Do Machine Learning Models Produce TypeScript Types That Type Check?

Ming-Ho Yee[1] and Arjun Guha[1,2]
[1]Northeastern University
[2]Roblox

July 20, 2023
ECOOP 2023

Type migration: JavaScript to TypeScript

- Incremental migration
- Static type checking
- Better documentation
- Editor integration

Type migration: JavaScript to TypeScript

- Incremental migration
- Static type checking
- Better documentation
- Editor integration

```
function f(s) {
    return s.
}
```

abc f
abc s

```
function f(s: string) {
    return s.
}
```

Symbol interface Symbolvar
charAt
charCodeAt
codePointAt
concat

Machine learning for type prediction

Predict the most likely type annotation for the given code fragment

Machine learning for type prediction

Predict the most likely type annotation for the given code fragment

DeepTyper [ESEC/FSE 2018]

Machine learning for type prediction

Predict the most likely type annotation for the given code fragment

DeepTyper [ESEC/FSE 2018]

LambdaNet [ICLR 2020]

Machine learning for type prediction

Predict the most likely type annotation for the given code fragment

DeepTyper [ESEC/FSE 2018]

LambdaNet [ICLR 2020]

InCoder [ICLR 2023]

Machine learning for type prediction

Predict the most likely type annotation for the given code fragment

DeepTyper [ESEC/FSE 2018]

LambdaNet [ICLR 2020]

InCoder [ICLR 2023]

```
function f(x) {
    return x + 1;
}
```

Type of x	Probability
`number`	0.4221
`any`	0.2611
`string`	0.2558
other	

Type of x	Probability
`string`	0.9512
`number`	0.0474
`Function`	0.0006
other	

Machine learning for type prediction

Predict the most likely type annotation for the given code fragment

DeepTyper [ESEC/FSE 2018]

LambdaNet [ICLR 2020]

InCoder [ICLR 2023]

```
function f(x) {
    return x + 1;
}
```

Type of x	Probability
number	0.4221
any	0.2611
string	0.2558
other	

Type of x	Probability
string	0.9512
number	0.0474
Function	0.0006
other	

```
function f(x: _hole_) {
    return x + 1;
}
```

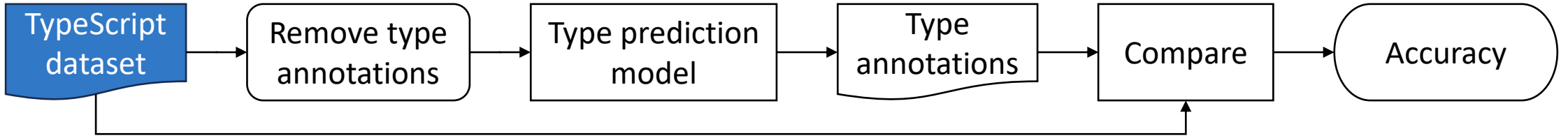
```
function f(x: number) {
    return x + 1;
}
```

TypeWeaver: type check the type annotations

TypeWeaver: type check the type annotations

Prior work:

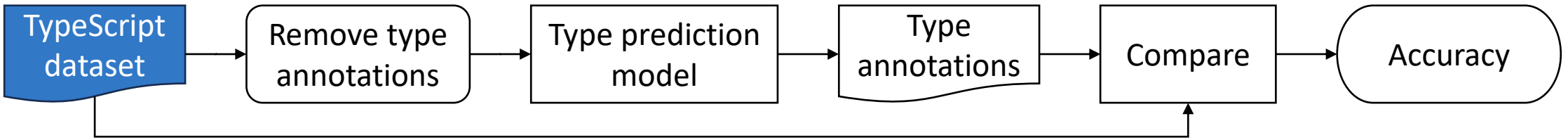
$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$



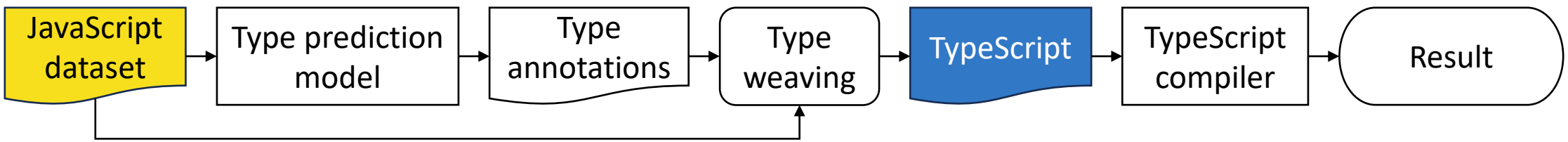
TypeWeaver: type check the type annotations

Prior work:

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$



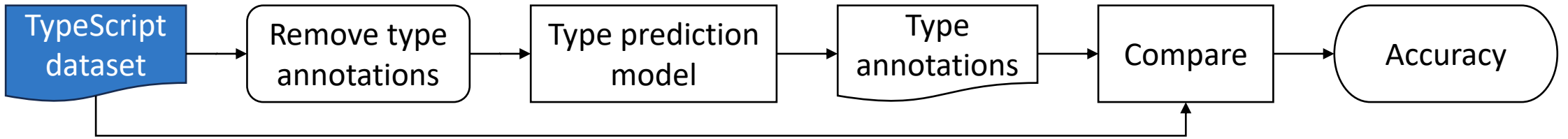
Our tool (TypeWeaver):



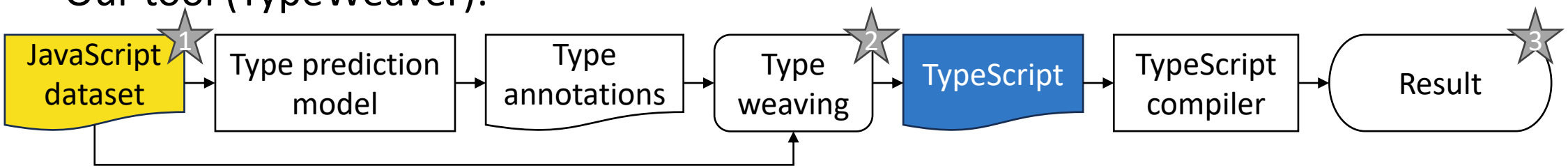
TypeWeaver: type check the type annotations

Prior work:

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$



Our tool (TypeWeaver):



This talk

Constructing the JavaScript dataset

1. Top 1,000 most downloaded packages

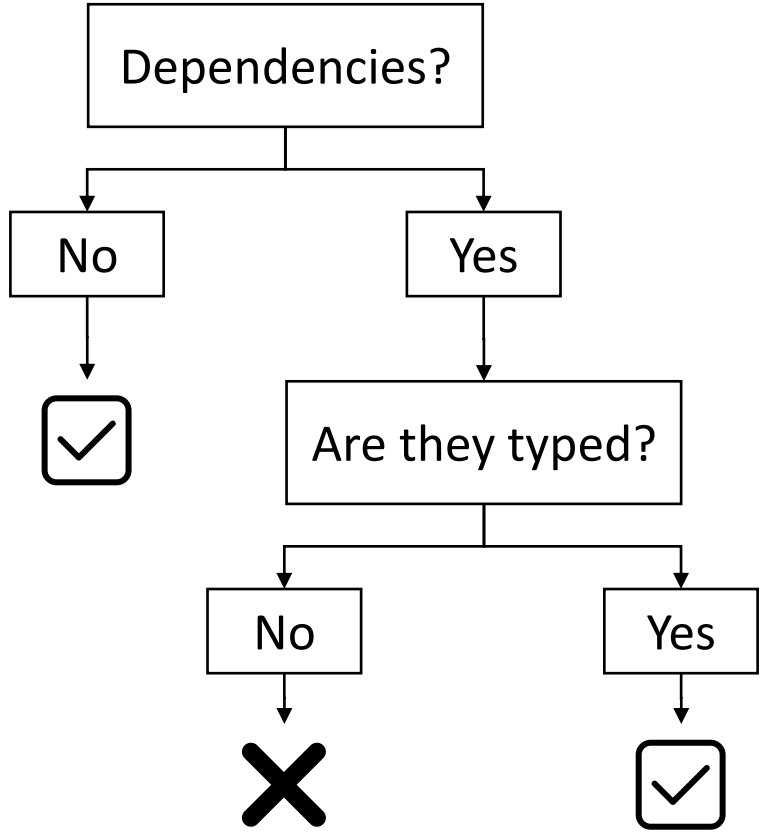
2. Download source code

3. Filter and clean

4. Check dependencies

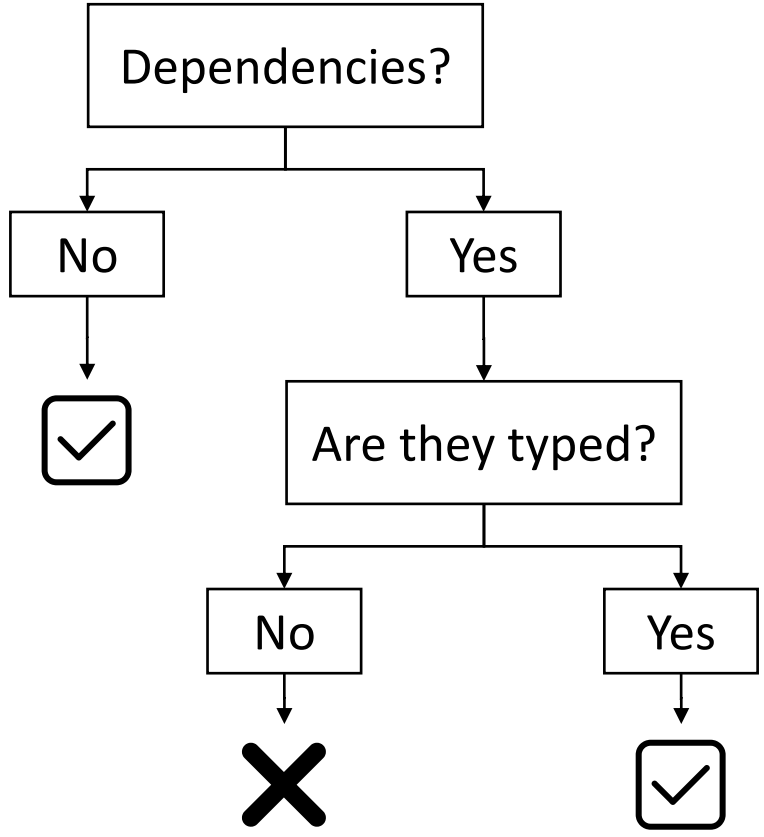
Constructing the JavaScript dataset

1. Top 1,000 most downloaded packages
2. Download source code
3. Filter and clean
4. Check dependencies



Constructing the JavaScript dataset

1. Top 1,000 most downloaded packages
2. Download source code
3. Filter and clean
4. Check dependencies



Result: 513 packages

Type weaving: JS + type annotations = TS

```
function f(x, y) {
    return x + y;
}
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Type weaving: JS + type annotations = TS

```
function f(x, y) {
    return x + y;
}
```

```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Type weaving: JS + type annotations = TS

```
function f(x, y) {
    return x + y;
}
```

```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Type weaving: JS + type annotations = TS

```
function f(x, y): string {
    return x + y;
}
```

```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Type weaving: JS + type annotations = TS

```
function f(x, y): string {
    return x + y;
}
```

```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Type weaving: JS + type annotations = TS

```
function f(x: string, y): string {
    return x + y;
}
```

```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Type weaving: JS + type annotations = TS

```
function f(x: string, y): string {
    return x + y;
}
```

```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Type weaving: JS + type annotations = TS

```
function f(x: string, y: number): string {
    return x + y;
}
```

```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Type weaving: JS + type annotations = TS

```
function f(x: string, y: number): string {
    return x + y;
}
```

```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

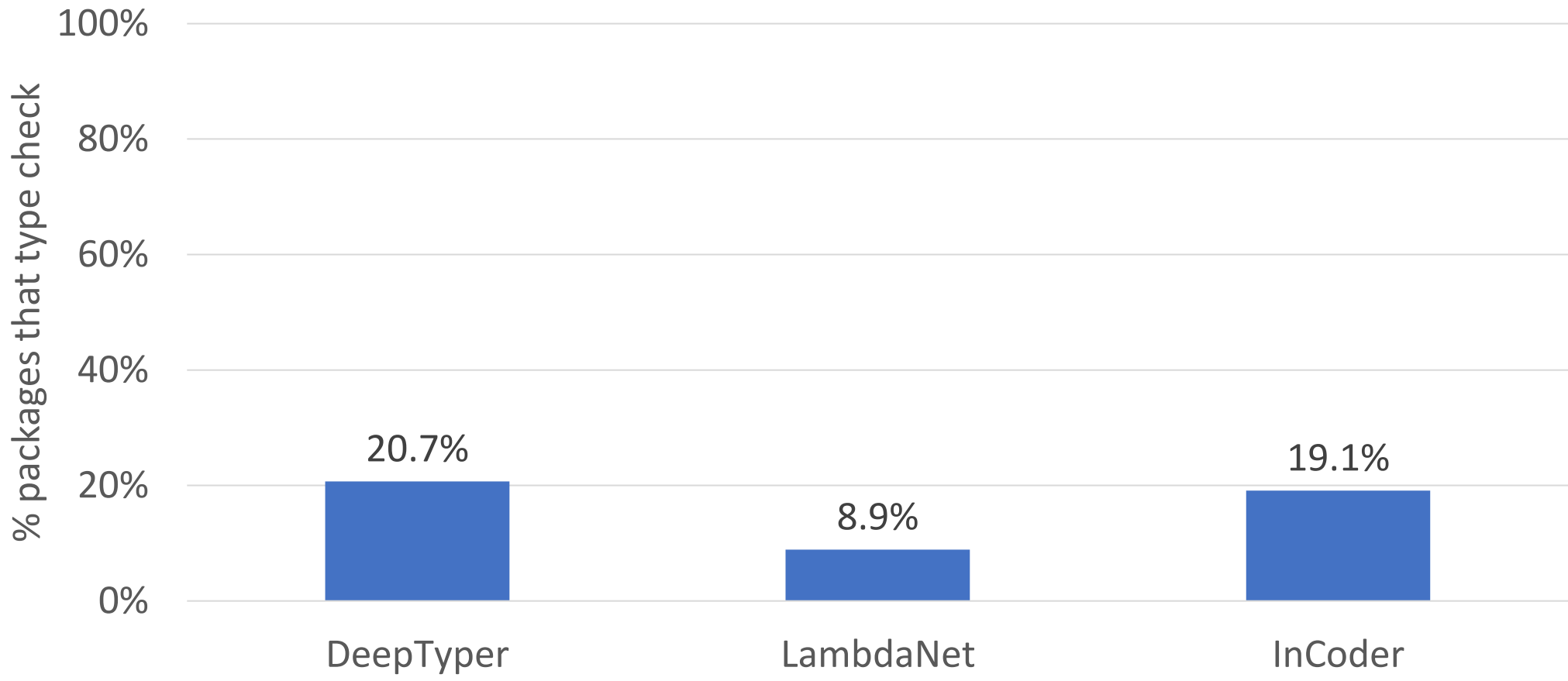
Type weaving: JS + type annotations = TS

```
function f(x: string, y: number): string {
    return x + y;
}
```

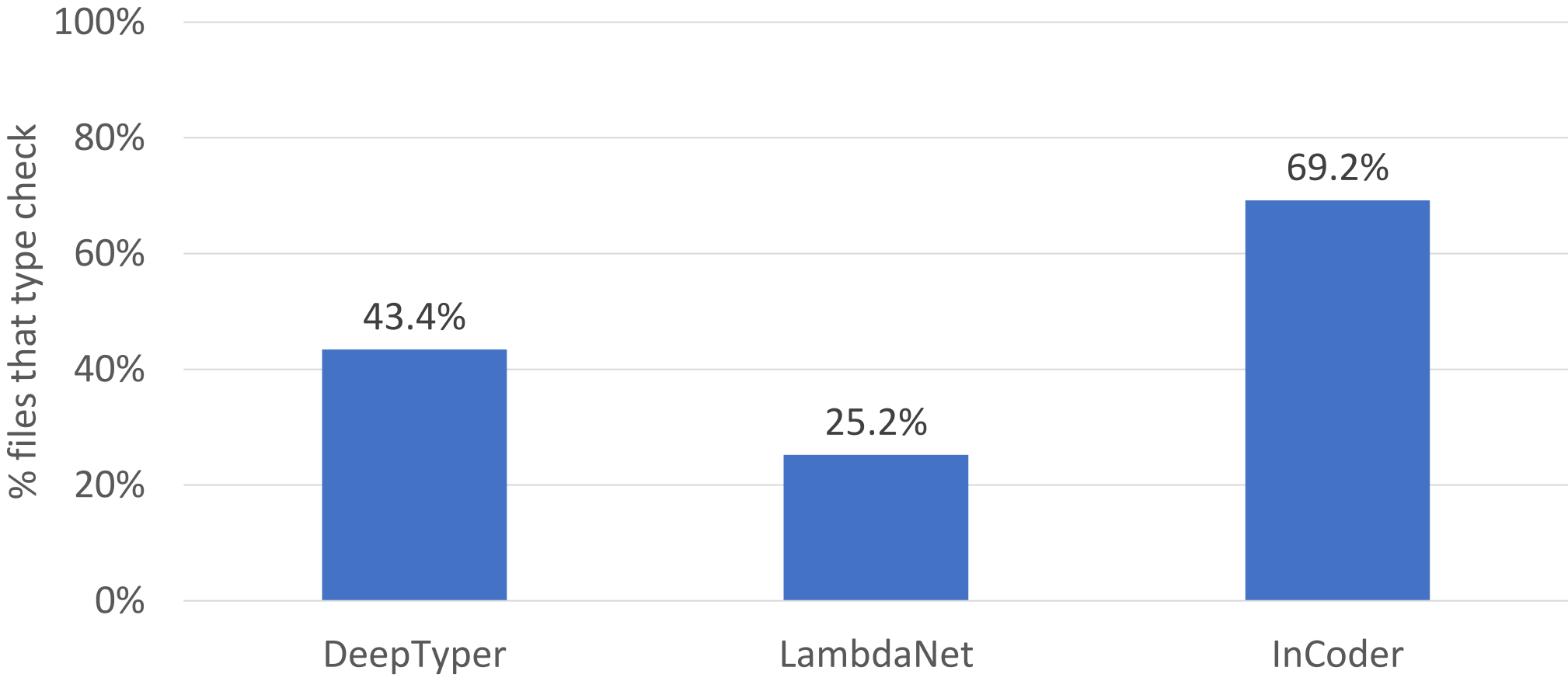
```
FunctionDeclaration
  Identifier
  Parameter
    Identifier
  Parameter
    Identifier
  Block
    ReturnStatement
      …
```

Token	Type	Probability
function		
f	string	0.6381
(		
x	string	0.4543
,		
y	number	0.4706
)		
{		
return		
x	number	0.3861
+		
y	number	0.5039
;		
}		

Do migrated packages type check?

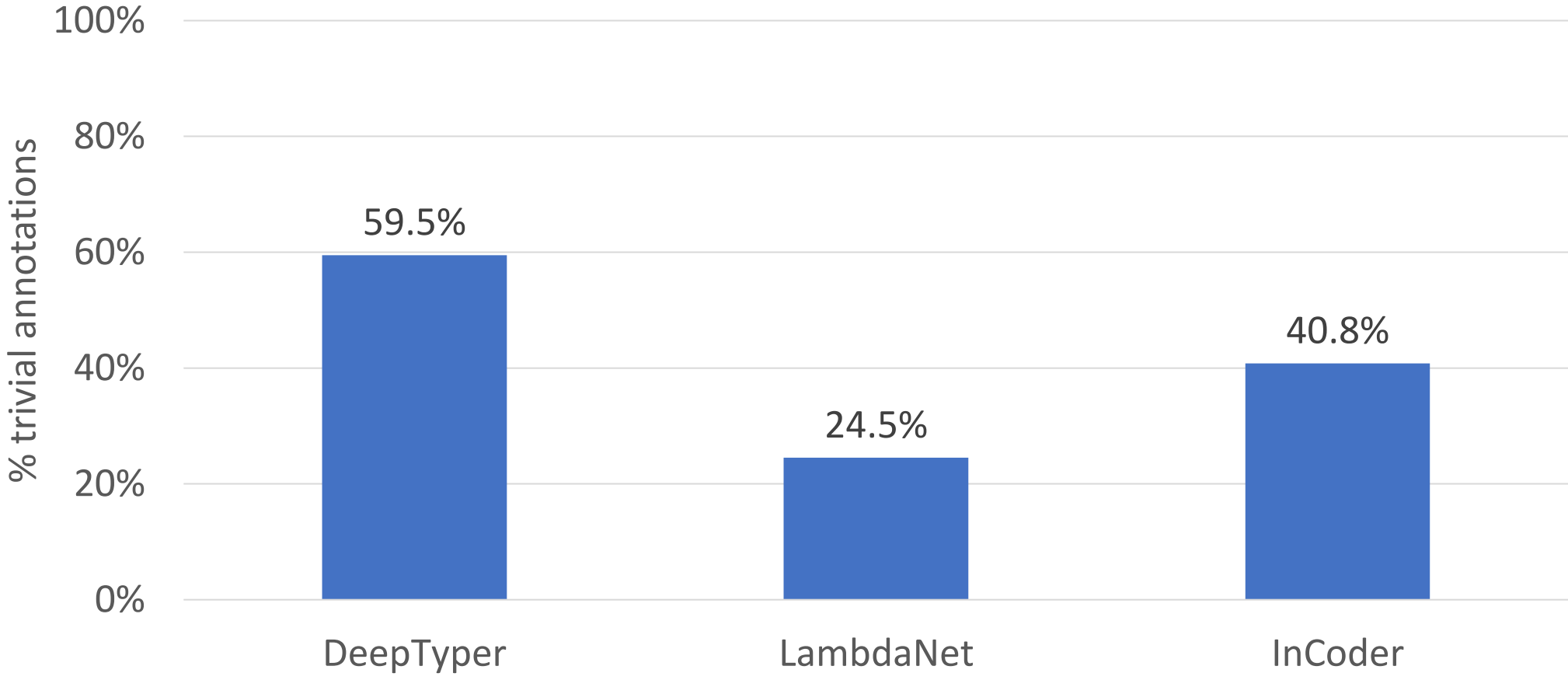


Do migrated files type check?

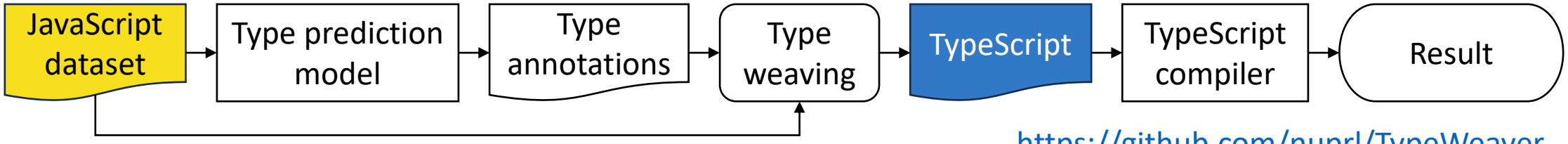


How many type annotations are trivial?

(within the files that type check)

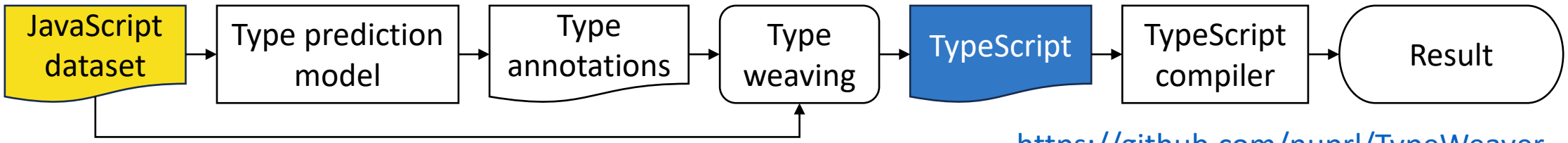


Conclusion: TypeWeaver

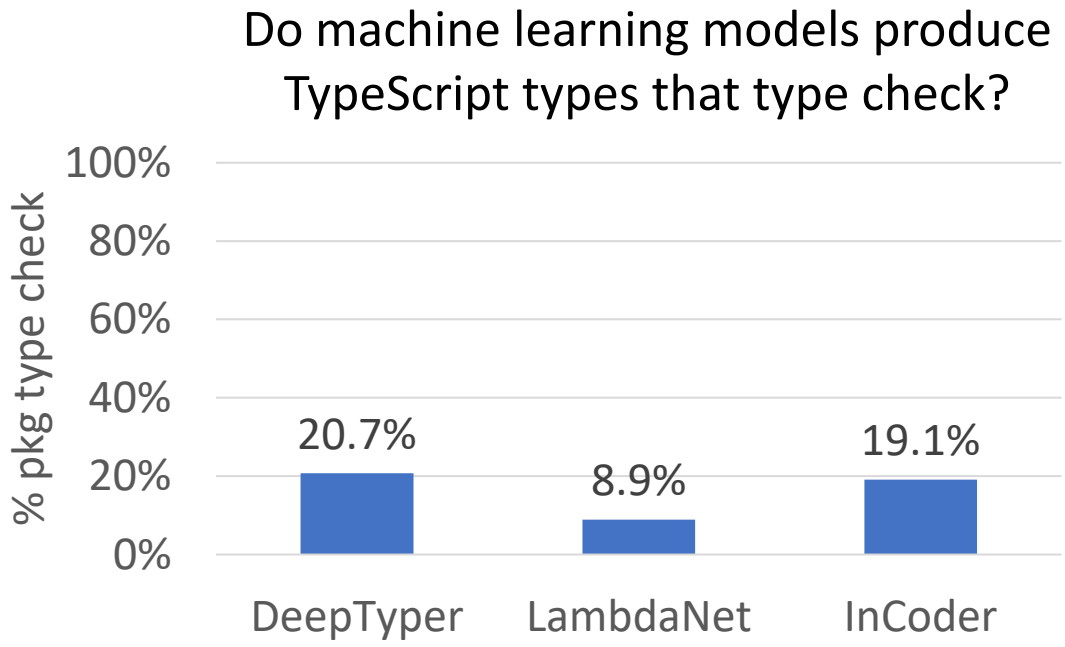


https://github.com/nuprl/TypeWeaver

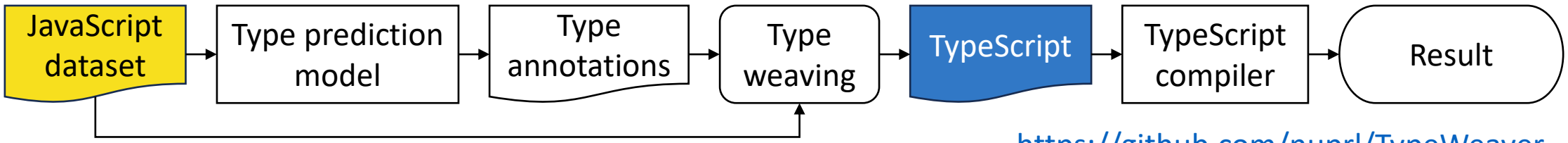
Conclusion: TypeWeaver



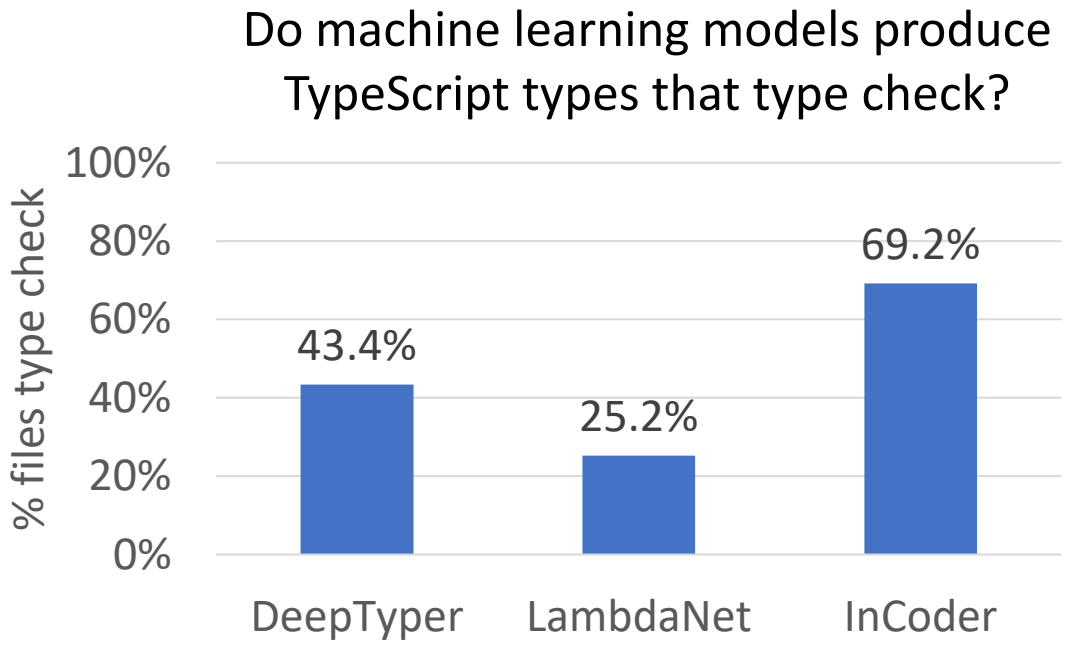
https://github.com/nuprl/TypeWeaver



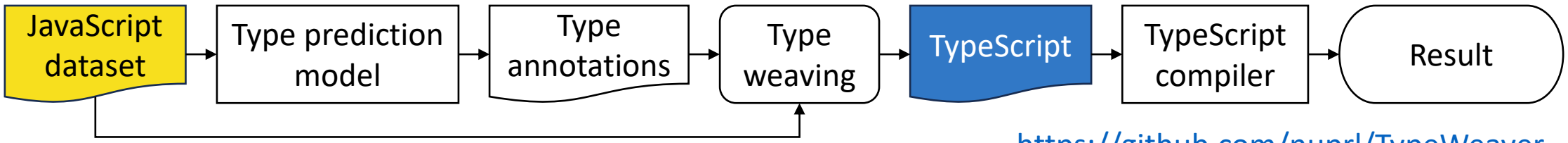
Conclusion: TypeWeaver



https://github.com/nuprl/TypeWeaver

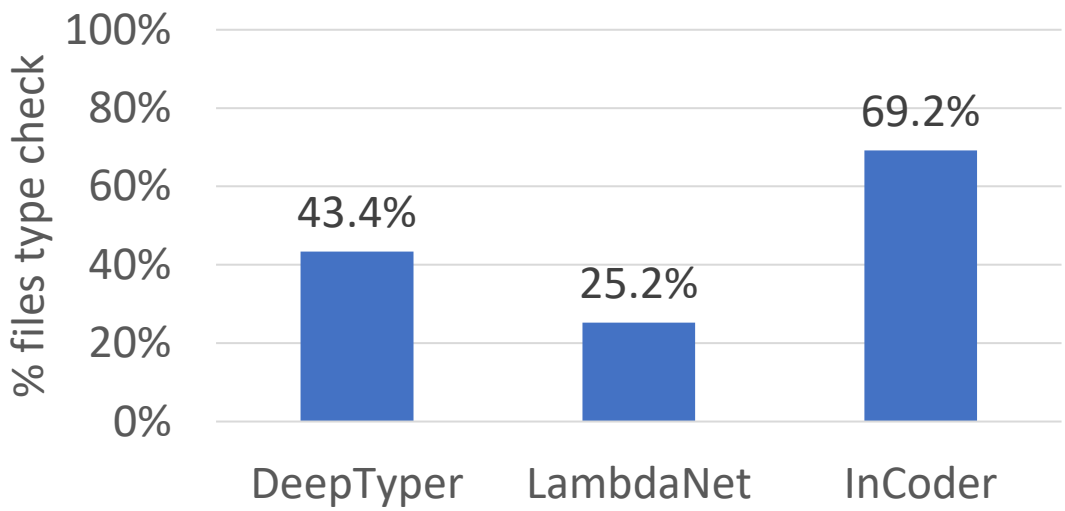


Conclusion: TypeWeaver



https://github.com/nuprl/TypeWeaver

Do machine learning models produce TypeScript types that type check?

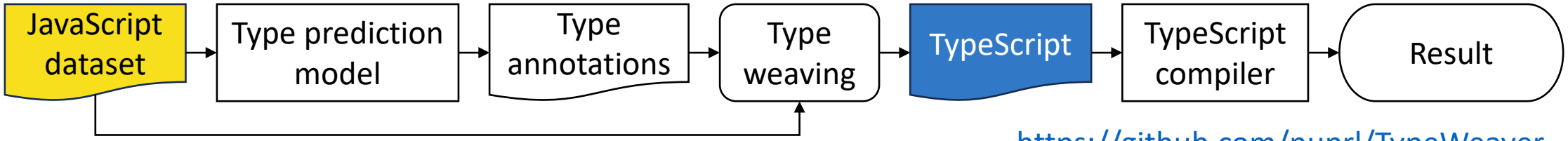


Open questions:

- How should we evaluate type prediction models?
- Can slightly wrong annotations be useful?

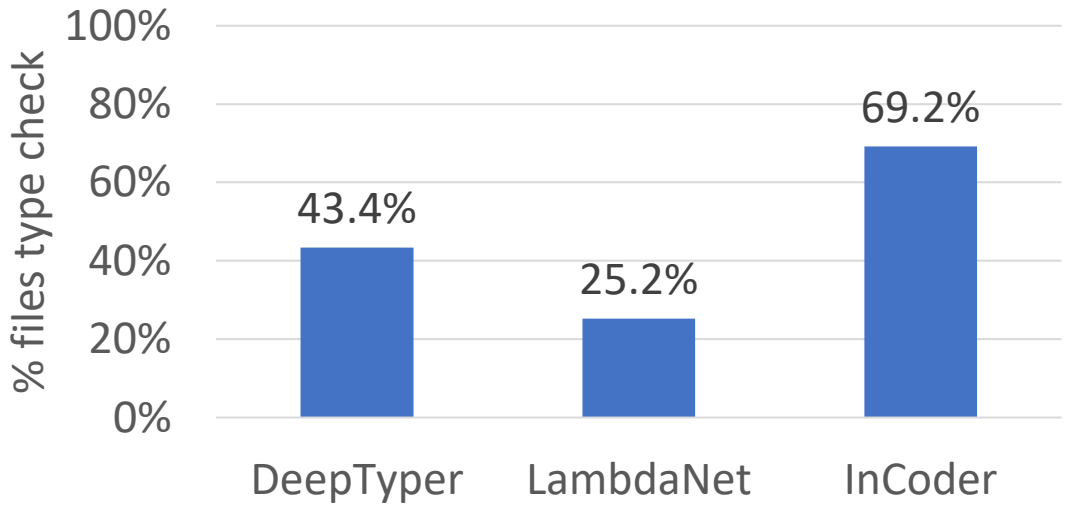
Conclusion: TypeWeaver

Thank you!



https://github.com/nuprl/TypeWeaver

Do machine learning models produce TypeScript types that type check?



Open questions:

- How should we evaluate type prediction models?
- Can slightly wrong annotations be useful?