

---

# 东北大学《自然语言课程》大作业

## 基于 Teacher Forcing 和 Beam Search 方法的机器翻译任务



姓 名 : 李 中  
学 号 : 2272001  
专 业 名 称 : 计算机技术  
学 院 : 计算机科学与工程  
任 课 老 师 : 肖 桐

二〇二二年秋季 12 月 21 日

---

## 实践报告

### 1、研究问题

近年来，自然语言处理的研究已经成为热点，而机器翻译作为自然语言研究领域的一个重要分支，同时也是人工智能领域的一个课题，同样备受关注。随着神经网络的发展和 Transformer 的出现，基于此类技术的翻译系统，展现出很好的性能。

### 2、研究方法

RNN 方法很难处理长距离依赖的问题，而 LSTM 可以捕捉长距离依赖，但也存在逐字翻译的问题，下一个词的翻译必须依赖于前一个的隐藏状态。CNN 可以对同一句子应用不同的内核来解决依赖关系，但是句子中单词所有组合之间的依存关系需要数量巨大的内核，因此也不是很好的选择。而 Transformer 的注意力机制，可以很好的解决长距离依赖问题，且能够实现高度的并行化处理，是机器翻译任务的非常好的模型选择。

### 3、Transformer

实践中使用的 Transformer 是基于 Harvard nlp 的源码，该源码逻辑非常的清晰，可读性强，适合初学者快速上手。

#### (1) Embedding 层

Encoder 和 Decoder 层共享相同的参数的 Embeddings 层。这里的词嵌入使用 `nn.Embedding` 随机初始化，并随着模型的训练获得合适的词向量。经 `nn.Embedding` 后需要乘以 `sqrt(d_model)` 扩大数值，目的减少位置编码对词向量过大的影响。

---

单词在句子中顺序和位置会影响句子的意思，所以需要添加上位置信息。位置信息 **encoding** 有两种方式，一种是通过训练学习得到，而这里采用公式计算得到。优点是不需要训练参数，在训练集中未出现过的句子长度上也能用。计算公式如下：

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

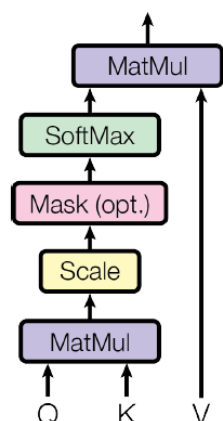
其中，**pos** 代表词在句子中的位置，**2i** 表示 **embedding** 词向量的偶数维度，**2i+1** 表示 **embedding** 词向量的奇数维度。

## (2) MultiHeadedAttention 层

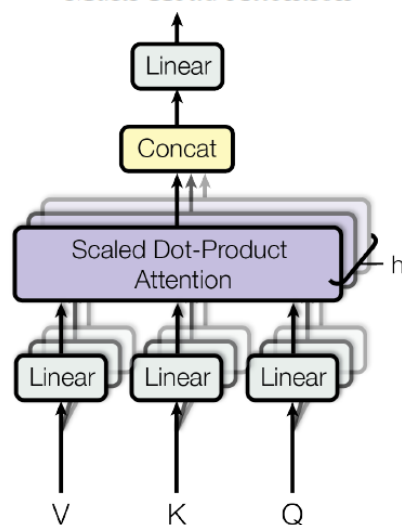
自注意力机制的缺陷，模型在编码时会过渡将注意力集中在自身位置处，而多头注意力机制，通过划分**特征子空间**的方式，对于相同的 **Q**, **K**, **V** 可以学习不同的行为，能够捕获序列各种范围内的依赖关系，有效的解决了过拟合的问题。

在这里首先克隆了 4 个线性层，分别用于生成 **Q**, **K**, **V** 和最后多头拼接。调节 **Q**, **K**, **V**, **Mask** 的形状，并进行 **attention** 计算。这里的打分函数使用的是 **Scaled Dot Product Attention**，得到 **Attention** 的分布，并和 **values** 计算权重和，融合得到相应的特征信息。由于 **d<sub>k</sub>** 越大，**Q\*K<sup>T</sup>** 越大，就可能落到 **softmax** 中梯度平稳区内，所以进行缩放。

Scaled Dot-Product Attention



Multi-Head Attention



### (3) FeedForward 层

因为注意力机制可能对复杂过程的拟合程度不够，通过增加两层网络来增强模型的能力。第一层使用激活函数 ReLU。

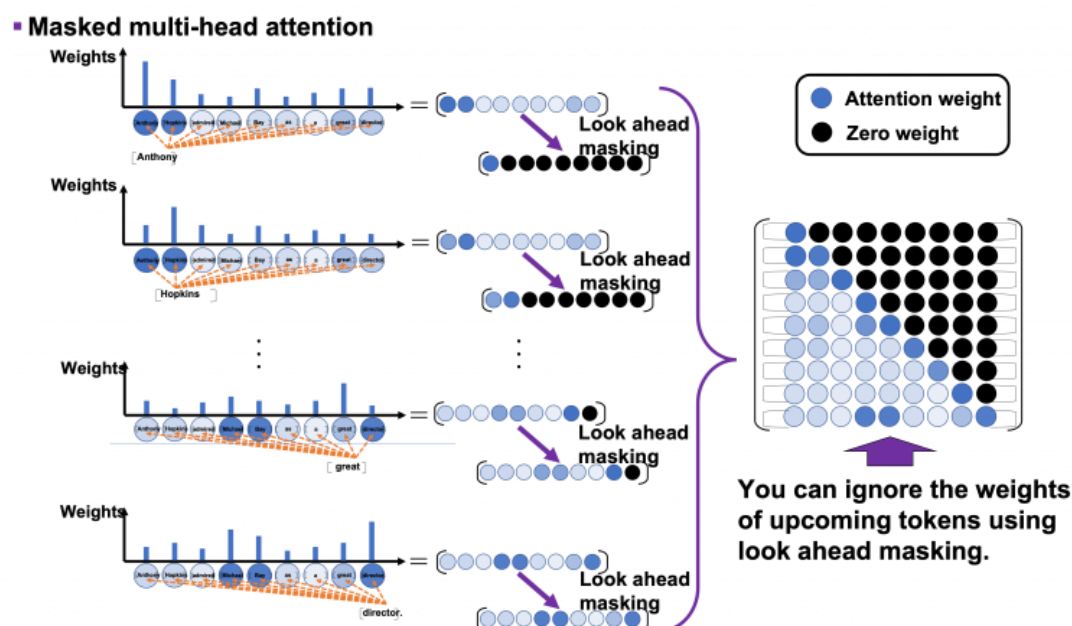
子层之间的连接采用残差连接，并进行层归一化。**残差连接**，可以改善反向传播中，**梯度消失**的问题，使得深层网络训练变得容易，同时打破了网络的对称性，提升了网络的表征能力，避免了**权重矩阵退化**的问题。

归一化可分为批量归一化（batch normalization, BN）和层归一化（layer Normalization, LN）。对于 BN，若 sigmoid 激活函数，如果先 BN 再 sigmoid，BN 使方差接近 1，均值接近于 0，BN 后的数据落在 sigmoid 的线性区内，降低了激活函数的非线性能力，所以应该先经过 sigmoid 函数，再 BN；若 ReLU 激活函数，如果先 ReLU 再 BN，ReLU 使部分神经元失活，再接 BN 后失活的神经元会对结果造成影响，所以这时应该先 BN 再 ReLU。层归一化，是对词向量维度的各个特征进行无量纲化处理，使得不同维度的特征具有可比性。

在 Decoder 中，含有 3 个子层，其中两个为 Multi-Head Attention 层，第一个为 Masked Multi-Head Attention 层，第二个 Multi-Head Attention 层的 K, V 来自 Encoder，而 Q 来自 Decoder。还有一个为前馈层与 Encoder 部分类似，增加网络的拟合能力。

#### (4) Masked Multi-Head Attention 层

这种 Masked 的 Attention 是考虑到输出 Embedding 会偏移一个位置，确保了生成位置  $i$  的预测时，仅依赖小于  $i$  的位置处的已知输出，相当于把后面不该看到的信息屏蔽掉。

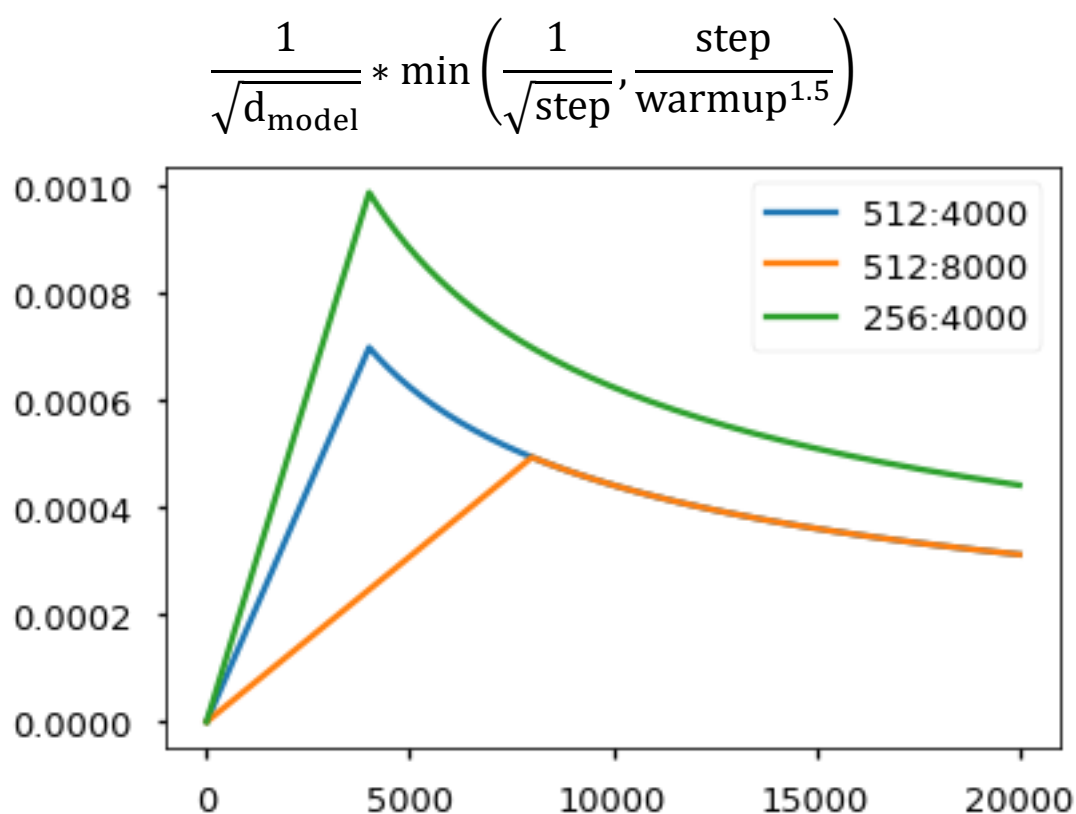


## 4、损失函数和优化器

采用 **label smoothing** 方式，通过添加一个均匀分布的噪声，防止模型在训练时过于相信 label，改善模型的泛化能力。ground truth 分布为  $q(k)$ ，模型预测分布为  $p(K)$ ，在 ground truth 分布中加入噪声，则  $q'(x) = (1-e) q(k) + e*u(k)$ 。交叉熵损失函数  $H(q', p) = (1-e)*H(q, p) + e*H(u, p)$ 。

学习率更新公式如下。以 warmup 为分界点的分段函数，该点之后  $lr_{rate} = \frac{1}{\sqrt{d_{model} * step}}$ ，是 decay 部分，这里采用负幂指数形式，衰减先快后慢。该点之前， $lr_{rate} = \frac{1}{\sqrt{d_{model}}} * \frac{step}{warmup^{1.5}}$ ，是 warm up 部分，这里采用线性函数增长形式，warmup 越大，斜率越小。

在训练初试阶段，模型尚不稳定，较大的学习率会增加收敛的难度，所以使用较小的学习率进行 warmup。在 loss 下降到一定程度后，再恢复为常规的学习率。



## 5、数据处理

### (1) corpus

神经机器翻译领域国际上最常用的数据集为 WMT，很多机器翻译任务都是基于这个数据集进行训练的，这里我们使用 WMT 2018 新闻领域的数据集 (train/dev/test) 作为本次实践的语料库。

---

## (2) word segmentation

使用 `sentencepiece` 工具，采用 BPE 的分词方式，将最常出现的字词对合并，直到词汇表达到预定的大小时停止。使用 `train/dev/test` 数据集构建中英文语料库 `corpus.en` 和 `corpus.ch`，再使用 `sentencepiece.SentencePieceTrainer.Train()` 去训练分词模型，其中中英文词表大小设为 32000，最终得到分词模型 `chn.model` 和 `eng.model`，词表 `chn.vocab` 和 `eng.vocab`。

## (3) Dataset

构建数据集 `train_dataset/dev_dataset/test_dataest`

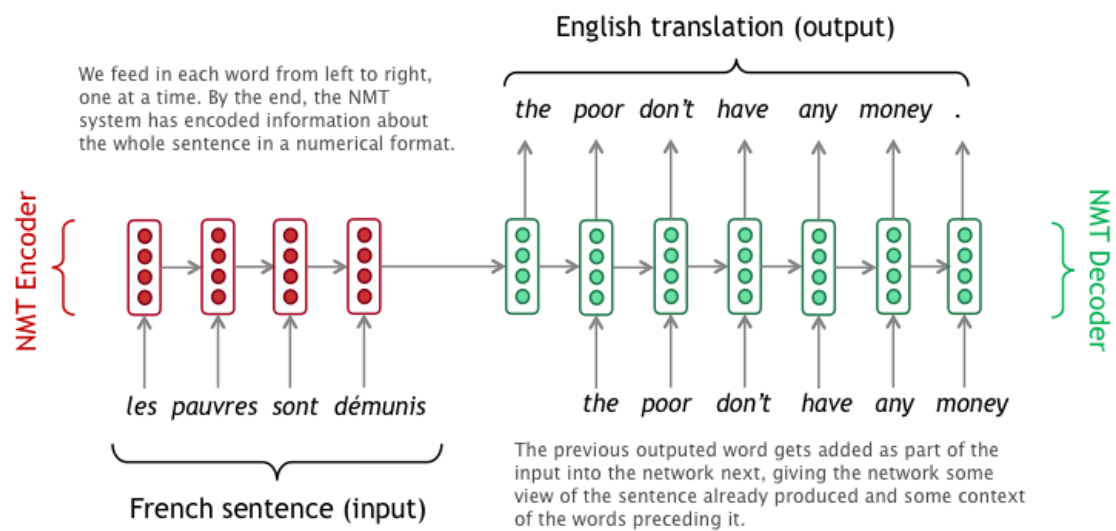
Dataset	Train	Dev	Test
Size	176943	25278	50556

构建数据迭代器 `train_dataloader/dev_dataloader/test_loader`。主要对数据集划分 `batch`，分词后转换为 `tokens`，构建 Transformer 输入格式，对齐处理，构建掩码等。

# 6、训练

## (1) Teacher Forcing

采用 `teacher forcing` 方法。相比于 `free-running` 方法，每一次不使用上一个 `state` 的输出作为下一个 `state` 的输入，而是直接使用训练数据的标准答案 (`ground truth`) 的上一项作为下一个 `state` 的输入，防止训练过程不收敛，出现一步错步步错的情况。



## (2) nn.DataParallel

采用 DP 方式，使用服务器中的 4 个 GPU（0，4，5，6）进行分布式训练：



NVIDIA-SMI 418.56 Driver Version: 418.56 CUDA Version: 10.1									
GPU Fan	Name Temp	Perf	Persistence-M Pwr:Usage/Cap	Bus-Id	Disp. A Memory-Usage	Volatile GPU-Util	Uncorr. Compute	ECC M.	
0 34%	TITAN Xp 57C	P2	92W / 250W	00000000:04:00.0	Off 9718MiB / 12196MiB	36%		N/A Default	
1 53%	TITAN Xp 83C	P2	195W / 250W	00000000:05:00.0	Off 11429MiB / 12196MiB	72%		N/A Default	
2 23%	TITAN Xp 27C	P8	9W / 250W	00000000:08:00.0	Off 10MiB / 12196MiB	0%		N/A Default	
3 23%	TITAN Xp 26C	P8	8W / 250W	00000000:09:00.0	Off 10MiB / 12196MiB	0%		N/A Default	
4 23%	TITAN Xp 33C	P8	8W / 250W	00000000:84:00.0	Off 3691MiB / 12196MiB	0%		N/A Default	
5 23%	TITAN Xp 22C	P8	8W / 250W	00000000:88:00.0	Off 3715MiB / 12196MiB	0%		N/A Default	
6 23%	TITAN Xp 27C	P8	9W / 250W	00000000:89:00.0	Off 3715MiB / 12196MiB	0%		N/A Default	
Processes:									
GPU	PID	Type	Process name	GPU Memory Usage					
0	3480	C	... hong/anaconda3/envs/py3.6/bin/python3.6	1033MiB					
0	12848	C	python	8675MiB					
1	24179	C	python	11419MiB					
4	12848	C	python	3681MiB					
5	12848	C	python	3705MiB					
6	12848	C	python	3705MiB					

### (3) BLEU 分数

每一个训练周期，将训练好的模型在验证集上计算 loss 和 bleu 分数，并在当前最好得分时，保存最佳模型。否则，在连续几个 epoch，bleu 分数均小于最好分数，则提前终止训练，

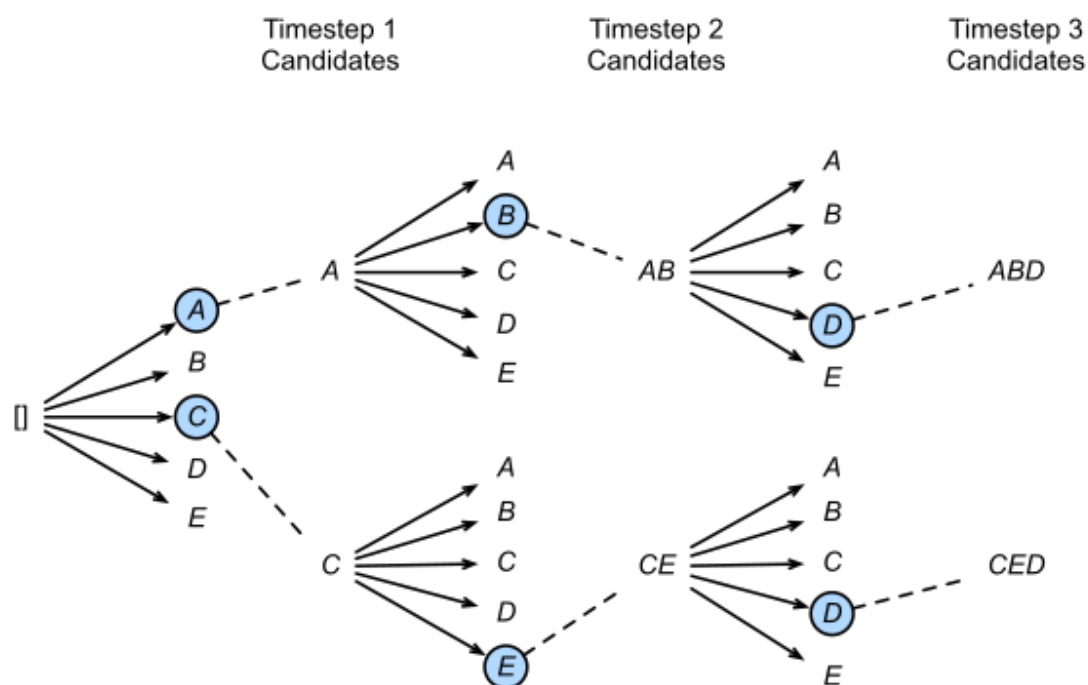
## 7、验证和测试

采用贪心搜索 (greedy search)，每一时间步都取条件概率最大的输出，再将从开始到当前步的结果作为输入去获得下一个时间步的输出，直到出现终结符或最大句子长度。这种方式将原来指数级别的求

解空间直接压缩到与长度线性相关的大小。但是由于舍弃了绝大多数的可能解，只关注于当下的策略无法保证最终得到的序列概率是最优的。

Time step	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

集束搜索（beam search）是对贪心策略的改进。基本思想是放宽考察条件，在每一时间步，不再只保留当前分数最高的 1 个输出，而是保留 num\_beams 个高分用于探测。



在第一个时间步，A 和 C 是最优的两个，结果为[A], [C]。第二步，基于这两个结果继续生成，基于 A 可得到 5 个结果，C 同理。对

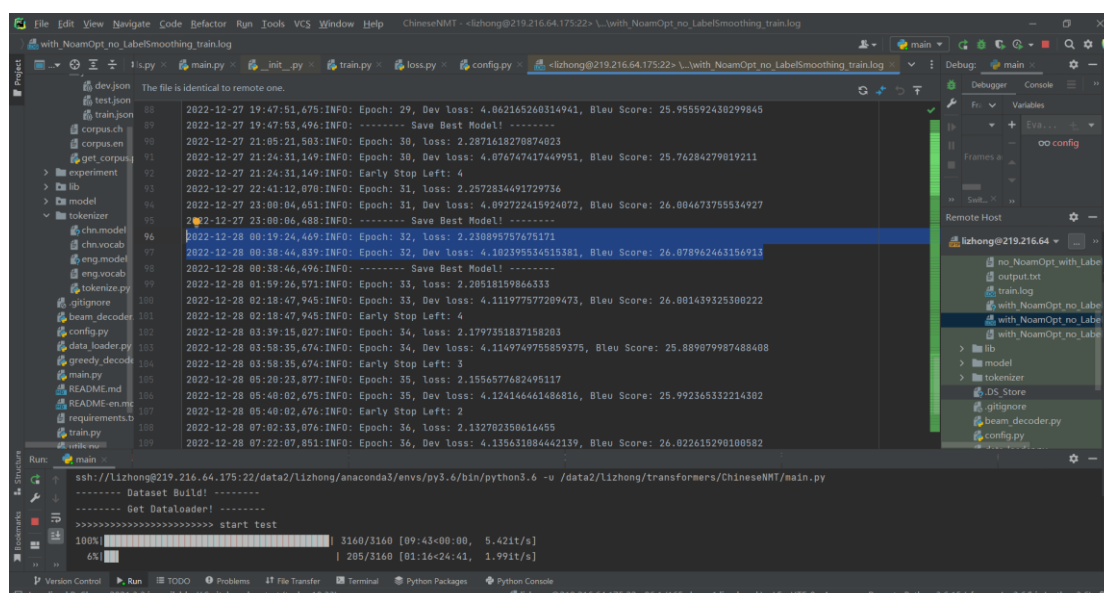
这 10 个结果进行统一排序，保留最优两个，即结果为[AB], [CE]。

第三步，从 10 个结果中保留两个最优，结果为[ABD], [CDE]。

由此，beam search 每一步考察候选结果数量是贪心策略的 num\_beams 倍，因此是一种以时间换性能的方式。

## 8、实践结果

在训练集上，训练 32 个周期后达到最佳模型，训练集上损失为 2.23，验证集上损失为 4.10，BLEU 分数为 26.07。



在测试集上，分别在不同的 beam\_size 上计算 BLEU 分数，结果如下：

Beam_size	2	3	4	5
bleu	26.67	26.81	26.84	<b>26.87</b>

由此可见，模型在 beam\_size 为 5 时，达到最好的效果。

使用模型进行翻译：

- 例子 1.
- src:  
The researchers concluded that selfreported overall health and depres

	sion improved among those who enrolled <b>in</b> Medicaid, <b>and</b> that there was an increase <b>in</b> the diagnosis <b>and</b> treatment of diabetes <b>for</b> this group.
3.	
4. tgt:	
5.	研究者的结论是,在参加了医疗补助的人中间,自我报告的健康和抑郁情况有所好转,并且这一群体的糖尿病诊断和治疗数量也有所增加。
6.	
7. trans:	
8.	研究人员认为,在 Medicaid 中,自我报告健康及抑郁症的研究者中,接受 Medicaid、治疗糖尿病这一组数量的增加了。
9.	
10. 例子 2.	
11. src:	
12.	There <b>is</b> a vast number of important Buddhist sites <b>in</b> Swat <b>and</b> other areas of northwest Pakistan.
13.	
14. tgt:	
15.	在斯瓦特河谷和巴基斯坦西北部有着大量重要的佛教文物。
16.	
17. trans:	
18.	斯德利斯和其他巴基斯坦北方的北方的佛教网站数量巨大
19.	
20. 例子 3.	
21. src:	
22.	But Howard Hughes's success as a film producer <b>and</b> airline owner made him one of the richest Americans to emerge during the first half of the twentieth century.
23.	
24. tgt:	
25.	但霍华德·休斯作为电影制片人和航空公司老板的成功使得他跻身 20 世纪前半叶最富有的美国人行列。
26.	
27. trans:	
	28. 但霍华德·赫伯特作为电影生产商,航空所有者他在 20 世纪首代美国人出现的最富有美国人之一。

## 9、可能存在的问题

(1) teacher-forcing 方法过于依赖 ground truth 数据,在训练过程中,模型的会有比较好的效果,但是在测试的时候,因为不能得到 g

---

round truth 的支持，如果和训练数据有很大差异，模型会变得脆弱。

(2) beam search 方法生成的结果比贪心搜索好一些，但还是会遇到诸如词语重复这样的问题，需要以后去解决。

(3) DP (nn.DataParallel) 分布式训练方式，会出现负载不均衡的现象，受 GPU0 (逻辑) 影响很大，并且通信开销很大，比单卡训练效率上并没有多大提高，可以采用 DDP (nn.parallel.DistributedDataParallel) 方式。

## 10、参考

[1] Transformer 代码详解 [EB/OL]. [2023-01-02]. [https://blog.csdn.net/weixin\\_40548136/article/details/100163727](https://blog.csdn.net/weixin_40548136/article/details/100163727).

[2] 教你用 PyTorch 玩转 Transformer 英译中翻译模型！[EB/OL]//知乎专栏. [2023-01-02]. <https://zhuanlan.zhihu.com/p/347061440>.

[3] 碎碎念：Transformer 的细枝末节[EB/OL]//知乎专栏. [2023-01-02]. <https://zhuanlan.zhihu.com/p/60821628>.

[4] VASWANI A, SHAZEER N, PARMAR N, 等. Attention is All you Need[C/OL]//Advances in Neural Information Processing Systems: 卷 30. Curran Associates, Inc., 2017[2023-01-02]. <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.