PR #25960 完整报告

sgl-project/sglang

bugfix: --decrypted-draft-config-file not applied

合并时间：2026-05-29 09:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25960

执行摘要

- 一句话：修复 --decrypted-draft-config-file 未生效的问题
- 推荐动作：本 PR 修改简单但定位准确，值得阅读的要点包括：如何通过 kwargs 机制向配置加载过程注入额外参数；以及可变默认参数的实际取舍方式。建议在类似功能中统一使用 None 默认值以提升安全性。

功能与动机

用户使用 --decrypted-draft-config-file 为草稿模型指定加密配置文件时，_resolve_speculative_algorithm_alias 调用 get_config 未传入该文件，导致模型类型无法识别而抛出 ValueError: Unrecognized model。需要将用户指定的配置文件路径传递到配置加载过程中。

实现拆解

1. 在 _resolve_speculative_algorithm_alias 的函数签名中新增 kwargs: Optional[dict] = {} 参数，使其能接收额外的关键字参数（如 _configuration_file）。
2. 在 handle_speculative_decoding 函数中，从 server_args.decrypted_draft_config_file 读取用户指定的加密配置文件路径，若存在且非空，则以 _configuration_file 键存入 kwargs 字典。
3. 调用 _resolve_speculative_algorithm_alias 时传入该 kwargs，并在其内部通过 **kwargs 解包传递给 get_config，最终使 AutoConfig.from_pretrained 使用指定的配置文件。
4. 测试配套：本 PR 未包含单元测试文件；但 PR body 引用了 NPU 上的测试脚本，验证 EAGLE3 + 加密配置场景可正常启动。

关键文件：

- python/sglang/srt/arg_groups/speculative_hook.py（模块 推测解码；类别 source；类型 core-logic；符号 _resolve_speculative_algorithm_alias, handle_speculative_decoding）：唯一修改的文件，对 _resolve_speculative_algorithm_alias 和 handle_speculative_decoding 两处作了改动，使 --decrypted-draft-config-file 能传递到配置加载流程中。

关键符号：_resolve_speculative_algorithm_alias, handle_speculative_decoding

关键源码片段

python/sglang/srt/arg_groups/speculative_hook.py

唯一修改的文件，对 _resolve_speculative_algorithm_alias 和 handle_speculative_decoding 两处作了改动，使 --decrypted-draft-config-file 能传递到配置加载流程中。

```
fromtypingimport Optional def_resolve_speculative_algorithm_alias( speculative_algorithm: Optional[str],  speculative_draft_model_path: Optional[str], trust_remote_code: bool = False,  # 新增 kwargs 参数，允许调用方传入额外配置，例如 _configuration_file  kwargs: Optional[dict] = {}, ) -> Optional[str]:  # 解析 CLI 推测算法；NEXTN/EAGLE 可能变为 FROZEN_KV_MTP（针对 Gemma4 助理草稿） is_gemma4_draft = False  if speculative_draft_model_path: fromsglang.srt.utils.hf_transformers_utilsimport get_config  # 调用 get_config 时解包 kwargs，传入用户指定的加密配置文件路径  cfg = get_config( speculative_draft_model_path,  trust_remote_code=trust_remote_code, **kwargs,  )  is_gemma4_draft = 'Gemma4AssistantForCausalLM' in (  getattr(cfg, 'architectures', None) or []  )  # ... 后续算法逻辑不变 ...  # 省略 Gemma4 判断和返回值 defhandle_speculative_decoding(server_args: 'ServerArgs') -> None:  # ... 前面代码 ...  if server_args.speculative_algorithm is not None: server_args.speculative_algorithm = server_args.speculative_algorithm.upper()  # 构造额外参数字典  kwargs = {}  override_config_file = server_args.decrypted_draft_config_file  if override_config_file and override_config_file.strip():  # 使用 _configuration_file 键，该键会被 get_config 识别并传递给 AutoConfig  kwargs['_configuration_file'] = override_config_file.strip() server_args.speculative_algorithm = _resolve_speculative_algorithm_alias( server_args.speculative_algorithm,  server_args.speculative_draft_model_path,  trust_remote_code=server_args.trust_remote_code,  kwargs=kwargs,  )  # ...后续代码...
```

注意：上述代码片段为整理后的摘录，省略了原函数的其他部分以突出关键改动。完整实现请查看合并后文件。

评论区精华

Review 中 gemini-code-assist[bot] 指出 kwargs: dict = {} 使用可变默认参数存在风险，建议改为 None 并在展开时用 **(kwargs or {})。作者 McZyWu 接受了类型提示修改（改为 Optional[dict] = {}），但拒绝了解包方案，认为当前调用路径不会并发修改 kwargs。该讨论启示了在公共 API 中使用可变默认参数的注意事项。

- 避免可变默认参数 (design): 作者保留了默认字典，未采用 None 解包方案，但将类型的 Optional[dict] 引入签名。合并时接受了该状态。

风险与影响

- 风险：主要风险是 kwargs 默认值仍然是一个可变字典，若未来出现并发调用该函数且修改字典，可能导致意外共享状态。当前调用仅在启动时的串行代码中进行，风险较低。此外，

本修改仅影响推测解码路径，可能对其他算法（如 NEXTN、FROZEN_KV_MTP）无影响且已经过 NPU 测试验证。缺少对 CPU/GPU 场景的测试覆盖。

- 影响：对用户：之前因加密配置文件不生效而无法启动的草稿模型，现在可正常使用 --decrypted-draft-config-file 参数启动。对系统：无性能或资源影响。对团队：需注意该修复对合并分支后其他推测解码功能可能产生交互，但逻辑简单风险可控。
- 风险标记：核心路径变更 ，缺少测试覆盖

关联脉络

- 暂无明显关联 PR