

Blurred Compliance Boundaries: Analyzing Open Source License Conflicts in MCP-Based Agent Workflows

Xing Cui ✉ 

Institute of Software, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China

Jingzheng Wu* ✉ 

Institute of Software, Chinese Academy of Sciences, Beijing, China

Tianyue Luo ✉ 

Institute of Software, Chinese Academy of Sciences, Beijing, China

Xiang Ling ✉ 

Institute of Software, Chinese Academy of Sciences, Beijing, China

Abstract

Background. Agent systems increasingly invoke heterogeneous third-party services through the Model Context Protocol (MCP), where service licenses, model licenses, and data licenses may be jointly applied during runtime execution. This emerging execution paradigm introduces cross-service compliance conflicts that are difficult to capture with conventional license analysis methods. Existing methods usually assume that the license set under analysis is statically predetermined, making them insufficient for handling novel term semantics in MCP licenses and for analyzing conflict patterns that arise from task-specific selected service subsets.

Aims. This paper investigates compositional license conflicts in MCP-based agent workflows. We seek to characterize how license terms interact across compositional MCP services and to provide an automated analysis framework that detects conflicts and identifies license-compatible service replacements.

Method. We propose MCP-Licentra, a license conflict analysis framework for MCP-based agent workflows. MCP-Licentra contains four integrated components. First, it extends an existing license term taxonomy with MCP-specific categories and constructs annotated training data for domain adaptation. Second, it fine-tunes License-Llama3-8B with supervised fine-tuning (SFT) and odds ratio preference optimization (ORPO) to recognize license terms and their associated attitudes. Third, it formulates task-specific service selection as a power-set legality verification problem, enabling conflict checking across possible service invocation subsets. Fourth, it recommends license-compatible replacements for conflicting services through semantic embedding retrieval and an iterative compliance checking algorithm under joint functional similarity and license compatibility constraints.

Results. We evaluate MCP-Licentra on 3,811 real-world MCP services and 500 compositional service scenarios. MCP-Licentra achieves F1-scores of 88.82% and 86.44% for term identification and license understanding, respectively. For group-level conflict detection, it obtains an FPR of 3.15% and an FNR of 8.77%. For compositional conflict detection, it achieves an overall FPR of 4.90% and an FNR of 3.33%. In addition, MCP-Licentra reaches a resolution success rate of 71.23%, outperforming all baselines.

Conclusion. The results show that MCP-based agent workflows introduce compositional compliance risks that are difficult to capture with conventional static license analysis. MCP-Licentra provides an automated framework and empirical evidence for detecting and mitigating such risks in agent systems.

2012 ACM Subject Classification Software and its engineering → Empirical software validation



© Xing Cui, Jingzheng Wu, Tianyue Luo, and Xiang Ling;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).

Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;

Article No. 33; pp. 33:1–33:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Keywords and phrases MCP, license conflict detection, compositional license analysis, large language model, agent compliance

Digital Object Identifier 10.4230/LIPIcs.ESEM.2026.33

Category Technical Track Paper

Funding This work is supported by the National Key R&D Program of China No. 2024YFB3108000, the Beijing Nova Program No. 202604841307, and the YuanTu Large Research Infrastructure.

1 Introduction

As artificial intelligence technology advances rapidly, agents that invoke multimodal tools and external services to accomplish complex tasks have become a prominent paradigm in large language model (LLM) applications. In this context, the MCP has been proposed as a standardized interaction mechanism for connecting models with external capabilities [16, 37]. MCP provides agents with a standardized interface for accessing external services, enabling them to compose and invoke tool services from diverse providers according to task requirements [12, 37]. These MCP services typically encompass a wide range of functional scenarios, including file processing, database access, and code execution [7, 17]. However, the code components, foundation models, and data resources associated with such services are often governed by distinct licenses or terms of use, exhibiting considerable diversity in licensing origins and heterogeneity in constraint specifications [35]. For instance, a service repository may be released under an open-source license such as MIT or Apache-2.0, the underlying model may carry usage restrictions such as RAIL, and the associated data resources may impose additional constraints regarding commercial use, redistribution, or access conditions. As agents compose and invoke multiple services within a single workflow, incompatibilities among different license terms pertaining to output attribution, derivative work classification, and automated access permissions may give rise to license conflicts, leading to distribution violations and restrictions on commercial deployment.

Prior research has produced a series of approaches to license conflict analysis. Early work focused on automated license identification and extraction. Kapitsaki et al. [18] formalize license terms using SPDX structured information for compatibility checking in package composition scenarios. Liu et al. [22] extend compatibility verification to complex dependency structures by propagating license constraints over dependency graphs. With advances in natural language processing, subsequent studies shifted toward semantic understanding of license terms. Xu et al. [42] propose LiDetector, which applies machine learning to identify key license terms through syntactic parsing and semantic feature extraction, inferring encoded rights and obligations to detect conflicts. Building on this, Xu et al. further propose LiResolver [41], which refines term identification and license extraction while supporting license recommendation and customized generation for compliance resolution. Cui et al. [5] propose DIKE, which automates license term extraction and conflict detection via NLP, revealing widespread license conflicts in popular GitHub projects. Their follow-up work [4] further leverages LLMs to improve detection accuracy in complex real-world scenarios.

Although prior studies advance license conflict detection for static software compositions, their direct application to MCP workflows faces several limitations. First, existing methods largely rely on traditional license datasets such as SPDX, providing limited coverage of

*Corresponding author

emerging AI-specific licensing constraints. As a result, they fail to capture conflicts involving output ownership, automated access, or user profiling inference. Second, they typically assume a static, predetermined license set and determine whether a given composition is conflicting, which is insufficient for analyzing conflicts across different service subsets in MCP workflows. Third, existing mitigation strategies mainly rely on license replacement or relicensing. In MCP settings, however, service licenses are usually predefined by third-party providers and cannot be changed by end users. A more practical direction is to retrieve functionally equivalent alternative services under license compatibility constraints, while this problem remains underexplored.

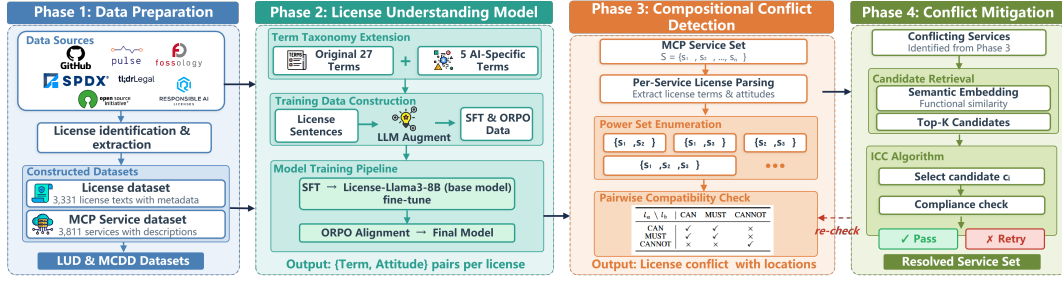
To address these challenges, we propose MCP-Licentra, a license conflict analysis framework for MCP-based agent workflows. MCP-Licentra fine-tunes License-Llama3-8B for domain adaptation by extending the existing license term taxonomy and constructing annotated training data; formalizes agent service selection as a power set legality verification problem to systematically detect conflicts across all possible service invocation subsets; and recommends license-compatible replacement services through semantic embedding retrieval and an Iterative Compliance Checking (ICC) algorithm. Experiments on 3,811 real-world MCP services and 500 combination scenarios demonstrate that MCP-Licentra consistently outperforms all baseline methods across four tasks, including term identification, license understanding, compositional conflict detection, and conflict resolution. The main contributions of this paper are as follows:

- (1) We formalize license conflicts in MCP-based agent workflows as a power set legality verification problem over candidate service subsets, identifying three fundamental challenges absent from conventional static license analysis.
- (2) We propose MCP-Licentra, a framework for license term understanding and compositional conflict analysis. It combines a domain-adapted license understanding model with extended term coverage, a power-set-based conflict detection mechanism, and an ICC algorithm for resolving conflicts under functional similarity and license compatibility constraints.
- (3) We construct two evaluation datasets, LUD and MCDD, for MCP license conflict analysis. Experiments on 3,811 MCP services and 500 combination scenarios demonstrate that MCP-Licentra achieves F1-scores of 88.82% and 86.44%, FPR of 4.90%, FNR of 3.33%, and RSR of 71.23%, outperforming all baselines.

2 Background

2.1 MCP Services

The MCP is an open protocol that standardizes interactions between LLMs and external services, enabling unified compositional invocation without custom interfaces [30]. Through standardized service descriptions and invocation mechanisms, MCP allows agents to access registered services for task execution and capability extension [14]. However, the rapid expansion of the MCP ecosystem introduces services from diverse maintainers into shared workflows, inevitably bringing heterogeneous license constraints. Multi-service compositions may yield license incompatibilities regarding distribution, derivative work obligations, or usage restrictions, posing legal risks [13, 24]. Such conflicts typically remain latent at the individual service level and emerge only when multiple services are combined within an agent workflow, leading to more complex compliance analysis.



■ **Figure 1** Overview of the MCP-Licentra framework for compositional license conflict analysis in MCP-based agent workflows.

2.2 Open source licenses and license conflicts

Open source licenses are legal instruments that govern the use, reproduction, modification, and distribution of software, defining permissible and prohibited behaviors through specific terms to protect developer rights and promote collaborative development [38]. A license conflict arises when multiple licenses within a project impose mutually contradictory terms, making simultaneous compliance infeasible. For instance, a component license may stipulate “CANNOT sublicense” while the project license requires “CAN sublicense”; similarly, one component may mandate “MUST contact the author” while another specifies “CANNOT contact the author”, inevitably violating at least one license [9].

3 Methodology

3.1 overview

This section presents MCP-Licentra, a compositional license conflict detection framework for MCP-based agent workflows. As shown in Figure 1, the framework includes four components.

3.2 Data preparation

This section describes the data collection and preprocessing pipeline for MCP services and license, which provides the foundation for subsequent model training and conflict analysis.

3.2.1 Data collection

► **MCP service collection.** We collect metadata for 3,821 MCP services from the pulseMCP platform [30], including natural language descriptions, functional summaries, usage examples, and GitHub repository links. After excluding 10 services without valid repository URLs, 3,811 samples are retained. All repositories are cloned locally and preprocessed uniformly, including binary artifact removal, directory encoding standardization, and metadata index construction. The resulting dataset preserves both service descriptions and full source structures, serving as the data foundation for license extraction and conflict analysis.

► **License collection.** We collect 3,331 open source licenses from authoritative sources. Among them, 3,324 conventional licenses are drawn from SPDX [39], OSI [25], TLDRLegal [40], and FOSSology [8], while seven AI-specific licenses are obtained from the Responsible AI Licenses (RAIL) project [36]. These AI-specific licenses cover constraints rarely captured

■ **Table 1** Extended license term taxonomy.

No.	Term	Explanation
1	User Owns Model Output	User owns model output; licensor claims no rights; user is responsible for its use
2	User Content Rights	Grant of rights to use, modify, distribute, and display user content
3	AI Output Attribution	Clear identification of AI-generated content
4	Automated Access Limitation	Prohibit excessive automated access via bots or crawlers
5	Predictive User Profiling	Infer characteristics related to insurance risk, health, criminal behavior, identity, or biometrics from personal data

by conventional license datasets, such as model outputs, data usage, and automated access. Each entry includes a standardized SPDX identifier, license name, version information, and complete legal text. We further build a cross-platform alias mapping to reduce naming inconsistencies, followed by text normalization and deduplication. The resulting dataset provides a structured basis for license identification and conflict analysis.

3.2.2 Training data construction

License-Llama3-8B [4] is an instruction-tuned model built on the Llama3 architecture and optimized for license understanding, with strong capabilities in recognizing open source license terminology, term attitudes, and rights obligation structures. However, its term and attitude recognition covers only 27 traditional open source license term categories, providing no support for the novel constraint types introduced by AI-specific licenses. To address this limitation, MCP-Licentra extends the existing term taxonomy with supplementary categories.

We collect the official texts of 7 RAIL-series licenses and apply NLTK [3] sentence segmentation, obtaining 1,169 candidate sentences. Manual filtering retains 689 sentences that express explicit rights or obligation constraints. Semantic analysis of these sentences reveals that a substantial portion encodes constraint types not representable by the existing term taxonomy, including model output ownership, AI-generated content attribution obligations, automated access limitations, and AI-based inference of sensitive personal characteristics.

We further remove two types of unsuitable sentences: ethical recommendations without clear permissive or restrictive attitudes, such as “prevent generating false information to harm others through AI outputs”, and broad statements that cannot be mapped to well-bounded terms, such as “ensure transparency in automated decisions affecting legal rights”. Two researchers experienced in license analysis independently conduct the filtering and achieve high agreement, with a Cohen’s Kappa of 0.95. Disagreements are resolved through discussion or removed. This process retains 365 sentences with clear semantic boundaries, grouped into 5 supplementary term categories and annotated with CAN, MUST, or CANNOT attitudes. Table 1 provides their definitions. To enable License-Llama3-8B to recognize the 5 supplementary term categories and their associated attitudes, we construct supplementary training data by generating semantically equivalent paraphrases from the 365 seed sentences using Qwen3 [44]. For each seed sentence, the corresponding term definition, attitude label, and sentence text are organized into a structured prompt, which is submitted to Qwen3 to generate 10 paraphrases that preserve the original term semantics and attitude while varying in phrasing, sentence structure, and expression angle. The prompt template is shown below.

OBJECTIVE

The following is a dict consisting of license terms, term attitudes, and an official term description. You need the terms and attitude in this dict to write 10 additional term descriptions that are different from the official term descriptions.

DICT CONTENT

```
{ "terms": {terms_definition}, "attitudes": {attitudes_definition}, "terms_description": "(b) You must give any Third Party recipients of the Model or Derivatives of the Model a copy of this License;" }
```

RESPONSE FORMAT

Output a JSON list of dicts, each containing keys: terms, attitudes, and terms_description. The generated descriptions must conform to the given terms and attitudes.

The definitions and attitude labels of the 5 supplementary terms are inserted into {terms_definition} and {attitudes_definition} respectively, while one seed sentence is placed in {terms_description}. The prompt is submitted to Qwen3, which returns a JSON list of paraphrased descriptions. The first three entries of an example response are shown below.

JSON Output Example

```
[{"terms": "Provide a copy of this license for the derivative products of the model", "attitudes": "MUST", "terms_description": "A copy of this license must be included with any distribution of the model's derivatives."}, {"terms": "...", "attitudes": "MUST", "terms_description": "You are required to supply this license alongside any derivative works derived from the model."}, {"terms": "...", "attitudes": "MUST", "terms_description": "Whenever distributing a derivative of the model, this license must be provided to all recipients."}, ...]
```

Qwen3 generates 10 paraphrases per seed sentence, producing 3,650 candidate samples. Two researchers experienced in software license analysis independently screen the candidates to ensure that each sentence maps to a specific supplementary term category, preserves the original constraint semantics, introduces no new rights or obligations, and keeps the same attitude label as the seed. Disagreements are resolved through discussion, and unresolved entries are discarded. The filtering retains 3,212 valid sentences, which are combined with the 365 seed sentences to form a 3,577-sample supplementary term dataset. We split the dataset into training, validation, and test datasets at a 7:1:2 ratio, resulting in 2,504, 358, and 715 samples, respectively.

3.3 MCP license understanding model

This section presents the domain adaptation training of License-Llama3-8B for term category and attitude recognition across traditional and AI-specific licenses, supporting subsequent conflict detection and mitigation.

3.3.1 SFT: term and attitude recognition

Following the construction of the supplementary term taxonomy and the preparation of training data, we apply SFT [21] to adapt License-Llama3-8B for accurate recognition of the extended term categories and their associated attitudes. The 2,504 training samples obtained in Section 3.2.2 are formatted using the following prompt:

OBJECTIVE

Identify the terms and corresponding attitudes contained in the given license text based on the definition of license terms and attitudes.

DEFINITIONS

Terms: {terms_definition}; Attitudes: {attitudes_definition}

LICENSE TEXT

"You may convey modified covered source (...) provided that you: a) retain notices as required in subsection 3.2; and b) add a notice to the modified covered source stating that you have modified it, with the date and brief description of how you have modified it."

RESPONSE FORMAT

Output JSON key-value pairs, where the key is the term name and the value is the corresponding attitude name.

In the prompt, `{terms_definition}` and `{attitudes_definition}` are populated with the supplementary term definitions and attitude labels respectively, while the `LICENSE TEXT` field is filled with the corresponding license description. The expected response is constructed by inserting the ground-truth term-attitude mappings into the response template, forming supervised input-output pairs for fine-tuning.

JSON Output Example

After analyzing the license text, I've identified the following terms and attitudes:
{"Modify": "CAN", "State Changes": "MUST", "Include Notice": "MUST"}

To reduce catastrophic forgetting of the original 27 term categories, SFT jointly uses the 2,504 supplementary training samples and the original License-Llama3-8B dataset. The original 19,883 samples are split at a 7:1:2 ratio, with 12,726 samples used for training. Following Li et al. [20], we adopt re-balanced sampling, where 80% of each mini-batch comes from the supplementary dataset and 20% from the original dataset. This increases exposure to new term semantics while preserving traditional term recognition. The SFT objective maximizes the conditional log-likelihood of the reference response.

$$\log P_{\theta}(y \mid x) = \frac{1}{M} \sum_{t=1}^M \log P_{\theta}(y_t \mid x, y_{<t}) \quad (1)$$

Here, x is the tokenized input prompt, $y = (y_1, \dots, y_M)$ is the tokenized output sequence, y_t is the t -th output token, $y_{<t}$ denotes the preceding tokens, $P_{\theta}(y_t \mid x, y_{<t})$ is the probability of y_t under model θ , and M is the output length. Through SFT training, the model establishes a stable mapping from license text to term-attitude pairs within a unified term space, preserving recognition performance on the original 27 term categories while substantially improving generalization to the 5 supplementary term categories.

3.3.2 ORPO: output consistency and hallucination reduction

Although SFT equips the model with term-to-attitude mapping capabilities, likelihood maximization alone may lead to output instability, attitude prediction bias, and hallucinations in complex license contexts. To address this issue, we apply ORPO-based preference alignment after SFT to strengthen the model's preference toward correct term-attitude combinations and suppress hallucination tendencies [15]. For each supplementary term training sample, we construct a preference pair consisting of a chosen response y_w and a rejected response y_l . The chosen response corresponds to the accurate term-attitude combination determined by human annotation or rule-based verification, while the rejected response is obtained by randomly perturbing attitude labels or mismatching term names. The SFT-trained License-Llama3-8B is then further fine-tuned with the ORPO objective $L_{\text{ORPO}} = L_{\text{SFT}} + \lambda L_{\text{OR}}$, where $L_{\text{SFT}} = -\frac{1}{M} \sum_{t=1}^M \log P_{\theta}(y_t \mid x, y_{<t})$ denotes the negative log-likelihood loss, consistent with the SFT objective, and λ is a weighting factor. The odds ratio loss is defined as $L_{\text{OR}} = -\log \sigma \left(\log \frac{\text{odds}_{\theta}(y_w \mid x)}{\text{odds}_{\theta}(y_l \mid x)} \right)$, where $\text{odds}_{\theta}(y \mid x) = \frac{P_{\theta}(y \mid x)}{1 - P_{\theta}(y \mid x)}$ and $P_{\theta}(y \mid x)$ denotes the sequence-level generation likelihood. By increasing the relative preference for y_w over y_l , L_{ORPO} improves the model's sensitivity to term attitude polarity and reduces misclassification risk in conflict detection.

Here, L_{SFT} denotes the negative log-likelihood loss, which is consistent with the objective used in the SFT stage, and λ is a weighting factor. L_{OR} denotes the odds ratio loss, where the odds function is defined as $\text{odds}_{\theta}(y \mid x) = \frac{P_{\theta}(y \mid x)}{1 - P_{\theta}(y \mid x)}$, and $P_{\theta}(y \mid x)$ denotes the sequence-level generation likelihood. L_{ORPO} encourages a higher probability for y_w and a lower probability for y_l , improving the model's sensitivity to term attitude polarity and reducing misclassification risk in conflict detection.

3.4 Compositional license conflict detection

3.4.1 License extraction

MCP-Licentra builds on ScanCode’s [1] rule-matching framework and extends its rule set to support AI-specific license detection. ScanCode’s matching mechanism relies on token subsequence matching, which requires exact matches between key tokens in license texts and standard sentences in the rule base. However, AI-specific license terms often appear in project documentation as paraphrased expressions or non-standard license statements, causing the native matching mechanism to miss licenses such as those in the RAIL series. To address this limitation, MCP-Licentra preserves the original rule base and introduces a fuzzy matching strategy based on semantic similarity, using the 365 seed term descriptions from Section 3.2.2 as supplementary rules. Each rule text rule_i is first normalized and then tokenized into $R_i = \text{tokenize}(\text{normalize}(\text{rule}_i)) = [r_{i1}, r_{i2}, \dots, r_{ik_i}]$, where R_i denotes the token sequence of the i -th rule, r_{ij} denotes its j -th token, and k_i denotes the sequence length. These sequences form an extended rule base $\mathcal{R} = \{R_1, R_2, \dots, R_m\}$, where $m = 365$.

The target document text is processed identically to obtain token sequence $T = [t_1, t_2, \dots, t_n]$. A sliding window of length $|R_i|$ is applied over T , and for each subsequence W the weighted similarity with rule R_i is computed as:

$$\text{Sim}(R_i, W) = \alpha_0 \cdot \text{Sim}_{\text{Jaccard}}(R_i, W) + \alpha_1 \cdot \text{Sim}_{\text{Cosine}}(R_i, W) \quad (2)$$

where $\text{Sim}_{\text{Jaccard}}(R_i, W) = \frac{|R_i \cap W|}{|R_i \cup W|}$, $\text{Sim}_{\text{Cosine}}(R_i, W) = \frac{\sum_j f_{R_i}(r_j) \cdot f_W(r_j)}{\|f_{R_i}\| \cdot \|f_W\|}$, $f_{R_i}(r_j)$ and $f_W(r_j)$ denote the term frequencies of token r_j in R_i and W respectively, and α_0, α_1 are weighting coefficients satisfying $\alpha_0 + \alpha_1 = 1$. A rule R_i is considered matched in the document if $\max_W \text{Sim}(R_i, W) \geq \gamma$, where γ is the similarity threshold. The extended rules are integrated into the ScanCode detection pipeline alongside the rule base, preserving coverage of traditional open-source licenses while closing the detection gap for AI-specific license declarations.

3.4.2 Compositional license conflict detection

Existing license conflict detection methods typically rely on pairwise compatibility checks over predefined licenses. MCP-based agent workflows, however, involve multiple selected services whose joint license constraints may introduce conflicts beyond individual pairs. Pairwise analysis therefore provides limited coverage for multi-service interactions. MCP-Licentra instead formulates conflict detection as legality verification over service subsets. Because invocation order does not affect the resulting license combination, each workflow configuration is represented as a subset of selected MCP services and verified independently, enabling systematic coverage of all license combinations.

In an MCP-based agent workflow, a user selects a base language model and a set of MCP services. The licenses associated with the base model form the Model License set $ML = [ML_1, ML_2, \dots, ML_k]$, and the licenses associated with the i -th MCP service form the Server License set $SL_i = [SL_{i1}, SL_{i2}, \dots, SL_{ik_i}]$. The selected MCP service set is denoted as $S = \{S_1, S_2, \dots, S_m\}$, where m denotes the number of services included in the workflow configuration. The non-empty power set of S contains $2^m - 1$ subsets. This design restricts detection to the selected services and avoids checking unrelated services in the MCP registry.

Since service invocation order does not affect license combinations, all activated service subsets within the selected MCP service set can be represented as the non-empty subsets of S , denoted as $C = \mathcal{P}(S) \setminus \{\emptyset\}$. For any service combination $S' = \{S_{i_1}, S_{i_2}, \dots, S_{i_n}\} \subseteq S$, the combined license set is defined as $\text{CLS}(S') = ML \cup \bigcup_{j=1}^n SL_{i_j}$.

■ **Table 2** Conflict detection between l_a and l_b .

$l_a \setminus l_b$	CAN	MUST	CANNOT
CAN	✓	✓	×
MUST	✓	✓	×
CANNOT	×	×	✓

To determine pairwise license compatibility within a combination, we define a compatibility relation over the three term attitudes CAN, MUST, and CANNOT. A conflict arises when one license expresses CAN or MUST on a term while another expresses CANNOT on the same term; all other attitude pairs are considered compatible, as summarized in Table 2. The conflict detection function is defined as:

$$\text{Conflict}(\text{CLS}(S')) = \begin{cases} 1 & \text{if } \exists (l_a, l_b) \in \text{CLS}(S') \times \text{CLS}(S'), l_a \not\sim l_b \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $l_a \not\sim l_b$ denotes a term-level conflict between licenses l_a and l_b . MCP-Licentra enumerates all subsets in C and partitions them into a compliant set and a conflict set:

$$C_{\text{legal}} = \{S' \in C \mid \text{Conflict}(\text{CLS}(S')) = 0\}, \quad C_{\text{conflict}} = C \setminus C_{\text{legal}} \quad (4)$$

In practice, MCP-Licentra first scans service repositories to extract license texts, then identifies the specific license terms and their attitudes, and finally performs compatibility checking across all service combinations to produce C_{legal} and C_{conflict} along with the corresponding conflicting terms.

3.5 Conflict mitigation

Existing license conflict mitigation methods rely on license substitution or relicensing, which is infeasible in MCP scenarios because service licenses are controlled by third-party providers. MCP-Licentra instead treats mitigation as compliant alternative service retrieval, identifying functionally similar and license-compatible replacements for conflicting services and resolving compositional conflicts through iterative compliance checking.

3.5.1 Alternative service retrieval

MCP-Licentra encodes all MCP services in the service catalog into dense vectors using Qwen3-Embedding-8B [45] to support efficient retrieval of functionally similar services. For each service, the name, description, and classification fields are extracted and concatenated into a standardized information string following the format below.

The standardized strings of all catalog services are encoded into dense vectors and stored in a vector index. For a conflicting service, its information string is encoded into a query vector $\mathbf{m} \in \mathbb{R}^d$ and compared with each candidate vector $\mathbf{n}_i \in \mathbb{R}^d$ in the index using cosine similarity, where the retrieval score is computed as $\text{score}_i = \frac{\mathbf{m} \cdot \mathbf{n}_i}{\|\mathbf{m}\| \cdot \|\mathbf{n}_i\|}$. where $\cdot$ denotes the dot product, $\|\mathbf{m}\|$ and $\|\mathbf{n}_i\|$ are the norms of the query and candidate vectors respectively, and d is the embedding dimension. All candidates are ranked in descending order of score_i , and those with $\text{score}_i \geq \tau$ are retained. The top- k services among the filtered candidates form the alternative service list:

$$\text{Recommended}(\mathbf{m}) = \{S_i \mid \text{score}_i \geq \tau \wedge \text{argmax}_k(\text{score}_i)\} \quad (5)$$

where τ is the similarity threshold, k is the maximum candidate list size, and S_i denotes the i -th candidate alternative service.

3.5.2 Iterative compliance checking

In MCP compositional conflict scenarios, conflicts do not always manifest as a single service being incompatible with both ML and the remaining services. More complex cases involve multiple conflict points across different service subsets simultaneously, where replacing a single service is insufficient to restore overall compliance. To handle such multi-point conflicts, MCP-Licentra introduces an ICC mechanism that replaces conflicting services and continuously updates the current compliant service set until a compliant configuration is obtained or no feasible replacement exists.

Given the conflict detection results C_{legal} and C_{conflict} , MCP-Licentra replaces conflicting services one by one using the license constraints of the current compliant combinations as the reference. The replacement order follows two priority principles. The replacement order follows two priority principles. Under the ML-first principle, services conflicting with ML are handled first, as such conflicts directly affect the compatibility of the entire combination with the underlying model. Under the minimal modification principle, when conflicts occur among MCP services, the service with the highest number of conflicts is prioritized, minimizing the total number of replacements required to restore compliance. The replacement priority of each conflicting service S_{ij} within conflict group S_i is quantified as:

$$\text{Priority}(S_{ij}) = \beta \cdot \mathbf{1}[\text{IsConflictWith}(S_{ij}, ML)] + (1 - \beta) \cdot \frac{\text{ConflictNum}(S_{ij})}{\text{ConflictNum}_{\max}} \quad (6)$$

where $\mathbf{1}[\cdot]$ is the indicator function, $\text{ConflictNum}(S_{ij})$ denotes the number of conflicts between S_{ij} and other services in S_i , and $\beta \in [0, 1]$ balances the two principles.

For each conflict group $S_i \in C_{\text{conflict}}$, MCP-Licentra sorts the conflicting services by $\text{Priority}(S_{ij})$ in descending order and generates a candidate replacement set for each S_{ij} using the retrieval method described above. A candidate S'_{ij} is accepted if:

$$\text{IsCompatible}(S'_{ij}, ML \cup \text{Licenses}(C_{\text{legal}})) = \text{True} \quad (7)$$

Upon acceptance, the conflict group is updated and the compliant set is revised as:

$$S_i \leftarrow (S_i \setminus \{S_{ij}\}) \cup \{S'_{ij}\}, C_{\text{legal}} \leftarrow C_{\text{legal}} \cup \{S_i\}, C_{\text{conflict}} \leftarrow C_{\text{conflict}} \setminus \{S_i\} \quad (8)$$

If no compatible replacement exists for a conflicting service, the corresponding conflict group is added to IrreplaceableList and removed from C_{conflict} . The process repeats until $C_{\text{conflict}} = \emptyset$, at which point the algorithm outputs the updated C_{legal} , the replacement record RecommendedList, and IrreplaceableList. Since C_{legal} is updated immediately after each replacement, all subsequent compatibility checks are performed against the latest state, ensuring that newly introduced replacements do not generate new license conflicts with existing compliant combinations.

4 Experimental setup

4.1 Datasets

License understanding dataset (LUD). LUD evaluates the model's ability to recognize license terms and infer their associated attitudes. The dataset combines 3,577 supplementary term samples from Section 3.2.2 and 19,883 samples from the original License-Llama3-8B [4] training dataset. Both subsets are partitioned at a 7:1:2 ratio, producing test splits of 715 and 3,977 samples respectively, which are merged to form a LUD test dataset of 4,692 samples covering both traditional open source licenses and AI-specific supplementary licenses.

MCP Conflict detection dataset (MCDD). MCDD is used to evaluate license conflict detection in MCP service combination scenarios. We randomly sample from the 3,811 MCP services described in Section 3.2.1 to construct service combinations of sizes 1 to 5, generating 100 groups per size for a total of 500 groups. Each group is assigned one of three base models, Llama3-8B, Qwen3-32B, or DeepSeek-V4-Pro, to simulate real-world agent workflow scenarios. Three domain experts independently annotate all 500 groups for license conflicts. Inter-annotator agreement is measured by Fleiss’ Kappa, resulting in a coefficient of 0.86. Disagreements are first resolved by majority voting, and remaining contested cases are adjudicated through expert discussion, producing the final ground-truth annotations.

4.2 Baselines

We compare MCP-Licentra with 7 license analysis baselines and 5 LLM baselines.

License analysis baselines. Text Similarity [11] is a heuristic method that matches license sentences to term categories by computing cosine similarity against term representative vectors. FOSS-LTE [19] applies LDA topic modeling to match license text topic distributions against manually defined term topics. DIKE [5] frames term identification as a multi-label classification task using a Label-specific attention network to predict multiple term labels simultaneously. LiDetector [42] employs a Bi-LSTM sequence labeling model to identify term entities token by token. LiResolver [41] extends LiDetector with fine-grained entity modeling, recognizing action, object, attitude, and condition entities to infer term attitudes. L³icNexus [4] combines regex-based term extraction with a language model for context-aware attitude inference. SPDX-VT [18] constructs a directed license compatibility graph based on SPDX rules for conflict detection.

LLM Baselines. DeepSeek-V4-Pro [6], Qwen3-32B [34], Llama-4-Scout-17B [23], GPT-4.5 [26], and Claude 3.5 Sonnet [2] perform term identification and attitude inference under zero-shot settings using the same prompt template as Section 3.3, with temperature set to 0.

4.3 Metrics

Accurate conflict detection requires correctly recognizing license terms and inferring their attitudes. Following prior work, we adopt Precision = $TP/(TP + FP)$, Recall = $TP/(TP + FN)$, and their harmonic mean $F1 = 2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall})$ as metrics, where TP , FP , and FN denote true positives, false positives, and false negatives, respectively.

Two evaluation metrics are designed to assess resolution effectiveness and replacement efficiency respectively. The Resolution Success Rate, defined as $RSR = N_{\text{resolved}}/N_{\text{conflict}}$, measures the proportion of conflict groups successfully restored to a compliant state, where N_{conflict} is the total number of conflict groups and N_{resolved} is the number of groups successfully incorporated into C_{legal} by the ICC algorithm. The Average Replacement Number, defined as $ARN = \frac{1}{N_{\text{resolved}}} \sum_{i=1}^{N_{\text{resolved}}} R_i$, measures the average number of replacement operations per successfully resolved group, where R_i denotes the number of replacements for the i -th resolved group. Note that R_i does not equal the number of conflicting services, as replacing a single service may simultaneously eliminate multiple conflicts. ARN therefore reflects the actual replacement efficiency under the minimal modification principle.

■ **Table 3** Comparison on term identification.

Method	Precision(%)	Recall(%)	F1-score(%)
Text Similarity	18.39	49.25	26.78
FOSS-LTE	16.58	67.12	26.59
DIKE	84.39	59.69	69.92
LiDetector	85.32	65.89	74.36
LiResolver	76.58	71.92	74.18
L ³ icNexus	90.31	78.33	83.89
DeepSeek-V4-Pro	49.82	76.71	60.41
Qwen3-32B	22.96	64.32	33.84
Llama-4-Scout	34.79	65.34	45.40
GPT-5.4	51.63	76.89	61.78
Claude 3.5 Sonnet	38.88	69.62	49.90
MCP-Licentra	92.83	85.15	88.82

4.4 Experimental settings

All experiments run on a server with 2×NVIDIA H100 GPUs (80GB HBM3), an Intel Xeon Platinum 8463B CPU, and 50GB memory, using Python 3.10.13 and PyTorch 2.2.0. During SFT, we train for 5 epochs with a learning rate of 2×10^{-5} and a batch size of 8, applying the re-balanced sampling strategy described in Section 3.3.1. During ORPO, we fine-tune for 3 epochs with a loss weight of $\lambda = 0.1$. Both stages use the AdamW optimizer with a cosine warmup schedule and a warmup ratio of 0.03.

5 Evaluation and discussion

5.1 Research questions

In this section, we evaluate MCP-Licentra on real-world datasets and answer the following research questions:

RQ1: How effective is MCP-Licentra in license term and attitude analysis?

RQ2: How do license extension, SFT, and ORPO contribute to MCP-Licentra?

RQ3: How effectively does MCP-Licentra detect and mitigate conflicts in MCP service combinations?

5.2 Evaluation on license understanding

We evaluate MCP-Licentra on the LUD test dataset for license term identification and attitude inference, comparing it with 6 applicable baselines and 5 LLM-based methods using precision, recall, and F1-score. SPDX-VT is excluded because it directly checks predefined license relationships without a standalone term or attitude analysis. Results are reported in Table 3.

As shown in Table 3, MCP-Licentra achieves the best performance across all metrics, with 92.83% precision, 85.15% recall, and an 88.82% F1-score. Compared with the strongest baseline, L³icNexus, MCP-Licentra improves precision, recall, and F1-score by 2.52%, 6.82%, and 4.93%, respectively. This gain mainly comes from targeted fine-tuning on supplementary term data, which improves the recognition of AI-specific license terms beyond zero-shot generalization. Text Similarity and FOSS-LTE perform worst, with F1-scores of 26.78% and 26.59%, respectively, because they assign only one term per sentence and cannot handle multi-term license statements. FOSS-LTE also shows severe over-prediction, as indicated by its low precision of 16.58% despite a recall of 67.12%. DIKE, LiDetector, and LiResolver

■ **Table 4** Comparison on license understanding.

Method	Precision(%)	Recall(%)	F1-score(%)
Text Similarity	14.93	38.75	21.56
FOSS-LTE	14.63	59.87	23.51
DIKE	71.37	50.97	59.47
LiDetector	72.65	57.02	63.89
LiResolver	64.01	64.23	64.12
L ³ icNexus	88.36	76.27	81.87
DeepSeek-V4-Pro	47.57	74.76	58.14
Qwen3-32B	20.91	60.31	31.05
Llama-4-Scout	31.76	60.67	41.69
GPT-5.4	50.12	69.92	58.39
Claude 3.5 Sonnet	36.96	65.43	47.24
MCP-Licentra	90.51	82.72	86.44

perform better, with LiDetector achieving the highest F1-score among them at 74.36%, but they remain behind L³icNexus and MCP-Licentra due to their limited ability to capture complex license language. Among zero-shot LLM baselines, GPT-5.4 and DeepSeek-V4-Pro achieve F1-scores of 61.78% and 60.41%, respectively, while the remaining LLMs perform notably worse, further confirming the need for domain-specific fine-tuning.

Based on the license term identification results, each identified term is further classified into one of three attitudes: CAN, CANNOT, or MUST. Compared with term identification, attitude inference requires deeper semantic reasoning because the model must infer normative intent from context, especially under negation, ambiguity, or complex syntax. We evaluate each method by predicting attitudes for the identified terms and report precision, recall, and F1-score. Since this task depends on the preceding term identification stage, missed terms cannot be evaluated. The results therefore reflect the end-to-end performance of the two-stage pipeline.

As shown in Table 4, MCP-Licentra achieves the best performance across all metrics, with Precision 90.51%, Recall 82.72%, and F1 86.44%, improving F1 by 4.57% over the strongest baseline L³icNexus (81.87%), primarily through better attitude inference on supplementary term categories. Text Similarity and FOSS-LTE perform worst (F1 21.56% and 23.51%), as static matching and topic modeling lack capacity for syntactic and contextual reasoning. DIKE, LiDetector, and LiResolver show limited attitude inference due to architectural constraints in handling domain-specific license language, compounded by error propagation from term identification. LLM baselines, despite strong general language modeling, all underperform fine-tuned methods due to insufficient domain-specific understanding of license semantics and compliance logic.

Answer to RQ1: MCP-Licentra achieves F1-scores of 88.82% and 86.44% on term identification and license understanding, consistently outperforming all baselines. These results validate that the extended term taxonomy and task-oriented training strategy effectively enhance license analysis in MCP scenarios.

5.3 Effect of license extension, SFT, and ORPO

To assess the contribution of each component in the license understanding module, we conduct ablation experiments using the same dataset and metrics as Section 5.2, constructing three variants. The first variant, w/o license extension, removes the supplementary term taxonomy and restricts training and inference to the original 27 terms, which is equivalent

■ **Table 5** Performance comparison of ablation variants.

Method	Term Ident.			License Und.		
	P	R	F1	P	R	F1
Llama3-8B	8.97	71.79	15.95	8.32	67.26	14.81
w/o License Ext.	90.31	78.33	83.89	88.36	76.27	81.87
w/o SFT	91.08	81.22	85.87	88.97	78.16	83.22
w/o ORPO	92.53	84.21	88.17	89.92	80.86	85.15
MCP-Licentra	92.83	85.15	88.82	90.51	82.72	86.44

to using the L³icNexus License-Llama3-8B model directly. The second variant, w/o SFT, retains the supplementary term taxonomy in the prompt but removes the additional SFT on supplementary term data, using the original L³icNexus fine-tuned model instead. The third variant, w/o ORPO, retains both license extension and SFT but removes the preference alignment stage. All variants use the same temperature setting as Section 5.2 during inference.

Table 5 presents the ablation results. All variants substantially outperform the unadapted Llama3-8B baseline, as they inherit the fine-tuning foundation of L³icNexus and maintain stable recognition of the original 27 term categories. Performance differences therefore primarily reflect each variant’s capability in handling AI-specific supplementary terms. Removing License Extension causes F1-score to drop by 4.93% on Term Identification and 4.57% on license understanding, with Recall declining more sharply than precision, confirming that License Extension primarily enhances term coverage rather than prediction precision. Removing SFT while retaining License Extension leads to an F1-score decline of approximately 2.95%. The observation that w/o SFT outperforms w/o License Extension suggests that injecting supplementary term definitions into the prompt provides limited zero-shot generalization, yet this cannot substitute for parameter-level adaptation. SFT remains essential for converting extended term knowledge into stable recognition capability. Removing ORPO produces marginal degradation, with F1-score declining by 0.65% and 1.29% on the two tasks respectively. The larger impact on license understanding indicates that ORPO improves the consistency and reliability of attitude inference, reducing inconsistent outputs that may persist after SFT. Overall, the three components address distinct performance bottlenecks. License Extension broadens term coverage, SFT internalizes extended knowledge into model parameters, and ORPO enhances attitude inference reliability. Their complementary contributions collectively support the superior performance of the MCP-Licentra.

Answer to RQ2: License extension and SFT are the primary drivers of performance improvement. License Extension broadens term coverage, SFT internalizes extended term semantics and attitude classification rules into model parameters, and ORPO further improves output consistency and reliability, with all three components collectively enabling the MCP-Licentra to achieve optimal performance on both tasks.

5.4 MCP license conflict detection and mitigation

This section evaluates MCP-Licentra’s conflict detection on the MCDD dataset (Section 4.1) at two levels. Group-level detection treats each base-model and MCP-service combination as a composite project and compares MCP-Licentra with SPDX-VT [18], DIKE [5], LiDetector [42], LiResolver [41], and L³icNexus [4]. Since these baselines target static single-project analysis, we adapt their inputs by treating the base model and each service as independent components

■ **Table 6** Group-level license conflict detection results.

Method	#Report	FPR(%)	FNR(%)
SPDX-VT	2310	4.14	72.89
DIKE	3516	4.49	49.45
LiDetector	3946	3.44	35.06
LiResolver	4152	3.23	29.86
L ³ icNexus	4962	3.31	13.13
MCP-Licentra	5132	3.15	8.77

in a unified detection target. Compositional detection further checks possible service subsets of different sizes, reflecting task-specific activation in agent workflows. Both settings use False Positive Rate (FPR) and False Negative Rate (FNR), computed as $FPR = FP/(FP + TN)$ and $FNR = FN/(FN + TP)$, where FP , TN , FN , and TP denote false positives, true negatives, false negatives, and true positives, respectively.

5.4.1 Group-level license conflict detection

For each sample group, all licenses associated with the base model and MCP service set are extracted and submitted as a unified detection target. The MCDD dataset contains 4,777 positive and 24,539 negative samples across the 500 annotated combination groups. Results are presented in Table 6.

MCP-Licentra achieves the best performance with FPR 3.15% and FNR 8.77%. SPDX-VT’s reliance on a predefined compatibility graph covering only a limited license set results in an FNR of 72.89%. DIKE, LiDetector, and LiResolver adopt LSan-based multi-label classification, Bi-LSTM sequence labeling, and RoBERTa-based entity modeling respectively, with FNR decreasing progressively, yet all three remain substantially behind L³icNexus and MCP-Licentra due to limited semantic representation capacity. MCP-Licentra and L³icNexus achieve comparable FPR values of 3.15% and 3.31%, while MCP-Licentra reduces FNR by 4.36%, indicating that the detection gap between the two methods is primarily attributable to false negatives. This FNR reduction is attributed to supplementary term fine-tuning, which enhances the model’s sensitivity to conflicts involving AI-specific license terms and compensates for the missed detections of L³icNexus on such cases.

5.4.2 Compositional license conflict detection

In real-world agent workflows, a task typically activates only a subset of registered MCP services, requiring conflict detection at both combination and subset levels. We therefore apply MCP-Licentra to all non-empty service subsets in the MCDD ground-truth dataset. For a combination of m services, exact verification requires checking $2^m - 1$ subsets. Our cost analysis shows that $m \leq 5$ remains tractable: each group contains at most 31 subsets and requires 5.60 seconds on average, supporting expert annotation, conflict verification, and error analysis. By contrast, $m = 6$ yields 63 subsets with an average detection time of 10.37 seconds, while $m = 10$ yields 1,023 subsets per group, or 102,300 subsets for 100 groups, about 33 times the scale of $m = 5$. Based on this analysis, we use $m \leq 5$ for MCDD construction and exact-verification experiments, balancing verification cost, annotation reliability, and experimental controllability. This setting is a practical evaluation choice rather than a theoretical limit of MCP-Licentra. Here, m denotes the number of MCP services involved in a task or verification instance, not the total number configured in the agent environment.

■ **Table 7** Compositional license conflict detection results.

Combination	1-MCP		2-MCP		3-MCP		4-MCP		5-MCP		Total	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
1-MCP	5.71	3.08	—	—	—	—	—	—	—	—	5.71	3.08
2-MCP	4.29	3.85	6.45	2.90	—	—	—	—	—	—	4.95	3.52
3-MCP	4.67	3.11	4.55	2.83	7.41	2.74	—	—	—	—	4.95	2.93
4-MCP	4.41	3.41	4.17	3.19	5.21	3.62	7.69	2.70	—	—	4.67	3.33
5-MCP	4.73	3.63	4.35	3.54	5.15	3.39	5.83	3.16	8.70	2.60	4.96	3.40
Total	4.64	3.46	4.42	3.29	5.34	3.41	6.16	3.08	8.70	2.60	4.90	3.33

■ **Table 8** License conflict resolution results.

Method	#Resolved	RSR(%)	ARN
LexicalMatch	193	53.91	1.67
StaticOrder	243	67.88	1.54
SinglePass	162	45.25	1.39
MCP-Licentra	255	71.23	1.42

Table 7 reports detection results by combination size and subset size, with 100 combinations evaluated for each setting. A 5-MCP combination contains 31 non-empty subsets and requires 5.6 seconds on average for detection. To evaluate MCP-Licentra in a realistic workflow, a project-management and document-synchronization agent is constructed with DeepSeek-V4-Pro, GitHub Projects [29], Filesystem [28], Todoist [32], and Notion [31]. When only GitHub Projects and Filesystem are activated, MCP-Licentra correctly identifies the GPL-2.0 and Apache-2.0 incompatibility, demonstrating its ability to detect conflicts within task-activated service subsets. As subset size increases, FPR rises while FNR decreases, reflecting more conservative conflict judgments under increasingly complex license interactions. For 5-MCP combinations, FPR increases from 4.73% for 1-MCP subsets to 8.70% for 5-MCP subsets, while FNR decreases from 3.63% to 2.60%. Overall, MCP-Licentra achieves an FPR of 4.90% and an FNR of 3.33%, indicating stable detection performance across subset sizes.

5.5 License conflict mitigation evaluation

This section evaluates MCP-Licentra’s conflict mitigation capability on 358 conflict groups randomly sampled from the detection results in Section 5.4. We set the candidate pool size to $k = 10$ and use $\tau = 0.75$ as the similarity threshold based on sensitivity analysis. This setting filters low-relevance candidates while preserving stable RSR and ARN; higher thresholds, such as 0.80 and 0.90, reduce candidate coverage and lower RSR.

To assess the contribution of each design in the conflict resolution module, we construct three ablation variants targeting semantic retrieval, priority ordering, and iterative compliance checking. LexicalMatch replaces Qwen3-Embedding-8B retrieval with Jaccard similarity to evaluate the role of semantic retrieval in candidate quality. StaticOrder follows the service order instead of priority scoring to assess the effect of ML-first and minimal-modification principles on resolution efficiency. SinglePass performs only one replacement attempt for each service in C_{conflict} , without iterative rechecking, to evaluate the benefit of iterative compliance verification for multi-point conflicts. All other components remain unchanged.

Table 8 presents the experimental results. MCP-Licentra achieves the best performance, resolving 255 out of 358 conflict groups with an RSR of 71.23% and an ARN of 1.42. A conflict group is counted as resolved when a replacement service satisfies functional similarity, license compatibility, and post-replacement compliance verification. Thus, the remaining

unresolved cases indicate that at least one of these conditions cannot be satisfied within the top- k candidate pool, due to insufficient candidate coverage, low functional similarity, or residual license conflicts after replacement. StaticOrder performs closest to the full method, with an RSR of 67.88% and an ARN of 1.54, indicating that removing priority ordering reduces resolution effectiveness and increases replacement cost. LexicalMatch further drops to an RSR of 53.91% and an ARN of 1.67, confirming that lexical similarity alone cannot reliably retrieve high-quality replacement candidates. SinglePass achieves the lowest RSR of 45.25%, showing that iterative compliance checking is necessary for resolving complex multi-point conflicts. Its ARN of 1.39, slightly lower than that of the full method, suggests that SinglePass mainly succeeds on simpler conflict groups requiring fewer replacements.

Answer to RQ3: MCP-Licentra achieves FPR of 3.15% and 4.90% and FNR of 8.77% and 3.33% on group-level and compositional conflict detection, respectively, and attains an RSR of 71.23% and ARN of 1.42 on conflict resolution, outperforming all ablation baselines and validating the three core design components.

6 Related work

6.1 License identification

License identification is fundamental to open source compliance. FOSSology [8] identifies licenses by aligning source text against templates using the binary Symbolic Alignment Matrix. Ninka [10] performs sentence-level matching in source code files. ScanCode [1] compiles a large-scale index of license texts and detection rules. SPDX [39] provides standardized license identifiers, TLDRLegal [40] summarizes terms in plain language, and OSI [25] maintains a curated list of certified licenses. These tools are designed for source code and package scenarios, leaving MCP services uncovered. MCP-Licentra addresses this gap by leveraging LLMs to identify and standardize license information for MCP services.

6.2 License compliance analysis

Ensuring compliance with all dependent license terms is central to open source research. Kapitsaki et al. [27] develop a graphical tool for visualizing license compatibility. Cui et al. [5] propose DIKE for automated conflict detection via NLP, with follow-up work [4] leveraging LLMs to improve accuracy. Other studies examine incompatibility in ecosystems such as NPM [33] and PyPI [43]. However, these approaches target conflicts in source code and package dependencies, relying on predefined rules or expert knowledge within traditional supply chains. In MCP settings, third-party services are combined within workflow configurations under distinct license terms, posing challenges that existing methods do not address. MCP-Licentra employs LLMs for license compatibility analysis across MCP services and recommends compatible alternatives via semantic retrieval.

7 Threats to Validity

Internal validity. The primary threat to internal validity lies in the annotation quality of training data. To mitigate this, we employ multiple annotators with cross-validation and resolve disagreements through discussion. Additionally, the fine-tuning process may introduce bias toward specific license styles; we address this by incorporating diverse license texts during training.

External validity. Our evaluation is based on 3,811 MCP services and 500 dynamic combination scenarios, which may not fully represent all real-world deployments. The generalizability of our findings to other service ecosystems or emerging license types requires further investigation.

8 Conclusion

This paper formalizes license conflict analysis in MCP-based agent workflows as a power-set legality verification problem. We propose MCP-Licentra, which integrates domain-adapted license understanding, compositional conflict detection over service subsets, and iterative conflict resolution. We further extend the license-term taxonomy and construct two datasets, LUD and MCDD, for license understanding and multi-service conflict evaluation. Experiments show that MCP-Licentra consistently outperforms existing baselines across license understanding, conflict detection, and resolution. Future work will focus on scaling to larger service ecosystems and adapting to evolving license and compliance requirements.

References

- 1 AboutCode Project. ScanCode – Open Source Code Scanning Engine. <https://aboutcode.org/scancode/>, 2024. Accessed: May 16, 2026.
- 2 Anthropic. Claude 3.5 Sonnet. <https://www.anthropic.com/news/claude-3-5-sonnet>, 2024. Accessed: May 16, 2026.
- 3 Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 interactive presentation sessions*, pages 69–72, 2006.
- 4 Xing Cui, Jingzheng Wu, Xiang Ling, Tianyue Luo, Mutian Yang, and Wenxiang Ou. 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)*, pages 291–302. IEEE, 2025. doi:10.1109/ICSE-COMPANION66252.2025.00083.
- 5 Xing Cui, Jingzheng Wu, Yanjun Wu, Xu Wang, Tianyue Luo, Sheng Qu, Xiang Ling, and Mutian Yang. 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)*, pages 495–505. IEEE, 2023. doi:10.1109/ICSE-SEIP58684.2023.00050.
- 6 DeepSeek-AI. DeepSeek-R1. <https://huggingface.co/deepseek-ai/DeepSeek-R1>, 2025. Accessed: May 16, 2026.
- 7 Herman Errico, Jiquan Ngiam, and Shanita Sojan. Securing the model context protocol (mcp): Risks, controls, and governance. *arXiv preprint*, 2025. doi:10.48550/arXiv.2511.20920.
- 8 FOSSology Project. FOSSology – Open Source License Compliance Software. <https://www.fossology.org/>, 2024. Accessed: May 16, 2026.
- 9 GR Gangadharan, Vincenzo D’Andrea, Stefano De Paoli, and Michael Weiss. Managing license compliance in free and open source software development. *Information Systems Frontiers*, 14(2):143–154, 2012. doi:10.1007/S10796-009-9180-1.
- 10 Daniel M German, Yuki Manabe, and Katsuro Inoue. 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*, pages 437–446, 2010.
- 11 Wael H Gomaa, Aly A Fahmy, et al. A survey of text similarity approaches. *international journal of Computer Applications*, 68(13):13–18, 2013.
- 12 Hechuan Guo, Yongle Hao, Yue Zhang, Minghui Xu, Peizhuo Lv, Jiezhi Chen, and Xiuzhen Cheng. A measurement study of model context protocol ecosystem. *arXiv preprint*, 2025. arXiv:2509.25292.

- 13 Yongjian Guo, Puzhuo Liu, Wanlun Ma, Zehang Deng, Xiaogang Zhu, Peng Di, Xi Xiao, and Sheng Wen. Systematic analysis of mcp security. *arXiv preprint*, 2025. doi:10.48550/arXiv.2508.12538.
- 14 Zikang Guo, Benfeng Xu, Chiwei Zhu, Wentao Hong, Xiaorui Wang, and Zhendong Mao. Mcp-agentbench: Evaluating real-world language agent performance with mcp-mediated tools. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40(37), pages 30888–30896, 2026. doi:10.1609/AAAI.V40I37.40347.
- 15 Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189, 2024. doi:10.18653/V1/2024.EMNLP-MAIN.626.
- 16 Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model context protocol (mcp): Landscape, security threats, and future research directions. *ACM Transactions on Software Engineering and Methodology*, 2025.
- 17 Huihao Jing, Haoran Li, Wenbin Hu, Qi Hu, Xu Heli, Tianshu Chu, Peizhao Hu, and Yangqiu Song. Mcip: Protecting mcp safety via model contextual integrity protocol. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1177–1194, 2025. doi:10.18653/V1/2025.EMNLP-MAIN.62.
- 18 Georgia M Kapitsaki, Frederik Kramer, and Nikolaos D Tselikas. Automating the license compatibility process in open source software with spdx. *Journal of systems and software*, 131:386–401, 2017. doi:10.1016/J.JSS.2016.06.064.
- 19 Georgia M Kapitsaki and Demetris Paschalides. Identifying terms in open source software license texts. In *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, pages 540–545. IEEE, 2017. doi:10.1109/APSEC.2017.62.
- 20 Yuan Li, Zhengzhong Liu, and Eric Xing. Data mixing optimization for supervised fine-tuning of large language models. *arXiv preprint*, 2025. doi:10.48550/arXiv.2508.11953.
- 21 Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023. doi:10.1145/3560815.
- 22 Ziang Liu, Xin Liu, Yingli Zhang, Song Li, Weina Niu, Qingguo Zhou, Rui Zhou, and Xiaokang Zhou. Towards a comprehensive framework for verifying open-source software license compatibility. *Empirical Software Engineering*, 31(1):15, 2026. doi:10.1007/S10664-025-10751-W.
- 23 Meta AI. Llama-4-Scout-17B-16E. <https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E>, 2025. Accessed: May 16, 2026.
- 24 Grigoris Ntousakis, Julian James Stephen, Michael V Le, Sai Sree Laya Chukkapalli, Teryl Taylor, Ian M Molloy, and Frederico Araujo. Securing mcp-based agent workflows. In *Proceedings of the 4th Workshop on Practical Adoption Challenges of ML for Systems*, pages 50–55, 2025. doi:10.1145/3766882.3767177.
- 25 Open Source Initiative. Open Source Initiative (OSI). <https://opensource.org/>, 2024. Accessed: May 16, 2026.
- 26 OpenAI. GPT-4.5. <https://openai.com/index/gpt-4-5-system-card/>, 2025. Accessed: May 16, 2026.
- 27 Demetris Paschalides and Georgia M Kapitsaki. Validate your spdx files for open source license violations. In *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, pages 1047–1051, 2016. doi:10.1145/2950290.2983939.
- 28 PulseMCP. Filesystem MCP Server. <https://www.pulsemcp.com/servers/quarkiverse-filesystem>, 2025. Accessed: May 16, 2026.
- 29 PulseMCP. GitHub Projects MCP Server. <https://www.pulsemcp.com/servers/taylor-lindores-reeves-github-projects>, 2025. Accessed: May 16, 2026.
- 30 PulseMCP. MCP Server Directory. <https://www.pulsemcp.com/servers>, 2025. Accessed: May 16, 2026.
- 31 PulseMCP. Notion MCP Server. <https://www.pulsemcp.com/servers/notion>, 2025. Accessed: May 16, 2026.

- 32 PulseMCP. Todoist MCP Server. <https://www.pulsemcp.com/servers/todoist>, 2025. Accessed: Mar 16, 2026.
- 33 Shi Qiu, Daniel M German, and Katsuro Inoue. Empirical study on dependency-related license violation in the javascript package ecosystem. *Journal of Information Processing*, 29:296–304, 2021. doi:10.2197/IPSJJIP.29.296.
- 34 Qwen Team. Qwen3-32B. <https://huggingface.co/Qwen/Qwen3-32B>, 2025. Accessed: May 16, 2026.
- 35 Brandon Radosevich and John Halloran. Mcp safety audit: Llms with the model context protocol allow major security exploits. *arXiv preprint*, 2025. doi:10.48550/arXiv.2504.03767.
- 36 RAIL Initiative. Responsible AI Licenses (RAIL). <https://www.licenses.ai/>, 2024. Accessed: May 16, 2026.
- 37 Partha Pratim Ray. A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions. *Authorea Preprints*, 2025.
- 38 Lawrence Rosen. Open source licensing: Software freedom and intellectual property law.(2004).
- 39 SPDX Project. Software Package Data Exchange (SPDX). <https://spdx.dev/>, 2024. Accessed: May 16, 2026.
- 40 TLDRLegal. TLDRLegal – Software Licenses Explained in Plain English. <https://www.tldrlegal.com/>, 2024. Accessed: May 16, 2026.
- 41 Sihan Xu, Ya Gao, Lingling Fan, Linyu Li, Xiangrui Cai, and Zheli Liu. Liresolver: License incompatibility resolution for open source software. In *Proceedings of the 32nd ACM SIGSOFT international symposium on software testing and analysis*, pages 652–663, 2023. doi:10.1145/3597926.3598085.
- 42 Sihan Xu, Ya Gao, Lingling Fan, Zheli Liu, Yang Liu, and Hua Ji. Lidetector: License incompatibility detection for open source software. *ACM Transactions on Software Engineering and Methodology*, 32(1):1–28, 2023. doi:10.1145/3518994.
- 43 Weiwei Xu, Hao He, Kai Gao, and Minghui Zhou. Understanding and remediating open-source license incompatibilities in the pypi ecosystem. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 178–190. IEEE, 2023. doi:10.1109/ASE56229.2023.00175.
- 44 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint*, 2025. arXiv:2505.09388.
- 45 Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint*, 2025. doi:10.48550/arXiv.2506.05176.