논문 2025-3-4 http://dx.doi.org/10.29056/jsav.2025.09.04

LLM 기반의 조항 이해 및 속성 추출을 활용한 라이선스 양립성 판단 모델 연구

김동완*, 박경엽*, 조용준*, 신동명*†

A Study on a License Compatibility Determination Model Using LLM-Based Clause Understanding and Attribute Extraction

Dong-Wan kim*, Kyung-Yeob Park*, YoungJoon Joe*, Dong-Myung Shin*†

요 약

OSS의 확산에 따라 라이선스 간 충돌이 주요한 법적·기술적 문제로 대두되고 있다. 그러나 기존 컴플라이언스 도구는 주로 라이선스 탐지에 머무르며, 양립성 판단은 전문가의 수작업에 의존한다. 실제 양립성은 라이선스 전문의 조항뿐 아니라 결합 방식과 배포 형태 등 실행 문맥이 함께 결정하므로 단순 자동화가 어렵다. 본 논문에서는 외부 규칙 엔진을 배제하고 규칙 지식을 내재화한 LLM을 학습하여, 라이선스 조항을 법적·기술적·맥락적 속성으로 분해하고 각 항목을 삼진 벡터(+1, 0, -1)로 예측한 뒤, 예측 속성과 실행 문맥을 결합해 양립성 여부를 판단하는 모델을 제안한다. 입력에는 라이선스 전문과 소프트웨어 간 결합·배포 정보가 포함되며, 모델은 문장 단위 근거 매핑을 함께 제공하여 판단 결과에 대한 설명 가능성을 제공한다. 본 연구는 LLM의 법률 해석력과 문맥 인식 능력을 결합하여 명시적 규칙 엔진에 의존하지 않는 양립성 판정 체계를 제안하고, OSS 컴플라이언스 자동화를 위한 기반을 마련한다.

Abstract

As OSS adoption grows, license conflicts emerge as critical legal and technical challenges. Existing compliance tools mainly focus on license identification and still rely on manual expert review for compatibility decisions. In practice, compatibility depends not only on license clauses but also on operational context, which makes automation difficult. We propose a structured approach in which an LLM internalizes licensing rules without an external rule engine, predicts legal/technical/contextual attributes as ternary vectors (+1, 0, -1), and directly decides compatibility given the context. The input includes full license texts and integration/distribution details, and the model outputs sentence-level rationales to ensure explainability. These results indicate that combining legal reasoning and contextual awareness in LLMs can enable compatibility judgments without explicit rule engines, providing a foundation for automated OSS license compliance.

한글키워드 : 오픈소스 소프트웨어, 대규모 언어모델, 라이선스 충돌 분석 컴플라이언스 자동화, 다중과제 학습

keywords : OSS, LLM, License Conflict Analysis, Compliance Automatio, Multi-Task Learning

* 엘에스웨어㈜
† 교신저자: 신동명(email: roland@lsware.co.kr)

접수일자: 2025.08.29. 심사완료: 2025.09.16.
게재확정: 2025.09.20.

1. 서 론

최근 다양한 산업 분야에서 오픈소스 소프트웨어(OSS, Open Source Software)가 핵심 기술로 부상하였다. 모듈화된 서비스 구조를 지향하는 마이크로서비스 아키텍처(MSA, Microservice Architecture)와 클라우드 네이티브(Cloud Native) 환경의 확산은 OSS의 재사용과 통합을 촉진하였다[1]. 그러나 OSS 활용 확산에 따라 이종 라이선스 간 법적 충돌이 심화되고[2], 소스코드 공개의 의무나 재라이선싱(relicensing) 가능 여부, 특허권 충돌 등 다양한 컴플라이언스 위험이 발생하였다[3]. 예를 들어, 2024년 독일 법원에서 AVM FRITZ!Box 4020 공유기 펌웨어의 LGPL(GNU Lesser General Public License) 라이브러리 소스코드와 설치 스크립트 제공 의무에 대한 소송이 제기되었다. 이에 법원은 비용 배상을 판결하였으며, 원고는 필요한 자료를 공유받았다[4][5]. 또한 이탈리아 정부 산하기관이 AGPL(Affero General Public License) 기반 프로젝트인 Globaleaks를 수정 및 재배포하는 과정에서 소스코드 공개 및 라이선스 고지 의무를 위반하였다[6]. 이 같은 사건은 OSS가 네트워크를 통한 제공 상황에서도 소스코드 공개 등 의무가 전파될 수 있음을 시사한다. 이와 관련하여 Synopsys의 '2024 OSSRA' 보고서에 따르면, 조사 대상 기업의 53%가 라이선스 충돌 또는 미준수 문제가 있는 OSS 구성 요소를 사용한다고 보고한다[7].

이 같은 결과는 OSS가 다양한 방식으로 통합되는 과정에서 발생하는 복잡한 라이선스 관계와 맥락 차이에서 비롯된다. 소프트웨어 통합 구조는 정적 링크(static linking)뿐만 아니라 동적 링크(dynamic linking), 프로세스 간 통신(IPC, Inter-Process Communication), RESTful API, SaaS(Software as a Service) 배포 등으로 다양화되고 있으며, 결합 방식과 배포 형태에 따라 동일한 라이선스 조합이라도 충돌 여부가 달라질 수 있다[8]. 예를 들어, MIT(Massachusetts Institute of Technology) 라이선스와 GPLv3(GNU General Public License version 3) 라이선스를 적용한 소스코드를 정적으로 결합하는 경우 GPLv3의 카피레프트 조건에 따라 결과물 전체에 소스코드 공개 의무가 확장된다. 반면 API를 통한 분리 구조에서는 소스코드 공개 의무가 발생하지 않는다[9]. 반면, AGPL(Affero General Public License)은 코드 결합이 없더라도 API 호출만으로 네트워크 전파 조항에 따라 소스코드 공개 의무가 발생할 수 있다[10].

그러나 기존 OSS 컴플라이언스 도구는 이러한 맥락 차이를 충분히 반영하지 못한다. 기존 도구나 연구들은 대부분 정적 분석 기반의 라이선스 탐지에 의존하며, 최종 충돌 여부는 전문가의 수작업 해석에 의존한다[11]. 이러한 방식은 기술적 환경이 동적으로 변화하는 상황에서 수작업 판정의 효율성과 일관성을 유지하기 어렵게 한다. 따라서 기존 방식은 복잡한 통합·배포 맥락을 고려한 라이선스 양립성 판단에 근본적인 한계를 가진다[12]. 또한, 카피레프트 전파 조건, 링크 허용 범위, 특허 부여 여부, 재라이선싱 가능성 등 주요 판단 요소를 체계적으로 정형화한 연구도 제한적이다[13].

이에 본 논문에서는 대규모 언어모델(LLM, Large Language Model)과 명시적 규칙 기반 추론을 결합한 속성 기반 라이선스 양립성 판단 모델을 제안한다. 제안 모델은 OSS 라이선스를 법적·기술적·맥락적 속성으로 분해하고, 각 속성을 삼진 벡터(+1/0/−1)로 구조화한다. LLM은 라이선스 전문과 실행 문맥을 입력받아 속성 벡터를 산출하며, 규칙 기반 추론을 통해 맥락상 유효한 속성을 선별하고 조합에 따른 양립 여부를 판단한다. 제안 방식은 기존의 정적 조항 비교와 수작업 해석 중심 방식이 갖는 한계를 완화하고, 다양한 통합·배포 환경에서의 판정 일관성을 확보한다.

2. 관련 연구

2.1 규칙 기반 오픈소스 라이선스 정적 분석 기법

기존 오픈소스 라이선스 분석은 소스코드, 바이너리, 메타데이터 등에서 라이선스 식별자를 추출한 뒤, 사전에 정의된 규칙에 기반하여 정적 분석을 수행한다. 문자열 매칭, 정규표현식, 토큰화와 같은 텍스트 기반 탐지 기법을 활용하며, 코드 저장소의 LICENSE 파일, 주석 영역, 배포 패키지의 메타데이터를 탐색한다. 대표적인 라이선스 정적 분석 도구는 FOSSology[14], Ninka[15], ScanCode Toolkit[16], SPDX Violation Tool[17] 등이 있다. FOSSology는 소스코드와 바이너리에서 라이선스를 식별하여 SPDX(Software Package Data Exchange) 형식의 결과를 출력한다. Ninka는 주석 내 문장을 사전 정의 패턴과 대조하는 패턴 매칭 기반 도구이다. ScanCode Toolkit은 소스코드와 패키지 의존성을 스캔하여 라이선스 및 취약점 정보 등을 식별하고 출력한다. SPDX Violation Tool은 공개 라이선스 목록과 호환성 매트릭스를 참조하여 종속성 기반으로 라이선스 조합을 점검한다.

이 같은 기존 도구들은 일관된 기준에 따라 라이선스를 식별하지만, 여러 구조적 한계를 지닌다. SPDX 목록에 없는 라이선스를 인식하기 어렵고, 정적·동적 링크, IPC, SaaS와 같은 통합·배포 맥락을 판정에 반영하지 못한다. 또한 '소스코드 공개 의무', '특허 부여·보복'과 같은 조항 의미를 추출하여 해석하기 어렵고, 고정된 호환성 매트릭스에 의존해 복합 환경에서 조건부 양립성 판단이 제한된다. 이러한 기존 도구는 라이선스 식별 이후 호환성 매트릭스, 정책 테이블 등의 외부 규칙 엔진을 별도 모듈로 호출하여 양립성을 판정한다. 이 방식은 규칙의 명시성과 반복성은 제공하지만, 변형 조항이나 비정형 결합 토폴로지에 대한 일반화가 낮고, 규칙셋 유지·버전 동기화 비용이 크다.

2.2 AI 기반 오픈소스 라이선스 조항 분석 기법

정적 분석의 한계를 보완하기 위해, 라이선스 전문을 입력으로 받아 특정 조항과 그 의무 수준을 자동 식별하고 이를 근거로 호환성을 추론하는 AI 기반 접근이 제시되었다. 자연어 처리의 발전과 함께 조항을 정형 데이터로 변환한 뒤 규칙 기반 엔진이나 분류기에 전달하는 하이브리드 구조가 주로 사용된다. 이 방식은 텍스트 수준의 식별을 넘어 조항 의미와 의무 강도를 모델에 반영한다는 점에서 정적 탐지와 구별된다.

AI 기반 오픈소스 라이선스 분석 도구에는 FOSS-LTE[18], DIKE[19], LiDetector[20], LiResolver[21] 등이 있다. FOSS-LTE는 LDA(Latent Dirichlet Allocation) 기반 토픽 모델링으로 라이선스 문서에서 주제별 조항을 추출하고 사전 정의된 의무 수준을 적용한다. 처리 흐름이 단순하여 일관성은 확보되지만, 조항 유형과 의무 수준 구분이 고정되어 새로운 조항이나 복합 조건에 대응하기 어렵다. DIKE는 LSAN(Label-Specific Attention Network)으로 조항을 식별하고 ALBERT 기반 분류기로 의무 수준을 판별하는 이중 구조로 모듈 분리가 명확하지만, 선행 단계의 오류가 후속 단계로 전파되는 오류 누적 문제가 있다. LiDetector는 Bi-LSTM 기반 명명된 엔티티 인식(NER, Named Entity Recognition)으로 조항을 추출하고, 규칙 및 감성 분석으로 의무 수준을 분류한다. 이러한 구조는 문장 구조 변형에 일정 수준 견고하지만 결합 방식, 배포 형태 등 같은 사용 맥락을 입력에 통합되지 않아 문맥 의존 판단에 한계가 있다. LiResolver는 RoBERTa 기반 문장 임베딩으로 조항 간 관계를 표현하고 SVM (Support Vector Machine) 분류기를 통해 라이선스 계층 구조를 고려한 충돌 탐지를 수행한다. 그러나 학습 데이터가 비공개이거나 제한된 범주에 한정되어 일반화 및 확장성이 부족하다.

AI 기반 분석은 정적 분석에 비해 표현의 유연성과 제한적인 맥락 이해 측면에서 향상되었으나, 법적·기술적·맥락적 속성을 통합적으로 반영하는 구조를 구현하지 못하는 한계가 존재한다. 특히, 원문을 입력하여 최종 라벨을 산출하는 엔드투엔드(end-to-end) 방식은 설명 가능성 부족, 법적 제약 위반 가능성, 데이터 희소성에 대한 취약성, 판단의 재현성 결여 등 구조적 문제를 야기한다. 이러한 한계를 보완하기 위해 최근에는 LLM을 활용한 연구가 시도되고 있다.

2.3 LLM을 활용한 OSS 라이선스 분석

GPT, LLaMA, Claude 등 LLM의 발전은 OSS 라이선스 분석 자동화에 새로운 가능성을 제시하였다. LLM은 대규모 사전학습을 기반으로 법률 문서와 같은 복잡한 문장을 처리하고, 특정 조항과 그 의미를 추론할 수 있다. 또한 단일 입력으로부터 조항(term)과 의무 수준(attitude)을 식별하며, 주어진 조건에 따라 의미를 다르게 해석한다. 이러한 구조는 파이프라인 분할 방식에서 발생하는 단계 간 오류 전파를 감소시킨다.

대표적인 사례로 L3icNexus는 LLaMA 3-8B를 라이선스 도메인 특화 데이터로 학습하여 조항·의무 수준 추출과 충돌 판단을 단일 흐름으로 통합한다[22]. 그러나 실제 소프트웨어 통합 과정에서의 정적 링크, 동적 링크, IPC, API 호출 등 다양한 결합 방식과 배포 구조는 고려하지 않았다. 이에 따라 동일한 라이선스 조합이라도 결합 형태에 따라 결과가 달라지는 상황에 대응하지 못하며, 기술적 맥락을 반영한 해석 절차 또한 제시하지 않았다. 한편, GPT-4, LLaMA 3-70B 등 범용 LLM을 제로 샷(zero-shot) 방식으로 활용한 연구도 진행되었으나, 도메인 특화 학습의 부족으로 인해 관련 없는 조항 생성, 의무 수준 오판 등 환각 현상이 빈번하게 발생하는 한계가 있다[22][23].

LLM 기반 라이선스 분석은 기존 AI 기반 기법에 비해 여러 측면에서 개선이 나타난다. 단일 입력으로 조항과 의무 수준을 동시에 식별하여 단계 간 오류 전파를 줄이고, 표현 변형이나 비정형 문장에도 안정적으로 대응하며, 주어진 맥락을 반영한 조건부 해석을 수행할 수 있다. 그러나 한계는 여전히 존재한다. 도메인 특화 학습 부족으로 실제 적용 정확도가 저하되며, 추출 결과를 법적·기술적·맥락적 속성으로 정형화하여 논리적으로 결합하는 과정은 미비하다. 또한, 일관된 판정을 뒷받침할 명시적 규칙 기반 판단 체계가 구축되지 않았다.

3. 라이선스 양립성 판단을 위한 주요 속성 정의

본 논문에서는 라이선스 양립성 판단을 위해 OSS 라이선스의 다양한 조건을 법적(Legal), 기술적(Technical), 맥락적(Contextual) 속성으로 구분한 27개 속성 체계를 정의한다. 제안 체계는 기존 연구[24][25][26][27]에서 제시한 라이선스 분류 기준과 충돌 요인 분석을 기반으로 하며, 각 범주 내 속성은 실제 통합 과정에서 양립성에 영향을 미치는 요소를 중심으로 도출한다.

3.1 법적 속성

라이선스 양립성 판단의 첫 번째 범주는 법적 속성(Legal Attributes)이다. 법적 속성은 OSS 라이선스 조항 중 충돌 가능성이 높은 법률적 조건을 정형화한 속성 집합으로, 파생 저작물 조건, 특허 정책, 재배포 의무, 상표권·관할 조항 등 실제 통합 시 직접적인 충돌 요인이 될 수 있는 요소를 포함한다. 본 논문에서는 표 1과 같이 14개 법적 속성과 각 항목의 의미를 정의한다. 각 속성은 필수(+1), 무관(0), 금지/상충(−1)의 삼진값으로 표현하며, 해석 기준은 다음과 같다.

표 1. OSS 라이선스 양립성 분석을 위한 법적 속성 목록 및 정의
Table 1. Legal Attribute Definitions for Analyzing Compatibility Between OSS Licenses

속성 코드	속성 이름	의미(+1=필수, 0=무관, −1=금지/예외)	규칙 분류
F1	Strong Copyleft	파생 저작물에 동일 라이선스 적용 요구(+1), 언급 없음(0), 동일 라이선스 금지/예외(−1)	의무형
F2	Weak Copyleft	링크/모듈 수준 동일 라이선스 요구(+1), 언급 없음(0), 금지/예외(−1)	의무형
F3	Permissive Grant	재라이선스/병합 허용(+1), 언급 없음(0), 제한/금지(−1)	허용형
F4	Linking Restriction	링크 방식에 따라 전파 의무 명시(+1), 언급 없음(0), 링크 자유 보장(−1)	의무형
F5	Patent Grant	특허 권한 부여 포함(+1), 언급 없음(0), 특허 사용 금지/보장 부재(−1)	정책형
F6	Patent Retaliation	특허 소송 시 라이선스 종료/보복 조항(+1), 언급 없음(0), 보복 금지/부재(−1)	정책형
F7	Modification Allowed	수정 허용(+1), 언급 없음(0), 수정 금지/제한(−1)	허용형
F8	Distribution Obligation	배포 시 공개 의무(+1), 언급 없음(0), 공개 의무 부재/금지(−1)	의무형
F9	Attribution Required	저작권자 표시 요구(+1), 언급 없음(0), 비요구/금지(−1)	의무형
F10	Jurisdiction-Specific	특정 관할 강제(+1), 중립/언급 없음(0), 특정 관할 금지/상충(−1)	정책형
F11	Trademark Use Restriction	상표 사용 제한(+1), 언급 없음(0), 자유 사용 보장(−1)	정책형
F12	Warranty Disclaimer	보증 부인(+1), 언급 없음(0), 보증 제공/제한 요구(−1)	허용형
F13	Notice Requirement	NOTICE/고지 의무(+1), 언급 없음(0), 고지 금지/무요구(−1)	의무형
F14	License Compatibility Clause	호환성/비호환성 명시(+1/−1), 언급 없음(0) ※대상 라이선스 정보 없음	정책형

- **+1 (필수):** 해당 조항이 명시되어야 하며, 충족되지 않으면 양립성이 저해됨
- **0 (무관):** 해당 조항의 존재 여부가 양립성 판단에 영향을 미치지 않음
- **-1 (금지 또는 예외):** 해당 조항이 존재하면 결합할 수 없거나 다른 라이선스와 상충함

특수 해석이 필요한 항목은 다음과 같다.

- **F10(관할 조항):** +1=특정 관할 강제, 0=언급 없음, −1=특정 관할 금지 또는 상충
- **F14(호환성 조항):** −1은 비호환 명시를 의미하며 일반적인 '금지/예외'와 구분

법적 속성에서 F1(Strong Copyleft)은 파생 저작물에 동일한 라이선스를 적용하도록 요구하는 조항이 명시된 경우, +1로 평가한다. 반대로 동일 라이선스 적용을 명시적으로 금지하거나 예외를 규정한 경우, −1, 관련 조항이 없거나 적용 의무가 명확하지 않은 경우, 0으로 분류한다. F4(Linking Restriction)은 정적 또는 동적 링크 방식에 따라 전파 조건이 달라질 수 있으며, 특정 결합에서의 의무 전파 가능성을 뜻한다. F5(Patent Grant)와 F6(Patent Retaliation)은 각각 특허 권한 부여와 특허 분쟁 대응 조건을 나타내며, 저작권 외 법적 책임 발생 여부를 판단

한다. F10(Jurisdiction-Specific)은 특정 관할 또는 국가의 법 적용을 요구하는 조항으로, 국제 환경의 일관성에 영향을 미친다. F13(Notice Requirement)와 F14(License Compatibility Clause)는 소스코드 배포와 라이선스 결합 시, 고지 의무와 결합 가능성 명시 여부를 나타낸다.

이러한 법적 속성들은 LLM 기반의 구조화 과정에서 입력 항목으로 활용되며, 각 조항의 존재 여부와 의미를 자동으로 추론된 속성 벡터로 표현한다. 이 속성 벡터는 링크 방식, 배포 조건 등 사용 맥락과 결합되어 라이선스 간 법적 충돌 가능성을 정량적으로 분석하는 데 사용된다. 제안 속성 체계는 OSS 통합 환경에서의 충돌 판정 자동화와 준수 검토의 기준으로 기능한다.

3.2 기술적 속성

오픈소스 라이선스의 양립성은 법적 조항뿐 아니라 실제 소프트웨어의 결합 방식에도 영향을 받는다. 기술적 속성은 OSS 간 통합 시 적용되는 링크 방식, 호출 구조, 배포 형태 등 기술적 연결 조건에 따라 라이선스 전파 범위와 의무 조건을 판단하는 기준이다. 본 논문에서는 표 2와 같이 7개의 기술적 속성과 의미를 정의한다. 각 속성은 특정 기술 조건에서 조항 적용 또는 전파를 요구(+1), 무관(0), 또는 적용되지 않음·예외적 허용(−1)을 나타내는 삼진 벡터 형태로 표현한다. 이는 단순한 조항 존재 여부를 넘어, 정적·동적 링크, 실행 환경, SaaS 제공 여부 등과 같은 사용 맥락에 따라 다르게 해석될 수 있다. 표 2에서 T1(Static Linking Propagates)은 정적 링크로 결합된 소프트웨어가 파생 저작물로 간주되어 동일 라이선스 적용이 필요한지를 평가한다. 요구가 명시된 경우 +1, 언급이 없거나 해석이 필요한 경우 0, 병합이 아님을 명시한 경우 −1로 분류한다. T2(Dynamic Linking Propagates)는 동적 링크에서 동일한 전파 조건이 적용되는지를 평가하며, 일반적으로 정적 링크보다 완화된 해석이 적용된다. T3(IPC Triggers Propagation)은 IPC 구조에서의 전파 여부를, T4(Containerization Propagates)는 컨테이너 단위 배포 시 구성 요소 간 결합이 전파 의무를 유발하는지를 평가한다. T5(SaaS Provision Triggers License)는 SaaS 제공이 배포에 해당하는지를, T6(Build Tool Affects Output)은 빌

표 2. OSS 라이선스 양립성 분석을 위한 기술적 속성 목록 및 정의
Table 2. Technical Attribute Definitions for Analyzing Compatibility Between OSS Licenses

속성 코드	속성 이름	의미 (+1=전파/트리거 발생, 0=무관/불명확, −1=전파 없음/예외)	규칙 분류
T1	Static Linking Propagates	정적 링크 시 전파(+1), 불명확(0), 전파 없음(−1)	의무형
T2	Dynamic Linking Propagates	동적 링크 시 전파(+1), 불명확(0), 전파 없음(−1)	의무형
T3	IPC Triggers Propagation	IPC 연결 시 전파(+1), 불명확(0), 전파 없음(−1)	의무형
T4	Containerization Propagates	컨테이너 내부 병합 전파(+1), 불명확(0), 전파 없음(−1)	의무형
T5	SaaS Provision Triggers License	SaaS 제공 시 전파(+1), 불명확(0), 전파 없음(−1)	의무형
T6	Build Tool Affects Output	빌드 도구 사용 시 전파(+1), 불명확(0), 전파 없음(−1)	의무형
T7	Plugin Architecture Propagates	플러그인 구조 전파(+1), 불명확(0), 전파 없음(−1)	의무형

드 도구 사용이 산출물의 라이선스에 영향을 미치는지를 판단한다. T7(Plugin Architecture Propagates)은 플러그인 구조에서 모듈 간 결합이 병합으로 간주 되는지 평가한다.

이러한 기술적 속성은 구조화 모델의 입력 항목으로 활용되며, 각 라이선스 조항이 기술 조건을 어떻게 규정하는지를 추론하는 데 사용된다. 이후 판단 단계에서는 링크 방식, 실행 환경, 배포 구조 등과 결합하여 라이선스 간 전파 조건 충돌 여부를 정량적으로 분석한다. 제안 속성 체계는 고정값이 아닌 맥락 반응형(Context-Sensitive) 구조를 지니며, 실제 통합 조건에 따른 양립성 판단의 정확도를 높인다.

3.3 맥락적 속성(Contextual Attributes)

오픈소스 라이선스의 양립성은 법적 조항이나 기술적 통합 방식뿐 아니라 제도적 승인 여부, 버전 간 호환성 등 외부 요인의 영향을 받는다. 이러한 요인은 실제 운영 환경의 제약 조건을 반영하며, 본 논문에서는 이를 구조화하여 표 3과 같이 6개의 맥락적 속성과 의미를 정의한다. 각 속성은 공식 승인/채택(+1), 불확실/해석 필요(0), 명시적 비승인(−1)의 삼진 값으로 표현한다. 표 3에서 C1(SPDX Listed)은 해당 라이선스가 SPDX License List에 등재된 경우 +1, 등재 여부가 불명확하거나 매핑이 모호한 경우 0, 공식 식별자가 없거나 'deprecated / withdrawn' 상태인 경우, −1로 평가한다. C2(FSF GPL-Compatible)는 FSF가 GPL과의 호환성을 인정하면 +1, 호환 불가를 명시하면 −1로 분류한다. C3(OSI Approved)은 OSI의 공식 승인 여부를 기준으로 하며, 승인 시 +1, 거절 시 −1로 평가한다. C4(License Frequently Enforced)는 해당 라이선스에 대한 판례 또는 강제 집행 사례가 있는 경우 +1, 유효성이 부정된 경우, −1로 분류한다. C5(Project Policy Compatible)는 주요 오픈소스 프로젝트의 채택 여부를 기준으로 하며, 명시적 채택은 +1, 명시적 배제는 −1로 평가한다. C6(Version Interoperability Risk)은 동일 라이선스 내 버전 간 호환성을 나타낸다. 예를 들어 GPLv2-only와 GPLv3과 같이 결합이 불가능한 경우 −1, 충돌 가능성이 없는 경우, +1로 평가한다.

이러한 맥락적 속성은 법적·기술적 속성과 별도로 평가하며, LLM 기반 속성 추론 과정에서 보조 지표로 활용된다. 이를 통해 정책적 제약, 사회적 수용성과 같은 현실 요인을 정량적으로 반영한다.

표 3. OSS 라이선스 양립성 분석을 위한 맥락적 속성 목록 및 정의
Table 3. Contextual Attribute Definitions for Analyzing Compatibility Between OSS Licenses

속성 코드	속성 이름	의미 (+1=긍정/채택, 0=무관/불명확, −1=부정/비호환. C6은 +1=리스크 없음, −1=리스크 존재)	규칙 분류
C1	SPDX Listed	SPDX 공식 식별자 존재(+1), 커스텀·복합 라이선스 등으로 판단 보류, 불명확 (0), 공식 식별자 부재 또는 부정 등재 상태 (-1)	맥락형
C2	FSF GPL-Compatible	FSF 기준 GPL 호환(+1), 불명확(0), 비호환(−1)	맥락형
C3	OSI Approved	OSI 승인(+1), 불명확(0), 거절(−1)	맥락형
C4	License Frequently Enforced	법적 판례/집행 사례 존재(+1), 불명확(0), 무효/부정 판례(−1)	맥락형
C5	Project Policy Compatible	주요 프로젝트 정책상 채택(+1), 중립/불명확(0), 배제(−1)	맥락형
C6	Version Interoperability Risk	버전 간 충돌 위험 없음(+1), 불명확(0), 위험 존재(−1) ※부호 방향 주의	맥락형

4. 제안 기법

4.1 제안 모델의 개요

본 논문에서 제안하는 OSS 라이선스 양립성 판단 모델은 속성 기반 분석과 규칙 지식을 학습하는 LLM을 결합한 구조적 방식이다. 입력은 라이선스 전문, 메타데이터, 실행 문맥으로 구성되며, 제안 모델은 입력값을 27차원 삼진 속성 벡터로 변환한 뒤 양립성 여부를 판단한다.

제안 모델은 외부 규칙 엔진을 사용하지 않고, 언어모델이 규칙 지식을 학습하여 속성 추출, 판단, 근거 선택을 통합적으로 수행한다. 본 논문에서는 OSS 라이선스 데이터 세트에 대해 gpt-oss[28]에 파인튜닝(Fine-tuning)을 수행하였다. 모델은 라이선스 전문과 속성 라벨링 데이터를 학습하고, 이를 통해 조항 단서와 속성값의 대응 관계를 모델 내부에 반영한다. 또한, 속성과 근거 문장을 활용하여 최종 판단을 수행한다.

본 논문에서 제안하는 모델의 구조는 그림 1과 같다. 두 라이선스 전문과 링크 방식, 결합 구조, 배포 방식 등의 실행 문맥을 입력값으로 받아 문장 단위 분석을 통해 속성과 근거 문장을 도출한다. 이후 속성과 문맥 정보를 결합하여 최종 판단 결과를 생성한다.

제안 모델은 복잡한 조항 표현이나 비정형 문장을 처리하며, 실행 문맥에 따른 조건부 해석을 수행한다. 이를 통해 기존 규칙 기반 접근법의 제약을 보완하고, OSS 활용 환경에서 발생하는 라이선스 충돌을 판단한다.

4.2 입력 구조 및 속성 표현 방식

모델의 입력은 라이선스 메타데이터, 라이선스 전문, 속성 벡터, 실행 문맥의 네 요소로 이루어진 직렬화된 JSON 형태이다. 메타데이터에는 비

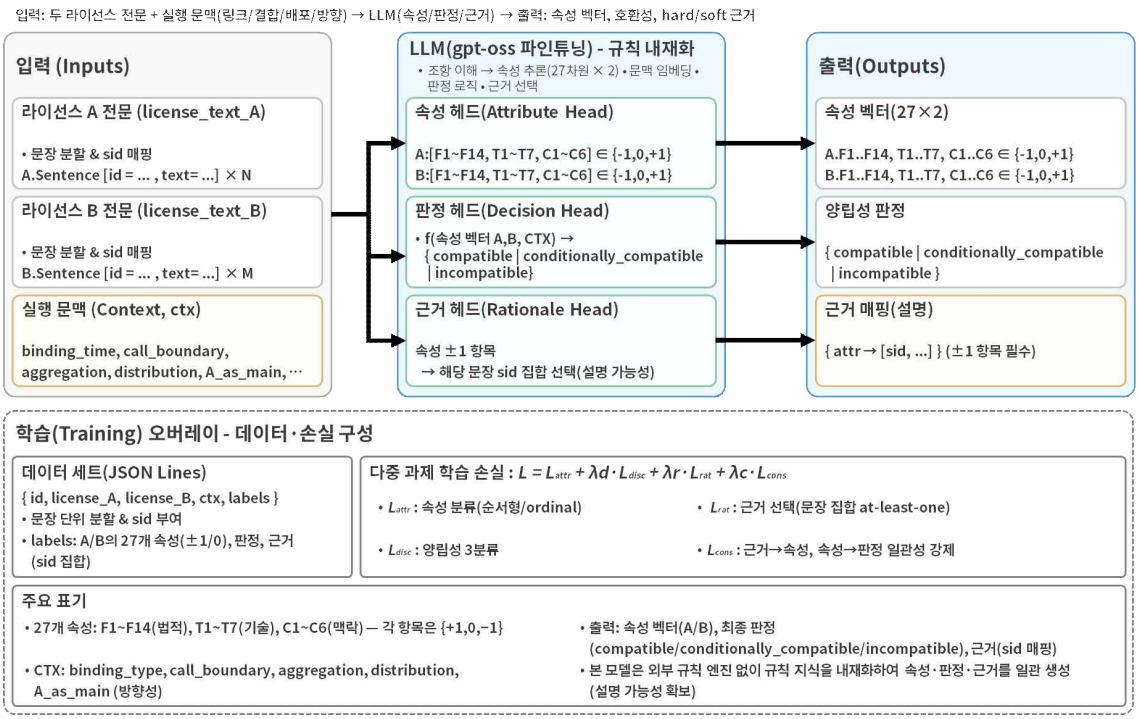


그림 1. 제안 모델 개요 - LLM 기반 속성 추출 · 문맥 통합 · 양립성 판정

Fig. 1. Overview of the proposed Model - LLM-based attribute extraction, context integration, and compatibility determination

교 대상 라이선스의 정식 명칭, 버전, SPDX 식별자가 포함된다. 버전 표기는 -only, -or-later와 같은 구분을 반영하여 적용 범위와 조건을 해석하는 기준으로 사용한다.

라이선스 전문은 license_text_A와 license_text_B 항목에 저장되며, 법적, 기술적, 맥락적 요소를 포함한다. 제안 모델은 이를 바탕으로 속성값을 추론하거나 기존 라벨을 검증한다. 전문은 속성 벡터 생성뿐 아니라 규칙 적용 과정에서 문맥 해석과 예외 조항 확인에도 활용된다.

속성 벡터는 법적 속성(F1~F14), 기술적 속성(T1~T7), 맥락적 속성(C1~C6)으로 이루어지며, 각 항목은 +1, 0, −1의 삼진 값으로 표현된다. 예외적으로 F14의 −1은 '비호환 명시', C6의 +1은 '리스크 없음', −1은 '리스크 존재'를 의미한다. 속성 벡터는 학습 데이터에서 제공되거나, 언어모델이 전문으로부터 직접 생성한다.

실행 문맥은 바인딩 시점(binding_time), 결합 구조(aggregation), 배포 여부(distribution), 호출 경계(call_boundary), 분석 주체(A_as_main), 으로 정의된다. 동일한 속성 벡터라도 실행 문맥이 달라지면 최종 판단 결과가 달라질 수 있다.

4.3 라이선스 양립성 판단 규칙

라이선스 양립성 판단 규칙의 기본 전제는 라이선스 A가 적용된 소프트웨어에 라이선스 B를 결합하는 방향이다. 판단 규칙은 27개 속성 벡터와 관계, 실행 문맥 조건을 고려한다.

판단 결과는 '확정적 충돌', '조건부 충돌', '양립 가능'의 세 범주로 나눈다. 확정적 충돌은 규칙에 따라 양립 불가로 판단되는 경우이다. 예를 들어, 한쪽이 특정 속성을 필수(+1)로 요구하고 다른 쪽이 동일 항목을 금지(−1)하는 경우, 강제 전파 속성(F1)을 요구하나 최종 라이선스가 이를 충족하지 못하는 경우, 서비스 전파 트리거(T5)가 활성화되었지만, F1을 충족하지 못하는 경우, 호환성 조항(F14)이 명시적으로 호환되지 않음을 규정한 경우이다. '조건부 충돌'은 속성 벡터값이 일치하지 않지만, 양립 불가로 단정할 수 없음을 의미한다. 예를 들어, 특허 보장 부재(F5), 특허 보복(F6)과 같은 정책형 속성의 불일치, 허용형 속성(F3, F7, F12)의 상호 충돌, 맥락적 속성(C1~C6)의 부정, F14의 모호한 표현 등이 이에 속한다. 프로젝트 정책, 기관 기준, 실행 환경 해석에 따라 수용 가능성이 달라질 수 있으므로 판단 단계에서는 조건부 충돌로 분류한다. '양립 가능'은 두 범주에 해당하지 않고 결합 및 배포 과정에서 충돌 위험이 없는 경우다.

평가 절차는 실행 문맥에 따라 확인 대상 속성을 제한한다. 링크 방식이 IPC인 경우 T1, T2, F7, F8을 제외하며, 정적 링크에서는 T2, 동적 링크에서는 T1을 제외한다. 결합 구조가 분리 배치라면 F1, F3, F7, F14를 제외하며, 배포가 없는 경우 F4, F11, F12, F13, T4를 제외한다. 실행 환경이 서비스 제공인 경우, F5, F6, F10, F14, T5만 유지한다. 실행 문맥에 따라 유효한 속성 벡터는 방향성에 따라 비교한다. 포함되는 라이선스가 특정 속성을 필수로 요구하고 포함하는 라이선스가 이를 금지하면 확정적 충돌로 판단한다. 다음으로 강제 전파 속성(F1)과 서비스 전파 트리거(T5)의 충족 여부를 확인한다. 포함되는 측에서 F1이 참이면 최종 적용 라이선스는 이를 충족해야 하며, 서비스 제공 환경에서 T5가 참이면 최종 적용 라이선스 역시 F1을 포함해야 한다. 위 과정에서 확정적 충돌 항목이 존재하면 확정적 충돌로 판단하며, 조건부 충돌 항목이 있다면 조건부 충돌로 간주한다. 두 경우 모두 해당하지 않으면 양립이 가능한 경우이다.

4.4 모델 구성

제안 모델은 외부 규칙 엔진에 의존하지 않으며, LLM이 규칙을 학습하여 속성 예측과 양립성

판단을 동시에 수행하는 구조를 갖는다. 입력은 두 라이선스의 전문과 실행 문맥이며, 결합 방향은 실행 문맥의 'A_as_main' 값으로 지정한다. 라이선스 전문은 문장 단위로 분할하고 각 문장에 고유 식별자(sid)를 매핑한다.

제안 모델의 백본 네트워크는 지시형 디코더 계열의 GPT 아키텍처를 기반으로 한 gpt-oss이다, 본 논문에서는 이를 파인 튜닝하여 속성 헤드, 판정 헤드, 근거 헤드의 세 가지 출력을 학습한다. 속성 헤드는 각 라이선스에 대해 27개 속성의 예측값을 출력한다. 판단 헤드는 속성 벡터와 실행 문맥을 종합하여 최종 양립성 결과를 판단한다. 근거 헤드는 속성값이 ±1인 항목에 대응하는 근거 문장을 선택하여 출력한다.

제안 모델은 이와 같은 통합 구조를 통해 외부 규칙 엔진 없이 속성, 판정, 근거를 일관되게 생성한다. 이를 통해 판단 과정의 투명성과 설명 가능성, 재현 가능성을 제공한다.

4.5 데이터 세트 구성 및 처리 방식

본 논문에서 사용하는 데이터세트는 라이선스 전문(A, B)과 실행 문맥(CTX)으로 이루어진 입력 단위로 구성된다. A와 B는 비교 대상 라이선스의 전문과 메타데이터이며, CTX는 이전 절에서 정의한 다섯 가지 실행 문맥 키(binding_time, call_boundary, aggregation, distribution, A_as_main)로 이루어진다. 메타데이터에는 SPDX 식별자, 정식 명칭, 텍스트 해시가 포함되며, 해시는 데이터 분할 시 중복을 차단하는 기준으로 사용된다.

라이선스 전문은 문장 단위로 분할하고 각 문장에는 고유 식별자(sid)를 부여한다. 분할 규칙은 약어, 조항 표기, 인용부호 내 마침표, 부속 문서 등을 예외 처리하여 문맥 단절을 방지한다.

속성 라벨은 27개 항목에 대해 '+1, 0, −1'값으로 정의한다. '+1'은 요구 또는 발동, '0'은 무관 혹은 불명확, '−1'은 금지 혹은 비 호환을 의미한다. ±1로 표기된 속성은 하나 이상의 근거 문장과 연결된다. 최종 라벨은 양측 속성 벡터, 판단 결과, 속성별 근거 매핑으로 구성된다.

데이터 전처리 과정은 규칙 기반 문장 분할, 라이선스 쌍 생성, 실행 문맥 샘플링, 방향성 전환으로 이루어진다. 데이터는 학습, 개발, 평가 세트로 분할되며, 동일한 라이선스 전문이 서로 다른 세트에 포함되지 않도록 SPDX 식별자와 텍스트 해시를 기준으로 중복을 차단한다.

최종 데이터 세트는 JSON Lines 형식이며, 각 샘플은 A, B, CTX, 라벨 블록으로 구성된다.

4.6 학습 방법 및 실험 설정

본 연구의 학습 목표는 라이선스 A와 B의 전문과 실행 문맥을 입력으로 받아 속성 예측, 양립성 판단, 근거 선택을 동시에 수행하는 것이다. 기존 end-to-end 방식은 중간 단계 오류가 누적되는 문제가 있으므로, 제안 모델은 단계별 손실을 포함한 다중 과제 학습(multi-task learning) 방식을 사용한다. 모든 학습 데이터는 전면 매핑 구조로 설계되었으며, 속성값이 ±1인 항목에는 반드시 하나 이상의 근거 문장을 포함한다. 최종 판정 레이블은 확정적 충돌, 조건부 충돌, 양립 가능의 세 범주로 학습·평가된다. 목적함수는 식(1)과 같이 네 가지 손실 항의 선형 결합으로 정의된다.

$$L = L_{attr} + \lambda_d L_{disc} + \lambda_r L_{rat} + \lambda_c L_{cons}$$ 식(1)

- L_{attr} **(속성 예측 손실):** 각 라이선스의 27개 속성을 {-1, 0, +1} 범주로 분류하며, 순서를 반영하는 ordinal loss를 적용한다. 임계값 τ_1, τ_2에 대해, L_{attr} 는 식(2)와 같으며 음이항 로그우도는 L_{attr} 이다.

$$P(y_k \le -1) = \sigma(\tau_1 - z_k),$$
$$P(y_k \le 0) = \sigma(\tau_2 - z_k), with\ \tau_1 < \tau_2 \quad 식(2)$$

- L_{disc}**(양립성 판단 손실):** 속성의 소프트 출력과 실행 문맥 임베딩을 입력으로 사용하며, 세 분류 교차 엔트로피 손실을 적용한다.

- L_{rat}**(근거 선택 손실):** 속성값이 ±1인 경우 정답 근거 집합 $G_k \subset \{1, ..., N\}$에 대해, 문장별 예측 확률 $\hat{r}_{k,i}$로 식(3)을 사용한다. 속성이 0인 경우는 마스킹하여 손실에 포함하지 않으며, 복수 정답 시 at-least-one 기준으로 평가한다.

$$L_{rat} = \sum_{k:|y_k|} \left(-\sum_{i \in G_k} \log \hat{r}_{k,i} - \sum_{i \notin G_k} \log(1 - \hat{r}_{k,i}) \right) \quad 식(3)$$

- L_{cons}**(일관성 손실):** $L_{cons(attr)}$(근거→속성) 및 $L_{cons(disc)}$(속성→판단) 간 합치성을 강제한다. $L_{cons(attr)}$ 합치는 선택된 근거들의 평균 로짓 $\bar{e}_k$ 와 속성 로짓 z_k가 같은 방향을 가리키도록 마진을 두며 이는 식(4)와 같다. 이때, $y_k = +1$이면 $\bar{e}_k + z_k$가 충분히 크도록, $y_k = -1$ 이면 충분히 작도록 유도한다 ($m > 0$은 마진).

$$L_{cons(attr)} = \sum_{k:|y_k|=1} \max\left(0, m - sign(y_k) \cdot (\bar{e}_k + z_k)\right) \quad 식(4)$$

$L_{cons(disc)}$ 합치는 속성 규칙에 비추어 기대되는 판정 보조 타깃 $\tilde{c}$를 만들어, 판정 로짓 ν에 보조 CE 손실을 더한다.

$$L_{cons(disc)} = -\log softmax(\nu)_{\tilde{c}}, \quad ...식(5)$$
$$L_{cons} = L_{cons(attr)} + L_{cons(disc)}$$

가중치 $\lambda_d, \lambda_r, \lambda_c$는 검증 세트에서 조정하였다.

5. 성능 평가

5.1 성능 평가 환경 및 데이터 세트 구성

평가 환경은 단일 GPU(GeForce RTX 3090, 24GB)에서 PyTorch/Transformers를 기반으로 구축하였다. GPT-OSS-20B를 기반 모델로 하여 QLoRA(4bit)로 미세조정하였고, 정밀도는 fp16을 기본으로 하되 bf16 운용도 가능하도록 구현하였다. 입력은 무손실 슬라이딩 윈도우 전략을 적용해 하나의 샘플을 토큰 길이 4096, 오버랩 256으로 겹치게 분할하여 학습하였고, 프롬프트 구간은 −100 마스킹으로 손실에서 제외하였다. 최적화는 learning rate 2e-4, cosine 스케줄러, warmup 0.03, per-device batch 1, gradient accumulation 8로 고정하고, seed 42로 재현성을 확보하였다. 평가는 모델 컨텍스트 한도 내에서 라이선스 본문·규칙·실행 문맥을 그대로 투입하고, 드물게 generate 실패 시 수동 greedy 폴백을 사용하였다.

데이터 세트는 라이선스 종류 쌍 55종에 실행 문맥(ctx) 시나리오를 평균 16개씩 결합해 총 880 샘플을 구성하였다. 각 샘플은 문장 식별자(sid)가 부여된 라이선스 전문/요약, 27개 속성(F1~F14, T1~T7, C1~C6), 링킹/IPC/배포/집적 형태 등 실행 문맥 메타데이터를 포함한다. 분할은 동일 라이선스 쌍이 교차 누출되지 않도록 pair-stratified 원칙을 적용했고, 학습/검증/시험은 704/88/88로 사용하였다. 라벨은 속성(다중라벨) - 근거(rationale) - 최종 판정(compatible, conditionally_compatible, in compatible)의 멀티

태스크 체계를 따른다. 평가지표로는 속성 매크로 F1(attr_macro_f1), 근거 일치(rat_hit_any, rat_f1_micro), 최종 판정 정확도(2-way, 3-way), 생성 길이 관련 지표를 산출하였다.

5.2 라이선스 충돌 판단 정확도 측정

학습은 1 - 10 epoch 범위에서 진행되었다. 그림 2(Validation Loss vs 2-way Accuracy) 에서 보이듯 검증 손실은 학습이 진행될수록 완만히 하강했고, 2-way 정확도는 전 구간에서 높은 수준을 유지하며 점차 소폭 상승하는 경향을 보였다. 이는 모델이 규칙과 문맥을 안정적으로 내재화함에 따라 이진(허용, 비허용) 판정의 일관성이 빠르게 확보되었다는 해석이 가능하다.

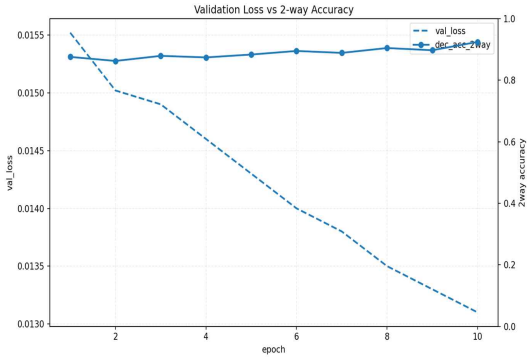


그림 2. Validation loss와 2-way 정확도의 epoch별 추이

Fig. 2. Trends of Validation Loss and 2-way Accuracy over Epochs

그림 3 (2-way vs 3-way Decision Accuracy)은 최종 판정의 정밀도를 비교해 보여준다. 2-way는 학습 내내 0.87 - 0.93 구간의 안정대를 형성한 반면, 3-way는 초반 0.73대에서 후반 0.86 수준으로 유의미하게 개선되었다. 특히 conditional 범주에서의 경계 인식이 강화되면서 compatible, conditional, conditional, incompatible 간 오인식이 점차 감소하는 추이가 관찰되었다.

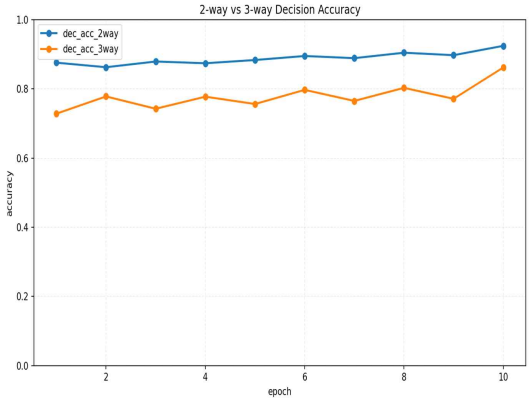


그림 3. 2-way와 3-way 최종 판정 정확도의 epoch별 비교

Figure 3. Comparison of 2-way and 3-way final determination accuracy over epochs

그림 4(Attribute F1 & Rationale Metrics)는 속성 인식(attr_macro_f1)과 근거 지표(rat_hit_any, rat_f1_micro)의 동반 개선을 보여준다. attr_macro_f1은 초기 0.68대에서 최종 0.82 내외로 상승했고, rat_hit_any와 rat_f1_micro도 후반부 에서 각각 약 0.80, 0.69 수준을 기록하였다. 중후반 epoch에서 근거 지표의 안정화가 속성 F1과 3-way 정확도의 재상승을 견인하는 패

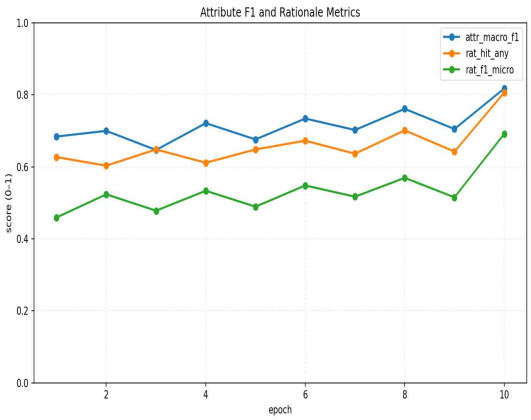


그림 4. 속성 매크로 F1(attr_macro_f1)과 근거 지표(rat_hit_any, rat_f1_micro)의 epoch별 추이

Figure 4. Trends of Attribute Macro F1 (attr_macro_f1) and Rationale Metrics (rat_hit_any, rat_f1_micro) over Epochs

턴이 뚜렷했으며, 이는 근거 문장(sid) 회수의 일관성이 속성 판단과 최종 판정의 품질을 향상시킬 수 있음을 의미한다.

6. 결 론

OSS 활용이 확산됨에 따라 라이선스 양립성 판단은 조항 해석과 실행 맥락을 동시에 고려해야 하는 과제로 대두되었다. 기존 도구는 주로 식별 기능에 집중하여 수작업 판정에 의존하였고, 외부 규칙 엔진 기반 접근에서는 규칙 유지·확장 비용과 오류 전파 문제가 나타났다.

이에 본 논문에서는 외부 규칙 엔진에 의존하지 않고 규칙 지식을 내재화한 LLM을 학습함으로써, 라이선스 전문과 실행 맥락을 입력으로 받아 문장 단위 근거와 함께 최종 양립성 라벨을 직접 산출하는 모델을 제안한다. 이를 위하여 gpt-oss 기반 파인튜닝과 다중 과제 학습을 적용하여 OSS 라이선스의 양립성을 판정한다.

본 연구의 주요 기여는 다음과 같다. 첫째, 외부 규칙 엔진 없이 판정과 근거를 일관되게 산출하는 규칙 내재화 프레임워크를 제안하였다. 둘째, 링크·배포·경계 등 실행 맥락을 반영한 양립성 판정 절차를 정립하였다. 셋째, 문장 단위 근거 매핑을 통해 설명 가능한 판정 결과를 제시하였다. 향후 연구에서는 데이터 및 라이선스 스펙트럼 확대, 컨테이너, 플러그인 등 복합 토폴로지 세분화에 대한 추가적인 모델링을 진행할 예정이다.

본 연구는 문화체육관광부 및
한국콘텐츠진흥원의 2025년도 SW저작권
생태계 조성 기술개발 사업으로 수행되었음
(과제명 : 클라우드 서비스 활용 구축 형태별
대규모 소프트웨어 라이선스 검증 기술개발,
과제번호 : RS-2023-00224818, 기여율: 100%)

참 고 문 헌

[1] Charlie Dai and Lee Sustar, “The Power Of Open Source: Cloud-Native Is Transforming As AI Takes The Limelight”, 2025, URL: https://www.forrester.com/blogs/the-power-of-open-source-cloud-native-is-transforming-as-ai-takes-the-limelight/

[2] Schoettle, H. Open source license compliance-why and how?. Computer, 52(8), pp.63-67, 2019, DOI:10.1109/MC.2019.2915690

[3] German, D., Di Penta, M. A method for open source license compliance of java applications. IEEE software, 29(3), pp.58-63, 2012, DOI: 10.1109/MS.2012.50

[4] T. Claburn, “Free-software warriors celebrate landmark case that enforced GNU LGPL,” The Register, Jan. 10, 2025. URL: https://www.theregister.com/2025/01/10/german_router_maker_avm_lgpl/

[5] D. Knop, “Confusion after lawsuit against AVM over Fritzbox firmware,” heise online, Jan. 17, 2025. URL: https://www.heise.de/en/news/Confusion-after-lawsuit-against-AVM-over-Fritzbox-firmware-10247040.html

[6] C. Piana, F. Pietrosanti, G. B. Gallus, and A. Pianon, “The first AGPL compliance case settled in an Italian Court: a tale of compliance, license compatibility and source code availability,” presented at the FOSDEM 2021, Brussels, Belgium, Feb. 6 - 7, 2021. URL:https://archive.fosdem.org/2021/schedule/event/agplcompliance/

[7] BLACKDUCK, “2024 OSSRA report: Open source license compliance remains problematic”, 2024. URL: https://www.blackduck.com/blog/ossra-license-compliance-risks.html

[8] Stoltz, M. L. The penguin paradox: How the scope of derivative works in copyright affects the effectiveness of the GNU GPL. BUL Rev., 85, 1439. 2005

[9] World Law Group, “Norway: Open source & copyleft licenses - how to ensure commercially acceptable use”, 2024, URL: https://www.theworldlawgroup.com/membership/news/open-source-copyleft-licenses-how-to-ensure-commercially-acceptable-use

[10] KEMP IT LAW, “Open Source Software: the Affero GPL, the ‘as a Service’ world and the CAL”, 2020, URL: https://kempitlaom/insights/open-source-software-the-affero-gpl-the-as-a-service-world-and-the-cal/

[11] Liu, T., Liu, C., Liu, T., Wang, H., Wu, G., Liu, Y., Zhang, Y. Catch the butterfly: Peeking into the terms and conflicts among spdx licenses. In 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) pp. 477-488, IEEE. 2024, March, DOI: 10.1109/SANER60148.2024.00056

[12] Pfeiffer, R. H. License incompatibilities in software ecosystems. arXiv preprint arXiv:2203.01634, 2022, DOI: https://doi.org/10.48550/arXiv.2203.01634

[13] Li, B., Liu, C., Fan, L., Chen, S., Zhang, Z., Liu, Z. Open Source, Hidden Costs: A Systematic Literature Review on OSS License Management. IEEE Transactions on Software Engineering. 2025, DOI: 10.1109/TSE.2025.3586411

[14] Gobeille, R. The fossology project. In Proceedings of the 2008 international working conference on Mining software repositories pp. 47-50, 2008, May, DOI: https://doi.org/10.1145/1370750.1370763

[15] German, D. M., Manabe, Y., Inoue, K. A sentence-matching method for automatic license identification of source code files. In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering pp. 437-446, 2010 September, DOI: https://doi.org/10.1145/1858996.1859088

[16] scancode-toolkit, “scancode-toolkit,” 2024. URL: https://github.com/aboutcode- org/scancode-toolkit,

[17] Kapitsaki, G. M., Kramer, F., Tselikas, N. D. Automating the license compatibility process in open source software with SPDX. Journal of systems and software, 131, 386-401, 2017, DOI: https://doi.org/10.1016/j.jss.2016.06.064

[18] Kapitsaki, G. M., Paschalides, D. Identifying terms in open source software license texts. In 2017 24th Asia-Pacific Software Engineering Conference (APSEC) pp. 540-545, IEEE. 2017, December, DOI: 10.1109/APSEC.2017.62

[19] Cui, X., Wu, J., Wu, Y., Wang, X., Luo, T., Qu, S., ... Yang, M. An empirical study of license conflict in free and open source software. In 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) pp. 495-505, IEEE. 2023 May, DOI: 10.1109/ICSE-SEIP58684.2023.00050

[20] Xu, S., Gao, Y., Fan, L., Liu, Z., Liu, Y., Ji, H. Lidetector: License incompatibility detection for open source software. ACM Transactions on Software Engineering and Methodology, 32(1), pp.1-28. 2023, DOI: https://doi.org/10.1145/3518994

[21] Xu, S., Gao, Y., Fan, L., Li, L., Cai, X., Liu, Z. Liresolver: License incompatibility resolution for open source software. In Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis pp. 652-663, 2023 July, DOI: https://doi.org/10.1145/3597926.3598085

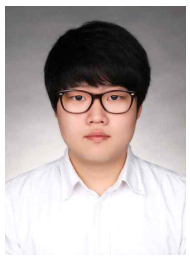
[22] Cui, X., Wu, J., Ling, X., Luo, T., Yang, M., Ou, W. Exploring Large Language Models for Analyzing Open Source License Conflicts: How Far Are We?. In 2025 IEEE/ACM 47th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) pp. 291-302,

IEEE. 2025, April, DOI : 10.1109/ICSE-Companion66252. 2025.00083
[23] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., ... Liu, T. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. ACM Transactions on Information Systems, 43(2), 1-55. 2025, DOI: https://doi.org/10.1145/370315
[24] C. De Roover, R. Wu, and E. Bouwers, “Mining open source license usage patterns,” in Proc. IEEE Int. Working Conf. Mining Softw. Repositories, pp. 222 - 233, 2016,
[25] P. Liang, D. Lo, and F. Thung, “Automatic discovery of license violation in open source projects,” in Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng., pp. 647 - 658, 2016
[26] D. M. German and A. E. Hassan, “License integration patterns: Addressing license mismatches in component-based development,” in Proc. 31st Int. Conf. Softw. Eng., pp. 188 - 198, 2009
[27] Software Package Data Exchange (SPDX), “SPDX specification version 2.3,” SPDX Workgroup, 2022.
[28] https://huggingface.co/openai/gpt-oss-20b

저 자 소 개

김동완(DongWan Kim)

2022.02 한국성서대학교 컴퓨터소프트웨어학과 졸업
2024.02 숭실대학교 컴퓨터학과 석사
2024-현재 엘에스웨어 소프트웨어연구소 연구개발본부 주임 연구원
<주관심분야> 클라우드, 인공지능, 빅데이터, 블록체인



박경엽(Kyung-Yeob Park)

2019.2 서울과학기술대학교 컴퓨터공학과 석사
2019-현재 엘에스웨어(주) 선임 연구원
<주관심분야> IoT 보안, 블록체인, 빅데이터, 메타버스

조용준(YongJoon Joe)

2011.3 큐슈대학교 전기정보공학과 졸업
2013.3 큐슈대학교 정보학부 석사
2016.3 큐슈대학교 정보학부 박사과정 수료
2013.4-2016.3 일본 학술진흥원 특별연구원
2016-현재 엘에스웨어(주) 이사
<주관심분야> 병렬·분산 컴퓨팅, 게임이론, 분산 제약 최적화 문제

신동명(Dong-Myung Shin)

2003.02	대전대학교 컴퓨터공학과 박사
2001-2006	한국정보보호진흥원 응용기술팀 선임연구원
2006-2014	한국저작권위원회 저작권기술팀 팀장
2014-2016	한국스마트그리드사업단보안 인증팀 팀장
2016-현재	엘에스웨어(주) 소프트웨어연구소 연구소장/ 전무이사
<주관심분야>	오픈소스 라이선스, 저작권 기술, 시스템/네트워크보안, SW 취약점 분석·감정, 블록체인 기술, 메타버스