"They've Stolen My GPL-Licensed Model!": Toward Standardized and Transparent Model Licensing

Anonymous Author(s)

Abstract

As model parameter sizes reach the billion-level range and their training consumes zettaFLOPs of computation, components reuse and collaborative development are become increasingly prevalent in the Machine Learning (ML) community. These components, including models, software, and datasets, may originate from various sources and be published under different licenses, which govern the use and distribution of licensed works and their derivatives. However, commonly chosen licenses, such as GPL and Apache, are software-specific and are not clearly defined or bounded in the context of model publishing. Meanwhile, the reused components may also have free-content licenses and model licenses, which pose a potential risk of license noncompliance and rights infringement within the model production workflow. In this paper, we propose addressing the above challenges along two lines: 1) For license analysis, we have developed a new vocabulary for ML workflow management and encoded license rules to enable ontological reasoning for analyzing rights granting and compliance issues. 2) For standardized model publishing, we have drafted a set of model licenses that provide flexible options to meet the diverse needs of model publishing. Our analysis tool is built on Turtle language and Notation3 reasoning engine, envisioned as a first step toward Linked Open Model Production Data. We have also encoded our proposed model licenses into rules and demonstrated the effects of GPL and other commonly used licenses in model publishing, along with the flexibility advantages of our licenses, through experiments.

CCS Concepts

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Keywords

License Analysis, AI Licensing, Automated Reasoning

1 Introduction

In recent years, the compelling generalization capabilities provided by billion-parameter models [32], along with the high computational and data costs associated with their training [16], have motivated ML project developers to collaborate incrementally rather than train models from scratch. For example, a common approach is to download a Pre-Trained Model (PTM) [11] and fine-tune it for downstream task [9]. However, these paradigms may face potential legal risks if the use and redistribution practices violate the governing licenses of the reused components, akin to the GPL violation issues in the field of Open Source Software (OSS) [17]. Another risk arises from the choice of license used to republish the work. Some developers adhere to traditional software publishing practices and select OSS licenses for their models [6, 20], which often lack clear definitions and conditions regarding ML activities and do not effectively prevent undesirable use. For example, a licensee can close-source your published models, even if they are licensed under GPL, without violating any terms.

There are three possible ways to address above challenges. First, developers could avoid using any third-party materials. However, this is extremely difficult for individual developers, as training PTMs is expensive and requires vast amounts of data. For instance, the training dataset for GPT-2 [25] was collected from 45 million web pages, governed by various licenses and terms of use. Second, a new licensing framework for ML projects could be developed, which might include drafting specific licenses for models and datasets [2, 5], along with a compatibility table to guide their reuse policies. However, this approach also has limitations, as it does little to address existing conflicts in ML projects that rely on components released under traditional licenses. Furthermore, it is impractical to expect all publishers to relicense their previous works. Third, we can scan the reused components in ML projects and analyze existing license compliance risks to eliminate them. This is a common solution applied to OSS projects [10] but it cannot be directly extended to ML projects. The reason is that ML components can involve complex coupling mechanisms and different licensing frameworks that are interwoven within a project.

Take MixLoRA [13] as an example: it is licensed under Apache-2.0 (an OSS license) and is fine-tuned on Llama2 [30] (governed by Llama2 Community License [18], a model license) using the Cleaned Alpaca Dataset [28], which licensed under CC BY-NC-4.0 (a free content license from Creative Commons [4]). Previous OSS license analysis tools [17, 21] that only consider package reference dependencies and focus on software licenses will fall short in such ML scenarios, which involve implicit nested dependencies and various licensing frameworks. Therefore, to provide license analysis for ML projects, the key is to develop an interpretative solution that can cover all licensing frameworks and disambiguate their mapping rules related to ML activities. Moreover, the lack of consensus in standard model publishing practices and the inflexibility of available model licenses have led many developers to publish their models under OSS licenses or even free-content licenses [3, 8], further increasing the complexity of designing license analysis methods.

In this paper, we propose a two-pronged approach to address these challenges. First, to resolve existing license conflicts in ML projects, we introduce *MG Analyzer*, a tool that constructs ML workflows as Resource Description Framework (RDF) graphs and assesses potential license compliance issues, improper license selection, rights grants, restrictions, and obligations within the projects. Second, to promote standardized model publishing in the future, we propose a new set of model-specific licenses, *MG Licenses*, offering Creative Commons-style licensing options for developers to choose from. To present potential risks of using traditional OSS, model, and free-content licenses in model publishing scenarios, we evaluate them with the *MG Analyzer* on a typical workflow. We also demonstrate the flexibility of *MG Licenses* in encompassing nearly all licensing conditions provided by other model licenses through comprehensive comparisons. The main contributions of our paper are:

- We identify the challenges of license compliance in ML projects and conduct a comprehensive analysis of the deficiencies in current model publishing practices.
- We developed MG Analyzer based on semantic technologies to enable automated reasoning for license analysis in ML projects. It incorporates a dedicated vocabulary capable of describing ML workflows with complex dependencies and the rules of OSS, data, and model licenses. We also provide a interface to convert user-input workflow descriptions into RDF graphs following this vocabulary. Based on these graphs, MG Analyzer constructs dependencies, performs reasoning, and detects potential license conflicts.
- We drafted a set of model licenses called MG Licenses to promote more standardized model publishing. These licenses are well-defined and cover a complete spectrum of model publishing scenarios. Furthermore, we have integrated support for MG Licenses within MG Analyzer.
- To the best of our knowledge, MG Analyzer and Licenses represent the first attempt at standardizing model publishing. The proposed code and license drafts are available at (link temporarily hidden due to double-blind policy).

The rest of the paper is organized as follows.

2 Background and Related Work

2.1 License Compliance Analysis

2.2 Model Licensing

3 MG Analyzer

This section aims to introduce the specific design of MG Analyzer by exploring three questions: *(i) How can we represent the workflows of ML projects? (ii) How do we establish a mapping from license text to reasonable rules? (iii) What types of license compliance issues can arise in ML projects, and how can we detect them?* Before delving into the detailed design that answers these questions, we first provide an overview that serves as a roadmap for this section. As illustrated in Figure 1, the process of MG Analyzer is divided into three main parts: Construction, Reasoning, and Analysis.

In the **Construction Stage**, user-input workflow descriptions (written in Python) are converted into an RDF graph (saved as *gen.ttl* in Turtle format) that contains the base information of the workflow. This conversion is achieved with the help of RDFLib [12] and the MG vocabulary, which is part of our analyzer. RDFLib provides an API for writing RDF graphs, while the MG vocabulary defines the specific semantics to represent the concepts and dependencies in ML projects. Then, we apply reasoning rules (written in Notation3 [1]) for the complete workflow construction using the EYE reasoner [31]. The reasoner concludes new properties that represent the input and output chains between components, enabling further reasoning about the *compositional dependencies* among them (reflecting Question (i); see Section 3.1 for details).

The main tasks in **Reasoning Stage** involve concluding the *definition dependencies* and *rights-using* dependencies. This is achieved through two substeps. First, a new property called *ruling* is created to record the definition of the output work in relation to the input work within the context of licensing. For example, if we merge GPL-licensed code into another software, the resulting work is considered a *derivative* of the original work. This relationship, which we refer to as *definition dependencies*, is crucial for determining the compliant license of the output work. We recursively identify such dependencies and ascertain the licenses of indeterminate works until all works in the workflow have a license. Based on the RDF workflow graph with complete license assignments, we can execute the second step of reasoning, termed *request*, which records *rights-using* dependencies that represent the rights required for the work according to practical reuse methods (reflecting Question (ii); see Section 3.2 for details).

So far, all necessary information for license analysis has been concluded before entering the final **Analysis Stage**. In this stage, MG Analyzer evaluates the validity of the base workflow information, checks for the satisfaction of rights granting, and assesses license compliance and conflicts. *Reports* are genreated to present these results in the workflow graph (reflecting Question (iii); see Section 3.3 for details).

3.1 ML Workflow Representation

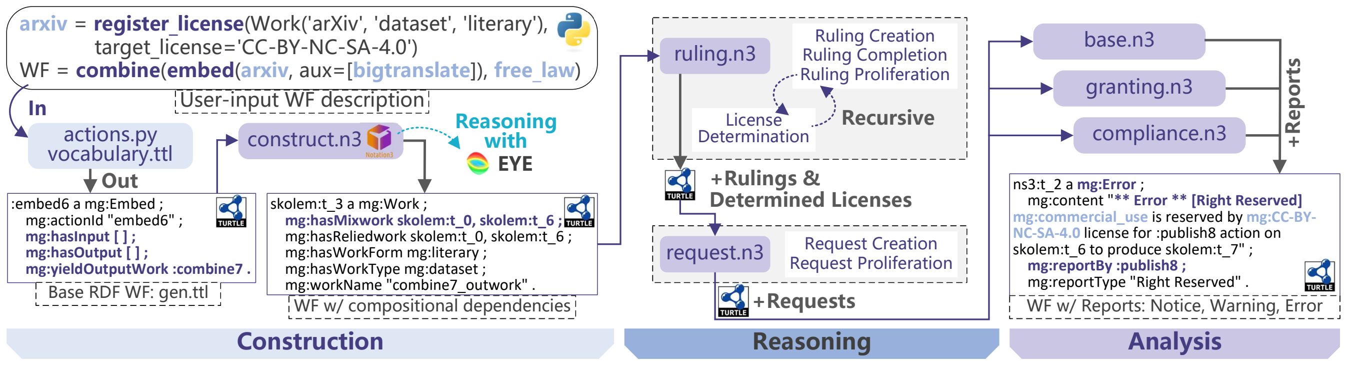
The representation of ML workflows, particularly when considering license analysis scenarios, differs significantly from common software workflows for the following three reasons. ① ML workflows often involve various components (e.g., code, datasets, images, model weights, services), each governed by licenses from different frameworks. Additionally, non-standard licensing practices are prevalent in current ML projects [7, 26], for instance, the C4AI Command R+ model [3] is licensed under a free-content license: CC BY-NC-4.0. Therefore, the representation should be flexible enough to cover such situations.

② The component dependencies in ML workflows may be implicit and nested. For instance, Openjourney [24] is fine-tuned based on the StableDiffusion [27] model and the data generated by Midjourney [23]. In this case, knowledge from Midjourney is transferred to Openjourney without explicit compositional inclusion. Therefore, the representation should consider the multifarious dependencies present within ML projects.

③ The components' dependencies are also defined by the components' practical licenses and the ways they are reused. A common case in the OSS field is that republishing Software as a Service (SaaS) is considered to convey a *derivative* under AGPL-3.0 but has *no definition* under GPL-3.0. Therefore, terms like *derivative* and *independent* should be contextualized within specific licenses, and our representation should be capable of reflecting such meanings.

Therefore, we propose the MG Vocabulary to describe the properties and classes in an ML workflow. For the flexibility issue ①, we use the following terms to abstract key concepts in the workflow:

Work: Represents the components (e.g., models, datasets), each with a unique Type and Form. A work can have a license assigned through a Register License action, or its license can be determined through rules applied in the Reasoning Stage (see Figure 1).



```
arxiv = register_license(Work('arXiv', 'dataset', 'literary'),
          target_license='CC-BY-NC-SA-4.0')
WF = combine(embed(arxiv, aux=[bigtranslate]), free_law)
```

```
:embed6 a mg:Embed ;
  mg:actionId "embed6" ;
  mg:hasInput [ ] ;
  mg:hasOutput [ ] ;
  mg:yieldOutputWork :combine7 .
```

```
skolem:t_3 a mg:Work ;
  mg:hasMixwork skolem:t_0, skolem:t_6 ;
  mg:hasReliedwork skolem:t_0, skolem:t_6 ;
  mg:hasWorkForm mg:literary ;
  mg:hasWorkType mg:dataset ;
  mg:workName "combine7_outwork" .
```

```
ns3:t_2 a mg:Error ;
  mg:content "** Error ** [Right Reserved]
mg:commercial_use is reserved by mg:CC-BY-
NC-SA-4.0 license for :publish8 action on
skolem:t_6 to produce skolem:t_7" ;
  mg:reportBy :publish8 ;
  mg:reportType "Right Reserved" .
```

Figure 1: Overview of MG Analyzer ("mg" is the prefix of our proposed vocabulary).

Action: Represents operations performed on a Work, including Modify, Train and Combine, etc. In practice, we broaden the definition of these operations to make the vocabulary adaptable to different types of works. (See Table 1 for more details.)

Work Type: Includes software, dataset, model, and mixed-type. It is used to describe both the nature of the work and to identify the types of materials intended by a license. We use this information to detect any mismatch between the work type and the license.

Work Form: Divided into three subclasses: Raw, Binary, and Service to provide flexibility. For example, source code, model weights, and corpus fall under Raw; compiled programs are considered Binary; and services like SaaS or online chat LLMs are categorized as Service. Additionally, three general terms are offered: raw-form, binary-form, and service-form, which can work in conjunction with the work type. This approach helps represent situations that lack a formal ontology, such as "a dataset published as a service". We use mixed-form to represent the cases involving collections of works.

LicenseInfo: This contains the essential license information derived from conditions, including the license name, ID, intended types of works, whether it is copyleft or permissive, as well as granted and reserved rights, etc. While the license name is sufficient to describe the base workflow, to enable reasoning, LicenseInfo should bind rules, which we will discuss in the next section.

At this stage, we can describe a base ML workflow, as illustrated[1] schematically in Figure 2 (The RDF graph can be referred to in *gen.ttl* in Figure 1). In this base workflow, the derived input and output of each Action can be represented as blank nodes, serving as placeholders. These placeholders will be populated by reasoning the output yielded by the previous action and the input for the current action, respectively.

For dependency issues discussed in ② and ③, we identify three potential types of dependencies in an ML project: **compositional, definition, and rights-using**. These dependencies are visually represented by different colored dashed arrows in Figure 2. *Compositional dependencies* are categorized into four types: Mixwork, Subwork, Auxwork, and Provenance, each representing the containment relationships between input and output works. For example, when the output work includes the input work or a part of it, the input is considered the Mixwork of the output. This type of dependency is vital for license analysis because all actions performed on the output work, including any rights usage, will proliferate to the Mixwork components. For instance, when fine-tuning a MoE model, the fine-tuning operation will cascade to all submodels. Consequently, the license terms related to fine-tuning for each submodel are triggered, meaning that any constraints or rights imposed on the submodels must be honored. Additionally, if a work includes Mixworks in different forms, such as code and weights, we need to generalize the output's form to raw-form to accommodate these variations. A similar approach applies to the work type as well. Subwork and Auxwork are used to track works that are utilized by other works in the workflow, such as training datasets or distilled models. The key distinction is that Subwork is intended to be published alongside the output, which necessitates additional license analysis related to republishing. Provenance is specifically used for the Register License action to indicate that the output is simply the input itself, bound to a license. In such cases, any further proliferation of dependencies should cease.

The *definition dependencies* represent the relationships between works based on the definitions established by their licenses. For example, if a new work is created by modifying GPL-licensed code, that modification is considered a *derivative* of the original work. These dependencies should be understood in the context of the original license and can extend to subsequent actions if the same conditions are activated again. These dependencies are the main factor in determining the applicable license and restrictions for the output work. For example, under GPL-3.0, the republication of *derivatives* must apply the same license. Additionally, a work may have multiple definition dependencies in a complex workflow, and these dependencies should be simultaneously satisfied (if possible; otherwise, an error should be reported) during license determination. The corresponding implementation in the MG Analyzer is illustrated in the Reasoning Stage of Figure 1. New instance nodes called *Ruling* are created to track the definition dependencies and triggered rules for each work, determining their applicable licenses in an alternating manner.

The *rights-using dependencies* describe the rights that must be granted for actions performed on works. For example, when executing a training action on a model, it requires the rights to *use* and *modify* (termed as Usage in MG vocabulary) from the model's

[1] The "mg" prefix is omitted and some properties are merged or filtered for presentation.

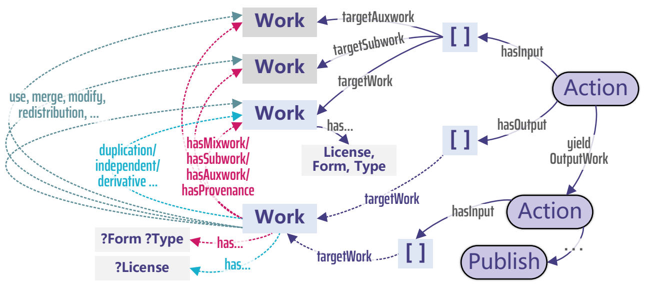


Figure 2: A Typical ML Workflow Represented by MG Analyzer. Dashed arrows with different colors indicate the properties related to three kinds of dependencies: compositional, definition, and rights-using dependencies.

license[2]. Similarly, the rights-using dependencies should proliferate according to compositional dependencies. For instance, the requirement to *modify* a model extends to all its submodels. In the MG Analyzer, we create new nodes called *Request* to represent this dependency. Additionally, it is insufficient to only check the granting rights; the reserved rights must also be verified. Depending on the clarity of the license text, some rights may either be explicitly granted or reserved. Furthermore, certain license clauses can waive the requirement for specific rights. For instance, both GPL-3.0 and CC licenses include automatic relicensing clauses for downstream recipients, which eliminate the need for a *sublicense* right.

By MG Vocabulary, we are able to describe complete ML workflows and represent the necessary dependencies for license analysis. The *compositional dependencies* are license-independent and can be reasoned from the base workflow. However, *definition dependencies* and *rights-using dependencies* are associated with specific rules expressed in natural language within each license. To facilitate automated reasoning for these dependencies, the next step is to develop a viable method for encoding license terms into formal logic rules.

3.2 License Rule Encoding and Reasoning

Typically, licenses are designed to govern the use and distribution of specific types of works. For example, the GPL-3.0 is tailored for source code and object code, while CC licenses focus on literary, musical, and artistic works. As a result, it is challenging to map their rules within a unified framework for logical reasoning. Meanwhile, many ML projects actually incorporate non-standard licensing components, as mentioned in Section 1. If we consider these claimed licenses to be invalid, then they would not pose any license compliance issues. However, the validity of these licenses depends on specific cases and the dispute resolution process by the jurisdictional courts in accordance with the applicable laws in different regions. As a license analyzer, we aim to maximize the detection of all potential legal risks under various interpretations, rather than merely granting a green light with low confidence. To this end, we perform three generalizations to encode license rules.

[2] An example of such rights granting can be found in the Llama2 Community License [18], which states, "You are granted a ... to use, reproduce, distribute, copy, create derivative works of, and make modifications to the Llama Materials.

Table 1: List of Supported Actions in MG Analyzer Following Rule Alignment. The symbols = and ≈ denote that the Types/Forms of output work and input work are the same and may differ, respectively. The corresponding license terms in OSS, Free-content, and Model for each action are listed.

Action	Type	Form	Composition	Terms
Copy	=	=	Output and input are exactly **same**.	Copy Duplicate
Combine	≈	≈	**Entire** input included in output.	Link Aggregate MoE Arrange Collect
Modify	=	=	Output includes a **significant** portion of input and **can be reverted**.	Modify Fine-tune
Amalgamate	=	=	Output includes portions of input but **cannot be reverted**.	Modify Remix Fusion
Train	=	=	Output has the **same structure** as input and may contain a **negligible** portion of it.	Alter Adapt Train
Generate	≈	≈	Output does not contain any portion of input and may **perform differently** from it.	Output Generate Synthetic
Distill	=	=	Output does not contain any portion of input but **performs similarly** to it.	Distill Transfer Extract
Embed	=	=	Output does not contain any portion of input, but there has a **mapping** that converts input to output.	Translate Transform
Publish	=	≈	Output is the **same** as the input but may have a **different form**.	Redistribute Perform Display Disseminate

The first generalization is called *fuzz form matching*. We broaden the definitions related to a work's form to encompass its general form. For instance, we expand the license terms of GPL-3.0 concerning source code to include all forms of work in the Raw categories, such as model weights and corpus. In this way, we can extend the scope of interpretation of GPL-3.0 to cover models and datasets.

The second generalization is called *composition-based rule alignment*, which aims to resolve the pervasive ambiguities across different licensing frameworks. This ambiguity often arises in non-standard scenarios, for example, when licensing a model under GPL-3.0, it may be unclear whether *model aggregation* (a technology used in federated learning [14]) triggers the "Aggregate" clause in GPL-3.0. Therefore, we propose a composition-based method to align these rules. Specifically, we generalize the concept of action to represent the compositional relationships between input and output. For instance, the action *Combine* signifies that the input work has been entirely included in the output work without modification. This action corresponds to terms such as "Link" and "Aggregate" in software licenses, "Collection" in CC licenses, and "MoE" in model licenses. In the case of *model aggregation*, which produces an output that contains parts of the input works and is difficult to separate, it is not considered a *Combine*. Consequently, according to our rule alignment, it will not activate the "Aggregate" clause in GPL-3.0.

The complete rule alignment method utilized in the MG Analyzer can be found in Table 1. It is worth mentioning that the meanings of these actions have been broadened and may differ from their original definitions. In some cases, multiple actions may align with the same license terms. For instance, the license term "Modify" can align with both the actions *Modify* and *Amalgamate*, as such licenses do not distinguish the extent of changes made or whether those changes can be reverted.

The final aspect is *applicable term generalization*, where we encode the triggering conditions of a license term into the following properties: range of input work forms, range of output work forms, and types of actions. Figure 3 illustrates an example of the GPL-3.0 derivatives rule[3], which represents the Combine action applied to input and output works in code format. This action triggers the

[3] The original GPL-3.0 license text reads: "You may convey a work based on the Program ... in the form of source code ... provided that you also meet all of these

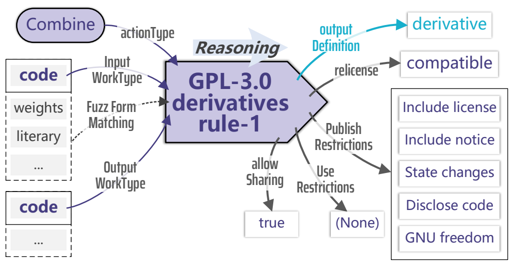


Figure 3: Example of a Generalized GPL-3.0 Derivatives Rule in MG Analyzer.

"derivative" clause in GPL-3.0, indicating that the license of the output work must be compatible with GPL-3.0 (e.g., APGL-3.0). Additionally, five restrictions apply to the output work if it is to be republished, as dictated by this *definition dependency*. This rule does not include any *Use Restrictions*, which apply to output works regardless of whether they are republished. We found this subtle distinction to be crucial in ML license analysis, as most OSS license terms are triggered by distribution, and their definitions of "distribution" typically exclude publishing as a service. However, in the case of models, the common deployment method is through a web interface, leading to many OSS license restrictions being circumvented in such scenarios.

Furthermore, multiple rules may lead to the same output definition, and we also provide the option of *fuzz form matching* within the rules to enhance the interpretive capabilities of the MG Analyzer. It is worth mentioning that the reasoning behind the restrictions derived from the rules requires further analysis for validation. Taking the rule in Figure 3 as an example, whether the final work remains in a Raw form determines whether the warnings about disclosing code should be reported. We present snippets of our encoded rules written in Turtle format in Appendix C. The list of supported licenses, whose terms have been encoded in MG Analyzer, is shown in Table 3, covering nearly all top-ranking licenses for published models on HuggingFace[4].

With the MG Vocabulary, the base ML workflow with compositional dependencies, and encoded license rules, we can reason to derive the definition and rights-using dependencies, thereby determining the applicable licenses for intermediate works. The license determination in the MG Analyzer follows an incremental and minimal non-compliance strategy, where the new license only applies to the incremental parts of the work without affecting the original work (a common practice in licensing). Furthermore, to avoid introducing additional compliance issues during analysis, we use the *Unlicense* as the default when applicable. However, an exception arises with the license proliferation clauses found in copyleft licenses, such as GPL-3.0 and OSL-3.0, which require that the entire new work be licensed under the same terms. Our analyzer incorporates reasoning logic to identify applicable licensing solutions (a snippet of logic in Notation3 can be found in Appendix C), but unresolved conflicts may occur if multiple copyleft clauses are

conditions: ... stating that you modified it ... it is released under this License ... keep intact all notices ... license the entire work, as a whole, under this License ..." .

[4] https://huggingface.co/models

Table 2: List of Notices, Warnings, and Errors reported by MG Analyzer. The triggered work is denoted as ?work.

Code	Report Type	Report Content
N1	Include License	The original license file from ?work should be retained.
N2	Include Notice	The notices (e.g., attribution, copyleft, patent, trademark) from ?work should be retained.
N3	State Changes	A notice stating the modifications made to ?work should be provided.
N4	ImpACT Reports	You need to complete a Derivative Impact Report.
W1	License Type Mismatch	Non-standard licensing of ?work.
W2	Revocable License	The license of ?work is revocable.
W3	Possibly Revocable License	The revocability of the license of ?work is not claimed.
W4	Right Not Granted	The required right is not explicitly granted by ?work.
W5	Disclose Source Code	This work should disclose its source code.
W6	Disclose Unmodified Code	The unmodified source code of ?work should be disclosed.
W7	Use Behavior	The use of this work must comply with the usage behavior restrictions of ?work.
W8	Runtime Control	There is a runtime restriction clause in ?work (e.g., forced updates).
E1	Wrong Work Type or Form	The type of ?work is inconsistent with its form.
E2	Right Reserved	The required right is reserved by the license of ?work.
E3	Not Allowed to Share	Redistribution of this work is prohibited.
E4	Not Allowed to Sublicense	Sublicensing of ?work is prohibited.
E5	Non-Commercial Use	Commercial use of ?work is prohibited.
E6	Cannot Be Relicensed	The license of this work is invalid because ?work cannot be relicensed, or relicensing is prohibited.
E7	GNU Freedom Conflict	The additional terms applied in this work may violate the GNU freedom clauses of ?work.
E8	CC Freedom Conflict	The additional terms applied in this work may violate the CC freedom clauses of ?work.
E9	Llama2/3 Exclusive	Using Llama2/3's output in non-Llama2/3 derivatives is prohibited.
E10	Exclusive License	The additional terms applied in this work are prohibited by the license of ?work.

triggered. In such cases, the MG Analyzer will select one of these copyleft licenses and report an error during the Analysis Stage.

3.3 Compliance Analysis

At this stage, we establish all necessary dependency properties through automated reasoning to enable compliant license analysis. The analysis rules are designed to assess and report the validity of the base workflow information, the fulfillment of granted rights, work restrictions, and overall license compliance. In addition, the Publish action should be invoked to signify the completion of the workflow, along with an assigned public manner and work form. MG Analyzer considers three republication scenarios: internal, share, and sell, each of which typically involves different terms and conditions in the licenses. For instance, if we publish the final work for sale, the related licenses should grant rights for redistribution, sublicensing, and commercial use.

Appendix C presents the logic code for rights granting analysis, and the full list of reported notices, warnings, and errors is shown in Table 2. Due to its staged logic rule reasoning design, MG Analyzer has considerable extensibility, enabling the incorporation of additional licenses and analysis targets, provided that it can reason based on our proposed dependencies information. Beyond compliance analysis, our vision is to promote ML workflow supply chain management and improve FAIRness [33] in model publishing. We position our vocabulary and tool as a first step toward a linked open model production data.

4 MG Licenses

Although the MG Analyzer can identify potential non-compliance in existing ML projects, it does not offer an effective solution to

Table 3: List of MG Analyzer-Supported Licenses & Aggrements (including MG Licenses) with Their Comparisons in Clarity and Freedom. Grouped by OSS, Free-Content (&Dataset), Model and Sorted First by Clarity Score, Then by Freedom Score.

License Name	Clarity of Definitions				Freedom of Verbatim Copy			Freedom of Derivative					Freedom of Use		Clarity	Freedom
	Prefixes	Rights	Rules	Remote	Share	Close	Non-excl	Share	Close	Non-excl	Sublicense	Attribute	Comm	Behav		
AGPL-3.0	✓	✓	≈	✓	≈	✗	≈	≈	✗	≈	≈	✗	✓	✓	3.5	4.5
AFL-3.0	≈	✓	≈	✓	≈	✓	✓	≈	≈	✓	✓	✗	✓	✓	3.0	7.5
OSL-3.0	≈	✓	≈	✓	≈	✓	✗	≈	✗	✗	✓	✗	✓	✓	3.0	5
Apache-2.0	✓	✓	≈	✗	≈	✓	✓	≈	≈	✓	✓	✗	✓	✓	2.5	7.5
LGPL-3.0	✓	✓	≈	✗	≈	✗	✓	≈	✓	✓	≈	✓	✓	✓	2.5	7.5
Artistic-2.0	≈	≈	≈	✓	≈	✗	✓	≈	✗	✓	✓	✗	✓	✓	2.5	6.0
GPL-3.0	✓	✓	≈	✗	≈	✗	≈	≈	✗	≈	≈	✗	✓	✓	2.5	4.5
ECL-2.0	≈	✓	≈	✗	≈	✓	✓	≈	≈	✓	✓	✗	✓	✓	2.0	7.5
Unlicensed	≈	≈	≈	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	1.5	10
MIT	≈	≈	≈	✗	≈	✓	✓	✓	✓	✓	✓	✗	✓	✓	1.5	8.5
GPL-2.0	✗	✓	≈	✗	≈	✗	✓	≈	✗	✓	≈	✗	✓	✓	1.5	5.5
LGPL-2.1	✗	✓	≈	✗	≈	✗	✓	≈	✗	✓	≈	✗	✓	✓	1.5	5.5
BSD-3-Clause	✗	≈	≈	✗	≈	✓	✓	≈	✓	✓	✓	✗	≈	✓	1.0	7.5
BSD-3-Clause-Clear	✗	≈	≈	✗	≈	✓	✓	≈	✓	✓	✓	✗	≈	✓	1.0	7.5
BSD-2-Clause	✗	≈	≈	✗	≈	✓	✓	≈	✓	✓	✓	✗	≈	✓	1.0	7.5
WTFPL-2.0	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0	10
CC0-1.0	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓*	✓	✓	✓	3.0	10
ODC-By-1.0	≈	≈	✓	✓	≈	✓	✓	≈	✓	✓	≈	✗	✓	✓	3.0	7.5
PDDL-1.0	✓	≈	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	2.5	10
CC-BY-4.0	✓	✓	≈	✗	≈	✓	≈	≈	≈	≈	≈	✗	✓	✓	2.5	6.0
CC-BY-SA-4.0	✓	✓	≈	✗	≈	✓	≈	≈	≈	≈	≈	✗	✓	✓	2.5	6.0
CC-BY-NC-4.0	✓	✓	≈	✗	≈	✓	≈	≈	≈	≈	≈	✗	✗	✓	2.5	5.0
CC-BY-NC-SA-4.0	✓	✓	≈	✗	≈	✓	≈	≈	≈	≈	≈	✗	✗	✓	2.5	5.0
CC-BY-ND-4.0	✓	✓	≈	✗	≈	✓	≈	✗	n/a**	n/a	n/a	n/a	✓	✓	2.5	4.0
CC-BY-NC-ND-4.0	✓	✓	≈	✗	≈	✓	≈	✗	n/a**	n/a	n/a	n/a	✗	✓	2.5	3.0
GFDL	✓	≈	≈	✗	≈	✗	✗	≈	✗	✗	≈	✗	✓	✓	2.0	3.5
C-UDA	✓	✗	≈	✗	✓	✓	✓	≈	✓	✓	✗	✗	✗	✗	1.5	5.5
LGPLLR	✗	✓	≈	✗	≈	✓	✗	≈	≈	✗	≈	✗	≈	✓	1.5	4.5
MG0	✓	✓	✓	✓	≈	✓	✓	≈	✓	✓	✓	≈	✓	✓	4.0	8.5
MG-BY	✓	✓	✓	✓	≈	✓	✓	≈	≈	✓	✓	✗	✓	✓	4.0	7.5
MG-BY-NC	✓	✓	✓	✓	≈	✓	✓	≈	✓	✓	≈	✗	✗	✓	4.0	6.5
MG-BY-RAI	✓	✓	✓	✓	≈	✓	✓	≈	≈	✓	✓	✗	✓	✗	4.0	6.5
‡ OpenRAIL-M	✓	✓	✓	✓	≈	✓	✓	≈	≈	✓	✓	✗	✓	✗	4.0	6.5
MG-BY-OS	✓	✓	✓	✓	≈	✗	✓	≈	✗	✓	✓	✗	✓	✓	4.0	6.0
MG-BY-NC-RAI	✓	✓	✓	✓	≈	✓	✓	≈	✓	✓	≈	✗	✗	✗	4.0	5.5
MG-BY-NC-OS	✓	✓	✓	✓	≈	✗	✓	≈	✗	✓	≈	✗	✗	✓	4.0	4.5
MG-BY-ND	✓	✓	✓	✓	≈	✓	✓	✗	n/a	n/a	n/a	n/a	✓	✓	4.0	4.5
MG-BY-NC-ND	✓	✓	✓	✓	≈	✓	✓	✗	n/a	n/a	n/a	n/a	✗	✓	4.0	3.5
† OPT-175B	✓	≈	✓	✓	≈	✓	✓	≈	✓	✓	✗	✗	✗	✗	3.5	5.0
† Llama3	≈	✗	✓	✓	≈	✓	✓	≈	✓	✓	✗	✗	≈	✗	2.5	5.5
† Llama3.1	≈	✗	✓	✓	≈	✓	✓	≈	✓	✓	✗	✗	≈	✗	2.5	5.5
† • AI2-ImpACT-LR	≈	✗	✓	✓	≈	✓	✓	≈	✓	✓	✗	✗	≈	✗	2.5	5.5
† • AI2-ImpACT-MR	≈	✗	✓	✓	✗	n/a	n/a	≈	✓	✓	✗	✗	✗	✗	2.5	2.0
† • AI2-ImpACT-HR	≈	✗	✓	✓	✗	n/a	n/a	✗	n/a	n/a	n/a	n/a	✗	✗	2.5	0
† Gemma	✗	✗	✓	✓	≈	✓	✓	≈	≈	✓	✗	✗	≈	✗	2.0	4.5
† Llama2	≈	✗	✓	✗	≈	✓	✓	≈	✓	✓	✗	✗	≈	✗	1.5	5.5

Header Definitions:
Prefixes: ✓ The license explicitly includes sufficient prefixes that clearly describe scope and conditions of granting rights (e.g., revocable, sublicensable); ≈ Some important prefixes are indeterminate; ✗ No prefixes are declared.
Rights: ✓ The license explicitly declares whether a patent license or a copyright license is granted; ≈ Only the granting of a patent license or copyright license is stated; ✗ No explicit grant of either is provided.
Rules: ✓ The license terms cover all actions listed in Table 1; ≈ Some actions fall outside the definition of this license; ✗ Almost no rules are set forth.
Remote: ✓ The license considers remote access situations (e.g., via API, Web, SaaS); ✗ No definitions or rules regarding remote access behaviors are set forth.
Share: ✓ The license permits the sharing of verbatim copies/derivatives created by you without any restrictions; ≈ Some restrictions apply to sharing; ✗ Sharing verbatim copies/derivatives is prohibited.
Close: ✓ The license does not require you to disclose the source files of verbatim copies/derivatives created by you; ≈ Modification statements are required; ✗ You must disclose the source files of your created copies/derivatives.
Non-exclusive: ✓ The license does not restrict you from adding new terms when republishing; ≈ Certain types of terms are prohibited in republishing; ✗ All republishing must adhere to the original terms and conditions.
Sublicense: ✓ The license explicitly grants sublicensing rights; ≈ The license prohibits sublicensing but offers automatic licensing instead; ✗ Sublicensing is either prohibited or not explicitly permitted.
Attribute: ✓ The license does not require retaining the original attribution and licenses in redistributed derivatives; ≈ Attribution or license must be retained; ✗ Redistributed derivatives must retain the attributions and licenses.
Commercial: ✓ The license explicitly grants commercial rights; ≈ Commercial rights are not explicitly granted but not reserved either, or compromised commercial rights are granted; ✗ Commercial rights are reserved.
Behavioral: ✓ The license does not restrict user behaviors; ≈ Includes runtime controls (e.g., forced updates); ✗ Certain behaviors involving the licensed materials or derivatives are prohibited (e.g., harming, medical advice).
Clarity/**Freedom** Score: ✓ +1.0, ≈ +0.5, ✗ +0, n/a: +0 . Maximum Clarity Score: 4.0, Maximum Freedom Score: 10.
Explanations:
* Although CC0-1.0 explicitly states that sublicensing is not allowed, sublicensing becomes unnecessary due to the Waiver of Rights.
** Since CC-BY-ND-4.0 and CC-BY-NC-ND-4.0 prohibit the sharing of derivatives, judgments regarding redistributed derivatives are marked as "n/a" in the table.
† These licenses (or terms of use, or agreements) are specifically drafted for certain products and are not intended for general model publishing purposes.
‡ As there are no fundamental differences between CreativeML Open RAIL-M, OpenRAIL++-M, BigCode Open RAIL-M, BigScience RAIL, and BigScience Open RAIL-M, these licenses are grouped under OpenRAIL-M.
• We have used an archive of the AI2 ImpaACT license; the version is 2.0, with an effective date of January 8, 2024.

prevent such issues in the future. To address this, we conducted a survey of the most widely used licenses for models published on HuggingFace and identified three major causes of license non-compliance in current ML projects: non-standard licensing, lack of general model licenses, and insufficiently defined licenses. The statistical results of a previous study [7] support part of our findings.

To reveal the underlying dilemmas in model licensing, we provide comprehensive comparisons of these licenses in Table 3. Based on the terms of these licenses, we evaluated each license's clarity score and freedom score, each encompassing sub-items as defined in the table. A higher clarity score indicates that the license is more clearly defined in model publishing scenarios, while a higher freedom[5] score signifies fewer restrictions on republished copies and derivatives. The significant findings are summarized below:

① **For OSS licenses**, most are not well defined in the context of model publishing, primarily due to the absence of clauses addressing ML activities (Rules) and the lack of coverage for publishing as a service (Remote). This implies that mainstream model deployment practices, which often provide models as web services, are likely to circumvent the governance of these licenses. Additionally, commercial use and behavioral restrictions are not stipulated in most OSS licenses, which are common requirements in model publishing.

② **For free-content and dataset licenses**, their average clarity score is only slightly better than that of OSS licenses, and they

[5] Our freedom score reflects only the amount of restrictions stipulated in a license and should not be confused with the definitions of freedom in free software [22].

also lack coverage for publishing as a service. However, some CC licenses, such as CC-BY-NC-4.0, offer additional options that prohibit the commercial use of the work, unlike OSS licenses. This may explain why many models are licensed under these licenses[6], despite the fact that they were not originally drafted for models.

③ **For model licenses**, aside from the proposed MG licenses, most model licenses are not intended for general publishing purposes. For example, the terms in Llama2 license is specifically drafted to govern the Llama2 model and its derivatives, failing to meet reusable standards. Additionally, since the purpose of these licenses is often to protect the IP rights of proprietary models, they are usually revocable and prohibit sublicensing. Although a set of well-defined licenses known as OpenRAIL-M [5] exists, their nearly identical rules make it difficult to accommodate the diverse needs in model publishing.

Based on the above findings, we conclude that it is necessary to draft new standardized and flexible licenses for general model publishing. To address this, we collaborated with a law firm to draft a set of model licenses, tentatively referred to as MG Licenses[7]. As reflecting in Table 3, our licenses include specific definitions and terms related to ML concepts, offering greater clarity compared to OSS and free-content licenses. Most importantly, following the philosophy of CC licenses, we designed MG Licenses to be flexible and easy to use, providing five options to fullfill various publishing scenarios. The options include: BY (Attribution), NC (Non-Commercial), ND (No Derivatives), RAI (Responsible Use of AI), and OS (Open Source[8]). We drafted nine preset licenses using these options, including MG0, which does not apply any options. The flexibility of our licenses is reflected in table 3, where their freedom scores are evenly distributed between 3.5 and 8.5, indicating that they form a superset of all other model licenses[9]. To promote transparency, we introduced a *Model Sheet* as an attachment to each MG License, inspired by MDL [2]. This sheet assists model users in understanding the rights and restrictions granted by the license terms and helps model developers identify the most suitable license for their needs.

In the next section, we explore which licenses can protect your model from misuse beyond your intended purposes. For example, licensee may close source your GPL-licensed model without violating your license, even if it does contains code disclosure clauses. To do this, we evaluate some of the popular licenses for model publishing, as well as the MG licenses, using MG Analyzer.

5 Experiments

This section seeks to answer a key question: *Should I continue using traditional OSS and free-content licenses to publish my model, and what are the associated risks?* To explore this question, we evaluate commonly adopted licenses in the context of model publishing by MG Analyzer to assess whether they are still effective as intended. The example workflow involves combining two models and publishing them as a service, as shown in Figure 4. Here, we consider

[6] There are 9659 models on HuggingFace licensed under CC-BY-NC-4.0 (more than Llama2), and 668 of them have garnered 1k+ downloads (accessed on October 8, 2024).
[7] Link to license text temporarily hidden due to double-blind policy.
[8] At the time of writing, the Open Source Initiative has only released a draft v. 0.0.9 version of the Open Source AI definition, and our MG licenses with the OS option are not OSI-approved at present.
[9] AI2-ImpACT-MR and AI2-ImpACT-HR prohibit sharing copies, we do not consider such restrictions to be common in model publishing, so we have excluded them.

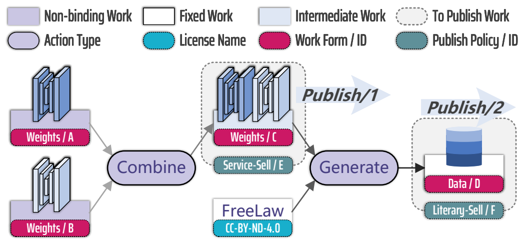


Figure 4: Example Workflow: Combine Models, Then Publish.

Table 4: Settings and Analysis Results from MG Analyzer with Fuzz Form Matching Enabled.

Work: License	Work: Report code.
(i) C: AGPL-3.0. D: Unlicense. (ii) D: Unlicense	(i) E: N1N2N3×2; **W5**×2; W1×3. F: W1×3. (ii) E: W1×2. F: W1.
OSS License Setting: (i) A←PhoBERT [19]: AGPL-3.0. B←CKIP-Transformers [15]: GPL-3.0. (ii) A←PhoBERT [19]: AGPL-3.0. B←*None.*	
C: Unlicense. D: Unlicense.	E: W1×2; **E2**×2; **E5**×2. F: W1×2 .
Free-content License Setting: A← MPT-Chat [29]: CC-BY-NC-SA-4.0. B←Command R+ [3]: CC-BY-NC-4.0.	
(iii) D: Unlicense. (iv) D: Unlicense.	(iii) E: N1; N2; **W5** . F: *None.* (iv) E: N1; N2; W2×2; **E2**; **E5**. F: W2, **E5**.
Model License Setting: (iii) A← MG-BY-OS. B←*None.* (iv) A← MG-BY-NC. B←*None.*	

two scenarios: 1) publishing the combined model as a service; 2) publishing the data generated from the combined model. Models A and B are non-binding works that correspond to the settings in Table 4, with their respective analysis results also presented in the table. Model C is an intermediate work created by combining A and B, and Data D is the generated output from C. Work E involves republishing C as a service with the intent to sell, while Work F involves republishing D as a literary form, also with the intent to sell. To be more convincing, we use real-world models and their respective licenses for demonstration.

First, we evaluate two OSS licenses: AGPL-3.0 and GPL-3.0, which are considered enforceable open-source licenses with copyleft clauses. In setting (i), Work E triggers two *Disclose Source Code* warnings (code W5, refer to Table 2 for code definitions), and AGPL-3.0 successfully proliferates[10] to Work C. However, with a small adjustment, we can circumvent these clauses by republishing the generated content rather than providing the model as a service. As demonstrated by Work D, there is no W5 warning, and the content is licensed under the Unlicense. Furthermore, the condition in AGPL-3.0 that triggers the disclose code clauses related to remote access is *you modify the Program*, which means you can directly republish copies as a service to circumvent this clause. As reflected in setting (ii), there are no more W5 warnings, only two warnings related to the non-standard licensing remain.

Second, we evaluate two free-content licenses: CC-BY-NC-SA-4.0 and CC-BY-NC-4.0, both of which prohibit the commercial use of the governed work. As shown in the results, the republication of Work E successfully triggers E2 errors because the rights to

[10] While two copyleft conditions are simultaneously triggered here, they can be resolved because GPL-3.0 is compatible with AGPL-3.0.

commercial use are reserved. However, we can still circumvent these clauses by generating and then sharing the output, as these licenses lack rules regarding the generated work.

Third, we evaluate MG Licenses: MG-BY-OS and MG-BY-NC, which contain open sourcing and non-commercial use clauses, respectively. In setting (iii), our MG-BY-OS license successfully triggers the W5 warning, indicating that Work E must disclose its source code. In setting (iv), the non-commercial use error E5 is reported by the generated Work F. As a model license, we do not enforce licensing on generated content, allowing Work D to be licensed under the Unlicense, albeit with certain restrictions. A summary of the rights granted and restrictions imposed by these licenses can be found in their *Model Sheet* provided in Appendix B.

It is worth mentioning that all results were obtained with fuzzy form matching enabled, maximizing the detection of potential risks. If these fuzz rules were disabled, fewer issues would be reported. Furthermore, our MG Analyzer is designed to help developers be aware of potential compliance issues in ML projects. Its results should not be considered legal advice or a defense in dispute resolution. Please refer to our disclaimers in Appendix A.

Summary: GPL, AGPL, and CC licenses can be easily circumvented, leading to unintended misuse of the ML models they govern. In contrast, MG Licenses offer greater clarity and flexibility tailored to various publishing scenarios, promoting a more standardized and transparent approach to model licensing.

6 Conclusion

Non-standard licensing is prevalent in ML projects, and the underlying risks are often neglected. To reveal these risks, we propose a vocabulary to describe ML workflows and develop the MG Analyzer to detect compliance issues based on it. To promote more standardized licensing in the future, we have drafted MG Licenses to provide flexible licensing solutions for model publishing. Our experiments show that commonly used OSS and CC licenses are unsuitable for model publishing, while MG Licenses provide a viable alternative.

References

[1] Dörthe Arndt and Stephan Mennicke. 2023. Notation3 as an existential rule language. In *International Joint Conference on Rules and Reasoning (RuleML+RR)*. Springer, 70–85. https://doi.org/10.1007/978-3-031-45072-3_5

[2] Misha Benjamin, Paul Gagnon, Negar Rostamzadeh, Chris Pal, Yoshua Bengio, and Alex Shee. 2019. Towards standardization of data licenses: The montreal data license. *arXiv preprint arXiv:1903.12262* (2019).

[3] CohereForAI. 2024. C4AI Command R+. Retrieved October 1, 2024 from https://huggingface.co/CohereForAI/c4ai-command-r-plus

[4] Creative Commons. 2024. Creative Commons Licenses List. Retrieved October 1, 2024 from https://creativecommons.org/licenses/

[5] Danish Contractor, Daniel McDuff, Julia Katherine Haines, Jenny Lee, Christopher Hines, Brent Hecht, Nicholas Vincent, and Hanlin Li. 2022. Behavioral use licensing for responsible AI. In *2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*. 778–788. https://doi.org/10.1145/3531146.3533143

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 4171–4186. https://doi.org/10.18653/v1/n19-1423

[7] Moming Duan, Qinbin Li, and Bingsheng He. 2024. ModelGo: A Practical Tool for Machine Learning License Analysis. In *Proceedings of the ACM on Web Conference 2024*. 1158–1169. https://doi.org/10.1145/3589334.3645520

[8] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. OLMo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838* (2024).

[9] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*.

[10] Michael C Jaeger, Oliver Fendt, Robert Gobeille, Maximilian Huber, Johannes Najjar, Kate Stewart, Steffen Weber, and Andreas Wurl. 2017. The FOSSology project: 10 years of license scanning. *International Free and Open Source Software Law Review* 9 (2017), 9.

[11] Wenxin Jiang, Nicholas Synovic, Matt Hyatt, Taylor R Schorlemmer, Rohan Sethi, Yung-Hsiang Lu, George K Thiruvathukal, and James C Davis. 2023. An empirical study of pre-trained model reuse in the hugging face deep learning model registry. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE)*. 2463–2475. https://doi.org/10.1109/ICSE48619.2023.00206

[12] Daniel Krech, Gunnar AAstrand Grimnes, Graham Higgins, Jörn Hees, Iwan Aucamp, Niklas Lindström, Natanael Arndt, Ashley Sommer, Edmond Chuc, Ivan Herman, Alex Nelson, Jamie McCusker, Tom Gillespie, Thomas Kluyver, Florian Ludwig, Pierre-Antoine Champin, Mark Watts, Urs Holzer, Ed Summers, Whit Morriss, Donny Winston, Drew Perttula, Filip Kovacevic, Remi Chateauneu, Harold Solbrig, Benjamin Cogrel, and Veyndan Stuart. 2023. *RDFLib*. https://doi.org/10.5281/zenodo.6845245

[13] Dengchun Li, Yingzi Ma, Naizheng Wang, Zhiyuan Cheng, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. 2024. MixLoRa: Enhancing large language models fine-tuning with lora based mixture of experts. *arXiv preprint arXiv:2404.15159* (2024).

[14] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10713–10722.

[15] Chin-Tung Lin and Wei-Yun Ma. 2022. HanTrans: An Empirical Study on Cross-Era Transferability of Chinese Pre-trained Language Model. In *Proceedings of the 34th Conference on Computational Linguistics and Speech Processing (ROCLING 2022)*. 164–173.

[16] Nestor Maslej, Loredana Fattorini, Raymond Perrault, Vanessa Parli, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, and Jack Clark. 2024. *The AI Index 2024 Annual Report.* Stanford University, Stanford, CA.

[17] Arunesh Mathur, Harshal Choudhary, Priyank Vashist, William Thies, and Santhi Thilagam. 2012. An empirical study of license violations in open source projects. In *2012 35th Annual IEEE Software Engineering Workshop (SEW)*. IEEE, 168–176. https://doi.org/10.1109/SEW.2012.24

[18] Inc. Meta Platforms. 2024. Llama2 Community License. Retrieved October 1, 2024 from https://ai.meta.com/llama/license/

[19] Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. PhoBERT: Pre-trained language models for Vietnamese. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 1037–1042. https://doi.org/10.18653/v1/2020.findings-emnlp.92

[20] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. 2022. Expanding language-image pretrained models for general video recognition. In *European Conference on Computer Vision (ECCV)*. Springer, 1–18. https://doi.org/10.1007/978-3-031-19772-7_1

[21] Philippe Ombredanne. 2020. Free and open source software license compliance: tools for software composition analysis. *Computer* 53, 10 (2020), 105–109. https://doi.org/10.1109/MC.2020.3011082

[22] Bruce Perens. 1999. The open source definition. *Open sources: voices from the open source revolution* 1 (1999), 171–188.

[23] Midjourney platform. 2024. Midjourney's Terms of Service. Retrieved October 1, 2024 from https://docs.midjourney.com/docs/terms-of-service

[24] PromptHero. 2024. Openjourney v4. Retrieved October 1, 2024 from https://www.openjourney.art/

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[26] Gopi Krishnan Rajbahadur, Erika Tuck, Li Zi, Dayi Lin, Boyuan Chen, Zhen Ming, Daniel M German, et al. 2021. Can I use this publicly available dataset to build commercial AI software?–A Case Study on Publicly Available Image Datasets. *arXiv preprint arXiv:2111.02374* (2021).

[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10684–10695. https://doi.org/10.1109/CVPR52688.2022.01042

[28] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm.stanford.edu/2023/03/13/alpaca.html* 3, 6 (2023), 7.

[29] MosaicML NLP Team. 2023. *Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs.* www.mosaicml.com/blog/mpt-7b Accessed:

2025-10-01.

[30] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[31] Ruben Verborgh and Jos De Roo. 2015. Drawing conclusions from linked data on the web: The EYE reasoner. *IEEE Software* 32, 3 (2015), 23–27.

[32] Jason Wei, Najoung Kim, Yi Tay, and Quoc Le. 2023. Inverse Scaling Can Become U-Shaped. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 15580–15591. https://doi.org/10.18653/v1/2023.emnlp-main.963

[33] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* 3, 1 (2016), 1–9. https://doi.org/10.1038/sdata.2016.18

A Disclaimers

The information in this article is for general informational purposes only and does not constitute legal advice. Views, opinions, and recommendations expressed are solely those of the author(s) and do not represent any organization. Do not rely on this material as a substitute for professional legal advice tailored to your specific circumstances.

B Model Sheet

Table 5: Model Sheet of MG-BY-NC.

Use & Modify	✓	Sublicensing	✗
Create Derivatives	✓	Irrevocable	✗
Share Verbatim Copy	✓	Trademark Use	✗
Share Derivatives	✓	Commercial Use of Model	✗
Share Output	✓	Commercial Use of Output	✗
Patent Use	✓	Commercial Use of Derivatives	✗
Copyright Use	✓	Commercial Use of Code & Docs	✗
Retain Original Attribution	✓	Disclose Source	✗
Retain Original License	✓	Responsible AI Restrictions	✗
Retain All Notices	✓		
Disclaimer of Warranty	✓		
Limitation of Liability	✓		

Table 6: Model Sheet of MG-BY-OS.

Use & Modify	✓	Trademark Use	✗
Create Derivatives	✓	Responsible AI Restrictions	✗
Share Verbatim Copy	✓		
Share Derivatives	✓		
Share Output	✓		
Patent Use	✓		
Copyright Use	✓		
Commercial Use of Model	✓		
Commercial Use of Output	✓		
Commercial Use of Derivatives	✓		
Commercial Use of Code & Docs	✓		
Retain Original Attribution	✓		
Retain Original License	✓		

C Encoded Rules

Here is a code snippet:

MGLicenseRule.ttl > GPL-3.0-derivative-rule-1

```
@prefix mg: <http://~/rdf/terms#> .
mg:GPL−3.0−derivative−rule−1 a mg:Rule ;
    mg:hasOutputDef mg:derivative ;
    mg:targetActionType mg:Combine ;
    mg:targetInputWorkForm mg:code ;
    mg:targetOutputWorkForm mg:code ;
    mg:relicense mg:compatible−license ;
    mg:hasPublishRestriction mg:include_license_restriction,
        mg:include_notice_restriction, mg:disclose_self_restriction,
        mg:state_changes_restriction, mg:gnu_freedom_restriction ;
    mg:allowSharing true .
```

MGLicenseRule.ttl > Llama2-derivative-rule

```
@prefix mg: <http://~/rdf/terms#> .
mg:Llama2−derivative−rule a mg:Rule ;
    mg:hasOutputDef mg:derivative ;
    mg:targetActionType mg:Amalgamate, mg:Combine,
      mg:Modify, mg:Train, mg:Embed, mg:Distill ;
    mg:targetInputWorkForm mg:weights, mg:exe ;
    mg:targetOutputWorkForm mg:weights, mg:exe ;
    mg:relicense mg:any−license ;
    mg:hasPublishRestriction mg:include_license_restriction ;
    mg:hasUseRestriction mg:llama2_exclusive_use_restriction,
        mg:use_behavior_restriction ;
    mg:allowSharing true .
```

ruling.n3 > License Determination Logic (snippet)

```
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix list: <http://www.w3.org/2000/10/swap/list#> .
@prefix mg: <http://~/rdf/terms#> .
{ # CASE−3: There have multiple rulings require the output work's license should be compatible, but these ruling has a same license.
    ?outw a mg:Work .
    _:x log:notIncludes { ?outw mg:hasLicense ?li } . # This work does not have a license yet
    # If all relied wroks have a license, we can determind the license of this work
    ( { ?outw mg:hasReliedwork ?relw } { ?relw mg:hasLicense ?li } ) log:forAllIn _:t .
    # There is no ruling that NOT allow relicesnse (For exclude CASE−2)
    ({?outw!mg:hasRuling!mg:hasRule mg:relicense ?reli } {?reli log:notEqualTo mg:none−license}) log:forAllIn _:t .
    # Collect all compatible−relicensable license into a list
    ( ?li
        {
            ?outw mg:hasRuling ?rling .
            ?rling!mg:hasRule mg:relicense ?rule .
            ?rule log:equalTo mg:compatible−license . # Compatible
            ?rling mg:hasLicense ?li . # This ruling should bind a license which provides a compatible license list
            ?li mg:hasCompatibleLicense ?clist.
        }
    ?li_list ) log:collectAllIn _:t . # (CASE−1 will yield an empty list here) .
    ?li_list list:first ?li_1st .
    ?li_1st mg:hasCompatibleLicense ?compat_list_1st .
    # Check wether exist a compatible license from 1st that also in all other (include self) compatible lists
    _:x log:includes { ?cli list:in ?compat_list_1st .
        ({?li_list!list:member mg:hasCompatibleLicense ?compat_list_other} {?cli list:in ?compat_list_other} ) log:forAllIn _:t . } .
} => { ?outw mg:hasLicense ?li_1st . } .
```

analysis_granting.n3 > Right Reserved Error

```
{ # Error [Right Reserved]. Report a error if the license reserves you the right to do your action
    ?req a mg:Request .
    ?req!mg:grant mg:hasUsage ?req_usage . # There may be multiple mg:Usage inside a mg:Right
    ?req mg:targetAction ?a .
    ?req mg:targetWork ?outw .
    ?req <− mg:hasRequest ?inw .
    ?inw mg:hasLicense ?li . # The checking target work must have a license, otherwise this checking will be skipped
    ( ?r { ?li!mg:reserve mg:hasUsage ?r . } ?reserved_list ) log:collectAllIn _:x . # Collect all reserved rights according to the license
    ?req_usage list:in ?reserved_list . # Check if the required usage is in the reserved list
    (?a ?inw ?outw ?req_usage) log:skolem ?geniri . # Keep unique
    ("**␣Error␣**␣[Right␣Reserved]␣" ?req_usage "␣is␣reserved␣by␣" ?li "␣license␣for␣" ?a "␣action␣on␣" ?inw "␣to␣produce␣" ?outw)
        string:concatenation ?report_content .
} => {
    ?geniri a mg:Error ;
        mg:reportBy ?a ;
        mg:reportType "Right␣Reserved" ;
        mg:content ?report_content .
} .
```