**摘要**

# **自然语言处理实践作业三报告**

利用ROC Stories数据实现故事生成（给第一句，生成后面四句）

**马煜 1120210526**

*https://github.com/TT2TER/Story-Gener*

本次作业分为两大部分。第一部分为自己搭建LSTM或Transformer架构网络，手动实现解码算法，并对结果进行评估。第二部分为调用GPT-2等预训练模型，使用不同的方法对模型进行微调，并对结果进行评估。在本次实践中，我发现，手动搭建的网络（LSTM）因**网络结构简单，训练数据简陋**，不能对语言的因果建立正确的感知，仅能生成四句内部逻辑适当的文字。而预训练大语言模型（GPT-2）因其**高质量的预训练方法和数据**，在微调之后能够完成故事生成任务。我将在本报告中对**建模过程**、**实验方案**和**实验结果**进行具体阐述。

1. **问题重述和背景介绍**

故事生成任务是自然语言处理中文本生成任务的一个子分支（文本生成任务是“文本到文本”的生成过程，即以文本作为输入，并将输入文本进行语义表示，最后生成期望输入的过程，也即Seq2Seq任务。机器翻译、摘要生成、问题回答和对话系统都是常见的文本生成任务），旨在探讨神经网络对句间关系的理解以及常识性知识的储备，可以促进带有时序特征的生成式模型的发展。

本次实验使用ROC Stories数据集[1]，这是一个为了解决常识知识的表示和学习的数据集。这个问题对于理解事件之间的偶然关系和相关关系尤其具有挑战性。虽然这个话题在 NLP 界引起了很大的兴趣，但由于缺乏适当的评估框架，研究受到了阻碍。此数据库试图通过一个评估故事理解和剧本学习的新框架来解决这个问题。当然本次实验只使用数据，并未针对其提出的任务进行训练。此数据集创建了一个包含约 5 万个五句常识故事的新语料库 ROC Stories，以实现相关评估。该语料库有两个独特之处：（1）它捕获了日常事件之间丰富的因果关系和时间常识关系。（2）它是日常生活故事的高质量集合，也可用于故事生成。根据其官方表述，许多基于浅层语言理解的基线和最先进的模型都很难在故事完形填空测试中获得高分。

传统的故事生成方法为结构化故事生成[2]，该工作将故事生成拆成了若干任务，逐个击破。简单说，就是根据提示条件，先生成谓词和参数序列。然后，使用占位符实体（如ent0）生成一个故事。最后，用特定的引用替换占位符，生成故事。

在本实验中，我将使用最简单的LSTM [3]模型和基于transformer [4]架构的预训练模型GPT-2 [5]实现给定一句话，生成后面四句话最终形成一个完整的五句话故事。

LSTM模型全称为Long short-term memory(长短期记忆)，是一种特殊的RNN，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。现在机器学习领域被广泛使用。简单来说，LSTM相比普通的RNN，能够在更长的序列中有更好的表现，它能通过门的控制保留很久之前的特征，这是它最大的特点。

GPT-2 是 OpenAI 在 2019 年 2 月创建的一种基于 Transformer 的无监督深度学习语言模型，其目的只有一个，就是预测句子中的下一个单词。模型开源，在超过 15 亿个参数上进行训练。奠定了Decoder-only范式在大语言模型上的霸主地位。

1. **评价指标的选取和建立**

针对本次文本生成任务，我选择了两种传统的评价指标，BLEU [6]和ROUGE [7]。这两种指标最初分别被用来评价机器翻译任务和文本摘要任务。

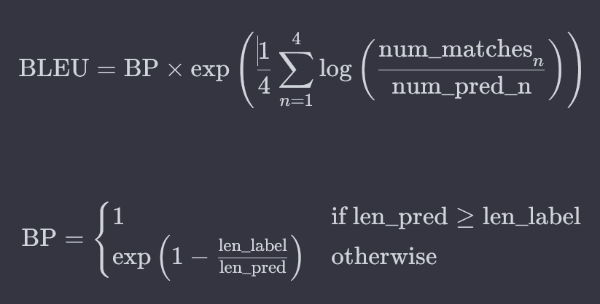
 BLEU的计算基于n-gram的精确匹配。它将生成的文本中的n-gram与参考文本中的n-gram进行比较，然后计算匹配的数量。BLEU还考虑了n-gram的长度惩罚，以防止较短的生成文本在评价中过于受到偏见。最后，通过将匹配的n-gram数标准化为生成文本的总n-gram数，得到最终的BLEU分数。BLEU指标简单优雅，能很好的根据生成内容的长短和短语重复程度量化生成质量，可以直接迁移到5句话故事生成任务中。本次使用BLEU4作为我们的BLEU分数。

图 1 BLEU公式

ROUGE关注生成的文本中包含的信息与参考文本的重叠程度，即它会查看有多少个参考译句中的n元词组出现在了输出之中。ROUGE-N将BLEU的精确率优化为召回率，而ROUGE-L 将BLEU的n-gram优化为公共子序列。为了与BLEU分数做更明显区分，本文使用更常用的ROUGE-L-f作为ROUGE分数。



图 2 ROUGE-L计算公式

当然，针对生成式的语言模型，还有其他新式的评价指标，比如UNION [8]，一种可学习的无参考指标，用于评估开放式故事生成，它可以在没有任何参考的情况下衡量生成故事的质量。 UNION 建立在 BERT 之上，经过训练可以区分人类编写的故事和负面样本，并恢复负面故事中的扰动。这项工作提出了一种通过模仿现有 NLG 模型中常见错误（包括重复图、逻辑冲突和长期不连贯）来构建负样本的方法。对两个故事数据集的实验表明，UNION 是评估生成故事质量的可靠方法，它与人类判断更好地相关，并且比现有的最先进的指标更具有普遍性。

由于时间关系以及深度学习框架选择（UNION为tenserflow框架），本次实验暂时未采用UNION评估办法。而**手写实现**了BLEU指标，同时**调库实现**了ROUGE指标。这部分代码见eval.p。

**补充说明：**

1. 在最后的实验中，我发现BLEU和ROUGE只能评估文本内容上的重合率，和故事逻辑无任何关联，即只评价内容有无，不评价好坏。为了评估生成的文本的质量，我额外对部分评估生成结果按照以下标准进行了人工盲打分（human evaluation）：

0分：**不能生成句子**（生成乱码）， 所有句子的句间句内均无逻辑；

1分：**能够生成句子**（句子数量不 一定5句话），**某些句子句内逻辑 通顺**（能部分说人话）；

2分：能够生成句子（不一定5句 话），**所有句子句内逻辑通顺**，语 法正确；

3分：能够生成5句话故事，所有 句子句内逻辑通顺，语法准确，**相 邻两句句间逻辑正确**。

4分：能够生成5句话故事，所有 句子句内逻辑通顺，语法准确，**5 句话居间逻辑为一个统一整体**；

5分：在4分基础上**故事精彩有细 节**，此项为加分项。

1. 此外，在**训练和验证**（train and valid）时，我提出了一个量化训练效果的指标，即Accuracy。计算公式如下：

可以在除loss之外为句子的训 练准确率提供参考。

1. **第一部分：LSTM**
   1. **建模过程[[1]](#footnote-1)**

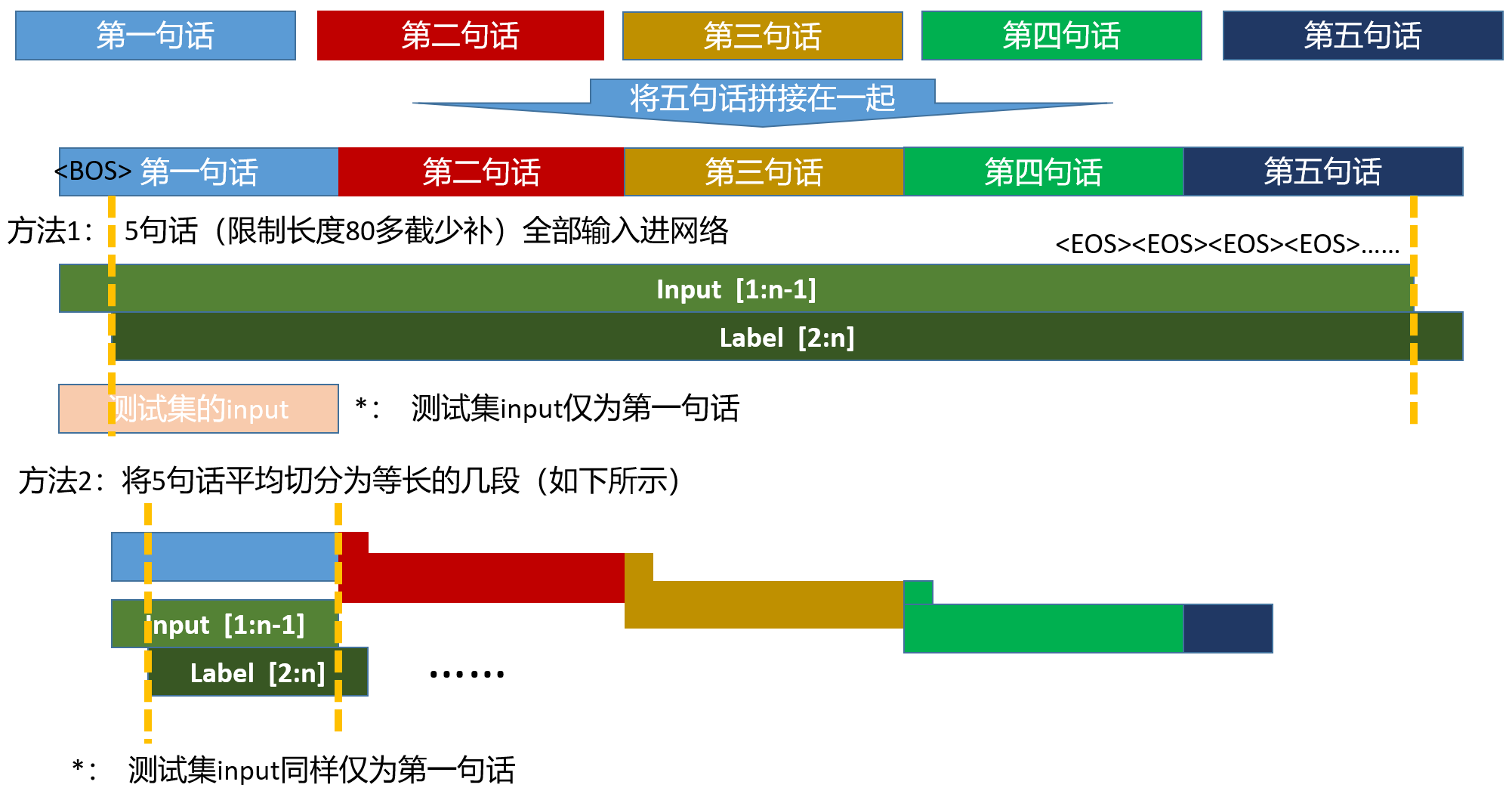


图 3 数据处理和切分方法示意图

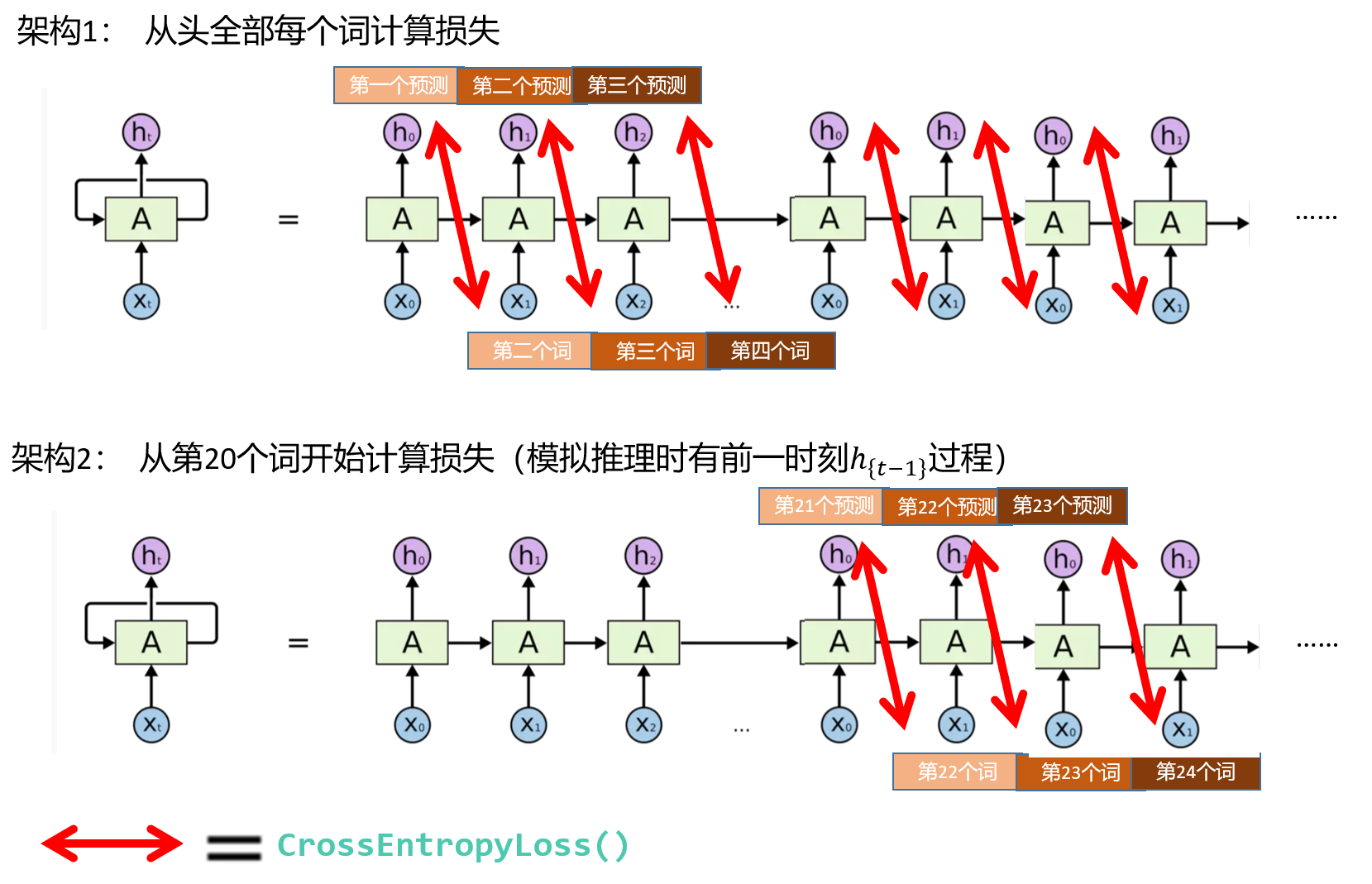


图 4 LSTM两种训练方式示意图

**3.1.1 数据处理**

首先介绍一下数据集的处理方式，在建模阶段，针对训练集和验证集我将5句话连在一起，只在第一句前加特殊字符<BOS>，在尾部少补全多截取，使用<EOS>作为输出结束标志以及padding标志（即不使用<PAD>）。

前期设计了两种数据切分方式。**第一种：**输入数据为第1至n-1个词（n为句子长度），ground-truth标签为第2至n个词。**第二种：**在第一种的基础上将每个故事平均切分为4份。每份依旧按照错位对其的方式进行处理（目的是为了减轻一次训练时网络复杂度）。两种方式的测试集输入均处理为第一句话。数据处理及切分示意如图3所示。

此外，针对embedding时使用的词向量，根据第二次作业的经验来看，加载预训练好的词向量对网络无任何明显提升（可通过增加训练轮数将效果追回），因此本次实验未采用现有词向量进行初始化。

**3.1.2 网络设计**

在网络架构设计方面，使用单向LSTM作为类似Decoder-Only的解码器，设计测试了词向量维度为100，300；LSTM隐藏层数量128，512；网络堆叠层数1，4。这几种不同情况的LSTM。

对于训练方式，根据LSTM网络特点和任务需求，设计了两种不同的训练损失计算方法。如图4所示。第一种方法，直接从第一个时刻的输出开始计算每一时刻的交叉熵损失。第二种方法，从第20个时刻的输出才开始计算交叉熵损失（为了模拟输入第一句生成后四句时有第一句的隐藏状态）。

在推理部分，我不能像训练一样批量处理输入文本，而是需要一个时刻一个时刻取下一时刻的预测，并用解码算法选择最合适的预测结果添加到输入序列中，如此往复循环。

**3.1.3手动实现不同解码算法**

在训练时求交叉熵损失时使用的是Greedy方法，而推理时则依次实现了按照概率采样（sample）、贪心算法（Greedy）、Beam-search、Temperat、Top-p、Top-k等6种解码方法。

下面对它们的实现细节进行简单介绍。

首先介绍Greedy算法，就是将下一时刻输出结果进行softmax求得概率，并直接选取概率最大的一个作为预测结果。

Sample算法则是在softmax后依照概率的可能性大小进行采样。

Temperature算法将softmax结果重新在[0,1]之间按照温度系数进行直方图排列，能够使原本差异较大的输出概率的差异缩小，使网络的多样性更好。

Beam-search算法在实现上维护了一个beam\_list = [(data\_num, 1)]队列，用来保存不同的输出和对应的概率。同时，根据我的实验尝试，在softmax过程中加入温度系数并设置为2能够提升算法的多样性，在上述参数下，beam=10时能生成较好的文本。

Top-K和Top-P算法思想其实是相近的，在实现上top-k在softmax概率分布中选择前K个最高概率的词汇，然后将其他概率置为零。而top-p通过指定一个累积概率阈值p，然后在概率分布中选择概率累积达到p的最小的词汇集合。

具体代码参见LSTM.py中predict()函数。

**3.2实验过程和结果**

实验环境为NVIDIA RTX 2080Ti 魔改22G显存版本， pytorch版本为2.1.2+cu118，系统为wsl2下的Ubuntu22.04（GPT-2实验环境相同）。

在正是实验之前，我对比了**不同的学习率调度器**的差别，分别尝试了StepLR（每特定步数后下降），MultiStepLR（指定几个步骤后下降），CosineAnnealingLR（余弦函数式周期上升下降），ReduceLROnPlateau（检测某项指标变化，不变化时再下降），get\_linear\_schedule\_with\_warmup（先上升再线性下降到0），对比认为ReduceLROnPlateau非常适合在这个数据集下的LSTM模型，因为其能检测当loss不下降时自动降低学习率，具有相当好的自适应能力，能够帮助我在有限的epoch下更多的学习到知识。

实验首先比较了**两种切分方式**（图3）对训练效果的影响。词向量128维，隐藏层数量128情况下，网络层数同样为1层，训练90个epoch。Bach\_size均设置为吃满显存状态（分别为256和1024）。lr初始为3e-3，使用adam优化器，学习率调度器使用ReduceLROnPlateau监测val部分的loss下降情况，每次调整下降到原来的0.3倍。90个epoch的实验结果如表1所示。由结果可以看到**一次送入整个故事对训练效果有显著提升**。此后所有测试（包括GPT-2）均采用方法1。

表格 1 不同数据切分方法对比实验结果

|  |  |  |
| --- | --- | --- |
| 不同数据切分 | 方法1 | 方法2 |
| 验证集损失 | **2.4951** | 4.6336 |
| Accuracy | **63.224 %** | 32.656 % |

随后，在其他条件不变的情况下，对比测试了**两种训练损失计算方法**（图4）对训练结果的影响，由于方法2每次训练获得信息较方法1少，因此方法2训练轮数增加到110 epoch。方法1保持90不变。结果如表2所示。结果显示采用方法1能更有效的训练网络（我认为ROUGE如此贴近是因为实际上两种网络学习到的内容是几乎一样的，在内容上都是重复相同的话，导致分数相近）。尽管方法2的设计思想是更贴近推理实际，但考虑到网络的自回归结构，这种方法设计并不能提升训练效果，而会因每次训练学习到知识过少而降低效果。因此在之后的实验中均采用方法1进行实验。

表格 2 不同损失计算方法对比

|  |  |  |
| --- | --- | --- |
| 不同损失计算 | 方法1（90epoch） | 方法2  (110epoch) |
| 验证集损失 | **2.4951** | 3.1931 |
| Accuracy | **63.224 %** | 54.284 % |
| GreedyBLEU | **0.076** | 0.067 |
| GreedyROUGE | 0.4020 | **0.4029** |

讨论完数据处理和训练方法后，采用最优方法进行测试比较，讨论**LSTM网络架构**对训练结果的影响：

**网络层数：**通过对比1层和4层的训练10epoch时的loss（2.927和3.327），我认为增加网络层数不能提升训练速度。而出于训练集客观较小因素影响，单纯的堆叠并不能使网络学习到更多有效的信息。

**词向量维度和隐藏层大小：**类似网络层数的实验，使用10epoch对比（128,128），（128,512），发现epoch相同时大隐藏层学习速率更快，对比（128,512），（300,512），在相同epoch下，大的词向量同样能获得更好的训练效果。

最终使用（300,512）方案，训练200个Epoch后结果如表3所示。

表格 3 LSTM模型最终训练效果

|  |  |
| --- | --- |
| 1层300词向量512隐藏层 | 200 epoch |
| 验证集损失 | **2.8613** |
| Accuracy | **55.426 %** |

* **特别说明**，可以看到这里的数据不如不同损失方式对比时的结果，原因如下：使用词表规模变化，在这两个实验之间我修改了代码，将测试集单词也加入了词表中，导致词表规模扩大，数据上效果下降。但实际测试是有增强的。

使用最终训练好的模型对不同的解码方式进行结果生成。在不断的测试之后，我努力找到语法准确性和结果多样性的平衡点，并在一些算法的求概率部分加入了温度参数（在前文算法介绍已经提及，温度均为2）。Beam=10，k=10，p=0.9。结果如表4所示。可以看到，实际上LSTM在传统评价指标上表现的并不好，而较高的ROUGE分数则是因为其重复出现的内容过多。人类评价实际观感上，如图5所示，各种方法生成的内容大部分在2-3分水平，少部分为1分或0分。在句间逻辑上体现出了力不从心。可以说LSTM部分在最终效果上是失败的。

表格 4 不同解码方法的LSTM比较

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 不同解码算法 | Sample | Greedy | Beam | Temperature | Top-k | Top-p |
| BLEU | **0.057** | **0.075** | 0.069 | 0.067 | 0.066 | 0.068 |
| ROUGE | **0.3284** | 0.4178 | **0.4301** | 0.3609 | 0.4122 | 0.3587 |

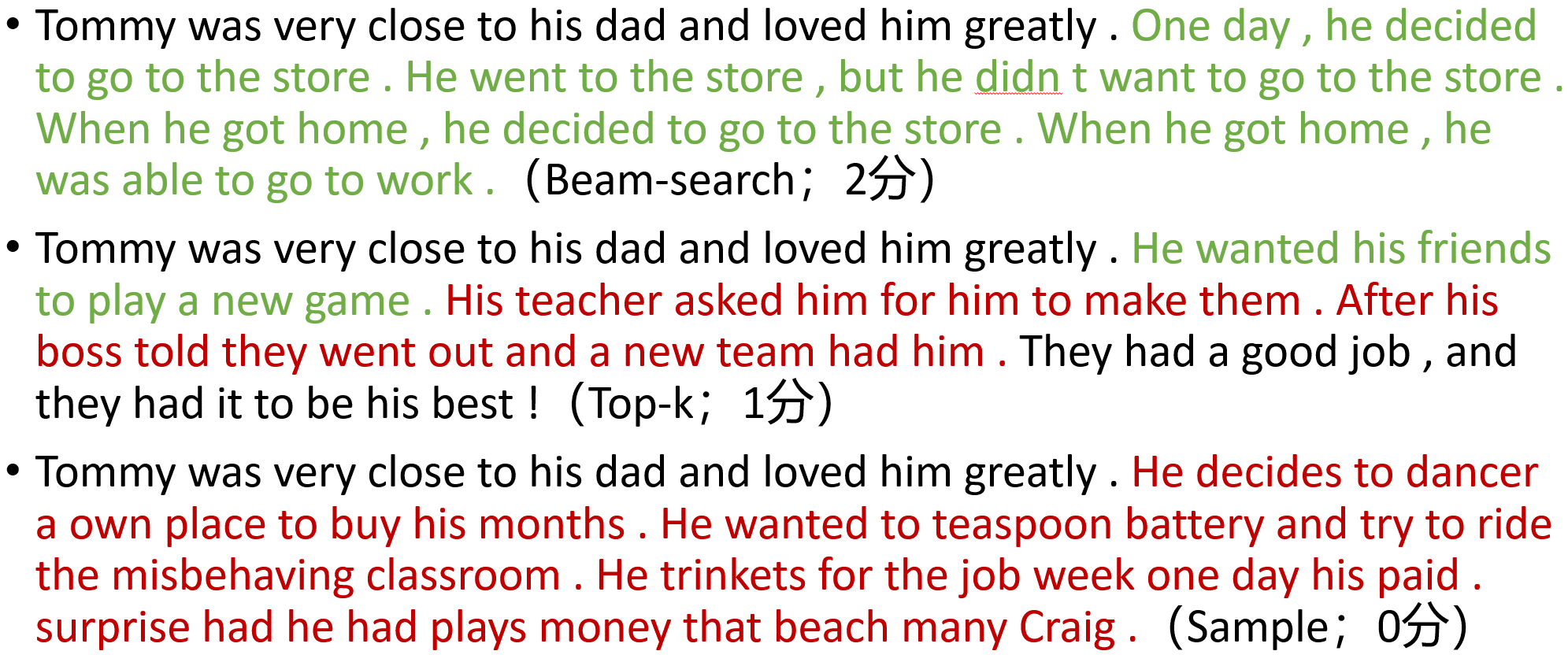


图 5 LSTM模型生成内容评估结果示例

绿色为句内语法和逻辑正确，红色为胡话。

1. **第二部分：GPT-2**
   1. **建模过程**
      1. **预训练模型选取**

LLaMA模型是真正意义上的大语言模型，其最小的模型LLaMA-7B也无法加载到显卡显存中（根据搜集到的信息，27G大小的预训练模型，需要设置为float16才能装在到16G显存的显卡中）。因此，本次实验选用GPT-2 small。如图6所示。其堆叠了12层transformer decoder，参数量来到了1.2B，虽然不能达到广义的大语言模型的标准，但其可训练参数依旧非常可观。

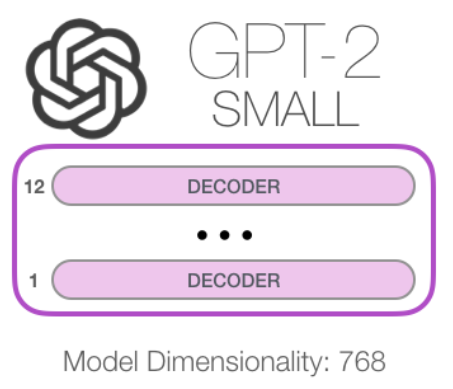


图 6 GPT-2-small示意图

* + 1. **数据处理**

GPT-2的数据处理方法和LSTM非常相似，根据LSTM的实验结果，采用**第一种**数据切分方式，即将四句话连在一起后不再进一步细分，直接一起送入训练。其余差别只是在技术实现上直接调用了GPT-2的tokenizer。

* + 1. **不同微调方法的实现介绍**

本次实践中分别实现了9种不同的有效微调方式，分别是：全参数微调、冻住部分参数（前8层）微调、逐层解冻微调、梯度累积微调、Lora方法、adaLora方法、Prefix tuning、Prompt tuning、P-tuning。其中前四种方法为**手动实现**，后五中方法为**调用PEFT库**实现。

下面将分别介绍每种方法实现的重点部分。

优化器：所有方法在加载优化器时需要筛选出可微调参数。

全微调：得益于魔改到22G的2080Ti，我能够完成全参数微调。将预训练模型加载后将所有参数设置为可训练，输出和直接使用GPT-2的输出和loss计算。可微调参数量为（124439808/124439808）。

冻住部分参数微调：在加载模型后逐层遍历模型，将前8层decoder冻结，只训练后4层。在初始化优化器时只将可微调参数加载。（67736832/124439808）

逐层解冻微调：起初同冻住部分参数微调一样，冻住8层decoder，而在最后8个epoch内在训练过程中逐层将参数解冻并重新初始化优化器。此方法的学习率调度器为手动调度，每解冻两层学习率降为原来的0.3倍。

可微调参数变化：67736832->74824704->81912576->89000448->96088320->103176192->110264064->117351936->124439808。

梯度累积微调：这个方法的目的是小的batch\_size模拟大的batch\_size的梯度。可微调参数为（124439808/124439808），在每256个batch都更新一次学习率，学习率调度器使用get linear schedule with warmup，warmup设置为200，因大的batch\_size梯度下降方向较为合理，因此将学习率逐步调大，累积的batch\_size等效为2048。

Lora方法：引入低秩矩阵适应参数，能够有效地利用大规模预训练模型的知识，同时在特定任务上进行微调。调用PEFT库，非常简便的实现了微调算法。其余训练代码不变。（294912/ 124734720）

adaLora：根据权重矩阵的重要性得分自适应地分配参数。(442512/ 124882332)

Prompt tuning和P-tuning理念很相似，通过在模型输入中引入任务特定的前缀，以引导模型产生更符合任务需求的输出。在验证集部分，需要将前缀长度截取掉。Prompt指令为：“give one sentence to start the five sentence story:”可微调参数分别为(6144/124445952)和(229376/ 124669184)

Prefix tuning：在模型输入前添加一个连续且任务特定的向量序列，只优化这个向量序列。(368640/ 124808448)

具体代码参见model.py中GPT-2类以及gpt2.py中train()函数。

* + 1. **解码算法介绍**

在解码部分，根据LSTM部分的实验经验结果，设计了三种解码算法：贪心搜索、带温度的top\_k+top\_p、带温度的beam search。均直接调用GPT-2的.generate()方法实现

具体代码参见gpt2.py中predict()函数。

**4.2 实验过程和结果**

实验环境同LSTM。

首先讨论lr优化器的影响。在本部分，因为GPT-2模型过于庞大，微调一个epoch也需要将近10分钟，因此除梯度累积微调和逐层解冻微调之外，我全部选用MultiStepLR指定在第2，3,5,8个epoch时将学习率下降到原来的0.3倍。同时，根据测试，在全微调方法下10个epoch时学习率已经下降到较低水平。

在此基础上，解码算法的温度设置为1，k=50，p=0.9，beam=10。

得到不同微调方式的比较如表5所示。BLUE和ROUGE分数为三种解码方式的平均。Human为本人按照提出的五档评分制对每种微调方法的打分平均（每种解码方式10条，每种微调三种解码方式，共30条）

从结果中可以肯定的是，所有微调方法相比于zoro-shot都有较大的提升，即所有微调方法都有效。

根据图7的趋势图可以看出，人类打分的变化和BLEU指标的变化趋势比较接近。从打分结果可以总结得到，全微调研究是最均衡最有效的微调方法之一，而梯度累积因为其微batch\_size等效较大效果也非常好。而逐层解冻和直接冻住部分微调因可调参数直接减半，且无有效的优化方法（如LORA），虽然可调参数量较多，但效果依旧远不如Lora系列。P tuning和Prompt tuning两种方法虽然可调22万和6千，但在最终生成效果上可谓惊艳。而Prefix方法在各项指标均达到了相对最差水平（虽然依旧比逐层解冻和冻住部分参数微调效果好）。

表格 5 不同微调和解码方法的评分比较（可调参数量从左到右由大到小，绿色为该项最佳，蓝色为次佳，红色为最差）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 不同微调方法 | 全微调 | 梯度累积 | 逐层解冻 | 冻住部分 | AdaLora | Prefix | Lora | P | Prompt | Zero-shot |
| Greedy-BLEU | 0.307 | 0.314 | 0.315 | 0.310 | 0.321 | 0.221 | 0.324 | 0.253 | 0.294 | 0.151 |
| Greedy-ROUGE | 0.399 | 0.394 | 0.389 | 0.393 | 0.397 | 0.387 | 0.405 | 0.407 | 0.408 | 0.293 |
| TOP-k-p-BLEU | 0.331 | 0.327 | 0.318 | 0.285 | 0.294 | 0.246 | 0.296 | 0.273 | 0.295 | 0.147 |
| TOP-k-p -ROUGE | 0.354 | 0.383 | 0.369 | 0.345 | 0.349 | 0.356 | 0.358 | 0.379 | 0.368 | 0.287 |
| Beam-BLEU | 0.336 | 0.297 | 0.308 | 0.290 | 0.319 | 0.276 | 0.327 | 0.293 | 0.274 | 0.191 |
| Beam-ROUGE | 0.404 | 0.399 | 0.371 | 0.362 | 0.383 | 0.369 | 0.395 | 0.412 | 0.398 | 0.309 |
| BLEU | 0.325 | 0.313 | 0.314 | 0.295 | 0.311 | 0.248 | 0.316 | 0.273 | 0.288 | 0.163 |
| ROUGE | 0.386 | 0.392 | 0.376 | 0.367 | 0.376 | 0.371 | 0.386 | 0.399 | 0.391 | 0.296 |
| HUMAN | 2.900 | 3.733 | 2.133 | 2.266 | 3.733 | 2.466 | 3.466 | 3.566 | 3.533 | 0.2 |

图 7 不同微调方法评分趋势图

以上现象能**总结出以下结论**：

1. 在有条件的情况下请尽量保障全参数微调，同时在一定限度内增加batch\_size（或者用梯度累积方法模拟大batch\_size情况）。
2. 在没有条件的情况下，lora系列微调效果最好，极大降低微调参数量的同时保证了微调质量。
3. 在文本生成任务中，Prompt微调和P微调能起到较好的效果，尤其是在条件极差的情况下，良好的Prompt微调能实现媲美、超越全参数微调的效果。
4. Prefix方法效果较差，可能不适合文本生成类任务的微调。
5. Zero-shot效果较差的主要原因是GPT预训练的任务为对话，与故事生成有较大偏差。
6. 现有的机器翻译评价指标对生成类任务评价有些许偏差，但也在可接受范围内，能够初步筛选出好模型与差模型。同时BLEU参数甚至能体现出模型的相对好坏。

在本实验基础上，选用人类评价相对较好的AdaLora模型（在主观观感上lAdaora模型的多样性比梯度累积更好），改变解码算法的temperature参数（改为0.7和2.0），与1做比较。结果如表格6。从结果比较可以看出来，温度的改变对Beam-search方法影响

表格 6 温度对GPT-2生成结果影响比较

|  |  |  |  |
| --- | --- | --- | --- |
| 不同温度 | 1 | 0.7 | 2.0 |
| TOP-k-p-BLEU | 0.294 | 0.336 | 0.191 |
| TOP-k-p -ROUGE | 0.349 | 0.398 | 0.290 |
| Beam-BLEU | 0.319 | 0.291 | 0.287 |
| Beam-ROUGE | 0.383 | 0.367 | 0.365 |
| HUMAN（avg） | 3.7 | 3.95 | 3.5 |
| TOP-k-p- HUMAN | 3.2 | 3.5 | 3.2 |
| Beam- HUMAN | 4.2 | 4.4 | 3.8 |

不大，而对Top-k+Top-p组合影响较大。我推测的解释为：beam-search方法因选取的是多个时间步概率的最大累积，所以只要概率的主次大小关系没有改变，在sample的时候选择的概率变动也不会太大。而Top-k+Top-p组合为单个时间步的概率选择，改变概率的平滑程度会更影响选择的结果。

对温度改变的生成效果比较来看，更尖锐的分布（温度0.7），会减小模型的多样性选择，这样模型会更倾向于说人话，说训练集见过的话。而温度过高会导致结果偏离训练集，说胡话，或者回答结构与微调任务不一致。

1. **其他结论及总结**

从以上各个实验可以总结得出的结论有，模型的架构对模型的影响有，但不如训练数据的质量影响大。本次GPT微调用的数据在有先验语言知识的GPT上效果较好，而在无先验知识的LSTM模型上展现出了重复等行为。因此，大语言模型的涌现在LLM中是比较关键的一个特性。

此外在实验过程中，对实验关键数据的记录我这次做的不是特别的好，log文件混乱，勉强能够通过回忆辨别，有的数据需要写报告的时候再次实验再次确认，希望下次有类似实验时把log文件维护的更好一点。

最后，感谢鉴萍老师这学期的三次大作业，让我对各种结构的网络有了实践上的认知！

* 生成的Log文件和结果展示可以在output文件夹中找到

**参考文献**

[1] N. Mostafazadeh et al., “A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories.” arXiv, Apr. 06, 2016. doi: 10.48550/arXiv.1604.01696.

[2] A. Fan, M. Lewis, and Y. Dauphin, “Strategies for Structuring Story Generation.” arXiv, Jun. 15, 2019. Accessed: Dec. 29, 2023. [Online]. Available: <http://arxiv.org/abs/1902.01109>

[3] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo, “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.” arXiv, Sep. 19, 2015. Accessed: Dec. 29, 2023. [Online]. Available: http://arxiv.org/abs/1506.04214

[4] A. Vaswani et al., “Attention Is All You Need.” arXiv, Aug. 01, 2023. Accessed: Nov. 16, 2023. [Online]. Available: <http://arxiv.org/abs/1706.03762>

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners”.

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a method for automatic evaluation of machine translation,” in Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2001, p. 311. doi: 10.3115/1073083.1073135.

[7] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries”.

[8] J. Guan and M. Huang, “UNION: An Unreferenced Metric for Evaluating Open-ended Story Generation.” arXiv, Sep. 16, 2020. doi: 10.48550/arXiv.2009.07602.

1. LSTM部分从构建dataset和data loader开始至训练推理解码代码除了必要部分调库外，其余部分全部参照实践作业2框架手写实现 [↑](#footnote-ref-1)