PR #32375 完整报告

sgl-project/sglang

model: support EmbeddingGemma

合并时间：2026-07-27 10:40

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32375

执行摘要

- 一句话：支持 EmbeddingGemma 模型，集成双向注意力与 Breakable CUDA Graph
- 推荐动作：值得精读：本 PR 展示了如何利用现有模型骨架（Gemma3）快速实现编码器变体，并巧妙通过条件注意类型实现注意力重用，以及通过 server_args 的后期处理阶段完成模型特定的 BCG 配置。均值池化的 cumsum 实现也是一个简洁的批量池化示例。

功能与动机

Google 发布了 EmbeddingGemma 模型，其使用 gemma3_text 架构但启用 use_bidirectional_attention，需要作为编码器（嵌入模型）运行。sglang 需要原生支持其双向注意力、均值池化，并禁用不兼容的缓存与分块预填充机制以确保正确性。

实现拆解

1. 检测 EmbeddingGemma 配置：在 model_config.py 中新增 is_embedding_gemma 函数，检查 model_type == "gemma3_text" 且 use_bidirectional_attention=True；在 ModelConfig.__init__中设置 self.is_embedding_gemma 标记，并利用该标记修正 self.is_generation 判断。
2. 调整注意类型：在 gemma3_causal.py 的 Gemma3TextAttention.__init__中，根据 config.use_bidirectional_attention 条件选择 AttentionType.DECODER_BIDIRECTIONAL 或 DECODER，使注意类型动态适配。
3. 新增 EmbeddingGemmaModel 类：继承 Gemma3ForCausalLM，初始化时跳过 LM head，仅保留 model（Gemma3TextModel）和池化层（Pooler(PoolingType.MEAN, normalize=True)）。forward 直接调用 self.model 得到 hidden_states，再通过池化层输出 EmbeddingPoolerOutput。load_weights 支持原生 Gemma3 和 Sentence Transformers 两种 checkpoint 格式的权重映射。
4. 调整池化层：在 pooler.py 中新增 PoolingType.MEAN，实现批量序列的均值池化，使用 cumsum 和 prompt_lens 计算每个序列的均值。
5. 服务器自动配置：在 server_args.py 的 _handle_model_capability_adjustments 中，检测到 is_embedding_gemma 后，设置 disable_radix_cache=True、chunked_prefill_size=-1、禁用 decode CUDA graph，并在 CUDA 环境下启用 Breakable CUDA Graph（prefill backend 设为 BREAKABLE，自动设置 max_bs 和 batch sizes）。非 CUDA 环境则禁用 prefill graph。

6. 调度器和 TP worker 数据流调整：在 scheduler.py 和 tp_worker.py 中，forward_batch_embedding 返回值扩展为 (pooler_output, can_run_cuda_graph)，传递给 EmbeddingBatchResult，以便控制 CUDA Graph 执行路径。
7. 单元测试：添加 test_model_config.py 中的 TestEmbeddingGemmaConfig（测试 is_embedding_gemma 对双向 / 因果配置的判断），test_multimodal_piecewise_cuda_graph.py 中的 test_embedding_gemma_forces_breakable_prefill（验证 server_args 的配置强制），test_pooler_score_and_pool.py 中的 test_mean_pooling_respects_packed_sequence_boundaries（验证均值池化正确性）。

关键文件：

- python/sglang/srt/models/gemma3_causal.py（模块 模型层；类别 source；类型 data-contract；符号 EmbeddingGemmaModel, init, forward, load_weights）：添加 EmbeddingGemmaModel 类，继承自 Gemma3ForCausalLM，实现归一化均值池化，重载 forward 和 load_weights 以支持嵌入模式；同时修改 Gemma3TextAttention 注意类型为条件判断。
- python/sglang/srt/configs/model_config.py（模块 模型配置；类别 source；类型 data-contract；符号 is_embedding_gemma）：新增 is_embedding_gemma 检测函数，在 ModelConfig 初始化时设置标志，并影响 is_generation 判断。
- python/sglang/srt/server_args.py（模块 启动参数；类别 source；类型 core-logic）：添加 EmbeddingGemma 自动配置逻辑：禁用 radix cache 和 chunked prefill，启用 Breakable CUDA Graph。
- python/sglang/srt/layers/pooler.py（模块 池化层；类别 source；类型 core-logic）：新增 PoolingType.MEAN 枚举和对应的均值池化逻辑。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：扩展 EmbeddingBatchResult 以包含 can_run_cuda_graph 字段，支撑 CUDA Graph 控制。
- test/registered/unit/configs/test_model_config.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestEmbeddingGemmaConfig, test_detects_bidirectional_gemma3_text_config, test_does_not_misclassify_causal_gemma3）：新增 TestEmbeddingGemmaConfig 测试用例，验证 is_embedding_gemma 对双向 / 因果配置的判断。

关键符号：EmbeddingGemmaModel.init, EmbeddingGemmaModel.forward, EmbeddingGemmaModel.load_weights, is_embedding_gemma, pool_hidden_states (MEAN case)

关键源码片段

python/sglang/srt/models/gemma3_causal.py

添加 EmbeddingGemmaModel 类，继承自 Gemma3ForCausalLM，实现归一化均值池化，重载 forward 和 load_weights 以支持嵌入模式；同时修改 Gemma3TextAttention 注意类型为条件判断。

```
class EmbeddingGemmaModel(Gemma3ForCausalLM):
```

```
    """EmbeddingGemma's Gemma3 encoder with normalized mean pooling."""

    def __init__(
        self,
        config: Gemma3TextConfig,
        quant_config: Optional[QuantizationConfig] = None,
        prefix: str = "",
    ) -> None:
        # 不初始化 LM head，让 BCG 只捕获 transformer 主体，池化作为 eager 尾部
        PreTrainedModel.__init__(self, config=config)
        self.config = config
        self.quant_config = quant_config
        self.model = Gemma3TextModel(
            config, quant_config, prefix=add_prefix("model", prefix)
        )
        self.pooler = Pooler(pooling_type=PoolingType.MEAN, normalize=True)
        self.capture_aux_hidden_states = False

    @torch.no_grad()
    def forward(
        self,
        input_ids: torch.Tensor,
        positions: torch.Tensor,
        forward_batch: ForwardBatch,
        input_embeds: torch.Tensor = None,
        get_embedding: bool = True,
        **kwargs,
    ) -> EmbeddingPoolerOutput:
        assert get_embedding, "EmbeddingGemmaModel is only used for embeddings"
        hidden_states = self.model(
            input_ids, positions, forward_batch, input_embeds, **kwargs
        )
        return self.pooler(hidden_states, forward_batch)

    def load_weights(self, weights: Iterable[Tuple[str, torch.Tensor]]):
        """支持原生 Gemma3 和 Sentence Transformers 两种检查点格式。"""
        backbone_prefixes = ("embed_tokens.", "layers.", "norm.")
        remapped_weights = (
            (
                f"model.{name}" if name.startswith(backbone_prefixes) else name,
                weight,
            )
            for name, weight in weights
            if name.startswith("model.") or name.startswith(backbone_prefixes)
        )
        # 委托给 Gemma3ForCausalLM 的权重加载器（跳过 LM head）
        Gemma3ForCausalLM.load_weights(self, remapped_weights)
```

python/sglang/srt/configs/model_config.py

新增 is_embedding_gemma 检测函数，在 ModelConfig 初始化时设置标志，并影响 is_generation 判断。

```
def is_embedding_gemma(config) -> bool:
    """判断 config 是否为 EmbeddingGemma 双向注意力配置。"""
    return getattr(config, "model_type", None) == "gemma3_text" and getattr(
        config, "use_bidirectional_attention", False
    )

# 在 ModelConfig.__init__ 中：
self.hf_text_config = get_hf_text_config(self.hf_config)
self.is_embedding_gemma = is_embedding_gemma(self.hf_text_config)

# 后续根据该标记修正 is_generation：
self.is_generation = not self.is_embedding_gemma and is_generation_model(
    self.hf_config.architectures, is_embedding
)
```

评论区精华

本 PR 无实质 review 讨论，仅包含自动 bot 通知和作者触发 CI 的命令。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险：修改了 Gemma3TextAttention 的注意类型条件（原本固定双向，现在根据配置条件），可能影响已存在的 Gemma3 因果模型（如 Gemma3ForCausalLM）的正确性，但通过 test_does_not_misclassify_causal_gemma3 测试确保。
  2. Breakable CUDA Graph 配置代码位于 server_args 的后期处理阶段，如果其他模型检测条件冲突可能导致配置覆盖顺序问题。
  3. 均值池化使用 cumsum 计算，在大 batch 下可能存在数值精度问题。
  4. EmbeddingGemmaModel 类直接继承 Gemma3ForCausalLM，若未来 Gemma3ForCausalLM 构造变化可能需同步更新。 - 影响：用户：现在可以使用 --model-type 加载 EmbeddingGemma 模型（如 google/embedding-gemma），并获得正确的嵌入输出和 BCG 加速。系统：新增一种模型架构和池化类型，调度器数据流增加 can_run_cuda_graph 字段。团队：需要维护 EmbeddingGemma 相关的配置检测和 BCG 兼容性，测试新增 3 个单元测试（约 50 行）。无明显性能或安全风险。 - 风险标记：新模型支持，Breakable CUDA Graph 依赖，池化类型扩展

关联脉络

- 暂无明显关联 PR