Learning to Find Naming Issues with Big Code and Small Supervision

Jingxuan He, Cheng-Chun Lee, Veselin Raychev, Martin Vechev

Existing Scheme of Learning a bug detector

- Non-buggy
  - Random bug injection
- Buggy
  - A dataset with synthetic bugs

Fundamental distribution mismatch

Existing deep networks (e.g., for VarMisuse)

High accuracy on synthetic bugs

~80%

~10%

Namer: combining two learning schemes

Big code

Smaller supervision

Pattern mining & matching

Run classifier on 300 violations

Pattern mining & matching

~65k patterns mined

~500k violations triggered

~90% of repos and files have violations

Evaluating Namer

GitHub

Pattern mining & matching

~33k Python repos*

~3 million source files (deduplicated)

* We also evaluated Namer on a large Java dataset. See paper.

Precision comparison

Classifier: 70%

W/o classifier: 46%

W/o analyses: 59%

W/o both: 40%

Examples

Semantic defects:

```python
self.assertEquals(3, val)
for i in xrange(10)
```

Code quality issues:

```python
num_or_process = 3
def evolve(..., **args):
```

> 86% chance accepted by professional developers at coding time in an IDE.