哈尔滨工业大学

**自然语言处理**

**实验报告**

|  |  |
| --- | --- |
| **姓名：** | **刘博飞** |
| **学号：** |  |
| **学院：** | **未来技术学院** |
| **日期：** | **2024年11月22日** |
| **教师：** | **车万翔** |

实验1：从零开始实现基于Transformer的语言模型

## 一、实验过程及结果

1.实验环境配置

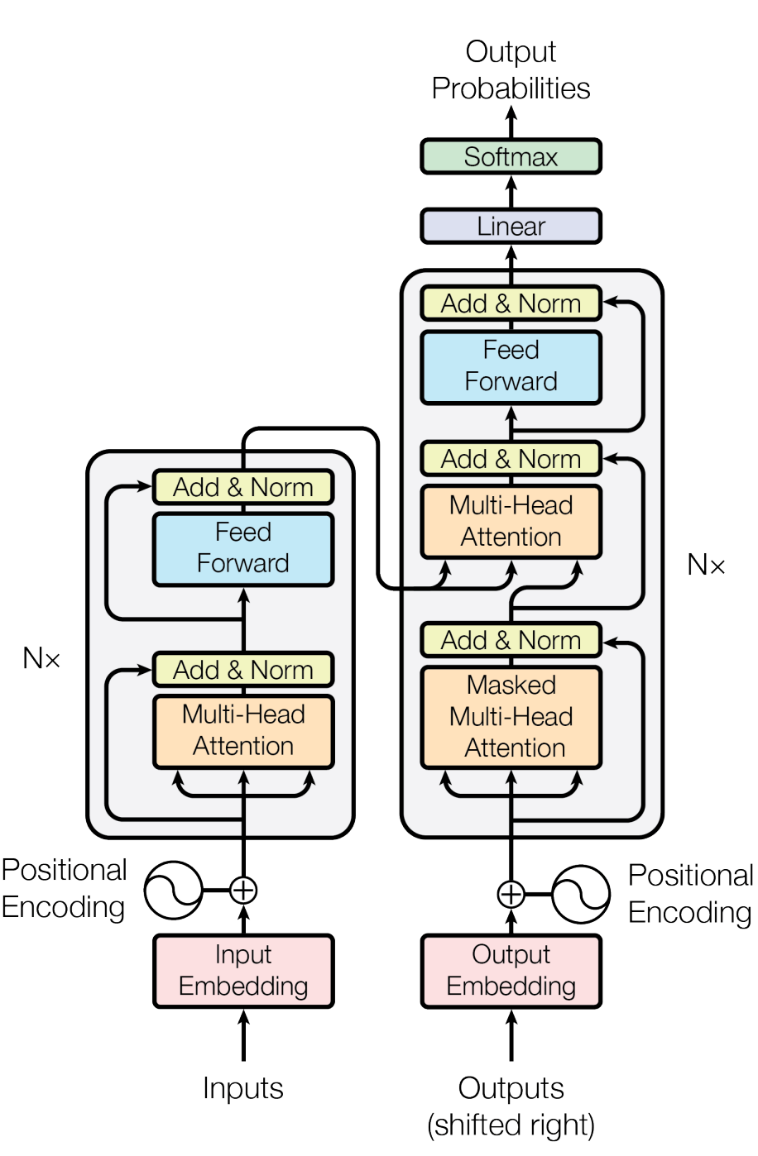
本实验要求实现一个基于Transformer架构的语言模型，首先需要按照requirements.txt配置实验环境，本人配置的实验环境主要安装包如下所示：

python 3.10.15 h4607a30\_1

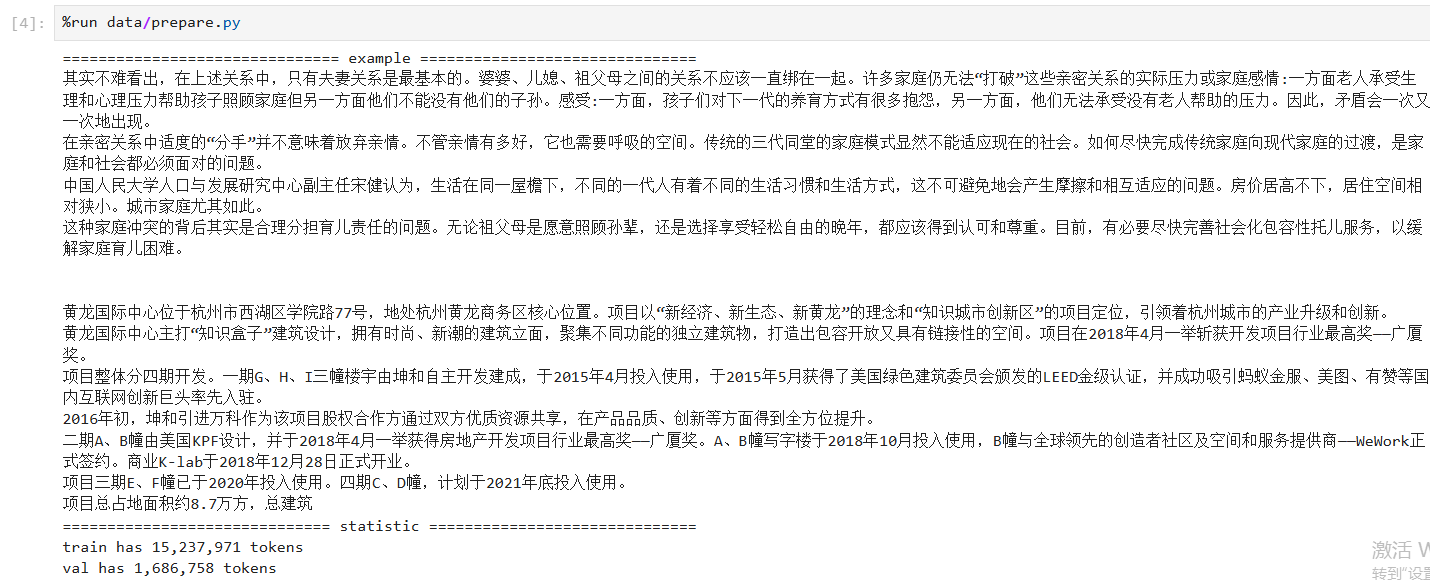
pytorch 1.13.1 py3.10\_cuda11.7\_cudnn8\_0

numpy 2.1.3 pypi\_0 pypi

2.字词划分



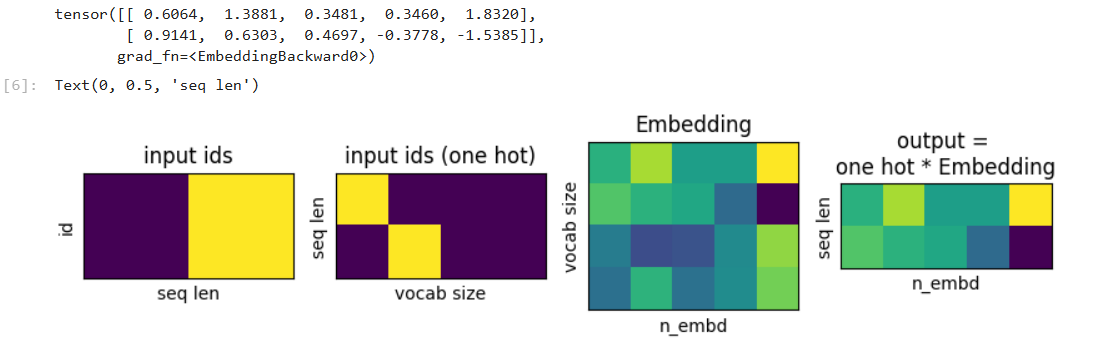
根据transformer的decoder部分的框架结构，首先需要对输入文本进行tokenize,即根据词表进行字词划分，顺便将划分好的字词串分成训练集和测试集。运行示例如下：



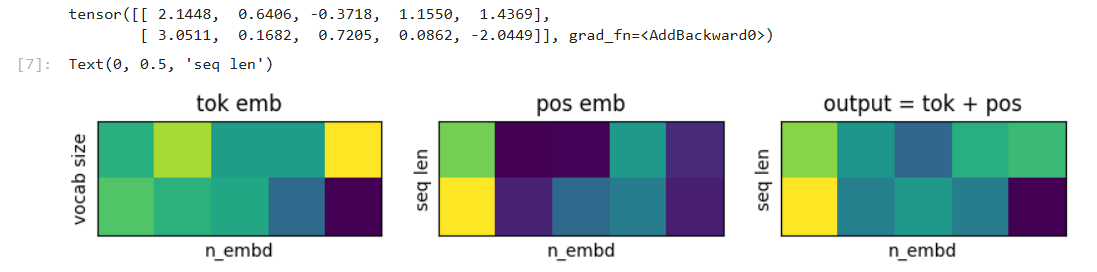
3.输入embedding

接下来根据decoder结构，需要对各个token进行embedding，即将其投影到n\_embd维度的隐空间中，以表征每个token的单独语义以及上下文语义。

Embedding过程包括token embedding和position encoding，其中，token embedding的运行逻辑为将输入token对于词表的独热编码矩阵乘以词表的embedding矩阵，实际上就是查找并取出词表中包含的输入序列的embedding并整合在一起，程序运行结果示意如下：



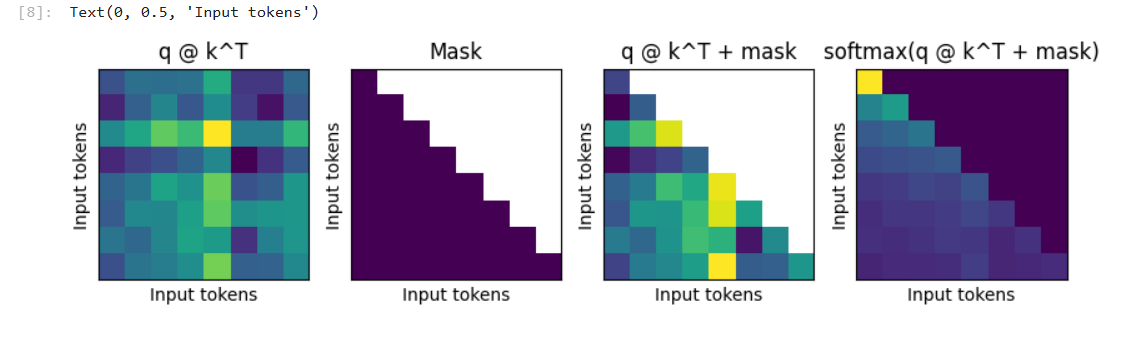
Position encoding的逻辑与embedding类似只不过查找字典的大小为输入序列的长度。Embedding最后的输出就是token embedding与position encoding的和，程序示例如下：



4.多头自注意力机制的实现

这一部分是GPT模型的核心，本实验中主要关注的是自注意力掩码的设置，由于模型是通过已经读入的序列来预测下一个位置的token，因此预测某一个位置的token时，只能应用该token之前所有位置的信息。因此我们需要对自注意力分数矩阵进行掩蔽，使得某一token对于其之后token的注意力分数为0.

具体实现时，由于需要用到softmax函数，因此在q@k矩阵的基础上，加上值为负无穷的上三角矩阵即可，实现效果如下：



5.各个模块的实现、拼接与整合

这一部分为组装模型，这里简要说明一下多头注意力的实现以及各个模块的包含关系。

多头注意力的实现主要在于q,k,v矩阵及其运算结果矩阵的形状变换。为了实现“多头”，需要将q,k,v矩阵的形状从[B, T, n\_embd]切分为[B, n\_head, T, head\_dim]，运算时，每个头下辖的embedding单独进行运算，最后再整合到一起。具体来说q@k后的矩阵形状为[B, n\_head, T, T]，再@v后的形状为[B, n\_head, T, head\_dim]，最后将多个头拼接成一个维度，形状变为[B, T, n\_embd]。

Decoder block一共包含如下模块：

self.LayerNorm1 = nn.LayerNorm(config.n\_embd)

self.LayerNorm2 = nn.LayerNorm(config.n\_embd)

self.MHSelfAtt = MultiHeadSelfAttention(config)

self.FFN = FFN(config)

整个GPT模型包含如下模块：

self.transformer = nn.ModuleDict(

dict(

wte=nn.Embedding(config.vocab\_size, config.n\_embd),

wpe = nn.Embedding(config.block\_size, config.n\_embd),

dropout = nn.Dropout(config.dropout),

blocks = nn.ModuleList([DecoderBlock(config) for \_ in range(config.n\_layer)]),

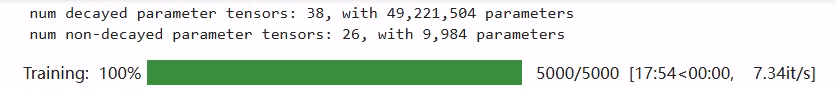
ln = nn.LayerNorm(config.n\_embd),

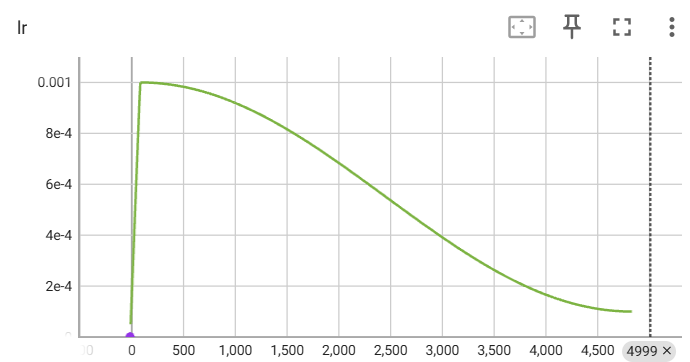
)

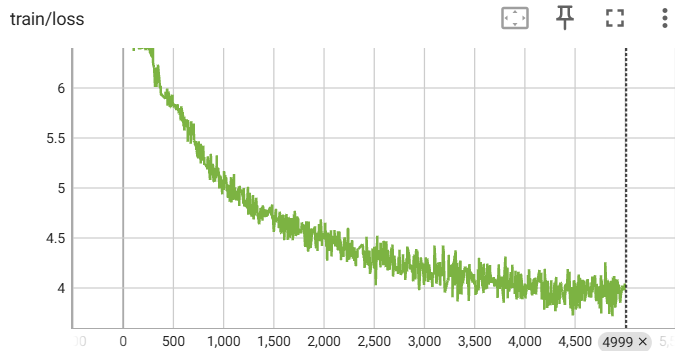
)

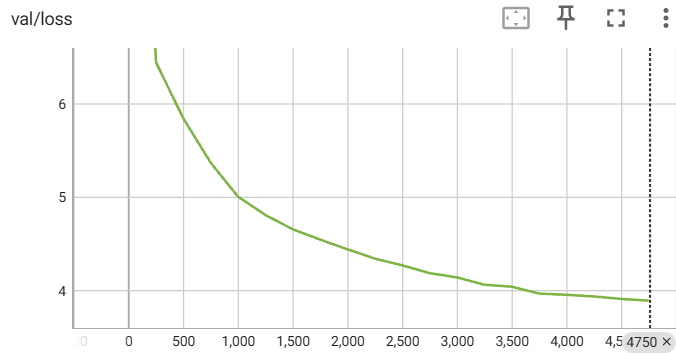
self.lm\_head = nn.Linear(config.n\_embd, config.vocab\_size, bias=False)

6.模型训练

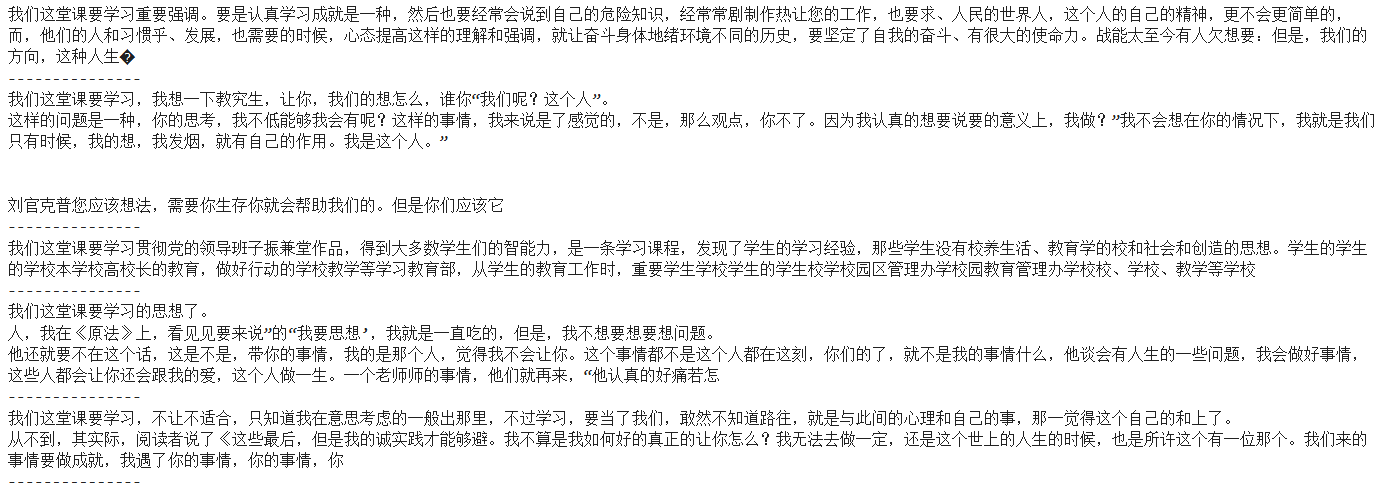








7.模型测试



## 二、实验心得

了解了GPT模型的组织架构模式，对于模型实现的细节有了更进一步的认识。

实验中遇到的问题包括以下几点：

1. 文本编码问题

需要在prepare.py文件中加入encoding=’utf-8’

1. 安装torch时，为了指定下载cuda支持的版本，需要使用以下安装命令

conda install pytorch==1.13.1 torchvision==0.14.1 torchaudio==0.13.1 pytorch-cuda=11.7 -c pytorch -c nvidia

|  |  |
| --- | --- |
| 实验成绩： |  |
| 教师签名： |  |