자율 연구 인텔리전스**(ARI):** 재귀적 과학 발견을 위한 아키텍처 및 구현 보고서

초록 **(Executive Summary)**

대규모 언어 모델(LLM)의 발전은 인공지능을 단순한 보조 도구에서 능동적인 행위자(Agent)로 전환시키는 패러다임의 변화를 이끌고 있다. 특히 과학적 발견의 전 과정을 자동화하는 자율 연구 인텔리전스(Autonomous Research Intelligence, ARI)는 가설 생성부터 실험 계획, 코드 실행, 논문 작성, 그리고 동료 평가(Peer Review)에 이르는 연구의 생명주기를 스스로 수행하는 것을 목표로 한다. 본 보고서는 Sakana AI의 'The AI Scientist', FutureHouse의 'PaperQA', 그리고 'FAIRBridge'와 같은 최신 연구 성과를 바탕으로, 사용자 쿼리에서 요구된 "가설 할당, 실험 계획 및 실행, 논문 작성 및 리뷰, 도메인별 DB 크롤링 및 RAG 기반의 지식 확장, 그리고 재귀적 가설 창조"를 수행하는 에이전트의 아키텍처를 심층적으로 분석한다. 특히 선형적 워크플로우를 넘어선 에이전트 트리 검색(Agentic Tree Search) 방식의 실험 설계, 도메인 특화 크롤링을 통한 벡터 데이터베이스(Vector DB)의 동적 확장, 그리고 다중 페르소나를 활용한 적대적 리뷰 시스템의 기술적 구현 방안을 제시한다.

I. 서론**:** 인공지능 과학자의 태동과 개념적 프레임워크

1.1 자율 연구의 정의와 패러다임 전환

전통적인 과학 연구는 인간 연구자가 가설을 수립하고, 실험을 설계하며, 데이터를 분석하여 결론을 도출하는 인간 중심의 프로세스였다. 그러나 ARI 시스템은 이러한 인지적, 물리적 작업을 알고리즘화하여 연구의 루프(Loop)를 닫는 것을 목표로 한다. 이는 단순히 실험실 장비를 자동화하는 것을 넘어, '무엇을 연구할 것인가'라는 상위 레벨의 의사결정까지 AI에게 위임하는 것을 의미한다. 최근 Sakana AI가 공개한 'The AI Scientist'는 이러한 가능성을 실증적으로 보여주었으며, 템플릿 기반의 초기 모델에서 벗어나 스스로 코드를 수정하고 실험 경로를 탐색하는 에이전트형 시스템으로 진화하고 있다.[1]

1.2 연구 생명주기의 **4**단계 자동화

ARI 시스템의 핵심은 과학적 방법론을 계산 가능한 모듈로 분해하는 것이다. 본 보고서에서 제안하는 아키텍처는 사용자의 요구사항을 반영하여 다음의 4단계 순환 구조를 따른다.

1. 지식 습득 및 가설 생성 **(Knowledge Acquisition & Ideation):** 도메인별 저널 사이트를 크롤링하여 벡터 DB를 구축하고, RAG(Retrieval-Augmented Generation)를 통해 기존 문헌을 분석한 후 새로운 가설을 창출한다.
2. 실험 계획 및 실행 **(Planning & Execution):** 가설을 검증하기 위한 실험 플랜을 수립하고, 실제 코드(Python 등)를 생성하여 가상 환경에서 실행한다. 이 과정에서 발생하는 오류를 스스로 수정(Self-Correction)한다.
3. 논문 작성 및 시각화 **(Manuscript Authoring):** 실험 결과를 바탕으로 학술 논문

형식(LaTeX)의 문서를 작성하고, 도표와 그래프를 생성한다.

4. 동료 평가 및 재귀적 발전 **(Review & Recursion):** 가상의 리뷰어 에이전트들이 논문을 비판적으로 평가하고, 저자 에이전트는 이를 반영하여 수정본을 제출한다. 최종 승인된 논문은 다시 시스템의 DB로 편입되어 새로운 가설의 씨앗이 된다.[2]

II. 지식 엔진: 도메인 특화 크롤링과 벡터 DB 확장 전략

과학적 발견은 기존 지식의 어깨 위에서 이루어진다. 따라서 ARI의 가장 기초적이면서도 중요한 기능은 최신 연구 동향을 실시간으로 파악하고 이를 자신의 지식베이스로 내재화하는 것이다. 사용자의 요구사항인 "도메인 별 크롤링 후 벡터 DB 확장"을 실현하기 위해, 단순한 웹 스크래핑을 넘어선 '지능형 크롤링 아키텍처'가 요구된다.

2.1 딥 웹(Deep Web) 접근을 위한 에이전트형 크롤러

일반적인 검색 엔진은 학술 데이터베이스의 깊은 곳에 있는 원천 데이터나 PDF 파일, 보충 자료(Supplementary Material)에 접근하는 데 한계가 있다. 이를 해결하기 위해 **FAIRBridge**와 같은 아키텍처는 LLM을 활용하여 웹 포털의 구조를 동적으로 해석한다.[4]

2.1.1 동적 쿼리 변환 및 폼 핸들링

과학 데이터베이스(예: PubMed, arXiv, PDB, Zenodo 등)는 각기 다른 검색 인터페이스와 API 규격을 가진다. ARI의 크롤러 모듈은 '타겟 리소스'에 대한 자연어 설명을 입력받아, 해당 사이트의 검색 필드(Form)를 인식하고 적절한 쿼리를 생성한다.

- 리소스 발견 요청: 에이전트는 "최신 단백질 접힘 구조 데이터셋"과 같은 추상적 목표를 수립한다.
- 인터페이스 해석: Selenium이나 Playwright와 결합된 시각-언어 모델(VLM)이 웹 페이지의 DOM(Document Object Model) 트리를 분석하여 검색창, 날짜 필터, 다운로드 버튼의 위치를 식별한다.
- 인증 및 접근: 유료 저널이나 기관 인증이 필요한 경우, 사전에 설정된 자격 증명(Credentials)을 활용하여 로그인 세션을 관리하고 쿠키를 유지한다.[4]

2.1.2 도메인별 파싱 전략

수집된 데이터는 비정형(PDF, HTML) 형태이다. 이를 벡터화하기 위해서는 정교한 파싱이 필요하다.

- 과학 논문 파싱: **Grobid**나 **Nougat**과 같은 전문 파서(Parser)를 사용하여 PDF에서 제목, 저자, 초록, 본문, 참고문헌을 구조적으로 분리한다. 특히 참고문헌 섹션을 본문과 분리하는 것은 RAG 수행 시 인용 정보가 내용으로 오인되는 환각(Hallucination)을 방지하는 데 필수적이다.[6]
- 메타데이터 추출: 논문의 텍스트뿐만 아니라 출판 연도, 피인용 수, 저널의 임팩트 팩터(IF) 등의 메타데이터를 함께 추출하여 벡터 DB의 필터링 조건으로 활용한다.

2.2 벡터 데이터베이스 구축 및 임베딩 최적화

수집된 텍스트는 기계가 이해할 수 있는 벡터로 변환되어야 한다. 이때 범용적인 임베딩 모델보다는 과학 도메인에 특화된 모델을 사용해야 정확도가 보장된다.

2.2.1 도메인 특화 임베딩 모델 선정

OpenAI의 text-embedding-3와 같은 범용 모델은 일반적인 의미론적 유사도는 잘 포착하지만, 전문 용어(예: 유전자 이름, 화학식, 수식) 간의 미세한 뉘앙스를 놓칠 수 있다. 따라서 ARI 시스템은 도메인에 따라 다른 임베딩 모델을 선택적으로 로드한다.

- 생명과학**/**의학**:** BioBERT 또는 PubMedBERT 기반의 임베딩을 사용하여 질병-약물 관계의 정확도를 높인다.
- 컴퓨터 과학**:** Specter2나 SciNCL과 같이 인용 그래프(Citation Graph)를 학습한 모델을 사용하여, 단순히 단어가 겹치는 논문이 아니라 논리적 맥락이 유사한 논문을 찾도록 한다.[7]

2.2.2 의미론적 청킹**(Semantic Chunking)**과 하이브리드 검색

긴 논문을 벡터화할 때 단순히 글자 수(예: 500 토큰)로 자르는 것은 문맥을 파괴할 위험이 있다. ARI는 논문의 논리적 구조(서론, 방법, 결과)를 인식하여 섹션 단위로 청킹(Chunking)을 수행한다. 또한, 벡터 DB(ChromaDB, Milvus 등) 구축 시 하이브리드 검색**(Hybrid Search)** 전략을 채택한다.

- 밀집 벡터 검색**(Dense Vector Search):** 의미적 유사성을 기반으로 관련 문서를 찾는다.
- 희소 벡터 검색(Sparse Keyword Search - BM25): 특정 고유명사나 수치(예: "ResNet-50", "p < 0.05")가 정확히 일치하는 문서를 찾는다.
  이 두 가지 점수를 가중 합산하여 검색의 재현율(Recall)과 정밀도(Precision)를 동시에 확보한다.6

기능	기존 키워드 검색	**ARI**의 벡터**/**하이브리드 검색
쿼리 이해	단순 매칭	의미론적 의도 파악
문맥 인식	불가능	인용 관계 및 섹션 구조 인식
데이터 소스	정적 인덱스	실시간 크롤링 기반 동적 업데이트
임베딩	없음 (TF-IDF 등)	도메인 특화 (Specter, BioBERT)

III. 가설 생성 및 신규성 평가: RAG와 LBD의 결합

지식 베이스가 구축되면, 에이전트는 이를 바탕으로 "아직 탐구되지 않은" 영역을 찾아내야 한다. 이는 문헌 기반 발견(Literature-Based Discovery, LBD) 방법론과 RAG 기술의 융합을 통해 이루어진다.

3.1 문헌 기반 발견(LBD)과 지식 그래프

LBD는 서로 연결되지 않은 두 개념 A와 C 사이의 관계를, 매개 개념 B를 통해 추론하는 기법이다(A→B, B→C, 따라서 A→C?). ARI는 벡터 DB의 내용을 바탕으로 동적인 **지식 그래프(Knowledge Graph)**를 생성한다.[10]

- 트리플 추출: LLM을 이용해 논문 초록에서 (개체1, 관계, 개체2) 형태의 지식을 추출한다. (예: (Transformer, suffers_from, quadratic_complexity))
- 경로 탐색 및 가설 도출: 지식 그래프 상에서 끊어진 링크를 예측하거나, **KG-CoI(Knowledge-Grounded Chain of Ideas)** 기법을 활용하여 논리적 비약이 없는 가설을 생성한다. 예를 들어, "이전 연구에서 기법 X가 CNN의 과적합을 해결했다면, 구조적으로 유사한 문제(Y)를 가진 Transformer에도 기법 X를 적용해 볼 수 있다"는 식의 추론을 수행한다.

3.2 의미론적 학술 검색(Semantic Scholar)을 이용한 신규성 검증

생성된 아이디어가 이미 존재하는 연구인지 확인하는 절차는 필수적이다. 'The AI Scientist'는 이 과정에서 **Semantic Scholar API**를 적극적으로 활용한다.[13]

1. 아이디어 쿼리 변환: 생성된 가설(예: "Adaptive LayerNorm for RNN")을 검색 쿼리로 변환한다.
2. 유사 논문 검색: API를 통해 가장 관련성이 높은 상위 20개의 논문을 검색한다.
3. **LLM** 심사관의 평가: 검색된 논문들의 초록과 생성된 가설을 비교하여 '유사도 점수'와 '차별성'을 평가한다. 만약 기존 논문과 아이디어가 너무 유사하다면(유사도 임계값 초과), 시스템은 해당 아이디어를 폐기하고 다시 가설 생성 단계로 돌아가는 **거부 샘플링(Rejection Sampling)**을 수행한다.[13]

3.3 아이디어 템플릿의 구조화

실행 가능한 실험 계획을 수립하기 위해, 가설은 단순한 문장이 아닌 구조화된 JSON 형태로 정의된다.

JSON

```
{
  "Title": "가설의 제목",
```

```
  "Research_Question": "해결하고자 하는 핵심 질문",
  "Methodology": "제안하는 방법론 (예: 모델 아키텍처 변경, 새로운 손실 함수 도입)",
  "Experiment_Plan": "검증을 위한 실험 단계 (베이스라인 설정 -> 비교 실험 -> 결과 분석)",
  "Expected_Results": "예상되는 결과 및 기여점",
  "Novelty_Score": 8.5,
  "Feasibility_Score": 9.0
}
```

이러한 구조화된 출력은 다음 단계인 실험 설계 에이전트가 코드를 작성할 때 명확한 지침(Instruction) 역할을 한다.[13]

IV. 자율 실험 설계 및 실행: 에이전트 트리 검색(Agentic Tree Search)

가설이 확정되면 ARI는 이를 검증하기 위한 실험 단계로 진입한다. 이는 단순한 스크립트 실행이 아니라, 복잡한 문제 해결 과정이며, Sakana AI의 'The AI Scientist v2'에서 도입된 에이전트 트리 검색이 핵심적인 역할을 한다.[15]

4.1 에이전트 트리 검색(Agentic Tree Search) 아키텍처

기존의 선형적(Linear) 방식은 실험 코드를 한 번 작성하고 실패하면 멈추거나 처음부터 다시 시작해야 했다. 반면, 트리 검색 방식은 실험 과정을 여러 분기(Branch)를 가진 탐색 트리로 모델링한다.

- 루트 노드(**Root Node**): 초기 실험 코드 템플릿이나 베이스라인 모델.
- 자식 노드(**Child Node**): 하이퍼파라미터 변경, 모델 구조 수정, 데이터 전처리 방식 변경 등의 구체적인 실험 시도.
- 평가 및 가지치기(**Pruning**): 각 노드에서 실험을 실행한 후, 결과(정확도, 수렴 속도 등)를 평가한다. 결과가 좋지 않거나 실행 오류(Bug)가 해결되지 않는 노드는 '가지치기' 되고, 에이전트는 부모 노드로 돌아가(Backtracking) 다른 전략을 시도한다.[17] 이는 연구자가 실패한 실험을 버리고 다른 방법을 시도하는 과정을 모사한 것이다.

4.2 코딩 에이전트(Aider)와 자기 수정(Self-Correction) 루프

실제 코드를 작성하고 수정하는 역할은 **Aider**와 같은 LLM 기반 코딩 도구가 담당한다.[1]

1. 코드 생성: 실험 플랜에 따라 Python 스크립트(예: experiment.py, plot.py)를 작성하거나 수정한다.
2. 실행 및 오류 포착: 작성된 코드를 격리된 환경(Docker Container)에서 실행한다. 이때 발생하는 표준 출력(stdout)과 표준 에러(stderr)를 캡처한다.
3. 디버깅 루프: 에러 로그를 분석하여 원인을 파악하고, 코드를 수정하여 다시 실행한다. 이 과정은 최대 N회(예: 5회) 반복되며, 해결되지 않으면 해당 실험 분기는 실패로 간주된다.
4. 라이브러리 관리: 실험에 필요한 라이브러리가 없으면 pip install 명령어를 통해 환경을

동적으로 구성한다.

4.3 진화적 알고리즘 최적화 (ShinkaEvolve)

단순한 코드 수정을 넘어, 더 나은 알고리즘(예: 새로운 손실 함수, 최적화 기법)을 찾기 위해 ARI는 진화 연산(Evolutionary Computation) 전략을 사용한다. **ShinkaEvolve** 프레임워크는 LLM을 '돌연변이 연산자(Mutation Operator)'로 활용한다.[18]

- 모집단 생성: 다양한 알고리즘 변종 코드를 생성한다.
- 토너먼트 선택: 각 코드를 실행하여 성능을 평가하고 우수한 개체를 선택한다.
- 교차 및 변이: 우수한 코드들의 특징을 LLM이 분석하여 결합(Crossover)하거나 창의적으로 변형(Mutation)하여 다음 세대의 코드를 생성한다. 이 방식은 인간이 고안하기 어려운 복잡하고 효율적인 알고리즘을 발견하는 데 효과적이다.[18]

V. 논문 작성 및 멀티모달 시각화

실험이 완료되면 ARI는 축적된 데이터(로그, CSV, 체크포인트)를 바탕으로 인간이 읽을 수 있는 학술 논문을 작성한다.

5.1 LaTeX 기반의 논문 자동 생성

전문적인 학술 문서를 생성하기 위해 ARI는 Markdown이 아닌 **LaTeX** 포맷을 사용한다.

- 섹션별 작성: 서론, 관련 연구, 방법론, 실험, 결론 등 표준적인 학술 논문 구조에 따라 텍스트를 생성한다.
- 문맥 인식 인용: '관련 연구' 섹션을 작성할 때는 앞서 구축한 벡터 DB를 조회(RAG)하여, 실험 주제와 가장 연관성이 높은 논문들을 찾아 적절한 위치에 인용(\cite{key})한다.[20]
- 컴파일 검증: 작성된 .tex 파일을 컴파일하여 PDF를 생성한다. 컴파일 오류가 발생하면 코딩 에이전트와 동일한 방식으로 오류 로그를 분석하여 LaTeX 문법을 수정한다.

5.2 VLM을 활용한 도표 검증 및 개선

AI가 생성한 논문의 가장 큰 약점 중 하나는 가독성이 떨어지거나 의미 없는 도표(Figure)였다. ARI v2는 이를 해결하기 위해 **시각-언어 모델(Vision-Language Model, VLM)**을 활용한 피드백 루프를 도입했다.[3]

1. 도표 생성: matplotlib 등을 이용해 결과 그래프를 이미지 파일로 저장한다.
2. 시각적 비평: VLM(예: GPT-4V)이 이미지를 보고 "범례가 데이터 포인트를 가린다", "축의 레이블이 너무 작다", "색상 대비가 낮아 구분이 어렵다" 등의 구체적인 피드백을 제공한다.
3. 코드 수정: 플롯 생성 코드를 수정하여 피드백을 반영, 출판 가능한 수준(Publication-Quality)의 이미지를 완성한다.

5.3 환각 방지 및 진실성 확보 (PaperQA)

논문의 신뢰성을 위해 **PaperQA** 스타일의 검증 메커니즘이 적용된다.

- 인용 검증: 본문에 삽입된 모든 인용구는 실제 DB에 존재하는 논문이어야 한다. 에이전트가 존재하지 않는 논문을 인용하려 하면(Hallucination), 시스템이 이를 감지하고 차단하거나 실제 문헌 검색을 강제한다.[21]
- 팩트 체크: "실험 결과 정확도가 10% 향상되었다"는 주장이 실제 실험 로그 데이터와 일치하는지 대조한다.

VI. 적대적 동료 평가(Peer Review) 및 수정 시스템

사용자의 핵심 요구사항 중 하나인 "각기 다른 리뷰어가 존재해 리뷰를 받고 수정하는" 프로세스는 ARI의 품질 관리를 위한 최후의 보루이다.

6.1 다중 페르소나 리뷰어 에이전트

단일 에이전트가 스스로를 평가하면 편향될 수 있으므로, 시스템은 서로 다른 관점을 가진 여러 리뷰어 에이전트를 생성한다.[23]

- **System Prompt:** "당신은 세계적인 AI 학회(ICLR/NeurIPS)의 엄격한 심사위원입니다. 논문의 독창성, 방법론적 건전성, 실험의 철저함을 비판적으로 평가하십시오."
- 리뷰어 페르소나 예시:
  - *Reviewer A (이론가):* 수식의 정합성과 이론적 배경의 탄탄함을 중점적으로 본다.
  - *Reviewer B (실험가):* 베이스라인 모델 선정의 적절성, 하이퍼파라미터 튜닝의 공정성을 본다.
  - *Reviewer C (응용가):* 실제 문제 해결에 대한 기여도와 임팩트를 평가한다.

6.2 리뷰-반박(Rebuttal)-수정의 반복 루프

단순히 점수만 매기는 것이 아니라, 저자 에이전트와 리뷰어 에이전트 간의 상호작용이 일어난다.[1]

1. 리뷰 생성: 각 리뷰어는 점수(1-10)와 상세한 코멘트(강점, 약점, 질문)를 작성한다.
2. 반박 및 계획 수립: 저자 에이전트는 리뷰를 분석하여 대응 전략을 짠다.
   - 텍스트 수정: 오해를 풀기 위해 설명을 보강한다.
   - 추가 실험: "베이스라인 모델 X와의 비교가 빠졌다"는 지적이 있으면, 실험 단계로 되돌아가 해당 모델을 추가 실험하고 결과를 업데이트한다. 이는 ARI의 가장 강력한 기능 중 하나이다.
3. 수정본 제출: 수정된 논문(Revision)과 반박문(Rebuttal Letter)을 제출한다.
4. 최종 결정: 리뷰어들은 수정본을 다시 평가하여 최종 게재 여부(Accept/Reject)를 결정한다.

VII. 재귀적 발견(Recursive Discovery): 닫힌 루프의 완성

ARI의 궁극적인 목표는 인간의 개입 없이 스스로 연구를 지속하는 개방형(Open-Ended)

발전이다.[18]

7.1 결과의 지식화 및 피드백

최종 승인된 논문은 단순한 결과물이 아니라, 시스템의 새로운 지식으로 환원된다.

- **DB 업데이트:** 생성된 논문은 다시 파싱되어 벡터 DB에 '신뢰할 수 있는 지식'으로 저장된다. 이는 다음 세대의 연구를 위한 기초 자료가 된다.
- 부정적 결과**(Negative Results)** 아카이빙**:** 실패한 실험(가지치기 된 트리 노드) 정보도 별도의 DB에 저장된다. 이는 향후 에이전트가 동일한 실패를 반복하지 않도록 하는 '반면교사' 데이터로 활용된다. 인간 연구 생태계에서는 부정적 결과가 잘 공유되지 않지만, ARI 생태계에서는 이것이 효율성의 핵심이 된다.[21]

7.2 미래 가설의 창조

생성된 논문의 '결론 및 향후 연구(Conclusion & Future Work)' 섹션은 다음 연구 사이클의 씨앗(Seed)이 된다.

- 재귀적 사이클**:**
  1. 연구 1: "A 기법이 B 문제에 효과적임을 증명함. 그러나 C 상황에서는 한계가 있음."
  2. 가설 생성기: "연구 1에서 C 상황에 한계가 있다고 했으므로, 이를 해결하기 위해 D 기법을 도입해보자."
  3. 연구 2: D 기법 실험 진행.
     이러한 연쇄 작용을 통해 ARI는 단편적인 연구를 넘어, 하나의 거대한 연구 줄기(Research Agenda)를 형성해 나갈 수 있다.

VIII. 기술적 구현 요약 및 성능 지표

8.1 시스템 아키텍처 다이어그램 **(**텍스트 기술**)**

1. **User Input:** 연구 주제 및 제약 사항.
2. **Knowledge Engine:** Crawler $\leftrightarrow$ Vector DB $\leftrightarrow$ Semantic Scholar API.
3. **Planner Agent:** Idea Generator $\rightarrow$ Tree Search Manager.
4. **Execution Layer:** Aider (Code) $\leftrightarrow$ Docker Sandbox (Runtime) $\leftrightarrow$ ShinkaEvolve (Optimization).
5. **Authoring Layer:** LaTeX Generator $\leftrightarrow$ VLM Critic $\leftrightarrow$ PaperQA Verifier.
6. **Review Layer:** Multi-Persona Reviewers $\leftrightarrow$ Author Agent (Rebuttal Loop).
7. **Recursion:** Generated Paper $\rightarrow$ Vector DB Ingestion.

8.2 비용 및 효율성

Sakana AI의 리포트에 따르면, 이러한 완전 자동화된 프로세스를 통해 논문 한 편을 작성하는 데

드는 비용(API 호출 비용 등)은 약 $15 (약 2만원) 수준이다.[27] 인간 연구자가 수 주에서 수 개월 걸릴 일을 수 시간 내에, 그것도 매우 저렴한 비용으로 수행할 수 있다는 점은 과학 연구의 경제성을 근본적으로 변화시킬 잠재력을 가진다.

결론 **(Conclusion)**

본 보고서에서 제시한 자율 연구 인텔리전스(ARI) 아키텍처는 사용자가 요구한 "가설-실험-논문-리뷰-재귀"의 전 과정을 포괄하는 통합 시스템이다. 특히 에이전트 트리 검색을 통한 비선형적 실험 관리, FAIRBridge 기반의 능동적 지식 습득, 그리고 페르소나 기반의 적대적 리뷰 시스템은 기존의 자동화 도구를 넘어선 진정한 의미의 '인공지능 과학자'를 구현하는 핵심 기술이다. 물론 환각(Hallucination)의 위험, 윤리적 문제, 그리고 완전히 새로운 패러다임을 창조하는 창의성의 한계 등은 여전히 해결해야 할 과제이나, 이러한 아키텍처의 발전은 과학적 발견의 속도와 규모를 기하급수적으로 확장시킬 것이다.

섹션 **1:** 자율 연구 인텔리전스**(ARI)** 아키텍처 개요

1.1 ARI의 핵심 철학**:** 도구에서 동료로

인공지능은 오랫동안 과학자들의 도구(Tool)였다. 데이터를 분석하거나, 시뮬레이션을 돌리거나, 특정 패턴을 찾는 데 사용되었다. 그러나 ARI는 AI를 **행위자(Agent)**로 격상시킨다. ARI는 연구의 주체로서 스스로 질문을 던지고 답을 찾는다. 사용자의 요구사항에 명시된 "가설을 할당받아 스스로 계획하고 수행하는" 능력은 ARI가 수동적인 계산기가 아니라 능동적인 연구자임을 의미한다.

이 시스템의 궁극적인 목표는 **'과학적 발견의 민주화'**와 **'연구 속도의 가속화'**이다. 인간의 인지적 한계(기억력, 주의력, 수면 필요 등)를 뛰어넘어, 24시간 쉬지 않고 가설을 검증하고 새로운 지식을 축적하는 시스템을 구축함으로써, 인류가 직면한 난제(신약 개발, 기후 변화, 신소재 등)를 해결하는 속도를 획기적으로 높일 수 있다.

1.2 시스템의 전체적인 워크플로우

본 아키텍처는 크게 네 가지의 순환 루프로 구성된다.

1.2.1 인지 루프 **(Cognitive Loop)**

- 입력**:** 사용자 쿼리 또는 이전 연구의 결론.
- 처리**:** RAG를 활용한 문헌 분석, 지식 그래프 탐색, 가설 생성(Ideation).
- 출력**:** 구체적이고 검증 가능한 가설 및 실험 설계안.

1.2.2 실행 루프 (Execution Loop)

- 입력: 실험 설계안.
- 처리: 코드 생성, 가상 환경에서의 실행, 오류 수정(Debugging), 데이터 수집.
- 출력: 실험 로그, 결과 데이터(CSV), 시각화 자료(PNG).

1.2.3 저작 및 평가 루프 (Publication Loop)

- 입력: 실험 결과 및 관련 문헌.
- 처리: 논문 작성, 시각적 비평(VLM), 동료 평가(Review), 수정(Revision).
- 출력: 최종 승인된 PDF 논문.

1.2.4 재귀적 학습 루프 (Recursive Loop)

- 입력: 최종 논문.
- 처리: 지식 추출, 벡터 DB 업데이트, 실패 사례 아카이빙.
- 출력: 향상된 지식 베이스 및 새로운 가설 시드(Seed).

표 1: 기존 자동화 시스템 vs. 제안된 ARI 시스템 비교

특징	기존 자동화 (Lab Automation)	초기 AI 과학자 (The AI Scientist v1)	제안된 ARI 시스템 (v2+ 확장형)
주요 대상	물리적 반복 작업 (피펫팅 등)	소프트웨어/알고리즘 연구	전 도메인 (물리/화학/바이오 포함)
실험 설계	인간이 코딩한 스크립트 실행	템플릿 코드 수정 (선형적)	에이전트 트리 검색 (비선형적)
지식 습득	없음 (입력된 데이터만 처리)	제한적 키워드 검색	도메인 특화 딥 웹 크롤링 & RAG
평가 방식	결과값 리포팅	단순 점수 평가	다중 페르소나 리뷰 & 반박/수정
연속성	단발성 실행	단일 논문 생성 후 종료	재귀적 가설 생성 및 지식 확장

섹션 **2:** 지식 엔진 **-** 도메인 특화 크롤링과 데이터 파이프라인

사용자가 요청한 "도메인 별 클롤링 후 각 도메인 별 벡터 DB 확장"을 구현하기 위해서는 고도화된 데이터 파이프라인이 필수적이다.

2.1 FAIRBridge 기반의 지능형 크롤링

과학 데이터는 표준화되어 있지 않다. 어떤 저널은 PDF를 제공하고, 어떤 데이터베이스는 CSV를 제공하며, 어떤 곳은 복잡한 웹 인터페이스를 통해서만 데이터를 보여준다. **FAIRBridge** 아키텍처는 이러한 이질적인 소스를 통합하는 데 최적화되어 있다.[4]

2.1.1 웹 에이전트의 작동 원리

크롤러는 단순한 스크립트가 아니라, LLM 두뇌를 가진 에이전트이다.

1. 목표 인식**:** "최신 머신러닝 최적화 기법에 관한 논문을 수집하라"는 명령을 받는다.
2. 소스 식별**:** 사전 정의된 도메인 맵을 통해 적절한 소스(예: arXiv, NeurIPS Proceedings, Papers With Code)를 선정한다.
3. 동적 탐색**:** 웹 페이지에 접속하여 Next Page, Download PDF, Export BibTeX 등의 요소를 시각적/구조적으로 인식하고 상호작용한다. 캡차(CAPTCHA)나 로그인 창이 뜨면 이를 감지하고 해결(가능한 경우)하거나 인간에게 알림을 보낸다.
4. 속도 조절**:** 서버에 과부하를 주지 않기 위해 robots.txt를 준수하고 요청 간격을 조절하는 '예의 바른(Polite)' 크롤링 정책을 수행한다.

2.2 멀티 도메인 벡터 데이터베이스 전략

수집된 데이터는 하나의 거대한 DB에 넣는 것이 아니라, 도메인별로 특화된 벡터 공간(Vector Space)에 저장되어야 한다.

2.2.1 도메인별 임베딩 모델 분리

- **Computer Science (CS):** 소스 코드와 수식에 대한 이해가 중요하다. 따라서 코드 데이터로 학습된 모델이나 Specter와 같은 인용 중심 모델을 사용한다.
- **Bio-Medical:** 유전자 서열, 화학식, 질병명 등 고유한 엔티티가 많다. PubMedBERT나 BioLinkBERT를 사용하여 이러한 엔티티 간의 관계를 벡터 공간에 보존한다.
- **Physics/Math:** LaTeX 수식의 의미를 보존해야 한다. 수식 토큰화에 특화된 모델을 적용한다.

2.2.2 벡터 **DB** 샤딩**(Sharding)** 및 확장

데이터의 양이 방대하므로(수천만 건의 논문), 벡터 DB는 샤딩(Sharding)되어 관리된다.

- 동적 확장**:** 새로운 논문이 크롤링되면 실시간으로 임베딩되어 해당 도메인의

샤드(Shard)에 추가된다. HNSW 인덱스는 실시간 업데이트가 가능하도록 구성하여, 크롤링 즉시 검색(Retrieval)이 가능하게 한다.[28]

2.3 지식 정제: 파싱과 청킹

- **PDF** 파싱의 난제 해결: PDF는 시각적 레이아웃 중심의 포맷이다. 이를 텍스트로 변환할 때 단락 순서가 섞이거나 표(Table)가 깨지는 문제가 발생한다. ARI는 **Nougat** (OCR 기반) 또는 **Grobid** (구조 인식 기반)를 사용하여 이를 해결한다. 특히 표 데이터는 Markdown이나 HTML 표로 변환하여 LLM이 구조를 이해할 수 있도록 한다.
- 인용**(Citation)** 분리: 본문 내의 ``와 같은 인용 부호를 실제 참고문헌 리스트의 메타데이터와 연결(Link)한다. 이는 나중에 RAG 수행 시 "누가 말했는가"를 정확히 추적하는 데 사용된다.

섹션 3: 가설 생성 및 실험 설계 (Ideation & Planning)

충분한 지식이 확보되었다면, 이제는 새로운 질문을 던질 차례이다. 사용자의 요구인 "새로운 가설을 창조"하고 "실험 플랜을 계획"하는 단계이다.

3.1 문헌 기반 발견(LBD)을 통한 가설 생성

단순히 "새로운 아이디어를 내봐"라고 LLM에게 묻는 것은 환각을 유발하거나 뻔한 답을 낼 가능성이 높다. ARI는 **LBD(Literature-Based Discovery)** 방법론을 사용하여 논리적 근거가 있는 가설을 생성한다.[10]

3.1.1 지식 그래프(Knowledge Graph) 추론

벡터 DB에서 추출된 엔티티와 관계를 바탕으로 지식 그래프를 구성한다.

- 연결 예측**(Link Prediction):** 그래프 상에서 직접 연결되지 않은 두 노드(예: 'Drug A'와 'Disease B') 사이에 잠재적인 연결 가능성을 계산한다.
- 유추**(Analogy)** 기반 생성: "문제 P1에 알고리즘 A1이 효과적이었다. 문제 P2는 P1과 구조적으로 유사하다(임베딩 거리가 가깝다). 따라서 P2에도 A1(또는 그 변형)이 효과적일 것이다."

3.2 신규성(Novelty) 검증 필터

생성된 가설이 이미 존재하는지 확인하기 위해 **Semantic Scholar** API를 활용한 엄격한 검증을 거친다.[13]

1. 아이디어 벡터화: 가설을 임베딩 벡터로 변환한다.
2. 유사도 검색: 벡터 DB 및 외부 API에서 가장 유사한 논문들을 검색한다.

3. 차별점 분석**:** LLM에게 "제안된 가설"과 "검색된 유사 논문"을 주고, 두 연구의 차이점을 분석하게 한다.
    - 차이점 없음*:* 기각(Reject).
    - 미미한 차이*:* 보류 또는 수정 제안.
    - 명확한 차이*(*새로운 방법, 새로운 데이터, 새로운 적용 분야*):* 채택(Accept).

3.3 실험 플랜의 구체화

채택된 가설은 실행 가능한 계획으로 변환된다.

- 실험 템플릿**:** 실험의 종류(머신러닝 모델 학습, 데이터 분석, 시뮬레이션 등)에 따라 적절한 코드 템플릿을 로드한다.
- 단계별 계획**(Step-by-Step Plan):**
    1. 데이터 준비 (전처리, 증강).
    2. 베이스라인 모델 구현.
    3. 제안 모델(가설) 구현.
    4. 비교 실험 및 지표 측정.
    5. 결과 시각화.

섹션 **4:** 에이전트 트리 검색을 통한 자율 실험 **(Autonomous Experimentation)**

이 단계는 ARI의 핵심 엔진이다. "The AI Scientist v2"에서 도입된 **에이전트 트리 검색(Agentic Tree Search)**은 실험의 불확실성을 관리하는 강력한 도구이다.[15]

4.1 선형적 실행의 한계와 트리 검색의 도입

과거의 자동화 스크립트는 일직선으로 실행되었다. 중간에 에러가 나면 멈췄다. 그러나 실제 연구는 시행착오의 연속이다. 하이퍼파라미터를 바꿔보기도 하고, 데이터 전처리를 다르게 해보기도 한다.

- 트리 구조**:** 실험의 각 결정(Decision)을 노드로, 그에 따른 실행을 엣지로 표현한다.
- 탐색 전략**:** 너비 우선 탐색(BFS)이나 깊이 우선 탐색(DFS)을 사용하여 가능한 실험 경로를 탐색한다. LLM은 각 노드의 상태(성공, 실패, 유망함)를 평가하여 어떤 가지(Branch)를 더 뻗어나갈지 결정한다.

4.2 코딩 에이전트**(Aider)**의 역할

실제 코드는 **Aider**와 같은 코딩 특화 에이전트가 작성한다.[1]

- 문맥 인식**:** 프로젝트의 전체 파일 구조를 이해하고, 필요한 파일만 수정한다.

- 테스트 주도 개발**(TDD)** 유사 방식**:** 먼저 실행 코드를 짜고, 실행해본 뒤, 에러 로그(stderr)를 보고 코드를 수정한다.
- 라이브러리 및 환경 관리**:** ImportError가 발생하면 스스로 requirements.txt를 수정하고 패키지를 설치한다.

4.3 진화적 최적화 **(ShinkaEvolve)**

단순한 버그 수정을 넘어, 성능을 극대화해야 할 때(예: 정확도 1% 높이기)는 **ShinkaEvolve** 알고리즘이 작동한다.[18]

- 여러 개의 코드 변형(Variant)을 생성하여 병렬로 실행한다.
- 가장 성능이 좋은 코드를 부모로 삼아, LLM이 이를 변형(Mutation)하거나 교차(Crossover)하여 더 나은 코드를 생성한다. 이는 유전 알고리즘을 코드 생성에 적용한 것으로, 인간이 생각하지 못한 창의적인 최적화 코드를 찾아낼 수 있다.

섹션 **5:** 논문 작성 및 동료 평가 **(Writing & Review)**

실험이 성공적으로 완료되면, 그 결과는 학계의 표준 형식인 논문으로 정리되어야 한다. 사용자가 요구한 "실제 논문 작성 플로우"와 "각기 다른 리뷰어" 시스템이 여기서 구현된다.

5.1 **LaTeX** 기반의 전문 논문 작성

ARI는 워드 프로세서가 아닌 **LaTeX**를 사용하여 논문을 작성한다. 이는 수식, 표, 인용 관리에 있어 학계 표준이기 때문이다.

- **RAG** 기반 저술**:** 논문의 각 섹션을 쓸 때마다 벡터 DB를 참조한다. '서론'을 쓸 때는 연구 배경 지식을 검색하고, '결과'를 쓸 때는 실험 로그 데이터를 참조한다.
- 인용 무결성**: PaperQA** 기술을 적용하여, 에이전트가 기억에 의존해 인용을 지어내지 않도록 한다. 반드시 DB에 존재하는 논문의 BibTeX 키만을 사용하도록 강제한다.[21]

5.2 시각적 비평**(Visual Critique)**

VLM(예: GPT-4V)이 생성된 도표(Plot) 이미지를 검사한다.

- 가독성 체크**:** 글자 크기, 색상 구분, 겹침 현상 등을 확인한다.
- 정보 전달력**:** 도표가 캡션의 설명과 일치하는지, 핵심 결과를 잘 보여주는지 평가하고, 필요하면 플로팅 코드를 수정하도록 지시한다.[3]

5.3 다중 페르소나 리뷰 시스템

논문 초안이 완성되면, 자동화된 동료 평가**(Automated Peer Review)** 단계로 넘어간다.

- 리뷰어 구성**:** 시스템은 3~4명의 가상 리뷰어를 생성한다. 각 리뷰어는 서로 다른 배경과 성향을 가진다.
  - *Reviewer 1 (The Skeptic):* 방법론의 약점을 집요하게 파고든다.
  - *Reviewer 2 (The Benchmark Nazi):* 비교 실험이 충분한지, SOTA 모델과 비교했는지 확인한다.
  - *Reviewer 3 (The Domain Expert):* 해당 분야의 기존 연구 흐름과 잘 맞는지를 본다.
- 리뷰 가이드라인**:** NeurIPS나 ICLR 같은 실제 학회의 리뷰 가이드라인을 프롬프트로 입력받아, 정량적 점수와 정성적 피드백을 산출한다.[23]

5.4 반박 및 수정**(Rebuttal & Revision)** 루프

단순히 평가만 받고 끝나는 것이 아니다. 저자 에이전트는 리뷰를 읽고 **반박(Rebuttal)**하거나 **논문을 수정(Revision)**해야 한다.[1]

- 실험 재수행의 판단**:** 만약 리뷰어가 "추가 실험이 필요하다"고 지적하면, 저자 에이전트는 실험 단계로 되돌아가 추가 실험을 수행할지, 아니면 논리적으로 방어할지 결정한다. 이 '루프백(Loop-back)' 기능이 ARI의 고도화된 자율성을 보여주는 특징이다.

섹션 **6:** 재귀적 발견과 미래 **(Recursion & Future)**

ARI의 진정한 가치는 이 모든 과정이 끝나고 나서 시작된다.

6.1 지식의 내재화와 선순환

승인된 논문은 새로운 지식으로 시스템에 흡수된다.

- 벡터 **DB** 업데이트**:** 생성된 논문은 다시 파싱되어 자신의 지식 베이스에 추가된다.
- **SOTA** 갱신**:** 만약 실험 결과가 기존 최고 성능(SOTA)을 경신했다면, 시스템 내부의 벤치마크 기준이 상향 조정된다.

6.2 새로운 가설의 창조 **(Recursive Discovery)**

사용자가 요구한 "스스로 논문을 분석해 새로운 가설을 창조"하는 기능이다.[26]

- **Future Work** 분석**:** 생성된 논문의 '결론'과 '향후 연구' 섹션은 다음 연구의 출발점이 된다.
- 한계점 돌파**:** 자신의 논문에서 스스로 밝힌 한계점(Limitation)을 극복하기 위한 새로운 가설을 자동으로 생성하고, 이를 큐(Queue)에 등록하여 다음 연구 사이클을 시작한다. 이는 24시간 멈추지 않는 '영구적인 연구 기계'의 가능성을 시사한다.

결론

본 보고서는 사용자의 요구사항을 충실히 반영하여, 가설 생성부터 실험, 논문 작성, 리뷰, 그리고 재귀적 발전에 이르는 완전 자율 연구 에이전트의 아키텍처를 상세히 기술하였다. **FAIRBridge** 방식의 딥 웹 크롤링과 도메인별 벡터 DB 확장을 통해 지식의 깊이를 더하고, **Agentic Tree Search**와 **ShinkaEvolve**를 통해 실험의 성공률과 창의성을 높이며, 다중 페르소나 리뷰 시스템을 통해 연구 결과의 품질을 보증한다. 이러한 시스템은 단순한 자동화를 넘어, 과학적 발견의 새로운 지평을 여는 **Artificial Research Intelligence (ARI)**의 청사진을 제시한다.

참고 자료

1. Evaluating Sakana's AI Scientist for Autonomous Research: Wishful Thinking or an Emerging Reality Towards 'Artificial Research Intelligence' (ARI)? - arXiv, 1월 9, 2026에 액세스, https://arxiv.org/html/2502.14297v3
2. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery - Sakana AI, 1월 9, 2026에 액세스, https://sakana.ai/ai-scientist/
3. [2504.08066] The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search - arXiv, 1월 9, 2026에 액세스, https://arxiv.org/abs/2504.08066?
4. A FAIR Resource Recommender System for Smart Open Scientific Inquiries - MDPI, 1월 9, 2026에 액세스, https://www.mdpi.com/2076-3417/15/15/8334
5. A GenAI System for Improved FAIR Independent Biological Database Integration - arXiv, 1월 9, 2026에 액세스, https://arxiv.org/html/2506.17934v1
6. LANGUAGE AGENTS ACHIEVE SUPERHUMAN SYNTHESIS OF, 1월 9, 2026에 액세스, https://ctstate.edu/images/Forms-Documents/AI-presidential-fellows/Language-Agents-Science.pdf
7. RAG-based Search Engine with an Autonomous Web Crawler - Theseus, 1월 9, 2026에 액세스, https://www.theseus.fi/bitstream/10024/904061/2/Hemmi_Pauli.pdf
8. Agentic Search Engine for Real-Time Internet of Things Data - PMC - NIH, 1월 9, 2026에 액세스, https://pmc.ncbi.nlm.nih.gov/articles/PMC12526645/
9. Future-House/paper-qa: High accuracy RAG for answering questions from scientific documents with citations - GitHub, 1월 9, 2026에 액세스, https://github.com/Future-House/paper-qa
10. Recent Advances and Future Directions in Literature-Based Discovery - arXiv, 1월 9, 2026에 액세스, https://arxiv.org/html/2506.12385v1
11. Looking for Hidden Gems in Scientific Literature - Elicit, 1월 9, 2026에 액세스, https://elicit.com/blog/literature-based-discovery
12. The AI Scientist: How done it - Wandb, 1월 9, 2026에 액세스, https://wandb.ai/wandb-japan/ai-scientists/reports/The-AI-Scientist-How-done-it--VmlldzoxMTE0ODAzMQ

13. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search - GitHub, 1월 9, 2026에 액세스, https://github.com/SakanaAI/AI-Scientist-v2
14. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search, 1월 9, 2026에 액세스, https://iclr.cc/virtual/2025/37267
15. Sakana unveils AI Scientist-v2 | ml-news – Weights & Biases - Wandb, 1월 9, 2026에 액세스, https://wandb.ai/byyoung3/ml-news/reports/Sakana-unveils-AI-Scientist-v2--VmlldzoxMjQxNTUzMw
16. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search - Sakana AI, 1월 9, 2026에 액세스, https://pub.sakana.ai/ai-scientist-v2/paper/paper.pdf
17. ShinkaEvolve: Evolving New Algorithms with LLMs, Orders of Magnitude More Efficiently, 1월 9, 2026에 액세스, https://sakana.ai/shinka-evolve/
18. ShinkaEvolve: Towards Open-Ended And Sample-Efficient Program Evolution - arXiv, 1월 9, 2026에 액세스, https://arxiv.org/html/2509.19349v1
19. AI-Assisted Tools for Scientific Review Writing: Opportunities and Cautions | ACS Applied Materials & Interfaces - ACS Publications, 1월 9, 2026에 액세스, https://pubs.acs.org/doi/10.1021/acsami.5c08837
20. Jr. AI Scientist and Its Risk Report: Autonomous Scientific Exploration from a Baseline Paper, 1월 9, 2026에 액세스, https://arxiv.org/html/2511.04583v2
21. PaperQA - paper-qa · PyPI, 1월 9, 2026에 액세스, https://pypi.org/project/paper-qa/4.2.0/
22. OpenReviewer: A Specialized Large Language Model for Generating Critical Scientific Paper Reviews - arXiv, 1월 9, 2026에 액세스, https://arxiv.org/html/2412.11948v3
23. TreeReview: A Dynamic Tree of Questions Framework for Deep and Efficient LLM-based Scientific Peer Review - ACL Anthology, 1월 9, 2026에 액세스, https://aclanthology.org/2025.emnlp-main.790.pdf
24. aiXiv: A Next-Generation Open Access Ecosystem for Scientific Discovery Generated by AI Scientists - arXiv, 1월 9, 2026에 액세스, https://arxiv.org/html/2508.15126v2
25. The AI Scientist Due Dillidence Report - Intor with AI, 1월 9, 2026에 액세스, https://www.intor.ai/ai-analysis/the-ai-scientist
26. [R] The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery - Reddit, 1월 9, 2026에 액세스, https://www.reddit.com/r/MachineLearning/comments/1eqwfo0/r_the_ai_scientist_towards_fully_automated/
27. Full article: Autonomous GIS: the next-generation AI-powered GIS - Taylor & Francis Online, 1월 9, 2026에 액세스, https://www.tandfonline.com/doi/full/10.1080/17538947.2023.2278895