

LLM 에이전트 기반 데이터 품질 검증을 위한 하네스 엔지니어링 방법론: 관찰-검증-종합 기반 에이전트 설계

최 권 택*

A Harness Engineering Methodology for LLM Agent-based Data Quality Validation: Observation-Validation-Synthesis-based Agent Design

Kwon-Taeg Choi*

본 연구는 2024학년도 강남대학교 교내연구비 지원에 의해 수행되었음

요 약

데이터 품질 검증은 사전 정의된 규칙에 의존할 경우 예상하지 못한 결함을 탐지하기 어렵다. 대규모 언어 모델 에이전트는 의미를 해석해 규칙에 의존하지 않고 결함을 탐지할 수 있으나, 자율성이 높아질수록 재현성과 검증 가능성은 낮아진다. 본 연구는 이 상충을 완화하기 위해 검증을 관찰·검증·종합 세 단계로 나누고 단계별 도구를 제한한 하네스를 제안한다. 도구 제한이 없는 조건을 대조군으로 4개 벤치마크에서 실행한 결과, 도구 통제는 성능을 희생하지 않았고 일부(Beers, Flights)에서는 오히려 향상되었으며(F1: Hospital 0.994/0.998, Beers 0.946/0.995, Flights 0.622/0.730, Rayyan 0.184/0.171), 재현성도 전 데이터셋에서 개선되었다(표준편차 감소, Flights 0.250→0.007). 다만 Rayyan은 두 조건 모두 F1이 0.2 이하로 낮아 도구 통제만으로는 해결되지 않는 난제로 남았다. 실험을 바탕으로 데이터셋·오류 유형별 하네스 구조의 영향과 한계를 분석하였다.

Abstract

Data quality validation relies heavily on predefined rules, limiting detection of unforeseen defects. LLM agents can interpret semantics to detect defects beyond predefined rules, but increasing autonomy reduces reproducibility and verifiability. To mitigate this trade-off, we propose a controlled harness dividing validation into three stages (observation, validation, synthesis) with stage-specific, constrained tools. Across four benchmarks, tool control did not sacrifice performance and even improved it on some (Beers, Flights) (F1: Hospital 0.994/0.998, Beers 0.946/0.995, Flights 0.622/0.730, Rayyan 0.184/0.171), with reproducibility improved across all (Flights: 0.250 to 0.007 in standard deviation). Rayyan, however, remained difficult under both conditions, with F1 at or below 0.2. These experiments reveal the harness structure's effects and limits across datasets and error types.

Keywords

data quality validation, large language model, harness engineering, agent autonomy, tool-based validation

* 강남대학교 ICT융합공학부 교수
- ORCID: <http://orcid.org/0000-0001-5331-321X>

• Received: Aug. 18, 2026, Revised: Sep. 03, 2026, Accepted: Sep. 06, 2026
• Corresponding Author: Kwon-Taeg Choi
Dept. of ICT Convergence Engineering, Kangnam University, Yongin, Korea
Tel.: +82-31-280-3660, Email: kwontaeg.choi@kangnam.ac.kr

I. 서론

데이터에서 도출되는 모든 분석과 의사결정은 그 품질에 따라 신뢰성이 좌우되지만, 품질 검증의 자동화는 오랫동안 검사 대상을 사용자가 사전에 정의해야 한다는 제약 아래 놓여 있었다. Deequ[1]로 대표되는 선언적 제약 프레임워크는 검증 규칙의 작성과 실행 절차를 표준화하였고, Raha[2]는 다수 탐지기의 출력을 특징으로 결합하여 소량의 레이블만으로 규칙 없이 오류를 탐지하는 방향을 제시하였다. 그러나 전자는 규칙 정의 비용을, 후자는 학습 데이터 확보 비용을 각각 사용자에게 전가한다는 점에서 공통의 한계를 지닌다.

대규모 언어 모델(LLM, Large Language Model)의 발전은 이 제약을 완화할 가능성을 열었다. LLM은 사전 계산된 통계 프로파일로부터 데이터의 의미를 해석하여 결함을 판정할 수 있으며[3], 나아가 추론과 도구 호출을 교대로 수행하는 에이전트 구조[4]를 통하여 스스로 데이터를 관찰하고 가설을 검증하는 능동적 탐색까지 수행할 수 있다. 그러나 자율성의 확대가 곧 유용성의 확대를 의미하지는 않는다. 데이터 에이전트의 자율성을 위계화한 최근 연구[5]는 자율성이 높아질수록 결과의 책임성과 검증 가능성은 오히려 낮아질 수 있음을 지적한다.

이는 데이터 품질 검증에서 구체적인 설계 문제로 이어진다. 도구 제한 없이 자유롭게 코드를 작성하게 하면 판단의 유연성과 탐지 범위를 높일 수 있지만 실행 과정의 재구성 과 재현이 어려워질 수 있다. 반대로 호출 가능한 도구를 제한하면 검증 과정의 통제와 재현은 용이해지는 대신 도구가 표현하지 못하는 결함은 탐지 범위에서 제외될 수 있다. 따라서 통제성과 탐지 범위가 상충한다는 것이 통념이나, 이러한 상충이 실제 오류 탐지 성능에 미치는 영향을 비교한 연구는 아직 제한적이다.

본 연구에서는 이러한 문제를 하네스 엔지니어링(Harness engineering)의 관점에서 다룬다. LLM 에이전트의 데이터 품질 검증 과정에서 도구 사용에 제약이 없는 조건과 단계별로 도구 사용을 통제하는 조건을 구성하고, 동일한 모델과 데이터 환경에서 오류 탐지 결과와 재현성을 비교한다. 이를 통해 도구 사용에 대한 통제가 성능과 재현성에 미치는 영

향을, 오류 유형과 모델 규모에 따라 분석한다.

본 연구의 기여는 세 가지이다. 첫째, LLM 에이전트의 데이터 품질 검증에서 자율성과 도구 통제 수준을 비교할 수 있는 실험적 프레임워크를 제시한다. 둘째, 자유 코드 실행 방식과 단계별 도구 제한 방식을 동일 조건에서 비교하여 도구 통제가 오류 탐지 성능과 재현성에 미치는 영향을 정량적으로 평가한다. 셋째, 모델 규모와 오류 유형에 따른 성능 차이를 분석한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 검토하여 연구 공백을 도출하고, 3장에서는 제안하는 하네스의 구축 방법과 도구 체계를 기술한다. 4장에서는 실험 결과를 제시·논의하며, 5장에서는 결론과 향후 연구를 기술한다.

II. 관련 연구

데이터 품질 검증 자동화 연구는 결함 판정의 주체와 그 자율성의 정도를 기준으로 구분할 수 있다. 한쪽에는 전문가가 규칙을 정의하는 전통적 접근이, 다른 한쪽에는 LLM이 데이터를 직접 해석·판단하는 접근이 위치한다. 본 장에서는 전통적 방법과 LLM 기반 방법을 검토하고, 각 접근의 성과와 한계를 비교하여 기존 연구의 공백을 제시한다.

전통적 방법은 사전 정의된 규칙을 적용하거나, 통계·확률 모형을 통해 규칙 정의를 학습으로 대체한다. Deequ[1]와 GX(Great Expectations)는 검증 규칙을 선언적 제약으로 기술하고 이를 대규모 데이터에 결정론적으로 적용하여 재현성을 확보하였다. 도구 비교 연구[6]는 오픈소스 도구가 유연성에 비해 구현 부담이 크며, LLM 통합 역시 규칙 생성에 국한되어 검사 항목의 결정은 여전히 수동으로 이루어짐을 지적하였다. Raha[2]는 여러 오류 탐지기의 출력을 결합해 소량의 레이블로 분류기를 학습하고, HoloClean[7]은 무결성 제약과 통계적 신호를 확률적 추론으로 통합하였다. 그러나 이러한 방법은 학습 데이터나 반복 구조가 부족한 희귀·의미적 오류에 대한 일반화에 한계가 있다.

LLM 기반 방법은 LLM에 부여되는 해석 권한에 따라 다시 나뉜다. 자연어 요구사항을 검증 규칙으로 변환하는 접근에서 Quality by Prompt[8]는 자연

어 요구사항을 GX 표현식으로 변환하고, DQGen[9]은 메타데이터를 품질 차원별 규칙으로 매핑하여 검증 스크립트 생성을 자동화한다. 이 경우 LLM은 판단보다 변환을 수행하므로, 사람이 인지하지 못한 결함을 데이터 자체로부터 발견하기 어렵다. 반면 사전 계산된 프로파일을 해석하는 접근에서 Cocoon[3]은 데이터 프로파일을 요약하여 의미를 해석하고 정제 질의를 생성하며, LLMClean[10]은 LLM이 추출한 온톨로지적 함수 종속성을 복구 과정에 활용한다. TableEG[11]는 실제 분포를 모사하는 오류를 LLM으로 생성한다. 이들은 LLM의 의미론적 판단을 활용하지만, 관찰 대상이 실행 전에 고정되어 있어 탐색 과정에서 대상을 스스로 확장하는 자율성은 제한적이다.

두 접근 방법을 종합하면 자율성이 높아질수록 탐지 범위는 넓어지지만 결과의 검증 가능성은 낮아지는 경향이 있다. ReAct[4], Toolformer[12] 등의 자율적 도구 사용 연구와 Self-RAG[13] 및 관련 연구[14]는 외부 피드백 없는 자기수정의 한계를 보여주며, 판단과 실행을 분리할 필요성을 시사한다. 데이터 에이전트의 자율성을 위계화한 연구[5]와 LLM 기반 품질 검증 연구[15] 역시 이러한 문제를 다루었으나, 동일한 모델과 데이터 조건에서 도구 사용의 통제 수준을 조절하여 검증 결과, 재현성, 오류 유형의 차이를 실증적으로 분석한 연구는 제한적이다. 본 연구는 이러한 공백을 대상으로 에이전트의 판단과 도구의 역할을 관찰·검증·종합 단계로 분리한 하네스를 설계하고, 자유 코드 실행 조건과 동일 벤치마크에서 비교하여 통제 수준에 따른 성능과 재현성의 차이를 분석한다.

III. 제안 방법

3.1 하네스 전체 구조

본 연구의 검증 방식은 프롬프트 하나에 완성된 결과를 한 번에 반환하는 일반 LLM 호출과 다르다. 하네스가 제공하는 도구와 환경 위에서 행동과 관찰을 반복하며, 식 (1)과 같이 최종 결과를 생성하는 에이전트 호출을 기반으로 한다.

$$result = Agent(M, prompt, H) \quad (1)$$

여기서 M은 추론을 담당하는 LLM이다. 에이전트 호출은 최근 에이전트 기반 개발 도구들이 공통적으로 지원하는 실행 방식을 따르는데, 이들 도구는 명령행 인터페이스(CLI, Command Line Interface)의 비대화형 모드와 전용 SDK를 통한 헤드리스(Headless) 실행을 함께 제공한다. 본 연구는 이를 활용해 에이전트 호출을 단일 호출로 구성하였으며, 이 방식은 데이터셋과 조건의 조합을 사용자의 개입 없이 일괄 실행할 수 있게 한다. 대화형 모드와 달리 실행 중 도구 사용을 사용자에게 확인받을 통로가 없으므로, 어떤 도구를 허용할지는 실행 전 권한 정책으로 미리 확정해 두어야 한다.

하네스 H는 에이전트의 실행 환경을 구성하는 요소들의 결합으로, 식 (2)와 같이 형식화된다.

$$H = (M, T, C, P, G) \quad (2)$$

여기서 T는 에이전트에게 노출되는 도구 집합, C는 실행 시점에 주입되는 컨텍스트이다, P는 도구 및 파일 접근을 규율하는 권한 정책이며, G는 실행 전후의 상태 변화를 기록하는 감사 장치이다. 기존 연구가 주로 C에 집중해 온 것과 달리, 본 연구는 T의 구성과 P·G에 의한 통제를 설계의 중심에 둔다. 그림 1은 이렇게 구성된 하네스의 전체 구조를 나타낸다.

에이전트가 정답 데이터를 참조하지 않았다는 전제는 검증 결과의 타당성을 좌우하므로 엄격히 보장되어야 한다. 다만 파일 시스템 접근에 제한이 없는 도구는 상대 경로 탐색 등으로 정답 경로에 접근할 수 있어, 물리적 분리만으로는 충분하지 않다. 이에 P·G는 물리적 분리, 샌드박스 파일시스템 읽기 차단, 파일 도구 권한 거부, 실행 전후 스냅샷·로그 기반 사후 감사의 네 가지 통제 수단으로 구성하였다. 앞의 세 수단은 접근을 사전에 차단하고, 사후 감사는 도구 제한이 없는 조건에서도 수행된 연산을 확인할 수 있는 유일한 수단이다. 탐지 결과는 컬럼명·행 인덱스 배열로만 저장되어 평가 절차와 분리된다.

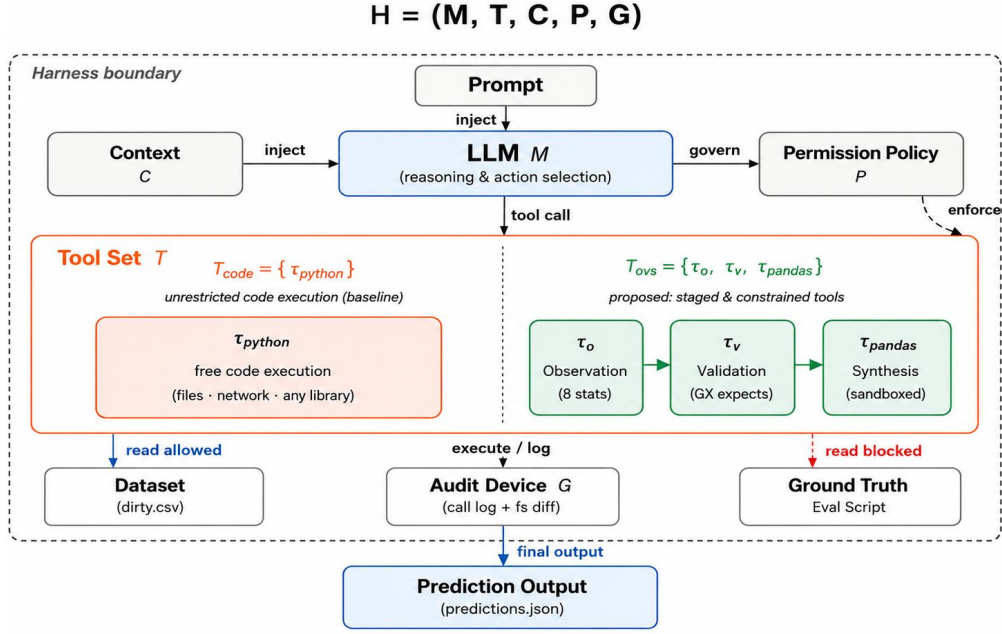


그림 1. 제안 하네스의 전체 구조

Fig. 1. Overall architecture of the proposed harness

이 하네스 위에서 도구 집합 T 를 구성하는 가장 단순한 방식은 에이전트에게 제약 없는 범용 코드 실행 도구 τ_{python} 하나만을 제공하는 것으로, 식 (3)과 같이 T_{code} 라 정의한다.

$$T_{code} = \{ \tau_{python} \} \quad (3)$$

τ_{python} 은 파일 시스템·네트워크 접근과 임의 라이브러리 import에 제한이 없는 일반 코드 실행 도구로, 에이전트는 이 도구 하나만으로 데이터 적재부터 결과 저장까지 전 과정을 스스로 작성한 코드로 수행한다. 이는 판단의 유연성을 극대화하는 대신, 연산 과정의 사후 재구성이나 실행마다 상이하게 생성되는 코드의 자원 사용량에 대한 사전 통제 수단은 제공하지 않는다. 실제로 동일한 프롬프트를 반복 입력해도 LLM이 매번 구조가 다른 코드를 생성한다는 점이 보고된 바 있다[16], [17]. 하네스가 도구 집합의 구성 자체를 설계 대상으로 삼아야 하는 이유가 여기에 있다.

3.2 관찰-검증-종합 3단계 도구 체계

본 절은 식 (3)의 T_{code} 가 지닌 문제를 해결하기

위해, 판단은 LLM이 수행하되 계산과 검증은 결정론적 도구에 위임한다는 원칙 아래 도구 집합을 관찰-검증-종합의 세 역할로 분리한다. LLM이 수치를 직접 집계하면 환각이 혼입될 수 있으므로 모든 수치 산출은 도구가 전담하고 LLM은 해석과 다음 행동 결정만 담당하도록 하였으며, 이에 따라 제안 방법의 도구 구성은 식 (4)와 같이 정의된다.

$$T_{ows} = \{ \tau_o, \tau_v, \tau_{pandas} \} \quad (4)$$

여기서 τ_o 는 결함 후보를 형성하는 관찰 도구군, τ_v 는 후보를 결함으로 확정하는 검증 도구군, τ_{pandas} 는 명명 도구로 표현되지 않는 연산을 위한 종합 도구이다.

관찰 도구가 적으면 포착 가능한 패턴이 좁아져 탐지 성능이 저하되고, 도구가 다양할수록 탐지 정확도가 높아지는 경향이 있다. 다만 데이터셋 특화 도구는 성능을 더 끌어올릴 수 있으나, 적합한 도구를 매번 설계해야 하는 비용을 초래한다. 이에 본 연구는 pandas 데이터프레임에 보편적으로 적용 가능한 list_columns, sample, value_counts, describe, null_ratio, dtype, correlate, cardinality_ratio의 8종만으로 τ_o 를 구성하였다. 이들은 모두 요약 통계만 반

환하도록 설계되어 응답 크기가 데이터 규모와 무관하게 일정하다. 이는 고카디널리티 컬럼을 원형 그대로 직렬화하면 컨텍스트가 급격히 팽창하고 구조 이해 성능도 저하된다는 선행 연구[18], [19]와, 이것이 LLM 기반 오류 탐지의 주요 병목으로 지적되어 온 점[20], [21]을 반영한 설계이다.

τ_o 가 자체 구현 도구로 구성된 것과 달리, τ_v 는 검증 결과가 결합 확정이라는 최종 판단으로 직결되어 계산 로직의 정확성이 신뢰성을 좌우한다. 따라서 직접 구현 대신 산업계에서 광범위하게 검증된 오픈소스 프레임워크 GX Core의 표현식(Expectation)을 사용한다(고유성 8종, 수치 13종, 유효성 15종, 완결성 2종, 스키마 7종, 규모 3종, 다중원천 2종 등 총 50종의 expect 계열 규칙제공). 다만 GX Core는 기본 설정이 위반 행 인덱스를 반환하지 않아 완전 반환 모드로 강제하였으며, 이때도 표시되는 위반 행 수에는 50건 상한이 존재해 후술할 종합 단계 발생의 구조적 원인이 된다.

τ_{pandas} 는 run_pandas_code 하나로 구성된, τ_{python} 과 유사하지만 동일하지 않은 코드 실행 도구이다. 실행 환경은 pandas, numpy, 이미 적재된 공유 데이터프레임만 참조하도록 제한되고 허용된 빌트인 함수 목록 밖의 호출(파일 열기, 모듈 import, 네트워크 접근 등)은 차단되며, 반환값은 2,000자로 절단되고 명명 도구를 먼저 시도하고 표현이 안 되는 사소한 경우에만 사용하라는 사용 우선순위가 명시되어 있다. τ_o , τ_v 는 검증된 관찰·검증 로직을 구조화해 제공하므로, 표현력의 상한이 낮더라도 실제로달 성능은 오히려 더 높고 안정적일 수 있다.

τ_o , τ_v , τ_{pandas} 는 모두 함수의 집합으로, 각 단계의 입력은 호출할 함수명과 파라미터이고 출력은 해당 함수의 출력값이다. 예를 들어 Beers 데이터셋의 state 컬럼을 검증하는 과정을 보면, 관찰 τ_o 단계는 value_counts(column="state")를 호출해 결측치 비율을 반환값으로 받고, 이를 근거로 state를 검증 후보로 지목한다. 검증 τ_v 단계는 expect_column_values_to_not_be_null(column="state")을 호출해 결측행의 인덱스 목록을 반환값으로 받는다. 종합(τ_{pandas}) 단계는 이 인덱스를 반영한 교차 컬럼을 검사하는 코드를 입력받아 실행하고, 결측이 정보 손실이 아닌 통합 표기임을 판정한 결과를 출력한다.

종합(τ_{pandas}) 단계가 분화되는 근본 이유는 τ_o , τ_v 가 고정된 도구로만 구성되어, 목록에 없는 오류 유형은 애초에 다룰 수 없기 때문이다. 이 경우 에이전트는 관찰 단계의 단서를 τ_{pandas} 로 직접 재구성해야 한다. 검증 도구의 위반 행 표시 상한도, 상한을 초과하는 컬럼에서는 τ_v 가 결합의 존재와 총 건수만 확정하고 전체 위반 행 목록은 τ_{pandas} 로 별도 산출해야 한다. 이는 저비용 요약으로 먼저 선별하고 확정된 컬럼에만 전체 목록 조회 비용이 발생하도록 강제한다는 점에서 효율적이다. 다만 종합 단계의 τ_{pandas} 호출은 새 판단이 아니라 이미 확정된 판단을 산출물로 조립하는 경우가 많기 때문에 자유 코드 의존과는 구분해서 해석해야 한다.

실행 도구 궤적에 대한 특성을 정량화하기 위하여 두 지표를 정의한다. 길이 K의 도구 호출 시, 관찰 대비 검증 호출 비율 ρ_{obs} 는 식 (5)로, 관찰 대비 종합 호출 비율 ρ_{free} 는 식 (6)으로 정의된다.

$$\rho_{obs} = \frac{\sum_{t=1}^K \Pi(\tau_t \in \tau_o)}{\sum_{t=1}^K \Pi(\tau_t \in \tau_v)} \quad (5)$$

$$\rho_{free} = \frac{\sum_{t=1}^K \Pi(\tau_t \in \tau_{pandas})}{\sum_{t=1}^K \Pi(\tau_t \in \tau_o)} \quad (6)$$

여기서 지시 함수 $\Pi(\cdot)$ 는 참이면 1, 거짓이면 0을 갖는다. 두 지표는 판단과 직접 관련된 호출만을 대상으로 하며, load_dataset처럼 판단과 무관한 호출은 분모·분자 어디에도 포함하지 않아 재적재가 지표를 희석시키지 않는다. 같은 이유로 분모도 전체 호출 수 K가 아니라 비교 대상 두 범주의 호출 수로 한정하였다. 두 지표가 포착하려는 것은 전체 실행에서의 절대적 비중이 아니라 인접 단계 사이의 상대적 무게이기 때문이며, 이 논리에서 ρ_{obs} 의 역수에 해당하는 검증 대비 관찰 비율은 별도로 정의하지 않았다. ρ_{obs} 는 결합 확정까지의 탐색 폭을, ρ_{free} 는 명명 도구 카탈로그가 해당 데이터셋의 결합을 얼마나 충분히 표현하는지를 반영한다.

IV. 실험

4.1 실험 데이터셋

실험은 오류 탐지 연구에서 널리 사용되는 4개 벤치마크 데이터셋을 대상으로 수행하였다. 각 데이터셋의 구성과 주된 오류 유형은 표 1과 같다. 네 데이터셋은 모두 셀 단위 정답을 제공하며, 선행 연구[2], [15]가 공통으로 사용한 벤치마크이다. 이 중 일부 데이터셋은 식별자 컬럼이 실제 행 위치보다 1만큼 큰 값을 가지는데, 이러한 불일치는 탐지 결과 전체를 무효화할 수 있으므로 하네스의 결과 파서가 정규식으로 0 기반 행 위치만을 추출하도록 하여 정규화하였다.

표 1. 실험 데이터셋의 구성

Table 1. Composition of experimental datasets

Dataset	Row	Col.	Main error type
Hospital	1000	20	character substitution typos missing values
Beers	2410	11	field merging unit-text contamination
Flights	2376	7	multi-source conflicts inconsistent time formats large amounts of missing data
Rayyan	1000	11	category fragmentation character encoding contamination date rotation contamination

실험은 모든 조건에서 컨텍스트를 동일하게 고정하여, 조건 간 차이가 컨텍스트 정보량이 아니라 도구 구성 및 모델 역량에서 기인하도록 통제하였다. 컨텍스트에는 결측을 나타내는 관용적 표기 (null/NaN, N/A, Not Available, Contact Airline 등)를 결측으로 인식할 것과, 무게·크기 등 숫자형 컬럼은 통상 단위 없이 사용된다는 것이라는 두 가지 일반 규칙을 포함하였다. 아울러 주입된 결측 오류 비율이 높았던 Beers(state), Flights(sched_dep_time, act_dep_time, sched_arr_time, act_arr_time), Rayyan(article_jvolumn, article_jissue)에는 결측을 허용하지 않는다고 명시하였다.

이러한 컨텍스트 위에서, 실험은 세 조건으로 구성하였다. $T_{code} \cdot T_{ous}$ 실험에서는 Sonnet 5(high 추론)

으로 모델 요인을 통제하였고, 모델 규모 비교 실험에서는 Haiku 4.5·Opus 5(high 추론)를 사용하였다.

4.2 자유 코드 실행 조건 실험(T_{code})

T_{code} 조건은 도구 제한이 없는, 자유도가 가장 높은 대조군이다. 이 조건에서 에이전트는 사전에 정의된 절차 없이 관찰과 가설 생성, 검증으로 이어지는 순환 루프를 스스로 구성하였다.

표 2는 5회 실행한 결과의 평균 탐지 성능과 표준편차를 나타낸다. Hospital과 Beers는 각각 F1 0.994와 0.946으로 높은 성능을 보였으며, 표준편차도 각각 0.008과 0.043으로 낮아 안정적인 결과를 보였다. 반면 Flights와 Rayyan은 F1이 각각 0.622와 0.184로 낮았으며, 특히 Flights는 표준편차가 0.250으로 높아 실행에 따른 성능 변동이 크게 나타났다.

표 2. T_{code} 조건의 데이터셋별 탐지 성능

Table 2. Detection performance of the T_{code} condition by dataset

Dataset	Recall	Precision	F1	F1(std.)
Hospital	0.997	0.992	0.994	0.008
Beers	0.962	0.937	0.946	0.043
Flights	0.485	0.987	0.622	0.250
Rayyan	0.127	0.348	0.184	0.071

Beers에서 관측된 오탐은 abv 컬럼과 style 컬럼에서 발생하였다. 두 컬럼은 단위 표기나 서식이 다양하게 변형되어 나타나는 특성을 지니며, τ_{python} 이 이 서식 변형 전반을 결함으로 과잉 일반화한 것으로 해석된다. τ_{python} 에는 판단을 최종 결과로 확정하기 전에 걸러 낼 검증 절차가 없으므로, 판단의 유연성으로 인해 오탐이 곧바로 유입된다.

생성된 τ_{python} 의 실행 통계를 살펴보면, 4개 데이터셋에서 Hospital 19회, Beers 13회, Flights 11회, Rayyan 11회씩 총 54회의 코드 실행이 발생하였고, 이 중 실패는 Hospital·Beers·Rayyan에서 각 1건씩 총 3건이다. 3건 모두 콘솔 문자 인코딩에 기인한 환경 문제였으며, 판단 로직의 오류로 인한 실패는 한 건도 없었다. 즉 τ_{python} 이 생성한 판단 로직 자체는 신뢰할 수 있었으며, 문제로 남는 것은 로직의 정확성보다 앞서 확인한 오탐의 유입 쪽이었다.

τ_{python} 코드를 분석하면 대체로 네 단계를 거친다. 컬럼별 프로파일링으로 이상 신호를 먼저 포착한 뒤 정규식이나 허용값 목록으로 기대 형식의 위반을 검증한다. 이어서 같은 대상을 나타내는 행들을 묶어, 다수결에서 벗어난 소수값을 이상값 후보로 찾아낸다. 마지막으로 대표 사례를 직접 대조하여 근본 원인을 역추적한 뒤 전체 데이터에 재검증한다. 이 절차는 제안 방법의 관찰-검증-종합 구조와 기능적으로 대응하나, 각 단계가 명명된 도구가 아니라 매 실행마다 새로 작성되는 코드로 구현된다는 점에서 구별된다.

새성된 코드를 분석하면 Hospital은 함수 종속성 교차검증에 반복문 기반 이중 순회를 12개 컬럼에 적용했고, Beers는 행 단위 함수와 정규식을 결합한 비벡터화 패턴을 6개 호출에서 사용해 세션 내 최장 실행 시간을 기록했으며, Rayyan은 문자 인코딩 오염 탐지에 문자 단위 순회를 사용했다. 현재 규모에서는 모든 호출이 수 초 내 종료되어 체감 문제는 없었으나, 이는 T_{code} 조건에서 자원 사용량을 사전에 예측·통제할 수단이 없음을 보여준다.

4.3 하네스 구조 조건 실험(T_{ovs})

이번 실험에서는 3.2절에서 정의한 관찰·검증·종합 도구 체계 T_{ovs} 를 적용하였다(표 3). Hospital(F1 0.998, 표준편차 0.002)과 Beers(F1 0.995, 표준편차 0.008)는 매우 높은 탐지 성능과 안정성을 보였다. Flights는 F1 0.730이었으나 재현율이 5회 모두 0.583으로 완전히 동일해 구조적 병목의 존재를 시사했고, Rayyan은 F1 0.171, 표준편차 0.057로 네 데이터셋 중 가장 낮고 불안정했다.

표 3. T_{ovs} 조건의 데이터셋별 탐지 성능

Table 3. Detection performance of the T_{ovs} condition by dataset

Dataset	Recall	Precision	F1	F1(std.)
Hospital	0.997	0.998	0.998	0.002
Beers	0.992	0.998	0.995	0.008
Flights	0.583	0.976	0.730	0.007
Rayyan	0.123	0.290	0.171	0.057

컬럼 단위로 오류 원인을 보면 주로 세 가지 원인으로 분류된다. 첫째, 값이 다른 그럴듯한 값으로 바뀌어 버려져 원본과 대조하지 않는 한 원칙적으로 탐지가 불가능한 오류다. Flights의 act_arr/dep_time은 비결측 오류의 85~88%가 이런 형태(예: 9:32 a.m.→9:22 a.m.)여서 재현율이 0.39~0.33에 그쳤다. 전체 주입 오류 948건 중 article_jcreated_at(월/일 순서 뒤 바뀜, 예: 2/15/04→4/2/15) 컬럼이 722건(76.2%)을 차지하였다. 둘째, 컨텍스트에 결측이면 안 됨으로 명시된 컬럼은 결측형 오류를 거의 완벽하게 탐지한 반면, 명시되지 않은 컬럼(Beers의 city, Rayyan의 mojobake 오염 3개 컬럼)은 별도 지침 없이 스스로 식별해야 해 탐지가 불완전했다. 이 중 Rayyan의 author_list는 전체 오류의 8.9%를 차지함에도 재현율 손실이 두드러졌다. 셋째, 정밀도 손실은 대부분 원본에 이미 존재하던 비표준 값을 정당하게 의심했으나 주입된 오류만 정답으로 인정한 평가 기준의 한계에 가까웠다. Flights의 src(항공사 코드 vs 사이트명 형식 혼재)와 Rayyan의 journal_issn·article_language가 대표적이다.

Rayyan에서는 article_jcreated_at, author_list, article_jissue가 전체 오류의 대부분을 차지하였다. Rayyan은 실행 5회 기준 표준편차(0.057)가 크게 나타나 10회로 표본을 늘려 재계산(0.048) 후, 편차의 원인을 컬럼별로 분석하였다. auth_list는 오류 후보를 찾는 규칙 자체는 동일했지만, 후보를 원본 데이터와 대조해 오류 여부를 재검증했는지에 따라 재현율이 0%에서 70%대로 양극화되며 F1을 크게 좌우했다. 반면 article_jissue 같은 수치형 컬럼은 재현율이 항상 100%였지만, 판정 허용폭이 좁고 넓음에 따라 후보 집합이 포함관계로 확장되면서 정밀도만 0.1~1.0 사이로 요동쳤다.

오류 패턴이 단순하고 균일한 Hospital·Beers는 매 실행 동일한 범위가 일관되게 검증되어 표준편차가 0.002~0.008에 그친 것과 대조적이다.

도구 호출 로그를 분류하여 산출한 궤적 지표는 표 4에 요약하였다. τ_{ovs} 는 1.14에서 8.50까지 데이터셋마다 큰 폭으로 나타났다. τ_v 호출이 τ_o 호출보다 많아지면 얼마든지 1 미만이 될 수 있다. 관찰은 전 컬럼에 넓게, 검증은 좁혀진 소수 컬럼에만 적용되는 패턴이 네 데이터셋 모두에서 공통적으로 나

타났다. 두 지표는 데이터셋에 따라서도 뚜렷한 차이를 보인다. Hospital은 τ_{ovs} 가 1.14로 가장 낮고 τ_{free} 도 0.11로 최저인데, 정규식 위반과 결측 규칙만으로 판단이 대부분 종결되어 τ_v 호출이 58회에 이른 결과다. 반대로 Rayyan은 τ_v 호출이 2회에 불과하고 τ_{free} 가 1.29로 가장 높다. 문자 인코딩 오염이나 날짜 회전 오염과 같이 검증 도구 카탈로그로 직접 표현하기 어려운 오류 유형이 다수를 차지하여, 관찰에서 얻은 단서를 τ_{pandas} 로 재구성하여야 하였기 때문으로 해석된다. Beers 역시 τ_{free} 가 1.03으로 높는데, 필드 병합 오류의 위반 행을 확정하는 과정에서 τ_{pandas} 호출이 반복되었기 때문이다.

표 4. 제안 방법의 데이터셋별 도구 호출 개적

Table 4. Tool-call trajectories of the proposed method by dataset

Metric	Hospital	Beers	Flights	Rayyan
τ_o	66	29	31	17
τ_v	58	5	15	2
τ_{pandas}	7	30	19	22
Load	19	19	19	25
Total calls	150	83	84	66
τ_{ovs}	1.14	5.80	2.07	8.50
τ_{free}	0.11	1.03	0.61	1.29

그림 2는 Beers 데이터셋의 도구 호출 타임라인으로, 세로축은 호출 순번이고 가로축은 도구 범주이다. τ_o 는 실행 초반에 밀집하여 전 컬럼을 광범위하게 조사하는 반면, τ_v 는 실행 후반에 소수의 군집으로 나타나 관찰로 좁혀진 컬럼에만 선택적으로 적용된다. 아울러 τ_{pandas} 호출이 데이터 적재 호출과 쌍을 이루어 반복적으로 등장하는 것은, τ_{pandas} 실행 전마다 데이터프레임을 다시 적재하는 방식이 일관되게 나타나기 때문이다.

다만 τ_{pandas} 개입이 모두 카탈로그 표현력 부족 때문인 것은 아니다. 세부적으로 컬럼 단위로 분석하면 Beers의 state는 다른 컬럼에서 이미 쓰인 `expect_column_values_to_not_be_null`을 호출하지 않고 τ_{pandas} 의 `isna()` 판단으로 대체되었고, Rayyan의 `article_title:journal_title:author_list`는 해당 오류 유형을 다룰 τ_v 표현식이 카탈로그에 없어 τ_{pandas} 가 판단을 전담하였다.

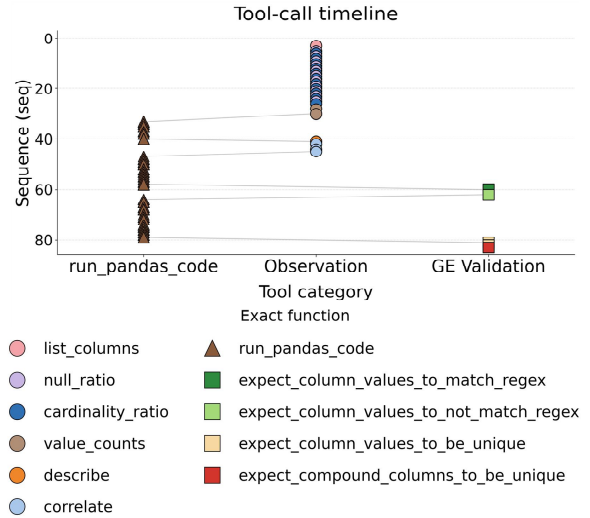


그림 2. Beers 데이터셋의 도구 호출 타임라인

Fig. 2. Tool-call timeline for the Beers dataset

4.4 모델 규모 비교 실험

마지막 비교 실험은 모델만 교체해 남은 성능 격차가 도구 설계가 아니라 모델 역량에서 비롯되는지를 표 5에서 확인할 수 있다.

Haiku 4.5는 Hospital·Beers에서 급격히 저하했으나(F1 0.095, 0.728), Sonnet 5에서 0.998, 0.995로 회복되어 컨텍스트 반영 능력의 한계로 해석된다. Flights는 F1이 규모에 비례해 개선되었으나(0.647→0.730→0.886), `act_arr/dep_time`의 분단위 이탈자는 도구 카탈로그의 표현력 한계로 규모와 무관하게 잔존하였다. Rayyan은 Opus 5로 갈수록 재현율은 개선되나(0.071→0.203) 정밀도가 하락하는데, 원본의 비표준 값을 오답 처리하는 평가 기준의 한계에 가깝다. 정리하면, 모델 규모 확대는 컨텍스트 반영 능력 부족(Hospital, Beers)은 해소하지만, 도구 카탈로그 표현력(Flights)이나 평가 기준(Rayyan)의 병목은 해소하지 못한다.

표 5. 모델 규모별 탐지 성능(F1)

Table 5. Detection performance by model(F1)

	Haiku 4.5	Sonnet 5	Opus 5
Hospital	0.095	0.998	1.000
Beers	0.728	0.995	1.000
Flights	0.647	0.730	0.886
Rayyan	0.071	0.171	0.203

V. 결 론

본 논문은 LLM 에이전트의 자율적 판단이 주는 탐지 유연성은 유지하면서도, 판단은 LLM이 맡고 계산·검증은 결정론적 도구가 담당하도록 관찰·검증·종합 세 단계에 도구를 배정하는 하네스 방법론을 제안하였다. 관찰 도구는 요약 통계만 반환하고 검증 도구가 확정한 위반만 최종 결과에 반영되도록 설계하여 재현성과 검증 가능성을 확보하였다.

이를 T_{code} 조건과 동일한 조건에서 4개 벤치마크에 걸쳐 5회씩 반복 실행한 결과, 도구를 단계별로 통제하더라도 성능은 저하되지 않았으며, 이미 F1 0.99 이상이었던 Hospital을 제외한 일부 데이터셋(Beers·Flights)에서는 오히려 다소 향상되었다. 재현성 또한 네 데이터셋 모두에서 표준편차가 감소해(Flights 0.250→0.007) 일관되게 개선되었음이 확인되었다. 다만 Rayyan은 두 조건 모두 F1이 0.2 이하로 낮아 난제로 남았다.

본 연구에서는 적은 수의 범용 관찰 도구만을 사용하였는데, 향후에는 필드 간 상관관계 분석 등으로 관찰·검증 도구의 표현 범위를 확장하고 평가 기준의 한계도 보완할 계획이다.

References

- [1] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Bießmann, and A. Grafberger, "Automating Large-Scale Data Quality Verification", Proc. VLDB Endow., Rio de Janeiro, Brazil, Vol. 11, No. 12, pp. 1781-1794, Aug. 2018. <https://doi.org/10.14778/3229863.3229867>.
- [2] M. Mahdavi, et al., "Raha: A Configuration-Free Error Detection System", Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD), Amsterdam, Netherlands, pp. 865-882, Jun.-Jul. 2019. <https://doi.org/10.1145/3299869.3324956>.
- [3] S. Zhang, Z. Huang, and E. Wu, "Data Cleaning Using Large Language Models", Proc. IEEE 41st Int. Conf. Data Eng. Workshops (ICDEW), Hong Kong, China, pp. 28-32, May 2025. <https://doi.org/10.1109/ICDEW67478.2025.00008>.
- [4] S. Yao, et al., "ReAct: Synergizing Reasoning and Acting in Language Models", Int. Conf. Learn. Represent. (ICLR), Kigali, Rwanda, pp. 1-33, May 2023. <https://doi.org/10.48550/arXiv.2210.03629>.
- [5] Y. Luo, G. Li, J. Fan, and N. Tang, "Data Agents: Levels, State of the Art, and Open Problems", Companion Proc. ACM SIGMOD/PODS Int. Conf. Manage. Data (SIGMOD), Bengaluru, India, pp. 1-9, May-Jun. 2026. <https://doi.org/10.48550/arXiv.2602.04261>.
- [6] T. Rehberger, T. Hütter, L. Ehrlinger, and W. Wöß, "Evaluating Data Quality Tools: Measurement Capabilities and LLM Integration", arXiv preprint, arXiv:2604.09163, pp. 1-16, Apr. 2026. <https://doi.org/10.48550/arXiv.2604.09163>.
- [7] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "HoloClean: Holistic Data Repairs with Probabilistic Inference", Proc. VLDB Endow., Munich, Germany, Vol. 10, No. 11, pp. 1190-1201, Aug.-Sep. 2017. <https://doi.org/10.48550/arXiv.1702.00820>.
- [8] M. Abughazala, M. Ibiyo, H. Muccini, and M. Sharaf, "Quality by Prompt: LLM-Powered Transformation of Data Quality Requirements Into Great Expectations", Proc. Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA), Salerno, Italy, LNCS Vol. 16081, pp. 130-147, Sep. 2025. https://doi.org/10.1007/978-3-032-04190-6_9.
- [9] M. Abughazala and H. Muccini, "DQGen: Scalable Metadata-Driven Automation for Data Quality Validation in Data-Intensive Applications", Proc. Eur. Conf. Softw. Archit. (ECSA) Tracks Workshops, Limassol, Cyprus, LNCS Vol. 15982, pp. 363-380, Sep. 2025. https://doi.org/10.1007/978-3-032-04403-7_31.
- [10] F. Biester, M. Abdelaal, and D. Del Gaudio, "LLMClean: Context-Aware Tabular Data Cleaning via LLM-Generated OFDs", New Trends in Database and Information Systems (ADBIS 2024 Short Papers), Commun. Comput. Inf. Sci., Vol.

- 2186, pp. 68-78, Aug. 2024. https://doi.org/10.1007/978-3-031-70421-5_7.
- [11] X. Liu, J. Chen, B. Hu, Y. Sun, X. Chen, and S. Song, "Towards Practical Benchmarking of Data Cleaning Techniques: On Generating Authentic Errors via Large Language Models", arXiv preprint, arXiv:2507.10934, pp. 1-14, Jul. 2025. <https://doi.org/10.48550/arXiv.2507.10934>.
- [12] T. Schick, et al., "Toolformer: Language Models Can Teach Themselves to Use Tools", Adv. Neural Inf. Process. Syst. (NeurIPS), New Orleans, Louisiana, USA, Vol. 36, pp. 68539-68551, Dec. 2023. <https://doi.org/10.48550/arXiv.2302.04761>.
- [13] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection", Int. Conf. Learn. Represent. (ICLR), Vienna, Austria, pp. 1-30, May 2024. <https://doi.org/10.48550/arXiv.2310.11511>.
- [14] J. Huang, et al., "Large Language Models Cannot Self-Correct Reasoning Yet", Int. Conf. Learn. Represent. (ICLR), Vienna, Austria, pp. 1-14, May 2024. <https://doi.org/10.48550/arXiv.2310.01798>.
- [15] A. Akella, A. Kaul, K. Narayanam, and S. Mehta, "Quality Assessment of Tabular Data using Large Language Models and Code Generation", Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP) Ind. Track, Suzhou, China, pp. 2713-2748, Nov. 2025. <https://doi.org/10.48550/arXiv.2509.10572>.
- [16] Y. Song, T. Sun, X. Tang, P. Rajput, T. F. Bissyandé, and J. Klein, "Measuring LLM Code Generation Stability via Structural Entropy", Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE) New Ideas Emerg. Results (NIER) Track, Seoul, Korea, pp. 1-5, Nov. 2025. <https://doi.org/10.48550/arXiv.2508.14288>.
- [17] A. Paleyes, R. Sendyka, D. Robinson, C. Cabrera, and N. D. Lawrence, "Code Roulette: How Prompt Variability Affects LLM Code Generation", Proc. Int. Workshop Large Lang. Models Code (LLM4Code), Rio de Janeiro, Brazil, pp. 1-11, Apr. 2026. <https://doi.org/10.48550/arXiv.2506.10204>.
- [18] M. Chen, et al., "Empowering Tabular Data Preparation with Language Models: Why and How?", arXiv preprint, arXiv:2508.01556, pp. 1-20, Aug. 2025. <https://doi.org/10.48550/arXiv.2508.01556>.
- [19] Y. Sui, M. Zhou, M. Zhou, S. Han, and D. Zhang, "Table Meets LLM: Can Large Language Models Understand Structured Table Data? A Benchmark and Empirical Study", Proc. ACM Int. Conf. Web Search Data Min. (WSDM), Mérida, Mexico, pp. 645-654, Mar. 2024. <https://doi.org/10.48550/arXiv.2305.13062>.
- [20] W. Ni, et al., "ZeroED: Hybrid Zero-Shot Error Detection Through Large Language Model Reasoning", Proc. IEEE Int. Conf. Data Eng. (ICDE), Hong Kong, China, pp. 3126-3139, May 2025. <https://doi.org/10.1109/ICDE65448.2025.00234>.
- [21] Z. Liang, "Efficient Representations for High-Cardinality Categorical Variables in Machine Learning", Int. Conf. Adv. Mach. Learn. Data Sci. (AMLDS), Tokyo, Japan, pp. 1-8, Jan. 2025. <https://doi.org/10.48550/arXiv.2501.05646>.

저자소개

최 권 택 (Kwon-Taeg Choi)



2006년 2월 : 연세대학교
컴퓨터공학과(공학석사)
2011년 2월 : 연세대학교
컴퓨터공학과(공학박사)
2016년 3월 ~ 현재 : 강남대학교
소프트웨어융용학부 교수
관심분야 : 가상현실, 증강현실,
모바일컴퓨팅, 기계학습, HCI