[LLaMA] Rotary positional embedding differs with official implementation #25199

Closed **lytning98** opened this issue on Jul 31, 2023 · 10 comments

lytning98 commented on Jul 31, 2023 • edited

`transformers` implement LLaMA model's Rotary Positional Embedding (RoPE) as follows:

transformers/src/transformers/models/llama/modeling_llama.py
Lines 173 to 188 in e42587f

```
173     def rotate_half(x):
174         """Rotates half the hidden dims of the input."""
175         x1 = x[..., : x.shape[-1] // 2]
176         x2 = x[..., x.shape[-1] // 2 :]
177         return torch.cat((-x2, x1), dim=-1)
178
179
180     def apply_rotary_pos_emb(q, k, cos, sin, position_ids):
181         # The first two dimensions of cos and sin are always 1, so we can `squeeze
182         cos = cos.squeeze(1).squeeze(0)  # [seq_len, dim]
183         sin = sin.squeeze(1).squeeze(0)  # [seq_len, dim]
```

This is **GPT-NeoX style** RoPE. But in Meta's official model implementation, the model adopts **GPT-J style** RoPE, which processes query and key vectors in an **interleaved way** instead of split into two half (as in `rotate_half` method).

Meta's official repo implements RoPE as (full code link):

```
def apply_rotary_emb(
    xq: torch.Tensor,
    xk: torch.Tensor,
    freqs_cis: torch.Tensor,
) -> Tuple[torch.Tensor, torch.Tensor]:
    xq_ = torch.view_as_complex(xq.float().reshape(*xq.shape[:-1], -1, 2))
    xk_ = torch.view_as_complex(xk.float().reshape(*xk.shape[:-1], -1, 2))
    freqs_cis = reshape_for_broadcast(freqs_cis, xq_)
    xq_out = torch.view_as_real(xq_ * freqs_cis).flatten(3)
    xk_out = torch.view_as_real(xk_ * freqs_cis).flatten(3)
    return xq_out.type_as(xq), xk_out.type_as(xk)
```

I'm confused with this difference, since `transformers.LlamaModel` can directly load weights converted from the officially released checkpoint, won't this lead to inconsistency in inference results? Is this difference expected?

👍 14

lytning98 changed the title ~~Rotary embedding in Llama model differs with official implementation~~ **[LLaMA] Rotary positional embedding differs with official implementation** on Jul 31, 2023

lytning98 closed this as completed on Jul 31, 2023

---

santiweide commented on Aug 22, 2023

same confusion

lytning98 commented on Aug 22, 2023 Author

> same confusion

@**santiweide** Params of some layers are re-permuted while converting weights in the official scripts. Check

transformers/src/transformers/models/llama/convert_llama_weights_to_hf.py
Lines 113 to 115 in `e42587f`

```
113     # permute for sliced rotary
114     def permute(w, n_heads=n_heads, dim1=dim, dim2=dim):
115         return w.view(n_heads, dim1 // n_heads // 2, 2, dim2).transpose(1, 2).reshap
```

👍 16 ❤️ 8 🚀 3

santiweide commented on Aug 22, 2023

ohhh thank you, we are converting the Megatron weight to ft weight, and we would check the shape of weights then

ffohturk commented on Oct 17, 2023

Awesome, thanks for clarifying this!

lwang2070 mentioned this issue on Nov 7, 2023

about rotary embedding in llama juncongmoo/pyllama#83

Closed

wangdongxuking61 commented on Feb 20

Awesome, thanks for clarifying this!

datourat mentioned this issue on Mar 30

Implementation of RoPE Lightning-AI/litgpt#1214

Closed

caixd-220529 commented on Mar 31

Thanks for the detailed illustration!!!

ShoufaChen commented on Apr 10

Thank you **@lytning98**, your answer saved my life.

May I ask the purpose behind this process?

transformers/src/transformers/models/llama/modeling_llama.py
Lines 177 to 181 in `6cdbd73`

```
177     def rotate_half(x):
178         """Rotates half the hidden dims of the input."""
179         x1 = x[..., : x.shape[-1] // 2]
180         x2 = x[..., x.shape[-1] // 2 :]
181         return torch.cat((-x2, x1), dim=-1)
```

I mean, why not use the interleaved pair as in Meta's official llama?
@zphang **@ArthurZucker** .

Thanks in advance.

4

Hannibal046 commented on Jun 12

> Thank you **@lytning98**, your answer saved my life.
>
> May I ask the purpose behind this process?
>
> **transformers/src/transformers/models/llama/modeling_llama.py**
> Lines 177 to 181 in `6cdbd73`
>
> ```
> def rotate_half(x):
> """Rotates half the hidden dims of the input."""
> x1 = x[..., : x.shape[-1] // 2]
> x2 = x[..., x.shape[-1] // 2 :]
> return torch.cat((-x2, x1), dim=-1)
> ```
>
> I mean, why not use the interleaved pair as in Meta's official llama?
> **@zphang** **@ArthurZucker** .
> Thanks in advance.

https://discuss.huggingface.co/t/is-llama-rotary-embedding-implementation-correct/44509/2

ArthurZucker commented on Jun 19 Collaborator

Few reasons that are already mentioned:

- first and foremost, and I can't stress this enough, licence
- second, eleuther's rope formulation (that we are using) is equivalent, maybe has one less operation that makes it more optimised

ArthurZucker mentioned this issue on Jul 9

ROPE implementation differs from official meta implementation #31859

Closed

4 tasks

Isotr0py mentioned this issue on Jul 11

[Core] Support loading GGUF model vllm-project/vllm#5191

Merged