Combining Type4Py's ML-based Type Inference with Static Type Inference for Python

Anhar Al Haydar
a.alhaydar@student.tudelft.nl
Supervisors: Amir M. Mir and Sebastian Proksch

1. Introduction

- **Dynamically typed languages, like Python, aren't type-checked, which can be problematic**
- **Optional static typing exists, but adding annotations to existing code is tedious**
- **Static type inference can create annotations, but it can be imprecise**
- **ML-based type inference models can be more performant, but have limits in the type annotations they can infer**
- **Type4Py can infer types for Python, but cannot infer user-defined types**

2. Research question

Would the general type prediction performance of Type4Py be improved if combined with static type inference, using the original dataset?

Sub-questions:

1. How does Type4Py perform compared to a static type inference tool, e.g. Pytype?
2. How does Type4Py perform when combined with a static type inference tool, e.g. Pytype?

4. Results

- **For Top-1:**
  - Regarding Exact Match and Base Type Match, hpredict outperforms Type4Py by *5.0%* and *13.8%* for All types, respectively
- **For Top-10:**
  - hpredicts is found to outperform Type4Py in every case
  - Regarding Exact Match on All types, hpredict outperforms Type4Py by *11.0%*
  - Regarding Base Type Match on All types, Type4Py is outperformed by *19.2%*

Top-n prediction	Metrics	%
Top-1	Exact Match All	21.8
	Exact Match Ubiquitous	54.1
	Exact Match Common	5.8
	Exact Match Rare	0.1
	Base Type Match All	24.9
	Base Type Match Common	12.9
	Base Type Match Rare	3.7
Top-10	Exact Match All	32.1
	Exact Match Ubiquitous	73.5
	Exact Match Common	18.0
	Exact Match Rare	0.6
	Base Type Match All	39.9
	Base Type Match Common	37.0
	Base Type Match Rare	9.3

Table 1: Performance evaluation of neural model Type4Py

Top-n prediction	Metrics	%
Top-1	Exact Match All	26.8
	Exact Match Ubiquitous	49.8
	Exact Match Common	18.7
	Exact Match Rare	9.6
	Base Type Match All	38.7
	Base Type Match Common	45.8
	Base Type Match Rare	23.8
Top-10	Exact Match All	43.1
	Exact Match Ubiquitous	84.9
	Exact Match Common	31.3
	Exact Match Rare	10.1
	Base Type Match All	59.1
	Base Type Match Common	67.8
	Base Type Match Rare	29.2

Table 2: Performance evaluation of hpredict

5. Discussion

- **Incremental approach to hpredict not researched**
- **Augmented dataset of ManyTypes4Py not used**
- **Type4Py results differ strongly from prior research**
- **The Evaluation implementation couild contain mistakes**
- **hpredict could outperform Type4Py because of the combination with static type inference**

6. Conclusions

- **hpredict outperforms Type4Py by 11.0% on Exact Match and All types**
- **Type4Py's type prediction can be improved by combining with static type inference**

Future work:

- Research incremental approach's hpedict performance
- Check evaluation's implementation for mistakes
- Research different prediction combination strategies

3. Method

ManyTypes4Py: the used dataset of Python projects

Top-n metric: percentage of times that correct type is among top-*n* predictions

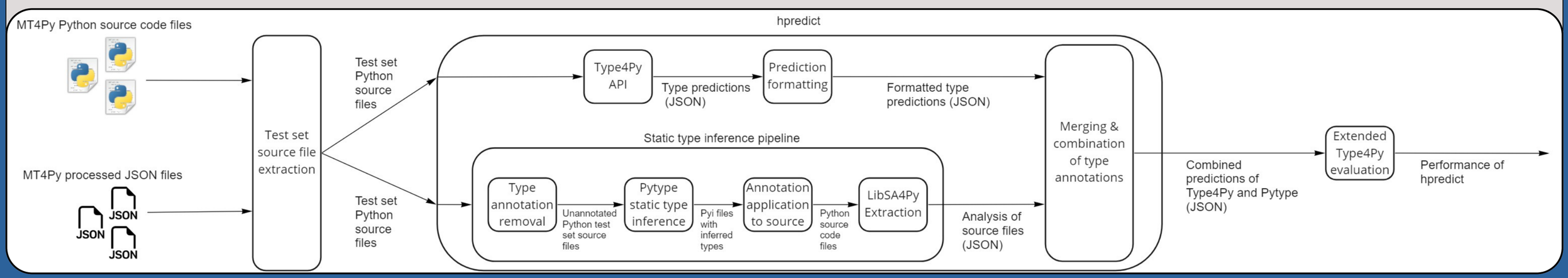

Figure 1: Pipeline overview for research sub-question 2