自然语言处理技术报告-机器翻译

袁昭新 2020K8009926029

2023 年 7 月 11 日

目录

1 概述

尽管使用 Pytorch 等开源框架可以方便地实现基于 Transformer 的机器翻译模型，但考虑到实现和超参调整的时间成本较高，在本次作业中我选择了开源的 Fairseq 来实现机器翻译模型。Fairseq 是 Facebook AI Research 开源的序列建模工具包，支持多种序列建模任务，包括机器翻译、语音识别、语言建模等，其内置了多种序列建模模型，包括 Transformer、LSTM、CNN 等，可以方便地进行序列建模任务的实现和调优。

在本次作业中，我使用了提供的中英双语数据 (TED)，使用 Faireseq 实现和对比了基于 Transformer、LSTM 的机器翻译模型，并对翻译结果进行了分析和评估。实验实现主要分为以下几个步骤：

1. 数据预处理：对原始数据进行预处理，包括分词、标点符号标准化、字节对编码以及过滤等，以便进行模型训练。
2. 模型训练：使用预处理后的数据训练模型，包括 Transformer、LSTM 等，调整超参数，优化模型性能。
3. 模型评估：使用 BLEU 等指标对模型进行评估，分析模型的性能。

2 数据预处理

Fairseq 要求将源语言和目标语言的数据分别存储在两个文件中，每个文件中的每一行分别为一条源语言和目标语言的数据，两个文件中的每一行数据一一对应。提供的数据已经满足这一要求，因此我们只需要对数据进行预处理，包括标点符号标准化、分词、token 化、字节对编码以及过滤等，以便进行模型训练。

数据预处理的过程参考了 Fairseq 中提供的英译德的例子，其中提供了详细的数据预处理脚本 (/fairseq/examples/translation/prepare-wmt14en2de.sh)。

2.1 标点符号标准化

首先对标点符号进行标准化，使用 mosesdecoder 中的脚本进行标准化。

```
perl mosesdecoder/scripts/tokenizer/normalize-punctuation.perl \
    -l en < TED2020.en-zh_cn.en > data/norm.en
perl mosesdecoder/scripts/tokenizer/normalize-punctuation.perl \
    -l zh < TED2020.en-zh_cn.zh_cn > data/norm.zh
```

其中例如对中文源文件中第 142 行的数据处理如下所示。

转换前：

这就是我的工作——光学精神控制。

转换后：

这就是我的工作 - - 光学精神控制。

2.2 分词

对于中文而言，需要先进行分词，再进行 token 化。分词使用 jieba 分词工具，对中文源文件中的每一行进行分词，分词后的结果保存在 data/norm_seg.zh 中。

```
python3 -m jieba -d ' ' < data/norm.zh > data/norm_seg.zh
```

分词后的结果如下所示：

这 就是 我 的 工作 - - 光学 精神 控制 。

2.3 token 化

使用 mosesdecoder 中的脚本进行 token 化。这可以将英文单词与标点符号分开，将多个连续的空格合并为一个空格，将某些符号替换为转义字符等。

```
perl mosesdecoder/scripts/tokenizer/tokenizer.perl -l en \
    < data/norm.en > data/norm_tok.en
perl mosesdecoder/scripts/tokenizer/tokenizer.perl -l zh \
    < data/norm_seg.zh > data/norm_tok.zh
```

token 化后的结果如下所示：

这 就是 我 的 工作 - - 光学 精神 控制 。

其中的多个空格被合并为一个空格，句尾的空格被删除。
对于标点符号的处理，例如对英文源文件中第 46 行的数据处理如下所示。
转换前：

It's like beach-combing, you know?

转换后：

It 's like beach-combing , you know ?

2.4 字节对编码

使用 subword-nmt 工具进行字节对编码，将英文和中文的 token 化后的文件分别进行字节对编码，得到英文和中文的 bpe 文件。

正确的处理流程是先划分训练集、验证集和测试集，然后使用训练集学习 bpe 模型，最后使用学习到的 bpe 模型对训练集、验证集和测试集进行编码。但为了方便起见，我们直接使用训练集对训练集、验证集和测试集进行编码。

```
# 学习bpe模型
python3 subword-nmt/subword_nmt/learn_joint_bpe_and_vocab.py --input \
    norm_tok.en -s 32000 -o ../model/bpecode.en --write-vocabulary ../model/
        voc.en
python3 subword-nmt/subword_nmt/learn_joint_bpe_and_vocab.py --input \
    norm_tok.zh -s 32000 -o ../model/bpecode.zh --write-vocabulary ../model/
        voc.zh

# 使用bpe模型对所有数据进行编码
python3 subword-nmt/subword_nmt/apply_bpe.py -c ../model/bpecode.en \
    --vocabulary ../model/voc.en < norm_tok.en > norm_tok_bpe.en
python3 subword-nmt/subword_nmt/apply_bpe.py -c ../model/bpecode.zh \
    --vocabulary ../model/voc.zh < norm_tok.zh > norm_tok_bpe.zh
```

使用 bpe 对数据进行编码，例如对中文源文件中第 41 行的数据处理如下所示。
编码前：

克里斯 安德森 ： 一言为定 ！

编码后：

克里斯 安德森 ： 一 @@ 言 @@ 为 @@ 定 ！

中文 bpecode 文件中的前 10 个 bpe 编码如下所示：

```
我 们</w>
o t
q u
qu ot
quot ;</w>
& quot;</w>
一 个</w>
他 们</w>
这 个</w>
可 以</w>
```

2.5 清洗

对上述处理后的双语文件进行清洗，过滤长度过短和过长的句对，从而有效删除文件中的空白行。使用 mosesdecoder 中的脚本进行清洗。

```
mv norm_tok_bpe.en toclean.en
mv norm_tok_bpe.zh toclean.zh
perl mosesdecoder/scripts/training/clean-corpus-n.perl \
    toclean zh en clean 1 256
```

例如中文源文件中第 124 行为空白行，在清洗后被删除。

2.6 分割

使用 python 脚本将清洗后的数据按照设定的比例随机地分割为训练集、验证集和测试集。将 90% 的数据作为训练集，5% 的数据作为验证集，5% 的数据作为测试集。

至此，数据预处理完毕，得到了最终的训练集、验证集和测试集。

3 模型训练

根据 Fairseq 的文档，首先使用 fairseq-preprocess 对数据进行预处理，将数据转换为二进制格式，以便于后续的训练和测试。相关的命令如下所示。

```
fairseq-preprocess --source-lang en --target-lang zh \
    --trainpref proced/train --validpref proced/valid \
    --testpref proced/test --destdir data-bin/
```

随后，使用 fairseq-train 对数据进行训练。分别使用了 Transformer、LSTM 两种模型进行训练。

3.1 Transformer

使用 Transformer 模型进行训练，相关的命令如下所示。

```
CUDA_VISIBLE_DEVICES=0 fairseq-train data-bin/en-zh \
    --arch transformer --source-lang en --target-lang zh \
    --max-epoch 6 --batch-size 256 \
    --optimizer adam  --lr 0.0007 --adam-betas '(0.9, 0.98)' \
    --lr-scheduler inverse_sqrt --max-tokens 1024  --dropout 0.3 \
    --criterion label_smoothed_cross_entropy  --label-smoothing 0.1 \
    --max-update 200000  --warmup-updates 4000 --warmup-init-lr '1e-07' \
    --num-workers 8 \
    --keep-best-checkpoints 1 \
    --save-dir model/checkpoints2 &
```

使用的损失函数为 label smoothed cross entropy，优化器为 Adam。交叉熵损失函数的计算公式如下所示。

$$H(p,q) = -\sum_{x} p(x) \log q(x) \tag{1}$$

label smoothed cross entropy 可以有效地防止过拟合，提高模型的泛化能力。

在训练过程中，神经网络会促使自身往正确标签和错误标签差值最大的方向学习。这样的学习方式会导致模型过于自信，从而导致过拟合。而 label smoothing 通过抑制正负样本之间的的差距，从而有效地防止过拟合。label smoothing 的计算公式如下所示。

$$p(x) = (1-\epsilon)p(x) + \frac{\epsilon}{|V|} \tag{2}$$

其中 V 为词表的大小，ϵ 为平滑系数。具体的实现方式可能与 fairseq 中的实现有所不同，但其思想是相同的。

训练过程中在训练集和验证集上的 loss 如图1所示。

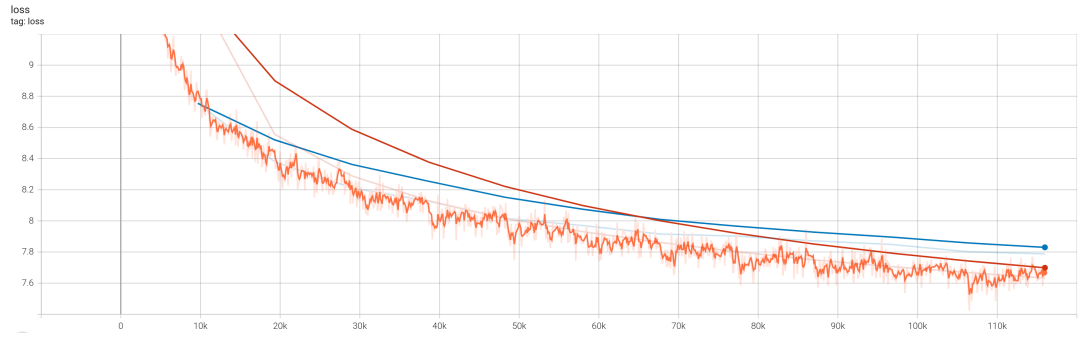

Figure 1: Transformer 模型训练过程中的 loss。其中，橘色的曲线代表训练时上的 loss，红色曲线代表整个训练集的 loss，蓝色曲线代表验证集上的 loss。

ppl 表示模型对于给定的输入序列的预测概率的倒数。ppl 越小，模型的预测越准确。训练过程中模型的困惑度 ppl 如图2所示。

模型在训练到第 60k 个 step 之后，即大约第 6 个 epoch 后，loss 和 ppl 的下降都趋于平缓，同时在验证集上的 loss 和 ppl 也趋于平缓，这表明模型已经收敛。

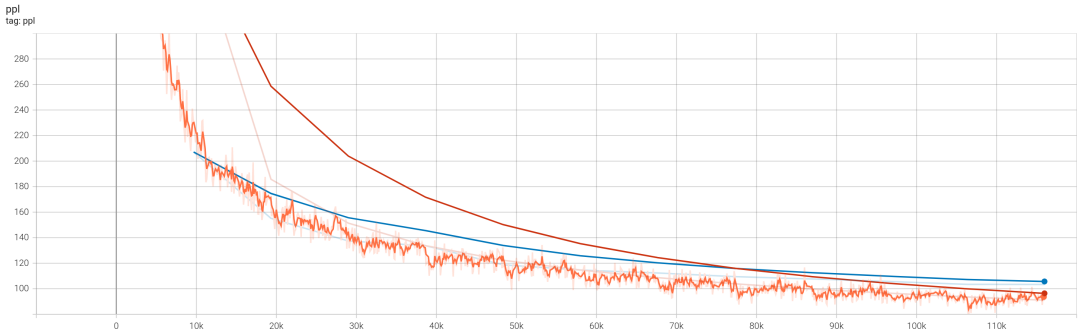

Figure 2: Transformer 模型训练过程中的 ppl。其中，橘色的曲线代表训练时上的 ppl，红色曲线代表整个训练集的 ppl，蓝色曲线代表验证集上的 ppl。

3.2 LSTM

使用 LSTM 模型进行训练，相关的命令如下所示。

```
CUDA_VISIBLE_DEVICES=0 fairseq-train data-bin/en-zh --arch lstm \
        --source-lang en --target-lang zh \
    --max-epoch 10 --batch-size 64 \
    --optimizer adam  --lr 0.0005 --adam-betas '(0.9, 0.98)' \
    --lr-scheduler inverse_sqrt --max-tokens 1024  --dropout 0.3 \
    --criterion cross_entropy \
    --max-update 200000  --warmup-updates 4000 --warmup-init-lr '1e-07' \
    --num-workers 1 --tensorboard-logdir log5/ \
    --keep-best-checkpoints 3 \
        --save-dir model/checkpoints6 &
```

在 LSTM 上，分别使用了 cross entropy 和 label smoothed cross entropy 两种损失函数进行训练。其效果差距不大，这里只展示使用 cross entropy 的结果。

训练过程中在训练集和验证集上的 loss 如图3所示。

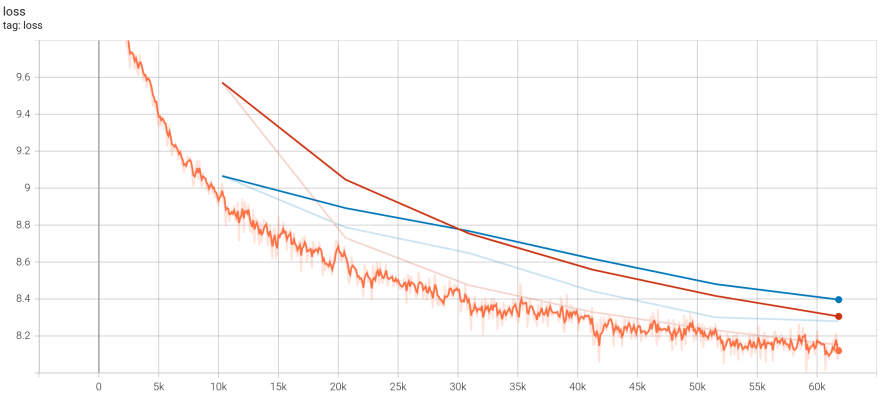

Figure 3: LSTM 模型训练过程中的 loss。其中，橘色的曲线代表训练时上的 loss，红色曲线代表整个训练集的 loss，蓝色曲线代表验证集上的 loss。

训练过程中模型的困惑度 ppl 如图4所示。

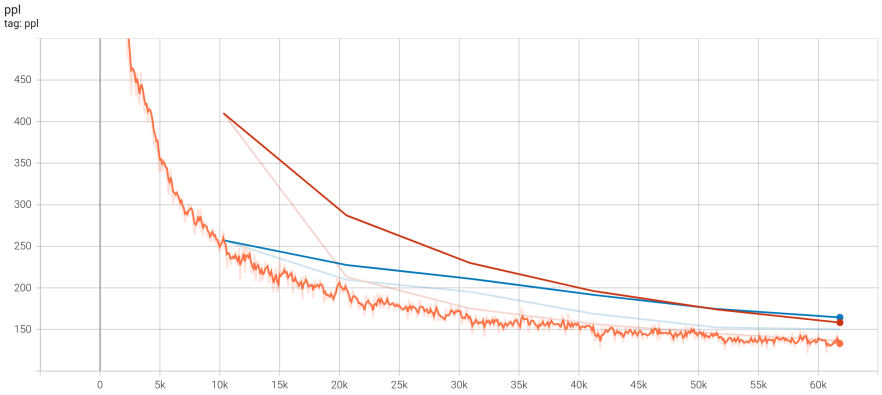

Figure 4: LSTM 模型训练过程中的 ppl。其中，橘色的曲线代表训练时上的 ppl，红色曲线代表整个训练集的 ppl，蓝色曲线代表验证集上的 ppl。

模型在训练到第 10 个 epoch 时，loss 和 ppl 的下降都趋于平缓，同时在验证集上的 loss 和 ppl 也趋于平缓，这表明模型已经收敛，可以停止训练。

4 模型评估

使用训练好的模型对测试集进行翻译，然后使用 BLEU 对翻译结果进行评估。为了达到最好的效果，使用训练过程中在验证集上表现最好的模型进行翻译。

4.1 翻译

翻译过程使用的脚本如下所示。为了减少显存占用，这里使用了 fp16 的精度，否则翻译过程将无法进行。同时，在翻译过程中去除了 BPE 编码。

```
file=original_transformer
fairseq-generate data-bin/en-zh \
    --path model/checkpoints/checkpoint_best.pt \
    --remove-bpe --fp16 --scoring bleu \
    --batch-size 64 --beam 8 > $file.txt
```

对于相同的两个句子，Transformer 和 LSTM 翻译的部分结果如下所示。

```
# Transformer
S-4404  Why would they care ?
T-4404  为什么 它们 会 关心 这个 呢 ?
H-4404  -1.0210976600646973     为什么 会 这样 ?
D-4404  -1.0210976600646973     为什么 会 这样 ?
P-4404  -0.4230 -3.1401 -0.6009 -0.7001 -0.2414
S-9148  Because this is wrong .
T-9148  因为 这样 是 错 的 。
H-9148  -1.0174709558486938     因为 这 是 错 的 。
D-9148  -1.0174709558486938     因为 这 是 错 的 。
P-9148  -0.5324 -3.2863 -1.9479 -0.3507 -0.1783 -0.6778 -0.1489
```

```
# LSTM
S-4404  Why would they care ?
T-4404  为什么 它们 会 关心 这个 呢 ？
H-4404  -1.107420563697815      他们 为什么 要 这样 做 ？
D-4404  -1.107420563697815      他们 为什么 要 这样 做 ？
P-4404  -1.5861 -0.1276 -1.2176 -3.7194 -0.0482 -0.8993 -0.1536
S-9148  Because this is wrong .
T-9148  因为 这样 是 错 的 。
H-9148  -1.1656571626663208     因为 这 是 不 可能 的 。
D-9148  -1.1656571626663208     因为 这 是 不 可能 的 。
P-9148  -0.9448 -2.9336 -2.6911 -0.9434 -0.7263 -0.4065 -0.5443 -0.1352
```

S 表示源语言，T 表示目标语言，H 表示模型翻译的结果，D 表示去除重复的结果，P 表示翻译的概率。

4.2 BLEU

BLEU 是一种常用的机器翻译评估指标，其计算方式如下所示。

$$
\begin{aligned}
\text{BLEU} &= \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \\
\text{BP} &= \begin{cases} 1 & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{if } c \leq r \end{cases} \\
p_n &= \frac{\sum_{\text{clipped } n\text{-grams}} \text{count}_{\text{clipped}}}{\sum_{\text{n-grams}} \text{count}}
\end{aligned}
\tag{3}
$$

使用 BLEU 对翻译结果进行评估，首先将翻译结果和参考结果从翻译后的文件中分别提取到 sys 和 ref 两个文件中，然后使用 fairseq-score 进行评估。相关的命令如下所示。

```
file=original_transformer
grep ^T $file.txt | cut -f2- | perl -ple 's{(\S)-(\S)}{$1 ##AT##-##AT## $2}g'
    > $file.ref
grep ^H $file.txt | cut -f3- | perl -ple 's{(\S)-(\S)}{$1 ##AT##-##AT## $2}g'
    > $file.sys
fairseq-score -s $file.sys -r $file.ref
```

评估结果如下所示。

```
# Transformer
BLEU4 = 2.53, 25.5/4.4/1.1/0.3 \
    (BP=1.000, ratio=1.122, syslen=203880, reflen=181744)
```

```
# LSTM
BLEU4 = 11.77, 44.2/16.9/7.5/3.4 \
    (BP=1.000, ratio=1.057, syslen=192077, reflen=181744)
```

其中，44.2/16.9/7.5/3.4 表示四个不同的 BLEU 值，分别对应 1-gram、2-gram、3-gram 和 4-gram 的 BLEU 值，BP 表示 brevity penalty，ratio 表示系统输出的句子长度和参考句子长度的比值，syslen 表示系统输出的句子长度，reflen 表示参考句子长度。

一般而言，BLEU 的值越高，表示翻译的效果越好。主流的翻译软件的 BLEU 值在 20 到 40 之间，而本次实验中基于 Transformer 的翻译结果的 BLEU 值为 2.53，基于 LSTM 的翻译结果

的 BLEU 值为 11.77，均低于主流翻译软件的 BLEU 值，说明本次实验中的翻译结果的质量较差。

Transformer 的 BLEU 值低于 LSTM 的 BLEU 值，可能是因为 Transformer 在训练时需要的资源较多，为了在有限的资源下进行训练，其中的一些参数可能遭到限制，例如 batch size 等，导致 Transformer 的 BLEU 值较低。

5 总结

本次实验通过 Fairseq 这一开源工具实现了 LSTM 和 Transformer 两种模型对英语翻译为中文的任务，通过 BLEU 对翻译结果进行了评估。通过本次实验，我对 Transformer 模型有了初步的认识。并且通过具体实践，加深了我对机器翻译相关内容的理解。