Ontology Crawler Platform 개선 전략

현재 프로젝트는 폐기 대상이 아니라, 구조는 유지하면서 핵심 파이프라인을 교체 및 강화해야 하는 단계이다.

현재 가장 큰 문제는:

- 웹페이지 본문 추출 실패
- footer/nav/배송정보/이벤트 오염
- LLM JSON 출력 실패
- ontology validation 부재
- fallback 규칙 오염
- entity/page classification 부재

이며,

이 문제는 단순히 LLM 모델 교체로 해결되지 않는다.

핵심은:

```
좋은 본문을 추출하고
→ 안정적으로 구조화하고
→ 검증된 ontology claim만 저장하는 것
```

이다.

현재 다운로드한 OSS 역할 정리

프로젝트	역할	현재 프로젝트에서 사용 목적
playwright	브라우저 렌더링	JS 렌더링 / SPA / 실제 페이지 획득
crawl4ai	AI 친화 크롤링	markdown 변환 / content cleaning
trafilatura	본문 추출	footer/nav 제거 / main content extraction
instructor	LLM JSON 안정화	strict schema output
guardrails	LLM validation	garbage output reject
ontocast	ontology 구조 참고	entity/relation/triple 구조
neo4j-graphrag	graph 구조 참고	ontology graph 저장
knowledge_agent	agent loop 참고	recursive crawl / relevance
OpenDeepResearcher	research flow 참고	탐색 loop 설계
firecrawl	AI friendly crawling 참고	markdown cleaning 구조

프로젝트 방향 재정의

현재 프로젝트는 단순 crawler가 아니라:

```
Semantic Ontology Research Platform
```

방향으로 정의한다.

최종 목표:

```
웹
→ 의미 추출
→ entity/relation 생성
→ ontology mapping
→ graph knowledge 구축
→ semantic exploration
```

현재 문제 분석

문제 1. 페이지 전체를 분석하고 있음

현재:

- footer
- 배송 국가 목록
- 이벤트 배너
- 공지
- navigation
- 게시판 링크
- 정렬 텍스트

까지 ontology claim 생성에 포함되고 있다.

결과:

```
CAFE24 = 브랜드
Green = accord
낮은가격 = entity
상품수 = entity
```

같은 잘못된 결과 발생.

이 문제는:

```
본문 추출 실패
```

문제이다.

문제 2. AI extractor가 실제로 거의 실패 중

현재 로그:

```
Expecting value: line 1 column 1
AI returned no usable entities or claims
```

의미:

- JSON 반환 실패
- 빈 문자열
- markdown/codeblock
- 설명문 반환
- malformed JSON

등이 발생 중.

현재는 fallback rule이 실제 ontology claim처럼 저장되고 있음.

이것은 매우 위험하다.

문제 3. Entity type 분리 없음

현재:

- Product
- Event
- Promotion
- Notice
- BrandStory
- CommunityPost

가 모두 같은 레벨에서 처리된다.

그래서 ontology graph가 오염된다.

문제 4. Validation 없음

현재 저장되는 값:

```
value
accord
keyword
""
```

이것은 ontology 데이터가 아니다.

Placeholder reject 필요.

개선 방향 전체 구조

현재 구조를 아래 파이프라인으로 재구성한다.

```
[Crawler]
→ [DOM/MainContent Extractor]
→ [Page Classifier]
→ [Structured Page Model]
→ [LLM Extraction]
→ [Schema Validation]
→ [Claim Validation]
→ [Ontology Mapping]
→ [Candidate Review]
→ [Graph Merge]
```

1단계 — Playwright 도입

사용 OSS:

- playwright

목표:

```
실제 브라우저 기반 페이지 획득
```

도입 목적:

- JS 렌더링

- lazy loading 대응
- dynamic page 대응
- 실제 DOM 안정화
- SPA 대응

구현:

```
1. 페이지 이동
2. network idle 대기
3. 특정 selector 대기
4. HTML snapshot 저장
5. canonical URL 저장
6. error/captcha 감지
```

추가:

- blocked page 감지
- unavailable page 감지
- region redirect 감지

예:

```
Page unavailable
Access denied
captcha
Reference ID
```

등 발견 시:

```
crawl_failed
```

상태 저장.

2단계 — Trafilatura 본문 추출

사용 OSS:

- trafilatura

현재 프로젝트에서 가장 중요한 단계.

목표:

```
웹페이지에서 실제 읽을 본문만 추출
```

제거 대상:

- footer
- nav
- shipping info
- recommendation
- banner
- copyright
- category
- board links
- country lists

출력:

```
clean_text
clean_html
main_content
```

현재 문제:

```
배송 국가 목록이 accord로 추출됨
```

같은 문제를 여기서 해결.

3단계 — Crawl4AI 도입

사용 OSS:

- crawl4ai

목적:

```
LLM 친화 markdown 생성
```

흐름:

```
HTML
→ markdown
→ cleaned markdown
→ semantic blocks
```

기대 효과:

- 긴 HTML 감소
- token 감소
- semantic extraction 안정화
- hallucination 감소

4단계 — Page Classification 추가

현재 가장 부족한 구조.

도입:

```
PageType classifier
```

분류:

- ProductPage
- BrandStoryPage
- PromotionPage
- CommunityPage
- EventPage
- CategoryPage
- NoticePage
- UnknownPage

예:

```
/product/detail
→ ProductPage

/board/free
→ CommunityPage

/shopinfo
→ BrandStoryPage
```

중요:

ProductPage가 아니면:

```
price extraction 금지
```

등 claim 제한 필요.

5단계 — Instructor 도입

사용 OSS:

- instructor

목표:

```
LLM output strict JSON 강제
```

현재 문제:

```
Expecting value
```

해결 목적.

예:

```
class ProductClaim(BaseModel):
    product_name: str
    brand: str
    price: int
    top_notes: list[str]
```

LLM이 schema에 맞지 않으면 reject.

6단계 — Guardrails Validation

사용 OSS:

- guardrails

목표:

```
garbage ontology reject
```

Reject 대상:

```
value
accord
keyword
```

```
""
null
unknown
n/a
```

추가:

- too short reject
- meaningless token reject
- duplicated relation reject
- invalid entity reject

7단계 — Claim 상태 분리

현재:

```
rule 결과가 ontology claim으로 저장됨
```

문제.

변경:

```
candidate_claim
validated_claim
rejected_claim
ai_claim
rule_candidate
```

분리.

rule 결과는:

```
candidate 상태
```

로만 저장.

8단계 — Ontology Graph 구조 개선

참고 OSS:

- ontocast
- neo4j-graphrag

목표:

```
Entity
→ Relation
→ Triple
→ Graph
```

예:

```
[Product] Cotton Hug
  ├─ hasBrand → Forment
  ├─ hasPrice → 49000 KRW
  ├─ hasTopNote → Pink Pepper
  ├─ suitableForSeason → Summer
```

현재처럼:

```
CAFE24
낮은가격
상품수
```

등은 graph merge 금지.

9단계 — Recursive Research Agent

후반 단계.

참고 OSS:

- knowledge_agent
- OpenDeepResearcher

목표:

```
탐색
→ relevance 판단
→ 링크 확장
→ ontology 성장
```

예:

```
브랜드 페이지 발견
→ 제품 페이지 탐색
→ 리뷰 페이지 탐색
→ 노트 정보 탐색
→ 경쟁 브랜드 탐색
```

추천 구현 순서

Phase 1

핵심 안정화.

```
Playwright
Trafilatura
Page Cleaner
```

우선.

현재 프로젝트의 가장 중요한 문제는:

```
LLM 이전 단계 실패
```

이기 때문.

Phase 2

```
Instructor
Guardrails
```

도입.

LLM JSON 안정화.

Phase 3

```
Page Classification
Claim Validation
Candidate Separation
```

추가.

Phase 4

```
Neo4j Graph Merge
Ontology Relation
Semantic Query
```

추가.

Phase 5

```
Knowledge Agent
Recursive Research
Semantic Expansion
```

추가.

최종 방향

이 프로젝트는:

```
단순 크롤러
```

가 아니라:

```
Semantic Knowledge Construction Platform
```

으로 발전해야 한다.

핵심은:

```
웹을 읽는 것
```

이 아니라:

```
웹의 의미 구조를 구축하는 것
```

이다.

따라서:

- 좋은 본문 추출
- strict validation
- ontology integrity
- semantic relation
- graph quality

가 가장 중요하다.

현재 문제는:

```
LLM 성능 부족
```

보다 먼저:

```
좋은 데이터를 제대로 추출하지 못하는 것
```

이다.