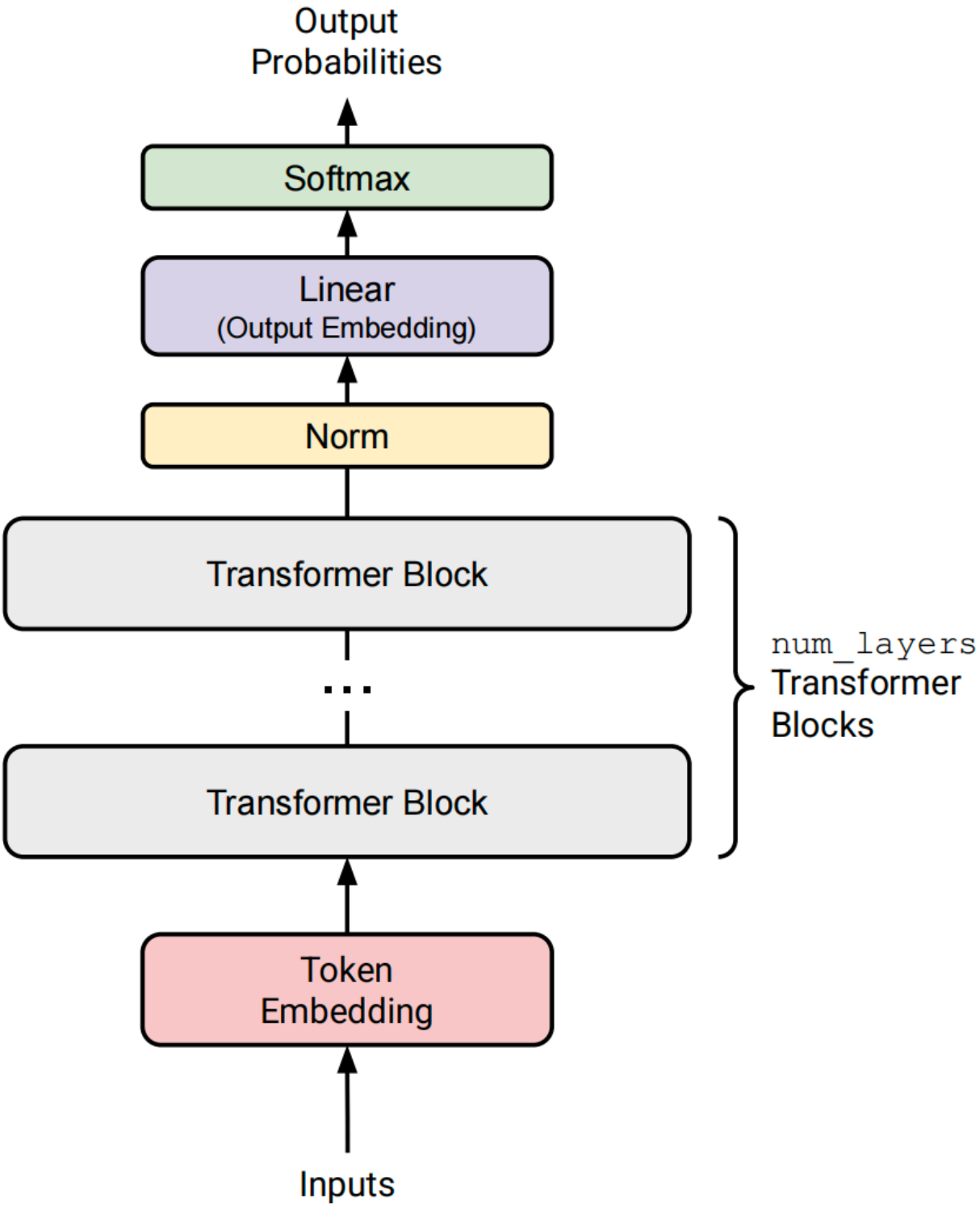
该文档记录了CS336作业1 architecture和simple LM部分的知识笔记,如有错误，欢迎指正。

Architecture

想要训练一个现代的大语言模型，我们通常可以将模型的结构划分为下面的层次



Pasted image 20251113193609.png

我们需要一个Tokenizer对文档进行Tokenization，将人可以看懂的文章转化为机器可以看懂的位置，在本课程的实践中，采用了byte-level的BPE算法训练了一个Tokenizer。对输入的材料进行Tokenization之后，我们经过一个embedding层，将一个one-hot的向量转化为一个高维空间中的dense tensor，这个tensor经过训练之后本身会携带该token的语义信息。接下来就是常规的Transformer decoder block模块进行自回归注意力的计算，最终经过FFN和softmax实现对下一个位置词的预测

Tokenization

训练一个BPE分词 -> 创建Tokenizer进行Tokenization

BPE分词算法

BPE分词从一个最基本的词汇表开始，逐步合并出现频率最高的字符对（合并之后词汇表表达，可以减少Tokenization之后长度，可以人为设定词汇表的大小来平衡词汇表的大小和句子长度），最终形成一个词汇表
BPE算法的流程：
1.预分词，将句子中按词进行划分，并且encode成utf-8形式，同时去除掉special_token（也就是special_token不参与合并）
2.统计不同字符对出现的频率，并且选取频率最高的单词进行合并

Tokenization

我们在语料库上训练BPE算法获得词汇表和merge_rules之后，在正式开始训练之前，需要对输入进行Tokenization
Tokenizaiton的流程：
1.预分词，和BPE算法一样我们首先需要对输入的文本进行预分词，转化成机器可读的形式
2.按照BPE获得的merge_rules进行合并，注意一定要按照merge_rules的顺序来进行合并
经历这一步之后，我们的文本就转化成了一个个的one-hot向量，经过embedding层之后，我们就获得了token的dense tensor

Transformer block

现代的Transformer block由MHA模块和FFN模块组成，在MHA模块之前我们需要对input进行投影，获得QKV三个矩阵，然后对Q，K应用RoPE位置编码，计算注意力分数，对V进行加权，最后拼接各个头的结果

RoPE

RoPE是旋转位置编码，是一种相对位置编码方式，公式表示如下

To inject positional information into the model, we will implement Rotary Position Embeddings [Su et al, 2021], often called RoPE. For a given query token $q^{(i)} = W_q x^{(i)} \in \mathbb{R}^d$ at token position i, we will apply a pairwise rotation matrix R^i, giving us $q'^{(i)} = R^i q^{(i)} = R^i W_q x^{(i)}$. Here, R^i will rotate pairs of embedding elements $q^{(i)}_{2k-1:2k}$ as 2d vectors by the angle $\theta_{i,k} = \frac{i}{\Theta^{(2k-2)/d}}$ for $k \in \{1, \ldots, d/2\}$ and some constant Θ. Thus, we can consider R^i to be a block-diagonal matrix of size $d \times d$, with blocks R^i_k for $k \in \{1, \ldots, d/2\}$, with

$$R^i_k = \begin{bmatrix} \cos(\theta_{i,k}) & -\sin(\theta_{i,k}) \\ \sin(\theta_{i,k}) & \cos(\theta_{i,k}) \end{bmatrix}. \tag{8}$$

Thus we get the full rotation matrix

$$R^i = \begin{bmatrix} R^i_1 & 0 & 0 & \ldots & 0 \\ 0 & R^i_2 & 0 & \ldots & 0 \\ 0 & 0 & R^i_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & R^i_{d/2} \end{bmatrix}, \tag{9}$$

Pasted image 20251113191406.png

简单来说，旋转的角度大小是跟token位置m以及位于特征向量中的位置有关系的。
同时我们可以发现，RoPE和输入的数据没有任何关系，只跟上述的两个因素有关系，因此我们可以预先计算出对应的cos和sin。并且借用self.register_buffer（这个函数是可以将某个数据注册为模型的一部分，随着模型在device间移动）保存称为model的一部分

SwiGLU

SwiGLU是Swish函数和GLU函数的综合体
Swish函数的表示为：
Swish(x)=x·σ(x)
GLU函数的表示为：
GLU(XW+b)=(XW1+b1)⊗σ(XW2+b2)
SwiGLU函数的表示为：
SwiGLU(XW+b)=(XW1+b1)⊗Swish(XW2+b2)
我们可以看到门控机制会引入额外的可学习的权重，所以课中提到的 d_ff = 4 *d_model 就需要变小一点来平衡SwiGLU所带来的额外计算。在实际中这个值通常为 3/8* d_model,同时也要求是64的倍数提高内存效率(d_model指的是transformer层输出的维度，d_ff指的是feedforward层的隐藏维度)

RMSNorm

RMSNorm是一种正则化手段，在现代的LLM中，大多数采用RMSNorm来代替LayerNorm，因为RMSNorm在和LayerNorm效果相当的同时，计算更为简单，计算复杂度低

$$\text{RMSNorm}(a_i) = \frac{a_i}{\text{RMS}(a)} g_i, \tag{4}$$

where $\text{RMS}(a) = \sqrt{\frac{1}{d_{\text{model}}} \sum_{i=1}^{d_{\text{model}}} a_i^2 + \varepsilon}$. Here, g_i is a learnable "gain" parameter (there are `d_model` such parameters total), and ε is a hyperparameter that is often fixed at 1e-5.

Pasted image 20251113200646.png

RMSNorm在归一化了输入值的同时，保持了输出值的均值非0的特性

Other features of modern LLM

Pre-norm

在Transformer原作之中，作者使用的是Post-norm，也就是在整个transformer模块之后进行norm，但是这种方式可以理解为破坏了残差连接的梯度流，因此，在现代的LLM结构中，一般采用Pre-norm的形式
Pre-norm的公式表示为(没有对残差连接应用Norm，确保了梯度流的顺利传播)
$x' = x + \text{MHA}(\text{RMSNorm}(x))$
但是笔者也看到过对于Pre-norm和Post-norm的更深刻的理解和对比，感兴趣的朋友可以看看苏神的blog或者搜索一下相关的论文的文章

Gradient clipling

在模型训练的过程中，如果梯度超过某一个阈值，会对grad进行缩放，确保数值稳定性

Softmax

在实际实现softmax函数时，为了确保数值稳定性，通常情况下我们需要先将所有的值减去最大值来保持数值稳定性，这种做法在数学上不会影响输出结果