中山大學
SUN YAT-SEN UNIVERSITY

《自然语言处理实验》

实验报告

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	17 计科大数据方向
学 生 姓 名 ：	张锦洋
学　　号 ：	17308213
时　　间 ：	2019　年　12　月　23　日

模　型：神经翻译模型

一. 实验准备

1、实验要求：

i. 两个LSTM分别作为Encoder和Decoder

ii. 实现基于注意力机制的机器翻译

iii. 改变teacher forcing ratio，观察效果

iv. 采用Beam Search策略对测试集进行解码，并用BLEU值（BLEU-4）进行评估

2、实验运行环境：Python3.6, cuda9, cudnn7,pytorch和nltk库

3、整个实验过程可以分为建立模型和训练模型两个阶段。

二. 建立模型

1) 数据预处理：

a) 对中文进行分词，并在每个词之间加上空格，对英文也要进行简单的预处理，在标点符号前加上空格。这样读入数据时，只需简单将空格作为分隔符，就可以得到独立的词单元。

b) 分别建立中文和英文的序号跟单词的映射词典，在每条句子开头和结尾分别添加“bos”和“eos”标记，用“padding”补齐较短序列并将每条句子转换为固定长度的数字序列，作为模型的输入。

2) 定义模型：

a) 我将attention机制单独封装成类，在decoder模型中直接引用:

attention机制中要训练的总共有两个全连接层

分别用于计算权重的score

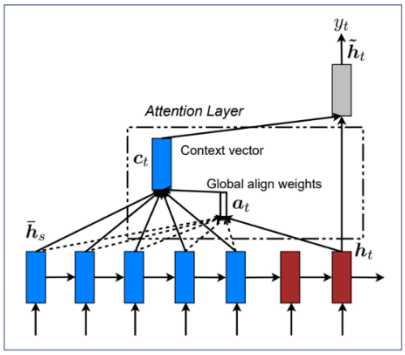
$$score(h_t, \bar{h}_s) = \begin{cases} h_t^T W \bar{h}_s & ^1 \\ V_a^T tanh(W_1 h_t + W_2 \bar{h}_s) \end{cases}$$

和attention vector输出

$$\tilde{h}_t = f(c_t, h_t) = tanh(W_c[c_t; h_t])$$

因此，初始化网络时，定义两个全连接层

```
    def __init__(self,hidden_size):
        super(attention, self).__init__()
        self.hidden_size = hidden_size
        # 要训练的网络总共有两个全连接层
        # 获取 score(ht,hs)，训练参数为 W
        self.linear = nn.Linear(hidden_size,hidden_size)
        # 获取 attention vector， 训练参数为 Wc
        self.attention_vetcor = nn.Linear(2*hidden_size,hidden_size)
```



1. Attention weight α_{ts} :

$$\alpha_{ts}=\frac{\exp(score(h_t,\bar{h}_s))}{\sum_{s'=1}^{S}\exp(score(h_t,\bar{h}_{s'}))}$$

$$score(h_t,\bar{h}_s)=\begin{cases} h_t^T W\bar{h}_s & ^1 \\ V_a^T tanh(W_1 h_t+W_2\bar{h}_s) \end{cases}$$

2.Context vector c_t :

$$c_t=\sum_s \alpha_{ts} * \bar{h}_s$$

3.Attention vector $\tilde{h}_t$:

$$\tilde{h}_t=f(c_t,h_t)=tanh(W_c[c_t;h_t])$$

[;]表示拼接操作

根据attention机制的流程，forward的过程可以表示如下

i. 首先计算score，我选择的score形式如下

$$\boldsymbol{score(h_t,\bar{h}_s)=h_t^T W\bar{h}_s}$$

$\bar{h}_s$为encoder所有单元隐藏层的输出结果

h_t为前一单元隐藏层的传递状态

变量	Tensor维度
$\bar{h}_s$	(batch_size, seq_len, hidden_size)
h_t	(batch_size, 1, hidden_size)

$\bar{h}_s$经过一层全连接层得到$W\bar{h}_s$

(batch_size, seq_len, hidden_size)->(batch_size, seq_len, hidden_size)

参数W维度为 (hidden_size, hidden_size)

然后与h_t在每个batch上进行矩阵乘法运算得到score

将$W\bar{h}_s$进行转置后在每个batch上与h_t进行矩阵相乘

(batch_size, 1, hidden_size) *(batch_size, hidden_size, seq_len)

得到结果score维度为(batch_size, 1, seq_len)

ii. 然后对score矩阵进行softmax操作

$$\alpha_{ts}=\frac{exp\ (score(h_t,\bar{h}_s))}{\sum_{s'=1}^{S} exp\ (score(h_t,\bar{h}_{s'}))}$$

注意操作要在第2维度上，对应每个batch中所有time_step的结果权重加和为1

得到α_{ts}的维度依然为(batch_size, 1, seq_len)

iii. 计算

Context vector c_t :

$$c_t=\sum_s \alpha_{ts} * \bar{h}_s$$

变量	Tensor维度
$\bar{h}_s$	(batch_size, seq_len, hidden_size)
α_{ts}	(batch_size, 1, seq_len)

等价于$\bar{h}_s$和α_{ts}也在每个batch上进行矩阵乘法

(batch_size, 1, seq_len)* (batch_size, seq_len, hidden_size)

=(batch_size, 1, hidden_size)

iv. 计算

Attention vector $\tilde{h}_t$:

$$\tilde{h}_t=f(c_t,h_t)=tanh\,(W_c[c_t;h_t])$$

首先拼接c_t和h_t得到$[c_t;h_t]$维度为(batch_size, 1, 2*hidden_size)

经过(2*hidden_size, hidden_size)的全连接层后得到结果维度为

(batch_size, 1, hidden_size)

再对每个元素进行tanh函数激活即得到注意力机制的输出结果

```
def forward(self,hs,ht):
    # hs 为 encoder 隐藏层所有 time_step 输出的结果，
    # ht 为 decoder 中一个 time_step 隐藏层输出结果
    # hs 的维度大小:(bs,seq_len,hidden_size) ht 的维度大小(batch_size,1,hidden_size)
    score = self.linear(hs)
    # (batch_size,seq_len,hidden_size)->(batch_size,seq_len,hidden_size)
    score = score.transpose(1,2)
    # 进行转置得到(batch_size,hidden_size,seq_len)
    score = torch.bmm(ht,score)
```

```
    #(batch_size,1,hidden_size)*(batch_size,hidden_size,seq_len)=(batch_size,1,seq_len)
     self.weight = F.softmax(score,dim=2)
     # 对第二个维度进行 softmax，使每个 batch 的每个 time_step 权重之和为 1
     ct = torch.bmm(self.weight,hs)
     # (batch_size,1,seq_len)*(bs,seq_len,hidden_size) = (batch_size,1,hidden_size)
     h = torch.cat((ct,ht),2)
     # 拼接操作后得到 (batch_size,1,2*hidden_size)
     output = self.attention_vetcor(h)
     # (batch_size,1,2*hidden_size)->(batch_size,1,hidden_size)
     output = F.torch.tanh(output)
     return output
```

b) encoder模型由embedding层后加上双向lstm组成。

其中embedding层将单词序号映射到embedding_size维度的向量空间，然后再经过双向lstm得到隐藏层输出和传递的状态。由于双向lstm隐藏层的输出维度为

(batch_size,seq_len, hidden_size*2)

在输出隐藏层状态时，我采用了简单的将两个方向的隐藏层状态进行相加，将维度压缩到了

(batch_size,seq_len, hidden_size)

c) decoder模型由embedding层加上单向lstm，然后再加上以及定义好的attention层，并使用常规方法在forward时对其进行正向传播，最后再将结果输入到全连接层，即可得到预测结果。

```
class decoder(nn.Module):
    def __init__(self,input_size,embed_size, hidden_size):
        super(decoder,self).__init__()
        self.embed = nn.Embedding(input_size, embed_size)
        self.lstm = nn.LSTM(embed_size, hidden_size,batch_first=True)
        # 引入 attention 层
        self.attention_layer = attention(hidden_size)
        self.linear = nn.Linear(hidden_size, input_size)

    def forward(self, x, hs, hm):
        # hs 为 encoder 隐藏层输出，hm 为前一单元的隐藏层状态传递
        x = self.embed(x)
        # (batch_size,1,voc_size)->(batch_size,1,embed_size)
        hidden_out,h = self.lstm(x,hm)
        # hidden_out:(batch_size,1,hidden_size)
```

```
        # h 为一个包含两个(1,batch_size,hidden_size)的元组
        hidden_out = self.attention_layer(hs,hidden_out)
        # hidden_out: (batch_size,1,hidden_size)
        out = self.linear(hidden_out)
        # (batch_size,1,hidden_size)->(batch_size,1,voc_size)
        return out,h
```

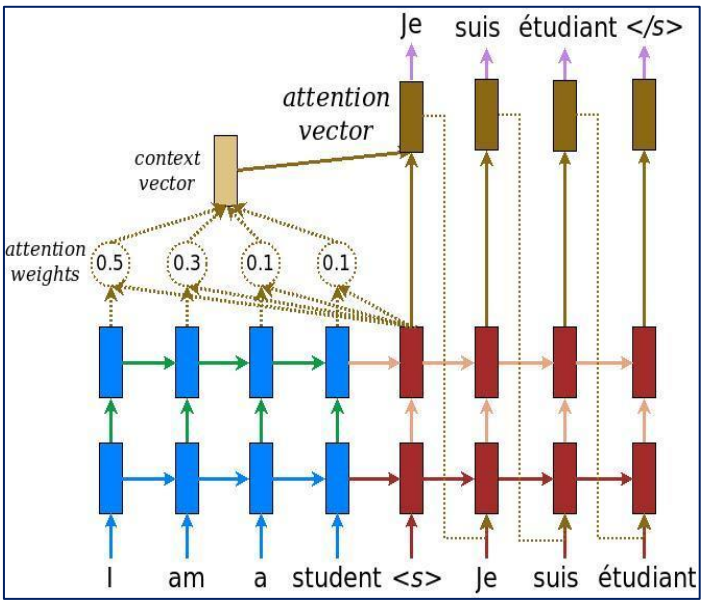
3) 训练模型

a) 训练过程

对encoder网络，可以直接输入(batch_size, seq_len)的训练数据，并返回每个单元隐藏层的输出$\bar{h}_s$和最后一个单元隐藏层的传递状态h_t。

而对于decoder网络，对每个time_step单元要进行迭代训练，第一个单元以起始标记”bos”作为输入，接下来每个单元都以前一单元的预测结果为输入，进行训练

由于我将attention机制包含在decoder网络中，因此decoder训练时一共接收三个参数，输入数据x，前一单元传递的隐藏层状态h_t，以及encoder网络每个单元隐藏层的输出$\bar{h}_s$。



由于padding是人为填充的，在计算损失函数时，要用ignore_index关键词，去除填充项对总体损失的影响

b) 在训练初期，由于预测结果准确率较低，错误的预测会影响后面的单元的训练，因此使用teacher forcing的方法，以一定概率随机决定使用ground

truth还是前一单元生成的预测，可以提高网络的训练的效率

```
for j in range(1,target.size()[1]-1):
    use_teacher_forcing = True if random.random() < teacher_force_rate else False
    # 通过设置 teacher_force_rate 决定随机输入正确标签的概率
    if use_teacher_forcing:
        x = target[:,[j]].to(device)
    output,h_c = decoding(x,hs,h_c)
    x = torch.max(output, 2)[1]
    # 得到预测结果
    result=torch.cat((result,output),1)
    # 将每一步结果拼接得到预测序列
```

4) 测试与评估

a) 集束搜索(beam search)

集束搜索在起始时间步的预测中选择前k个概率最大的预测，作为下一个单元的输入，然后在后续预测中，总是只保留当前概率前k大的序列作为候选序列，并在最后将最大概率的预测序列作为结果输出。

集束搜索是本次实验难点之一，主要突破口在于用简洁，不易出错且便于进行切片操作的数据结构来存储候选序列的所有中间信息。

我将在数组索引上比较容易出错的操作封装成函数

其中candidate是前一时间步产生的候选序列，prob为当前时间步预测的序列的概率集合，对prob进行排序后，选出前beam_width大的序列作为新的candidate，并为每个序列包括概率score和隐藏层状态。

```
# beam_search 搜索
def beam_search(beam_width,candidate,score,prob,index,h):
    prob = prob+score
    prob = torch.sort(prob.reshape(-1),0,True)
    new_candidate = []
    for i in range(beam_width):
        score[i]=d[0][i]
        r = (d[1][i])//beam_width
        l = (d[1][i])%beam_width
        h[i] = h[r]
        new_candidate.append(candidate[i]+[index[r][l]])
    return (new_candidate,score,h)
```

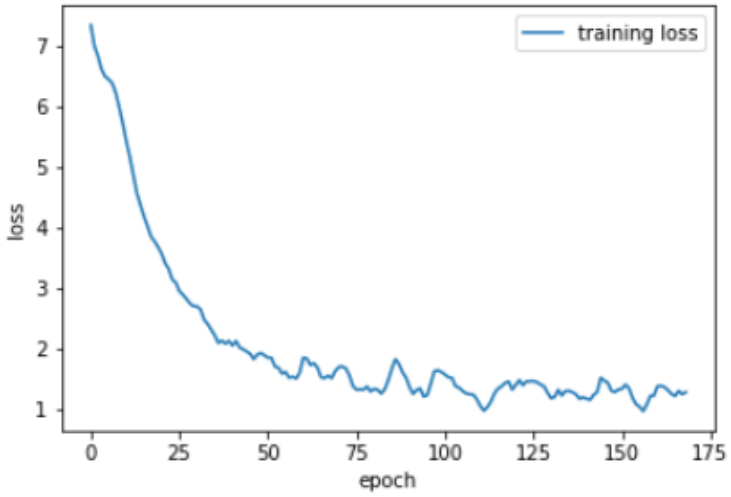
b) 计算预测结果与ground true之间的BLEU值；使用NLTK包可以计算BLEU值，我对每个样例都计算了BLEU值，并将平均值作为最终预测评估结果

四. 训练结果

1) 我选择是10k规模大小的数据集，对模型进行训练，在训练集上的损失显著有减小趋势，最开始的teacher_force_ratio设定为0.8

```
Epoch:  0 | train loss: 7.3382
Epoch:  5 | train loss: 6.4439
Epoch:  10 | train loss: 5.4261
Epoch:  15 | train loss: 4.1819
Epoch:  20 | train loss: 3.5666
Epoch:  25 | train loss: 2.9492
Epoch:  30 | train loss: 2.6943
Epoch:  35 | train loss: 2.2172
Epoch:  40 | train loss: 2.0498
```

训练到100轮后，发现损失已经接近收敛了



总共对训练集和测试集都计算了BLEU值，并观察了他们的输出结果

在测试集上的BLEU值可到达0.944

截取其中部分预测结果：

发现翻译效果很好，几乎翻译出了原句，并且原句里有些省略的成分，比如there is等训练后的模型会补充翻译。

训练集翻译结果

1中文: bos 1929 年 还是 1989 年 ? eos

预测翻译: 1929 or 1989 ? eos
正确翻译: 1929 or 1989 ? eos

2中文: bos 巴黎 - 随着 经济危机 不断 加深 和 蔓延 , 整个 世界 一直 在 寻找 历史 上 的 类似 eos

预测翻译: PARIS - As the economic crisis deepens and widens , the world has been searching for historical analogies eos
正确翻译: PARIS - As the economic crisis deepens and widens , the world has been searching for historical analogies eos

3中文: bos 一 开始 , 很多 人 把 这次 危机 比作 1982 年 或 1973 年 所 发生 的 情况 eos

预测翻译: At the start of the crisis , many people likened it to 1982 or 1973 , which was eos
正确翻译: At the start of the crisis , many people likened it to 1982 or 1973 , which was eos

4中文: bos 如今 人们 的 心情 却是 沉重 多 了 , 许多 人 开始 把 这次 危机 与 1929 年 eos

预测翻译: Today , the mood is much grimmer , with references to 1929 and 1931 beginning to abound , eos
正确翻译: Today , the mood is much grimmer , with references to 1929 and 1931 beginning to abound , eos

5中文: bos 目前 的 趋势 是 , 要么 是 过度 的 克制 （ 欧洲 ） , 要么 是 努力 的 eos

预测翻译: The tendency is either excessive restraint (Europe) or a diffusion of the treatment of law eos
正确翻译: The tendency is either excessive restraint (Europe) or a diffusion of the effort (the United eos

6中文: bos 欧洲 在 避免 债务 和 捍卫 欧元 的 名义 下正 变得 谨慎 , 而 美国 已经 在 许多 eos

预测翻译: Europe is being cautious in the name of avoiding debt and defending the euro , whereas the US eos
正确翻译: Europe is being cautious in the name of avoiding debt and defending the euro , whereas the US eos

1225中文: bos 通过 有 约束力 的 联合国安理会 决议 , 科索沃 可以 被 授予 对 其 公民 和 领土 全面 和 eos

预测翻译: By means of a binding UN Security Council resolution , Kosovo could be granted full and exclusive authority eos
正确翻译: By means of a binding UN Security Council resolution , Kosovo could be granted full and exclusive authority eos

1226中文: bos 可以 允许 其 签署 贸易 协议 以及 与 个人 相关 的 协议 （ 比如 , 允许 外国人 入境 eos

预测翻译: It could be authorized to enter into trade agreements as agreements concerning individuals (for example , eos
正确翻译: It could be authorized to enter into trade agreements as well as agreements concerning individuals (for example eos

1227中文: bos 科索沃 将 因此 获得 某些 表明 国家 地位 的 重要 外部 标志 。 eos

预测翻译: Kosovo would thus gain some essential trappings of statehood . eos
正确翻译: Kosovo would thus gain some essential trappings of statehood . eos

1228中文: bos 然而 , 一个 由 科索沃 、 塞尔维亚 和 欧盟 代表 组成 的 决策机构 将 全权处理 重大 的 外交政策 eos

预测翻译: However , a decision - making body consisting of delegates from Kosovo , Serbia , and the European eos
正确翻译: However , a decision - making body consisting of delegates from Kosovo , Serbia , and the European eos

但在测试集上效果就比较糟糕， BLEU值也只能到达0.139

但当我将翻译结果打印并进行对照后，发现大部分句子翻译效果都很糟糕，很多语句只能翻译出开头短语。像more importantly, meanwhile, so等常放在句子开头或连接句子的短语，容易被正确预测出。但带有实际含义的许多名词动词等，翻译难度较大。

测试集翻译结果:

22中文: bos 与此同时 , 美国 从 中产阶级 社会 变得 日益 贫富 分化 。 eos

预测翻译: Meanwhile , America ' s strategic partnership , the US partnership eos
正确翻译: In the meantime , America went from being a middle - class society to one increasingly divided between eos

```
43中文: bos 因此 下 一个 任务 是 设计 明智 、 新颖 、 低成本 的 方案 解决 这些 挑战 。 eos

预测翻译: So he is the to reflects the to win the next of neither next next month , eos
正确翻译: So the next task is to design wise , innovative , and cost - effective programs to address eos

44中文: bos 不幸 的 是 ， 当 需要 采取 大胆 的 创新 方案 满足 关键 人类 需求 时 ， 美国 eos

预测翻译: Unfortunately , though most less a full cuts in the US and that fear that the likely to eos
正确翻译: Unfortunately , when it comes to bold and innovative programs to meet critical human needs , America is eos

45中文: bos 美国 是 时候 脱胎换骨 了 ， 而 奥巴马 全力 为 进步 主义 愿景 辩护 的 举动 为 美国 eos

预测翻译: America is that that the growth in America , this is so more that a view that the eos
正确翻译: It is time to begin anew , and Obama ' s full - throated defense of a progressive eos

46中文: bos 美国 新 的 贸易 伪善 eos

预测翻译: America ' s United hope . eos
正确翻译: America ' s New Trade Hypocrisy eos
```

分析翻译结果，我认为原因还是由于训练集太小导致的过拟合，训练集包含的句子类型和内容不够多，导致模型学习到的是训练集上的特征，泛化性不强。

对不同的 teacher_force_ratio 进行测试

teacher_force_ratio	Bleu-4 on train_data after 120 epoch	Bleu-4 on test_data after 120 epoch
0.8	0.944	0. 139
0.5	0.771	0. 126
0.3	0.632	0.0838

在训练时，我发现当 teacher_force_ratio 太小，即给予的正确答案提示很少时，网络收敛得极慢，对训练数据的翻译结果也较差。

但在测试集上，在这三种参数设置的模型的翻译能力都比较弱。

五. 总结与感想:

(1) 这学期搭建的两个模型，可以说让我终于对神经网络有了具体的了解，直观体验了它的强大功能，同时也掌握了在构建神经网络过程中的许多方法要点。搭建网络、训练网络和调试网络的过程都充满了趣味性。同时经过第一次实验，我也明显感觉到了我构建神经网络的能力，有了显著的提升，对 pytorch、tensorflow 等库的使用也更加熟练。这是宝贵的经验。

(2) 在实验中感悟最深的就是过拟合的问题，尽管在训练集上有些极高的准确率，在测试集上还是出现了糟糕的预测结果。当数据集较小时，往往会学到训练集上的部分特征，而泛化性不强。数据和算力，可以说是限制我的神经网络的能力的最主要的原因了。

(3) 经过这两次实验，我现在看神经网络相关的资料文献，也更容易理解了。对我的能力有很大的提高，也激发了我对 nlp 和其它人工智能领域的浓厚兴趣。希望在未来，我能够对这些技术有更深入的研究。