

디지털 연구콘텐츠 객체 인벤토리 분석: arXiv 논문과 GitHub 저장소를 중심으로

강 근 영*

연세대 학교 대학도서관발전연구소 연구원

Inventory Analysis of Digital Research-Content Objects in arXiv Papers and GitHub Repositories

Keun Young Kang*

Researcher, Research Institute for Academic Library Development, Yonsei University, Seoul 03722, Korea

[요 약]

인공지능의 발전과 더불어 연구 결과물은 코드 저장소, 실행환경, 모델, 데이터셋 등 다양한 디지털 객체로 확장되고 있다. 본 연구는 디지털 연구콘텐츠 객체의 구성과 메타데이터 표현 가능성을 살펴보기 위해 2025년 1월~2026년 6월 arXiv 제출 논문 중 공식 GitHub 저장소가 검증된 120편을 대상으로 인벤토리를 구축하였다. 분석 결과 PDF, 공식 저장소, README는 모든 논문에서 확인되었으나, 문서화 메타데이터, 라이선스 파일, 모델 카드, 데이터셋 카드, 프로젝트 페이지 등 보조 객체의 보유 양상은 유형에 따라 차이를 보였다. 즉, 저장소 URL만으로는 논문 주변 콘텐츠를 충분히 설명할 수 없음을 보여준다. 이에 객체 유형, 출처, 접근 상태, 버전 정보를 함께 기술하는 관계지향 메타데이터가 필요하며, 표본 범위를 확장한 후속 검증이 요구된다.

[Abstract]

Research on artificial intelligence produces increasingly diverse digital objects, including code repositories, execution environments, models, and datasets. This study investigated the composition and metadata representativeness of digital research-content objects in 120 arXiv papers submitted between January 2025 and June 2026 with verified official GitHub repositories. PDFs, official repositories, and README files were found for all papers, whereas the presence of supporting objects, including documentation metadata, license files, model cards, dataset cards, and project pages, varied by object type. These findings indicate that repository URLs alone do not sufficiently describe the digital content surrounding research papers. The study therefore highlights the need for relation-oriented metadata describing object type, source, access status, and version information, though broader samples are needed to further validate these results.

색인어 : 디지털 연구콘텐츠 객체, 관계지향 메타데이터, 콘텐츠 인벤토리, 스키마 상호운용성, 디지털 연구 인프라

Keyword : Digital Research-Content Objects, Relation-Aware Metadata, Content Inventory, Schema Interoperability, Digital Research Infrastructure

<http://dx.doi.org/10.9728/dcs.2026.27.9.2439>



This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Received 28 July 2026; **Revised** 14 August 2026

Accepted 31 August 2026

***Corresponding Author; Keun Young Kang**

Tel: +82-2-2123-2405

E-mail: ky.kang@yonsei.ac.kr

1. 서 론

전통적인 학술 커뮤니케이션 체계에서 논문은 연구의 방법과 결과를 완결된 형태로 담는 최종 산출물로 기능해 왔다. 그러나 실증 연구가 소프트웨어와 데이터에 의존하는 정도가 커지면서, 논문이 유일한 커뮤니케이션 단위라는 전제 자체를 재검토해야 한다는 문제의식이 오래전부터 제기되어 왔다[1]. 데이터셋, 시뮬레이션, 소프트웨어, 동적 지식표현까지도 논문과 대등한 독립적 커뮤니케이션 단위로 인정하고, 이들을 재귀적으로 결합할 수 있는 복합 객체 구조로 학술 커뮤니케이션 체계를 재설계할 것을 제안한 연구도 있다[1]. 이러한 문제의식은 이후 저장소 간 상호운용성 실험으로 이어졌으며, 하나의 학술 산출물을 데이터스트림이나 문서, 계보(lineage) 정보 등 이질적 구성요소의 그래프로 표현하고 여러 다른 기종의 저장소에 걸쳐 재사용하는 것이 기술적으로 가능함을 보인 연구도 있다[2]. 즉, 연구성과가 논문 하나로 환원되지 않고 여러 이질적 객체로 분산된다는 인식 자체는 특정 분야에 국한된 새로운 발견이 아닌, 이미 반복적으로 확인되어 온 현상이다.

인공지능(AI) 분야는 논문과 함께 코드, 데이터, 모델 등 다양한 디지털 연구 산출물을 공개하는 양상이 확산되고 있다. 특히 기계학습 연구에서는 GitHub와 같은 공개 저장소가 논문과 연결된 기술적 산출물을 유통하는 주요 경로로 활용되고 있으며, 저장소의 존재와 활용은 연구의 확산과도 관련되는 것으로 보고되었다[3]. 그러나 연구 산출물의 공개 여부와 실제 재사용 가능성은 동일한 문제로 볼 수 없다. AI 연구를 대상으로 한 메타연구에서도 코드와 데이터의 공개율뿐 아니라 공개된 코드의 재현 가능성에 상당한 차이가 나타난 바 있다[4]. 따라서 AI 연구에서 하나의 논문은 공식 코드 저장소, 여러 개의 하위 문서, 모델 카드, 데이터셋 카드 등 이질적인 객체들과 연결될 수 있으며, 이 객체들은 서로 다른 기능과 접근 조건을 가진다. 논문이 유일한 참조 단위였던 시기에 형성된 서지 정보 관리, 인용 체계, 발견 시스템은 주로 논문과 논문 사이의 관계를 다루는 데 맞추어져 있어 이러한 다층적 구조를 충분히 반영하기 어려운 측면이 있다. 실무에서는 흔히 저장소 링크 하나를 추가하는 방식으로 다루어 왔는데, 이는 연구성과의 재현, 재사용, 인용에 필요한 객체의 구성을 충분히 나타내지 못할 수 있다.

GitHub와 같은 코드 호스팅 플랫폼의 학술적 활용 역시 이러한 흐름 속에서 빠르게 확대되어 왔다. Escamilla 외[5]는 arXiv와 PubMed Central 출판물에서 코드 호스팅 플랫폼을 인용한 사례가 2007년 160건에서 2021년 76,746건으로 증가했음을 보고한 바 있다. 다만 이 증가 추세에 대한 논의는 주로 인용 빈도라는 단일 지표에 초점을 두어 왔으며, 그 저장소 안에 실제로 어떤 종류의 객체가 존재하고 그것들이 어떤 조건에서 접근 가능한지는 상대적으로 덜 다루어졌다. 한편 Heil 외[6]는 기계학습 연구의 재현성을 위해 데이터, 모

델, 코드와 실행환경의 명시적 제공이 중요함을 제시하였고, Wattanakriengkrai 외[7]는 GitHub 저장소와 학술논문 간 추적 가능성과 연결 관계를 분석하였다. 그러나 이들 연구는 논문과 연결된 저장소를 구성하는 README, 실행환경 파일, 모델 카드, 데이터셋 카드, 라이선스, 인용 파일, 이슈 채널 등의 객체를 하나의 통합된 인벤토리로 분석하지는 않았다. 즉, 저장소의 존재 여부나 특정 차원의 품질을 측정하는 연구는 존재하지만, 논문을 중심으로 이 모든 객체가 어떤 유형적 구성과 분포로 나타나는지를 체계적으로 목록화한 AI 분야 연구는 상대적으로 드물다.

객체 수준의 구분은 기존 연구에서 다루지 않은 항목을 추가로 목록화한다는 데 의미가 있는 것은 아니다. 저장소의 존재 여부만으로는 실행환경, 문서화, 모델·데이터셋 설명, 라이선스, 인용정보처럼 서로 다른 기능을 수행하는 객체가 실제로 제공되는지를 판단할 수 없기 때문이다. 따라서 논문과 저장소의 연결을 하나의 이진적 관계로 다루는 것과 그 연결을 구성하는 객체를 구분하여 파악하는 것은 서로 다른 수준의 정보이다. 객체 유형별 구성과 접근 상태를 확인하는 것은 저장소 연결 여부와 실제 연구콘텐츠의 구성 사이의 차이를 구체화하고, 향후 객체 수준의 기술과 발견을 논의하기 위한 기초 자료를 제공한다.

이 점에서 메타데이터 설계와 콘텐츠 발견 시스템 구축의 관점에서 살펴볼 만한 함의를 지닌다. 논문 레코드에 ‘코드 있음’이라는 단일 속성만 추가하는 방식은, 실행 가능성을 보여주는 환경 파일, 적용 범위와 한계를 설명하는 모델·데이터셋 카드, 재사용 조건을 명시하는 라이선스와 인용 파일, 유지보수 상태를 보여주는 이슈 채널처럼 서로 다른 성격의 객체를 하나의 값으로 함께 다루게 된다. 이러한 접근은 몇 가지 층위에서 한계를 가질 수 있다. 첫째, 콘텐츠 발견의 층위에서, 이용자는 저장소 링크만으로는 실행환경이 문서화되어 있는지, 모델의 적용 범위가 기술되어 있는지를 사전에 판단하기 어렵다. 둘째, 메타데이터 상호운용성의 층위에서, DataCite, Schema.org, CodeMeta, RO-Crate와 같은 기존 스키마가 논문·저장소·데이터셋 같은 핵심 객체는 비교적 직접적으로 표현할 수 있지만, README나 이슈 채널처럼 맥락적 성격이 강한 객체는 관계 어휘와 공식성 검증 규칙이 갖추어지지 않으면 일관되게 표현되기 어렵다. 셋째, 장기적 접근성의 층위에서, 객체별로 상이한 접근 상태(접근 가능, 메타데이터만 확인, 접근 불가)를 구분하여 기록하지 않으면, 시간이 지남에 따라 어떤 객체가 유실되었는지 파악하기 어려워질 수 있다. 이러한 점들은 여러 플랫폼에 분산된 객체를 연결하는 과정에서 객체 유형, 논문과의 관계, 공식성의 근거, 접근 가능 여부, 갱신·버전 정보를 함께 다루는 접근이 유용할 수 있음을 시사한다.

이에 본 연구는 공식 GitHub 저장소가 검증된 arXiv 내 논문 120편을 대상으로 논문 주변의 공개 디지털 객체를 객체 수준에서 조사한다. 연구의 목적은 저장소 링크가 가리키는 연구콘텐츠 생태계가 실제로 어떤 객체들로 구성되어 있으며,

이 구성이 메타데이터 설계에 어떠한 함의를 갖는지를 실증적으로 살펴보는 데 있다. 이를 위해 다음의 연구문제를 설정하였다.

RQ1. 공식 GitHub 저장소와 연결된 arXiv 논문에서는 어떤 유형의 디지털 연구콘텐츠 객체가 나타나며, 그 보유 비율은 어떠한가?

RQ2. 이 논문들의 공식 GitHub 저장소는 어떠한 문서화 및 지원 특성을 보이는가?

RQ3. 관찰된 디지털 연구콘텐츠 객체는 DataCite, Schema.org, CodeMeta, RO-Crate에서 어떻게 표현될 수 있는가?

이를 통한 본 연구의 기여는 다음과 같다. 첫째, 공식 저장소가 검증된 논문 표본에 대해 객체 수준의 기술적 인벤토리를 제시함으로써, 저장소 연결 여부와 실제 콘텐츠 구성을 구분하여 살펴볼 수 있는 실증 자료를 제공한다. 둘째, DataCite, Schema.org, CodeMeta, RO-Crate의 공식 어휘와 구조를 기준으로 관찰된 객체 유형의 표현 가능성을 검토하고, 추가적인 프로파일링이 필요한 지점을 논의함으로써 객체 수준 메타데이터 기술을 위한 기초 자료를 제공한다. 세부 분야별 비교는 객체 분포의 차이 가능성을 점검하기 위한 탐색적 분석으로 수행하였다. 본 연구는 현행 도서관 목록, 기관 리포지터리, 검색 서비스의 성능을 평가하지 않으며, 분석 범위는 공식 GitHub 연계 arXiv AI 논문으로 한정한다.

II. 이론적 배경

2-1 학술 커뮤니케이션 단위의 다변화

학술 커뮤니케이션 체계에서 논문은 오랫동안 연구성과를 유통시키는 기본 단위로 기능해 왔다. 인쇄 매체를 전제로 형성된 이 체계에서는 하나의 완결된 서사 안에 연구 문제, 방법, 결과, 해석을 담아내는 것이 연구성과를 전달하는 통로였다. 그러나 실증 연구가 계산 자원과 대규모 데이터에 점점 더 의존하게 되면서, 논문이라는 서사적 텍스트만으로는 그 연구가 실제로 수행된 과정, 즉, 어떤 데이터를 어떤 절차로 처리했으며, 어떤 코드로 결과를 산출했는가를 충분히 재구성할 수 없다는 문제가 제기되기 시작했다[8],[9]. 이러한 문제의식은 2000년대 초반 디지털 도서관 및 학술 정보 인프라 연구에서 구체적인 형태를 갖추었다. Van de Sompel 외[1]는 기존 학술 커뮤니케이션 체계를 등록, 인증, 인지, 보존, 보상이라는 다섯 가지 기능으로 분해하여 분석하고, 이 체계가 데이터셋, 소프트웨어, 동적 지식표현까지 포괄하도록 재설계되어야 하며, 이러한 복합 산출물은 논문과 마찬가지로 재귀적으로 결합 가능한 커뮤니케이션 단위로 인정되어야 한다고

주장하였다.

이는 이후 실제 저장소 간 상호운용성을 검증하는 연구로 이어졌다. Warner 외[2]는 서로 다른 아키텍처를 가진 네 개의 저장소 시스템(Fedora, aDORe, DSpace, arXiv)에 걸쳐 학술 산출물을 유통시키는 실험을 수행하였다. 이들은 하나의 학술 산출물을 데이터스트림, 문서, 계보 정보 등 이질적 구성요소로 이루어진 그래프 구조로 표현하고, 이 구조화된 표현을 저장소들 사이에서 상호 교환할 수 있음을 실증하였다. 즉, 복합 산출물을 다루기 위해 반드시 모든 콘텐츠를 하나의 플랫폼에 물리적으로 통합할 필요는 없으며, 각 구성요소의 출처와 관계를 명시하는 구조화된 기술 방식만으로도 상호운용이 가능하다는 점이다. 이는 본 연구가 취하는 입장과도 일치한다. arXiv 논문과 GitHub 저장소, 그리고 그 저장소 내부의 여러 객체들은 물리적으로 하나의 시스템에 속하지 않지만, 이들 사이의 관계가 적절히 기술된다면 하나의 연구성과를 구성하는 요소로 취급될 수 있다.

이러한 문제의식은 특정 플랫폼이나 기술 프로젝트에 국한되지 않고, 향상된 출판물(enhanced publication), 복합 디지털 객체, 연구객체라는 세 갈래의 개념적 흐름으로 정착되었다[10]~[13]. 향상된 출판물은 서사적 논문과 데이터, 이미지, 워크플로, 소프트웨어 등 관련 부분의 관계를 함께 기술하는 출판 모델로 정리되어 왔다[14],[15]. 이 모델의 핵심은 논문 텍스트와 그 근거 자료를 서로 분리된 산출물로 다루지 않고, 하나의 출판 단위 안에서 그 관계를 명시적으로 선언한다는 데 있다. 한편 연구객체 개념은 재사용 가능성이라는 문제에 좀 더 직접적으로 초점을 맞춘다. 연구객체 논의에서 디지털 객체의 재사용은 다운로드나 접근을 넘어, 검증·변형·재맥락화까지 포괄하는 행위로 다루어진다[16]. 재사용 가능성 자체는 객체 간 관계와 실행·생성 맥락을 얼마나 충실히 기록하는지에 좌우된다는 논의도 제기되었다[17]. 이 지점은 본 연구가 다루는 문제와 직접 맞닿아 있다. 논문 주변에 존재하는 객체들을 단순히 열거하는 작업과, 그 객체들 사이의 관계 및 공식성 근거를 함께 기록하는 작업은 서로 다른 수준의 작업이며, 후자가 실질적인 재사용 가능성을 좌우한다.

FAIR 원칙은 이러한 논의를 데이터 관리의 실무 지침으로 구체화한 사례로 볼 수 있다. FAIR는 연구 산출물의 검색·접근·상호운용과 재사용을 위해 식별자, 메타데이터, 관계 정보의 중요성을 강조한다[18]. 여기서 강조되는 지점은 산출물 간의 관계가 인간뿐 아니라 기계도 해석 가능한 형태로 명시되어야 한다는 점이다. 이에 따라 본 연구가 다루는 논문 주변의 디지털 객체들은 서로 어떻게 연결되어 있으며 그 연결의 근거가 무엇인지가 기록되어야 할 대상이다. 이상의 논의를 종합하여 본 연구는 ‘디지털 연구콘텐츠 객체’를 논문과 공식 링크, 저장소 메타데이터 또는 수집 근거로 연결된 공개 객체로 정의한다. 구체적으로 논문 PDF와 공식 저장소처럼 핵심적인 객체가 있는 반면, README, 실행환경 파일, 예제, 라이선스, 인용 파일, 이슈 채널은 이용·재현·지원의 맥락을 제공하는 보조 객체로 구분된다. 따라서 본 연구의 객체 인벤토리

토리는 연구성과의 발견과 재사용에 관여할 수 있는 연결 대상을 식별하는 기술적 분류이다.

2-2 AI 연구에서의 저장소 기반 콘텐츠 생산과 재사용성

앞 절에서 다룬 다객체적 성격은 AI 관련 연구에서 두드러지게 나타난다. AI 관련 논문과 연계된 GitHub 저장소에는 소스코드뿐만 아니라 README, 요구사항 및 환경 파일, 노트북과 예제, 문서 링크, 릴리스, 라이선스, 인용 파일, 이슈 채널이 함께 포함될 수 있다. 이 객체들은 저장소 안에서 각각 설치, 실행, 검증, 인용, 유지보수라는 서로 다른 기능적 역할을 수행하도록 의도된다. 그러나 이러한 저장소가 실제로 이 기능들을 충족시키는 정도는 균질하지 않다는 점이 선행연구를 통해 반복적으로 확인된다. 기계학습 연구의 재현성을 위해서는 논문 자체의 방법 기술뿐 아니라 데이터, 모델, 소스코드와 실행 환경을 함께 제공하고 명확히 기술하는 것이 중요하다. Heil 외[6]는 기계학습 연구의 계산적 재현성을 높이기 위한 기준으로 데이터, 모델, 코드의 공개와 실행 워크플로의 명시를 제시하였다. 이는 재현에 필요한 데이터, 모델, 코드와 실행 맥락이 어떻게 제공되는지를 함께 살펴볼 필요가 있음을 보여준다.

Wattanakriengkrai 외[7]는 학술 논문을 참조하는 GitHub 저장소를 분석하여 논문과 저장소 사이의 연결 및 추적 가능성이 균일하지 않음을 보고하였다. 논문과 저장소 사이에 연결이 존재하는지 여부와 그 저장소가 실제로 어떤 콘텐츠를 제공하는지를 구분하여 살펴볼 필요가 있음을 보여준다. 이 결과는 논문과 저장소 사이에 연결이 존재하는지 여부와 그 저장소가 실제로 어떤 콘텐츠를 제공하는지를 구분하여 살펴볼 필요가 있음을 보여준다. 연결의 존재만으로는 저장소 내부의 문서화, 실행환경, 라이선스, 인용정보 등 구체적인 콘텐츠 구성을 판단할 수 없으며, 본 연구는 이에 저장소 내부 객체의 유형별 분포를 조사 단위로 삼고자 한다.

Kang 외[3]는 Papers with Code에 수록된 기계학습 연구와 GitHub 저장소를 연결하여, 코드 저장소와 같은 기술적 연구 산출물이 논문의 확산과 관련될 수 있음을 실증하였다. 그러나 저장소의 존재나 연결 자체는 저장소 내부의 문서화, 실행환경, 라이선스, 인용정보 등 구체적인 지원 객체의 구성을 설명하지 않는다. 한편 Venkatesh 외[4]는 공개된 코드의 재현 가능성에도 차이가 있음을 보고하였다. 해당 연구의 결과는 연구 산출물의 공개 여부와 그 내부 콘텐츠의 구성 및 이용 가능성을 구분하여 살펴볼 필요가 있음을 보여주며, 이는 본 연구가 다루는 RQ1과 RQ2에 연결된다.

이와 같은 재사용성 논의는 연구 소프트웨어에 관한 문헌에서도 폭넓게 뒷받침된다. 공개 개발, 버전 관리, 라이선스 명시, 소프트웨어 인용 정보, 사용자 문서와 같은 실천은 소프트웨어의 신뢰성과 발견성을 높이는 요소로 제시되어 왔다[19]–[22]. 이 가운데 문서화는 특히 중요한 역할을 하는데, 이는 개발자가 아닌 이용자가 설치, 입력과 출력, 제약 조건을

이해하도록 돕는 핵심적인 매개이기 때문이다[23]. 최근 제안된 FAIR4RS 원칙은 이러한 논의를 연구 소프트웨어에 특화된 형태로 발전시켜, 연구 소프트웨어를 데이터와는 구분되는 별도의 디지털 연구 산출물로 취급하고 식별, 접근, 상호운용, 재사용 조건을 명시적으로 갖출 것을 제안한다[24]. 기계학습 연구의 재현성에 관한 논의에서도 유사한 문제의식이 확인된다. 코드, 데이터, 환경, 평가 절차의 공개 수준이 재현 가능성 판단에 직접적으로 관여한다는 논의가 축적되어 있다[25],[26].

모델 카드와 데이터셋 카드는 인공지능 콘텐츠 생태계에서 비교적 최근에 자리 잡은 문서화 장르로, 코드 자체와는 구분되는 별도의 정보 요구를 충족시킨다. 모델 카드는 모델의 의도된 용도, 성능, 한계 등을 설명하며[27], 데이터셋 문서화는 수집 과정, 구성, 권장 사용, 주의사항을 기록한다[28]. 이 두 유형의 객체는 코드 저장소 내부에 존재할 수도 있고, 별도의 플랫폼에 존재할 수도 있다. 위치의 비교정성은 이 객체들이 단일한 저장소 레코드로 온전히 환원되기 어렵다는 것을 의미하며, 본 연구가 이들을 별도의 객체 유형으로 다루는 근거가 된다.

2-3 메타데이터 표준과 콘텐츠 발견성

앞서 다룬 논의를 콘텐츠 발견 시스템의 설계로 연결하기 위해서는 메타데이터 표준에 대한 검토가 필요하다. 디지털 콘텐츠의 발견성은 URL의 나열보다 구조화된 관계 기술에 의존한다.

DataCite는 관련 식별자와 관계 유형을 제공하며, Schema.org는 ScholarlyArticle, SoftwareSourceCode, Dataset 등 웹 기술에 적합한 유형 체계를 제공한다. CodeMeta는 연구 소프트웨어에 특화된 메타데이터를 지원하며, RO-Crate는 관련 산출물의 집합을 맥락 정보와 함께 기술할 수 있는 패키징 방식을 제공한다[29]–[32],[13]. 이 네 가지 표준은 각기 다른 배경에서 발전해 왔지만, 공통적으로 연구 산출물을 개별 파일이 아니라 관계로 연결된 구조로 기술하려 한다는 지향점을 공유한다.

이 표준들이 다루는 범위는 인용과 관계의 명시화라는 운영 과제와도 맞닿아 있다. Cousijn 외[33]는 출판 과정에서 데이터 인용을 구현하기 위한 구체적 절차를 제시하였으며, 이 절차는 메타데이터를 생성하는 주체와 출판 단계별 책임을 구분하여 배분할 것을 요구한다. Sansone 외[34]는 표준, 리포지터리, 정책 정보를 서로 연결하는 메타데이터 레지스트리의 역할을 강조하였다. 이보다 앞선 의미론적 출판 논의는 데이터와 메타데이터의 구조화된 연결이 학술 콘텐츠의 탐색과 해석을 확장할 수 있다는 관점을 제시하였으며[35], FAIR는 이러한 관점에서 단일한 기술 표준이라기보다 기계가 처리 가능한 표현과 연결을 지향하는 원칙의 집합으로 이해될 수 있다[36]. 즉, 객체를 인간과 기계 모두가 해석할 수 있도록 관계·유형·접근 조건을 명시적으로 기술해야 한다는 방향

으로 수립할 수 있다.

그러나 README와 일반 문서, 프로젝트 페이지와 보충자료, 이슈 채널 상태와 개별 이슈 레코드를 서로 구분하여 기술하기 위해서는 표준 자체의 채택을 넘어 응용 프로파일 차원의 세부 규칙이 요구된다. 예를 들어, Schema.org는 README와 일반 문서를 모두 DigitalDocument라는 동일한 유형으로 다룰 수 있지만, 이 둘은 논문과의 관계나 갱신 빈도, 공식성 검증 절차에서 서로 다른 성격을 가진다. 이러한 차이를 포착하려면 표준이 제공하는 일반적 어휘 위에 개별 연구 커뮤니티가 합의한 세부 규칙이 추가될 필요가 있다. 데이터 인용의 실제 구현에서도 메타데이터를 누가 생성하고 출판 과정의 어느 단계에서 책임을 지는지에 대한 배분이 함께 고려되어야 한다[33]. 본 연구는 표준이 제공하는 개념적 자원과 실제 콘텐츠 인벤토리에서 관찰되는 객체들 사이의 접점을 실증적으로 검토하고자 한다.

III. 연구방법

3-1 연구설계와 분석 단위

본 연구는 공식 GitHub 저장소와 연결된 arXiv 인공지능 분야 논문의 디지털 연구콘텐츠를 기술적으로 인벤토리화한 단면적 기술 연구다. 표집 단위는 논문이며, 주 분석 단위는 각 논문과 공식적으로 연결된 공개 디지털 객체다. 여기서 '인공지능 분야 논문'은 arXiv의 cs.CL, cs.CV, cs.LG, cs.AI 주범주에 속하는 논문을 뜻한다. 표본은 이 네 주범주를 대상으로 수집하되, 분석에서는 cs.CL, cs.CV, cs.LG/cs.AI의 세 층으로 구성하였다. 객체는 논문 PDF, arXiv HTML, 저장소 메타데이터, README, 파일 트리에서 탐지된 문서화·실행환경 관련 경로, 모델·데이터셋 카드, 프로젝트 페이지, 보충자료, 라이선스·인용 파일, 릴리스 및 이슈 채널 상태를 포함한다. 해당 연구에서의 정의는 논문과의 연결·유형·접근 상태를 기록하기 위한 기술적 분류다.

3-2 표본 구성 및 검증

표본들은 arXiv API를 이용해 구축하였다. cs.CL, cs.CV, cs.LG 또는 cs.AI에 속하고 2025년 1월 1일부터 2026년 6월 30일까지 최초 제출된 레코드를 검색하였다. 여기서 '최초 제출'은 논문이 이후 개정본으로 갱신되었는지와 무관하게, arXiv에 처음 등록된 시점을 기준으로 하였다. API 요청은 제출일 조건을 포함하고 제출일 오름차순으로 정렬했으며, 기간별 요청을 주 단위 창으로 나누어 실행하였다.

arXiv 논문의 분야 배정에는 Atom feed의 주범주(primary category)를 사용하였다. 수집 과정에서는 주범주와 전체 등재 범주를 별도로 저장하였으며, 최종 본수집과 보

충수집에서는 논문의 주범주가 해당 목표층의 허용 범주에 포함되는 경우에만 해당 분석층에 배정하였다. 이에 따라 주범주가 cs.CL인 논문은 cs.CL 층, cs.CV인 논문은 cs.CV 층, cs.LG 또는 cs.AI인 논문은 통합 cs.LG/cs.AI 층으로 배정하였다. 복수 범주에 교차 등재된 경우에도 주범주를 기준으로 하나의 분석층에만 배정하였다. 최종 표본에서 cs.LG/cs.AI 층 40편은 cs.LG 주범주 32편과 cs.AI 주범주 8편으로 구성되었다.

표본 구성은 네 단계로 진행하였다. 첫째, 3개 분야와 3개 기간(2025년 상반기, 2025년 하반기, 2026년 상반기)을 교차한 9개 층을 설계하고, 층당 40편을 목표로 arXiv API에서 1차 후보 360건(9개 층 × 40편)을 수집하였다. 이 단계의 후보는 공식 GitHub 저장소 검증을 거치기 전의 원시 목록이다. 둘째, 일부 층에서는 공식 저장소가 확인되지 않는 논문이 많아 목표 편수를 채우지 못했다. 이를 보충하기 위해 해당 층에서 160건을 추가로 수집하였다. 셋째, 이렇게 확보한 총 520건의 후보 전체에 공식성과 접근성, 저장소 구성 규칙을 적용해 검증한 결과, 162편만이 적격 판정을 받아 최종 적격 후보를 이루었다. 총 520편의 후보 가운데 358편이 적격 기준을 충족하지 못하였다. 제외 사유를 살펴보면 공식 GitHub 저장소를 확인할 수 없는 경우가 264편(73.7%)으로 가장 많았다. 저장소 후보는 확인되었으나 수작업 검토 후에도 논문과의 공식적·직접적 연계를 충분히 확정하기 어려운 경우는 39편(10.9%)이었으며, 비공식 또는 제3자 저장소로 확인된 경우는 22편(6.1%)이었다. 이외에 README 부재 또는 접근 불가 14편(3.9%), 연구와 관련된 코드·모델·데이터를 저장소에서 확인할 수 없는 경우 11편(3.1%), 공식성을 판단할 근거가 충분하지 않은 경우 8편(2.2%)이 제외되었다. 제외 사유별 분포는 표 1에 제시하였다. 넷째, 이 162편 중에서 분야별 균형, 기간별 분포, 동일 저장소 중복 배제, 제1저자 집중 완화 규칙을 적용해 각 분야 40편씩, 총 120편을 최종 표본으로 확정하였다.

표 1. 후보 논문 제외 사유

Table 1. Reasons for exclusion of candidate papers

Exclusion Reason	N	%
Official GitHub repository not identified	264	73.7
Official or direct linkage could not be confirmed after manual review	39	10.9
Unofficial or third-party repository	22	6.1
README missing or inaccessible	14	3.9
Research-related code, model, or data not identifiable in repository	11	3.1
Insufficient evidence to establish repository officiality	8	2.2
Total	358	100.0

공식 GitHub 저장소 여부는 다음 세 가지를 결합해 판정하였다. 1) arXiv 초록 페이지 또는 논문 PDF·HTML의 링크.

2) 저자 프로젝트 페이지, 공식 연구그룹 페이지, Hugging Face 논문 페이지 등 외부 근거. 3) 저장소 README 또는 메타데이터에 담긴 논문 제목, arXiv ID, DOI, 인용 정보, 고유하게 일치하는 프로젝트명 가운데 하나 이상. 검색엔진 결과에서만 발견되고 위 근거로 확인되지 않는 저장소는 공식 저장소로 인정하지 않았다. GitHub 검색은 후보 발견에만 사용하였다. 일반 도구, 프레임워크, 제3자 구현은 제외하였다.

3-3 객체 유형 정의와 수집 절차

2장에서 논의한 바와 같이 논문 주변 객체는 핵심 객체와 보조 객체로 구분된다. 논문 PDF와 공식 GitHub 저장소는 핵심 객체이며, README, 실행환경 파일, 모델 카드, 데이터셋 카드, 프로젝트 페이지, 보충자료, 라이선스 파일, 인용 파일, 릴리스 노트, 이슈 채널은 이용·재현·지원의 맥락을 제공하는 보조 객체다. 본 연구는 이 구분에 따라 논문별로 13가지 객체를 수집하였다. 각 레코드에는 객체 유형, URL, 접근 상태, 출처 위치, 공식 연결 근거, 수집 시각, 텍스트 이용 가능 여부, 라이선스, 버전·릴리스 정보를 함께 수집하였다.

표 2. 객체 유형 및 수집 항목 정의

Table 2. Object type and collection item definition

Object Type	Category	Collected Items
Paper PDF	Core	URL, access status, text availability
arXiv HTML status record	Core	Access status, HTTP status code
Official GitHub repository	Core	Repository URL, official-link evidence, version/release info
README	Supplementary	Presence, path
Documentation/support metadata	Supplementary	Doc path, requirements/environment files, examples/notebooks
Model card	Supplementary	URL, source location (in-repo / external platform)
Dataset card	Supplementary	URL, source location, license
Project page	Supplementary	URL, source location
Supplementary material	Supplementary	URL, source location
License file	Supplementary	License type, path
Citation file	Supplementary	File presence, path
Release notes	Supplementary	Number of releases, version info
GitHub issue channel status	Supplementary	Issue feature enabled, sampled issue status (open/closed)

저장소 객체는 GitHub 저장소 메타데이터 API, README API, 기본 브랜치의 재귀적 파일 트리 API로 수집하였다. 저

장소 메타데이터와 파일 트리 스냅샷은 별도의 레코드로 각각 수집하였다. 자동 탐지의 재현성을 확보하기 위해 객체 유형별 탐지 규칙을 사전에 정의하였다. 문서화 경로, 예제·튜토리얼·노트북, 요구사항·실행환경 파일, 라이선스 파일, 인용 파일은 GitHub 기본 브랜치의 재귀적 파일 트리에서 경로명 및 파일명 패턴을 적용하여 탐지하였으며, 실제 적용한 탐지 규칙은 부록에 제시하였다. 문서화·실행환경·예제 경로는 파일 트리 메타데이터로만 수집하였다. 프로젝트 페이지와 보충 자료는 arXiv 페이지에서 추출한 외부 링크와 저장소 메타데이터의 홈페이지 링크로 수집하였다. 모델·데이터셋 카드는 README, 저장소 설명·홈페이지, 프로젝트 페이지에서 탐지된 Hugging Face URL을 대상으로 수집하였다. GitHub 이슈는 저장소의 이슈 기능 여부를 메타데이터로 수집하였으며, 열람·단합 이슈 목록에서 사전 정의한 키워드가 포함된 이슈를 우선하여 저장소당 최대 20건까지 수집하였다.

또한 패턴 기반 자동 탐지의 타당도를 확인하기 위해 최종 표본 120편 중 20편을 무작위로 선정하여 저장소의 파일 트리와 관련 페이지를 수작업으로 대조하였다. 수작업 확인 결과를 기준으로 자동 탐지 결과가 실제 존재하는 객체를 정확하게 식별했는지와 실제 존재함에도 탐지되지 않은 객체가 있었는지를 확인하고 정밀도(precision)와 재현율(recall)을 산출하였다. 검증 결과 전체 정밀도는 1.000, 재현율은 .872로 나타났다. 요구사항·실행환경 파일과 라이선스 파일은 검증 표본에서 모두 탐지되었으나, 문서화 경로와 인용 파일에서는 실제 존재하는 객체 중 일부가 자동 탐지 과정에서 확인되지 않았다. 이는 비표준적인 경로나 파일명이 사용된 경우 일부 객체가 탐지되지 않을 수 있음을 의미하며, 해당 유형의 보율은 실제 수준보다 다소 낮게 산출되었을 가능성이 있다.

한편 GitHub 이슈 채널에 대해 수집한 항목(github_issues)은 저장소 수준에서 이슈 채널의 접근 가능 여부와 활성화 상태만을 수집한 레코드이며, 개별 이슈 각각의 목록을 포함하지 않았다.

3-4 변수 구성 및 분석 절차

각 객체 레코드에는 객체 유형, URL, 접근 상태, 공식 연결 근거, 출처 위치, 원시 파일 위치, 텍스트 이용 가능 여부, 라이선스, 버전·릴리스 정보, 수집 시각을 수집하였다. 저장소 메타데이터에는 소유자·저장소명, 기본 브랜치, 생성·갱신·푸시 시각, 보관 상태, 라이선스, 이슈 사용 여부, README 경로, 문서·예제·요구사항·인용 파일 탐지 여부, 릴리스 수, 선택 이슈 수를 수집하였다.

객체 유형별 보유 현황은 레코드 수와 논문 단위 보유 비율로 분석하였으며, 저장소의 문서화 및 지원 특성은 README, 명시적 문서 경로·링크, 예제·튜토리얼·노트북, 요구사항·환경 파일, 인용 파일, 이슈 활성화, 보관 상태의 빈도와 비율로 분석하였다. 이후 논문별 객체 보유 여부를 0/1로 재구성하고, 문서화·지원 메타데이터, 라이선스 파일, 모델 카

드, 데이터셋 카드, 프로젝트 페이지, 보충자료의 보유 여부와 세 분야 간의 독립성을 Pearson 카이제곱 검정으로 분석하였다. 이 여섯 유형은 3×2 분할표의 모든 기대빈도가 5 이상인 경우로 한정하였다. 여섯 검정의 다중 비교는 Holm 방법으로 보정하였다. 인용 파일과 릴리스 노트는 기대빈도 조건을 충족하지 않아 검정에서 제외하였다. 이 분석은 사전에 설정한 가설 검증이 아니라, 기술적 인벤토리에서 관찰된 분야별 차이 가능성을 점검하는 보조적 탐색 절차로 하였다.

저장소 운영기간이 객체 보유 결과에 미칠 수 있는 영향을 추가로 확인하였다. 저장소 운영기간은 GitHub 저장소의 생성 시각과 객체 수집 시각의 차이를 일수로 산출하였다. 각 객체 유형의 보유 여부에 따른 저장소 운영기간의 차이는 Mann-Whitney U 검정으로 비교하였으며, 11개 객체 유형에 대한 반복 검정의 유의확률은 Holm 방법으로 보정하였다. 또한 분야별 비교에서 통계적으로 유의한 연관성이 관찰된 모델 카드에 대해서는 보유 여부를 종속변수로 하고 세부 분야와 저장소 운영기간을 독립변수로 투입한 이항 로지스틱 회귀분석을 실시하였다. 회귀분석에서는 저장소 운영기간을 100일 단위로 환산하였으며, cs.CL을 기준범주로 설정하였다.

3-5 객체 유형의 메타데이터 표현 가능성 검토

본 연구는 DataCite, Schema.org, CodeMeta, RO-Crate 네 가지 표준을 대상으로, 각 객체 유형이 해당 표준에서 직접 표현 가능한지, 관련 객체로서 간접적으로만 표현 가능한지, 표준의 확장이나 로컬 프로파일링이 필요한지를 검토하였다. 분석 항목은 객체 유형, 논문과의 관계, 출처·검증 근거, 접근 상태, 버전·갱신 정보이다.

메타데이터 표현 가능성에 대한 검토는 각 객체 유형에 대해 네 가지 표준의 공식 문서와 어휘 정의를 대조하는 방식으로 진행하였다. 논문 PDF는 Schema.org의 ScholarlyArticle 유형이 직접 대응하는지, 공식 GitHub 저장소는 CodeMeta나 Schema.org의 SoftwareSourceCode 유형이 직접 대응하는지를 확인하였다. README나 이슈 채널처럼 특정 표준에 전용 유형이 없는 객체는, 해당 표준이 제공하는 관련 객체 속성이나 맥락 파일 개념으로 간접 표현이 가능한지를 분석하였다. 표현 가능성 검토에서는 표현 가능성을 직접 표현, 간접 표현, 부분 표현, 적용 불가의 네 범주로 구분하였다. 직접 표현은 해당 표준이 객체를 독립적인 자원으로 표현할 수 있는 명시적이고 의도된 구성요소를 제공하는 경우로 정의하였다. 객체별 전용 클래스뿐 아니라 파일 엔터티, 맥락적 엔터티, 소프트웨어 자원, 웹페이지, 데이터셋 등 자원 수준의 표현구조를 포함하였다. 간접 표현은 객체가 관련 자원 링크, 일반적 관계 속성, URL 또는 상위 수준의 일반 자원을 통해 표현되는 경우로, 부분 표현은 일부 속성이나 관계는 기술할 수 있으나 객체 고유의 역할을 충분히 구분하기 어려운 경우로 정의하였다. 대응 가능한 표현구조를 확인

하기 어려운 경우는 적용 불가로 판정하였다.

판정의 신뢰도를 확인하기 위해 9개 객체 유형과 DataCite, Schema.org, CodeMeta, RO-Crate의 4개 표준으로 구성된 36개 조합을 제2판정자가 독립적으로 판정하였다. 판정자 간 일치도를 Cohen's kappa로 확인한 결과, 관찰 일치율은 91.7%, Cohen's κ 는 .881로 나타났다. 불일치 항목은 해당 표준의 공식 문서와 어휘 정의를 다시 대조하여 최종 판정에 반영하였다.

IV. 연구결과

4-1 객체 유형별 보유 현황

최종 데이터셋(표 3)은 논문 120편과 객체 레코드 1,467건으로 구성하였다. 논문당 객체 수는 최소 6건에서 최대 54건이며, 중앙값은 11.0건이다. 본 연구에서는 객체 유형별 현황을 객체 레코드 수와 해당 객체를 하나 이상 보유한 논문 수로 구분하여 집계하였다. 객체 레코드 수는 동일 논문에서 같은 유형의 객체가 복수로 확인된 경우 각각을 별도로 계산한 값이므로, 문서화·지원 메타데이터, 모델 카드, 보충자료, 인용 파일 등에서는 객체 레코드 수가 보유 논문 수보다 많을 수 있다. 반면 라이선스 파일은 수집 단계에서 저장소별 1건을 기록하도록 하였기 때문에 레코드 수와 보유 논문 수가 동일하게 나타났다.

표 3. 디지털 연구콘텐츠 객체 유형별 보유 현황

Table 3. Holding status of digital research-content objects by type

Digital Research-Content Object	Object Records	Papers Holding Object
Paper PDF	120	120
arXiv HTML status record	120	106 (accessible)
Official GitHub repository	120	120
README	120	120
Documentation/support metadata	356	99
Supplementary material	156	31
Model card	70	35
Dataset card	38	25
Project page	42	34
Issue channel status record	120	120
License file	68	68
Citation file	8	7
Release notes	9	9

논문 PDF, 공식 GitHub 저장소, README, 이슈 채널 상태 레코드는 120편 전체에서 확인되었다. 이 네 유형의 보유 비율은 모두 100.0%다. arXiv HTML 상태 레코드는 120건을 시도하여 106건이 접근 가능했고(88.3%), 14건은 HTTP 404 오류로 접근이 불가능했다. 나머지 8개 유형의 보유 비율은 논문마다 상이했다. 문서화·지원 메타데이터는 356건의 레코드가 확인되었으며, 이를 보유한 논문은 99편(82.5%)이다. 보충자료는 156건의 레코드가 확인되었으며 보유 논문은 31편(25.8%)이다. 모델 카드는 70건의 레코드, 보유 논문 35편(29.2%)이다. 데이터셋 카드는 38건의 레코드, 보유 논문 25편(20.8%)이다. 프로젝트 페이지는 42건의 레코드, 보유 논문 34편(28.3%)이다. 라이선스 파일은 68건의 레코드, 보유 논문 68편(56.7%)으로 레코드 수와 보유 논문 수가 일치한다. 인용 파일은 8건의 레코드, 보유 논문 7편(5.8%)이다. 릴리스 노트는 9건의 레코드, 보유 논문 9편(7.5%)이다. 객체 유형별 보유 비율은 최저 5.8%(인용 파일)에서 최고 100.0%(PDF·저장소·README·이슈 채널)까지 분포한다.

4-2 AI 세부 분야별 객체 보유 결과

분야별 비교는 논문 단위의 객체 보유 여부(0/1)를 기준으로 하였다(표 4). 문서화·지원 메타데이터 보유 논문은 cs.CL 34편(85.0%), cs.CV 31편(77.5%), cs.LG/cs.AI 34편(85.0%)이며, $\chi^2=1.039$, $p=.595$, Holm 보정 $p=1.000$ 이다. 라이선스 파일 보유 논문은 cs.CL 22편(55.0%), cs.CV 25편(62.5%), cs.LG/cs.AI 21편(52.5%)이며, $\chi^2=0.882$, $p=.643$, Holm 보정 $p=1.000$ 이다.

모델 카드 보유 여부는 단순 분야 비교에서 통계적으로 유의한 연관성이 나타났다($\chi^2=10.245$, $p=.006$, Holm 보정

$p=.036$, Cramér's $V=.292$). 여섯 객체 유형 가운데 모델 카드에서만 Holm 보정 이후에도 유의한 연관성이 관찰되었다. 다만 이 결과는 저장소 운영기간을 고려하여 추가분석 결과와 함께 해석하였다.

데이터셋 카드 보유 논문은 cs.CL 7편(17.5%), cs.CV 12편(30.0%), cs.LG/cs.AI 6편(15.0%)이며, $\chi^2=3.133$, $p=.209$, Holm 보정 $p=.626$ 이다. 프로젝트 페이지 보유 논문은 cs.CL 16편(40.0%), cs.CV 10편(25.0%), cs.LG/cs.AI 8편(20.0%)이며, $\chi^2=4.268$, $p=.118$, Holm 보정 $p=.473$ 이다. 보충자료 보유 논문은 cs.CL 13편(32.5%), cs.CV 5편(12.5%), cs.LG/cs.AI 13편(32.5%)이며, $\chi^2=5.567$, $p=.062$, Holm 보정 $p=.309$ 이다.

4-3 저장소 운영기간을 고려한 추가분석

최종 120개 공식 GitHub 저장소의 운영기간 중앙값은 349.6일이었으며($Q1=157.5$, $Q3=504.9$), 최소 20.7일에서 최대 2,818.1일까지 분포하였다. 객체 유형별 보유 여부에 따라 저장소 운영기간에 차이가 나타나는지를 확인하기 위해 11개 객체 유형에 대해 Mann-Whitney U 검정을 실시하고 Holm 방법으로 유의확률을 보정하였다. 그 결과 Holm 보정 이후 저장소 운영기간에 통계적으로 유의한 차이가 확인된 객체 유형은 없었다(표 5).

모델 카드의 분야별 차이가 저장소 운영기간을 고려한 이후에도 유지되는지를 확인하기 위해 이항 로지스틱 회귀분석을 실시하였다. 저장소 운영기간을 100일 단위로 함께 투입한 결과, cs.CL을 기준으로 cs.CV의 모델 카드 보유 오즈는 1.890배였으나 통계적으로 유의하지 않았으며($OR=1.890$, 95% $CI=0.748\sim4.773$, $p=.178$), cs.LG/cs.AI의 오즈 역시 0.330배로 나타났으나 유의하지 않았다($OR=0.330$, 95% $CI=0.103\sim1.054$, $p=.061$). 저장소 운영기간 역시 모델 카드 보유 여부와 유의한 관련이 확인되지 않았다(100일 증가당 $OR=0.990$, 95% $CI=0.879\sim1.115$, $p=.865$). 따라서 단순 분야 비교에서 관찰된 모델 카드 보유 차이는 저장소 운영기간을 함께 고려한 모형에서는 통계적으로 유의한 분야 효과로 확인되지 않았다.

4-4 저장소 및 문서화 특성

표 6에 정리한 바와 같이 120개 저장소 전체에 README 경로가 존재하였다. 요구사항 또는 환경 파일은 89개 저장소(74.2%)에서 확인되었다. 예제·튜토리얼·노트북은 37개 저장소(30.8%)에서, 명시적 문서 디렉터리 또는 링크는 17개 저장소(14.2%)에서 확인되었다. 인용 파일은 7개 저장소(5.8%)에서 확인되었다. 이슈 기능은 118개 저장소(98.3%)에서 활성화되어 있었고, 2개 저장소(1.7%)에서는 비활성 상태였다. 선택 이슈는 60개 저장소를 대상으로 수집하였으며, 총 257건 중 114건(44.4%)은 열림 상태, 143건(55.6%)은

표 4. AI 세부 분야별 객체 보유와 독립성 검정

Table 4. Object holding by AI subfield and independence test

Object Type	Holding by Subfield(cs.CL/cs.CV/cs.LG·AI)	χ^2 , p , Holm-adjusted p	Cramér's V
Documentation/support metadata	34(85.0%)/31(77.5%)/34(85.0%)	$\chi^2=1.039$, $p=.595$, Holm $p=1.000$	.093
License file	22(55.0%)/25(62.5%)/21(52.5%)	$\chi^2=0.882$, $p=.643$, Holm $p=1.000$	.086
Model card	12(30.0%)/18(45.0%)/5(12.5%)	$\chi^2=10.245$, $p=.006$, Holm $p=.036$	.292
Dataset card	7(17.5%)/12(30.0%)/6(15.0%)	$\chi^2=3.133$, $p=.209$, Holm $p=.626$	.162
Project page	16 (40.0%)/10(25.0%)/8(20.0%)	$\chi^2=4.268$, $p=.118$, Holm $p=.473$	.189
Supplementary material	13(32.5%)/5(12.5%)/13(32.5%)	$\chi^2=5.567$, $p=.062$, Holm $p=.309$	.215

표 5. 객체 보유 여부에 따른 저장소 운영기간 비교 (일)

Table 5. Comparison of repository age by object holding status (days)

Object Type	Present, n; Median (Q1-Q3)	Absent, n; Median (Q1-Q3)	U; Holm p
Documentation/ support metadata	99; 333.8 (135.6-491.3)	21; 418.7 (272.9-531.3)	879.0; 1.000
Explicit documentation directory/link	17; 315.9 (128.2-566.7)	103; 361.7 (163.1-498.4)	896.0; 1.000
Requirements/ environment	89; 333.8 (134.5-474.7)	31; 418.7 (213.9-539.1)	1146.0; 1.000
Examples/tutori- als/notebooks	37; 333.8 (160.1-566.7)	83; 361.7 (149.6-473.3)	1661.0; 1.000
License file	68; 349.6 (166.1-519.5)	52; 356.2 (133.3-474.8)	1880.0; 1.000
Citation file	7; 348.6 (121.6-544.4)	113; 350.6 (160.1-501.1)	393.0; 1.000
Release notes	9; 602.7 (348.6-1136.8)	111; 341.0 (155.0-481.0)	693.0; .545
Model card	35; 345.5 (198.5-507.9)	85; 371.3 (134.5-501.6)	1498.0; 1.000
Dataset card	25; 311.8 (196.1-501.1)	95; 361.7 (142.2-508.1)	1203.0; 1.000
Project page	34; 372.2 (201.9-523.1)	86; 337.4 (128.6-484.0)	1662.0; 1.000
Supplementary material	31; 271.2 (117.7-404.6)	89; 403.9 (196.1-522.1)	925.0; .071

표 6. 공식 GitHub 저장소의 문서화 및 지원 특성

Table 6. Documentation and support characteristics of official GitHub repositories

Repository/Documentation Characteristic	Repositories	Rate
README path present	120	100.0%
Explicit documentation directory/link	17	14.2%
Examples/tutorials/ notebooks	37	30.8%
Requirements/ environment files	89	74.2%
Citation file	7	5.8%
Issues enabled	118	98.3%
Sampled issues collected	60	50.0%
Archived repository	1	0.8%

단합 상태였다. 보관(archived) 처리된 저장소는 1개(0.8%)였다.

4-5 객체 유형의 메타데이터 표현 가능성 검토 결과

표 7은 9가지 객체 유형이 DataCite, Schema.org, CodeMeta, RO-Crate 네 표준에서 얼마나 직접적으로 표현 가능한지를 검토한 결과이다. 논문 PDF, 공식 GitHub 저장

소, 데이터셋 카드는 최소 한 표준에서 직접적인 자원 수준의 표현이 가능한 반면, 나머지 6개 유형(README·문서화, 요구사항·환경 파일, 모델 카드, 프로젝트 페이지·보충자료, 라이선스·인용 파일·릴리스, 이슈 채널 상태)은 네 표준 모두에서 간접 표현이나 부분 표현에 그친다. 이는 객체 유형에 따라 기존 표준에서 직접적인 자원 수준의 표현이 가능한 정도가 서로 다르며, 일부 저장소 내부 객체의 역할을 구분하기 위해서는 추가적인 관계 기술이나 프로파일링이 필요할 수 있다.

직접 표현이 가능한 세 유형과 그렇지 않은 여섯 유형이 요구하는 핵심 메타데이터도 성격이 다르다. 전자는 식별자·저자·URL·버전처럼 비교적 표준화된 서지 정보를 요구하는 반면, 후자는 경로·접근 상태·실행환경·활성화 상태처럼 저장소 내부 구조에 특화된 정보를 요구한다.

표 7. 디지털 연구콘텐츠 객체의 메타데이터 스키마 표현 가능성
Table 7. Representability of digital research-content objects in metadata schemas

Object Type	Representability across DataCite/Schema.org/CodeMeta/RO-Crate	Key Metadata for Implementation
Paper PDF	Direct(Schema.org: ScholarlyArticle; RO-Crate: included); not applicable in CodeMeta	Identifier, author, publication date
Official GitHub repository	Direct(Schema.org: SoftwareSourceCode; CodeMeta; RO-Crate); related identifier in DataCite	Official-link evidence, repository URL, version
README/ documentation	Indirect only (Schema.org: DigitalDocument; CodeMeta: partial; RO-Crate: context file)	Object type, path, access status
Requirements/ environment files	Indirect only (Schema.org: MediaObject; CodeMeta: partial; RO-Crate: context file)	File type, execution environment, update date
Model card	Indirect only across all four standards; can be included in RO-Crate	Model identifier, intended use, limitations
Dataset card	Direct in Schema.org (Dataset); limited in CodeMeta	Dataset identifier, license, access status
Project page/ supplementary material	Indirect (Schema.org: WebPage/CreativeWork); not applicable in CodeMeta	Relation type, URL, provenance
License/ citation file/ release	Indirect only across all four standards	License, citation info, version/ update date
Issue channel status	Indirect/limited across all four standards	Channel URL, active status, collection time

V. 논 의

5-1 콘텐츠 진입점으로서의 논문과 저장소

본 연구가 확인한 것은 접근점과 콘텐츠 구성 사이의 차이이다. 논문 PDF, 공식 저장소, README는 표본 전체에서 확인된 반면, 인용 파일과 릴리스 노트 등 일부 지원 객체의 보유는 제한적으로 나타났다. 이는 저장소 링크가 콘텐츠 구성을 그대로 반영하지 않을 수 있음을 보여준다.

다만 이러한 보유율은 표본 구성 방식과 분리하여 해석할 수 없다. 본 연구는 공식 GitHub 저장소와의 연결이 검증된 논문만을 분석 대상으로 선정하였으므로, 공식 저장소의 보유는 표본 정의에 의해 보장된 특성이다. README 역시 적격성 검토 과정에서 접근 가능성과 저장소 구성 여부를 확인하는 데 활용되었기 때문에, arXiv AI 논문 전체의 일반적인 문서화 수준으로 해석하기 어렵다.

따라서 RQ1의 결과는 공식 GitHub 저장소가 확인된 논문 집단 내부에서 보조 객체가 어떠한 구성과 편차를 보이는지를 나타낸 결과로 한정하여 해석할 필요가 있다. 또한 공식 저장소와의 연결을 명확하게 확인할 수 있는 논문은 그렇지 않은 논문보다 코드 및 관련 연구 산출물의 공개와 문서화가 상대적으로 체계적일 가능성이 있으므로, 본 연구에서 확인된 보조 객체의 보유율은 전체 arXiv AI 논문을 기준으로 할 경우 다소 높게 나타난 값일 가능성도 있다.

Van de Sompel 외[1]는 학술 커뮤니케이션을 등록, 인증, 인지, 보존, 보상이라는 다섯 기능으로 분해하고, 데이터셋과 소프트웨어가 논문과 대등한 단위로 이 기능들을 수행해야 한다고 제안하였다. 이 틀에 비추어 보면, 공식 저장소 링크는 무엇이 연결되어 있는지 알리는 등록 기능은 수행하지만, 인증이나 보존 기능은 저장소 내부 구성에 따라 선택적으로만 수행되는 것으로 보인다. 인용 파일의 낮은 보유율은 이 점에서 특기할 만하다. Smith 외[21]가 제시한 소프트웨어 인용 원칙은 소프트웨어를 학술 산출물로서 인용 가능하게 만드는 것을 목표로 하는데, 본 표본에서 이 원칙이 구현된 저장소는 120개 중 7개에 그친다. 이는 소프트웨어 인용 원칙의 이행 여부를 의미하지는 않지만, 저장소 수준에서 명시적으로 식별 가능한 인용 정보를 제공하는 사례가 제한적이었다. 추가분석에서는 인용 파일과 릴리스 노트의 보유 여부에 따른 저장소 운영기간 차이가 Holm 보정 이후 통계적으로 유의하게 확인되지 않았다. 다만 두 객체의 보유 사례가 각각 7개와 9개로 적어, 낮은 보유율이 저장소 운영기간과 무관하다고 단정할 수는 없다.

결과에서 보인 비대칭은 향상된 출판물 논의가 지적해 온 문제와도 연결된다. Bardi와 Manghi[14],[15]는 향상된 출판물을 논문과 관련 산출물의 관계를 명시적으로 기술하는 모델로 정의하였는데, 이 모델은 관련 산출물 각각이 논문과 유사한 수준으로 기술된다는 것을 전제로 한다. 본 연구의 결과는 이 전제가 객체 유형에 따라 상당히 다른 정도로 충족되고 있

음을 보여준다. Belhajjame 외[12]와 Soiland-Reyes 외[13]가 발전시킨 연구객체 개념 역시 논문·데이터·소프트웨어·워크플로 사이의 관계 기술을 다루는데, 문서화 및 지원 객체 유형별로 나타난 보유율 편차는 관계의 명시화가 객체마다 다른 정도로 이루어지고 있을 가능성을 시사한다. 또한 Yuan 외[17]이 지적했듯, 연구객체의 재사용 가능성은 구성요소의 단순 집합보다 객체 간 관계와 생성 맥락의 기록에 좌우될 수 있다. 저장소 링크 하나는 콘텐츠의 존재를 알릴 뿐, 그 내부의 관계(어떤 파일이 재현에 필요하고 어떤 파일이 필수적인지)를 드러내지는 않는다. Stein Kenfield 외[16]가 제시한 디지털 객체 재사용의 정의(다운로드를 넘어 검증·변형·재맥락화까지 포괄하는 행위)에 비추어 보면, 본 연구가 확인한 낮은 인용 파일·릴리스 노트 보유율은 검증과 재맥락화 단계에서 참고할 수 있는 공식적 근거가 상대적으로 적을 수 있음을 뜻한다.

5-2 문서화 편차와 재현성 논의의 접점

모든 저장소에서 README가 확인되었고 요구사항과 환경 파일이 74.2%에서 확인되었다. 이는 다수의 저장소가 기본적인 문서 진입점과 실행환경 관련 파일을 제공하고 있음을 의미한다. Peng[8]은 계산과학 연구의 재현성을 완전한 독립 재현이 어려울 때 적용할 수 있는 최소 기준으로 제시하였고, Stodden 외[9]는 이 기준의 구체적 요건으로 코드, 데이터, 실행 환경의 명시적 공개를 제시하였다. 이 틀에 비추어 보면, 요구사항·환경 파일 보유율 74.2%는 이 요건 가운데 하나가 다수 저장소에서 충족되고 있음을 보여주는 한편, 나머지 25.8%에서는 그렇지 않다는 것도 함께 보여준다.

추가적으로 Gundersen과 Kjensmo[25], Pineau 외[26]는 코드, 데이터, 환경, 평가 절차의 공개 수준이 기계학습 연구의 재현 가능성 판단에 관여한다고 지적하였다. 본 연구는 이러한 관찰을 120편 규모의 표본에서 객체 단위로 다시 확인한다. 예제·튜토리얼·노트북(30.8%)과 명시적 문서 경로·링크(14.2%)의 낮은 보유율은, 저장소들이 실행 환경에 관한 정보는 상대적으로 많이 제공하지만 실제 사용 예시를 보여주는 정보는 그렇지 않을 수 있음을 시사한다. Lee[23]가 강조한 사용자 중심 문서화라는 목적에 비추어 보면, 이 두 종류의 정보는 서로 다른 기능을 하므로 보유율이 다르게 나타날 수 있다.

이러한 결과는 기계학습 연구의 재현성을 위해 데이터, 모델, 코드와 실행 환경을 명시적으로 제공해야 한다는 Heil 외[6]의 논의와 연결된다. 본 연구에서는 지원 요소가 저장소 내 객체 유형별로 균등하게 나타나지 않았으며, 특히 인용·릴리스·명시적 문서 관련 객체의 보유가 제한적으로 나타났다. Wattanakriengkrai 외[7]가 보고한 논문과 저장소 간 연결의 희소성과 함께 고려하면, 저장소 존재 여부와 저장소가 실제로 제공하는 지원 정보의 범위는 서로 다른 속도로 개선되고 있을 가능성이 있다. Venkatesh 외[4]가 AI 연구에서 확인한 코드, 데이터 공개와 재현 가능성의 차이를 고려하면, 연구 산출물이 공개되어 있다는 사실과 그 산출물이 실제 이용

과 재현을 지원하는 데 필요한 정보를 갖추고 있다는 것은 구분하여 평가할 필요가 있다.

분야별 비교에서는 모델 카드 보유 여부에서만 Holm 보정 이후 유의한 연관성이 관찰되었으나, 저장소 운영기간을 함께 고려한 로지스틱 회귀에서는 분야계수가 통계적으로 유의하지 않았다. 또한 저장소 운영기간 자체도 모델 카드 보유 여부와 유의한 관련이 확인되지 않았다. 따라서 단순 비교에서 나타난 모델 카드 보유율의 차이를 저장소 운영기간이나 분야 자체의 독립적인 효과로 단정하기는 어렵다. 본 연구에서 확인된 차이는 후속 연구에서 보다 큰 표본과 모델 배포 플랫폼, 연구과제 유형 등 관련 요인을 고려하여 재검증할 필요가 있는 탐색적 결과로 보는 것이 적절하다.

다만 본 연구는 문서의 내용적 충실도, 실행 성공 여부, 개별 이슈의 실질적 의미를 평가하지 않았다. 확인한 것은 특정 유형의 파일이 파일 트리 수준에서 존재하는지 여부뿐이며, 존재가 곧 품질이나 정확성을 보장하지 않는다는 점은 Prlić와 Procter[19]가 이미 지적한 바와 부합한다.

5-3 관계지향 메타데이터 설계로의 합의

메타데이터 표현 가능성 검토 결과는 객체 유형에 따라 현존 표준에서 직접적 또는 간접적으로 기술할 수 있는 범위가 있음을 보여주었다. DataCite의 관련 식별자·관계 유형[29],[30], Schema.org의 콘텐츠 유형[32], CodeMeta의 소프트웨어 기술[31], RO-Crate의 맥락화된 객체 묶음[13]은 논문, 소프트웨어 저장소, 데이터셋처럼 학술 커뮤니티가 오랫동안 다루어 온 산출물 유형에는 비교적 정비된 어휘를 제공한다. 반면 README, 이슈 채널, 프로젝트 페이지와 같은 객체는 네 표준에서 주로 간접적이거나 부분적인 방식으로 표현되므로, 객체의 역할을 구분하여 기술하기 위해서는 추가적인 프로파일링이 필요할 수 있다.

데이터 인용 구현 절차[33]나 메타데이터 레지스트리의 역할을 강조한 연구는[34] 학술 출판 생태계가 데이터라는 새로운 산출물 유형에 적응해 온 과정을 보여준다. FAIR가 단일 표준이 아니라 기계가 처리 가능한 표현과 연결을 지향하는 원칙의 집합이라는 지적에 비추어 보면[36], README나 이슈 채널이 현재 검토한 표준에서 주로 간접적 또는 부분적으로 표현된다는 점은, 객체의 역할을 보다 세부적으로 구분하기 위한 응용 프로파일의 확장 가능성을 보여준다. 이에 본 연구가 확인한 접근 상태 구분(접근 가능, 메타데이터만 확인, 접근 불가)이나 객체 위계(핵심·보조)는 관계지향 메타데이터를 설계할 때 참고할 수 있는 최소한의 틀을 제공한다. 다만 구조화된 연결이 학술 콘텐츠의 탐색과 해석을 확장하려면 저장소 메타데이터와 파일 트리 스냅샷, README와 일반 문서, 프로젝트 페이지와 보충자료를 구분하는 응용 프로파일 차원의 합의가 함께 필요하다는 지적도 있다[35]. 이 합의는 표준 채택만으로 자동으로 따라오지 않으며, 본 연구가 제시하는 객체 유형 분류와 표현 가능성 검토가 이에 필요한

자료로서 기능할 수 있을 것으로 사료된다.

VI. 결 론

본 연구는 공식 GitHub 저장소가 검증된 arXiv 내 AI 관련 분야 논문 120편을 대상으로, 저장소 링크로 연결된 연구 콘텐츠가 실제로 어떠한 객체들로 구성되어 있는지를 살펴보았다. 본 연구의 목적은 저장소 링크가 가리키는 콘텐츠 구성이 실제로 어떠한지, 그리고 그 구성이 메타데이터 설계에 어떤 함의를 갖는지를 밝히는 데 있다.

본 연구의 의의는 학술적 측면과 실무적 측면으로 나누어 볼 수 있다. 학술적으로는 논문과 저장소 연결의 존재 여부와 저장소의 전반적인 문서화 수준을 넘어, 하나의 저장소 링크가 실제로 서로 다른 기능을 가진 객체들로 어떻게 구성되는지를 동일한 분석 단위에서 구분하여 제시하였다는 데 의의가 있다. 이를 통해 저장소의 존재와 그 내부에서 이용·재현·인용을 지원하는 객체의 구성은 서로 구분하여 살펴볼 필요가 있음을 객체 수준에서 확인하였다. 실무적으로는 논문과 공식 저장소의 연결만으로 연구 주변의 디지털 콘텐츠를 충분히 설명하기 어렵다는 점을 보여주고, 주요 메타데이터 표준에서 각 객체 유형의 표현 가능성을 검토하였다는 데 의미가 있다. 이는 향후 논문과 저장소 주변의 다양한 연구콘텐츠를 객체 단위로 기술하고 연결할 때, 기존 메타데이터 요소를 활용할 수 있는 부분과 추가적인 기술이 필요한 부분을 검토하는 데 참고자료로 활용될 수 있다. 이에 저장소 링크만을 추가하는 방식에서 나아가, 논문과 연결된 주요 객체를 구분하여 기술하는 방안을 우선 검토할 수 있다. 본 연구의 수집 항목을 기준으로 할 경우 객체 유형과 URL, 논문과의 공식 연결 근거, 접근 상태, 출처 위치, 라이선스, 버전·갱신 정보가 이러한 기술에 활용할 수 있는 기본 요소에 해당한다. 공식성 확인 역시 논문 본문이나 arXiv 페이지의 링크, 저자, 연구그룹의 공식 페이지, 저장소 내부의 논문 식별정보 등 논문과 저장소의 직접적 관계를 확인할 수 있는 근거와 함께 기록할 필요가 있다.

본 연구는 공식 GitHub 저장소와의 연결이 확인된 arXiv AI 관련 논문으로 표본을 한정하였기 때문에, 일부 객체의 높은 보유율은 표본 선정 조건의 영향을 받았으며 이를 전체 arXiv AI 논문의 특성으로 일반화하는 데 한계가 있다. 저장소 운영기간에 따른 추가분석에서는 객체 보유 여부에 따른 운영기간 차이가 Holm 보정 이후 통계적으로 유의하게 확인되지 않았으나, 인용 파일과 릴리스 노트처럼 보유 사례가 적은 객체가 포함되어 있어 운영기간의 영향을 배제할 수는 없다. 또한 본 연구는 동일 저장소의 장기적 변화를 추적한 종단 연구가 아니므로 운영기간과 객체 축적의 관계를 확정적으로 판단하기 어렵다. 모델·데이터셋 카드와 외부 문서화 객체는 사전에 정한 수집 경로와 링크 유형을 중심으로 확인하였으므로, 비표준적인 경로나 다른 플랫폼·외부 문서 서비스에 존재하는 객체가 일부 집계되지 않았을 가능성이 있다.

분야별 비교는 각 40편의 제한된 표본을 대상으로 한 탐색적 분석이라는 점에서 결과를 제한적으로 해석할 필요가 있다.

향후 연구에서는 동일 저장소를 여러 시점에 반복 수집하여 운영기간에 따른 객체 구성의 변화를 중단적으로 추적하고, 보다 큰 표본을 활용하여 본 연구에서 탐색적으로 관찰된 분야별 차이를 재검증할 필요가 있다. 또한 객체 유형과 관계를 구조화하여 제공하는 방식이 실제 이용에 미치는 효과를 탐색 성공률, 탐색 소요시간, 설치, 실행 및 재현 성공률 등의 지표로 검증할 수 있다.

부 록

부록 A. GitHub 저장소 객체의 자동 탐지 규칙

본 연구에서는 GitHub 저장소의 기본 브랜치에서 확보한 개체적 파일 트리를 대상으로 사전에 정의한 경로명, 파일명, 확장자 및 정규표현식 조건을 적용하여 문서화 경로, 예제·튜토리얼·노트북, 요구사항·실행환경 파일, 인용 파일 및 라이선스 파일을 자동 탐지하였다. 탐지 규칙은 실제 자료수집 코드의 detect_paths(tree_items)함수에 적용된 조건을 기준으로 하였으며, 별도의 의미 기반 분류기는 사용하지 않았다.

표 8. GitHub 저장소 객체의 자동 탐지 규칙

Table 8. Automated detection rules for GitHub repository objects

Object Type	Exact Pattern Used	Detection Logic	Output Cap
Documentation/ explicit documentation path	p.lower().startswith(("docs/", "doc/")) or "documentation" in p.lower()	Detected when the recursive tree path starts with docs/or doc/, or contains documentation	First 5 paths
Examples/ tutorials/ notebooks	p.lower().startswith(("examples/", "example/", "tutorials/", "notebooks/")) or p.lower().endswith(".ipynb")	Detected when the path starts with one of the predefined directory names or ends with .ipynb	First 5 paths
Requirements/ environment files	^(	/(requirements.*.txt	environment.*.ya?ml
Citation files	^(	/(citation.cff	codemeta.json
License files	^(	/(license	licence
README path	^(	/readme(..*)?\$	README path detected in the recursive tree; the README object itself was collected through the GitHub README API

참고문헌

- [1] H. Van de Sompel, S. Payette, J. Erickson, C. Lagoze, and S. Warner, "Rethinking Scholarly Communication: Building the System that Scholars Deserve," *D-Lib Magazine*, Vol. 10, No. 9, September 2004.
- [2] S. Warner, J. Bekaert, C. Lagoze, X. Liu, S. Payette, and H. Van de Sompel, "Pathways: Augmenting Interoperability across Scholarly Repositories," *International Journal on Digital Libraries*, Vol. 7, pp. 35-52, 2007. <https://doi.org/10.1007/s00799-007-0016-7>
- [3] D. Kang, T. Y. Kang, and J. Jang, "Papers with Code or without Code? Impact of GitHub Repository Usability on the Diffusion of Machine Learning Research," *Information Processing & Management*, Vol. 60, No. 6, 103477, 2023. <https://doi.org/10.1016/j.ipm.2023.103477>
- [4] K. Venkatesh, S. M. Santomartino, J. Sulam, and P. H. Yi, "Code and Data Sharing Practices in the Radiology Artificial Intelligence Literature: A Meta-Research Study," *Radiology: Artificial Intelligence*, Vol. 4, No. 5, e220081, 2022. <https://doi.org/10.1148/ryai.220081>
- [5] E. Escamilla, M. Klein, T. Cooper, V. Rampin, M. C. Weigle, and M. L. Nelson, "The Rise of GitHub in Scholarly Publications," in *Proceedings of the 26th International Conference on Linking Theory and Practice of Digital Libraries*, Padua, Italy, pp. 187-200, 2022. https://doi.org/10.1007/978-3-031-16802-4_15
- [6] B. J. Heil, M. M. Hoffman, F. Markowitz, S.-I. Lee, C. S. Greene, and S. C. Hicks, "Reproducibility Standards for Machine Learning in the Life Sciences," *Nature Methods*, Vol. 18, pp. 1132-1135, 2021. <https://doi.org/10.1038/s41592-021-01256-7>
- [7] S. Wattanakriengkrai, B. Chinthanet, H. Hata, R. G. Kula, C. Treude, J. Guo, and K. Matsumoto, "GitHub Repositories with Links to Academic Papers: Public Access, Traceability, and Evolution," *Journal of Systems and Software*, Vol. 183, 111117, 2022. <https://doi.org/10.1016/j.jss.2021.111117>
- [8] R. D. Peng, "Reproducible Research in Computational Science," *Science*, Vol. 334, No. 6060, pp. 1226-1227, 2011. <https://doi.org/10.1126/science.1213847>
- [9] V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, ... and M. Taufer, "Enhancing Reproducibility for Computational Methods," *Science*, Vol. 354, No. 6317, pp. 1240-1241, 2016. <https://doi.org/10.1126/science.aah6168>
- [10] C. Lagoze, H. Van de Sompel, M. L. Nelson, S. Warner, R. Sanderson, and P. Johnston, "Object Re-Use & Exchange: A Resource-Centric Approach," arXiv:0804.2273, April 2008. <https://doi.org/10.48550/arXiv.0804.2273>

- [11] S. Bechhofer, D. De Roure, M. Gamble, C. Goble, and I. Buchan, "Research Objects: Towards Exchange and Reuse of Digital Knowledge," *Nature Precedings*, pp. 1-6, 2010. <https://doi.org/10.1038/npre.2010.4626.1>
- [12] K. Belhajjame, J. Zhao, D. Garijo, K. Hettne, R. Palma, Ó. Corcho, ... and C. Goble, "The Research Object Suite of Ontologies: Sharing and Exchanging Research Data and Methods on the Open Web," arXiv:1401.4307, January 2014. <https://doi.org/10.48550/arXiv.1401.4307>
- [13] S. Soiland-Reyes, P. Sefton, M. Crosas, L. J. Castro, F. Coppens, J. M. Fernández, ... and C. Goble, "Packaging Research Artefacts with RO-Crate," *Data Science*, Vol. 5, No. 2, pp. 97-138, 2022. <https://doi.org/10.3233/DS-210053>
- [14] A. Bardi and P. Manghi, "Enhanced Publications: Data Models and Information Systems," *LIBER Quarterly: The Journal of the Association of European Research Libraries*, Vol. 23, No. 4, pp. 240-273, 2014. <https://doi.org/10.18352/lq.8445>
- [15] A. Bardi and P. Manghi, "A Framework Supporting the Shift from Traditional Digital Publications to Enhanced Publications," *D-Lib Magazine*, Vol. 21, No. 1/2, 2015. <https://doi.org/10.1045/january2015-bardi>
- [16] A. Stein Kenfield, L. Woolcott, S. Tompson, E. J. Kelly, A. Shiri, C. Muglia, ... and M. E. Morales, "Toward a Definition of Digital Object Reuse," *Digital Library Perspectives*, Vol. 38, No. 3, pp. 378-394, 2022. <https://doi.org/10.1108/DLP-06-2021-0044>
- [17] Z. Yuan, D. H. T. That, S. Kothari, G. Fils, and T. Malik, "Utilizing Provenance in Reusable Research Objects," *Informatics*, Vol. 5, No. 1, 14, 2018. <https://doi.org/10.3390/informatics5010014>
- [18] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, ... and B. Mons, "The FAIR Guiding Principles for Scientific Data Management and Stewardship," *Scientific Data*, Vol. 3, 160018, 2016. <https://doi.org/10.1038/sdata.2016.18>
- [19] A. Prlić and J. B. Procter, "Ten Simple Rules for the Open Development of Scientific Software," *PLoS Computational Biology*, Vol. 8, No. 12, e1002802, 2012. <https://doi.org/10.1371/journal.pcbi.1002802>
- [20] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, ... and P. Wilson, "Best Practices for Scientific Computing," *PLoS Biology*, Vol. 12, No. 1, e1001745, 2014. <https://doi.org/10.1371/journal.pbio.1001745>
- [21] A. M. Smith, D. S. Katz, K. E. Niemeyer, and FORCE11 Software Citation Working Group, "Software Citation Principles," *PeerJ Computer Science*, Vol. 2, e86, 2016. <https://doi.org/10.7717/peerj-cs.86>
- [22] R. C. Jiménez, M. Kuzak, M. Alhamdoosh, M. Barker, B. Batut, M. Borg, ... and S. Crouch, "Four Simple Recommendations to Encourage Best Practices in Research Software," *F1000Research*, Vol. 6, ELIXIR-876, June 2017. <https://doi.org/10.12688/f1000research.11407.1>
- [23] B. D. Lee, "Ten Simple Rules for Documenting Scientific Software," *PLoS Computational Biology*, Vol. 14, No. 12, e1006561, December 2018. <https://doi.org/10.1371/journal.pcbi.1006561>
- [24] M. Barker, N. P. Chue Hong, D. S. Katz, A.-L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, ... and T. Honeyman, "Introducing the FAIR Principles for Research Software," *Scientific Data*, Vol. 9, 622, 2022. <https://doi.org/10.1038/s41597-022-01710-x>
- [25] O. E. Gundersen and S. Kjensmo, "State of the Art: Reproducibility in Artificial Intelligence," in *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans: LA, pp. 1644-1651, 2018. <https://doi.org/10.1609/aaai.v32i1.11503>
- [26] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Larivière, A. Beygelzimer, F. d'Alché-Buc, ... and H. Larochelle, "Improving Reproducibility in Machine Learning Research," *The Journal of Machine Learning Research*, Vol. 22, No. 1, pp. 7459-4778, 2021.
- [27] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, ... and T. Gebru, "Model Cards for Model Reporting," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, Atlanta: GA, pp. 220-229, 2019. <https://doi.org/10.1145/3287560.3287596>
- [28] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford, "Datasheets for Datasets," *Communications of the ACM*, Vol. 64, No. 12, pp. 86-92, 2021. <https://doi.org/10.1145/3458723>
- [29] DataCite Metadata Working Group. DataCite Metadata Schema Documentation for the Publication and Citation of Research Data and Other Research Outputs, Version 4.6 [Internet]. Available: <https://doi.org/10.14454/mzv1-5b55>.
- [30] DataCite Metadata Working Group. DataCite Metadata Schema for the Publication and Citation of Research Data and Other Research Outputs, Version 4.6 [Internet]. Available: <https://doi.org/10.14454/csba-e454>.
- [31] CodeMeta Project. The CodeMeta Project [Internet]. Available: <https://codemeta.github.io/>.
- [32] Schema.org Community Group. Schema.org [Internet]. Available: <https://schema.org/>.
- [33] H. Cousijn, A. Kenall, E. Ganley, M. Harrison, D. Kernohan, T. Lemberger, ... and T. Clark, "A Data

Citation Roadmap for Scientific Publishers,” *Scientific Data*, Vol. 5, 180259, 2018. <https://doi.org/10.1038/sdata.2018.259>

- [34] S.-A. Sansone, P. McQuilton, P. Rocca-Serra, A. Gonzalez-Beltran, M. Izzo, A. L. Lister, ... and the FAIRsharing Community, “FAIRsharing as a Community Approach to Standards, Repositories and Policies,” *Nature Biotechnology*, Vol. 37, pp. 358-367, 2019. <https://doi.org/10.1038/s41587-019-0080-8>
- [35] D. Shotton, “Semantic Publishing: The Coming Revolution in Scientific Journal Publishing,” *Learned Publishing*, Vol. 22, No. 2, pp. 85-94, 2009. <https://doi.org/10.1087/2009202>
- [36] B. Mons, C. Neylon, J. Velterop, M. Dumontier, L. O. Bonino da Silva Santos, and M. Wilkinson, “Cloudy, Increasingly FAIR,” *Information Services & Use*, Vol. 37, No. 1, pp. 49-56, 2017. <https://doi.org/10.3233/ISU-170824>



강근영(Keun Young Kang)

2014년 : 연세대학교 대학원

(문헌정보학 석사)

2025년 : 연세대학교 대학원

(문헌정보학 박사)

2025년 ~ 현재 : 연세대학교 대학도서관발전연구소 연구원

※ 관심분야 : 텍스트마이닝(Text-mining), 자연어처리(Natural Language Processing), 인간-AI 상호작용(Human-AI interaction)