# 자율적 가설 검증 및 논문 생성 에이전트 시스템: 독창성 평가와 다중 모달 가상 실험을 위한 아키텍처 및 구현 심층 보고서

## 1. 서론: 과학적 발견의 자동화와 에이전트 워크플로우의 필요성

과학적 발견(Scientific Discovery)의 과정은 역사적으로 인간의 직관과 반복적인 실험, 그리고 방대한 문헌 조사를 통한 지식의 확장에 의존해 왔습니다. 그러나 최근 대규모 언어 모델(Large Language Models, LLM)과 에이전트 기반 인공지능(Agentic AI)의 비약적인 발전은 연구의 전 주기를 자동화할 수 있는 새로운 가능성을 열었습니다. 본 보고서는 사용자로부터 가설이나 질문을 입력받아, 기존 학술 자료를 바탕으로 독창성을 판단하고, 가상 실험을 수행하며, 최종적으로 논문을 작성하는 'AI 과학자(AI Scientist)' 시스템의 구현 방안을 상세히 기술합니다.1

특히 본 시스템은 사용자의 입력이 단순한 '질문'인지 검증이 필요한 '가설'인지에 따라 분기되는 동적 워크플로우를 채택하며, 가설 검증을 위해 서로 다른 세 가지 접근 방식을 제안하고 사용자가 이를 선택하게 함으로써 인간-AI 협업(Human-AI Collaboration)을 극대화하도록 설계되었습니다. 이는 단순히 텍스트를 생성하는 것을 넘어, 코드 실행(Code Execution), 문헌 기반의 논리적 추론, 그리고 데이터 시뮬레이션을 포함한 실질적인 연구 행위를 수행하는 시스템입니다.3

본 보고서는 시스템의 전체 아키텍처를 설계하고, 핵심 모듈인 독창성 판단 엔진, 실험 설계 및 실행 모듈, 그리고 결과 저장 및 논문 생성 파이프라인을 기술적인 깊이를 담아 제안합니다. 또한, 각 단계에서 발생할 수 있는 환각(Hallucination) 문제를 최소화하고 연구의 신뢰성을 확보하기 위한 검증 메커니즘과 데이터베이스 스키마 설계까지 포괄적으로 다룹니다.

## 2. 시스템 매크로 아키텍처 및 사용자 인터랙션 흐름

전체 시스템은 **사용자 인터페이스(UI) 레이어**, **오케스트레이터(Orchestrator) 에이전트**, **전문가 에이전트 풀(Pool)**, **지식 및 데이터 레이어**, 그리고 \*\*실행 샌드박스(Execution Sandbox)\*\*로 구성됩니다. 이 구조는 확장성과 모듈성을 보장하며, 복잡한 연구 과제를 관리 가능한 단위 작업으로 분해하여 처리합니다.5

### 2.1 사용자 의도 파악 및 분기 처리 (Intent Classification)

시스템의 진입점에서는 사용자의 입력이 단순한 정보 검색을 위한 '질문(Question)'인지, 새로운 과학적 사실을 입증하려는 '가설(Hypothesis)'인지를 구별하는 것이 필수적입니다. 이를 위해 자연어 이해(NLU) 모듈이 입력 텍스트의 언어적 특성과 요구 사항을 분석합니다.

* **질문 모드(Question Mode):** 사용자가 기존 지식에 대한 확인이나 설명을 요구할 때 활성화됩니다. 시스템은 즉시 RAG(Retrieval-Augmented Generation) 파이프라인을 가동하여 학술적 근거를 검색하고, 해당 아이디어의 존재 여부나 실현 가능성을 보고합니다. 이 결과는 '질문 DB'에 저장되어 향후 아이디어 뱅크로 활용됩니다.
* **가설 모드(Hypothesis Mode):** 사용자가 "만약 ~라면 ~일 것이다" 형태의 인과적 주장을 제시할 때 활성화됩니다. 이 경우 시스템은 더욱 복잡한 '독창성 판단' 프로세스를 시작하며, 기존 논문과의 유사성을 검토한 후, 실험을 통한 검증 단계로 진입합니다.

### 2.2 핵심 워크플로우 설계

사용자가 가설을 입력했을 때의 워크플로우는 다음과 같이 세분화됩니다.

1. **독창성 및 선행 연구 검증:** Semantic Scholar 또는 OpenAlex API를 활용하여 입력된 가설과 의미적으로 유사한 선행 연구를 검색합니다. 만약 동일한 연구가 존재한다면 이를 사용자에게 알리고 프로세스를 종료하거나 수정을 제안합니다.7
2. **접근 방식(Approach) 제안:** 가설이 독창적이라고 판단되면, 시스템은 이를 검증하기 위한 3가지 서로 다른 실험적 접근 방식(예: 전산 시뮬레이션, 통계적 메타 분석, 이론적 모델링)을 생성하여 사용자에게 제시합니다.9
3. **가상 실험(Virtual Experiment) 수행:** 사용자가 선택한 접근 방식에 따라 에이전트는 Python 코드를 작성하거나, 논리적 추론 체인을 형성하여 가상의 실험을 수행합니다. 이 과정은 Docker 컨테이너와 같은 격리된 환경에서 이루어집니다.11
4. **결과 리포팅 및 DB 축적:** 실험의 성공/실패 여부, 생성된 데이터, 로그, 중간 분석 결과가 구조화된 형태(JSON)로 데이터베이스에 저장됩니다. 사용자는 도메인별, 시간순으로 정렬된 결과를 열람하고 수정할 수 있습니다.13
5. **논문 생성(Paper Generation):** 사용자가 축적된 실험 결과 중 특정 항목들을 선택하면, 시스템은 이를 종합하여 서론, 방법론, 결과, 고찰을 포함한 완전한 형태의 학술 논문을 작성합니다.14

## 3. 독창성 판단 엔진: RAG 기반의 차별성 평가

과학적 발견의 핵심은 기존 지식과의 차별성, 즉 '독창성(Novelty)'에 있습니다. 본 시스템은 단순한 키워드 매칭을 넘어선 심층적인 독창성 판단 알고리즘을 구현합니다.

### 3.1 상대적 이웃 밀도(Relative Neighbor Density, RND) 기반 평가

기존의 코사인 유사도(Cosine Similarity) 방식은 특정 가설이 기존 문헌과 얼마나 '비슷한가'만 측정할 뿐, 해당 연구 분야의 밀집도를 고려하지 못하는 한계가 있습니다. 이를 극복하기 위해 **상대적 이웃 밀도(RND)** 알고리즘을 도입합니다.16

* **원리:** 가설을 고차원 벡터 공간에 임베딩한 후, 주변에 위치한 기존 논문들의 밀도를 계산합니다. 만약 가설이 기존 연구들이 밀집된 클러스터(Dense Cluster) 내부에 위치한다면 독창성이 낮다고 판단합니다. 반면, 연구가 희소한 영역(Sparse Region)에 위치하면서도 논리적 타당성을 유지한다면 높은 독창성 점수를 부여합니다.
* **구현:** OpenAI의 text-embedding-3-large 또는 과학 기술 특화 임베딩 모델(SciBERT 등)을 사용하여 가설과 검색된 상위 $k$개의 논문 초록을 벡터화합니다. 이후, 가설 벡터와 이웃 벡터들 간의 거리 분포를 분석하여 '독창성 스코어'를 산출합니다.

### 3.2 지식 그래프(Knowledge Graph)를 이용한 갭 분석

단순한 텍스트 유사성을 넘어 개념 간의 관계를 파악하기 위해 지식 그래프를 활용합니다. 시스템은 가설에 포함된 핵심 엔티티(예: '단백질 X', '질병 Y', '알고리즘 Z')를 추출하고, 학술 지식 그래프상에서 이들 사이의 연결 경로를 탐색합니다.17

* **링크 예측(Link Prediction):** 그래프상에서 직접적인 연결(Edge)이 없는 두 개념을 연결하는 가설은 독창적일 가능성이 높습니다. 시스템은 경로 탐색을 통해 기존에 보고된 적 없는 메커니즘이나 상관관계를 제안하는지 검증합니다.
* **SciSpace 및 Semantic Scholar 연동:** API를 통해 실시간으로 최신 논문 데이터를 조회하여, 가설과 정확히 일치하는 선행 연구가 있는지 '침해 분석(Infringement Analysis)' 수준의 정밀 검사를 수행합니다.19

## 4. 가설 검증을 위한 3가지 접근 방식 제안

시스템은 사용자가 입력한 가설의 성격과 도메인에 따라, 이를 검증할 수 있는 최적의 접근 방식 3가지를 생성합니다. 이는 사용자가 연구의 방향성을 결정하는 데 중요한 역할을 하며, 각 방식은 서로 다른 방법론적 깊이와 자원을 요구합니다.

### 4.1 접근 방식 A: 코드 기반 자율 시뮬레이션 (The In-Silico Computational Approach)

이 방식은 가설을 검증하기 위해 실제 실행 가능한 코드를 생성하고 시뮬레이션을 돌리는 가장 적극적인 형태의 검증입니다. 주로 전산학, 데이터 과학, 시스템 생물학 등의 분야에 적합합니다.11

* **워크플로우:**
  1. 에이전트가 가설을 검증하기 위한 알고리즘이나 수학적 모델을 Python 코드로 작성합니다.
  2. **E2B** 또는 **Docker** 샌드박스 환경에서 코드를 실행합니다. 이때 필요한 라이브러리(PyTorch, NumPy, Scikit-learn 등)를 동적으로 설치합니다.
  3. 합성 데이터(Synthetic Data)를 생성하거나 공개 데이터셋을 다운로드하여 모델을 학습 또는 테스트합니다.
  4. 실행 결과(정확도, 수렴 속도, p-value 등)와 시각화된 그래프를 결과 보고서로 출력합니다.
* **장점:** 정량적인 데이터와 재현 가능한 코드를 제공하므로 결과의 신뢰도가 높습니다.
* **구현 기술:** LangChain의 Code Interpreter 도구, Jupyter Kernel 실행 환경, E2B Sandboxing API.12

### 4.2 접근 방식 B: 문헌 기반 논리적 합성 및 메타 분석 (The Literature Synthesis Approach)

실험 데이터가 없거나 코드 실행이 불가능한 인문사회과학, 이론 물리학, 의학 리뷰 등의 분야에서 유용한 방식입니다. 기존 문헌들의 파편화된 증거를 수집하고 논리적으로 연결하여 가설을 지지하거나 반박합니다.23

* **워크플로우:**
  1. 가설과 관련된 수백 편의 논문을 검색하고, 'Tree of Thoughts' 프롬프팅을 통해 논리적 구조를 수립합니다.25
  2. 각 논문에서 핵심 결과(Experimental Results)와 결론을 추출하여 비교 테이블을 생성합니다.
  3. 상충되는 주장들을 대조(Contrastive Analysis)하고, 가설이 이러한 모순을 어떻게 해결할 수 있는지 논리적으로 서술합니다.
  4. 다중 문서 요약(Multi-document Summarization) 기술을 적용하여 기존 연구들의 합의점과 한계점을 도출하고, 이를 바탕으로 가설의 타당성을 평가합니다.
* **장점:** 방대한 지식을 종합하여 새로운 통찰을 제공하며, 물리적 실험 없이도 높은 수준의 이론적 검증이 가능합니다.

### 4.3 접근 방식 C: 데이터 기반 확률적 예측 모델링 (The Probabilistic Predictive Approach)

이는 실제 실험을 수행하는 대신, LLM이 가진 방대한 지식과 추론 능력을 활용하여 실험 결과를 '예측'하는 방식입니다. 특히 사회과학 설문조사 시뮬레이션이나 경제학적 행동 모델링에 유효합니다.27

* **워크플로우:**
  1. 가설 검증을 위한 가상의 피실험자(Persona) 집단을 생성합니다 (예: "30대 미국 거주 IT 종사자 100명").
  2. LLM에게 각 페르소나를 부여하고, 가설 상황에 대한 반응이나 의사결정을 시뮬레이션하도록 요청합니다(Generative Simulation).
  3. 수집된 시뮬레이션 데이터를 통계적으로 분석하여 가설의 지지 여부를 판단합니다.
  4. 이 과정에서 LLM의 편향을 보정하기 위해 다양한 프롬프트 변형(DiVeRSe Prompting)을 사용하여 결과의 강건성(Robustness)을 확보합니다.29
* **장점:** 비용과 시간이 많이 소요되는 인간 대상 실험을 예비적으로 수행해볼 수 있으며, 다양한 시나리오(What-if analysis)를 빠르게 테스트할 수 있습니다.

## 5. 가상 실험 실행 및 결과 보고서 생성

사용자가 특정 접근 방식을 선택하면, 시스템은 \*\*자율 실험 에이전트(Autonomous Experiment Agent)\*\*를 가동합니다. 이 에이전트는 계획(Planning), 실행(Execution), 디버깅(Debugging), 분석(Analysis)의 순환 루프를 통해 실험을 완수합니다.4

### 5.1 실험 계획 및 코드 생성 (Planner & Coder)

'Planner' 에이전트는 실험의 목표를 단계별로 분해합니다(Task Decomposition). 예를 들어, "데이터 전처리 -> 모델 구현 -> 학습 -> 평가 -> 시각화"의 단계를 수립합니다. 이어 'Coder' 에이전트는 각 단계에 맞는 실행 가능한 코드를 생성합니다. 이때 LLM이 생성한 코드가 문법적으로 정확할 뿐만 아니라 논리적으로도 타당한지 검증하는 것이 중요합니다.

### 5.2 샌드박스 실행 및 자동 디버깅 (Executor & Debugger)

생성된 코드는 보안이 확보된 **E2B 샌드박스** 내에서 실행됩니다. 만약 실행 중 에러가 발생하면, 'Debugger' 에이전트가 에러 로그(Traceback)를 분석하여 코드를 수정하고 재실행합니다. 최근 연구에 따르면, 이러한 자율적 디버깅 루프는 실험 성공률을 비약적으로 높여줍니다.31 시스템은 최대 $N$번까지 재시도를 허용하며, 실패 시 사용자에게 수동 개입을 요청하거나 다른 접근 방식을 제안합니다.

### 5.3 결과 보고서 및 데이터베이스 저장

실험이 완료되면 'Analyst' 에이전트가 결과 데이터(CSV, JSON)와 이미지(PNG)를 분석하여 텍스트 형태의 보고서를 작성합니다. 이 보고서는 **실험의 목적, 방법, 주요 결과, 결론**으로 구성됩니다.

데이터베이스 저장 스키마 (PostgreSQL 예시):

데이터의 체계적인 관리를 위해 관계형 데이터베이스와 벡터 스토어를 결합한 하이브리드 스키마를 사용합니다.32

| **Table Name** | **Description** | **Key Columns** |
| --- | --- | --- |
| hypotheses | 사용자 가설 및 메타데이터 | id, user\_id, domain, content, novelty\_score, created\_at |
| experiments | 수행된 실험 정보 | id, hypothesis\_id, approach\_type, status, code\_snippet |
| results | 실험 결과 데이터 및 보고서 | id, experiment\_id, summary\_text, data\_json, image\_paths |
| qa\_logs | 질문 모드에서의 질의응답 | id, question, answer, embedding\_vector |

특히 results 테이블은 사용자가 나중에 논문을 작성할 때 선택할 수 있는 단위 블록이 되며, 도메인(domain)과 타임스탬프(created\_at)를 기준으로 인덱싱되어 빠른 정렬과 조회를 지원합니다.

## 6. 논문 생성 및 통합 (Paper Composition)

사용자가 DB에 축적된 실험 결과들 중 유의미한 것들을 선택하면, 'Writer' 에이전트가 이를 바탕으로 논문을 작성합니다. 이 과정은 단순한 나열이 아니라, 하나의 일관된 논리적 흐름(Narrative)을 갖추도록 설계됩니다.14

### 6.1 계층적 개요 작성 (Hierarchical Outlining)

먼저 선택된 실험 결과들을 종합하여 논문의 전체 개요를 작성합니다. 이때 **WriteHERE**와 같은 계층적 계획 기법을 사용하여 논문의 각 섹션(서론, 관련 연구, 방법론, 결과, 논의)에 들어갈 내용을 배치합니다. 실험 결과가 서로 상충되거나 부족한 부분이 있다면, 이를 '한계점(Limitations)' 섹션에 포함시키거나 추가 실험을 제안하는 문장을 생성합니다.34

### 6.2 섹션별 집필 및 인용 관리 (Drafting & Citation)

각 섹션은 해당 역할에 특화된 프롬프트를 통해 작성됩니다.

* **관련 연구(Related Work):** 독창성 판단 단계에서 수집된 선행 연구들을 요약하고, 본 연구와의 차별성을 강조하는 방식으로 기술합니다.
* **방법론(Methodology):** 실험 실행 단계에서 기록된 코드와 파라미터 설정을 자연어로 변환하여, 재현 가능할 정도로 상세하게 기술합니다.
* **결과(Results):** 생성된 그래프와 통계 수치를 캡션과 함께 배치하고, 이에 대한 해석을 덧붙입니다.

모든 주장은 \cite{} 포맷으로 인용 부호를 달아 BibTeX 데이터베이스와 연동되도록 하여 학술적 엄밀성을 보장합니다. 최종 결과물은 LaTeX 소스 코드 또는 PDF 형태로 렌더링되어 사용자에게 제공됩니다.

## 7. 구현을 위한 기술 스택 및 전략

성공적인 시스템 구현을 위해 다음과 같은 최신 기술 스택을 권장합니다.

* **LLM Orchestration:** **LangGraph** 또는 **CrewAI**. 순환적 흐름(Loop)과 상태 관리(State Management)가 필수적이므로, 단순한 Chain보다는 Graph 기반의 프레임워크가 적합합니다.35
* **LLM Model:** 추론과 코딩에는 **Claude 3.5 Sonnet**이나 **GPT-4o**와 같은 고성능 모델을 사용하고, 단순 요약이나 데이터 처리에는 **DeepSeek-V3**나 **Llama 3**와 같은 비용 효율적인 모델을 사용하는 하이브리드 전략을 취합니다.37
* **Knowledge Retrieval:** **Semantic Scholar API**와 **Tavily Search API**. 학술적 깊이와 최신 웹 정보의 폭을 동시에 커버하기 위함입니다.39
* **Execution Environment:** **E2B**. AI가 생성한 코드를 안전하고 빠르게 실행할 수 있는 클라우드 기반 샌드박스로, 장시간 실행되는 실험 세션을 안정적으로 지원합니다.12
* **Frontend:** **Streamlit** 또는 **React**. 사용자가 실험 진행 상황을 실시간으로 확인하고, 생성된 중간 결과물(차트, 텍스트)을 상호작용적으로 검토할 수 있는 인터페이스가 필요합니다.42

## 8. 결론 및 향후 전망

본 보고서에서 제안한 시스템은 연구자의 아이디어 발상부터 논문 작성까지의 전 과정을 AI가 보조하거나 주도하는 'Agentic Science'의 실체적 구현 방안입니다. 독창성 판단을 위한 RND 알고리즘, 도메인에 따른 3가지 실험 접근 방식, 그리고 샌드박스 기반의 실행 환경은 기존 LLM 서비스들이 제공하지 못했던 실질적인 '검증' 기능을 제공합니다.

이러한 시스템은 연구의 진입 장벽을 낮추고, 반복적이고 소모적인 실험 설계 및 코딩 작업에서 연구자를 해방시킬 것입니다. 나아가, 축적된 실험 데이터와 가설들이 상호 연결되면서 시스템 자체가 스스로 새로운 연구 주제를 발굴하고 진화하는 '자기 진화형 AI 과학자(Self-Evolving AI Scientist)'로 발전할 잠재력을 가지고 있습니다.43 다만, AI가 생성한 결과물의 윤리적 책임과 환각으로 인한 오류 가능성에 대해서는 인간 연구자의 최종 검토와 엄격한 검증 프로세스가 반드시 수반되어야 할 것입니다.

#### 참고 자료

1. LLM-Based Scientific Agents - Emergent Mind, 1월 16, 2026에 액세스, <https://www.emergentmind.com/topics/llm-based-scientific-agents>
2. Autonomous Agents for Scientific Discovery: Orchestrating Scientists, Language, Code, and Physics - arXiv, 1월 16, 2026에 액세스, <https://arxiv.org/html/2510.09901v1>
3. From Models to Scientists: Building AI Agents for Scientific Discovery - Kempner Institute, 1월 16, 2026에 액세스, <https://kempnerinstitute.harvard.edu/research/deeper-learning/from-models-to-scientists-building-ai-agents-for-scientific-discovery/>
4. Evaluating Sakana's AI Scientist for Autonomous Research: Wishful Thinking or an Emerging Reality Towards 'Artificial Research Intelligence' (ARI)? - arXiv, 1월 16, 2026에 액세스, <https://arxiv.org/html/2502.14297v2>
5. Agentic Workflows for Economic Research: Design and Implementation - arXiv, 1월 16, 2026에 액세스, <https://arxiv.org/html/2504.09736v1>
6. What Are Agentic Workflows? Patterns, Use Cases, Examples, and More | Weaviate, 1월 16, 2026에 액세스, <https://weaviate.io/blog/what-are-agentic-workflows>
7. 1월 16, 2026에 액세스, <https://scispace.com/agents/paper-novelty-check-kz3yroxz#:~:text=Paper%20Novelty%20Check%20Using%20SciSpace,all%20with%20reproducible%2C%20transparent%20workflows.>
8. And that is semantic scholar, https://www.semanticscholar.org/ | Hacker News, 1월 16, 2026에 액세스, <https://news.ycombinator.com/item?id=42178675>
9. LLMs for Experiment Design in Scientific Domains: Are We There Yet? - OpenReview, 1월 16, 2026에 액세스, <https://openreview.net/pdf?id=dIEeOwrmOe>
10. Hypothesis Generation with Large Language Models - arXiv, 1월 16, 2026에 액세스, <https://arxiv.org/html/2404.04326v1>
11. Code execution with MCP: building more efficient AI agents - Anthropic, 1월 16, 2026에 액세스, <https://www.anthropic.com/engineering/code-execution-with-mcp>
12. E2B: Give Your AI Agent a Safe Workspace, 1월 16, 2026에 액세스, <https://medium.com/@ecommerce_plan/e2b-give-your-ai-agent-a-safe-workspace-f080f9981dd0>
13. A Database Schema for Standardized Data and Metadata Collection in Agricultural Experiments - MDPI, 1월 16, 2026에 액세스, <https://www.mdpi.com/2073-445X/14/9/1816>
14. The First 'AI Scientist' to Publish a paper: Sakana AI | by Priyanshu Pansari - Medium, 1월 16, 2026에 액세스, <https://medium.com/@priyanshu.pansari/the-first-ai-scientist-to-pass-peer-review-what-this-means-for-research-43011dca62de>
15. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, 1월 16, 2026에 액세스, <https://www.semanticscholar.org/paper/The-AI-Scientist%3A-Towards-Fully-Automated-Discovery-Lu-Lu/33161a5a9b5dcb635b5a97475e6a6209a69ada7d>
16. Enabling AI Scientists to Recognize Innovation: A Domain-Agnostic Algorithm for Assessing Novelty - arXiv, 1월 16, 2026에 액세스, <https://arxiv.org/html/2503.01508v1>
17. SciAgents: Automating Scientific Discovery Through Bioinspired Multi‐Agent Intelligent Graph Reasoning - PMC - PubMed Central, 1월 16, 2026에 액세스, <https://pmc.ncbi.nlm.nih.gov/articles/PMC12138853/>
18. Relational Memory in LLMs: How Relational Memory Solves Finite Context - DEV Community, 1월 16, 2026에 액세스, <https://dev.to/kgthedev/relational-memory-in-llms-how-relational-memory-solves-finite-context-ej5>
19. PatentBench - AI-driven Novelty Search Benchmark - Patsnap, 1월 16, 2026에 액세스, <https://www.patsnap.com/benchmark>
20. [PDF] Using Scopus and OpenAlex APIs to retrieve bibliographic data for evidence synthesis. A procedure based on Bash and SQL | Semantic Scholar, 1월 16, 2026에 액세스, <https://www.semanticscholar.org/paper/64baac0c663e6381b2334ae79ba3202104320971>
21. [2402.18679] Data Interpreter: An LLM Agent For Data Science - arXiv, 1월 16, 2026에 액세스, <https://arxiv.org/abs/2402.18679>
22. e2b/code-interpreter - NPM, 1월 16, 2026에 액세스, <https://www.npmjs.com/package/@e2b/code-interpreter>
23. Towards Multi-Document Summarization of Scientific Articles:Making Interesting Comparisons with SciSumm, 1월 16, 2026에 액세스, <https://gvrkiran.github.io/content/Towards-Multi-Document-Summarization-of-Scientific-Articles.pdf>
24. Do Multi-Document Summarization Models Synthesize? - MIT Press Direct, 1월 16, 2026에 액세스, <https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00687/124262/Do-Multi-Document-Summarization-Models-Synthesize>
25. What is Tree Of Thoughts Prompting? - IBM, 1월 16, 2026에 액세스, <https://www.ibm.com/think/topics/tree-of-thoughts>
26. Tree of Thoughts Prompting (ToT) - Humanloop, 1월 16, 2026에 액세스, <https://humanloop.com/blog/tree-of-thoughts-prompting>
27. Social science researchers use AI to simulate human subjects | Stanford Report, 1월 16, 2026에 액세스, <https://news.stanford.edu/stories/2025/07/ai-social-science-research-simulated-human-subjects>
28. Using Large Language Models to Simulate and Predict Human Decision-Making, 1월 16, 2026에 액세스, <https://eilamshapira.com/blog/2025/using-large-language-models-to-simulate-and-predict-human-decision-making/>
29. DiVeRSe (Diverse Verifier on Reasoning Step) - Learn Prompting, 1월 16, 2026에 액세스, <https://learnprompting.org/docs/advanced/ensembling/diverse_verifier_on_reasoning_step>
30. Towards Fully Autonomous Research Powered by LLMs: Case Study on Simulations - arXiv, 1월 16, 2026에 액세스, <https://arxiv.org/html/2408.15512v2>
31. Helping LLMs Improve Code Generation Using Feedback from Testing and Static Analysis, 1월 16, 2026에 액세스, <https://arxiv.org/html/2412.14841v1>
32. A Metadata Schema for Data from Experiments in the Social Sciences, 1월 16, 2026에 액세스, <https://documents1.worldbank.org/curated/en/099945502062327217/pdf/IDU081c960a8049b504197099ff0d12be0b95375.pdf>
33. How Can I Connect AI to My PostgreSQL? Full Guide for 2025 - Julius AI, 1월 16, 2026에 액세스, <https://julius.ai/articles/how-can-i-connect-ai-to-my-postgres>
34. Beyond Outlining: Heterogeneous Recursive Planning for Adaptive Long-form Writing with Language Models - arXiv, 1월 16, 2026에 액세스, <https://arxiv.org/html/2503.08275v3>
35. Run Python code for AI data analysis with LangGraph - DeployApps, 1월 16, 2026에 액세스, <https://deployapps.dev/blog/data-analyst-agent-langgraph-genezio/>
36. Building a LLM Agent to Directly Interact with a Database | by Ayush Gupta | Medium, 1월 16, 2026에 액세스, <https://medium.com/@ayush4002gupta/building-an-llm-agent-to-directly-interact-with-a-database-0c0dd96b8196>
37. Best Open Source LLM for Scientific Research & Academia in 2025 - SiliconFlow, 1월 16, 2026에 액세스, <https://www.siliconflow.com/articles/en/best-open-source-llm-for-scientific-research-academia>
38. Top 10 open source LLMs for 2025 - NetApp Instaclustr, 1월 16, 2026에 액세스, <https://www.instaclustr.com/education/open-source-ai/top-10-open-source-llms-for-2025/>
39. What's the best search API for LLM pipelines? Looking for something that integrates search + content extraction. : r/n8n - Reddit, 1월 16, 2026에 액세스, <https://www.reddit.com/r/n8n/comments/1ozlrlk/whats_the_best_search_api_for_llm_pipelines/>
40. The Complete Guide to Web Search APIs for AI Applications in 2025 - Firecrawl, 1월 16, 2026에 액세스, <https://www.firecrawl.dev/blog/top_web_search_api_2025>
41. AutoGen Code Interpreter with E2B, 1월 16, 2026에 액세스, <https://e2b.dev/blog/community-contribution-autogen-code-interpreter-with-e2b>
42. Building A RAG Agent for Research Paper Analysis with CrewAI - Turing College, 1월 16, 2026에 액세스, <https://www.turingcollege.com/blog/building-a-rag-agent-for-research-paper-analysis-with-crewai>
43. Autonomous Scientific Discovery Through Hierarchical AI Scientist Systems - Preprints.org, 1월 16, 2026에 액세스, <https://www.preprints.org/manuscript/202507.1951>