北京大学

Python 程序设计与数据科学导论（期末大作业 Part 2）

机器翻译

目标

1. 利用 PyTorch 复现 Seq2Seq with Attention，以实现简单的机器翻译任务。

主要任务

1. 数据预处理 （已由助教在 Dataloader 类中实现）
2. 模型：Encoder、Decoder（+ Attention）、Seq2Seq
3. 测试效率（生成文本等） + 调参

目录

姓　　名：	尤俊浩
学　　号：	1900094810
院　　系：	信息科学技术学院
联系方式：	1900094810@pku.edu.cn

二〇二一年 六月

模型

模型主要有 3 个部分

1. Encoder

采用单层双向 GRU，需要注意的是要在最后通过一个全连接层将输出的维度转换成 [batch_size, dec_hid_dim]

```
class Encoder(nn.Module):
    def __init__(self, input_dim, emb_dim, enc_hid_dim, dec_hid_dim, dropout):
        super().__init__()

        ##构建embedding
        self.embedding = nn.Embedding(input_dim, emb_dim)

        ##构建rnn模块
        self.rnn = nn.GRU(emb_dim, enc_hid_dim, bidirectional = True)

        self.fc = nn.Linear(enc_hid_dim * 2, dec_hid_dim)

        ##构建dropout层
        self.dropout = nn.Dropout(dropout)


    def forward(self, src_info):
        src, src_len = src_info

        ## 补充embedding层（+dropout）代码
        embedded = self.dropout(self.embedding(src))
        packed_embedded = nn.utils.rnn.pack_padded_sequence(embedded, src_len.to('cpu'))

        ## 补充rnn层代码
        packed_outputs, hidden = self.rnn(packed_embedded)
        outputs, _ = nn.utils.rnn.pad_packed_sequence(packed_outputs)
        hidden = torch.tanh(self.fc(torch.cat((hidden[-2,:,:], hidden[-1,:,:]), dim = 1)))

        return outputs, hidden
```

2. Decoder（带 Attention 机制）

Decoder 与 Encoder 类似，采用的是单层单向的 GRU，但增加了 Attention 机制，以实现对序列上下文的重点进行关注。Attention 机制的实现如下图，主要思想是按照元素重要程度加权平均。

```
class Attention(nn.Module):
    def __init__(self, enc_hid_dim, dec_hid_dim):
        super().__init__()
        self.attn = nn.Linear((enc_hid_dim*2) + dec_hid_dim, dec_hid_dim)
        self.v = nn.Linear(dec_hid_dim, 1, bias=False)


    def forward(self, hidden, encoder_outputs, mask):
        #hidden = [batch size, dec hid dim]
        #encoder_outputs = [src len, batch size, enc hid dim * 2]
        batch_size = encoder_outputs.shape[1]
        src_len = encoder_outputs.shape[0]

        #将decoder的隐状态重复src_len次
#         print("in attention, hidden shape:", hidden.shape)
        hidden = hidden.unsqueeze(1).repeat(1, src_len, 1)
        encoder_outputs = encoder_outputs.permute(1, 0, 2)
        #hidden = [batch size, src len, dec hid dim]
        #encoder_outputs = [batch size, src len, enc hid dim * 2]

        ##请补充计算attention score的代码
        energy = torch.tanh(self.attn(torch.cat((hidden, encoder_outputs), dim = 2)))
        attention = self.v(energy).squeeze(2)
        attention.masked_fill(mask == 0, -1e10)
        ##mask掉padding部分，计算softmax
        return F.softmax(attention, dim = 1)
```

3. Seq2Seq 模型

最后是整合了 Encoder 和 Decoder 的 Seq2Seq 模型。有别于传统的 Seq2Seq 模型中直接将句子中的每个词都不断传入 Decoder 中训练，引入了 Attention 机制可以自己控制将词一个一个的输入。具体实现如下：

```
class Seq2Seq(nn.Module):
    def __init__(self, encoder, decoder, device, pad_idx):
        super().__init__()
        self.encoder = encoder
        self.decoder = decoder
        self.pad_idx = pad_idx
        self.device = device

    def forward(self, src_info, trg = None, teacher_forcing_ratio = 0.5):
        src, src_len = src_info
        batch_size = src.shape[1]
        max_len = trg.shape[0] if trg is not None else MAX_LEN
        trg_vocab_size = self.decoder.output_dim

        ##存储所有decoder输出的结果
        outputs = torch.zeros(max_len, batch_size, trg_vocab_size).to(self.device)
        attn_scores = []

        ## encoder
        encoder_outputs, hidden = self.encoder(src_info)

        ##初始化decoder的输入是<sos>token
        input = trg[0, :] if trg is not None else src[0, :]

        #mask = [batch size, src len]
        mask = self.create_mask(src)


        ## decode过程，每个step decode出一个token
        for t in range(1, max_len):
            ## 请补全decoder的代码，得到output, hidden, atten_score
            output, hidden, a = self.decoder(input, hidden, encoder_outputs,mask)
            attn_scores.append(a)
            outputs[t] = output
            teacher_force = random.random() < teacher_forcing_ratio
            top1 = output.argmax(1)
            input = trg[t] if teacher_force else top1


        return outputs, torch.cat(attn_scores, dim = 1).to(self.device)

    def create_mask(self, src):
        mask = (src != self.pad_idx).permute(1, 0)
        return mask
```

模型效果：

对训练集训练 100 个 epoch 后，表现如下：

```
100%|██████████| 176/176 [00:17<00:00, 10.01it/s]
Best BLEU: 0.081 | Best Loss:3.899 |  Epoch: 99 |  BLeu :  0.085 | Loss:6.032081181352789
```

利用测试集对最佳模型进行评估，平均 BLEU score 为 0.076，其中翻译效果仍然差强人意。

```
src:return a dictionary of the builtins in the current execution frame, or the interpreter of the thread state if no frame is currently executing.
trg:<unk> 返回 当前 帧 中 的 当前 帧 中 ， 如果 当前 帧 的 " ， " " " " "
golden:返回 当前 执行 帧 中 内置 函数 的 字典 ， 如果 当前 没有 帧 正在 执行 ， 则 返回 线程 状态 的 解释器

src:many editors and ides provide syntax <unk> debugging tools, and pep 8 <unk>
trg:<unk> 许多 、 和 分发 和 语法 和 语法 和 语法 和 和 pep <unk> 和 pep <unk> 和 pep <unk>
golden:大多数 编辑器 和 集成 开发 环境 支持 语法 <unk> ， 调试 工具 和 pep 8 检查

src:the cython and <unk> libraries offer <unk> over python's c api
trg:<unk> " 和 " <unk> " <unk> " <unk> " python 的 c api
golden:cython 和 <unk> 库 提供 了 对于 python 的 c api 的 抽象

src:for integer and pointer types, it is an integer, for character types, it is a single character bytes object or string, for character pointer types it is a python bytes object or string.
trg:<unk> 对于 整数 和 指针 ， 对于 整数 ， 对于 整数 代表 一个 字符 ， bytes like object 或 bytes like object 或 字节 串 对象
golden:对于 整数 和 指针 类型 ， 它 是 一个 整数 ， 对于 字符 类型 ， 它 是 一个 单 字符 字符串 对象 或 字符串 ， 对于 字符 指针 类型 ， 它 是 一个 python 字节 串 对象 或 字符串

0.0760201715402007
```

参考文献:

[1] Seq2Seq (Attention) 的 PyTorch 实现，https://wmathor.com/index.php/archives/1451/，2020-07-02

[2] Pytorch-seq2seq，https://github.com/bentrevett/pytorch-seq2seq，2020-05-12

[3] Seq2Seq 实现神经机器翻译，https://zhuanlan.zhihu.com/p/73141235，2020-11-07