基于 Transformer 的端到端语音翻译模型

一、 项目概述

语音翻译任务的主要目标，是完成从源语言的语音信号到目标语言文本的转换过程。目前广泛使用的方法是级联的语音翻译系统，即把语音识别模型和机器翻译模型进行级联，但是这种方法具有错误传播和翻译效率低的问题。因此，端到端语音翻译成为了新的研究热点。本项目在 Fairseq 框架基础上，利用数据增强、多任务学习等多种技术，设计并实现了基于 Transformer 的端到端语音翻译系统，该系统可以完成由英语语音到法语文本的翻译任务。

二、 模型总体设计

2.1 模型结构

模型总体结构如下图所示：

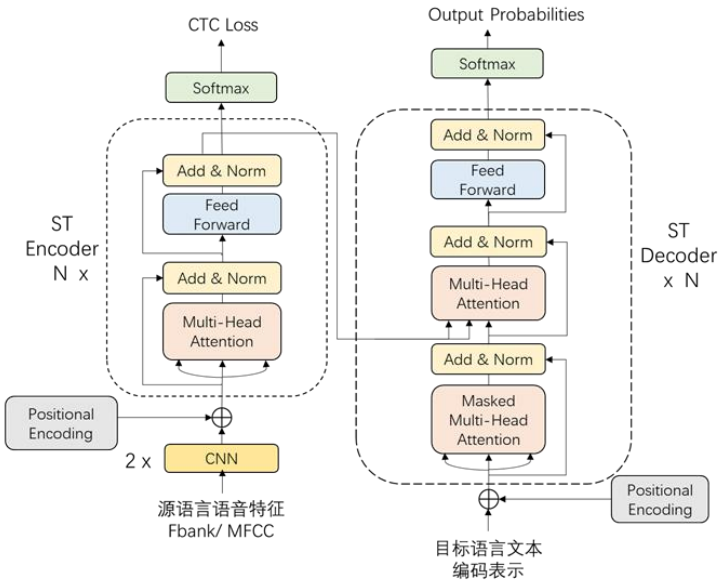


图 2.1 模型总体结构

如图 2.1 所示，本项目中使用的模型是基于 Transformer 结构的，主要分为编码端和解码端两部分。该模型与 Transformer 都使用了多头注意力、前馈神经网络、残差连接以及层标准化技术。输入的序列经过编码器的处理后，最上层的编码结果传入解码器。解码器与编码器相比增加了一个多头注意力子层，其他的结构相似，也有输入的词序列来帮助进行目标语言的生成，最终模型的输出由 Softmax 层完成。

基于 Transformer 的语音翻译模型与基本的 Transformer 相比主要有两点区别。首先就是在编码器的底层使用了一个 CNN 网络。语音翻译的输入与机器翻译的输入不同，机器翻译输入的是文字序列，但是语音翻译输入的是从语音波形中提取出的声学特征序列。声学特征序列相比于文字序列来说更冗长，也更复杂。因此在语音识别和语音翻译任务中通常会使用 CNN 来减小输入的声学特征序列长度。CNN 的加入可以缓解特征序列过长所引起的显存占用和训练时间长等问题。

其次就是在编码器顶层上添加一个额外的输出层，以输出 CTC 损失。本系统中使用了连接时序分类技术（CTC），CTC 技术可以将输入序列每个位置与标注文本相对应，学习语音和文字的软对齐关系。CTC 能够将音频序列上每个位置分别对应到同一个词上，其在预测序列时会在没有声音的位置判断出空，以ϵ来表示。对齐之后，将预测序列中连续相同的词合并，去除ϵ之后得到最终输出结果。CTC 的对齐方式有两个特性：

（1）输入与输出之间对齐单调。对于输入序列$\{x_1,\ldots, x_m\}$和对应的预测序列$\{y_1,\ldots, y_n\}$，若x_i对应预测结果为y_j，则x_{i+1}对应的预测结果必定是y_j、y_{j+1}或ϵ。

（2）输入与输出的对齐是多对一的。对于一个输入序列可以有多种方式与预测序列相对应。多个输入会对应到同一个输出上。

在编码器顶层上添加一个额外的输出层，就可以把 CTC 应用到语音翻译中，实现方法简单。这意味着不需要增加太多的参数就可以为模型引入较强的监督信息。

2.2 模型具体设计

针对语音翻译任务，在搭建模型时的功能需求主要分为三个部分：数据处理、模型训练、译文生成与评价。本项目针对这三方面，对模型进行了具体的设计。

2.2.1 数据处理

语音翻译数据集中包含了：带有编号的 wave 语音音频文件、源语言以及目标语言的参考译文。但是由于一个音频文件包含多句语音，并且时间较长，所以不能直接提取一个 wave 文件的特征作为模型的输入。需要先对音频进行切分处理，让一个切分大致对应源语言中的一句话。通常情况下，数据集中会给出对音频逐句切分的 yaml 文件，里面包含了切分语音片段的开始时间、持续时间、说话者编号和 wave 文件名称。这样在数据处理时直接读取对应的 yaml 文件，对相关数据进行特征提取即可。

数据处理中需要完成数据集初始化和数据处理任务。

数据集初始化需要做到提取语音翻译语料库中的数据集（训练集、校验集和测试集）里基本信息，将 yaml 文件中的语音切分信息与对应的参考英语文本和法语文本做对齐。之后再把 wave 文件路径、语音切分偏移与持续时长、音频采样率、英语文本、法语文本、说话者编号以及索引信息存储到一个 data 列表中，在数据处理时会使用 data 列表中的信息进行后续特征提取、生成 tsv 文件等工作。

数据处理时，会对语料库内部的训练集、校验集、测试集等数据集依次进行数据集初始化，提取各个切分片段的 Fbank 特征，以索引信息命名并存储到设置好的路径下，压缩为 zip 格式。之后依次对数据集进行 tsv 文件的生成，在其中存储之前的各种信息与特征路径。再根据 tsv 文件中的信息生成词表和分词器。最后会生成一个设置的 yaml 文件，记录数据处理中使用的操作和一些文件的路径。

2.2.2 模型训练

在模型训练部分，使用了 Fairseq 框架来搭建语音翻译模型。除了使用基础的语音翻译数据集，本项目中还制作了一些伪数据，用于缓解训练数据稀缺的问题。模型设计为编码器 12 层，解码器 6 层。首先需要检测是否有处理好的数据，根据训练设置文件中的要求，检测 gpu 个数并使用，创建保存模型的文件夹，初始化各个参数，开始迭代训练。每一轮 epoch 训练结束后使用校验集进行校验，并保存检查点文件。

2.2.3 译文生成与评价

语音翻译的评价指标也为 BLEU 值，BLEU 值越大说明模型的翻译质量越好。本部分会加载之前训练好的模型，根据设置文件的需求加载数据集，检索或生成译文并输出，最后根据句子得分算出 BLEU 值。

三、 模型具体实现方法

3.1 数据处理实现

语音翻译数据处理的实现通过 run.sh 中调用 prep_libtrans_data.py 文件来实现。run.sh 需要向该文件传输若干的参数，prep_libtrans_data.py 根据参数来进行数据处理工作。该 py 文件从待处理数据集中提取出各种数据，对语音进行特征提取并保存，制作 tsv 文件保存处理后的数据信息。再生成分词器和词

表，和数据处理的 yaml 设置文件。

在数据处理中使用了速度扰动，该方法可以作用于语音翻译和语音识别的数据处理工作中。在 ST 任务的 run.sh 中加入了 speed_perturb 参数，用来设置是否进行速度扰动的操作。在 perp_libritrans_data.py 中加入了 speed_perturb 列表，其中有 0.9、1.0 和 1.1，这三个系数分别表示对 wave 文件进行处理时的速度系数，这样一条数据就对应生成了三条不一样的数据。速度扰动只对训练集上的数据进行操作，校验集和测试集不变。此方法可以大量增加训练的数据，使数据包含多种不同语速的语音，有效提升模型的性能。

除此之外还使用了 SpecAugmet，该技术在 Fairseq 中已有应用，其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。

除了使用该方法生成对应语音翻译数据集的 tsv 文件用于训练之外，本项目中还使用了类似的方法处理了语音识别数据集生成相应的 tsv 文件，以扩增训练数据，提高模型性能。本项目中使用了 Lbrispeech 数据集，以及其中的语音翻译数据集 Lbri_trans。Lbrispeech 数据集为语音识别数据集，该语料库包括 1000 小时的英文语音 wave 文件以及目标语言文本，项目中使用了一个预先训练好的英法机器翻译（MT）模型将数据集中的目标语言文本翻译成了法语文本，将该法语文本和数据集中的数据组合形成 tsv 文件，得到伪数据并用于后续的模型训练工作，伪数据数量为 281241 条。

3.2 模型训练实现

train.sh 的 shell 脚本，完成相关参数的设置，并通过调用 run.sh 中的 stage 1，来进行网络训练。在 run.sh 中主要通过调用 fairseq_cli 中的 train.py 实现模型训练。本文训练的 ST 模型在编码器的输入前设置了两层一维 CNN 网络，卷积核大小为 5×5，通道数为 1024。模型的编码器为 12 层，解码器为 6 层，使用的是 RELU 激活函数，丢弃率为 0.1。编码器和解码器的嵌入层维度均为 256，前馈神经网络嵌入层维度为 2048，多头注意力个数设定为 4。通过对 trian_ctc.yaml 文件中参数的修改，在 train.py 中读取 yaml 文件的信息并进行模型的训练工作，搭建了若干不同的 ST 模型。模型在训练过程中会保存检查

点 checkpoint.pt 文件，在这里规定了只保存最后 10 个检查点。

在 ST 模型的搭建过程中使用了 CTC 技术。CTC 的实现只需要在模型的编码器端加入一个输出层即可。在 train_ctc.yaml 文件中，加入了 ctc-weight 来规定 CTC 权重，在本系统中统一采用 ctc-weight 值为 0.3。compute_ctc_loss()函数根据模型编码器的输出信息，计算 ctc 损失，并与其他损失一起在 forward()中输出到训练日志中，作为训练时的监督信息。下表展示了训练过程中使用的一些参数信息。

表 4.1 train_ctc.yaml 主要参数

参数名称	参数含义	参数设定值
train-subset	训练集	train_st 等
valid-subset	校验集	dev_st,dev_gtranslate_st
max-epoch	最大迭代次数	50
max-update	最大更新次数	300000
update-freq	参数更新频率	2
max_tokens	每个 batch 包含词数量	2000
num-workers	加载 batch 的线程数目	8
patience	多少轮不变化时停止	10
arch	模型类型	s2t_transformer_s
optimizer	优化器类型	adam
lr	学习率	2e-3
warmup-init-lr	学习率预热初始化大小	1e-7
warmup-updates	学习率预热步数	10000
criterion	评价准则	label_smoothed_cross_entropy_with_ctc
label_smoothing	标签平滑度	0.1

3.3 译文生成与评价实现

本模块的主要任务是译文生成与译文评价。编写了 decode.sh 脚本用于此功能的实现。decode.sh 中规定了模型解码相关的参数，然后调用 run.sh 中的 stage 2 来使用相关模型进行解码，进而得到译文，并利用参考译文对翻译结果进行评价，计算 BLEU 值。在模型解码前，需要先对这十个检查点进行平均计算，得到

一个 avg_10_checkpoint.pt，作为最终的模型。得到平均计算后的模型，就调用 generate.py 完成译文的生成与 BLEU 值的计算。

四、 实验结果与分析

本系统中搭建了多种不同的 ST 模型，并使用 BLEU 值来对 ST 模型性能进行比较，BLEU 值越大说明模型性能越好。测试集选用的是 Libri_trans 中的 test_st。在表 4.1 中介绍了搭建的 3 种 ST 模型和测试结果。下面将进行实验结果的分析和讨论。

表 4.2 ST 模型测试结果

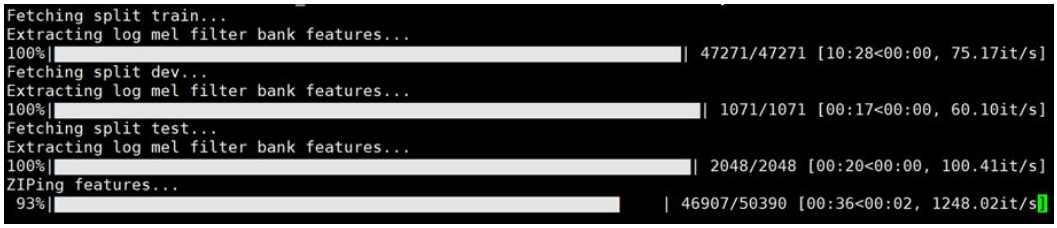
序号	模型名称	训练数据	BLEU
1	Baseline ST	Libri_trans	15.99
2	Speed Perturb ST	Libri_trans	16.63
3	PseudoData ST	Libri_trans+伪数据	18.89

表 4.1 中模型 1 为基线模型，其中只使用了 CTC 和 SpecAugment 技术。模型 2 是在 1 的基础上使用了速度扰动技术。模型 2 取得了比基线提高了 0.64 个 BLEU 值。由此可见，速度扰动是提升 ST 模型性能的有效方法。在其后的模型 3 也应用了该方法。模型 3 展示了使用一定数量伪数据时的模型性能，该模型达到了实验中最高的 BLEU 值 18.89，与基线模型相比提高了 2.9 个 BLEU 值。这也进一步表明了训练数据的规模对模型性能有很大的影响，使用了伪数据的 ST 模型在性能上有明显的提升。

下面展示一些实验过程中的输出结果截图：

```
stage 0: ASR Data Preparation
Run command:
python /home/liuxiaowen/st/Fairseq-S2T/examples/speech_to_text/prep_libritrans_data.py
        --data-root /media/liuxiaowen/st/libri_trans
        --output-root /home/liuxiaowen/st/data/libri_trans/st_sp_lcrm
        --task asr
        --vocab-type unigram
        --vocab-size 5000
        --speed-perturb
stage 0: ST Data Preparation
Run command:
python /home/liuxiaowen/st/Fairseq-S2T/examples/speech_to_text/prep_libritrans_data.py
        --data-root /media/liuxiaowen/st/libri_trans
        --output-root /home/liuxiaowen/st/data/libri_trans/st_sp_lcrm
        --task st
        --add-src
        --cmvn-type utterance
        --vocab-type unigram
        --vocab-size 10000
        --share
        --speed-perturb
        --lowercase-src
        --rm-punc-src
```

(a) 数据处理命令



(b) 特征提取

图 4.1 数据处理输出

```
stage 2: ST Decoding
Run command:
python /home/liuxiaowen/st/Fairseq-S2T/scripts/average_checkpoints.py
        --inputs /home/liuxiaowen/st/Fairseq-S2T/../checkpoints/libri_trans/st/0508_1154_lcrm_train_ctc_base
line
        --num-epoch-checkpoints 10
        --output /home/liuxiaowen/st/Fairseq-S2T/../checkpoints/libri_trans/st/0508_1154_lcrm_train_ctc_base
line/avg_10_checkpoint.pt
```

(a) 解码命令

```
Run command:
python /home/liuxiaowen/st/Fairseq-S2T/fairseq_cli/generate.py
        /home/liuxiaowen/st/data/libri_trans/st_lcrm/en-fr
        --config-yaml config_st_share.yaml
        --gen-subset test_st
        --task speech_to_text
        --path /home/liuxiaowen/st/Fairseq-S2T/../checkpoints/libri_trans/st/0508_1154_lcrm_train_ctc_baseli
ne/avg_10_checkpoint.pt
        --results-path /home/liuxiaowen/st/Fairseq-S2T/../checkpoints/libri_trans/st/0508_1154_lcrm_train_ct
c_baseline
        --max-tokens 10000
        --beam 5
        --lenpen 1.0
        --scoring sacrebleu
Generate test_st with beam=5: BLEU = 15.99 46.9/21.5/11.5/6.3 (BP = 0.973 ratio = 0.973 hyp_len = 42568 ref_
len = 43743)
```

(b) BLEU 值输出

图 4.2 译文生成与评价

```
1319        demanda Château-Renaud. "Dit Chateau-Renaud.    -0.4270661147439704
116         reprenez ça.    Vous pouvez le ramener. -0.5235346662615997
1071        - Comment cela s'est-il fait?   "Comment était-ce?      -0.8392439493767238
1350        --Laquelle?     " "Qui? -1.1236120171075266
1072        demanda le prisonnier.  demanda le prisonnier.  -0.37944218391523643
1687        s'écria le vieux soldat,        S'écria le vieux soldat.        -0.3355034113378654
1397        demanda le président.   Dit le président.       -0.5415489030348992
1387        répéta le magistrat.    répéta le président.    -0.3294191596004604
1570        Que Dieu vous protège!  Dieu vous bénisse!      -0.3738297450826423
1443        --Oui.  " "Oui. -0.7756822573055452
1420        je vais vous en donner. Je leur donnerai.       -0.6037563607175213
1390        continua le président.  continua le président.  -0.46357884843973624
1179        - Eh bien?      "" Eh bien?     -0.7594429674804754
1681        -- Comment?     -- Comment?     -1.0043724896633714
1293        --Moi!  -- Moi? -0.986367625944544
216         Vous m'en donnez votre parole?  Sur votre parole maintenant!    -0.9789038368710744
2010        -- Parfaitement.        Certainement.   -0.4950355501741118
586         « Conseil!      Conseil!        -1.0976111640021344
1102        demanda Cornélius.      demanda Cornelius.      -0.2952691020644165
```

图 4.3 生成译文结果

五、 总结

本项目重点的工作在于端到端语音模型的搭建以及伪数据的生成与使用。本项目使用了 Fairseq 框架，搭建了基于 Transformer 的端到端语音翻译模型基线，并使用了 CTC 技术。对该基线模型的性能进行了测试，实现了基本的英法语音翻译。使用了数据增强方法来提升语音翻译模型的性能，在数据处理时利用了速度扰动技术。使用机器翻译模型和语音识别数据制作了伪数据，用作语音翻译的训练数据。搭建了新的模型并进行测试，对比使用不同的模型优化方法带来的增益效果。