PR #28691 完整报告

sgl-project/sglang

Add LFM2.5 embedding model support

合并时间：2026-07-29 19:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28691

执行摘要

- 一句话：添加 LFM2.5 双向嵌入模型支持
- 推荐动作：值得精读，特别是 AttentionType.ENCODER_ONLY 的引入以及 is_generation_model 注册机制，是 SGLang 嵌入模型集成的标准路径。可作为添加类似 bidirectional encoder 模型的模板。

功能与动机

LiquidAI/LFM2.5-Embedding-350M 是一个密集双编码器，其 HuggingFace checkpoint 暴露 Lfm2BidirectionalModel，但 SGLang 之前只支持因果 LFM2 路径。此 PR 添加最小模型支持，使 /v1/embeddings 能够通过现有的 prefill-only 嵌入流程返回模型的 CLS 池化归一化向量。

实现拆解

实现分为以下步骤：

1. 修改 Lfm2Attention 添加 attn_type 参数，使注意力层可切换为 ENCODER_ONLY（无因果掩码）。
2. 新增 Lfm2BidirectionalShortConv 类，继承 Lfm2ShortConv，重写 forward 为 same-padding 非因果卷积，支持非等长 batch 的 padding/unpadding 操作。
3. 修改 Lfm2DecoderLayer，通过 bidirectional 标志控制传递 AttentionType.ENCODER_ONLY 或 DECODER。
4. 新增 Lfm2BidirectionalModel 类，继承 Lfm2ForCausalLM，重写 get_num_kv_cache_layers 返回 0，添加 Pooler 实现 CLS 池化 + L2 归一化，复用原权重加载逻辑。
5. 在 model_config.py 的 is_generation_model 中注册 Lfm2BidirectionalModel，使其自动识别为 embedding 模型，无需命令行参数。

关键文件：

- python/sglang/srt/models/lfm2.py（模块 模型层；类别 source；类型 core-logic；符号 Lfm2BidirectionalShortConv, forward, Lfm2BidirectionalModel, init）：核心模型文件，新增 Lfm2BidirectionalModel 类及其组件，是主要变更

- python/sglang/srt/configs/model_config.py（模块 配置；类别 source；类型 configuration）：注册新架构为非生成模型，使自动检测生效

关键符号：Lfm2BidirectionalShortConv.forward, Lfm2BidirectionalModel.init, Lfm2BidirectionalModel.forward, Lfm2BidirectionalModel.get_num_kv_cache_layers, Lfm2BidirectionalModel.load_weights

关键源码片段

python/sglang/srt/models/lfm2.py

核心模型文件，新增 Lfm2BidirectionalModel 类及其组件，是主要变更

```
# Lfm2BidirectionalShortConv 使用 same padding 的卷积代替 causal 卷积
class Lfm2BidirectionalShortConv(Lfm2ShortConv):
    # 与父类的 causal 卷积不同，此处使用 F.conv1d 的 padding 参数实现非因果效果
    def forward(self, hidden_states, forward_batch):
        if forward_batch.forward_mode.is_idle():
            return hidden_states

        proj, _ = self.in_proj(hidden_states)
        B_gate, C_gate, x = proj.chunk(3, dim=-1)
        Bx = B_gate * x

        seq_lens = forward_batch.extend_seq_lens_cpu
        if seq_lens is None:
            seq_lens = forward_batch.extend_seq_lens.detach().cpu().tolist()
        max_len = max(seq_lens)

        padded = Bx.new_zeros((len(seq_lens), max_len, Bx.shape[-1]))
        offset = 0
        for batch_idx, seq_len in enumerate(seq_lens):
            seq_len = int(seq_len)
            end = offset + seq_len
            padded[batch_idx, :seq_len] = Bx[offset:end]
            offset = end

        # 使用 same padding，保证输出长度与输入长度一致（自动截断或填充）
        conv_out = F.conv1d(
            padded.transpose(1, 2),
            weight=self.conv_weight.unsqueeze(1),
            bias=self.conv_bias,
            padding=self.conv_kernel // 2,
            groups=self.hidden_size_per_partition,
        )
        if conv_out.shape[-1] > max_len:
            conv_out = conv_out[..., :max_len]
        elif conv_out.shape[-1] < max_len:
            conv_out = F.pad(conv_out, (0, max_len - conv_out.shape[-1]))
        conv_out = conv_out.transpose(1, 2)
```

```
        unpadded = Bx.new_empty(Bx.shape)
        offset = 0
        for batch_idx, seq_len in enumerate(seq_lens):
            seq_len = int(seq_len)
            end = offset + seq_len
            unpadded[offset:end] = conv_out[batch_idx, :seq_len]
            offset = end

        output, _ = self.out_proj(C_gate * unpadded)
        return output
```

评论区精华

PR review 仅含一条 approve 评论（'LGTM, very clean'），内部讨论集中在应依赖架构自动检测而非 --is-embedding 参数。作者最终选择注册非生成架构列表，避免了用户额外配置。

- 自动检测模型类型 (design): 采用注册方式，合并 PR。

风险与影响

- 风险：主要风险：修改了共享的 LFM2 模型文件（lfm2.py），可能影响原有因果 LFM2 模型的加载与推理行为。但新增代码均封装在新类或条件分支中，原有路径不受影响。缺少专用单元测试，回归依赖手动验证。另外，非因果短卷积中的 padding 逻辑要求正确计算长度，若扩展序列长度与假设不符可能导致越界。
- 影响：对用户：可直接使用 LFM2.5 嵌入模型，无需手动指定 --is-embedding，降低使用门槛。对系统：新增架构注册，不影响已有模型。对团队：为后续嵌入模型集成提供了参考模式。影响程度中等偏小。
- 风险标记：共享文件修改，缺少单元测试，padding 边界处理

关联脉络

- 暂无明显关联 PR