Untersuchung der Domänenabhängigkeit tiefer Lernverfahren für die Typinferenz in Python

Bernd Gruner[1], Tim Sonnekalb[1], Thomas S. Heinze[2] und Clemens-Alexander Brust[1]

[1] DLR-Institut für Datenwissenschaften, Mälzerstraße 3-5, 07747 Jena
{bernd.gruner,tim.sonnekalb,clemens-alexander.brust}@dlr.de
[2] Duale Hochschule Gera-Eisenach, Weg der Freundschaft 4, 07546 Gera
thomas.heinze@dhge.de

Zusammenfassung. In diesem Beitrag stellen wir unsere in [3] veröffentlichten Arbeiten zur Untersuchung der Domänenabhängigkeit von tiefen Lernverfahren für die Typinferenz vor. Wir zeigen am Beispiel Type4Py und Python, dass ein Wechsel zwischen den Anwendungsdomänen Webprogrammierung und wissenschaftliches Rechnen zu einer Verschlechterung in der Vorhersagequalität der Typvorhersage führt und diskutieren verschiedene Ursachen für die Verschlechterung.

1 Einführung

Moderne dynamisch typisierte Programmiersprachen erlauben durch Spracherweiterungen wie *PEP-484* für Python und TypeScript für JavaScript optionale Typannotationen zu nutzen und damit auf die Vorteile der statischen Typisierung zurückzugreifen. In Verbindung mit einer Typinferenz zur automatisierten Ableitung von Typannotationen, ergibt sich ein hilfreiches Werkzeug um Schnittstellen zu dokumentieren, Programmanalysen zu unterstützen und nicht zuletzt typbezogenen Laufzeitfehlern vorzubeugen. Klassische Verfahren der Typinferenz basieren auf statischen oder dynamischen Programmanalysen, jeweils einhergehend mit Problemen: mangelnde Präzision aufgrund der fehlenden Typeinschränkungen in dynamischen Sprachen und der in statischen Analysen angewandten Abstraktionen oder der unvollständigen Abdeckung des Programmverhaltens im Fall dynamischer Analysen. Die Typinferenz auf Grundlage tiefer Lernverfahren bietet einen aktuellen, alternativen Ansatz der vielversprechende Ergebnisse liefert.

Der große Teil der zur Typinferenz mittels tiefer Lernverfahren veröffentlichten Forschungsarbeiten betrachtet dynamische Programmiersprachen. Für die Sprache Python, die auch im Zentrum unseres Beitrags steht, wurden eine Reihe von Verfahren vorgeschlagen, so auch *Type4Py* [5]. In Type4Py wird ein hierarchisches rekurrentes neuronales Netz mit über 600.000 Typannotationen aus ca. 200.000 Python-Dateien öffentlicher Repositorien trainiert, um die charakteristischen Muster von Typannotationen anhand von auftretenden Bezeichnern und Programmstrukturen zu lernen. Das Netz lernt dazu einen Vektorraum, in dem ähnliche Typen zusammengehörige Gruppen bilden und sich die Typannotationen ungesehener Programmbeispiele mittels Ähnlichkeitssuche vorhersagen lassen. Die Autoren von Type4Py konnten zeigen, dass ihr Verfahren im Vergleich mit verwandten Ansätzen bessere Ergebnisse bei der Typvorhersage liefert [5].

Tabelle 1. Datensatz *CrossDomainTypes4Py* (Aufteilung in 70%, 10% und 20% Trainings-, Validierungs- und Evaluationsdaten; seltene Typen mit <100 Vorkommen)

	Webprogrammierung			Wissenschaftliches Rechnen		
Repositorien	3.129			4.783		
Python-Dateien	166.505			470.011		
	Train.	Valid.	Eval.	Train.	Valid.	Eval.
Typannotationen	251.064	27.987	61.978	476.768	56.854	148.732
Typen, darunter	7.588	1.195	8.475	14.973	2.218	14.960
... häufige Typen	232	158	192	363	252	332
... seltene Typen	7.356	1.037	8.283	14.610	1.966	14.628

2 Domänen und Datensatz *CrossDomainTypes4Py*

Die Anwendung und Entwicklung von Lernverfahren, so auch für die Typinferenz, erfolgt im Allgemeinen unter der Annahme, dass sich die statistischen Verteilungen in den Merkmalen der für das Training genutzten Daten und der vorherzusagenden Daten nicht unterscheiden. In der Praxis können aber Probleme bei Verletzung dieser unter dem Begriff *independent and identically distributed (iid)* bekannten Annahme auftreten. In dem vorliegenden Beitrag untersuchen wir die Verteilungen der für die Typinferenz mit Type4Py relevanten Programmmuster unter Berücksichtigung unterschiedlicher Anwendungsdomänen. Im Fokus steht die Frage, inwiefern sich das auf einem Datensatz erlernte Modell auch für Vorhersagen zu Programmen anderer Domäne nutzen lässt. Type4Py wurde als ein aktueller Repräsentant tiefer Lernverfahren für die Typinferenz ausgewählt.

Um diese Frage zu beantworten, wird zunächst ein Datensatz benötigt, der neben einer großen Zahl an Typannotationen auch Informationen zu den zugehörigen Domänen bereitstellt. Vergleichbare Datensätze, wie der ursprünglich zum Training von Type4Py genutzte *ManyTypes4Py*-Datensatz [4], enthalten keine Informationen zu Anwendungsdomänen. Mittels Repository Mining durchsuchen wir zu diesem Zweck systematisch öffentliche Repositorien nach Python-Dateien, die einerseits eine Abhängigkeit zur Programmbibliothek *mypy* aufweisen, da wir in diesem Fall von vorhandenen Typannotationen ausgehen können. Andererseits filtern wir zusätzlich nach Abhängigkeiten zu den Bibliotheken *Flask* und *NumPy*. Diese zwei Bibliotheken sehen wir jeweils für die Domänen Webprogrammierung und wissenschaftliches Rechnen als kennzeichnend. Anschließend werden Duplikate entfernt, dies betrifft zum einen Doppelungen aufgrund von gleichzeitigen Abhängigkeiten zu den oben genannten zwei Bibliotheken und zum anderen auch direkte Dateiduplikate. Weitere Schritte zur Vorverarbeitung, analog [4], dienen der Extraktion relevanter Programmmerkmale (Bezeichner, Symbolsequenzen, usw.) und der Normalisierung von Typannotationen: Entfernen von `Any` und `None`, Auflösen von Typaliasen, Qualifizierung von Typen und Beschränkung der Verschachtelungstiefe generischer Typen. Kennzahlen für den sich ergebenden und öffentlich verfügbaren Datensatz *CrossDomainTypes4Py* [2] mit über einer Million Typannotationen sind in Tabelle 1 angegeben.

Tabelle 2. Experimente zu *Type4Py* mit nach Domänen getrennten Trainings- und Evaluationsdaten (Top-1-Vorhersagegenauigkeit, seltene Typen mit <100 Vorkommen)

Training \ Evaluation	Wissenschaftliches Rechnen (alle Typen)	
	alle Typen	bekannte Typen
Wissenschaftliches Rechnen	62,7	76,9
Webprogrammierung	51,6	69,5
ManyTypes4Py	48,1	66,4

Training \ Evaluation	Wissenschaftliches Rechnen (häufige Typen)	
	alle Typen	bekannte Typen
Wissenschaftliches Rechnen	81,1	81,1
Webprogrammierung	74,0	74,0
ManyTypes4Py	72,1	72,1

Training \ Evaluation	Wissenschaftliches Rechnen (seltene Typen)	
	alle Typen	bekannte Typen
Wissenschaftliches Rechnen	32,0	50,0
Webprogrammierung	13,0	42,8
ManyTypes4Py	8,2	30,6

3 Experimente zur Domänenabhängigkeit

In unseren Experimenten zur Domänenabhängigkeit von tiefen Lernverfahren für die Typinferenz untersuchen wir verschiedene Fragestellungen (vergleiche auch [3]). Zunächst interessiert uns, ob Unterschiede in den statistischen Verteilungen der zur Typinferenz mit Type4Py genutzten Programmmuster zwischen den zwei Anwendungsdomänen Webprogrammierung und wissenschaftliches Rechnen existieren. Tatsächlich fällt bei Betrachtung der zehn häufigsten Typen unseres Datensatzes CrossDomainTypes4Py auf, dass beispielsweise die Typen `str`, `Optional[str]` und `dict` vermehrt in Programmen der Anwendungsdomäne Webprogrammierung vorkommen, wohingegen `int`, `float` und `numpy.ndarray` häufiger in Programmen für das wissenschaftliche Rechnens auftreten. Gleichzeitig kann beobachtet werden, dass beide Domänen lediglich 3.755 Typen gemeinsam haben, bei insgesamt 15.177 beziehungsweise 27.611 vorkommenden Typen. Der Grund hierfür ist in der schiefen Verteilung der Typen zu suchen, mit vielen weniger häufig auftretenden Typen, die oft projekt- oder domänenspezifisch sind.

Um die Auswirkung dieser Verteilungsunterschiede auf die Typinferenz mit Type4Py zu beurteilen, werden unterschiedliche Modelle trainiert und evaluiert, wobei sich die für Training und Evaluation der Modelle jeweils genutzten Programmbeispiele hinsichtlich der Anwendungsdomänen aus denen die Beispiele stammen unterscheiden. In Tabelle 2 ist die Top-1-Vorhersagegenauigkeit, das heißt das Verhältnis von Typannotationen für die eine übereinstimmende Typvorhersage erzielt werden konnte bezüglich aller zur Evaluation genutzten Typannotationen, für drei verschiedene Modelle angegeben. Für das Anlernen

des ersten Modells werden Programmbeispiele aus der Anwendungsdomäne wissenschaftliches Rechnen verwendet, das zweite Modell nutzt dafür Programmbeispiele der Webprogrammierung und das dritte Modell beruht schließlich auf dem ursprünglich zum Training von Type4Py verwendeten ManyTypes4Py-Datensatz. All drei Modelle nutzen als Evaluationsdaten Programmbeispiele aus der Domäne wissenschaftliches Rechnen. Beim Vergleich der angegebenen Top-1-Vorhersagegenauigkeiten über alle Typen hinweg fällt auf, dass sich die Genauigkeit bei einem Wechsel der Domäne zwischen Training und Evaluation um bis zu 14% verringert. Bemerkenswerterweise trifft dies auch für das mit dem ManyTypes4Py-Datensatz angelernte Modell zu. Auch wenn nur die Typannotationen häufiger Typen betrachtet werden, also von Typen die mindestens 100 Mal vorkommen, verbessert sich zwar die Vorhersagegenauigkeit spürbar, eine Verringerung der Genauigkeit beim Wechsel der Domäne für Training und Evaluation kann aber weiterhin beobachtet werden. Tiefe Lernverfahren haben im Allgemeinen Probleme bei der Vorhersage unbekannter Typen, das heißt von Typen die nicht in den Trainingsdaten vorkommen [5]. Werden unbekannte Typen bei der Evaluation vernachlässigt, ergibt sich zwar wieder eine erhebliche Verbesserung der Vorhersagegenauigkeit. Der Unterschied hinsichtlich der unterschiedlichen Domänen lässt sich aber dadurch wieder nicht vollständig erklären.

Um die beobachtete Verringerung der Vorhersagegenauigkeit aufgrund eines Wechsels der Anwendungsdomäne für tiefe Lernverfahren wie Type4Py einzuschränken, kann auf eine nachträgliche Feinabstimmung der Modelle oder auf eine Erweiterung der Type4Py zugrundeliegenden Ähnlichkeitssuche (*TIPICAL* [1]) zurückgegriffen werden, wie wir in weiteren Experimenten zeigen konnten [3].

Literatur

[1] Elkobi, Jonathan ; Gruner, Bernd ; Sonnekalb, Tim ; Brust, Clemens-Alexander: TIPICAL - Type Inference for Python In Critical Accuracy Level. In: *21st IEEE/ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2023, Orlando, FL, USA, May 23-25, 2023*, IEEE, 2023, 16–21

[2] Gruner, Bernd ; Heinze, Thomas S. ; Brust, Clemens-Alexander: *CrossDomainTypes4Py: A Python Dataset for Cross- Domain Evaluation of Type Inference Systems*. http://dx.doi.org/10.5281/zenodo.5747024. Version: Januar 2022

[3] Gruner, Bernd ; Sonnekalb, Tim ; Heinze, Thomas S. ; Brust, Clemens-Alexander: Cross-Domain Evaluation of a Deep Learning-Based Type Inference System. In: *20th IEEE/ACM International Conference on Mining Software Repositories, MSR 2023, Melbourne, Australia, May 15-16, 2023*, IEEE, 2023, 158–169

[4] Mir, Amir M. ; Latoskinas, Evaldas ; Gousios, Georgios: ManyTypes4Py: A Benchmark Python Dataset for Machine Learning-based Type Inference. In: *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021, Madrid, Spain, May 17-19, 2021*, IEEE, 2021, 585–589

[5] Mir, Amir M. ; Latoskinas, Evaldas ; Proksch, Sebastian ; Gousios, Georgios: Type4Py: Practical Deep Similarity Learning-Based Type Inference for Python. In: *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, ACM, 2022, 2241–2252