Author: Merlijn Mac Gillavry (merlijnmac@gmail.com)
Supervisors: Amir Mir, Sebastian Proksch

# Validating Type4Py with MyPy: MyType4Py

TUDelft

## 1. Keywords

- DPL: Dynamically-typed Programming Language
- SPL: Statically-typed Programming Language
- MRR: Mean Reciprocal rank is a statistical measure for evaluating any process that produces a list of responses to a sample of queries
- Type-Correct: Mypy validated (e.g.: MyPy returns no errors when analyzing the function)
- Ground-truth: Type4Py's prediction matches the annotation of developer
- Accuracy: Amount of Type-correct predictions divided by total amount of predictions

## 2. Background

- Python is a DPL, DPL's are generally not checked for type-errors which makes them more prone to run-time type-errors
- Type4Py uses Machine Learning to predict type annotations for Python code. This has been evaluated by using MRR to reflect the performance perceived by users and has achieved an MRR of 77.1% [1]
- Type4Py has, however, not been validated by a static type-checker that checks code for typing errors
- A comparable Type predictor tool Typilus has reached an accuracy of 89% when checked by Mypy

## 3. Research Questions

*How well does Type4Py perform when validated by the static type-checker Mypy?*

1. How many of Type4Py's predictions are type-correct?
2. How many of Type4Py's predictions are type-correct and match ground-truth?

## 4. Methodology

1. Use clean version of ManyTypes4Py dataset
2. Use Type4Py to gather list of predictions for all type-slots
3. Apply P1 strategy with a threshold to greedily take the first prediction that has the highest confidence above that threshold per type-slot (predictable element in source code)
4. Apply predictions to source files
5. Use Mypy to type-check predictions based on line-numbers
6. Evaluate the  results

**ManyTypes4Py dataset (clean):** type-checked subset of 5382 Python projects with more than 869K type annotations. Duplicate source code files were removed to eliminate the negative effect of the duplication bias. [2]

**Type4Py:** Deep similarity learning-based hierarchical neural network model. Used for predicting types in python code.

Type-applier: The type-applier will apply the different type predictions made by Type4Py to the source files. For this libSA4Py was used (a static analysis library for python)

**MyPy:** Mypy is a static type-checker for Python that will check the predictions for type errors

**Evaluation:** Evaluate results of type-checked predictions and calculate accuracy for different match cases

## 5. Results

Using P1 Strategy:
- on a threshold of 0.25, Type4Py's accuracy was 88%
- on a threshold of 0.5, Type4Py's accuracy was 91%
- on a threshold of 0.75, Type4Py's accuracy was 95%

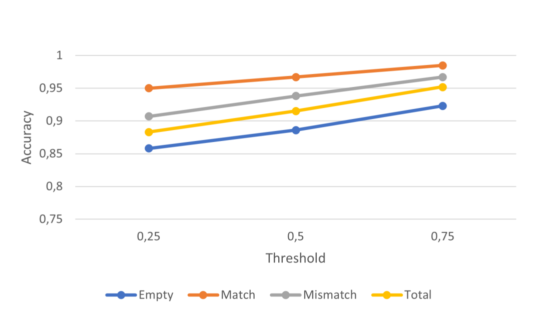
Using P1 Strategy where Type4Py's predictions match ground-truth:
- on a threshold of 0.25, Type4Py's accuracy was 95%
- on oa threshold of 0.5, Type4Py's accuracy was 97%
- on a threshold of 0.75, Type4Py's accuracy was 98%

| Threshold = 0.25 | Proportion (%) | Accuracy (%) | Total Predictions | Type-correct Predictions |
|---|---|---|---|---|
| Empty | 62 | 86 | 1,649,185 | 1,415,684 |
| Match | 15 | 95 | 393,856 | 373,985 |
| Mismatch | 23 | 91 | 602,465 | 546,296 |
| Total | 100 | 88 | 2,645,506 | 2,335,965 |

| Threshold = 0.5 | Proportion (%) | Accuracy (%) | Total Predictions | Type-correct Predictions |
|---|---|---|---|---|
| Empty | 56 | 89 | 982,696 | 870,443 |
| Match | 22 | 97 | 383,297 | 370,506 |
| Mismatch | 22 | 94 | 392,896 | 368,350 |
| Total | 100 | 91 | 1,758,889 | 1,609,299 |

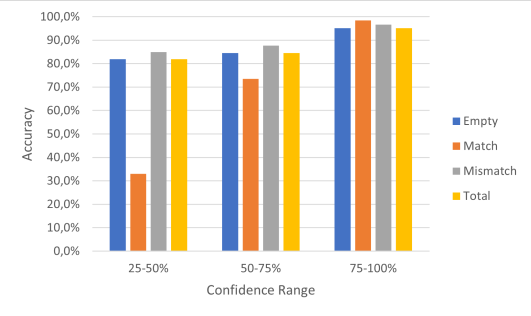
| Threshold = 0.75 | Proportion (%) | Accuracy (%) | Total Predictions | Type-correct Predictions |
|---|---|---|---|---|
| Empty | 46 | 92 | 531,125 | 490,082 |
| Match | 31 | 98 | 355,706 | 350,235 |
| Mismatch | 23 | 97 | 266,337 | 257,430 |
| Total | 100 | 95 | 1,153,168 | 1,097,747 |

Table 1,2 and 3: Type checking accuracy of Type4Py's predictions with the P1 Strategy on the thresholds of 0.25, 0.5 and 0.75 for different match cases.



<- Figure 1: Type checking accuracy of Type4Py's predictions with the P1 Strategy on the thresholds of 0.25, 0.5 and 0.75 for different match cases.

Figure 2: Type checking accuracy of Type4Py's predictions with the P1 Strategy on the confidence ranges of 25-50%, 50-75% and 75-100% for different match cases. ->



## 6. Conclusions

- With a threshold of at most 0.25 Type4Py's predictions reach an accuracy of 88%
- Type4Py's predictions with confidence ranges of 25-50%, 50-75% and 75-100% reach accuracy percentages of 82%, 84% and 95% respectively
- For the case where Type4Py's predictions matched ground-truth, Type4Py's predictions reached accuracy percentages of 33%, 73% and 98% for the confidence ranges of 25-50%, 50-75% and 75-100%
- With a threshold of 0.5, Type4Py's predictions are more accurate than Typilus's predictions [3]

## 7. Discussion

- Accuracy results are highly dependent on the selection of errors that are disabled when using Mypy
- Linking specific predictions to specific errors was programatically hard especially in the time-frame of this research

## 8. Future work

- A combinatoral approach for finding sets of type-correct predictions for a complete file would also be particularly interesting for future research.
- Adding Mypy checking of predictions to Type4Py's VS code extension, would make the extension even more valuable to developers

## References

[1]. A. M. Mir, E. Latoskinas, S. Proksch, and G. Gousios,  "Type4py: Deep similarity learning-based type inference for python," CoRR, vol. abs/2101.04470, 2021.  [Online]. Available: https://arxiv.org/abs/2101.04470

[2] A. M. Mir, E. Latoskinas, and G. Gousios, "Manytypes4py:  A benchmark python dataset for machine learning-based type inference," in 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). IEEE, 2021, pp. 585–589. [Online]. Available: https://ieeexplore.ieee.org/document/9463150

[3]  M. Allamanis, E. T. Barr, S. Ducousso, and Z. Gao, "Typilus: neural type hints," in Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. ACM, jun 2020. [Online]. Available: https://doi.org/10.1145%
2F3385412.3385997