史记知识库构造管线技能手册

SKILL 00-09 技术规范与实践指南

作者：鲍捷

电子邮件：baojie@gmail.com
微信：baojiexigua
GitHub: @baojie

文档在线目录：
github.com/baojie/shiji-kb/tree/main/skills

生成日期：2026-03-19

完整知识库构造管线 · 从文本到知识图谱

目录

SKILL 04: 事件构建 — 从标注文本到结构化事件

SKILL 04a: 古籍历史事件识别

SKILL 04b: 十表事件处理

SKILL 04c: 事件年代推断

SKILL 04d: 事件年代审查管线

SKILL 04e: 古籍年份消歧 — 从在位纪年到公元绝对年

SKILL 04f: 动词标注

SKILL 05: 关系构建 — 从事件与实体到关系网络

SKILL 05a: 事件关系发现 — 从孤立事件到关系网络

SKILL 05b: 实体关系构建 — 人物/地点/制度间的结构化关系

SKILL 05c: 人物关系构建 — 从共现到家谱的多层次人物关系网络

SKILL 05d: 事实发现 — 原子事实三元组抽取

SKILL 05e: 战争事件识别与知识融合

SKILL 06: 本体构建 — 从实体实例到分类本体

SKILL 06a: 实体到本体管线 — 从标注文本到 RDF 分类本体

SKILL 07: 逻辑推理 — 矛盾检测与洞见挖掘

SKILL 07a: 人物生卒年推断 — 区间估算

SKILL 07b: 姓氏推理 — 先秦人物姓与氏的区分与标注

SKILL 07c: 反常推理 — 不符合常识的事实检测与解读

SKILL 08: SKU构造 — 从结构化知识到知识单元

SKILL 09: 应用构造 — 从知识到产品

SKILL 09a: 认知辅助阅读器 — 语法高亮 + 结构语义化

SKILL 09b: 知识库游戏化 — 用史记知识库设计游戏

SKILL 00: 管线总览 — 从原始文本到知识应用

> 本文件是总控文档，只描述管线工序顺序和依赖关系，具体方法见各阶段SKILL。

一、九大阶段

```
原始文本 (archive/)
    │
    ▼ 01 校勘
定本 (docs/original_text/)
    │
    ▼ 02 结构分析
结构化文本 (段落编号 + 区块 + 注释层)
    │
    ▼ 03 实体构建
标注文本 + 实体索引 (chapter_md/*.tagged.md)
    │
    ├──▼ 04 事件构建
    │    事件索引 + 公元纪年 (kg/events/)
    │
    ├──▼ 05 关系构建
    │    关系网络 + SPO三元组 + 原子事实 (kg/relations/, kg/facts/)
    │
    ├──▼ 06 本体构建
    │    OWL/RDF 分类本体 (kg/rdf/)
    │
    ▼ 07 逻辑推理
推断知识 (矛盾检测 / 生卒年推断 / 姓氏推理)
    │
    ▼ 08 SKU构造
```

```
知识单元（Factual / Procedural / Relational）
    |
    ▼ 09 应用构造
用户产品（时间线 / 问答 / 阅读器）
```

二、核心原则

1. **校勘先行** — 底本文字有误，后续一切工作建立在错误基础上
2. **先结构后标注** — 段落编号完成后再做实体标注，避免频繁返工
3. **保持原貌** — 校勘/切分/标注都不改变原文字符（异体字、标点层级等）
4. **LLM + 规则混合** — 实体标注和事件提取用LLM，关系发现和纪年标注用自动规则+LLM
5. **迭代验证** — 每个工序都有对应的lint/validate工具，形成闭环

三、元技能（Meta-Skills）

元技能是对所有工序通用的抽象方法论，与具体数据类型无关，指导工序设计与执行。

3.0 元元技能（所有元技能的根源）

编号	文件	内容
00-M00	00-META-00_好东西都是总结出来的.md	**元技能的元技能**，OTF+JIT+Bootstrap方法论，贯穿所有工序

3.1 核心方法论（01-06）

编号	文件	内容
00-M01	00-META-01_反思.md	Agent驱动的质量审查通用方法论（按章/按类型/多轮收敛）

编号	文件	内容
00-M02	00-META-02_迭代工作流.md	MVP→生产的系统化迭代（2-5轮迭代方法论）
00-M03	00-META-03_冷启动.md	新工序启动方法（手工样本→MVP规则→试点→扩展）
00-M04	00-META-04_柳叶刀方法.md	精细分解与混合解决方案（LLM+规则+代码分工）
00-M05	00-META-05_知识作为上下文压缩.md	KG价值论证（10x-10000x压缩比，成本线性收益平方）
00-M06	00-META-06_SKILL优化与演化.md	SKILL从日志总结、持续抽象、版本迭代、渐进载入

3.2 工序相关（07-13，按知识抽象层级）

编号	文件	内容
00-M07	00-META-07_可读性.md	人类优先的数据格式设计（Token/Markdown/Git友好）
00-M08	00-META-08_标注体系设计.md	Token标注体系设计原则（v1.0→v2.5演化/18类实体）
00-M09	00-META-09_Agent提示词工程.md	SKILL文档驱动的提示词设计（五层金字塔结构）
00-M10	00-META-10_质量控制.md	Lint/Validate工具链设计（五层质检体系）
00-M11	00-META-11_数据体感培养.md	直接观察数据的十层体系（统计无法替代观察）

编号	文件	内容
00-M12	00-META-12_数据融合.md	多源数据对齐/消歧/去重（字符串→实体→事件→SKU）
00-M13	00-META-13_技能迁移.md	SKILL跨项目迁移方法论（史记→汉书→通鉴，试点反思）

四、SKILL 文件索引

阶段	编号	文件	内容
总览	00	SKILL_00_管线总览.md	本文件
校勘	01	SKILL_01_古籍校勘.md	底本校正，生成定本
结构分析	02	SKILL_02_结构分析.md	阶段总览
	02a	SKILL_02a_章节切分与编号.md	段落编号 + 小节划分 + 对话拆分
	02b	SKILL_02b_区块与韵文处理.md	太史公曰/赞/策论 + 韵文检测
	02c	SKILL_02c_三家注标注.md	裴骃/司马贞/张守节 注释层
	02d	SKILL_02d_结构语义分析.md	句间/段间/章间语义关系 + 注释对齐 + 排版映射
	02e	SKILL_02e_词法分析.md	字级词性标注、遗漏实体发现

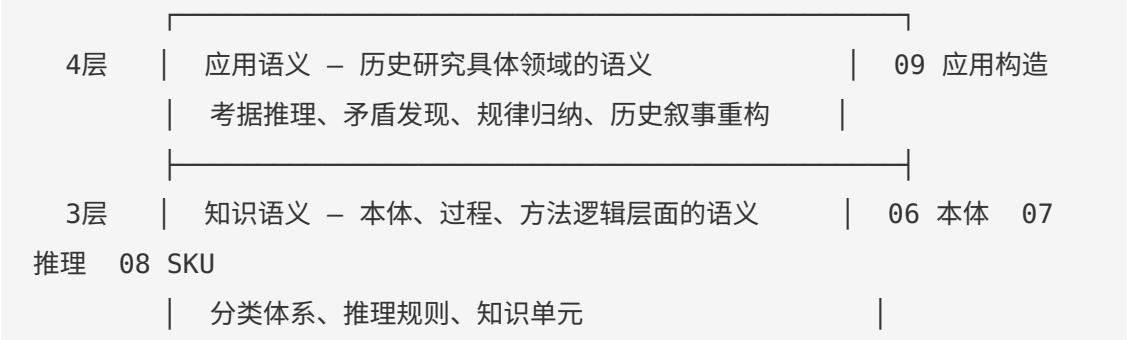
阶段	编号	文件	内容
	02f	SKILL_02f_文本统计.md	字数/句长/词频/实体密度/五体对比
	02g	SKILL_02g_表格结构语义分析.md	史记十表维度定义、关系挖掘
	02h	SKILL_02h_词表构建.md	19类专项词表识别与结构化
实体构建	03	SKILL_03_实体构建.md	阶段总览
	03a	SKILL_03a_实体标注.md	18类实体NER标注
	03b	SKILL_03b_实体消歧.md	同名异人/异名同人消歧
	03c	SKILL_03c_实体标注反思.md	按章/按类型多轮反思审查，系统性误标修正
	03d	SKILL_03d_渲染与发布.md	渲染与发布（衔接，详见SKILL 09a）
事件构建	04	SKILL_04_事件构建.md	阶段总览
	04a	SKILL_04a_事件识别.md	事件定义/Schema/类型/提取
	04b	SKILL_04b_十表事件处理.md	十表（013-022）专用补充
	04c	SKILL_04c_事件年代推断.md	分层纪年解析

阶段	编号	文件	内容
	04d	SKILL_04d_事件年代审查.md	Agent反思审查工作流
	04e	SKILL_04e_年份消歧.md	年号/纪年消歧
	04f	SKILL_04f_战争事件识别.md	战争事件专项提取（736个）
关系构建	05	SKILL_05_关系构建.md	阶段总览（7652条关系，9种类型）
	05a	SKILL_05a_事件关系发现.md	9种关系类型/自动+LLM发现
	05b	SKILL_05b_实体关系构建.md	地点/制度/事物关系（非人物实体关系）+ SPO三元组
	05c	SKILL_05c_人物关系构建.md	人物共现→家族推理→政治社会关系（三层递进）
	05d	SKILL_05d_事实发现.md	原子事实三元组（~10万，SPO全汉字，含ctx上下文）
本体构建	06	SKILL_06_本体构建.md	阶段总览
	06a	SKILL_06a_实体到本体.md	实体词表 → 分类树 → OWL/RDF
逻辑推理	07	SKILL_07_逻辑推理.md	矛盾检测 + 洞见挖掘 + 本体/时间/关系推理
	07a		

阶段	编号	文件	内容
		SKILL_07a_人物生卒年推断.md	人物生卒年区间推断（年龄/在位年/亲属约束）
	07b	SKILL_07b_姓氏推理.md	先秦人物姓/氏区分，多轮迭代推理（R1–R7）
	07c	SKILL_07c_反常推理.md	单条事实违反常识/制度/规律的异常点检测
SKU构造	08	SKILL_08_SKU构造.md	Factual/Procedural/Relational 知识单元
应用构造	09	SKILL_09_应用构造.md	阶段总览
	09a	SKILL_09a_认知辅助阅读器.md	语法高亮渲染 + 实体配色 + 段落锚点 + 发布
	09b	SKILL_09b_知识库游戏化.md	知识库→游戏：SKU→技能卡、事件→剧情、实体→角色

四、四层语义模型

本项目的研究方法建立在四层语义递进模型上，每一层在前一层的基础上构建更高级的语义理解：



```
          ├──────────────────────────────────────────────┤
  2层     │  图谱语义 — 实体、关系、事实层面的语义          │  03 实体  04 事件  05 关系
          │  人物/地点/事件识别、消歧、关系发现、原子事实   │
          ├──────────────────────────────────────────────┤
  1层     │  结构语义 — 篇章、段落、句子层面的语义结构      │  01 校勘  02 结构分析
          │  切分、编号、因果/并列/递进、注释对齐           │
          └──────────────────────────────────────────────┘
```

层次	关注对象	核心问题	对应工序
结构语义	篇章、段落、句子	文本怎么组织的？句子间什么关系？	01校勘、02结构分析（02a-02d）
图谱语义	实体、事件、关系、事实	谁做了什么？在哪里？什么时候？原子事实是什么？	03实体、04事件、05关系
知识语义	本体、规则、方法	这些实体怎么分类？事件遵循什么规律？	06本体、07推理、08 SKU
应用语义	考据、叙事、发现	历史真相是什么？有哪些矛盾？能发现什么规律？	09应用构造

每一层都是下一层的前提：没有结构语义就无法准确提取图谱语义，没有图谱语义就无法构建知识语义，没有知识语义就无法支撑应用语义。

五、方法论

人机协作分工

角色	职责
人（决策者）	定义愿景、设计标注体系、审美判断、纠正语义错误、质量把控

角色	职责
AI（执行者）	大规模NER、代码开发、批量文件操作、渲染管线、文档生成

关键设计决策

1. **用大模型做NER** — 古文多义词极多，关键词匹配无法处理
2. **内容与样式分离** — Token标记系统让原文可读，渲染交给CSS
3. **先做完再做好** — 先快速覆盖全部章节，再迭代提升质量

六、扩展到其他古籍

本管线可直接应用于二十四史及其他大规模古籍，需适配：实体类型、体裁分类、提示词、别名规则、年份体系。

古籍	字数	篇章数	预估Token
史记	57万	130	5.7亿
汉书	74万	120	~7亿
资治通鉴	300万	294	~25亿

SKILL 01: 古籍校勘 — 从底本到定本的文字审校

> 校勘是所有工序（切分、标注、事件提取、知识图谱化）的前提。底本文字有误，后续一切工作都建立在错误基础上。

一、工序位置

```
docs/original_text/*.txt      底本（网络来源，可能有错）
        ↓ 校勘（本SKILL）
docs/original_text/*.txt      定本（校正后）
        ↓ 切分+编号（SKILL_古籍章节切分与编号）
chapter_md/*.tagged.md        标注文本
        ↓ 实体标注（SKILL_03a_实体标注）
        ↓ 事件提取 ...
```

核心原则： 校勘只改原文txt，不改标注文件。标注文件应忠实反映定本。

二、校勘来源

优先级	来源	说明
1	中华书局点校本（1959/2013）	学术界公认权威底本，以金陵书局本为底本校勘
2	武英殿二十四史本	清代官刻本，校勘精善

优先级	来源	说明
3	百衲本二十四史	商务印书馆影印宋元善本
4	数字化资源	维基文库·史記、国学网·史记、ctext.org、国学大师等可作辅助参照

三、常见底本错误类型

3.1 网站爬取垃圾

- 整章内容被替换为网站索引（如022章"中秋/怀古/爱国"）
- 混入HTML标签、导航文字

3.2 缺字与乱码

- 原文缺字显示为 `？`（如 `季?` → `季历`）
- PUA私用区字符（如 `{单心}` → `惮`）

3.3 衍文与脱文

- 衍文（多余重复）：`蹇叔止蹇叔止臣` → `蹇叔止臣`
- 脱文（遗漏）：`马` → `马鷙`

3.4 现代编辑附加

- 公元纪年数字（206、205…）不属于原文
- `公元前` 表头

3.5 异体字

- 原文使用古字/异体字时**保持原貌**，不做繁简转换
- 例：`太山` 不改为 `泰山`，`後` 不改为 `后`，`彊` 不改为 `强`

四、校勘原则

1. **保持原貌** — 异体字、古字、通假字保留，不做现代化
2. **有据可查** — 每处校改应有版本依据（ctext或其他权威版本）
3. **标注不改原文** — 标注过程中发现原文有误，应修正txt底本，而非在标注文件中悄悄改
4. **分段不改字** — 切分段落时只改换行位置，不改任何文字（包括引号层级 `"` / `'`、标点 `；` / `。` 等）

五、校勘工具

5.1 lint完整性检查

```
python scripts/lint_text_integrity.py          # 全部130章
python scripts/lint_text_integrity.py 005 006  # 指定章节
```

5.2 白名单

已确认的正确校勘记录在 `scripts/lint_whitelist.txt`，lint时自动跳过。

格式：`章节号<TAB>原文片段<TAB>标注片段<TAB>说明`

5.3 校勘参照

以权威刊本或可靠数字化资源为参照，核对存疑文字。

六、已完成校勘

章节	校勘内容	状态
001	双重迁移修复 + `唯` 补回	√
004	`季?→季历` 、 `天王→天子` 、 `{单心}→惮`	√ 白名单
005	重复句号、衍文去重、 `饹→奔` 、 `騺` 补入、 `虓→察` 、顿号恢复	√ 部分白名单
006	`；` / `。` 恢复、繁简恢复（78处）、分段引号清理	√
022	垃圾内容替换为正确序言	√
十表	公元纪年删除、 `。` 恢复（430处）	√

七、字典关联

校勘过程中遇到的古字、异体字、通假字，应关联到字典资源，为读者提供查阅入口。

7.1 关联场景

场景	示例	字典用途
异体字	`後` / `后` 、 `彊` / `强`	查明字形演变，确认是否同字
通假字	`饹` 通 `奔` 、 `蚤` 通 `早`	查明通假关系，辅助校勘判断
古字	`埶`（势）、 `𨸏`（陀）	查明古今字对应
疑难字	PUA字符、罕见字	查明字义和读音

7.2 字典资源

字典	用途	访问方式
漢典	字义/字形/异体字查询	`https://www.zdic.net/hans/X`
教育部異體字字典	台湾教育部异体字权威	在线查询
說文解字	许慎原典	维基文库
康熙字典	字形收录最全	维基文库

7.3 渲染集成

在HTML渲染中，校勘异体字可添加字典链接tooltip：

```
<span class="variant-char" title="異體字：後，通行字：后">
  後
  <a href="https://www.zdic.net/hans/後" class="dict-link"
target="_blank">📖</a>
</span>
```

八、待办

- [ ] 系统性检查剩余542处lint差异，区分校勘/标注错误/格式差异
- [ ] 建立各章校勘记录

SKILL 02: 结构分析 — 从定本到结构化文本

> 将校勘后的定本拆分为段落编号的结构化文本，处理特殊区块（太史公曰/赞/韵文）和三家注释层。

核心设计决策：为什么用〖〗Token而不用XML/HTML标签

本项目的语义标注采用轻量CJK符号（`〖@人名〗`、`〖=地名〗`等），而非XML（`<person>人名</person>`）或HTML（`<span class="person">人名</span>`）。这是在**可读性**与**语义精确性**之间的刻意权衡。

方案对比

```
原文：黄帝者，少典之子，姓公孙，名曰轩辕。

XML标注：
<person>黄帝</person>者，<person>少典</person>之子，
姓<clan>公孙</clan>，名曰<person>轩辕</person>。

HTML标注：
<span class="person">黄帝</span>者，<span class="person">少典</span>之子，
姓<span class="dynasty">公孙</span>，名曰<span class="person">轩辕</span>。

Token标注：
〖@黄帝〗者，〖@少典〗之子，姓〖&公孙〗，名曰〖@轩辕〗。
```

为什么不用XML/HTML

维度	XML/HTML	〖〗Token
人眼可读性	标签淹没原文，标注后几乎无法直接阅读	标记轻量，标注后仍可流畅阅读原文
标注效率	打开/关闭标签冗长，LLM输出Token消耗大	单字符类型标记，LLM输出紧凑
git diff	标签噪音严重，差异难以review	diff清晰，改动一目了然
嵌套处理	允许任意嵌套，但解析复杂	拍平策略：`〖;安国君〗`（外层保留）
语义精确性	可附加任意属性（id/type/ref等）	仅类型标记，属性通过外挂JSON补全
编辑门槛	需要了解XML语法，标签易写错	选中文本加前后缀即可，直觉操作
格式冲突	与Markdown语法无冲突	v1.0的`$官职$`与Markdown math冲突（v2.0已解决）
渲染无关性	HTML标注与渲染耦合	同一份标注可渲染为HTML/PDF/LaTeX/JSON

关键洞察

1. **古文标注的首要矛盾是可读性**。130篇×57万字的标注文件，研究者需要频繁阅读和修改。XML标注后原文几乎不可读（标签字符数常超过原文字符数），这在实际工作中不可接受。

2. **语义精确性通过分层补全**。Token标记只承载"类型"这一维信息，更丰富的语义（消歧、别名、公元纪年）通过外挂JSON元数据层在渲染时注入，而非塞进标记本身：
   `源文件：〖@武王〗 ← 只标类型 消歧层：{"004": {"武王": "周武王"}} ← JSON`

`补全语义 别名层：{"周武王": ["武王", "发"]} ← JSON补全别名 渲染时：<a href="person.html#周武王"><span class="person" title="即周武王">武王</span></a>`

3. **v1→v2的教训验证了这一选择**。项目初期曾用HTML `<span style="color:...">` 标注，标注后原文完全不可读，被迫用 `convert_spans_to_simple.py` 批量回退为Token标记。

与TEI/XML标准的关系

学术数字人文项目（如TEI P5）通常采用XML标注。本项目的选择并非否定TEI，而是针对**AI Agent驱动的大规模标注**场景做的工程权衡：

- TEI适合：小规模、高精度、学术发表、需要复杂属性和关系
- Token适合：大规模（57万字）、AI批量标注、频繁迭代、需要人眼审核
- 两者可桥接：Token标注 + JSON元数据 → 可导出为TEI XML（`export_tei.py`，待开发）

子工序

编号	文件	内容
02a	SKILL_02a_章节切分与编号.md	段落编号（Purple Numbers）+ 小节划分 + 对话拆分
02b	SKILL_02b_区块与韵文处理.md	太史公曰/赞/策论围栏 + 韵文检测
02c	SKILL_02c_三家注标注.md	裴骃集解/司马贞索隐/张守节正义 注释层
02d	SKILL_02d_结构语义分析.md	句间/段间/区块间/章间语义关系挖掘、注释对齐与排版映射
02e	SKILL_02e_词法分析.md	

编号	文件	内容
		字级词性标注（虚词/动词/候选实体）、遗漏实体发现
02f	SKILL_02f_文本统计.md	字数/句长/词频/实体密度/五体对比等定量分析
02g	SKILL_02g_表格结构语义分析.md	史记十表维度定义、关系挖掘、HTML→JSON/CSV 数据转换
02h	SKILL_02h_词表构建.md	19类专项词表（人名/地名/官职等）识别与结构化

SKILL 02a: 古籍章节切分与编号 — Purple Numbers与语义分段

> 从《史记》130篇的章节结构化实践中提炼，适用于大规模古籍的篇章拆分、段落编号和小节划分。

一、核心概念

1.1 三层结构

古籍文本需要三层结构化：

```
篇章（Chapter）      130个独立文件，三位数编号
  └─ 小节（Section）   语义段落组，二/三级标题
      └─ 段落（Paragraph）   带Purple Numbers的原子文本单元
```

1.2 Purple Numbers（紫色编号）

定义：每个段落的唯一、永久、不可变标识符。

```
[0]       篇名段
[1]       第一段
[1.1]     第一段的第一子段
[1.1.1]   三级子段
[2]       第二段
```

为什么叫"Purple Numbers"：
- 源自Project Xanadu的概念 — 文档中每个段落都有永久地址
- 一旦分配，永不改变（即使周围内容重组）
- 渲染时显示为可点击的锚点链接

为什么重要:
- 跨文档引用：`092_淮阴侯列传#pn-3.1` 永远指向同一段文字
- 实体索引关联：`entity_index.json` 中每个实体引用格式为 `[chapter, paragraph_id]`
- 事件索引关联：事件定位到具体段落
- 搜索定位：精确到段落级别的全文检索

二、阶段1：篇章拆分

2.1 拆分规则

将完整古籍文本按卷/篇拆分为独立文件。

命名规范：`{三位编号}_{篇名}.txt`

```
001_五帝本纪.txt
002_夏本纪.txt
...
130_太史公自序.txt
```

《史记》的体裁分布:

体裁	篇数	编号范围	特征
本纪	12	001-012	帝王编年，时间跨度大
表	10	013-022	表格数据，结构特殊
书	8	023-030	制度专题，论述为主
世家	30	031-060	诸侯/功臣，叙事+编年
列传	70	061-130	人物传记，叙事为主

2.2 拆分脚本逻辑

```
# 识别篇章边界：标题行以体裁关键词结尾
CHAPTER_PATTERNS = ['本纪', '表', '书', '世家', '列传', '太史公自序']

def split_by_chapter(full_text):
    chapters = []
    current = []
    for line in full_text.split('\n'):
        if any(line.strip().endswith(p) for p in
CHAPTER_PATTERNS):
            if current:
                chapters.append(current)
            current = [line]
        else:
            current.append(line)
    return chapters
```

注意事项:
- 编号补零到3位，确保文件系统排序正确
- 保持原文完整性，不做任何修改
- 表类篇章（013-022）可能需要特殊处理（表格结构）

三、阶段1.5：段落修复（误分段合并）

原始文本拆分后，常因源文件换行位置不当而产生误分段（一个句子被截断为两行）。需在段落编号之前修复。

误分段特征

```
於是九州攸同，四奥既居，九山        ← 行末无句末标点
旅，九川涤原，九泽既陂...           ← 下一行直接续前句
```

合并规则（4条）

规则	条件	操作	理由
1	当前行以句末标点结尾（。！？」』：；…）	不合并	完整句子
2	当前行或下一行为空	不合并	段落分隔
3	当前行很短（<10字符）	不合并	可能是标题或小节标记
4	其他情况	合并	行末无句末标点，很可能是误分段

合并算法

```
ending_marks = ['。', '！', '？', '」', '』', '：', '；', '…']

def should_merge(current_line, next_line):
    if not current_line.strip() or not next_line.strip():
        return False                          # 规则2
    if len(current_line.strip()) < 10:
        return False                          # 规则3
    if current_line.rstrip()[-1] in ending_marks:
        return False                          # 规则1
    return True                               # 规则4：合并

# 循环合并：当前行可能需要与连续多行合并
merged = current_line
while should_merge(merged, next_line):
    merged += next_line
    # 继续检查下一行
```

扩展到其他古籍

句末标点列表需按古籍标点系统调整。未标点的古籍（如原始刻本）不适用此规则，需先完成标点再修复。

四、阶段2：段落编号（Purple Numbers）

3.1 编号规则

基本规则:
- 每个非空文本行获得一个编号
- `[0]` 保留给篇名
- 正文从 `[1]` 开始递增
- 子段落用点分层：`[1.1]`、`[1.2]`、`[2.1]`

何时使用子编号:
- 原文一段落过长（>500字），按语义切分后用子编号
- 对话穿插叙事，对话和叙事分别编号
- 引用块（`> "..."`）不独立编号，从属于上一段

短列表不编号：简单列举项（方位、物品等短词组）不需要子编号；长列表项（完整句子、对话）需要子编号：

```
# ✓ 短列表（无子编号）：
[2]
- 东至于=海=，登=丸山=，及=岱宗=。
- 西至于=空桐=，登=鸡头=。

# ✓ 长列表/对话（有子编号）：
[12]
- [12.1] @尧@曰："谁可顺此事？"
- [12.2] @放齐@曰："嗣子@丹硃@开明。"
```

编号放置位置：行首，紧跟空格

```
[1] 太史公曰：余登箕山，其上盖有许由冢云。

[1.1] 孔子序列古之仁圣贤人，如吴太伯、伯夷之伦详矣。

> "余以所闻由光义至高，其文辞不少概见，何哉？"
```

3.2 编号的不可变性

关键原则：Purple Numbers一旦分配，永不更改。

即使后续编辑（插入段落、重组内容），已有编号保持不变。新插入的段落使用子编号：

```
[3] 原有段落
[3.1] 后续插入的段落（不改变[3]和[4]的编号）
[4] 原有段落
```

这确保了所有外部引用（实体索引、事件索引、跨文档链接）始终有效。

五、阶段3：语义小节划分

4.1 小节层级

```
# [0] 篇名                    ← 一级标题（每篇唯一）

## 大节标题                   ← 二级标题（主要叙事段落）

### 小节标题                  ← 三级标题（细分主题）

[1.1] 正文段落...
```

4.2 两种小节格式

格式A：标题带编号

```
## [1] 早年困顿
[1.1] @淮阴侯@韩信@者，=淮阴=人也...
[1.2] @信@钓於城下...

## [2] 投奔项羽与汉王
[2.1] 及@项梁@渡=淮=...
```

锚点直接取自标题中的编号：`#pn-1`、`#pn-2`。

格式B：标题不带编号

```
## 项羽早年
### 项氏家世
[1.1] @项籍@者，=下相=人也...

## 起兵反秦
### 斩会稽守
[7.1] &秦&%二世元年七月%...
```

锚点取自该节第一个段落的编号：`#pn-1.1`、`#pn-7.1`。

4.3 按体裁定制小节策略

体裁	划分原则	典型小节
本纪	按帝王/时期	"黄帝"、"帝颛顼"、"帝喾"
世家	按诸侯世代	"太伯奔吴"、"寿梦立"、"阖闾争位"
列传（单传）	按人生阶段	"早年"、"拜将"、"封王"、"陨落"
列传（合传）	按人物分段	"管仲"、"晏婴"

体裁	划分原则	典型小节
列传（类传）	按人物分段	"总论"、"循吏某某"、"循吏某某"
书	按制度主题	"礼之起源"、"三代之礼"、"秦汉之变"
表	按表格结构	保留原表，可能需特殊处理

4.4 小节标题规范

好的标题:
- "鸿门宴" — 具体事件
- "萧何月下追韩信" — 经典典故
- "背水之战" — 著名战役

差的标题:
- "第一部分" — 无信息量
- "段落1-10" — 纯位置描述
- "内容" — 过于笼统

标题中清理实体标记：小节标题在 `sections_data.json` 中存储时，去除所有实体标记符号：

```
def clean_title(title):
    """去除标题中的实体标记"""
    for ch in ['@', '=', '$', '%', '&', '^', '~', '*', '!', '〖+', '〗', '〖?', '#']:
        title = title.replace(ch, '')
    return title.strip()
```

4.5 自动小节生成

对于没有明确分段的章节（约41篇），使用启发式算法生成建议：

```
def auto_generate_sections(paragraphs, chunk_size=10):
    """每10段为一组，分析前3段内容生成标题"""
    sections = []
```

```
    for i in range(0, len(paragraphs), chunk_size):
        chunk = paragraphs[i:i+chunk_size]
        sample = ' '.join(chunk[:3])
        title = infer_title(sample)
        sections.append({"anchor": chunk[0].pn, "title": title})
    return sections

def infer_title(text):
    """基于关键词的启发式标题推断"""
    if '曰' in text or '云' in text: return '评论与论述'
    if '年' in text or '月' in text: return '编年记事'
    if '战' in text or '伐' in text: return '战争与征伐'
    if '封' in text or '侯' in text: return '封赏与爵位'
    if '太史公' in text: return '太史公评论'
    return '叙事'  # 默认
```

重要：自动生成的标题仅为建议，必须人工审核后使用。

六、数据结构

5.1 sections_data.json

```
{
  "001_五帝本纪": [
    {"anchor": "1", "title": "黄帝"},
    {"anchor": "5", "title": "帝颛顼"},
    {"anchor": "7", "title": "帝喾"},
    {"anchor": "9", "title": "帝尧"},
    {"anchor": "15", "title": "帝舜"}
  ],
  "092_淮阴侯列传": [
    {"anchor": "1", "title": "早年困顿"},
    {"anchor": "2", "title": "投奔项羽与汉王"},
    {"anchor": "3", "title": "萧何月下追韩信"},
```

```
    {"anchor": "4", "title": "拜将坛上论天下"}
  ]
}
```

规则:
- 每篇最多15个小节（超过则需合并）
- `anchor` 指向该节第一个段落的Purple Number
- `title` 不含实体标记符号
- 不包含固定尾部（太史公曰、赞等）

5.2 HTML锚点格式

所有锚点统一使用 `pn-` 前缀:

```
<!-- 段落锚点 -->
<a href="#pn-1.1" id="pn-1.1" class="para-num" title="点击复制链
接">[1.1]</a>

<!-- 索引页的小节链接 -->
<a href="chapters/092_淮阴侯列传.tagged.html#pn-1">
  [1] 早年困顿
</a>
```

七、工具链

6.1 脚本一览

脚本	位置	功能
`section_extract.py`	scripts/	提取 `## [N] 标题` 格式的小节
`section_extract_all.py`	scripts/	完整提取（含无编号标题），生成 `sections_data.json`

脚本	位置	功能
`section_add_to_chapter.py`	scripts/	为指定章节添加小节标题
`section_auto_generate.py`	scripts/	为无分段章节自动生成小节建议
`section_add_ids_to_html.py`	scripts/	为HTML的h2/h3标签添加id属性
`section_update_index.py`	scripts/	用小节链接更新index.html导航
`lint_markdown.py`	scripts/	段落编号验证 + 标题层级检查

6.2 处理流程

```
# 1. 提取所有章节的小节结构
python scripts/section_extract_all.py

# 2. 对缺少小节的章节生成建议（需人工审核）
python scripts/section_auto_generate.py

# 3. 为指定章节添加小节
python scripts/section_add_to_chapter.py 092

# 4. 更新索引页的小节导航
python scripts/section_update_index.py

# 5. 验证格式
python scripts/lint_markdown.py chapter_md/092_淮阴侯列传.tagged.md
```

八、质量检查

7.1 lint_markdown.py 相关检查项

段落编号检查:
- 格式正确：`[数字]` 或 `[数字.数字...]`（不允许 `[1.a]`、`[一]`）
- 顶层编号连续（[1]→[2]→[3]，不允许跳跃 [1]→[3]）
- 子编号连续（[1.1]→[1.2]→[1.3]）

标题层级检查:
- 一级标题唯一（每篇只有一个 `#`）
- 不允许层级跳跃（`#` 直接到 `###`，跳过 `##`）
- 标题非空

常见问题及修复:

问题	表现	修复方式
编号跳跃	[1]→[3] 缺少[2]	检查是否遗漏段落或编号写错
层级跳跃	# 后直接 ###	补充 ## 二级标题
重复编号	两个 [3.1]	给后一个重新编号 [3.2]
空标题	`##` 后无内容	添加有意义的标题文字

九、特殊情况处理

8.1 "赞"段落（太史公评论）

每篇末尾通常有"太史公曰"或"赞"段落，格式特殊：

```
## 赞
[N] 太史公曰：...评论文字...
多行评论合并为一个段落编号
不需要每行单独编号
```

修复工具：`fmt_fix_zan.py` — 将多编号的赞合并为单编号，各句换行但不分段。

8.2 对话段落

包含多人对话的长段落需要拆分，拆分后每行转为列表项（`-` 开头）：

```
# 拆分前：
[5] 齐王曰："寡人闻之。"晏子对曰："臣请言之。"

# 拆分后：
- [5.1] 齐王曰："寡人闻之。"
- [5.2] 晏子对曰："臣请言之。"
```

修复工具：`fmt_split_dialogues.py` — 在"X曰:"边界处自动拆分。

详细规则与示例见 SKILL_02a_章节切分与编号.md §五。

8.2.1 嵌套引号的拆分规则

将包含嵌套引号的长段落拆分为列表项时，**必须保持原文的引号层级**，不得将内层单引号 `'...'` 改为外层双引号 `"..."`。

```
# 原文（一整段）：
作汤诰："维三月，王自至於东郊。告诸侯群后：'毋不有功於民，勤力乃事。予乃大罚殛女，毋予怨。'曰：'古禹、皋陶久劳于外，...先王言不可不勉。'曰：'不道，毋之在国，女毋我怨。'"以令诸侯。

# ✗ 错误拆分（内层引号被改为双引号）：
- [8.1] "维三月，...毋予怨。'"
- [8.2] 曰："古禹、皋陶...不可不勉。"     ← 错：'→"
- [8.3] 曰："不道，...女毋我怨。"           ← 错：'→"
```

```
# ✓ 正确拆分（保持内层单引号）：
- [8.1] "维三月，...毋予怨。'
- [8.2] 曰：'古禹、皋陶...不可不勉。'
- [8.3] 曰：'不道，...女毋我怨。'"
```

原则：拆分只改变物理换行，不改变原文的任何字符（包括引号类型）。外层 `"` 的开闭仍然在首段和末段，内层 `'` 保持不变。

8.3 表类篇章（十表 013-022）

年表（013-022）以表格形式呈现，每行数据对应一个历史条目（侯国、诸侯、君主、将相等），**必须为每行分配独立 Purple Number**，以支持精确引用和事件链接。

8.3.1 行号格式

表格行使用 `[rN]` 格式（`r` 前缀区别于段落编号 `[1.1]`），章内全局递增，跨多张表格连续计数：

```
| 国名 | 侯功 | 元光 | 元朔 | … |
| --- | --- | --- | --- | … |
| [r1] 翕。 | 匈奴相降，侯。 | 三。四年七月壬午，侯赵信元年。 | … |
| [r2] 持装。 | 匈奴都尉降，侯。 | … |
| [r5] 长平。 | 以元朔二年再以车骑将军击匈奴… | … |
```

渲染器读取 `[rN]` 并生成锚点 `id="pn-r5"`，HTML URL 为 `020_建元以来侯者年表.html#pn-r5`。

8.3.2 生成脚本

`scripts/add_table_row_pn.py` 可一键为十表 MD 文件添加 `[rN]` 标记：

```
python scripts/add_table_row_pn.py          # 处理全部 013-022 章
python scripts/add_table_row_pn.py 020       # 只处理 020 章
```

已有 `[rN]` 的行会自动跳过，支持幂等运行。

8.3.3 在事件索引中引用

十表来源的事件，`段落位置`字段应使用`r{N}`格式，而非`表（X条目）`：

```
- **段落位置**: r5          ← 对应 #pn-r5（长平侯条目）
- **段落位置**: r8          ← 对应 #pn-r8（平津侯/公孙弘条目）
```

多行事件可记为范围`r12-r15`。

8.3.4 CSS 渲染

行号列（`<td class="row-pn-col">`）样式在`docs/css/shiji-styles.css`中定义，与段落`.para-num`使用相同颜色（淡紫色`#C8A8FF`）。表格其余样式（全视口宽度、sticky 表头、间隔行色）由`.shiji-table-wrapper`/`.shiji-table`/`.even`/`.odd`控制，也在同一 CSS 文件中。

8.4 超长小节

如果一个小节超过20个段落，应进一步拆分：

```
MAX_SECTIONS_PER_CHAPTER = 15  # 小节上限
MIN_PARAGRAPHS_PER_SECTION = 3  # 太短的小节应合并
MAX_PARAGRAPHS_PER_SECTION = 20 # 太长的小节应拆分
```

十、文件命名规范

文件类型	格式	示例
标注 Markdown	`NNN_章节名.tagged.md`	`001_五帝本纪.tagged.md`
发布HTML	`NNN_章节名.html`	`001_五帝本纪.html`（不含`.tagged`）

文件类型	格式	示例
原始文本	`NNN_章节名.txt`	`001_五帝本纪.txt`

- 三位数字编号补零（001-130）
- 下划线分隔编号和篇名
- `.tagged.md` 表示已完成实体标注
- HTML发布时移除 `.tagged` 后缀

十一、经验教训

1. **编号是基础设施** — Purple Numbers是整个知识系统的定位基础（实体索引、事件索引、跨文档引用都依赖它），设计必须在最早期确定且不可变。
2. **语义分段优于机械分段** — "每500字切一段"远不如"按帝王更迭切一段"有价值。小节标题是读者的导航地图。
3. **体裁决定结构** — 本纪按时间、列传按人生、世家按世代，不同体裁的最佳分段策略完全不同。
4. **自动生成只是起点** — 启发式规则生成的小节标题准确率约60%，必须人工审核。但它大幅降低了从零开始的工作量。
5. **小节数量要克制** — 每篇5-15个小节是合理范围。太多（>20）等于没有分段，太少（<3）等于只有标题。
6. **两种标题格式都要支持** — 有些章节天然适合"## [N] 标题"格式（列传），有些适合无编号格式（本纪），工具链必须兼容两者。
7. **锚点格式统一** — `#pn-{number}` 作为唯一锚点格式贯穿整个系统（源文件、HTML、索引、API），避免多套ID体系。
8. **先切分后标注** — 先完成篇章拆分和段落编号，再做实体标注。如果顺序反过来，每次修改编号都要重新关联所有实体引用。

十二、扩展到其他古籍

10.1 直接复用

- Purple Numbers编号系统（格式和规则通用）
- `lint_markdown.py` 编号验证逻辑
- `section_extract_all.py` 小节提取逻辑
- `sections_data.json` 数据格式
- HTML锚点渲染（`#pn-` 格式）

10.2 需要适配

项目	需调整内容
篇章拆分	不同古籍的卷/篇结构不同
体裁分类	编年体（资治通鉴）vs 纪传体（二十四史）vs 纪事本末体
小节策略	编年体按年份，纪传体按人物/事件
自动生成关键词	不同时代用语不同（"曰"→"说"、"伐"→"征"）
特殊段落	"赞"是《史记》特有，其他书有"论"、"赞"、"评"

10.3 规模参考

古籍	篇章数	预估段落数	小节数
史记	130	~8,000	~1,200
汉书	120	~10,000	~1,100
资治通鉴	294	~30,000	~3,000

本SKILL文档基于《史记》130篇的结构化处理经验提炼（2025-02 至 2026-03）。
核心产出：130篇×平均60段落 ≈ 8,000个Purple Numbers，~1,200个语义小节。

SKILL 02b: 区块与韵文处理 — Fenced Div 语义块与赞体韵文的标注、检测与渲染

> 从《史记》130篇的太史公曰/赞标注实践中提炼，适用于带评论和韵文的古籍文本。
>
> **前置**: 实体标注和章节结构已完成，详见 SKILL_02a_章节切分与编号.md。

一、问题定义

《史记》每篇末尾通常有两部分:

1. **太史公曰**（散文评论）— 司马迁的史论
2. **赞**（韵文总结）— 四字为主的押韵评语

这两部分需要用 Pandoc Fenced Div 语法（`::: tag` / `:::`）标注为语义块，使渲染器能:
- 应用不同的 CSS 样式（边框、背景、字体）
- 韵文保持硬换行（每句一行，`<br>` 分隔）
- 标签显示为 tooltip（`title` 属性），而非标题

二、Fenced Div 语法

2.1 基本格式

```
::: 标签名
内容...
:::
```

· 开头 `::: 标签名`，标签名为中文语义标记

· 结尾 `:::` 单独一行

· 块内内容无需 `>` 前缀，直接书写

2.2 标签类型

标签	CSS 类	用途
太史公曰	`.note-太史公曰`	史论散文评论
赞	`.note-赞`	韵文总结
诗歌	`.note-诗歌`	篇中出现的诗歌
谏言	`.note-谏言`	大臣谏言
策论	`.note-策论`	纵横家论述
制度	`.note-制度`	制度性文献
传说	`.note-传说`	神话传说
宋昌说辞 等	`.note-box`	自定义标签

2.3 渲染规则

```
::: tag → <div class="note-box note-{tag}" title="{tag}">
内容    → <p>...</p>
:::     → </div>
```

关键设计决策：
- **标签不渲染为 `<h4>` 标题**，而是 `title` 属性（鼠标悬停显示 tooltip）
- **赞/诗歌块**内连续行用 `<br>` 连接（`flush_para()` 函数），保持硬换行
- CSS 使用 `white-space: pre-line` + `line-height: 2.0`

三、核心规则

3.1 区块不跨节

规则：`::: 块必须在 ## 或 ###` 标题之前关闭，不可跨越节边界。

```
# 错误 ❌
::: 太史公曰
[20] 太史公曰: ...

## 赞                ← 标题在块内，块跨节了
[21] 韩襄遗孽...
:::

# 正确 ✓
::: 太史公曰
[20] 太史公曰: ...
:::

## 赞
::: 赞
[21] 韩襄遗孽...
:::
```

3.2 太史公曰结构

标准模式（大多数篇目）：

```
## 太史公论XXX

::: 太史公曰
[N] 太史公曰: 散文评论...
:::

::: 赞
```

```
[N+1] 四字韵文，每句独立。
句句相对，押韵整齐。
:::
```

3.3 特殊章节

章节类型	处理方式
年表（017/018）	有 `## 太史公曰` 标题但不加 `:::` 包裹（整章即太史公曰）
书（023-030）	太史公曰可能在开头或中间，按实际位置标注
有自定义标签的（025律书）	保留 `::: 太史公评文帝` 等变体标签
褚先生补注（058/060）	含未标签的 `:::` 块，是预存问题

四、韵文检测

4.1 赞的特征

赞是四言韵文，典型特征：
- **句式整齐**：以四字（偶有三字或五字）为一个语义单元
- **对仗工整**：上下句结构对称
- **押韵**：偶数句末字韵脚相同

```
韩襄遗孽，始从汉中。
剖符南面，徙邑北通。
穰当归国，龙雒有功。
```

4.2 verse_score 评分函数

```
def verse_score(line):
    """0-1 分，表示该行"像韵文"的程度"""
    segments = split_by_punctuation(strip_tags(line))
    lengths = [len(s) for s in segments]

    four_char = count(l == 4 for l in lengths)
    near_four = count(3 <= l <= 5 for l in lengths)

    if four_char / total > 0.5: return 0.9    # 强韵文
    if near_four / total > 0.7: return 0.7     # 中等
    if avg_length > 6: return 0.1              # 长句 = 散文
    return 0.3
```

4.3 verse vs prose 判断

- **整组判断**（majority vote）：组内所有行的 verse_score 平均值 > 0.6 → 韵文
- **散文陷阱**：某些古文散文也有四字节奏（如"荆王王也，由汉初定"），需整段判断而非逐句

五、格式修复脚本

5.1 修复流程（推荐顺序）

```
1. fmt_fix_zan.py         — 赞块编号合并（多个 [X.Y] → 保留第一个）
2. fmt_fix_verse.py       — 赞块内韵文换行（单行多句 → 每句独立）
3. fix_zan_prose_v3.py    — 散文/韵文分离（从赞移回太史公曰）
4. fix_block_sections.py — 区块不跨节（在标题前自动关闭）
5. auto_tag_taishigong.py — 自动补全缺失的 ::: 标注
```

5.2 各脚本说明

脚本	位置	功能
`fmt_fix_zan.py`	`scripts/`	赞块内多个段落编号合并为一个
`fmt_fix_verse.py`	`scripts/`	赞块内单行多句拆分为每句一行
`fix_zan_prose_v3.py`	`/tmp/`	散文从赞块移至太史公曰（verse_score 评分）
`fix_block_sections.py`	`/tmp/`	自动关闭跨 `##` / `###` 的区块
`auto_tag_taishigong.py`	`/tmp/`	批量为缺失标注的章节补 `::: 太史公曰` 和 `::: 赞`

5.3 审计检查

```
# 检查空块
grep -l '::: 太史公曰' chapter_md/*.tagged.md | \
  xargs -I{} python -c "..."

# 检查跨节
grep -n ':::' + grep -n '## ' → 确保没有交叉

# 检查孤立 :::
python /tmp/check_blocks.py
```

六、CSS 样式

```
/* 太史公曰 */
.note-太史公曰 {
    border-left: 3px solid #8B7500;
```

```
}

/* 赞（韵文） */
.note-赞 {
    border-left: 3px solid #a0522d;
    background-color: #fefcf5;
    font-style: italic;
}
.note-赞 p {
    line-height: 2.0;
    white-space: pre-line;    /* 保持硬换行 */
}

/* 诗歌 */
.note-诗歌 {
    border-left: 3px solid #6b8e23;
    background-color: #fcfef5;
    font-style: italic;
}
.note-诗歌 p {
    line-height: 2.0;
    white-space: pre-line;
}
```

七、关键经验

1. **区块绝不跨节** — 标题是结构边界，区块必须在标题前关闭。违反此规则会导致 HTML 嵌套错误。
2. **韵文检测用整组投票** — 逐句判断会被古文散文的四字节奏误导（如"及汉兴，依日月之末光"），整组平均分更可靠。
3. **verse_score 阈值**: 0.6 是分界线 — >0.6 韵文，<0.4 散文，0.4-0.6 需人工确认。
4. **标签用 tooltip 不用标题** — `<h4>` 标签会破坏文档结构和目录层级，`title` 属性既提供信息又不干扰布局。

5. **多遍修复有序执行** — 先合并编号 → 再拆韵文行 → 再分散文/韵文 → 最后检查跨节。顺序错误会导致中间状态混乱。
6. **年表不包裹** — 017/018 整章就是太史公曰，加 `:::` 包裹反而多余。
7. **自动脚本后必须人工抽查** — 特别关注：散文伪装成韵文的章节（如051荆燕世家）、书类章节（太史公曰在开头）、有褚先生补注的章节（058/060）。

八、对话分析

《史记》大量篇幅是对话（X曰："…"），对话是区块分析的重要组成部分。

8.1 对话识别

- **直接引语**：`X曰："…"` / `X曰：'…'`
- **嵌套引语**：外层 `""` 内层 `''`
- **间接引语**：`X以为…` / `X谓…`

8.2 对话统计特征

体裁	对话占比	特点
本纪	30-40%	诏令、廷议为主
表	<5%	几乎无对话
书	10-15%	引经据典式
世家	25-35%	君臣对话
列传	35-45%	人物对话最丰富

8.3 与其他工序的关系

- 对话拆分规则详见 SKILL_02a § 8.2
- 对话的嵌套引号处理是校勘（01）的重要工作内容

- 对话往返是结构语义分析（02d）的段间关系类型之一

九、区块标签查询与专项索引应用

9.1 核心概念

区块标签是可查询的结构化元素

- 每个 `::: tag` 是一个带标签的文本区块
- 可以按标签查询、提取、重组
- 形成独立的专项索引页面

9.2 查询框架

```
def extract_blocks_by_tag(tag_name, chapter_range=None):
    """
    通用区块标签查询接口

    参数:
        tag_name: 标签名称（如"太史公曰"、"赞"、"诏"）
        chapter_range: 章节范围过滤（可选）

    返回:
        List[Dict]: 包含章节信息和区块内容的列表
    """
    results = []
    for chapter in chapters:
        # 提取带指定标签的区块
        blocks = find_blocks(chapter, tag_name)
        results.extend(blocks)
    return results
```

9.3 已实现的专项索引

太史公曰专项 (2026-03-18)

- **标签**: `太史公曰`
- **数量**: 125篇（覆盖96.2%章节）
- **输出**:
- `docs/special/taishigongyue.json` — 结构化数据
- `docs/special/taishigongyue.md` — Markdown格式
- `docs/special/taishigongyue.html` — HTML页面（实体高亮）
- **脚本**:
- `scripts/extract_taishigongyue.py` — 提取脚本
- `scripts/render_taishigongyue_html.py` — HTML生成
- **访问**: 主页 → 📖 专项索引 → 太史公曰

实现特点:
1. 支持多种格式（`::: 太史公曰` 和段落格式）
2. 支持段落编号（`[27]` 和 `[27.1]`）
3. 保留实体标注并在HTML中彩色渲染
4. 章节链接直达原文

9.4 可扩展专项

基于相同框架，可以快速实现：

专项	标签模式	数量估计	用途
赞	`赞`	~110篇	韵文评价汇总
诏书	`.*诏`	~50篇	皇帝诏令文献
书信	`书`	~30篇	历史书信
传说	`传说`	~20篇	引用传闻评论

专项	标签模式	数量估计	用途
谏言	`谏言`	~40篇	大臣谏言
策论	`策论`	~30篇	纵横家论述
诗歌	`诗歌`	~15篇	篇中诗歌
制度	`制度`	~25篇	制度性文献
对话	(特殊)	全书	人物对话（需NLP提取）

9.5 技术栈

```
区块标签系统
    ├── 标注层: ::: tag 语法 (Markdown)
    ├── 存储层: Purple Numbers 段落编号
    ├── 查询层: Python + 正则表达式
    ├── 数据层: JSON + Markdown
    ├── 渲染层: 自定义HTML生成器
    └── 展示层: 专项索引页面
```

9.6 设计优势

1. **统一语法**: 所有区块使用相同的 `::: tag` 格式
2. **可扩展性**: 新增专项只需指定tag，复用提取框架
3. **结构化**: 每个区块有Purple Number，可精确引用
4. **语义保留**: 实体标注在提取后仍然保留
5. **多格式输出**: 同时生成JSON/MD/HTML三种格式

9.7 工作流程

```
# 1. 标注阶段（已完成）
# 在 chapter_md/*.tagged.md 中用 ::: tag 标记区块

# 2. 提取阶段
python scripts/extract_blocks.py --tag="太史公曰"

# 3. 生成阶段
python scripts/render_blocks_html.py --input=taishigongyue.json

# 4. 发布阶段
# 文件生成在 docs/special/ 目录
# 主索引页自动添加入口链接
```

9.8 质量保证

- **完整性检查**: 对比维基文库版本，确保无遗漏
- **格式兼容**: 支持多种区块格式变体
- **人工抽查**: 特殊章节（书类、年表）需人工验证
- **版本跟踪**: 所有提取结果包含章节版本信息

9.9 未来扩展

1. **智能分类**: 用NLP自动识别未标注的区块类型
2. **交叉索引**: 区块之间的引用关系（如赞引用太史公曰）
3. **主题聚类**: 按主题对区块进行聚类分析
4. **对比研究**: 跨章节的区块内容对比分析
5. **可视化**: 区块分布的时间轴/地图可视化

十、参考文档

- **设计文档**: doc/BLOCK_TAG_SYSTEM.md — 区块标签系统完整设计

- **专项索引**: docs/special/README.md — 专项索引说明
- **渲染规范**: SKILL_03d_渲染与发布.md — HTML渲染规则

> 另见:
> - SKILL_02a_章节切分与编号.md — 段落编号、对话拆分
> - SKILL_02d_结构语义分析.md — 句间/段间语义关系
> - SKILL_02e_词法分析.md — 字级词性标注
> - SKILL_02f_文本统计.md — 定量统计分析

SKILL 02c: 三家注标注 — 《史记》三家注语义化与数据结构设计

> 从《史记》三家注（裴骃《集解》、司马贞《索隐》、张守节《正义》）整理与标注实践中提炼，
> 适用于古籍注疏类文本的结构化处理。
>
> **注意**：主文本的实体标注规范详见 SKILL_03a_实体标注.md。
> 本文档聚焦**注疏文本**的特殊处理规则及配套数据结构。

一、三家注的性质与内容特点

1.1 三注概览

注家	朝代	作者	主要内容
《集解》	南朝宋（5世纪）	裴骃	训诂字义、考证人物地名、汇集前人注解、校勘异文
《索隐》	唐（8世纪）	司马贞	音韵注音（反切/直音）、文字辨析、补正《集解》、补录佚文
《正义》	唐（8世纪）	张守节	地理考证（引《括地志》等）、历史背景阐释、较长的解说性注文

1.2 三注内容分类

注疏文本的典型内容类型，决定了标注策略的差异：

内容类型	典型示例	标注策略
释义注	"某，某也"	标准实体标注
音韵注	"音X"、"X反"（反切）、"读如X"	**不标注实体，保留原文**
引书注	"《汉书》曰……"、"《说文》云……"	书名用 `〖^书名〗` 标注
地理注	"某地，在今某州某县"	标准地名标注
人物注	"某，某国人，仕为某官"	标准人名/官职标注
校勘记	"一本作某"、"某本无此字"	保留纯文本，不标注
佚文辑录	引自已佚文献的史料	标准实体标注

1.3 与主文本标注的核心差异

1. **语言风格**：注疏语言以解释性散文为主，多"某曰""某云"引文格式，不同于主文本的叙事文体
2. **音韵注特殊性**：音韵内容（"音X"、"X反"）是注疏独有类型，不含语义实体，不参与实体标注
3. **书名引用密集**：注家大量引用他书，书名需单独标记以便溯源
4. **引文格式**：注疏中的"某曰"是**文献引用格式**，非叙事对话，处理规则不同（见第五节）
5. **地理考证为主**：《正义》大量地名多为考证语境，与主文本地名功能不同但标注规则相同

二、数据结构设计

2.1 维基文库HTML中的注文结构

维基文库三家注的HTML采用**内联嵌入**模式——注文直接穿插在正文段落中：

```
<p>
  主文本片段1
  <small>
    <span style="background-color: green; ...">集解</span>
    <span style="color:green">注文内容...</span>
    <span style="background-color: deepPink; ...">索隱</span>
    <span style="color:deeppink">注文内容...</span>
  </small>
  主文本片段2
  <small>
    <span style="background-color: #966; ...">正義</span>
   注文内容...
  </small>
  主文本片段3
</p>
```

关键特征:
- 注文被 `<small>` 标签包裹，与前面的主文本片段形成"被注词组→注释"的对应
- 同一个 `<small>` 中可包含多家注（集解+索隐+正义），通过颜色标签分隔
- 注文中的实体已用内联样式标注（实线下划线=人名/地名，波浪下划线=书名）
- 全书的 `<small>` 标签是自动分离注文与正文的关键标记

2.2 文件命名与存放位置

```
kg/sanjia/data/NNN_*.sanjia.json        ← 结构化三家注数据
kg/sanjia/data/NNN_*.collation.json     ← 校勘异文数据（可选）
```

2.3 JSON 结构规范

```
{
  "chapter": "001",
  "chapter_title": "五帝本紀",
  "notes": [
    {
      "target_phrase": "黃帝者，少典之子",
```

```
      "target_pn": "1",
      "source": "集解",
      "text": "譙周曰：「有熊國君，少典之子也。」皇甫謐曰：「有熊，今河南新鄭是也。」",
      "ws_entities": {
        "solid_underline": ["譙周", "河南"],
        "wavy_underline": []
      },
      "tagged_text": ""
    },
    {
      "target_phrase": "黃帝者",
      "target_pn": "1",
      "source": "索隱",
      "text": "案：有土德之瑞，土色黃，故稱黃帝……",
      "ws_entities": {
        "solid_underline": ["孔安國", "皇甫謐"],
        "wavy_underline": ["帝王代紀", "系本"]
      },
      "tagged_text": ""
    },
    {
      "target_phrase": "軒轅之丘",
      "target_pn": "1",
      "source": "正義",
      "text": "《括地志》云：軒轅丘在鄭州新鄭縣西北。",
      "ws_entities": {
        "solid_underline": ["鄭州", "新鄭縣"],
        "wavy_underline": ["括地志"]
      },
      "tagged_text": ""
    }
  ],
  "collation": [
    {
      "original": "筴",
      "corrected": "策",
```

```
      "context": "迎日推筴"
    }
  ]
}
```

2.4 字段说明

字段	类型	必填	说明
`chapter`	string	是	三位数章节编号
`chapter_title`	string	是	章节标题
`notes`	array	是	注文条目列表
`notes[].target_phrase`	string	是	被注的主文本片段（`<small>` 前的文本）
`notes[].target_pn`	string	否	关联的 Purple Number（后期匹配填入）
`notes[].source`	string	是	注家：`集解` / `索隐` / `正义`
`notes[].text`	string	是	注文纯文本（去除HTML标签）
`notes[].ws_entities`	object	是	维基文库已有标注（从HTML下划线样式提取）
`notes[].ws_entities.solid_underline`	array	是	

字段	类型	必填	说明
			实线下划线实体列表（人名+地名）
`notes[].ws_entities.wavy_underline`	array	是	波浪下划线实体列表（书名）
`notes[].tagged_text`	string	否	带〖〗Token的标注文本（后期LLM标注填入）
`collation`	array	否	校勘异文对（从删除线+点线红底提取）
`collation[].original`	string	是	被校改的原文字（删除线文本）
`collation[].corrected`	string	是	校勘替换文字（点线红底文本）
`collation[].context`	string	否	所在上下文

2.5 target_pn 关联规则

- `target_pn` 对应主文本 `chapter_md/NNN_*.tagged.md` 中的 Purple Number
- **首次生成时可为空**：`parse_sanjia.py` 从HTML提取注文时，先用 `target_phrase` 记录被注词组
- **后期匹配填入**：用 `target_phrase` 在主文本中搜索，找到所在段落的 Purple Number
- 若注文位置不确定（如章首综合注），使用 `"0"` 表示章级注

三、标注要点

3.1 适用 17 类实体 Token

注疏文本中出现的人名、地名、官职等实体，适用与主文本相同的标注规则：

Token	注疏中的典型用法
〖@人名〗	注家引用的历史人物，如 〖@郑玄〗曰
〖=地名〗	地理考证中的地名，如 在=长安=城西
〖;官职〗	人物介绍中的官职，如 仕为〖;丞相〗
〖%时间〗	年代考证，如 %汉武帝元狩元年%
〖&朝代〗	朝代名，如 &汉&初
〖^制度〗	制度名（含书名扩展，见3.3节）
〖~族群〗	民族、族群
〖•器物〗	器物名
〖!天文〗	天象、历法
〖?神话〗	传说人物（注疏中引用神话时）
〖+名称〗	生物（较少见）

参考完整规则：SKILL_03a_实体标注.md。

3.2 音韵注：不标注，保留原文

音韵注是《索隐》的重要组成部分，内容为字音说明，不含语义实体，**一律不做实体标注**：

类型	示例	处理
直音	"音皇"、"读若某"	保留原文，不标注
反切	"呼光反"、"古朗切"	保留原文，不标注
声调说明	"上声"、"去声"	保留原文，不标注

示例（正确处理）：

```
原文：索隐：典，音展。
tagged_text："典，音展。"    ← 不标注，保持原文
```

若一条注文同时含音韵说明和实体内容，仅对非音韵部分标注：

```
原文：索隐：禅，音善。禅让之义也，言尧让位於舜。
tagged_text："禅，音善。〖^禅让〗之义也，言〖@尧〗让位於〖@舜〗。"
```

3.3 书名：用 〖{书名〗 标注

注疏大量引用他书，书名作为 〖{典籍〗 Token 标记（v2.8起，含书名号《》）：

```
正义：〖{《括地志》〗云：=轩辕丘=在=郑州新郑县=西北。
集解：〖{《汉书》〗音义曰："胡苏，地名也。"
索隐：〖{《说文》〗云：轩，曲輈藩车也。
```

书名标注的范围：
- 明确的引书名称（有《》标记或习知书名）标注
- 泛指称谓（"古书云"、"或说"）不标注

3.4 校勘记：保留纯文本

校勘记记录版本异同，不含可标注实体，保持原文不标注：

```
原文：一本作"轩辕之野"。
tagged_text: "一本作\"轩辕之野\"。"    ← 不标注
```

四、与主文本的差异

4.1 语言特征对比

维度	主文本	三家注
文体	叙事散文	解释性散文
人称	第三人称叙述	注家以第一人称或引述他人
动词密度	较高（叙事行为）	较低（解释性陈述）
引用格式	少有引书	大量引书，"某曰""某云"密集
音韵内容	无	《索隐》中大量出现
地理细节	简略	《正义》多详细坐标性描述

4.2 实体密度

注疏文本的实体密度通常**低于主文本**，原因：
- 音韵注占相当比例，无可标实体
- 解释性句式（"某，某也"）实体密度低于叙事句式
- 《正义》地理注虽地名多，但单条注文人名稀少

五、引文格式处理（"某曰"不做对话拆分）

5.1 规则

注疏中的 `某曰：` 是**文献引用格式**，不是叙事对话，**不适用**主文本的对话格式化规则：

- 主文本：`〖@尧〗曰："……"` → 可拆分为对话列表项
- 注疏引文：`〖^《汉书》〗音义曰："……"` → **保持整行，不拆分**

5.2 引文格式的识别标准

注疏引文的特征（满足任一即为引文格式，不做对话处理）：

1. 说话主体为书名（`〖^某书〗曰`）
2. 说话主体为注家本人（`骃案`、`正义曰`、`索隐曰`）
3. 说话主体为已故学者的学术性陈述（`〖@郑玄〗曰：某字之义……`）

5.3 示例

```
{
  "source": "集解",
  "text": "韦昭曰：少典，有熊国君名也。",
  "tagged_text": "〖@韦昭〗曰：少典，〖&有熊〗国君名也。"
}
```

注意：`〖@韦昭〗曰：` 保持整行，不拆分为列表项。

六、数据来源：维基文库三家注

6.1 已有资源

维基文库三家注版（`archive/wikisource_sanjia/`）已从 epub 提取为130卷独立 HTML，包含丰富的内联标注：

标注样式	全书数量	实体类型	HTML特征
实线下划线	11,730	人名 + 地名（混合）	`text-decoration-style: solid`
波浪下划线	1,198	典籍/书名	`text-decoration-style: wavy`
删除线	512	校勘异文	`text-decoration: line-through`
点线红底	513	校勘注记	`border-bottom: 1px dotted red`

三家注结构标记：
- **集解**：绿色标签（`background-color: green`）+ 绿色正文
- **索隐**：粉色标签（`background-color: deepPink`）+ 粉色正文
- **正义**：褐色标签（`background-color: #966`）+ 默认色正文

6.2 三家注对本项目的三大价值

价值一：校勘（SKILL_01）

三家注是传统校勘的核心工具，维基文库版已标注校勘信息：

- **删除线**（512处）：标示被校改的原文字（异体字/误字）
- **点线红底**（513处）：与删除线配合，标示校勘替换文字
- **注文中的校勘记**：集解/索隐中大量"一本作某"、"某本无此字"等异文记录
- **利用方式**：提取删除线+点线红底校勘对 → 与 `docs/original_text/` 对照 → 生成校勘异文表

价值二：实体标注增强（SKILL_03a）

三家注中的实体标注可直接增强主文本的实体识别：

实线下划线（11,730处）— 人名 + 地名混合：

排名	实体	出现次数	类型
1	徐廣	2,251	注家（集解引）
2	韋昭	618	注家
3	鄭玄	614	注家
4	服虔	572	注家
5	孔安國	531	注家
…	河南	138	地名
…	兗州	82	地名

地名约1,390处（含州/縣/山/河/郡等特征字），可按特征字与人名分离。

波浪下划线（1,198处）— 典籍书名：

地理志(464)、系本(177)、百官表(78)、大戴禮(38)、別錄(36)等，可扩充《》标注词表。

利用方式：
1. 提取实线下划线实体 → 按地名特征字分离人名/地名 → 与 `entity_index.json` 比对 → 输出未覆盖实体候选
2. 提取波浪下划线书名 → 与典籍词表比对 → 扩充《》标注
3. 注文中提及的人名/地名/官职 → 扩充实体词表（尤其是注家引用的古人名和古地名）

价值三：音韵数据（未来方向）

《索隐》中含大量音韵注释，是古汉语语音研究的珍贵资料：

类型	示例	说明
直音	"音皇"、"读若某"	同音字注音
反切	"呼光反"、"古朗切"	声母+韵母组合注音

类型	示例	说明
声调	"上声"、"去声"	声调说明

潜在价值:
- 提取全书反切/直音数据 → 构建《史记》古音索引
- 与《广韵》等韵书对照 → 研究上古/中古音演变
- 当前阶段不做标注处理，仅保留原文

价值四：地理知识图谱（SKILL_05/06）

《正义》大量引用《括地志》等唐代地理文献，提供"古地名→今地名"的对应关系：

> 正义：〖=轩辕丘〗在〖=郑州新郑县〗西北。
> 正义：〖=涿鹿〗故城在〖=妫州怀戎县〗西南。

- 提取"在今某州某县"模式 → 构建古今地名对照表
- 与CHGIS等历史地理数据库关联 → 为事件地铁图提供坐标数据
- 扩充地名别名（古称→今称映射）

价值五：人物关系与背景（SKILL_05b）

注文中包含大量主文本未提及的人物信息：

- **人物身份**："某，某国人"、"仕为某官" → 扩充人物本体的属性
- **人物关系**："某，某之子"、"某之弟" → 补充家族关系图谱
- **人物评价**：注家对历史人物的评价 → 情感分析/观点挖掘的素材
- **佚文史料**：引用已佚文献中的人物事迹 → 补充事件索引

价值六：纪年与历法考证（SKILL_04c）

三家注中含有大量年代考证信息，可辅助事件年代推断：

- 集解/正义常引《竹书纪年》等对年代的校订
- "某事在某王某年"的明确纪年 → 交叉验证事件索引中的公元纪年
- 历法相关注释 → 辅助理解《历书》等天文历法章节

价值七：繁简对照与繁体支持

维基文库三家注为繁体，本项目主文本为简体。两个版本的对照可以：

- 构建《史记》繁简字对照表（覆盖古汉语专用繁体字，如"爲/為→为"、"後→后"等）
- 为阅读器提供繁体显示功能（TODO中已列"繁体支持"）
- 通过繁简差异发现异体字/通假字（如"筴/策"、"脩/修"等），辅助校勘

价值八：典籍引用网络（知识图谱扩展）

三家注引用了数百种古籍，波浪下划线标注了1,198处书名引用：

- 构建"三家注引书目录" → 了解注家的知识来源
- 引用频次分析 → 识别核心参考文献（地理志464次、系本177次等）
- 引书→被引段落的关联 → 构建文献引用网络（知识溯源）

6.3 利用策略

```
archive/wikisource_sanjia/*.html
  │
  ▼ Step 1: 提取标注实体
  ├── 实线下划线 → 按地名特征字分离人名/地名
  ├── 波浪下划线 → 典籍书名列表
  └── 删除线+点线红底 → 校勘异文对
  │
  ▼ Step 2: 与现有数据比对
  ├── 人名/地名 → entity_index.json 比对，发现遗漏实体
  ├── 书名 → 典籍词表比对，扩充《》标注
  ├── 校勘 → 辅助 SKILL_01 校勘（异文表）
  ├── 古今地名 → 提取"在今某州某县"→ 古今地名对照表
  ├── 人物背景 → 提取身份/关系/评价 → 扩充人物本体
  ├── 纪年考证 → 交叉验证事件年代
  ├── 引书目录 → 构建文献引用网络
  ├── 繁简对照 → 繁简字映射表 + 繁体显示支持
  └── 音韵 → 提取反切/直音索引（未来方向）
  │
  ▼ Step 3: 结构化解析
```

```
   ├── 按集解/索隐/正义标签拆分注文
   ├── 关联主文本段落（被注词组定位）
   └── 输出 NNN_*.sanjia.json
```

七、工作流程

```
维基文库三家注 HTML（archive/wikisource_sanjia/）
   │
   ▼ 第一步：parse_sanjia_html.py
   ├── 解析 <p> 结构：<small> 前文本 = target_phrase，<small> 内 = 注文
   ├── 按颜色标签分离集解（green）/索隐（deeppink）/正义（#966）
   ├── 提取实线下划线实体 → ws_entities.solid_underline
   ├── 提取波浪下划线书名 → ws_entities.wavy_underline
   ├── 提取删除线+点线红底 → collation 校勘对
   └── 输出 NNN_*.sanjia.json（text + ws_entities，tagged_text 为空）
   │
   ▼ 第二步：match_target_pn.py
   ├── 用 target_phrase 在 chapter_md/NNN_*.tagged.md 中搜索
   ├── 繁简转换后模糊匹配（维基文库为繁体，主文本为简体）
   ├── 结合注文顺序与正文顺序对齐
   └── 填入 target_pn 字段
   │
   ▼ 第三步：annotate_sanjia.py（大模型 NER）
   ├── 读取 sanjia.json
   ├── ws_entities 作为种子标注（已知的人名/地名/书名）
   ├── LLM 补充标注（官职/时间/制度等 ws_entities 未覆盖的类型）
   ├── 音韵注识别（跳过标注）
   └── 写入 tagged_text 字段
   │
   ▼ 第四步：extract_sanjia_knowledge.py
   ├── 提取古今地名对照（"在今某州某县"模式）
   ├── 提取人物背景（"某，某国人，仕为某官"）
   ├── 提取引书目录（波浪下划线书名统计）
   ├── 提取音韵数据（反切/直音索引）
```

```
   └── 输出知识图谱增量数据
   │
   ▼ 第五步：compare_entities.py
   ├── 三家注实体 vs entity_index.json 比对
   ├── 输出：未覆盖人名/地名/书名候选列表
   └── 输出：校勘异文表（辅助 SKILL_01）
   │
   ▼ 第六步：render_shiji_html.py 更新
   └── 渲染时注入注文展示（见第八节）
```

7.1 第一步的关键：从 `<small>` 分离注文

维基文库HTML中，`<small>` 是注文与正文的分界标记：

```
for para in soup.find_all('p'):
    parts = []
    current_main = []
    for child in para.children:
        if child.name == 'small':
            target = ''.join(current_main)  # <small>前的文本 = 被注
词组
            notes = parse_note_block(child)  # 按颜色标签拆分集解/索
隐/正义
            parts.append((target, notes))
            current_main = []
        else:
            current_main.append(child.get_text())
```

7.2 Purple Number 关联方法

1. **繁简转换匹配**：维基文库为繁体，主文本为简体，需先做繁→简转换再搜索
2. **顺序对齐**：同一章内注文顺序与正文顺序一致，可用顺序辅助定位
3. **模糊匹配**：异体字（如"爲/為/为"）需归一化后匹配
4. **人工补正**：自动匹配失败时人工指定

7.3 ws_entities 作为标注种子

维基文库已有的下划线标注（11,730实线 + 1,198波浪）可直接作为LLM标注的种子输入：

```
已知实体（ws_entities）→ 提供给LLM作为参考 → LLM补充标注未覆盖的类型
```

这比从零标注效率更高：ws_entities 已覆盖人名+地名+书名，LLM只需补充官职/时间/制度/族群等类型。

八、渲染设计

8.1 展示形式：段落级内联展开

每个主文本段落 `[N]` 下方，显示该段落对应的三家注情况：

```
<!-- 主文本段落 -->
<p id="pn-1" class="paragraph">
  [1] 〖@黄帝〗者，〖@少典〗之子…
  <!-- 三家注指示器 -->
  <span class="sanjia-indicator">
    <button class="sanjia-btn 集解" title="集解">集</button>
    <button class="sanjia-btn 索隐" title="索隐">索</button>
    <button class="sanjia-btn 正义" title="正义">正</button>
  </span>
</p>

<!-- 展开后的注文区块（默认折叠） -->
<div class="sanjia-notes" id="sanjia-pn-1">
  <div class="note-item 集解">
    <span class="note-source">集解</span>
    <span class="note-phrase">黄帝者，少典之子</span>
    <p>皇甫谧曰：有熊国君，少典之子也。</p>
  </div>
```

```
  <div class="note-item 索隐">…</div>
  <div class="note-item 正义">…</div>
</div>
```

8.2 三注指示器设计原则

- 有注文的按钮**高亮显示**，无注文的**灰显**
- 点击按钮**切换**对应注家的注文显示/隐藏
- 三个按钮可**独立切换**（可同时展示多家注）
- 注文中的实体 Token 与主文本使用**相同的 CSS 样式**（复用现有样式表）

8.3 CSS 类命名

```
.sanjia-indicator { }          /* 指示器容器 */
.sanjia-btn { }                /* 单个注家按钮 */
.sanjia-btn.集解 { }           /* 集解专属颜色 */
.sanjia-btn.索隐 { }           /* 索隐专属颜色 */
.sanjia-btn.正义 { }           /* 正义专属颜色 */
.sanjia-notes { }              /* 注文展开区块 */
.note-item { }                 /* 单条注文 */
.note-source { }               /* 注家标签 */
.note-phrase { }               /* 被注词组（斜体显示） */
```

九、工具需求

9.1 新增脚本

脚本	位置	功能
`parse_sanjia.py`	`scripts/`	解析三家注原始文本 → `NNN_*.sanjia.json`（仅 text 字段）

脚本	位置	功能
`annotate_sanjia.py`	`scripts/`	调用 LLM 对 `text` 做实体标注，写入 `tagged_text`

9.2 parse_sanjia.py 设计要点

```
输入：三家注原始文本文件（按章组织）
输出：NNN_*.sanjia.json

主要逻辑：
1. 识别注家标记（"集解"/"索隐"/"正义"开头）
2. 提取被注词组（通常在注文标记之前的正文片段）
3. 调用词组匹配关联 target_pn
4. 输出结构化 JSON，tagged_text 初始为空字符串
```

9.3 annotate_sanjia.py 设计要点

```
输入：NNN_*.sanjia.json（含 text，tagged_text 为空）
输出：更新 tagged_text 字段

主要逻辑：
1. 读取 sanjia.json
2. 对每条 notes[i]：
   a. 判断是否为纯音韵注（正则：仅含"音X"/"X反"/"X切"）
      → 是：tagged_text = text（原样复制）
   b. 调用 LLM 进行实体标注（Prompt 指定18类 Token + 书名规则）
   c. 写回 tagged_text
3. 输出更新后的 JSON
```

LLM Prompt 要点:
- 提供与主文本相同的18类实体标注规范
- 额外规则：书名用 `〖^书名〗`（含《》）；音韵注不标注

- 提供2-3条注疏标注示例（few-shot）
- 要求输出仅为 `tagged_text` 字符串，不含解释

9.4 render_shiji_html.py 更新

在现有渲染器中新增三家注数据加载与注入逻辑：

```
修改点：
1. 检测同目录是否存在对应 .sanjia.json
2. 若存在，按 target_pn 构建注文索引 dict
3. 在渲染每个段落 [N] 时，若有对应注文：
   a. 在段落末尾注入 .sanjia-indicator 按钮组
   b. 在段落后注入 .sanjia-notes 折叠区块
4. 在 HTML head 注入三家注的 CSS 和 JS（展开/折叠交互）
```

十、关键设计决策

1. **JSON 而非 Markdown**：注文与主文本分离存储，避免混入主文本文件，便于独立维护和版本管理
2. **target_pn 关联而非嵌入**：通过 Purple Number 关联而非直接嵌入主文本，保持主文本文件格式稳定
3. **tagged_text 与 text 并存**：保留原文字段，标注结果独立，便于重新标注或修正
4. **书名扩展 〖^制度〗 Token**：复用现有 CSS 类 `.institution`，无需新增样式，可区分于其他制度标注
5. **音韵注不标注**：避免错误标注，字形相似的字（如反切用字）不构成历史实体
6. **引文"某曰"不拆分**：区别于主文本对话拆分规则，注疏引文格式用整行保留
7. **段落级折叠渲染**：每段独立可展开，比全章注文列表更利于对照阅读

> 另见：
> - SKILL_03a_实体标注.md — 主文本标注规范（18类实体 Token、对话拆分、NOTE 块）

> - SKILL_03a_实体标注.md — 18类实体标注规范详情
> - SKILL_02a_章节切分与编号.md — Purple Numbers 系统
> - SKILL_03a_实体标注.md — Lint 工具与完整语法规则

SKILL 02d: 结构语义分析 — 语义关系挖掘、注释对齐与排版映射

> 从已切分的结构化文本中，识别句子、段落、区块、章节之间的语义关系，将三家注等注释与原文精确对齐，并将关系与注释映射为排版样式，让读者通过视觉布局直观理解文本的逻辑结构。

核心理念

古文的语义关系隐含在文字中，传统排版无法体现。本工序通过识别关系类型并映射为排版样式，将"理解文章结构"的认知负担从读者转移到排版系统。

```
原始文本（线性）  →  语义关系标注  →  排版映射  →  可视化呈现
连续段落          因果/并列/递进     缩进/对照/色块   读者一眼看懂结构
```

四层语义关系

第一层：句间关系

句子与句子之间的逻辑连接，是最细粒度的语义单元。

关系类型	信号词/模式	排版映射	示例
因果	故/乃/於是/遂/则	原因正常，结果缩进+左边线	蚩尤作乱 → 黄帝徵师诸侯
并列	东…西…南…北… / 其一…其二…	横向flex或grid对照	东至于海…西至于空桐…

关系类型	信号词/模式	排版映射	示例
递进	且/又/乃/更	缩进+左边线	乃修德振兵，治五气，蓺五种…
转折	然/而/虽…但	对比色或转折标记	虽强，不如德之固
对举	A也…B也 / 正反对比	两列grid并排	赏以春夏，刑以秋冬
总结	是以/此所以/故号	背景色块+左粗边线	诸侯咸尊轩辕为天子

第二层：段间关系

段落与段落之间的结构关系。

关系类型	识别方式	排版映射
叙事推进	时间标记递进（元年→二年→三年）	时间轴左边线
论证展开	论点→论据→结论	论点加粗，论据缩进，结论色块
对话往返	X曰…Y曰…X曰…	交替缩进（左/右对齐）
列举枚举	同类事项罗列	列表或grid卡片
插叙/背景	打断叙事的背景说明	浅灰背景或缩小字号

第三层：区块间关系

太史公曰/赞与正文之间、引用文献与叙事之间的关系。

关系类型	场景	排版映射
评论	太史公曰 vs 正文	围栏块+不同底色
引用	引《诗》《书》佐证	引用缩进+典籍样式
诗歌/韵文	赞/歌/铭	居中+硬换行
文件/诏令	嵌入的正式文书	全宽色块+边框

第四层：章间关系

不同章节之间的呼应、矛盾、互见关系。

关系类型	场景	排版映射
互见	同一事件在不同章节的记载	侧栏链接+tooltip
前后呼应	前文铺垫，后文揭晓	脚注式交叉引用
矛盾	不同章节的冲突记载	红色标记+对比面板
系谱承接	世家/本纪的世系延续	家谱图链接

第六层：表格语义分析（史记十表）

史记十表（013-022）的语义分析已独立为专项工序，见 **SKILL_02g_表格结构语义分析.md**。

涵盖内容：十表语义维度定义、三类表结构（编年/封侯/世系）、表格内部关系类型、tagged.md 标注方法、JSON/CSV 数据格式与现状、与实体库/事件库的链接。

第五层：注释与原文的对齐关系

三家注（集解/索隐/正义）及其他注释与原文之间的锚定关系。

关系类型	场景	排版映射
词语注释	注释解释原文某个词/短语	右栏注释与左栏锚点垂直对齐
句义注释	注释解释整句含义	右栏注释对齐到该句所在段落
异文校勘	注释记录不同版本的文字差异	右栏标注原字+异文，校勘badge
考证补充	注释提供地理/人物/制度等考证	右栏展开式卡片
引经据典	注释引用其他典籍佐证	典籍名高亮+外链

注释对齐机制

```
原文锚点                              右栏注释
————————                              ————————————

〖@黄帝〗者 ←— data-note="n1" ——→ [集解] 徐广曰：號有熊
                                        [索隐] 案：有土德之瑞…
                                        [正义] 輿地志云…

〖@少典〗之子 ←— data-note="n2" —→ [集解] 譙周曰：有熊國君
                                        [索隐] 少典者，諸侯國號…
```

- 原文中用 `<span class="note-anchor" data-note="nX">` 标记注释锚点
- 右栏注释用 `id="nX"` 对应
- 点击锚点 → 右栏滚动到对应注释并高亮
- 点击注释 → 左栏滚动到对应原文并高亮
- 移动端降级：注释折叠为点击展开的tooltip

三家注分色方案

注释来源	Badge颜色	左边线颜色
集解（裴骃）	绿底绿字 `#d0f0d0`	`#1a6e1a`

注释来源	Badge颜色	左边线颜色
索隐（司马贞）	橙底棕字 `#ffe8cc`	`#b45309`
正义（张守节）	蓝底蓝字 `#d0e0ff`	`#1a4fb4`

关系识别方法

自动识别（规则+LLM混合）

规则驱动（高精度）：
- 因果：检测"故/乃/於是/遂"等连接词
- 并列：检测"东…西…南…北…"或"其一…其二…"结构
- 总结：检测"是以/此所以/故曰"句式
- 时序：检测"N年"递增模式

LLM驱动（高召回）：
- 输入一个段落，让LLM标注每对相邻句子之间的关系类型
- 输出JSON：`[{sent_pair: [0,1], relation: "causal", signal: "乃"}]`

标注格式

在tagged.md文件中，语义关系用HTML注释标注（不影响strip_markup）：

```
<!-- rel:causal -->
[1.1] 蚩尤作乱，不用帝命。
<!-- rel:effect -->
於是黄帝乃徵师诸侯，与蚩尤战於涿鹿之野。

<!-- rel:parallel:4 -->
- 东至于海，登丸山，及岱宗。
- 西至于空桐，登鸡头。
- 南至于江，登熊、湘。
- 北逐荤粥，合符釜山。
```

```
<!-- rel:summary -->
[1.3] 诸侯咸尊轩辕为天子，代神农氏，是为黄帝。
```

排版映射规则

CSS类与关系类型的对应

```
.causal { }                     /* 因果容器 */
.cause  { }                     /* 原因句 */
.effect { padding-left: 1.5em;  /* 结果句：缩进+左边线 */
          border-left: 2px solid #d4c5a9; }

.parallel { display: flex; }    /* 并列：横排 */
.contrast { display: grid;      /* 对举：两列 */
            grid-template-columns: 1fr 1fr; }

.escalation { margin-left: 1.5em;  /* 递进：缩进 */
              border-left: 2px solid #c8a882; }

.summary-line { background: #f3ede3;  /* 总结：色块 */
                border-left: 3px solid #c8a882; }
```

可开关设计

语义排版通过CSS类 `.semantic-off` 控制，关闭时所有语义结构恢复为连续段落：

```
function toggleSemantic() {
    document.querySelector('.main-
text').classList.toggle('semantic-off');
}
```

原型

- `labs/prototype_marginal_notes.html` — 右侧注释+语义排版原型（五帝本纪前3段）
- `labs/排版实验_弋射说楚王.html` — 句间关系排版实验（因果/并列/递进/总结/警示）

与其他工序的关系

上游	本工序	下游
02a 章节切分（段落边界）	02d 结构语义分析（关系标注）	03c 渲染与发布（排版输出）
02b 区块处理（围栏块）	→ 区块间关系	09a 认知辅助阅读器
04a 事件识别（事件链）	→ 叙事推进关系	05a 事件关系发现
十表原始文本（013-022）	→ **第六层：表格语义分析**	04b 十表事件处理（事件提取）
tables/README.md（数据模型）	→ 表格维度定义 + 实体链接	实体库合并、tables/data/ JSON

SKILL 02e: 词法分析 — 字词级标注、统计与遗漏发现

> 对标注文本进行字词级分析：词性标注、词频统计、别名扫描、候选实体发现，为标注质量评估和实体补全提供依据。

一、任务定义

实体标注（03a）标记了102,851次实体出现，但剩余约488,674个汉字未被标注。词法分析的目标：

1. **分类未标注文字** — 区分虚词（不应标注）和候选实体（应补充标注）
2. **量化标注覆盖率** — 各类词性分布、实体密度
3. **发现遗漏实体** — 高频候选词、未标注别名、单字省称

二、已完成工作总结

2.1 字级词性分析

分类	字数	占比	说明
虚词	131,317	26.9%	之乎者也等，封闭字表（~230字）
动词	80,625	16.5%	注意词类活用
形容词	20,052	4.1%	偶尔实体化
数词	33,182	6.8%	一二三…百千万

分类	字数	占比	说明
量词	16,009	3.3%	石/斗/升/里/步
候选实体	~207,489	42.4%	需二次判断

产出：130章JSON（`doc/analysis/pos/`）+ 汇总报告（`doc/analysis/pos_summary.md`）

2.2 词频统计

统计原文（去标注后）的字频和高频词，产出：
- `doc/analysis/词频统计结果.json` — 结构化数据
- `doc/analysis/词频统计表.md` — 可读报告

2.3 未标注别名扫描

将已知别名表（entity_aliases.json，595条）与未标注文字交叉匹配，发现遗漏的别名出现。

2.4 候选实体词发现

从POS分析的"候选实体"中提取高频词：
- `doc/analysis/unknown_candidate_words.tsv` — 按出现次数排序
- 示例：`群臣(202)、左右(158)、发兵(157)` → 人工审核后决定是否补标

2.5 单字人名省称推断

从章内已标注全名中提取末字作省称候选，用句法框架过滤。
- 局限：单字与普通汉字形同，误报率高，当前不大规模使用

2.6 专项实体扫描

- **生物实体反思**：跨类型冲突检测（同时标为biology和place/astronomy）、词表外标注 → `doc/analysis/biology_scan_report.tsv`
- **身份词补标**：基于封闭词表（天子/诸侯/兄弟/子孙等）批量补标〖#〗
- **亲属词补标**：父/母/子/女/兄/弟等亲属关系词补标

三、方法论

3.1 字级规则分析（核心方法）

文言文以单字词为主（字即词），无需分词即可做词性分类：

分类	方法	准确率
虚词	封闭字表	高（字表高度封闭）
动词/形容词	半封闭字表	中（词类活用）
数词/量词	字符匹配	高
候选实体	剩余未归类	需二次判断

3.2 两遍分析

第一遍：字级分类
- 去除已标注实体（〖…〗内的内容）
- 对剩余每个汉字按字表分类

第二遍：复合实体lint
- 检测"夹心"模式：`[候选字][非候选字][候选字]`
- 识别被虚词拆开的复合实体（如"无忌"中"无"是虚词但整体是人名）

3.3 交叉匹配法（别名/词表扫描）

已有实体词表 × 未标注文字 → 交叉匹配 → 遗漏候选

四、Agent提示词模式

4.1 实体候选筛选

以下是《史记·XXX》中未被标注的高频词（出现>10次）。
请判断每个词是否应标注为实体，如果是，给出实体类型。

词表：
群臣 202次
左右 158次
发兵 157次
...

输出JSON：[{word: "群臣", should_tag: true, type: ";", reason: "官职泛称"}]

4.2 单字词性判断

以下是文言文中出现频率较高的单字。请判断每个字的主要词性。
注意文言文词类活用现象（如"王"可作名词或动词"称王"）。

字表：王(3201) 之(2890) 年(2145) ...

输出：{字："王"，词性：["名词","动词"]，说明："作名词=君王，作动词=称王/封王"}

4.3 复合实体识别

以下未标注文字序列中，哪些连续字组合可能是实体？
每个候选给出类型和置信度。

文字："乃令将军击匈奴出代郡斩首虏万馀级"

```
已标注实体：将军、匈奴、代郡（已标注，不用重复）

输出：[{text: "斩首虏", type: "非实体", confidence: 0.9}]
```

五、工具索引

脚本	功能	用法
`scripts/pos_analysis.py`	字级词性分析（130章）	`--all` / `--chapter NNN` / `--report`
`scripts/analyze_word_frequency.py`	原文词频统计	直接运行
`scripts/scan_untagged_aliases.py`	未标注别名扫描	直接运行
`scripts/infer_single_char_names.py`	单字人名省称推断	`--chapter NNN`
`scripts/scan_biology_reflect.py`	生物实体反思扫描	直接运行
`scripts/tag_identity_words.py`	身份词批量补标	`--all`
`scripts/tag_kinship_words.py`	亲属词批量补标	`--all`

六、与其他工序的关系

上游	本工序	下游
02a 章节切分（段落边界）	词性分析	03a 实体标注（遗漏发现）
03a 实体标注（已标注实体）	去标注后分析	02d 结构语义（虚词→因果/并列信号）
03b 实体消歧（别名表）	别名交叉扫描	03a 实体补标

对结构语义分析（02d）的支撑

词法分析识别出的虚词中包含大量语义连接词：
- **因果**：故、乃、遂、於是、以
- **并列**：且、又、及、与
- **转折**：然、而、虽
- **递进**：况、且、甚

这些连接词是02d自动识别句间关系的基础信号。

SKILL 02f: 文本统计 — 定量分析与分布特征

> 对标注文本进行多维度定量统计，揭示《史记》的文本规模、结构特征和实体分布规律。

一、统计维度

1.1 基础规模

指标	数值
总章节数	130篇
纯汉字	576,915字
含标点	~706,000字
实体词条	15,190个（18类）
标注次数	102,851次

1.2 五体分布

体裁	篇数	字数	占比
本纪	12	76,730	13.3%
表	10	70,525	12.2%

体裁	篇数	字数	占比
书	8	41,761	7.2%
世家	30	132,853	23.0%
列传	70	255,046	44.2%

1.3 可统计指标

类别	指标	工具
文本长度	章节字数、段落字数、句长	`analyze_tagged.py`
词频	字频、实体频次、虚词频次	`analyze_word_frequency.py`
实体密度	每百字实体数、分类占比	`analyze_tagged.py`
对话占比	引文字数/总字数	`analyze_tagged.py`
标点密度	每百字标点数	`analyze_tagged.py`

二、工具

```
python scripts/analyze_word_frequency.py          # 词频统计
python analyze_tagged.py                           # 标注文本分析
```

三、产出数据

文件	说明
`doc/analysis/史记统计分析.md`	全书统计报告

文件	说明
`doc/analysis/词频统计表.md`	高频字词表
`doc/analysis/词频统计结果.json`	结构化词频数据

四、与其他工序的关系

上游	本工序	下游
02a 切分（段落边界）	02f 文本统计	研究分析（远观/宏观模式发现）
03a 实体标注	→ 实体密度统计	标注质量评估（覆盖率）
02e 词法分析	→ 词性分布统计	语言风格分析（五体对比）

SKILL 02g: 表格结构语义分析 — 史记十表的维度定义、关系挖掘与数据转换

> 史记十表（013-022）是《史记》中唯一采用二维表格形式的篇章，与叙事类文本有本质区别：**文本本身就是结构化数据，语义不在句子关系中，而在行/列/单元格的维度定义上**。本工序负责定义十表的语义维度、标注表格内部关系，并将 HTML 表格转换为可查询的 JSON/CSV 数据。

十表概览

章号	名称	类型	时间跨度	行数	列数
013	三代世表	世系表	前2300—前841年	3段共61行	8/1/12
014	十二诸侯年表	编年表	前841—前477年	365	16
015	六国年表	编年表	前476—前207年	270	9
016	秦楚之际月表	月表	前209—前202年	2段共91行	10/21
017	汉兴以来诸侯王年表	王表	前202—前101年	106	28
018	高祖功臣侯者年表	封侯表	前202—前87年	143	9
019	惠景间侯者年表		前194—前141年	93	8

章号	名称	类型	时间跨度	行数	列数
		封侯表			
020	建元以来侯者年表	封侯表	前140—前87年	119	8
021	建元已来王子侯者年表	封侯表	前127—前87年	162	8
022	汉兴以来将相名臣年表	编年表	前206—前20年	187	6

语义维度定义

三个基本维度

维度	含义	语义角色
行维度	时间轴（年/月）或实体轴（人/国）	定义"描述对象"
列维度	并列的政治实体（国、职位）或属性（功绩、封地）	定义"描述视角"
单元格	具体事件/人物/状态	承载实际语义内容

三类表的语义结构

编年表（014 / 015 / 016 / 022）

行 = 时间切片（年 / 月）
列 = 并列政治实体（齐、楚、燕…）或职位（丞相、将军…）
单元格语义：
- 君主名 → 当前执政者
- 事件描述 → 该时期该实体的重大事件
- 空单元格 → 无记录（非"无事件"）

语义关系：
- 同行跨列 = **共时并列**（同年各国并列视角）
- 同列跨行 = **时序推进**（一国/一职位的历史纵线）
- 对角线读法 = **跨国互动**（如战争须同时看双方列）

封侯表（018 / 019 / 020 / 021）

行 = 侯国（每侯一行）
列 = 属性维度（封地、户数、功绩、传承链、国除原因…）
单元格语义：
- 初封信息列 → 建立节点
- 传承链列（子继/孙继/外孙继） → 世代传递关系
- 国除原因列（酎金/谋反/无后…） → 终止节点

语义关系：
- 同行读法 = **侯国生命周期**（从封到除的完整叙事）
- 同批次跨行 = **同年封侯群体**（政治事件的集体行动）
- 国除原因分布 = **制度史语义**（如酎金案的集体国除）

世系表（013）

013 三代世表含**三个独立子表**，结构各异：

子表	行维度	列维度	特点
五帝·夏世系	世代层级（黄帝→桀）	颛顼/喾/尧/舜/夏/殷/周七支系	多支系并行，空格表示无传承
殷世系	帝王顺序（外丙→纣）	单列	线性序列，每行含事件注记
周·诸侯世系	周王世代（成王→共和）	鲁/齐/晋/秦/楚/宋/卫/陈/蔡/曹/燕	横向同世代并列

```
纵向 = 传承链（父→子→孙）
横向 = 同世代并列（兄弟/同代各国）
跨支系连线 = 姻亲/分封关系
```

王表（017）

```
行 = 年份
列 = 各诸侯王国（最多28国，含楚、齐、荆、淮南、燕、赵、梁等）
单元格语义：
  - 王国名 → 该年该王国存在
  - 空格 → 该年该王国已除/未立
```

表格内部关系类型

关系类型	场景	标注方式
传承	侯国父子相继	`rel:succession`
共时	同年多国并列	`rel:concurrent`
终止	国除/灭国	`rel:termination reason="酎金"`

关系类型	场景	标注方式
分封批次	同日封多人	`rel:batch date="前201年"`
互动	跨列的战争/联盟	`rel:interaction parties="齐,楚"`

表格语义标注方法

在 tagged.md 中，表格区块用 `<!-- block:table -->` 围栏标记：

```
<!-- block:table type="chronological" id="014" -->
<!-- table:row year="-841" era="共和元年" -->
| 周 | 鲁 | 齐 | 晋 | 秦 | 楚 |
|---|---|---|---|---|---|
| 共和行政 | 真公濞元年 | 武公寿元年 | 靖侯司徒七年 | 秦侯（无名）三年 | 熊渠卒，熊挚红立 |
<!-- /table:row -->
<!-- /block:table -->
```

单元格实体链接：

```
| [@周厉王](person:周厉王) | [@真公濞](person:真公濞) |
```

数据存储与转换

文件结构

```
resources/table_html/016_秦楚之际月表_table.html   ← 来源 HTML
  ↓ tables/convert_tables.py（MultiTableParser）
tables/data/016_秦楚之际月表.json                   ← JSON 输出
tables/data/016_秦楚之际月表.csv                    ← CSV 输出（UTF-8
BOM）
```

转换脚本：`tables/convert_tables.py`，支持单段和多段（如013三段）结构。

JSON 格式

单段表（大多数表）：

```
{
  "table_info": {
    "id": "016",
    "title": "秦楚之际月表",
    "columns": ["公元前", "秦", "西楚", "汉", ...],
    "row_count": 91
  },
  "rows": [
    {"公元前": "209", "秦": "二世元年", "西楚": "", ...},
    {"公元前": "",    "秦": "七月",     "西楚": "楚隐王陈涉起兵入
秦。", ...}
  ]
}
```

多段表（013三代世表）：

```
{
  "table_info": {
    "id": "013",
    "title": "三代世表",
```

```
    "section_count": 3,
    "total_rows": 61
  },
  "sections": [
    {
      "section_title": "五帝·夏世系",
      "columns": ["帝王世国号", "颛顼属", "喾属", ...],
      "row_count": 22,
      "rows": [...]
    },
    {
      "section_title": "殷世系",
      "columns": ["帝王"],
      "row_count": 29,
      "rows": [...]
    },
    {
      "section_title": "周·诸侯世系",
      "columns": ["周", "鲁", "齐", ...],
      "row_count": 10,
      "rows": [...]
    }
  ]
}
```

CSV 格式规则

- 编码：`utf-8-sig`（含 BOM，兼容 Windows Excel）
- 空单元格保留为空字符串（不填充、不推断）
- 行数不足表头列数时末尾补空字符串
- 多段表：每段前加 `## 分段标题` 行，段间空行分隔
- 跨行/跨列合并单元格**不展开**，以空值填充（保留原表位置信息）

数据现状

文件	行数	列数	主要列
013_三代世表	3段共61行	8/1/12	①五帝·夏世系(8列)；②殷世系(1列)；③周·诸侯世系(12列)
014_十二诸侯年表	365	16	公元前、干支、周、鲁、齐、晋、秦、楚、宋、卫、陈、蔡、曹、郑、燕、吴
015_六国年表	270	9	公元前、秦、魏、韩、赵、楚、燕、齐
016_秦楚之际月表	2段共91行	10/21	①秦二世时期(10列)；②项羽分封后(21列)
017_汉兴以来诸侯王年表	106	28	公元前、纪年、楚、齐、荆、淮南、燕、赵、梁、淮阳、代、长沙等
018_高祖功臣侯者年表	143	9	国名、侯功、高祖十二、孝惠七、高后八、孝文二十三、孝景十六、建元至后元、侯第
019_惠景间侯者年表	93	8	同018类（惠帝景帝间封侯）
020_建元以来侯者年表	119	8	同018类（武帝建元后封侯）
021_建元已来王子侯者年表	162	8	同018类（推恩令后王子封侯）
022_汉兴以来将相名臣年表	187	6	公元前、纪年、大事纪、相位、将位、御史大夫位

与知识库的链接

表格是实体库和事件库的另一个数据源，与叙事类章节互补：

```
表格单元格
  ↓ 实体链接
实体库（persons/places/offices）
  ↓ 合并去重
主实体索引

表格行/批次
  ↓ 事件提取（见 SKILL_04b_十表事件处理）
事件库
  ↓ 年代标注
timeline
```

当前格式（扁平行结构）是原始数据层（保真转换）。更丰富的语义模型（编年表 `timeline_data[]`、封侯表 `persons[]`、世系表 `genealogy{}`）是未来的语义增强层，需在此基础上进一步解析构建。

与其他工序的关系

上游	本工序	下游
十表 HTML 原始文件（`resources/table_html/`）	02g 表格结构语义分析（维度定义 + 格式转换）	`tables/data/` JSON/CSV
`tables/convert_tables.py`（行列提取）	→ 表格语义标注	03a 实体标注（单元格实体）
02d 结构语义分析（四层关系框架）	→ 表格专属关系类型	04b 十表事件处理（事件提取）

上游	本工序	下游
tagged.md 表格区块	→ `<!-- block:table -->` 标注	实体库合并、事件库年代标注

SKILL 02h: 词表构建 — 专项词汇识别与结构化

> 从标注文本中提取特定领域的词汇表，包括成语典故、专有名词、术语等，构建结构化词表并定位到原文段落。

一、任务定义

词表构建是从史记全文中系统地提取和组织特定类型的词汇，包括：

1. **成语典故** — 成语、典故、经典名句的识别、定位与释义
2. **专有名词** — 官职、地名、族群、制度等封闭词表
3. **术语体系** — 礼制、历法、度量衡等专业术语
4. **文化词汇** — 器物、服饰、音乐、祭祀等文化概念

与词法分析（02e）的区别：
- **02e 词法分析**：字词级分类、词性标注、统计分析（面向语言学）
- **02h 词表构建**：领域词汇提取、语义标注、知识组织（面向知识图谱）

二、已完成工作总结

2.1 成语典故词表

数据来源：`kg/vocabularies/data/史记成语典故.md`（351条原始数据）

处理流程：
1. **文本清理** — 处理17种实体标注标记（`〖@人名〗`等）
2. **模糊匹配** — 基于Levenshtein编辑距离，滑动窗口搜索
3. **段落定位** — 将成语定位到具体章节和段落编号（Purple Numbers）

4. **关键词提取** — 自动提取成语关键字并在原文中高亮
5. **质量控制** — 排除括号概括（后人总结）和非史记来源（《战国策》）

最终成果:
- **340条成语** — 100%定位到史记原文
- **涵盖70章** — 占全书54%
- **TOP章节**：项羽本纪(22条)、秦始皇本纪(18条)、五帝本纪(15条)

技术亮点:
- **文本清理算法** — 移除实体标注但保留内容，处理标点、引号、段落编号
- **模糊匹配** — 相似度阈值：80%（短语≤5字），70%（长文）
- **手动修正** — 修正7条章节标注错误（如"唇亡齿寒" 043→039）

输出文件:

```
data/chengyu.json           # 结构化JSON数据（340条）
docs/special/chengyu.html   # HTML网页（带实体标注着色）
docs/special/chengyu.pdf    # PDF文档（8.4MB）
```

数据结构:

```
{
  "chapter_num": "007",
  "chapter_title": "项羽本纪",
  "chengyu": "破釜沉舟",
  "original": "项羽乃悉引兵渡河，皆沉船，破釜甑",
  "meaning": "形容下定决心，不顾一切",
  "context": "...[上下文200字]...",
  "paragraph": "15.2"
}
```

三、子任务规划

3.1 成语典故识别（已完成 ✅）

输入：`kg/vocabularies/data/史记成语典故.md`
输出：340条成语，100%定位到原文
脚本：
- `scripts/extract_chengyu.py` — 提取与定位
- `scripts/apply_manual_fixes.py` — 手动修正
- `scripts/render_chengyu_html.py` — HTML渲染

3.2 官职词表构建（待开发）

目标：提取并分类史记中的所有官职名称

数据来源：
- 已标注的`〖;官职〗`实体（约X条）
- 未标注的候选官职词（从02e词法分析获取）

分类体系：
- **中央官职**：丞相、太尉、御史大夫等
- **地方官职**：郡守、县令、刺史等
- **军职**：将军、都尉、校尉等
- **宫廷官职**：宦者、郎中、侍中等
- **特殊官职**：博士、太史、太卜等

输出：

```
data/guanzhi.json           # 官职词表（含释义、出现次数）
docs/special/guanzhi.html   # 官职体系可视化
```

3.3 地名词表构建（待开发）

目标：提取并地理标注史记中的所有地名

数据来源：
- 已标注的`〖=地名〗`实体（约X条）
- 地名消歧数据（`kg/entities/disambiguation/place.json`）

地理标注:
- **古今对照**: 古地名 → 今地名
- **坐标标注**: 经纬度（基于谭其骧《中国历史地图集》）
- **行政层级**: 国/郡/县/乡
- **地名演变**: 同一地点的历代名称

输出:

```
data/diming.json            # 地名词表（含坐标、古今对照）
docs/special/diming.html    # 地名地图可视化
```

3.4 礼制术语词表（待开发）

目标: 提取礼制、祭祀、典礼相关术语

分类:
- **祭祀**: 封禅、郊祀、宗庙、社稷
- **礼仪**: 朝觐、会盟、燕飨、冠礼
- **丧葬**: 谥号、庙号、陵寝、祭祀
- **音乐**: 雅乐、俗乐、乐器、乐律

3.5 度量衡词表（待开发）

目标: 提取并换算史记中的度量衡单位

分类:
- **长度**: 里、步、尺、寸
- **容量**: 石、斗、升、合
- **重量**: 斤、两、铢
- **货币**: 金、钱、刀、布

现代换算:

```
{
  "unit": "里",
  "type": "length",
  "qin_value": "约415米",
```

```
  "han_value": "约350-450米",
  "occurrences": 2145
}
```

3.6 族群词表（待开发）

目标：提取史记中的族群、部落、民族名称

分类：
- **华夏诸国**：齐、楚、燕、赵、韩、魏、秦
- **边疆族群**：匈奴、东胡、西羌、南越、闽越
- **古代部落**：鬼方、猃狁、戎狄

四、通用工作流程

4.1 词汇收集

方法一：基于已标注实体

```
# 从标注文件中提取特定类型实体
grep -oP '〖;[^〗]+〗' chapter_md/*.tagged.md | sort | uniq -c
```

方法二：基于候选词表
- 从02e词法分析的候选实体中筛选
- 人工审核 + Agent批量判断

方法三：基于外部知识
- 参考《史记索隐》《史记正义》
- 对照《汉书》《说文解字》等典籍

4.2 定位与标注

核心算法：文本清理 + 模糊匹配

```
def clean_text(text):
    """清理实体标注、标点、段落编号"""
    # 移除实体标注但保留内容：〖@人名〗 → 人名
    text = re.sub(r'〖[@=^%•{&\'~#+?;!$:\[\]_\\]([^〗]+)〗', r'\1',
text)
    # 移除段落编号 [N] 或 [N.M]
    text = re.sub(r'\[\d+(?:\.\d+)?\]\s*', '', text)
    # 统一标点、引号
    text = re.sub(r'["""\'\'\'`]', '', text)
    return text

def fuzzy_find(pattern, text, threshold=0.7):
    """基于Levenshtein距离的模糊匹配"""
    # 滑动窗口搜索（窗口大小 = pattern长度 ± 30%）
    # 返回最佳匹配位置和相似度
```

4.3 释义与标注

释义来源:
1. **现代工具书** — 《汉语大词典》《辞海》
2. **古注** — 裴骃集解、司马贞索隐、张守节正义
3. **AI生成** — Claude提供释义初稿，人工校对

结构化标注:

```
{
  "word": "词条",
  "type": "类型",
  "meaning": "释义",
  "chapter_num": "章节号",
  "paragraph": "段落号",
  "original": "原文引用",
  "context": "上下文",
  "notes": "备注"
}
```

4.4 可视化与发布

HTML渲染:
- 实体标注着色（17种颜色）
- 关键词加粗高亮
- 段落编号链接（点击跳转原文）

PDF导出:
- 使用WeasyPrint生成高质量PDF
- 保留HTML样式和颜色

数据发布:
- JSON格式（供程序调用）
- Markdown格式（供人类阅读）
- 集成到专项索引总页（`docs/special/special_index.html`）

五、质量控制

5.1 定位率要求

- **目标**：≥95% 的词条定位到原文段落
- **手段**：模糊匹配 + 手动修正
- **例外**：括号概括（后人总结）、外典来源（如《战国策》）应排除

5.2 释义准确性

- **多源校验**：至少2个工具书来源
- **古注对照**：与三家注释内容一致
- **专家审核**：疑难词条请教古文献学者

5.3 数据一致性

- **章节编号** — 三位数格式（001-130）
- **段落编号** — Purple Numbers格式（[N] 或 [N.M]）
- **实体标注** — 与kg/entities/数据一致

六、Agent提示词模式

6.1 词条释义生成

以下是《史记》中的词条，请给出简明释义（不超过20字）。

词条：破釜沉舟
原文：项羽乃悉引兵渡河，皆沉船，破釜甑
上下文：[提供200字上下文]

输出格式：
释义：形容下定决心，不顾一切地战斗
典故：项羽渡河前砸破做饭的锅，凿沉渡河的船，示意士兵决一死战

6.2 词条分类判断

以下词条应归入哪个类别？

词条：丞相
候选类别：中央官职 / 地方官职 / 军职 / 宫廷官职 / 特殊官职

分析：丞相为秦汉最高行政长官，辅佐皇帝处理政务
输出：中央官职

6.3 批量匹配验证

以下是自动匹配的词条定位结果，请验证是否正确。
如有错误，给出正确的章节和段落。

词条：唇亡齿寒
原记录：043_赵世家
原文片段："脣之与齿，脣亡则齿寒"

```
自动定位：039_晋世家 [30.33]

判断：[正确 / 错误 / 两处都有]
```

七、工具索引

脚本	功能	用法
`scripts/extract_chengyu.py`	成语提取与定位	直接运行
`scripts/apply_manual_fixes.py`	手动修正章节错误	直接运行
`scripts/render_chengyu_html.py`	成语HTML渲染	直接运行
`scripts/build_guanzhi_vocab.py`	官职词表构建（待开发）	`--output data/guanzhi.json`
`scripts/build_diming_vocab.py`	地名词表构建（待开发）	`--with-coords`

八、与其他工序的关系

上游	本工序	下游
02e 词法分析（候选词发现）	词表构建	06 本体构建（术语体系）
03a 实体标注（已标注实体提取）	词条定位	09a 认知辅助阅读器（词条注释）

上游	本工序	下游
03b 实体消歧（地名/官职消歧）	释义标注	应用层（词典查询、知识问答）

对专项索引（special/）的贡献

词表构建为专项索引提供了内容支撑：
- **成语典故** — `docs/special/chengyu.html`（已完成）
- **太史公曰** — `docs/special/taishigongyue.html`（已完成）
- **韵文** — `docs/special/yunwen.html`（已完成）
- **官职词典** — `docs/special/guanzhi.html`（待开发）
- **地名地图** — `docs/special/diming.html`（待开发）

九、成语典故子任务详细说明

9.1 数据清洗流程

原始数据问题：
1. 括号概括 — 9条（如"（萧何定法，曹参守而勿失）"）
2. 非史记来源 — 2条（"狡兔三窟"出自《战国策》）
3. 章节标注错误 — 7条（如"唇亡齿寒"记为043实为039）

清洗策略：

```
# 1. 排除括号概括
if original.startswith('（') and original.endswith('）'):
    remove_item()

# 2. 排除非史记来源
if verify_source() == "战国策":
    remove_item()

# 3. 修正章节错误
apply_manual_fixes({
```

```
    "唇亡齿寒": ("039", "晋世家", "30.33"),
    "过犹不及": ("067", "仲尼弟子列传", "52"),
    ...
})
```

9.2 模糊匹配算法参数

相似度阈值:
- 短语（≤5字）：80%
- 长文（>5字）：70%

滑动窗口:
- 窗口大小 = 模式长度 × (0.7 ~ 1.3)
- 步长 = 1字符

特殊处理:
- 省略号拆分：`text.split('…')[0]` 取首段匹配
- 实体标注穿透：`〖@人名〗` → `人名`

9.3 关键词高亮算法

提取规则:

```
# 1. 拆分复合成语（如"浑沌/穷奇/梼杌/饕餮"）
if '/' in chengyu:
    keywords = chengyu.split('/')

# 2. 提取2-4字关键词
elif len(chengyu) >= 4:
    keywords = [chengyu[:2], chengyu[2:4], chengyu]

# 3. 按长度排序（长的优先匹配）
keywords.sort(key=len, reverse=True)
```

高亮样式:

```
.item-original strong {
    color: #8b4513;           /* 棕色 */
    font-weight: 700;         /* 粗体 */
    text-decoration: underline;
    text-decoration-color: #d4a373;
    text-decoration-thickness: 2px;
}
```

9.4 输出统计

指标	数值	说明
成语总数	340	排除括号和战国策后
定位率	100%	所有条目均定位到原文
涵盖章节	70/130	53.8%
JSON文件大小	130KB	结构化数据
HTML文件大小	280KB	含实体标注着色
PDF文件大小	8.4MB	高质量排版
平均上下文长度	~200字	段落前后各100字

TOP 5 成语章节:
1. 项羽本纪 — 22条（破釜沉舟、四面楚歌、鸿门宴、项庄舞剑等）
2. 秦始皇本纪 — 18条（指鹿为马、图穷匕见、焚书坑儒等）
3. 五帝本纪 — 15条（生而神灵、垂拱而治等）
4. 周本纪 — 15条（烽火戏诸侯、桐叶封弟等）
5. 孔子世家 — 15条（韦编三绝、食不厌精等）

十、未来扩展

10.1 多语言词表

- **拉丁化** — Wade-Giles罗马化拼音
- **英文释义** — 面向国际学者
- **日文对照** — 史记汉字在日本的训读

10.2 词表关联

- **成语 → 人物** — "破釜沉舟" → 项羽
- **成语 → 事件** — "鸿门宴" → 前206年鸿门会
- **成语 → 地点** — "围魏救赵" → 大梁、邯郸

10.3 词表对比

- **史记 vs 汉书** — 成语源流对比
- **史记 vs 现代汉语** — 词义演变追踪

十一、参考资源

工具书:
- 《史记成语典故词典》
- 《汉语大词典》
- 《中国古代官职词典》
- 谭其骧《中国历史地图集》

学术文献:
- 王力《古代汉语》
- 裴学海《古书虚字集释》
- 杨伯峻《古汉语虚词》

数字资源:
- 汉典（zdic.net）

- 教育部异体字字典
- 中央研究院汉籍电子文献

SKILL 03: 实体构建 — 从结构化文本到实体索引

> 对结构化文本进行18类实体标注（v2.5），消歧同名异指，生成实体索引和可视化页面。

子工序

编号	文件	内容
03a	SKILL_03a_实体标注.md	18类实体NER标注（LLM驱动，v2.5标注体系）
03b	SKILL_03b_实体消歧.md	同名异人/异名同人消歧
03c	SKILL_03c_实体标注反思.md	按章/按类型多轮反思审查
03d	SKILL_03d_渲染与发布.md	HTML渲染 + 实体索引 + GitHub Pages发布
03e	SKILL_03e_按类型反思.md	按实体类型的系统性审查与修正

SKILL 03a: 古籍实体标注 — 从原文到结构化实体的完整方法

> 从《史记》130篇×57万字的实体标注实践中提炼，适用于大规模古籍命名实体识别（NER）与消歧。

一、标注体系设计

1.1 核心原则

"内容与样式分离" — 用轻量Token标记原文，渲染交给程序。

Token标记的优势：
- 原文保持可读（对比HTML `<span>` 标注，标注后的markdown仍可直接阅读）
- 版本可控（git diff友好，纯文本差异清晰）
- 渲染无关（同一份标注文件可渲染为HTML、PDF、LaTeX等）

1.2 18类实体标记符号

原始11类（v1.0）+ 2026-03-13扩展4类（v1.1）+ 2026-03-13迁移为〖〗格式（v2.0）+ 封国类型（v2.1）+ 身份类型（v2.3）+ 数量类型（v2.5）。

符号	类型	说明	示例
〖@X〗	人名	历史人物、传说人物	〖@刘邦〗、〖@项羽〗、〖@台|吕台〗
〖=X〗	地名	城邑、山川、国名	〖=长安〗、〖=泰山〗
〖;X〗		正式任命的职衔、封号	〖;丞相〗、〖;太尉〗

符号	类型	说明	示例
	官职		
〖#X〗	身份	社会角色、地位、血缘关系（非正式官职）	〖#天子〗、〖#太后〗、〖#外戚〗
〖%X〗	时间	年号纪年、月份	〖%元年〗、〖%三月〗
〖&X〗	氏族	宗族、家族、姓氏集团	〖&刘氏〗、〖&嬴氏〗
〖'X〗	邦国	统一王朝、诸侯国、外邦政权	〖'汉〗、〖'齐〗、〖'匈奴〗
〖^X〗	制度	典章制度、礼法规范	〖^郡县制〗、〖^封禅〗
〖~X〗	族群	民族、部落	〖~匈奴〗、〖~三苗〗
〖•X〗	器物	礼器、兵器	〖•宝鼎〗、〖•印绶〗
〖!X〗	天文	星象、历法	〖!岁星〗、〖!闰月〗
〖?X〗	神话	传说、神话事件	〖?黄帝〗、〖?女娲〗
〖+X〗	生物	名词性生物	〖+龙〗、〖+百穀〗
〖{X〗	典籍	古籍书名（v1.1新增）	〖{春秋〗、〖{史记〗

符号	类型	说明	示例
〖:X〗	礼仪	礼仪制度、宗庙祭祀（v1.1新增）	〖:宗庙〗、〖:籍田〗
〖[X〗	刑法	刑罚、法律条文（v1.1新增）	〖[族诛〗、〖[腰斩〗
〖_X〗	思想	哲学概念、文体体裁（v1.1新增）	〖_仁义〗、〖_本纪〗
〖$X〗	数量	非时间性计量：军队/距离/重量/金额	〖$三万人〗、〖$千里〗

v2.7　内联消歧语法扩展到所有实体类型（2026-03-16），v2.8　五类专用括号迁移至〖TYPE〗（2026-03-16）：

- 语法 〖TYPE 显示名|规范名〗：显示短名，链接/tooltip/索引用规范名
- 适用 18 种 〖TYPE〗 实体（人名/地名/官职/时间/氏族/邦国/制度/族群/身份/天文/生物/数量/器物/神话/典籍/礼仪/刑法/思想）
- 优先级高于 disambiguation_map.json，标注即消歧，无需维护 JSON

类型	示例	用途
人名 @	〖@台|吕台〗、〖@桓公|齐桓公〗	单字省称、同名消歧
地名 =	〖=函谷|函谷关〗、〖=沛|沛县〗	地名简称
官职 ;	〖;太尉|太尉职〗、〖;守|郡守〗	官职消歧
邦国 '	〖'汉|大汉〗、〖'楚|楚国〗	邦国消歧
氏族 &	〖&羋|羋姓〗、〖&嬴|嬴姓〗	氏族正名
时间 %	〖%元|元年〗	年份省称

类型	示例	用途
器物 `•`	`〖•鼎|九鼎〗`	器物简称
制度 `^`	`〖^郡县|郡县制〗`	制度全名
族群 `~`	`〖~胡|匈奴〗`	族群全称
身份 `#`	`〖#外戚|外戚家族〗`	身份消歧
天文 `!`	`〖!彗|彗星〗`	天象简称
生物 `+`	`〖+龙|蟠龙〗`	生物消歧
数量 `$`	`〖$千|千金〗`	数量消歧

- 渲染输出：`<span class="TYPE" title="类型：规范名" data-canonical="规范名">显示名</span>`
- 索引：以规范名（`|` 后）入索引，显示名（`|` 前）显示在正文

v2.0迁移（2026-03-13）：
- 9类对称符号（`@人名@` 等）迁移为 `〖TYPE content〗` 统一格式（白色六角括号+类型标记）
- 官职类型标记从 `$` 改为 `;`（避免Markdown math渲染冲突）
- 嵌套标注拍平：`〖;〖@安国君〗〗` → `〖;安国君〗`（外层保留，内层标记去除）
- 迁移脚本：`scripts/migrate_to_lenticular.py`

v2.1 邦国/氏族类型（2026-03-14）：
- `〖'X〗` = **邦国**：一切政治实体——统一王朝（`〖'汉〗`、`〖'周〗`）、诸侯封国（`〖'齐〗`、`〖'楚〗`）、外邦政权（`〖'匈奴〗`、`〖'大宛〗`）
- `〖&X〗` = **氏族**：以血缘/姓氏为纽带的宗族集团（`〖&刘氏〗`、`〖&嬴氏〗`、`〖&有扈氏〗`）
- 与 `〖~族群〗` 的区别：族群=以文化/地域/民族定义（匈奴人），氏族=以宗法血缘定义（嬴氏宗族）
- 秦：前221年前 = `〖'秦〗`（封国），前221年后 = `〖'秦〗`（统一王朝，仍是邦国）
- 迁移脚本：`scripts/migrate_dynasty_to_feudal.py`

v2.3 身份类型（2026-03-14）：
- 〖#X〗= **身份**：有制度含义的社会/政治角色标签
- 与〖;官职〗的区别：官职=朝廷正式册封的职位（丞相/太尉/御史大夫），身份=社会角色（天子/太后/外戚/食客/布衣）
- **不标注亲属称谓**：父子/兄弟/长子/少子/中子/苗裔/子孙 等纯粹描述家庭关系或血缘溯源的词，即使带"关系"义也不标。区别：太子有法定制度地位 → 标；长子仅说明排行 → 不标
- 身份词表：`kg/entities/data/identity_wordlist.json`
- 迁移脚本：`scripts/migrate_official_to_identity.py`

v2.5 数量类型（2026-03-14）：
- 〖$X〗= **数量**：非时间性计量——军队规模（三万人）、距离（千里）、度量（三寸/五丈）、金额（千金/万钱）、行政户数（千户/二千石）、城邑数（十城/七十馀城）
- 与〖%时间〗的区别：时间=年号纪年/月份/日期/时长（三年/正月/岁馀），数量=物理计量/人数/距离/重量/货币
- 不标注：代词性用法（朕一人）、年龄（年二十）、时长（数十年）、历法结构数（历书月数）、虚指（千秋万岁）
- 数量词表：`kg/quantity/data/quantity_wordlist.json`（281词条，6大类）
- 标注脚本：`scripts/tag_quantity_entities.py`（掩码匹配+长词优先）
- 实施方法：词表自动标注(690处) → 跨类型迁移〖%〗→〖$〗(1,894处) → LLM反思清洗(146处去标) → 最终2,438处

1.3 实体分类核心原则

原则一：具体的人是人名，对应多人是官职

模式	判断	归类	示例
国+谥号+爵	特指唯一一人	人名〖@〗	〖@齐桓公〗、〖@晋文公〗、〖@秦昭王〗
国+爵（无谥号）	泛称，可指任一任	官职〖;〗	〖;吴王〗、〖;燕王〗、〖;卫君〗
国+公子/王子	"公子"是泛称	官职〖;〗	〖;楚公子〗、〖;魏公子〗

模式	判断	归类	示例
姓+太后/皇后	特指某人	人名 〖@〗	〖@吕太后〗、〖@窦皇后〗
太后/皇后（无姓）	泛称身份	身份 〖#〗	〖#太后〗、〖#皇后〗

原则二：正式任命 vs 社会角色 vs 亲属称谓

归类	判断标准	示例
官职 〖;〗	朝廷正式册封/任命的职位	丞相、太尉、御史大夫、郡守、县令
身份 〖#〗	有制度含义的社会/政治角色标签	天子、太后、诸侯、外戚、食客、布衣、太子、大臣
不标注	纯亲属关系描述词，无政治/制度含义	父子、兄弟、长子、少子、中子、苗裔、子孙

身份 vs 亲属称谓的关键区分：

- 〖#太子〗 √ — 法定政治头衔，有继承权，有制度地位
- 长子 ✗ — 排行描述，仅说明出生顺序，如"生子三人：长子宣公，中子成公，少子穆公"中的长子/中子/少子均不标
- 〖#诸侯〗 √ — 封建等级称谓，有政治含义
- 父子 ✗ — "父子俱以材力事殷纣"里 父子 = "两人，关系为父子"，是关系陈述非身份
- 兄弟 ✗ — "我兄弟多"是亲属关系陈述，不标
- 苗裔 ✗ — "帝颛顼之苗裔孙"是血缘溯源词，描述辈分关系，不标
- 子孙 ✗ — "後子孙饮马於河"是后代泛称，不标

原则三：邦国不存人名

〖'邦国〗　仅标注政治实体本身。"赵王"、"齐桓公"等是人对国的引用，不是国家实体。
- ~~〖'齐〗〖;桓公〗~~ → 〖@齐桓公〗（合并为人名）
- ~~〖'吴〗〖;王〗~~ → 〖;吴王〗（合并为官职）

原则四：标注完整性

每个〖•〗〖;〗等标记必须完整包裹一个实体词，不得跨越标点或吞噬句子。
- ✓ 〖•驷马〗— 正确
- ✗ 〖•驷〗马〖•，迎梁王於关下...〗— 错误（标记边界断裂，吞噬后文）

原则五：数量与时间的边界

〖$数量〗仅标注非时间性计量。判断顺序：
1. 以年/月/日/岁结尾 → 〖%时间〗（三年、正月、岁馀）
2. 年号/干支/季节开头 → 〖%时间〗（元光、甲子、春）
3. 以量词结尾（人/里/城/户/骑/乘/斤/金/尺/丈/寸/匹等）→ 〖$数量〗
4. 代词性用法 → 不标注（朕一人 = I alone）
5. 年龄（年二十/年五十）→ 不标注
6. 时长（数十年/千馀岁）→ 不标注
7. 历法结构数（历书中的月数十二/十三）→ 不标注

1.4 君主实体的特殊处理

君主（帝王、诸侯、藩王）是《史记》中最重要的人名实体类，也是歧义最严重的类群，需要特殊处理。

标注原则

君主的标注依循"具体的人是人名，对应多人是官职"的核心原则：

场景	标注示例	说明
国+谥号全称（特指一人）	〖@齐桓公〗、〖@秦昭王〗	**人名**，唯一标识某位君主
谥号称呼（本章上下文）	〖@武王〗、〖@桓公〗	**人名**，须消歧（见 SKILL_03b_实体消歧.md）
个人名		**人名**，以别名形式存在

场景	标注示例	说明
	〖@嬴政〗、〖@刘邦〗	
国+爵（无谥号，泛称）	〖;吴王〗、〖;燕王〗	**官职**，可指多人
天子/太后等通称	〖#天子〗、〖#太后〗	**身份**，社会角色

四级类型体系

《史记》君主分四个等级（存储在 `kg/relations/rulers.json`，651条）：

类型	典型例子	标注时的识别特征
天子/皇帝	周武王、秦始皇帝、汉高祖	所属国为夏/商/周/汉，或秦皇帝
诸侯王	楚武王、秦昭王、西楚霸王	谥号含"王"，但非天子
诸侯侯君	齐桓公、晋文公、鲁隐公	谥号含公/侯/伯，或无爵称
汉诸侯王	楚元王刘交、齐悼惠王刘肥	汉代藩王，臣属于汉帝

与 rulers.json 的关联

- **数据库位置**：`kg/relations/rulers.json`（651条，字段：名/谥号/别名/所属国/类型/登基年/退位年等）
- **消歧利用**：可用 `(所属国, 谥号)` 唯一确定一位君主
- **别名系统**：君主别名（如\"沛公\"→\"汉高祖\"）维护在 `entity_aliases.json` 中
- **在位年辅助消歧**：当同一短名有多个候选时，用上下文年份与 `登基年/退位年` 区间匹配

1.4 标注规则

标什么：
- 名词性用法的实体（"秦灭六国"中的"秦"标为 `〖&秦〗`）
- 同一段落中首次出现必须标注，后续可选
- 嵌套标注允许（外层保留，内层标记去除）：`〖;安国君〗`（官职包裹人名，拍平后取官职标记）

不标什么：
- 动词/形容词用法（"蚕食"的"蚕"不标）
- 比喻用法（"龙颜"不标〖+〗）
- 普通名词（"山"、"水"不标，除非是专有地名）

⚠️ 铁律：标注只能添加标记符号，不得修改原文字符
- 标注操作仅在原文字符上添加 `〖TYPE 〗` 标记，不得增加、删除或替换任何原文字符
- 禁止将省称/合称拆解为全称后再标注。例如"文成之世"中的"文成"是对文公和成公的合称省写，**不得**拆分为 `〖@文公〗〖@成公〗`（这会插入原文没有的"公"字）
- 禁止将全称缩写为省称后标注。例如原文"刘邦"不得标为 `〖@高祖〗`
- 验证方法：将标注文件去除所有 `〖TYPE 〗` 符号后，所得纯文本必须与原始 `.txt` 文件逐字相同

二、标注流程

2.1 五阶段流水线

```
原始文本(.txt)
    ↓ 阶段1: 拆分 + 段落编号
章节文件(chapter_md/NNN_title.md)
    ↓ 阶段2: AI语义标注（Claude）
标注文件(chapter_md/NNN_title.tagged.md)
    ↓ 阶段3: 质量控制（lint + validate）
验证通过的标注文件
    ↓ 阶段4: 实体提取 + 索引构建
```

```
entity_index.json + entity_aliases.json
    ↓ 阶段5: 消歧 + 年份映射
disambiguation_map.json + year_ce_map.json
```

2.2 阶段1：文本准备

> 篇章拆分、段落编号（Purple Numbers）和小节划分的详细方法，见 **SKILL_02a_章节切分与编号.md**。

前置条件：实体标注开始前，文本必须已完成：

- 篇章拆分（130个独立文件）
- Purple Numbers段落编号（`[1]`、`[1.1]`等）
- 语义小节划分（二/三级标题）

2.3 阶段2：AI语义标注

批量处理模式：

```
SYSTEM_PROMPT = """你是《史记》标注专家。
## 输出要求
1. 使用18类Token标记所有实体
2. 添加[章.节]段落编号
3. 按内容语义划分小节（二级/三级标题）
4. 直接输出Markdown，不加解释
## 实体标注规则（18类，v2.5格式）
- 〖@人名〗 〖=地名〗 〖;官职〗 〖%时间〗 〖'邦国〗 〖&氏族〗
- 〖^制度〗 〖~族群〗 〖•器物〗 〖!天文〗 〖?神话〗 〖+生物〗
- 〖#身份〗 〖$数量〗 〖{典籍〗 〖:礼仪〗 〖[刑法〗 〖_思想〗
## 特殊注意
- 标注名词性用法，不标动词/形容词/比喻
- 同段落首次出现必须标注
- 允许嵌套（外层保留，内层标记去除）：〖;安国君〗（安国君为官职+人名，取外层官职）
- 书名用〖{〗，不用〖^制度〗（如〖{春秋〗而非〖^春秋〗）
- 刑罚/法律条文用〖[〗（如〖[腰斩〗〖[族诛〗）
```

```
- 哲学概念/文体体裁用〖_〗（如〖_仁义〗〖_本纪〗）
- 各级标题（# 开头行）不做实体标注
"""

def process_chapter(client, chapter_name, input_text):
    response = client.messages.create(
        model="claude-sonnet-4-20250514",
        max_tokens=16000,
        temperature=0,        # 确保一致性
        system=SYSTEM_PROMPT,
        messages=[{"role": "user", "content": f"请处理:
{chapter_name}\n\n{input_text}"}]
    )
    return response.content[0].text
```

关键经验：

- `temperature=0` 确保标注一致性
- 按体裁定制提示词（本纪/世家/列传/书/表各有侧重）
- 批量处理必须支持断点续传（`progress.json`）
- 每批次5-10章，处理完立即验证

2.4 阶段3：质量控制

三层验证：

层级	工具	检查内容
语法检查	`lint_markdown.py`	未闭合标记、空标注、段落编号连续性、标题层级
文本完整性	`validate_tagging.py`	去除所有标记后与原文比对，确保标注仅增不改
人工抽检	人工	每批次抽检2-3章，关注实体遗漏和误标

lint_markdown.py 检查项：

```
错误级（必须修复）：
- 未闭合标记（未配对的〖〗、〖?〗、〖+〗等）
- 空标注（〖@〗、〖=〗）
- 无效段落编号（[1.a]）
- 缺少一级标题

警告级（建议修复）：
- 段落编号跳跃
- 标题层级跳跃（# → ###）
- 嵌套标注冲突（〖=〖@name〗〗）
- 超长行（>200字符）
```

validate_tagging.py 核心逻辑：

```
def strip_all_tags(text):
    """去除所有实体标记，还原纯文本"""
    # v2.8格式：〖TYPE content〗 统一格式，18类全覆盖
    text = re.sub(r'〖[@=;%&\$\^~•!#\+\?\{\:\[\_][^〖〗\n]+?〗',
lambda m: re.sub(r'^〖.', '', m.group()).rstrip('〗'), text)
    return text

def validate(tagged_md, original_txt):
    stripped = strip_all_tags(read(tagged_md))
    original = read(original_txt)
    # 逐字比对，确保标注没有改变原文
    return stripped == original
```

2.5 阶段4：实体提取与索引

两套产出数据：

同一份标注文件产出两套数据，统计口径不同：

	命名实体索引（entity_index.json）	分类词表（kg/vocabularies/）
脚本	build_entity_index.py	build_vocabularies.py
统计单位	规范实体（别名合并后）	表面形式（每个写法独立）
人名示例	沛公/汉王/高祖/高帝 → 合并为「刘邦」1条	沛公、汉王、高祖、高帝 各1条 = 4条
用途	实体目录：查同一实体的所有称谓和出处	词汇字典：查每个表面形式的用法

因此词表词条数 ≥ 索引条目数（人名差异最大，因古人别名众多）。

build_entity_index.py 流程：

```
扫描130个 *.tagged.md
    ↓ 正则提取每类实体
{ type: { surface_form: [chapter, paragraph] } }
    ↓ 加载 entity_aliases.json 做别名合并
{ type: { canonical_name: { aliases: [...], refs: [...] } } }
    ↓ 输出
entity_index.json + docs/entities/*.html
```

产出数据规模（130篇，v2.8 统一〖TYPE X〗格式后）：

实体类型	词条数	出现次数
人名	3,638	29,961
事件	3,198	—
官职	2,144	13,530

实体类型	词条数	出现次数
地名	1,865	15,177
器物	961	3,443
时间	690	9,441
制度	654	3,254
生物	388	1,031
数量	363	2,438
天文	257	1,008
神话	216	822
族群	168	1,487
氏族	153	364
邦国	147	10,470
身份	114	4,353
典籍	90	150
刑法	66	325
思想	55	1,738
礼仪	23	105
总计	**15,190**	**102,851**

2.6 阶段5：消歧与增强

消歧的核心原则：**不修改源文件**，通过独立的JSON元数据层在渲染时补全。

人名消歧

古文中同一称谓可指不同人物（"武王"可指周武王、秦武王、楚武王等）。

四层启发式策略（disambiguate_names.py）：

```
优先级1：共现全名
  同段落出现 〖@周武王〗 和 〖@武王〗 → "武王"指周武王

优先级2：附近国名（80字窗口）
  "...〖&秦〗...〖@昭王〗..." → "昭王"指秦昭王

优先级3：前置国名
  "秦昭王...王曰" → "王"指秦昭王

优先级4：章节主题国
  周本纪中的"武王" → 默认为周武王
```

输出格式（disambiguation_map.json）：

```
{
  "004": {"武王": "周武王", "成王": "周成王", "幽王": "周幽王"},
  "005": {"昭王": "秦昭王", "惠王": "秦惠王"},
  ...
}
```

保守原则：不确定的case跳过不处理，宁缺勿错。实际覆盖率~90%，138个不确定case保留原样。

别名合并

古人一生有名、字、号、谥号、庙号、封号多种称谓。

别名来源：
- 人工整理常见别名

- `auto_detect_aliases.py` 自动检测（同章共现的长名-短名对）
- 帝号统一（孝文帝 ← 孝文皇帝/孝文/文帝）

entity_aliases.json 结构：

```
{
  "person": {
    "刘邦": ["沛公", "汉王", "高祖", "高帝", "汉高祖"],
    "项羽": ["项王", "项籍", "西楚霸王"],
    "秦始皇帝": ["始皇", "始皇帝", "秦始皇"]
  },
  "place": {
    "长安": ["京师"],
    ...
  }
}
```

年份映射

```
叙事年份（"元光六年"）→ 公元年份（前129年）
```

- 数据来源：年表章节（014/015/022）提取君主在位年
- 支持288种年份格式
- 输出：`year_ce_map.json`

三、HTML渲染

3.1 正则替换顺序（关键）

`render_shiji_html.py` 的核心是**有序正则替换**，顺序错误会导致标记冲突：

```
# v2.8格式：〖TYPE content〗统一包裹，18类全部迁移完毕（2026-03-16）
ENTITY_PATTERNS = [
    (r'\*\*(.+?)\*\*', 'bold'),          # 1. 粗体必须先于 〖•器物〗
```

```
    (r'〖•(.+?)〗', 'artifact'),          # 2. 器物（黑点，v2.8）
    (r'〖;(.+?)〗', 'official'),          # 3. 官职（;替代$避免
Markdown math冲突）
    (r'〖=(.+?)〗', 'place'),             # 4. 地名
    (r'〖%(.+?)〗', 'time'),              # 5. 时间
    (r'〖&(.+?)〗', 'dynasty'),           # 6. 朝代
    (r'〖\^(.+?)〗', 'institution'),      # 7. 制度
    (r'〖~(.+?)〗', 'tribe'),             # 8. 族群
    (r'〖!(.+?)〗', 'astronomy'),         # 9. 天文
    (r'〖@(.+?)〗', 'person'),            # 10. 人名（嵌套内层时最后处
理）
    (r'〖\+(.+?)〗', 'biology'),          # 11. 生物
    (r'〖\?(.+?)〗', 'mythical'),         # 12. 神话（v2.8）
    (r'〖\{(.+?)〗', 'book'),             # 13. 典籍（v2.8）
    (r'〖:(.+?)〗', 'ritual'),            # 14. 礼仪（v2.8）
    (r'〖\[(.+?)〗', 'legal'),            # 15. 刑法（v2.8）
    (r'〖_(.+?)〗', 'concept'),           # 16. 思想（v2.8）
    (r'〖\$(.+?)〗', 'quantity'),         # 17. 数量
]
```

替换规则：

- `**粗体**` 必须在 `〖•器物〗` 之前处理
- `〖@人名〗` 放最后（最常作为嵌套内层，如嵌套拍平前的 `〖;〖@张良〗〗`）
- 每次替换后排除已生成的HTML标签内容

3.2 渲染时元数据注入

渲染器在生成HTML时加载三个JSON文件：

```
aliases = load_json("entity_aliases.json")     # 别名→规范名
disamb = load_json("disambiguation_map.json")  # 短名→全名
years  = load_json("year_ce_map.json")         # 叙事年→公元年

# 人名实体：添加链接和tooltip
# <span class="person" title="即刘邦">
#   <a href="../entities/person.html#刘邦">沛公</a>
# </span>
```

```
# 时间实体：添加公元年
# <span class="time" title="公元前202年">五年</span>
```

四、工具链一览

4.1 正式脚本

脚本	位置	功能
`build_entity_index.py`	kg/entities/scripts/	从tagged.md提取实体，构建索引
`disambiguate_names.py`	kg/entities/scripts/	四层启发式人名消歧
`auto_detect_aliases.py`	kg/entities/scripts/	自动检测别名候选
`augment_sku_entities.py`	kg/entities/scripts/	为知识单元(SKU)增补实体标注
`validate_tagging.py`	kg/entities/scripts/	验证标注未改变原文
`validate_all_chapters.py`	kg/entities/scripts/	批量验证所有章节
`find_differences.py`	kg/entities/scripts/	比较标注差异
`report_differences.py`	kg/entities/scripts/	生成差异报告
`lint_markdown.py`	scripts/	Markdown标注语法检查

脚本	位置	功能
`render_shiji_html.py`	项目根	标注Markdown→HTML渲染

4.2 数据文件

数据文件（v2.0）：

文件	位置	大小	说明
`entity_index.json`	kg/entities/data/	~4.5MB	15,190实体的全量索引（18类+事件）
`entity_aliases.json`	kg/entities/data/	22KB	586条规范名→别名映射
`disambiguation_map.json`	kg/entities/data/	9.5KB	按章节的人名消歧规则

五、技术演进

5.1 三代标注方案

版本	方式	问题
v1	HTML `<span style="color:#8B4513">` 内联样式	原文不可读，难维护
v2	Markdown Token `〖@人名〗`（〖〗格式）	可读、可控、可扩展，无嵌套歧义
	〖〗格式 Token + 外挂JSON元数据	

版本	方式	问题
v3（当前）		源文件不变，消歧/年份通过元数据层增强

5.2 关键设计演进

v1→v2 的转折： 发现HTML span标注让原文完全不可读。一个 `convert_spans_to_simple.py` 脚本完成了批量转换：

```
<span style="color:#8B4513">刘邦</span> → @刘邦@
<span style="color:#B8860B">长安</span> → =长安=
```

v2→v3 的转折（2026-03-13）：9类对称符号（ `@人名@` 等）存在嵌套解析脆弱问题，且 `$官职$` 的 `$` 字符导致Markdown math渲染冲突。迁移为 `〖TYPE content〗` 格式：

```
@刘邦@ → 〖@刘邦〗
$丞相$ → 〖;丞相〗    （官职改用 ; 避免 $ 的 math 冲突）
&汉&   → 〖&汉〗
嵌套 $@安国君@$ → 〖;安国君〗   （拍平，外层保留）
```

迁移脚本： `scripts/migrate_to_lenticular.py` （130章全量批量迁移，含备份）

v3→v4 的转折： 发现消歧和年份映射如果直接修改源文件，会导致：
- 原文失真（"武王"变成"周武王"不是原文）
- 无法回溯修改
- 多人协作时频繁冲突

解决方案：**元数据外挂** — 消歧/别名/年份作为独立JSON，渲染时注入。

5.3 v2.1→v2.2 实体清理（2026-03-13～14）

初始AI标注完成后，通过多轮「按类型反思」逐类型清理标注错误。整个过程在两天内完成，共修正约15,400处标注。

> 详细清理历史、各轮修正统计和方法论见 **SKILL_03c_实体标注反思.md §五**。

核心动因：AI初始标注存在系统性错分——朝代混合三种语义、官职混入身份/时间/人名、族群混入氏族/邦国、地名混入天文/器物/人名。

清理方法论

三阶段迁移法（复用于每轮清理）：

1. **Phase 1 自动替换** — 词表中 `always_XXX` 列表，高置信度直接替换
2. **Phase 2 上下文审查** — 歧义词生成 TSV 报告（含前后文窗口），人工/LLM 填写 decision 列后批量应用
3. **Phase 3 新词扫描** — 在未标注文本中搜索 `new_candidates` 词表，发现遗漏

关键脚本：

脚本	功能
`scripts/migrate_dynasty_to_feudal.py`	朝代→邦国/氏族迁移（Phase 1+2+3）
`scripts/migrate_official_to_identity.py`	官职→身份迁移（Phase 1+2+3）
`scripts/tag_new_entity_types.py`	新类型词表扫描（典籍/礼仪/刑法/思想）
`kg/entities/data/identity_wordlist.json`	身份词表（always/context_dependent/new_candidates 三层）

反思模式： 对每个类型，LLM逐章审查该类型的所有标注，输出修正JSON（corrections列表），脚本批量应用。每轮反思独立聚焦一个类型，避免修正冲突。

5.4 v2.5 数量类型新增（2026-03-14）

新增第18类实体 `〖$X〗`（数量），覆盖军队规模、距离、重量、金额、行政户数等非时间性计量。

新增动因

扫描发现130章中约1,200处数量表达未被标注（掩码已有标注后统计）。这些数量信息对军事力量分析、地理距离、经济数据等知识图谱维度具有价值。

符号选择：`$` 的复用

v2.0 中 `$` 从官职释放（改用 `;`）。`〖$X〗` 中的 `$` 在〖〗内单个出现，**不触发 Markdown math 渲染**（需 `$...$` 配对且不在CJK括号内）。实际在 VS Code / GitHub 中验证无冲突。

数量与时间的边界规则

这是本轮最关键的设计决策 — 数量 `〖$〗` 与时间 `〖%〗` 的划分：

归入数量 〖$〗	归入时间 〖%〗	不标注
三万人、十万众、千骑	三年、五月、元年	纯虚指（千秋万岁）
千里、百步、数百里	正月、三十日、七日	序数（第一、第二）
二千石、千户、万户	岁馀、月馀、四时	代词性数词（朕一人）
千金、万钱、百斤	万世、三代、百世	历法结构数（历书月数）
五十城、七十馀城	年二十（年龄）	时长（数十年、千馀岁）
三寸、六尺、五丈		

判断流程：
1. 以年/月/日/岁结尾 → 时间
2. 干支日（甲子/丙寅等）→ 时间
3. 年号开头（元光/太初等）→ 时间
4. 以量词结尾（人/里/城/户/骑/乘/斤/金/尺/丈/寸/匹/级等）→ 数量
5. 纯数词无量词（数百/千馀/十馀等）→ 数量（通常后接量词在标注外）
6. 其他 → 保留原标注

四步实施法

此次实施形成了「**新增类型四步法**」，可复用于未来新增实体类型：

Step 1：词表驱动自动标注

- 构建三层词表（`always_quantity` / `context_dependent` / `exclude_patterns`）
- 脚本 `tag_quantity_entities.py` 实现：掩码已有标注 → 长词优先匹配 → 数词前缀检查（避免拆分更大表达）
- 首轮产出：690处新标注

Step 2：跨类型迁移扫描

- **关键发现**：现有 `〖%时间〗` 中混入了大量数量表达（1,894处）
- 方法：扫描全部 `〖%X〗` 标注，用规则分类器（后缀/前缀/正则）自动判定时间 vs 数量
- 这一步贡献了 **73% 的总标注量** — 说明新增类型时，**存量误标往往多于增量新标**

Step 3：LLM反思清洗四类误标

误标类型	识别方法	修正量	示例
代词性数词	前缀匹配（朕/予/余 + 一人）	8处	「朕一人」→ 去标注
历法结构数	按章识别（026历书月数表）	76处	历术表「十二」→ 去标注
年龄	模式「年〖$X〗」+ 非量词后缀	46处	「年二十」→ 去标注
时长	模式「〖$X〗年/岁/月/日」	16处	「数十年」→ 去标注

Step 4：重建全链路

- 130篇HTML重新生成
- 实体索引重建（quantity.html，351条目）
- 18类词表重建
- 最终产出：**2,438处标注，351种表达，126/130章**

方法论总结

1. **存量扫描优先**：新增类型前，先扫描现有标注中的误标候选，往往比新标注更多

2. **规则分类器足够**：时间/数量的边界用后缀+前缀规则即可覆盖95%+，无需LLM逐条审查
3. **四类误标清洗模式**具有通用性：代词性用法、特殊体裁（历法表）、语义角色（年龄 vs 计量）、时间段 vs 时间点 — 每种新类型都可能遇到类似边界问题
4. **符号可以复用**：只要新语境下不产生渲染冲突，释放的符号可安全回收（`$` 从官职→数量）

关键脚本

脚本	功能
`scripts/tag_quantity_entities.py`	数量实体自动标注（词表驱动+掩码匹配）
`kg/quantity/data/quantity_wordlist.json`	数量词表（281个词条，6大类）

六、常见标注错误

> 完整错误目录和类型边界判断见 **SKILL_03c_实体标注反思.md §二**。

错误类型	错误示例	正确示例	说明
未闭合标记	`〖@黄帝 征战`	`〖@黄帝〗征战`	缺少闭合 `〗`
空标注	`〖@〗`	`〖@黄帝〗`	标记内无实体
嵌套标注	`〖=〖@name〗〗`	`〖=地名〗` 或 `〖@人名〗`	选择最外层类型
人名+官职合并	`〖;萧丞相〗`	`〖@萧何〗为〖;丞相〗`	人名和官职分开标
	`〖=秦〗攻〖=邯郸〗`	`〖'秦〗攻〖=邯郸〗`	

错误类型	错误示例	正确示例	说明
地名+国名混淆			秦是邦国不是地名
段落编号格式	`[1.a]` 或 `[1-2]`	`[1.1]` 或 `[2]`	只允许数字和点
标记吞噬句子	`〖•驷〗马〖•，迎梁王…〗`	`〖•驷马〗，迎梁王…`	标记边界断裂

七、候选扩展实体类型

以下6类在《五帝本纪》分析中识别，当前未实现，供未来扩展参考：

优先级	类型	建议标记	标注价值	说明
高	方位词	待定	与华夷观念、疆域治理相关	东/西/南/北/四方
高	颜色词	待定	与五行、礼制、等级相关	黄/彤/白/赤
高	自然现象	待定	与灾异思想、政治合法性相关	日月星辰/洪水/风雨
中	德行品质	待定	儒家史学核心评价标准	孝/仁/惠/信
低	材料物质	待定	与经济、礼制、工艺相关	金玉/水火

注：数字/数量已在v2.5实现为 `〖$数量〗`。新增类型时应参照"新增类型四步法"（§五.4）。

八、经验教训

6.1 标注阶段

1. **先粗后细** — 先用AI快速标注全部130章（允许错误），再迭代修正。等做完才做好。
2. **AI标注90%正确** — 人工修正剩余10%的价值远超AI自动修正。人工纠错最有价值。
3. **按体裁定制提示词** — 本纪/世家/列传的结构差异大，通用提示词效果差。
4. **批量处理必须支持断点续传** — 130章跑完需要数小时，中断后从头来不可接受。

6.2 消歧阶段

1. **消歧比NER更难** — 同名人物的消歧需要深度语义理解（"武王"在不同上下文指不同人）。
2. **别名是冰山一角** — 古人一生有名、字、号、谥号、庙号、封号，同一人可能有10+种称谓。
3. **保守策略优于激进** — 宁可留下未消歧的case，也不要错误消歧。错误传播比缺失更有害。
4. **章节上下文是最强信号** — "周本纪"里的"武王"几乎肯定是周武王，无需复杂算法。

6.3 工程实践

1. **内容与样式分离是根基** — Token标记系统让标注、渲染、索引三个环节完全解耦。
2. **不修改源文件** — 消歧/别名/年份通过外挂JSON实现，源标注文件是"真理之源"。
3. **渲染顺序至关重要** — `**粗体**` 和 `〖•器物〗` 的冲突，`〖@人名〗` 嵌套在 `〖;官职〗` 内拍平，都需要精心设计处理顺序。
4. **lint先行** — 每次批量标注后先跑lint，系统性错误要在源头修复，不要等到渲染/索引阶段才发现。

6.4 类型演化

> 类型演化的反思方法论详见 **SKILL_03c_实体标注反思.md §六**。

1. **新增类型时，存量误标 > 增量新标** — v2.5新增数量类型时，从 〖%时间〗 中迁移出1,894处误标，远多于词表新标注的690处。每次新增类型都应先扫描现有类型中的误分。
2. **类型边界需要明确的判断流程** — 数量与时间的区分看似简单，实际有四类边界case（代词性/历法/年龄/时长）。先用规则分类器处理95%，再用LLM反思清洗长尾。
3. **符号可以安全复用** — `$` 从官职释放后用于数量，在 〖〗 内不触发Markdown math。
4. **掩码匹配是批量标注的核心技术** — 对已标注文本做新标注时，必须先掩码所有现有标注区域，否则会在标注内部产生嵌套。

6.5 扩展到其他古籍

1. **18类实体覆盖完整** — 从11类（v1.0）演化至18类（v2.5），覆盖人/地/官/时/国/族/制/器/天/神/生/典/礼/刑/思/身/邦/量。可能需要微调（佛经需增"佛教术语"类），但Token标记体系本身可直接复用。
2. **消歧规则需要重写** — 每部书的人物称谓习惯不同，启发式规则不通用。
3. **年份体系需要重建** — 不同朝代的纪年方式不同，但映射框架（reign_periods.json）可复用。

九、可复用组件清单

直接复用，无需重写：
- Token标记体系（18类符号，v2.5）
- `lint_markdown.py`（标注语法检查）
- `build_entity_index.py`（实体索引生成器）
- `auto_detect_aliases.py`（别名检测逻辑）
- `validate_tagging.py`（标注完整性验证）
- `tag_quantity_entities.py`（词表驱动自动标注——掩码+长词优先的通用框架）

- CSS样式体系（实体高亮颜色方案）
- Purple Numbers段落编号

需要适配调整：
- `disambiguate_names.py`（消歧策略框架通用，规则需按书调整）
- SYSTEM_PROMPT（按体裁/时代定制）
- `entity_aliases.json`（每部书独立维护）

可复用的方法论模式：
- **三阶段迁移法**：always自动替换 → context_dependent审查 → new_candidates扫描（适用于任何类型新增/重分类）
- **新增类型四步法**：词表自动标注 → 跨类型迁移扫描 → LLM反思清洗 → 重建全链路
- **掩码匹配技术**：对已标注文本做增量标注时，先掩码现有标注避免嵌套干扰

本SKILL文档基于《史记》130篇×57万字的实体标注实践（2025-02 至 2026-03）提炼。
v1.1（2026-03-13）扩展为15类实体（+典籍/礼仪/刑法/思想），
v2.0（2026-03-13）迁移为〖〗格式（官职符号`$`→`;`），
v2.2（2026-03-14）实体清理重构（朝代→邦国`〖'〗`+氏族`〖&〗`，外邦归族群，官职/地名多轮反思），
v2.3（2026-03-14）新增身份类型`〖#〗`+标注完整性修复+国谥号人名重分类，
v2.5（2026-03-14）新增数量类型`〖$〗`（词表标注+跨类型迁移+反思清洗），累计修正~15,400处，
v2.8（2026-03-16）器物类型`*`→`•`（黑点，避免Markdown冲突）3,390处；五类专用括号（〖?〗/《》/〈〉/【】/〔〕）迁移为`〖TYPE〗`统一格式（神话`?`/典籍`{`/礼仪`:`/刑法`[`/思想`_`），共3,293处。
实际投入：~120小时人工 + ~30亿Token AI处理，产出11,992个实体（18类）、~102,851次标注。

SKILL 03b: 古籍人名消歧 — 从短称到唯一指称

> 从《史记》130篇3,797个人名的消歧实践中提炼，适用于古籍中同名异指的自动化解决。

一、问题定义

1.1 为什么需要消歧

古文中人物常以短称出现：谥号（"武王"）、爵号（"桓公"）、通称（"太子"）。同一短称在不同上下文可指不同人物：

短称	周本纪	秦本纪	楚世家
武王	周武王	秦武王	楚武王
昭王	周昭王	秦昭王	楚昭王
文公	—	秦文公	—
桓公	—	秦桓公	—

数据规模：《史记》130篇中有87个歧义短名（如"武王"、"桓公"、"惠王"），涉及261条（短名, 国名）→全名的映射规则。当前已消歧644条章节级映射，覆盖71个章节（其余53章无歧义短名，6章为同章多指代局限）。

1.2 核心约束

两种消歧方式（可混用）：

方式A：外部JSON消歧（章节级，不改源文件）

```
源文件(.tagged.md)  →  不改动，〖@武王〗保持原样
消歧元数据(.json)   →  {"004": {"武王": "周武王"}}
渲染HTML            →  <span title="人名：周武王">武王</span>
```

方式B：内联消歧（v2.7，适用所有13种〖TYPE〗实体类型）

```
源文件(.tagged.md)  →  〖@台|吕台〗（人名：显示"台"，全名"吕台"）
                    →  〖=函谷|函谷关〗（地名：显示"函谷"，全名"函谷关"）
                    →  〖'汉|大汉〗（邦国：显示"汉"，全名"大汉"）
无需JSON            →  内联 | 后的规范名优先级高于JSON消歧
渲染HTML            →  <span class="TYPE" title="类型：规范名" data-
canonical="规范名">显示名</span>
```

各类型典型使用场景：

类型	消歧场景	示例
人名	单字省称、同名消歧	〖@台|吕台〗、〖@桓公|齐桓公〗
地名	地名简称、多义地名	〖=函谷|函谷关〗、〖=沛|沛县〗
官职	同类官职区分	〖;守|郡守〗、〖;将|上将军〗
邦国	同名邦国（汉/楚多指）	〖'汉|汉王国〗、〖'楚|楚国〗
氏族	姓/氏规范化	〖&羋|羋姓〗、〖&嬴|嬴姓〗
时间	纪年省称	〖%元|元年〗
制度	制度全称	〖^郡县|郡县制〗
族群	族群全称	〖~胡|匈奴〗

适用场景对比：

	JSON消歧	内联消歧 〖TYPE X|Y〗
优点	不改源文件，集中管理	标注即消歧，无需维护JSON
缺点	需维护disambiguation_map.json	标注文件稍冗长
适用	章内一义的谥号（武王→周武王）	单字省称（台→吕台）、非人名消歧

二、歧义短名的类型学

2.1 三层歧义结构

```
第一层：谥号歧义（最常见，81个）
  "武王" → 周武王 / 秦武王 / 楚武王
  "桓公" → 齐桓公 / 宋桓公 / 晋桓公 / 郑桓公 / ...
  "惠王" → 秦惠王 / 魏惠王 / 燕惠王 / 周惠王

第二层：同国同谥号歧义（罕见但存在）
  齐"桓公" → 齐桓公（小白）vs 齐桓侯（午）
  晋"献公" → 晋献公（诡诸）vs 历史上其他晋献公

第三层：通称歧义（如"太子"、"王"、"公子"）
  → 不在当前消歧范围内（需段落级而非章节级处理）
```

2.2 歧义最严重的36个短名（跨章节指代不同人物）

高歧义（5+种指代）：
- **武公**（7种）：卫武公、宋武公、晋武公、秦武公、赵武公、鲁武公、武公寿
- **襄王**（5种）：周襄王、楚顷襄王、秦昭襄王、赵悼襄王、魏襄王
- **昭王**（5种）：秦昭王、燕昭王、楚昭王、周昭王、魏昭王
- **襄公**（5种）：秦襄公、宋襄公、齐襄公、晋襄公、鲁襄公

中歧义（3-4种指代）：
- **惠王**（4种）、**悼公**（4种）、**厉公**（4种）
- **武王**（3种）、**文王**（3种）、**桓公**（4种）、**景公**（3种）

2.3 不需消歧的名称（SKIP_NAMES）

以下名称虽然表面上像短称，但在史记中指代唯一，不应进入消歧流程：

类别	示例	原因
帝号	孝文帝、孝景帝、孝武帝	汉代帝号在史记中无歧义

类别	示例	原因
上古人物	黄帝、炎帝	传说人物，唯一指代
已含国名的全称	齐桓公、秦昭王、晋文公	已经消歧，不需处理
汉代封号	梁孝王、淮南王、平津侯	史记中唯一指代
通称性短名	太后、太公、周公	语义过于宽泛，章节级消歧不可靠
带前缀的复合谥号	惠文王、武灵王、庄襄王	已含国名信息，唯一指代

三、四层启发式消歧策略

3.1 策略优先级

```
优先级1：前缀国名（最精确，前10字符内有&国名&）
  "&秦&@昭王@" → 秦昭王（置信度：极高）

优先级2：附近国名（80字窗口内的&国名&或@国名X@）
  "...&楚&...诸事...@怀王@..." → 楚怀王（置信度：高）

优先级3：章节主题国（由CHAPTER_STATE表提供）
  周本纪(004)中的"武王" → 默认周武王（置信度：中高）

优先级4：共现全名（同章出现@秦昭王@和@昭王@）
  → "昭王"指秦昭王（置信度：中）
```

3.2 策略1：前缀国名

```
def find_preceding_state_prefix(content, pos):
    """检查@tag@前10字符内是否有&state&"""
    pre_context = content[max(0, pos - 10):pos]
    m = re.search(r'&([^&]+)&$', pre_context)
    if m and m.group(1) in ALL_STATES_SET:
        return m.group(1)
    return None
```

示例：`&秦&@昭王@十三年` → 前缀"秦" → 查RULER_DB → 秦昭王

局限：原文中国名前缀不总是紧邻短名（"秦之昭王"中间隔了"之"）。

3.3 策略2：附近国名

```
def find_state_tags_nearby(content, pos, window=80):
    """在pos前后window字符内查找所有&国名&和@国名X@"""
    context = content[max(0, pos-window):min(len(content),
pos+window)]
    # 查找 &国名& 标记
    for m in re.finditer(r'&([^&\n]+)&', context):
        if m.group(1) in ALL_STATES_SET:
            found.append(m.group(1))
    # 查找 @国名XX@ 标记中的国名前缀
    for m in re.finditer(r'@([^@\n]+)@', context):
        if person[0] in ALL_STATES_SET:
            found.append(person[0])
    return found
```

窗口大小：80字符（约2-3句古文）。太大则噪声增多，太小则覆盖不足。

局限：多国并提时可能选错（"秦昭王与楚怀王会于...@王@曰"中的"王"可能指任一方）。

3.4 策略3：章节主题国

每个章节有一个"主题国"（CHAPTER_STATE表），作为兜底消歧依据：

```
CHAPTER_STATE = {
    '004': '周',   # 周本纪 → 默认周
    '005': '秦',   # 秦本纪 → 默认秦
    '032': '齐',   # 齐太公世家 → 默认齐
    '039': '晋',   # 晋世家 → 默认晋
    '040': '楚',   # 楚世家 → 默认楚
    '043': '赵',   # 赵世家 → 默认赵
    ...
}
```

强信号章节（主题国几乎涵盖所有短名）：
- 本纪（001-012）：周、秦、汉，短名几乎都指该朝人物
- 世家（031-060）：各诸侯国，短名基本指该国人物
- 列传（061-130）：混合度高，主题国信号较弱

弱信号章节（主题国不可靠）：
- 书（023-030）、表（013-022）：跨国综述
- 合传列传（如069苏秦、070张仪）：涉及多国

3.5 策略4：共现全名

```
def find_cooccurring_fullnames(content, short_name):
    """在同一章节中查找包含short_name的全名@标记@"""
    pattern = r'@([^@\n]*' + re.escape(short_name) + r'[^@\n]*)@'
    # 如果只找到一个全名候选，且该全名的国名前缀在附近出现 → 采用
```

示例：章节中同时出现 `@秦昭王@` 和 `@昭王@` → "昭王"指秦昭王。

局限：如果章节中出现多个含"昭王"的全名（如 `@秦昭王@` 和 `@燕昭王@`），则无法仅凭共现消歧，需结合附近国名。

3.6 多数投票与冲突处理

同一章节同一短名可能出现多次，每次可能由不同策略消歧到不同人物。处理方法：

```
# 收集每个(章节, 短名)的所有消歧结果
results = [("秦昭王", "nearby_state"), ("秦昭王",
"chapter_state"), ...]

# 多数投票：≥2/3一致 → 采用
if top_count * 3 >= total * 2:
    chapter_map[chapter_id][short_name] = top_name
else:
    # 票数分裂 → 跳过，不消歧（保守原则）
    pass
```

四、已知错误模式与手动修正

模式1：附近国名误导

问题：nearby_state窗口内捕获到的国名不是短名所指的国。

```
章004（周本纪）：
  "...&燕&师伐...@惠王@..." → nearby_state误判为燕惠王
  实际应为：周惠王（周本纪中的惠王）
```

修正：MANUAL_CORRECTIONS 表覆盖。

模式2：多国并提段落

问题：段落同时讨论多个国家的事务，窗口内多国名共存。

```
章071（樗里子甘茂列传）：
  叙述秦楚交涉，&楚&频繁出现
  "@武王@" 被误判为楚武王，实为秦武王
```

修正：MANUAL_CORRECTIONS + 提升前缀国名策略的优先级。

模式3：主题国与短名不匹配

问题：章节主题国与短名的国籍不一致时，兜底策略失效。

```
章005（秦本纪）：
  "@惠王@" → chapter_state判为秦惠王
  但此处引述燕国事务，实为燕惠王
```

修正：当策略2（附近国名）与策略3（主题国）冲突时，应优先采用策略2。

模式4：同国多位同谥号君主

问题：齐国有齐桓公（小白，前685）和齐桓侯（午，前374），谥号相近。

当前处理：SKIP_NAMES中包含"齐桓侯"，避免将"桓公"错误映射；大多数章节中"桓公"默认指齐桓公（小白）。

MANUAL_CORRECTIONS 表

对已知启发式错误的精确修正（优先级最高）：

```
MANUAL_CORRECTIONS = {
    ('004', '元王'): '周元王',     # nearby_state:楚 误判
    ('004', '惠王'): '周惠王',     # nearby_state:燕 误判
    ('044', '惠王'): '魏惠王',     # nearby_state:秦 误判
    ('044', '襄王'): '魏襄王',     # nearby_state:秦 误判
    ('071', '武王'): '秦武王',     # nearby_state:楚 误判
    ('071', '昭王'): '秦昭王',     # nearby_state:楚 误判
```

```
    ('079', '昭王'): '秦昭王',     # nearby_state:周 误判
    ('087', '惠王'): '秦惠王',     # nearby_state:楚 误判
}
```

模式5：上古章节无消歧需求

五帝本纪(001)、夏本纪(002)等上古章节中，人名均为唯一指代（黄帝、颛顼、尧、舜等），不存在谥号歧义问题。消歧工作可跳过这些章节。

模式6：单字人名假阳性

"象""益""会""代""舍""段"等单字既是人名也是常见动词/副词/形容词。在古文中频繁产生假阳性标注。

```
已知假阳性：
- 001 [5]  @象@天 — "象"=效法，非舜弟象
- 001 [20] 皆@益@笃 — "益"=更加，非伯益
```

检测方法：上下文模式匹配（`以.{0,2}@象@天`、`皆@益@笃`、`日@益@`、`@益@盛/强/弱`）。

模式7：破损标记污染（已大部分修复）

原有655处破损@标记@，已通过 `fix_broken_tags.py` 五层策略自动修复331处，降至155处（76%↓）。

五层修复策略：
1. L1: `$@X@Y@$` → `$@XY@$`（封号+名字合并，167处）
2. L2: `@Title@KnownName@` → `@TitleKnownName@`（已知实体合并，52处）
3. L3: `$X@@Y@$` → `$X$@Y@$`（$/@混用修正，54处）
4. L4: 奇数@行三连@修复（27处）
5. L5: `$$Name@` / `弒其君Name@` 等常见误用（31处）

剩余155处：主要在世家章节（039/036/040/032等），偶数@行配对错位，需语义理解。

脚本：`kg/entities/scripts/fix_broken_tags.py`（--scan/--fix/--chapter/--summary）

模式8：引用性短名的默认消歧

列传和书/表中引述历史典故时使用的"武王""文王"等短名，绝大多数指周武王/周文王。可为非本纪/世家章节的这些短名设置默认映射：

短名	默认指代	适用场景
武王	周武王	非秦/楚本纪章节的引述
文王	周文王	非秦/楚本纪章节的引述
成王	周成王	非楚世家章节的引述

五、数据结构

5.1 输入

文件	位置	说明
`*.tagged.md`	chapter_md/	130个标注文件，包含 `@人名@` 标记
RULER_DB	脚本内嵌	261条 (短名, 国名) → 全名 映射规则
CHAPTER_STATE	脚本内嵌	130章 → 主题国 映射
SKIP_NAMES	脚本内嵌	不参与消歧的名称集合
MANUAL_CORRECTIONS	脚本内嵌	手动修正表

5.2 输出

disambiguation_map.json（章节级消歧映射）：

```
{
  "004": {
    "武王": "周武王",
    "成王": "周成王",
    "幽王": "周幽王",
    "文王": "周文王",
    ...
  },
  "005": {
    "昭王": "秦昭王",
    "惠王": "燕惠王",
    ...
  }
}
```

语义：在章节 `NNN` 中，所有 `@短名@` 的出现都映射到同一个全名。

局限：章节级消歧假设同一章节中同一短名只指代一个人。当同一章节中"昭王"在不同段落指不同人时（极罕见），需要升级到段落级消歧。

5.3 渲染集成

`render_shiji_html.py` 中的 `_add_entity_links()` 函数加载消歧映射：

```
# 消歧流程（渲染时）
resolved = entity_text                    # "武王"
if css_class == 'person' and chapter_id:
    chapter_disambig = disambig_map.get(chapter_id, {})
    if entity_text in chapter_disambig:
        resolved = chapter_disambig[entity_text]  # "周武王"

# 别名解析 → 规范名
canonical = alias_map.get(resolved, resolved)  # 可能进一步合并

# 生成HTML：显示原文，tooltip显示全名
```

```
# <a href="person.html#entity-周武王" target="_blank">
#   <span class="person" title="人名: 周武王">武王</span>
# </a>
```

六、Canonical命名规范

消歧后的全名通过别名系统（entity_aliases.json）进一步归一化到canonical name。

6.1 命名规则

类别	Canonical模式	示例	别名示例
先秦诸侯	国+谥号	周武王, 秦昭王, 齐桓公	武王, 昭王, 桓公
秦始皇	秦始皇帝	—	始皇, 嬴政, 秦王政
汉帝	汉+通称	汉高祖, 汉文帝, 汉武帝	刘邦/沛公/高祖, 刘彻/武帝
诸侯王	封国+谥号	楚元王, 梁孝王	—
其他人物	最常用名	项羽, 商鞅, 韩信	项王/项籍, 卫鞅/商君

6.2 设计原则

1. **唯一性** — canonical name在整个知识库中无歧义
2. **源文忠实** — 优先使用史记自身的称呼习惯（汉帝用谥号而非个人名）
3. **个人名降为alias** — 刘邦/嬴政等作为别名存在，方便搜索但不作canonical
4. **旧canonical自动合并** — 如"厉王胡"→"周厉王"后，"厉王胡"加入别名列表

6.3 处理流程

```
源文本 @武王@
  → 消歧(chapter_level) → 周武王
  → 别名解析(global) → 周武王 (canonical，无进一步合并)

源文本 @沛公@
  → 消歧(无需) → 沛公
  → 别名解析 → 汉高祖 (canonical)

源文本 @厉王胡@
  → 消歧(无需) → 厉王胡
  → 别名解析 → 周厉王 (canonical)
```

七、覆盖率与改进方向

7.1 当前覆盖率

指标	数值
RULER_DB规则数	261条（87个歧义短名）
已标注章节	130/130
已消歧章节	71/130（有歧义短名的章节全部覆盖）
无需消歧章节	53/130（无歧义短名出现）
章节级映射数	644条
缺失消歧映射	6条（6章，均为同章多指代）
幽灵条目（无效映射）	0条

指标	数值
别名组	586组
Canonical命名规范	汉帝=汉+通称、先秦=国+谥号
人名实体数	3,797条

7.2 剩余覆盖缺口

经全量反思修正后，剩余6条无法章节级消歧（同章多指代）：

章节	短名	原因
005	平公	同章含晋平公(L226)和齐平公(L259)
033	景公	同章含晋景公、齐景公、鲁景公
039	桓公	同章含曲沃桓叔和齐桓公
050	哀王	汉代封国王号，非先秦谥号体系
098	肃侯/顷侯	汉代侯爵谥号
111	哀侯/烈侯	汉代侯爵谥号（卫青后人）

7.3 改进路径

路径A：扩展RULER_DB
- 补充缺失的 (短名, 国名) → 全名 映射
- 特别是侯爵体系（韩/赵/魏的侯爵称号）

路径B：段落级消歧
- 当前是章节级（一章一个映射），升级为段落级（每个段落独立消歧）
- 解决同一章节中短名指代切换的问题（如列传中叙述多国事务）

路径C：AI辅助消歧
- 对启发式无法覆盖的case，用LLM阅读上下文判断

- 输入：段落文本 + 候选全名列表 → 输出：最可能的全名
- 适合处理策略2-4都无法覆盖的复杂语境

路径D：别名系统联动
- 将 entity_aliases.json（别名→规范名）与消歧映射打通
- 例如"沛公"→"刘邦"是别名关系，不需要消歧
- "孝公"在秦本纪中→消歧→"秦孝公"→别名→"秦孝公"（规范名一致）

八、关键经验

8.1 消歧原则

1. **保守优于激进** — 不确定的case跳过不处理，错误消歧比缺失更有害
2. **不改源文件** — 消歧结果存储在独立JSON，渲染时注入，源文件是"真理之源"
3. **章节上下文是最强信号** — "周本纪"里的"武王"几乎肯定是周武王
4. **前缀国名最可靠** — `&秦&@昭王@` 的消歧置信度远高于窗口国名

8.2 工程实践

1. **RULER_DB要全** — 每个 (短名, 国名) 对都需要有映射，缺失导致消歧失败
2. **SKIP_NAMES要准** — 误加导致漏消歧，误删导致错误消歧
3. **MANUAL_CORRECTIONS是安全网** — 启发式永远有错，手动修正表是最后防线
4. **多数投票防震荡** — 同一章节同一短名的多次出现，取2/3多数决

8.3 古籍特殊性

1. **谥号是主要歧义源** — 先秦人物以谥号称呼是惯例，谥号字数少且重复率高
2. **国名是消歧关键** — 古文中国名几乎总是与人物短称同段出现
3. **体裁决定歧义密度** — 本纪/世家（单国叙事）歧义少，书/表/合传（跨国）歧义多
4. **同国同谥号极罕见** — 261条规则中几乎无重复，说明谥号制度本身有一定的去重机制

九、工具链

脚本	位置	功能
`disambiguate_names.py`	kg/entities/scripts/	四层启发式消歧，生成disambiguation_map.json
`review_disambiguation.py`	kg/entities/scripts/	消歧+别名反思审查工具（质检脚本）
`render_shiji_html.py`	项目根	渲染时加载消歧映射，生成tooltip和链接
`build_entity_index.py`	kg/entities/scripts/	构建实体索引（消歧后的规范名）

数据文件	位置	说明
`disambiguation_map.json`	kg/entities/data/	章节级消歧映射（71章644条）
`entity_aliases.json`	kg/entities/data/	别名→规范名（586组）
`entity_index.json`	kg/entities/data/	全量实体索引（11,069实体）

十、君主实体的专项消歧

10.1 君主数据库（rulers.json）

位置：`kg/relations/rulers.json`
规模：651条，覆盖上古圣王至汉代藩王
核心字段：`所属国`、`谥号`、`名`、`别名`、`类型`、`登基年`、`退位年`

利用此数据库，可对87个歧义短名进行系统性查找：

```
# 按(所属国, 谥号)精确匹配
def lookup_ruler(state, short_title, rulers_db):
    for r in rulers_db:
        if r['所属国'] == state and r['谥号'] == short_title:
            return r['谥号']  # 返回 canonical 全称（如"楚武王"）
    return None
```

10.2 在位年辅助消歧

当同一短名有多个候选（如"惠王"可能是秦惠王/魏惠王/燕惠王）时，可用上下文年份缩小范围：

```
def disambig_by_year(short_name, context_year, rulers_db):
    """context_year 为公元前年份（负整数）"""
    candidates = []
    for r in rulers_db:
        if short_name in (r.get('谥号',''), r.get('名',''),
*r.get('别名',[])):
            start = r.get('登基年') or -9999
            end   = r.get('退位年') or 9999
            if start <= context_year <= end:
                candidates.append(r)
    return candidates  # 通常缩减到1-2个候选
```

示例：

上下文年份	短名	在位区间匹配	消歧结果
-334	惠王	秦惠王(-337~-311)、魏惠王(-369~-319)、燕惠王	需结合国名进一步缩小
-279	昭王	秦昭王(-306~-251)、燕昭王(-311~-279)	两者均在位，仍需国名
-238	庄王	楚庄王(-613~-591) 不在位 → 楚幽王(-237~-228)	年份排除法

10.3 谥号 vs 名 vs 别名的标注选择

rulers.json 中每位君主有多个字段，对应标注中的不同表现形式：

字段	典型值	史记中的出现形式
谥号	楚武王	主要称谓，多见于本纪/世家
名	熊通	偶见于列传或初次介绍
别名	武王	章节内短称（需消歧）

标注建议：按原文形式标注，消歧映射在 `disambiguation_map.json` 中维护，不修改源文件。

10.4 四级君主类型与消歧难度

类型	歧义程度	原因
天子/皇帝	低	天子称谓少且唯一（"孝武帝"无歧义）
诸侯王	高	

类型	歧义程度	原因
		多国并用"武王"/"昭王"等同谥（87个歧义短名主要在此类）
诸侯侯君	高	"桓公"/"献公"跨国重复率高，5+国共用同一谥号
汉诸侯王	低	封国+谥号在史记中唯一指代（"梁孝王"无歧义）

10.5 消歧规则与rulers.json联动

RULER_DB（261条）中的 `（短名，国名）→ 全名` 映射规则，可从 rulers.json 自动生成初稿：

```
def generate_ruler_db(rulers_json):
    """从rulers.json生成RULER_DB初稿"""
    ruler_db = {}
    for r in rulers_json:
        state = r['所属国']
        title = r.get('谥号', '')
        if title and state:
            short = title.replace(state, '')  # 去掉国名前缀得到短名
            if short and len(short) >= 2:
                ruler_db[(short, state)] = title
    return ruler_db
```

注：自动生成的规则需人工审核，特别是吴国（寿梦等名字不含"王"字）和越国（允常/句践）的特殊情况。

本节基于651条rulers.json数据（2026-03整理完成），覆盖上古至汉代全部君主。
相关数据文件：`kg/relations/rulers.json`、`kg/relations/rulers.csv`
整理过程详见：`doc/entities/史记君主列表整理过程.md`

当前覆盖率：130/130章节已标注+已分析，71章有消歧映射(644条)，53章无需消歧。
剩余6条已知局限（同章多指代）。

Canonical命名规范：汉帝=汉+通称(汉高祖/汉文帝)、先秦=国+谥号(周武王/秦昭王)、其他=最常用名。
2026-03-09：全量反思+修正、补标035-040六章(3780处标记)、+25 canonical调整、+204消歧映射。
2026-03-13（v2.0迁移）：9类对称符号迁移为〖〗格式（`@人名@` → `〖@人名〗`，`&朝代&` → `〖&朝代〗` 等），本文档算法示例暂保留旧格式说明，实际标注文件已更新为新格式。
反思详情：doc/entities/实体消歧别名反思报告.md

> 另见：SKILL_04e_年份消歧.md — 在位纪年→公元年的消歧方法论

SKILL 03c: 按章反思 — 实体标注逐章审查

> **使用方式**：复制 §一 提示词，替换 `NNN` / `章名`，提供给 Agent 执行。
> **适用场景**：单章语境误判、遗漏实体、章节特有错误。跨章系统性错误见 SKILL_03e_按类型反思.md。
> **关键约束**：每个Agent只处理一章，不批量处理多章。
>
> **第二轮反思**：2026-03-17启动。背景：第一轮中013章后修正量骤降，可能反思深度不够。
> 第二轮要求：像001-012章那样深度思考，逐段审查，质量基线30-80处修正/章。
> 报告写入：`doc/entities/第二轮按章实体反思报告.md`

一、按章反思提示词（即用）

```
你是《史记》实体标注质量审查员。请对章节 NNN_章名 的已标注文件执行深度按章反思。

## 任务说明

读取文件：chapter_md/NNN_章名.tagged.md

审查对象是**已标注文件**——关注"标错了什么"和"漏标了什么"。
不修改原文字符，只修改标注符号。

**质量基线**：正常章节（300-1000行）预期30-80处修正。若低于20处，需再次逐段复查。

---

## Step 0：预备（必须先执行）

1. 读取 `skills/SKILL_03c_按章反思.md` §三（已积累规律），了解所有已知错
```

````
误模式
2. 运行格式检查:
   ```bash
   python scripts/lint_markdown.py chapter_md/NNN_*.tagged.md
   ```
3. 运行边界损坏检测:
   ```bash
   grep -n '〖@[^〗]*[，。、；：]' chapter_md/NNN_*.tagged.md
   ```
4. 运行单字名推断（若脚本存在）:
   ```bash
   python scripts/infer_single_char_names.py chapter_md/
NNN_*.tagged.md
   ```

---

## Step 1: 格式与边界扫描

### 1a. 旧格式残留
- `〖X〗` → `〖+X〗`（旧生物格式）
- `@X@`、`=X=` → 对应新格式
- `〖=、〗`、`〖=。〗` → 去掉
- `<span style="...">` → 对应新格式

### 1b. 标注边界损坏（AI生成高频错误）
| 错误模式 | 修复 |
|---------|------|
| `〖@X弟〗Y` | → `〖@X〗弟〖@Y〗` |
| `〖@，次曰〗X` | → `，次曰〖@X〗` |
| `〖@，是为〗X` | → `，是为〖@X〗` |
| `〖@弒〗`/`〖@杀〗`/`〖@攻〗` | → 去标注（动词） |
| `〖@X。〗Y` | → `〖@X〗。〖@Y〗` |
| `〖@X，〗Y` | → `〖@X〗，Y` |
| `〖@曰："〗` | → `曰："` |
| 标注吞噬后续文字 `〖•X〗，...后续〗` | 找到错位`〗`并移正 |
| 数字前缀漏标 `三百六〖$十六日〗` | `〖$三百六十六日〗` |
````

```
| 同一词重复标注 | 删除重复 |

---

## Step 2：现有标注审查（逐段阅读，每段都要看）

⚠️ **逐段阅读全文**，不要跳过任何段落。对每个标注判断类型和边界。

### 2a. 人名 `〖@〗` 误标（最高频错误类型）

| 误标 | 正确 | 说明 |
|------|------|------|
| 单字邦国名（赵/韩/魏/秦/齐/楚/晋/吴/越/燕/宋/卫/陈/郑/鲁等） | `〖'邦国〗` | "伐赵"/"与韩攻"→邦国；"太史赵"→人名省称 |
| `〖@X〗氏`/`〖@X〗宗`/`〖@X〗孤` | `〖&氏族〗` | 赵氏/韩氏/刘氏→氏族名 |
| 帝号/谥号/庙号 | `〖@人名〗` | 高帝/孝武/幽王→人名，不是`〖;官职〗` |
| 通称（天子/皇帝/太后/陛下） | `〖#身份〗` | 泛指不特指某人 |
| 特指后妃（康后/窦太后等） | `〖@人名〗` | 能替换为具体人→人名 |
| 侯号作职衔引用 | `〖;官职〗` | 文中已有本名时，侯号→官职 |
| 动词误入人名标注 | 去标注 | 起/杀/攻/弒/说/封/立/会/及/为 |
| `〖@帝颛顼高阳〗` | 拆分 | `〖@帝颛顼〗〖&高阳〗` |
| 长子/少子/兄/弟 | 不标注 | 亲属关系描述 |
| "以X为Y"中Y | `〖;官职〗` | 任命语境中的职位名 |

### 2b. 官职 `〖;〗` 误标

| 误标 | 正确 |
|------|------|
| 天子/皇帝/太后/皇后/陛下/先帝/先王 | `〖#身份〗` |
| 诸侯（泛称）/寡人/今王/人主/单于 | `〖#身份〗` |
| 太牢（祭祀规格） | `〖^制度〗` |
| 文学（"文学之士"的学问类别） | `〖^制度〗` |
| 太史公/褚先生（撰述者） | `〖@人名〗` |
| `〖;官名〗〖;人名〗`中第二个是人名 | 第二个改`〖@〗` |

### 2c. 时间 `〖%〗` 误标
```

误标	正确
时长（岁馀/月馀/数月/百岁/数年/二日/万岁）	不标注
距离/人数（X里/X人/X众）	`〖$数量〗`
年龄（年二十/年九十馀）	不标注或`〖$〗`
居官时长（居丞相十年）	不标注或`〖$〗`
✅ 纪年格式（元年/三年/八月丙子）	保留`〖%〗`

2d. 制度 `〖^〗` 误标

误标	正确
刑名（五刑/弃市/城旦/坑杀/车裂/菹醢）	`〖[刑法〗`
思想概念（仁义/王道/五德/礼义/道德/天命）	`〖_思想〗`
五行之德（土德/水德/火德）	`〖_思想〗`
日月（天体语境）	`〖!天文〗`

2e. 地名 `〖=〗` 误标

误标	正确
邦国名（秦/汉/齐/楚等政治实体）	`〖'邦国〗`
方位副词（东北/西南/东方/西方/南方）	不标注
泛指地貌（山川/海/野）	不标注
九鼎等器物	`〖•器物〗`
同师（同一师父）	不标注

2f. 语境判断（同一词汇不同含义）

词汇	语境A（标注）	语境B（不标或改类型）
日月	`〖!〗`天体（日月所照）	`〖%〗`时间 / `〖^〗`礼制
成都	`〖=〗`地名（四川成都）	不标（成为都邑）
封禅	`〖^〗`典章制度	`〖:〗`具体礼仪行为
共工	`〖@〗`人名（独立提及）	`〖;〗`官职（任命语境）
符节	`〖•〗`器物（持节/矫节）	—

```
| 说 | 不标（动词，往说X） | 人名一部分（夏说/傅说） |

---

## Step 3：遗漏检查（按类型逐项扫描）

⚠️ 遗漏往往是修正数量最大的来源，务必仔细。

### 3a. 人名遗漏
| 句型/场景 | 说明 |
|---------|------|
| 段落内已标注人名的后续未标出现 | 同段后文的省称/再次出现 |
| 修饰语位（X大祖也/X之子） | "X"漏标 |
| 嵌入主语位（於是X妻之/乃X使之） | "X"漏标 |
| "以X为Y"中X | X 通常是人名 |

### 3b. 官职遗漏
| 句型 | 说明 |
|------|------|
| `以X为Y`（任命） | Y 是官职 |
| `拜X为Y` / `封X为Y` | Y 是官职或爵号 |
| 具体官名独立出现 | 丞相/太守/将军/御史/太尉等 |

### 3c. 地名/邦国遗漏
| 句型 | 说明 |
|------|------|
| `至于X` / `居于X` / `迁于X` | X 是地名 |
| `伐X` / `攻X` / `入X` | X 是邦国/地名 |
| 具体地名独立出现 | 咸阳/长安/邯郸等 |

### 3d. 其他类型遗漏
| 类型 | 高频遗漏 |
|------|---------|
| 生物`〖+〗` | 蝗/马/牛/鱼/鸟/犬/虎/象/龙（非神话语境）/草木 |
| 刑法`〖[〗` | 弃市/诛/坑/车裂/族诛/大辟/腰斩/大赦/赦天下 |
| 器物`〖•〗` | 金/玉/印/剑/车/旗/鼎/弓/节/玺/冠/衣 |
| 典籍`〖{〗` | "《X》曰"/"作《X》"/"传《X》" |
```

```
| 天文`〖!〗` | 星宿/历法名词/日食/彗星 |
| 氏族`〖&〗` | "姓X"/"氏X"中的X |

---

## Step 4: 单字人名消歧

若 Step 0 的脚本产出有结果，逐一验证:
1. **时代交叉**: 段落年代是否在人物活动期内
2. **汉姓约定**: 皇室/同姓诸侯→刘X
3. **远古单名**: 舜/尧/禹/鲧/桀/纣/汤等不消歧
4. **歧义无法消解**: 同章多位同末字人物→保留`〖@X〗`

---

## 输出格式

### A. 格式与边界修复
| 段落 | 原文 | 修正 | 说明 |
|------|------|------|------|

### B. 类型误标修正
| 段落 | 原标注 | 修正后 | 规则 |
|------|--------|--------|------|

### C. 遗漏补充
| 段落 | 原文片段 | 建议标注 | 说明 |
|------|---------|---------|------|

### D. 单字名消歧
| 段落 | 单字 | 全名 | 验证 |
|------|------|------|------|

### E. 章节特有发现
记录本章发现的、尚未写入 SKILL 的新规律或异常模式。

### F. 修正统计汇总
```

```
| 修正类别 | 数量 |
|---------|------|
| 边界/格式修复 | N处 |
| 类型误标修正 | N处 |
| 遗漏补充 | N处 |
| 单字名消歧 | N处 |
| **合计** | **N处** |

---

## 完成后操作

> ⚠️ **核心约束：只改标注符号，不改原文字符。**
> 反思修正只允许增删 `〖〗` 标注符号（v2.8格式），**原文汉字一个都不能增加、删除或修改**。
>
> **禁止操作示例**：
> - ❌ 删字：`皆贤妇人` 中若 `〖;妇人〗` 标注有误，**不能**删掉"妇人"二字
> - ✅ 正确：去掉标注符号还原为 `皆贤妇人`
> - ❌ 增字：`族` → `〖[族诛〗`（增加了"诛"字）
> - ✅ 正确：`族` → `〖[族〗` 或 `〖[族|族诛〗`（消歧说明加到|后）
> - ❌ 增字：`斩` → `〖[斩首〗`（增加了"首"字）
> - ✅ 正确：`斩` → `〖[斩〗` 或原文就是"斩首"时才能标`〖[斩首〗`
>
> **消歧标注规则**：
> - 如需说明词义，使用扩展标注：`〖[原文|说明〗`
> - 标注内容必须与原文字符完全一致，说明信息加到 `|` 后面

1. **应用修正**：将 A/B/C/D 中所有修正应用到 `chapter_md/NNN_章名.tagged.md`
2. **完整性验证（必须执行）**：
   ```bash
   python scripts/lint_text_integrity.py chapter_md/NNN_*.tagged.md
   ```
   结果须为 `✓ 0处实质差异`。若有差异，立即排查修复。
```

3. **写反思报告**：修正统计和特有发现追加到 `doc/entities/第二轮按章实体反思报告.md`
4. **写回 SKILL**：E 中的新规律写入 SKILL_03c §三对应小节

二、反思工具

工具	用途
`scripts/lint_text_integrity.py`	**原文完整性验证**（实质差异=0 才通过）
`scripts/lint_markdown.py`	标注格式检查（未闭合/空标注）
`scripts/infer_single_char_names.py`	单字人名全名推断
`grep -n '〖@[^〗]*[，。]' chapter_md/NNN_*.tagged.md`	边界损坏检测

三、已积累的错误模式（按类别）

A. 类型判断规则

A1. 帝号/谥号/庙号→人名（009-010章归纳）

帝号/谥号/庙号指代具体某人→ `〖@人名〗`，不是身份 `〖#〗` 或官职 `〖;〗`。

类型	标注	示例
帝号/谥号/庙号	`〖@〗`	高帝、高后、孝武皇帝、幽王、厉王
通称	`〖#〗`	天子、皇帝、太后、陛下、上（指当朝皇帝）

类型	标注	示例
泛称	〖#〗	今王、寡人、人主、单于

判断标准：能替换为具体人名→〖@〗；泛指一类身份→〖#〗。

A2. 特指后妃→人名（012章首发现）

特指某具体后妃的称号→〖@〗，泛称→〖#〗。
- 康后（胶东康王后）→〖@〗；太后/皇后（泛称）→〖#〗

A3. 先帝/先王→身份（006章首发现）

"先帝""先王"是追述已故帝王的称号→〖#身份〗，不是官职〖;〗。

A4. 侯号作职衔→官职（011章首发现）

文中已有本名时，侯号引用→〖;官职〗；全章只出现侯号→可保留〖@〗。

A5. 职官+人名拆分（012章归纳）

〖;官名〗〖;人名〗中第二个是人名→改〖@〗。

A6. 太牢→制度（012章首发现）

太牢是祭祀规格→〖^〗，不是官职〖;〗。

A7. 日月三义（001章首发现）

- 天体（日月所照）→〖!〗；绘于器物上→〖!〗
- 时间纪年→〖%〗
- 礼制规范→〖^〗（极少）

A8. 时长vs纪年（008章归纳）

不标〖%〗：岁馀/月馀/数月/百岁/万岁/围城二日/苦战数岁/居丞相十年
标〖%〗：元年/三年/八月丙子等纪年格式

A9. 五行之德→思想（001/005/010章）

土德/水德/火德/金德/木德→〖_思想〗。

A10. 思想vs制度（014-030章）

仁义/王道/五德/礼义/道德/天命→ `〖_思想〗` ；儒家/墨家/封禅/祭祀→ `〖^制度〗` 。

A11. 刑法词汇（004/006/010章）

弃市/城旦/五刑/肉刑/坑杀/屠城/菹醢/车裂/大辟/大赦→ `〖[刑法〗` 。

补充（003章第二轮）：动词形式的刑法词汇易遗漏：
- 残酷刑罚：醢（剁成肉酱）、脯（做成肉干）、剖（剖心）、车裂、菹醢
- 刑罚执行：殛（诛杀）、诛、族诛、弃市、僇
- 赦免类：赦、大赦、赦天下

检查方法： `grep -E '殛|醢|脯|剖|车裂|族诛|僇|赦' chapter_md/NNN_*.tagged.md`

A12. 符节→器物（009章首发现）

持节/矫节中的"节"→ `〖•器物〗` 。

A13. 灾害生物→生物（012章首发现）

蝗灾之"蝗"→ `〖+生物〗` 。

A14. 庙号消歧（013章首发现）

殷商庙号加消歧前缀：中宗→ `〖#中宗|帝太戊〗` 。

A15. 撰述者→人名（013章首发现）

太史公/褚先生在"X曰"语境→ `〖@人名〗` 。

A16. 稷字语境判断（001第二轮）

"稷"需根据语境判断：
- 任命语境（"以X为稷"/"使X主稷"）→ `〖;官职〗`
- 与人名并列（"让於稷、契与皋陶"）→ `〖@人名〗` （后稷省称）

A17. 器物类高频遗漏（001第二轮）

易遗漏器物词：衣/车/琴/笠/仓廪/瑞/服/鼎/弓/节/玺/冠

补充（003章第二轮）：
- 捕猎工具：网（"网开三面"）
- 财物类：宝玉、珠玉、金银（作为战利品/财物时）

- 服饰类：宝玉衣（具体服饰名称）
- 标识类：白旗、黄旗、旗帜
- 兵器类：斧钺、弓矢、剑戟

补充（011章第二轮）：
- 布料等级：七布（七升布）、十五布、粗布、细布等布料规格标注为器物 〖•〗

区别：
- "网"在捕猎语境→器物；"天罗地网"抽象比喻→不标注
- "宝玉"作为具体物品/战利品→器物；抽象品德描写→不标注

检查方法：

```
grep -nE '衣|车|琴|笠|廪|瑞|服|鼎|弓|节|玺|冠|网|宝玉|旗|斧钺'
chapter_md/NNN_*.tagged.md
```

A18. 数量标注完整边界（001第二轮）

错误：二十有〖$二人〗 → 数字前缀漏标
正确：〖$二十有二人〗 → 完整数量短语

其他示例：
- 〖$三百六十六日〗（不是三百六〖$十六日〗）
- 〖$五千里〗（不是五〖$千里〗）

A19. 交通工具全覆盖（002第二轮）

易遗漏交通工具：船/舟/橇/暐/四载

说明：
- 单个交通工具：车/船/舟/橇/暐 → 〖•器物〗
- 集合名词：四载（指车船橇暐四种工具）→ 〖•四载〗

检查方法：

```
grep -nE '船|舟|橇|暐|四载' chapter_md/NNN_*.tagged.md
```

注意平行段落：如果某段描述"陆行乘车，水行乘船…"，往往会在其他段落重复出现，需对照检查。

A20. 测量工具识别（002第二轮）

准绳、规矩等测量工具应标注为 `〖•器物〗` 。

示例：
- "左准绳，右规矩" → `左〖•准绳〗，右〖•规矩〗`

A21. 集体性身份/官职（002第二轮）

易遗漏的集体性称谓：

词汇	标注	说明
群后	`〖#身份〗`	众诸侯
百官	`〖;官职〗`	官员群体
百工	`〖;官职〗`	百官，工匠群体
子弟	`〖#身份〗`	年轻人群体
父老	`〖#身份〗`	年长者群体
诸将	`〖#身份〗`	将领群体

A22. 典籍vs器物辨析（002第二轮）

文献名称应标注为 `〖{〗` 而非 `〖•〗` 。

常见错误：
- ❌ `〖•五子作歌〗`
- ✅ `〖{五子之歌〗`

A19. "得意"判断规则（006第二轮）

"得意"需根据语境判断：

人名语境：`〖@张得意〗`（明确指某个叫"得意"的人）

动词短语语境：不标注（表示满意、称心、实现意图）
- "明得意"：彰显志得意满

- "章得意"：显示功绩得偿所愿
- "不得意於海内"：不能称心如意於天下

类似易误标词汇：大业、大毕、昭明、天下大旱等也易误标为人名，需仔细辨别语境。

A20. "今上"→身份标注（006第二轮）

"今上"指当今皇帝，是尊称→〖#今上〗，不是人名。

类似词汇（皆为身份〖#〗）：
- 今上、陛下、圣上（指当今皇帝）
- 先帝、先王（指已故帝王）

判断标准：指代当时在位或已故帝王的通称→身份类。

A21. 诗歌铭文实体标注（006第二轮）

诗歌格式中的实体词汇也需完整标注，不因格式特殊而降低标注标准。

器物类示例：
- 戎兵、甲兵、章旗、备器等武器/旗帜
- 金石、珍宝、器械等物品

思想类示例：
- 仁义、礼法、王道等思想概念

注意：碑刻铭文是重要历史材料，标注价值高，需完整审查。

B. 邦国/氏族判断规则

A23. 战争场景器物密集检查（004第二轮）

战争场景（战前誓师、交战过程、战后处置）器物密集，需逐句审查。

关注动词：执、持、把、握、击、射、驰、下、佩、悬、挥、载、乘、骑、驱等

高频器物：
- 兵器：剑、车、弓、矢、旗、钺、戈、矛、盾、甲
- 战具：戎车、兵车、战车、罕旗、常车
- 防具：甲、盾、胄

检查方法：

```
grep -nE '(执|持|把|握|击|射|驰|下|佩|悬|挥).*[剑车弓矢旗钺戈矛盾甲]'
chapter_md/NNN_*.tagged.md
```

典型段落特征:
- 牧野之战、巨鹿之战等大型战役描写
- 单段可能出现5-10处器物遗漏
- 动词+器物的组合密集出现

A24. 旗名判断规则（004第二轮）

旗帜名称常见格式："颜色+旗"/"形状+旗"/"用途+旗"

器物标注:
- 大白旗、小白旗、罕旗、常旗 → `〖•器物〗`
- 黄旗、赤旗、黑旗、青旗 → `〖•器物〗`

避免误标为天文:
- 大白本是旗名，不是太白金星
- 区分方法：带"旗"字明确指旗帜→器物；单独"太白"且谈天象→天文

检查方法:

```
grep -nE '(大白|小白|罕旗|常旗|黄旗|赤旗)' chapter_md/NNN_*.tagged.md
```

A25. 器物省称识别（004第二轮）

在简洁语境（赞文、诗句、列举）中，器物常用省称。

常见省称:
- 乐 → 乐器（"太师抱乐"）
- 车 → 戎车/兵车/常车（"载以车"）
- 兵 → 兵器/甲兵
- 旗 → 某某旗（"县之旗"）

判断方法:
1. 查看同章其他段落是否有完整表达
2. 根据语境推断（祭祀→乐器，战争→兵器）
3. 简洁文体（赞、颂、铭）中省称频率更高

检查方法：在赞文段落特别关注单字器物词

A26. 礼器类词汇表（004第二轮）

礼仪场景（祭祀、封建、朝会）常见礼器，容易与制度混淆。

礼器分类:

类别	器物	说明
青铜器	鼎、彝、爵、觚、尊、卣、簋	祭祀用青铜礼器
玉器	玉、璧、圭、璋、琮、琥	礼仪用玉器
祭品容器	木主、神主、牌位	祭祀用牌位
祭祀用品	明水、牲、粟、酒、血	祭祀供品
仪仗	旗、车、剑、节	礼仪场合的器物

易混淆点:
- 彝 vs 礼制：祭祀用的"宗彝"→器物；"典章彝器"→制度
- 明水 vs 概念：祭祀用的清水→器物；抽象"明德"→不标注
- 木主 vs 神祇：具体牌位→器物；神灵本身→神祇

检查方法:

```
grep -nE '鼎|彝|璧|圭|木主|明水|宗彝' chapter_md/NNN_*.tagged.md
```

B1. 单字邦国名误标人名（跨章反思）

语境	正确标注	示例
军事/外交主体	〖'〗	伐赵、与韩攻、魏败我
领土/地理	〖'〗	赵北有代、入晋
氏族/家族	〖&〗	赵氏、刘氏

语境	正确标注	示例
人物省称	〖@〗	太史赵（赵某）

B2. 动词误标人名（跨章反思）

起/杀/攻/弒/说/封/立/会/及/为 等高频动词须看语境。
"起"：起身/建造/调动/发生→不标；白起/吴起省称→〖@〗。

B3. 人名重复出现时省称遗漏（001第二轮）

现象：同一人名在段落中首次出现标注，后续再现遗漏。

示例（001章）：
- "於是尧妻之二女…尧善之" → 两处"尧"都应标注
- "尧二女不敢以贵骄…尧九男皆益笃" → 两处"尧"都应标注
- "象喜…本谋者象…象取之…象乃止" → 四处"象"都应标注

检查方法:
1. 使用grep查找未标注的高频人名：`grep -n '[^〗]尧[^〗]' chapter_md/NNN_*.tagged.md`
2. 对于远古单名（尧/舜/禹/象等），特别关注段落中的重复出现

C. 单字人名消歧规则

C1. 处理流程

1. 脚本推断 → 2. 人工审查（时代/姓氏/歧义） → 3. 应用

C2. 汉代姓氏约定

皇室/同姓诸侯→刘X（友→刘友）。异姓王不适用。

C3. 不处理的情况

- 远古单名（舜/尧/禹/鲧/启/桀/纣/汤/契/稷/益等）
- 前文全名紧随的省称
- 歧义无法消解

C4. 统计

章	替换数	典型映射
003	9	微→微子, 说→傅说, 辛→帝辛
004	37	武→周武王, 文→周文王, 犯→马犯
005	62	鞅→卫鞅, 圉→公子圉, 胡→胡陽
006	75	高→赵高, 斯→李斯, 毒→嫪毒
007	103	籍→项籍, 梁→项梁, 良→张良
008	93	布→黥布, 信→韩信, 豨→陈豨
009	48	产→吕产, 友→刘友, 禄→吕禄
010	31	勃→周勃, 婴→灌婴, 长→刘长
011	33	彻→刘彻, 戊→刘戊, 系→萧系
012	0	无有效映射
合计	**491**	

附：001章反思结果摘要（参考案例）

类别	修正数	主要内容
旧格式 〔〕 →〖+〗	15处	动物标注旧括号
天子〖;〗→〖#〗	9处	全章"天子"一律是身份
制度→刑法	7处	典刑/五刑/流宥五刑/官刑/教刑/赎刑/五服

类别	修正数	主要内容
年龄去标注	6处	年二十/三十/五十/五十八/六十一
制度→天文	3处	日月（"历日月""日月所照"）
制度→思想	2处	土德/天地（五行哲学概念）
地名→其他/去标注	3处	白马→生物，山川/东方→不标
人名→其他/去标注	3处	公孙→氏族，昭明→不标，夔夔→不标
人名拆分	2处	帝颛顼高阳/帝喾高辛→人名+氏族号
官职误标人名	1处	"以垂为共工"中共工→官职
数量边界修复	1处	三百六十六日
时间→数量	3处	二人/五千里/七世
遗漏补充	1处	七十载
合计	**~57处**	

上游/下游

- **上游**：SKILL_03a_实体标注.md（18类定义）、SKILL_03b_实体消歧.md
- **下游**：修正后重新生成 `kg/entities/data/entity_index.json`
- **跨章系统性错误**：见 SKILL_03e_按类型反思.md

A23. 群体称谓的身份/官职判断（003章第二轮）

泛指群体且无具体指向→身份 `〖#〗` 或官职 `〖;〗`：

词汇	标注	说明
群后	〖#〗	诸侯集合称谓
群臣	〖#〗	臣子群体
百吏	〖#〗	官吏群体
百工	〖#〗 或 〖;〗	工匠群体，语境判断
百官	〖;〗	官员群体

A27. 年表体裁刑法词汇密集检查（014章第二轮）

年表章节记录重大历史事件，刑法词汇密度极高，需系统性扫描。

高频词汇:
- 杀/弒/诛/族/醢/脯/剖/车裂（处决类）
- 灭/屠/坑（灭国类）
- 获/虏/执/囚（俘虏类）
- 鸩（毒杀）

固定搭配:
- 杀+人名：`杀〖@某某〗` → `〖[杀〗〖@某某〗`
- 灭+国名：`灭梁` → `〖[灭〗梁`
- 诛+人名：`诛〖@某某〗` → `〖[诛〗〖@某某〗`
- 弒君：`弒公` → `〖[弒〗公`

扫描方法:

```
grep -nE '(杀|弒|诛|灭|获|虏|鸩)' chapter_md/NNN_*.tagged.md | grep
-v '〖\['
```

014章统计:
- 杀：28处遗漏
- 灭：21处遗漏
- 诛：13处遗漏
- 弒：2处遗漏

A28. "释之"消歧规则（014章第二轮）

"释之"有两种含义，需根据语境判断：

情况A - 人名：张释之（汉朝廷尉，见其他章节）
- 标注：〖@张释之〗
- 判断：前文明确提及此人

情况B - 动词短语：释放之义
- 标注：不标注
- 判断：释放/赦免语境，如"肉袒谢，释之"

014章实例：
- L297: "〖@郑伯〗肉袒谢，释之" → 去标注（释放之义）
- L389: 类似语境 → 去标注

判断流程：
1. 前文是否提及"张释之"等具体人物 → 人名
2. 语境是"释放/赦免" → 动词短语，不标注
3. 无法确定 → 保守不标注

A29. "不礼"动词短语误标（014章第二轮）

错误：王〖@不礼〗、蔡〖@不礼〗

正确：去标注

说明：
- "不礼"：不以礼待之（动词短语）
- "王不礼" = 周王不以礼待之
- "蔡不礼" = 蔡国不以礼待之

类似易误标词汇：得意、大业、昭明、不礼等

A30. 年表"公"的标注豁免（014章第二轮）

现象：年表中"公"（指诸侯）大量出现，理论上属身份类〖#〗，但密度极高。

决策：在年表/表类章节中，"公"作为通称可不标注，避免过度标注。

适用条件：
- 文体：年表/表类章节
- 密度：单行出现3次以上"公"字
- 语境：明确指诸侯国君的通称

说明：014章每行可能包含12国多个事件，"公"字密集出现（几乎每行5-10处），全部标注会导致标注噪音过大。

A31. 氏族标注在年表权力语境（014章第二轮）

年表常记录"X氏作乱"/"X氏专政"等权力斗争，此时X氏应标注为氏族类。

修正示例：
- `〖@田氏〗有德于齐` → `〖&田氏〗`（氏族势力）
- `齐政归〖@田氏〗` → `〖&田氏〗`（氏族夺权）
- `〖@季氏〗` → `〖&季氏〗`（鲁国权臣家族）
- `〖@赵氏〗赐〖^公族〗` → `〖&赵氏〗赐〖^公族〗`

判断标准：X氏在权力、政治、家族势力语境中 → 氏族类 `〖&〗`

014章统计：赵氏、崔氏、栾氏、田氏（3处）、季氏，共7处误标

A43. 哲学列传思想词汇密集检查（062章第二轮）

哲学思想类列传（老子韩非列传等）中，思想类词汇密度极高，需系统性扫描：

道家核心词汇：
- 道、德、虚、无为、清静、自然、虚无、微妙

法家核心词汇：
- 法、术、势、法制、刑名、刑名法术

伦理概念：
- 仁、义、孝、恩、圣、贤

标注原则：
- 即使在抽象讨论语境也必须标注
- 与制度类区分：思想概念→〖_〗，学派/典章→〖^〗
- 预期哲学列传思想类标注占比>10%

检查方法：

```
grep -nE '(道|德|仁|义|孝|法|术|势|虚|无为|自然)' chapter_md/
NNN_*.tagged.md
```

A44. 著述语境典籍名称识别（062章第二轮）

讨论著作的章节（如老子韩非列传）中，典籍名称识别规则：

固定搭配:
- "著书X篇" / "作X" → X是典籍
- "X之书" / "X曰" → X是典籍
- "为X书" → X是典籍

易混淆情况:
- "著书二篇" → 数量"二"标注为〖$二〗，不是典籍
- "五蠹之书" → "五蠹"是篇名，标注为〖{五蠹〗

062章案例:
- 孤愤、五蠹、内外储、说林、说难（韩非著作）
- 渔父、盗跖、胠箧、畏累虚、亢桑子（庄子篇名）

A45. "子将"等复合结构的语法识别（062章第二轮）

错误: `〖@子将〗隐矣`（误作人名）
正确: 子将隐矣（"子"=您，"将"=将要）

识别规则:
- "子+动词"结构 → "子"是尊称代词，不是人名
- 常见模式：子欲、子所、子言、子独、子将

对比真实人名:
- 孟子、韩非子、老莱子 → "子"是敬称后缀，前有明确人名
- 判断依据：前文是否已引入此人物

类似易误标: 出于、得意、不礼等动词短语

A46. 人名谱系传承中的单字名遗漏（062章第二轮）

谱系传承语境（"X子Y，Y子Z"）中，单字人名极易遗漏:

062章案例:

> 老子之子名宗，宗子注，注子宫，宫玄孙假，假之子解

原标注: 只标注"老子"
应标注: 宗、注、宫、假、解 全部需标注

检查方法:

```
grep -nE '(之子|子名|玄孙|曾孙)' chapter_md/NNN_*.tagged.md
```

对匹配行逐个检查人名是否完整标注。

A47. 刑法动词在法家文献的密度（062章第二轮）

法家思想讨论（如韩非列传）中，刑法词汇密度远超普通章节：
- 军事类：伐、攻、袭
- 惩罚类：戮、诛、杀、刖、族
- 赦免类：赦、释

062章统计：
- 伐：4处
- 戮：2处
- 刖：2处
- 杀：2处
- 其他：攻、袭、诛、赦各1处

检查策略：法家/军事相关列传需对刑法动词进行系统性扫描。

A32. 军功传记刑法动词密集检查（057章第二轮）

军功传记（世家/列传中的军事人物传）刑法动词密度极高，需系统性扫描。

高频动词清单：
- 攻城类：攻、袭、围、下
- 作战类：击、破、败、战、追
- 平定类：定、取、降、平
- 处决类：杀、斩、诛、屠、灭
- 叛乱类：反、叛

典型句式：
- "攻X，破之" → "〖[攻〗X，〖[破〗之"
- "击X军，斩X" → "〖[击〗X军，〖[斩〗X"
- "下X城，定X郡" → "〖[下〗X城，〖[定〗X郡"
- "别破军二，下城三，定郡五" → "别〖[破〗军二，〖[下〗城三，〖[定〗郡五"

扫描方法：

```
grep -nE '(攻|击|破|下|定|围|杀|斩|诛|屠|灭|降|反|取|追|袭|败)'
chapter_md/NNN_*.tagged.md | grep -v '〖\['
```

057章统计：93处刑法动词遗漏，占修正总数56%，为该章核心问题。

适用章节：世家/列传中的将帅传记，如绛侯周勃世家、淮阴侯列传、卫将军骠骑列传等。

A33. 主人公省称全覆盖原则（057章第二轮）

世家/列传主人公的姓名省称应全覆盖，不论首次或后续出现。

常见遗漏位置：
- 动词前主语：`勃为人` → `〖@勃〗为人`
- 介词后宾语：`说勃曰` → `说〖@勃〗曰`
- 被动句施事：`告勃欲反` → `告〖@勃〗欲反`
- 叙事句主语：`勃既定燕` → `〖@勃〗既定燕`
- 动词宾语：`拜勃为太尉` → `拜〖@勃〗为太尉`

检查方法：

```
grep -n '[^〗]主人公省称[^〗〖]' chapter_md/NNN_*.tagged.md
```

057章统计：22处"勃"字省称遗漏。

注意事项：
- 远古单名（尧/舜/禹等）同样适用全覆盖原则（参考001章第二轮）
- 同章多人物时，主人公省称优先级最高

A34. 官职+人名拆分标注（057章第二轮）

职官与人名连用时需拆分标注。

标注格式：`〖;官职〗〖@人名〗`

修正示例：

错误	正确	说明
`〖;南阳守齮〗`	`〖;南阳守〗〖@齮〗`	职官+人名拆分
`〖@内史保〗`	`〖;内史〗〖@保〗`	内史是官职
`〖@守陘〗`	`〖;守〗〖@陘〗`	守是官职，陘是人名
`〖;丞相程纵〗`	`〖;丞相〗〖@程纵〗`	正确示例

判断方法：
1. "X+Y"结构中，X为职官名词（将军/丞相/太守/都尉等）→ X标注为〖;〗

2. Y为人的姓名 → Y标注为〖@〗
3. 遇到单字人名易判断错误，需根据语境确认

057章统计：官职+人名拆分不足，涉及齮、保、陉等单字名。

SKILL 03d: 渲染与发布 — 标注文本输出为HTML

> 将标注完成的 `chapter_md/*.tagged.md` 输出为可发布的HTML。本SKILL仅描述从实体构建阶段到输出的衔接；渲染器、配色、交互等详细设计见 **SKILL 09a 认知辅助阅读器**。

一、工序位置

```
标注文本 (chapter_md/*.tagged.md)
    ↓ 渲染 (SKILL 09a)
HTML页面 (docs/chapters/*.html)
    ↓ 发布
GitHub Pages
```

与主管线并行，标注完成后即可渲染。

二、快速命令

```
# 渲染单章
python render_shiji_html.py chapter_md/001_五帝本纪.tagged.md

# 批量生成全部130章
python generate_all_chapters.py

# 完整发布流水线（生成 + 路径修复 + lint检查）
bash publish_to_docs.sh
```

三、质量检查

```
# HTML格式检查
python scripts/lint_html.py docs/chapters/

# Markdown标注格式检查
python scripts/lint_markdown.py chapter_md/001_五帝本纪.tagged.md
```

四、详细设计

渲染器实现、18类实体配色、段落结构语义化、交互功能、实体索引页、发布流水线等详见：

→ **SKILL 09a 认知辅助阅读器**

SKILL 03e: 按类型反思 — 批量系统性错误修正

> **适用场景**：同一类错误在130章中大量重复（系统性误标），一次修正全书。
> **单章语境误判**：见 SKILL_03c_按章反思.md。

一、按类型反思提示词（即用）

```
你是《史记》实体标注质量审查员。执行一轮按类型反思，目标：

- **源类型**：SOURCE_TYPE（如 `〖^制度〗`）
- **目标类型**：TARGET_TYPE（如 `〖[刑法〗`）
- **迁移描述**：DESCRIPTION（如"刑法词汇从制度类型迁出"）

工作目录：/path/to/shiji-kb
章节文件：chapter_md/NNN_*.tagged.md（130章）

---

## Step 1：侦察（统计现有分布）

```bash
grep -roh '〖SOURCE_MARKER[^〗]*〗' chapter_md/ | sort | uniq -c |
sort -rn
```
```

（将 SOURCE_MARKER 替换为实际类型标记，如 `^`、`;`、`=` 等）

列出：
- 全部不同词汇及出现次数
- 总出现章节数
- 总出现次数

Step 2：词表分类

将 Step 1 的词汇分为三类：

always_list（无歧义，直接替换）

满足：**在任何上下文中都属于 TARGET_TYPE**，不存在歧义。

格式：

```
词汇 | 出现次数 | 理由
```

context_list（需上下文判断）

满足：**同一词汇在不同语境可能属于不同类型**，需逐条审核。

格式：

```
词汇 | 出现次数 | 歧义说明
```

exclude_list（保留在源类型）

满足：**确实属于源类型**，不应迁移。

格式：

```
词汇 | 理由
```

Step 3：Phase 1 执行（always_list 自动替换）

对 always_list 中的每个词，在所有章节文件中执行替换：

```
# 示例：〖^五刑〗 → 〖[五刑〗
python scripts/migrate_TYPE_fixes.py \
  --source SOURCE_TYPE \
  --target TARGET_TYPE \
  --words always_list.txt \
  --apply
```

替换完成后验证：

```
python scripts/lint_markdown.py chapter_md/NNN_*.tagged.md
```

lint 必须通过，无新增错误。

报告：
- 实际替换了多少处
- 哪些章节被修改

Step 4：Phase 2 准备（context_list 上下文报告）

为 context_list 生成 TSV 审查报告（含前后20字上下文窗口）：

```
python scripts/migrate_TYPE_fixes.py \
  --source SOURCE_TYPE \
  --words context_list.txt \
  --report context_review.tsv
```

TSV 格式：

```
章节 | 段落 | 词汇 | 前文（20字） | 后文（20字） | decision（keep/
change）
```

decision 列留空，由人工或下一轮 LLM 审查填写。

Step 5：Phase 3 新词扫描（遗漏发现）

在**未被任何标注包围的裸文本**中搜索 TARGET_TYPE 的典型词汇：

```
python scripts/tag_new_entity_types.py \
  --candidates new_candidates.txt \
  --output new_candidates_found.tsv
```

`new_candidates.txt`：TARGET_TYPE 的典型词汇列表（根据类型定义和本次 Step 1 所见词汇推断）。

报告：发现了哪些新词，在哪些章节，建议标注为 TARGET_TYPE 的理由。

输出

A. 词表分类结果

类别	词汇数	词汇列表
always_list	N个	词1/词2/...
context_list	N个	词1/词2/...
exclude_list	N个	词1/词2/...

B. Phase 1 执行报告

章节	替换处数
...	N处
合计	**N处**

C. Phase 2 TSV（context_list 审查表）

（输出为文件 `logs/context_review_TARGET.tsv`）

D. Phase 3 新候选词

词汇	出现次数	建议标注	代表语境

E. 本轮发现的新规律

记录执行过程中发现的、超出预期的误标模式，写入 SKILL_03e §二或 §六。

完成后操作

1. **应用 Phase 1**：already done in Step 3
2. **lint 验证**：确认无格式错误
3. **交付 Phase 2 TSV**：等待人工审核 context_list
4. **更新 §六 已知存量错误**：将本轮完成的迁移标记为 √，更新数量
5. **更新词表 JSON**：将 always_list 并入对应词表文件

```
---

## 二、三阶段迁移法（方法论参考）

每轮按类型反思固定三步，提示词中 Step 3-5 即对应这三阶段：

### Phase 1：自动替换（高置信度）

构建 `always_list`（该类型中无歧义、可直接替换的词）：
```

```bash
python scripts/migrate_TYPE_fixes.py --always   # 130章批量
```

验证：`python scripts/lint_markdown.py` 确认无格式破坏。

Phase 2：上下文审查（歧义词）

构建 `context_list`（需要看上下文才能判断的词）：

```
python scripts/migrate_TYPE_fixes.py --context  # 生成 TSV 报告
```

TSV 报告含前后文窗口，人工或 LLM 填写 `decision` 列（keep/change），批量应用。

Phase 3：新词扫描（遗漏发现）

在未标注文本中搜索 `new_candidates` 词表，发现应标但未标的实体：

```
python scripts/tag_new_entity_types.py --candidates
new_candidates.txt
```

三、常见误标方向速查

用于确定每轮迁移的目标。

源类型	目标类型	典型词汇	已处理?
〖;官职〗	〖#身份〗	天子/太后/皇后/食客/处士	√ v2.3
〖;官职〗	〖@人名〗	高祖/沛公/元年（作人名）	√ v2.3
〖%时间〗	〖$数量〗	X里/X人/X众/百里-三百里	√ v2.5

源类型	目标类型	典型词汇	已处理?
〖%时间〗	不标注	年龄（年X十）/时长（数十年）	√ v2.5
〖&氏族〗	〖'邦国〗	汉/周/秦/齐等统一王朝/封国	√ v2.1
〖&氏族〗	〖~族群〗	匈奴/月氏/大宛等外邦政权	√ v2.1
〖^制度〗	〖[刑法〗	五刑/官刑/教刑/赎刑/典刑/五服（刑制）	待执行
〖^制度〗	〖:礼仪〗	祭祀/朝觐（具体行为语境）	待执行
〖^制度〗	〖!天文〗	日月（"所照/所至/历X迎送"语境）	待执行
〖X〗	〖+X〗	所有旧括号生物标注	待执行
〖=地名〗	〖'邦国〗	秦/汉/齐等（指政治实体时）	部分

四、执行优先级

高影响优先:

1. **格式修复**（〖〗旧格式）— 自动化，无损，约未统计处
2. **制度→刑法**（五刑/官刑等）— 中频，估计全书200+处
3. **制度→礼仪**（祭祀/朝觐）— 中频
4. **制度→天文**（日月天体语境）— 低频，约50处
5. **地名→邦国**（残余误分） — 低频

五、组合策略

阶段	推荐方式
初始标注完成后	**按类型**：优先修正高频系统性错误（旧格式/天子/年龄）
新增实体类型后	**按类型**：扫描存量误标 + 新词扫描
重点章节打磨	**按章**：本纪/世家主要章节精细审查
发布前质检	**按章**：抽检5-10章评估整体质量

原则：新增类型前必须先扫描存量误标——v2.5 新增数量类型时，存量迁移1,894处，远多于新标注690处。

六、脚本参考

脚本	功能
`scripts/migrate_dynasty_to_feudal.py`	朝代→邦国/氏族迁移
`scripts/migrate_official_to_identity.py`	官职→身份迁移
`scripts/tag_new_entity_types.py`	新类型词表扫描（典籍/礼仪/刑法/思想）
`scripts/apply_reflect_fixes.py`	批量应用修正JSON
`kg/entities/data/identity_wordlist.json`	身份词表（always/context_dependent/new_candidates）

七、已知存量错误（待执行）

系统性错误	估计规模	修正脚本
〖〗旧生物格式	未统计	`scripts/migrate_old_biology.py`（待写）
制度→刑法（五刑等）	~200处	三阶段法，目标 `〖[刑法〗`
制度→礼仪（祭祀/朝觐）	未统计	三阶段法，目标 `〖:礼仪〗`

已完成的存量清洗：
- 天子→身份：2,235处（v2.3）
- 年龄不标注：146处（v2.5）
- 时间中的距离→数量：1,894处（v2.5）
- 朝代→邦国/氏族：~6,100处（v2.1）

附：v2.1→v2.2 历史清理记录

14轮按类型反思，共修正约15,400处。

轮次目标	修正量	主要操作
邦国/氏族重构	~6,100处	〖&〗拆分为〖'〗+〖&〗
外邦归位（匈奴等）	326处	邦国→族群
氏族误标清理	85处	地名39/人名24/时间20/思想2
族群/氏族交叉	195处	姜氏→氏族，秦/齐→邦国，单于→官职
官职反思（元年/高祖）	2,290处	元年1,497→时间，793→人名

轮次目标	修正量	主要操作
地名反思	60处	共和→时间，太阴→天文，九鼎→器物
官职→身份（天子等）	2,235处	33种always + 21种新词扫描
破损标注修复	358处	边界错误（吞噬句子）
裸星号转换	129处	`*X*` → `〖•X〗`（v2.8器物格式）
国+爵合并	273处	分裂标注合并
封国谥号→人名	527处	齐桓公/晋文公等165种
封国→人名	120处	赵王/秦王等24种
数量类型新增	2,584处	新标690 + 时间误标迁移1,894
数量误标清洗	146处	代词/年龄/时长去标

上游/下游

- **上游**：SKILL_03a_实体标注.md（18类定义和词表）
- **下游**：修正后重新生成 `kg/entities/data/entity_index.json`，更新词表JSON
- **单章精修**：见 SKILL_03c_按章反思.md

SKILL 04: 事件构建 — 从标注文本到结构化事件

> 从标注文本中提取结构化事件，标注公元纪年，通过Agent反思审查年代准确性。

一、工序位置

```
标注文本 (chapter_md/*.tagged.md)
    |
    ▼ 04a 事件识别
事件索引 (kg/events/data/NNN_*_事件索引.md)
    |
    ▼ 04c 事件年代推断
公元纪年标注
    |
    ▼ 04d 事件年代审查
年代修正 (Agent反思)
    |
    ▼ 04e 年份消歧
年号/纪年消歧
```

十表（013-022）有专用补充流程（04b），与主流程并行。

二、方法论

为什么用LLM提取事件?

古籍事件不同于新闻事件:
- 一个"事件"可能跨越数段叙述（"鸿门宴"跨十余段）

- 事件粒度需要语义判断（"秦灭六国"是一个事件还是六个？）
- 因果关系深藏在叙事逻辑中，非字面可提取

自动 vs LLM 分工

任务	方法	原因
事件提取	LLM	需理解叙事语义和粒度判断
公元纪年标注	规则	年号表+在位年可计算
纪年审查	LLM (Agent)	需综合判断年代合理性

三、执行流程

```
# 1. 提取事件
python kg/events/scripts/extract_events.py

# 2. 构建年份映射
python kg/events/scripts/build_year_map.py

# 3. 标注公元纪年
python kg/events/scripts/annotate_ce_years.py

# 4. 纪年质检（迭代修复）
python kg/events/scripts/lint_ce_years.py

# 5. Agent反思审查（按章循环）
python kg/events/scripts/run_review_pipeline.py --prompt NNN

# 6. 格式验证
python kg/events/scripts/validate_events.py
```

四、产出统计

指标	数值
总事件数	3,185
覆盖章节	130/130 (100%)
事件类型	11种
有CE标注	3,051 (98.7%)
战争事件	736
时间范围	前2700年 ~ 前87年
年代修正	~2,119次（五轮反思）

五、子工序索引

编号	文件	内容
04a	SKILL_04a_事件识别.md	事件定义/Schema/11种类型/体裁策略
04b	SKILL_04b_十表事件处理.md	十表（013-022）专用补充提取
04c	SKILL_04c_事件年代推断.md	分层纪年解析/君主在位数据库
04d	SKILL_04d_事件年代审查.md	Agent反思审查工作流
04e	SKILL_04e_年份消歧.md	同一年号在不同语境中的消歧

SKILL 04a: 古籍历史事件识别

1. 任务定义

从已完成实体标注的古籍文本中，识别和提取结构化的历史事件记录。

输入: 已标注的Markdown文件（含@人名@、=地名=等实体标注）
输出: 结构化的事件索引文件（Markdown表格 + 详细记录 + 事件链）

---

2. 事件的定义与粒度

什么是一个"事件"

一个事件是一个**有完整起因-经过-结果（或至少经过-结果）的行为单元**。它通常包含：
- 一个或多个行为主体（人物）
- 一个行为或状态变化
- 可选的时间、地点、因果

粒度控制（最关键的判断）

太粗	适中	太细
"黄帝一生"	"涿鹿之战"	"黄帝征师"
"夏朝兴衰"	"启伐有扈氏"	"启召六卿"
"商鞅变法"	"商鞅第一次变法"	"商鞅废井田"

判断标准:
1. 能否用4-8字概括？能 → 粒度合适
2. 能否写出2-4句事件描述？不能（太简单） → 可能太细
3. 事件描述超过8句？ → 可能太粗，应拆分

特殊情况处理

世系传承: 连续的"X崩，子Y立"（无额外叙述）→ 合并为一个"世系传承"事件

```
### 002-036 夏朝中期世系
- **事件类型**: 家族事件
- **事件描述**: 中康崩，子帝相立...帝扃崩，子帝廑立。
```

重复叙述: 同一事件在不同段落被提及（如001章和002章都提到"鲧治水"）→ 各章独立提取，不去重。去重是后处理的工作。

太史公曰: 评论本身不是事件，但如果评论中**首次提到**某个史实，应当提取。

---

3. 事件类型体系

共11个主要类型:

类型	说明	典型例子
战争	战役、军事征伐、平叛	涿鹿之战、长平之战
继位	即位、崩逝、禅让、篡位、废立	舜践天子位、桀亡国
政治活动	朝会、外交、联盟、谏言、决策	众臣荐举虞舜
政治改革	变法、制度建设、任命	舜任命禹为司空
政治整顿	惩处、流放、诛杀、清洗	舜惩办四罪
家族事件	婚姻、生子、世系、家族纷争	黄帝娶嫘祖生子
建设	治水、筑城、迁都、工程	禹治水十三年
文化活动	礼乐、祭祀、著述、教育	尧命羲和制历
经济活动	商贸、赋税、盐铁、货殖	平准均输

类型	说明	典型例子
自然灾害	洪水、地震、旱灾、天象异变	洪水滔天
改革	法律变革、官制改革	禹定五服制度

类型归属原则：一个事件只标注一个主类型。当多个类型可选时：
- "战争"优先级最高（凡涉及军事行动的归入战争）
- "继位"次之（凡涉及帝位变更的归入继位）
- 其余按事件核心行为判断

---

4. 不同章节类型的处理策略

4.1 本纪（001-012）

帝王年表式叙事，事件密度高。

- "X年，Y事" → 每条有独立意义的记录都应提取
- 注意区分同一帝王不同时期的事件
- 秦始皇本纪、高祖本纪事件量最大（可达100+）

4.2 表（013-022）

结构化年代表，选择性提取。

- 只提取有叙事性的重大事件
- 忽略纯年月记录（如"某年某月某日"无实质内容的）
- 关注跨国并列的重大事件

4.3 书（023-030）

典章制度论述，侧重制度变革节点。

- 天官书：天象异变事件
- 封禅书：历次封禅事件

- 河渠书：水利工程事件
- 平准书：经济政策变革事件

4.4 世家（031-060）

诸侯国通史，时间跨度长。

- 建国、迁都、灭国是关键事件
- 世系传承可合并
- 重大战役需详细记录
- 与其他世家/本纪重复的事件，各章独立提取

4.5 列传（061-130）

人物传记，关注生平转折点。

- 合传中不同人物分别提取
- 关键节点：出仕、建功、获罪、结局
- 类传（循吏、酷吏等）每人单独处理
- 对话/谏言中的事件也应提取

---

5. 输出格式规范

5.1 事件索引文件结构

```
# {章节名} 事件索引

## 事件列表概览

| 事件ID | 事件名称 | 事件类型 | 时间 | 地点 | 主要人物 | 朝代 |
|--------|---------|---------|------|------|---------|------|
| 003-001 | 殷商始祖契 | 家族事件 | - | =商= | @契@、@简狄@、@帝喾@ | &有娀氏& |
| ... | ... | ... | ... | ... | ... | ... |
```

```
---

## 详细事件记录

### 003-001 殷商始祖契
- **事件类型**: 家族事件
- **时间**: -
- **地点**: =商=
- **主要人物**: @契@、@简狄@、@帝喾@
- **涉及朝代**: &有娀氏&
- **事件描述**: 殷契的母亲简狄是有娀氏之女、帝喾次妃。简狄吞玄鸟卵而生契。契长大后辅佐禹治水有功，帝舜封其于商，赐姓子氏。
- **原文引用**: "@殷契@，母曰@简狄@，&有娀氏&之女，为@帝喾@次妃。三人行浴，见〖+玄鸟〗堕其卵，@简狄@取吞之，因孕生@契@。"
- **段落位置**: [1]

---

## 统计信息

- **总事件数**: N
- **战争事件**: X
- ...

---

## 关键事件链

1. **商朝建立** (003-001 → 003-002 → ...)
   - 契受封 → 先公世系 → 汤建商
```

5.2 事件ID编码规则

- 格式：`{3位章节号}-{3位序号}`
- 章节号与文件名一致（001-130）

- 序号从001开始，按事件在原文中出现的顺序编号

5.3 字段填写规范

字段	有值时	无值时
时间	保留原文标注 `%时间%`	`-`
地点	保留原文标注 `=地名=`	`-`
主要人物	`@人名@`，顿号分隔	不应为空
朝代	`&朝代&`	`-`

---

6. 工作流程

6.1 大模型事件识别（主体）

```
chapter_md/*.tagged.md  →  Claude API  →  kg/events/*_事件索引.md
```

脚本：`scripts/extract_events.py`

- 读取已标注的tagged.md文件作为输入
- 根据章节类型（本纪/表/书/世家/列传）微调提示词
- 大模型输出结构化事件索引
- 支持断点续传（progress.json）

6.2 后处理验证

脚本：`scripts/validate_events.py`

- 检查事件ID格式和连续性
- 检查必须字段是否完整

- 生成统计汇总

6.3 人工审校

大模型输出需要人工审校的常见问题：
1. **粒度不一致**：有的章节拆得太细，有的太粗
2. **类型归属**：边界情况的类型判断
3. **遗漏事件**：特别是对话中隐含的事件
4. **重复事件**：同一事件被拆成多个

---

7. 质量标准

7.1 事件数量参考范围

章节类型	典型事件数	说明
本纪（短）	30-50	五帝、夏
本纪（长）	80-150	秦始皇、高祖
表	10-30	选择性提取
书	15-40	侧重制度节点
世家（短）	20-40	小国、短篇
世家（长）	60-100	晋、楚、齐
列传（短）	10-25	单人传
列传（长）	30-60	合传、民族传

7.2 完整性检查清单

- [ ] 每个事件有唯一ID
- [ ] 每个事件有事件名称（4-8字）

- [ ] 每个事件有类型分类
- [ ] 每个事件有事件描述（2-4句）
- [ ] 每个事件有原文引用（保留标注）
- [ ] 每个事件有段落位置
- [ ] 概览表格与详细记录一一对应
- [ ] 有统计信息部分
- [ ] 有关键事件链部分

---

8. 已知难点与解决方案

8.1 长章节超出输出限制

问题：秦始皇本纪、高祖本纪等章节很长，事件量大，可能超出大模型单次输出限制。

方案：
- 方案A：将长章节按section拆分，分批处理后合并
- 方案B：增大max_tokens（当前16000），必要时用extended thinking

8.2 年表章节的处理

问题：十二诸侯年表等以表格形式存在，不是传统叙事。

方案：只提取有叙事性的重大事件（如"灭国"、"迁都"），忽略纯年月记录。

8.3 事件重复

问题：同一历史事件在多个章节被记述（如"武王伐纣"出现在周本纪、齐太公世家、鲁周公世家等）。

方案：各章独立提取，不在提取阶段去重。后续可构建跨章事件关联表。

8.4 模糊边界

问题：有些段落难以判断是否构成独立事件（如纯粹的地理描述、制度罗列）。

方案：优先提取有人物行为的段落。纯描述性内容（如禹划定九州的地理描写）视为一个大事件的子部分。

9. 扩展方向

> 以下方向已在 SKILL_05a_事件关系发现.md 中实现完整方案，包括8种关系类型、自动+LLM发现算法、事件链提取、实体三元组、地铁图可视化等。

9.1 跨章事件关联

从130章的事件索引中，构建跨章节的事件关联图：
- 同一事件的多章记述 → 合并为统一事件，标注多个来源
- 因果关系 → 事件A导致事件B
- 时序关系 → 事件时间线

9.2 事件知识图谱

将事件与已有的实体索引关联：
- 人物参与了哪些事件
- 地点发生了哪些事件
- 朝代包含哪些事件

9.3 事件分类增强

基于已提取的事件，训练事件分类模型，用于：
- 自动分类验证
- 发现遗漏的事件类型
- 跨文本的事件对比

10. 适用范围

本方法适用于：
- 纪传体史书（如《汉书》、《后汉书》、《三国志》等二十四史）
- 编年体史书（如《资治通鉴》、《春秋》）
- 其他有叙事性的古籍文本

关键前提：
1. 文本已完成实体标注（提供上下文信息帮助事件识别）

2. 文本有段落编号（提供精确定位）
3. 有明确的章节分类（影响提示词设计）

SKILL 04b: 十表事件处理

适用范围

史记十表（013-022）的事件提取与年代标注。十表与叙事类章节（本纪/世家/列传）有本质区别：**表体自带纪年结构，无需推断年代**。

十表分类

类型	章号	名称	时间跨度	特点
世表	013	三代世表	前2300—前841年	世系表，按代列国，少事件
年表	014	十二诸侯年表	前841—前477年	逐年记录，序言重
年表	015	六国年表	前476—前207年	六国逐年，覆盖最好
月表	016	秦楚之际月表	前209—前202年	**按月记录**，精度最高
王表	017	汉兴以来诸侯王年表	前202—前101年	诸侯王封/除一览
侯表	018	高祖功臣侯者年表	前202—前87年	功臣侯国表
侯表	019	惠景间侯者年表	前194—前141年	惠帝景帝间封侯

类型	章号	名称	时间跨度	特点
侯表	020	建元以来侯者年表	前140—前87年	武帝时期封侯
侯表	021	建元已来王子侯者年表	前127—前87年	推恩令后王子侯
年表	022	汉兴以来将相名臣年表	前206—前20年	逐年：大事/丞相/将/御史大夫

与叙事类章节的核心差异

维度	叙事类（本纪/世家/列传）	十表
年代来源	需推断（插值/交叉验证）	表体自带纪年
单锚点塌缩风险	高（60+事件塌缩到一年）	无
确定性标注	混合（()、[]、[约]）	绝大多数用（）精确标注
事件粒度	按叙事段落	按年/月/封侯批次
序言vs表体	全文为叙事	序言=论述，表体=结构化数据
反思修正量	平均15-30处/章	0-5处/章（格式清理为主）

事件提取策略

序言（太史公曰/序论）

序言通常是太史公对本表主题的综论，提取方法同叙事类章节：

- 识别关键论述段落
- 年代需从上下文推断或交叉验证

- 注意：序言中的年代多为概括性描述（如"齐晋秦楚四海迭兴"），可用时间范围标注

表体（结构化数据）

按表体结构不同采用不同提取策略：

逐年型（014、015、022）

表体按公元年逐行排列，每行含多列（各国/各职位），提取粒度：

- **重大政治军事事件**（战争、即位、灭国、变法等）→ 每事一条
- **年代直接取表头纪年** → 用（公元前XXX年）精确标注
- 段落位置标为 `表（前XXX年）`

月表型（016）

按月排列，精度最高：

- **每月重大事件** → 每事一条
- **年代精确到月** → `%二世元年七月%（公元前209年）`
- **注意秦历岁首十月问题**：秦及汉初（前221—前104年）以十月为岁首，十月至十二月属于该年（非次年）。核对时以年表为准

世表型（013）

按世系排列，时间跨度极大（2000+年），每个世系节点可提取：

- **始封/初建** → 取该君主即位年
- **重大世系变动**（灭国、篡位） → 交叉验证年表
- 不要在相距数百年的节点之间插值

侯表型（018-021）

表体按侯国逐条排列，提取粒度：

- **按封侯批次分组**：同日/同年封侯的多人合为一条事件
- 例：高祖六年（前201年）大封功臣百余人 → 一条事件
- **重大政治事件**：国除（酎金、坐罪等）按批次合并
- 例：元鼎五年酎金坐罪国除106人 → 一条事件

- **典型个案**：特别重要的封侯/国除单列
- 例：韩信谋反被诛、张良封留侯

王表型（017）

表体按诸侯王列表：

- **分封批次** → 如高祖六年立同姓九王
- **重大叛乱/废除** → 如七国之乱
- **制度变革** → 如推恩令

年代标注规则

表体事件

表体事件自带纪年，标注规则简单：

1. **表头有明确公元年** → `（公元前XXX年）` 精确标注
2. **表头有纪年标记**（如 `%元朔二年%`）→ 用基准年换算后 `（公元前XXX年）`
3. **跨年/跨月事件** → 用范围标注 `（公元前XXX年–前YYY年）`

序言事件

序言事件按叙事类规则处理，参见 `SKILL_04c_事件年代推断.md`。

秦历月份偏移

秦至汉初使用颛顼历，以十月为岁首：

颛顼历月份	对应位置	注意事项
十月—十二月	该年开头	不是上年末
正月—九月	该年后半	正常对应

例：016-005　陈涉死于 `%二世元年十二月%`，对应公元前208年（非前209年），需以年表确认。

事件提取现状与补充计划

现有事件数

章号	名称	现有事件	目标	补充方向
013	三代世表	32	32	✅已完成（修复表格：补充殷世系29行、五帝夏世系22行；新增殷代12条事件）
014	十二诸侯年表	12	~25	表体按年提取
015	六国年表	33	~35	基本覆盖
016	秦楚之际月表	19	~25	表体按月补充
017	诸侯王年表	11	~20	表体分封/废除批次
018	高祖功臣侯者年表	11	~20	封侯批次分组
019	惠景间侯者年表	8	~15	封侯批次+国除
020	建元以来侯者年表	10	~15	封侯批次+军功
021	王子侯者年表	3	~15	**最需补充**：推恩令分封批次
022	将相名臣年表	16	~30	前157—前20年缺失

推荐执行顺序

1. **022**（原型）— 最清晰的逐年结构，现仅覆盖前206—前158年，缺前157—前20年
2. **018-021**（侯表组）— 封侯批次+国除分组
3. **014-015**（年表组）— 表体按年提取关键事件
4. **016**（月表）— 按月补充
5. **013**（世表）— 主要从序言补充

质量检查清单

- [ ] 概览表与详情表年代一致
- [ ] 纪年标记（%...%）与公元年换算正确
- [ ] 精确标注（）/ 推算标注 [] 使用得当
- [ ] 无时间字段多重标注冗余
- [ ] 无脚本修正指令残留（"统一为"等文本）
- [ ] 秦历月份偏移已核对
- [ ] 侯表事件按批次合理分组（非逐人列举）
- [ ] 序言事件与表体事件区分标注段落位置

SKILL 04c: 事件年代推断

任务定义

将史记事件索引中的事件，通过《中国历史大事年表》交叉验证和reign-year换算，标注公元前纪年，并记录推断依据。

标记规范

标记	含义	使用条件
（公元前XXX年）	精确纪年	年表有明确记载，或reign-year可精确换算
[公元前XXX年]	确定推算	年表无直接记载，但可从基准年精确推算
[约公元前XXX年]	近似推算	只能估算范围，无法精确定位

字段位置

年代推断放在每个事件详情section的**最后一行**（段落位置之后）：

```
### 046-001 陈完出生与预言
- **事件类型**: 家族事件
- **时间**: [约公元前710年]
- **地点**: &陈&
- **主要人物**: @陈完@
- **涉及朝代**: &陈&
- **事件描述**: ……
- **原文引用**: "……"
- **段落位置**: [1]
- **年代推断**: 陈厉公被杀于前707年，陈完为厉公之子，出生当在厉公在位期间
```

核心方法：年表交叉验证

第一优先：查年表

读取 `kg/chronology/data/中国历史大事年表.md`，按 `### 前XXX年` 逐条查找事件。年表覆盖从远古到前87年共684个年份条目。

年表中有明确记载的事件 → 直接用（公元前XXX年）

第二优先：reign-year换算

原文有 `%秦昭王十三年%` 等纪年标记时，用基准年换算。

关键基准年表（130章review实证）

远古至西周:
基准	公元前	来源
武王克殷	前1046年	年表
共和元年	前841年	年表（中国纪年确切起点）

春秋:
基准	公元前	来源
鲁隐公元年	前722年	春秋纪年起点
齐桓公（小白）元年	前685年	年表
晋献公元年	前676年	年表
楚庄王元年	前613年	年表
吴王阖闾元年	前514年	年表
勾践元年	前496年	年表
齐景公元年	前547年	年表

战国:
基准	公元前	来源
田氏代齐（田和列为诸侯）	前386年	年表
商鞅变法（孝公元年）	前361年	年表
齐威王元年	前356年	年表
齐宣王元年	前319年	年表
齐湣王元年	前300年	年表

燕昭王即位	前311年	年表
秦惠文王元年	前337年	年表
秦武王元年	前310年	年表
秦昭王元年	前306年	年表（最常用基准之一）
赵惠文王元年	前298年	年表
赵孝成王元年	前265年	年表
魏安釐王元年	前276年	年表
楚顷襄王元年	前298年	年表
楚考烈王元年	前262年	年表

秦:

基准	公元前	来源
秦始皇元年	前246年	年表
秦二世元年	前209年	年表

秦汉之际至西汉:

基准	公元前	来源
汉元年	前206年	年表
高祖元年（称帝）	前202年	年表
惠帝元年	前194年	年表
高后元年（称制）	前187年	年表
文帝元年（前元）	前179年	年表
文帝后元元年	前163年	年表
景帝元年（前元）	前156年	年表
景帝中元元年	前149年	年表
景帝后元元年	前143年	年表
武帝建元元年	前140年	年表
元光元年	前134年	年表
元朔元年	前128年	年表
元狩元年	前122年	年表
元鼎元年	前116年	年表
元封元年	前110年	年表
太初元年	前104年	年表

第三优先：已知重大事件定年

事件	公元前	适用章节
长平之战	前260年	073白起、076平原君、079范雎等
邯郸之围	前258—前257年	076平原君、077魏公子、078春申君
乐毅伐齐	前284年	080乐毅、082田单
田单复齐	前279年	046田敬仲完、080乐毅、082田单
陈胜起义	前209年	048陈涉、007项羽等
鸿门宴	前206年	007项羽、008高祖
垓下之战	前202年	007项羽、008高祖、092淮阴侯
白登之围	前200年	008高祖、110匈奴
诛诸吕	前180年	009吕太后、057绛侯周勃
七国之乱	前154年	011孝景、058梁孝王、106吴王濞

第四优先：章节插值

同章已有精确纪年事件作为锚点，按时间顺序前后推断。**但必须警惕以下陷阱**。

常见错误模式（130章review实证总结）

错误1：单锚点塌缩

最常见的系统性错误。当章节只有一两个锚点时，所有无纪年事件被插值到同一年份。

130章中发现的典型案例：

章节	塌缩年份	实际跨度	受影响事件数
001 五帝本纪	前2290年	前2700—前2100年	60+
046 田敬仲完世家	前655年	前710—前259年	12
053 萧相国世家	前202/前196年	前220—前192年	14
057 绛侯周勃世家	前206/前180/前154年	三组塌缩	21
130 太史公自序	前104年	前220—前91年	13

识别方法：如果一章中多个事件共享完全相同的 `[约公元前XXX年]`，几乎可以确定是单锚点塌缩。

错误2：锚点本身计算错误

比塌缩更隐蔽——锚点日期就是错的，导致所有依赖它的事件全部偏移。

实证案例：
- **046-008 田和代齐**：原锚点计算为前602年，实为前386年，偏差216年。原因：把"陈完奔齐"后的世代数错误计算为年数
- **046-003 陈完奔齐**：原标注前655年，年表确认为前672年
- **046-020 田单复齐**：因046-008锚点错误，标注为前647年，实为前279年，偏差368年

教训：不能只检查插值事件，**必须首先验证锚点本身是否正确**。

错误3：前元/中元/后元混淆

西汉文帝、景帝有前元/中元/后元多段年号，同一个"N年"对应不同公元年。

实证案例（011孝景本纪，28处修正）：
- "中元年"≠"元年"：景帝中元元年=前149年，景帝元年=前156年，差7年
- "后元年"≠"元年"：景帝后元元年=前143年，差13年
- "后三年"=前141年（景帝崩），不是前154年

规则：遇到文帝/景帝纪年，必须先确定是前元、中元还是后元，再换算。

错误4：年号/帝王混淆

"N年"指代的君主不是你以为的那位。

实证案例：
- **053-017 萧何卒**："二年"被标注为汉二年（前205年），实为惠帝二年（前193年），偏差12年。"二年"在萧何传的上下文中指惠帝二年，而非高祖汉二年
- **046-009 齐桓公午**：被标注为（前681年），那是春秋齐桓公（小白）的年份。田齐桓公午元年=前370年，完全不同的人

规则：
- "汉N年"专指刘邦称汉王后的纪年（前206—前202），不等于惠帝N年
- 同名君主（如"齐桓公"）要区分是春秋齐桓公小白还是田齐桓公午
- 从上下文判断"N年"的归属，不能默认用章节主角

错误5：追叙事件误用当前时间

传记中经常追叙早年经历，这些事件的时间不等于叙述所在位置的时间。

实证案例：
- **006-002 秦王政即位**：在始皇出生（前259年）附近叙述，被标注为前259年，实为前247年（庄襄王死年）
- **006-003 吕不韦为相**：同样被塌缩到前259年，实为前249年（庄襄王元年）
- **053-001 萧何为沛主吏掾**：在汉朝建立后追叙秦代经历，被标注为前202年，实为约前220年

规则：传记开头常有追叙段落（"高祖微时"、"未起时"），不能用后文锚点标注追叙事件。

错误6：写作时间≠事件时间

序言、总论性质的章节，锚点往往是写作时间。

实证案例（130太史公自序）：
- 唯一锚点"太初元年始作太史公书"（前104年）是写作时间
- 但本章回顾了前220年（秦统一）到前91年（史记完成）的历史
- 13个事件全部被错误标注为"约前104年"

错误7：确定性标注不足

年表有明确记载的事件，却被标注为 `[约公元前XXX年]`（推断），应升级为 `（公元前XXX年）`（精确）。

实证案例：
- **006-015 嫪毐封侯**：年表明确记载前239年，原标 `[约公元前239年]`
- **006-104 项籍灭秦**：年表明确记载前206年，原标 `[约公元前206年]`

规则：标注后回查年表，凡年表有载的事件一律用圆括号。

错误8：世代数≠年数

世家类章节常列某人"N世孙"，不能把世代数直接换算成年数。

实证案例（046田敬仲完世家）：
- 从陈完奔齐（前672年）到田和代齐（前386年），经历约十代约286年
- 原计算把这段算成了约53年（前672→前602？），相当于每代5年，明显不合理
- 合理的世代间隔为25—30年/代

推理逻辑模式（2018条年代推断实证总结）

以下是从130章共2018条年代推断中归纳的推理逻辑，按使用频率排序。

逻辑1：年表直接查证（~800例）

最可靠的方法。在年表中按 `### 前XXX年` 查找事件关键词。

典型推断写法：
- "年表前672年明确记载'陈宣公杀太子御寇。陈完奔齐'"
- "据《中国历史大事年表》前284年条：'燕乐毅联合五国伐齐'"
- "年表前260年条明确记载长平之战"

注意：年表条目中的事件描述可能与事件索引的命名不完全一致，需按人物+事件性质匹配，而非精确字符串匹配。

逻辑2：Reign-Year精确换算（~400例）

原文有 `%秦昭王十三年%` 等纪年标记时，用公式换算：

```
公元前年份 = 基准元年 - (N - 1)
```

例：秦昭王十三年 = 306 - (13-1) = 前294年

特殊情况：

情况	规则	实证
"元年"	= 基准年本身	考烈王元年 = 前262年
"前元/中元/后元"	各有各的元年	景帝中元元年=前149年≠景帝元年前156年
"汉N年"	以前206年为元年	汉四年=前203年（非高祖四年前199年）
"高祖N年"（称帝后）	以前202年为元年	高祖六年=前201年
诸侯王"第N年"	以封王年为元年	英布淮南王第七年=前203封→前196年反
"後N年"/"明年"	从上文事件起算	"其明年"=上文年份+1

易错的Reign-Year：

原文	易错理解	正确理解	实证
"二年"	汉二年（前205）	惠帝二年（前193）	053-017萧何卒
"七年"	高祖七年（前200）	淮南王第七年（前196）	017-007英布反
"六年"	高祖六年（前201）	梁王第六年（前196）	017-008彭越反
"四年"	诸樊四年（前557）	馀祭四年（前544）	031-009季札聘鲁

原文	易错理解	正确理解	实证
"七年"	孝惠七年（前188）	高后七年（前181）	009吕太后本纪

逻辑3：事件顺序与"紧接"推断（~200例）

原文按时间顺序叙事，利用相邻事件的时间关系。

模式:
- **紧接**："紧接八年成蟜叛事之后、九年嫪毐之乱之前" → 前239年
- **同年**："与灭夏同年约前1600年"、"与窃符同日"
- **同期**："与封功臣侯、封吕台为王同期"
- **此后/明年**："封博望侯(前123年)之翌年" → 前122年
- **N年后**："灭夏后十三年而死" → 前1600+13=前1587年
- **期间**："留赵十年"=前257—前247年

逻辑4：人物活跃期定位（~150例）

无纪年标记时，用人物的在位期间或活跃期来约束。

模式:
- **君主在位期**："齐景公在位前547—前490年，晏子约卒于前500年"
- **任职期间**："春申君相楚八年=前262-8+1=前255年"
- **生年推算**："刘邦生于前256年，约三十余岁任亭长" → 约前220年
- **卒年/崩年**：死亡事件取去世年而非生卒中点
- "汤崩约前1587年" → 用前1587而非在位中点
- "武王崩约在前1043年"

关键规则:
- 事件名含"崩"、"薨"、"卒"、"死"、"自杀"、"被杀" → 取去世年
- 事件名含"即位"、"立"、"践祚" → 取即位年
- 事件名含"封"、"拜" → 查年表确认具体年份

逻辑5：跨章节交叉定位（~100例）

> 跨章互见关系的完整发现算法和置信度分级，详见 SKILL_05a_事件关系发现.md 第三节。

同一事件在多个章节出现，取精确度最高的标注。

优先级：本纪 > 年表 > 世家 > 列传

实证案例：
- 长平之战：073白起传、076平原君传、079范雎传均涉及 → 以年表前260年为准
- 田单复齐：046田敬仲完世家、080乐毅传、082田单传 → 以年表前279年为准
- 诛诸吕：009吕太后本纪、057绛侯周勃世家 → 以本纪前180年为准

逻辑6：追叙段落独立定年（~50例）

传记开头的追叙（"未起时"、"高祖微时"、早年经历）不能用后文锚点。

识别标志：
- "秦统一后设亭长之制" → 追叙秦代经历，不是汉代事件
- "为沛主吏掾" → 秦代任吏，约前220年
- "少时学书/学剑" → 童年期，用生年推算

实证：
- 053-001"萧何为沛主吏掾"：原标前202年（汉朝建立年），实为约前220年（秦代）
- 007-002"项籍少时学书剑"：项羽生于前232年，"少时"约前220年
- 008-002"高祖为亭长"：刘邦生于前256年，约三十余岁任职，约前220年

逻辑7：时间跨度与区间标注（~80例）

有些事件跨越多年，应标注时间区间而非单一年份。

模式：
- "前284年—前279年"：乐毅伐齐至田单复齐
- "前258—前257年"：邯郸之围
- "前206—前202年"：楚汉相争
- "约前320—前312年"：孟子游说齐梁

逻辑8：世系推算与世代间隔（~30例）

世家类章节常有"自X至Y凡N世"的记载。

规则：
- 合理的世代间隔：25—30年/代
- "自太伯至寿梦十九世" → 跨约500年

- "自契至汤十四世" → 跨约500年（前2100→前1600）
- 每代5年或更短 → 明显计算错误

逻辑9：春秋纪年换算（~30例）

春秋时期常用鲁公纪年。《春秋》纪年自鲁隐公元年（前722年）至鲁哀公十四年（前481年）。

常用鲁公元年基准：

鲁公	元年
隐公	前722年
桓公	前711年
庄公	前693年
僖公	前659年
文公	前626年
宣公	前608年
成公	前590年
襄公	前572年
昭公	前541年
定公	前509年
哀公	前494年

交叉验证：春秋事件同时可查《左传》纪年和年表，三源比对最可靠。

逻辑10：反向推算（从已知结果倒推）

知道结果年份，倒推过程年份。

实证:
- "灭夏约前1600年，汤即位后十七年灭夏" → 即位约前1617年
- "信陵君卒后十八年秦灭魏=前225年" → 信陵君卒约前243年
- "秦灭魏前225年，信陵君在位期间" → 信陵君活跃期约前276—前243年

逻辑11：制度/事件背景定年

利用制度实施时间来约束事件年份。

实证:
- "秦统一后设亭长之制" → 事件在前221年之后
- "推恩令" → 元朔二年（前127年）
- "诸侯丞相改称为相" → 景帝中五年（前145年）
- "西门豹治邺" → 年表前403年"魏文侯用西门豹治邺"

逻辑12：年表中的人物生卒标注

年表在某些条目中附注人物生卒年，可直接使用。

实证:
- 前202年条"项羽（前232—）死" → 项羽生年前232年
- 前195年条"高祖死（前256或前247—）" → 刘邦生年存疑
- 前479年条"孔子卒（前551—）" → 孔子生年前551年

1. 夏代事件章节兜底值塌缩

问题: 002-028至002-039大量事件被二轮插值为前1900年（章节兜底值），而概览表中有不同的递减年份。单锚点塌缩问题。

典型案例: 002-028太康失国：概览表标前2110年，详情却塌缩到前1900年（章节兜底值）。002-033少康复国：概览前1890年，详情前1900年。实际应按年表"夏始前2070年、灭夏约前1600年"区间内按帝王世系分配。

识别方法: 检查夏本纪002-028至002-039各事件详情时间字段，若全部为同一年份（前1900年），即为此类错误。修正方法：以年表朝代起止年（前2070—前1600年）为区间，按帝王在位顺序和世代间隔逐一定年。

2. 夏代概览-详情双轨年份分裂

问题：概览表与详情年份大面积不一致：002-028至002-039的概览表有递减序列但详情全部为前1900年。

典型案例： 002-030相失国：概览表标[约公元前2000年]（等距插值），详情标[约公元前1900年]（兜底值），两者均非正确年份。002-035少康中兴后事迹：概览前1780年，详情前1900年，两套错误值互相矛盾。

识别方法： 逐事件比对概览表时间列与详情时间字段，若系统性不一致（概览有递减序列、详情全部相同），即为此类双轨偏差。修正时以年表为准独立定年，再同步概览与详情。与模式31（概览与详情年份双轨偏差）为同一类错误在夏本纪的具体表现。

3. 禹事件误用生卒中点替代年表即位年

问题：禹相关事件（002-006至002-027）全部使用禹生卒中点前2150年，但禹即位应为前2070年（夏商周断代工程）。

典型案例：002-025禹即天子位：标注[约公元前2150年]（禹生卒中点），年表明确记载夏始前2070年，偏差80年。002-027禹东巡会稽崩：标注前2150年，实应为禹在位末年（约前2050年代）。

识别方法： 远古章节中，检查事件年份是否恰好等于人物生卒年中点（可查person_lifespans.json）。若年表有该人物的即位年或朝代起始年，应优先使用年表数据。禹的正确锚点：即位=前2070年（夏始），而非生卒中点前2150年。与模式30（传说时代生卒中点不可靠）为同一类错误。

4. 复合总结段落单点标注

问题： 传记末尾常有总结性段落，将主角在位期间多个不同年份的德政、功绩合并叙述。自动标注倾向于选取其中一个可查证的子事件年份作为整段的时间标注，遗漏了段落实际涵盖的时间跨度。与'单锚点塌缩'不同，这里并非缺乏锚点，而是段落本身就是跨年度的综合叙述。

典型案例： 010-043以德化南越：段落包含南越称臣(前179年)、匈奴和亲(前162年等)、吴王赐杖(约前170年代)等多个子事件，但仅标注为前179年。010-042文帝节俭德政：'即位二十三年'的全朝总结仅标注后元年间。

识别方法： 识别方法：(1)事件描述中出现多个独立子事件且涉及不同人物/事件背景；(2)原文有'即位N年'、'自…以来'等总括性用语；(3)段落位置在章节末尾（传记总评区域）。修正方法：标注为在位全期的时间区间。

5. 连续计年与改元计年混淆风险

问题: 当帝王在位期间有改元（如文帝前元→后元），原文有时使用从即位起的连续计年（如'十七年'），有时使用改元后的新计年（如'后二年'）。两种计年混用可能导致换算错误。

典型案例: 010-038新垣平伏诛：原文'十七年'为即位后连续第17年=后元年=前163年，而非'前元十七年'（前元只有16年）。若误以为'十七年'是前元纪年，会尝试计算179-(17-1)=前163年，结果恰好相同，但推理过程有误。在其他帝王（如景帝前元7年→中元→后元）情况下，连续计年与分段计年的混淆会导致实际错误。

识别方法: 识别方法：(1)检查'N年'是否超出该元的年数范围（如前元只有16年却出现'十七年'）；(2)查看原文上下文是否有'更为元年'等改元标志；(3)将连续计年和分段计年分别换算，比对结果是否一致。

6. 祖先事件误用后裔年代

问题: 世家/世表类章节中，始祖（如契、后稷）的事件被标注为其著名后裔（如汤、文王）的年代，因为自动标注脚本在人物生卒年交集计算中使用了后裔的生卒年而非始祖本人的活跃期。这与'世代数≠年数'（错误8）相关但不同：错误8是换算公式错误，此处是直接选错了参照人物。

典型案例: 013-010契无父而生：契与尧同时代（约前2300年），但因主要人物字段包含@汤@，被标注为汤的生卒中点前1629年，偏差约700年

识别方法: 事件描述中出现'N世至XX'、'后世至XX'等跨世代叙述时，检查标注年份是否误用了后世人物的年代而非事件当事人的年代

7. 原文已含精确纪年但被插值覆盖

问题: 事件的%纪年标记%中已包含精确的公元前年份（如%公元前293%），但自动标注脚本忽略了%标记中的年份信息，仍然使用插值算法生成错误的推断年份。这是比'单锚点塌缩'更严重的错误，因为正确答案已经存在于数据中却被忽略。

典型案例: 015-017白起击伊阙：%公元前293%已给出精确年份，但被插值为[约公元前307年]；015-020长平之战：%公元前260%被插值为[约公元前214年]或[约公元前307年]

识别方法: 检查%纪年标记%中是否包含'公元前NNN'格式的精确年份，若有则直接采用，无需插值。对比%标记年份与[]推断年份，若不一致则为此类错误。

8. 秦历岁首十月导致的跨年误标

问题: 秦及汉初以十月为岁首，同一个reign-year（如二世元年）的十月至十二月在太阳历上跨越两个公元年。简单地将整个reign-year映射为单一公元年（如二世元年=前209年）会导致十月至十二月的事件被错误地归入前一个公元年。年表按太阳历编排，与reign-year整体映射产生矛盾。

典型案例: 016-005陈涉死：二世元年十二月，按reign-year映射标为前209年，但年表明确记载于前208年条。段落位置亦标'表（前208年十二月）'，说明原始月表数据与概览表年份不一致。

识别方法: 检查秦至汉初（前221至前104年太初改历前）事件中，凡纪年标记含十月、十一月、十二月者，需核对年表确认其实际公元年是否与reign-year整体映射一致。若年表将该事件置于下一个公元年，则应以年表为准。

9. 确定性标注批量不足

问题: 本章大量事件的年代推断理由中已明确引用年表条目作为依据，但标注仍使用[约公元前XXX年]而非（公元前XXX年）。这属于已知错误模式7（确定性标注不足）的批量出现。030章41个事件中，有4个事件（030-025、030-028、030-030、030-033）存在此问题，推断理由已查到年表记载但未升级标注格式。这表明自动标注脚本在生成推断标注后缺少一个'回查升级'步骤。

典型案例: 030-025桑弘羊盐铁官营：推断理由写"年表前119年条记载"，但标注仍为[约公元前119年]，应升级为（公元前119年）。030-033均输法实施：推断理由引用年表前110年条，标注却用[约]。

识别方法: 检查年代推断行中是否出现"年表""年表前XXX年条""年表明确记载"等措辞，若有则该事件的标注应为圆括号（精确）而非方括号（推断/近似）。批量扫描：搜索推断行含"年表"但时间字段含"[约"的事件。

10. 通论章节多时代单锚点塌缩

问题: 封禅书等通论性质章节横跨上古到汉武帝数千年，但自动标注仅以秦二世元年(前209年)为唯一锚点，导致远古殷商、西周春秋、战国秦国诸事件全部塌缩到前209年。这比一般的单锚点塌缩更严重，因为涉及的时间跨度可达2000年以上。

典型案例: 028-003至028-013共9个事件（殷太戊至秦缪公，跨约前1535年至前659年）全部被标为[约公元前209年]，偏差从450年到1300年不等。

识别方法：检查通论/八书类章节（封禅书、礼书、乐书等），若发现跨多个朝代的事件共享同一推断年份，即为此类错误。修正方法：按各事件所属帝王的在位期查年表独立定年。

11. 景帝元年误标为武帝元年

问题：　武帝即位于景帝后元三年(前141年)，建元元年=前140年。但自动标注将武帝'元年'错误对应为前156年（景帝前元年），导致武帝在位期间所有事件偏差16年。这是错误4（年号/帝王混淆）的一个具体变种，在封禅书中影响极大。

典型案例：　028-035/036标注'元年（公元前156年）'，实为武帝建元元年=前140年；028-036赵绾被诛实为前139年。后续028-037至028-048共12个事件因此锚点全部错误。

识别方法：　凡涉及'今上'（武帝）的事件，检查是否误用前156年（景帝元年）作为武帝元年。正确基准：武帝建元元年=前140年。

12. 前后君主纪年衔接处的reign-year错位

问题：　当一位君主被弒、新君即位时，事件索引容易将旧君末年与新君纪年标记混淆。如032-032中，'四年'指简公四年（前481年），但标注使用了悼公四年/悼公被杀年（前485年），将悼公末年的公元前年份直接赋给了简公的reign-year标记。这与'年号/帝王混淆'（错误4）相关，但更具体：发生在同一家族/国家的相邻两任君主之间，因弒君-立新君的叙事紧密衔接而导致'N年'的归属君主被错误判定。

典型案例：　032-032田常弒简公：'四年'为简公四年=前481年，被错标为前485年（悼公被杀年）

识别方法：检查弒君/立新君事件中的'N年'标记，确认是前任还是继任的纪年。特别注意当同一段落叙述了前任被杀和继任被立时，后文'N年'通常指继任者的纪年。

13. 同名君主跨国混淆（国君号混淆）

问题：　不同诸侯国的君主有相同的谥号（如郑灵公vs秦灵公、郑缪公vs秦缪公），自动标注将一国君主的元年误用为另一国君主的元年，导致年份大幅偏移。与已有'错误4年号/帝王混淆'类似但更具体：不是同一国的同名君主，而是不同国家的同谥号君主。

典型案例：042-016染指尝鼋弒灵公：郑灵公元年=前605年，但被标注为前424年（秦灵公元年）。042-015弦高犒秦师：郑缪公元年=前627年，但被标注为前659年（秦缪公元年）。

识别方法: 当事件所属章节为某国世家、原文纪年标记含某国君主名号时，检查所标年份是否对应了同谥号的他国君主。尤其注意灵公、穆公、文公、桓公等高频谥号。

14. 即位年与元年纪年错位

问题: 古代新君即位的物理行为（被立、即位）与其'元年'纪年常不在同一公元年。一般做法是即位当年仍用前君纪年，次年才算新君元年。当事件时间字段使用'元年'却标注即位的物理发生年份时，产生矛盾。039-033悼公即位：物理即位在前573年（厉公被杀当年），但悼公元年是前572年。事件标注'元年正月庚申（公元前573年）'将元年与即位年混为一谈。

典型案例: 039-033晋悼公即位：厉公被杀于前573年，悼公同年被立，但悼公元年=前572年（次年改元）。标注"元年（公元前573年）"混淆了即位年与元年。又如汉高祖：前206年入关称汉王（即位年），但汉元年起算亦为前206年（此处恰好一致，因楚义帝以该年封诸侯王）。

识别方法: 当事件名含"即位"且纪年标记含"元年"时，检查标注年份是否为前任君主被杀/去世的同一年。若是，需确认该国是否即位当年即改元，还是次年才改元。春秋列国多次年改元，秦汉则多即位当年改元。查年表中新君元年条目可确认。

15. 概览表与详情表年份大面积系统性分裂

问题: 概览表与详情表对同一批事件使用了不同的年份生成策略（概览用等距插值或人物中点，详情用单锚点塌缩或兜底值），导致同一事件在两处标注完全不同的年份，两个值可能都不正确。与模式31（双轨偏差）相同，但此处强调"大面积系统性"——整章数十个事件均受影响。

典型案例: 002-028至002-039共12个事件：概览从前2110到前1670年等距分布，详情全部为前1900年。121儒林列传：概览用人物生卒中点（如前158年），详情全部塌缩到前209年（陈涉起义锚点），两套年份均与真实年份相差甚远。

识别方法: 批量比对概览表时间列与详情时间字段：若概览有规律递变而详情全部相同，或两者系统性不一致（差异>10个事件），即为此类错误。通常出现在远古章节（001-003）和通论章节（121儒林、128龟策等）。

16. 魏惠王纪年off-by-one问题

问题: 魏惠王纪年存在学术争议，《史记》与《竹书纪年》等文献的惠王元年差1年（前370年vs前369年）。年表以前369年为惠王元年，但桂陵之战（年表前353年=惠王十七年）、马陵之战（年表前341年=惠王二十九年）等事件的reign-year标记与年表元年换

算结果差1年。事件索引直接采用年表事件年份是正确做法，但标注'原文记载十八年，即公元前353年'的推断理由在数学上不严密

典型案例：044-011桂陵之战：%十八年%标注为前353年，但按年表惠王元年=前369年换算，十八年=前352年。年表前353年确有桂陵之战，但前352年标'惠王十八年'另有他事

识别方法：检查魏惠王纪年事件：将%N年%按369-(N-1)换算后与年表事件年份比较，若差1年则需确认是否采用了不同的元年基准

17. 田齐世系纪年分歧导致系统性偏移

问题：《史记·田敬仲完世家》的威王/宣王纪年与竹书纪年及年表存在约22年的系统性偏差。《史记》将威王元年定在约前378年，而年表（采竹书纪年）定在前356年。自动标注若直接采用《史记》原文中的reign-year数字换算，会导致威王至宣王期间所有事件偏早22年。这与'锚点本身计算错误'（错误2）不同：此处不是计算公式错误，而是两套权威史料体系本身对同一国君的在位年份有根本性分歧。

典型案例：046-010威王九年：按《史记》=前370年，按年表=前348年，偏差22年。046-012威王二十四年：按《史记》=前355年，按年表=前333年。046-013桂陵之战：《史记》威王二十六年=前353年与年表齐威王四年=前353年恰好同值（因年表记载的是实际事件年份而非reign-year换算结果）。

识别方法：当同一章节中部分reign-year换算值与年表一致（如桂陵、马陵之战）而另一部分不一致时，检查是否存在两套纪年系统的差异。关键判据：年表中同一国君的'N年'标注与《史记》文本的'N年'指向不同公元年。

18. 年代推断理由中锚点年份引用错误

问题：详情的年代推断行引用其他事件作为锚点时，写入了错误的年份数字。如053-013/014推断理由中写'后锚053-017(前205年)'，但053-017实际年份为前193年（惠帝二年）；053-018推断理由写'前锚053-017(前205年)'，同样引用错误。这可能是脚本在处理'%二年%'时误将其解读为汉二年（前205年）而非惠帝二年（前193年），属于错误模式4（年号/帝王混淆）在推断理由层面的延伸表现。

典型案例：053-013年代推断：'后锚053-017(前205年)'，实际053-017萧何卒为惠帝二年=前193年，非汉二年=前205年。脚本将'%二年%'误解为汉王二年而非惠帝二年。

识别方法：检查年代推断行中引用的锚点事件ID及其括号内年份，与该锚点事件的实际标注年份进行比对。若不一致，则为此类错误。特别注意含'%N年%'纪年标记的事件，需确认N年归属的帝王。

19. 复合传记事件以封王年代替核心事件年

问题: 世家/列传中，一段文字先叙封王年，再叙该王一生行状（如击吴、好儒、卒年等），自动标注倾向于将%纪年标记%（封王年）作为整个事件的时间标注，即使事件名称指向的核心行为（如'击吴'、'卒'）发生在封王后若干年。这与'追叙事件误用当前时间'（错误5）不同：此处不是追叙早年，而是叙述起始年后发生的后续事件。

典型案例: 059-007江都易王非击吴：标注为前155年（封王年），实际击吴在前154年（七国之乱）。059-003临江哀王阏于卒：标注为前155年（封王年），实际卒于前153年（在位三年）。

识别方法: 当事件名含'击'、'卒'、'自杀'、'国除'等动作性词语，但标注年份恰为该人物封王/即位年时，检查事件核心行为是否发生在封王年之后。特别在五宗世家、诸侯王世家等章节中，每段以'以孝景N年用皇子为X王'开头，后叙一生事迹，需逐一区分封王年与事件实际发生年。

20. 《史记》系年与现代学术定年的系统性偏差

问题: 《史记》对某些人物的活跃时代存在系统性错误，导致以《史记》原文为锚点的年代标注与年表（基于现代学术研究）产生大幅偏差。苏秦列传是典型案例：《史记》将苏秦活动定在前334-前333年（与张仪同时代），但1973年马王堆帛书《战国纵横家书》出土后，学界普遍认为苏秦活跃于前311-前284年（燕昭王时期），比《史记》晚约25年。这与'锚点本身计算错误'（错误2）不同：错误2是换算公式错误，此处是《史记》原文叙事框架本身就与史实不符，所有依据《史记》叙事定年的事件都会产生系统性偏移。

典型案例: 069-006至069-014共9个事件以燕文侯（前333年卒）为锚点定在前333年，但年表记苏秦前311年始赴燕昭王处；069-008称'韩宣王'（前332年始立）、069-009称'魏襄王'（前318年始立）、069-010称'齐宣王'（前319年始立），均与前333年的标注产生君主在位期矛盾。

识别方法: 当章节中出现的多位君主的在位期与标注年份产生系统性矛盾（如标注年份时某些君主尚未即位），应检查《史记》原文系年是否与年表存在系统性偏差。特别注意战国纵横家（苏秦、张仪等）的年代问题。

21. 传记'十岁/N岁不得调'导致升迁序列年代压缩

问题: 传记中明确记载人物'事某帝N岁不得调'，说明入仕后经历了N年沉寂期才被推荐升迁。自动标注忽略了这段沉寂期，将升迁事件紧接在入仕年份之后，导致整个升迁序

列被压缩到入仕后1-2年内，而实际应在N年之后。这与'追叙事件误用当前时间'（错误5）不同：此处不是追叙，而是正序叙事中遗漏了明确记载的时间跨度。

典型案例：102-001至102-005：释之入仕(约前179年)后'十岁不得调'，约前170年才被袁盎推荐升迁，前177年为廷尉。但自动标注将谏嗇夫(102-002)、论秦弊(102-003)、劾太子(102-004)全部标注为前178年，忽略了十年沉寂期。

识别方法：检查传记中是否有'N岁/N年不得调/不迁/不用'等表述，若有则升迁事件应在入仕年份+N年之后，而非紧接入仕年份。

22. 传说性人物跨世纪活动期的年代矛盾

问题：扁鹊诊赵简子（约前500年）与扁鹊见齐桓侯（前357年）相隔143年，又与入秦被杀（约前310年）相隔近200年，远超正常人寿命。文本将不同时代的'扁鹊'（可能是医者通称）事迹合并叙述为同一人传记，导致自动标注时无法在合理的人物活跃期内定位所有事件。按文本叙事顺序排列则产生跨越数百年的不合理时间跨度。

典型案例：105-002扁鹊诊赵简子（约前500年）与105-004扁鹊诊齐桓侯（前357年）相隔143年。如按人物生卒中点插值，会将早期事件拉晚或晚期事件推早。

识别方法：检查列传中同一人物的事件时间跨度是否超过合理寿命（80-100年）。若超过，该人物可能是传说性人物或通称，各段事件应按各自的历史背景独立定年，而非用统一的人物活跃期插值。

23. 年代推断行位置错位至非事件区域

问题：事件详情的年代推断行未出现在该事件的详情section中，而是被错误地放置到了文件末尾的统计信息区域，导致该事件详情缺少年代推断行，同时统计区域出现了不属于该区域的内容。

典型案例：118-025衡山王自刭国除：详情部分在段落位置[28]后直接结束，无年代推断行；而'年代推断:　　原文记载元狩元年，即公元前122年'出现在文件末尾的'统计信息'section中。

识别方法：检查每个事件详情section是否包含年代推断行；检查统计信息、关键事件链等非事件区域是否混入了年代推断行。

24. 统一前后时间标注不区分称号语境

问题：秦始皇'帝'号始于前221年统一之后，但自动标注将涉及'秦始皇帝'称号的事件标注为前220年（统一前），未注意到'始皇帝'称号本身暗示事件发生在称帝之后。类似

地，'汉兴'不等于刘邦起兵年（前209年），而是建政年（前206或前202年）。称号/政权名本身蕴含时间下限信息。

典型案例: 129-007乌氏倮比封君、129-008巴寡妇清守丹穴：原文用'秦始皇帝'而非'秦王政'，表明事件在前221年之后，但被标注为前220年

识别方法: 检查事件原文中的政权称号或帝号是否与标注年份矛盾：'秦始皇帝'→前221年后；'汉'→前206年后；'太祖高皇帝'→前202年后

25. 通论章节跨千年事件共享汉代锚点

问题: 儒林列传为通论性质章节，从孔子（前551-前479）到武帝时期（前140-前87年），跨度约400年。唯一的精确纪年锚点是121-004陈涉起义（前209年），导致17/23个事件详情全部塌缩到前209年。概览表使用人物生卒中点做了二次修正（如前158年、前165年等），但这些中点值仍然大量偏离实际年份。这是'通论章节多时代单锚点塌缩'（已有模式10）在列传类章节的又一典型案例，但本章的特殊之处在于：概览表与详情表的年份完全分裂（概览用人物中点，详情用前209年锚点），形成双重错误叠加。

典型案例: 121-003秦焚诗书：概览前158年（人物中点）、详情前209年（锚点塌缩），实为前213年。121-009公孙弘设学官：概览前159年、详情前209年，实为前124年。两套错误年份均与真实年份相差甚远。

识别方法: 检查列传类通论章节（儒林、游侠、酷吏等），若概览表与详情表年份系统性不一致，且详情全部指向同一年份，则为此类双重错误。

26. 裸年份缺少确定性标记

问题: 部分事件的时间字段直接写'公元前XXX年'而不带任何标记符号（既不是圆括号也不是方括号），不符合三级标注规范。这些裸年份实际上都有明确依据，应使用（公元前XXX年）精确标注。

典型案例: 100-001至100-011中有7个事件使用裸年份格式，应统一为（公元前XXX年）圆括号格式

识别方法: 检查时间字段中是否存在不带（）或[]标记的'公元前XXX年'格式

27. 锚点间时序倒置

问题: 当锚点值本身有误时，基于错误锚点的插值会产生时序倒置（前锚反而晚于后锚）

典型案例: 065-012田文为相吴起不悦：标注[约公元前382年]，但后续事件065-013（年表确认前390年）反而更早，违反叙事时间顺序。修正为[约公元前392年]后时序恢复正确。046田敬仲完世家中，因046-008锚点错误（前602年→前386年），导致后续事件046-020（前647年）反而早于046-008，时序倒置。

识别方法: 按事件编号顺序检查公元前年份是否单调递减（越往后越接近公元元年）。若出现后续事件年份反而更早（公元前数字更大），即为时序倒置，需回溯检查相关锚点是否有误。

28. 时间字段重复标注

问题: 部分事件的时间字段中，同一年份标注被重复粘贴2-3次（如009-003重复2次、009-015重复3次），可能是脚本处理时的bug导致。这不影响年份正确性，但影响数据质量和可读性。

典型案例: 009-003惠帝即位：时间字段为"（公元前194年）（公元前194年）"，重复2次。009-015诸吕封王：时间字段重复3次。037卫康叔世家13个事件出现双重标注（如"（公元前813年）（公元前813年）"），为多轮修正叠加未去重所致。

识别方法: 用正则表达式检查时间字段中是否出现连续重复的标注模式：`（公元前\d+年）\s*（公元前\d+年）` 或 `\[.*?\]\s*\[.*?\]`。发现后保留一份，删除重复。

29. 非时间信息混入时间字段

问题: 009-071的时间字段为'夜 （公元前180年）'，将事件发生的时间段描述（'夜'）混入了年份标注字段。时间字段应仅包含标准年份标注，时间段信息应在事件描述中体现。

典型案例: 009-071太尉勃入北军：时间字段为"夜 （公元前180年）"，"夜"为事件发生时段而非年份信息。又如部分事件时间字段混入"战国中期""春秋末年"等叙事性文本替代标准日期格式。

识别方法: 检查时间字段中是否包含非标准年份格式的文本（如汉字时段"夜""晨"、时代描述"战国中期""春秋末年"等）。时间字段应仅含`（公元前XXX年）`、`[公元前XXX年]`、`[约公元前XXX年]`三种格式及纪年标记`%...%`。

30. 传说时代生卒中点不可靠

问题: 远古传说时代（五帝至夏）人物的生卒年本身就是后人推算的虚构数据，用生卒中点作为事件年份缺乏可靠基础。应优先使用年表记载的朝代起止年（如夏始前2070年、汤灭夏约前1600年）作为锚点，再按叙事时序分配各事件年份。

典型案例: 002-025禹即天子位标注前2150年（禹生卒中点），实际年表明确记载夏始前2070年，偏差80年；002-042汤伐桀标注前1629年（汤生卒中点），实际年表记载约前1600年。

识别方法: 远古章节（001-003）中，如果事件年份恰好等于某人物person_lifespans.json中记录的生卒年中点，且年表有更可靠的朝代起止记载，则很可能是此错误。

31. 概览与详情年份双轨偏差

问题: 概览表使用了等距插值法分配年份，但详情部分全部塌缩到单一默认值（如前1900年），导致同一事件在概览和详情中标注完全不同的年份。两个值可能都不正确，且互相矛盾。

典型案例: 002-028至002-039共12个事件，概览从前2110到前1670年等距分布，详情全部为前1900年。如002-028概览=前2110年但详情=前1900年，实际应为前2061年。

识别方法: 自动检查脚本报告概览=前XXX年，详情=前YYY年不一致，且详情中大量事件指向同一年份时即为此模式。

32. 复合事件单年标注遗漏跨年子事件

问题: 事件描述中包含发生在不同年份的两个子事件（如'窦太后崩'前135年和'其明年征文学'前134年），但仅标注了第一个子事件的年份。这与已有的'复合总结段落单点标注'（模式4）类似，但此处不是总结性叙述，而是原文以'其明年'明确标记了时间间隔的两个事件被合并为一条。

典型案例: 028-041窦太后崩与征文学：窦太后崩于前135年，"其明年"征文学为前134年，但事件仅标注前135年。121-008公孙弘上书与设学官：上书在前124年，"明年"设博士弟子在前124年，两个子事件合并为一条。

识别方法: 检查事件描述中是否出现"其明年""明年""后N年"等跨年标记，若有则当前标注仅覆盖第一个子事件。修正方法：标注为区间（如"前135年—前134年"），或以核心子事件的年份为主标注。

33. 时间字段多重标注冗余

问题: 019-003的时间字段中出现三个并列的时间标注（%五十载% [约—前141年] [约—前141年] [公元前194年—前141年]），其中[约—前141年]重复出现两次。这是自动标注脚本在多次修正过程中叠加标注而未清理旧标注导致的格式冗余。

典型案例: 019-003惠景间追修遗功：时间字段含三个并列标注，应统一为单一标注

识别方法: 检查时间字段中是否出现多个方括号标注并列的情况，若有则需要清理合并为单一标注

34. 格式伪迹重复

问题: 脚本自动修正时可能导致时间字段中同一标注被重复插入多次，需检查并去重。

典型案例: 004-034周厉王在位：时间字段"（公元前899年-前878年）（公元前899年-前878年）（公元前899年-前878年）"，同一区间标注出现三次。037卫康叔世家多个事件出现双重标注。

识别方法: 用正则检查时间字段中是否存在完全相同的标注片段连续出现两次以上。与模式28（时间字段重复标注）本质相同，但此处强调区间格式（含"—"或"-"）的重复。修正方法：保留一份，删除重复。

35. 脚本修正指令残留

问题: 前轮反思中的修正指令文字（如'概览表补充'、'统一为'）被错误地保留在时间字段中，而非仅执行修正操作。apply_fixes脚本在替换时可能将指令文字也写入了字段。

典型案例: 005-128时间字段出现'%三十三年% 统一为 统一为（公元前273年）'，其中'统一为'是指令残留。005-094/095/096时间字段出现"概览表补充"文字，为apply脚本将修正指令当数据写入。

识别方法: 检查时间字段中是否包含中文动词短语（"统一为""概览表补充""修正为""改为"等）。这些文字不属于标准年份标注格式，是脚本处理的副作用。修正方法：删除指令文字，保留正确的年份标注。

审查检查清单

对每一章事件索引进行review时，按以下顺序：

1. **验证锚点**：先检查章节中标注为（公元前XXX年）的精确纪年是否正确

2. **查年表**：逐事件在年表中查找，能找到的升级为精确纪年
3. **换算reign-year**：有纪年标记的事件，用基准年表换算
4. **检查"N年"归属**：确认"二年"、"七年"等指的是哪位君主的纪年
5. **检查单锚点塌缩**：多个事件共享同一推断年份 → 逐个核实
6. **检查前元/中元/后元**：涉及文帝/景帝的章节必查
7. **检查追叙段落**：传记开头"未起时"段落的事件需独立定年
8. **检查同名混淆**：齐桓公（小白vs午）、"二年"（汉二年vs惠帝二年）等
9. **检查死亡事件**：崩/薨/卒/死事件应取去世年，非生卒中点
10. **跨章节比对**：同一事件在不同章节的年份应一致，以本纪/年表为准
11. **世代合理性**：世家章节检查世代间隔是否在25—30年/代范围内
12. **确定性升级**：推断标注 `[约]` 是否可升级为精确标注 `()`

130章review统计

类别	数量
年代推断条目总数	2018 条
年表明确记载	~800+ 事件
推算标注	~1000+ 事件
修正错误	~400+ 事件
新增标注（071-080原无标注）	161 事件

参考文件与输出产物

输入参考

文件	用途
`kg/chronology/data/中国历史大事年表.md`	交叉验证的ground truth，684个年份条目
`kg/chronology/data/史记编年表.md`	《史记》编年大纲（上古→西汉），本工序的汇总输出，反向用作整体校验
`kg/entities/data/person_lifespans.json`	人物生卒年（外部来源），用于无纪年事件的近似定位

核心数据输出

文件	说明
`kg/events/data/NNN_*_事件索引.md`	130章事件索引，每条事件含公元纪年标注
`kg/events/事件时间索引.md`	全部3185事件的时间索引（Markdown汇总格式）
`kg/chronology/data/史记编年表.md`	编年大纲：将事件索引按年代重组为传统年表格式

可视化输出（→ 09 应用层）

文件	说明
`docs/entities/event.html`	事件时间索引网页（按历史分期分组，含搜索筛选）

文件	说明
`docs/entities/timeline.html` + `docs/css/timeline.css`	编年索引：每年显示十表诸侯纪年 + 文本引用 + 事件列表
`app/metro/`	史记事件地铁图（130章×3185站，横轴为公元纪年）
`kg/entities/scripts/build_entity_index.py`	生成event.html的脚本（同时生成全部实体索引）
`kg/events/scripts/write_inferred_years.py`	批量推断年代脚本
`kg/events/scripts/fix_undated_known_events.py`	修正已知事件年份
`kg/events/scripts/batch_fix_collapsed_dates.py`	批量修正年代推断位置

编年索引 timeline.html 生成流程

```
python kg/events/scripts/build_year_map.py
```

完整四阶段管线：

1. **Phase 1 — 解析十表 → reign_periods + year_state_map**
   - `parse_table_014()`：十二诸侯年表（014），841–425 BCE，14国君主 × 417行
   - `parse_table_015()`：六国年表（015），475–145 BCE，8国君主 × 331行
   - `parse_table_022()`：将相名臣年表（022），206 BCE–20 CE，帝号+年号
   - 核心解析器 `_parse_table_by_rows()`：行号计数法（r1=起始年，逐行-1）
   - 输出 `kg/chronology/data/reign_periods.json`（~382君主）
   - 输出 `kg/chronology/data/year_state_map.json`（635年，每年→所有在位君主）

2. **Phase 2 — 年份消歧 → year_ce_map**
   - 扫描130章 tagged.md 中 〖%X年〗 标记
   - 用 reign_periods 消歧为公元纪年（Mapped: 2127，Unresolved: 134）
   - 输出 `kg/chronology/data/year_ce_map.json`
3. **Phase 3 — 事件索引年份 → event_year_map**
   - `load_event_index_years()`：解析130章事件索引 `kg/events/data/`
   - 提取时间字段的公元纪年（精确/推算/近似/区间）
   - 输出：3077个带纪年的事件
4. **Phase 4 — 生成 timeline.html**
   - 合并 year_ce_map + event_year_map（文本引用 + 事件）
   - 用 year_state_map 为每年显示所有诸侯国君主纪年（>4国则折叠展示）
   - 总计：835个年份，1956次文本引用，3077个事件

解析器设计要点：
- 十表无 `| 公元前` 列，用**行号计数法**（不是从单元格读年份）
- r1 = 起始BCE年，每行 -1 年
- 表格触发：检测含 ≥3 个诸侯国名的 `|` 行
- 新君主识别：`元年` 或 `RulerNameX年` 格式
- 续位格式：纯数字（year_num 续计）或空格（inherit）

event.html 生成流程

事件时间索引页面（`docs/entities/event.html`）由 `kg/entities/scripts/build_entity_index.py` 生成：

```
python kg/entities/scripts/build_entity_index.py
```

数据流：
1. 读取 `kg/events/data/*_事件索引.md`（130个文件）
2. 解析每个文件的概览表格，提取事件名、类型、时间、人物、地点、段落位置
3. 从时间字段提取公元年（精确纪年 `（公元前XXX年）` 和推断纪年 `[公元前XXX年]` / `[约公元前XXX年]`）
4. 为无纪年事件推断近似日期（人物生卒年交集 → 章节内插值 → 章节时代兜底）
5. 按历史分期分组，生成HTML页面

页面功能:
- 按历史分期（五帝→夏→商→西周→春秋→战国→秦→楚汉→西汉各帝）分组展示
- 每个事件显示：事件编号、事件名（可点击跳转原文）、时间标签、类型标签、人物标签、地点标签
- 精确纪年（深色标签）与推断纪年（浅色斜体标签）视觉区分
- 支持文本搜索和事件类型筛选
- 折叠/展开各历史分期

SKILL 04d: 事件年代审查管线

> 基于Agent反思的逐章年代标注审查与迭代修正工作流，适用于大规模历史事件索引的质量保障。

一、问题定义

1.1 为什么需要审查管线

事件索引的年代标注来自自动推断脚本（`write_inferred_years.py`），存在系统性错误：

错误类型	典型偏差	130章review实证
单锚点塌缩	60+事件共享同一年份	001五帝、046田敬仲完等
锚点本身计算错误	216-368年	046田和代齐偏差216年
前元/中元/后元混淆	7-13年	011孝景本纪28处修正
年号/帝王混淆	12-300年	053萧何卒、046齐桓公午
追叙事件误用当前时间	20+年	053萧何为沛吏掾

自动检测只能发现格式和一致性错误，**语义错误**（如"二年"指惠帝二年还是汉二年）必须由AI反思判断。

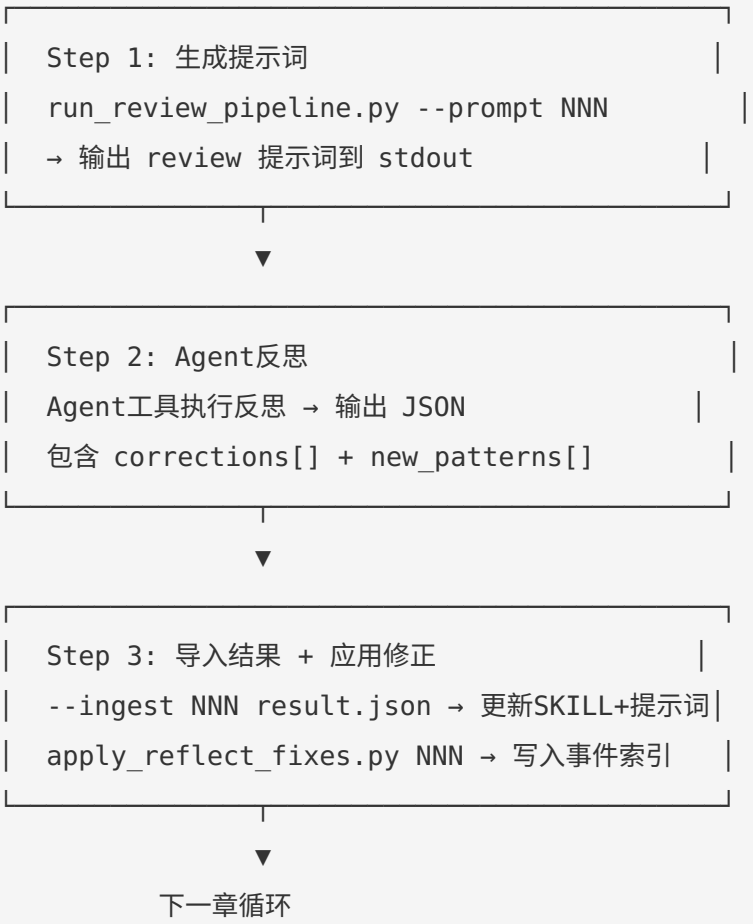
1.2 管线设计原则

1. **逐章审查**：每章独立反思，避免上下文过长
2. **知识累积**：每章发现的新错误模式写入SKILL，后续章节自动获得
3. **人机协作**：Agent反思 + 脚本自动应用 + 人工抽检

4. **可恢复**：管线状态持久化，中断后可从断点继续

二、管线架构

2.1 三步循环



2.2 关键脚本

脚本	用途	路径
`run_review_pipeline.py`	管线主控（--prompt/--ingest/--report）	`kg/events/scripts/`
`generate_review_prompts.py`	批量生成130章提示词	

脚本	用途	路径
		`kg/events/scripts/`
`apply_reflect_fixes.py`	将修正写入事件索引	`kg/events/scripts/`

2.3 数据流

```
事件索引(.md)
    ↓ parse
review提示词(.md)  ←── SKILL_04c_事件年代推断.md（累积的错误模式+推理逻辑）
    ↓ Agent反思
reflect结果(.json)  →  corrections[] + new_patterns[]
    ↓ --ingest
    ├── SKILL_04c_事件年代推断.md  ← 追加new_patterns
    ├── pipeline_state.json    ← 更新进度
    └── 下一章提示词(.md)      ← 包含新学到的模式
    ↓ apply_reflect_fixes.py
事件索引(.md)  ← 概览表+详情表年份同步更新
```

三、Step 1: 生成提示词

3.1 命令

```
python kg/events/scripts/run_review_pipeline.py --prompt NNN
```

输出到 stdout，包含：
- 章节元信息（章名、时代分类）
- 该章事件索引全文
- 适用的错误模式（从SKILL中筛选该时代相关的）

- 年表中该时代的关键条目
- 输出格式要求（JSON schema）

3.2 提示词也可预生成

```
python kg/events/scripts/generate_review_prompts.py          # 全部130章
python kg/events/scripts/generate_review_prompts.py 001-012  # 指定范围
```

输出到 `kg/events/prompts/review_NNN.md`。

四、Step 2: Agent反思

4.1 反思任务

Agent收到提示词后，对每个事件执行：

1. **验证锚点**：查年表确认精确纪年是否正确
2. **检查reign-year换算**：纪年标记 → 基准年 → 公式换算
3. **识别"N年"归属**：从上下文判断指哪位君主
4. **检查单锚点塌缩**：多个事件共享同一推断年份
5. **检查追叙段落**：传记开头的早年经历
6. **确定性升级**：`[约公元前XXX年]` → `（公元前XXX年）`

4.2 输出JSON格式

```
{
  "chapter_id": "046",
  "corrections": [
    {
      "event_id": "046-008",
      "original_time": "[约公元前602年]",
      "suggested_year": "（公元前386年）",
```

```
      "reason": "年表前386年明确记载'田和列为诸侯'",
      "error_type": "锚点计算错误",
      "confidence": "high"
    }
  ],
  "new_patterns": [
    {
      "name": "世代数≠年数",
      "description": "世家类章节不能把世代数直接换算成年数",
      "example": "046-008: 从陈完到田和约十代286年，原计算为53年",
      "detection": "检查世代间隔是否在25-30年/代范围内"
    }
  ]
}
```

4.3 字段说明

corrections[]:

- `event_id`：事件编号（如 `046-008`）
- `original_time`：原时间标注
- `suggested_year`：建议修正值，使用三级标注格式
- `reason`：修正理由（必须引用具体依据）
- `error_type`：对应SKILL中的错误模式名
- `confidence`：`high`（年表确认）/ `medium`（reign-year换算）/ `low`（推断）

new_patterns[]:

- 本章发现的、SKILL中尚未记录的新错误模式
- 会被 `--ingest` 追加到 `SKILL_04c_事件年代推断.md`

五、Step 3: 导入与应用

5.1 导入反思结果

```
python kg/events/scripts/run_review_pipeline.py --ingest NNN /tmp/
reflect_NNN.json
```

执行:
1. 解析JSON，验证格式
2. 将 `new_patterns` 追加到 `SKILL_04c_事件年代推断.md`
3. 更新 `pipeline_state.json` （进度、累积统计）
4. 生成下一章提示词（自动包含新学到的模式）

5.2 应用修正到事件索引

```
python kg/events/scripts/apply_reflect_fixes.py NNN
```

执行:
1. 读取 `kg/events/reports/reflect_NNN.json`
2. 逐条匹配事件索引中的概览表和详情表
3. 同步更新概览表时间列 + 详情的时间字段
4. 保留原有的纪年标记（ `%N年%` ），只替换公元年部分
5. 输出修改统计

5.3 修正应用规则

```
原时间字段: "%秦昭王十三年% [约公元前307年]"
suggested_year: "（公元前294年）"
结果: "%秦昭王十三年% （公元前294年）"
```

- 纪年标记（ `%...%` ）保留
- 旧的公元年标注被替换
- 新标注使用 `suggested_year` 的原始格式（含括号类型）

六、管线状态管理

6.1 状态文件

`kg/events/reports/pipeline_state.json`：

```
{
  "last_chapter": "046",
  "completed_chapters": ["001", "109"],
  "total_corrections": 56,
  "total_new_patterns": 26,
  "accumulated_patterns": ["单锚点塌缩", "锚点计算错误", ...]
}
```

6.2 断点恢复

```
python kg/events/scripts/run_review_pipeline.py --resume
```

从 `last_chapter` 的下一章继续。

6.3 查看报告

```
python kg/events/scripts/run_review_pipeline.py --report
```

输出各章审查统计汇总。

七、自动检测模式（无需Agent）

管线也支持纯脚本自动检测（快速，无API调用）：

```
python kg/events/scripts/run_review_pipeline.py 001-130
```

自动检测项:
- 概览表与详情表年份不一致
- 同一年份被多个事件共享（塌缩嫌疑）
- 时间字段格式不规范
- 纪年标记与公元年不匹配
- 事件时间超出合理范围

八、实证统计

130章review结果

指标	数量
年代推断条目总数	2,018
修正错误	~400+
新增错误模式	26种
新增推理逻辑	12种
新增标注（原无标注）	161

已完成的Agent反思章节

章节	事件数	修正数	关键发现
001 五帝本纪	66	44	单锚点塌缩到前2290年
109 李将军列传	16	12	详情塌缩到前166年

九、参考文件

文件	用途
`SKILL_04c_事件年代推断.md`	累积的错误模式和推理逻辑（Agent反思的知识库）
`kg/events/data/NNN_*_事件索引.md`	130章事件索引（审查对象）
`kg/events/prompts/review_NNN.md`	130章预生成的审查提示词
`kg/events/reports/review_NNN.json`	自动检测结果
`kg/events/reports/reflect_NNN.json`	Agent反思结果
`kg/events/reports/pipeline_state.json`	管线状态
`kg/chronology/data/中国历史大事年表.md`	交叉验证的ground truth

SKILL 04e: 古籍年份消歧 — 从在位纪年到公元绝对年

> 从《史记》130篇年份消歧实践中提炼，适用于以在位纪年记述历史的古籍文本。

一、问题定义

史记正文中的年份均为**在位纪年**（如 `%三年%` = 某位君主的第三年），不含绝对时间信息。年份消歧需解决两个子问题：

1. **归属**：确定每个年份属于哪位君主
2. **换算**：将在位纪年转换为公元纪年

脚本：`build_year_map.py`
输出：`reign_periods.json` + `year_ce_map.json` + `docs/entities/timeline.html`

二、数据来源

2.1 年表章节（841 BCE 之后）

项目中三个年表章节包含完整的公元↔在位年映射：

年表	覆盖范围	格式特点
014 十二诸侯年表	841–478 BCE	14列（周+13诸侯国），每行=1公元年
015 六国年表	476–207 BCE	8列（周+秦+六国），每行=1公元年

年表	覆盖范围	格式特点
022 将相年表	206–20 BCE	纪年列含帝号+年号，每行=1公元年

"元年"条目标志新君即位，后续行递增计数。

2.2 夏商周断代工程（841 BCE 之前）

841 BCE 之前的年代采用夏商周断代工程（2000年发布）官方年表：

- **西周王室**（周文王～周厉王）：11位周王的在位年
- **商后期**（武丁～帝辛）：依据断代工程考古+天文校准
- **鲁国早期**（伯禽～献公）：由鲁真公（855 BCE）向前推算

三、三阶段处理管线

Phase 1：提取在位年表

从年表章节解析 `reign_periods.json`：

```
{
  "rulers": {
    "秦孝公": {"state": "秦", "start_bce": 361, "end_bce": 338},
    "周宣王": {"state": "周", "start_bce": 827, "end_bce": 782}
  },
  "eras": {
    "建元": {"ruler": "孝武", "state": "汉", "start_bce": 140,
"end_bce": 134}
  },
  "aliases": {"秦惠王": "秦惠文王", "高祖": "高皇帝"}
}
```

共提取 **380位君主**（含断代工程补充的 pre-841 BCE 君主）、**32个汉代年号**。

公元换算公式：`bce_year = start_bce - year_num + 1`

Phase 2：年份消歧

扫描所有 `*.tagged.md`（排除年表章节自身），对每个 `%X年%` 实体执行消歧。

预过滤

跳过非历法年份（时长/年龄）：

```
立%十二年%卒      → 跳过（前文含"立"、"居"、"凡"、"共"、"在"）
%十馀年%          → 跳过（含"馀"）
```

消歧策略（按优先级）

优先级	策略	说明
1	**年号直接映射**	`%建元六年%` → 汉武帝建元六年 → 公元前135年
2	**近距离君主**	60字符范围内最近的 `@person@` 或 `$title$` 是已知君主
3	**序列延续**	本段落前面已有已解析年份 → 沿用同一君主
4	**小节标题**	`## 秦王政时期` 标题含君主名 → 该节内年份归属该君主
5	**近距离原始名**	找到人名但无法映射到已知君主 → 以"君主+年份"为索引键

优先本章节所属国：在秦本纪中，优先匹配秦国君主，叙事中提及的外国君主不参与竞争。

国名前缀推断：如 `宣公` → 在秦本纪中推断为 `秦宣公`。

结果校验

计算出的公元年份必须落在该君主已知在位范围内（容差5年），否则仅记录 `ruler_key`，不生成 `ce_year`。

越界恢复机制

当年份超出已匹配君主的在位范围时（多因连续归因跨越君主更替），自动在同国君主序列中查找正确归属，恢复公元纪年映射。

Phase 3：编年索引页

`docs/entities/timeline.html` 包含两部分：

1. **公元纪年索引**：按世纪分组，每条显示公元年份 + 所有并行在位君主的纪年别名（如"秦孝公元年、周显王八年"），附章节段落链接。
2. **先秦早期（年代不详）**：以"君主+年份"为索引键，用于五帝、夏、殷、早周等年表未覆盖的年代。

四、数据结构

4.1 输出：year_ce_map.json

```
{
  "005": {
    "7": {
      "元年": {"ce_year": -765, "ruler": "秦文公", "method":
"nearby_ruler"}
    },
    "7.2": {
      "三年": {"ce_year": -763, "ruler": "秦文公", "method":
"sequential"}
    }
  },
  "001": {
    "13.8": {
      "三年": {"ruler": "舜", "method": "raw_nearby", "ruler_key":
"舜三年"}
```

```
    }
  }
}
```

键层次：`chapter_id → para_id → 表面年份文本`。值字段：

字段	含义	是否必有
`ruler`	归属君主	是
`method`	消歧方法	是
`ce_year`	公元年（负数=BCE）	仅有完整映射时
`ruler_key`	索引键（如"舜三年"）	仅无公元年时

4.2 渲染效果

```
# render_shiji_html.py 中的年份渲染逻辑
if css_class == 'time' and para_id in year_map:
    entry = year_map[para_id].get(entity_text)
    if entry:
        if 'ce_year' in entry:
            tooltip = f"公元前{abs(entry['ce_year'])}年
 ({entry['ruler']}{entity_text}) "
            href = f"timeline.html#{entry['ce_year']}"
        else:
            tooltip = entry['ruler_key']
            href = f"timeline.html#ancient-{entry['ruler_key']}"
```

- **有公元年**：悬停显示 `公元前763年（秦文公三年）`，点击跳转到 timeline.html 对应条目
- **仅有君主+年份**：悬停显示 `舜三年`，点击跳转到先秦早期对应条目
- **未映射**：保持原有链接（指向 `time.html`）

五、覆盖统计

指标	数值
数据源	年表章节(014/015/022) + 夏商周断代工程
处理范围	71篇（排除年表自身及无年份标注的篇章）
提取君主数	380位
汉代年号数	32个
映射总数	1,498条
公元年覆盖	前1248年～前91年
未解决率	~7%（上古无年表章节覆盖的早期年份）

六、关键经验

1. **年表章节是年份消歧的核心资源** — 三个年表章节提供了几乎所有841 BCE后年份的基础数据
2. **序列延续是最高频的成功策略** — 史记叙述通常连续记录同一君主的多个年份，前一年份已解析则后续可顺推
3. **本章节所属国优先** — 消歧时本国君主权重高于叙事中提及的外国君主，大幅降低误判率
4. **年号直接映射最精确** — 汉代年号（如建元、元光）与公元年一一对应，无歧义
5. **保守策略优于激进** — 超出在位范围的年份宁可不映射，不做错误猜测
6. **上古年份只做索引，不强求公元换算** — 五帝、夏、殷等时期以"君主+年份"为索引键即可

七、工具链

脚本/文件	位置	功能
`build_year_map.py`	kg/events/scripts/	三阶段年份消歧管线
`generate_all_chapters.py`	项目根	渲染时加载year_ce_map.json
`reign_periods.json`	kg/chronology/data/	380位君主在位年表
`year_ce_map.json`	kg/events/data/	章节段落级年份→公元年映射
`timeline.html`	docs/entities/	编年索引页（公元年+先秦早期）

管线集成顺序：

```
build_year_map.py  →  reign_periods.json + year_ce_map.json + timeline.html
generate_all_chapters.py  →  渲染所有章节HTML（读取上述元数据）
```

> 另见：SKILL_03b_实体消歧.md — 人名消歧的完整方法论

SKILL 04f: 动词标注

编号: SKILL_04f
父技能: SKILL_04_事件构建
版本: v3.2
创建日期: 2026-03-19
最后更新: 2026-03-19

技能定位

动词标注是**事件构建与关系构建的桥梁**，专门处理历史叙事中的核心动作动词。动词既是事件的核心要素，又是关系的连接纽带（主语-**动词**-宾语）。

在管线中的位置:

```
SKILL_04_事件构建
├── SKILL_04a_事件识别
├── SKILL_04b_十表事件处理
├── SKILL_04c_事件年代推断
├── SKILL_04d_事件年代审查
├── SKILL_04e_年份消歧
└── SKILL_04f_动词标注 ← 当前（事件到关系的桥梁）
    ↓
SKILL_05_关系构建
├── SKILL_05a_事件关系发现
├── SKILL_05b_实体关系构建
└── ...
```

为什么放在SKILL_04最后?
1. **依赖实体标注**：动词标注需要先完成名词实体标注（SKILL_03），才能识别主语和宾语
2. **支撑事件提取**：标注的动词直接用于识别军事事件、刑罚事件等（SKILL_04a）

3. **铺垫关系抽取**：标注的动词是建立三元组关系的核心（SKILL_05a/05b）
4. **桥梁作用**：动词是从事件到关系的自然过渡

技能概述

目标

为《史记》文本中的动词进行语义标注，将历史叙事中的核心行为动作标记为结构化实体，用于：
- **事件提取**：自动识别军事行动、刑罚事件等历史事件
- **关系挖掘**：建立主语-动词-宾语的三元组关系
- **语义检索**：按动词类型检索特定事件
- **统计分析**：量化分析军事冲突频率、刑罚执行模式等

核心价值

1. **事件核心要素**：动词是历史事件的核心，标注动词等于标注事件
2. **关系桥梁**：动词连接主体与客体，是关系抽取的关键
3. **语义区分**：区分动词与名词（〖◉杀〗vs〖[斩首〗），语义更精确
4. **可计算性**：结构化的动词标注便于量化分析和模式挖掘

标注体系

1. 符号系统

外层符号：`〖〗` (U+27E6/27E7 数学双方括号)

与名词实体的区分：
维度	名词实体	动词实体
外层符号	`〔〕` (白色龟甲括号)	`〖〗` (数学双方括号)
TYPE符号	文本符号（@=;%等）	几何符号（◈◉○◇）
语义功能	事件参与者（主/宾语）	事件核心（谓语）
用途	实体识别	关系识别

2. 动词类型

类型	符号	格式	示例	核心词数	状态
军事动词	◈	〖◈verb〗	〖◈伐〗 〖◈攻〗	17个	✅ 已实施
刑罚动词	◉	〖◉verb〗	〖◉杀〗 〖◉诛〗	15个	✅ 已实施
政治动词	○	〖○verb〗	〖○立〗 〖○封〗	待扩展	🔄 预留
经济动词	◇	〖◇verb〗	〖◇赐〗 〖◇贡〗	待扩展	🔄 预留

3. 消歧语法

格式: 〖TYPE动词|消歧说明〗

示例:

```
〖◈伐|征伐〗        # 军事征伐
〖◈走|逃亡〗        # 逃走（区分于行走）
〖◉死|处死〗        # 判处死刑（区分于自然死亡）
〖◈取|攻取〗        # 军事攻取（区分于一般之取）
```

核心词表

军事动词（17个）

进攻类（5个）：

```
伐、攻、击、袭、侵
```

交战类（7个）：

战、破、败、灭、围、下、拔

机动类（5个）：

救、取、定、降、走

刑罚动词（15个）

处决类（7个）：

杀、诛、斩、弒、族、戮、刺

处罚类（5个）：

废、囚、执、笞、刑

其他类（3个）：

反、亡、死

关键区分：动词 vs 制度名词

类别	标注格式	特征	示例
刑罚动词	`〖◉verb〗`	单字动词，表示动作	`〖◉杀〗` `〖◉诛〗`
刑罚制度名词	`〖[legal〗`	2字+名词短语	`〖[斩首〗` `〖[腰斩〗`

示例对比:

```
〖◉杀〗〖;令尹〗〖@子西〗，〖[斩首〗八万
  ↑动词                    ↑制度名词

〖◉斩〗〖@颜良〗，〖[弃市〗
  ↑动词          ↑制度名词
```

标注规则

规则1：识别原则

✅ **应该标注**：
- 军事行动：伐/攻/击/战/破/败/灭等
- 刑罚处决：杀/诛/斩/弒/族等
- 明确的及物动词，有主语和宾语
- 可用于事件提取和关系抽取的关键动词

❌ **不应标注**：
- 日常动词：行/坐/立/言/语
- 助动词：能/可/是/为
- 趋向动词：来/去/上/下（除非军事机动）
- 一般性动作：见/闻/知/思

规则2：消歧原则

何时需要消歧：
1. 同形异义（如：走=逃亡/行走，取=攻取/获取）
2. 跨类歧义（如：亡=军事逃亡/逃避刑罚）
3. 与名词混淆（如：围=动词包围/名词包围圈）

规则3：标注边界

- **单字优先**：〖◈伐〗 〖◉杀〗
- **复合动词拆分**（推荐）：〖◈击〗〖◈破〗 而非 〖◈击破〗

规则4：语法约束

✅ **正确**：`〖◈伐〗〖'随〗` `〖◈击|攻击〗〖'楚〗`

❌ **错误**：`〖〖◈伐〗〗` `〖◈〗` `〖◈伐〗` `〖@伐〗`

工作流程

Phase 1: 文本分析

1. 阅读章节内容，理解历史叙事
2. 识别关键事件（战争/刑罚/政治/经济）
3. 标记候选动词
4. 确认上下文

Phase 2: 动词分类

```
是否涉及战争、军事行动?
  ├─ 是 → 军事动词 〖◈〗
  └─ 否 → 继续

是否涉及处决、刑罚?
  ├─ 是 → 刑罚动词 〖◉〗
  └─ 否 → 继续

是刑罚制度名称?
  └─ 是 → 使用 〖[legal〗（非动词）
```

Phase 3: 标注执行

```
原文：楚伐随。杀令尹子西，斩首八万。
标注：〖'楚〗〖◈伐〗〖'随〗。〖◉杀〗〖;令尹〗〖@子西〗，〖[斩首〗八万。
```

```
结构分析:
- 三元组1: (楚, 伐, 随) → 军事关系
- 三元组2: (?, 杀, 子西) → 刑罚关系
```

Phase 4: 质量检查

```
# 运行验证脚本
python kg/entities/scripts/validate_verb_tagging.py --chapter 040

# 检查输出
# - 未分类的动词?
# - 格式错误?
# - 遗漏的动词?
```

Agent提示词

系统提示（System Prompt）

```
你是《史记》动词标注专家。你的任务是识别、分类和标注历史叙事中的核心动词，为事件提取和关系挖掘提供基础。

**核心能力**:
1. 识别军事动词（伐/攻/击/战/破/败/灭等17个）
2. 识别刑罚动词（杀/诛/斩/弒/族等15个）
3. 区分动词与名词（〖◉杀〗 vs 〖[斩首〗）
4. 判断是否需要消歧（〖◈走|逃亡〗）
5. 识别三元组结构（主语-动词-宾语）

**标注格式**:
- 军事: `〖◈verb〗` 或 `〖◈verb|说明〗`
- 刑罚: `〖◉verb〗` 或 `〖◉verb|说明〗`
- 外层: 〖〗(U+27E6/27E7)
```

工作原则:
1. 只标注军事和刑罚动词（政治/经济预留）
2. 单字动词优先
3. 有歧义必须消歧
4. 刑罚制度名词用 〖[legal〗
5. 日常动词不标注
6. 关注动词的主语和宾语（为关系抽取做准备）

任务提示（Task Prompt）

请为以下段落标注动词:

章节: {chapter_name}
段落: [{para_id}]
原文: {original_text}

要求:
1. 识别所有军事动词（17个核心词）
2. 识别所有刑罚动词（15个核心词）
3. 区分动词与制度名词
4. 对歧义词添加消歧说明
5. 标注后分析三元组结构

示例:
原文：楚伐随。杀令尹子西，斩首八万。
标注：〖'楚〗〖◈伐〗〖'随〗。〖◉杀〗〖;令尹〗〖@子西〗，〖[斩首〗八万。

三元组:
- (楚, 〖◈伐〗, 随)
- (?, 〖◉杀〗, 子西)

请输出标注后的文本和三元组分析:

反思提示（Reflection Prompt）

检查你的动词标注，回答清单：

完整性：
- [ ] 所有军事行动已标注？
- [ ] 所有刑罚事件已标注？
- [ ] 有无遗漏？

准确性：
- [ ] 动词分类正确？
- [ ] 动词/名词区分正确？
- [ ] 消歧说明准确？

一致性：
- [ ] 同一动词标注一致？
- [ ] 格式符合规范？
- [ ] 无格式错误？

关系识别：
- [ ] 每个动词的主语清晰？
- [ ] 每个动词的宾语清晰？
- [ ] 可以形成有效的三元组？

边界情况：
- [ ] 无日常动词误标？
- [ ] 无制度名词误用动词格式？
- [ ] 无不必要的消歧？

如有问题，列出修正方案。

核心脚本

1. 查询脚本

文件: `kg/entities/scripts/query_verbs_by_type.py`

用法:

```
# 统计军事动词
python kg/entities/scripts/query_verbs_by_type.py --type military

# 统计刑罚动词
python kg/entities/scripts/query_verbs_by_type.py --type penalty

# 分析指定章节
python kg/entities/scripts/query_verbs_by_type.py --chapter 040

# 生成完整报告
python kg/entities/scripts/query_verbs_by_type.py --all --report
```

2. 验证脚本

文件: `kg/entities/scripts/validate_verb_tagging.py`

用法:

```
# 验证所有章节
python kg/entities/scripts/validate_verb_tagging.py

# 验证指定章节
python kg/entities/scripts/validate_verb_tagging.py --chapter 040

# 严格模式
python kg/entities/scripts/validate_verb_tagging.py --strict
```

```
# 生成报告
python kg/entities/scripts/validate_verb_tagging.py --report
report.txt
```

3. 迁移脚本

文件: `kg/entities/scripts/migrate_verb_tags.py`

用法:

```
# 预览迁移
python kg/entities/scripts/migrate_verb_tags.py --chapter 040 --
dry-run

# 执行迁移
python kg/entities/scripts/migrate_verb_tags.py --chapter 040 --
backup

# 批量迁移
python kg/entities/scripts/migrate_verb_tags.py --all --backup

# 生成报告
python kg/entities/scripts/migrate_verb_tags.py --all --report
report.md
```

常见错误及修正

错误1：混淆动词与名词

❌ `〖◉斩首〗八万`
✅ `〖[斩首]〗八万`

说明: 「斩首」是刑罚制度名词，应用 `〖[legal]〗`

错误2：过度标注日常动词

❌ `〖@项羽〗⟦行⟧至〖=鸿门〗`
✅ `〖@项羽〗行至〖=鸿门〗`

说明：「行」是日常动词，不标注

错误3：缺少必要消歧

❌ `〖@子常〗⟦◈走⟧奔〖'郑〗`
✅ `〖@子常〗⟦◈走|逃亡⟧奔〖'郑〗`

说明：「走」有歧义（逃亡/行走），需消歧

错误4：错误的TYPE符号

❌ `⟦@伐⟧〖'随〗`
✅ `⟦◈伐⟧〖'随〗`

说明：军事动词用 `◈`，非人名符号 `@`

错误5：错误的外层符号

❌ `〖◈伐〗〖'随〗`
✅ `⟦◈伐⟧〖'随〗`

说明：动词用 `⟦⟧`，名词用 `〖〗`

性能指标

质量目标

指标	目标值
准确率	≥95%
召回率	≥90%

指标	目标值
一致性	≥98%
格式正确率	100%
三元组可提取率	≥85%

当前统计（v3.2）

- **已标注章节**：130/130 (100%)
- **军事动词**：4,085 次
- **刑罚动词**：2,172 次
- **总计动词**：6,257 次
- **平均每章**：48.1 次

参考文档

规范文档

1. **动词标注规范** - 完整的语法规范和分类体系
2. **动词渲染规则** - HTML渲染和CSS样式
3. **标注格式规范** - 整体标注格式

数据文档

1. **动词分类体系** - 详细词表和扩展规划
2. **动词标注迁移报告** - v3.1迁移记录
3. **实体标注统计报告** - 包含动词的完整统计

相关技能

前置技能:
- **SKILL_03a_实体标注** - 名词实体标注（提供主语/宾语）
- **SKILL_03b_实体消歧** - 消歧语法
- **SKILL_04a_事件识别** - 事件识别

后续技能:
- **SKILL_05a_事件关系发现** - 基于动词的事件关系
- **SKILL_05b_实体关系构建** - 基于动词的实体关系

迭代历史

版本	日期	变更	成果
v1.0	2026-03-14	初步使用 `〖[legal〗` 标注动词	误将动词当作制度名词
v2.0	2026-03-18	发现问题，设计动词标注语法	创建verb_taxonomy.md
v3.0	2026-03-18	实施迁移，完成130章	6,257次动词标注
v3.1	2026-03-19	建立skill，创建agent提示词	完整的标注工作流
v3.2	2026-03-19	重新定位为SKILL_04f	明确事件-关系桥梁作用

下一步计划

短期（已完成）

- ✅ 建立军事/刑罚动词标注体系
- ✅ 完成130章迁移

- ✅ 创建验证和查询工具
- ✅ 更新HTML渲染规则

中期（规划中）

- 🔄 扩展政治动词（封/立/废/拜/召等）
- 🔄 扩展经济动词（赐/贡/贿/赂等）
- 🔄 人工审查805个未分类词
- 🔄 建立动词-事件映射规则
- 🔄 基于动词自动提取三元组

长期（愿景）

- 🔮 动词驱动的事件抽取
- 🔮 基于动词的关系挖掘
- 🔮 历史模式的量化分析
- 🔮 可视化动词网络

技能维护者: Claude AI Agent
最后更新: 2026-03-19
状态: ✅ 正式发布
适用范围: 史记知识库全部130章
定位: 事件构建与关系构建的桥梁

SKILL 05: 关系构建 — 从事件与实体到关系网络

> 发现事件间的因果/时序/互见关系，抽取实体间的结构化三元组（SPO）。

一、产出统计

知识类型	数量
事件关系	7,652条（自动5,126 + LLM 2,525）
关系类型	9种（延续/因果/跨章因果/包含/对立/互见/共人/共地/同期）
跨线换乘	1,876条（互见294/共人867/共地542/同期173）

二、子工序

编号	文件	内容
05a	SKILL_05a_事件关系发现.md	9种事件关系（自动+LLM混合发现）
05b	SKILL_05b_实体关系构建.md	地点/制度/事物关系（非人物实体关系）+ SPO三元组
05c	SKILL_05c_人物关系构建.md	人物共现→家族推理→政治社会关系（三层递进）

编号	文件	内容
	SKILL_05c1_人物共现关系.md	段落级共现抽取、PMI计算、候选关系对
	SKILL_05c2_家族关系推理.md	帝王/邦国/大族家谱构建、传递推理、验证
	SKILL_05c3_政治社会关系.md	君臣/师徒/敌对等关系抽取
05d	SKILL_05d_事实发现.md	原子事实三元组抽取（~10万规模，独立于事件和关系）
05e	SKILL_05e_战争事件识别.md	战争事件专项提取/多源融合/事实关联

SKILL 05a: 事件关系发现 — 从孤立事件到关系网络

> 基于《史记》130篇3,185个事件、7,652条关系的实践提炼。整合全项目关系发现规则的唯一权威文档。
>
> 合并来源：`SKILL_04c_事件年代推断.md`（跨章定年）、`SKILL_04a_事件识别.md`（事件链）、`SKILL_05b_实体关系构建.md（实体三元组）。

一、关系类型体系

1.1 八种关系类型

类型	方向	来源	数量	说明
sequel	A→B	LLM	1,623	B是A的直接后续发展
causal	A→B	LLM	407	A是B的直接原因（章内，A不发生则B不发生）
cross_causal	A→B	LLM	352	跨章因果：含direct_cause/prerequisite/background/trigger/precedent五种子类型
part_of	子→父	LLM	107	A是B的子事件/组成环节
opposition	A↔B	LLM	50	对立双方的行动
cross_ref	A↔B	自动	294	不同章节记述同一事件

类型	方向	来源	数量	说明
co_person	A↔B	自动	969	跨章事件共享>=2个人物
co_location	A↔B	自动	705	跨章事件共享地点+>=1人物
concurrent	A↔B	自动	230	同一公元年+共享>=1人物

1.2 核心分工原则

章内语义关系用LLM，跨章结构关系用自动算法，跨章因果关系用LLM二次推理。

- LLM擅长理解叙事因果逻辑（sequel/causal/part_of/opposition + cross_causal）
- 自动方法擅长跨文档实体匹配（cross_ref/co_person/co_location/ concurrent）
- **第二轮LLM推理**：以自动关系的候选对为输入，专门挖掘跨章因果（cross_causal）
- 自动方法产出5,126条（含年代全覆盖后concurrent大幅增加），LLM产出2,188条章内关系 + 352条跨章因果

二、十表特有的关系模式

十表（013-022）共226个事件，参与458条跨章关系。十表作为史记的"纪年骨架"，与纪传体章节形成独特的互补关系网络。

2.1 表-纪互见（Table-Annals Cross-Reference）

定义：同一历史事件同时在年表和本纪/世家/列传中被记述。表提供精确年代框架，纪传提供人物与细节叙事。

典型案例：

表中事件	纪传中事件	关系	互补价值
016-009 项羽大破秦军巨鹿	007-016 巨鹿之战	cross_ref	表给月份，纪给战术细节
016-018 杀项籍天下平	007-061 垓下之围	cross_ref	表给月份，纪给四面楚歌/乌江自刎
015-020 长平之战	043-025/081-017 长平之战	cross_ref	表给年份，世家/列传给白起坑杀赵卒
022-018 吴楚七国反	011-007 七国之乱	cross_ref	表给将相调动，纪给平叛经过
018-007 张良封留侯	055-022 封留侯	cross_ref	侯表给封邑/户数，世家给封侯背景

发现规则:
1. 事件名称相似度>=0.6 且共享>=1人物 → 候选互见
2. 表中事件的精确年代可校验纪传中的推算年代
3. 互见关系置信度：名称完全相同(1.0)+共享2+人物 > 名称近似(0.6-0.9)+共享1人物

2.2 跨表同事件（Cross-Table Same Event）

定义：同一历史事件在多个年表中各自记载，因视角/主题不同，各表记述侧重不同。

核心案例：酎金国除（元鼎五年，前112年）

章节	事件ID	侧重	涉及侯国数
018 高祖功臣侯者年表	018-020	高祖功臣后裔被除	~20家
019 惠景间侯者年表	019-015	惠景间封侯者被除	~7家
020 建元以来侯者年表	020-014	武帝时封侯者被除	~6家
021 王子侯者年表	021-013	王子侯56家被除	56家

章节	事件ID	侧重	涉及侯国数
030 平准书	—	经济政策背景（夺侯充国用）	—

发现规则:
1. 同一公元年 + 同类事件名（如都含"国除"/"酎金"） → 跨表同事件
2. 方向：标注为 cross_ref，附加说明"同一事件不同视角"
3. 这类关系自动算法可能遗漏（因事件名称不完全相同），需补充人工规则

其他跨表同事件:

事件	涉及表	说明
七国之乱（前154年）	017, 019, 022	017记王国除，019记平乱封功臣，022记将相调动
高祖六年大封（前201年）	017, 018, 022	017记封同姓九王，018记封功臣百余人，022记将相任命
推恩令（前127年）	017, 021	017记制度颁布，021记王子侯分封实施
诛诸吕（前180年）	017, 019, 022	017记吕氏王国除，019记功臣封侯，022记将相变动

2.3 制度因果链（Institutional Causal Chain）

定义：政策颁布→实施→后果的多步因果关系，跨越多个表和多年时间。

链条1：分封—削藩—推恩—国除

```
017-002 高祖立同姓九王（前201年）
  → 017-004 诸侯骄逸自大
  → 017-010 晁错上削藩策
  → 017-006 吴楚七国之乱（前154年）
  → 017-019 七国诸侯国除
```

```
   → 017-005 推恩令分封（前127年）
   → 021-006~012 各年王子侯分封（前127–前112年）
   → 021-013 酎金国除56侯（前112年）
```

链条2：功臣封侯—富厚骄溢—国除殆尽

```
018-002 汉兴功臣受封百余人（前201年）
   → 018-003 萧曹绛灌富至四万户
   → 018-004 子孙骄溢淫嬖
   → 018-020 酎金坐罪大量国除（前112年）
   → 018-011 太初百年间见侯仅五
```

链条3：武帝击匈奴—军功封侯—财政困难—酎金夺爵

```
020-003 卫青击匈奴封长平侯（前127年）
   → 020-004 霍去病封冠军侯（前123年）
   → 020-011 漠北大战封侯（前119年）
   → 030-* 平准书：财政空虚
   → 020-014 酎金坐罪国除（前112年）
```

发现规则：
1. 同主题事件按时间排序 → 寻找因果/延续链
2. 政策事件后10-50年内的同类批量事件 → 疑似实施/后果
3. 跨表因果链需人工确认，自动算法难以发现（时间跨度大、名称不相似）

2.4 人物生命线（Person Lifeline）

定义：同一人物在不同表/章中的事件形成传记时间线。

案例：周勃的跨章生命线

时间	事件	章节	类型
前209年	周勃从沛公灭秦	057-001	世家
前206—前202年	定三秦击楚	057-002	世家

时间	事件	章节	类型
前201年	封绛侯	018-009	侯表
前196年	以将军击燕王卢绾	022-008	将相表
前180年	诛诸吕立文帝	019-005/022-011	侯表+将相表
前177年	就国为庶人	057-017	世家
前169年	卒	057-019	世家

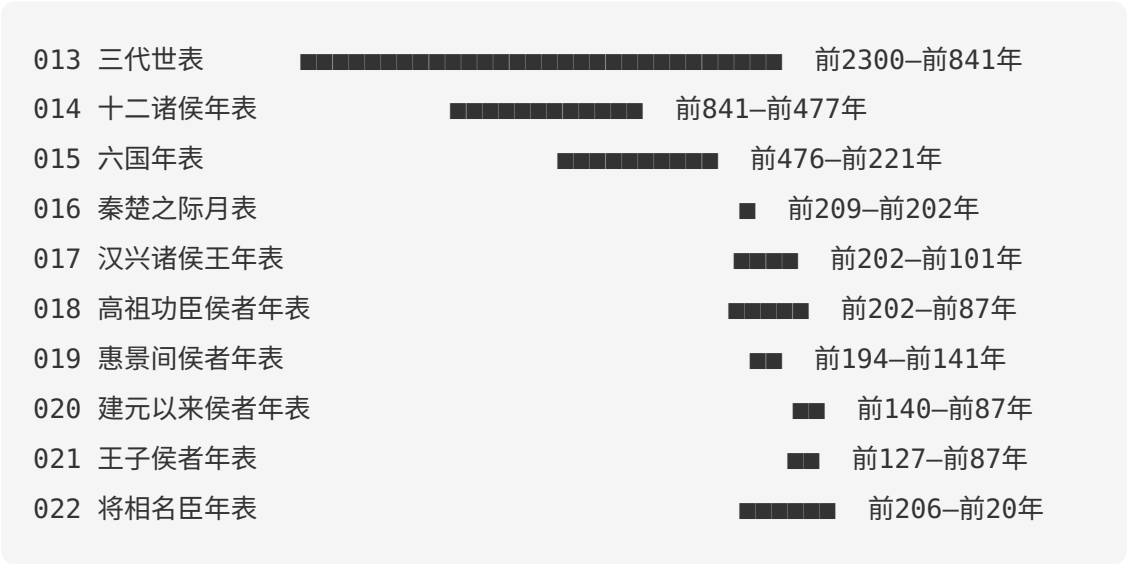
高频跨表人物:

人物	涉及表章	关系数
高祖（刘邦）	016, 017, 018, 022	最密
周勃	018, 019, 022 + 057世家	封侯→诛吕→将相
卫青	020, 022 + 111列传	军功→封侯→大将军
霍去病	020, 022 + 111列传	军功→封侯→骠骑将军
霍光	013, 022 + 068世家	辅政→大将军→身后族灭
张良	018 + 055世家	封留侯→辟谷→国除
萧何	018, 022 + 053世家	封酂侯→相国→后嗣续封
韩信	016, 018 + 092列传	灭赵定齐→封侯→谋反诛杀

发现规则:
1. 构建 人物→事件列表 倒排索引
2. 同一人物跨2+章节出现 → 按时间排序形成生命线
3. co_person关系是自动发现的基础（共享>=2人物）

2.5 时间线贯通（Timeline Continuity）

定义：不同年表覆盖不同时段，拼接后形成连续历史时间线。



关系发现要点：

1. 相邻时段的表之间存在时间重叠区（如015末段与016全段重叠）
2. 重叠区的同年事件 → concurrent关系候选
3. 017-022六表时间高度重叠，同一年可能在多表都有记载

2.6 主题聚合（Thematic Clustering）

定义：围绕同一历史主题，跨章节、跨时段的事件形成主题簇。

主题1：匈奴战争

- 014: 周室东迁与戎狄（背景）
- 015: 赵武灵王胡服骑射
- 020-003/004/011: 卫青霍去病封侯、漠北大战
- 022-023~030: 武帝朝将相调动（马邑→河南→右贤王→祁连→漠北）
- 110 匈奴列传: 匈奴方叙事
- 111 卫将军骠骑列传: 将领方叙事

主题2：削藩体制

- 017: 诸侯王封/废全景
- 019-013: 平七国乱封功臣
- 021: 推恩令实施细节（王子侯分封批次）
- 022-018/020: 将相参与平乱与分国

- 101 袁盎晁错列传: 晁错削藩策
- 118 淮南衡山列传: 淮南王安谋反

主题3：功臣兴衰
- 018: 高祖功臣百余人封侯→太初仅存五
- 019: 惠景间追封+国除
- 022: 将相任免更替
- 053/055/057: 萧何/张良/周勃世家（个案叙事）
- 092: 韩信从封王到诛杀

发现规则:
1. 事件描述关键词聚类（如"匈奴""封侯""国除""削藩"）
2. 同主题事件跨3+章节 → 建立主题簇
3. 主题簇内事件按时间排序 → 形成主题时间线

三、关系发现方法

3.1 自动发现算法（4种，5,126条）

cross_ref（互见）发现 — 294条

不同章节记述**同一事件**。

```
def find_cross_refs(events_by_chapter):
    """跨章互见：名称相似+共享人物"""
    for ch_a, ch_b in combinations(chapters, 2):
        for ev_a in events_by_chapter[ch_a]:
            for ev_b in events_by_chapter[ch_b]:
                sim = name_similarity(ev_a.name, ev_b.name)
                shared = shared_persons(ev_a, ev_b)
                if sim >= 0.6 and len(shared) >= 1:
                    yield CrossRef(ev_a, ev_b, sim, shared)
```

名称相似度计算：字符交集 / min(len_a, len_b)。阈值0.6。

置信度三级分档:

置信度	条件	典型案例
high	相似度>=0.8 且 共享>=2人物 且 同年	005-135↔043-025 长平之战（白起）
medium	相似度>=0.6 且 共享>=1人物	016-018↔007-061 垓下（名称不完全一致）
low	相似度>=0.6 但 共享0人物（仅名称相似）	需人工确认

典型高置信度互见:

事件A	事件B	相似度	共同人物
005-135 长平之战	043-025 长平之战	1.0	白起
006-034 荆轲刺秦王	086-017 荆轲刺秦王	1.0	荆轲
006-083 沙丘之谋	087-007 始皇崩沙丘之谋	1.0	李斯、胡亥、赵高
002-044 汤践天子位	003-009 汤践天子位	1.0	汤
004-045 犬戎灭西周	110-003 犬戎杀幽王西周灭亡	1.0	幽王、申侯

co_person（共人）发现 — 969条

跨章事件共享**>=2个关键人物**。

```
def find_co_persons(events_by_chapter):
    """跨章共人: 共享>=2个人物"""
    person_index = defaultdict(list)  # 人物 → 事件列表
```

```
    for ev in all_events:
        for person in ev.persons:
            person_index[person].append(ev)
    # 对每对跨章事件，计算共享人物数
    # 共享>=2人 → 建立co_person关系
```

关键：人物匹配需经过别名合并（如 刘邦=沛公=高祖=汉王），参考 `entity_aliases.json`。

co_location（共地）发现 — 705条

跨章事件共享**地点 + >=1人物**。

```
def find_co_locations(events_by_chapter):
    """跨章共地：共享地点+>=1人物（过滤高频地名噪音）"""
    # 纯地点相同但无共人的不算（避免"长安"产生噪音）
    # 高频地名黑名单：长安、咸阳、洛阳 — 出现次数>100的地名需要更严格的共人条件
```

concurrent（同期）发现 — 230条

同一公元年的跨章事件共享**>=1人物**。

```
def find_concurrent(events_by_chapter):
    """同年+共人：同一公元年的跨章事件共享人物"""
    year_index = defaultdict(list)  # ce_year → 事件列表
    for ev in all_events:
        if ev.ce_year:
            year_index[ev.ce_year].append(ev)
    # 同年 + 跨章 + 共享>=1人物 → concurrent
```

3.2 LLM推理方法（4种，2,187条）

LLM只处理章内关系，跨章由自动方法处理。 避免重复发现，降低成本。

```
def extract_relations(chapter_events):
    """对每章内的事件列表提取语义关系"""
    prompt = f"""分析以下事件之间的关系。

## 关系类型（只使用这4种）
- sequel: B是A的直接后续（A的结果创造B的前提）
- causal: A是B的直接原因（比sequel更强：A不发生则B不发生）
- part_of: A是B的子事件（source=子，target=父）
- opposition: 对立双方的行动

## 输出格式
JSON数组: [{{"type": "...", "source": "事件ID", "target": "事件ID",
"reason": "<=10字"}}]

## 注意
- sequel只针对直接后续，不要跳跃
- causal要求严格因果（A不发生则B不发生）
- part_of方向：source=子事件，target=父事件
- opposition仅限明确的对立行动
- reason必须<=10个字

## 事件列表
{format_events(chapter_events)}
"""
    # 每章单独处理，避免跨章关系
```

四种LLM关系的判断标准：

关系	判断问题	正例	反例
sequel	"B是A的直接后续吗？"	斩宋义→拜上将军	陈胜起义→垓下之战（跳跃太远）
causal	"A不发生，B还会发生吗？"	商鞅变法→秦国强大	秦统一→汉兴（中间有太多环节）

关系	判断问题	正例	反例
part_of	"A是B的一个环节吗？"	四面楚歌 part_of 垓下之围	长平之战 part_of 秦灭六国（粒度差太大）
opposition	"双方行动都有明确记述吗？"	荥阳相持↔项羽拔荥阳	秦攻赵↔赵抵抗（赵方无具体记述）

章内 causal 的完整推理逻辑：

causal 要求**严格充分因果**，推理时逐步检查：

```
① 时序检查：A 是否早于 B？（必要条件，时序颠倒直接否定）
② 充分性检查：A 不发生，B 是否仍然可能发生？
   - 若 B 有其他独立路径 → 降级为 sequel（时序关联）
   - 若 B 完全依赖 A → 确认 causal
③ 直接性检查：A 到 B 之间是否有跳跃的中间事件？
   - 若中间有 3+ 个独立事件 → 降级为 sequel 链（分多段记录）
   - 若直接相连或仅有一步过渡 → 确认 causal
④ 章内限定：跨章因果不用 causal，改用 cross_causal（见 3.7）
```

causal vs sequel 的边界判断：

情形	判断	理由
斩宋义 → 项羽拜上将军	causal	斩宋义是获得军权的直接原因
垓下之战 → 项羽乌江自刎	causal	兵败是自刎的直接触发
鸿门宴 → 刘邦汉中称王	sequel	鸿门宴不是称王的充分原因（项羽分封才是）
陈胜起义 → 垓下之战	不建立	跳跃太远，中间事件太多

3.3 事件链提取（LLM辅助）

事件提取阶段同时要求LLM输出2-5条**因果链**，这些链直接成为sequel/causal关系的初始数据。

```
## 关键事件链

1. **商朝建立** (003-001 → 003-002 → 003-009)
   契受封 → 先公世系 → 汤践天子位

2. **楚汉争霸** (008-008 → 008-009 → ... → 008-063)
   陈胜起义 → 起兵沛县 → ... → 垓下之战
```

事件链的价值:
- 串联孤立事件为有意义的叙事线
- 自动生成sequel关系的候选对
- 是地铁图中"线路"的核心数据

3.4 实体关系三元组提取

除事件间关系外，还可从标注文本中提取实体级别的关系三元组:

```
(主语实体, 关系类型, 宾语实体)
```

关系类型	示例	来源
击败	(刘邦, 击败, 项羽)	事件描述
辅佐	(张良, 辅佐, 刘邦)	事件描述
封为	(高祖, 封为, 张良→留侯)	侯表事件
父子	(项燕, 父, 项梁)	家族事件
君臣	(秦昭王, 君, 白起)	列传叙事

三元组与事件关系互补：事件关系连接**事件节点**，三元组连接**实体节点**，共同构成知识图谱的两层网络。

3.5 跨章节交叉定年（关系发现的副产品）

发现跨章互见关系时，可同时进行年代交叉验证。

定年优先级（来源：`SKILL_04c_事件年代推断.md` 逻辑5）：

```
本纪 > 年表 > 世家 > 列传
```

- 同一事件在多个章节出现，取**精确度最高**的标注
- 年表有明确年份 → 用年表
- 本纪有纪年标记 → 用本纪
- 世家/列传有推算 → 仅作参考

已知重大事件定年表（跨章验证锚点）：

事件	公元前	适用章节
长平之战	260年	005/015/043/073/076/079/081
邯郸之围	258—257年	076/077/078
乐毅伐齐	284年	080/082
田单复齐	279年	046/080/082
陈胜起义	209年	016/048/007
鸿门宴	206年	007/008
垓下之战	202年	007/008/016/092
白登之围	200年	008/110
诛诸吕	180年	009/017/019/022/057

事件	公元前	适用章节
七国之乱	154年	011/017/019/022/058/101
酎金国除	112年	018/019/020/021/030

3.6 十表特有的补充规则

自动算法对十表事件的发现率偏低（021仅4条跨章关系），原因：
- 表中事件名多为批次性描述（"元朔三年大批王子封侯"），与纪传事件名不相似
- 表中事件人物标注少（批量事件不逐人列出），共人条件难满足

补充规则：

1. **年代精确匹配**：表中事件有精确公元年，与纪传中同年事件直接建立concurrent关系（降低共人门槛到0）
2. **关键词匹配**：表中"封侯"/"国除"/"反"等关键词与纪传事件描述匹配
3. **侯表-世家/列传对应**：侯名/人名直接查找对应章节
4. **跨表同事件**：同一公元年+同类关键词（"酎金""七国""封"）→ cross_ref

3.7 跨章因果推理（LLM二次推理）

目的：自动关系（cross_ref/co_person/co_location）识别了跨章关联，但不判断因果。此步骤对这批候选对进行LLM推理，提取跨章因果链。

五种跨章因果子类型：

类型	定义	典型案例
direct_cause	A直接导致B发生（充分条件）	田光荐荆轲自刎→荆轲刺秦王（conf 0.95）
prerequisite	A是B得以发生的必要前提	萧何荐曹参→曹参入相（conf 0.95）
background	A构成B的历史背景条件	

类型	定义	典型案例
		吕氏封侯擅权→诸吕之乱平定（conf 0.80）
trigger	A成为B的导火索（时间紧密相连）	韩信袭齐→郦生被烹（conf 0.90）
precedent	A为B提供历史先例或模板	孔子作春秋→左丘明作左氏春秋（conf 0.85）

与章内causal的区别：

- 章内 `causal`：同一章内，A→B满足"A不发生则B不发生"（严格因果）
- 跨章 `cross_causal`：不同章节，细分为5种强度，置信度0-1浮点

候选对筛选规则：

```
# 从 event_relations.json 中筛选候选对
for r in relations:
    src_ch = r['source'].split('-')[0]
    tgt_ch = r['target'].split('-')[0]

    if src_ch == tgt_ch: continue          # 排除章内（已由causal处理）
    if r['type'] not in ('cross_ref', 'co_person', 'co_location'): continue

    # 时序过滤：确保 source 年代早于 target
    yr_src = events[r['source']]['year'] or 0
    yr_tgt = events[r['target']]['year'] or 0

    if r['type'] == 'cross_ref':
        # cross_ref 双向，取时序较早的为 source
        if yr_src > yr_tgt: src, tgt = tgt, src
    else:
```

```
        if yr_src == yr_tgt: continue       # 同年 co_person 不做因果推断
        if yr_src > yr_tgt: src, tgt = tgt, src
```

LLM推理提示词结构:

```
分析事件A（源章节、年代、类型、人物、描述）与
事件B（目标章节、年代、类型、人物、描述）之间是否存在因果关系。

输出:
{
  "has_causal": true/false,
  "causal_type": "direct_cause/prerequisite/background/trigger/precedent/no_causal",
  "confidence": 0.0-1.0,
  "reasoning": "1-2句说明因果逻辑"
}
```

判断原则（来自实践经验）:

1. **cross_ref ≠ 因果**：同一事件跨章重复记载（如"长平之战"同时在赵世家和六国年表），无论相似度多高，均为 `no_causal`。仅当A章描述起因、B章记录结果且时序不同时，才存在因果。
2. **时序颠倒判 no_causal**：目标事件年代早于源事件，则无因果。
3. **空描述判 no_causal**：共同描述字段为空时（部分十表事件），无法判断因果，默认 `no_causal`。
4. **同期平行事件区分**：同年共人但各自独立的政治行动，判 `no_causal`，而非background。
5. **并行多链因果**：同一后果可能有多个前因（如诛吕事件有张辟彊献策→direct_cause，也有吕氏封侯擅权→background），两者同时成立，分别记录。

规模与效率:

指标	数值
候选对总数	1,490
实际分析（过滤后）	1,457
确认因果	352（24.2%）
高置信度（≥0.8）	177
涉及章节	102 / 130
推理方法	8 Agent 并行，每批 ~187 对

输出文件：`kg/relations/causal_relations.json`


```
{
  "meta": {
    "total_pairs_analyzed": 1457,
    "causal_relations_found": 352,
    "method": "llm_reasoning",
    "model": "claude-sonnet-4-6",
    "by_causal_type": {
      "background": 138, "prerequisite": 100,
      "direct_cause": 96, "trigger": 15, "precedent": 3
    }
  },
  "relations": [
    {
      "source": "086-014",
      "target": "006-034",
      "source_name": "田光荐荆轲自刎",
      "target_name": "荆轲刺秦王",
      "source_chapter": "刺客列传",
      "target_chapter": "秦始皇本纪",
      "source_year": -228,
      "causal_type": "direct_cause",
```

```
      "confidence": 0.95,
      "reasoning": "田光荐荆轲并自刎以激励其成行，直接促成荆轲接受刺秦任
务",
      "base_relation": "co_person"
    }
  ]
}
```

常见高质量跨章因果示例:

因果对	类型	conf	说明
田光荐荆轲自刎→荆轲刺秦王	direct_cause	0.95	086→006
赵高谮李斯→赵高杀李斯	direct_cause	0.95	087→006
萧何荐曹参继相→曹参守成不变	direct_cause	0.95	053→054
周勃诛吕安刘→文帝论功行赏	direct_cause	0.95	057→010
刘敬献和亲之策→汉匈和亲始约	direct_cause	0.95	099→110
吕氏封侯擅权→诸吕之乱平定	background	0.80	019→049
窦太后崩→武帝推儒术	direct_cause	0.95	121→107
专诸刺吴王僚→阖庐即位	direct_cause	0.95	066→031
火牛阵破燕→田单复齐	direct_cause	0.95	082→046
越王栖会稽→计然助越灭吴	direct_cause	0.90	031→129

- 表中事件名多为批次性描述（"元朔三年大批王子封侯"），与纪传事件名不相似
- 表中事件人物标注少（批量事件不逐人列出），共人条件难满足

补充规则:

1. **年代精确匹配**：表中事件有精确公元年，与纪传中同年事件直接建立concurrent关系（降低共人门槛到0）
2. **关键词匹配**：表中"封侯"/"国除"/"反"等关键词与纪传事件描述匹配
3. **侯表-世家/列传对应**：侯名/人名直接查找对应章节
4. **跨表同事件**：同一公元年+同类关键词（"酎金""七国""封"）→ cross_ref

四、关系模式识别清单

4.1 必查的高频模式

模式	信号	关系类型	示例
同一事件跨章记述	名称相似+共人+同年	cross_ref	长平之战(015↔043↔081)
政策→实施→后果	同主题+时间递进	causal链	推恩令→分封→国除
人物生命线	同一人物跨章出现	co_person	周勃：057↔018↔019↔022
跨表同事件	同年+同类关键词	cross_ref	酎金国除(018↔019↔020↔021)
战争-封赏对	战争后1-3年内封侯	sequel	平七国→019-013封功臣
谋反-诛杀对	反→诛/国除	causal	淮南王谋反→国除(017-018)
制度讨论-实施	书中论述→表中执行	causal	030平准书→酎金政策

4.2 十表间的对应矩阵

	013	014	015	016	017	018	019	020	021	022
013	-	弱	弱	-	-	-	-	-	-	弱
014	弱	-	强	-	-	-	-	-	-	-
015	弱	强	-	中	-	-	-	-	-	-
016	-	-	中	-	中	中	-	-	-	强
017	-	-	-	中	-	强	强	-	强	强
018	-	-	-	中	强	-	中	-	-	强
019	-	-	-	-	强	中	-	-	-	中
020	-	-	-	-	-	-	-	-	中	强
021	-	-	-	-	强	-	-	中	-	中
022	弱	-	-	强	强	强	中	强	中	-

强关联：同一时段+同一主题（如017-018高祖封侯/封王）
中关联：时段重叠+偶有同事件（如016-018秦楚过渡期功臣）
弱关联：仅时段相邻或序言互引

4.3 表与纪传的对应关系

表章	强关联纪传	关联类型
013 三代世表	001五帝、002夏、003殷、004周	世系互见
014 十二诸侯年表	031-040世家（鲁齐晋楚等）	逐年互见
015 六国年表	043-046世家（赵魏韩齐）、005秦本纪	逐年互见
016 秦楚之际月表	007项羽、008高祖本纪	逐月互见（最密）
017 诸侯王年表	050-052楚齐世家、058梁孝王、118淮南衡山	封/废互见
018 高祖功臣侯者年表	053萧何、055张良、057周勃、092韩信	封侯/诛杀互见

表章	强关联纪传	关联类型
019 惠景间侯者年表	057周勃、049外戚、101袁盎晁错	封侯/政变互见
020 建元以来侯者年表	111卫青霍去病、112公孙弘、113南越	军功/外交互见
021 王子侯者年表	059五宗世家、112平津侯主父偃	推恩实施互见
022 将相名臣年表	几乎所有汉代纪传	任免时间互见

五、实操流程

5.1 跨章关系发现步骤

```
步骤1：自动发现基础关系
  python kg/events/scripts/extract_event_relations.py
  → 产出 cross_ref / co_person / co_location / concurrent

步骤2：LLM推理章内关系
  对每章事件列表 → sequel / causal / part_of / opposition

步骤3：LLM推理跨章因果关系
  python kg/relations/scripts/build_causal_relations.py
  → 以步骤1的cross_ref/co_person/co_location为候选对输入
  → 产出 cross_causal（352条，含5种子类型）
  → 输出 kg/relations/causal_relations.json

步骤4：补充十表特有关系（人工/半自动）
  4a. 跨表同事件：按公元年+关键词匹配
  4b. 制度因果链：按主题+时间递进串联
  4c. 人物生命线：按人物倒排索引串联
  4d. 表-纪互见补充：对自动发现遗漏的低相似度互见，用年代精确匹配补充
```

```
步骤5：合并去重
  按(source, target)去重，优先保留LLM结果（有reason字段）
```

5.2 质量验证

检查项	方法
互见年代一致	cross_ref两端事件的公元年差<=3年
因果方向正确	causal的source时间早于target
sequel不跳跃	sequel两端事件之间无遗漏的中间事件
跨表同事件完整	酎金/七国等已知跨表事件在所有相关表中都建立了关系
人物生命线连续	高频人物（高祖/周勃/卫青等）的跨章事件按时间有序

六、关系数据格式

6.1 event_relations.json 结构

```
{
  "metadata": {
    "total_events": 3185,
    "total_relations": 7652,
    "auto_relations": 5126,
    "llm_relations": 2525
  },
  "relations": [
    {
      "type": "cross_ref",
      "source": "015-020",
      "target": "043-025",
```

```
      "confidence": "high",
      "similarity": 1.0,
      "shared_persons": ["白起"],
      "reason": "长平之战跨章互见"
    },
    {
      "type": "sequel",
      "source": "017-006",
      "target": "017-019",
      "reason": "七国乱后国除"
    }
  ]
}
```

6.2 关系统计产出

```
| 类型 | 数量 | 占比 | 来源 |
|------|------|------|------|
| concurrent | 2,997 | 39% | 自动 |
| sequel | 1,624 | 21% | LLM |
| co_person | 1,071 | 14% | 自动 |
| co_location | 737 | 10% | 自动 |
| causal | 407 | 5% | LLM |
| cross_causal | 338 | 4% | LLM |
| cross_ref | 321 | 4% | 自动 |
| part_of | 107 | 1% | LLM |
| opposition | 50 | 1% | LLM |
```

七、可视化：事件地铁图

关系数据的最终消费形态之一是**事件地铁图**——将130篇章节映射为地铁线路，事件为站点，跨章关系为换乘。

7.1 概念映射

地铁概念	历史概念	数据来源
线路	章节（130条线路）	事件索引文件
站点	事件（3,185个站点）	事件索引概览表
换乘	跨章关系（1,876条换乘）	event_relations.json
X轴	公元年时间线（前2700~前87年）	CE年标注
线路颜色	体裁分组（本纪红/表灰/书紫/世家多色/列传多色）	章节分类

7.2 换乘数据

从 `event_relations.json` 中筛选跨章关系作为换乘：

```
TRANSFER_TYPES = ['cross_ref', 'co_person', 'co_location',
'concurrent']
# 排除 sequel/causal/part_of/opposition（章内关系，不形成"换乘"）
```

换乘数据结构：

```
{
  "events": ["005-135", "043-025"],
  "lines": ["ch005", "ch043"],
  "name": "长平之战",
  "type": "cross_ref",
  "year": -260,
  "shared_people": ["白起"]
}
```

7.3 站点定位算法

X坐标（时间轴）：

```
def compute_x_pos(events):
    """为每个事件计算时间轴位置"""
    for event in events:
        if event.ce_year:
            event.x_pos = event.ce_year  # 有CE年直接用
        else:
            # 线性插值：找前后最近的有日期事件
            prev = find_nearest_dated(events, event.index,
backward=True)
            next_ = find_nearest_dated(events, event.index,
backward=False)
            if prev and next_:
                t = (event.index - prev.index) / (next_.index -
prev.index)
                event.x_pos = prev.ce_year + t * (next_.ce_year -
prev.ce_year)
            elif prev:
                event.x_pos = prev.ce_year + 2  # 向后外推
            elif next_:
                event.x_pos = next_.ce_year - 2  # 向前外推
            else:
                event.x_pos = -500 + event.index * 5  # 完全无日期的
兜底
```

77%的事件有CE年，剩余23%通过线性插值获得X坐标，确保地铁图没有"断站"。

7.4 可视化规模

指标	数值
线路	130

指标	数值
站点	3,185
换乘	1,876
章内链	2,188
数据文件大小	3.2MB
时间跨度	前2700年 ~ 前87年

八、关系合并与去重

8.1 合并策略

自动方法和LLM可能发现同一对关系，合并规则：

1. 按 `(source, target)` 去重
2. 同一对存在多种关系类型时，**全部保留**（如A→B既是sequel又是causal）
3. 同一对同一类型重复时，优先保留LLM结果（有reason字段）
4. cross_ref关系保留相似度最高的记录

8.2 关系统计产出格式

```
| 类型 | 数量 | 占比 | 来源 |
|------|------|------|------|
| concurrent | 2,997 | 39% | 自动 |
| sequel | 1,624 | 21% | LLM |
| co_person | 1,071 | 14% | 自动 |
| co_location | 737 | 10% | 自动 |
| causal | 407 | 5% | LLM |
| cross_causal | 338 | 4% | LLM |
```

```
| cross_ref | 321 | 4% | 自动 |
| part_of | 107 | 1% | LLM |
| opposition | 50 | 1% | LLM |
```

九、经验教训

9.1 事件提取阶段

1. **体裁决定粒度** — 本纪平均63事件/篇，列传20个，表5个（补充后22.6个）。用统一粒度标准会让本纪太粗或列传太细。
2. **事件链比孤立事件更有价值** — 提取时要求LLM同时输出2-5条因果链，这些链直接成为sequel/causal关系的初始数据。
3. **保留实体标记** — 事件描述中保留 `@` `=` `$` `%` `&` 标记，后续自动化（共人/共地计算）直接从标记中提取，无需二次NER。

9.2 关系发现阶段

1. **章内用LLM，跨章用自动** — 核心分工。LLM擅长理解叙事因果，自动方法擅长跨文档匹配。各占一半。
2. **sequel是最大类别（37%）** — 反映古籍叙事的线性特征。历史事件天然形成时间链。
3. **cross_ref置信度分级** — 名称相似度1.0+共人是最可靠信号。相似度0.6-0.8需人工确认。"长平之战"在秦本纪和赵世家各出现一次，名称完全相同且共享白起。
4. **co_location需+共人过滤** — 纯地点匹配噪音大（"长安"出现在太多事件中），加上共人条件大幅提升精度。

9.3 十表特有难点

1. **批量事件匹配难** — 表中"元朔三年大批王子封侯"不会与任何纪传事件名称匹配。需要用年代+关键词补充规则。
2. **人物标注稀疏** — 侯表中批量事件不逐人列出，co_person算法失效。解法：对侯表事件降低共人门槛，改用"同年+同主题"匹配。

3. **跨表同事件容易遗漏** — 酎金国除在四个侯表中各有记载，但事件名称不完全相同（"酎金坐罪大量国除" vs "酎金国除56侯详情"）。需要建立"已知跨表事件清单"人工补充。

4. **制度因果链时间跨度大** — 推恩令（前127年）到酎金国除（前112年）跨15年，sequel算法默认只找"直接后续"会跳过。需要允许制度链的时间跨度到50年。

9.4 工程实践

1. **先提取后标注后关联** — 三个阶段严格顺序执行。事件提取完成后再标注纪年，纪年稳定后再发现关系。逆序会导致频繁返工。

2. **关系合并需要去重** — 自动方法和LLM可能发现同一对关系，合并时按(source, target)去重，优先保留LLM结果。

3. **插值算法让可视化完整** — 77%的事件有CE年，剩余23%通过线性插值获得X坐标，确保地铁图没有"断站"。

4. **表-纪互补是核心价值** — 十表提供时间骨架（精确到年/月），纪传提供叙事血肉。关系发现的最大价值是把这两层连接起来。

十、工具链

脚本	功能	输入	输出
`extract_events.py`	从标注文本提取事件	tagged.md	事件索引.md
`extract_event_relations.py`	自动+LLM关系发现	事件索引	event_relations.json
`annotate_ce_years.py`		事件索引+reign_periods	更新的事件索引

脚本	功能	输入	输出
	公元纪年标注		
`lint_ce_years.py`	纪年质检	事件索引	验证报告
`validate_events.py`	事件格式验证	事件索引	验证报告
`build_year_map.py`	年份消歧映射	年表章节+tagged.md	year_ce_map.json
`build_metro_map_data.py`	生成地铁图数据	事件索引+关系	metro_map_data.json

关系推理脚本位于 `kg/relations/scripts/`：

脚本	功能	输入	输出
`build_causal_relations.py`	LLM跨章因果推理	event_relations.json+事件索引	causal_relations.json

执行顺序：

```
# 1. 提取事件（需Claude API）
python kg/events/scripts/extract_events.py
```

```
# 2. 构建年份映射
python kg/events/scripts/build_year_map.py

# 3. 标注公元纪年
python kg/events/scripts/annotate_ce_years.py

# 4. 纪年质检 → 修复 → 重新标注（迭代）
python kg/events/scripts/lint_ce_years.py

# 5. 发现事件关系（需Claude API做LLM部分）
python kg/events/scripts/extract_event_relations.py

# 6. 格式验证
python kg/events/scripts/validate_events.py

# 7. 生成地铁图数据
python kg/events/scripts/build_metro_map_data.py
```

extract_event_relations.py 运行模式

```
# 全流程（自动关系 + LLM章内关系）
python kg/events/scripts/extract_event_relations.py

# 仅重算自动关系（无API成本，用于新增事件后更新）
python kg/events/scripts/extract_event_relations.py --auto-only

# 仅跑LLM关系（指定章节）
python kg/events/scripts/extract_event_relations.py --llm-only --
chapters 013 014 015

# 预览将要处理的内容，不实际写文件
python kg/events/scripts/extract_event_relations.py --dry-run
```

关键注意事项:

- `--auto-only` 会**覆盖** `event_relations.json` 中原有的LLM关系（sequel/causal/part_of/opposition），需要先备份或之后手动合并。
  合并方法:

```
python # 从git恢复LLM关系后合并 old = json.loads(subprocess.run(['git','show','HEAD:kg/events/data/event_relations.json'], capture_output=True, text=True).stdout) llm_rels = [r for r in old['relations'] if r['type'] in {'sequel','causal','part_of','opposition'}] new_data['relations'] = new_auto_rels + llm_rels
```

- 重新运行自动关系后，**concurrent 数量会随CE年覆盖率大幅变化**（年代审查五轮后从230条增至2,997条，因覆盖率25%→98.7%）。
- 输出文件:
- `kg/events/data/event_relations.json` — 所有关系数据
- `kg/events/data/event_relations_summary.md` — 统计摘要
- 自动关系计算不需要API，几秒内完成；LLM关系按章逐一推理，需要API Token。

十一、扩展到其他古籍

11.1 直接复用

- 事件Schema（7个字段）
- 8种关系类型定义
- 自动关系发现算法（cross_ref/co_person/co_location/concurrent）
- 纪年验证逻辑（时序检查/大跨度/已知日期）
- 地铁图可视化框架
- 插值定位算法

11.2 需要适配

项目	需调整内容
体裁分类	编年体（资治通鉴）无本纪/列传之分，事件粒度统一
reign_periods.json	每部书覆盖的朝代不同，需重建在位年数据库
年号表	汉以后年号系统更复杂（改元频繁）
已知日期表	按所涉历史时期定制
事件类型	11种基本够用，可能需增加"外交"等细分类
提取提示词	按具体古籍的行文风格调整

11.3 规模估算

古籍	预估事件数	预估关系数	Token成本（提取+关系）
史记	3,185	7,314	~4M
汉书	~4,000	~6,000	~4M
资治通鉴	~15,000	~25,000	~20M

本SKILL文档基于《史记》130篇事件关系发现实践（2025-02至2026-03）提炼。
核心产出：3,185个事件、7,652条关系、1,876条跨章换乘、十表231事件参与458条跨章关系。
本文档为事件关系发现的唯一权威SKILL，整合了原分散在多个SKILL文档中的关系发现规则。

SKILL 05b: 实体关系构建 — 人物/地点/制度间的结构化关系

> 从标注文本中抽取实体之间的结构化关系（地点、制度、事物），构建关系网络的非人物部分。

零、任务定位

实体关系构建负责处理**非人物实体间的结构化关系**，包括：
- **地点关系**：行政隶属、地理邻接、历史沿革
- **制度关系**：官职等级、法律继承、礼仪演变
- **事物关系**：因果链条、引用关系、同义异名

人物关系已独立为 SKILL 05c（人物共现关系），本工序与之配合，共同构建完整的实体关系网络。

一、与其他关系类型的区别

维度	事件关系（05a）	实体关系（05b）	人物关系（05c）
主体	事件与事件	地点/制度/事物与实体	人物与人物
类型	因果/时序/互见/包含	隶属/邻接/沿革/等级	父子/君臣/师徒/共现
粒度	段落/章节级	实体级	人物级

维度	事件关系（05a）	实体关系（05b）	人物关系（05c）
产出	event_relations.json	geo_relations.json, office_relations.json	person_relations.json

---

二、关系类型体系

2.1 地点关系（41-46）

编号	关系	示例	数据文件
41	隶属	咸阳→关中	41_地点关系_隶属.md
42	邻接	齐↔鲁	42_地点关系_邻接.md
43	沿革	大梁→开封	43_地点关系_沿革.md
44	封地	韩信→齐	44_地点关系_封地.md
45	战场	垓下之战→垓下	45_地点关系_战场.md
46	都城	秦→咸阳	46_地点关系_都城.md

2.2 制度关系（51-56）

编号	关系	示例
51	官职等级	丞相>御史大夫
52	官职隶属	太史令→太常
53	法律继承	汉律←秦律

编号	关系	示例
54	礼仪演变	周礼→汉礼
55	爵位等级	王>侯>伯>子>男
56	谥号规则	文帝、武帝

2.3 事物关系（61-66）

编号	关系	示例
61	同义	太史公≈司马迁
62	异名	项羽=西楚霸王
63	引用	史记→尚书
64	因果	焚书→坑儒
65	包含	本纪∈史记
66	度量	一里=300步

三、规模估算（待构建）

3.1 地点关系规模

关系类型	预估数量	说明	状态
隶属	300–500	郡县隶属、都城隶属	🔜 待构建
邻接	200–400	诸侯国邻接、地理邻接	🔜 待构建

关系类型	预估数量	说明	状态
沿革	100–200	地名变迁、行政区划变化	🔜 待构建
封地	500–800	诸侯封地、功臣封地	🔜 待构建
战场	200–300	重大战役发生地	🔜 待构建
都城	50–100	历代都城	🔜 待构建
合计	**1350–2300**		

3.2 制度关系规模

关系类型	预估数量	说明	状态
官职等级	100–200	丞相/御史大夫/太尉等	🔜 待构建
官职隶属	200–300	九卿/诸侯官制	🔜 待构建
法律继承	20–50	秦汉法律沿革	🔜 待构建
礼仪演变	30–60	周礼→汉礼	🔜 待构建
爵位等级	50–100	王侯伯子男	🔜 待构建
谥号规则	100–200	帝王谥号	🔜 待构建
合计	**500–910**		

四、工具

脚本	功能
`kg/relations/scripts/extract_geo_relations.py`	提取地点关系
`kg/relations/scripts/extract_office_relations.py`	提取制度关系
`kg/relations/scripts/extract_entity_relations.py`	提取事物关系
`kg/relations/scripts/build_geo_network.py`	构建地理网络
`kg/relations/scripts/build_office_hierarchy.py`	构建官制等级树

五、产出数据

文件	说明
`kg/relations/geo_relations.json`	地点关系（JSON）
`kg/relations/office_relations.json`	制度关系（JSON）
`kg/relations/entity_relations.json`	事物关系（JSON）
`kg/relations/geo_network.html`	地理网络可视化
`kg/relations/office_hierarchy.json`	官制等级树

六、SPO三元组输出

实体关系以主-谓-宾（SPO）三元组形式结构化存储，可直接导入知识图谱：

地点关系示例：

```
{
  "subject": "咸阳",
  "predicate": "隶属",
  "object": "关中",
  "chapter": "006",
  "evidence": "[1.2] 秦都咸阳，关中之地..."
}
```

制度关系示例:

```
{
  "subject": "丞相",
  "predicate": "官职等级",
  "object": "御史大夫",
  "relation": "高于",
  "chapter": "087",
  "evidence": "[3.1] 丞相、御史大夫、太尉，号三公..."
}
```

关系	主语类型	宾语类型	示例
隶属	地	地	咸阳 → 关中
邻接	地	地	齐 ↔ 鲁
封地	人	地	韩信 → 齐
都城	国	地	秦 → 咸阳
官职等级	官	官	丞相 > 御史大夫
官职隶属	官	官	太史令 → 太常
同义	实体	实体	太史公 ≈ 司马迁

七、与其他工序的关系

上游	本工序	下游
03a 实体标注（实体识别）	实体关系构建	06a 本体构建（关系本体化）
03b 实体消歧（唯一指称）	→ 消歧后才能准确建关系	07 逻辑推理（地理推理）
05a 事件关系（事件共现）	→ 共地关系补充	09a 阅读器（关系网络可视化）
05c 人物共现关系	→ 人地关系补充（封地/战场）	

SKILL 05c: 人物关系构建 — 从共现到家谱的多层次人物关系网络

> 从标注文本中构建人物关系网络：共现分析→家族推理→政治社会关系抽取，最终形成完整的人物知识图谱。

零、任务定位

人物关系构建是关系构建（SKILL 05）的核心子任务，专注于人物与人物之间的结构化关系。与地点关系（05b）并列，共同构建完整的实体关系网络。

与其他关系类型的区别

维度	人物关系（05c）	实体关系（05b）	事件关系（05a）
主体	人物与人物	地点/制度/事物	事件与事件
类型	家族/政治/社会/共现	隶属/等级/沿革	因果/时序/互见
粒度	人物级	实体级	段落/章节级
产出	person_relations.json, family_tree.json, imperial_genealogy.json	geo_relations.json, office_relations.json	event_relations.json

一、三层关系体系

人物关系构建采用**从弱到强、从浅到深**的三层递进架构：

```
Layer 1：共现关系（弱关联）
  ↓ 提供候选关系对
Layer 2：家族关系（强关联，可推理）
  ↓ 传递推理扩展
Layer 3：政治社会关系（强关联，需上下文）
```

层次	子任务	输入	输出	方法
L1: 共现	05c1 人物共现关系	`*.tagged.md`	`person_cooccurrence.json`	滑动窗口+PMI
L2: 家族	05c2 家族关系推理	共现+关键词	`family_relations.json`，帝王家谱	规则+传递推理
L3: 政治社会	05c3 政治社会关系	共现+上下文	`political_relations.json`，`social_relations.json`	规则+LLM

二、子工序

编号	文件	内容	状态
05c1			

编号	文件	内容	状态
	SKILL_05c1_人物共现关系.md	段落级共现抽取、PMI计算、候选关系对生成	📝 规划中
05c2	SKILL_05c2_家族关系推理.md	父子/兄弟/夫妻关系抽取、传递推理、帝王家谱构建	✅ 部分完成
05c3	SKILL_05c3_政治社会关系.md	君臣/师徒/敌对等关系抽取	📝 规划中

三、规模估算

3.1 基于前5章词频外推

关系大类	代表词	全书估计次数	抽取难度
家庭（父子/兄弟/长子/少子…）	父子、兄弟、长子、少子、子孙、苗裔	~470词×全书倍率 ≈ **9000–12000**	中（词表+句法）
依附（舍人/宾客/食客/门下…）	舍人、宾客、食客、从者、门下	~410×倍率 ≈ **7000–9000**	中
师徒（弟子/受业…）	弟子、受业	~100×倍率 ≈ **1500–2000**	低（词表）
君臣	君臣、臣子	~46×倍率 ≈ **600–900**	低
友人/知己	故人、知己、旧交	~36×倍率 ≈ **400–600**	高（动态陈述为主）
姻亲	外戚、婚姻	~18×倍率 ≈ **200–400**	高（多用婚姻事件描述）

关系大类	代表词	全书估计次数	抽取难度
仇敌	仇雠、仇人	~12×倍率 ≈ **150–250**	高
合计（关系词可见部分）		**~19000–25000**	

> **注**：上表为关系词的"显式出现次数"，实际三元组抽取量预估为词频的 30–50%，即 **6000–12000 条关系三元组**。

3.2 各层关系规模对比

层次	关系数量	精度	召回率
L1 共现关系	**20000–30000** 对（段落级）	低（包含大量噪音）	高（覆盖所有可能关系）
L2 家族关系	**2000–3000** 条（直接+推理）	高（规则可靠）	中（受限于关键词）
L3 政治社会关系	**4000–9000** 条	中高（依赖 LLM）	中

四、关系类型体系

4.1 家庭关系（01-06）

编号	关系	示例	数据文件	工序
01	父子	黄帝→玄嚣、昌意	01_家庭关系_父子.md	05c2
02	母子	嫘祖→玄嚣	02_家庭关系_母子.md	05c2

编号	关系	示例	数据文件	工序
03	兄弟	玄嚣↔昌意	03_家庭关系_兄弟.md	05c2
04	夫妻	黄帝↔嫘祖	04_家庭关系_夫妻.md	05c2
05	祖孙	黄帝→高阳	05_家庭关系_祖孙.md	05c2
06	叔侄	—	06_家庭关系_叔侄.md	05c2

4.2 政治关系（11-17）

编号	关系	示例	工序
11	君臣	秦始皇↔李斯	05c3
12	推荐	百里奚推荐蹇叔	05c3
13	封赏	高祖封韩信为齐王	05c3
14	辅佐	张良辅佐刘邦	05c3
15	献策	陈平献计	05c3
16	背叛	韩信叛汉	05c3
17	出使	张骞出使西域	05c3

4.3 社会关系（21-31）

编号	关系	示例	工序
21	师徒	孔子↔颜回	05c3
22	友人	管仲↔鲍叔牙	05c3

编号	关系	示例	工序
31	敌对	项羽↔刘邦	05c3

五、工具与脚本

5.1 共现关系抽取（05c1）

脚本	功能	状态
`kg/relations/scripts/extract_cooccurrence.py`	提取人物共现关系	📝 待开发
`kg/relations/scripts/calculate_pmi.py`	计算共现强度	📝 待开发
`kg/relations/scripts/filter_candidate_relations.py`	过滤候选关系对	📝 待开发

5.2 家族关系推理（05c2）

脚本	功能	状态
`kg/genealogy/scripts/extract_family_relations.py`	提取家族关系	✅ 已完成
`kg/genealogy/scripts/extract_imperial_genealogy.py`	构建帝王世系	✅ 已完成
`kg/genealogy/scripts/infer_indirect_relations.py`	传递推理间接关系	📝 待开发
`kg/genealogy/scripts/build_family_tree.py`	构建家族树	

脚本	功能	状态
		📝 待开发
`kg/genealogy/scripts/validate_family_tree.py`	验证家族树逻辑一致性	📝 待开发

5.3 政治社会关系（05c3）

脚本	功能	状态
`kg/relations/scripts/extract_political_relations.py`	提取政治关系	📝 待开发
`kg/relations/scripts/extract_social_relations.py`	提取社会关系	📝 待开发

六、产出数据

6.1 共现层数据

文件	说明
`kg/relations/person_cooccurrence.json`	人物共现统计
`kg/relations/person_cooccurrence_pmi.json`	人物共现强度（PMI）
`kg/relations/candidate_relations.json`	候选关系对（高频/高PMI）

6.2 家族层数据

A. 帝王家谱（已完成）

文件	说明	状态
`kg/genealogy/data/五帝朝帝王家谱.md`	五帝朝帝王世系（38位，5对关系）	✅ 已生成
`kg/genealogy/data/夏朝帝王家谱.md`	夏朝帝王世系	✅ 已生成
`kg/genealogy/data/商朝帝王家谱.md`	商朝帝王世系	✅ 已生成
`kg/genealogy/data/周朝帝王家谱.md`	周朝帝王世系	✅ 已生成
`kg/genealogy/data/秦朝帝王家谱.md`	秦朝帝王世系	✅ 已生成
`kg/genealogy/data/秦末朝帝王家谱.md`	秦末帝王世系	✅ 已生成
`kg/genealogy/data/汉朝帝王家谱.md`	汉朝帝王世系	✅ 已生成
`kg/genealogy/imperial_genealogy.json`	帝王家谱汇总（JSON）	✅ 已生成

B. 诸侯邦国家谱（待构建）

邦国	文件	说明	状态
齐		姜姓齐国君主世系（世家032）	

邦国	文件	说明	状态
	kg/genealogy/data/齐国家谱.md		📝 待生成
楚	kg/genealogy/data/楚国家谱.md	芈姓楚国君主世系（世家040）	📝 待生成
燕	kg/genealogy/data/燕国家谱.md	姬姓燕国君主世系（世家034）	📝 待生成
赵	kg/genealogy/data/赵国家谱.md	嬴姓赵国君主世系（世家043）	📝 待生成
魏	kg/genealogy/data/魏国家谱.md	姬姓魏国君主世系（世家044）	📝 待生成
韩	kg/genealogy/data/韩国家谱.md	姬姓韩国君主世系（世家045）	📝 待生成
晋	kg/genealogy/data/晋国家谱.md	姬姓晋国君主世系（世家039）	📝 待生成
宋	kg/genealogy/data/宋国家谱.md	子姓宋国君主世系（世家038）	📝 待生成
卫	kg/genealogy/data/卫国家谱.md	姬姓卫国君主世系（世家037）	📝 待生成
鲁	kg/genealogy/data/鲁国家谱.md	姬姓鲁国君主世系（世家033）	📝 待生成
吴	kg/genealogy/data/吴国家谱.md	姬姓吴国君主世系（世家031）	📝 待生成
越			

邦国	文件	说明	状态
	kg/genealogy/data/越国家谱.md	姒姓越国君主世系（世家041）	📝 待生成
陈	kg/genealogy/data/陈国家谱.md	妫姓陈国君主世系（世家036）	📝 待生成
郑	kg/genealogy/data/郑国家谱.md	姬姓郑国君主世系	📝 待生成
蔡	kg/genealogy/data/蔡国家谱.md	姬姓蔡国君主世系	📝 待生成

C. 重要家族家谱（待构建）

家族	文件	说明	状态
孔氏	kg/genealogy/data/孔氏家谱.md	孔子家族世系（列传047）	📝 待生成
孟尝君家族	kg/genealogy/data/孟尝君家族.md	田文家族世系（列传075）	📝 待生成
春申君家族	kg/genealogy/data/春申君家族.md	黄歇家族世系（列传078）	📝 待生成
平原君家族	kg/genealogy/data/平原君家族.md	赵胜家族世系（列传076）	📝 待生成
信陵君家族	kg/genealogy/data/信陵君家族.md	魏无忌家族世系（列传077）	📝 待生成
汉初功臣	kg/genealogy/data/汉初功臣家谱.md	萧何、韩信、张良等家族	📝 待生成

D. 通用家族数据

文件	说明	状态
`kg/relations/family_relations.json`	全部家庭关系三元组	📝 待生成
`kg/relations/family_relations_inferred.json`	推理后的家庭关系	📝 待生成
`kg/relations/family_tree.json`	家族树（DAG图）	📝 待生成
`kg/genealogy/state_genealogies.json`	诸侯邦国家谱汇总	📝 待生成
`kg/genealogy/clan_genealogies.json`	重要家族家谱汇总	📝 待生成

6.3 政治社会层数据

文件	说明
`kg/relations/political_relations.json`	政治关系三元组
`kg/relations/social_relations.json`	社会关系三元组

七、SPO三元组输出

人物关系以主-谓-宾（SPO）三元组形式结构化存储：

家族关系示例：

```
{
  "subject": "黄帝",
  "predicate": "父子",
  "object": "玄嚣",
  "evidence": [
    {
      "chapter": "001",
      "section": "[1.1]",
      "text": "黄帝居轩辕之丘...生二子，其后皆有天下：其一曰玄嚣...",
      "keywords": ["生", "子"]
    }
  ],
  "inference_method": "rule-based",
  "confidence": 0.95,
  "direct": true
}
```

政治关系示例：

```
{
  "subject": "秦始皇",
  "predicate": "君臣",
  "object": "李斯",
  "chapter": "006",
  "evidence": "[6.2] 秦始皇以李斯为丞相...",
  "confidence": 0.90
}
```

八、与其他工序的关系

上游	本工序	下游
	人物关系构建	06a 本体构建（关系本体化）

上游	本工序	下游
03a 实体标注（人物识别）		
03b 实体消歧（唯一指称）	→ 消歧后准确建关系	07 逻辑推理（家族推理、矛盾检测）
05a 事件关系（事件共现）	→ 事件参与者关系补充	09a 阅读器（关系网络可视化）
05b 实体关系（地点/制度）	→ 人地关系（封地/战场）	

九、实施路线图

Phase 1: 共现关系抽取（05c1）

- [ ] 实现段落级滑动窗口共现抽取
- [ ] 计算PMI/Jaccard强度
- [ ] 生成候选关系对

Phase 2: 家族关系推理（05c2）

A. 帝王家谱（已完成）
- [x] 提取帝王家谱（已完成：7个朝代）
- [x] 提取家族关系基础数据（已完成）

B. 诸侯邦国家谱（待完成）
- [ ] 提取齐国君主世系（姜姓齐国，世家032）
- [ ] 提取楚国君主世系（芈姓楚国，世家040）
- [ ] 提取燕/赵/魏/韩/晋/宋/卫/鲁/吴/越/陈/郑/蔡等国家谱
- [ ] 合并诸侯邦国家谱汇总

C. 重要家族家谱（待完成）
- [ ] 提取孔氏家谱（列传047）

- [ ] 提取战国四公子家谱（孟尝/春申/平原/信陵）
- [ ] 提取汉初功臣家谱（萧何/韩信/张良等）

D. 家族关系推理（待完成）
- [ ] 实现传递推理（祖孙/叔侄/堂兄弟/曾祖）
- [ ] 构建家族树DAG图（帝王+诸侯+重要家族）
- [ ] 验证家族树逻辑一致性（无环/性别/时代一致性）
- [ ] 可视化家族树（Graphviz/D3.js）

Phase 3: 政治社会关系（05c3）

- [ ] 提取君臣关系
- [ ] 提取师徒关系
- [ ] 提取敌对/友人关系
- [ ] LLM辅助复杂关系抽取

Phase 4: 整合与验证

- [ ] 合并三层关系数据
- [ ] 交叉验证关系一致性
- [ ] 生成人物关系网络可视化

十、质量控制

10.1 共现关系质量控制

指标	阈值	说明
共现频次	≥ 2	过滤偶然共现
PMI值	≥ 3.0	保留强关联
Jaccard系数	≥ 0.2	段落重叠度

10.2 家族关系质量控制

验证规则	说明
无环检测	家族树不应有循环（A→B→C→A）
性别一致性	父亲必为男性，母亲必为女性
时代一致性	父辈早于子辈（黄帝不能是颜回的父亲）
单一父母	每人最多一父一母（不能有两个生父）
兄弟对称性	A是B的兄弟 → B是A的兄弟

10.3 政治社会关系质量控制

验证规则	说明
时序一致性	君臣关系需在同一时代
角色一致性	师徒关系中师早于徒
证据充分性	每条关系需有明确文本证据

SKILL 05d: 事实发现 — 原子事实三元组抽取

> 将《史记》文本分解为最小可验证单元：每个事实是 1–3 个三元组（主–谓–宾），可独立成立、可独立引用、可独立核验。全书约十万规模。

一、什么是事实

1.1 定义

事实（Fact） = 一个原子陈述，用 1–3 个三元组（Subject–Predicate–Object）完整表达，来源于《史记》单句或单个从句。

核心特征：
- **原子性**：不可再拆分为更小的有意义单元
- **可引用性**：有明确的来源句，可回溯原文
- **可核验性**：可独立判断真/假/存疑
- **无叙事依赖**：不需要上下文就能理解其含义

1.2 与其他层的区别

层级	颗粒度	示例	数量级
实体	命名对象	〖@秦始皇〗、〖=咸阳〗	15,190
事实	原子陈述（1–3个三元组）	秦始皇于前210年死亡	**~100,000**
事件	叙事单元（多参与者+上下文）	沙丘之变（政变过程）	3,185
关系	持久性实体间结构	秦始皇↔胡亥（父子）	~10,000

事实 vs 事件：
事件是有叙事结构的复合单元（有起因、经过、影响），包含多个事实。
"沙丘之变"是事件；"秦始皇崩于沙丘"、"赵高矫诏"、"扶苏自杀"是三个独立事实。

事实 vs 关系：
关系是无时间戳的持久结构（父子、君臣）；事实是有时态的原子陈述（"前210年封X为Y"）。
"秦始皇是胡亥之父"是关系；"秦始皇于前247年即位"是事实。

事实 vs 姓氏推理（07b）：
姓氏是事实的一个子类——人物属性事实。

1.3 典型示例

原文	事实	三元组
武王崩	周武王死亡	(周武王, 死亡, ?)
孝公封商鞅商地十五邑	秦孝公封商鞅商地	(秦孝公, 封赏, 商鞅), (商鞅, 封地, 商地十五邑)
赵括代廉颇将	赵括接替廉颇为将	(赵括, 替代, 廉颇), (赵括, 担任, 赵国将领)
坑杀降卒四十余万	长平降卒被杀数量	(白起, 坑杀, 赵降卒), (赵降卒数量, =, 40万)
屈原者，名平，楚之同姓也	屈原之名与姓族	(屈原, 名, 平), (屈原, 姓族, 芈姓楚国)

二、设计原则

SPO 全用汉字，不设 schema。

- 主语（S）、谓语（P）、宾语（O）均直接用原文汉字或最简规范化汉字表达
- 不定义谓词枚举、不定义类型代码

- 同一类事实自然会收敛到相似的谓语（"崩"/"薨"/"卒"/"立"/"封"等），但这是归纳出来的，不是预先规定的
- 特殊情况：O 可为 null（无宾语的动词，如"崩"）；多个三元组共享同一来源时，作为同一事实的多条记录

这样做的好处：
- 不需要维护谓词本体；
- 《史记》语言本身就是最好的谓语来源；
- 后期可以从数据中归纳谓语聚类，而不是从设计中强加分类。

三、数据格式

3.1 单条事实

```
{
  "id": "004-f0023",
  "s": "周武王",
  "p": "崩",
  "o": null,
  "ctx": {
    "time": -1043,
    "place": null
  },
  "src": "004_周本纪",
  "text": "武王崩，太子诵代立，是为成王。",
  "conf": "high",
  "event": "004-e031"
}
```

`ctx`（context）记录事实成立时的背景条件，与 SPO 本身无关。时间和地点是最常见的 context，但不限于此——凡是"在什么情况下"的信息都放在 ctx 里。

更多示例：

```
{ "id": "005-f0011", "s": "秦孝公", "p": "封",   "o": "商鞅",
  "ctx": { "time": -340, "place": null },
  "src": "005_秦本纪", "text": "孝公封商鞅商地十五邑。" }

{ "id": "005-f0012", "s": "商鞅",   "p": "封地", "o": "商地十五邑",
  "ctx": { "time": -340, "place": null },
  "src": "005_秦本纪", "text": "孝公封商鞅商地十五邑。" }

{ "id": "073-f0041", "s": "白起",   "p": "坑杀", "o": "赵降卒四十余
万",
  "ctx": { "time": -260, "place": "长平" },
  "src": "073_白起王翦列传", "text": "坑杀降卒四十余万。" }

{ "id": "007-f0088", "s": "项羽",   "p": "破釜沉舟", "o": null,
  "ctx": { "time": -207, "place": "漳河南", "occasion": "救赵攻
秦" },
  "src": "007_项羽本纪", "text": "项羽乃悉引兵渡河，皆沉船，破釜甑。" }

{ "id": "084-f0001", "s": "屈原",   "p": "名",   "o": "平",
  "ctx": {},
  "src": "084_屈原贾生列传", "text": "屈原者，名平，楚之同姓也。" }
```

ctx　可扩展任意汉字键值，不预设字段枚举。常见键：`time`（公元纪年）、`place`（地点）、`occasion`（场合/背景）、`reign`（在位年号/君主）。

3.2 字段说明

字段	说明	必填
`id`	`{章节}-f{序号}`，全库唯一	是
`s`	主语，汉字规范名	是
`p`	谓语，直接用汉字（崩/封/名/坑杀…）	是
	宾语，汉字或 null	否

字段	说明	必填
`o`		
`ctx`	事实成立的背景条件（时间/地点/场合等），可为 `{}`	是
`ctx.time`	公元纪年（负数=公元前）	推荐
`ctx.place`	发生地点，汉字	推荐
`ctx.*`	其他背景键值，自由扩展，汉字	可选
`src`	来源章节	是
`text`	原始句子（≤50字）	是
`conf`	exact / high / medium / low	是
`event`	关联事件 ID，可选	否

3.3 数据文件

```
kg/facts/
  data/
    {章节}_事实索引.json     # 每章一个文件（类比事件索引）
    fact_index.json          # 全库汇总索引
  scripts/
    extract_facts.py         # 抽取脚本
    validate_facts.py        # 验证脚本
    build_fact_index.py      # 汇总索引生成
```

四、规模估算

来源	估算方式	事实数
十表（013–022）	每行1–3个事实，共约5,000行	~10,000–15,000
本纪（001–012）	密度最高，每章约1,000–1,500事实	~12,000–18,000
世家（031–060）	每章约600–1,000事实	~18,000–30,000
列传（061–130）	每章约400–800事实	~28,000–56,000
全库合计		**~70,000–120,000**

> 十万规模是合理预期。密度：本纪>世家>列传>书。

五、抽取方法

5.1 多轮迭代（类比姓氏推理）

```
Round 1：规则抽取
  - 高信号动词句式：立/薨/封/拜/以X为Y/名X/字X 等
  - 数字+量词正则：N万/N人/N年/N里
  - P 直接取原文动词，不映射到枚举
  → 覆盖率约40–60%，精度高

Round 2：句法框架抽取
  - 施事-动词-受事 框架
  - "以X为Y"结构（任命模式）
  - "X之Y"结构（归属/属性模式）
  → 覆盖更多隐式事实，精度中

Round 3：LLM批量抽取
  - 输入：段落文本 + 已知实体列表
```

```
  - 输出：事实三元组 + 置信度 + 原文依据
  - 质量控制：与 Round 1/2 结果交叉验证
  → 覆盖语义复杂案例，精度取决于模型

Round 4：验证与去重
  - 同一事实在多章出现 → 保留最高置信度，记录多源
  - 与事件索引、关系索引交叉核验
  - 数字类事实：与 SKILL_07 矛盾检测联动
```

5.2 抽取优先级

1. **十表（013–022）**：结构化最强，规则抽取可覆盖80%+
2. **本纪（001–012）**：密度最高，投入产出比最好
3. **世家（031–060）**：人物传承事实丰富（即位/薨/封）
4. **列传（061–130）**：细节丰富但叙事较散

六、与其他工序的关系

上游	本工序	下游
03a 实体标注（提供规范实体名）	事实抽取，建立 fact_index.json	07 逻辑推理（数字矛盾检测的输入）
04 事件构建（事件→分解为事实）	事实←→事件双向链接	07b 姓氏推理（人物属性事实）
05b 实体关系（关系←→事实互补）	与关系区分：有时态→事实，无时态→关系	08 SKU（事实作为知识单元最小原子）
04b 十表事件处理	十表是事实密度最高的来源	09 应用（事实问答、核验功能）

七、下一步

- [ ] 确定 `kg/facts/` 目录结构和 `fact_id` 命名规范
- [ ] 从十表（013–022）试点：规则脚本抽取立/薨/封/拜类句式
- [ ] 建立事实←→事件 ID 的双向链接规范
- [ ] 评估与 05b 实体关系的边界：哪些关系退化为事实，哪些事实升格为关系

SKILL 05e: 战争事件识别与知识融合

1. 任务定义

从已提取的3,185个历史事件中识别、提取并融合所有战争相关事件，构建统一的战争事件知识库。

输入:
- 130章的事件索引文件（`kg/events/data/*_事件索引.md`）
- 标注文本（`chapter_md/*.tagged.md`，用于补充细节）

输出:
- 战争事件索引（`kg/events/data/战争事件索引.md`）
- 战争事件JSON数据（`kg/events/data/wars.json`）
- 跨章战争关联表（`kg/events/data/war_links.md`）

---

2. 战争事件的定义

什么是战争事件

战争事件是指涉及**军事征伐、战役、平叛、攻城、围困**等武装冲突的历史事件。包括:

- **国家间战争**: 灭国之战、兼并战争、边境冲突
- **内部平叛**: 叛乱镇压、诸侯征伐
- **民族战争**: 华夷战争、少数民族征讨
- **军事行动**: 包围、袭击、救援、退兵

不属于战争事件

- **政治暗杀**: 无军事对抗的刺杀（如荆轲刺秦王）
- **武力威慑**: 陈兵示威但未发生战斗
- **军事演习**: 非实战的练兵活动

- **个人决斗**: 非军事组织的武力冲突

简略攻伐记录（Fact级别）

史记中大量存在极简略的攻伐记录，如:
- "攻魏"、"伐赵"（仅2-3字）
- "取某城"、"围某地"（无战争经过描述）
- 年表中的"X攻Y"条目（无详细叙述）

处理策略:

记录类型	建模方式	说明
有完整叙述的战争	Event（战争事件）	本skill（05e）处理
简略攻伐记录	Fact（事实）	由SKILL_05d处理
融合	Fact关联到Event	本skill将简略fact与战争事件关联

融合规则:

当简略fact与战争事件可能相关时，建立关联:

```
### WAR-042 长平之战
- **关联Facts**:
  - FACT-043-025: %四十五年%，攻赵长平（赵世家）
  - FACT-005-087: %昭王四十五年%，伐韩野王（秦本纪）
```

这样既保留了简略记录的事实性，又将其与完整战争事件关联起来。

---

3. 战争事件关键信息Schema

3.1 核心字段

字段名	说明	示例
war_id	战争统一ID	`WAR-001`
war_name	战争名称（4-8字）	涿鹿之战、长平之战
war_type	战争类型	国家战争/平叛/民族战争/兼并战争
time_range	时间范围	前260年—前260年（单次战役） 前230年—前221年（系列战争）
duration	持续时间	三年、数月、一日
location	主要战场	=长平=、=涿鹿之野=
belligerents	交战双方	
belligerents.attacker	进攻方	&秦&
belligerents.defender	防守方	&赵&
commanders	指挥官	
commanders.attacker	进攻方将领	@白起@
commanders.defender	防守方将领	@赵括@
scale	战争规模	兵力、死伤人数
outcome	战争结果	秦胜/赵败/双方议和

字段名	说明	示例
casualties	伤亡情况	赵军坑杀四十万、秦军死伤过半
consequences	战争影响	赵国元气大伤、秦统一进程加速
sources	来源章节	005-034, 043-012, 073-005
conflicts	矛盾记载	不同章节对伤亡数字的差异
related_facts	关联的简略事实	FACT-043-025, FACT-005-087

3.2 可选补充字段

字段名	说明	示例
trigger	战争起因	赵孝成王中秦反间计
strategy	战术策略	白起坑杀降卒、韩信背水一阵
turning_point	转折点	赵括代廉颇、秦军断赵粮道
related_events	相关事件	上党之争（前期）、邯郸之战（后续）
historical_assessment	史评	太史公曰中的评价

---

4. 战争事件提取流程

4.1 第一阶段：初步识别

从130章事件索引中筛选所有`事件类型 = 战争`的事件：

```
# 提取所有战争事件
grep "| 战争 |" kg/events/data/*_事件索引.md > kg/events/tmp/
wars_raw.txt
```

```
# 统计数量
wc -l kg/events/tmp/wars_raw.txt
```

预期结果: 约200-300个战争事件条目（需验证）

4.2 第二阶段：事件归并

同一战争在不同章节可能被多次提及，需要识别并归并：

自动归并规则

归并条件	示例
战争名称完全相同	"长平之战" = "长平之战"
时间+地点一致	前260年 + =长平= → 同一战争
交战双方+时间一致	秦vs赵 + 前260年 → 同一战争

LLM辅助归并

对于模糊情况，使用LLM判断：

```
输入：
- 事件A：003-015 商汤伐桀
- 事件B：004-012 桀亡国于鸣条

问题：这两个事件是否描述同一场战争？

输出：
- 是/否
- 理由：两者都描述商汤灭夏的最终战役，时间、人物、结果一致。003章侧重商汤视角，004章侧重夏桀视角。应归并为"鸣条之战"。
```

4.3 第三阶段：信息融合

将同一战争的多个来源记录融合为统一条目：

融合策略

信息类型	融合规则
战争名称	优先采用最常用名称
时间	取最精确的时间标注
地点	合并所有提及的战场
指挥官	合并所有提及的将领（标注来源）
规模/伤亡	保留所有不同说法，标注来源
结果	若有矛盾，全部保留并标注
战争经过	按时间顺序整合叙述

矛盾处理

不同章节对同一战争的记载可能矛盾，处理原则：

```
### WAR-042 长平之战
- **伤亡情况**:
  - 《白起王翦列传》(073-005): 坑杀赵卒四十五万
  - 《赵世家》(043-012): 前后死者四十万
  - **矛盾说明**: 两说相差5万，可能因统计口径不同（是否含战死者）
```

4.4 第四阶段：细节补充

从原文中补充关键细节：

- 回到标注文本（`*.tagged.md`）查找战争描述段落
- 提取战术、转折点、名言等细节

· 补充太史公曰中的战争评价

---

5. 战争类型体系

5.1 按性质分类

类型	说明	典型例子
国家战争	诸侯国间的正式战争	长平之战、邲之战
兼并战争	以灭国为目的的战争	秦灭六国、楚灭鲁
平叛战争	镇压叛乱	七国之乱、陈胜吴广起义
民族战争	华夏与周边民族	汉匈战争、楚伐濮
统一战争	改朝换代的系列战争	秦灭六国、汉灭项羽
内战	同一政权内部	楚国白公之乱、晋国内斗

5.2 按规模分类

规模	标准	典型例子
特大	参战10万+或改变历史格局	长平之战、巨鹿之战
大型	参战5-10万或影响国运	城濮之战、马陵之战
中型	参战1-5万或区域性影响	鄢陵之战
小型	局部冲突	边境小规模战斗

5.3 按时代分类

时代	时间范围	战争特点
上古	前2700—前1600年	神话传说色彩浓厚（涿鹿之战）
三代	前1600—前771年	宗法制下的征伐（武王伐纣）
春秋	前770—前476年	礼仪战争、车战为主
战国	前475—前221年	兼并战争、步兵崛起
秦汉	前221—前87年	统一战争、汉匈战争

---

6. 输出格式规范

6.1 战争事件索引文件结构

```
# 史记战争事件索引

## 统计概览

- **总战争数**: N
- **已归并**: M
- **单一来源**: X
- **多源融合**: Y
- **时间跨度**: 前2690年（阪泉之战）— 前119年（漠北之战）

---

## 按时代索引

### 上古时代战争（前2700—前1600年）
```

```
| 战争ID | 战争名称 | 时间 | 交战方 | 结果 | 来源章节数 |
|--------|---------|------|--------|------|-----------|
| WAR-001 | 阪泉之战 | [约前2690年] | 轩辕 vs 炎帝 | 轩辕胜 | 1
(001-002) |
| WAR-002 | 涿鹿之战 | [约前2679年] | 黄帝 vs 蚩尤 | 黄帝胜 | 1
(001-003) |
| ... | ... | ... | ... | ... | ... |

### 夏商西周战争（前1600—前771年）

...

### 春秋战争（前770—前476年）

...

### 战国战争（前475—前221年）

...

### 秦汉战争（前221—前87年）

...

---

## 详细战争记录

### WAR-001 阪泉之战

#### 基本信息
- **战争ID**: WAR-001
- **战争名称**: 阪泉之战
- **战争类型**: 部落战争
- **时间**: [约公元前2690年]
- **地点**: =阪泉之野=
- **持续时间**: 三战（具体时长不详）
```

```
#### 交战双方
- **进攻方**: @轩辕@
- **防守方**: @炎帝@

#### 战争起因
炎帝欲侵陵诸侯，诸侯归附轩辕。

#### 战争经过
轩辕修德振兵，治五气，种五谷，抚万民，训练熊罴貔貅貙虎等军队，与炎帝战于阪泉之野，三战后得其志。

#### 战争结果
- **胜负**: 轩辕胜
- **影响**: 轩辕确立霸主地位，为统一诸侯奠定基础

#### 来源章节
- **001-002** (五帝本纪): "@炎帝@欲侵陵诸侯，诸侯咸归@轩辕@...三战,然後得其志。"

#### 年代推断
阪泉之战发生于神农氏衰落、黄帝即位初期，叙事上紧接黄帝成长（001-001，前2698年）。"诸侯归轩辕"说明黄帝已有一定威望，在位数年后方能发动此战。推算为即位后约8年，即前2690年。

---

### WAR-042 长平之战（多源融合示例）

#### 基本信息
- **战争ID**: WAR-042
- **战争名称**: 长平之战
- **战争类型**: 国家战争
- **时间**: (公元前260年)
- **地点**: =长平=
- **持续时间**: 数月
```

```
#### 交战双方
- **进攻方**: &秦&
  - **主将**: @白起@（秦将）
  - **参谋**: @范雎@（丞相）
- **防守方**: &赵&
  - **初期主将**: @廉颇@
  - **后期主将**: @赵括@（替代廉颇）
  - **君主**: @赵孝成王@

#### 战争起因
秦攻打韩国上党，上党守将降赵。秦因此攻赵，争夺上党郡。

#### 战争经过
1. **初期**: 廉颇坚守不出，秦军久攻不下
2. **转折**: 秦使反间计，赵王中计换将赵括
3. **决战**: 赵括出击，秦军佯败，分兵断赵军粮道
4. **结局**: 赵军被围四十六日，赵括突围战死，赵军投降

#### 战争结果
- **胜负**: 秦胜，赵惨败
- **伤亡**:
  - 《白起王翦列传》(073-005): 坑杀赵卒四十五万
  - 《赵世家》(043-012): 前后死者四十万
  - **矛盾说明**: 两说相差5万，可能因统计口径不同
- **影响**: 赵国元气大伤，东方六国再无力抗秦

#### 战术要点
- **秦军策略**: 反间计、分割包围、断粮困敌
- **赵军失误**: 轻敌冒进、粮道被断

#### 来源章节
- **005-034** (秦本纪): 概述秦军胜利
- **043-012** (赵世家): 详述赵军失败经过
- **073-005** (白起王翦列传): 白起视角，详述战术
- **081-015** (廉颇蔺相如列传): 廉颇被换将经过

#### 太史公评价
```

```
（引用太史公曰中对此战的评述）

---

...

## 附录

### A. 战争名称索引（按拼音排序）

- 阪泉之战 → WAR-001
- 长平之战 → WAR-042
- ...

### B. 人物参战索引

#### @白起@
- WAR-042 长平之战（主将）
- WAR-038 伊阙之战（主将）
- ...

#### @项羽@
- WAR-085 巨鹿之战（主将）
- WAR-092 彭城之战（主将）
- ...

### C. 国家战争索引

#### &秦&
- WAR-042 长平之战（vs赵）
- WAR-065 秦灭韩（vs韩）
- ...

### D. 战争时间线
```

前2690年 —— WAR-001 阪泉之战
前2679年 —— WAR-002 涿鹿之战
…
前260年 —— WAR-042 长平之战
…
前119年 —— WAR-150 漠北之战（史记最后一场大战）

6.2 JSON数据结构

```
{
  "wars": [
    {
      "war_id": "WAR-042",
      "war_name": "长平之战",
      "war_type": "国家战争",
      "time": {
        "start": "-260",
        "end": "-260",
        "certainty": "precise",
        "original": "(公元前260年)"
      },
      "location": ["长平"],
      "belligerents": {
        "attacker": {
          "state": "秦",
          "commanders": ["白起"],
          "advisors": ["范雎"]
        },
        "defender": {
          "state": "赵",
          "commanders": ["廉颇", "赵括"],
          "ruler": "赵孝成王"
        }
      },
      "scale": {
```

```
        "troops": "数十万",
        "casualties": [
          {
            "source": "073-005",
            "description": "坑杀赵卒四十五万"
          },
          {
            "source": "043-012",
            "description": "前后死者四十万"
          }
        ]
      },
      "outcome": {
        "victor": "秦",
        "description": "赵军投降被坑杀"
      },
      "sources": [
        "005-034",
        "043-012",
        "073-005",
        "081-015"
      ],
      "conflicts": [
        {
          "field": "casualties",
          "descriptions": ["四十五万 (073)", "四十万 (043)"],
          "note": "统计口径可能不同"
        }
      ]
    }
  ]
}
```

7. 执行流程

7.1 脚本化提取

```
# 1. 初步提取所有战争事件
python kg/events/scripts/extract_wars.py --stage=identify

# 2. 自动归并（基于规则）
python kg/events/scripts/extract_wars.py --stage=merge_auto

# 3. LLM辅助归并（处理模糊情况）
python kg/events/scripts/extract_wars.py --stage=merge_llm

# 4. 信息融合
python kg/events/scripts/extract_wars.py --stage=consolidate

# 5. 细节补充
python kg/events/scripts/extract_wars.py --stage=enrich

# 6. 生成JSON和索引文件
python kg/events/scripts/extract_wars.py --stage=export

# 完整流程
python kg/events/scripts/extract_wars.py --all
```

7.2 质量验证

```
# 验证战争事件完整性
python kg/events/scripts/validate_wars.py

# 生成统计报告
python kg/events/scripts/war_statistics.py
```

---

8. 质量标准

8.1 归并质量

- [ ] 同一战争的多个来源已识别并归并
- [ ] 战争名称统一（无"长平之战"和"长平大战"并存）
- [ ] 无明显遗漏（如重大战役未被提取）

8.2 信息完整性

每个战争事件应具备：

- [ ] 唯一war_id
- [ ] 明确的战争名称
- [ ] 时间标注（精确或范围）
- [ ] 交战双方
- [ ] 战争结果
- [ ] 至少一个来源章节
- [ ] 详细的事件描述

8.3 矛盾记录

对于多源融合的战争：

- [ ] 已标注所有来源章节
- [ ] 矛盾之处已明确记录
- [ ] 不同说法都已保留
- [ ] 对矛盾有合理解释或说明

---

9. 预期产出

9.1 数量预估

指标	预估	说明
原始战争事件	200-300	从130章提取的战争类型事件
归并后战争数	120-180	去重后的独立战争
多源战争	50-80	在多个章节被记载的重要战争
单源战争	70-100	仅在一章提及的小规模战争

9.2 覆盖时间

- **最早**: 约前2690年（阪泉之战）
- **最晚**: 约前119年（漠北之战）
- **跨度**: 2571年

9.3 产出文件

1. **战争事件索引**（`kg/events/data/战争事件索引.md`）
   - 按时代分类的战争列表
   - 每场战争的详细记录
   - 附录索引（人物、国家、时间线）
2. **JSON数据**（`kg/events/data/wars.json`）
   - 结构化战争数据
   - 便于程序化分析和可视化
3. **跨章关联表**（`kg/events/data/war_links.md`）
   - 同一战争的多个来源映射
   - 便于溯源和验证

---

10. 扩展应用

10.1 战争地图可视化

基于地点信息，在地图上标注战争位置：

- 时间轴动画展示战国兼并过程
- 不同颜色标识战争类型和规模

10.2 战争网络分析

- **国家战争关系图**：哪些国家之间战争最频繁
- **将领战绩图**：名将参与的所有战争
- **战争因果链**：一场战争如何引发下一场

10.3 战争知识问答

基于结构化数据，支持问答：

```
Q: 秦国在统一过程中打了哪些重要战役?
A: 长平之战(前260)、伊阙之战(前293)、华阳之战(前273)...

Q: 白起一生参与了哪些战争?
A: 长平之战、伊阙之战、郢都之战...共歼敌百万余

Q: 史记中伤亡最惨重的战争是哪场?
A: 长平之战，赵军四十万被坑杀
```

10.4 跨文本对比

将史记战争数据与其他史书（战国策、汉书等）对比，发现记载差异。

---

11. 已知难点与解决方案

11.1 战争名称不统一

问题: 同一战争在不同章节可能用不同名称:
- "长平之战" vs "秦赵长平之役"
- "鸿沟之约" vs "楚汉相约中分天下"

方案:
- 建立战争名称同义词表
- LLM辅助识别同一战争的不同称谓
- 采用最常用名称作为主名称，其他作为别名

11.2 系列战争的粒度

问题: "秦灭六国"是一个战争还是六个?

方案:
- **战役级**: 每场独立战役为一个事件（秦灭韩、秦灭赵...）
- **战争级**: 系列战役合并为大战争（秦统一战争）
- 两个层级都保留，建立父子关系

11.3 时间跨度大的战争

问题: 有些战争持续数年（如汉匈战争），时间标注如何处理?

方案:
- 使用时间范围标注: `（公元前133年–前119年）`
- 将长期战争拆分为多个阶段战役
- 建立总体战争条目 + 具体战役条目的层级结构

11.4 传说与史实混杂

问题: 上古战争（阪泉、涿鹿）史实性存疑。

方案:
- 在 `certainty` 字段标注可信度（legend/uncertain/confirmed）
- 年代标注使用 [约...] 表示不确定性
- 在描述中注明"传说"或"据《史记》记载"

---

12. 后续迭代方向

第一期（MVP）

- [x] 定义战争事件Schema
- [ ] 提取所有战争类型事件
- [ ] 自动归并明显重复
- [ ] 输出基础索引文件

第二期（融合）

- [ ] LLM辅助归并模糊情况
- [ ] 多源信息融合
- [ ] 矛盾记录与解释
- [ ] 补充细节（战术、转折点）

第三期（增强）

- [ ] 建立战争层级体系（战争-战役-战斗）
- [ ] 生成JSON结构化数据
- [ ] 构建战争时间线
- [ ] 人物/国家参战索引

第四期（可视化）

- [ ] 战争地图可视化
- [ ] 战争网络分析
- [ ] 与其他史书对比
- [ ] 知识问答系统

---

13. 参考资源

- **事件提取基础**: SKILL_04a_事件识别.md

- **年代标注**: SKILL_04c_事件年代推断.md
- **事件关系**: SKILL_05a_事件关系发现.md
- **实体关联**: SKILL_05b_实体关系构建.md

---

14. 与其他Skill的协作

14.1 与SKILL_05d（事实抽取）的协作

分工:

Skill	处理对象	粒度
05e	有完整叙述的战争	Event级别（战争事件）
05d	简略攻伐记录	Fact级别（事实三元组）

工序位置:

本skill（05e）位于以下工序之后:
- **04e 年份消歧** → 才能准确判断同一战争（如"前260年"统一后才能归并）
- **05d 事实发现** → 才能关联简略攻伐记录

协作流程:

```
04e 年份消歧完成
    ↓
05d 事实发现（提取简略攻伐）
    → 输出: facts.json（可能500+条攻伐记录）
    ↓
05e 战争事件识别（本skill）
    → 从事件索引中提取战争
    → 输出: wars.json（736个战争）
    → 关联facts: 将时间、地点、交战方相近的fact关联到war
    → 输出: wars_enriched.json（战争+关联facts）
```

关联示例:

```
{
  "war_id": "WAR-042",
  "war_name": "长平之战",
  "time": {"start": "-260", "end": "-260"},
  "sources": ["005-034", "043-012", "073-005"],
  "related_facts": [
    {
      "fact_id": "FACT-043-025",
      "triple": ["秦", "攻", "长平"],
      "time": "赵孝成王四十五年",
      "source": "043_赵世家"
    },
    {
      "fact_id": "FACT-005-087",
      "triple": ["秦", "伐", "野王"],
      "time": "昭王四十五年",
      "source": "005_秦本纪",
      "note": "长平之战前期，切断上党"
    }
  ]
}
```

14.2 与SKILL_05a（事件关系发现）的协作

战争事件之间存在丰富的关系:

- **因果关系**: 长平之战 → 赵国衰弱 → 邯郸之战
- **时序关系**: 伊阙之战(前293) → 长平之战(前260) → 邯郸之战(前259)
- **对立关系**: 秦军视角 vs 赵军视角

这些关系由SKILL_05a处理，战争事件作为其输入。

14.3 与SKILL_05b（实体关系构建）的协作

战争事件是实体关系的重要来源：

- **人物-参与-战争**: @白起@ → 参与 → WAR-042（长平之战）
- **国家-交战-国家**: &秦& → 交战 → &赵&（在长平之战中）
- **地点-发生-战争**: =长平= → 发生 → WAR-042

这些关系由SKILL_05b提取并纳入知识图谱。

---

15. 适用范围

本方法适用于：

- 纪传体史书中的战争事件提取
- 编年体史书的战争记录融合
- 多源历史文献的战争信息整合

关键前提：

1. 已完成基础事件提取（有事件类型标注）
2. 有明确的事件来源标注（章节+事件ID）
3. 事件描述中包含足够信息供归并判断

附录：待创建的SKILL_05d预览

> SKILL_05d（事实抽取）将处理简略攻伐记录，与本skill互补。

SKILL_05d核心内容：

- 从原文中提取简略的动作-对象模式（"攻X"、"伐Y"、"取Z城"）
- 建模为事实三元组：(主体, 动词, 客体)
- 标注时间、地点元信息
- 与战争事件关联（时间、地点、交战方匹配）

· 输出：facts.json、fact_war_links.json

示例：

```
FACT-015-234
- **三元组**：（秦，攻，长平）
- **时间**：%昭王四十五年%（前262年）
- **来源**：015_六国年表
- **关联战争**：WAR-042（长平之战）
```

SKILL 06: 本体构建 — 从实体实例到分类本体

> 从标注实体中归纳分类层次，构建OWL/RDF本体。

子工序

编号	文件	内容
06a	SKILL_06a_实体到本体.md	实体词表 → 分类树 → OWL/RDF本体

SKILL 06a: 实体到本体管线 — 从标注文本到 RDF 分类本体

> 从《史记》标注实体出发，构建 OWL/RDF 本体的完整工作流。以生物类（388种）为首个实例。

一、管线概览

```
chapter_md/*.tagged.md      标注文本（18类实体标记）
        ↓ build_entity_index.py
kg/entities/data/entity_index.json     实体索引（按类型+出现次数+章节引用）
        ↓ 人工/LLM 分类
kg/{type}/data/{type}_wordlist.json    分类词表（多层分类树 + 歧义标记）
        ↓ 脚本生成
kg/{type}/data/{type}_taxonomy.md      可读分类树（Markdown）
        ↓ 脚本生成
kg/rdf/{type}.ttl                      OWL/RDF 本体（类+实例+属性）
```

核心原则： 本体不是凭空设计的，而是从标注实例中逐层归纳涌现。

二、步骤详解

Step 1. 从实体索引中提取类型词表（含词条简介）

输入：`entity_index.json` 中某个类型的全部条目

操作：按出现次数排序，人工审视词条，归纳分类维度

关键步骤：为每个词条生成简介

词表不应只是词条列表——每个词条应附带一个从上下文自动提取的简介（1-2句话），描述此实体是什么、做了什么。这个简介在后续本体分类阶段极其有用：

```
{
  "韩信": {
    "count": 333,
    "summary": "汉初将领，助刘邦灭项羽，封齐王/楚王，后被吕后杀于长乐宫",
    "chapters": ["008_高祖本纪", "092_淮阴侯列传"]
  },
  "管仲": {
    "count": 77,
    "summary": "齐桓公相，辅佐齐桓公称霸，主张尊王攘夷",
    "chapters": ["032_齐太公世家", "062_管晏列传"]
  }
}
```

为什么简介很重要：

在人物本体构建中，我们遇到了反复的分类困难：
- 韩信（333次）主要出现在高祖本纪 → 按章节分应归"本纪臣子"，但他实际是"将领"
- 司马相如（104次）主要出现在列传 → 按章节分应归"将相"，但他实际是"文学家"
- 赵高（139次）主要出现在秦始皇本纪 → 按章节分应归"本纪臣子"，但他实际是"佞臣/宦官"

如果在词表阶段就有简介 `"汉初将领，封齐王"` / `"辞赋家，作子虚赋"` / `"秦宦官，指鹿为马"`，本体分类就可以直接基于语义而非章节位置。

简介生成方法：

1. 从 `entity_index.json` 提取每个实体的所有出现上下文（±15字）
2. 用 LLM 从上下文中总结一句话简介
3. 或：从章节标题 + 共现的身份词（`〖#〗`）/ 官职词（`〖;〗`）自动拼接

管线升级：词表 → 带简介的词表 → 分类树

```
entity_index.json
    ↓ Step 1a: 提取词表
wordlist.json (词条 + 出现次数)
    ↓ Step 1b: 生成简介（LLM或规则）
```

```
wordlist_with_summary.json (词条 + 次数 + 简介)
    ↓ Step 2: 基于简介归纳分类
taxonomy.md → person.ttl
```

这是从人物本体构建实践中发现的关键方法论改进：**词表阶段的简介投资，在本体分类阶段获得十倍回报**。

输出：`{type}_wordlist.json`（三层结构，可扩展简介字段）

```
{
  "always_{type}": {
    "category_1": ["word_a", "word_b"],
    "category_2": ["word_c", "word_d"]
  },
  "context_dependent": {
    "type_vs_other": { "word_x": "判断说明" }
  },
  "new_candidates": { ... }
}
```

生物类实例：

```
{
  "always_biology": {
    "animals_domestic": ["马", "牛", "羊", "狗", "鸡", "豕"],
    "animals_wild":     ["虎", "鹿", "兔", "狼", "豹"],
    "birds":            ["鸟", "雉", "燕", "鸿鹄"],
    "fish_aquatic":     ["鱼", "鳖", "鼋"],
    "insects":          ["蝗", "蛇", "蜂"],
    "plants_crops":     ["稻", "黍", "稷", "菽", "麻"],
    "plants_trees":     ["桑", "松", "柏", "竹"],
    "plants_herbs":     ["兰", "芝", "椒"]
  },
  "context_dependent": {
    "biology_vs_mythical": { "龙": "天降龙=生物; 龙颜=比喻" },
```

```
    "biology_vs_metaphor": { "虎狼": "虎狼心=比喻; 手搏虎狼=生物" }
  }
}
```

Step 2. 设计分类树（类层次）

类名命名原则：

1. **自明性** — 类名必须一目了然，不需要依赖树结构上下文来理解。
   - ✗ `燕 [7人]` — 是诸侯/燕国？还是臣/春秋/燕？
   - √ `燕国臣子 [7人]` — 清晰无歧义
2. **无歧义** — 同名不能出现在不同父类下。如"齐"在春秋和战国下各出现一次，必须改为"齐国臣子(春秋)"和"齐国臣子(战国)"，或合并。
3. **语义完整** — 子类名应包含父类语义。不要用 `将` `相`，用 `秦国将领` `秦国相臣`。
4. **用中文** — URI 和标签都用中文，避免拼音碰撞（Han=汉≠韩）。

方法：从词表的分类维度出发，构建 3-4 层的类层次。

设计原则（对应 Ontology 101 Step 4）：

1. **自底向上** — 从实例归纳类，不从理论推演类
2. **中间层最丰富** — 顶层（动物/植物）过粗，底层（单个物种）过细，中间层（家畜/野兽/鸟类）承载分类信息
3. **允许集合类** — 古文中"六畜"、"五穀"、"鸟兽"等是独立概念，不归入动物或植物，单列"集合泛称"类
4. **神话类单列** — 龙、凤、麒麟等跨越生物/神话边界，单设"神话动物"子类

生物类分类树：

```
LivingThing (生物)
├── Animal (动物)
│   ├── Domestic (家畜)
│   │   ├── Equine (马属) — 马(102), 驹(9), 骥(5)...
│   │   ├── Bovine (牛属) — 牛(36)
│   │   ├── Ovine (羊属) — 羊(26)
```

```
│   │   ├── Canine（犬属）— 狗(32), 犬(4)
│   │   ├── Poultry（禽属）— 鸡(8)
│   │   └── Swine（猪属）— 彘(5), 豕(2)
│   ├── WildBeast（野兽）— 虎(23), 鹿(19), 熊(8)...
│   ├── Bird（鸟类）— 鸟(19), 雉(6)
│   ├── Aquatic（鱼类水族）— 鱼(33), 鳖(3)
│   ├── Insect（虫类）— 蛇(4), 蝗(3)
│   └── MythicalAnimal（神话动物）— 龙(9), 凤皇(6)
├── Plant（植物）
│   ├── Grain（穀物）— 稻(4), 禾(4)
│   ├── Tree（树木）— 木(23), 桑(7)
│   ├── Herb（草本花卉）— 兰(9), 芝(4)
│   └── FruitVeg（果蔬）— 橘(3), 枣栗(3)
└── Collective（集合泛称）— 草木(14), 鸟兽(10)
```

Step 3. 生成可读分类树（Markdown）

输出：`{type}_taxonomy.md`

脚本从 `entity_index.json` + 分类规则自动生成，格式：

```
## 动物
### 家畜
#### 马属（33 种，162 次）
- 马（102）
- 驹（9）
- 骥（5）
...
```

每种实体列出出现次数，按次数降序排列。

Step 4. 生成 RDF/OWL 本体（Turtle）

输出：`kg/rdf/{type}.ttl`

命名约定：
- 命名空间：`http://memect.cn/baojie/ontologies/2025/1/shiji/biology/`
- 类名：ASCII 英文（`bio:Equine`），中文放 `rdfs:label`

- 实例名：`bio:i_{pinyin}`（如 `bio:i_ma`），中文放 `rdfs:label`
- 属性：`:count`（出现次数）

Turtle 模式：

```
bio:Equine a owl:Class ;
    rdfs:subClassOf bio:Domestic ;
    rdfs:label "马属"@zh, "Equine"@en .

bio:i_ma a bio:Equine ;
    rdfs:label "马"@zh ;
    :count 102 .
```

验证：用 `rdflib` 解析确认语法正确。

```
from rdflib import Graph
g = Graph()
g.parse('kg/rdf/biology.ttl', format='turtle')
print(f'{len(g)} triples, {n_classes} classes, {n_instances}
instances')
```

Step 5. 反思验证（Reflection）

自动分类必须经过反思验证。 初次分类的错误率可能极高——人物帝王类首次分类437人中仅49人正确（88%误分率）。

反思方法：逐类审查

对每个分类类别，列出所有实例，逐一检查是否符合该类的定义：

```
帝王类反思流程：
1. 列出所有被分类为"帝王"的人物（437人）
2. 对每个人，检验分类信号：
   - 是否在帝王名册中？（明确帝王名）
   - 名称是否匹配帝号模式？（X帝/X祖/X王）
```

```
    - 是否为本纪标题人物?
3. 三个信号均不满足 → 判定为误分类
4. 分析误分原因 → 发现规则缺陷 → 修正规则
```

误分根因分析（以帝王类为例）：

误分原因	数量	示例	修正
"主章=本纪"≠"此人是帝王"	384人	韩信主出现在高祖本纪 → 被误分为帝王	改用帝王名册+名称模式
单信号分类不可靠	—	仅凭章节信号，准确率仅12%	多信号叠加

反思产生的分类规则升级：

```
v1（首次）：主章为本纪 → 帝王                准确率 12%
v2（反思后）：帝王名册 ∪ 帝号模式 ∪ 本纪标题人物   准确率 ~95%
```

反思轮次示例（人物本体，10+轮）：

轮次	类	修正	发现
1	帝王	437→49人	"主章=本纪"≠帝王（韩信等臣子被误分），准确率仅12%
2	帝王	49→90人（含子类修正）	秦国诸侯≠秦朝帝王；周公旦是摄政非天子
3	诸侯	609→432人	世家中提到的帝王/诸侯/后妃不应重复归入
4	臣子类	—	"主章世家"≠所属国（韩康子出现在魏世家因三家分晋叙事）

轮次	类	修正	发现
5	臣·强制拆分	20+人叶子类全部拆分	晋国臣子→六卿/世族/宗室/将领/卿大夫/谋臣
6	子类实例审查	多处修正	吴楚臣子混入国王（白公/平王）；汉初诸将混入诸侯（黥布/魏豹）
7	外邦审查	5+人移出	匈奴/大宛西域混入汉人（张骞/赵破奴）
8	中间层设计	引入王室/诸子百家/社会人物分组	合并诸侯重臣+将相→臣；功臣侯爵归入臣
9	(本级)清理	多处修正	帝武乙/帝太丁残留商代(本级)而非商帝王
10	类名规范化	全面修正	拼音URI碰撞(Han=汉=韩)→改用中文URI

详细过程记录：见 `doc/methodology/人物本体构建过程.md`

通用反思原则：

1. **高频实例优先审查** — 出现次数最多的实例最容易暴露分类错误
2. **单信号不可靠** — 至少两个独立信号叠加才可信（章节+名称、章节+共现）。人物帝王类仅凭章节信号准确率仅12%
3. **误分析产出规则** — 分析"为什么错"比"哪些错"更重要，它能改进分类规则
4. **先审大类后审小类** — 大类（帝王437→90人）的修正幅度远大于小类（刺客17人基本正确）
5. **保留"其他"类** — 无法分类的实例放入"其他"，不强行归类
6. **区分分组维度与语义断言** — "按主要出现章节分组"是数据组织方式，不是"此人属于该国"的语义断言。子类标签应注明是分组还是断言
7. **历史语境影响分类** — 同一政体在不同时期有不同性质（秦国=诸侯，秦朝=帝国），分类必须考虑时间维度

8. **拆分后必须清理(本级)残留** — 每次拆分子类后，检查是否有实例被遗忘在父类的"本级"中
9. **章节信号 ≠ 所属国家** — 跨国叙事（三家分晋、合纵连横、出使征战）导致人物出现在非本国章节
10. **逐一审查子类实例** — 每次拆分出子类后，必须逐一检查实例归属是否正确（国王混入臣子、汉人混入外邦等）

Step 6. 迭代优化

常见迭代:

1. **合并过细的类** — 如果某个叶子类只有 1-2 个实例，考虑合并到父类
2. **拆分过粗的类** — 如果某个类超过 50 个实例且内部有明显子群，考虑拆分
3. **处理多归属** — 某些实体可归入多个类（如"蛟龙"可归鱼类或神话），选择主要归属，在 `rdfs:comment` 中注明
4. **补充属性** — 除 `:count` 外，可添加 `rdfs:comment`（典故/来源说明）、`rdfs:seeAlso`（关联章节）

三、分类判断的难点

3.1 生物 vs 神话

词	用法	归类
龙	"天降龙" → 实际出现	神话动物
龙	"龙颜" → 比喻	不标注
凤皇	"凤皇来翔" → 祥瑞	神话动物
白鹿	"获白鹿" → 实际猎获	野兽

3.2 集合 vs 具体

"六畜"不等于马+牛+羊+鸡+犬+豕的并集。它是一个独立的语义概念（"六畜兴旺"），应作为独立实例归入"集合泛称"类。

3.3 跨类型实体

词	生物类标注	其他类型标注	说明
白云	〖+白云〗	—	实为地名或天文，误标
犀象	〖+犀象〗	—	集合泛称
苍狗	〖+苍狗〗	—	实为天文（白衣苍狗）

这些需要在分类时修正，或移到正确的实体类型。

3.4 人物分类的典型错误模式

从 10+ 轮反思中总结的七种反复出现的错误模式：

错误模式	说明	防范方法
章节信号误导	人物出现在某章 ≠ 属于该章的类别（韩信→帝王）	多信号叠加验证
国王混入臣子	拆分后未逐一审查（白公→吴楚臣子）	拆分后逐一检查
汉人混入外邦	跨国叙事信号误导（张骞→匈奴）	区分叙事语境与归属
类名歧义	"燕"不明确是诸侯/燕还是臣/燕	类名自明原则
平铺非嵌套	子类显示为同级而非子级	递归渲染器+审查
(本级)残留	拆分后遗忘在父类的实例	拆分后检查本级

错误模式	说明	防范方法
URI碰撞	拼音 Han=汉=韩	全部使用中文URI

四、已完成的本体

类型	文件	类数	实例数	三元组	反思轮次
生物	`kg/rdf/biology.ttl`	20	70	294	1
人物	`kg/rdf/person.ttl`	142	1,800	~5,826	10+

待建本体（按优先级）

1. **地名** — 1,800+ 实体，可按地理类型（城邑/山/川/关隘）分类
2. **官职** — 2,100+ 实体，可按朝代/级别/文武分类
3. **器物** — 1,000+ 实体，可按功能（礼器/兵器/车马/货币）分类

人物本体的特殊之处

人物本体与生物本体有两个关键差异，形成了新的方法论模式：

差异一：实例 vs 类型

生物本体中，"马"既是类（马属），也是实例（出现102次的词条）。人物本体中，每个人（刘邦、项羽）都是纯粹的实例，分类（帝王、将相、刺客）是后归纳的。

	生物	人物
类的来源	词表中预设（animals_domestic 等）	从章节标题和出现模式归纳
实例的来源	标注提取的物种名	标注提取的人名
	1:1（马 → 马属）	

	生物	人物
类-实例关系		1:N（项羽 → 帝王 + 将相）

差异二：多重分类

人物天然具有多重身份。10.6% 的高频人物属于两个以上分类：

人物	分类
项羽	帝王 + 将相
李斯	法家 + 将相
吕不韦	商贾 + 将相
吕太后	后妃 + 帝王

在 RDF 中用 `a per:Emperor , per:GeneralMinister` 表达多类型。

差异三：分类信号来自章节结构

史记的体例本身就是一种分类系统：本纪（帝王）、世家（诸侯重臣）、列传（各类人物）。列传标题直接给出人物类型："刺客列传"→刺客类，"游侠列传"→游侠类。这是古籍本体构建特有的信号来源——文献体例即分类学。

分类方法论：三层信号叠加

1. **章节信号**：主要出现章节的体裁/标题（最强信号）
2. **名称模式**：人名后缀（X帝/X太后/X公 → 帝王/后妃/诸侯）
3. **共现信号**：与身份词（`〖#〗`）、官职词（`〖;〗`）的共现关系（待实现）

五、工具链

工具	用途
`build_entity_index.py`	从标注文本提取实体索引
`build_taxonomy.py`	从 TTL 本体生成 Markdown 分类树（通用，适用于任意实体类型）
`rdflib` (Python)	Turtle 语法验证和查询
Protege	可视化浏览/编辑本体（可选）
`git diff`	跟踪本体变更（Turtle 是纯文本，diff 友好）

build_taxonomy.py 用法

```
# 基本用法：TTL → 同名_taxonomy.md
python kg/rdf/scripts/build_taxonomy.py kg/rdf/person.ttl
python kg/rdf/scripts/build_taxonomy.py kg/rdf/biology.ttl

# 指定输出路径和单位
python kg/rdf/scripts/build_taxonomy.py kg/rdf/person.ttl -o
output.md --unit 人

# 自定义顶层子类排列顺序
python kg/rdf/scripts/build_taxonomy.py kg/rdf/person.ttl \
  --order "王室,臣,策士,诸子百家,社会人物,方术,外邦,虚构人物,疑似误标"

# 调整每类最多显示的实例数
python kg/rdf/scripts/build_taxonomy.py kg/rdf/person.ttl --max-
show 30
```

设计原则：TTL 本身是权威数据源，不需要额外的 JSON 分类数据。脚本从 TTL 中解析 `owl:Class` + `rdfs:subClassOf`（类层次）、`a <class>`（实例归类）、

`rdfs:label`（中文标签）、`:count`（出现次数），自动生成可读的 Markdown 分类树。

六、与 Ontology 101 的对应

Ontology 101 步骤	本管线对应
Step 1: 确定领域和范围	由标注文本的实体类型天然确定
Step 2: 考虑复用现有本体	可参考 DBpedia/Wikidata 的分类，但古汉语实体需自建
Step 3: 列举重要术语	= entity_index.json（已由 AI 标注+脚本提取完成）
Step 4: 定义类层次	= wordlist.json 的分类维度 → taxonomy.md
Step 5: 定义属性	= `:count`、`rdfs:label`、`rdfs:comment`
Step 6: 定义约束	= context_dependent 歧义规则
Step 7: 创建实例	= AI 标注提取的实体直接成为实例

核心差异：Ontology 101 从 Step 1 开始自顶向下设计；本管线从 Step 7（实例）开始自底向上归纳。

七、实战案例：人物本体（10轮反思详录）

> 完整过程记录见 `doc/methodology/人物本体构建过程.md`。以下为方法论精华摘要。

7.1 规模与难度

从1,825个人名实体出发，经10+轮反思，构建142类、~5,826三元组的OWL/RDF人物本体。与生物本体的关键差异：

	生物（先行试验）	人物
类的来源	词表预设（动物/植物）	需从实例中归纳
实例-类关系	1:1	1:N（韩信→将领+臣）
分类信号	词义本身	章节位置+名称模式+历史知识

7.2 反思迭代轨迹

轮次	操作	关键发现
1	章节信号初始分类	帝王类88%误分（437→49人，准确率12%）
2	三信号叠加修正帝王	437→90人，秦国诸侯≠秦朝帝王
3	诸侯类排除法	609→432人，帝王/后妃不重复归入
4	臣类展开（最大挑战）	800+人，采用 `时代×国家×性质` 三维拆分
5	叶子类强制拆分	所有20人以上叶子类继续拆分
6	中间层与顶层调整	引入王室/诸子百家/社会人物分组
7	URI与命名规范	拼音碰撞（Han=汉=韩），全改中文URI
8	(本级)残留清理	每次拆分后检查父类遗留实例
9	格式演进	pickle→JSON→TTL，脚本抽象化
10	收敛验证	剩余问题集中在20-50人叶子类

7.3 典型错误模式（可复用于其他实体类型）

错误模式	频次	示例	防范方法
章节信号误导	50+	韩信→帝王（因出现在本纪）	多信号叠加验证
国王混入臣子	10+	白公胜→吴楚臣子（实为楚国君主）	拆分后逐一审查
汉人混入外邦	5+	张骞→匈奴（因出现在匈奴列传）	区分叙事场景与所属
诸侯混入将领	5+	黥布→汉初诸将（实为楚汉诸侯）	身份优先于出场角色
类名歧义	10+	"燕"→燕国诸侯 or 燕国臣子?	类名必须自明
(本级)残留	10+	拆分后遗忘在父类的实例	每次拆分后检查本级
拼音URI碰撞	1	Han=汉=韩	用中文URI

7.4 关键方法论

1. **单信号分类灾难性不可靠**。帝王类仅凭"主章=本纪"分类，准确率12%。至少需要两个独立信号叠加验证。
2. **分类从边缘案例中涌现**。不是先设计分类树再填充，而是实例的误分暴露规则缺陷，修正规则→改进分类树。
3. **反思收敛**。每轮发现的问题越来越少。10轮后，剩余问题集中在需要词条简介才能判断的叶子类。
4. **章节信号 ≠ 所属国家**是最大陷阱。跨国叙事（三家分晋、出使匈奴）导致人物出现在非本国章节。

5. **拆分后必须清理(本级)残留**。每次拆分子类，必须检查是否有实例被遗忘在父类。

7.5 最终分类树

```
人物 [1825人]
├── 王室
│   ├── 帝王 [90人]（8个朝代子类）
│   ├── 诸侯 [432人]（16个国家子类）
│   ├── 后妃 [38人]
│   └── 汉宗室 [45人]
├── 臣 [800+人]
│   ├── 先秦早期 → 春秋（按国） → 战国（按国）
│   ├── 秦臣 → 楚汉（刘邦/项羽阵营）
│   └── 汉臣（文帝朝→景帝朝→武帝朝，各含将领/文臣）
├── 策士 [68人]
├── 诸子百家（儒生/法家/兵家/思想家/文学家/隐士/史家）
├── 社会人物（刺客/游侠/佞幸/滑稽/商贾）
├── 方术 [23人]
├── 外邦 [58人]
├── 虚构人物 [5人]
└── 疑似误标 [37人]
```

7.6 工作量参考

- 分类反思（10+轮）：~8小时对话
- 脚本开发调试：~4小时
- 树渲染修复：~2小时
- 总计：~14小时 + ~5亿tokens

本 SKILL 基于史记生物本体（388 种，20 类，294 三元组）和人物本体（1,825 人，142 类，~5,826 三元组，10+ 轮反思）的构建实践提炼。核心发现：生物类是"类型先行"（词表分类 → 实例归入），人物类是"实例先行"（人名提取 → 归纳分类 → 允许多重归属）。文献体例（本纪/世家/列传）是古籍本体构建特有的分类信号来源。

SKILL 07: 逻辑推理 — 矛盾检测与洞见挖掘

> 基于已构建的实体、事件、关系和本体，进行跨章节逻辑推理，发现矛盾事实和隐含知识。

一、子工序

编号	文件	内容
07a	SKILL_07a_人物生卒年推断.md	生卒年区间推断，置信度分级
07b	SKILL_07b_姓氏推理.md	先秦人物姓/氏区分，"同姓"语义澄清
07c	SKILL_07c_反常推理.md	违反常识/制度/逻辑的单条事实检测与研究线索挖掘

二、矛盾检测

2.1 矛盾类型

类型	定义	示例
DATE	同一事件记载不同时间	陈涉起义：七月 vs 秋（007/008）

类型	定义	示例
DETAIL	同一事件的数量/地点/参与者不一致	东城斩首：数十百人 vs 八万（007/008）
ATTRIBUTION	责任/功劳归属不同	函谷关：关守卒 vs 沛公守关（007/008）
SEQUENCE	事件顺序或因果链不一致	下宫之难：前583年 vs 前597年（039/043）

2.2 检测流程

```
事件索引A + 事件索引B
    ↓ 事件对齐（公元纪年 + 实体共现 + 关键词匹配）
重叠事件对列表
    ↓ 逐维度对比（时间/地点/人物/数量/归因/顺序）
差异列表
    ↓ 分类（真矛盾 vs 视角差异 vs 详略不同）
矛盾报告
```

2.3 可自动化的检测规则

规则	方法	精度
日期矛盾	同一事件公元纪年不一致	高
数量矛盾	同一事件数字差异>10倍	高
人物矛盾	同一事件参与者集合不一致	中
地点矛盾	同一事件发生地不一致	中

2.4 需LLM判断的检测

- 归因矛盾：需理解叙事语境才能判断
- 视角 vs 矛盾：需判断详略不同是否构成事实矛盾
- 因果链矛盾：需追踪多个事件的前后关系

2.5 已完成实验

实验	章节对	重叠事件	矛盾数	核心发现
007 vs 008	项羽本纪 vs 高祖本纪	15	4	东城斩首数（百 vs 八万）、函谷关归因
039 vs 043	晋世家 vs 赵世家	10	1	下宫之难日期（前597 vs 前583，差14年）
086 vs 006	刺客列传 vs 秦始皇本纪	3	1	太子丹之死（前222 vs 前226，差4年）

三、洞见挖掘

3.1 研究方法参考

李开元《秦崩》《楚亡》《汉兴》三部曲的研究方法可作为洞见挖掘的范式：

方法	说明	可机器化程度
交叉验证	将同一事件在不同章节的记载交叉比对，拼合完整图景	高（事件对齐已可自动化）
沉默证据	关注《史记》没有记载的事情——为什么不记？谁被遗忘了？	中（需LLM推理）

方法	说明	可机器化程度
反常检测	寻找不合常理的记载——违反常识、逻辑矛盾、前后不一	中（规则+LLM）
补白推理	从已知事实推断缺失的环节——A到C之间必然有B	中（需领域知识）
动机分析	分析人物行为背后的利益驱动——谁得利？谁受损？	低（需深度理解）
制度推理	从制度规则推断具体事件——封侯规则、继承规则	高（规则明确）

3.2 可挖掘的洞见类型

类型	示例	数据源
时间线空白	某人物在某年段完全无记载，之前之后都有	事件索引+公元纪年
异常关系	应有父子/君臣关系但缺失	关系网络+本体
统计异常	某章节实体密度/事件密度异常高或低	文本统计
地理矛盾	同一时间出现在两个不可能同时到达的地点	事件+地名+时间
制度违反	行为违反当时已知的制度规则	制度本体+事件
传承断裂	世系/官职传承链中断	家谱+官职关系

3.3 沉默证据检测方法

核心思路：本体定义了"应有"的关系，知识图谱记录了"实有"的关系。两者之差就是"沉默证据"。

自动检测方法：
1. 从本体获取规则（如：君主应有配偶关系）

2. 从关系图谱获取实际记录的关系
3. 差集 = 应有但缺失的关系 = **候选洞见**
4. LLM评估候选洞见的研究价值

缺失模式分类:

缺失类型	检测规则	自动化
配偶缺失	君主无王后记载	本体差集
死因缺失	重要人物无死亡记载	事件索引检查
动机缺失	重大决策无理由说明	LLM判断
后续缺失	事件有开始无结果	事件链断裂检测
世系断裂	父子链突然中断	家谱完整性检查
地理跳跃	人物突然出现在远方	时间+地点一致性
年龄矛盾	生育年龄/兄弟年龄差异常	生卒年+亲属关系计算

具体案例见 doc/analysis/推理案例集.md（秦始皇无皇后、刘邦年龄矛盾等）。

3.4 自动化洞见发现流程

```
结构化数据（实体+事件+关系+本体）
    ↓ 规则检测（时间线空白/关系缺失/统计异常）
候选洞见列表
    ↓ LLM评估（是否有历史意义？是否可能有合理解释？）
有意义的洞见
    ↓ 生成研究问题（"为什么X在Y年之后再无记载？"）
研究线索报告
```

四、基础推理能力

4.1 本体推理

X是Y的子类 → X的实例也是Y的实例

4.2 时间推理

A在B之前 + B在C之前 → A在C之前；人物死亡后不可能参与后续事件

4.3 关系推理

A是B之父 + B是C之父 → A是C之祖父；生育年龄异常 → 亲属关系存疑

4.4 地理推理

同一人同日出现在相距千里的两地 → 记载有误或为不同人

五、与其他工序的关系

上游	本工序	下游
04 事件构建（事件+纪年）	矛盾检测（跨章事件对比）	09 应用（矛盾可视化）
05 关系构建（实体关系网络）	洞见挖掘（缺失关系/异常模式）	08 SKU（研究洞见→知识单元）
06 本体构建（分类体系）	本体推理+制度推理	学术研究（论文素材）

六、研究模式挖掘方法论

6.1 从研究著作中提取推理模式

用 `pdf2skills` 工具链处理历史研究书籍，自动提取其中的研究模式和推理方法，反哺矛盾检测和洞见挖掘的规则库。

```
研究书籍（PDF/EPUB）
    ↓ pdf2skills 提取
研究模式清单（每个模式：前提→推理→结论）
    ↓ 归类
矛盾检测规则 / 洞见挖掘模板
    ↓ 编码为自动化规则或LLM提示词
SKILL_07 规则库扩展
```

6.2 候选书目

书名	作者	可提取的研究模式
《秦崩》	李开元	交叉验证、沉默证据、人物行踪重建
《楚亡》	李开元	动机分析、多源拼图、制度推理
《汉兴》	李开元	补白推理、反常检测、时间线重建
《秦谜》	李开元	缺失证据分析（秦始皇皇后、身世）
《史记的读法》	杨照	叙事结构分析、互见法解读
《中国历史研究法》	梁启超	史料鉴别、考证方法论
《二十史朔闰表》	陈垣	历法换算、日期验证

6.3 提取目标

从每本书中提取：

1. **推理模式**：作者如何从已知事实推导出新结论
   - 前提条件（需要哪些数据）
   - 推理逻辑（什么规则）
   - 结论类型（发现了什么）
2. **检测规则**：可编码为自动化检测的规则（如人物年龄一致性检测）

具体案例见 doc/analysis/推理案例集.md

1. **提问模板**：将学者的研究问题抽象为可复用的模板
   - 例："为什么《史记》对X事件的记载在Y章和Z章中不一致？" → 模板化为跨章矛盾追问

七、下一步

- [ ] 编写自动化事件对齐脚本（基于公元纪年+实体共现）
- [ ] 扩展矛盾检测到更多章节对（如031吴太伯世家 vs 039晋世家）
- [ ] 实现时间线空白检测（人物消失期）
- [ ] 实现地理矛盾检测（同时出现在两地）
- [ ] 基于李开元方法论设计"沉默证据"发现提示词

SKILL 07a: 人物生卒年推断 — 区间估算

> 从《史记》原文和事件索引中，为人物推断生卒年的**可信区间**，而非单一点值。区间宽度反映证据强度：证据充分时区间窄（±5年），仅有间接推算时区间宽（±50年以上）。

一、现有数据状态

文件	内容	问题
`kg/entities/data/person_lifespans.json`	~60人，点值格式 `{"birth": -551, "death": -479}`	来自外部百科，非从史记文本推断；无置信度；无推断依据

目标：扩展为区间格式，覆盖更多人物，记录推断来源。

二、数据格式

2.1 区间表示

```
{
  "孔子": {
    "birth_min": -556, "birth_max": -550,
    "death_min": -480, "death_max": -478,
    "birth_label": "[约公元前551年]",
    "death_label": "（公元前479年）",
    "confidence": "high",
    "evidence": ["《史记·孔子世家》：孔子生鲁昭公二十二年（前520年说存疑）", "年表：前479年卒有明确记载"],
```

```
    "note": "生年有前551/前552年两说，取551年"
  },
  "晋文公": {
    "birth_min": -700, "birth_max": -694,
    "death_min": -629, "death_max": -627,
    "birth_label": "[约公元前697年]",
    "death_label": "[公元前628年]",
    "confidence": "medium",
    "evidence": ["《史记·晋世家》：重耳出亡时年已壮，在位9年卒", "年表：前636年归国即位，即位后9年卒=前628年"]
  }
}
```

2.2 置信度分级

confidence	含义	区间宽度典型值
`exact`	史书明确记载年份（少见）	0年（点值）
`high`	年表有记载或可精确换算	≤5年
`medium`	可从在位年数+即位年推算	5–20年
`low`	仅有间接线索（父子关系、同时代人）	20–50年
`legend`	传说时代，现代学界推算	>50年

三、推断方法（按证据强度排序）

方法1：直接卒年记载（最可靠）

原文有"X年，某人卒"或年表有明确记载。

```
"三十九年，缪公卒" + 缪公元年=前659年 → 卒年=前621年（exact）
```

方法2：在位年数反推

原文有"立X年卒"，即位年已知。

```
"德公生三十三岁而立，立二年卒"
+ 德公即位年=前677年
→ 卒年=前677-2=前675年（high）
→ 生年=前677-33=前710年（high）
```

方法3：年龄+事件锚点

原文有"年X岁立/仕/行某事"，事件年份已知。

```
"孝公立，年已二十一岁"，孝公元年=前361年
→ 生年=前361-21=前382年（high）
```

方法4：相对关系约束

无直接年龄，但有亲属/同时代约束。

```
A是B之父，B生于前X年
→ A生于前(X+20)至前(X+40)年（low，假设生育年龄20-40岁）

A与B同朝为官，B卒于前X年
→ A活跃期包含前X年（约束区间）
```

方法5：世代数推算（低可靠度）

史书有"某人X世孙"，始祖年代已知。

```
"大廉玄孙曰孟戏"，大廉活跃于商初约前1600年
→ 孟戏活跃于前1600-(4×25)=前1500年（low，每代25年估算）
```

> ⚠️世代数×年数换算误差大，传说时代尤不可靠，区间应扩大到±100年。

四、推断流程

```
输入：标注文本 (chapter_md/*.tagged.md)
    + 事件索引 (kg/events/data/*_事件索引.md)，已含公元纪年
    |
Step 1：从原文提取年龄/在位年数证据
    扫描格式：
    - "生X岁"/"年X岁立"/"年已X岁"
    - "立X年卒"/"在位X年"/"为X X年"
    |
Step 2：从事件索引获取锚点事件年份
    关联事件：即位、卒、出奔、仕于某君
    |
Step 3：计算区间
    按方法1→2→3→4→5优先级，取最强证据
    区间 = 计算值 ± 误差（误差来自方法本身精度）
    |
Step 4：亲属关系交叉约束
    父卒年 < 子生年（不合理则标注矛盾）
    兄弟生年差 < 40年（超出则标注存疑）
    |
输出：person_lifespans_v2.json（区间格式）
    + 推断报告（每人：证据链 + 置信度 + 矛盾标记）
```

五、与其他工序的关系

方向	工序	说明
输入	04c 事件年代推断	提供锚点事件的公元纪年
输入	05b 实体关系构建	提供亲属关系（用于交叉约束）
输出→	07 矛盾检测	生卒年区间用于检测年龄矛盾、同时代人冲突
输出→	04c 事件年代推断	反向校验：事件年份应落在当事人生卒区间内
输出→	09 应用构造	人物时间轴可视化

六、已有数据升级计划

`person_lifespans.json` 现有约60人，点值格式，来源为外部百科。

升级步骤:
1. 将现有点值转为区间：`birth → birth_min=birth-5, birth_max=birth+5`（标记 confidence 为外部来源，待验证）
2. 从事件索引批量提取年龄证据（正则扫描"生X岁"/"立X年"等）
3. 对前5章（本纪）人物逐一推断，验证方法
4. 逐步扩展到130章

输出文件: `kg/entities/data/person_lifespans_v2.json`

七、传说时代特殊处理

五帝至夏（001–003章）人物生卒年本身是后人推算，现有数据（如黄帝前2717–前2599）来自学界估算，区间极宽。

原则:
- 传说人物统一标记 `"confidence": "legend"`
- 区间宽度 ⩾ 100年

- 不用生卒年中点作为事件锚点（见 SKILL_04c 错误模式30）
- 优先用朝代起止年（夏始前2070年、汤灭夏前1600年）作为叙事定位

SKILL 07b: 姓氏推理 — 先秦人物姓与氏的区分与标注

> 从《史记》文本推断先秦人物的姓/氏/名/字，建立传承记录，辅助消歧和阅读注释。

背景与数据模型见：doc/spec/姓氏制度.md

一、总体思路：多轮迭代

姓氏推理是一个**传播问题**：有明确记载的人是种子，通过邦国、氏族、父系三条路径向未知人物扩散。每轮迭代都会新增已知条目，从而解锁更多下一轮的推理。

```
Round 1：直接提取（明确记载）
    ↓ 已知集合扩大
Round 2：邦国推理（属于已知邦国 → 继承邦国之姓）
    ↓ 已知集合扩大
Round 3：氏族推理（属于已知氏族 → 继承氏族之氏）
    ↓ 已知集合扩大
Round 4：父系推理（父已知 → 子可继承姓，未创新氏则继承氏）
    ↓ 已知集合扩大
Round 5：验证与冲突消解
    ↓
输出：person_xingshi.json 增量更新
```

每轮结束后将新推理出的条目写入 `person_xingshi.json`，供下一轮使用。

二、推理规则

规则 R1：直接记载

```
IF 文本中有"X者，Y姓也" / "X，姓Y氏Z" / "以Z为氏"
THEN xing(X) = Y，shi(X) = Z
confidence = exact / high
```

示例：`屈原者，名平，楚之同姓也` + 世家"以屈为氏" → xing=芈, shi=屈

规则 R2：邦国推理

```
IF 人物X属于邦国S（出仕、封于、为S大夫等）
AND 邦国S的姓为Y（来自 xing_index.json）
AND X无更明确的姓记录
THEN xing(X) = Y
confidence = medium（若仅有邦国关系）
```

主要邦国姓族对照：

邦国	姓	备注
周王室	姬	所有姬姓封国同此
鲁、卫、晋、燕、郑、吴、虞	姬	周王室分封
齐（姜齐）	姜	田氏代齐前
齐（田齐）	妫	陈完后裔，妫姓田氏
楚	芈	楚王室及屈/景/昭三族
秦	嬴	

邦国	姓	备注
赵	嬴	与秦同祖
宋	子	殷商后裔
薛	任	奚仲后裔，统治者任姓薛氏
陈	妫	舜后裔

规则 R3：氏族推理

```
IF 文本中人物X与氏族Z有明确关联（"Z氏" / "Z家" / 属于Z氏族群体）
AND Z 已知为某氏
THEN shi(X) = Z
confidence = high（若关联明确）
```

示例：文中"屈氏"成员 → shi = 屈

规则 R4：父系传姓

```
IF father(X) = F，且 xing(F) = Y 已知
THEN xing(X) = Y（姓沿父系不变）
confidence = 继承自F的置信度，不超过 medium
```

规则 R5：父系传氏

```
IF father(X) = F，且 shi(F) = Z 已知
AND X 未有以下创氏事件：
    - 受封新地（以封地为氏）
    - 担任新官职（以官职为氏）
```

```
    - 以祖名另立新氏
THEN shi(X) = Z（继承父氏）
confidence = 继承自F的置信度，不超过 medium
```

规则 R6：新氏创立（阻断继承）

```
IF X 被封于地Y → shi(X) = Y（封地氏）
IF X 的后代以X的官职P为氏 → shi(后代) = P（官职氏）
IF X 的后代以X的名/字为氏 → shi(后代) = 名/字（祖名氏）

以上三种情况下，X的后代不再继承F的氏
```

规则 R7：反向推理（约束传播）

```
IF son(X) = S，且 xing(S) = Y 已知
THEN xing(X) = Y（父子同姓）
confidence = low（仅作为辅助约束，不单独成立）

IF 两人被称"同姓"
THEN 他们共享相同的 xing
confidence = high（同姓记载可信）
```

规则 R8：人名首字推断（姓名模式）

```
IF 人名X的首字或首二字S为已知姓氏（含复姓）
AND X不属于已知的误匹配名单
THEN xing(X) = S
confidence = medium
```

要点:
- 复姓优先匹配（"司马""公孙""淳于"等先于单字）
- 单字名（如"赵""韩"）同时是国名时需人工消歧，不自动推断
- 此规则覆盖面最广（~1100人），但置信度统一为 medium

规则 R9：邦国名+称号模式

```
IF 人名形如"国名 + 王/公/侯/伯/太子/公子/世子/太伯"
AND 国名在 state_to_xing 映射中
THEN xing(X) = state_to_xing[国名]
confidence = medium
```

陷阱与例外:
- **汉代封王排除**：以先秦国名开头的汉代封王实为刘姓（如楚元王刘交、鲁元公主、赵王敖刘如意），必须排除
- **齐国时代分治**：姜齐（齐桓公等，前386年前）→ 姜姓；田齐（齐威王等，前386年后）→ 妫姓田氏
- **身份混淆排除**："卫子夫"是人名不是卫国公族，"秦相文信侯"是吕不韦不是嬴姓，"滕公"可能是汉代夏侯婴的封号

规则 R10：秦汉姓氏合一

```
IF period(X) ∈ {qin, han}
THEN xing = shi（姓氏已合流，不再区分）
confidence = 直接记载为 exact，名字推断为 medium
```

秦统一后封建分封制被郡县制取代，"氏"失去政治意义。现代意义的"姓"实为古代的"氏"。

规则 R11：外邦人物姓氏

```
IF X为非华夏体系人物（匈奴、越、西域等）
THEN 使用该族姓氏体系（如匈奴单于为挛鞮氏）
confidence = high（有族属记载时）
```

三、规则优先级与冲突处理

优先级	规则	覆盖条件
最高	R1 直接记载	文本明确陈述，覆盖其他推理结果
高	R3 氏族推理	氏族关联明确时覆盖父系推理
中	R4/R5 父系传播	无更高置信度来源时使用
中	R9 邦国名+称号	名称匹配，需排除汉代封王
中	R8 人名首字推断	覆盖面广但置信度统一为 medium
低	R2 邦国推理（出仕）	仅在无其他来源时兜底
辅助	R7 反向约束	不直接赋值，仅用于一致性检验

冲突处理原则：
- 同一人物出现两个不同姓 → 检查是否为同名不同人（触发 SKILL_03b 消歧）
- 同一人物出现两个不同氏 → 检查是否中途改氏（记录变化时间点）

四、数据输入/输出

输入	来源
实体标注文本	`chapter_md/` 各章
现有人物实体索引	`kg/entities/data/entity_index.json`
邦国归属关系	`kg/entities/data/entity_index.json`（`〖'邦国〗`类型）
氏族归属关系	`kg/entities/data/entity_index.json`（`〖&氏族〗`类型）
父系关系	`kg/relations/` 实体关系索引
姓族索引	`kg/entities/data/xing_index.json`

输出	位置
人物姓氏记录	`kg/entities/data/person_xingshi.json`
更新后的标注	`〖&X|姓〗` / `〖&X|氏〗` 子类型

五、标注规范

在 `〖&氏族〗` 基础上增加子类型：

标注形式	含义
`〖&X|姓〗`	X 为血缘之姓
`〖&X|氏〗`	X 为分家之氏

标注形式	含义
〖&X|氏族〗	无法区分，或秦后混用

"同姓"语义澄清：凡文中"同姓"，HTML 注释层注明实际姓族及分家代数。

六、工具链

文件/脚本	位置	说明
`person_xingshi.json`	`kg/entities/data/`	产出：1469条人物姓氏记录
`person_xingshi.md`	`kg/entities/data/`	可读表格（按轮次分组）
`xing_index.json`	`kg/entities/data/`	输入：11大姓族 + 31个邦国→姓映射
`prompt_template_姓氏推理.md`	`kg/entities/scripts/`	Agent 提示词模板
`build_xingshi.py`	`kg/entities/scripts/`	批量推理脚本（待建）
`validate_xingshi.py`	`kg/entities/scripts/`	验证覆盖率（待建）
`inject_xingshi_tooltip.py`	`kg/entities/scripts/`	HTML 渲染注入（待建）

七、与其他工序的关系

上游	本工序	下游
03a 实体标注（〖@人名〗〖&氏族〗〖'邦国〗）	姓氏推理（多轮迭代）	03b 实体消歧（姓氏辅助区分同名人物）
05b 实体关系（父子/家族关系）	父系传播（R4/R5）	07 矛盾检测（"同姓"一致性核验）
—	子类型标注	09a 认知辅助阅读器（悬浮注释）

八、执行记录

第一轮（2026-03-16）

八轮迭代完成，覆盖率 **56.6%**（2053/3630）：

轮次	方法	新增	累计
R1	直接记载（6本纪+12世家+10列传）	34	34
R2	邦国名+称号推理	306	340
R3	人名首字/复姓推理	1099	1439
R4-5	世系补充+重要人物手工	30	1469
R6	深度反思：重要人物逐个确认	277	1746
R6b	批量规则：帝X→殷商、子X→楚、世家章节→邦国、复姓扩展	242	1988

轮次	方法	新增	累计
R6c	高频人物最终补充（单字简称、汉代封王等）	68	2053

置信度分布：exact 172、high 87、medium 1688、low 106

发现的典型陷阱

1. **汉代封王假阳性**：楚元王(刘交)、鲁元公主(刘乐)等以先秦国名开头但实为刘姓
2. **齐国时代分治**：齐桓公=姜姓，齐威王=妫姓田氏，分界线前386年
3. **单字名消歧困难**："赵"225次出现中混有国名指代和人名指代
4. **官职氏vs封地氏**：司马=官职氏、卫=封地氏，同一人可能有多重称呼
5. **秦相文信侯**=吕不韦（吕氏），不是嬴姓；"卫子夫"是人名不是卫国公族

反思中发现的新推理模式

1. **R12 章节归属推理**：纯谥号人物（"X公""X侯"）若仅出现在某世家，可推断为该国君主
2. **R13 帝号→朝代推理**："帝X"出现在殷本纪→子姓殷氏，出现在夏本纪→姒姓夏后氏
3. **R14 楚国"子X"推理**：先秦"子X"双字人名若主要出现在楚世家→芈姓楚公族
4. **R15 汉代封王推理**：菑川王/胶东王/济北王/淮南王等汉代封王→刘姓

覆盖率瓶颈分析

剩余未覆盖 1587 人（43.4%），构成：
- **单字简称** 400+：多为消歧问题，同一字指代不同人物
- **低频人物** 1400+（出现1-4次）：文本中无姓氏线索
- **纯称号/泛称** 100+：如"别将""市人""路人"等非具体人名

九、下一步

- [x] 创建 `xing_index.json`（11大姓族 + 31邦国映射）

- [x] 创建 `person_xingshi.json`（八轮迭代，2053条，56.6%覆盖率）
- [x] 新增 R8–R15 推理规则
- [x] 深度反思轮：逐个确认重要人物 + 批量规则扩展
- [ ] 提升覆盖率至 70%+：逐章精读补充低频人物
- [ ] 完善父子关系数据质量（ `kg/relations/` 噪音多），支撑 R4/R5 自动传播
- [ ] 消歧单字名（"赵""韩""魏"等 400+ 个）
- [ ] 更新 SKILL_03a：增加 `|姓` / `|氏` 子类型说明
- [ ] 构建验证脚本 `validate_xingshi.py`（检测冲突、统计覆盖率）
- [ ] 清理低质量条目（confidence=low 的 106 条需人工复核）

SKILL 07c: 反常推理 — 不符合常识的事实检测与解读

> 从《史记》文本中识别"反常事实"：单条记载本身没有矛盾，但违反常识、违反制度规则、或在当时语境下极为罕见。目标是发现值得深究的历史异常点。

与 SKILL_07 矛盾检测 的区别：矛盾检测关注**两处记载互相打架**；反常推理关注**单条记载本身令人意外**。

一、反常类型分类

1.1 数字反常

子类型	判断标准	典型案例
兵力异常	单支军队超过50万，或双方兵力差距>10:1	长平之战赵军四十余万降卒
年龄异常	人物记载行为时年龄超出生理极限或常规	廉颇八十余岁仍欲为将
时间异常	事件发展速度违反地理/行军常识	数日内跨越千里
财富异常	赏赐/财富数量超出当时国家财政常规	陶朱公三致千金

1.2 行为反常

子类型	判断标准	典型案例
身份越轨	庶民/女性/奴隶做了通常只有贵族/男性/自由人才能做的事	吕后临朝称制
亲情反常	父杀子、子弑父、手足相残且无明显动机	晋献公杀申生
忠奸反转	被《史记》正面评价的人物做了反道德的事，或反之	—
动机缺失	重大行为找不到合理动机，或现有解释自相矛盾	—

1.3 制度反常

子类型	判断标准	典型案例
礼制违反	行为违反当时已知周礼/汉制的明文规定	诸侯僭用天子礼乐
继承异常	世系/王位继承跳过通常规则（嫡长子制等）且无记载原因	—
封赏异常	封号/爵位/赏赐规格与功绩明显不相称	—
刑罚异常	同类罪行处置轻重悬殊	—

1.4 叙事反常

子类型	判断标准	典型案例
沉默异常	重要人物/事件在相关章节完全缺席（应有而无）	秦始皇无皇后记载
详略失衡	次要事件大书特书，重要事件一笔带过	—
评价矛盾	太史公曰与正文叙事传递出相反的态度	—
时序倒置	叙述顺序与事件时序明显不符且无明显原因	—

二、检测流程

```
结构化数据（实体+事件+关系+本体规则）
      ↓ 规则扫描（数字阈值/制度违反/关系缺失）
候选反常清单
      ↓ LLM评估（是真反常？还是有常见解释？）
有价值的反常列表
      ↓ 分类标注（类型 + 置信度 + 可能解释）
输出：anomaly_report.json + 研究线索
```

2.1 规则检测（可自动化）

规则	数据源	输出
军队规模 > 阈值	事件索引 `count` 字段	候选异常
人物年龄 > 80 且仍有军事行动	生卒年推断 + 事件	候选异常
嫡长子未继位 且无废嫡记载	世系关系 + 继承事件	候选异常
重要人物（出现频次>N）无死亡事件	实体索引 + 事件索引	候选异常

2.2 LLM评估提示词框架

```
你是一位审视《史记》异常事实的历史学助手。

已知事实：[原文引用]
所属章节：[篇名]
类型标注：[数字反常 / 行为反常 / 制度反常 / 叙事反常]

请判断：
1. 这条记载是否违反[常识/制度/逻辑]？（是/否/存疑）
2. 有无通常的历史学解释？（列出1-3条）
3. 若无合理解释，这是否值得深入研究？（高/中/低价值）
4. 推荐追问方向：（一句话）
```

三、输出格式

anomaly_report.json 条目结构

```
{
  "id": "ANOM-001",
  "type": "数字反常/行为反常/制度反常/叙事反常",
  "subtype": "兵力异常",
  "chapter": "073",
  "chapter_name": "白起王翦列传",
  "original_text": "括军败，卒四十余万人降武安君",
  "anomaly_description": "单次战役降卒四十余万，超出多数学者对当时总人口的估算比例",
  "confidence": "high",
  "possible_explanations": [
    "古代数字夸大惯例（实际可能为4万或14万）",
    "包含非战斗人员",
    "历史学家的不同考证"
  ],
  "research_value": "high",
```

```
  "follow_up": "对照《战国策》《资治通鉴》的记载，以及现代人口史研究",
  "related_anomalies": []
}
```

四、与其他子工序的关系

依赖工序	关系
07a 生卒年推断	年龄异常检测依赖生卒年区间
07 矛盾检测	反常往往伴随矛盾，可互相触发
04 事件构建	数字/时间异常从事件索引中提取
05 关系构建	制度/继承异常从关系网络中检测

五、优先场景

以下场景反常密度高，适合作为首批检测对象：

1. **长平之战**（073白起王翦列传）：降卒数字、坑杀规模
2. **秦始皇无皇后**（006秦始皇本纪）：叙事沉默
3. **刘邦年龄**（008高祖本纪）：起兵年龄与生卒年的张力
4. **吕后称制**（009吕太后本纪）：女性执政
5. **项羽破釜沉舟**（007项羽本纪）：以少胜多的数字与战术逻辑
6. **商鞅徙木立信**（068商君列传）：行政成本与政治信号

六、下一步

- [ ] 建立 `anomaly_report.json` 初始文件（从已有矛盾案例库迁移）

- [ ] 编写数字异常自动扫描脚本（扫描事件索引中的数字字段）
- [ ] 设计 LLM 批量评估流水线（复用 SKILL_07 的矛盾检测 prompt 框架）
- [ ] 优先处理上述六个场景，产出首批反常条目

SKILL 08: SKU构造 — 从结构化知识到知识单元

> 将实体、事件、关系、本体等结构化知识封装为可独立分发的知识单元（SKU, Stock Keeping Unit），面向不同用途和受众。

一、工序位置

```
实体索引 + 事件索引 + 关系网络 + 本体 + 推断知识
    |
    ▼ 08 SKU构造（本SKILL）
知识单元 (ontology/SKUs/)
    |
    ▼ 09 应用构造
用户产品
```

二、三类知识单元

类型	说明	示例
Factual（事实知识）	人物传记、诸侯国、战役、制度等事实性知识卡片	韩信传记、长平之战、郡县制
Procedural（技能知识）	可复用的策略、方法、模式	军事战略、治国理政、外交谋略
	标签体系、术语表、实体关系图	

类型	说明	示例
Relational（关系知识）		人物关系网、事件因果链、地名沿革

三、SKU Schema

```
{
  "id": "SKU-person-韩信",
  "type": "factual",
  "title": "韩信",
  "summary": "汉初将领，助刘邦灭项羽，封齐王/楚王，后被吕后杀于长乐宫",
  "entities": ["韩信", "刘邦", "项羽"],
  "events": ["092-001", "092-015", "092-016"],
  "relations": [...],
  "chapters": ["008", "092"],
  "tags": ["军事", "汉初", "功臣"]
}
```

四、待定义

- [ ] SKU粒度标准（人物级/事件级/主题级）
- [ ] SKU生成脚本（从实体索引+事件索引自动生成初稿）
- [ ] SKU质量标准（必填字段、最小信息量）
- [ ] SKU发布格式（JSON/Markdown/API）

SKILL 09: 应用构造 — 从知识到产品

> 将结构化知识转化为面向用户的应用产品。

一、已有应用

应用	目录	说明
认知辅助阅读器	`docs/chapters/`	语法高亮 + 段落锚点 + 实体索引链接，详见 SKILL 09a
事件地铁图	`app/metro/`	130条线路/3,185站点的SVG可视化，支持缩放/拖拽/搜索/实体链接/原文引用
史记争霸游戏	`app/game/`	策略卡牌+文字冒险，基于SKU技能卡，详见 SKILL 09b

二、应用与知识层的关系

```
知识层（KG + SKU）
    ├── entities/ → 阅读器实体链接、游戏人物系统
    ├── events/   → 地铁图站点、游戏剧情事件
    ├── relations/ → 地铁图换乘、游戏技能组合
    ├── chronology/ → 阅读器年份tooltip、地铁图时间轴
    └── skus/       → 游戏技能卡数据源
          ↓
应用层（HTML/JS/CSS）
```

```
├── 阅读器：render_shiji_html.py 渲染
├── 地铁图：metro.js 从 metro_map_data.json 渲染SVG
└── 游戏：game.js 从 SKU 构建卡牌系统
```

三、待开发应用

- [ ] 史记知识问答机器人（基于知识图谱+SKU的QA，支持原文检索/事件查询/人物关系问答）
- [ ] 人物关系图谱（社会网络可视化，D3.js force graph）
- [ ] 交互式年表（时间轴+事件筛选）
- [ ] 古籍对照阅读器（原文+标注+注释三栏联动）

SKILL 09a: 认知辅助阅读器 — 语法高亮 + 结构语义化

> 将标注Markdown渲染为带语法高亮的HTML阅读器，借鉴IDE代码高亮思路降低古文阅读认知负担。

一、设计理念

语法高亮是当代更强大的"标点符号"：

- 古代"句读"帮助断句 → 语法高亮帮助理解语义结构
- 18类实体用不同颜色/样式标注 → 一眼识别关键信息（人物、地点、官职、时间）
- 段落结构语义化 → 适当字体、缩进、分段、对齐降低认知负担
- 从"标点断句"进化到"语义着色"——认知辅助的革命性提升

效果：就像IDE帮助程序员理解代码，认知辅助阅读器帮助读者理解历史文献。

二、渲染管线

```
chapter_md/*.tagged.md
    ↓ render_shiji_html.py（正则替换：Token→HTML span）
docs/chapters/*.html
    ↓ CSS（shiji-styles.css：配色/字体/布局）
    ↓ JS（purple-numbers.js：锚点点击复制）
浏览器渲染
```

核心渲染器 `render_shiji_html.py`

有序正则替换链，将〖〗Token标记转为HTML `<span>`：

1. **粗体** `**text**` → `<strong>`（最先处理，优先级最高）
2. **内联消歧（所有实体类型）** `〖TYPE 显示名|规范名〗` → `<span class="TYPE" title="类型：规范名" data-canonical="规范名">显示名</span>`（v2.7：13类均支持，优先于普通匹配）
3. **13类〖TYPE〗标记** → `<span class="TYPE" title="类型名">`
4. **5类新格式标记**（〖?〗/〖{〗/〖:〗/〖[〗/〖_〗）→ 对应 `<span>`
5. **引号** → `<span class="quoted">`（对话标识）
6. **段落编号** `[N.N]` → `<a class="para-num">` 可点击锚点
7. **实体链接** → 后处理包裹 `<a>` 指向实体索引页（内联消歧的 `data-canonical` 优先于 disambiguation_map）

处理顺序：粗体 → 内联消歧（各类型消歧版优先于普通版）→ 器物/官职/地名等外层 → 普通人名（最后，常作内层）

批量生成

```
python generate_all_chapters.py      # 调用渲染器处理130章
bash publish_to_docs.sh              # 完整发布流水线
```

三、视觉设计

3.1 实体渲染（18类名词实体）

设计文档:
- 实体渲染规划 v5.0 - v5.0设计方案
- CSS版本历史 v5.4 - v5.4版本演进
- 更新日志 2026-03-19 - v5.4发布说明

核心理念（v5.4）：
- **text-decoration风格** + 下划线组合（实线/虚线/点线/波浪线/无）

- **朝代紫底** + **其他淡黄底** - 与动词明确区分
- **参考古籍专名线**（人名线/地名线）和 **IDE语法高亮**（数字/关键字）
- **v5.4新增**：典籍/神话波浪线，朝代/思想下划线加粗

配色分组总览

褐色组（空间叙事）：人名、地名、官职
青色（时间维度）：时间（独立）
紫色组（政治组织）：朝代、封国、族群
蓝色组（社会制度）：制度、刑法、礼仪、身份
墨绿组（文化思想）：典籍、思想
橙褐组（物质世界）：器物、生物、天文、神话
海绿（数量度量）：数量（独立）

18类实体速查表

类型	CSS class	颜色	下划线	特殊样式	v5.4变化
人名	`.person`	#8B4513 褐色	实线	淡黄底	-
地名	`.place`	#B8860B 深金	双实线	-	-
官职	`.official`	#8B0000 暗红	无	中粗	-
时间	`.time`	**#008B8B 深青**	**无**	**斜体**	-
朝代/氏族	`.dynasty`	**#9370DB 中紫**	**实线2px**	**淡紫底**	**v5.4加粗**
封国	`.feudal-state`	#B266B2 梅红	实线	-	-
族群	`.tribe`	#2F4F4F 墨绿	无	中粗	-
制度	`.institution`	#4682B4 钢青	无	中粗	-
刑法	`.legal`	#8B0000 暗红	无	淡红底	-

类型	CSS class	颜色	下划线	特殊样式	v5.4变化
礼仪	`.ritual`	#8B6914 深金	无	中粗	-
身份	`.identity`	-	-	-	(未明确)
典籍	`.book`	**#4B0082 靛蓝**	**波浪线2px**	**斜体**	**v5.4新增**
思想	`.concept`	**#2F4F4F 墨绿**	**点线2px**	**-**	**v5.4加粗**
器物	`.artifact`	#CD853F 秘鲁棕	无	中粗	-
生物	`.biology`	#228B22 森林绿	无	中粗	-
天文	`.astronomy`	#483D8B 深岩蓝	无	斜体	-
神话	`.mythical`	**#8B008B 深品红**	**波浪线2px**	**中粗**	**v5.4新增**
数量	`.quantity`	#2E8B57 海绿	无	中粗	-

设计原则:

- **text-decoration方案**：v5.3+采用text-decoration替代border-bottom，避免背景重叠
- **下划线增强**：v5.4加粗关键类型(朝代2px、思想2px、典籍/神话波浪线2px)
- **朝代特殊标识**：紫色背景#F0E6FF + 紫色下划线，突出政权/氏族/封国
- **波浪线应用**：典籍(传统书名号延续)、神话(虚构性标识)
- **tooltip提示**：悬停显示实体类型名，消歧后显示全名

3.2 动词渲染（4类动词实体）

设计文档: 动词渲染规则 v4.0

核心理念:
- **鲜明背景色** - 红/蓝/金独立背景，与实体淡黄底明确区分
- **统一加粗** - font-weight:600/500，视觉强调
- **关系核心** - 动词是三元组（主-谓-宾）的连接纽带

4类动词速查表

类型	符号	CSS class	颜色	背景色	样式
军事动词	◈	`.verb-military`	#DC143C 深红	rgba(220,20,60,0.08)	加粗 600
刑罚动词	◉	`.verb-penalty`	#8B0000 暗红	rgba(139,0,0,0.06)	加粗 600 + 底线
政治动词	○	`.verb-political`	#4169E1 皇家蓝	rgba(65,105,225,0.06)	加粗 500 (预留)
经济动词	◇	`.verb-economic`	#DAA520 金杆	rgba(218,165,32,0.06)	加粗 500 (预留)

视觉区分:

实体：淡黄底 🟡 + 5色分组 + 下划线变化
动词：鲜明底 🔴🔵 + 加粗 + 无下划线（除刑罚底线）

四、段落结构语义化

字体与排版

- **正文**：Noto Serif SC（思源宋体），行高2.0，最大宽度900px
- **背景**：#fdfdf8 暖白色（古籍纸张感）
- **段间距**：2em（宽松排版，适合长文阅读）

语义区块（::: fenced div）

Markdown中用 `::: 标签` 围栏标注语义区块，渲染为带样式的 `<div>`：

区块类型	视觉样式
太史公曰	暗金左边框 + 暖白背景
赞	棕色左边框 + 斜体 + 保持硬换行
诗歌	橄榄绿左边框 + 斜体 + 保持硬换行

区块标签渲染为tooltip（悬停显示），不占用正文空间。

对话处理

- **引号内容**：斜体 + 极淡褐色底色 + 底部虚线
- **长对话**（>15字）：另起一行，缩进两个汉字（`display: block; padding-left: 2em`）
- **引用块 `>`**：缩进两个汉字，斜体

韵文自动分行

渲染器自动检测赞中的韵文（短句+逗号对仗），在句末标点后插入 `<br>` 换行。判定条件：

- ≥3个句末标点
- 平均句长≤12字

- ≥60%的句子含逗号
- ≥70%的句子≤14字

五、交互功能

Purple Numbers（段落锚点）

致敬 Doug Engelbart 1960年代的概念：

- 每个段落 `[1.2.3]` 渲染为淡紫色小字锚点
- 点击复制带 `#pn-1.2.3` 锚点的完整URL
- 表格行有独立行号锚点 `#pn-rN`

实体索引链接

每个实体 `<span>` 自动包裹 `<a>` 链接：

- **人名**：经消歧（章节级短名→全名）+ 别名解析后链接到实体索引页
- **时间**：年份查询 `year_ce_map.json` 添加公元纪年tooltip，链接到编年索引
- **其他实体**：直接链接到对应类型的索引页

表格渲染

十表（013-022）的Markdown表格渲染为HTML表格：

- 全视口宽度 + sticky表头 + 交替行背景
- 表头不做实体标注（避免年号/国名被样式干扰）
- 数据行自动添加行号锚点

六、实体索引页

```
扫描 tagged.md → 正则提取实体 → 别名合并 → 拼音排序 → 生成HTML索引页
```

- 18类实体各生成独立索引页（`docs/entities/*.html`）
- 条目含：出现次数、章节分布、别名列表
- 搜索过滤：`entity-filter.js` 按名称和别名实时过滤
- 拼音导航：按拼音首字母分节

七、发布流水线

```
bash publish_to_docs.sh
```

完整步骤：

1. 准备 `docs/` 目录结构
2. 复制 CSS/JS 文件
3. `python generate_all_chapters.py` 批量渲染
4. 移除文件名 `.tagged` 后缀
5. 修复HTML中的路径引用（CSS/JS/原文/导航链接）
6. 运行 `lint_html.py` 质量检查（章节/首页/实体索引）

八、文件清单

文件	用途	版本
`render_shiji_html.py`	核心渲染器：Token→HTML	-

文件	用途	版本
`generate_all_chapters.py`	批量生成130章	-
`publish_to_docs.sh`	发布流水线	-
`css/shiji-styles.css`	主样式表	**v5.4**
`docs/css/shiji-styles.css`	部署版样式表(同步)	**v5.4**
`docs/css/shiji-styles-v5.4.css`	v5.4存档版本	**v5.4**
`docs/css/chapter-nav.css`	章节导航样式	-
`docs/css/entity-index.css`	实体索引页样式	-
`docs/js/purple-numbers.js`	PN点击复制	-
`docs/js/entity-filter.js`	实体搜索过滤	-
`scripts/lint_html.py`	HTML质量检查	-
`kg/entities/data/entity_aliases.json`	别名映射	-
`kg/entities/data/disambiguation_map.json`	消歧映射	-
`kg/year_ce_map.json`	年份→公元纪年映射	-

时间轴与可视化（04c 输出的应用层展现）

文件	说明	数据来源
`docs/entities/timeline.html`		`build_year_map.py` 四阶段输出

文件	说明	数据来源
	编年索引：835年×（十表诸侯纪年+文本引用+事件）	
`docs/css/timeline.css`	时间轴样式	—
`app/metro/index.html`	史记事件地铁图	04c 事件索引（3185站，横轴=公元纪年，每章一线）
`app/metro/data/metro_map_data.json`	地铁图数据	`kg/events/scripts/build_metro_map_data.py` 生成

九、上游依赖与整合

9.1 上游技能依赖

上游工序	提供	阅读器用途
SKILL_01 古籍校勘	异体字/字典关联	字典链接tooltip
SKILL_02a 章节切分	段落编号（Purple Numbers）	段落锚点导航、精确引用
SKILL_02b 区块处理	太史公曰/赞/韵文围栏块	区块差异化渲染（背景色、字体）
SKILL_02c 三家注	集解/索隐/正义注释层	右栏注释面板、分色badge
SKILL_02d 结构语义分析	**句间/段间语义关系标注**	**因果缩进、并列对照、总结色块、语义排版开关**

上游工序	提供	阅读器用途
SKILL_03a 实体标注	18类实体Token	语法高亮（颜色/样式）
SKILL_03b 实体消歧	别名/消歧映射	实体链接跳转、tooltip
SKILL_03d 渲染与发布	**（已整合到本技能）**	**HTML生成、CSS样式、发布流水线**
SKILL_04c 事件年代	公元纪年映射	纪年tooltip
SKILL_04f 动词标注	4类动词Token	动词语法高亮（v4.0新增）

9.2 从SKILL_03d整合的内容

SKILL_03d（渲染与发布） 原职责现已全部整合到本技能：

1. **HTML渲染**：`render_shiji_html.py`（见 §二 渲染管线）
2. **CSS样式设计**：
   - 实体渲染：见 §三.1 + 实体渲染规划 v5.0
   - 动词渲染：见 §三.2 + 动词渲染规则 v4.0
3. **批量生成**：`generate_all_chapters.py`（见 §二 批量生成）
4. **发布流水线**：`publish_to_docs.sh`（见 §七 发布流水线）
5. **质量检查**：`scripts/lint_html.py`（见 §七 步骤6）

结论：SKILL_03d 作为独立技能已废弃，所有功能并入 SKILL_09a 认知辅助阅读器。

9.3 核心依赖说明

02d 结构语义分析 是核心依赖：
- 没有语义关系标注 → 阅读器只能做实体高亮（一维）
- 加上语义排版后 → 体现文本的逻辑结构（二维）

SKILL 09b: 知识库游戏化 — 用史记知识库设计游戏

> 将知识图谱（实体、事件、关系、SKU）转化为可玩的游戏机制，实现"寓教于乐"。

一、设计理念

知识→游戏的转化路径:

- SKU技能知识 → 技能卡牌（破釜沉舟、约法三章、鸿门脱险等）
- 事件索引 → 剧情场景（商周之变、楚汉相争等）
- 人物实体+关系 → 可招募角色（张良+鸿门宴逃生加成）
- 事件关系（因果/对立） → 技能组合效果（连击/互斥）
- 纪年体系 → 时代背景和回合制时间推进

核心原则：每个游戏元素都有《史记》原文支撑，技能卡附带原文段落引用。

二、当前实现：史记争霸（`app/game/`）

技术栈

- HTML5 + CSS3 + Vanilla JavaScript（无框架依赖）
- 中国风视觉设计（楷体字体、水墨色调、竹简风格UI）
- 响应式设计（支持移动端）

已实现功能

系统	状态	说明
技能卡牌	8/15张	5类：政治/军事/历史/文化/领导
资源管理	✅	四大资源：财富/民心/军力/威望
回合制	✅	情报→决策→执行→事件→结算
剧情模式	第1章	"商周之变"（2个场景）
技能图鉴	✅	查看详情和《史记》原文

文件结构

```
app/game/
├── index.html          # 主页面
├── styles.css          # 样式（中国风）
├── game.js             # 游戏逻辑
├── game_design.md      # 完整设计文档
├── add_chapters.py     # 添加新章节剧情
└── README.md           # 使用说明
```

三、知识库→游戏元素映射

3.1 SKU → 技能卡

每张技能卡直接来源于 `ontology/skus/` 中的技能知识单元：

```
SKU（技能知识）
    ├── name → 卡牌名称（如"破釜沉舟战术"）
    ├── category → 卡牌分类（军事/政治/文化）
```

```
    ├── source_text → 卡牌上的《史记》原文引用
    ├── entities → 关联人物和地点
    └── relations → 技能组合/互斥规则
```

技能卡属性设计:
- 稀有度：1-5星（根据SKU的复杂度和历史影响力）
- 资源消耗：军事类消耗军力+财富，政治类消耗威望+民心
- 冷却回合：高威力技能需等待多回合
- 升级路径：5级，逐步增强效果

3.2 事件索引 → 剧情场景

从 `kg/events/data/` 提取事件链，构建剧情线：

```
事件索引
    ├── 事件类型（战争/继位/政治）→ 场景类型
    ├── 事件关系（因果链）→ 剧情分支
    ├── 公元纪年 → 时代设定
    └── 参与人物 → 场景NPC
```

四章剧情设计:

章	时代	核心事件来源	核心技能
商周之变	前1046年	003殷本纪+004周本纪事件	牧野之战、殷商继承分析
秦国崛起	前221年	006秦始皇本纪事件	废分封、焚书坑儒
楚汉相争	前206-202年	007项羽+008高祖本纪事件	鸿门脱险、破釜沉舟
汉朝建立	前202年后	008高祖本纪事件	约法三章、知人善任

3.3 人物实体+关系 → 角色系统

从 `kg/entities/` 和 `kg/relations/` 构建可招募角色：

```
实体索引（人名）
    ├── 出现章节 → 角色所属时代
    ├── 别名/消歧 → 角色简介
    └── 关系（君臣/师徒）→ 阵营归属

关系网络
    ├── 人物→事件参与 → 角色技能加成
    └── 人物→人物关系 → 阵营内buff/debuff
```

3.4 事件关系 → 技能组合

从 `kg/events/data/event_relations.json` 中的因果/对立关系设计组合效果：

关系类型	游戏效果
因果关系	技能连击（先用A再用B，效果加成）
对立关系	技能互斥（A和B不能同回合使用）
包含关系	技能依赖（解锁A需要先获得B）
共人关系	人物加成（招募角色解锁专属技能）

四、扩展新剧情的方法

添加章节剧情

```
python app/game/add_chapters.py
```

从知识库提取新技能

1. 从 `ontology/skus/` 选择技能知识单元
2. 提取关联的《史记》原文段落（通过Purple Number引用）
3. 设计卡牌属性（分类/消耗/效果/冷却）
4. 关联人物加成（查询 `kg/relations/` 中的角色-事件关系）

设计新剧情线

1. 从 `kg/events/data/` 选择目标章节的事件索引
2. 沿事件链（因果关系）构建场景序列
3. 在分支点（对立关系）设计玩家选择
4. 标注各场景可用的技能卡

五、美术风格指南

- **色调**：金色+棕色主题（竹简和古籍感）
- **字体**：楷体为主，标题用行楷
- **卡牌**：水墨画风格背景 + 《史记》原文片段
- **人物**：传统工笔画风格立绘
- **地图**：古代舆图样式
- **UI**：竹简风格菜单，毛笔书写动画
- **战斗**：兵棋推演风格

六、教育价值设计

每个技能卡包含：
- 历史事件背景说明
- 《史记》原文引用（可跳转到阅读器对应段落）
- 历史人物介绍
- 策略思维训练

学习闭环：游戏中遇到技能 → 查看原文 → 跳转阅读器 → 阅读完整上下文 → 返回游戏应用知识