The ACM International Conference on the
Foundations of Software Engineering (FSE 2024)

北京大学
PEKING UNIVERSITY

PyRadar: Towards Automatically Retrieving and Validating Source Code Repository Information for PyPI Packages

Kai Gao (高恺)
gaokai19@pku.edu.cn

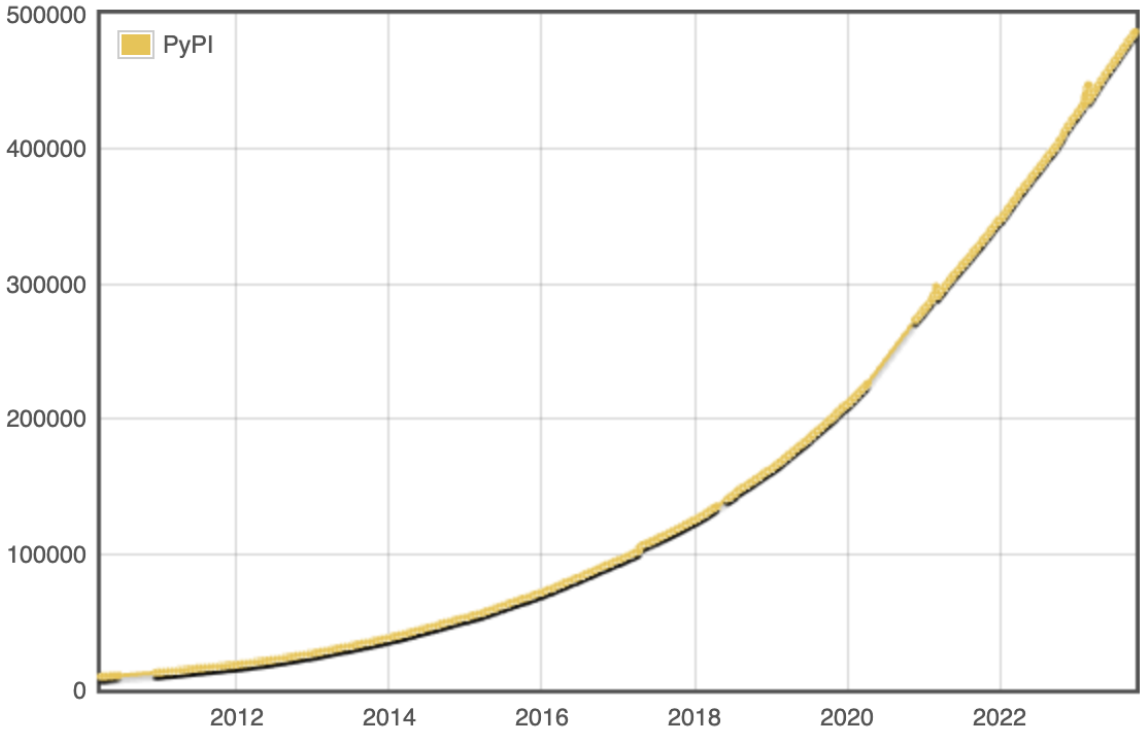
Weiwei Xu
xuww@stu.pku.edu.cn

Wenhao Yang
yangwh@stu.pku.edu.cn

Minghui Zhou
zhmh@pku.edu.cn

Porto de Galinhas, July 18, 2024

Rapid Growth and Wide Adoption of PyPI Packages



PyPI Package Count
(http://www.modulecounts.com/)

1.6 Billion

Downloads per day

9.2 Billion

Downloads per week

34.2 Billion

Downloads per month

https://pypistats.org/packages/__all__

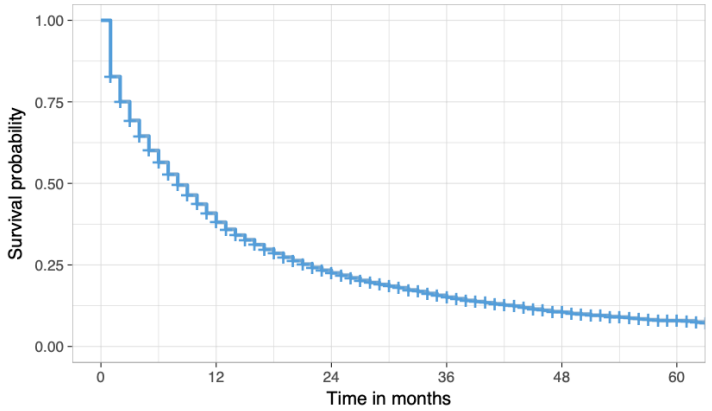
Critical Problems of Reusing Third-party Packages

Which package to use?

Heartbleed

Log4Shell

Security Vulnerabilities



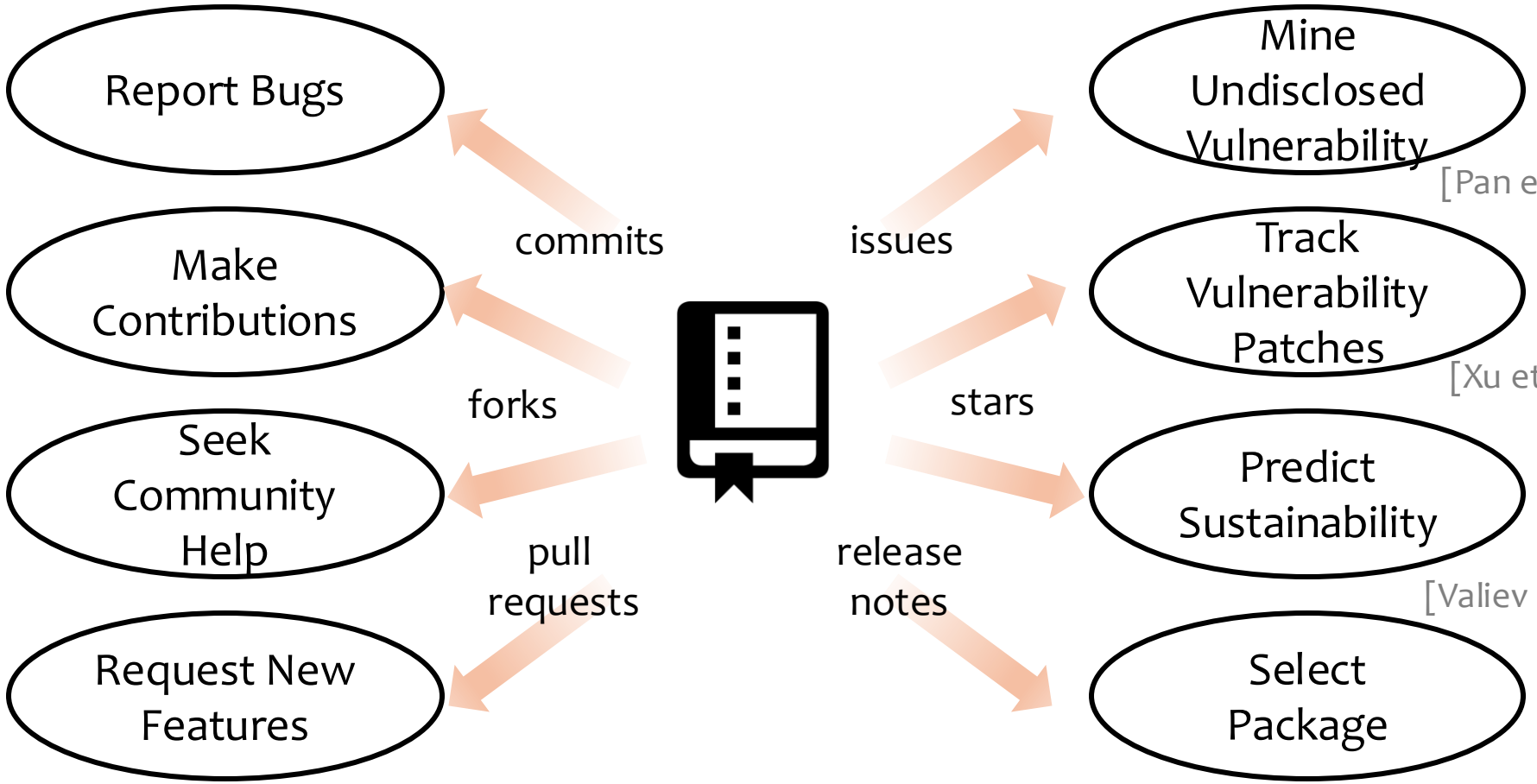
[Valiev et al., 2018]

Lack of Maintenance

Package Selection

Risk Monitoring

The Package's Source Code Repository Comes to the Rescue



[Pan et al., 2022]

[Xu et al., 2022]

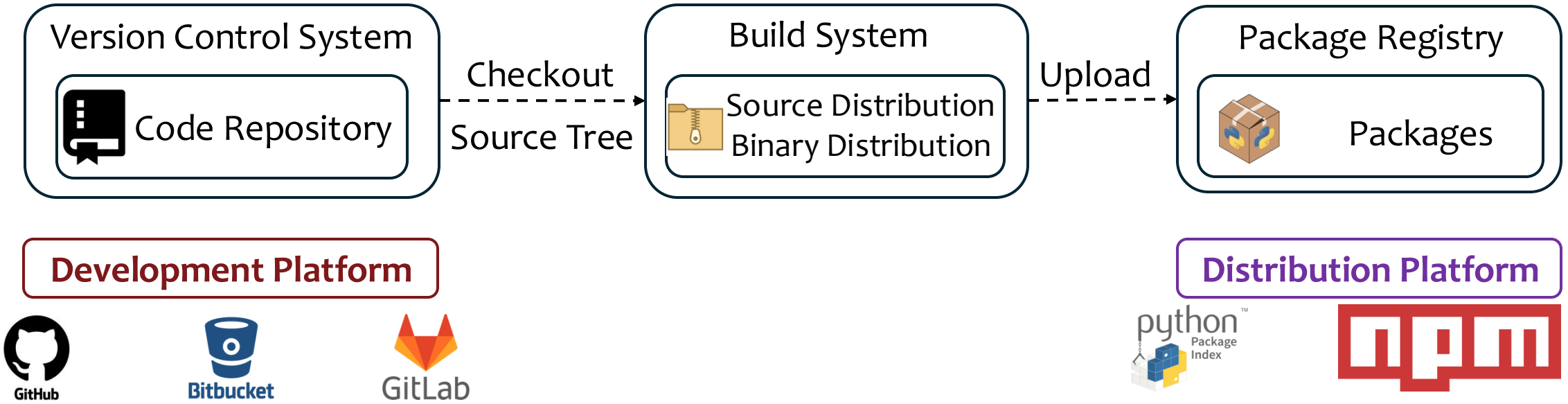
[Valiev et al., 2018]

[Larios Vargas et al. 2020]

usage, risk identification and mitigation

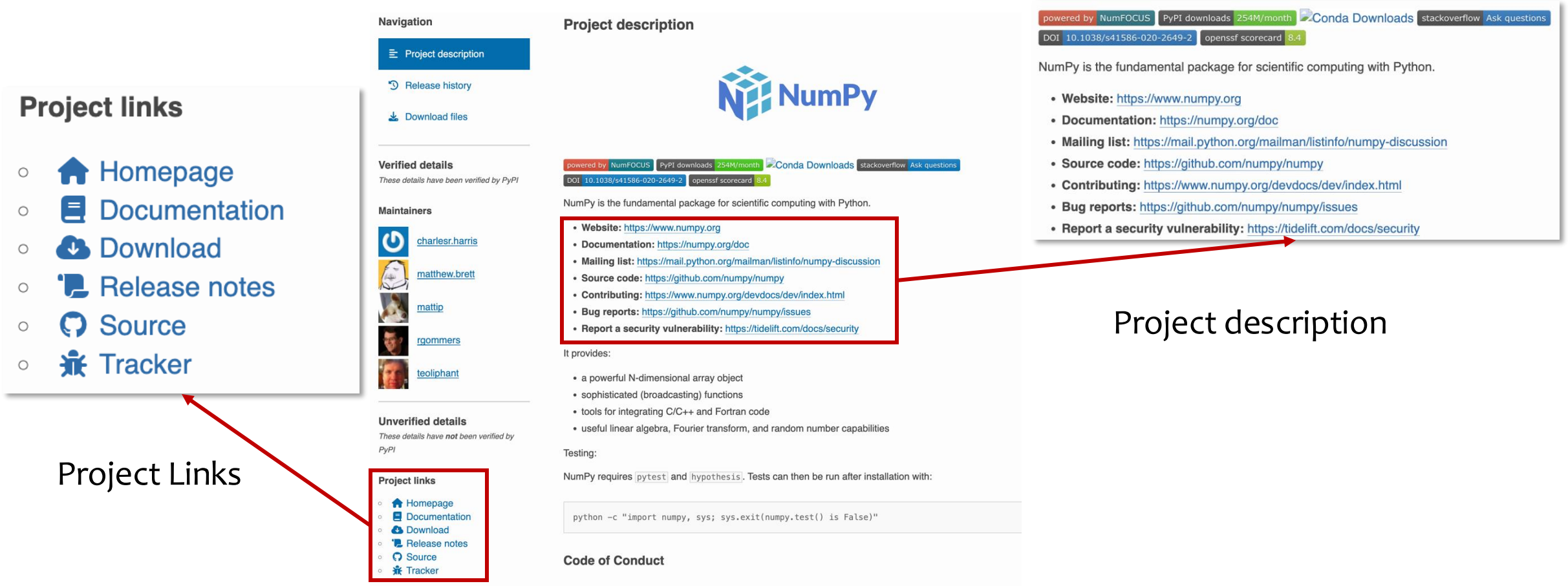
Disconnections between Packages and Their Source Code Repositories

- Most programming languages communities adopt the **development-distribution separation** strategies to manage third-party packages



The typical workflow of publishing packages

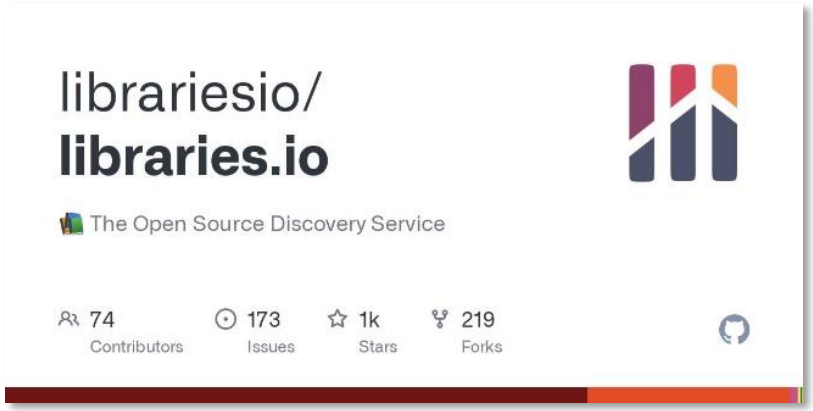
How to Manually Recover Links Between Packages and Their Source Code Repositories



The PyPI project page of the numpy package, which are generated from the package's **metadata.**

Existing Automated Tools: Metadata-based

PyPI GitHub Statistics



Libraries.io

OSS Find Source

py2src [Vu D L, 2021]

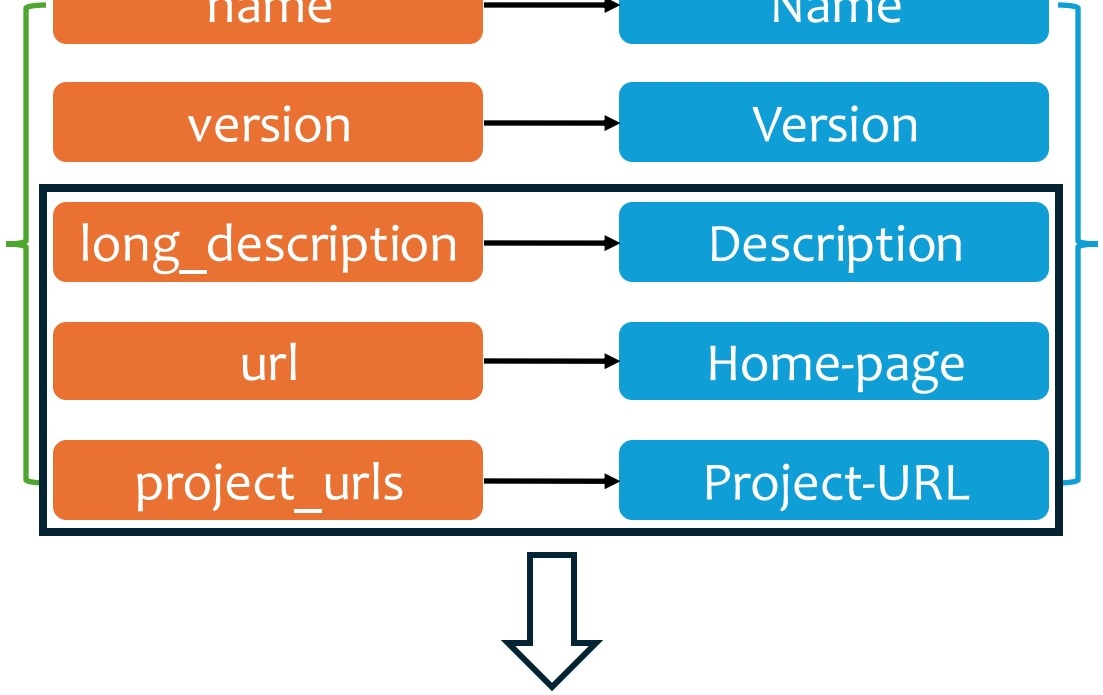
Employ different heuristics to retrieve a source code repository URL from the package's **metadata.**

How are Metadata Generated?

Package Specification Files in Code Repository

- setup.py
- pyproject.toml
- setup.cfg

Manually specified by package developers

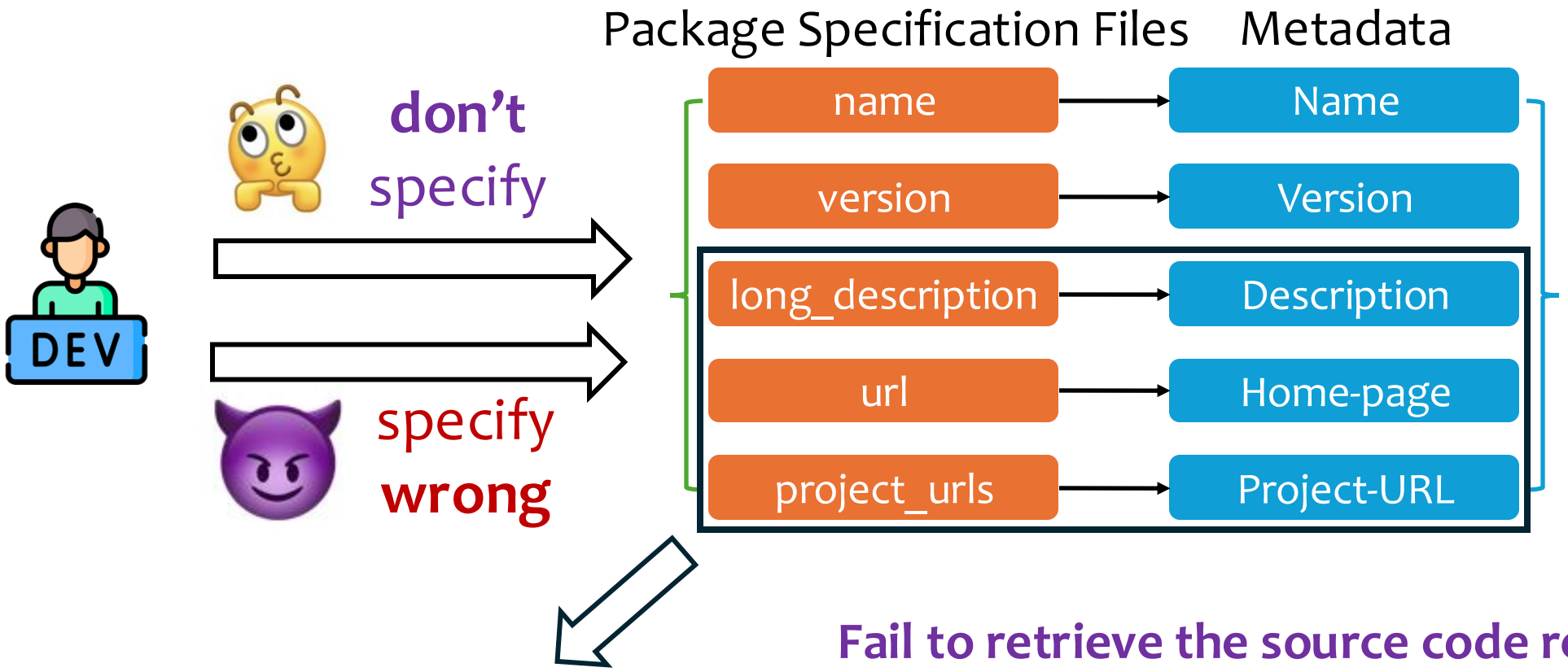


Used by metadata-based approaches

Metadata of PyPI Package

Automatically generated by build tools

Limitations of Metadata-based Approaches

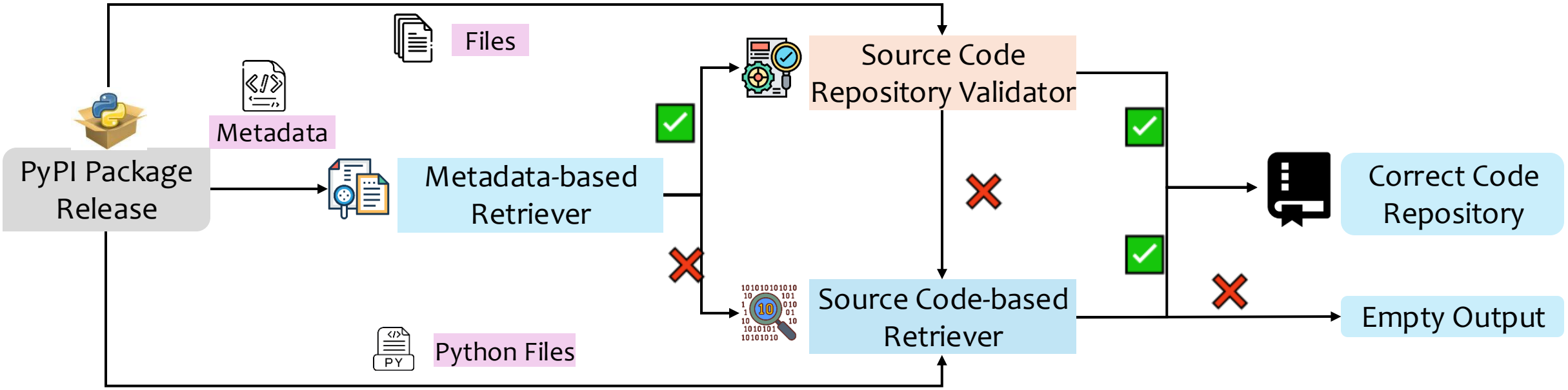


Fail to retrieve the source code repository URL

Retrieve the incorrect source code repository URL

Our Solution to Address the Two Limitations—— PyRadar

Intuitions: PyPI packages do not just have metadata, they also have **source code** in their distributions!



Overview of PyRadar

How to Collect the Correct and Incorrect Package-Repository Links?

A heuristic approach: collect correct links first, then incorrect links

Assumption 1: the linkage between **popular packages** and **popular source code repositories** should be correct.

12,463
GitHub packages

4,000
top downloaded
PyPI package

14,375
correct links

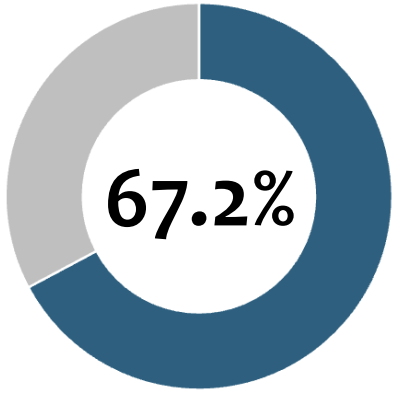
Assumption 2: Packages published by the same source code repository should have **the same PyPI maintainers.**

Package's PyPI
maintainer information

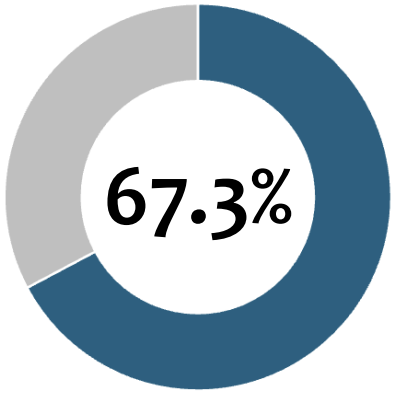
2,064
incorrect links

The Metadata-based Retriever: Evaluation of existing methods

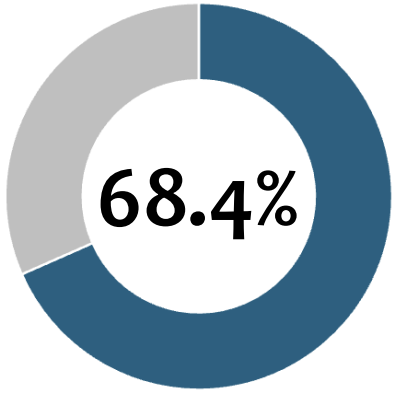
Collect metadata for **4,227,425** releases of 423,726 packages



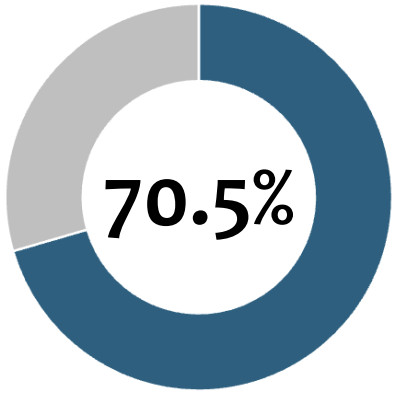
PyPI GitHub Statistics



OSS Find Source



Libraries.io



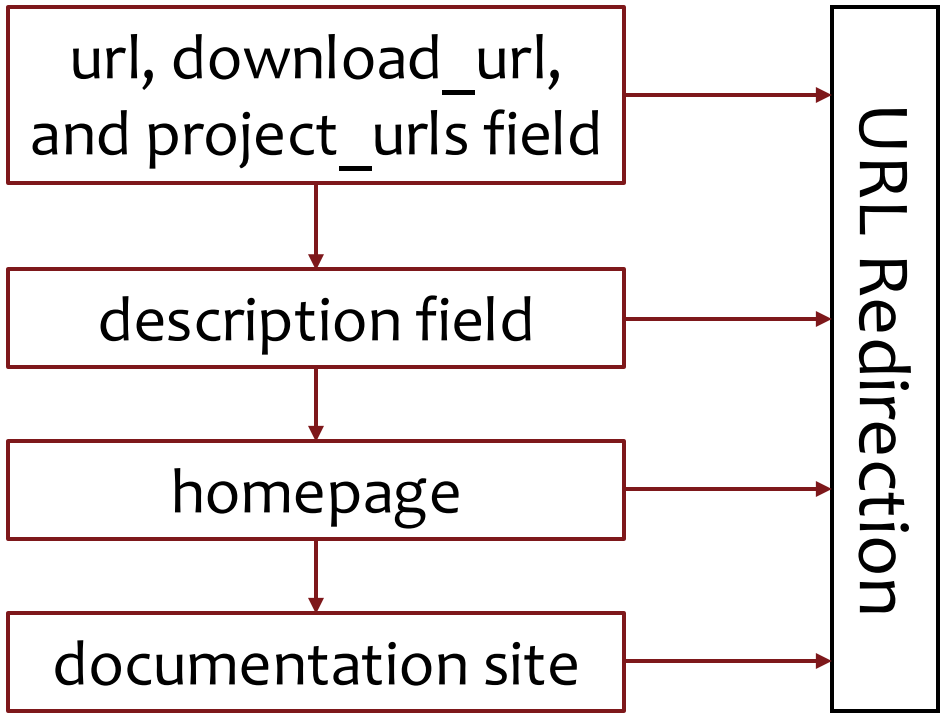
py2src

Differences in search strategies

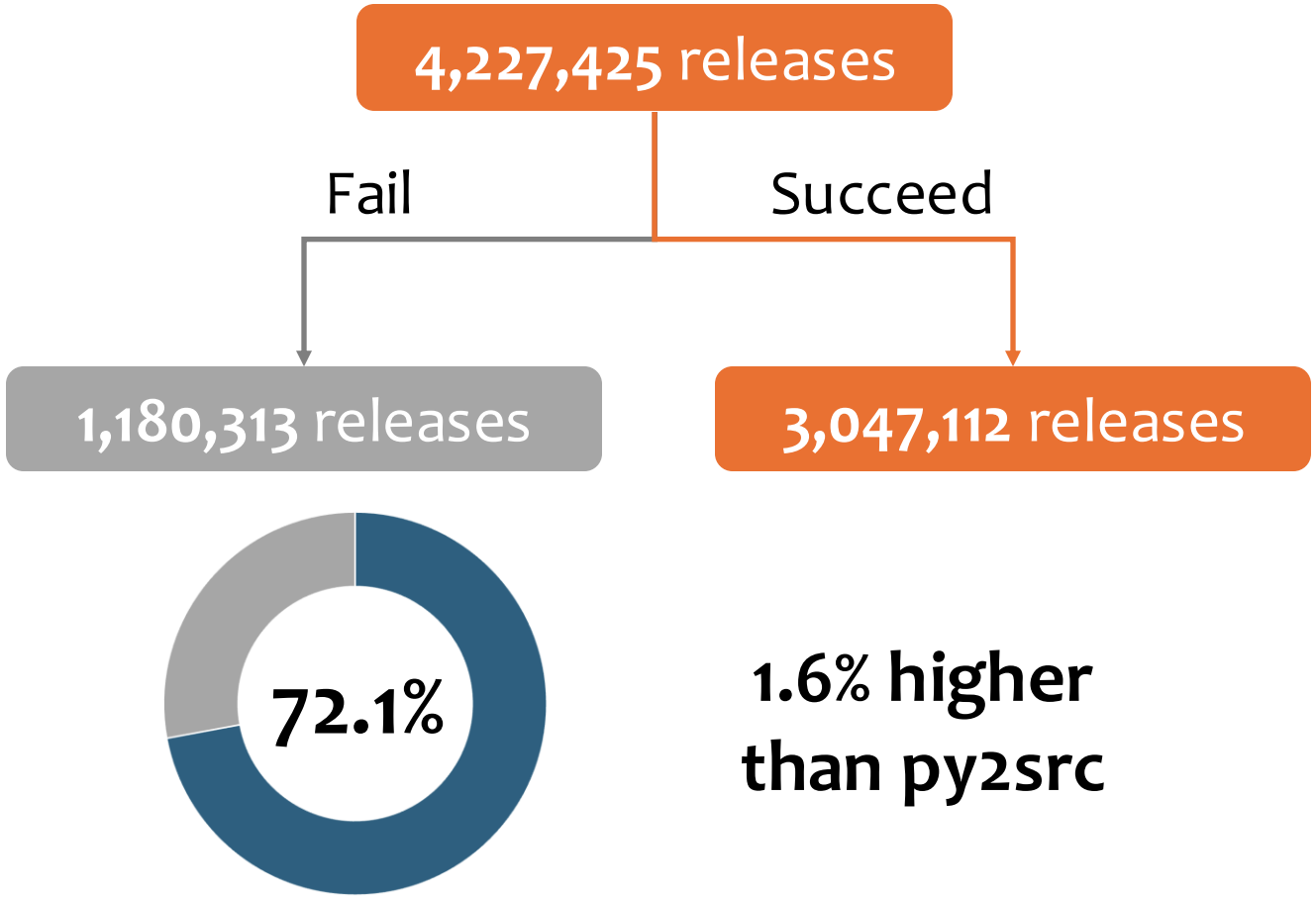
- URL Redirection
- Project-URL Field Searching
- Badge URL Searching
- Readthedocs Searching
- URL Extraction Method
- Homepage Searching
- Other

The Metadata-based Retriever: Design & Evaluation

Design



Evaluation



1.6% higher than py2src

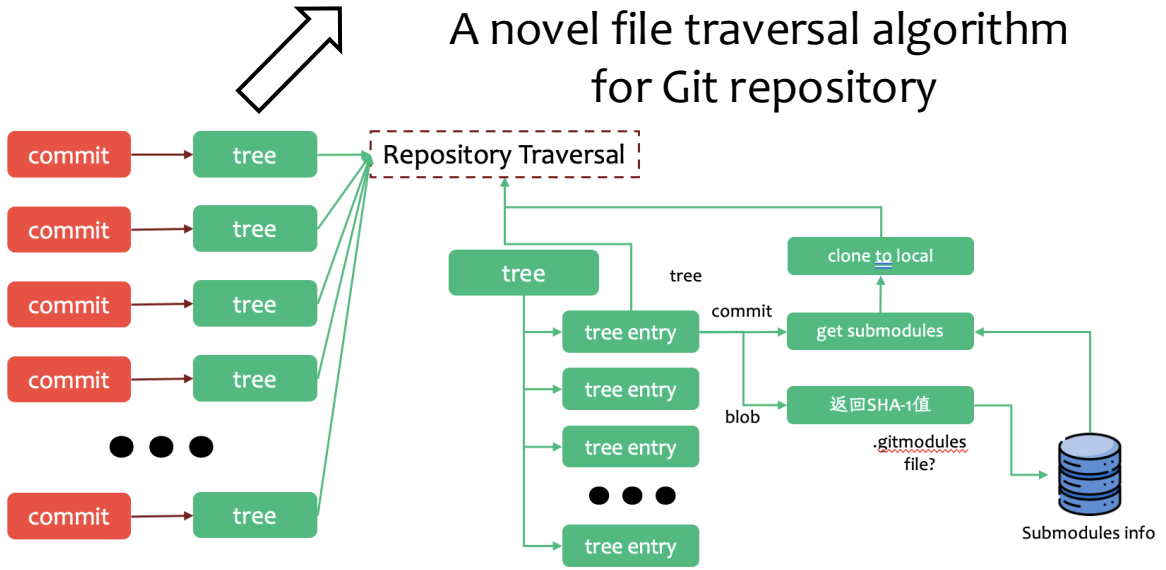
The Source Code Repository Validator: Phantom File Analysis

Phantom files[1]: files appearing in the release's distribution but not in the package's source code repository

hashes of files in the release's distribution — hashes of files in the package's source code repository

Key Findings:

1. Incorrect links have **more phantom (Python) files** than correct links.
2. The package specification file **setup.py or pyproject.toml** is more likely to be a phantom file in the incorrect links.
3. **Python files** typically remain the same in the correct links.



1. Vu D L, Massacci F, Pashchenko I, et al. Lastpymile: identifying the discrepancy between sources and packages. ESEC/FSE 2021

The Source Code Repository Validator: Design & Evaluation

Design

6 features

#phantom_pyfiles

pkg_spec_change

tag_alignment

#maintainers

name_similarity

#pkgs_by_maintainer

7 common ML algorithms

Logistic Regression

Decision Tree

Random Forest

SVM

AdaBoost

Gradient Boosted Decision Tree

XGBoost

Evaluation

14,375 correct links

2,064 incorrect links

AUC: 0.995

4,227,425 releases

Fail

Succeed

1,180,313 releases

3,047,112 releases

1 release per package

228,448 releases

accuracy: 0.85

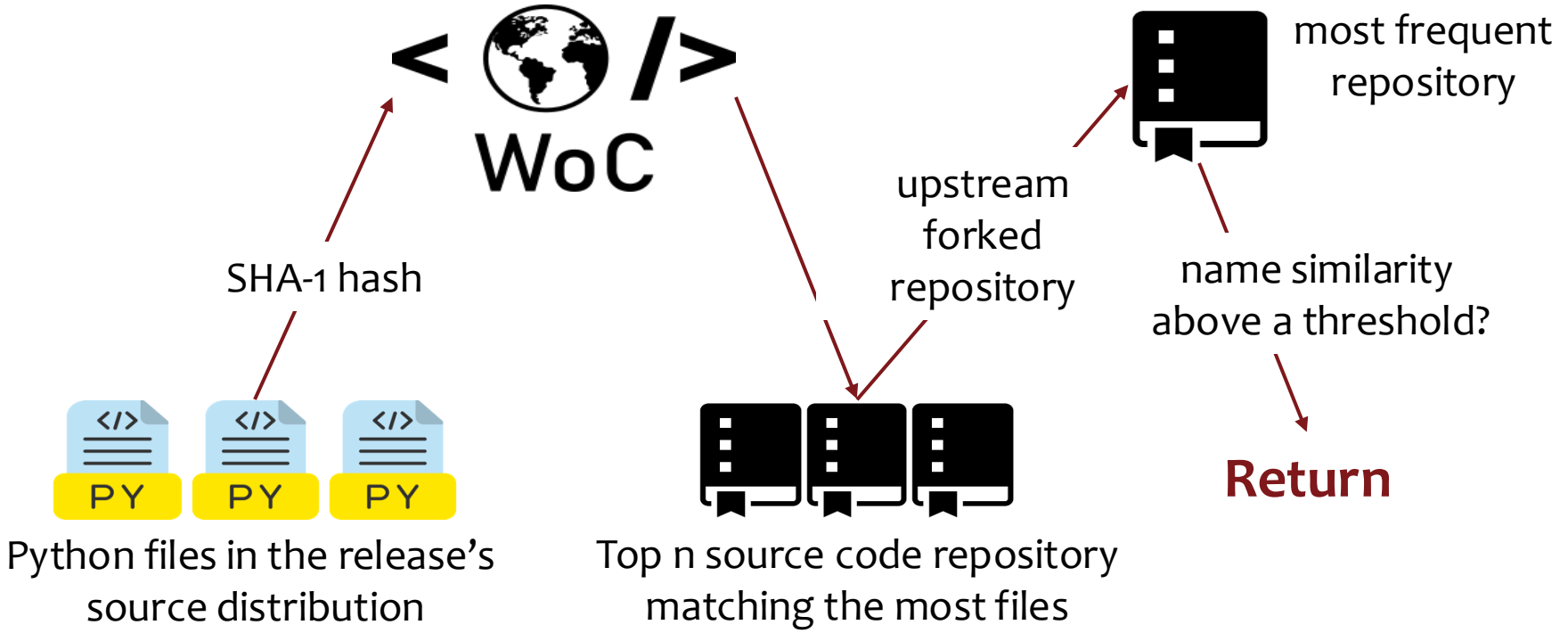
highest probability

100 releases

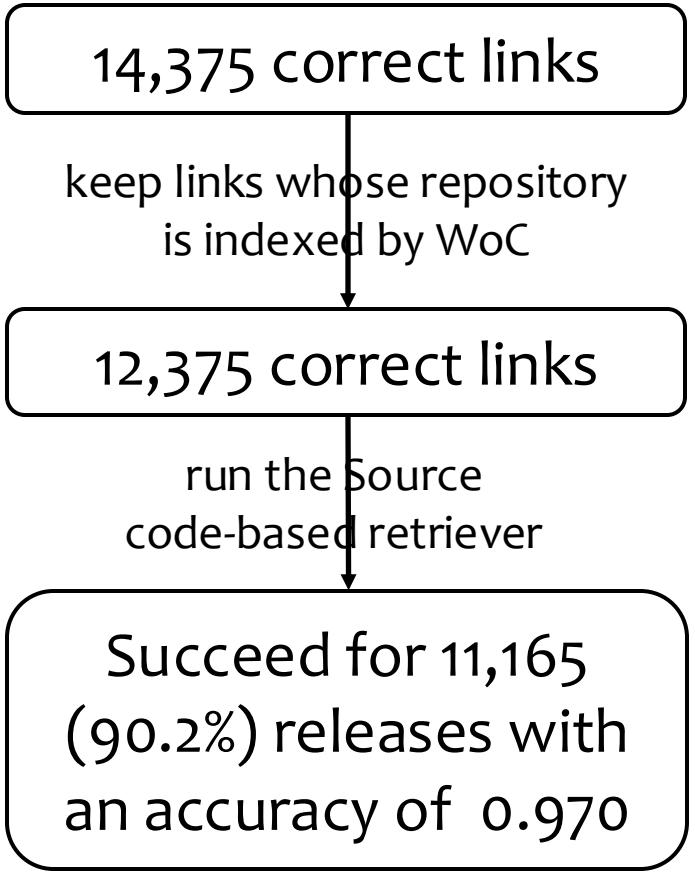
The Source Code-based Retriever: Design

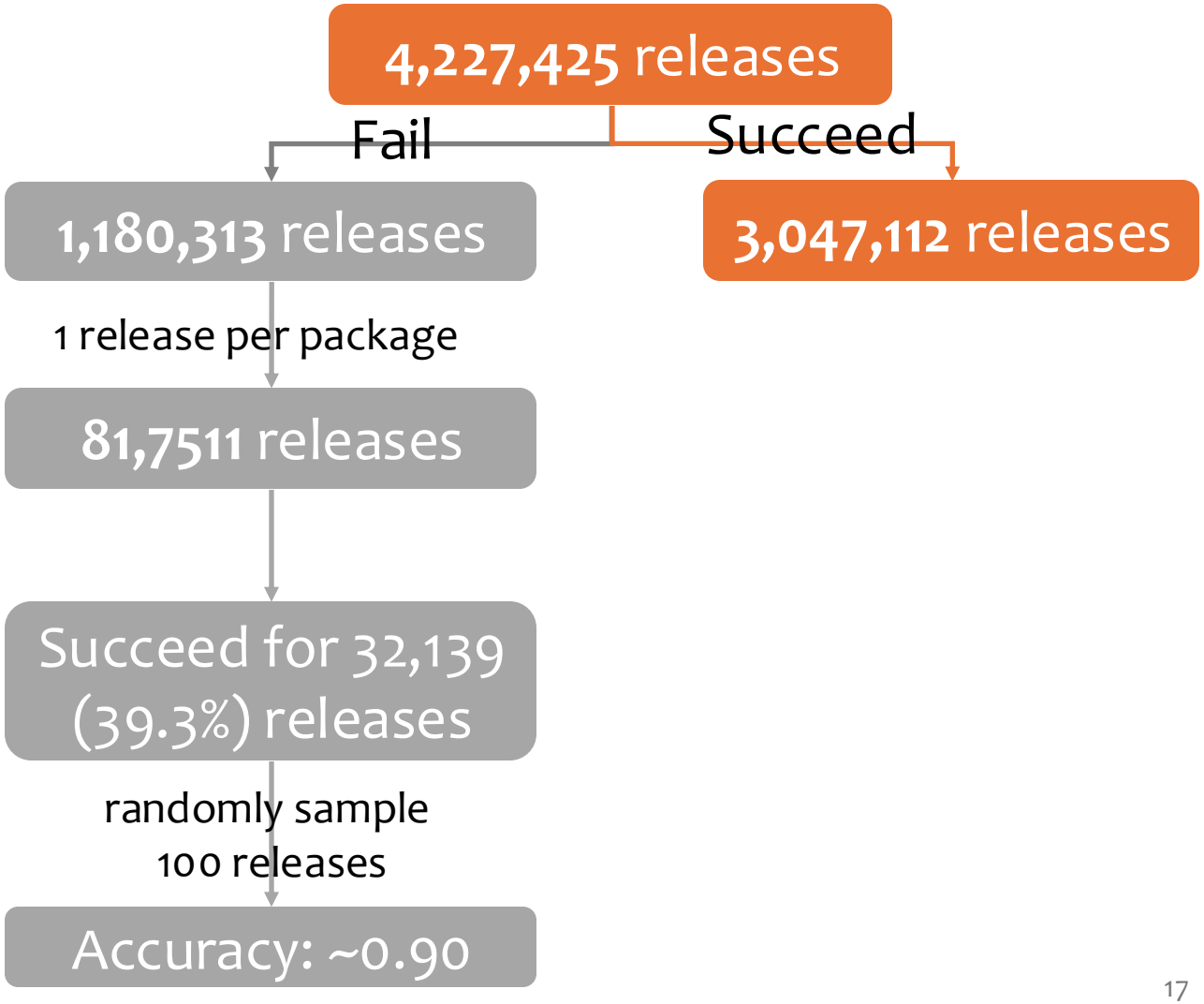
Key evidence: Python files are bridge between the package and its source code repository based on the results of phantom file analysis

Design: A **hash matching** and **name similarity**-based heuristic retrieval algorithm



The Source Code-based Retriever: Evaluation

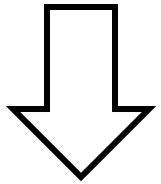




PyRadar: Overall Evaluation

14,375
correct links

2,064
incorrect links



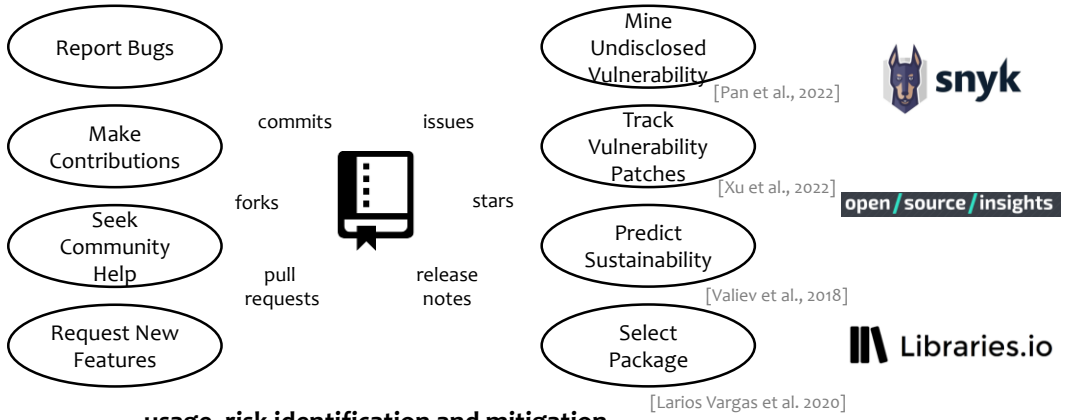
Accuracy: 0.88

Discussion

- Future Improvement
  - **Cross-link accounts** between code hosting platforms and package registries.
    - **Mechanisms:** account binding and account mutual authentication
    - **Automatic methods**.
  - Finer-grained **code analysis** to identify normal and abnormal changes in the build process.
  - Package registries (e.g., PyPI) and package management tools (e.g., pip) should **integrate validation mechanisms** to notify users if a package's repository information is problematic when searching or installing package.
- What about **other PL communities** that adopts the development-distribution separation strategies?
  - NPM, Maven, etc
  - Go

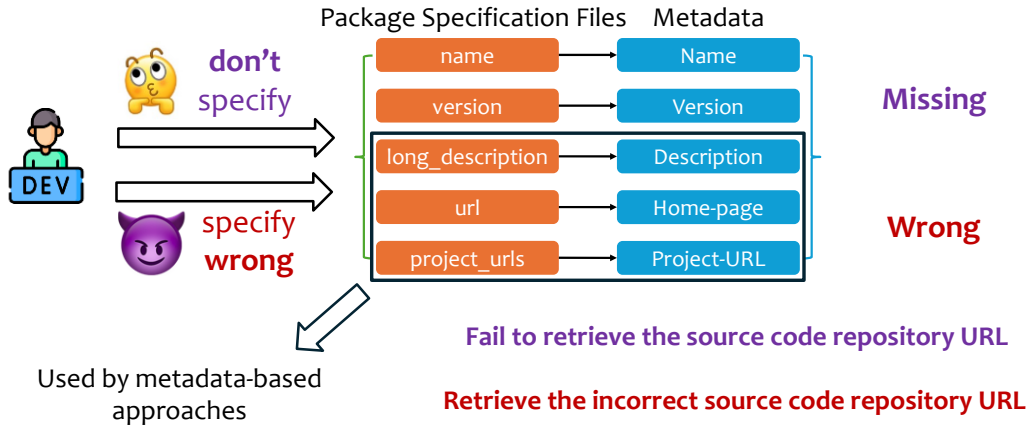
Summary

The Package's Source Code Repository Comes to the Rescue



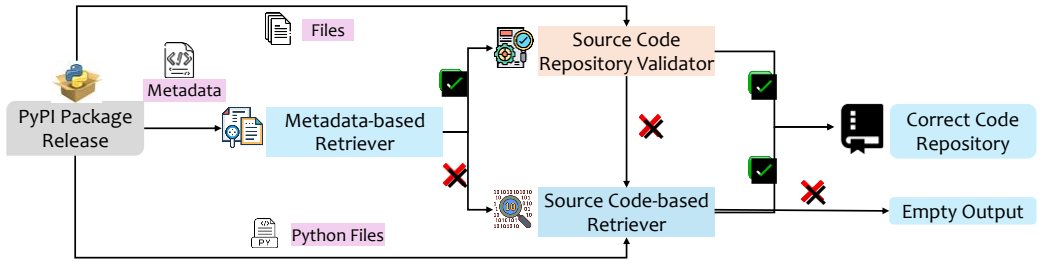
usage, risk identification and mitigation

Limitations of Metadata-based Approaches



Our Solution to Address the Two Limitations—— PyRadar

Intuitions: PyPI packages do not just have metadata, they have **source code** in their distributions!



Overview of PyRadar

Discussion

- Future Improvement
  - **Cross-link accounts** between code hosting platforms and package registries.
    - **Mechanisms:** account binding and account mutual authentication
    - **Automatic methods.**
  - Finer-grained **code analysis** to identify normal and abnormal changes in the build process.
  - Package registries (e.g., PyPI) and package management tools (e.g., pip) should **integrate validation mechanisms** to notify users if a package's repository information is problematic when searching or installing package.
- What about **other PL communities** that adopts the development-distribution separation strategies?
  - NPM, Maven, etc
  - Go

The ACM International Conference on the Foundations of Software Engineering (FSE 2024)

北京大学 PEKING UNIVERSITY

Thank You!

Paper

Code

Kai Gao (高恺)

gaokai19@pku.edu.cn

Weiwei Xu

xuww@stu.pku.edu.cn

Wenhao Yang

yangwh@stu.pku.edu.cn

Minghui Zhou

zhmh@pku.edu.cn

Acknowledgement: National Natural Science Foundation of China 61825201 and 62332001.