**华东师范大学数据科学与工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：当代人工智能 | **年级**：2021级 | **上机实践成绩**： |
| **指导教师**：李翔 | **姓名**：李睿恩 | **学号**：10215501434 |
| **上机实践名称**：文本摘要 | | **上机实践日期**：2023.12.22 |
| **上机实践编号**：4 | **组号**：无 | **上机实践时间**：0:33 |

1. 实验目的

* 实现文本摘要在医学领域的应用。

1. 实验任务

* 构建Seq2Seq模型完成此次文本摘要任务。
* 选用一种Encoder-Decoder结构，选择包括但不限于RNN / LSTM / GRU / Transformer / BERT / BART / T5 / OPT等。本次实验实现了BART与T5。
* 使用一种或多种评估指标对文本摘要结果进行分析。

1. 实验环境

本次实验基于**AutoDL算力云**平台。我们在AutoDL算力云平台中租用了2块RTX 4090(24GB)的GPU，并在其提供的JupyterLab平台中进行了本次实验。

我们所租用的实验环境基于Ubuntu 20.04，采用Python 3.8作为编程语言，并使用了PyTorch 2.0.0框架进行深度学习任务，同时借助CUDA 11.8实现对GPU的加速。

1. 实验过程

**4.1 Seq2Seq与Encoder-Decoder结构介绍**

Seq2Seq（Sequence-to-Sequence）是一种神经网络架构，用于处理序列到序列的任务。它是自然语言处理（Natural Language Processing）领域目前主流的神经网络架构。

Seq2Seq模型一般由两个主要部分组成：Encoder（编码器）与Decoder（解码器）。

Encoder一般负责将输入序列转换为一个固定维度的表示，这个表示通常被称为隐藏状态向量，该向量包含了输入序列的信息。Encoder的目标就是捕捉输入序列的语义信息，使得这些信息能够之后被Decoder有效地使用。

Decoder会接受Encoder生成的向量，将其解码为目标序列。解码器逐步生成目标序列的元素，每一步都依赖于前一步生成的元素。Decoder的目标就是根据隐藏状态向量生成与输入序列对应的输出序列。

Encoder-Decoder架构如今被广泛使用，他们被使用在机器翻译、语音识别、文本摘要等任务重，有着非常不俗的表现。

**4.2 Tranformer架构简介**

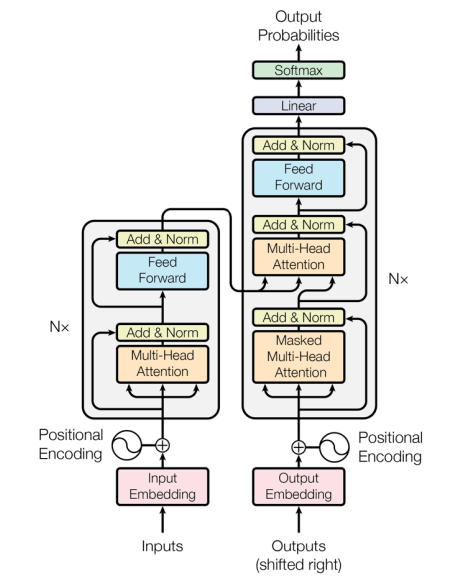
传统的循环神经网络（RNN）、长短期记忆神经网络（LSTM）与门控循环单元（GRU）是早期用于进行文本摘要的工具。但是，他们的效果并不让人满意。一方面，他们会存在较长的链条，这使得他们存在长期依赖问题，进而导致难以解决的梯度消失或梯度爆炸。另外，他们是逐步处理序列的，这使得他们难以掌握全局的信息。

Transformer是一种更现代、基于自注意力机制（Self-attention）的神经网络架构。自注意力机制允许Transformer直接对整个序列进行并行处理，提高了计算效率和模型的学习能力。Transformer还引入了位置编码，通过在输入嵌入中添加位置信息，模型可以区分不同位置的元素。

最新的Transformer还会使用多头注意力机制（Multi-head Attention），使得Transformer能够学习多个不同的注意力权重，从而更好地捕捉输入序列中的不同关系，使得模型更具表达能力。

基于上述原因，Transformer成为了目前最流行的架构。

目前最流行的自然语言处理中会使用的架构有BERT, GPT, T5等，他们也是基于Transformer架构的Encoder-Decoder架构。区别在于，BERT是只使用了Encoder的架构，GPT是只使用了Decoder的架构，而T5是完整使用了Encoder与Decoder的架构。图1展示了Transformer架构的一个示意图。



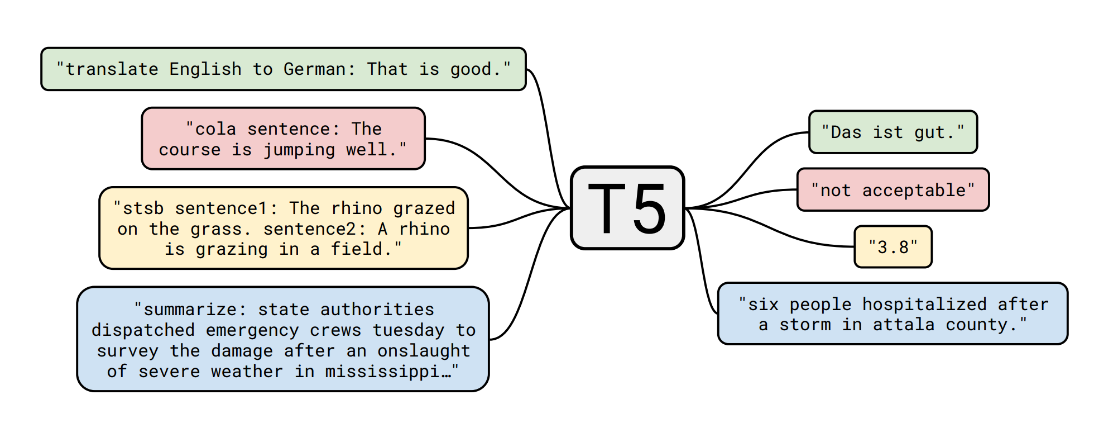
**图1 Transformer架构示意图**

**4.3 T5架构简介**

T5框架将所有自然语言处理任务重构为统一的文本到文本格式。其输入和输出始终是文本字符串，这与BERT模型形成了鲜明对比，因为BERT类的模型的输入有一定的局限性。T5框架允许我们在任何自然语言处理任务上使用相同的模型、损失函数与超参数，这包括机器翻译、文本摘要、机器问答和分类任务。图2展示了T5架构能实现的功能。

T5框架是一个有效的迁移学习技术。在开源的网站上获取的T5框架是受到过预训练的，我们只需要对框架进行微调，就可以将其作为有效的模型投入到使用中。

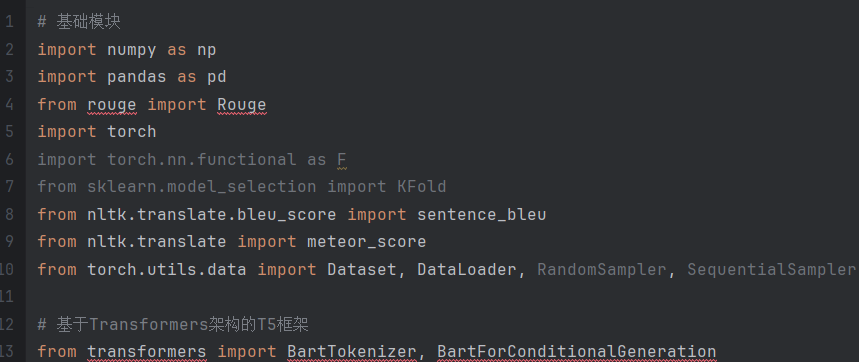
T5的模型有许多规模，如T5-Small，T5-Base，T5-Large。在这里，我们将尝试使用T5-Base框架，基于AutoDL云上服务器，使用训练集对模型进行微调，从而尝试实现文本摘要。

****

**图2 T5架构功能介绍**

**4.3.1 引入必要模块**

为了完成此次试验，一些模块的调用是必须的。调用的模块如图3.



**图3 调用模块**

其中，numpy与pandas是用于数据处理的模块；torch模块用于创建张量、加载数据、创建优化器与构建神经网络；nltk.translate模块可以协助我们使用BLEU-4和METEOR评估指标观测模型效果；transformers架构可以允许我们调用T5的预训练模型；我们还调用了Rouge模块，使用Rouge评估指标观测模型效果。

**4.3.2 自定义数据集**

本次实验我们将使用给定的train.csv文件作为训练集，test.csv文件作为测试集。为了能够更适配深度学习的模型，我们编写了MedicalDataset函数，从而使得数据集被处理成PyTorch更能适应的数据格式。

在该函数中，我们将数据统一了格式。首先，如果数据文本中有多个连续空格，我们会将这些空格合并为一个空格，避免数据格式的不统一。之后，我们会将文本转为模型可以处理的编码（即Encoder），但这些数据有一个最大长度限制，并且如果编码后的长度不满足最大长度限制，我们会要求编码对空缺位进行填充。我们会要求最后返回的编码是PyTorch可以处理的张量的形式，并且如果最后得到的编码超出了最大长度，我们会截断超出最大长度的部分。最后，我们会使用PyTorch中的squeeze函数，移除得到的编码中大小为1的维度，使得张量形状更加紧凑。其中，source\_ids与target\_ids返回了编码后的token IDs，source\_mask返回了二进制掩码，表明哪些位置是填充位，是注意力机制不应该关注的地方。图4展示了我们编写的MedicalDataset函数。



**图4 MedicalDataset的实现**

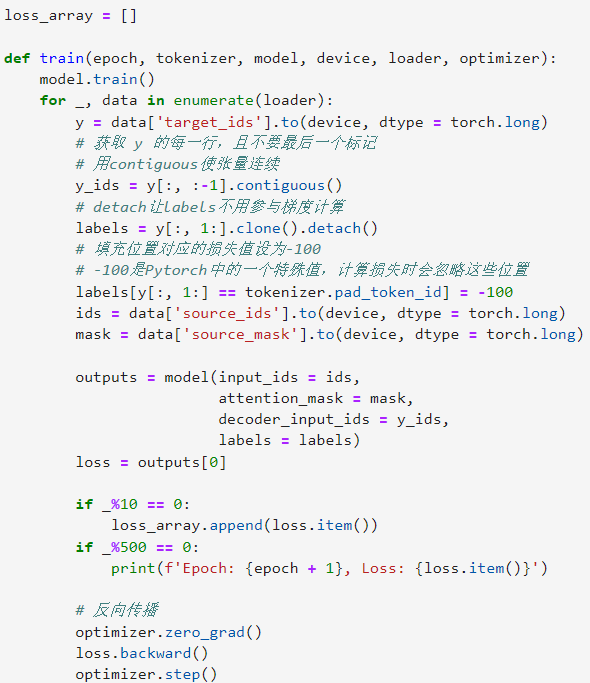
**4.3.3 编写训练函数**

我们需要基于确定格式的数据，设计训练函数。

在训练的初期，我们会要求获取输入数据的每一行，并摒弃最后一个标记位。随后，我们提取摘要文本除最后一个编码以外的所有编码作为Decoder的输入，记为y\_ids，并且在这里我们还会使用contiguous函数使得张量连续。我们也提取了摘要文本除第一个文本以外的所有编码作为Decoder正确的输出，记为labels，并且在这里我们还使用了detach函数，要求这部分数据不参与梯度的计算。接下来，考虑到labels中存在一些因为填充而存在的没有意义的编码位，我们要求这些位置的数值被计为-100，这个数字在PyTorch中是一个特殊值，能使得我们在计算时可以忽略这些位置。

基于此，我们可以将这些数据投入到模型的训练中。我们在训练的过程中还要求，每遍历10个数据，就必须记录当时的损失函数，每遍历500个数据，就必须输出目前的损失函数值。这样的方式有助于我们可视化训练的过程，确保我们的模型的确学习到了文本的特征。

最后，我们使用反向传播，真正完成深度学习模型的训练过程。图5展示了train函数的编写。



**图5 train的实现**

**4.3.4 编写测试函数**

测试函数与训练函数在逻辑上有不小的区别。

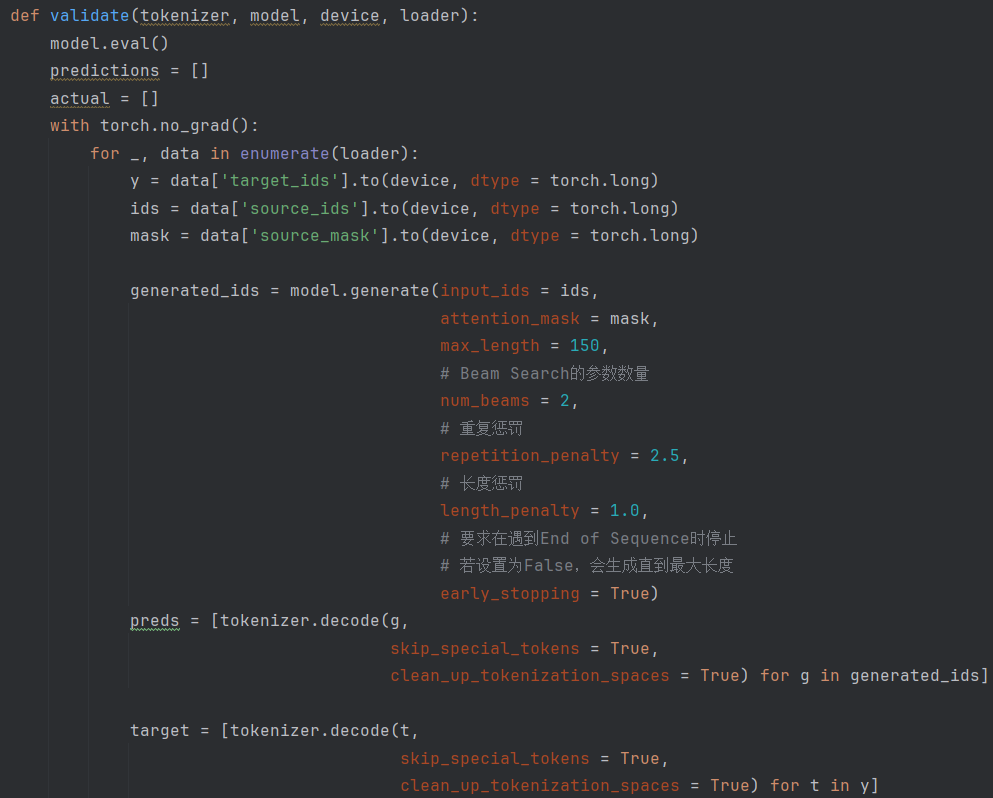
我们开启模型的验证模式，并且设置一定的参数。我们对该模型输入原始文本，设置注意力机制中的掩码（attention\_mask），并要求其最终生成的摘要的长度（max\_length）不可以超过150。在这里，我们还结合了束搜索（Beam Search）的思路，在每个时间步，选择当前最有可能的2个假设序列（num\_beams），从而保留文本的多样性。我们还为重复标记设置惩罚（repetition\_penalty），这可以鼓励模型生成不包含重复标记的序列。我们设置了长度惩罚（length\_penalty），防止模型输出过短或过长的序列。另外，我们还设置了早停（early\_stopping），要求模型在遇到序列结束标记的时候停止生成，如果不设置该值为True，则模型会生成直到达到最大长度。

最后使用Decoder，我们可以获得生成的文本与原始的文本。图6展示了验证过程的代码。

**4.3.5 设置训练背景**

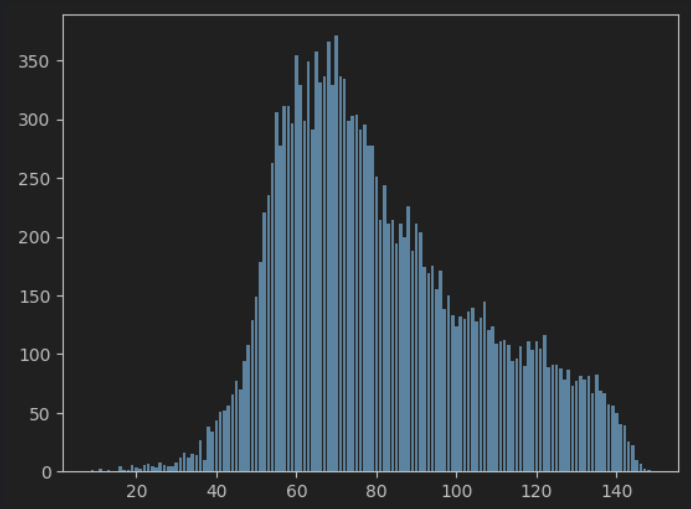
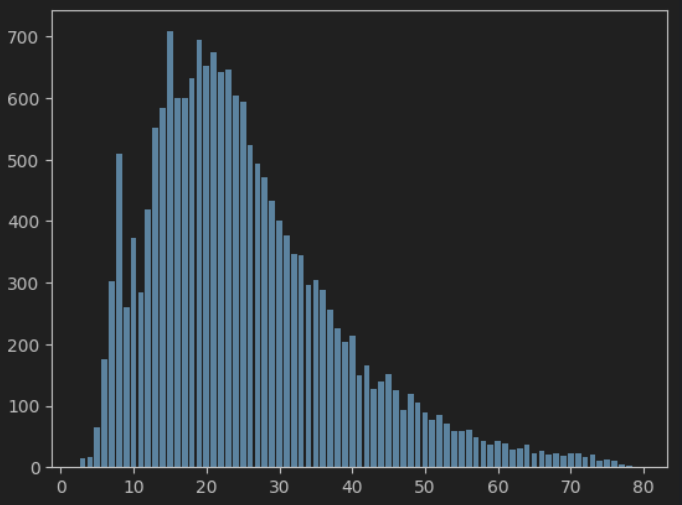
在真正投入预测使用前，我们需要设置一些常数与训练背景。

我们设置了tokenizer是基于Hugging Face Transformers库中的T5Tokenizer类中的预训练过的T5-Base模型的分词器。这个分词器专门用于训练T5-Base的模型。



**图6 validate的实现**

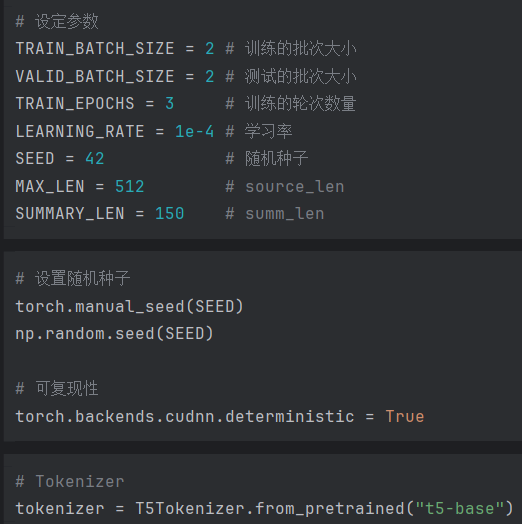
我们先观测数据集中文本的大小。通过数据处理与绘图，我们可以得到如图7的原始文本长度统计与如图8的摘要文本长度统计。这有助于我们确定编码的合适长度，以得到更好的模型表现。



**图7 原始文本长度统计 图8 摘要文本长度统计**

我们设定了训练的批次大小（TRAIN\_BATCH\_SIZE）为2，测试的批次大小（VALID\_BATCH\_SIZE）为2，从而让GPU可以发挥其效用。我们还设置了训练轮数（TRAIN\_EPOCHS）为3，这主要是因为在我们的多次实验中，我们发现过多的轮数不但耗时，最后得到的损失函数值也没有显著的下降，而且它还可能还导致了过拟合现象。我们另外设置了学习率为10-4，设置一个用于numpy与torch随机划分的随机数种子（SEED）为42，规定上文提到的原始文本编码最大长度为512，摘要编码最大长度为150。最后，我们设置torch.backends.cudnn.deterministic = True，这确保了每次训练的可复现性。

训练背景的设置整体如图9所示。

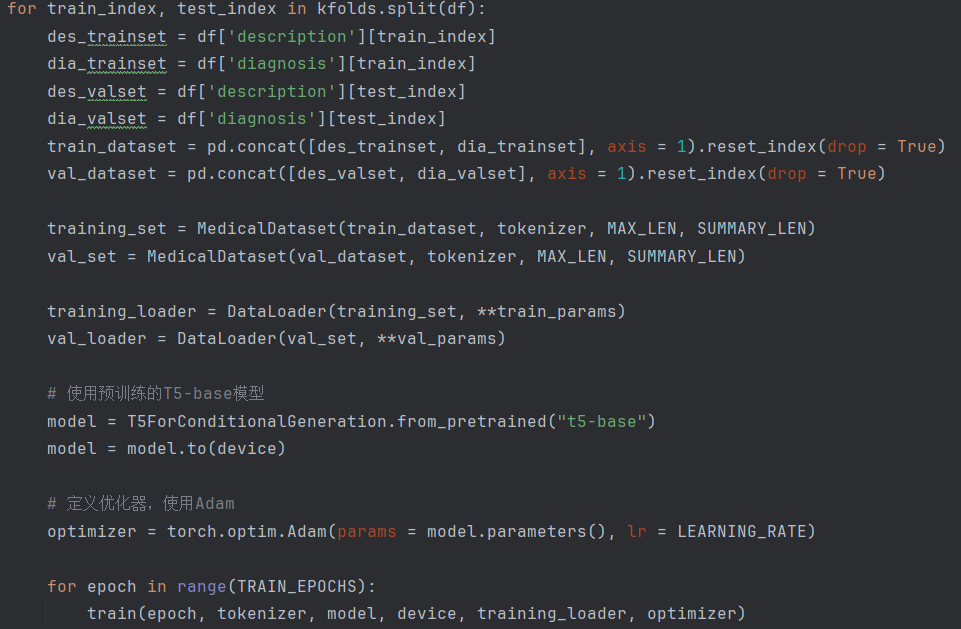


**图9 训练背景设置**

**4.3.6 五折交叉验证**

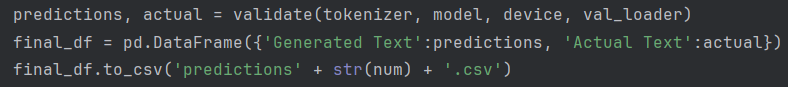
为了验证我们的模型是否是合理的，我们在此进行五折交叉验证。我们调用KFold模块，从而确保我们每次都采取了完全没有重叠的验证集。

我们需要先进行模型的训练。我们使用了自定义数据集的函数MedicalDataset，使用DataLoader进一步统一数据格式。随后，我们调用T5ForConditionalGeneration中的T5-Base预训练模型，我们将基于该预训练模型进行微调。最后，我们选择Adam作为优化器以进行对T5-Base模型的微调。五折交叉验证的训练代码执行流程如图10.



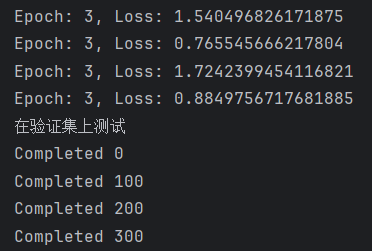
**图10 五折交叉验证的训练代码实现**

然后我们需要进行模型的验证。只需要直接调用validate函数即可实现模型的验证。最后，我们将每次交叉验证使用的原始摘要文本与生成的摘要文本存储在一个.csv文件中。五折交叉验证的验证代码如图11所示。



**图11 五折交叉验证的验证代码实现**

最终我们可以得到如图12的输出。

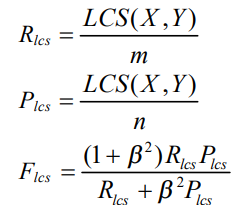
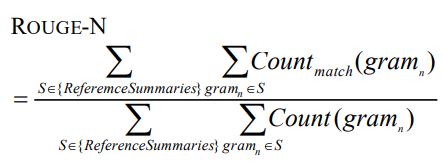


**图12 五折交叉验证的训练与验证输出**

**4.3.7 生成摘要的评估**

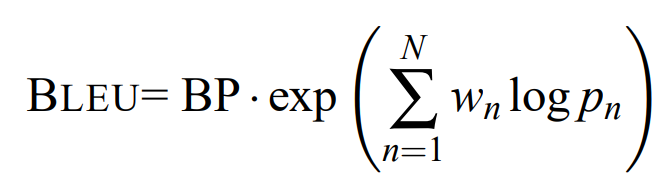
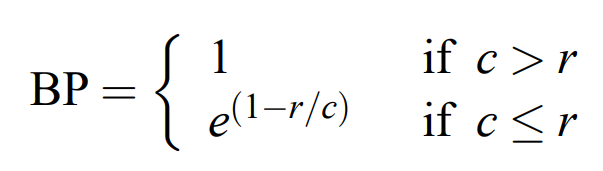
在这里，我们将使用Rouge（Recall-Oriented Understudy for Gisting Evaluation）、BLEU（Bilingual Evaluation Understudy）和METEOR（Metric for Evaluation of Translation with Explicit Ordering）来评估生成文本与原始文本的相似度。

Rouge-N一般的计算方法为，分母是标准答案中n-gram的总数量，分子是模型生成的摘要和标准答案共有的n-gram的个数，即召回率。在Python中的Rouge模块中，Rouge生成的还有准确率与F1（即*β*=1的情况下的F值）分数。Rouge-L则是参考了最长公共子序列的计算方式，分子总是最长公共子序列的长度。如果我们设*m*为原始文本的长度，*n*是生成文本的长度，那么Rouge-N方法的公式如图13，Rouge-L方法的公式如图14.



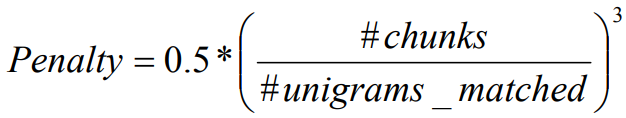
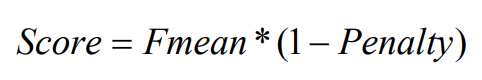
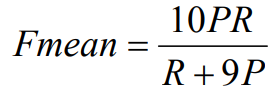
**图13 Rouge-N公式 图14 Rouge-L公式**

BLEU通常被用于机器翻译。它通过比较输出和多个参考文本之间的n-gram重叠来工作。通常我们使用的为BLEU-4，这个指标会考虑1-gram，2-gram，3-gram和4-gram的精度综合计算得到BLEU-4的数值。在计算BLEU时，我们会设置对短句的惩罚因子BP，如果设*r*为参考语料库总长度，*c*为翻译语料库总长度，那么BP的计算方法如图15所示。最后，我们结合BP和先前计算的n-gram精度，可以得到BLEU-4的计算公式，如果我们设置*wn*为n-gram的权重，*pn*为n-gram的取值，我们可以得到如图16的计算公式。



**图15 惩罚因子计算公式 图16 BLEU计算公式**

METEOR也是通常被用于机器翻译的方法。它考虑了同义词匹配、词干匹配以及词序。我们需要先结合准确率P和召回率R计算Fmean，计算公式如图17。之后，我们计算碎片度惩罚。我们记在生成摘要与原始摘要中能够对齐的、且在各自句中连续排列的单词形成一个chunk，#chunk为chunk的数量。而#unigram\_matched则被我们记为生成摘要中能够被匹配的一元词组的数量。惩罚的计算公式如图18。之后，使用如图19的公式即可计算出METEOR分数。

****

**图17 Fmean计算公式 图18 惩罚计算公式 图19 METEOR计算公式**

这些方法各自存在一定的缺点。Rouge方法在评估生成文本时，对于不同的词汇表达可能不是很敏感。BLEU只关注n-gram的匹配度，也因此忽略了语法结构和语义的一致性。METEOR虽然考虑地比较全面，但设计最为复杂。

我们将使用这些指标评估我们的模型。

我们在五折交叉验证中，得到的每次验证的Rouge、BLEU-4与METEOR得分如表1.

**表1 T5五折交叉验证的评估**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 评估指标 | 验证1 | 验证2 | 验证3 | 验证4 | 验证5 |
| Rouge 1-P | 0.633 | 0.623 | 0.623 | 0.593 | 0.670 |
| Rouge 1-R | 0.406 | 0.399 | 0.418 | 0.367 | 0.402 |
| Rouge 1-F1 | 0.462 | 0.457 | 0.470 | 0.426 | 0.474 |
| Rouge 2-P | 0.426 | 0.413 | 0.419 | 0.396 | 0.459 |
| Rouge 2-R | 0.265 | 0.262 | 0.276 | 0.240 | 0.263 |
| Rouge 2-F1 | 0.302 | 0.299 | 0.310 | 0.278 | 0.312 |
| Rouge L-P | 0.608 | 0.599 | 0.599 | 0.571 | 0.647 |
| Rouge L-R | 0.389 | 0.384 | 0.402 | 0.353 | 0.389 |
| Rouge L-F1 | 0.442 | 0.439 | 0.453 | 0.410 | 0.458 |
| BLEU-4 | 0.309 | 0.304 | 0.323 | 0.278 | 0.291 |
| METEOR | 0.368 | 0.361 | 0.380 | 0.331 | 0.365 |

经过比对，我们认为这个五折交叉验证的结果是比较好的。因此，我们将该模型正式投入训练与测试中。

**4.3.8 T5进行实际测试**

我们通过与上述过程类似的方法，用完整的训练集对模型进行了微调。我们最终得到了如图20所示的损失函数曲线，这可以说明我们实际微调的过程是很有效的。



**图20 T5架构微调的损失函数曲线**

最终我们将实际测试得到的摘要文本进行评估，得到了如表2的评估结果。

**表2 T5模型测试的评估**

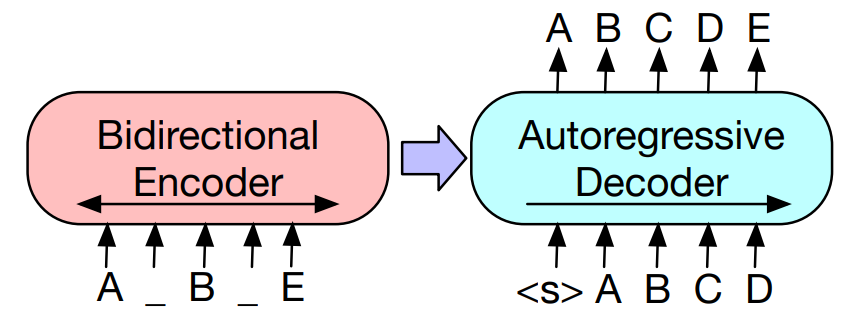
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rouge 1-P | Rouge 1-R | Rouge 1-F1 | Rouge 2-P | Rouge 2-R | Rouge 2-F1 |
| 0.666 | 0.404 | 0.472 | 0.460 | 0.272 | 0.318 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rouge 3-P | Rouge 3-R | Rouge 3-F1 | BLEU-4 | METEOR |
| 0.645 | 0.392 | 0.458 | 0.302 | 0.371 |

这个表现是比较好的，因此我们可以认为T5在文本摘要中可以有不错的表现。

**4.4 BART架构简介**

BART（Bidirectional and Auto-Regressive Transformers）是由Facebook AI Research的研究人员发布的一种架构，在自然语言处理领域有较好的效果。它的特点在于，它的Encoder是双向的，可以同时考虑输入序列的上下文信息，有助于捕获更丰富的语义信息。它的Decoder是自回归的，这意味着在生成输出序列时，模型会逐步生成每个词，将前面的词作为上下文信息，这使得BART在文本生成、文本摘要类任务中有不错的表现。而且，BART也是经过预训练的，只需要我们进行微调就可以得到不错效果的架构。BART的示意图如图21所示。



**图21 BART示意图**

**4.4.1 BART进行实际测试**

对BART架构进行测试非常简单，我们只需要将之前为T5架构编写的代码中的T5Tokenizer改为BartTokenizer，T5ForConditionalGeneration改为BartForConditionalGeneration，代码中的“t5-base”改为“facebook/bart-base”即可。

其余对数据集的预处理，训练背景的设置，大致均与先前的设置一样，不作过多修改。这里有一个地方的修改必须提及，如果我们完全按照T5模型所使用的参数来运行代码，我们意外发现最后无论我们输入的是怎么样的数字序列，最后得到的摘要都是一模一样的。由于最后生成的摘要长度都比较短，我们怀疑是生成摘要的编码长度限制太短考虑到该模型的学习出现了问题，因此我们增加了生成摘要的编码长度为200。经过这样的修改后，我们得到了不错的结果。

BART也是大模型之一，进行五折交叉验证的时间非常久，碍于实验的时间限制，我们没有对模型进行五折交叉验证，而是直接将其投入到实际测试中，观察实验结果。如果其最终实验结果较好，我们可以认为该模型的训练较为成功。

图22展示了BART架构的微调过程中的损失函数曲线。



**图22 BART架构微调的损失函数曲线**

最终我们将实际测试得到的摘要文本进行评估，得到了如表3的评估结果。

**表3 BART模型测试的评估**

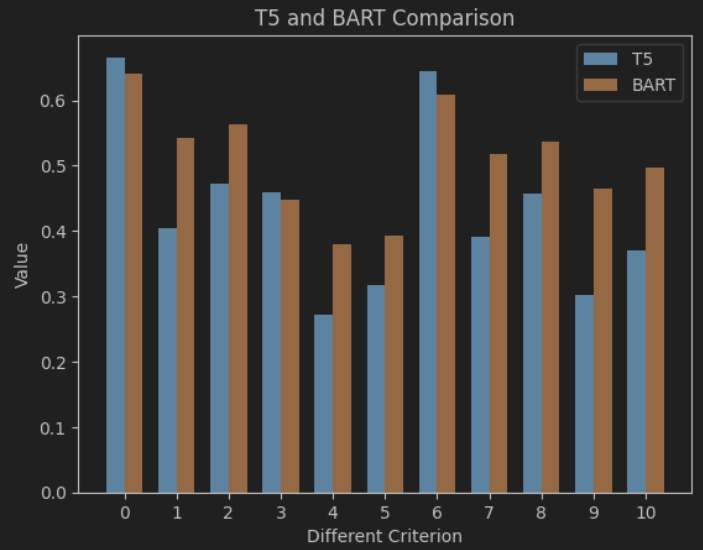
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rouge 1-P | Rouge 1-R | Rouge 1-F1 | Rouge 2-P | Rouge 2-R | Rouge 2-F1 |
| 0.640 | 0.543 | 0.564 | 0.448 | 0.380 | 0.393 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rouge 3-P | Rouge 3-R | Rouge 3-F1 | BLEU-4 | METEOR |
| 0.609 | 0.517 | 0.537 | 0.465 | 0.497 |

**4.5 模型对比**

作为大模型，他们的训练都需要耗费极大的算力与时间成本。即使是租用了具有2块RTX-4090，T5-Base与Bart-base每一次训练3个Epoch都需要耗费30~40分钟的时间。在这一点上，他们并没有过多的区别。

在最终的模型测试中，我们可以发现他们在各个评估指数中有着不同的表现。图23展示了二者的显著对比。



**图23 T5与BART各指标对比**

我们会发现，T5模型在衡量Rouge中的准确度指标时，表现通常略好于BART，但一旦涉及召回率、F1分数、考虑n-gram的BLEU与考虑较为综合的METEOR，BART的表现就会明显优于T5。

如果这不是由于我们对T5的参数设置过差，这可以说明我们所编写的BART在此次实验中，通常意义上的效果最好。

**4.6 难点与解决**

在本次实验中，我们的确遇到过多种问题。

由于我们知道RNN、LSTM等传统模型已经不再流行，因此我们大胆挑战了搭建大模型。但是，大模型的搭建是我们从未涉足过的领域，因此理解大模型的编写逻辑对我们来说是挑战的第一关。本人甚至在尚未完全理解大模型总体结构的基础上，连微调都尚未进行就去尝试对摘要进行生成。但最终，我们查阅大量技术文档与参考文献，找到了一定的思路。

我们此次的代码参考了一个在github中的仓库。

（URL: https://github.com/abhimishra91/transformers-tutorials）

尽管该仓库在本作业完成时（2023年12月26日）已经获得了778个Stars，但在我们试图复现其中的代码的过程中，我们发现代码会出现错误。经过研究报错，我们发现其中一个函数的参数名字是错误的，他们误将自己命名的参数lm\_labels作为了函数中的本应名为labels的参数的名字，这导致代码的运行无法正常运作。目前，本人对该仓库提交了一个Pull Request，并对仓库的拥有者发出了一封邮件，但目前对方尚未有任何回复，仓库仍然停留在十个月前的最后一次更新。

在我们运行Bart-base模型时，我们误以为只需要维持在T5-base中使用的参数就可以获得不错的结果，但仔细观察生成的预测文件，我们发现每一个文本对应的生成摘要居然都是一模一样的数字串，这让我们感到非常沮丧。但我们通过观察这些生成摘要的特征，怀疑我们是将生成摘要的最长编码设置地过短，进而修改了参数，最终得到了比T5-base还要优秀的结果。

1. 总结与分析

Seq2Seq（Sequence-to-Sequence）是一种神经网络架构，用于处理序列到序列的任务。它是自然语言处理（Natural Language Processing）领域目前主流的神经网络架构。不过，传统的结构如RNN，LSTM，GRU因为效果相对较差已经失势，目前的主流是使用了注意力机制（Attention）的Transformer架构。自注意力机制允许Transformer直接对整个序列进行并行处理，提高了计算效率和模型的学习能力。目前最流行的自然语言处理中会使用的架构有BERT, GPT, T5等，他们也是基于Transformer架构的Encoder-Decoder架构。

Rouge、BLEU、METEOR是自然语言处理领域常用的指标。Rouge考虑了生成文本的内容覆盖度，但在评估生成文本时对于不同的词汇表达形式不太敏感。BLEU使用n-gram精确匹配度衡量两个文本的相似性，但忽略了语法结构和语义的一致性。METEOR相对来说综合考虑的内容较多，考虑了近义词和词形变化，但也因此设计较为复杂。

从实验结果来看，BART在通常意义上可能比T5在文本摘要上有着更好的表现，尤其是在召回率，F1分数，BLEU与METEOR上。这可能可以启发我们之后在选择文本摘要的模型时，更多关注BART模型的表现。

大模型的运算非常消耗算力，一台普通的计算机可能对此无能为力。这需要我们寻求更高的算力平台（如租用服务器，置备更多GPU）。大模型的代码编写相较于常见的代码而言，内容更长，函数更复杂，理解与编写有不小的难度，但大模型也是目前人工智能发展的重要趋势，因此我们在大模型上的学习道阻且长。

1. 参考文献与技术文档
2. Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).
3. Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).
4. Banerjee, S., & Lavie, A. (2005, June). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization* (pp. 65-72).
5. NLP机器翻译全景：从基本原理到技术实战全解析（知乎）

URL: https://zhuanlan.zhihu.com/p/665885910

1. abhimishra91 / transformers-tutorials（github Repository）

URL: https://github.com/abhimishra91/transformers-tutorials