송련 코어 README

Source: README.ko.md - Generated: 2026-06-30 18:15 KST

키워드: LLM 에이전트, 출처 추적, 런타임 정직성, traceability, 로컬 우선 AI, smoke-tested agent architecture.

송련 코어는 LLM 에이전트가 코드가 확인한 사실과 모델이 해석한 말을 섞어 말하지 않게 만드는 작은 런타임 실험입니다.

목표는 단순합니다. 에이전트가 답변할 때 사실, 추측, 요약, 도구 결과, 내부 라우팅 판단을 한 덩어리의 그럴듯한 문단으로 뭉개지 않게 하는 것입니다.

피드백을 받고 싶습니다

송련 코어는 아직 초기 연구/런타임 프로토타입입니다. 특히 아래 질문에 대한 피드백을 받고 싶습니다.

- 절대정보 / 상대정보 / 혼합정보 구분이 더 신뢰할 수 있는 에이전트를 만드는 데 유용해 보이나요?
- 송련은 에이전트 런타임, 감사 레이어, 연구 프로토타입 중 무엇으로 보이나요?
- 다른 개발자가 작은 신뢰 에이전트를 조립할 수 있게 하려면 무엇을 먼저 단순화해야 할까요?
- Qwen 단독과 SongRyeon 런타임 비교가 프로젝트의 가치를 잘 보여주나요?

비판, 혼란스러운 점, LangGraph/LangSmith/Dify/OpenHands/SWE-agent 같은 프로젝트와의 비교 의견은 feedback issue에 남겨 주세요.

실제로 뭐가 다르냐

보통 에이전트는 이렇게 말할 수 있습니다.

> 문서 3개를 읽었고 충분한 근거를 찾았습니다.

송련 코어는 되도록 이렇게 말하게 만들려고 합니다.

> 코드가 확인한 count:
> - reportable_documents = 2
> - raw_extract_records = 3
> - empty_extract_records = 1
>
> LLM의 의미 판단:
> - 읽을 수 있는 문서는 2개였으므로 답변은 부분 답변으로 제한된다.
>
> 런타임 정직성:
> - 상위 L 재라우팅 요청은 정책에 의해 차단되었다.
> - 최종 보고는 오래된 legacy ID가 아니라 최신 L run을 기준으로 한다.

이 차이가 이 프로젝트의 핵심입니다.

무엇을 추적하나

송련 코어는 런타임 정보를 세 종류로 나눕니다.

- 절대정보: 코드, 스키마, 파일, trace event, data record로 존재와 값을 확인할 수 있는 정보.
- 상대정보: 하나의 특정 source record 또는 field에 직접 대응하는 의미 판단.
- 혼합정보: 하나의 source로 못 박으면 부정확한, 여러 source 묶음에 근거한 종합 판단.

짧게 말하면 이렇습니다.

> 코드 사실은 코드 사실로 둔다.
> LLM 판단은 LLM 판단으로 둔다.
> 여러 근거를 종합한 말은 여러 근거를 종합했다고 드러낸다.

왜 만들었나

많은 에이전트 데모는 처음 보면 그럴듯합니다. 하지만 조금만 파고들면 이런 질문이 나옵니다.

- 이 말은 코드가 확인한 사실인가, LLM이 해석한 말인가?
- 최종 답변은 어떤 내부 단계에서 나온 것인가?
- LLM 라우터가 실패했는데 코드 fallback이 조용히 대신 판단한 것은 아닌가?
- L 루프가 두 번 돌았을 때 최신 run을 보고 있는가, 예전 기록을 보고 있는가?
- 최종 답변에 나온 숫자는 LLM이 센 것인가, 코드가 센 것인가?

송련 코어는 이런 질문이 런타임 안에서 보이게 만드는 작은 로컬 실험입니다.

현재 하이라이트

- TraceStore와 DataStore를 통한 사건/데이터 출처 추적.
- 내부 문서 검색과 근거 수집을 위한 L 루프.
- 최종 보고서의 grounding count를 code가 고정 생성.
- 라우터 fallback 정직성 기록.
- same-turn L reroute guard.
  - 기본값은 L 1회.
  - 정책을 켜면 L 2회.
  - 3회차 이상은 차단.
- 최근 턴 capsule과 raw conversation alignment packet.
- 상대정보/혼합정보 분리 및 smoke-test.
- pretty runtime 출력에서 생성자, 정보 등급, source ID, 의미 판단 상태 표시.

추천 GitHub Topics

GitHub 저장소 오른쪽 About 영역에 아래 topics를 붙이면 검색과 유입에 도움이 됩니다.

```
llm
agents
python
local-first
provenance
traceability
runtime
agent-architecture
```

빠른 실행

전체 로컬 기준선은 pytest를 dev/test dependency로 사용합니다. 기존 CLI smoke-test는 계속 `python main.py smoke-test`로 실행합니다.

```
python -m pip install -r requirements-dev.txt
python -m compileall songryeon_core main.py
python -m pytest
python main.py smoke-test
```

기대 결과:

```
SMOKE_TEST_OK
```

실제 LLM 없이 deterministic fake turn 실행:

```
python main.py fake-turn "송련의 문서 메모리 인덱스가 무엇인지 알려줘" --pretty
```

dry run 실행:

```
python main.py dry-run
```

선택 기능: 로컬 LLM

Qwen 경로는 선택 기능입니다. Ollama와 호환 모델이 있으면 다음처럼 실행할 수 있습니다.

```
pip install ollama
python main.py qwen-ping --timeout 60
python main.py qwen-turn "송련의 문서 메모리 인덱스가 무엇인지 알려줘" --timeout 120 --pretty
```

`QWEN_LOCAL_ENDPOINT`를 OpenAI 호환 로컬 HTTP endpoint로 지정해서 쓸 수도 있습니다.

설계 원칙

1. 절대정보는 코드가 쓴다.
2. 의미 판단은 LLM이 쓴다.
3. 혼합정보는 source bundle을 드러낸다.
4. 코드는 LLM인 척하지 않는다.
5. LLM 판단을 코드 사실처럼 보여주지 않는다.

6. 휴리스틱은 숨기지 않고 명시적 정책으로 둔다.
7. smoke-test 없는 데모 감성에 속지 않는다.

현재 기준선

2026-06-27 기준:

- `python -m compileall songryeon_core main.py` 통과.
- `python -m pytest` 통과.
- `python main.py smoke-test` 통과.
- pytest는 import, schema split compatibility, 도메인별 smoke case를 검사함.
- 하나의 source field에 직접 대응하는 claim은 relative info로 테스트됨.
- source bundle 기반 planner claim은 mixed info로 유지됨.
- node_3 report grounding count는 code가 공급함.
- node_4는 위험하거나 count가 맞지 않는 report를 차단할 수 있음.

테스트 계층:

- `compileall`: 문법/임포트 최소 검사.
- `pytest`: 단위/도메인 회귀 검사.
- `smoke-test`: 통합 런타임 기준선 검사.
- `qwen-turn` / `qwen-chat`: 수동 live LLM 검사이며 CI 필수 조건이 아님.

폴더 구조

- `songryeon_core/core/`: schema, trace store, data store, registry, failure signal.
- `songryeon_core/state/`: zero state, unified state, turn capsule helper.
- `songryeon_core/nodes/`: node 구현.
- `songryeon_core/loops/`: L loop runtime과 loop policy.
- `songryeon_core/tools/`: 문서 도구, hash embedding 검색, tool result distillation.
- `songryeon_core/llm/`: LLM adapter, fake adapter, Qwen/Ollama adapter.
- `songryeon_core/runtime/`: dry run, user turn, terminal view, smoke test, replay.
- `songryeon_core/prompts/`: node별 prompt 파일.
- `Administrative_Reform_1/`: 설계 문서, 지도, 발주서, 실행 기록.
- `main.py`: CLI 진입점.

참고

이 프로젝트는 production assistant가 아닙니다.

송련 코어는 provenance, runtime honesty, agent self-reporting을 공부하기 위한 학습형 아키텍처 프로토타입입니다. 겉보기에 화려한 데모보다, 기록이 남고 검증 가능한 작은 MVP를 우선합니다.