Prepacking: A Simple Method for Fast Prefilling and Increased Throughput in Large Language Models

SL LV

2024 年 8 月 20 日

1 摘要

在使用基于Transformer的大型语言模型（LLM）进行推理时，预填充（prefilling）是一个关键步骤，即在自回归生成之前计算输入token的键值（KV）缓存。对于较长的输入提示（prompt），预填充将对解码时间产生显著的开销。本文主要讨论了预填充过程中存在的一个重要问题：当批次中包含长度差异较大的提示时，大量计算资源浪费在填充token上，因为通常的做法是将所有序列填充到最大长度。

随着LLM越来越支持更长的上下文长度（可能达到1000万token），批次中提示长度的变化将变得更加显著。这使得优化预填充计算变得愈发重要。为了解决这一问题，本文提出了一种名为**prepacking**的方法，这是一种简单但有效的优化预填充计算的方法。为了避免在填充token上的冗余计算，prepacking方法将不同长度的提示组合到一个序列中，并使用**bin-packing**算法将多个序列打包成一个紧凑的批次。然后，它修改了注意力掩码和位置编码，以在单个序列内计算多个提示的预填充KV缓存。

在包含长度变化的标准数据集上进行实验时，我们发现与Huggingface中的默认基于填充的预填充计算相比，prepacking方法在速度和内存效率方面都有显著的改进，无论是在基本模型配置还是推理服务场景中。

2 引言

基于Transformer的大型语言模型（LLMs）已经成为处理自然语言查询的强大通用工具。随着语言模型规模的不断扩大及其在各个领域中的广泛应用，如何在保持生成速度和效率的前提下优化计算资源的分配成为了一个关键问题。

优化LLMs的挑战与传统软件不同。由于LLMs的通用性，它们可以接收非常多样化的查询，从简单的问题到长篇摘要任务。Transformer的二次运行时间意味着，较长的提示词需要比较短的提示词进行更多的计算。当同时请求长短提示词时，如何合理分配计算资源以确保推理效率是LLM推理面临的主要挑战之一。在当前的LLM框架下，随着模型规模的增加以及更长、更复杂的查询的出现，如何有效地分配计算资源成为比以往任何时候都更加重要的问题。

传统的LLM推理方法使用不同长度的输入时效率低下，这一点在Huggingface Transformers库中得到了充分的体现。尽管Huggingface库在NLP社区中得到了广泛应用，但其处理不同长度提示词的方式是将所有提示词填充到最长序列的长度，并通过Transformer模型处理整个批次。这导致了大量的内存利用率低下和计算效率问题。LLMs在预填充期间受到计算瓶颈的限制，在生成期间也会受到内存瓶颈的影响，因此优化内存和GPU利用率对于实现高效的推理和可扩展性至关重要。

在本文中，我们通过提出一种名为**prepacking**的替代预处理步骤来减轻这种浪费计算。Prepacking方法专门针对改进LLM预填充的速度和内存使用，预填充是生成前填充键值缓存（KV cache）的初始计算。Prepacking的概念非常简单：与其将每个序列填充到相同的长度，我们将多个提示词紧凑地打包到一个序列中，使用现成的bin-packing算法代替填充token。通过自定义的注意力掩码和位置编码，这使得在单个序列内计算一个批次中的多个提示成为可能。位置编码对序列中的每个提示词重新启动索引，掩码防止提示词在打包序列中相互影响（见图1）。预填充的KV缓存可以解包，用于后续生成阶段的提示词。

我们通过实验证明，与Huggingface中使用的完整批次方法相比，prepacking在NVIDIA A6000 GPU上将预填充和首次生成时间缩短了多达6倍。为了评估prepacking在代表真实世界用户流量条件下的运行时性能，我们在六个不同的语言数据集上测试了该方法，这些任务包括问答、摘要、指令跟随、语言建模和人类偏好建模，模型规模从1B到13B参数不等。Prepacking在批次内存在显著长度差异且批次较大时表现出更大的加速效果。此外，在预填充过程中，prepacking通过允许将批次大小扩大到16倍显著减少了内存消耗。在初步实验中，我们还证明了prepacking在生成过程中也能带来显著的加速和内存节省。