

Jev 사용법

확률 기반 판정을 안전한 애플리케이션 동작으로



Jev는 무엇을 **작성할지**가 아니라, 주어진 근거에서 무엇을 **판단할지**에 특화된 모델이다. 모델이 의미를 판정하고, 코드가 정책과 실행 권한을 통제한다.

이 리포트에서 다루는 것

- Noul · Choice · Score의 실제 계약
- API 요청부터 보류·검토까지의 흐름
- RAG · 에이전트 평가 · 이미지 · 채용
- 평가·운영 + 영상·공간·null 연구 확장

읽기 전에

- 공개 문서 기반 · 독립 커뮤니티 가이드 · Public Edition
- 공식 문서의 현재 사양을 추가 대조
- 작성자 성능 주장 ≠ 독립 벤치마크
- 유료 모델 호출·제품 배포는 수행하지 않음

핵심 결론

타입이 맞는 답변도 틀릴 수 있다. **판정 정확성, 데이터 권한, 실행 권한**은 각각 검증해야 한다.

문서 대상 · 개발자 / 기술 PM / 평가·운영 담당자

산출물 · PDF + 오프라인 HTML + 편집 가능한 SVG/생성기

근거 시점 · 2026-10-20 공식 웹 문서 조회

읽는 순서와 근거 수준

기본 아키텍처와 책임 경계	03	Noul · Choice · Score 읽는 법	04
데이터·API 계약	05	첫 요청: 낮은 위험의 티켓 분류	06
판정에서 행동으로: fail-closed 정책	07	RAG: 검색과 생성 사이의 검토층	08
에이전트 평가: JevEval과 Opik	09	SQL 검토와 에이전트 메모리	10
이미지·PDF·영상: 입력 경로를 분리한다	11	채용·포트폴리오 검토	12
생태계: 모델·도구·관측을 구분한다	13	공급자 교체는 adapter와 재평가로	14
평가·확률 보장·비용	15	운영: shadow부터 시작한다	16
TypeSafe: HTTP와 SDK의 계약을 구분한다	17	Confidence: 확률·척도·정답률은 다르다	18
한도·오류·보관: 운영 전 확인할 계약	19	Provider·이미지: 공통화하지 말아야 할 차이	20
DeepEval·Opik: 점수 1도 실패할 수 있다	21	Beacon·평가 튜토리얼: 기록에서 검토된 지식으로	22
AnyJev·JevGrep·Opik 예제: 채택할 부분만	23	Unsloth·Colab: 자체 모델 학습의 재현 조건	24
소셜 원문: 활용 아이디어와 사실 정정	25	구현: 판단과 실행 사이에 경계를 둔다	26
검증: 정상 입력보다 실패 경로를 먼저	27	스킬로 사용하기: 설계부터 오프라인 구현까지	28
공개판 범위와 안전한 재사용	29	링크별 독해 기록 1	30
링크별 독해 기록 2	31	링크별 독해 기록 3	32
링크별 독해 기록 4	33	링크별 독해 기록 5	34
검증 범위와 다음 적용 단계	35	연구 확장: VideoJev · MJev-Spatial · Jeva	36-43

이 문서는 원문을 복사한 것이 아니라 기술 가이드로 재구성한 것이다

공개 기술 문서와 저장소를 바탕으로 개념·설계·도입 절차를 정리했다. 문서에 기록된 버전·가격은 작성 시점의 참고값이며, 실제 도입 전 공식 문서에서 다시 확인해야 한다.

표시	해석
공식 문서	API·모델·SDK의 공개 계약. 실제 계정에서의 호출 성공을 뜻하지 않음.
프로젝트 설명	저장소 저자·제품 소개의 주장. 운영 준비도·벤치마크 재현 여부와 별개.
권장 설계	이 리포트가 제안하는 정책·보안·운영 구조. Jev가 자동 제공하는 기능이 아님.
원문 관찰	공개 게시물의 사례 설명. 원본 미디어 전체 검증·벤치마크 재현을 뜻하지 않음.

기본 아키텍처와 책임 경계

Jev는 **state와 질문**을 함께 받아 질문별 타입 답변을 반환한다. 자유 형식 설명, 코드 생성, 웹 크롤링, DB 실행은 담당하지 않는다. 넓은 질문을 원자 판단으로 쪼개고 결과의 조합은 코드에 둔다. [S1-S3]

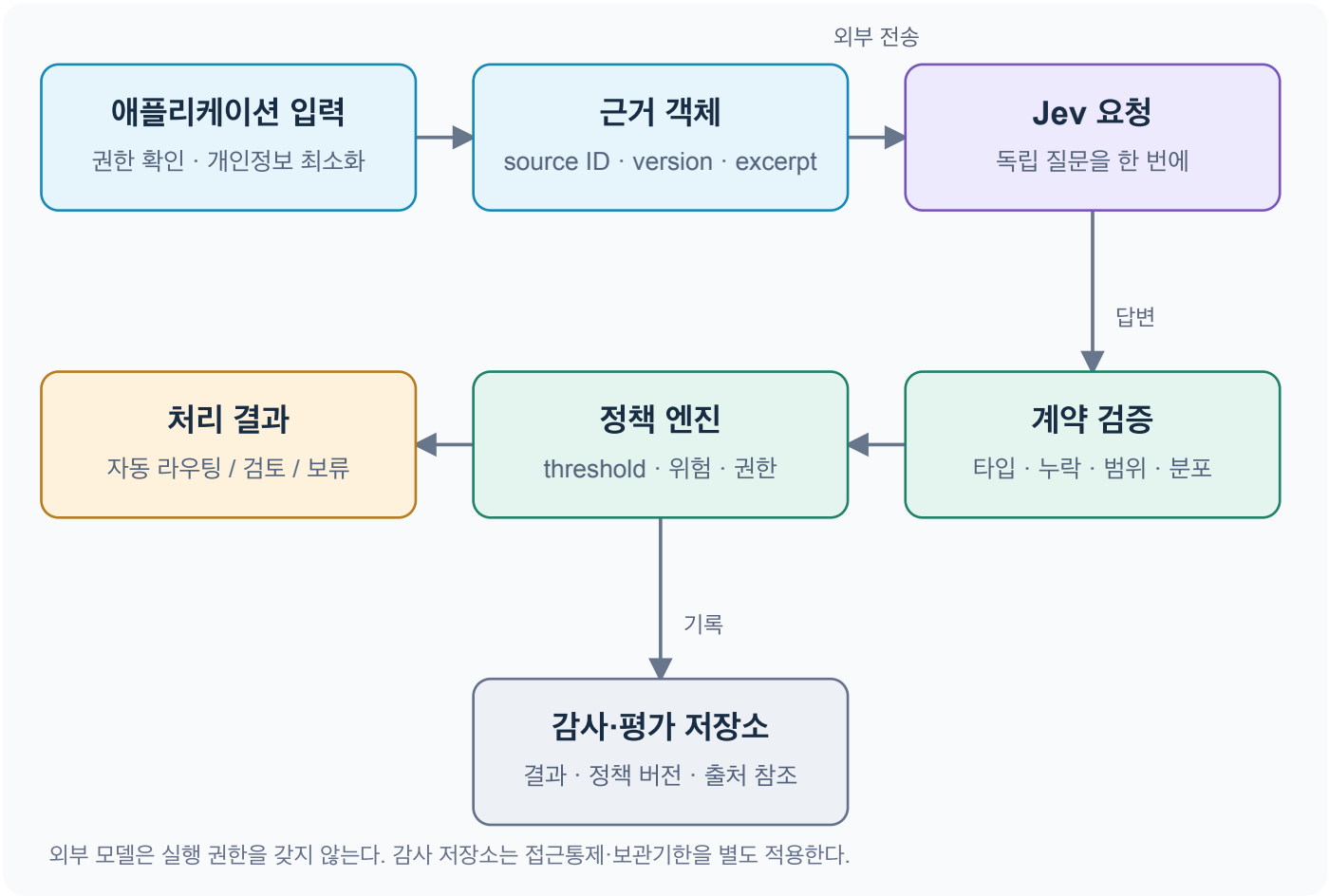


그림 1. 권장 결정 계층 · 입력·외부 추론·정책·감사를 분리한다.

담당	해야 하는 일	맡기지 않을 일
Jev	의미적 관련성·분류·등급 판정	최종 권한 승인, 자의적 도구 실행
일반 코드	파싱/검증, 정책, 임계값, 재시도, 감사	모델 확률을 정확성 보증으로 해석
생성형 LLM	설명·요약·SQL·계획 작성	생성한 설명을 Jev의 실제 사고 과정으로 표시
사람	고위험·불확실·신규 케이스 검토	근거 부재를 자동으로 부정 판정

권한 경계

높은 확신은 권한을 만들어 주지 않는다. 인증·허용 목록·결제 승인·개인정보 동의는 모델과 독립적인 코드 관문이다.

Noul · Choice · Score 읽는 법

타입	입력 설계	답변 해석
Noul	참/거짓 명제 1개. 필요하면 true/false 기준 추가.	$noul = P(\text{true})$, 0~1. 별도의 confidence 필드 없음.
Choice	순서 없는 선택지 맵. 선택지 설명을 서로 구분.	최고 확률의 choice + 전체 probabilities + confidence. 최대 255개 선택지.
Score	낮은→높은 순서의 criteria 배열, 2~10개 레벨.	레벨 인덱스의 확률 가중 평균. 0~(레벨 수-1). 정수가 아닐 수 있음.

세 타입 모두 “설명 생성”이 아니다. 다중 라벨 분류라면 하나의 Choice에 여러 정답을 강요하지 말고, 라벨별 Noul을 독립 질문으로 구성한다. [S4-S6]

Score와 confidence를 혼동하지 않기

```
# 공식 Score 문서 예제 분포를 재계산 (실제 API 호출 결과 아님)
P(level=0,1,2) = [0.00, 0.57, 0.43]
score = 0*0.00 + 1*0.57 + 2*0.43 = 1.43
normalized_score = 1.43 / (3-1) = 0.715

# Choice, 3개 선택지, 최고 확률 0.90인 설명용 계산
confidence = (p_max - 1/n) / (1 - 1/n) = 0.85
```

confidence는 정답률이 아니다

Choice confidence는 최대 확률을 균등 분포 기준으로 변환한 값이다. Score confidence는 레벨 간 거리도 반영한다. **confidence=0.85**를 “정답률 85%”라고 표시하면 안 된다. [S6]

질문 설계 체크리스트

- 한 질문에 의미 판단 하나만 넣는다. 품질·안전·완료 여부를 섞지 않는다.
- Choice의 other/unknown은 유용하지만 만능이 아니다. “어느 선택지도 적합한가?”라는 별도 Noul도 평가한다.
- 같은 요청의 질문은 서로의 답을 보지 못한다. 앞선 답에 의존하는 질문은 다음 요청으로 보낸다.
- Score의 각 레벨은 관찰 가능한 차이로 정의한다. ID 이름만으로 모델에 설명했다고 가정하지 않는다.
- 높을수록 위험한 점수와 높을수록 좋은 점수는 방향을 맞춘 뒤 결합한다. 평균으로 치명적 위반을 상쇄하지 않는다.

데이터·API 계약

TypeSafe 직접 호출의 기본 경로는 POST `https://api.typesafe.ai/v1/systemone`이다. 인증은 서버의 Bearer 토큰으로 처리하며 브라우저 번들·문서·로그에 키를 넣지 않는다. [S2]

필드	의미	구현상 주의
model	별칭 또는 버전 모델 ID	평가 완료 버전을 고정. 응답 model도 기록.
state	문자열 / 객체 / 배열	모든 질문이 공유하는 근거. 최소화·익명화·길이 제한.
questions	질문 ID → 질문 객체	type, instructions, criteria. ID는 결과 연결용.
answers	질문 ID → 타입 답변	요청한 모든 ID의 존재·타입·범위·유한값 검사.
usage	입출력 토큰	실제 과금 토큰 기준으로 비용 기록. 공급자별 추가 필드는 별도.

권장 evidence 객체 — 애플리케이션 내부 형식

```
{
  "record_id": "ticket-example-001",
  "evidence": [{
    "source_id": "ticket-body",
    "source_version": "revision-1",
    "excerpt": "구독 요금이 두 번 청구되었습니다.",
    "content_type": "text/plain",
    "extraction_status": "complete"
  }],
  "context": {"task": "support-routing", "language": "ko"}
}
```

위 필드는 TypeSafe 예약 스키마가 아니라 **권장 내부 계약**이다. 상세 URL·개인 식별자는 로컬 권한 저장소에 두고 모델에는 판정에 필요한 발체와 익명 ID만 보낼 수 있다.

컨텍스트 한도는 두 개다

TypeSafe 기본값은 요청당 64k, 질문당 32k tokens. state+모든 질문과 state+개별 질문이 각각 한도 안에 있어야 한다. Long Context 헤더로 두 한도를 262k로 높일 수 있지만 공식 문서는 속도·정확도 저하를 경고한다. 문자 수와 토큰 수는 같지 않다. [S3]

최신 버전 ≠ latest 별칭

2026-10-21 별칭 확인: 공식 모델표는 1.14를 최신 릴리스로 소개하지만 `jev-latest` → `jev-1.13.0`, `jev-preview` → `jev-1.14.0`로 기재한다. 운영은 별칭보다 검증한 버전 ID를 사용한다. [S3]

첫 요청: 낮은 위험의 티켓 분류

아래는 공식 HTTP 스키마에 맞춰 작성한 **설명용 요청**이다. JSON 파일 생성·문법 확인은 수행하되 유료 API 호출은 하지 않았다. 응답 숫자나 분류 정확도를 실측으로 제시하지 않는다. [S2, S7]

```
{
  "model": "jev-1.14.0",
  "state": {
    "message": "구독 요금이 두 번 청구되었습니다. 확인해 주세요."
  },
  "questions": {
    "department": {
      "type": "choice",
      "instructions": "이 문의의 주 담당 팀은 어디인가?",
      "criteria": {
        "billing": "청구·결제·환불 문의",
        "technical": "제품 오류·기술 연동 문의",
        "account": "로그인·계정 관리 문의",
        "other": "위 범주에 속하지 않음"
      }
    },
    "evidence_sufficient": {
      "type": "noul",
      "instructions": "문의의 담당 팀을 정할 근거가 message에 충분한가?"
    },
    "urgency": {
      "type": "score",
      "instructions": "message에 명시된 업무 영향의 긴급도는?",
      "criteria": [
        "긴급한 업무 영향이 명시되지 않음",
        "일부 업무 지연, 우회 가능",
        "핵심 업무 중단, 우회 불가"
      ]
    }
  }
}
```

보안 저장소로 환경 변수를 주입한 뒤 호출

```
curl --fail-with-body --silent --show-error https://api.typesafe.ai/v1/systemone -H
"Authorization: Bearer $TYPESAFE_API_KEY" -H "Content-Type: application/json" --data-
binary @example-request.json
```

운영에서는 타임아웃·응답 스키마 검증·비밀 제거 로깅을 추가한다. SDK를 쓴다면 Python `typesafe-sdk` 또는 JS `@typesafe-ai/sdk`를 사용하며 버전과 lockfile을 고정한다. **담당 팀 분류가 환불 실행 승인으로 이어져서는 안 된다.**

판정에서 행동으로: fail-closed 정책

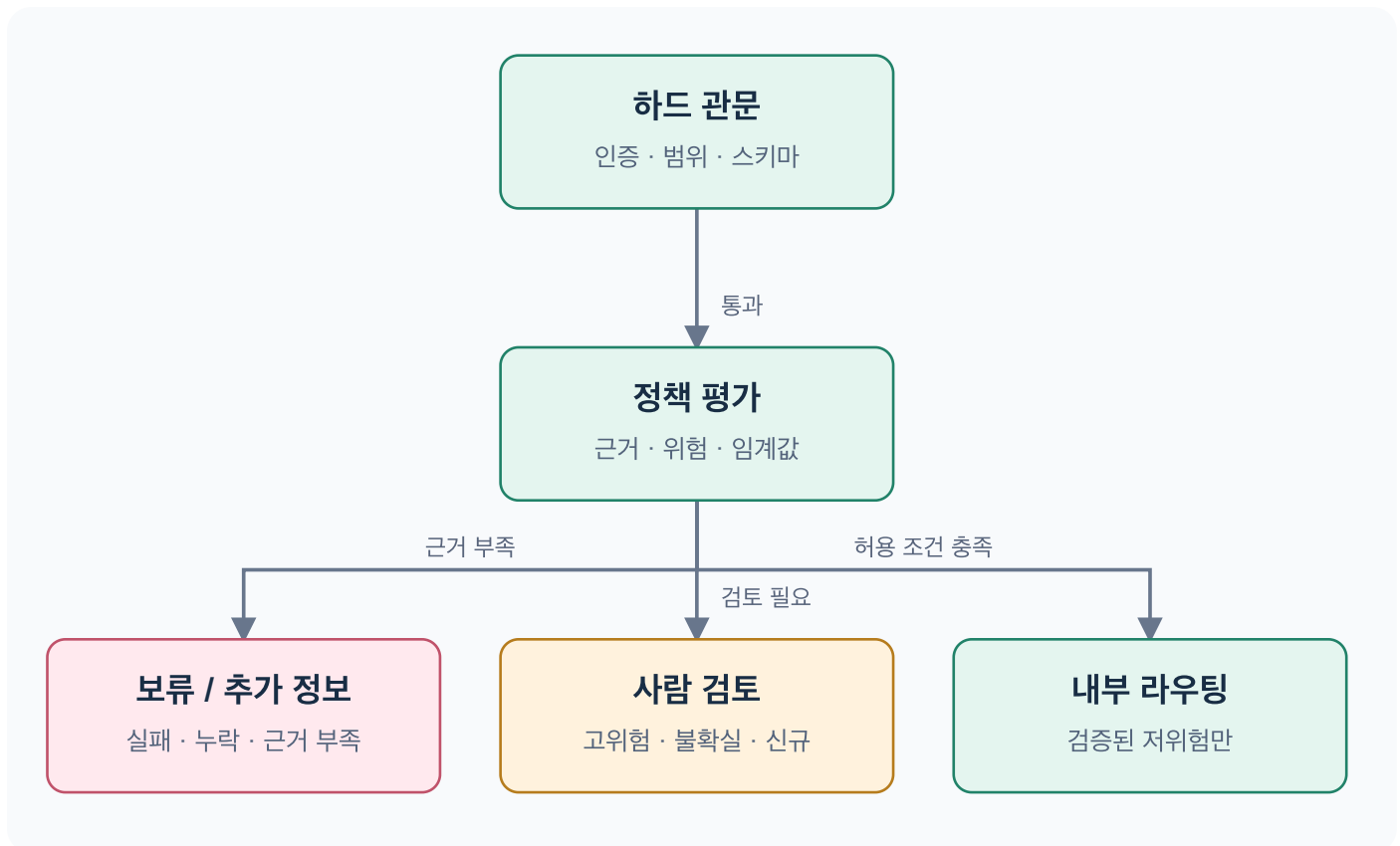


그림 2. 모델 출력 이후의 정책 분기 · 하드 관문 실패는 모델 호출/자동 행동 전에 중단한다.

```

# 설명용 의사코드: 임계값은 평가셋에서 결정한 설정값
if not authorized or not input_valid:
    return DENY
result = decision_provider.evaluate(state, questions)
if result.error or result.refused or not schema_valid(result):
    return HOLD
if evidence_missing(result) or not scope_supported(result):
    return ASK_FOR_MORE_EVIDENCE
if high_impact_action or needs_review(result, calibrated_policy):
    return HUMAN_REVIEW
return ROUTE_TO_INTERNAL_QUEUE # 금전·권한 변경 없음
  
```

- unknown / missing / error / refusal을 false 또는 0점으로 바꾸지 않는다.
- 자동 처리율을 높이려고 API 실패를 “통과”로 대체하지 않는다.
- 금전·계정 권한·채용 결과는 별도 승인·감사·되돌리기 경로를 둔다.
- 정책 결정 이유는 적용된 규칙 ID와 근거 참조로 설명한다. Jev의 내적 사고를 만들어내지 않는다.

RAG: 검색과 생성 사이의 검토층

검색기는 후보를 찾고 Jev는 후보의 의미적 유용성을 판단한다. Jev는 검색 후보 밖의 문서를 발견하지 못한다. 문단 관련성과 “답할 만큼 충분한가”는 별도 기준이다.

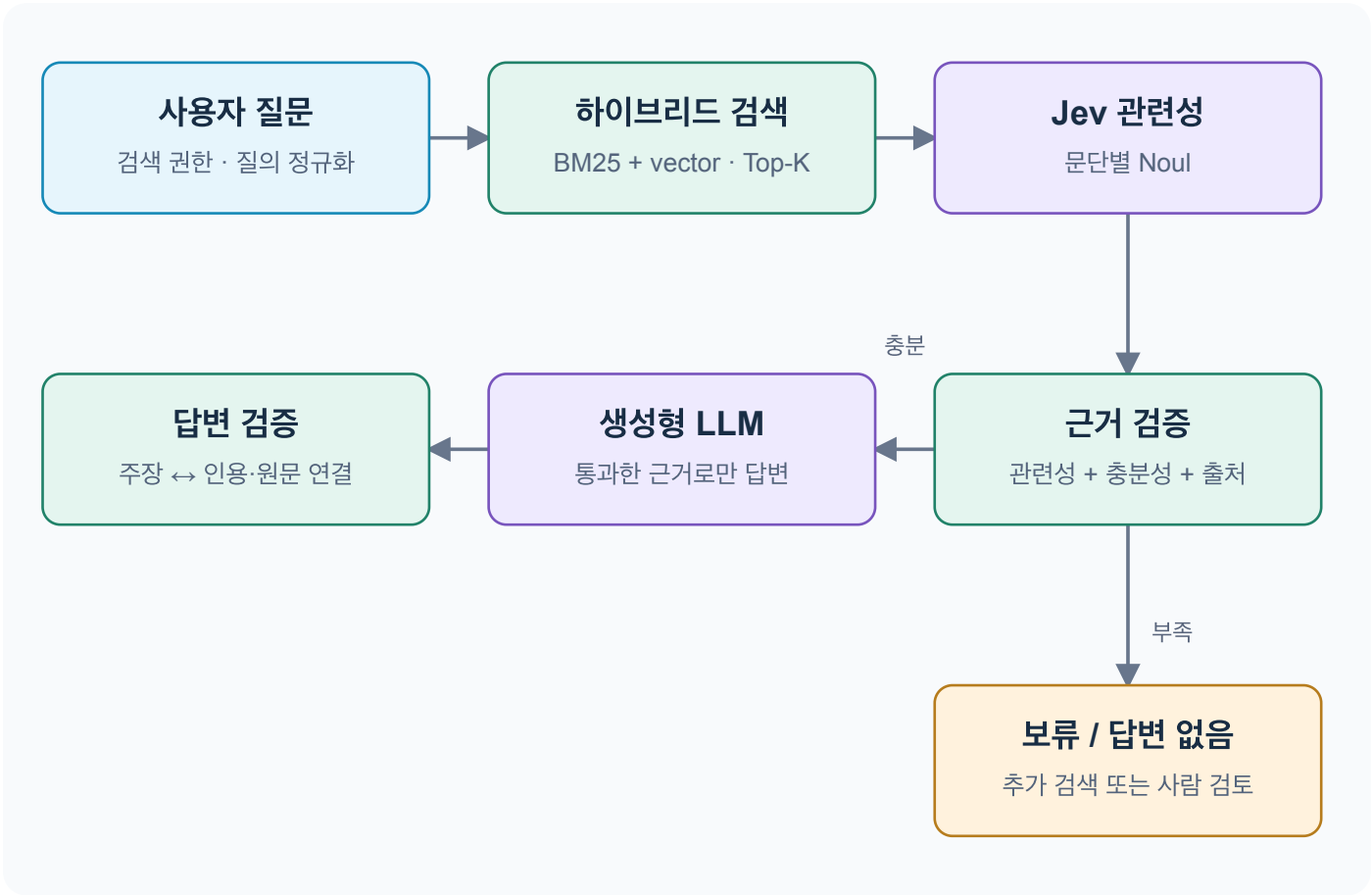


그림 3. RAG evidence gate · 원문의 RAG 설명을 기반으로 한 권장 구조. [원문 관찰, S1]

단계	검증할 것
검색	Top-K recall, 테넌트/사용자 접근권한, 최신성, 중복 제거
문단 판정	질문과의 직접 관련성, 필요한 사실 포함, 잘린 문맥 여부
충분성	여러 질문 요소에 대한 coverage, 상충 근거, 날짜·단위 일치
생성 후	인용 문단이 실제 주장을 지지하는지, 추가된 무근거 주장 여부

프롬프트 인젝션은 별도 보안 문제

검색 문서의 명령문을 실행 지침으로 신뢰하지 않는다. Jev의 인젝션 분류는 보조 신호일 뿐이다. 도구 allowlist·권한 검사·외부 전송 정책은 독립 관문으로 유지한다.

에이전트 평가: JevEval과 Opik

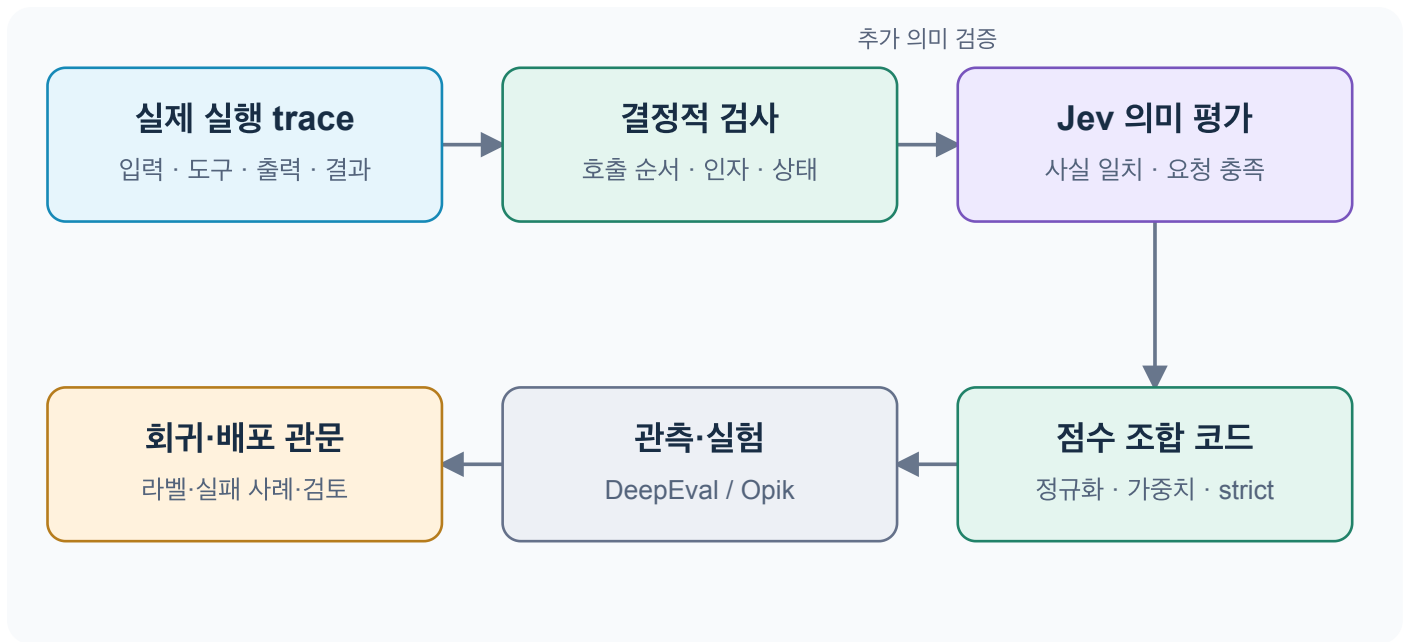


그림 4. 규칙 검사 → 의미 판정 → 회귀 · 규칙 검사에서 판별 가능한 사실을 모델에 재판정시키지 않는다. [S8]

JevEval은 Jev 답변을 코드로 결합하는 **DeepEval의 지표**다. Opik은 trace·데이터셋·실험 결과를 저장·비교하는 **관측 플랫폼**이다. 둘을 함께 쓸 수 있지만 둘 다 Jev 모델 자체는 아니다.

JevEval 동작	정확한 의미
일반 모드	Noul은 P(true), Score는 기대 레벨을 0~1로 정규화, Choice는 적용되는 선택지의 credit 가중 확률. 그 뒤 질문별 가중 평균.
N/A Choice	credit=None인 선택지 확률 합이 0.5 이상이면 해당 질문은 지표에서 제외. 누락된 테스트 근거를 임의로 N/A로 처리하면 안 됨.
strict 모드	Noul ≥ 0.5 , Score 최빈 레벨이 최고 레벨, Choice 최빈 적용 옵션의 credit=1.0일 때 개별 통과. 적용 질문이 모두 통과해야 최종 1.
reason	답변을 바탕으로 코드가 만드는 설명. 생성형 rationale이나 모델 사고 과정이 아님.

strict는 확률 100%를 요구하는 모드가 아니다

strict의 이진 규칙과 안전 임계값은 구분한다. 전부 N/A이면 score=10이므로 적용 질문 수가 0인 경우 별도 보류한다. high-risk 관문을 DeepEval 기본 threshold나 평균 점수에 위임하지 않는다. [S8]

trace 기반 평가에서 evaluation_params를 생략한 JevEval은 span tree를 사용하며 토큰·비용·프롬프트를 제외한다고 문서가 설명한다. 그런 필드를 보지 못하는 질문으로 비용 적정성을 평가하지 않는다.

SQL 검토와 에이전트 메모리

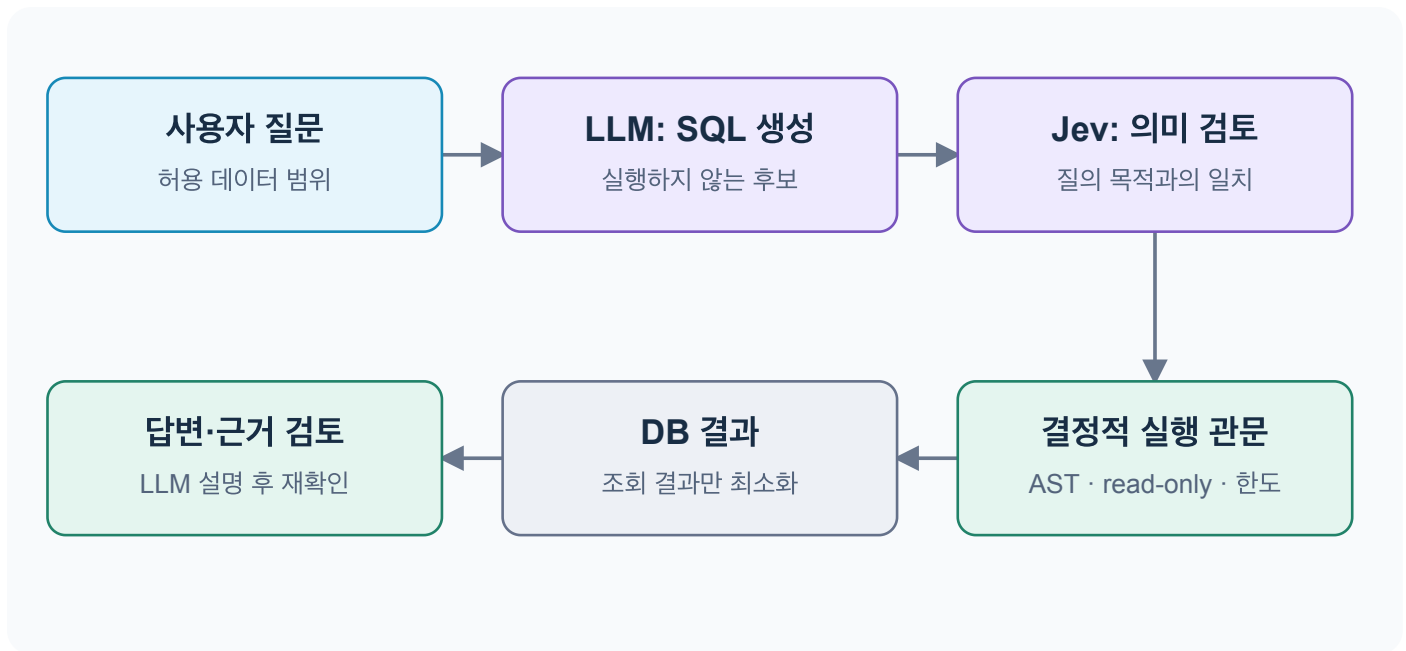


그림 5. 데이터 분석 에이전트의 이중 검토 · 화살표는 처리 순서. 도식은 권장 설계이며 설치·배포된 시스템의 실측도가 아니다.

SQL 실행의 보안 경계는 코드와 DB 권한이다

read-only 계정, 허용 스키마·테이블, AST 기반 정책, 행수·시간·비용 한도, 파라미터 바인딩을 사용한다. Jev가 “요청과 일치”라고 판단해도 임의 SQL을 실행하지 않는다.

Beacon·JevGrep은 어디에 놓이는가?

도구	역할	Jev와의 연결
Beacon	harness 실행 기록·세션 간 지식 재사용	공식 opt-in memory 평가가 Jev를 호출한다. 일반 capture/hooks는 호출하지 않는다. [S12]
JevGrep	자연어 동작 설명에서 관련 코드·경로·발체 탐색	검색 결과는 원본 파일을 읽을 출발점. 코드 변경·테스트 성공을 보장하지 않음.
Jev 판정 추가	기억의 재사용성·근거 충실도·검색 관련성 검토	커밋 SHA·원문 위치·유효기간을 함께 기록. 낡은 기억은 갱신·폐기.

Beacon은 `beacon memory evaluations run --dry-run`으로 전송 전 범위를 확인하도록 문서화한다. 실제 평가 명령은 제한·가림 처리한 trace를 외부로 보낼 수 있다. Jev는 교훈 문장을 생성하지 않고, 사람이 원본 trace를 검토해 승인한다. Beacon Cloud는 별도로이며 현재 문서는 evaluator가 없다고 설명한다. [S12] 외부 사례의 실행 환경·영상 전체 재현은 이 가이드의 검증 범위에 포함하지 않는다.

이미지·PDF·영상: 입력 경로를 분리한다

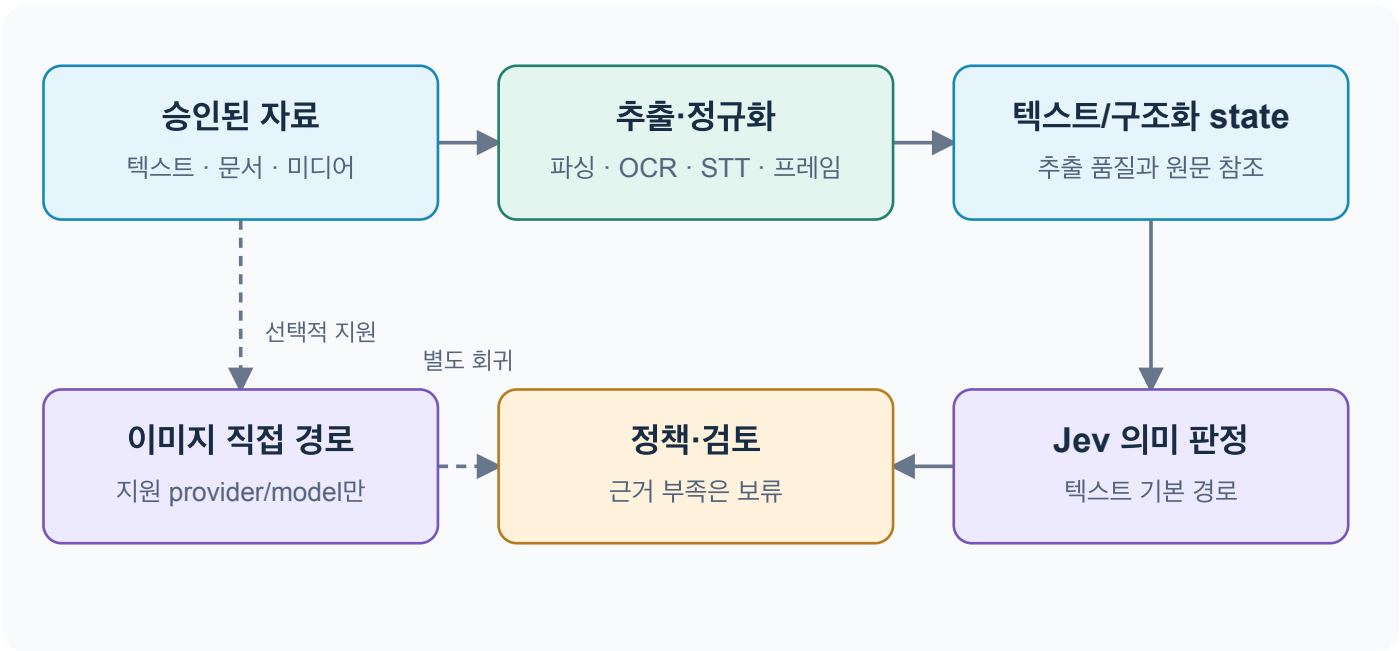


그림 6. 추출 경로와 이미지 직접 경로 · 텍스트 변환과 이미지 자체 판정은 동등한 검증이 아니다.

TypeSafe 이미지·Box / 2026-10-21 확인 · S13 Images · S14 Box

버전	이미지 입력	Box 위치 판정
Jev 1.13 jev-1.13.0	지원 · opt-in 필요	미지원 · 1.14 필요
Jev 1.14 jev-1.14.0	지원 · opt-in 필요	지원 · opt-in 필요

기본 OFF · Dreamer-ready / not production-ready

이미지·Box 사용 시 **매 요청**에 `TypeSafe-Danger-Zone-Images: true`가 필요하다. 모델 선택만으로 활성화되지 않는다. PNG/JPEG 를 assets에 넣고 `state.instructions.criteria`에서 참조한다.

Box는 부재에도 항상 좌표를 반환한다. not-found 결과가 없으므로 존재 여부 Noul을 **같은 요청**에 넣고 존재 확률이 낮으면 Box를 무시한다. Box에는 `criteria`가 없다. 좌표·confidence 주의를 p20 참고.

PDF는 파싱/OCR, 영상은 전사·프레임 추출 경로를 구분한다. OCR 텍스트 검증은 이미지 전체 검증이 아니다. OpenRouter 형식도 TypeSafe와 다르다. **기존 오프라인 starter는 텍스트 전용**이며 `assets·Box`를 지원하지 않는다. 별도 연구 모델과 입력 검증은 36-43쪽에서 구분한다.

채용·포트폴리오 검토

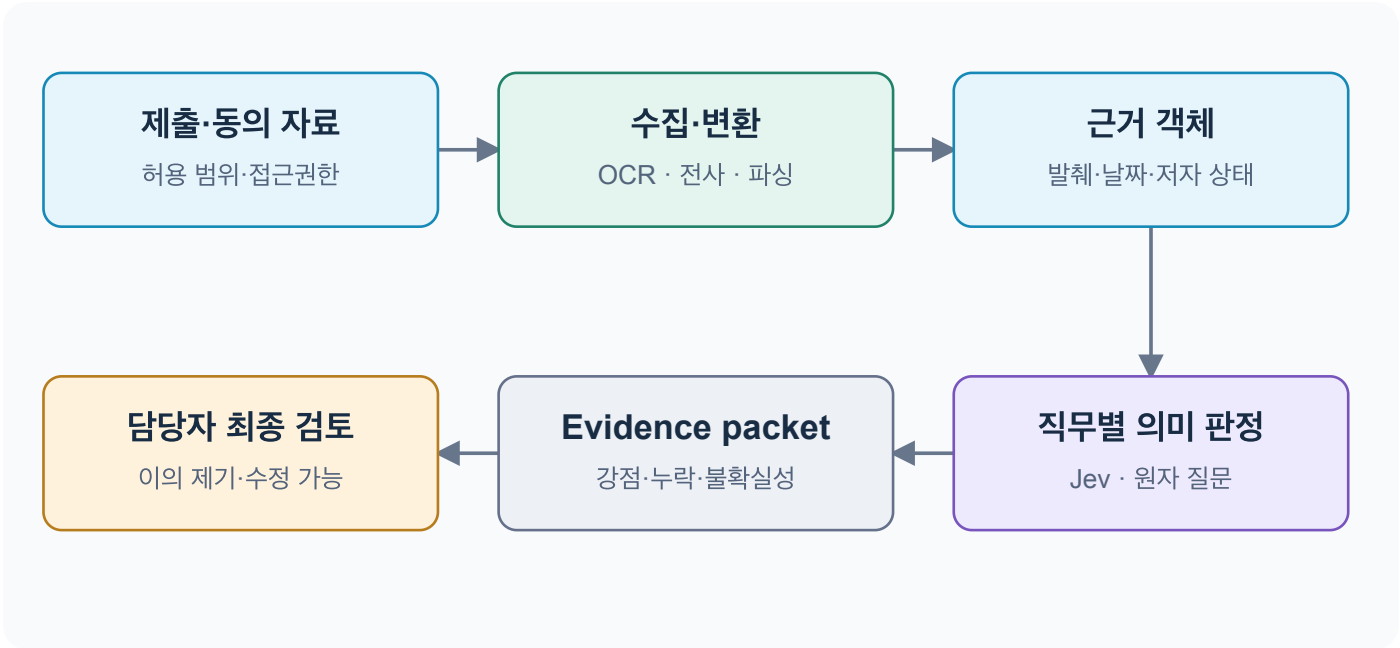


그림 7. 채용 포트폴리오 검토 지원 · 화살표는 처리 순서. 도식은 권장 설계이며 설치·배포된 시스템의 실측도가 아니다.

다음 도식은 특정 제품의 실측 사례가 아닌 권장 설계다. 권장 통합의 결과물은 **자동 합격/탈락**이 아니라 검토자가 확인할 evidence packet이다.

질문·기록	권장 내용
직무 기준	“이 제출물에 SQL 성능 분석의 근거가 있는가?”처럼 직무 관련 관찰로 한정.
저자 기여	팀 성과와 개인 역할을 구분. 저자 확인 상태를 unknown으로 남길 수 있음.
자료 누락	비공개 링크·깨진 URL·OCR 실패는 역량 없음이 아니라 missing evidence.
공정성	보호 특성·대리 변수에 의한 평가를 피하고 일관된 rubric과 사람 재검토를 제공.
데이터 권리	제출·동의 범위, 목적 제한, 최소 수집, 접근통제·보관·삭제·이의 제기 절차.

고위험 결정의 사람 관문

Jev 확률 하나로 채용 거절·순위 확정·신용/금융 의사결정을 자동 실행하지 않는다. 자료 부재와 능력 부재는 다르다.

생태계: 모델·도구·관측을 구분한다

프로젝트	위치와 역할	도입 시 확인
AnyJev [R1]	공개 causal LLM으로 제한된 결정을 읽는 연구·라이브러리. training-free 및 Tacit 경로.	TypeSafe 공식 구현 아님. 모델 weights·라이선스·서빙·보정 별도. BANKING77-20 수치는 저자 실험.
DeepEval / JevEval [S8,R2]	평가 지표·테스트·trajectory 프레임워크.	질문 state·가중치·N/A·strict 의미 고정. Jev 호출 비용 별도.
jev-as-judge [R3]	합성 trace 10건에 Noul 3개+Score 1개를 묶어 평가하는 Opik custom adapter 예제.	demo 확률은 수작업이며 저자도 실제 인증 API/업로드 미실행을 명시. 환불·자동 escalation 기능 없음.
Opik [R4]	trace·데이터셋·실험·평가 관측 플랫폼.	평가 모델이 아님. trace에 비밀·개인정보가 저장되지 않도록 필터.
dzhng/jevgrep [R5]	자연어 기반 코드 탐색 CLI.	8/10은 검색 정확도가 아니라 SWE-bench 과제 해결 수. 소스 외부 전송 주의. 동명 저장소와 구별.
agent-beacon [R6]	에이전트 세션 기록·메모리·검토된 지식 재사용 계층.	Jev 모델은 아니지만 명시적 memory 평가 명령에 Jev 통합이 있다. dry-run→평가→사람 승인. [S12]
evals-with-deepeval-tute [R7]	구조적 검사·답변 비교·Jev judge를 다루는 튜토리얼.	예제 학습용. 제품 수준 접근권한·운영 게이트는 별도.
Unsloth 가이드 [R8]	LoRA 기반 자체 결정 모델 훈련·보정 실험.	상용 Jev 자체를 미세조정하는 경로 아님. 학습·holdout·서빙을 직접 관리.

비교 단위부터 맞추기

TypeSafe Jev와 AnyJev는 같은 제품이 아니다. “모델”, “추론 추출 방식”, “평가 프레임워크”, “관측 시스템”, “검색 CLI”를 하나의 정확도/비용 순위로 섞지 않는다.

저장소 설명은 저자 주장과 독립 검증을 구분한다. upstream 설치·실행·전체 보안 감사·벤치마크 재현은 수행하지 않았다. VideoJev·MJev·Spatial·Jeva의 논문·실제 코드 대조와 분석은 36-43쪽에 추가했다.

공급자 교체는 adapter와 재평가로

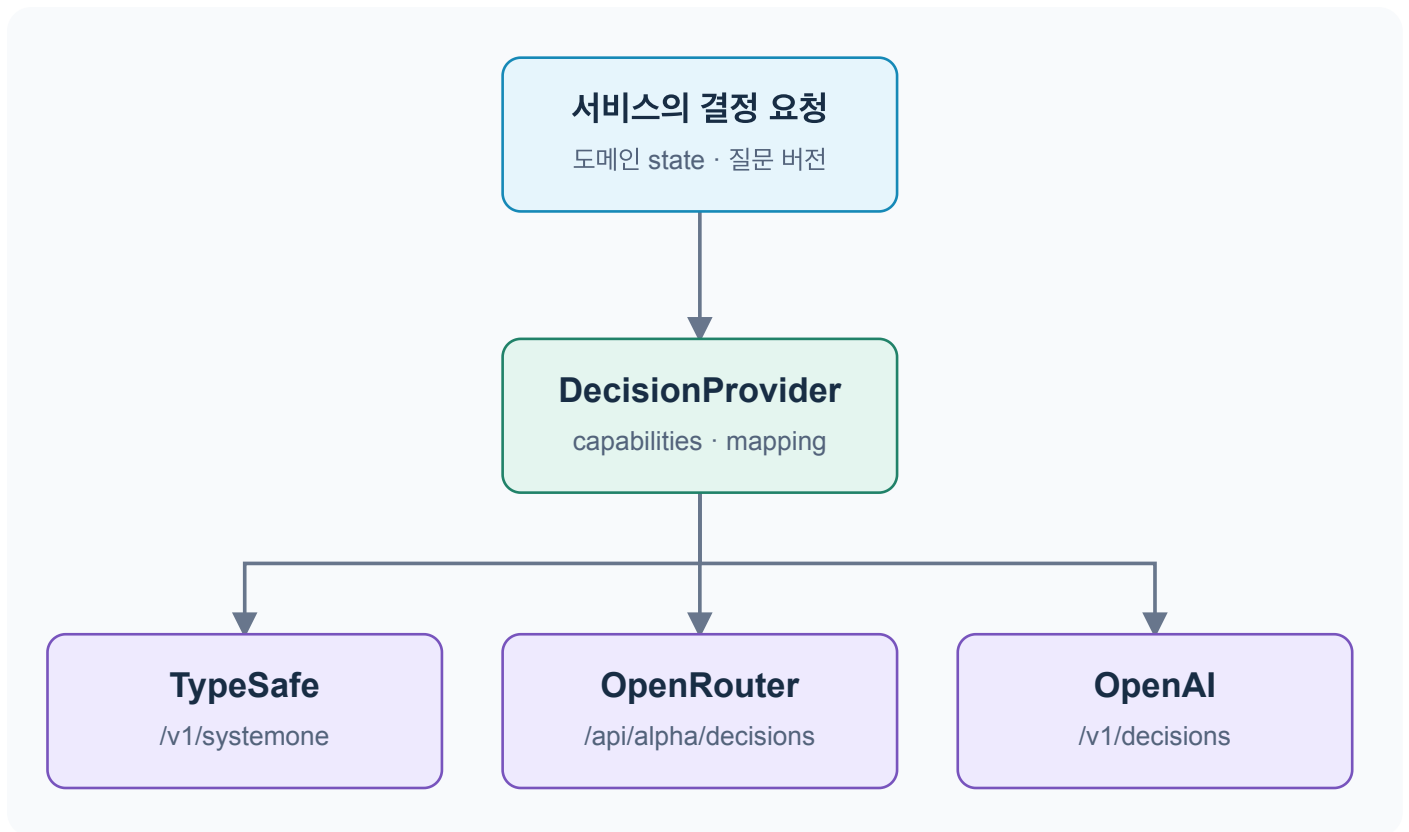


그림 8. 권장 DecisionProvider 경계 · 공통 타입은 유지하되 공급자별 semantics와 capability를 지우지 않는다.

항목	TypeSafe Jev	OpenAI Decisions
입력 / 질문	state + ID별 questions 맵	input + name이 있는 questions 배열
타입	noul / choice / score	predicate / choice / score
응답	ID별 answers 맵	name을 포함한 answers 배열, refusal 처리
모델 예	jev-1.14.0 (직접 API)	gpt-6-luna
입력 단가	입력 \$0.042 / 1M tokens	입력 \$0.10 / 1M tokens
미디어	기본 Text + 별도 이미지 경로	텍스트·이미지 문서화

가격은 조회 시점의 공개 기본 단가이며 계약·지역 처리·장문·중개 비용은 별도다. OpenAI 문서는 Decisions를 public beta로 표시한다. [S3, S11]

정규화할 것 / 유지할 것

공통화: question ID, answer kind, 확률, 오류 상태, 모델 버전, usage. 보존: 원본 응답, refusal/unsupported, confidence 정의, 입력 미디어·한도. 공급자 교체 후 임계값과 한국어 평가를 다시 수행한다.

평가·확률 보정·비용

정확도 하나로 운영을 결정하지 않는다. 실제 업무 표본·한국어·애매한 사례·정보 누락·공격 입력을 포함한 독립 평가셋이 필요하다. 동일 state·동일 라벨·동일 작업 목적에서만 공급자를 비교한다.

측	기록할 지표	판정에 쓰는 방법
품질	정확도, macro-F1, 클래스별 precision/recall, 오탐·미탐	비용이 큰 오류 유형별로 허용 한도 설정
보정	Brier score, ECE, reliability curve	출력 확률과 실제 빈도가 맞는지 확인. 빈 크기·표본 불확실성 명시.
선택적 자동화	자동 처리 coverage, 자동 구간 오류율, 검토량	불확실 사례를 보류해도 전체 오류를 숨기지 않음.
실행 성능	p50/p95 지연, timeout/429/오류율, retry 포함 비용	네트워크·동시성·컨텍스트 길이를 함께 기록
분포 변화	언어·입력 길이·카테고리·모델 버전별 품질	질문/선택지/버전 변경 시 동일 회귀셋 재실행

데이터를 세 구간으로 분리한다

개발셋으로 질문을 다듬고, 검증·보정셋으로 임계값을 선택하며, 고정 테스트셋으로 최종 판정한다. 동일 문서의 중복·파생 사례가 분리 구간을 넘지 않게 하고, 실제 빈도와 어려운 사례의 평가를 각각 보고한다.

설명용 입력 비용 계산 — 성능 측정값 아님

가정: 10,000건 × 요청당 실제 과금 입력 1,000 tokens
총 입력 = 10,000,000 tokens
TypeSafe 기본 단가: 10 × \$0.042 = \$0.42
OpenAI 기본 단가: 10 × \$0.10 = \$1.00

OCR·검색·이미지 토큰·질문 길이·재시도·후속 LLM·인간 검토·관측 저장 비용을 제외한 산술 예다. 실제 비용은 각 공급자의 usage와 청구 데이터를 사용한다. 질문 배치는 state 반복을 줄일 수 있지만 추가 질문은 무료가 아니다.

한국어는 별도 합격 기준

공식 문서는 영어 우세와 CJK 성능 차이를 명시한다. 번역 후 평가가 자동으로 해결책이 되지는 않는다. 한국어 원문과 실제 도메인 표현으로 품질·보정을 확인한다. [S3]

운영: shadow부터 시작한다

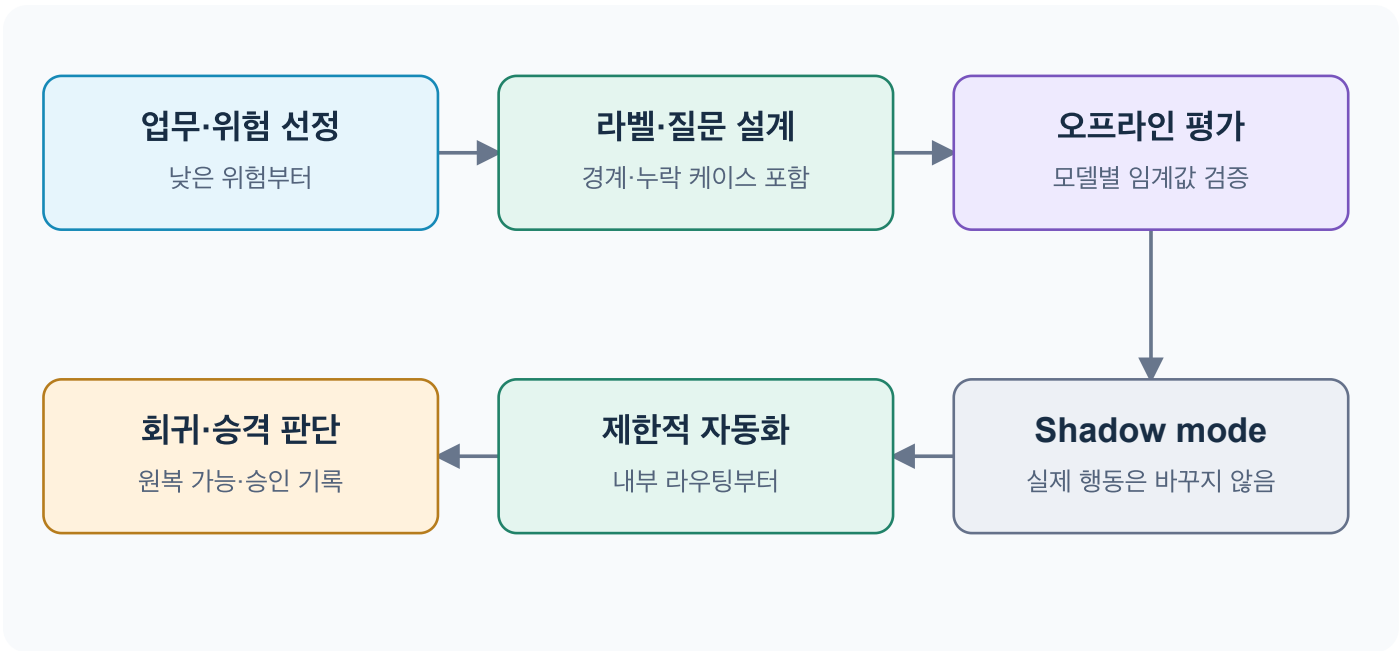


그림 9. 도입과 운영의 단계별 관문 · 화살표는 처리 순서. 도식은 권장 설계이며 설치·배포된 시스템의 실측도가 아니다.

영역	권장 기본값
오류·재시도	400/401 등 설정·인증 오류는 무한 재시도 금지. 429/5xx/timeout은 제한된 backoff+jitter, Retry-After 존중. 실패는 검토/보류.
이중 실행	추론 retry와 후속 액션을 분리. 액션에 idempotency key·중복 방지·감사 ID 적용.
버전 관리	provider/model, questions/rubric, threshold/policy, parser/schema를 함께 버전 관리.
감사 로그	근거 참조/해시·질문 ID/버전·모델 버전·원본 판정·정책 결과·검토자/행동. 원문은 제한된 저장소에 보관.
캐시	tenant/권한 범위·state hash·질문·모델 버전으로 분리. 원본 판정과 정책 결과를 분리해 threshold 변경 시 재해석 가능하게.
롤백	평가된 모델·질문·정책 버전을 묶어 되돌리고 자동 처리를 끌 수 있는 kill switch 확보.

최종 수용 조건

① 권한/개인정보 관문 ② 스키마·실패 경로 테스트 ③ 독립 한국어 평가 ④ 위험별 임계값 ⑤ shadow 오류·검토량 수용 ⑥ 관측·원복 절차.
이 조건의 실측 결과 없이는 “운영 준비 완료”라 부르지 않는다.

TypeSafe: HTTP와 SDK의 계약을 구분한다

공식 API 리디렉션과 OpenAPI를 대조했다. SDK 예제에서 model을 생략할 수 있어도 raw HTTP의 필수 필드는 model·state·questions 다. 질문은 공유 state를 독립적으로 평가하며, 앞 질문의 답이 필요한 후행 질문은 별도 요청으로 만든다.

```
POST https://api.typesafe.ai/v1/systemone
Authorization: Bearer <server-side secret>
Content-Type: application/json

model: exact model ID
state: string | object | array
questions: {question_id: {type, instructions?, criteria?}}
```

항목	공식 계약 / 실무 정책
State	최상위 null·number·boolean은 요청 state 타입이 아니다. 개인정보를 최소화한 string/object/array를 전달한다.
Choice	criteria는 옵션→설명 map. OpenAPI는 1~255개를 허용하지만 분류/로컬 confidence 계산에는 2개 이상을 요구한다.
Score	criteria는 낮은→높은 배열. OpenAPI는 1~10개지만 의미 있는 척도·정규화에는 2~10개를 요구한다.
Instructions	스키마상 선택이며 문자열 외 구조화 값도 가능하다. 스킴/스타터는 의미가 명확한 문자열을 요구하는 더 좁은 정책이다.
Response	model·answers·usage를 검증한다. Score 확률 키는 HTTP에서 문자열, Python SDK에서는 정수 변환 가능.
SDK	정확한 이름과 경계
Python	typesafe-sdk / import typesafe_sdk / system_one / Python 3.10+.
JavaScript	@typesafe-ai/sdk / systemOne / Node 20+. 직접 TS 실행 조건은 별도.
Credential	TYPESAFE_API_KEY. 문서 조회·오프라인 테스트 단계에서는 읽지 않는다.

완전한 API 클라이언트가 아닌 이유

오프라인 starter는 텍스트·문자열 설명·제한된 envelope만 지원한다. 구조화 state, assets, Box, SDK 변환, 공급자 확장 필드를 자동 처리하지 않는다. 이 제한을 Jev 자체의 기능 제한으로 옮기지 않는다.

Confidence: 확률·척도·정답률은 다르다

타입	의미 / 공식
Noul	$p = P(\text{true})$. 별도 confidence 필드는 없다. 필요 시 $\text{abs}(2p-1)$ 를 derived 값으로 별도 저장하며 이것을 긍정 확률과 혼동하지 않는다.
Choice	$C = (p_{\text{max}} - 1/n) / (1 - 1/n)$. 최고 확률만 보므로 나머지 분포의 모양은 구분하지 못한다. $n=1$ 에서는 식이 정의되지 않는다.
Score	$\text{score} = \sum(i \times p_i)$. 최빈값이 아니라 루브릭 위치의 기대값이며 범위는 $0..n-1$ 이다.
Score confidence	$m = \text{argmax}(p_i)$, $D = \sum i - (n-1)/2 / n$, $C = \max(0, 1 - \sum p_i i - m / D)$. 기대점수가 아닌 최빈 레벨과의 거리다.

문서 분포 (0, 0.57, 0.43)의 산식 계산은 score=1.43, confidence=0.355다. 실제 API 응답을 받은 결과가 아니라 로컬 산술 확인이다.

질문 설계에서 바뀌는 행동	이유
ID에만 의미를 넣지 않는다	Choice 질문 ID는 모델에 전달되지 않지만 옵션명·설명은 전달된다. 지시와 기준을 명시한다.
Score 레벨을 독립적으로 설명한다	모델은 각 레벨의 숫자·이웃 설명을 보지 않는다. “전보다 심함”보다 구체적 관찰 기준을 쓴다.
Other 하나만으로 불확실성을 처리하지 않는다	후보 중 해당되는 것이 있는지 companion Noul로 분리하고 적용 가능성을 먼저 검사한다.
필수 위험 조건을 평균으로 지우지 않는다	법적 위험·브랜드 적합성·품질은 독립적인 gate다. 평균 점수가 높아도 필수 항목 실패는 남긴다.

동률·단일 옵션·분포 검증

문서 예제의 argmax 동률 규칙을 서버 전체 보장으로 확대하지 않는다. 분포 합의 허용오차는 명시하고 잘못된 분포를 조용히 정규화하지 않는다. confidence 상승만으로 정확도 향상을 선언하지 않는다.

한도·오류·보관: 운영 전 확인할 계약

TypeSafe 직접 API	확인한 경계
기본 토큰 예산	요청 state+전체 질문 64k; state+개별 질문 32k. 두 조건을 동시에 충족해야 한다.
Long-context opt-in	TypeSafe-Danger-Zone-Long-Context: true이면 두 한도가 262k tokens. 느려지거나 정확도가 낮아질 수 있다.
HTTP body	32 MiB 제한은 토큰 제한과 다르다. UTF-8, 중첩 깊이, 압축 전후 크기를 별도 검사한다.
Alias	2026-10-21 별칭 확인: jev-latest는 1.13.0, jev-preview는 1.14.0. latest를 최신 출시와 동일시하지 않는다.
언어	영어 우세/CJK 품질 차이가 문서화되어 있다. 한국어 실제 라벨 평가를 별도로 통과해야 한다.
오류/운영	기본 대응
400 / 413 / 415 / 422	입력·크기·압축·필드 계약을 수정한다. 같은 payload를 반복 전송하지 않는다.
401 / 429 / 529	인증 오류는 중단, rate limit·overload는 승인된 요청/시간 예산 안에 서만 재시도.
SDK retry	Python 기본은 최초 후 2회 재시도·30초 총 retry budget. JS 시간 단위와 상한은 다르므로 중첩 retry를 피한다.
에러·debug 로그	422 detail과 debug body에는 입력 원문이 포함될 수 있다. header 가림만으로 PII 제거가 되지 않는다.

학습 미사용 ≠ Zero Data Retention

문서는 학습 미사용을 설명하지만 기본 무보관을 뜻하지 않는다. ZDR은 enterprise 별도 협의다. 미국 처리·보관 및 하위 처리자, 이용 목적·보유 기간·삭제를 확인한다. Opik/Confident AI를 켜면 별도의 데이터 저장 경계가 추가된다.

Provider·이미지: 공통화하지 말아야 할 차이

경로	요청/응답과 미디어 차이
TypeSafe direct	/v1/systemone, state + questions map. PNG/JPEG assets를 state-instructions-criteria에서 참조. Box에는 criteria 없음. 공급자 별 전송 형식은 별도 계약.
OpenRouter Decisions	/api/alpha/decisions, model/state/questions. 응답 id/provider/model/usage를 보존. 최신 브라우저 본문은 Jev 1.14·262,144-token context를 표시.
OpenRouter images	state의 image_url + data URL. 원격 URL 자동 fetch 없음. PNG/JPEG·해상도/픽셀/파일/총량 조건과 EXIF 전처리 필요.
OpenAI direct	/v1/decisions, input + questions 배열. predicate/choice/score, name 연결, type:refusal을 질문별로 처리.
OpenAI images	input_image.image_url. 공개 HTTP(S) 허용 여부가 가이드와 API reference 사이에 달라 공통 예제는 data URL로 제한.

2026-10-21 한정 갱신 · S13 Images · S14 Box · S3 Models. 이미지·Box·별칭만 확인했으며 전체 보고서를 재검증한 것은 아니다. 실제 API 호출은 하지 않았다.

TypeSafe 버전	이미지 / Box
jev-1.13.0	이미지 지원 / Box 미지원
jev-1.14.0	이미지·Box 지원

기본 OFF · Dreamer-ready / not production-ready

매 요청 TypeSafe-Danger-Zone-Images: true 필요. Box의 top·bottom·left·right는 0..1 정규화 좌표다. 부재에도 항상 Box가 나오며 not-found는 없다. 같은 요청의 Noul 존재 확률이 낮으면 무시한다. 공식 권장 confidence 임계값은 없다.

과거 OpenAPI의 nullable confidence를 현행 보장으로 옮기지 않는다. 향후 parser는 고정된 계약에서 누락·null을 별도 검증한다. 공급자별 크기·형식·잘림 조건은 도입 직전 재확인한다.

내부 adapter가 보존할 것	이유
REFUSED / ERROR / NO_APPLICABLE	false·0점·정상 N/A로 뭉개면 잘못된 승격이 발생한다.
Reported vs estimated cost	OpenRouter usage.cost와 OpenAI token usage는 다르다. USD 누락을 0원으로 만들지 않는다.
Raw vs derived confidence	OpenAI와 TypeSafe가 같은 confidence 공식을 쓴다는 근거는 확인하지 못했다.
Truncation / modality	특정 중개 모델의 무오류 절단이나 다른 모델의 audio/video 지원을 Jev 기능으로 일반화하지 않는다.

DeepEval·Opik: 점수 1도 실패할 수 있다

JevEval은 애매한 의미를 Jev로 평가하고 점수 조합은 일반 코드에서 계산한다. DeepEval 자체의 pass와 애플리케이션의 배포 허용은 별개다.

동작	검증/구현 규칙
N/A mass	credit=None인 Choice 확률 합이 0.5 이상이면 제외. 미만이면 적용 옵션 확률을 재정규화.
전부 N/A	일반·strict 모두 score=1이다. applicable_count=0이면 별도 NO_APPLICABLE/PENDING으로 보내고 승격하지 않는다.
Strict	Noul≥0.5, Score 최빈 레벨=최상위, Choice 최빈 적용 옵션 credit=1. 원확률 100%가 아니다.
Reason / confidence	reason은 코드가 만든 표시문이며 사고과정이 아니다. metric confidence의 최솟값에는 N/A 항목도 포함된다.
관찰 범위	evaluation_params 밖의 사실은 알 수 없다. trajectory에서 제거한 비용·토큰·prompt를 평가할 수 없다.

꼭 추가할 경계 테스트

N/A 질량 0.499/0.5/0.501, 전부 N/A, 동률 argmax, 누락 답변, Boolean/String Choice 충돌, 질문별 refusal 혼합, 누락 usage와 0원의 구분.

Opik에는 TypeSafe SDK용 track_typesafe 통합이 있다. 직접 HTTP는 별도 span 연결이 필요하다. OpenAI Decisions wrapper는 고정 공개 소스·mock 테스트에서 확인했으며 배포 release 지원은 별도로 확인해야 한다. 단순 provider label 변경은 schema 변환이 아니다.

두 번째 반출 경계

trace에는 state·tool output·base64 이미지·예외 원문이 들어갈 수 있다. DeepEval의 .env 자동 로딩, telemetry, 로그인 후 test case 업로드와 외부 judge 호출을 각각 통제한다. self-host Opik도 외부 judge 호출까지 로컬로 만들지는 않는다.

Beacon·평가 튜토리얼: 기록에서 검토된 지식으로

Beacon 단계	실제 의미
Capture / dry-run	일반 hook과 dry-run은 evaluator를 호출하지 않는다. 기록 수집과 Jev 평가는 별도.
Explicit evaluation	beacon memory evaluations run이 TypeSafe를 호출한다. 기본 latest alias는 고정 버전이 아니다.
Candidate	task_success≥0.50 및 세 확률 평균≥0.60은 후보 생성 기준이다. 교훈의 진실성·채택 승인이 아니다.
Approval	점수에서 교훈이 자동 생성되지 않는다. 원trace를 읽어 작성하고 사람 승인·skill 설치 승인으로 분리한다.
Local vs Cloud	로컬 memory.db/read-only MCP는 프로젝트 범위. Cloud에는 evaluator가 없고 local과 자동 복제되지 않는다. wizard 선택값을 확인한다.

DeepEval 튜토리얼은 구조 검사→문구 coverage→Jev 의미 판정을 순차 실행한다. 동일 응답/공유 judge 요청을 재사용하지만 각각의 필수 질문은 독립적인 AND 관문으로 평가한다.

튜토리얼 도입 함정	바로 적용할 규칙
문서 태그 없음	tutorial-01~09를 요구하지만 공개 demo tags가 비어 있다. 검토한 최종 main SHA와 제한을 명시한다.
Private 내부 API	_system_one_outcomes에 의존한다. DeepEval 업그레이드 전에 contract fixture를 재검증한다.
검사 비용과 전체 비용	구조/fuzzy metric이 무료여도 live agent가 먼저 호출되면 비용이 든다. mock agent여도 Jev judge는 호출될 수 있다.
Fail-fast 순서	query별 순차 실행 전제. xdist/random ordering을 추가하기 전에 계층 상태 전이를 재설계한다.

무과금 확인과 실모델 평가를 구분

이 조사에서는 golden JSON 6파일·72 records의 정합성과 제한된 소스 AST를 확인했다. 이것은 72개의 모델 평가를 실행했다는 뜻이 아니다.

AnyJev·JevGrep·Opik 예제: 채택할 부분만

프로젝트	이번 심층 조희로 확인한 사항
AnyJev 0.3.0	raw/L0와 Tacit 경로. L1/L2는 제거되어 이전 버전 안내와 다르다. L0는 옵션 회전·prior 보정이지만 정답 라벨 기반 calibration이 아니다.
Tacit adaptive	추가 추론 cap은 예산 제한이다. cap에 닿으면 저확신 답을 사람에게 보내는 것이 아니라 기존 one-forward 답을 유지한다.
AnyJev benchmark	BANKING77 20-way 300건 raw→L0 accuracy 0.747→0.803, ECE 0.240→0.184. 저자 보고이며 업무 위험 보장이 아니다.
jev-as-judge	합성 trace 10건·Noul 3+Score 1. demo, live judge, Opik experiment는 비용/외부쓰기 경로가 다르다.
JevGrep	eligible source를 외부 provider로 전송한다. ignore/secret heuristic은 완전한 제거 보장이 아니다. doctor도 synthetic live 요청이다.
JevGrep exit	resource_limit으로 incomplete여도 exit 0 가능. status/issues와 누락 원문을 별도 확인한다.

비용 수치 정정

JevGrep headline 약 30%는 Jev 비용을 제외한 10-task tuned cohort다. 별도 0.4.3 합산 재측정은 25.8% 절감, 0.5.0은 Jev 비용 감소에도 합산 비용이 2-3% 늘었다는 저자 보고다. 8/10은 검색 정확도가 아니라 과제 해결 수다.

재사용 가치: state allowlist, strict response validation, 오류와 품질 실패 구분, rubric version, audit. 별도 구현 필요: 사람 검토 큐, 승인 저장소, 운영 권한, 전송 정책, 실제 품질 보정. jev-as-judge는 명시 라이선스를 확인하지 못했으므로 무조건 코드를 복사 배포하지 않는다.

Unsloth·Colab: 자체 모델 학습의 재현 조건

이 경로는 TypeSafe Jev 가중치를 로컬에서 학습하는 것이 아니라, 별도 causal model에 decision head/LoRA를 적용하는 경로다. 모델 확보·GPU·학습·서빙 비용과 데이터 권리는 직접 관리한다.

자료 위치	서로 다른 설정을 섞지 않는다
문서 상단 benchmark	LoRA r64·1 epoch, 일부 표는 Qwen3.5-0.8B/2B.
UI / 권장값	r16·2 epochs·lr2e-4.
문서 code	batch2·gradient accumulation4·max_steps240·lr2e-5.
Colab code	Qwen3.5-4B, r16/alpha16, batch8·GA4·60 steps·lr2e-4.

Colab은 고정 GitHub revision에서 34셀(코드 16셀)을 읽었다. execution_count는 모두 null이고 저장 output도 없었다. 노트북 문장의 정확도·VRAM·시간은 작성자 설명이지 이번 실행 또는 저장 로그의 측정값이 아니다.

학습 전 관문	필요한 근거
데이터	state/questions/gold 정합성, skipped report, 이용권·개인정보 제거.
분리	train/calibration/test 분리 및 중복/파생 샘플 누출 검사.
환경	모델·dataset revision·dependency lock. notebook SHA만 고정해도 Git main 설치/upgrade는 고정되지 않는다.
검증	holdout calibration 뒤 별도 test, local save/reload consistency, 언어·오류비용별 threshold.
서빙	local alias jev-latest가 자체 모델을 가리킬 수 있다. 호스팅 TypeSafe provenance와 구분. 0.0.0.0 bind·Hub upload는 별도 승인.

이번에 실행하지 않은 것

노트북 실행, 모델 다운로드, GPU 학습, 추론, Hub upload. 일반 Python 코드 셀의 AST 검사는 실행 성공·품질·재현성 검증이 아니다.

공개 사례: 활용 아이디어와 검증 한계

링크	원문 본문 재조회 결과 / 한계
X · RAG	dense+keyword/RRF 후보 → passage relevance → 코드 threshold → 근거 충분성 → 생성. 검색 밖의 문서를 복구하지 못하며 injection 판정은 보안 경계가 아니다.
X · TrueForge	40건의 issue matching, decision cost 150x 절감, 검토 모델이 wrong match 5개를 찾았다는 저자 주장. review-sandbox-retry 포 함 전체 비용인지는 미확인.
X · Data analyst	Qwen의 SQL/설명 생성, Jev의 질문 명확성·SQL 의도 ·groundedness 검토. read-only SQLite·SELECT·row cap·승인은 코 드 계층.
첨부 미디어	이번 재조회는 본문이다. 카드 원본·영상 전체 동작·보안·처리량을 검 증한 것이 아니다.

저자의 주장과 독립 검증을 구분

소셜 게시물의 처리량·비용·정확도는 작성자가 제시한 사례일 뿐이다. 재현 가능한 평가 데이터와 실행 조건 없이는 일반 성능이나 운영 안
전의 근거로 사용하지 않는다.

채용에서 공개 footprint가 적거나 비공개 업무를 수행한 사람의 자료 누락은 능력 부족이 아니다. 모델의 자동 합불 대신 직무 관련 근거·
출처·저자 확인 상태·누락·불확실성을 사람에게 전달하는 검토 packet으로 한정한다.

공개 본문을 참고한 사실과 연결 구현의 코드·실행·보안 검증을 구분한다. 도입 전에는 자신의 평가 데이터, 승인 정책, 배포 환경에서 별도
로 검증한다.

구현: 판단과 실행 사이에 경계를 둔다

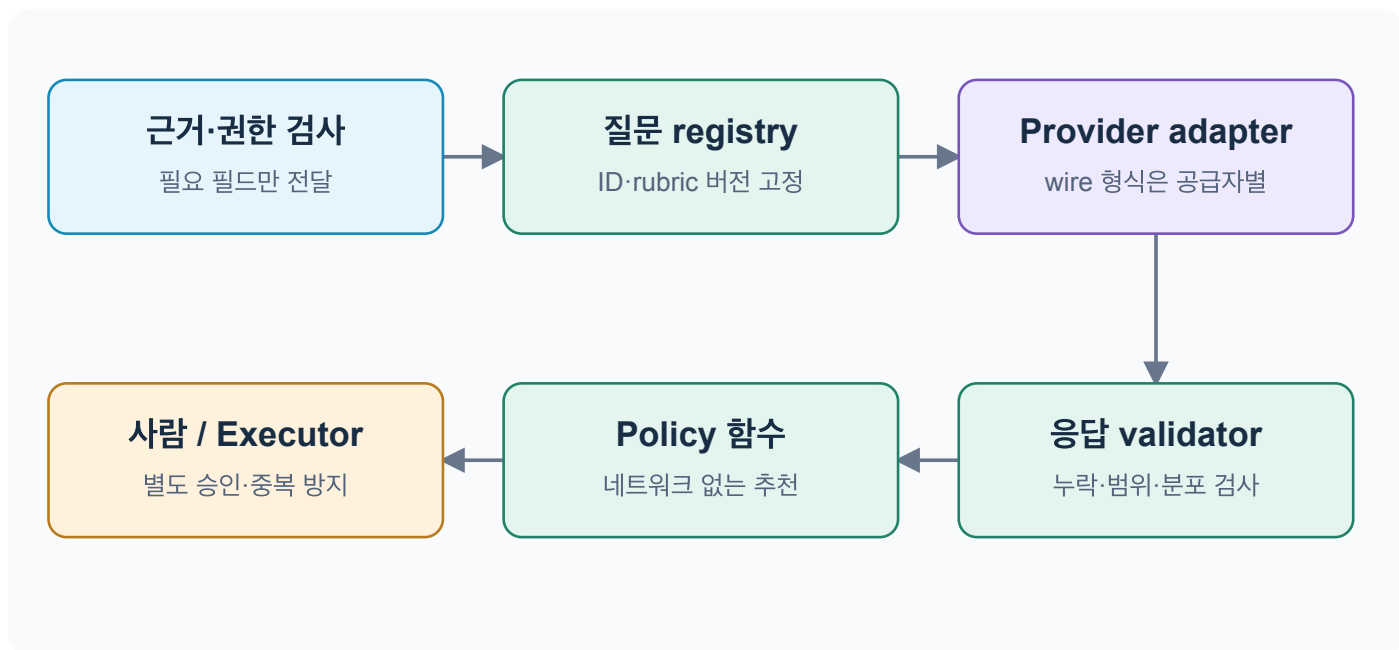


그림 11. 실제 구현에서 유지해야 하는 코드 경계 · 화살표는 처리 순서. 도식은 권장 설계이며 설치·배포된 시스템의 실측도가 아니다.

레이어	개발자가 구현할 계약
Input / evidence	출처·원문 위치·수집 시점·tenant/권한을 확인한다. evidence ID만 만들어 붙여도 사실 검증이 되는 것은 아니다.
Registry / adapter	atomic 질문·rubric을 버전 관리하고, 공급자별 endpoint/header/request/response 변환을 분리한다.
Validator / policy	빠진 ID, unknown choice, 잘못된 분포/scale, NaN을 거부한다. 오류 시 임의 0점 또는 PASS를 만들지 않는다.
Executor / audit	judge 추천과 실제 실행 결과를 따로 남긴다. 권한·idempotency·kill switch는 executor 책임이다.

권장 코드 단위

contracts → questions → adapters → policy → audit → service. 기존 프로젝트의 구조와 테스트를 우선하며, 이 패턴 때문에 새 서비스·DB·worker를 만들지는 않는다.

검증: 정상 입력보다 실패 경로를 먼저

테스트 범주	필수 사례 / 기대 동작
입력 계약	근거 없음·권한 없음·질문 ID 중복·빈 criteria → 호출 전 거부 또는 검토.
응답 계약	누락/추가 ID·type 불일치·unknown option·비유한 값·잘못된 분포 합·Score 범위 오류 → contract failure.
Provider 경계	timeout/429/5xx는 bounded retry, 400/401/403은 수정 전 중단. 404는 body와 endpoint를 확인.
행동 중복	추론 재시도와 액션 재시도 분리. 동일 요청을 받아도 외부 행동은 중복 실행하지 않음.
정보 격리	tenant·권한·model/question version이 다르면 캐시 격리. 민감 state·auth header를 일반 로그에 남기지 않음.
의미 품질	한국어·모순·경계·누락·out-of-scope의 실제 라벨 데이터는 parser unit test와 별도 평가.

Brier = $\text{mean}((p-y)^2)$. ECE는 binning과 기준(positive-event probability 또는 predicted-label confidence)을 명시해야 비교 가능하다. 순서형 Score는 평균점수만 보지 말고 분포와 업무상 오류 비용을 함께 본다.

승격 단계	증거가 없으면 다음 단계로 가지 않는다
Offline	합성 fixture의 parser/policy 테스트 통과. 실제 provider 호환·모델 품질을 의미하지 않는다.
Live evaluation	전송 데이터·권리·보관·provider·요청/지출 cap에 별도 승인.
Shadow	사용자 행동을 바꾸지 않고 검토량·오탐/미탐·지연·비용 실측.
Limited rollout	독립 QA/기존 리뷰 승인, 제한 범위·kill switch·버전 번들 rollback 확보.

평가 누출 금지

동일 사람·대화·문서가 calibration과 holdout에 섞이지 않게 나눈다. holdout 결과로 재튜닝하면 새 검증 데이터가 필요하다. 공개 벤치마크의 오류 상한은 내 업무의 안전 보장이 아니다.

스킬로 사용하기: 설계부터 오프라인 구현까지

공개 **jev-decisions** 스킬을 설치해 자신의 프로젝트에 적용한다. 기존 프로젝트의 명령·설정·승인 절차는 유지한다. 스킬 자체는 API를 호출하는 서비스가 아니라 작업할 때 읽는 실행 절차와 재사용 파일이다.

jev-decisions로 고객 문의 라우팅을 설계해. API 호출 없이.
jev-decisions로 RAG 판정 adapter와 테스트를 구현해.
jev-decisions로 에이전트 trace 평가 질문과 회귀 테스트를 만들어.

구성	어떻게 사용하는가
설계 템플릿	목적·위험·권한·evidence·provider·버전·지출·리뷰 관문을 먼저 채운다.
Question registry	Noul·Choice·Score의 예제를 업무 기준으로 교체한다. 예제 회사명/threshold를 제품에 고정하지 않는다.
오프라인 starter	표준 라이브러리만 사용. 질문 정의 검증·내부 정규화 응답 검증·정책 추천을 로컬에서 테스트한다.
문서 references	TypeSafe 계약, provider 통합, 구현 플레이북, 오프라인 starter 설명을 필요할 때만 읽는다.
평가 기록	미측정 값은 null/미측정으로 유지한다. 합성 결과를 실험 결과로 채우지 않는다.

설치 후 검증 명령

```
python3 -m unittest discover -s scripts -p 'test_decision_kit.py' -v
python3 scripts/decision_kit.py --help
```

위 명령은 skill_dir 안에서 실행한다. 구체적인 starter 입력 형식은 references/offline-starter.md를 따른다.

오해하면 안 되는 경계

starter는 **TypeSafe-style 텍스트 계약의 제한된 부분만** 검증한다. TypeSafe/OpenRouter/OpenAI raw JSON을 그대로 넣는 universal client가 아니며, 실제 API·credential·외부 실행 기능은 없다. 라이브 adapter는 선택한 provider 계약에 맞게 별도로 구현·검증한다.

공개판 범위와 안전한 재사용

이 배포판은 공개 기술 문서를 바탕으로 작성한 독립 커뮤니티 가이드다. TypeSafe 및 언급된 공급자의 공식 제품·인증 플러그인이 아니다.

포함한 것

- 타입이 있는 질문 설계, 확률 해석, 실패 처리, 평가와 단계적 도입 절차.
- 공개 문서·저장소 링크와 근거의 한계.
- 합성 예제로만 실행하는 오프라인 도구·테스트와 재사용 템플릿.
- 이 HTML 원본, PDF 생성 도구 및 직접 만든 SVG 도식.

포함하지 않은 것

- 개인 대화, 내부 프로젝트·고객·업무 데이터, 비공개 저장소 이력.
- 계정 설정, API 키, 토큰, 인증 쿠키, 자동 설치 hooks.
- 타사 모델 가중치, 자동 유료 호출(동봉 client는 --live 명시 때만 호출), 자동 외부 전송이나 실제 실행 권한.

워크플로에 적용하는 순서

- 스킬을 설치하고 낮은 위험의 의사결정 하나를 고른다.
- 도메인·권한·증거·질문·정책을 템플릿으로 정의한다.
- 오프라인 테스트로 잘못된 입력과 보류 경로를 검증한다.
- 별도 승인된 데이터와 예산으로 live adapter를 구현·평가한다.
- 사람 검토를 유지한 shadow 단계부터 점진적으로 도입한다.

합성 예제는 실측 데이터가 아니다

예제 확률·임계값은 동작을 설명하기 위한 값이다. 정답률·안전성·실무 보정·공급자 호환을 보장하지 않는다. TypeSafe 직접 API와 다른 공급자의 응답 형식도 서로 같다고 가정하지 않는다.

링크별 독해 기록 1

아래는 공개 자료별 핵심 내용과 한계다. 공개 출처 목록은 sources.json에 제공한다. 문서 검토는 모델 실행·미디어 검증을 뜻하지 않는다.

S1 · TypeSafe Introduction

<https://docs.typesafe.ai/introduction>

읽은 내용: 질문은 공유 state를 독립 평가. 복합 판단은 atomic 질문과 코드 조합으로 분리.

적용 주의: 선행 답이 필요한 후행 판단은 별도 요청.

S2 · TypeSafe System One API

<https://docs.typesafe.ai/api>

읽은 내용: System One으로 리디렉션. model/state/questions 필수, 답 타입·usage·OpenAPI 대조.

적용 주의: SDK 기본값과 raw HTTP 계약, 구조화 state와 좁은 starter 구분.

S3 · TypeSafe Models

<https://docs.typesafe.ai/models>

읽은 내용: 모델 alias, 기본/opt-in 토큰 한도, 가격, CJK 한계와 개인정보 정책.

적용 주의: latest≠최신 출시. 토큰≠바이트. 학습 미사용≠ZDR.

S4 · TypeSafe Choice

<https://docs.typesafe.ai/primitives/choice>

읽은 내용: 옵션명·설명이 분류 기준. 선택 확률과 분포를 코드가 소비.

적용 주의: 2개 이상을 실무 정책으로 요구하고 해당 없음은 companion Noul로 검사.

S5 · TypeSafe Score

<https://docs.typesafe.ai/primitives/score>

읽은 내용: 루브릭 위치의 기대값. 각 레벨은 독립적으로 읽히는 설명이어야 함.

적용 주의: Score는 0~1 확률이 아니며 필수 위험 조건을 평균으로 숨기지 않음.

링크별 독해 기록 2

아래는 공개 자료별 핵심 내용과 한계다. 공개 출처 목록은 sources.json에 제공한다. 문서 검토는 모델 실행·미디어 검증을 뜻하지 않는다.

S6 · TypeSafe Confidence

<https://docs.typesafe.ai/confidence>

읽은 내용: Choice·Score confidence 공식, Noul 파생 confidence의 의미.

적용 주의: 단일 옵션 분모 0, 분포·동률과 확신/정답률의 차이.

S7 · TypeSafe Quickstart

<https://docs.typesafe.ai/introduction/quickstart>

읽은 내용: Python/JS package·import·method, SDK 기본값과 Playground 경로.

적용 주의: 설치·인증 호출은 미실행. SDK retry 중첩과 debug 원문 유출 방지.

S8 · DeepEval JevEval

<https://deepeval.com/docs/metrics-jev-eval>

읽은 내용: JevEval의 가중치·N/A·strict·trajectory·reason을 문서와 순수 함수로 대조.

적용 주의: 전부 N/A일 때 score=1. applicable_count gate를 별도 구현.

S9 · OpenRouter Jev

<https://openrouter.ai/docs/guides/community/jev>

읽은 내용: 브라우저 본문으로 Jev 1.14·이미지/Box·context를 재확인.

적용 주의: extractor 구분문 차이를 기록. SDK compatibility URL과 Decisions endpoint 구분.

S10 · OpenRouter Multimodal Decisions

<https://openrouter.ai/docs/guides/community/multimodal-decisions>

읽은 내용: provider별 image/audio/video schema와 크기·pixel·truncation 조건.

적용 주의: 다른 모델의 기능을 Jev에 전이하지 않으며 inline data URL 계약 검사.

링크별 독해 기록 3

아래는 공개 자료별 핵심 내용과 한계다. 공개 출처 목록은 sources.json에 제공한다. 문서 검토는 모델 실행·미디어 검증을 뜻하지 않는다.

S11 · OpenAI Decisions guide

<https://developers.openai.com/api/docs/guides/decisions>

읽은 내용: 직접 /v1/decisions, input/questions 배열, predicate/choice/score/refusal.

적용 주의: Boolean/string choice, 질문별 refusal, 이미지 URL 문서 충돌과 usage 차이.

S12 · Beacon Cross-Harness Memory

<https://docs.beacon.sh/concepts/cross-harness-memory.md>

읽은 내용: 명시적 memory 평가, 후보 생성, 사람 승인, 프로젝트 범위와 Cloud 차이.

적용 주의: capture hook은 judge를 부르지 않음. local/Cloud 자동 동기화 아님.

R1 · AnyJev

<https://github.com/nokia-applied-research/AnyJev>

읽은 내용: AnyJev 0.3.0 raw/L0/Tacit, gateway·계약·저장된 benchmark.

적용 주의: L1/L2 제거. L0≠label calibration. 작은 cohort 및 독립 TypeSafe 비제휴.

R2 · DeepEval

<https://github.com/confident-ai/deepeval>

읽은 내용: JevEval 구현·privacy·환경 로딩·evaluation semantics 확인.

적용 주의: 판정 API와 Confident AI 기록은 각각 데이터 반출 경계.

R3 · Jev-as-a-Judge

<https://github.com/patchy631/jev-as-judge>

읽은 내용: 합성 trace10, Noul3+Score1, demo/live/Opik 경로와 strict response validation.

적용 주의: 데모 확률은 수작업. 라이선스 명시 미확인. 운영 guardrail 아님.

링크별 독해 기록 4

아래는 공개 자료별 핵심 내용과 한계다. 공개 출처 목록은 sources.json에 제공한다. 문서 검토는 모델 실행·미디어 검증을 뜻하지 않는다.

R4 · Opik

<https://github.com/comet-ml/opik>

읽은 내용: trace/dataset/experiment, TypeSafe tracking, OpenAI Decisions 관련 고정 소스.

적용 주의: SDK wrapper와 release 지원 구분. 추적 시 state 비밀/PII 별도 제거.

R5 · JevGrep

<https://github.com/dzhng/jevgrep>

읽은 내용: 현재 CLI/provider/secret heuristic/incomplete 처리 및 과거 benchmark 회계.

적용 주의: ~30%는 Jev 제외. tuned10 tasks·8/10 solve. doctor도 live 호출.

R6 · Beacon

<https://github.com/Asymptote-Labs/agent-beacon>

읽은 내용: 세션 수집·메모리와 명시적 Jev evaluator 소스 대조.

적용 주의: Jev 점수≠교훈 생성. 별도 사용자 승인·skill 설치 승인.

R7 · DeepEval tutorial

<https://github.com/bguiz/evals-with-deepeval-tute>

읽은 내용: 9부작 문서·최종 demo main, 6 golden files/72 records 정합성 확인.

적용 주의: 문서 tutorial 태그 부재. private API·순차 fail-fast·live agent 비용.

R8 · Unsloth Decision Model

<https://unsloth.ai/docs/basics/train-your-own-decision-model-with-unsloth>

읽은 내용: 별도 decision head/LoRA 학습·보정·서빙 경로.

적용 주의: 문서/코드/Colab 설정 불일치, 모델·데이터·dependency revision 고정.

링크별 독해 기록 5

아래는 공개 자료별 핵심 내용과 한계다. 공개 출처 목록은 sources.json에 제공한다. 문서 검토는 모델 실행·미디어 검증을 뜻하지 않는다.

P1 · Jev / RAG

https://x.com/akshay_pachaar/status/2105281570898849903

읽은 내용: 하이브리드 검색→문단 관련성→충분성→생성의 원문 본문.

적용 주의: 검색 누락 복구 불가. injection score는 보안 경계 아님.

P2 · Jev / agent harness

https://x.com/akshay_pachaar/status/2109909532352917658

읽은 내용: TrueForge/Daytona/Jev issue matching의 원문 본문.

적용 주의: 40건·150x는 저자 decision-cost 주장이며 전체 비용/보안 검증 아님.

P3 · Jev 데이터 분석 에이전트

https://x.com/Sumanth_077/status/2107800513681182737

읽은 내용: Qwen 생성과 Jev 의미 검토, SQL 실행 코드 경계의 원문 본문.

적용 주의: SQLite가 로컬이어도 schema/rows를 prompt로 보내면 외부 반출.

사례 인용의 경계

사례 설명은 공급자별로 구분한다. 다른 공급자의 활용 사례를 Jev 통합 증거로 인용하지 않으며, 속도·공정성·채용 자동화의 일반 근거로 확대하지 않는다.

P5 · Unsloth Colab

https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Qwen3_5_%284B%29-Decision.ipynb

읽은 내용: raw GitHub 고정 SHA의 Colab 34셀·코드16셀을 읽음.

적용 주의: 미실행·저장 output 없음. 일반 Python15셀 AST만 검사.

검증 범위와 다음 적용 단계

공개 출처를 바탕으로 정리한 기술 가이드다. 출처와 내용은 작성 시점 기준이며, 공급자의 현재 계약·모델·가격·한도는 도입 직전에 재확인해야 한다. 실제 API 호출·성능 벤치마크·영상 전체 재현 결과를 제공하지 않는다.

확인한 것	확인하지 않은 것
문서·bounded source·OpenAPI 독해	유료 Jev/OpenAI 추론, 실제 계정 API 호출
소셜 본문·Colab 코드 셀	소셜 미디어 전체 재생, notebook/GPU 학습
오프라인 starter parser/policy 테스트	실제 데이터 정확도·확률 보정·운영 threshold
리포트 HTML/PDF 렌더링과 레이아웃 QA	서비스 배포·실제 티켓 라우팅·결제·채용 판정

재사용 산출물

공개 저장소에는 이 가이드의 HTML 원본과 PDF 생성기, 공개 출처 목록, 범용 skill/plugin, 질문·설계·평가 템플릿 및 오프라인 스크립트·테스트가 포함된다. 개인 대화·비공개 프로젝트·내부 연구 원장은 포함하지 않는다. 설치·오프라인 검증·실제 API 도입은 서로 다른 단계다. 연구 확장 장은 세 논문의 요약·분석·구현 차이와 추가 검증 경계를 다룬다.

첫 적용은 정답 공간이 작고 근거가 명확하며 되돌릴 수 있는 내부 라우팅이다. 설계 템플릿을 채우고 질문·adapter·실패 테스트를 만든 뒤, 승인된 실제 라벨 데이터로 provider별 threshold를 평가한다. Shadow 단계에서는 실제 행동을 바꾸지 않는다.

한 문장으로

좁은 의미 판정은 Jev에 맡기되, 불확실성·권한·근거·최종 행동은 검증 가능한 코드와 사람의 책임으로 남긴다.

영상·공간·null-aware 판정으로 확장하기

이 장은 **VideoJev**, **MJev-Spatial**, **Jeva**의 논문 본문·부록과 공개 구현을 대조한 요약·분석이다. 세 프로젝트는 TypeSafe 서비스의 추가 모델 ID가 아니다. 공통점은 자유로운 답변 생성보다 **주어진 후보 공간에서 판정하는 구조**에 있고, 입력·학습·추론·출력 계약은 서로 다르다.

프로젝트	핵심 질문	설계에서 가져올 점
VideoJev 37-38쪽	영상에서 사건이 언제 일어나는가? 제시한 답 중 무엇이 맞는가?	영상 특징과 질문을 분리하고 시간 후보·선택지를 점수화한다.
MJev-Spatial 39-40쪽	대상이 어디에 있는가? 관계·수치·좌표는 무엇인가?	native attention으로 후보를 비교하고 단계적으로 정밀도를 좁힌다.
Jeva 41-42쪽	어떤 후보가 적절한가? 후보 집합 안에 답이 있기는 한가?	후보 간 경쟁과 후보 집합의 coverage를 분리한다.

같은 축으로 읽되, 같은 성능표로 합치지 않는다

- **표현**: 원본 영상·이미지·텍스트 중 무엇을 받는가?
- **후보**: 시간 셀, 텍스트 옵션, 공간 codebook 중 무엇을 비교하는가?
- **결과**: 구간·단일 좌표·조건부 확률·joint mass를 어떻게 해석하는가?
- **운영**: 실제 공개 진입점·캐시 범위·단위·실행 비용은 어디까지인가?

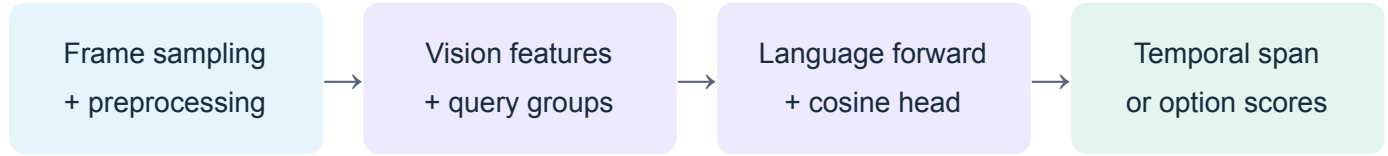
읽기·검증 수준

전체 텍스트·부록 독해와 고정 소스의 call graph를 기록했다. 그림별 시각 확인 범위는 상세 reference에 구분했다. 가중치 다운로드, 모델 추론, 학습과 논문 벤치마크 재현은 하지 않았다. 아래 활용 예시는 권장 설계이지 모델 실행 결과가 아니다.

43쪽의 원문·코드 링크와 오프라인 도구로 이어진다. 세 모델을 조합할 수도 있지만 이 패키지는 통합된 멀티모달 실행 backend를 제공하지 않는다.

VideoJev: 생성 대신 시간·선택지 판정

한 줄 요약: 영상과 질문을 처리한 뒤, 시간 후보 또는 제시된 답변 후보를 공유 cosine head로 점수화한다. 영상 이해 연구지만 임의의 대화 답변 생성·오디오 이해·도구 조작까지 자동 제공하는 것은 아니다.



고정 소스에서 추적한 개념 흐름. 모델을 실행해 관찰한 결과가 아니다. [R9]

논문 아이디어

두 프레임으로 이루어진 temporal group을 시간 셀로 보고, 질문의 사건과 관련된 셀을 판정한다. 추론은 threshold-adjusted logits에 maximum-sum decoding을 적용해 **비어 있지 않은 연속 구간**을 고른다. 선택형 QA는 사용자가 준 옵션 안에서 답을 고른다.

공개 구현에서 확인한 계약

- `predict_video`는 영상 준비·vision encode 후 `predict_features`로 이어진다.
- 질문은 `id/task/query`, 선택형이면 `options`를 사용한다. 우리 검증기는 질문 목록만 검사하며 영상 파일을 받거나 열지 않는다.
- Joint 묶음의 뒤 질문은 앞 질문 텍스트를 볼 수 있다. 질문 독립성이 필요한 평가에서는 이 조건을 통제한다.

구간 출력 ≠ 사건 존재

threshold-adjusted logits가 전부 음수여도 raw decoder는 가장 강한 단일 셀을 고르며 후처리로 구간이 조정될 수 있다. 따라서 사건이 없는 영상에도 구간이 나올 수 있다. 구간만으로 존재를 입증하거나 여러 분리 사건의 탐지 결과로 확장하지 않는다.

근거: 논문 §2·관련 부록, 공개 HF processing/modeling/temporal 경로. 상세 파일·함수·행 번호는 [VideoJev reference](#).

VideoJev: 캐시를 구분해야 도입을 판단한다

항목	논문·설명과 실제 코드의 관계
영상 특징 재사용	공개 HF API에서 vision features를 재사용한다. 매 질문 묶음의 언어 forward까지 생략하는 뜻은 아니다.
Prefix cache	공개 HF forward는 use_cache=False. 별도 연구 코드에는 실제 prefix-cache 경로가 있다. 두 경로를 같은 기능으로 설명하지 않는다.
평가 스크립트 명칭	evaluate_cached_contexts.py는 Joint 경로를 호출하지만 report label에는 private cache forks가 언급된다. 파일명·보고 문자 열보다 call graph를 확인한다.
학습 재현	공개 custom 4B 설정의 192K/ZeRO-3와 논문 경로의 144K/ZeRO-2가 다르다. 설정 차이는 곧 결과 오류라는 뜻은 아니지만 동일 재현의 증거도 아니다.

어디에 유용한가 — 권장 활용 설계

긴 영상에서 사람이 검토할 **사건 후보 구간을 좁히거나**, 동일 영상에 대한 정해진 질문 묶음을 평가하는 보조 도구가 후보가 된다. 예를 들어 “물체가 프레임에 들어오는 구간”을 찾고, 실제 존재 여부와 최종 구간을 사람이 확인하는 방식이다.

실행 전 평가할 것

- 사건 없음·짧은 사건·작은 물체·반복 사건·모호한 선택지를 포함한다.
- Temporal IoU와 선택형 정확도뿐 아니라 false positive·검토율을 기록한다.
- 프레임 sampling·decode·vision encode·언어 forward·후처리의 지연을 구분한다.
- 옵션 순서·질문 순서·Joint와 단독 호출·실제 cache 경로를 따로 비교한다.

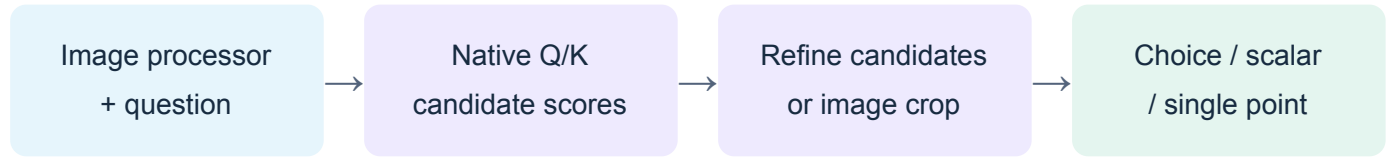
우리 workflow에 반영한 것

video-grounding·video-choice 후보 계획과 질문 계약 검사. 아직 GPU 추론 adapter나 사건 부재 검증기는 아니다. 논문의 speedup을 다운로드·영상 decode까지 포함한 로컬 end-to-end 성능으로 옮겨 쓰지 않는다.

읽은 범위: 본문 §§1-6, 표·참고문헌, Appendix A/B. 시각 확인 범위와 미재현 지표는 상세 reference에 별도 표시. [R9]

MJev-Spatial: 공간 판단을 후보 선택으로

한 줄 요약: 이미지·질문에서 공간 관계, 수치 또는 위치를 판단한다. 기존 attention의 Q/K 표현으로 후보를 점수화하고, 수치·좌표는 coarse-to-fine 단계로 좁혀 간다.



단계마다 후보를 좁히는 개념도. 한 단계의 forward와 전체 요청의 forward 수는 다르다. [R10]

작업	입력·출력의 해석
Choice	공간 관계 등 주어진 옵션을 비교한다. 안정적인 option ID와 텍스트를 구분한다.
Scalar	$m \cdot \text{count} \cdot \text{deg}$ 를 구분한다. 단위가 다른 MAE를 하나의 수치로 합치지 않는다.
Point	공개 기본 경로는 단일 <code>normalized_xy</code> point. 원본 이미지의 정규화 좌표이며 화면·실세계 좌표와 자동 호환되지 않는다.

코드에서 확인한 흐름

CLI의 입력 정규화 → image processor → native Q/K scoring → scalar codebook 또는 point refinement → decode. 학습의 teacher-forced 단계와 추론에서 예측한 단계는 구분해야 한다. 앞 단계 오류가 후속 crop·후보 선택으로 전달될 수 있다.

정밀도 ≠ 정확도, 좌표 ≠ 권한

세분화된 codebook은 출력 해상도를 높일 뿐 정확도를 보증하지 않는다. point를 반환해도 대상 존재·좌표 변환·DOM 매칭·클릭 승인·로봇 안전을 별도로 검증해야 한다.

본문 §§1-5·Appendix A-H와 실제 입력·Q/K·decode·학습 경로를 대조했다. [상세 reference](#).

MJev-Spatial: 설계 채택과 실행 승격은 별개

확인한 문제	정확한 범위·해석
Dependency 충돌	선언된 inference/train extras의 transformers==5.17.0는 Hub >=1.5,<2를 요구하지만 저장소는 Hub <1을 요구한다. 모든 설치 방법이 실패한다고 일반화하지 않는다.
Checkpoint 저장	documented train 경로가 output directory를 만들고 save 함수가 같은 경로를 다시 exist_ok=False로 생성한다.
공개 API export	top-level JevSpatial export가 존재하지 않는 inference 모듈을 참조한다. 다른 직접 진입점과는 구분한다.
Checkpoint 대응	공개 metadata와 논문 최종 학습 설정의 대응이 불명확하다. metadata만으로 가중치 오염이나 논문 결과의 부정을 단정할 수 없다.
평가 범위	공개 기본 evaluator의 accuracy·MAE·point distance는 논문의 mask·multi-point·macro benchmark 전체 harness가 아니다.

도입 순서 — 권장 설계

1. 고정 revision에서 의존성·진입점·저장 lifecycle을 별도 scope로 수정·검증한다.
2. 체크포인트의 학습·평가 데이터 계보와 코드·논문 대응을 확인한다.
3. 라이선스·backbone 조건·환경·메모리를 기록하고 작은 실제 추론 smoke를 실행한다.
4. 독립 holdout으로 단위별 수치 오차, point 위치 오차, 대상 부재·가림·복잡한 장면을 평가한다.

현재 채택 수준

공간 관계·수치·단일 point의

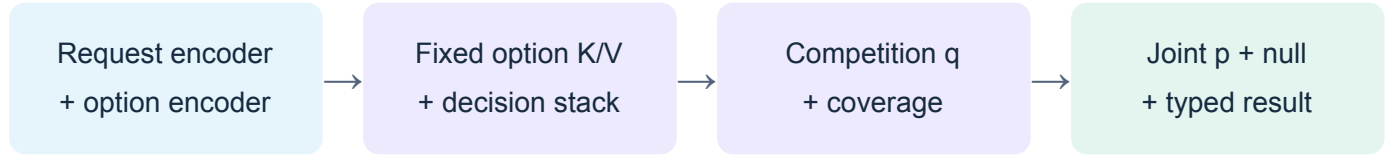
설계 reference와 오프라인 입력 계약

만 제공한다. BLOCKED_SOURCE_ISSUES를 유지하며 실제 backend는 켜지 않는다. 도구가 유효한 요청이라고 판정해도 이 차단 상태는 해제되지 않는다.

활용 후보는 이미지의 위치·관계를 사람이 재검토하는 보조 분석이다. 실시간 조작·정밀 측정·자동 클릭 등은 별도 교정과 권한 검증 없이 연결하지 않는다. [R10]

Jeva: 후보 경쟁과 후보 부재를 분리한다

한 줄 요약: 요청과 옵션을 따로 인코딩하고, 고정된 option memory를 읽어 요청 상태를 갱신한다. “후보 중 무엇이 좋은가”와 “이 후보 집합에 답이 있는가”를 서로 다른 양으로 표현한다.



옵션 재사용이 가능한 구조와 결과 해석의 분리. 캐시 효율·품질을 로컬에서 재현한 것은 아니다. [R11]

양	의미
조건부 확률 q	제시된 후보 안에서의 경쟁 분포. 공개 <code>candidate_probabilities</code> · <code>conditional_probabilities</code> 는 이 값을 가리킨다.
Joint mass p	$p_i = (1 - p_{\text{null}}) \times q_i$. 후보 coverage를 반영한 질량이다. 조건부 q 와 섞지 않는다.
Null	적절한 답이 후보 집합에 없다는 결과. 일반적인 불확실성·정답률·검토 승인과 같지 않다.
Typed value	Noul은 positive option의 조건부 질량, Score는 명시한 values와 조건부 q 로 계산한다. 보편적인 고정 점수 척도가 아니다.

실제 API 경계

우리 검증기는 raw predict의 `state/question/decision_type/options/allow_null` 형태를 제한적으로 검사한다. Score는 values, Noul은 `positive_index`를 명시한다. TypeSafe 요청 JSON이나 평가 label 파일을 그대로 받는 API가 아니다.

닫힌 집합과 열린 집합을 명시한다

typed Noul/Score는 closed-set이지만 raw 경로의 허용 범위는 다르다. 우리 adapter 정책은 후보 두 개 이상, null은 Choice에만 허용한다. 이는 독립적인 안전한 subset이지 upstream 전체 계약을 바꾼 것이 아니다.

Jeva: 아키텍처의 장점과 재현의 빈틈

왜 이 구조가 의미 있는가

옵션 표현을 요청과 분리하면 반복되는 후보 집합을 재사용할 수 있고, 후보 순서 변화에 대한 평가도 명시적으로 설계할 수 있다. coverage를 따로 두면 후보가 모두 부적절한 상황을 단순한 “가장 덜 나쁜 답”과 구분할 여지가 생긴다. 다만 이는 **구조적 설계 의도**이며 운영 품질·보장·캐시 이득의 실측 보장은 아니다.

대조 항목	발견한 차이와 적용 규칙
후보 수	raw 및 typed Choice/Score의 공통 inference validator는 singleton을 허용한다. 논문·evaluator의 최소 두 후보 조건과 구분하고 우리 검사에서는 두 개 이상을 요구한다.
Permutation 평가	문서의 any-flip/all-order/TV와 공개 summary의 changed-variant agreement·maximum drift는 같은 지표가 아니다. 필요한 지표를 별도 계산한다.
학습	공개 trainer는 논문의 auxiliary loss·단계별 curriculum·별도 optimizer parameter groups를 모두 재현하지 않는다.
Cache 표본	본문·그림의 50과 부록의 46이 다르다. 원인이나 정답 denominator를 추정하지 않고 불일치를 기록한다.
배포 라이선스	코드·가중치·backbone 조건을 각각 확인한다. 기본 download/export 경로만으로 모든 LICENSE·NOTICE가 보존된다고 가정하지 않는다.

권장 평가 시나리오

- 정답 후보 있음·없음, 비슷한 오답 추가, 후보 순서 변경을 분리한다.
- 안정적 option ID로 정렬한 뒤 확률 변화를 비교한다. 텍스트 중복을 임의 제거하지 않는다.
- 같은 모델·tokenizer·옵션·설정에서 cache/no-cache 품질과 실제 지연을 비교한다.
- null false accept/reject와 review 정책을 평가한다. null을 곧바로 승인·거절 명령으로 쓰지 않는다.

우리 workflow에 반영한 것

text-choice·text-noul·text-score 계획과 입력 검증. 후보를 추천·분류하는 보조 설계로 사용하며 모델 호출·업무 라우팅·자동 권한 부여는 수행하지 않는다.

전체 본문·Appendix A.1-C.7과 typed/raw API·평가·학습 경로를 대조. [상세 reference](#). [R11]

사용 순서, 출처, 남겨 둔 검증

스킬에서 적용하기

공개 플러그인은 `/jev:jev-decisions`에 작업을 자연어로 요청한다. 이 reference가 연결된 로컬 `/jev`에서도 같은 설계·검증 흐름을 사용할 수 있다. 명령 이름만으로 모델이 설치되거나 실행되는 것은 아니다.

```
/jev:jev-decisions VideoJev로 사건 구간 평가를 설계하고 합성 질문을 검증해. 추론은 하지 마.
/jev:jev-decisions MJev-Spatial의 단일 point 계약과 실행 차단 사항을 확인해.
/jev:jev-decisions Jeva의 null·조건부 확률을 분리한 평가를 설계해.
```

저장소 루트에서 stdlib 도구와 합성 데모를 실행한다.

```
python3 plugins/jev/skills/jev-decisions/scripts/research_kit.py plan --task video-
grounding
python3 examples/research_demo.py
```

통과 상태를 정확히 읽는다

`VALID_CONTRACT_ONLY`는 입력 계약 통과다. 미디어 경로를 열거나 출처·권한을 인증하지 않으며 모델·GPU·정확도를 검증하지 않는다. 런타임은 모두 `UNVERIFIED`, `spatial`은 추가 차단 상태를 유지한다.

추가 원문·코드 출처

R9 · VideoJev — 논문 2610.18615 · GitHub MCG-NJU/VideoJev

고정 코드: 7f8b38aa7df67721303cdc61f06d8435f3c7db89

R10 · MJev-Spatial — 논문 2610.22276 · GitHub Fr0zenCrane/MJev-Spatial

고정 코드: df37e9e5a3455c642f7b43cf96abc4f950b4fa6d

R11 · Jeva — 논문 2610.18773 · GitHub LiQiiii/Jeva

고정 코드: e2104e0720100f979f2cdd456a1cbb96da0abd

논문은 각각 v1 기준. 개별 논문·소스 항목은 `sources.json`, 체크포인트·추가 public artifact revision은 `research-sources.json`, 파일·행별 근거는 모델별 상세 reference에 보존한다. LinkedIn 추적 링크·내부 데이터·자격증명은 포함하지 않는다.

다음 단계: source/release 문제 해결 → 라이선스·데이터·예산 승인 → 실제 추론 smoke → 독립 holdout 평가 → shadow 운영. 이 가이드는 그 단계를 건너뛴 근거가 아니다.