字符模式 SAC 的工程实现与数学化描述v4.0.0

- 作者：GaoZheng

- 日期：2025-09-27
- 版本：v1.0.0

摘要

v4.0.0 在“词包语义 + 前后对称拓扑（v3.0.1）”的基础上，提出“摘要 → 迭代摘要 → 摘要的摘要 → 摘要展开”的端到端生成框架：先对长输入形成短摘要，再以“分段+回放”的方式进行迭代摘要（累积对齐），用“摘要的摘要”形成全局纲要，最后通过“摘要展开”将纲要逐段充实为高一致性的长上下文回答。该流程将词包作为一等公民参与命中/检索/展开，统一了控制旋钮与可观测指标，并给出可回滚的配置接口与评测标准。

- 语义目标：以“词包（非交换短语）”做为抽象锚点，贯穿摘要/迭代/纲要与展开四阶段；
- 算子统一：迭代与展开均可在 `hit_mode ∈ {catalog, packs, union}` 下运行；
- 工程可用：给出伪代码与配置接口，保持与 v3.0.1 的向后兼容（配置回退即可恢复 v3 语义）。

---

1. 形式化与对象

- 词表与长度集合（承接 v2/v3）：

$$\mathcal{C} = \mathrm{Catalog}, \quad U = \mathrm{union.lengths} \subseteq \mathbb{N}.$$

- 词包族（承接 v3）：

$$\mathfrak{P} = \{P_i\}_{i=1}^{M}, \quad P_i = \{\omega_{i,j}\}_{j=1}^{k_i},\ \omega_{i,j} \in \Sigma^{+}.$$

- 输入分段：给定长文本 X，分段为 $\{x_1, \ldots, x_T\}$；
- 摘要算子 Σ：$\Sigma(x;\theta)$ 生成摘要片段；
- 迭代摘要算子 Σ_{iter}：以“上次摘要 + 当前片段”为输入累积生成；
- 纲要算子 Σ_{meta}：对所有摘要段的再摘要，得到“摘要的摘要”（纲要）；
- 展开算子 Φ_{expand}：将纲要逐条扩写为长上下文回答 Y。

---

2. 核心算子与伪代码

2.1 初始摘要（分段）

```
function SEGMENT_SUMMARIZE(chunks, summarizer, packs, U):
    S = []
    for x in chunks:
        s = summarizer(x)                # 基础摘要
        s = PACK_ALIGNED(s, packs, U)    # 基于词包的锚定与对齐（可记录命中）
        S.append(s)
    return S  # [s_1, ..., s_T]
```

2.2 迭代摘要（回放对齐）

```
function ITERATIVE_SUMMARIZE(chunks, summarizer, packs, U):
    prev = ""
    I = []
    for x in chunks:
        source = prev + x                # 上一轮摘要 + 当前片段
        s = summarizer(source)
        s = PACK_ALIGNED(s, packs, U)    # 词包校正/命中记录
        I.append(s)
        prev = s                         # 迭代回放
    return I   # [i_1, ..., i_T]
```

说明：`PACK_ALIGNED` 不改变语义内容，只做术语归一/别名对齐/命中记录，用于可观测与后续检索约束。

2.3 摘要的摘要（形成全局纲要）

```
function META_SUMMARY(summaries, meta_summarizer, packs, U):
    concat = JOIN(summaries, sep="\n")
    g = meta_summarizer(concat)          # 形成纲要（摘要的摘要）
    return PACK_ALIGNED(g, packs, U)
```

2.4 摘要展开（纲要 → 长上下文回答）

```
function EXPAND_SUMMARY_TO_LONG_CONTEXT(guide, retriever, generator, packs, U, budget):
    # guide: 纲要 (bullet 或 numbered)
    Y = []
    for item in PARSE_GUIDE(guide):
        # 检索：以词包为锚点的语义检索（可 union catalog）
        ctx = retriever(item, mode="union", packs=packs)
        # 生成：受 packs/U 的术语约束与长度预算控制
        y = generator(item, ctx, constraints={"packs": packs, "U": U}, budget=budget)
        Y.append(y)
    return COALESCE(Y)  # 长上下文回答
```

可选：生成时对关键术语（包）进行显式地标注与高亮，便于审计与对齐。

---

3. 配置接口（建议）

```
{
  "summary": {
    "segmenter": "by_paragraph",            // 片段切分
    "summarizer": "gpt-mini-sum",           // 初始摘要模型
    "meta_summarizer": "gpt-mini-meta"      // 纲要模型
  },
  "iterative": {
    "enabled": true,
    "summarizer": "gpt-mini-iter",
    "pack_align": { "normalize": {"alias": true, "fullwidth": true} }
  },
  "expand": {
    "retriever": "bm25+pack-union",
    "generator": "gpt-mini-expand",
    "budget": { "tokens": 2048 },
    "constraints": { "hit_mode": "union" }   // catalog | packs | union
  },
  "packs": {
    "path": "data/topology_word_packs.json",
    "hit_mode": "union"
  }
}
```

与 v3.0.1 兼容：将 `hit_mode` 设置为 `catalog` 即可退化为“单词驱动”的摘要/展开路径。

---

4. 评价与奖励（建议）

- 摘要阶段：覆盖/一致/去冗（Coverage/Consistency/Redundancy），可复用 $\mathcal{N}_\gamma$ 变换；
- 迭代阶段：逐步一致性（`len_ratio/similarity/pack_hit_rate`）与稳定度（方差、熵）；
- 纲要阶段：结构完整性（要点召回、包覆盖率）与可展开性（每条可检索到足够证据的比例）；
- 展开阶段：

$$R = \alpha\,\mathrm{FCT}(Y,X) + \beta\,\mathrm{CoT}(Y) + \gamma\,\mathrm{PackHit}(Y) - \lambda\,\mathrm{Cost},$$

其中 FCT 为事实一致性、CoT 为思维链完整度、$\mathrm{PackHit}$ 为词包召回比例，Cost 为推理/检索成本。

---

5. 可观测性与日志

- 摘要/迭代：`pack_hit_rate`、`alias_normalized_terms`、`len_ratio`、`similarity`；
- 纲要：`guide_items`、`pack_coverage`；
- 展开：`retrieval_hits`、`evidence_ids`、`generated_spans`、`violations`（如越权扩写）。

---

6. 上线与回滚

- 灰度方案：分流 `hit_mode ∈ {catalog, packs, union}` 与不同 `budget.tokens`；
- 回滚路径：`expand.constraints.hit_mode="catalog"` + 关闭迭代摘要（使用分段摘要）；
- 故障保护：检索失败时降级为仅根据纲要生成，并在日志中标注“无检索”。

---

7. 与 v3.0.1 的关系

- 继承：词包作为一等公民与前后对称拓扑；
- 提升：从“命中”走向“抽象-纲要-展开”的生成闭环；
- 兼容：所有新配置均可回退为 v3 行为，以确保平滑迁移。

2′. v4.0.0 构造（压缩迭代 → 扩展迭代 → 长上下文）

为与“词包语义”完全对齐，这里用“正文分段词包/摘要词包”的序列化方式给出完整构造流程。记：

- 正文分段的词包序列为 $\{\mathsf{P}_t\}_{t=0}^{T}$，其中 P_t 表示第 t 段正文的“正文词包”；
- 迭代产生的摘要词包序列为 $\{\mathsf{S}_t\}_{t\geq 0}$，其中 S_t 表示“摘要词包”；
- 用户本次提问映射成词包 $\mathsf{P}_{\mathrm{ask}}$（可经 `PACK_ALIGNED` 归一）。

步骤一：压缩迭代（Compressive Loop）

1. 初始摘要词包预测：

$$\mathsf{S}_0 = \Sigma_{\text{seg}\to\text{pack}}(\mathsf{P}_0) \quad \text{（对第 0 段正文词包进行摘要，得到摘要词包）.}$$

2. 迭代压缩：对 $t = 1, 2, \ldots, n$，

$$\mathsf{S}_t = \Sigma_{\text{iter}\to\text{pack}}\big(\mathsf{S}_{t-1} \oplus \mathsf{P}_t\big),$$

其中 $\oplus$ 表示词包级“对齐+累积”组合（不改变词义，仅做别名归一与去冗）。

3. 与用户提问融合：

$$\mathsf{S}_{n+1} = \Sigma_{\text{iter}\to\text{pack}}\big(\mathsf{S}_n \oplus \mathsf{P}_{\text{ask}}\big).$$

步骤二：扩展迭代（Expansion Loop）

4. 从 S_{n+1} 开始做扩展生成，使每一步既产出“正文词包”又产出“下一步摘要词包”：

$$\big(\mathsf{P}_{n+1},\ \mathsf{S}_{n+2}\big) = \Phi_{\text{expand}}\big(\mathsf{S}_{n+1}\big), \qquad \big(\mathsf{P}_{n+2},\ \mathsf{S}_{n+3}\big) = \Phi_{\text{expand}}\big(\mathsf{S}_{n+2}\big),\ \ldots$$

5. 经过 m 步扩展，得到正文词包集合 $\{\mathsf{P}_{n+1}, \ldots, \mathsf{P}_{n+m}\}$。

步骤三：长上下文组装与文法风格补全

6. 将正文词包序列合并并检索证据片段构造上下文：

$$\mathsf{P}_{\text{long}} = \bigoplus_{k=1}^{m} \mathsf{P}_{n+k}, \qquad \mathsf{Ctx} = \text{Retrieve}(\mathsf{P}_{\text{long}};\ \text{hit\_mode}).$$

7. 文法/风格补全，得到长上下文回答 Y：

$$Y = \text{StyleComplete}\big(\mathsf{P}_{\text{long}},\ \mathsf{Ctx};\ \text{register} \in \{\text{tech}, \text{plain}\}\big),$$

其中 `StyleComplete` 负责术语一致、句法流畅与语篇连贯的最终润色（可控温度/长度预算）。

伪代码（贯通流程）

```
function V4_CONSTRUCT(chunks, ask, packs, U, cfg):
    # 压缩迭代
    P = [PACK_ALIGNED(x, packs, U) for x in chunks]       # 正文词包序列
    S0 = summarize_pack(P[0])
    S0 = PACK_ALIGNED(S0, packs, U)
    S = [S0]
    for t in range(1, n+1):
        S_t = summarize_iter_pack(S[-1] ⊕ P[t])
        S_t = PACK_ALIGNED(S_t, packs, U)
        S.append(S_t)
    S_ask = summarize_iter_pack(S[-1] ⊕ PACK_ALIGNED(ask, packs, U))
    S_ask = PACK_ALIGNED(S_ask, packs, U)

    # 扩展迭代
    body_packs = []
    cur = S_ask
    for j in range(m):
        P_next, S_next = expand_from_pack(cur, retriever=cfg.retriever, hit_mode=cfg.hit_mode)
        body_packs.append(PACK_ALIGNED(P_next, packs, U))
        cur = PACK_ALIGNED(S_next, packs, U)

    # 组装与风格补全
    P_long = reduce_oplus(body_packs)
    ctx = retrieve_context(P_long, mode=cfg.hit_mode)
    Y = style_complete(P_long, ctx, register=cfg.register, budget=cfg.budget)
    return Y
```

说明：以上每一步均保留“词包命中/命中长度/步数”等可观测字段，便于复盘与 A/B 评测。
