s5/
   #compute_cmvn
   steps/compute_cmvn_stats.sh data/mfcc/$x exp/mfcc_cmvn/$x mfcc/$x || exit 1;
done
cd /home/li/kaldi/egs/thchs30/s5/
#copy feats and cmvn to test.ph, avoid duplicated mfcc & cmvn
cp data/mfcc/test/feats.scp data/mfcc/test_phone && cp data/mfcc/test/cmvn.scp data/mfcc/test_phone || exit 1;

local/thchs-30_decode.sh --nj $n "steps/decode.sh" exp/mono data/mfcc
cd /home/li/kaldi/egs/thchs30/s5/
export PATH=$KALDI_ROOT/src/latbin:$KALDI_ROOT/egs/wsj/s5/utils:$PATH

echo $PATH
lattice-1best --acoustic-scale=0.1 'ark:gunzip -c exp/mono/decode_test_word/lat.1.gz |' ark:- | nbest-to-linear ark
```

我们选取了“媒体与认知真有趣”与“今天是 5 月 30 日，媒体与认知的展示将在主楼进行”作为自采数据，得到的结果分别是“媒体 人士 这 有趣”与“经纪人 是 五月 三十 枚 其余 人质 合成 的 展示 将 在 政府 楼 进行”

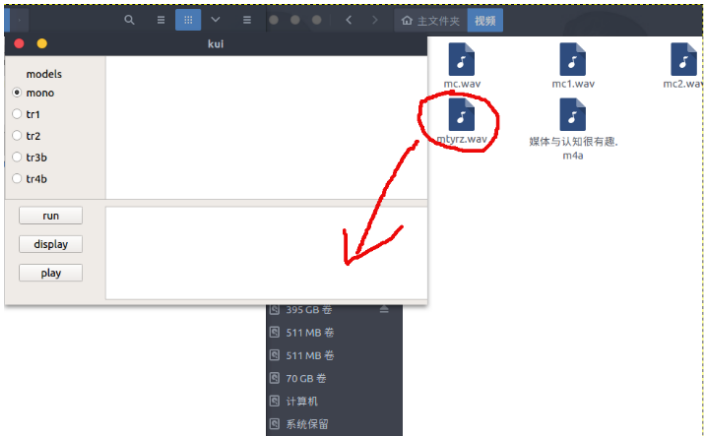
3. 图形界面展示

本例对自采集语音数据“媒体与认知真有趣”进行识别，并通过编写的图形界面展示解码识别结果。

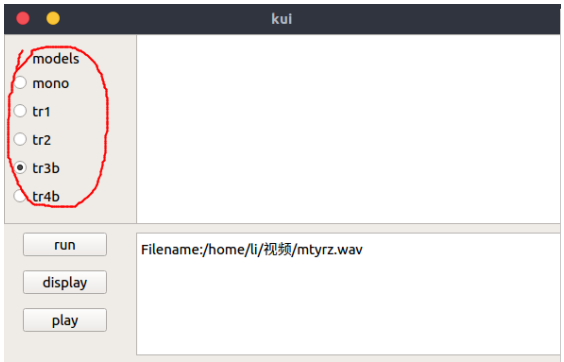
a. 图形界面概览：左侧可选择采取的模型种类



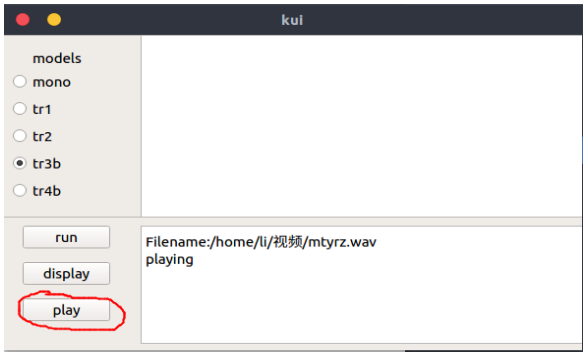
b. 将采集到的 wav 文件拖入图形界面下方对话框，作为输入语音信号



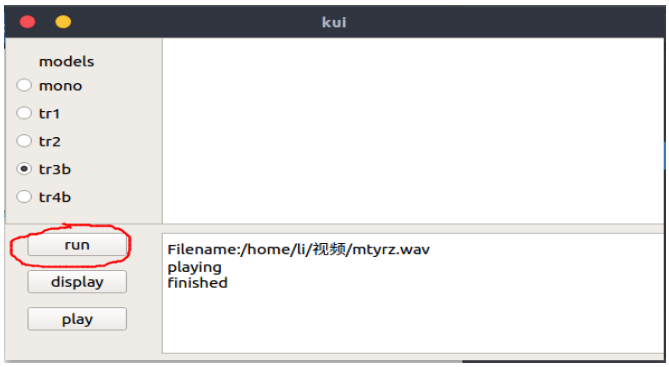
c. 左侧勾选 tr3b：基于特征空间的最大似然线性回归（fMLLR）进行说话人自适应训练的模型



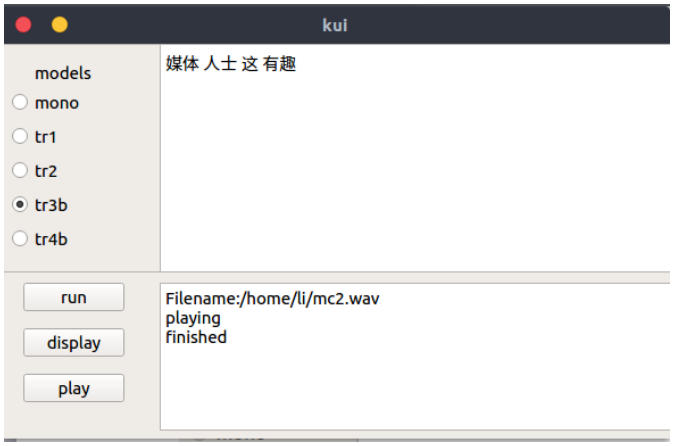
d. 点击 play，进行音频的播放



e. 点击 run，执行脚本，得到输出



f. 解码结果：“媒体 人士 这 有趣”，与输入语音相比，正确率处于可以接受的范围。可以通过识别结果基本确定语音含义。



六、 思考讨论与实验总结

(1)这次媒认的大作业是我们小组所有成员第一次大型的项目，从环境配置、运行脚本到最后图形界面制作以及成果展示，几乎经历了完整的项目开发的流程，虽然可能很多时候主要依据实验指导书来操作，但其中出现的很多 bug 或者是设计问题都由自己解决，可谓是收获良多，不仅是知识上，更是完成大型工程项目的素养上。

(2)这次实验耗时和跨度都很长，所以尤其训练了我们组织、规划和合作的能力，窃以为我们小组应该是分工很明确、任务执行力很强的一个小组，每个人都很好的完成了自己分内的工作，基本上只需提供对应的“接口”，对应的同学就能高效的完成自己的任务。

(3)在训练过程中，给我们最大的启发和教训就是在完成比较大型的项目时，一定要有提前的设计和规划，最好能分步进行，每一步都留下对应的 log 和副本，这样就避免了做很多的重复工作。

(4)自采数据识别与图形界面编写需要对 kaldi 的运行流程有大致的了解，由于 kaldi 提供了很多必要的脚本，比如音频格式转化，解码与结果输出等，大大简化了脚本编写过程。在熟悉了如何运用一个已训练好的模型之后，也能对模型的编写原理有更进一步的认识。

七、 组内分工

1. 数据准备、环境配置——徐远帆、李加捷
2. 模型训练、解码识别——徐远帆、李明哲
3. 图形界面制作——李明哲
4. 基于 WaveNet 的声学模型——徐远帆
5. 实验报告撰写——李加捷