



# Understanding and Remediating Open-Source License Incompatibilities in the PyPI Ecosystem

Weiwei Xu \*,



Hao He \*,



Kai Gao,



Minghui Zhou†



School of Computer Science, Peking University

\*Co-first author †Corresponding author

# Problem: license compatibility is critical in OSS

Ensuring comply with licensing requirements is a crucial aspect in the process of software reuse. Neglecting this can result in ethical, legal, and financial consequences.

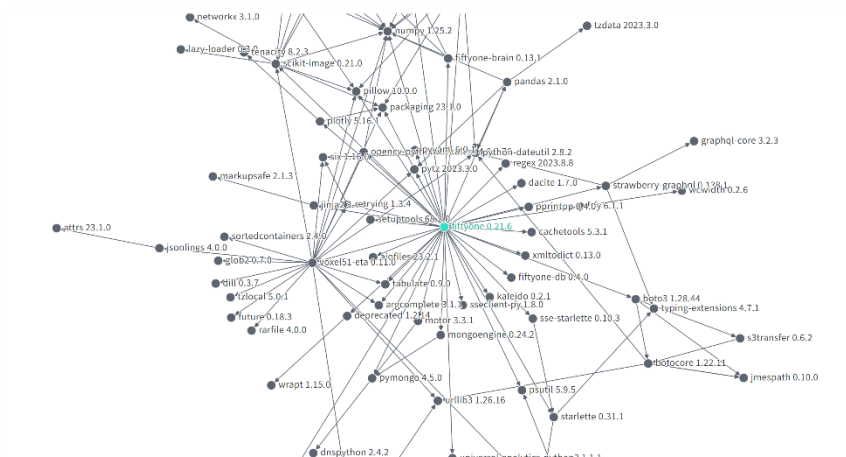


Free Software Foundation (FSF) v. Cisco Systems (2008)

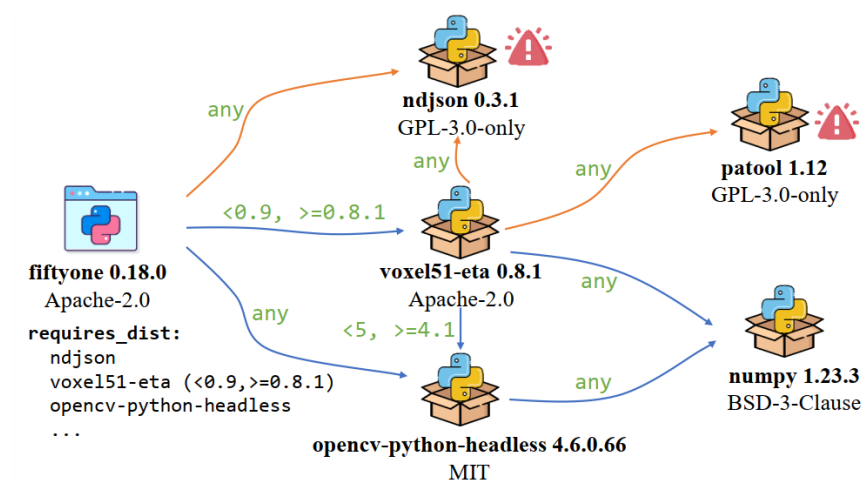
*The FSF sued Cisco in 2008 for not complying with the terms of the GPL and LGPL in their use of FSF software within numerous products, leading to Cisco agreeing to improve its license compliance processes in 2009.*

# Problem: Complex dependency relationship

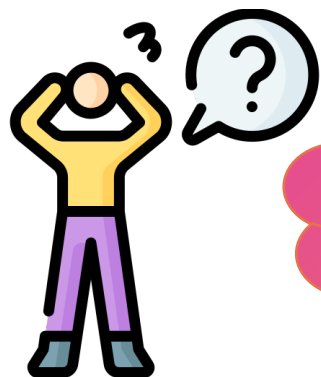
The complex dependency relationships in the package ecosystem make license incompatibility issues extremely easy to spread and very tricky to handle.



Complex dependency network in package ecosystem



License incompatibilities can be caused by both direct and transitive dependencies



developers may have insufficient knowledge about OSS licensing

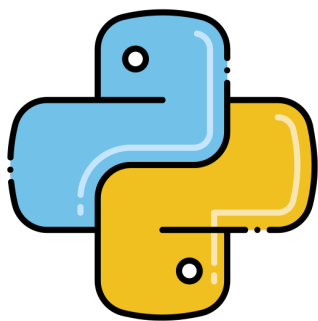
dependency graphs dynamically change over time and packages may change licenses in new releases

developers may only manage direct dependencies, overlooking or lacking enough control over transitive dependencies

# Problem defined in this study

The problem we want to address:

How to remediate the license incompatibility issues in the context of complex dependency relationships?



## Dataset:

Take the PyPI ecosystem as our research subject. Construct a dataset of dependency and licensing information of PyPI ecosystem.

## Empirical study :

Conduct an empirical study to unveil the characteristics of license incompatibilities in PyPI package ecosystem and reveal developers' remediation practices.

## Approach:

Propose an SMT-solver-based approach, **SILENCE**, to remediate license incompatibilities automatically.

## PYPI DEPENDENCY DATA

We begin with a complete PyPI distribution metadata dump obtained from the official dataset hosted on Google BigQuery in **November 2022**.



The dump contains **438,967** packages with **3,622,711** different releases.

## LICENSING DATA

The licensing information of a release can be found in three possible data sources:

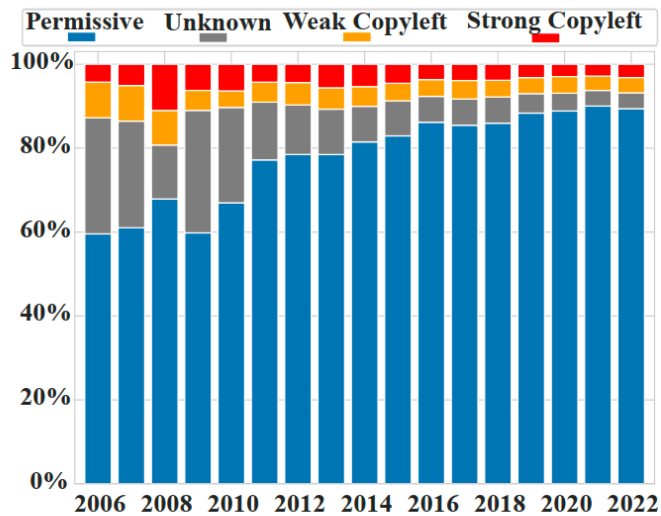


- The **license field** in its distribution metadata.
- The **classifier field** in the metadata.
- The wheel distribution files, which can include **LICENSE and README files** with licensing information.

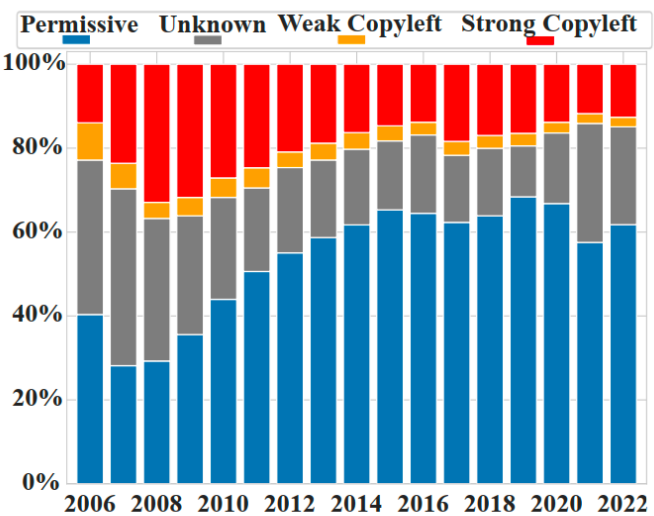
We get licensing information from classifier tags, the license field, and distribution files for **500,457 (13.8%)**, **2,465,863 (68.1%)**, **135,590 (3.7%)** releases respectively, leaving **520,801 (14.4%)** releases with Unrecognizable licensing.

# Empirical Study: Package version needs to be considered

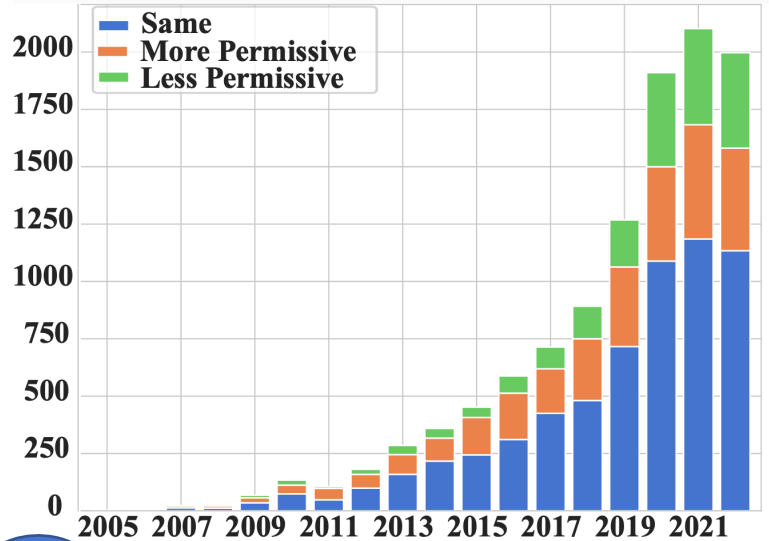
The presence of strong copyleft licenses in PyPI potentially elevates the risk of license incompatibilities. Furthermore, our findings confirm that licensing changes are a frequent occurrence in PyPI packages.



(a) TOP



(b) ALL



licensing changes in ALL

TOP: The top 5000 most downloaded PyPI packages.

The yearly distribution of licensing categories in the two groups.

ALL: All the 438,967 PyPI packages in our dataset.

To take licensing changes into consideration, a precise and versioned dependency graph is necessary for license incompatibility analysis.



# Empirical Study: global perspective solution

License incompatibilities form a significant problem in the PyPI ecosystem. Most of incompatible dependencies in dependency graphs of ALL are transitive dependencies that may reside in deep and sophisticated dependency graph positions.

TABLE I

THE LICENSE COMPATIBILITY STATUS OF PYPI RELEASES

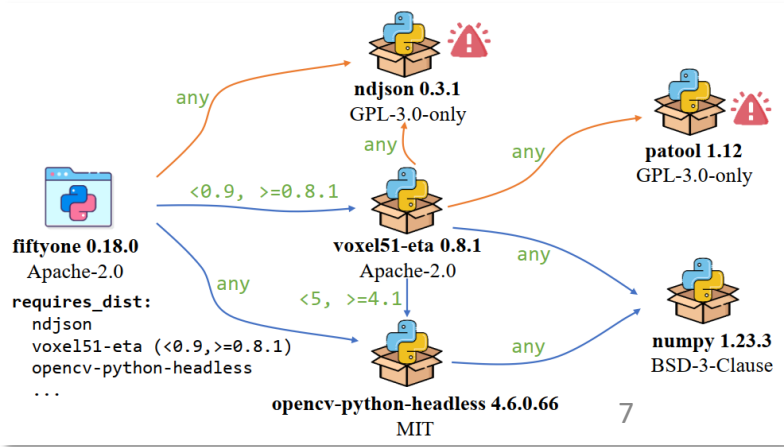
Compatibility Label	TOP (10,282 releases)		ALL (271,811 releases)	
	Count	Percentage	Count	Percentage
Compatible	5,731	55.64%	114,135	41.99%
Incompatible	202	1.96%	19,772	7.27%
Unknown	4,349	42.30%	137,904	50.74%

TABLE II

THE CUMULATIVE DISTRIBUTION OF DEPENDENCY GRAPH METRICS FOR ALL INCOMPATIBLE DEPENDENCIES

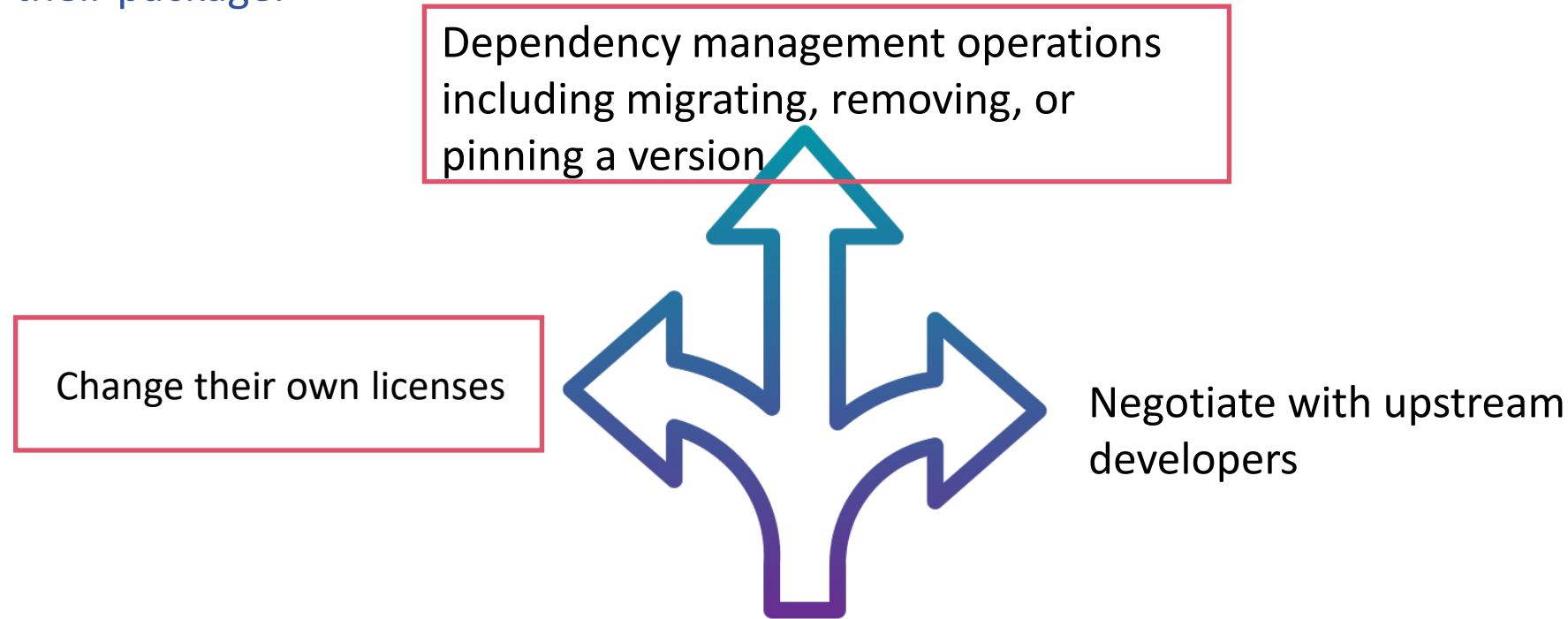
		= 0	≤ 1	≤ 2	≤ 3	≤ 4	≤ 5
Depth	TOP	-	74.0%	96.2%	100%	-	-
	ALL	-	38.7%	71.8%	89.1%	95.2%	97.6%
In-degree	TOP	-	95.9%	100%	-	-	-
	ALL	-	75.4%	87.7%	91.9%	94.2%	95.8%
Out-degree	TOP	60.8%	68.7%	81.9%	90.2%	97.4%	97.7%
	ALL	45.6%	57.9%	63.7%	68.6%	74.1%	79.0%

Remediating license incompatibilities in transitive dependencies requires searching for a feasible solution from a global perspective in the entire dependency graph.



# Empirical Study: solutions adopted by developers

PyPI package developers remediate license incompatibilities by 1) migrating, removing, or pinning a version of the incompatible dependency; 2) changing their own licenses; or 3) asking upstream developers to change the licensing of their package.



How do PyPI package developers respond to and remediate license incompatibilities in practice?

The practices taken by developers can serve as the solution space to be explored by automated approaches.



# SILENCE: SMT solver-based incompatibility mediator for licenses



## Input:

A package version  $\langle p, v \rangle$  along with its dependency graph  $G$  at a certain moment.

## Output:

- Change the root node's license strategy:  $M$  candidate licenses  $l_1, \dots, l_M$ , such that  $\langle p, v \rangle$  has no incompatible licenses in  $G$ .
- Change the root node's dependency strategy:  $N$  candidate dependency graphs  $G_1', \dots, G_N'$ , where there are no incompatible dependencies, and the changes to  $G$  are minimal.



## Algorithm 1: The SILENCE Approach

**Input:**  $\langle p, v \rangle$ ,  $\mathcal{G}(p, v)$ , and the PyPI dataset (Section V-A)  
**Output:**  $G = \{G_1', \dots, G_N'\}$ ,  $L = \{l_1, \dots, l_M\}$

```

1  $G \leftarrow \emptyset$ ,  $L \leftarrow \emptyset$ 
2 foreach  $l \in \mathcal{L}$  do # find compatible licenses
3   if  $\langle l(p', v'), l \rangle \notin \mathcal{I}$  for all  $\langle p', v' \rangle \in N(p, v) \setminus \langle p, v \rangle$  then
4      $L \leftarrow L \cup \{l\}$ 
5 Keep only top- $M$  licenses in  $L$  ordered by their popularity
6  $vars \leftarrow \{p, \text{plus all packages reachable from } \text{deps}(p, v)\}$ 
7  $clauses \leftarrow \text{build\_constraints}(p, v, vars)$ 
8 while  $G' \leftarrow \text{find\_solution}(vars, clauses, objective)$  do
9   if  $|G| \geq N$  or  $G' = \text{unsat}$  then break
10   $G \leftarrow G \cup \{G'\}$ 
11  Add new constraints to exclude solutions similar to  $G'$ 
12 return  $G, L$ 
    
```

For each package version, we add constraints on the corresponding dependency versions:

$$(p' = v') \implies \bigwedge_{\langle p_d, C \rangle \in \text{deps}(p', v')} \left( \bigvee_{v_d \in C} (p_d = v_d) \right)$$

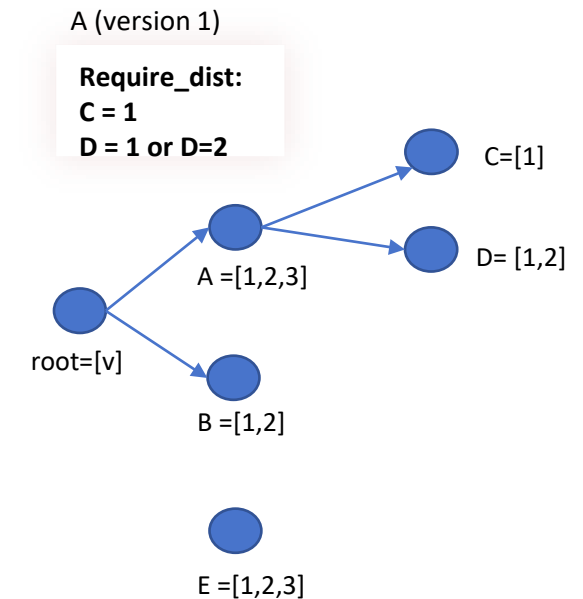
Add license constraints:

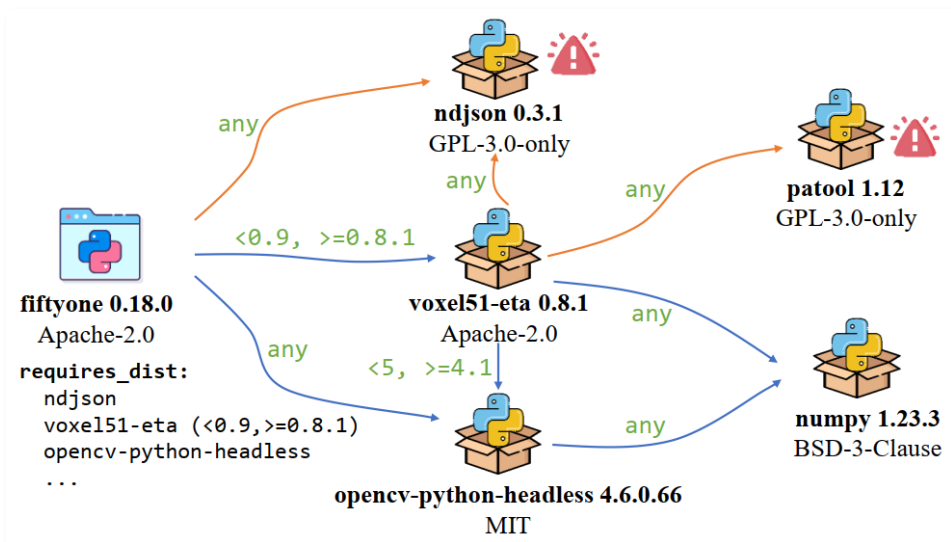
$$\bigwedge_{p' \in vars} (\langle l(p', v'), l(p, v) \rangle \in \mathcal{I} \implies p' \neq v')$$

Define the objective of minimizing configurable costs:

$$\min_{G'} \sum_{\langle p_{old}, p_{new} \rangle \in \text{diff}(G, G')} \begin{cases} c_{\text{migration}}, & \langle p_{old}, p_{new} \rangle \in \mathcal{M} \\ c_{\text{removal}}, & p_{new} = 0 \\ |p_{new} - p_{old}|, & \text{otherwise} \end{cases}$$

Using an SMT-Solver, we solve for the values of each package version that satisfy all the constraints, resulting in a compliant dependency graph.

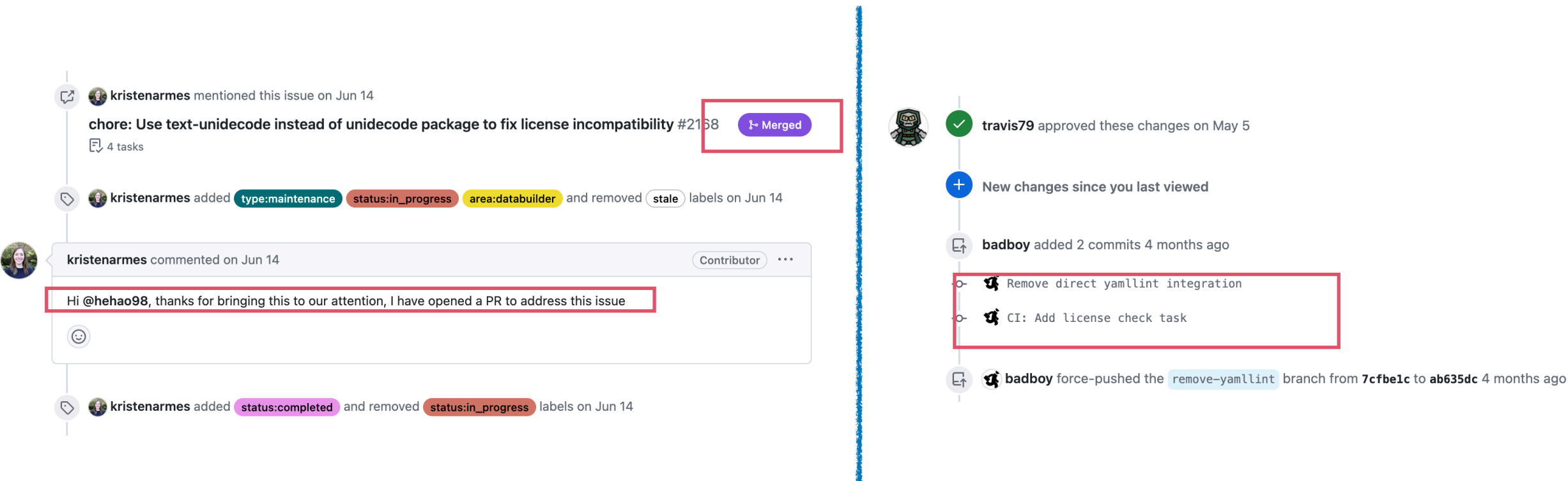




## Output:

- Possible Remediations for fiftyone 0.18.0:
1. Change project license to GPL-3.0-only, GPL-3.0-or-later, or AGPL-3.0-only;
  2. Or make the following dependency changes:
    - a) Migrate ndjson to jsonlines;
    - b) Pin voxel51-eta version to 0.1.9;
    - c) Pin pillow to 6.2.2;
    - d) Pin h11 to 0.11.0;
    - e) Pin imageio to 2.9.0.
  3. Or make the following dependency changes:
    - a) Migrate ndjson to jsonlines;
    - b) Remove voxel51-eta;
    - c) Pin h11 to 0.11.0.

Out of the 9 issues reported, we received responses in seven issues, among which five packages have completely or partially adopted one of the remediations suggested by SILENCE.



The image shows two screenshots from GitHub. The left screenshot displays issue #2168, titled "chore: Use text-unidecode instead of unidecode package to fix license incompatibility". It includes a comment from kristenarmes dated June 14, which is highlighted with a red box: "Hi @hehao98, thanks for bringing this to our attention, I have opened a PR to address this issue". The issue is marked as "Merged". The right screenshot shows pull request #2169, titled "Remove direct yamllint integration". It includes a comment from badboy dated 4 months ago, which is highlighted with a red box: "Remove direct yamllint integration". The pull request is marked as "Approved" by travis79 on May 5.

The high adoption rate signifies the relevance of license incompatibilities to PyPI developers, their positive attitude towards SILENCE, and the effectiveness of SILENCE in addressing incompatibilities.

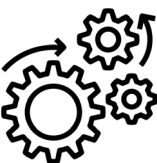
In summary, the contributions of this paper are as follows:



We build an up-to-date dependency and licensing dataset for the PyPI ecosystem, laying the foundation for license incompatibility analysis and remediation.



We conduct the first large-scale empirical study to confirm the prevalence of license incompatibilities in PyPI and reveal developers' remediation practices.



We design and evaluate a novel SMT-solver-based approach, SILENSE, for recommending actions to remediate license incompatibilities in Python dependency graphs.



**Thanks!**