Future-Seed：一种基于跨层状态回灌的原地约束修复机制（RWKV7-CUDA 技术报告：从 Future-Aware Copy 到置换修复与数独）

yanghu819

2026 年 2 月 15 日

摘要

固定记忆的递归语言模型（RNN/SSM 系）在“原地（in-place）约束修复”任务上常出现结构性脆弱：早期位置的决策难以有效利用序列右侧信息，且并行填空容易产生全局一致性冲突（重复/遗漏/违约束）。本文基于 `future-seed` 的核心思想：将上一层扫描完整段序列后的终止状态作为下一层的初始状态（跨层回灌），从而在深度方向形成“再读一遍序列”的信息通路。我们在单卡 RTX 4090 上，使用 RWKV7 `cuda_wind` 实现并评测一组可复现的 toy 任务族：**RIGHTCOPY/CONSTR**（future-aware sanity）、**KVSORT**（key-value 置换修复）、**PERMFILL**（置换填空与长度外推）、以及 **4×4 数独**（二维约束）。结果显示：Future-Seed 在 sanity 任务上稳定提升；在 KVSORT 设定（keys36, $n = 20$）中，**FS=0 完全失败（exact=0）而 FS=1 达到 exact=1**；在 PERMFILL 中，引入极小的 **anchor**（masked span 内保留 $k = 2$ 个真值 token）可显著改善长度外推（例如训练 $n_{\max} = 32$ 时对 $n = 36$ 仍保持 ≈ 0.935 的 exact/valid）；在 4×4 数独中 Future-Seed 明显推迟相变点，并可与一个极简软约束正则协同提升（holes=12：$\approx 0.48 \to 0.59$）。这些结果支持 Future-Seed 作为“全局约束记账/传播通道”的机制解释，并为后续扩展到更真实的原地编辑与结构修复任务提供了可操作的实验信号。

1 引言

我们关注的一类任务并非“从头生成文本”，而是**原地（in-place）修复**：在不重写整体结构的前提下，局部填补缺失并满足全局一致性约束。其困难点通常不是局部 token 分类，而是全局一致性：置换一致性（不重不漏）、括号/JSON 结构合法性、实体/变量命名一致性、以及约束满足问题（如数独）。

对固定记忆的递归模型而言，困难来自强因果扫描：序列左侧的预测很难利用右侧信息；而并行填空（一次性给出所有 mask 的预测）又容易产生冲突解。Future-Seed 通过跨层状态回灌，在深度方向提供额外的信息通路，使早期位置能间接依赖“未来”上下文。本文将该直觉落地为一组**极小但硬**的任务族，并给出可复现的对照证据。

2 方法：Future-Seed

设模型由 L 层堆叠而成。对第 l 层（$l \geq 1$），Future-Seed 将上一层在读完整段长度 T 后的终止状态 $s_T^{(l-1)}$，作为本层的初始状态 $s_0^{(l)}$（或其门控版本）：

$$s_0^{(l)} \leftarrow s_T^{(l-1)}.$$

直观上，这等价于在深度方向对同一序列进行“多次线性扫描”，使后层在处理早期 token 时携带更多来自整段序列的信息摘要。本文实验使用 RWKV7 的 `cuda_wind` 实现以保证吞吐与可复现性。

3 任务族与指标

所有任务均采用统一的 in-place 编码骨架：`P=…|M=___…|R=…`，其中 R（右侧）提供信息，M（中间）为需要填补/修复的 mask 段；训练损失仅在 mask 位置计算。

3.1 RIGHTCOPY / CONSTR（future-aware sanity）

RIGHTCOPY/CONSTR 是最小化的 future-aware 探针任务，用于检验“右侧信息能否有效影响左侧 mask 的预测”。指标为 mask token 的准确率（acc）。

3.2 KVSORT（Key-Value Sort, in-place 置换修复）

KVSORT 将右侧 R 设为乱序的 key-value 信息，要求在中间 M 输出按 key 排序后的序列（置换修复）。我们报告 `kvsort_id exact` 与 `kvsort_ood exact`（exact-match）。

3.3 PERMFILL（Permutation Fill, 长度外推）

PERMFILL 是更直接的置换填空任务。我们额外引入一个极简设定：在 masked span 内保留极少量真值 token 作为 **anchor**（例如 $k = 2$），以减少“全是 mask”带来的对齐与收敛不稳定。指标为 exact/valid（两者在该实现中一致）。

3.4 4×4 数独（二维约束修复）

我们合成生成合法 4×4 数独解盘（数字 1–4，2×2 宫），并随机挖空 h 个位置作为输入。报告：

- **solve-rate**：预测盘满足行/列/宫约束（每组恰好包含 {1,2,3,4}）。
- **exact**：预测盘与隐藏合成解完全一致的比例。

4 实验设置

实验在单卡 NVIDIA RTX 4090 上运行，采用 RWKV7 `cuda_wind` 内核。我们使用同一训练脚本 `rwkv_diff_future_seed.py`，通过环境变量切换任务与超参。对比为：

- **FS=0**：不启用 Future-Seed。
- **FS=1**：启用 Future-Seed（跨层回灌）。

5 结果

5.1 RIGHTCOPY / CONSTR：sanity 上的稳定提升

表 1 汇总了 3 个随机种子在训练末端的准确率（均值 ± 标准差）。

表 1: RIGHTCOPY/CONSTR 的最终准确率（3 seeds, mean±sd）。

task	FS=0	FS=1
rightcopy acc	0.1046 ± 0.0074	0.1550 ± 0.0176
constr acc	0.0977 ± 0.0028	0.2114 ± 0.0179

5.2 KVSORT：FS=0 与 FS=1 的分水岭

在 keys36、$n = 20$、nosep、$L = 8$、SEQ_LEN = 256 的设定下，FS=0 完全失败，而 FS=1 达到完美 exact-match（表 2）。

表 2: KVSORT（keys36, $n = 20$, nosep, L8, SEQ_LEN=256）的 exact 指标。

metric	FS=0	FS=1
kvsort_id exact	0.0	1.0
kvsort_ood exact	0.0	1.0

5.3 Transformer 注意力基线（负结果）

为排除“KVSORT 只是因为看不见右侧信息才失败”的解释，我们在同一代码中实现了多种 Transformer 基线：(i) 双向 Transformer-MLM（`MODEL=transformer`），(ii) 因果 decoder-only Transformer（`MODEL=transformer_causal`），以及 (iii) attention-side Future-Seed：跨层用全局 token 回传右侧摘要（`ATTN_FS=1`, K=32）。在 KVSORT（keys36, n=20, nosep）上，这些 Transformer 基线均无法恢复正确排序（exact=0.0）。Hungarian 解码可以强制置换合法性（`key_valid=1.0`）但排序仍为 0。我们也测试了迭代式 refinement（`REFINE_STEPS=8`）与 Sinkhorn 置换损失（`PERM_SINKHORN=1`），同样未能解决该失败模式。

表 3: KVSORT 的 Transformer 基线（keys36, $n = 20$, nosep）。Hungarian 可保证合法性，但排序仍失败。

model	decode	exact	key_valid	key_order
Transformer-MLM	argmax	0.0	0.0	0.0
Transformer-MLM	hungarian	0.0	1.0	0.0
Transformer-Causal	hungarian	0.0	1.0	0.0
Transformer-Causal + ATTN_FS (K=32)	hungarian	0.0	1.0	0.0
Transformer-MLM + Sinkhorn loss	hungarian	0.0	1.0	0.0

5.4 PERMFILL：极小 anchor 改善长度外推

表 4 给出加载不同权重并在不同 n_{test} 下的 exact/valid（同值）结果摘要。

表 4: PERMFILL 的长度外推（exact/valid）。anchor 表示 masked span 内保留 k 个真值 token。

weights	anchor	n=24	n=28	n=32	n=36
permfill_n24_fs1_L12_seq256	off	1.000	0.000	0.000	0.000
permfill_anchor2_n24_fs1_L12_seq256	k=2	0.990	0.830	0.120	0.000
permfill_anchor2_n32_fs1_L12_seq256	k=2	1.000	1.000	1.000	0.935

5.5 4×4 数独：Future-Seed 推迟相变点

表 5 给出 solve-rate 随 holes 增长的变化。Future-Seed 在 holes=10 与 holes=12 仍保持较高 solve-rate，而 FS=0 很快塌陷至接近 0。

表 5: 4×4 数独 in-place 填空：solve-rate 相变曲线（trials=2000）。FS=0 使用 L4/e128；FS=1 使用 L12/e128。

holes	FS=0 solve	FS=1 solve
4	0.3530	1.0000
6	0.1665	1.0000
8	0.0520	1.0000
10	0.0095	0.9510
12	0.0000	0.5510
14	0.0000	0.0000

5.6 数独软约束正则：提升 holes=12，但 holes=14 仍困难

为缓解冲突解，我们加入一个简单的软约束：对每行/列/宫，要求“每个数字的概率计数之和接近 1”。在 holes $\in [4, 14]$ 的课程训练下，加入 $\lambda = 0.1$ 的一致性正则能显著提升 holes=12 的 solve-rate（约 0.48→0.59），但 holes=14 仍然难以解出（表 6）。

表 6: 一致性正则的影响（FS=1，L12/e128，trials=2000）。

setting	holes=10	holes=12	holes=14
$\lambda = 0.0$	0.9385	0.4795	0.0000
$\lambda = 0.1$	0.9515	0.5855	0.0000

5.7 负结果

我们观察到：

- 仅训练极少参数（`FS_MASK_ONLY=1`）无法学习数独任务（solve-rate ≈ 0）。
- 将数独 mask 固定为 prefix（更强“用未来”压力）在当前设定下训练不稳定（solve-rate ≈ 0）。这提示仅靠单次前向的 argmax 填充可能不足，可能需要迭代式推理协议。

6 讨论与后续工作

本文结果跨任务一致指向同一解释：Future-Seed 通过跨层回灌提供了一个额外的“全局约束记账/传播通道”，使得左侧 mask 的预测能够更稳定地依赖右侧信息与全局一致性。在置换类任务（KVSORT/PERM-FILL）中，anchor 进一步提供了极轻量的对齐锚点，从而显著改善长度外推。

后续可沿两条路径推进：

- **推理协议**：迭代填充（mask-predict / refinement）在 KVSORT 上不足以修复排序失败；后续可在更强约束（更大数独/结构文本）下系统比较“并行一次性填空”与“迭代修复”的 trade-off。
- **更强基线**：我们已测试 Hungarian 解码与 Sinkhorn 置换损失；后续可补充 pointer/ILP 等结构化模型以进一步区分“建模”与“解码”因素。
- **任务扩展**：扩展到更真实的结构修复（JSON/括号/一致性编辑）与更大规模的置换修复（key-value 排序 + mask + 外推）。

复现说明

代码与脚本基于 `git@github.com:yanghu819/future-seed.git`。主要入口为 `rwkv_diff_future_seed.py`。

通过环境变量启用：

`RIGHTCOPY_TASK` / `CONSTR_TASK` / `KVSORT_TASK`

`PERMFILL_TASK` / `SUDOKU_TASK`

并设置 `FUTURE_SEED`。

数独复现脚本见 `rwkv-diff-future-seed/run_sudoku.sh`。