华东师范大学数据学院上机实践报告

课程名称：当代人工智能　　年级：2018　　上机实践成绩:
指导教师：李翔　　姓名：孙秋实
上机实践名称：基于 BERT 的机器翻译　　学号：10185501402　　上机实践日期：2021/10/21

Part 1
实验目的

(1) 在 WMT16 数据集上完成英语 → 德语翻译任务

(2) 完成神经机器翻译任务性能评估

Part 2
实验任务

(1) BERT 预训练模型介绍与理解

(1) 使用 BERT 预训练模型完成机器翻译任务

(2) 模型评估

Part 3
使用环境

(1) Google Colab

(2) Python3.7

(3) Pytorch 1.8

(4) GPU 型号：Tesla K80/V100

Part 4
实验过程

Section 1
数据集简介

本次实验使用的是 wmt-16 的英德数据集，通过 Hugginface 的 Datasets 库直接加载在 Colab 中

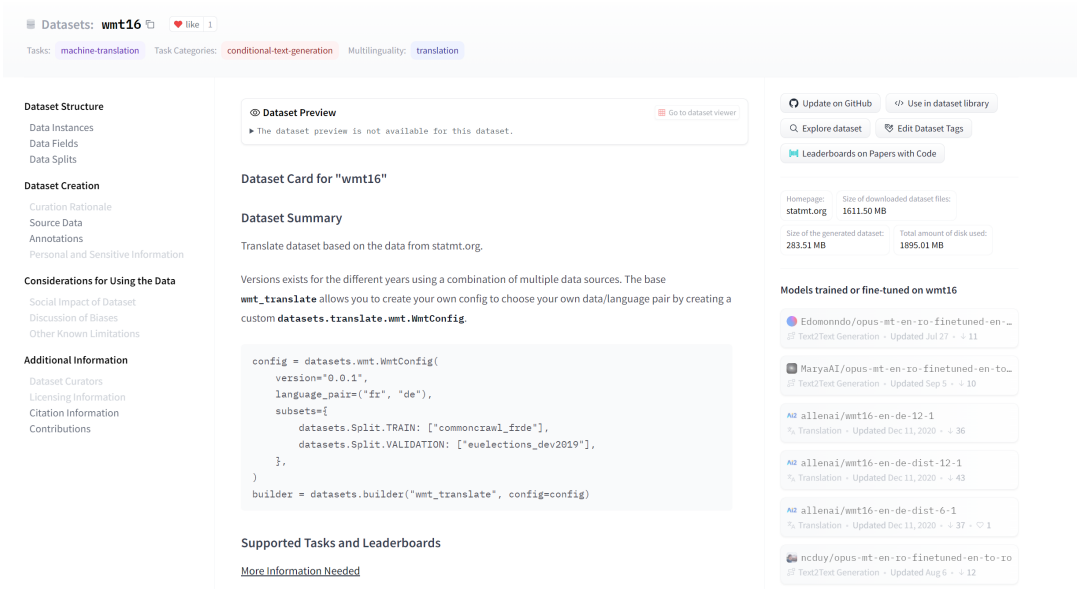

图 1: WMT 16 Dataset

调用方法也很容易，如下

```
from datasets import load_dataset
en_de_raw_datasets = load_dataset("wmt16", "de-en")
```

随后随机抽取几个样本查看数据形式，如图 2 所示

	translation
0	{'de': 'Sobald ein Land die Konvertierbarkeit in Gold abschaffte, was es in der Lage, einen Kurs der Geldmengenexpansion zu verfolgen, was tendenziell zu einem Ende der Deflation führte.', 'en': 'Once a country eliminated convertibility into gold, it was free to pursue monetary expansion, and deflation tended to end.'}
1	{'de': 'Es kann nicht sein, daß lose abgepackte Lebensmittel nicht gekennzeichnet werden müssen.', 'en': 'It is unacceptable that loose foodstuffs need not be labelled.'}
2	{'de': 'Die Läufer, klassiert vom 61. bis zum letzten Platz nach zwei Etappen, starten deshalb in Lacanau morgens um 7 Uhr 30.', 'en': 'The runners ranking from the 61st to the last position after 2 stages will, therefore, start out from Lacanau at 7h30 in the morning.'}
3	{'de': 'Einige Kilometer von Colle di Val d'Elsa entfernt, in der Nähe der Quelle des Flusses Elsa.', 'en': 'A few kilometres from Colle di Val d'Elsa, close to the springs of the river Elsa.'}
4	{'de': '(Beifall)', 'en': '(Applause)'}

图 2: WMT16 En-DE Dataset Samples

Section 2
BERT 模型

*考虑到从零训练 *Transformer* 类模型会受到算力的限制，在经过授课老师的同意的前提下，本次实验使用了已经预训练好的模型，以在本次实验报告中，我主要依赖 *Attention is all you need* 以及 *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* 这两篇文章，以及动手深度学习，结合课堂知识，以及我自己对模型的理解，对模型做一个解释。

Part 1
（自）注意力机制

自注意力，即将词元序列输入注意力池化中，在注意力机制的基础上让同一组词元同时充当 Q(Query), K(Key), V(Value)

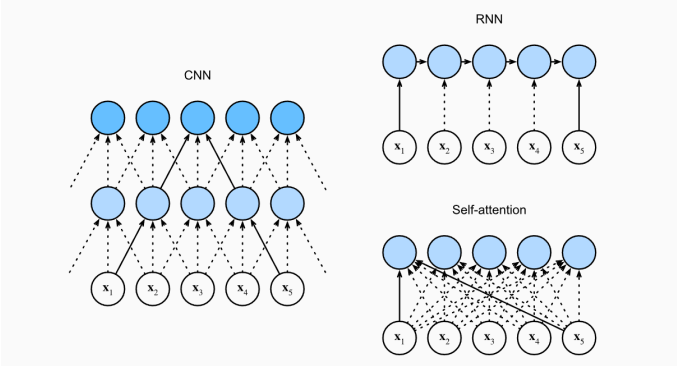

图 3: CNN vs RNN vs Self-Attention

以往我们使用卷积神经网络（CNN）或循环神经网络（RNN）对序列进行编码，目标都是将由词元组成的序列映射到另一个长度相等的序列，其中的每个输入词元或输出词元都由 d 维向量表示。具体来说，我们将比较的是卷积神经网络、循环神经网络和自注意力这几个架构的计算复杂性、顺序操作和最大路径长度。

	CNN	RNN	Self-Attention
复杂度	$O\left(knd^2\right)$	$O\left(nd^2\right)$	$O\left(n^2\ d\right)$
并行度	O(n)	O(1)	O(n)
最长路径	O(n/k)	O(n)	O(1)

表 1: CNN vs RNN vs Self-Attention

需要注意

- RNN 为顺序操作，会严重影响并行度
- 路径越短，就能更轻松地学习序列中的依赖关系

因此，在算力允许的情况下，我们显然选择自注意力机制用于对文本序列的编码。

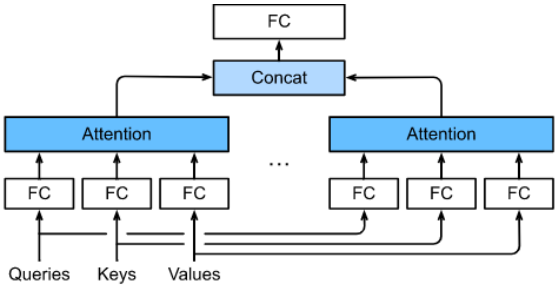

图 4: Multi-Head Attention

在 Transformer 模型中使用了多头注意力机制，它允许注意力机制学习多组 Q,K,V 的线性投影，最后将其组合作为输出这会增加模型计算的开销，但是对模型的性能显然是有益的。在 Transformer 中，每一个自注意力计算的输出都被称作一个“头”。

Part 2
Transformer

Transformer 是 Attention is all you need 提出的模型结构，如图 9 所示，Transformer 作为 Encoder-Decoder 结构的一个实例，Transformer 的编码器和解码器是基于自注意力的模块叠加而成的，源（输入）序列和目标（输出）序列的嵌入（embedding）表示将加上位置编码（positional encoding），再分别输入到编码器和解码器中。Transformer 模型完全基于注意力机制，没有任何卷积层或循环神经网络层。

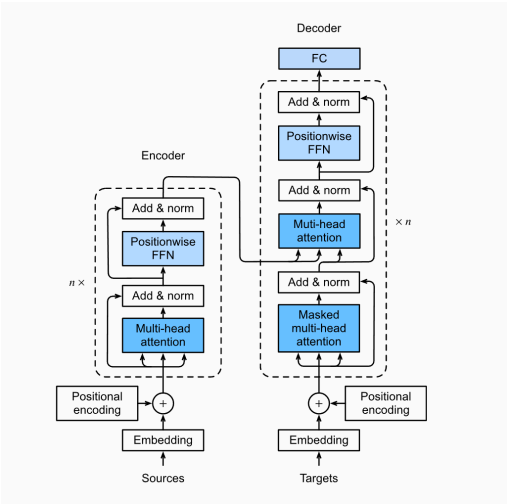

图 5: The Transformer Architecture

在 Transformer 结构中：

- 每个层都有两个子层（子层表示为 Sublayer ）
  - Layer1: 多头自注意力
  - Layer2: Positionwise feed-forward network（其实就是前馈神经网络）
- 子层之间采用残差链接，然后使用 Layer Norm

同样的，解码器也是由多个相同的层叠加而成的，并且层中使用了类似的残差连接和层归一化。除了编码器中描述的两个子层之外，解码器还在这两个子层之间插入了第三个子层，称之编码器 - 解码器注意力（encoder-decoder attention）层。在这个编码器 - 解码器注意力中，Q 来自前一个解码器层的输出，而 K 和 V 来自整个编码器的输出。

更多关于 Transformer 的内容可以参考：The Annotated Transformer

Part 3
Pre-training BERT

BERT 是基于 Transformer 的改进工作，原文中的实现完全照搬了 Transformer 的工程实现，摘要中对其描述为“conceptually simple and empirically powerful”。在 BERT 出现之前，面向下游任务的预训练模型可以被分为两种：

- Feature-based model
- Fine-tuning model

其中，Feature-based 的 ELMo 仍然使用了双层 RNN 架构，而 Fine-tuning model，如 GPT，使用了 unidirectional language models 学习一个具有泛化能力的语言表示。其缺点是，在做 question answering 一类的 token-level tasks 时候，无法考虑到文本的双向信息。

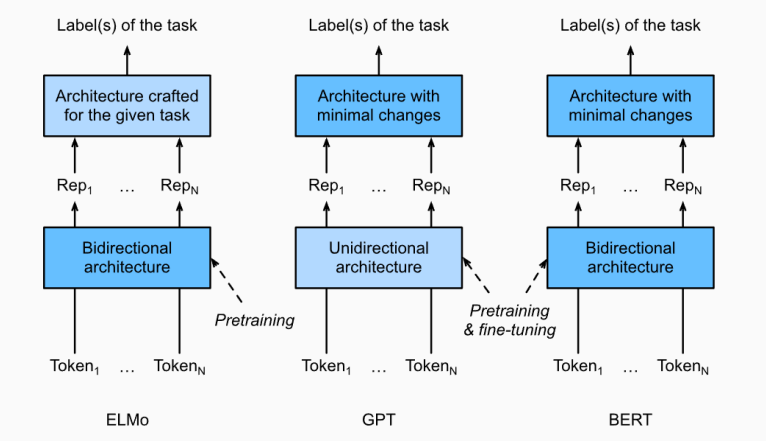

图 6: A Comparison of ELMo, GPT, and BERT

而 BERT，解除了上述工作的单向限制，并受到 Cloze 任务的启发，在预训练阶段使用了遮蔽语言模型。此外，还使用了 NSP('Next Sentence Prediction') 任务来训练文本对。三者的对比如图 6 所示。

对 NSP 任务的解释

以原始论文附录中的一个 case 进行简单解释

正例

$Input = [CLS]\ the\ man\ went\ to\ [MASK]\ store\ [SEP]$

$he\ bought\ a\ gallon\ [MASK]\ milk\ [SEP]$

$Label = IsNext$

负例：

$Input = [CLS]\ the\ man\ [MASK]\ to\ the store [SEP]$

$penguin\ [MASK]\ are\ flight\ \#\#less\ birds\ [SEP]$

$Label = NotNext$

可以看到在第二组数据中出现了 $flight\ \#\#less$，这是因为原始论文使用 WordPiece 处理后将这个不常见词切分为了 subword 导致的。

在预训练阶段，BERT 使用了两个相当大的数据源进行训练

- 800M words 的 BooksCorpus
- 2,500M words 的 English Wikipedia

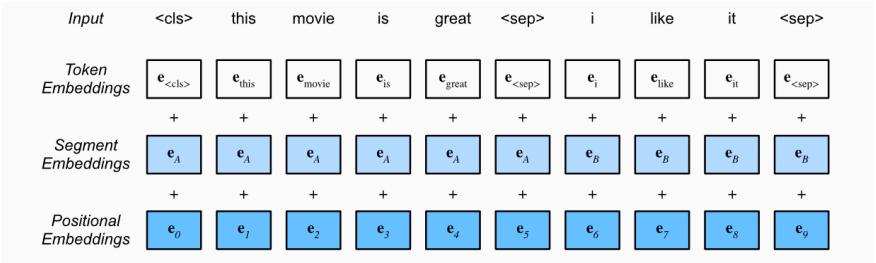

图 7: The Embeddings of the BERT Input Sequence

BERT 的输入表示可以被分为三个部分，如图 7 所示：

- Token Embeddings，可以将其理解为词源
- Segment Embeddings，可以将其理解为“文本中的第几句话”
- Position Embeddings，可以将其理解为当前词在文本中的位置

而 BERT 的输入表示，就是这三个 Embedding 的相加。

Part 4
Fine-tuning BERT

在已有预训练好的 BERT 时，就可以将其部署在下游任务上，值得一提的是，对于每一个下游任务，都需要启用一个独立的 BERT 模型，加载好权重后分别进行训练。

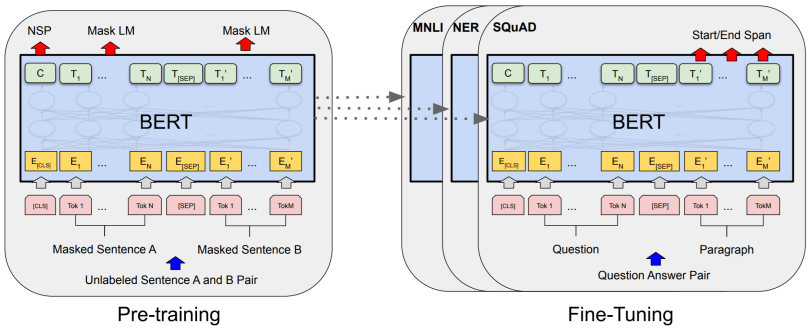

图 8: Overall Pre-Training and Fine-Tuning Procedures for BERT

在该微调阶段，对数据集的预处理为

- Normalization
- SentencePiece

Remark：SentencePiece 是一种分词的方法，在 BERT 原文中使用了 WordPiece，其目的为通过将不常见词切分为 subword 来压缩词典的大小，而 SentencePiece 是一个和它类似的 Google 开源工具，把英语单词切分更小的语义单元（grams），缩小词典大小，具体可参考 Google-SentencePiece

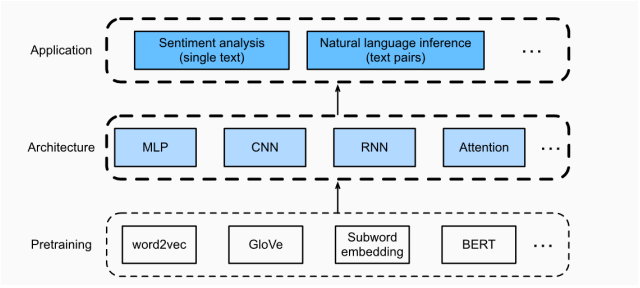

图 9: Different Downstream NLP Tasks

现在我们就可以将 BERT 运用于机器 Neural Machine Translation 任务上了，针对这个任务，需要为 BERT 再提供一个机器翻译任务的输出层，并设定参数。

- max sequence length = 64（受 Colab 资源限制）
- batch size = 8 （受 Colab 资源限制）
- learning rate = 2e-5

- weight decay = 0.01
- Optimizer = Adam

* 在微调阶段可以使用 AutoModelForSeq2SeqLM 这个类，直接帮助初始化一个 Seq2Seq 的机器翻译任务输出层。

Section 3
评价指标

本次实验中使用了 Hugging Face 提供的 sacrebleu 进行评价，以下为微调 BERT 模型时的 sacrebleu 评估结果

表 2: BERT Neural Machine Translation (Metrics=Sacrebleu)

Step	Training Loss	Validation Loss	Bleu
1000	1.7219	1.38277	28.7245
1500	1.7506	1.398325	28.7016
2000	1.9825	1.395685	28.6763
2500	2.1241	1.389101	28.9019
3000	2.1783	1.390327	29.0592
3500	2.098	1.383869	29.1901
4000	2.0543	1.395263	28.6873
4500	2.1105	1.3779	29.3813
5000	2.1623	1.380088	29.1342
5500	2.0758	1.382722	28.9144
6000	2.1188	1.382792	29.2525
6500	2.0746	1.381315	29.3518
7000	2.0952	1.382055	29.3291
7500	2.1224	1.37487	29.6879
8000	2.0744	1.37798	29.1137
8500	2.0886	1.377004	29.328
9000	2.082	1.380615	29.2838
9500	2.1228	1.377854	29.288
10000	2.1325	1.376926	29.2611
10500	2.1515	1.385205	29.254
11000	2.0974	1.374222	29.6186

可以看到微调过程中验证集性能可以达到 29.6 左右的 BLEU 值，测试集性能可以达到 29.3，虽然 BERT 的原文没有涉及机器翻译任务，但可以参考 Attention is all you need 中 Transformer 的实验结果进行对比。

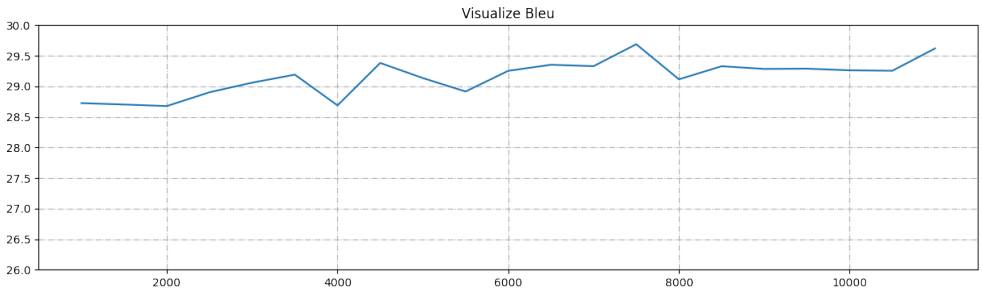

图 10: BLEU Visualization

图 11 为 Transformer 的实验结果

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

图 11: Transformer results

Remark: SacreBLEU是简单 BLEU 的一个变种，传统计算 BLEU 的方式存在一些问题，其中最主要的问题是已有的计算方式需要用户自己提供 Tokenize 过的结果，而且还需要 Tokenize 过的译文，而不同实验中 Tokenize 的方式不同，产生的结果就会不同，例如是不是会把复合词中间的连接符，标点等分开，<unk> 如何计算等等。

如果要替换为原始 BLEU，则使用下方所展示的代码：

```
from datasets import load_metric
metric = load_metric("bleu")
res = metric.compute(predictions=predictions, references=references) #
    参数分别为模型译文和译文参考
```

可见，使用 BERT 预训练模型来完成 EN-DE 的翻译任务，在非常不充分的微调下也可以比 Transformer(base) 和 Transformer(big) 分别高 2 个 BLEU 和 1 个 BLEU。

（受到 Colab 的 GPU 使用时间限制，这个实验微调过程中只训练了 1/3 个 Epoch，理论上若能继续 Fine Tuning 性能可以更高一些）

Section 4
Reference

参考文献（链接）：

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- The Annotated Transformer

- Helsinki-NLP/opus-mt-en-de