Self-Attention

1. 概述

为了让注意力更好的发挥性能，学者提出了多头注意力的思想，其实就是将每个 query、key、value 分出来多个分支，有多少个分支就叫多少头，对 Q, K, V 求多次不同的注意力计算，得到多个不同的 output，再把这些不同的 output 拼接起来得到最终的 output。本 self_attention 函数实现了多头自注意力机制，这是 Transformer 架构中的关键部分。自注意力机制允许模型在处理序列数据时，动态地关注序列中不同位置的信息，从而捕捉序列内的依赖关系。函数接收查询（Q）、键（K）、值（V）张量以及相关的超参数，计算注意力分数并生成最终的隐藏状态。

2. 输入参数

hidden_states：用于存储最终的隐藏状态，形状为 (seq, n_kv_h * n_groups * dqkv)。
att_scores：用于存储注意力分数，形状为 (n_kv_h, n_groups, seq, total_seq)。
q：查询矩阵，形状为 (seq, n_kv_h * n_groups * dqkv)。
k：键矩阵，形状为 (total_seq, n_kv_h * dqkv)。
v：值矩阵，形状为 (total_seq, n_kv_h * dqkv)。
n_kv_h：键值头的数量。
n_groups：分组的数量。
seq_len：当前序列的长度。
total_seq_len：总序列的长度。
dqkv：查询、键和值向量的维度。

3. 核心难点及解决方法

3.1 多头部与分组的处理
难点：代码需要处理多个键值头（n_kv_h）和多个分组（n_groups），每个头部和分组的计算逻辑需要分别处理，同时要保证数据的正确索引和组织。。
解决方法：使用嵌套循环来遍历每个键值头和分组。对于每个键值头，提取对应的键和值数据，重组为合适的形状。对于每个分组，提取对应的查询数据，进行后续的注意力计算。
3.2 矩阵形状的调整与重组
难点：输入的 Q、K、V 矩阵的形状可能不适合直接进行注意力计算，需要将它们调整为合适的形状。例如，需要将 K 和 V 矩阵从 (total_seq, n_kv_h * dqkv) 重组为 (total_seq, dqkv)，以便与查询矩阵进行点积运算。
解决方法：通过循环遍历每个序列位置，提取每个头部的数据，并将其存储在新的向量中。然后使用这些向量创建新的张量，调整矩阵的形状。
3.3 注意力分数的计算与归一化
难点：计算注意力分数需要进行矩阵乘法，并且需要对结果进行缩放和归一化处理。缩

放操作是为了避免点积结果过大，归一化操作是为了将分数转换为概率分布。此外，还需要应用掩码操作来处理序列中的填充部分。

解决方法：使用 OP::matmul_transb 函数进行矩阵乘法，并在乘法过程中进行缩放。然后使用 OP::masked_softmax 函数对注意力分数进行归一化处理，确保每个位置的注意力权重之和为 1。

3.4 数据的存储与更新

难点：计算得到的注意力分数和隐藏状态需要正确地存储在对应的张量中。由于涉及多个头部和分组，需要确保数据的存储位置正确，避免数据覆盖或混淆。

解决方法：通过计算偏移量和索引，将注意力分数和隐藏状态数据复制到对应的张量位置。使用 copy_from_slice 方法确保数据的准确复制。

文本生成

1. 文本生成过程

难点：文本生成的核心在于如何根据输入的提示文本逐步生成后续内容。生成过程中需要处理多个关键步骤，包括：

（1）将输入文本转换为模型可理解的 token 序列。

（2）通过模型的前向传播计算每个 token 的概率分布。

（3）根据概率分布采样下一个 token，并确保生成的文本符合预期的长度和内容质量。

解决方法：

（1）使用 tokenizer 将输入文本编码为 token 序列（input_ids）。在生成过程中，使用模型的 forward 方法计算每个 token 的 logits（未归一化的概率分布）。

（2）使用 random_sample 函数根据 top_p、top_k 和 temperature 参数对 logits 进行采样，确保生成的文本既具有多样性又符合上下文逻辑。

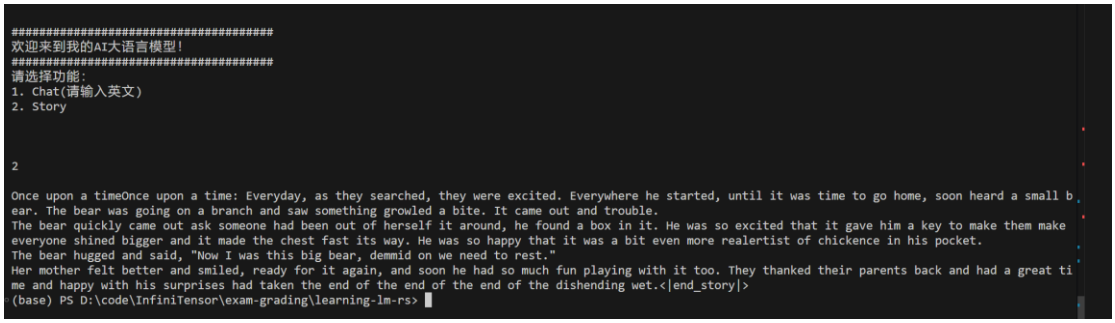
通过循环生成 token，直到达到最大生成长度（max_len）或遇到结束（eos_token_id）。

2. 关键参数的作用

top_p（核采样）：控制采样时考虑的 token 集合的大小。通过设置 top_p，可以限制采样范围，避免选择概率过低的 token，从而提高生成文本的连贯性。

top_k：限制采样时只考虑概率最高的前 k 个 token。与 top_p 类似，top_k 也可以控制生成文本的多样性，但更加直接。

temperature：调节概率分布的平滑程度。较高的 temperature 会使概率分布更加平滑，生成更多随机性文本；较低的 temperature 会使概率分布更加尖锐，生成更加确定性的文本。



AI 对话

1. 对话生成的核心逻辑

难点：

对话生成的核心在于如何根据用户输入和历史对话生成连贯的回复。生成过程需要处理以下问题：

（1）如何将用户输入和历史对话整合为模型可理解的输入格式。

（2）如何确保生成的回复符合上下文逻辑，同时避免重复或无意义的输出。

（3）如何控制生成的长度和质量，避免生成过长或不相关的文本。

解决方法：

（1）使用格式化字符串将用户输入和历史对话整合为模型可理解的输入格式。例如，使用<|im_start|>和<|im_end|>标记区分用户输入、系统消息和助手回复。

（2）通过历史记录管理将每次对话的用户输入和助手回复保存到 history 列表中，确保模型能够基于完整的上下文生成回复。

（3）通过 max_len、top_p、top_k 和 temperature 参数控制生成的长度和质量。

2. 历史记录管理

难点：

在对话场景中，历史记录的管理至关重要。模型需要基于完整的对话历史生成回复，但历史记录的存储和格式化可能带来以下问题：

（1）如何高效存储和检索历史记录。

（2）如何将历史记录整合为模型可理解的输入格式。

（3）如何避免历史记录过长导致输入超出模型的最大长度限制。

解决方法：

（1）使用 Vec<(String, String)>存储历史记录，每对字符串分别表示用户输入和助手回复。

（2）在每次生成回复前，遍历历史记录并将其格式化为模型可理解的输入字符串。例如，使用<|im_start|>user 和<|im_start|>assistant 标记区分用户和助手的对话内容。

（3）通过限制历史记录的长度或截断过长的历史记录，避免输入超出模型的最大长度限制。

```
######################请输入聊天的模式#################
1. 普通聊天模式
2. 历史记录模式
3. 退出


1
User:
hello,how are you?
Assistant: about buing things around.


######################请输入聊天的模式#################
1. 普通聊天模式
2. 历史记录模式
3. 退出


2

######################History############################
User: what is your name?,
Assistant: ly. "Momoe you make it better!"
Momoe a delicious treat, and it sparkled with excited when she got there."
Suddenly, the animal happened. The animal started to shake and started to ra
User: hello,how are you?,
Assistant: about buing things around.
######################History############################
```

3. 缓存机制

难点：

在对话生成过程中，模型需要多次调用 forward 方法计算每个 token 的 logits。为了提高效率，避免重复计算，需要使用缓存机制存储中间结果。然而，缓存的初始化和传递可能带来以下问题：

（1）如何初始化缓存并确保其在多次生成过程中保持一致。

如何在生成过程中正确传递和更新缓存。

解决方法：

（2）使用 llama.new_cache()初始化缓存，确保每次对话生成过程都有一个独立的缓存空间。

（3）在 chat_generate 函数中将缓存作为可变引用（&mut KVCache<f32>）传递，确保模型能够复用之前的计算结果，从而提高生成效率。

```
#######################################
欢迎来到我的AI大语言模型!
#######################################
请选择功能:
1. Chat(请输入英文)
2. Story



1


####################请输入聊天的模式#################
1. 普通聊天模式
2. 历史记录模式
3. 退出


1
User:
what is your name?
Assistant: ly. "Momoe you make it better!"
Momoe a delicious treat, and it sparkled with excited when she got there."
Suddenly, the animal happened. The animal started to shake and started to rain.
```