语言分析与机器翻译课程报告

姓　　名：王 杰

专　　业：计算机科学与技术

学　　号：2101798

任课教师：肖 桐

ASR 问题描述

语音识别技术（Automatic Speech Recognition，ASR）作为人工智能领域里的关键技术，在实现人机交互上有着重要的作用。其主要任务目标是把一段音频中的语音信号转换为相应的文本。早在上世纪 50 年代，研究人员就已经开始了对语音识别的探索。早期的语音识别研究只针对少量孤立词汇，后来基于 GMM-HMM 框架的语音识别占据了主导地位。这种传统的语音识别模型类似于统计机器翻译，要联合利用声学模型、语言模型和发音词典进行识别，比较复杂。因为语音识别的准确度和性能没有达到可应用级别，所以对此领域的研究陷入了瓶颈期。

深度学习的发展让语音识别迎来了新生。基于 RNN 和 CNN 等深度学习框架的语音识别模型在实际应用中获得了较好的性能，而且模型的训练流程也得到了简化。近年来端到端的语音识别模型成为研究热点，其大多是基于序列到序列的结构。语音的声学特征输入到编码器中并提取出高级特征，解码器利用这个高级特征识别出对应的文本。

本项目在 Fairseq 框架的基础上，设计并实现了基于 Transformer 的端到端 AS 系统，该系统可以完成由英语语音到英语文本的识别。

背景知识

音频处理，不同于文本，音频本质上是经过若干信号处理之后的波形。具体来说，声音是一种空气的震动，因此可以被转换为模拟信号。模拟信号是一段连续的信号，经过采样变为离散的数字信号。采样是每隔固定的时间记录一下声音的振幅，采样率表示每秒的采样点数，单位是赫兹（Hz）。采样率越高，采样的结果与原始的语音就越相像。通常来说，采样的标准是能够通过离散化的数字信号重现原始语音。日常生活中使用的手机和电脑设备的采样率一般为 16Hz，表示每秒 16000 个采样点；而音频 CD 的采样率可以达到 44.1kHz。经过进一步的量化，将采样点的值转换为整型数值保存，从而减少占用的存储空间，通常采用的是 16 位量化。将采样率和量化位数相乘，就可以得到比特率（Bits Per Second，

BPS），表示音频每秒占用的位数。例如 16kHz 采样率和 16 位量化的音频，比特率为 256kb/s。音频处理的整体流程如图 2.1 所示。

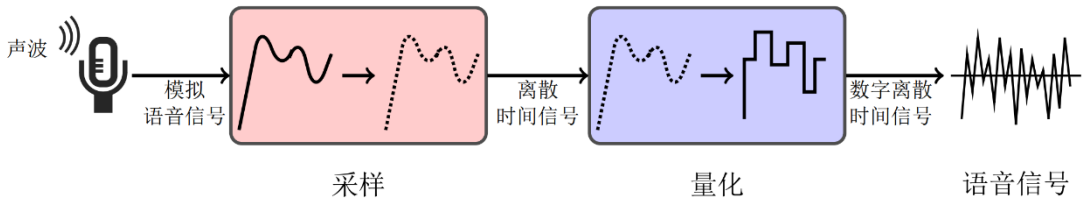

图 2.1 音频处理过程

ASR 模型

端到端的 ASR 模型主要基于序列到序列结构，编码器根据输入的声学特征进一步提取高级特征，解码器根据编码器提取的特征识别对应的文本。相比于文本翻译，本项目使用的 ASR 模型结构上主要的区别在于编码器的输入为声学特征，以及编码器底层会使用额外的卷积层来减小输入序列的长度。这是由于语音对应的特征序列过长，在计算注意力模型的时候，会占用大量的内存和显存，使得训练时间过长。因此在语音特征上进行两层步长为 2 的卷积操作，从而将输入序列的长度缩小为之前的四分之一。

模型的示例如图 3.1 所示，本模型共有 12 层 encoder 和 6 层 decoder，模型的输入为 80 维的 log Mel Fbank 声学特征。声学特征提取的第一步是预处理。其流程主要是对音频进行预加重、分帧和加窗。预加重是通过增强音频信号中的高频部分来减弱语音中对高频信号的抑制，使频谱更加顺滑。分帧是基于短时平稳假设，即根据生物学特征，语音信号是一个缓慢变化的过程，10ms~30ms 的信号片段是相对平稳的。基于这个假设，一般将 25ms 作为一帧来提取特征。这个时间称为帧长。同时，为了保证不同帧之间的信号平滑性，使每两个相邻帧之间存在一定的重合部分。每隔 10ms 取一帧，这个时长成为帧移。为了缓解分帧带来的频谱泄露问题，还会采用汉明窗来对每帧的信号进行加窗处理是的其幅度在两端渐变到 0。接着对得到的数据进行短时傅里叶变换、mel 滤波、去均值等便可以得到实验用的声学特征。

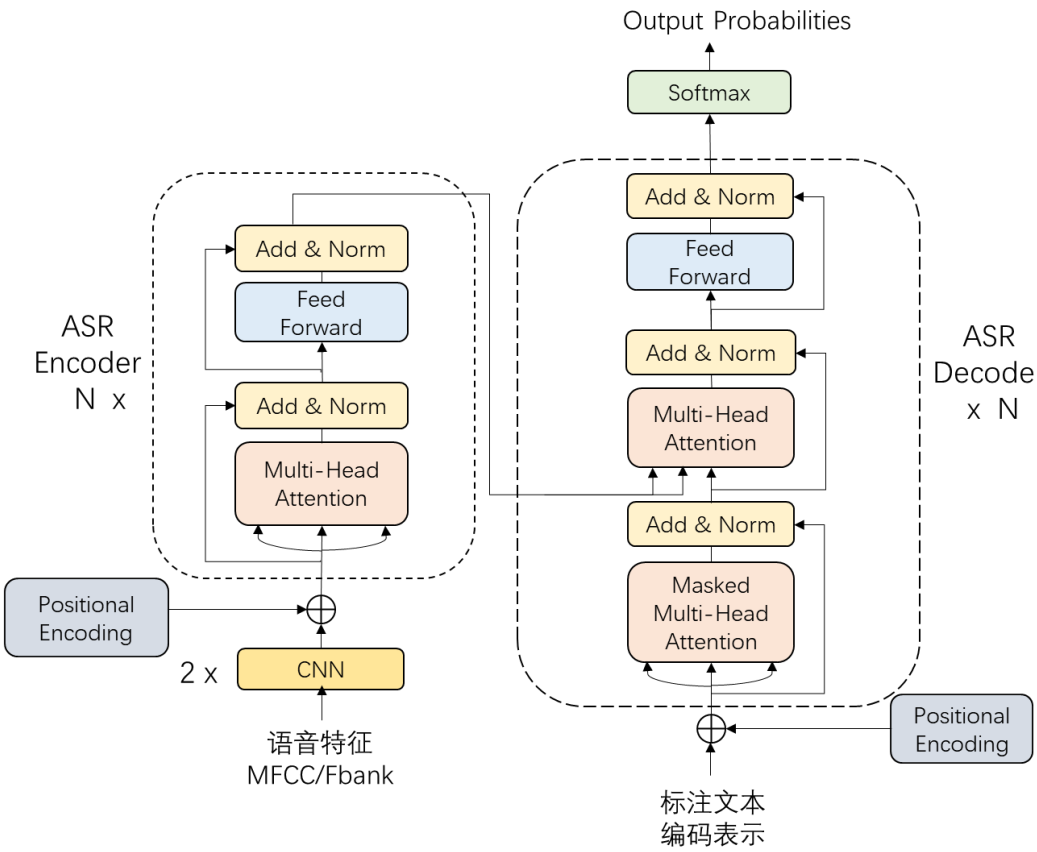

图 3.1 基于 Transformer 的语音识别模型

数据处理

数据处理的处理过程具体为，设置好特征保存路径，对语料库的各个数据集依次进行数据集初始化、特征提取并压缩保存。再生成 tsv 文件，tsv 文件中存储了索引 id、音频特征、持续时间、目标语参考译文、说话者 id 信息。之后生成词表、分词器和 yaml 文件记录数据处理的设置。语音识别的数据处理如图 4.4 所示。

具体操作时，ASR 数据处理使用脚本 run.sh 调用 prep_librispeech_data.py 文件进行数据处理。表 4.1 展示了对于语音识别数据处理的 run.sh 参数以及其含义。

process()函数在设置完特征保存路径和数据集路径后，在数据集各子集中使用 torchaudio.datasets 中的 LIBRISPEECH 类，提取出数据并生成 dataset 列表，之后对 dataset 中每一项数据使用 extract_fbank_features()提取并保存特征到设定路径下，压缩为 zip 格式。将数据编号、特征路径、帧数、目标文本和说话者编号的信息使用 save_df_to_tsv()存储到 tsv 文件中。最后使用 gen_config_yaml()生

成 yaml 文件。

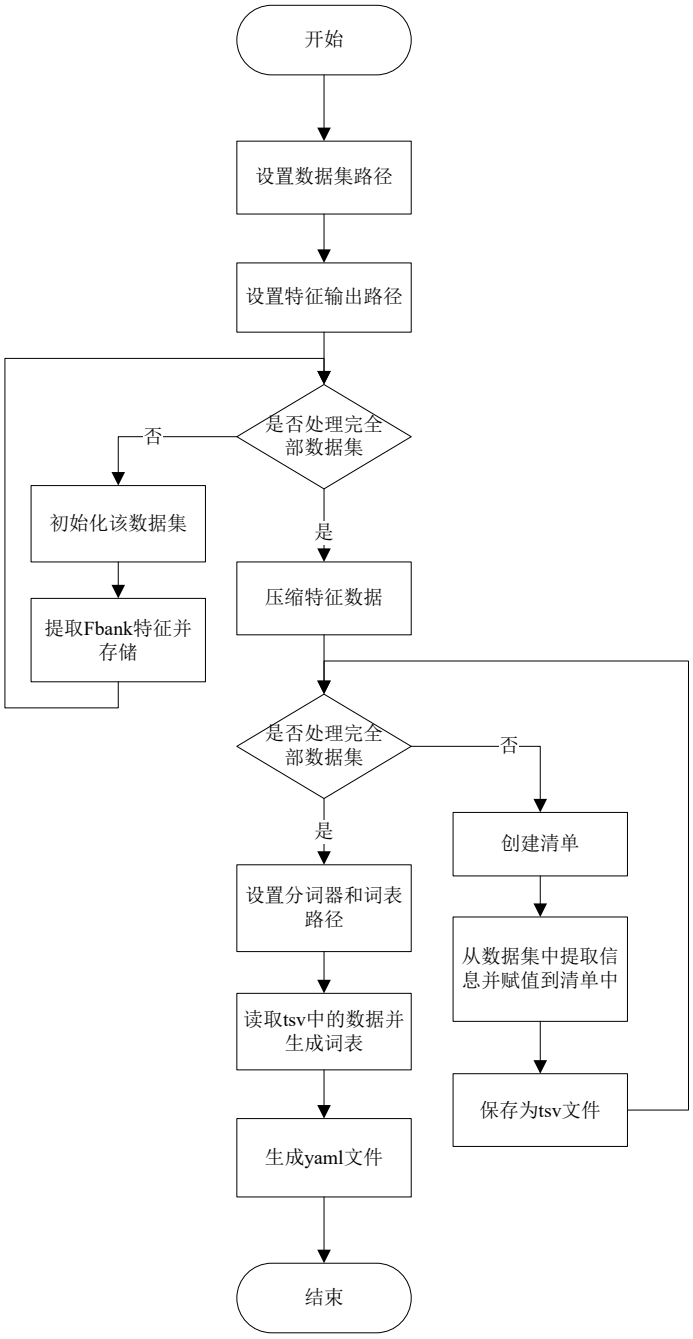

图 4.1 语音识别数据处理流程图

语音识别数据处理后，得到了 test_asr.tsv 等文件、特征文件 fbank80.zip、词表文件，分词器，以及 config_asr.yaml 文件，部分实验截图如图 4.2 所示。

此外，模型还使用了 SpecAugmet 数据增强方法，其中使用了频域掩盖和时域掩盖两种方式。通过在数据处理时设置 specaugment_policy 参数，使用不同的

表 4.1 asr run.sh stage 0 参数

参数名称	参数含义
src_lang	源语言
Dataset	数据集名称
vocab_type	词表类型
vocab_size	词表大小
speed_perturb	是否执行速度扰动
org_data_dir	数据集所在路径
data_dir	处理后数据保存路径
use_specific_dict	是否使用其他模型的词表
specific_prefix	处理数据名称标记
specific_dir	使用词表所在路径
asr_vocab_prefix	ASR 词表文件名称前缀

策略对音频数据进行处理，得到一些具有损失的数据，可以有效防止模型过拟合问题。

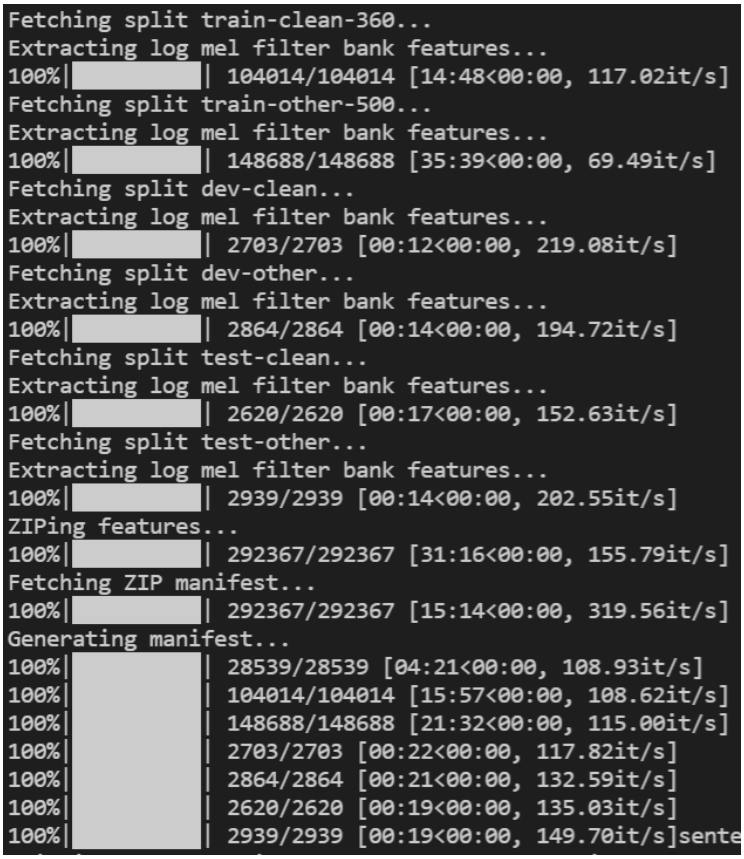

图 4.2 部分数据处理实验截图

实验

1）数据集 LibriSpeech

大小为 60GB，时长 1000 小时左右。采样率为 16kHz。该数据集为包含文本和语音的有声读物数据集，由 Vassil Panayotov 编写的大约 1000 小时的 16kHz 读取英语演讲的语料库。数据来源于 LibriVox 项目的阅读有声读物，并经过细致的细分和一致。经过切割和整理成每条 10 秒左右的、经过文本标注的音频文件。

2）训练过程

本项目基于 Fairseq 架构训练出基于 Transformer 的 ASR 模型。本文设计的模型其编码器为 12 层，解码器为 6 层，各层参数配置参照基础配置。调用 fairseq_cli 中的 train.py 来进行训练。下面将用图 5.1 来展示 train.py 进行模型训练的流程。

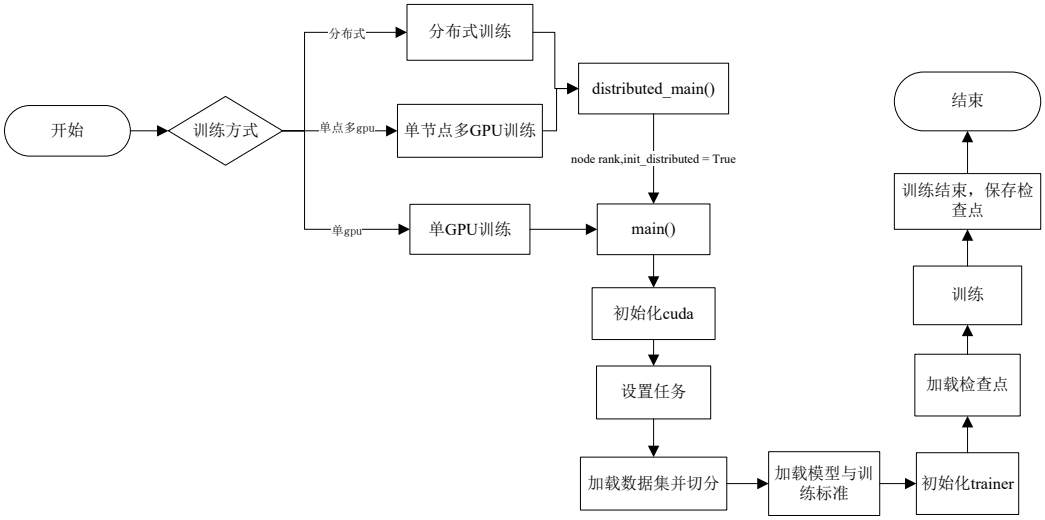

图 5.1 train.py 执行流程

如图 4.7 所示，展示了 train.py 进行训练的总体流程，train.py 会调用其中的 train.cli_main()函数开始训练，有三种可选的训练方式：分布式训练、单点多 gpu 训练以及单 gpu 训练。前两者会调用 distributed_main()，而后者调用 main()。distributed_main()会传递使用的 gpu 的 rank 编号和一个训练参数给 main()来实现前两种训练方式。之后在 main()函数中依次执行初始化 cuda，设置 task 任务、加载数据集并切分、加载模型与训练标准、初始化 trainer、加载检查点、训练模型和训练结束等一系列的操作。其中训练模型过程中一个 epoch 将所有数据送入网络中完成迭代后，会使用校验集计算出损失，进行学习率的调整，这样完成一个

epoch。在损失超过一定步数不再减小或者 epoch 和更新次数大于最大值时停止训练，保存检查点。

3）评价指标

WER： Word error rate，词错率，但一般称为字错率，是语音识别领域的关键性评估指标，WER 越低表示效果越好！计算公式如下：

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + H}$$

其中 S 为替换的字数，常用缩写 WS；D 为删除的字数，常用缩写 WD；I 为插入的字数，常用缩写 WI；H 为正确的字数，维基百科是 H；N 为（S 替换+D 删除+H 正确）的字数。

4）实验结果

Generate dev-clean with beam=5: WER: 6.23

Generate dev-other with beam=5: WER: 13.02

Generate test-clean with beam=5: WER: 6.44

Generate test-other with beam=5: WER: 12.60

训练 25 个 epoch 之后模型可以在 test-clean 测试集上达到 93.54 的识别准确率。

5）部分 ASR 示例展示

参考文本	输出生成文本
this is our last feast with you i said	this is our last feast with you i said
we have a commander who's game for anything	we have a commander whose game for anything
unlucky me and the mother that bore me	and lucky me into the mother that bore me
but now nothing could hold me back	but now nothing could hold me back
he never loses sight of the purpose of his epistle	he never loses sight of the purpose of his epistle
oh but i'm glad to get this place mowed	oh but i'm glad to get this place mode

it's tremendously well put on too	it's tremendously well put on too
we never had so many of them in here before	we never had so many of them in here before
there are few changes in the old quarter	there are few changes in the old quarter
i didn't preach without direction	i didn't preach without direction

总结

通过这次实验，我对 Transformer 的结构和 ASR 模型任务有了更为清晰的认识，熟练了脚本语言的编写和 fairseq 框架的使用，同时体验了如何在巨大的心理波动之下还能按时在 ddl 之前提交项目。

同时，跑语音方向的实验真的需要很大的硬盘空间，数据处理过程中我就因为空间不够程序中断了两次。断舍离删掉很多自己（和别人）的东西后才成功提取训练所需数据。

此外还存在很多不足的地方，比如模型的创新性不够，没有采用速度扰动来增强数据。没有采用尝试更多的方法和更多的数据集。

值得注意的是我跑的数据集是语音识别通用数据集中最小的一个，就已经需要数百 G 的空间和动辄按天计算的训练时间。这让我对语音方向训练模型的成本有了更为深刻的认识，我曾有过一段分布式机器学习相关的项目经历，借此机会我更加深刻地理解了在 ASR、ST、乃至 S2ST 任务中使用训练集群的重要性。

最后十分感谢这门课的开设，肖老师，朱老师和胡驰学长生动幽默、深入浅出的讲解让人茅塞顿开，感谢王成龙学长和严钟响学长的认真指导，同时感谢许晨学长和刘晓雯同学在过程中给予的帮助。