PR #1820 完整报告

radixark/miles

feat(session): pass request chat_template_kwargs to apply_chat_template

合并时间：2026-07-29 14:57

原文链接：http://prhub.com.cn/radixark/miles/pull/1820

执行摘要

- 一句话：会话支持请求级 chat_template_kwargs 覆盖渲染
- 推荐动作：值得精读。该 PR 展示了如何在共享底层 tokenizer 的前提下为每个请求创建轻量配置副本，以及如何处理模型家族的同义参数别名。合并逻辑 merge_chat_template_kwargs 和 clone_with_chat_template_kwargs 的设计可作为后续其他请求级配置覆盖的参考模板。

功能与动机

PR body 明确指出：Session requests accepted chat_template_kwargs, but local apply_chat_template calls did not receive them. 即客户端设置的响应模式（如 enable_thinking=false）只影响后端解析，而 session 本地渲染 input_ids 仍使用启动参数，导致本地 token 流与 sglang 解析视角分裂，影响 TITO 增量 token 跟踪的正确性。

实现拆解

1. 在 miles/utils/chat_template_utils/tito_tokenizer.py 中为 TITOTokenizer 新增 clone_with_chat_template_kwargs 方法，基于同一个 HF tokenizer 创建请求级 tokenizer 副本，并通过 template.merge_chat_template_kwargs 合并参数；同时为 DeepSeekV32/V4 声明 chat_template_kwarg_aliases（thinking_mode/enable_thinking/thinking）。
2. 在 miles/utils/chat_template_utils/template.py 中新增 merge_chat_template_kwargs 工具函数，支持别名键组整体替换：当请求中出现别名组任意键时，先清空 base 中整组键再应用请求层，避免同义键并存。
3. 在 miles/rollout/session/core.py 的 chat_completions 中，从 request body 读取 chat_template_kwargs，校验类型后调用 clone_with_chat_template_kwargs 获得请求级 tokenizer，并将解析后的 kwargs 原样写回 request_body（转发给后端），同时用于本地 prepare_pretokenized 渲染。
4. 测试配套：tests/fast/router/test_sessions.py 新增端到端测试验证请求 override 同时影响本地 input_ids 和后端转发；tests/fast/utils/chat_template_utils/test_tito_tokenizer.py 新增 DeepSeek V3.2/V4 请求 thinking=True 覆盖启动 enable_thinking=False 的单元测试。

关键文件：

- miles/utils/chat_template_utils/tito_tokenizer.py（模块 模板渲染；类别 source；类型 core-logic；符号 clone_with_chat_template_kwargs）：核心源码：新增 clone_with_chat_template_kwargs 方法，并为 DeepSeek V3.2/V4 声明 chat_template_kwarg_aliases，实现请求级 tokenizer 副本。
- miles/rollout/session/core.py（模块 会话服务；类别 source；类型 core-logic）：核心集成：chat_completions 中读取请求级 chat_template_kwargs，克隆 tokenizer 并在本地渲染使用，同时转发合并后的 kwargs 给后端。
- miles/utils/chat_template_utils/template.py（模块 模板工具；类别 source；类型 core-logic；符号 merge_chat_template_kwargs）：新增 merge_chat_template_kwargs 工具函数，实现别名组整体替换的合并语义，是解决冲突键问题的关键。
- tests/fast/router/test_sessions.py（模块 会话测试；类别 test；类型 test-coverage；符号 test_chat_template_kwargs_override_reaches_render_and_backend）：端到端测试验证请求级 override 对本地 input_ids 和后端转发的影响，确保行为符合预期。
- tests/fast/utils/chat_template_utils/test_tito_tokenizer.py（模块 分词器；类别 test；类型 test-coverage；符号 test_deepseek_request_thinking_overrides_startup_mode）：单元测试覆盖 DeepSeek 家族请求 thinking 覆盖启动模式的场景，验证别名组合并逻辑。

关键符号：clone_with_chat_template_kwargs, merge_chat_template_kwargs, chat_completions

关键源码片段

miles/utils/chat_template_utils/tito_tokenizer.py

核心源码：新增 clone_with_chat_template_kwargs 方法，并为 DeepSeek V3.2/V4 声明 chat_template_kwarg_aliases，实现请求级 tokenizer 副本。

```
# miles/utils/chat_template_utils/tito_tokenizer.py

class TITOTokenizer:
    # 每个模型家族可声明一组互斥的模板参数别名，合并时按组整体替换
    chat_template_kwarg_aliases: frozenset[str] = frozenset()

    def clone_with_chat_template_kwargs(self, request_kwargs):
        # 创建请求级副本：复用同一个 HF tokenizer 底层对象，仅替换
        # chat_template_kwargs，保证本地渲染与后端解析使用同一份参数
        return type(self)(
            self.tokenizer,
            chat_template_kwargs=template.merge_chat_template_kwargs(
                self.chat_template_kwargs,
                request_kwargs,
                alias_keys=self.chat_template_kwarg_aliases,
            ),
            assistant_start_str=self._assistant_start_str,
        )

# DeepSeek 家族将 thinking_mode / enable_thinking / thinking 视为同义键
```

```
_DEEPSEEK_MODE_KWARG_ALIASES = frozenset({'thinking_mode', 'enable_thinking', 'thinking'}
)

class DeepSeekV32TITOTokenizer(TITOTokenizer):
    chat_template_kwarg_aliases = _DEEPSEEK_MODE_KWARG_ALIASES

class DeepSeekV4TITOTokenizer(TITOTokenizer):
    chat_template_kwarg_aliases = _DEEPSEEK_MODE_KWARG_ALIASES
```

miles/rollout/session/core.py

核心集成：chat_completions 中读取请求级 chat_template_kwargs，克隆 tokenizer 并在本地渲染使用，同时转发合并后的 kwargs 给后端。

```
# miles/rollout/session/core.py

# FIXME(session): 只有嵌套的 chat_template_kwargs 能到达本地渲染,
# 顶层 reasoning / reasoning_effort 尚未映射到模板 kwargs。
request_ctk = request_body.get('chat_template_kwargs')
if request_ctk is not None and not isinstance(request_ctk, dict):
    raise MessageValidationError('chat_template_kwargs must be an object')

tito_tokenizer = self.registry.tito_tokenizer
if request_ctk:
    try:
        # 请求级副本：合并后的 kwargs 同时驱动本地渲染与后端转发
        tito_tokenizer = tito_tokenizer.clone_with_chat_template_kwargs(request_ctk)
    except ValueError as e:
        raise MessageValidationError(str(e)) from e

if tito_tokenizer.chat_template_kwargs:
    request_body['chat_template_kwargs'] = dict(tito_tokenizer.chat_template_kwargs)
else:
    request_body.pop('chat_template_kwargs', None)

request_messages = request_body.get('messages', [])
prompt_token_ids = session.prepare_pretokenized(
    request_messages,
    tools=request_body.get('tools'),
    tito_tokenizer=tito_tokenizer,
)
```

miles/utils/chat_template_utils/template.py

新增 merge_chat_template_kwargs 工具函数，实现别名组整体替换的合并语义，是解决冲突键问题的关键。

```
# miles/utils/chat_template_utils/template.py

def merge_chat_template_kwargs(base, overrides, *, alias_keys=()):
    # 将 overrides 合并到 base；若 overrides 命中任一 alias_keys,
```

```
    # 先移除 base 中整组别名，再应用请求层，避免同义键同时存在
    merged = dict(base)
    if any(key in overrides for key in alias_keys):
        for key in alias_keys:
            merged.pop(key, None)
    merged.update(overrides)
    return merged
```

评论区精华

Shi-Dong 在 issue 评论中提出关键疑虑：如果服务器启动时 enable_thinking=False，而后续请求发送 thinking=True，合并后的 kwargs dict 会同时包含 enable_thinking 和 thinking 两个冲突键。guapisolo 随后在 9b742314df 修复：请求参数通过 template.merge_chat_template_kwargs 按别名组整体替换，DeepSeek 请求提供 thinking_mode/enable_thinking/thinking 任一键时，先移除启动值整组再应用请求层。此外 Shi-Dong 对 core.py 中冗长的 FIXME 注释提出 nit，guapisolo 在 e8cc944 中简化注释，仅保留当前 request-shape 约束。

- 请求 thinking=True 与启动 enable_thinking=False 的冲突 (correctness): guapisolo 通过 merge_chat_template_kwargs 的 alias_keys 按组替换修复：DeepSeek 请求含任一别名键时，先清空启动值的整组键再应用请求层。
- FIXME 注释可读性 (style): guapisolo 在 e8cc944 简化注释，仅保留当前 request-shape 约束：只有嵌套 chat_template_kwargs 到达本地渲染。
- 顶层 reasoning 字段未映射 (design): 作为已知限制保留 FIXME，后续可能通过单独 PR 补齐。

风险与影响

- 风险：
  1. 核心请求路径变更：SessionCore.chat_completions 是 session 服务的关键路径，变更后每次请求都会额外创建 tokenizer 副本，虽共享底层 HF tokenizer，但需关注极端高并发下的对象创建开销。
  2. mid-session 模式切换未保护：FIXME 指出 DeepSeek V3.2/V4 在会话中途切换 thinking 模式会遗留旧模式前缀 token，导致历史与新模式不匹配，当前未加防护。
  3. 顶层 reasoning 字段未映射：FIXME 明确只有嵌套 chat_template_kwargs 生效，顶层 reasoning/reasoning_effort 不会被处理，用户若按 SGLang 习惯传顶层字段会静默失效。
  4. 合并顺序依赖 alias_keys：非 DeepSeek 家族使用普通同键覆盖，若后续新家族加入而忘记配置 aliases，可能出现同义键冲突。 - 影响：用户侧：session 客户端现在可以在单次请求中控制响应模式（如 enable_thinking），无需重启服务或新建 session，提升 API 灵活性。系统侧：session server 请求处理增加一次轻量 tokenizer 克隆，开销可忽略；本地 input_ids 渲染与后端 sglang 使用的 kwargs 一致，避免 token 流分裂。团队侧：提供了请求级模板参数的统一合并范式，并留下 FIXME 指引后续补齐顶层 reasoning 映射和 mid-session 安全保护。 - 风险标记：核心请求路径变更，

mid-session 模式切换未保护，顶层 reasoning 未映射

关联脉络

- PR #1759 (2/2) refactor(session): assemble training samples on the session server; records never leave it: 同样修改了 miles/rollout/session/core.py，是 session 服务器重构主线，本 PR 在该基础上继续扩展请求处理能力。
- PR #2028 session: collect speculative-decoding counters: 同属 session 模块的探索，改动 session 样本合并路径，与本 PR 的请求处理路径相邻，体现 session 功能的持续演进。
- PR #1916 (1/2) refactor(rollout): drop --generate-multi-samples and its per-turn sample semantics: rollout 样本语义调整，与本 PR 的模板渲染链路相关，均服务于 session 服务的输入输出对齐。