字符模式 SAC 的工程实现与数学化描述v3.0.0

- 作者：GaoZheng

- 日期：2025-09-27
- 版本：v1.0.0

摘要

在 v2.0.0 基于“长度集合 U 的可变后缀命中”基础上，v3.0.0 将“向前拓扑命中”从单一词扩展为“拓扑词包命中”（可配置的一组词/短语，支持非交换的专有词组），并形式化为“拓扑词包算子”；同时将“向后拓扑”从单字符扩展为“迭代多字符预测”，定义为“多字符迭代算子”。这两类算子以统一接口接入合规模块与奖励记录，兼容 v1/v2 的行为，并通过配置文件灵活开关与调参，便于在产线场景下做可审计、可回放的策略治理。

- 向前拓扑命中：由“单词”升级为“词包命中”，可配置词/短语集合，支持非交换短语（顺序敏感）。
- 向后拓扑扩展：由“1 字 bigram 扩展”升级为“多字符迭代预测”，步数与停止条件可配置。
- 算子化接口：引入“拓扑词包算子”与“多字符迭代算子”，提供统一的伪代码与复杂度评估。
- 配置化落地：新增 `topology_word_packs`、`forward_pack_match.*`、`backward_iter.*` 配置段，默认兼容 v2 行为。

关键词：拓扑词包；非交换短语；多字符迭代预测；可配置；SAC；字符模式

---

1. 形式化与符号

- 词表 Catalog（与 v2 相同）

  $\mathcal{C} = \texttt{Catalog} = \texttt{chinese\_name\_frequency\_word.json} \cup \texttt{chinese\_frequency\_word.json}.$

- 长度集合（与 v2 相同）

  $$U = \texttt{union.lengths} \subset \mathbb{N}, \quad \text{来自 } \texttt{data/word\_length\_sets.json}.$$

- 词包族（新增）
  - 给定一组“词包”集合 $\mathfrak{P} = \{P_1, \ldots, P_M\}$，每个 P_i 为若干“词/短语”的有序集合：

    $$P_i = \{\omega_{i,1}, \ldots, \omega_{i,k_i}\}, \quad \omega_{i,j} \in \Sigma^+.$$

  - 非交换短语指顺序敏感的 token 序列（如专有名词、术语搭配）；简记检查谓词 $\mathrm{hit}(s, \omega)$ 为“ω 是否在 s 的指定作用域内命中（如尾部/子串）”。

- 文本片段
  - 目标章节 χ_t，上一轮摘要 prev_t，拼接源 $\mathrm{source}_t = \mathrm{prev}_t \oplus \chi_t$。
  - 行为前缀轨迹 q：由首字符与未来候选字符逐步拼接得到。

---

2. 拓扑词包算子（向前命中）

目的：在 $s = \chi_t \oplus q$ 的尾部（或子串）上，基于 U 与词包族 $\mathfrak{P}$，寻找“最长命中”的词或短语，返回命中项与更新后的 s，用于合规则与奖励记录。

定义（尾部作用域）

$$\exists L \in U \cap [1..|s|],\ \exists P \in \mathfrak{P},\ \exists \omega \in P,\ \text{s.t. } \mathrm{tail}(s, L) = \omega.$$

伪代码

```
function FORWARD_PACK_HIT(chapter, q, future_chars, U, Packs, scope="tail"):
    s = chapter + q
    best = tail(s, min(2, len(s)))  # 兼容日志展示
    for ch in future_chars:
        s += ch
        if scope == "tail":
            candidates = [tail(s, L) for L in sort_desc(U ∩ {1..len(s)})]
        else:  # substring
            candidates = all_substrings_bounded_by_U(s, U)
        for seg in candidates:
            if is_cjk(seg) and PACK_HIT(seg, Packs):
                return seg, s
    return best, s

function PACK_HIT(seg, Packs):
    for P in Packs:
        for ω in P:
            if match_phrase(seg, ω):  # 顺序敏感（非交换）
                return True
    return False
```

匹配语义

- `match_phrase` 默认精确匹配；可在配置中切换到“最长可用/别名归一/大小写/全半角”规则。
- `scope` 支持 `tail` 与 `substring`；生产默认 `tail` 以对齐 v2 习惯与复杂度上界。

复杂度（单步）

$$\mathcal{O}(|U|\cdot\bar{k}) \quad \text{或} \quad \mathcal{O}(|\mathrm{sub}(s,U)|\cdot\bar{k}) \text{ (子串作用域)},$$

其中 $\bar{k}$ 为词包内平均候选数，`sub(s,U)` 为受 U 约束的候选子串集。

---

3. 多字符迭代算子（向后扩展）

目的：将 v2 的单字符 bigram 拓展，改为一次“最多 K 字”的迭代扩展，按步采样或贪心解码，直到命中/封顶/早停，输出轨迹 q 与命中项（若有）。

定义（后缀作用域）

$$\exists L \in U \cap [1..|q|],\ \mathrm{tail}(q,L) \in \mathcal{H},\ \mathcal{H} \subseteq \Sigma^{+},$$

其中 $\mathcal{H}$ 可取为 Catalog、或拓扑词包的并集，或其它约束词表。

伪代码

```
function ITER_BACKWARD_EXTEND(initial_char, sample_next, U, H, K_max, stop_on_hit=True):
    q = dedup_head_repeat(initial_char)
    for step in range(K_max - 1):  # 已有 1 字，最多补 K_max-1 字
        ch = sample_next(q)  # 来自策略/温度/Top-p 等
        q += ch
        for L in sort_desc(U ∩ {1..len(q)}):
            seg = tail(q, L)
            if is_cjk(seg) and in_set(seg, H):
                if stop_on_hit: return q, seg
                else: break  # 命中但继续累计，取更长命中
    # 未命中则回退为“可读性最强”的后缀（如最长 CJK 尾部）
    return q, longest_cjk_tail(q, max(U))
```

注：`sample_next` 可为贪心、温度采样或 Top-p，仍服从合规模块的掩码约束。

---

4. 配置项（新增）

建议在 `res/config.json` / `config_template.json` 中新增：

```
{
  "forward_pack_match": {
    "enabled": true,
    "scope": "tail",                // tail | substring
    "packs_path": "data/topology_word_packs.json", // 若为空则退化为 v2（单词）
    "normalize": { "alias": true, "casefold": true, "fullwidth": true }
  },
  "backward_iter": {
    "enabled": true,
    "k_max": 4,                     // 单步最多扩展的字符数（含首字 -> 轨迹总长 ≤ k_max）
    "stop_on_hit": true
  }
}
```

并新增数据文件示例 `data/topology_word_packs.json`：

```
{
  "packs": [
    { "id": "药物别名", "phrases": ["奥司他韦", "达菲"], "noncommutative": true },
    { "id": "疾病短语", "phrases": ["流行性感冒", "甲流"], "noncommutative": true }
  ]
}
```

兼容性：当 `packs_path` 为空或解析失败时，前向命中退化为 v2 的 Catalog 单词命中；`k_max=2, stop_on_hit=true` 时，后向扩展近似 v2 bigram。

---

5. 与合规模块/奖励的接口

- 合规模块 `_mask_logits` 仅需将“基于 H 的命中检查”作为可选诊断信号，不改变禁用/允许集的语义。
- 奖励日志：保留 `lexical_bigram_bonus` 字段名，但记录更一般的：`lexical_topo_bonus`、`topo_hit_phrase`、`topo_pack_id`、`iter_k`。
- 评分延续 v2：

$$\delta_t = \begin{cases} 1.0, & \text{命中拓扑词（包/单词）,} \\ 0.5, & \text{仅 1 字对齐目标字符,} \\ 0, & \text{否则.} \end{cases}$$

$$B_t^{\text{char}} = B_t + 0.5\,\chi_t^{\text{soft}} + \delta_t, \quad \Delta_t^{\text{char}} = \Delta_t + 0.25\,\chi_t^{\text{soft}}.$$

---

6. 复杂度与实现建议

- 复杂度：词包匹配令 $\bar{k}$ 增大，建议为 `packs` 建立 Trie/AC 自动机或散列桶以维持均摊 $\mathcal{O}(1)$ 命中；子串作用域慎用。
- 数据预处理：将常用短语与别名统一到 `packs`，并在规范化流程中做别名归一（与 `normalize` 同步）。
- 可观测性：在 HTML/CSV 可视化中新增列：`pack_hit`、`pack_id`、`hit_len`、`iter_len`，便于离线分析。

---

7. 实现映射（代码位置建议）

- `src/character_sac_trainer.py`
  - 新增/改造：
    - `TopologyWordPacks` 装载与规范化（读取 `packs_path`）。
    - `PACK_HIT` 与 `FORWARD_PACK_HIT`；`ITER_BACKWARD_EXTEND`；
    - 日志与奖励记录字段扩展。
  - 兼容：当未启用新特性时，路径回落到 v2 的 `FORWARD_EXTEND_BIGRAM`/后缀命中逻辑。

---

8. 回滚与 A/B

- 参数回滚：`enabled=false` 即回到 v2 行为；`k_max=2` 近似 bigram；`packs_path` 置空即退化为单词。
- A/B 方案：对 `packs` 的不同定义做流量切分，观察 `lexical_topo_bonus` 分布与收敛稳定性差异。
