

# Intelligent Run-time Reliability Engineering for Python Software

Yun Peng Supervisor: Prof. Michael R. Lyu

25 June, 2024





# Outline



- Introduction & Background
- Dynamic Type System
- Dynamic Run-time Environment
- Conclusion & Future Work

#### Introduction & Background

ONE

# The Popularity of Python Language

- Python has been the 2nd most popular language at GitHub since 2019.
- Python has been used by more than 5 million developers at GitHub in 2023.



GitHub Octoverse: The state of open source and rise of AI in 2023.

Reference: GitHub Octoverse - https://github.blog/2023-11-08-the-state-of-open-source-and-ai/

#### Dynamic Features → Fast Prototyping

#### • Dynamic Type System

As a dynamic language, Python does not require type declarations in code. The types of variables are determined at run time. This makes it easier to write generic functions.

#### • Dynamic Run-time Environment

Python does not require compilation before execution, making it portable on different platforms and systems.

# **Dynamic Type System**

• Dynamic type system makes it easier for Python developers to write generic functions.



# **Reliability Issue #1 – Type Error**

Python allows flexible operations between
Python allows heterogeneous data types.

>>> "100" \* 2 >>> "100100"

>>> "100" / 2 >>> TypeError def divide(input\_list):
result = input\_list[0]
for item in input\_list[1:]:
 result = result / item
 return result

>>> divide([100, 100]) >>> 1

>>> divide([100, "100"]) >>> TypeError

#### **Reliability Issue #1 – Type Error**

	Туре	Attribute	Value	Key	Import
StackOverflow	31.5%	19.4%	27.8%	8.3%	13.0%
GitHub	29.2%	19.4%	28.2%	12.9%	10.3%

**Type errors** are the most mentioned Python program errors.



About 50% of type errors cost **more than one week** to be fixed.

Reference: PyTER: effective program repair for Python type errors

### **Dynamic Run-time Environment**



Developer

User

### **Reliability Issue #2 – Run-time Environment Conflicts**

 Run-time environment conflicts occur when the implementations of external APIs used in the Python software cannot be found or are not the required ones in the run-time environment.



• Configuration files written by developers are not validated before distribution.

#### **Outline of Thesis**



#### Dynamic Type System

# Improve the Reliability of Dynamic Type System

• Pathway #1 (Prevention): Use type inference to statically get the types of variables so that common static checking techniques can be used to detect potential issues.



Type Checking

Blynduin

**Test Case Generation** 

### **Type Inference**

• Type Inference aims to determine the types of each variable in code.

- 1 def add(num1, num2):
- 2 a = num1 + num2
- 3 b = 1 + 2
- 4 return a + b

**Parameters**: num1 : ? num2 : ? Local Variables: a : ? b:? **Return Value:** add : ?

# How to Do Type Inference?

• Static type inference, which is frequently used in compilers.

```
1 def add(num1, num2):
```

- 2 a = num1 + num2
- 3 b = 1 + 2
- 4 return a + b

Premise 1, ..., Premise N conclusion

**Typing Rule Format** 

$\pi \vdash 1: int$	$\pi \vdash 2: in$	t	(Constant)
$\pi \vdash 1: int \pi \vdash 2$	2 : int		
$\pi \vdash 1 + 2$ : int		(Add)	
$\pi \vdash 1 + 2 : int$			
$\pi \vdash b: int$ (	Assign)		

- Very accurate (sound).
- Suffer from the low coverage problem.

### **How to Do Type Inference?**

• Supervised Type Inference.



- Address the low coverage problem.
- Require high-quality type annotations to train, may not be accurate.

### How to Do Type Inference?

• Static type inference vs. Supervised type inference.



A performant type-checker for Python 3

- Very accurate (sound).
- Suffer from the low coverage problem.



- Address the low coverage problem.
- Require high-quality type annotations to train, may not be accurate.

#### **Outline of Thesis**



# Hybrid Type Inference - HiTyper



# **Type Dependency Graph (TDG)**

• To bridge static type inference and supervised type inference.

Graph G = (N, E) N: nodes E: edges

Four kinds of nodes:

- symbol node / type slot
- expression node
- branch node
- merge node



# Hybrid Type Inference - HiTyper



21

#### **Static Inference and Rejection**

• To Infer correct types as many as possible and reject incorrect type predictions from deep learning models.



#### **Expressions**



### **Static Inference and Rejection**

• To Infer correct types as many as possible and reject incorrect type predictions from deep learning models.

#### **Forward Type Inference:**

- Start from nodes with no *input* nodes.
- Forward traverse the whole TDG.
- Activate typing rules in expression nodes.

#### **Backward Type Rejection:**

- Start from nodes with no *output* nodes.
- Backward traverse the whole TDG.
- Activate type rejection rules in expression nodes.



# Hybrid Type Inference - HiTyper



### **Neural Type Prediction**

#### **Hot Type Slot Finder:**

Identify the key variables that hinders the static inference part from inferring other variables.

 $\rightarrow$  reduce the variables that predicted by DL models.

#### **Similarity-based Type Correction:**

Map the never imported type predictions from DL models into valid types.

 $\rightarrow$  enhance the ability of predicting user-defined/third-party types.

import torch

prediction: tf.Tensor, mapping to: torch.Tensor

# Hybrid Type Inference - HiTyper



#### **Evaluation**

	Type Category		Top-1		Top-3		Top-5	
Dataset		Approach	Exact	Match to	Exact	Match to	Exact	Match to
			Match	Parametric	Match	Parametric	Match	Parametric
		Naive Baseline	0.14	0.16	0.33	0.38	0.43	0.51
	Argument	Type4Py	0.61	0.62	0.64	0.66	0.65	0.68
	-	HITYPER	0.65	0.67	0.70	0.74	0.72	0.76
	Return Value	Naive Baseline	0.07	0.10	0.19	0.28	0.28	0.42
		Type4Py	0.49	0.52	0.53	0.59	0.54	0.63
ManyTypes4Py		HiTyper	0.60	0.72	0.63	0.76	0.65	0.77
	Local Variable	Naive Baseline	0.13	0.17	0.33	0.45	0.47	0.65
		Type4Py	0.67	0.73	0.71	0.78	0.72	0.79
		HITYPER	0.73	0.85	0.74	0.86	0.75	0.86
	All	Naive Baseline	0.13	0.16	0.31	0.40	0.43	0.57
		Type4Py	0.62	0.66	0.66	0.72	0.67	0.73
		HiTyper	0.69	0.77	0.72	0.81	0.72	0.82
Typilus's Dataset		Naive Baseline	0.19	0.20	0.38	0.42	0.46	0.50
	Argument	Typilus	0.60	0.65	0.69	0.74	0.71	0.76
		HITYPER	0.63	0.68	0.72	0.76	0.76	0.79
		Naive Baseline	0.11	0.11	0.28	0.31	0.36	0.43
	<b>Return Value</b>	Typilus	0.41	0.57	0.48	0.62	0.50	0.64
		HiTyper	0.57	0.70	0.63	0.75	0.64	0.77
		Naive Baseline	0.17	0.18	0.35	0.39	0.44	0.48
	All	Typilus	0.54	0.62	0.63	0.70	0.65	0.72
		HITYPER	0.61	0.69	0.69	0.76	0.72	0.78

HiTyper shows great improvement  $(11\% \sim 15\%)$ on **overall type inference** performance, and the most significant improvement is on **return value inference** (22% ~ 39%).

# **Limitations of HiTyper**



- The performance upper bound of HiTyper depends on the performance of deep learning models used in the framework.
- If deep learning model cannot give correct type predictions, static inference cannot validate and give the final predictions.

#### A Recent New Approach – Cloze-Style Type Inference

- 1 def add(num1:<mask0>, num2:<mask1>) -> <mask2>:
- 2 c:<mask3> = num1 + num2
- 3 d:<mask4> = 1 + 2
- 4 return c + d



Parameters: <mask0> (num1) : int <mask1> (num2) : int Local Variables: <mask3> (c) : int <mask4> (d) : int

Return Value: <mask2> (add) : int

#### A Recent New Approach – Cloze-Style Type Inference

1 def add(num1:<mask0>, num2:<mask1>) -> <mask2>:

- 2 c:<mask3> = num1 + num2
- 3 d:<mask4> = 1 + 2
- 4 return c + d
  - Do not require a high quality training set.
  - Lack of static domain knowledge: With internal knowledge only in the pre-trained code models.
  - Lack of interpretability:

No idea about how the model reaches the prediction.

#### **Outline of Thesis**



#### **Generative Type Inference - TypeGen**

Input prompt with static domain knowledge

Python Code: DATABASES = { 'default': { 'ENGINE': 'django.db.backends.sqlite3', 'NAME': os.path.join(BASE_DIR, 'db.sqlite3'), } DATABASES['default'].update(db_from_env) Available User-defined Types:	
os.Mapping, os.MutableMapping, os.PathLike, osAddedDllDirectory, osEnviron, oswrap_close	
Q: What"s the type of the variable DATABASES?	LLIVI

**Output** chain-ofthought prompt making predictions

**A: First**, the variable DATABASES is assigned from a dict. **Second**, the key of the dict is a str. The value of the dict is a dict. **Third**, the keys of the dict are a str and a str. The values of the dict are a str and a function call os.path.join. **Therefore**, the type of the variable DATABASES is `dict[str, dict[str, str]]`.

Let LLMs act like a static type inference tool! See what static inference sees, think how static inference thinks.

#### **Generative Type Inference - TypeGen**

Challenge 1: Lack of static domain knowledge

What knowledge should a model have to infer a type for a variable? (See what static inference sees)

Knowledge 1: The context of the target variable – Code slicing.

Knowledge 2: The valid type set of the variable – Type hint collection.

#### **Code Slicing**

- Remove all statements without data dependencies with the target variable.
- Remove statements with very far data dependencies with the target variable.



Type Dependency Graph

Source Code

## **Type Hint Collection**

#### Imported types = third-party types + user-defined types

#### **User-defined types:**

• Collect all classes in the current source file.

#### Third-party types:

- Download top 10,000 popular Python packages in Libraries.io.
- Collect all classes and their paths as a third-party type database.
- Query the database based on the import statements in current source file.

#### **Generative Type Inference - TypeGen**

Challenge 2: Lack of Interpretability

How to know/guide the model to reach a type prediction like static inference? (Think how static inference thinks)

Simulate the inference steps of static inference!
# **Chain-of-Thought Prompt Generation**



Translate the Type Dependency Graph into a Chain-of-Thought prompt.

value of/argument [NAME] is [GTTYPE].

# **Chain-of-Thought Prompt Generation**



- First, the variable DATABASES is assigned from a dict.
- Second, the key of the dict is a str.
   The value of the dict is a dict.
- Third, the keys of the dict are a str and a str. The values of the dict are a str and a function call os.path.join.
- Therefore, the type of the variable DATABASES is `dict[str, dict[str, str]]`.

Translate the Type Dependency Graph into a Chain-of-Thought prompt.

# **In-Context Learning**

		Example Prompt:
	1. Code Slice	Python Code:
Static Analysis	:	DATABASES = {     'default': {         'ENGINE': 'django.db.backends.sqlite3',         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),     } } DATABASES['default'].update(db_from_env)
Generated	<ol> <li>Type Hint</li> </ol>	Available User-defined Types:
Generateu	i	os.Mapping, os.MutableMapping, os.PathLike, osAddedDllDirectory, osEnviron, oswrap_close
		Q: What's the type of the variable DATABASES?
	3. COT Prompt	<b>A: First</b> , the variable DATABASES is assigned from a dict. <b>Second</b> , the key of the dict is a str. The value of the dict is a dict. <b>Third</b> , the keys of the dict are a str and a str. The values of the dict are a str and a function call os.path.join. <b>Therefore</b> , the type of the variable DATABASES is `dict[str, dict[str, str]]`.
		Target Variable Prompt: +
	①. Code Slice	Python Code: [Code]
LLM Predicted	<ol> <li>Type Hint</li> </ol>	Available User-defined Types: [User-defined types from static analysis]
		Q: What's the type of the variable [name]?
		A: [To be generated]

# **Performance of TypeGen**

Metric	Category	Approach	Top-1			Top-3					Top-5				
	gy		Arg	Ret	Var	All	Arg	Ret	Var	All	-	Arg	Ret	Var	All
		TypeBERT	28.0	38.5	51.1	45.4	34.8	52.6	55.8	51.4		36.5	57.1	58.6	54.1
	Supervised	TypeWriter	53.3	52.8	-	-	61.1	60.7	-	-		65.8	65.3	-	-
		Type4Py	66.5	56.1	82.0	76.6	72.0	59.2	83.8	79.3		73.8	60.7	84.3	80.1
Exact		InCoder-1.3B	20.9	20.5	15.1	16.7	21.3	20.8	15.5	17.1		21.3	21.0	15.6	17.2
Match (%)	Cloze	InCoder-6.7B	24.1	42.0	18.7	21.9	24.6	42.7	19.1	22.3		24.7	43.1	19.2	22.4
(70)	Style	UniXcoder	55.0	49.2	35.9	40.9	66.9	64.6	42.1	49.0		70.6	69.8	45.2	52.4
	-	CodeT5-base	51.1	57.6	21.7	30.7	59.3	64.4	28.0	37.4		62.0	66.9	30.7	40.1
		CodeT5-large	56.2	60.2	44.7	48.4	61.6	64.5	50.4	53.9		63.9	66.3	53.4	56.6
	Generative	TypeGen	73.1	68.7	82.2	79.2	81.0	77.1	87.9	85.6		82.7	79.1	89.1	87.0
		TypeBERT	29.8	41.4	54.0	48.1	36.0	55.9	58.0	53.5		37.7	60.8	61.2	56.5
	Supervised	TypeWriter	54.4	54.1	-	-	63.4	63.5	-	-		68.8	69.3	-	-
		Type4Py	68.0	59.0	86.2	80.2	74.1	64.1	88.3	83.3		75.9	66.3	88.8	84.3
Match to		InCoder-1.3B	22.9	22.8	18.7	19.9	23.3	23.1	19.1	20.3		23.4	23.3	19.2	20.4
Parametric	Cloze	InCoder-6.7B	28.8	51.6	25.0	28.1	29.3	52.1	25.3	28.5		29.4	52.5	25.3	28.6
(70)	Style	UniXcoder	61.9	61.8	44.3	49.3	72.3	76.0	51.2	57.6		75.0	80.1	53.8	60.4
	-	CodeT5-base	54.8	66.7	27.7	36.6	62.9	74.2	34.4	43.6		65.6	76.4	37.1	46.3
		CodeT5-large	61.4	69.4	55.7	58.0	66.8	74.3	61.2	63.5		68.9	76.2	63.7	65.9
	Generative	TypeGen	78.7	75.6	91.2	87.3	84.9	83.0	93.7	91.0		86.1	84.5	94.1	91.7

# **Performance of TypeGen**

Base Model	Approach	Top-1 ( $\triangle$ )	<b>Top-3</b> (△)	<b>Top-5</b> (△)
CPT Neo	Zero-Shot	31.5	40.6	42.8
(1.3B)	Standard ICL	44.0 (40%)	50.0 (23%)	50.8 (19%)
	TypeGen	57.0 (81%)	61.5 (51%)	62.8 (47%)
CPT Neo	Zero-Shot	43.2	50.0	51.9
(2.7B)	Standard ICL	46.6 (8%)	52.3 (5%)	52.8 (2%)
(2.1.2)	TypeGen	55.5 (28%)	61.9 (24%)	63.0 (21%)
GPT-I	Zero-Shot	42.4	43.7	43.9
(6.7B)	Standard ICL	50.8 (20%)	54.9 (26%)	55.3 (26%)
(0.0-)	TypeGen	62.7 (48%)	67.3 (54%)	68.4 (56%)
CodeGen	Zero-Shot	34.7	44.0	45.5
(6B)	Standard ICL	54.1 (56%)	60.5 (38%)	61.9 (36%)
(02)	TypeGen	63.7 (84%)	69.1 (57%)	70.8 (56%)
CPT 3 5	Zero-Shot	62.0	65.4	66.3
(175B)	Standard ICL	69.7 (12%)	74.2 (13%)	75.8 (14%)
(1102)	TypeGen	78.9 (27%)	85.0 (30%)	86.2 (30%)
ChatGPT	Zero-Shot	61.3	66.1	67.5
(175B)	Standard ICL	68.0 (11%)	71.8 (9%)	73.1 (8%)
()	TypeGen	78.8 (29%)	85.3 (29%)	86.7 (28%)

Ablation	Arg	Ret	Var	Ele	Gen	Usr	All
w/o Code Slice	74.8	77.0	68.8	75.1	75.5	73.9	70.8
w/o Type Hint	76.1	75.9	89.3	94.1	77.2	75.9	85.5
w/o COT Prompt	82.3	78.6	86.4	92.9	70.8	84.3	84.9
TypeGen	83.5	79.4	89.7	94.3	77.8	84.6	87.5

TypeGen is capable of **consistently improving** the zero-shot performance of type inference for language models **with different parameter sizes** and achieves **2x** ~ **3x** of improvements made by the Standard ICL setting.

# Improve the Reliability of Dynamic Type System

• Pathway #1 (Prevention): Use type inference to statically get the types of variables so that common static checking techniques can be used to detect potential issues.

Pynquin

Type Checking

**Test Case Generation** 

 Pathway #2 (Repair): Implement automatic repair methods to fix issues caused by the dynamic type system.



## **Outline of Thesis**



# How to Fix Existing Type Errors?

• Incorporate domain knowledge into prompt templates  $\rightarrow$  Domain-aware prompts.

Complete mask	if (fnType != null) {
line replace	<mask><mask> <mask><mask></mask></mask></mask></mask>
line after	<mask><mask> <mask><mask> if (fnType != null) { <mask><mask> <mask><mask></mask></mask></mask></mask></mask></mask></mask></mask>
Partial mask	return foundDigit && !hasExp;
partial after	<mask><mask> <mask> !hasExp;</mask></mask></mask>
partial before	return <mask><mask> <mask></mask></mask></mask>
Template mask	<pre>primitiveValues.put(double.class, 0);</pre>
method replace	<mask><mask><mask>(double.class, 0);</mask></mask></mask>
parameter replace	<pre>primitiveValues.put(<mask><mask><mask>);</mask></mask></mask></pre>
single replace	<pre>primitiveValues.put(double.class, <mask>);</mask></pre>
add parameter	<pre>primitiveValues.put(double.class, 0, <mask>);</mask></pre>
Template mask	if (endIndex < 0) {
expression replace	<pre>if (<mask><mask><mask>) {</mask></mask></mask></pre>
more   /&& cond	<pre>if (endIndex &lt; 0    <mask><mask>) {</mask></mask></pre>
replace operator	if (endIndex <mask> 0 ) {</mask>

#### Prompt-based program repair

- + Do not need training set.
- + Most effective.
- Require good prompt template design.

#### **Bug Line:**

user\_pass = '%s:%s' % (unquote(user), unquote(password))

### General prompt template:

user\_pass = '%s:%s' % (<mask>...<mask>unquote(password))

#### **Domain-aware prompt template:**

user\_pass = <mask>...<mask>('%s:%s' % (unquote(user), unquote(password)))

# **Domain-Aware Type Error Repair - TypeFix**



# **Phase I: Fix Parsing**



- Fix Pattern: Imdicate the exact coble change.
- Internal Context: Indicate the statement wheeefilis patterns is bold a phyly.
- External Context: Indicate the location of the statement in riterral cortext.

# **Phase I: Fix Parsing**



• Definition of Template Tree Node

A node is a quadruple (*bt, t, v, i*) where

 $bt \in \{Variable, Op, Literal, Builtin, Type, Attribute, Expr, Stmt\}$  is the base type of node,

t is the AST node type, v is the value, and i is the id.

# **Phase II: Fix Template Mining**

**Overall Methodology: Hierarchical Clustering** 

- Distance of Fix Patterns
  - Defined as 1 minus the rate of same nodes in two template trees.
  - Calculate from the root to leaves, i.e., two nodes can be compared if and only if their parent nodes are the same (top-down).
- Distance of Contexts
  - Defined as 1 minus the rate of same nodes in two template trees.
  - Calculate from leaves to the root, i.e., two nodes can be compared if and only if their children nodes in the leaf-root path are the same (bottom-up).

# **Phase II: Fix Template Mining**

- Abstraction of 2 Nodes *a* and *b* in Fix Patterns or Contexts
  - Same Node: *a* and *b* are exactly the same, and they can be reserved for the generalized fix template.
  - Value Abstraction: *a* and *b* have the same types but different values. We create a node with the same type and set the value as a special *ABS* token to indicate a hole.
  - **Type Abstraction:** *a* and *b* have the same base types but different types and values. We create a node with the same base type, and set the type and value as a special *ABS* token to indicate a hole.
  - Node Removal: *a* and *b* have no common attributes. We directly remove the two nodes.

Node: (*bt, t, v, i*)

```
a: (Literal, int, 1, 1024)
b: (Literal, int, 1, 2024)
res: (Literal, int, 1, -)
```

```
a: (Literal, int, 1, 1024)
b: (Literal, int, 2, 2024)
res: (Literal, int, ABS, -)
```

```
a: (Literal, int, 1, 1024)
b: (Literal, str, "a", 2024)
res: (Literal, ABS, ABS, -)
```

```
a: (Literal, int, 1, 1024)
b: (Op, add, -, 2024)
res:-
```

# **Phase III: Fix Template Matching**

- Select Fix Templates
  - We match the ASTs of the buggy programs with the contexts of mined fix templates.
  - For the same type of fix templates, we select the most detailed fix template since it has the most domain knowledge.

- Rank Different Types of Fix Templates
  - Group different fix templates with the same contexts.
  - Rank the fix templates in one group according to the occurrence frequency obtained in the mining phase.
  - Rank the groups according to the abstraction ratio, i.e., the rate of program holes that require LLMs to synthesize.

# **Phase IV: Patch Generation**

#### **Bug Line:**

user\_pass = '%s:%s' % (unquote(user), unquote(password))

#### **Domain-aware prompt template:**

user\_pass = <mask>...<mask>('%s:%s' % (unquote(user), unquote(password)))

### Patch:

user\_pass = to\_bytes('%s:%s' % (unquote(user), unquote(password)))



## **Evaluation**

TypeBugs										
Project	#B	TypeFix	PyTER	Codex	AlphaRepair	CoCoNuT				
airflow	14	9/9	4/4	7/7	1/6	0/4				
beets	1	0/0	0/1	0/0	0/0	0/0				
core	9	7/7	5/7	4/5	4/4	2/3				
kivy	1	0/0	0/1	0/0	0/1	0/1				
luigi	2	0/2	0/0	0/2	1/2	0/0				
numpy	3	0/3	0/2	0/1	0/2	0/0				
pandas	48	21/32	17/27	18/19	11/22	3/10				
rasa	2	2/2	0/0	2/2	0/0	0/0				
requests	4	4/4	4/4	2/2	0/1	0/0				
rich	4	2/3	0/1	1/1	0/0	0/0				
salt	8	5/8	5/5	4/5	1/5	0/2				
sanic	2	0/0	2/2	0/0	0/0	0/0				
scikit-learn	7	2/3	2/3	1/2	0/0	0/0				
tornado	1	0/0	1/1	0/0	0/0	0/0				
Zappa	3	3/3	1/1	0/0	1/3	0/1				
Total	109	55/76	41/59	39/46	19/46	5/21				
Fix Rate (%)	-	50.5	37.6	35.8	17.4	4.6				
			BugsInPy							
Project	#B	TypeFix	PyTER	Codex	AlphaRepair	CoCoNuT				
ansible	1	0/0	0/0	0/0	0/0	0/0				
fastapi	1	1/1	0/0	1/1	0/0	0/0				
keras	7	4/6	1/1	0/3	0/4	0/4				
luigi	7	4/5	3/5	3/3	0/0	0/0				
pandas	19	4/13	4/6	2/6	3/10	3/8				
scrapy	12	10/11	5/7	10/12	1/4	2/4				
spacy	1	0/1	0/1	0/0	0/1	0/1				
tornado	2	1/1	1/1	1/1	0/1	0/1				
youtube-dl	4	2/3	1/1	0/2	1/1	1/1				
Total	54	26/41	15/22	17/28	5/21	6/19				
Fix Rate (%)	-	48.1	27.8	31.5	9.3	11.1				



Unique type errors fixed by each approach.

TypeFix successfully fixes 55 and 26 type errors in two benchmarks, outperforming state-of-the-art approaches by at least 14 type errors and 9 type errors, respectively. Meanwhile, TypeFix obtains the most unique type error fixes in two benchmarks.

## **Evaluation**

Ammaaah	Түрі	EBUGS	Bug	sInPy
Арргоасп	#Unique	Coverage	#Unique	Coverage
TypeFix	24	83 (76.1%)	16	40 (74.1%)
PyTER	10	46 (42.2%)	5	18 (33.3%)

Comparison with rule-based approach

TypeFix achieves a template coverage of about 75% on both benchmarks, which is 30% larger than that achieved by fix templates manually defined in PyTER.

	Түр	eBugs	Bug	sInPy	
	#Correct	#Plausible	#Correct	#Plausible	
No Template	19	41	6	20	
Add	+11	+10	+3	+5	
Remove	+1	+1	+0	+0	
Replace	+6	+8	+4	+9	
Insert	+18	+16	+13	+17	
Total	55	76	26	41	

Ablation results also demonstrate the usefulness of fix templates mined by TypeFix under each category.

**Ablation Results** 

## Dynamic Run-time Environment

**HRE** 

## Improve the Reliability of Dynamic Run-time Environment



• Pathway #1 (Prevention):

Run-time environments provide implementations for external APIs used in the code.

 $\rightarrow$  Provide high-quality API recommendation to avoid API misuse.

## **Outline of Thesis**



# **Two Categories of API Recommendation Approaches**



# **APIBench-Q for Query-Based Approaches**

- Mining Stack Overflow
  - All posts from Aug 2008 to Feb 2021
  - 1,756,183 Java posts and 1,661,383 Python posts

Format Check

• 148,938 Java posts and 156,493 Python posts

Keyword Filtering

• 13,755 posts

Manual Check

• 1,320 Java queries and 1,925 Python queries

• Mining Tutorial Websites



ng Tutorials	lava		Intervice in Coogle Search
excript	Java		
(covers JDK 9-15)			
/L5 and C853	Pasia	Contification	How to
NSI, C99 C11)	Dasic	Gertification	HOW to
ion NumPy	1. Java Utility Methods	Interview	1. Java Algorithms How to
(C++ 20)	2. Java Examples by API	1 Jana Totadal / Oxfo	2. Java Collection How to
arp (8.0)	3. Java Examples	2 OCA Into Building Birchy	3. Java Data Type How to
ile els	4. Scala Tutorial	3 OCA law Operators	4. Java Data Structure How to
QL (8.0)	5. Java Design Patterns	Statements	5. Java I/O How to
	Tutorial	4 OCA Inva Method	6. Java Graphics How to
	6. Java Object Oriented Design	5 OCA Java Class Design	7. Java XML HTML How to
	Tutorial	6 OCA Java ABla	8. JavaFX How to
	7, Java Data Type Tutorial	7 OCA Java Exception	9. Java Stream How to
	8. Java I/O Tutorial	B. OCA Java Baulaw	10. Java Swing How to
	9. Java XML Tutorial	9 OCA Java Mork Evan	11. JDBC How to
	10. Java Collection Tutorial	10 OCA Java Mock Exam 2	12. Java Thread How to
	11. Java Regex Tutorial	11. OGA Java Mock Exam 3	13. Java Network How to
	12. Java Format Tutorial	12 OCA Java Mock Evan 4	
	13. Java Reflection Tutorial	13 OCA Java Mock Evan 5	
	14. Java Language Tutorial	14. OCA Java Mock Exam 6	JEE
	15. Java	15. OCA Java Mock Exam 7	1. ISE Tutorial
	16. Java Tutorial	16 OCA Java Mode Evan 8	2 ISP Treated
	17. Jar File Download	17. OCA Java Mock Evan 9	1 DA Tutorial
	18. Jar File / POM / Source	18. OCA Java Mock Exam 10	4. Spring Tutorial
	19. Java Products	19. OCA Java Mock Evant 11	5 JDBC Tutorial
	20. Java by API	20. OCA Java Mock Exam 12	6. Lucene Tutorial
	21. SCJP	21. OCA Java Mock Exam 13	7 MonooDB Tutorial
	22. Java Free Code	22. OCA Java Mock Fears 14	<ul> <li>konji Tutorial</li> </ul>
	23. Popular Jar Files from	23. OCA Java Mock Exam 15	9. JSON Tutorial
	Maven		10. Junit Tutorial
			11 Mayor Tironial
			12. Mayan/POM Bapository
	Advanced	GUI	Java 8
	1. Java Thread Tutorial	1. Java Swing Tutorial	1, Java Streams Tutorial
	2. Java Network Tutorial	2. JavaFX Tutorial	2. Java Date Time Tutorial
			3. Java Lambda Tutorial
			4. Scripting in Java Tutorial



DI	St	ack Over	flow	Т	itorial Web	sites
r L	Ori.	Exp.	Mod.	Ori.	Exp.	Mod.
Python	1,925	78,157	100,100	2,384	95,360	123,968
Java	1,320	80,343	68,640	5,243	319, 783	272,636

# **APIBench-C for Code-Based Approaches**

Product 🗸 Solutions 🗸 Open Source 🗸 Pricing Sign in Sign up Mining GitHub Explore Topics Trending Collections Events GitHub Sponsors Get email updates • General: 1,000 most starred + 1,000 most forked Topics Browse popular topics on GitHub repos at entire Github. Specific Domain: 500 most starred + 500 most forked repos under one topic. COVID-19 Homebridge Chrome The coronavirus disease 2019 Homebridge is a utility for tying Chrome is a web browser from the (COVID-19) is an infectious disease smart home devices together into tech company Google. caused by SARS-CoV-2. Apple's HomeKit framework, LOC Threshold LOC LOC Threshold #API Total number of APIs (only testset) **ДТ:1**. 

PL	Domain	#Projects	#Files	(per func)	(per func)	Standard	User-defined	Popular	of Short Func	of Long Func
	General	899	230,064	15.24	5.55	1,363,240	1,747,878	54,244	8.875	54.875
	ML	323	46,556	13.89	6.08	629,437	339,821	125,377	12.65	46.05
Duthon	Security	126	15,785	18.98	6.72	111,393	64,809	3,613	6	86.5
ryuton	Web	568	82,771	14.14	5.05	369,114	241,602	11,832	7.35	51.625
	DL	307	39,577	14.58	6.25	413,295	220,228	76,654	11.675	52.525
	General	935	1,056,790	11.16	4.06	5,164,481	3,808,124	36,178	6.26	19.2
	Android	377	87,468	8.24	2.91	517,461	267,141	75,069	7.28	16.8
Java	ML	52	41,377	12.82	4.77	194,013	136,963	0	7.52	19.74
	Testing	55	23,618	9.93	3.98	105,577	55 <b>,2</b> 41	22	6.44	15.68
	Security	58	20,445	12.35	5.32	125,558	74,471	1,243	6.88	20.78

## **Query-Based Baselines**

Approach	Category/ Data Source	PL	Venue	Year
	Query Reformulation			
Google Prediction Service [22]	Query expansion, modification	Any	-	2021
NLPAUG [40]	Query expansion, modification	Any	-	2021
SEQUER [8]	Query expansion, modification	Any	ICSE	2021
NLP2API [56]	Query expansion	Java	ICSME	2018
Quer	y-Based API Recommenda	ation		
RACK [58]	Official documentation, Stack Overflow	Java	ICSE	2016
KG-APISumm [38]	Official documentation, Wikipedia	Java	FSE	2019
Naive Baseline	Official documentation	Any	-	2021
DeepAPI [23]	Official documentation	Java	FSE	2016
Lucene <sup>[16]</sup>	Official documentation	Any	-	2021
BIKER [27]	Official documentation, Stack Overflow	Java	ASE	2018

	G	S	earc	h		÷ 🤅		
R.	2016 IEEE 234 ACK: A Univers	d Internation Autom Crov oharmad M fity of Saskan (masod rai Gener	nal Conference natic A vdsour Masudur Rahm cherwan, Canada Immon, chandral rating Q	on Software Anal PI Reco ced Kno xan Chanchal x "Singapore Mana roy)@usask.ca, "d Query-Spe	rsis, Evolution, and Reengineering mmeendation using weledge K. Key Denki Lo genere University, Stagarov and the Varsachag cific Class API Summaries			
ch. Un effective					Deen API Learning			
elevant uploitis	1				Boop Air Eduning			
stedge g an ex s posts xh que e that e	Austra		Xiao	dong Gu <sup>1</sup> , H The Hong Kong	ongyu Zhang <sup>1</sup> , Dongmei Zhang <sup>1</sup> , and S i University of Science and Technology, Hong I (xguaa,hunkim)@cse.ust.hk	Sunghun Kim <sup>1</sup> Kong, China	9 mako	edward /
tap 10 nising. nisper a				-	'Microsoft Research, Beijing, China		⇔ Code	<li>Issue</li>
in Top n recal	ABSTRAC		API	Method	Recommendation without	Worrying about the	1	moster -
ciation,	and/or code, o opers gain a q	ABST			Task-API Knowledge G	ар		marter -
Indies	specific tasks. a challenge in	Develop tionality		Qiao Huang	Xin Xia	Zhenchang Xing	-3	makced
ut 19%	an approach fi for API classe	ing an A languag		Aut	omated Ouery Reformul	ation for Efficient		aithub
. Code ch and	natural langua sentences extr	existing search \$		Search	based on Ouery Logs I	From Stack Overflow		docs
[21].	External ev	standing		betaen	cused on Query Logo	for black of error		ecorrela
word m ilarity b	tasks. The maj	generate		1	Kaibo Cao <sup>†</sup> . Chunvane Chen <sup>‡+</sup> . Sebastian Baltes <sup>‡</sup> . G	'hristoph Treude <sup>5</sup> . Xiane Chen <sup>¶+</sup>		ninaun
fally é nethodi	baseline appri	sequeno APL:	ABSTRA		NLP2API: Query Reform	mulation for Code Search		mproog
with u uring	baselines sum that the summ	RNN E	programmin documentati		using Crowdsourced Kn	owledge and Extra-Large		res
ut relev suming	retrieval sesual	API seq	lexical gap a	imkbcao@gmai	Data A	nalytics		scripts
gested t is perfi	CCS CONG	of indiv	documentatia semantic me	Abstract-As	Mohammad Masadur Ra Department of Company Science,	hman Chanchal K. Roy University of Saskatchewan, Canada		test
code :	engineering	from Gi	knowledge g describes AF	questions and a developers to eff	(masud.rahman, ch	anchal.rey]@tesank.ce	Ľ	] .codacy;
greatly	KEYWORI	approact	information i	for. There are b between the us	Alabraz-Software developers frequently inne generic natural language (NL) queries for code sourch. Underimately, such	them. They are carefully collected from the Q & A threads of four totarial sites-EodeArea, AreaDA Area2x	C	gitattrib.
resses i ressure	Croc summar	CCS (	recommenda based Knowl	fore, developers correcting misso	queries often do not load to any relevant results with contempo- rary code (or web) search engines due to vocabulary mismatch problems. In our isochoical research paper incorpied at BCSME	and CodeSires. From each Q & A thread, the question title is captured as a query, the code example is considered	C	.gitignor
c natur	"M. Liu, X. Pong, 1 Shanghai Key Lah "X. Penet in the co	<ul> <li>Softwa</li> </ul>	bridge the le calculate the	gramming langu is tedious for de	2010), we propose a technique-NLP2API-that reformulates such NL queries using erweitseutred knowledge and extra-large data analytics detined from Stack Chardine ID & A site. In this paper	as a ground truth for code search, and the API classes discussed in the accompanying prose are captured as the manufacth API classes (12) approximately and the first	_	
9998-185		Keywe	spired by ou Overflow not	on deep learning we construct a k	we discuss all the artifacts produced by our work, and provide necessary details for downloading and verifying them.	file required for the prototype's internal use. - cardid date/ contains the candidate API classes for 310		
0.1109/5	Chargeon use is g for profit or comm	API, de	gap, NIKIR	the original que approach trains	I. NLP2API: ARTIFACT DETAILS Our technique, NLP2API 121, takes a generic natural lan-	NL queries before their making with the hearistics. • SLPZAPI-Results-Borda contains the API classes		
	on the first page. ( must be howeved.)	L IN	the query's s	generate candid original query.	gaage query as an input and then reformulates the query for code search using appropriate API classes mixed from	suggested by our technique when only <i>Royala</i> acrow heuristic is employed.		
	for. Request perm ESEC/FSE '19, Aug	to parse	API descript	outperforms five to 33.5% boost	Stack Overflow, Our algorithm design and empirical evaluation of the technique produced different sets of artifacts. These	<ul> <li>MEP2API-Results-Q-A-Proximity contains the API classes suggested by our technique when only ar-</li> </ul>		
	© 2019 Annotatio ACM INBN 978-1-	Obtaini	API to help - their tasks. C	Index Terms	artifacts can be divided into three major groups as follows: A. Algorithm Device and Tool Deplementation	<ul> <li>NLP2RPT-Regults centains the API classes for 310</li> <li>NL queries suggested by our technique.</li> </ul>		
	adding and a	method ful in th	the effectives ommendatio	and boy ou	<ul> <li>nlp2api-runner is our prototype. It takes a generic sector of the sector and sector a could be a</li> </ul>	C. Tool Evaluation: Query Reformulation Performance		
			study with 21 of NKER for	Stack Overfle	of Java API classes relevant to the query.	<ul> <li>code-ext-index is the corpus containing 4,170 code segments including the ground trath. The corpus is in-</li> </ul>		
		Permission classroom		(Q&A) site for acquisition. Ov	mands for running and evaluating our prototype. A screenshot of the prototype's run is also attached.	deted and used by Larow for performing code search. . search-angine-reax contains original search re-		
		on the first must be ha		mulated a large valuable reposit	<ul> <li>data/ contains Java programming keywords and step words used for the natural language pre-processing.</li> </ul>	saits and our improved results for 310 NL queries. II. IMPLICATIONS OF OUR ARTIFACTS		
		to pest on the Reque	Permission to m	developers enco as how to ane	<ul> <li>dataset/contains texts and code segments from 656K.</li> <li>Stack OverBow Q &amp; A threads, and the Lacree index descined from them.</li> </ul>	<ul> <li>Benchmarking: Our dataset and results can be used as the invacionant far future tools and techniques.</li> </ul>		
		FSE-16, 5 (c) 2006 / Max/Mr-	for profit or com on the first pute	two languages? [3], they tend a	driveneed from them. Compressed files should be de- compressed in the same directory.	<ul> <li>Reproducibility &amp; Kick-starting: Our prototype and intermediate data offer replication and possible repro-</li> </ul>		
		.aqural.t	must be honored to post on server	assist developer	<ul> <li>xmerwettr comms our imperimentation of fastTeat.</li> <li>[1], and our skip-gram model trained on the Stack Over- flow Q &amp; A shreads. Is movides word-architect for</li> </ul>	duchtity. Our source onde can kiele-start the next tool. • Remarkility: Our skip-gram models, 1.3 million Q & A		
			ASE '17, Septemi 0, 2018, Association	provides a seat	both a natural language layword and an API class which are then used to determine their semantic menimize	second and core represent true Stars therefore can be reused for various other purposes (e.g., code completion).		
			ACM ISBN 978-1 https://doi.org/1	However, eve	<ul> <li>jdk-fasttest-checker centains necessary com- mands to check fast Text and JDK 8 installations.</li> </ul>	III. NLP2API: DOWSLOAD LINK The artifacts are uploaded in a Google Drive, and can be		
				not easy for des [6]. There are	<ul> <li>LICENSE outlines the licensing details of our replication package and experimental data.</li> </ul>	downloaded from: https://goo.gl/Meujau REFERENCES		
				First, there exis	B. Soil Evaluation: API Suggestion Performance	<ol> <li>F. Bejanereda, E. Dram, A. Juslin, and T. Hiladim. Enriching word weines with automal information. arXiv preprint arXiv:1617.0608, 2014.</li> <li>M. M. Robranz and C. K. Ray, Effective enformation of query for coin search.</li> </ol>		
				<sup>1</sup> Carresponding <sup>2</sup> https://rackevet	guage queries and ground with Java API classes for	unter aventement knowledge and searchege data analytics. In Proc. A2007, page 11, 2018.		
					2576-3148/18/531.00-02018 IEEE 71 DOI 10.1108/ICSME.2018.00088	4 @computer		

# 

nakcedward / nlpaug Pub	lie)	Sponsor Q Notifications & Pork 433	17 Star (3.9k)
Code 🕑 Issues 59 🗈 F	Pull requests (7) 💿 Actions 🛞 Projects 🛈	Security 🗠 Insights	
P master + P 3 branch	tes 🛇 26 tags	Go to file Code + About	
🕹 makcedward Merge pull	request #306 from makcedward/dev	23869cb on Jul 7, 2022 ③738 commits Pata augmentation for Pata augmentation for Pata augmentation for Pata augmentation for	NLP
.github	Update FUNDING.yml	3 years ago nip data-acience n	nachine-learning
docs docs	release 1.1.11	10 months ago natural-language-process	ing al ml
example	Typo in Word Embeddings Augmenter e	sample last year artificial-intelligence a	Augmentation
nipaug 📫	release 1.1.11	10 months ago	
nes 🖿	add LAMBADA training example	2 years ago the MIT license	
scripts	(#295) fix incorrect lambda label	10 months ago	
test.	(#295) fix incorrect lambda label	10 months ago	
.codacyymi	ignore MD file in code checking	4 years ago 😌 433 forks	
	Initial commit	4 years ago	
.gitignore	fix skipping model dir	3 years ago	
-		Releases 25	

# **Effectiveness of Query-based Approaches**

### Class Level vs. Method Level

Baseline	Level	Success Rate@k			MAP@k				MRR	NDCG@k				
	20101	Top-1	Top-3	Top-5	Top-10	Top-1	Тор-3	Top-5	Top-10		Top-1	Тор-3	Top-5	Top-10
RACK	Class	0.17	0.30	0.35	0.41	0.17	0.23	0.24	0.24	0.25	0.17	0.24	0.26	0.28
KG-APISumm	Class	0.19	0.33	0.40	0.50	0.19	0.25	0.26	0.27	0.28	0.19	0.24	0.27	0.31
Naivo Bacolino	Class	0.07	0.13	0.16	0.21	0.07	0.10	0.10	0.10	0.11	0.07	0.09	0.10	0.13
Inalve Dasellile	Method	0.02	0.03	0.04	0.05	0.01	0.02	0.03	0.03	0.03	0.07	0.09	0.10	0.13
DoopAPI	Class	0.19	0.27	0.29	0.30	0.19	0.22	0.23	0.23	0.23	0.17	0.22	0.23	0.24
DeepAri	Method	0.05	0.09	0.10	0.11	0.05	0.07	0.07	0.07	0.07	0.17	0.22	0.23	0.24
Lucono	Class	0.15	0.21	0.24	0.29	0.15	0.17	0.18	0.17	0.19	0.12	0.15	0.16	0.20
Lucene	Method	0.04	0.08	0.10	0.14	0.04	0.06	0.06	0.06	0.07	0.12	0.15	0.16	0.20
RIVED	Class	0.33	0.51	0.59	0.67	0.33	0.41	0.41	0.39	0.44	0.27	0.32	0.35	0.42
DINEN	Method	0.12	0.23	0.29	0.37	0.12	0.16	0.18	0.18	0.19	0.27	0.32	0.35	0.42

**Finding**: Existing approaches fail to predict 57.8% method-level APIs that could be successfully predicted at the class level. Accurately recommending the method-level APIs still remains a great challenge.

# **Impact of Query Reformulation Techniques**

Finding: Query reformulation techniques are quite effective in helping query-based API recommendation approaches give the correct API by adding an average boost of 27.7% and 49.2% on class-level and method-level recommendations, respectively.





(a) Query Expansion

**Finding**: Query expansion is **more stable and effective** to help current querybased API recommendation approaches give correct APIs than query modification.

(b) Query Modification

## **Code-Based Baselines**

Approach	Representation	PL	Venue	Year							
Practical IDE											
PyCharm [30]	Code tokens	Python	-	2021							
Visual Studio Code [43]	Code tokens	Python	-	2021							
Eclipse [17]	Code tokens	Java	-	2021							
IntelliJ IDEA [29]	Code tokens	Java	-	2021							
Approach in Academia											
TravTrans [32]	AST	Python	ICSE	2021							
PyART [24]	Token flow Data flow	Python	ICSE	2021							
Deep3 [59]	AST, DSL	Python	ICML	2016							
FOCUS [49]	API Matrix	Java	ICSE	2019							
PAM [18]	API sequence	Java	FSE	2016							
PAM-MAX	API sequence	Java	FSE	2016							



Code Prediction by Feeding Trees to Transformers Jinman Zhao Yuchi Tian Satish Chardra

#### PyART: Python API Recommendation in Real-Time

Xincheng He\*, Lei Xu\*1, Xiangyu Zhang1, Rai Hao\*, Yang Feng\* and Baowen Xu\* State Key Laborator, For Novie Schware Technology, Nanjia University, China <sup>1</sup> Paroke University, USA <sup>3</sup> sinchenghe 2016 @ gmail.com, forgyang @#ja.edu.cn, bwrat@sji.edu.cn <sup>4</sup> raikao, gm@gmail.com, forgyang@#ja.edu.cn, bwrat@sji.edu.cn

<text><text><text><text><text>

L DETEINCTUNE
 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

 L DETEINCTUNE

opending author

<sup>2</sup>https://githuk.com/Tellinins/python-skeletons <sup>2</sup>https://githuk.com/python/typeshed <sup>2</sup>https://scaletonlo.com/python/typeshed

#### Probabilistic Model for Code with Decision Trees

Veselin Raychey Pavol Bielik Martin Vechev ment of Computer Science epartment of Computer Science ETH Zürich. Switzerland artment of Computer Science ETH Zärich, Switzerland ETH Zarich, Swit

Ab

In th

plos

abili

appr Cali

ods; Aut Key

2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) FOCUS: A Recommender System for Mining API

Function Calls and Usage Patterns

Phaong T. Nguyen, Jari Di Rocco, Davide Di Ruscio Università degli Studi dell'Aquila Arresterdam, Netherlands Massimiliano Di Penta Contram Washande & Informatica Ansterlaum, Netherlands (feutrame Jastuares) @cwi.ul dipenta@ausionis.it L'Aggila, Italy L'Aquila, Italy (firstname.lastname)@univaq.it

Above—before forbance interact with APS on a data on the second second

I. INTRODUCTION invoke, considering that it has already invoked these other API methods?"

<section-header><section-header><section-header><section-header><text><text><text><text>

#### Parameter-Free Probabilistic API Mining across GitHub

Jaroslav Fowkes Charles Sutton School of Informatics University of Edinburgh, Edinburgh, EH8 9AB, UK ilfowkes, csuttoni@ed.ac.uk

00

ABSTRACT	Despite a number
Existing API miming algorithms can be difficult to use and the require sequence parameter trating call the intrust of set of API alion can be larger, highly remaindant and difficult to the API and the approximation of the approximation of the animage the mass hierarchical parameters. We show that a strategies a strategies are approximately a strategies of the approximation of the approximation of the API approximation of the approximation of the API approximation of	well known ome suc Al <sup>17</sup> mining tools ha development environ We suggest that the quality of the extr the patterns esturns highly reducations on a methods. To a large are variations on a ways to repeat the The fundamental Al <sup>2</sup> I mining method Specifically, API an sequence raising, We mining Direature the
CCS Concepts	and much those all

 Software and its engineering → Documentation
 Keywords

API mining, sequential pattern mining 1. INTRODUCTION

 INTRODUCTION Learning the application programming interface (API) of an unfamiliar library or software framework can be a signifi-cant shocks of the evolosping [10, 31]. This is only concritent by the fact that API decumentarians can often be incomplete or analyzeness [17]. Fortunatioly, an operaturity to address this problem has arise out of the simulaneous growth in the smooth of same ore due that is saidable out into and the growth of large scale data mining and machine learning methods, periode (11). A sub-the simulation of the second scale outperiod periods (11). and the soft of the simulation of the second periods (11). and the simulation of the second scale outperiods periods (11). and the simulation of the software outperiods (11). This confinence has enabled the development of APT missing methods [44, 03], which aim to automatically extract a set of API patterns, which are lists of API methods that are usually used together, and that together characterize how as API is used.

Premission to make cligitid or hand copies of all or part of this work to classroom use is granted without for percisided that copies are not made for profit or commercial advantage and that copies bear the startics and the on the first cape. Copyrights for components of this sock smooth by to base the historical Advancing with copies point this sock smooth by to base the historical Advancing with copies point single points. In Regard permission, from Permission Phase ang PEE'16, November 13-18, 2016, Seattle, WA, USA ACM, 978-1-4500-4218-4918/11\_\$15:00

254

er of interesting proposed tools, including ach as MAPO [40] and UPMiner [40], so far save not yet gained wide-spread adoption in connecte and as Eclipse and Vienal Studio, he fundamental reason for this is that the he fundamental reason for this is that the tracted patterns is not yet high enough nod by current methods are numerous and (Section 5). For example, Figure 5 shows patterns extracted by two state of the art or sector 4 the nuttoes from both methods extent, the patterns from both met a theme, repeating different variations of e same few API methods. al reason for this, we argue, is that curren-ods are built on statistically shaky groun-The fundamental means in this we maps, in take summa processing the strength of the strength property appears means, which is a fairly of techniques from the data of from these strength of techniques from the data in the strength of techniques of the strength of the transmitter of the strength of the strength of the strength of the strength of techniques are of the strength of the transmitter of the strength o

r independence among items in the sequence. Within th API mining literature, methods like MAPO and UPMin API mining illerature, northods like MAPO and UPMiner apple chatering and archical priority for orders of the number of apple chatering and archical priority for orders of the number of the state of the state of the state of the state of the respective state of the state of the state of the state of the even with this state, polationtial extendingly remains. We address this problem by developing new a mining algorithm that returns API patterns that at state of algorithm angested by chates, this is, the mean information sequences aspected by chates, this is, the mean information sequences aspected by chates, this is, the mean information sequences

# **Cross-Domain Performance**

## General: multiple domains

Training Domain	TravTrans					Deep3					PyART				
	ML	Security	Web	DL	Μ	Ĺ	Security	Web	DL	M	L Se	curity	Web	DL	
ML	0.64	0.58	0.53	0.71	0.4	2	0.41	0.36	0.48	0.3	9 (	0.35	0.40	0.40	
Security	0.40	0.54	0.54	0.39	0.3	1	0.51	0.42	0.29	0.3	6	0.48	0.47	0.36	
Web	0.54	0.63	0.64	0.51	0.3	3	0.42	0.46	0.31	0.4	2 (	0.47	0.50	0.40	
DL	0.66	0.58	0.50	0.68	0.4	4	0.39	0.33	0.44	0.4	3 (	0.36	0.38	0.45	
General	0.72	0.76	0.78	0.74	0.5	5	0.65	0.62	0.57	0.4	4 (	0.44	0.46	0.46	

**Finding**: Training on **multiple domains** helps the current approaches to recommend APIs in different single domains, and the performance is **even better than** only training on the certain single domain.

## Improve the Reliability of Dynamic Run-time Environment

- Pathway #1 (Prevention): Provide accurate API recommendation to avoid external API misuse.
- Pathway #2 (Detection): Detect compatibility issues in the configuration files before software usage.



## **Outline of Thesis**



# **Source-level Run-time Environment Conflict Check**

- Version constraints defined in configuration files are not reliable.
- We should validate whether the **import statements** in source code can be successfully executed.



### Version-level Check

# **Source-level Run-time Environment Conflict Check**

• Installation Check: Assign the correct Python version and check whether a package can be successfully installed based on configuration files.



# **Installation Check**

Python Version Assignment



# **Source-level Run-time Environment Conflict Check**

• **Dependency Check:** Check potential conflicts between indicated versions in configurations and installed versions via pip.



# **Dependency Check**

### Metadata Check

- Existence of file <package>-<version>.dist-info.
- Top modules in file top\_level.txt.

## • Run-time Environment Check

- Solve the valid dependencies in configurations provided by developers.
- Collect the installed dependencies in the run-time environment built by Installation Check.
- Check inconsistences between the required dependencies and installed dependencies.

## Source Files Check

- Locate the source files based on the modules provided in the configurations.
- Check the syntax of all source files.

# **Source-level Run-time Environment Conflict Check**

• Import Validation: check potential conflicts between import statements in source code and the installed run-time environment.


# **Import Validation**

- Imports
  - Internal Imports: introduce local modules within the project.
  - External Imports: require third-party packages from the run-time environment.
- Collect Local Modules
  - All Python files and sub-directories with \_\_init\_\_.py file in the same directory.
  - Image files such as .so and .pyd.



# **Import Validation**

- Import Blocks
  - Developers employ different methods to handle different run-time environments, such as using if-else statements and try-except statements to incorporate import statements.

```
if sys.platform.startswith("java"):
    import platform
    os_name = platform.java_ver()[3][0]
    if os_name.startswith("Windows"): # "Windows XP", "Windows 7", etc.
        system = "win32"
    elif os_name.startswith("Mac"): # "Mac OS X", etc.
        system = "darwin"
    else: # "Linux", "SunOS", "FreeBSD", etc.
       # Setting this to "linux2" is not ideal, but only Windows or Mac
       # are actually checked for and the rest of the module expects
       # *sys.platform* style strings.
        system = "linux2"
else:
    system = sys.platform
```

• Block analysis aims to reformulate an import block to a boolean expression, so that we know whether an import block is successfully executed.



If con1: import a import b try: import c except Error1: import d import e else: import f import g







#### **Boolean Expression**



# **Import Validation**

• Execute imports to validate boolean expression:

```
((a and b) and (c or (d and e))) or (f and g)
```



- All Libraries
  - 8,282 packages and 338,069 releases on PyPI Platform.
- Installed Libraries (pass the Installation Check)
  - 7,830 (95%) packages and 303,377 (90%) releases.
- Validated Libraries (pass all checks)
  - 5,371 (65%) packages and 131,720 (39%) releases.

- Incomplete Configuration
  - Missing configuration files 281
  - Missing required libraries for setup 3,318
  - Missing Python versions 55,138 (16%)
  - Missing required libraries for direct imports 142,521 (42%)

Finding: Developers tend to provide **inadequate** configurations for the usage of libraries, especially for **Python versions and direct imports in source code**.

- Incorrect Configuration
  - Dependency conflicts in setup 6,318
  - Incorrect Python versions 4,155
  - Other run-time errors in setup 3,464
  - Inconsistent configurations with metadata 592
  - Inconsistent version numbers with release dates 12,018
  - Missing required modules for indirect imports 11,023
  - Inconsistent modules in direct imports with installed dependencies 6,678
  - Other run-time errors in imports 8,178

Finding: Developers make mistakes in writing configurations since **19% of configuration issues are incorrect configurations**. What's more, about 50% of incorrect configuration issues can only be detected by Import Validation, indicating the **importance of sourcelevel validation**.

- Incorrect Code
  - Missing source code 2,588
  - Parsing error 431
  - Multiple version control failure 15,507 (5%)

Finding: Incorrect configurations **can hardly be handled** by the multiple version control logic in source code, as there are 5% of library releases suffering from import block failures.

#### Conclusion & Future Work

FOUR

#### Conclusion



#### **Future Work**



#### Synergistic Type Inference:

- LLMs and static inference handle different kinds of variables
- LLMs get instant feedbacks from type checkers and improve the predictions
- Single-variable inference  $\rightarrow$  multiple-variable inference  $\perp$



#### **Future Work**



High-quality API Recommendation:

- Make use of query reformulation techniques.
- Train the deep learning model on multiple-domain data.

Run-time Environment Dependency Inference:

• Infer correct configurations of run-time environments based on the source code of software.

# **List of Publications**

- [ICSE'24] Less is More? An Empirical Study on Configuration Issues in Python PyPI Ecosystem.
- [ICSE'24] Domain Knowledge Matters: Improving Prompts with Fix Templates for Repairing Python Type Errors.
- [FSE'24] Less Cybersickness, Please: Demystifying and Detecting Stereoscopic Visual Inconsistencies in Virtual Reality Apps.
- [LLM4Code] Enhancing LLM-Based Coding Tools through Native Integration of IDE-Derived Static Context.
- [ASE'23] Generative Type Inference for Python.
- [ASE'23 Industry Challenge] REEF: A Framework for Collecting Real-World Vulnerabilities and Fixes. 88

# **List of Publications**

- [TSE'23] Prompt Tuning in Code Intelligence: An Experimental Evaluation.
- [TSE'23] API Usage Recommendation via Multi-View Heterogeneous Graph Representation Learning.
- [TSE'22] Revisiting, Benchmarking and Exploring API Recommendation: How Far Are We?
- [FSE'22] No More Fine-tuning? An Experimental Evaluation of Prompt Tuning in Code Intelligence.
- [ICSE'22] Static Inference Meets Deep Learning: A Hybrid Type Inference Approach for Python.

#### **Awards**

- ACM SIGSOFT Distinguished Paper Award (ASE'23).
- ACM SIGSOFT Distinguished Paper Award Nomination (ICSE'22).
- Distinguished Paper Award of Industry Challenge Track (ASE'23).





 Static Inference Meets Deep Learning: A Hybrid Type Inference Approach for Python Yun Peng, Cuiyun Gao, Zongjie Li, Bowei Gao, David Lo, Qirun Zhang, Michael Lyu
 ONINA
 OOI S Pre-print I Media Attached

NOMINATED FOR DISTINGUISHED PAPER







香港中文大學 The Chinese University of Hong Kong