Wieviel Ähnlichkeit ist in Programmierprüfungen normal?

Christoph Olbricht [1], Rafael Schypula[2] und Michael Striewe [3]

Abstract: Werkzeuge zur Identifikation von Plagiaten in Programmieraufgabe messen häufig die Ähnlichkeit zwischen Abgaben. Die Interpretation dieser Werte zur Identifikation von Verdachtsmomenten ist jedoch schwierig, da eine gewisse Ähnlichkeit von Abgaben unvermeidlich ist. Es ist insbesondere unklar, welche Ähnlichkeitsverteilung innerhalb einer Kohorte ohne Plagiate als normal angesehen werden kann und welche Abweichungen davon als Verdachtsmoment genutzt werden können. Der vorliegende Beitrag vergleicht dazu unter Nutzung zweier Werkzeuge zur Plagiatserkennung unter Aufsicht erstellte Abgaben zu Prüfungsaufgaben mit solchen, die ohne Aufsicht entstanden sind. Die Ergebnisse zeigen, dass die Ermittlung einer „Baseline“ für die erwartbare Ähnlichkeit schwierig ist, da auch schon kleine Änderungen der Aufgabenstellung starke Auswirkungen auf die Ähnlichkeit der Lösungen haben können.

Keywords: Programmieraufgaben, Plagiate, Programmähnlichkeit, Täuschungsversuche, JPlag, Sherlock

1 Einleitung

Plagiate bei der Lösung von Programmieraufgaben sind ein lange bekanntes Problem in der Lehre, zu dessen Bewältigung zahlreiche Werkzeuge zur Erkennung von Plagiaten entwickelt wurden. Diese Werkzeuge erkennen Plagiate nicht direkt, sondern messen die Ähnlichkeit zweier Lösungen. Es bleibt letztlich Aufgabe der Lehrenden zu entscheiden, ob es sich dabei um ein Plagiat handelt [NJK19]. Ein Stolperstein ist dabei, dass Programmierung ein strukturiertes Vorgehen bei der Entwicklung einer Lösung erfordert. Folgen Studierende unabhängig voneinander demselben – beispielsweise in der Lehre vermittelten – Vorgehen, so weisen auch ihre Lösungen mit einer gewissen Wahrscheinlichkeit eine hohe Ähnlichkeit auf. Bei der Verfolgung eines Einzelfalls muss daher immer mit einer rein zufälligen Ähnlichkeit gerechnet werden. Treten jedoch in einer Menge von Lösungen deutlich mehr Fälle von hoher oder sehr hoher Ähnlichkeit auf, als im Schnitt zu erwarten sind, ist dies ein Indikator dafür, dass nicht alle diese Fälle rein zufällig entstanden sind. Unabhängig von der verwendeten Methode eines Werkzeugs zur Plagiatserkennung und dessen Trefferquote beim Aufspüren ähnlicher Lösungen ist es daher für

[1] Universität Duisburg-Essen, paluno – The Ruhr Institute for Software Technology, Gerlingstraße 16, 45127 Essen, christoph.olbricht@paluno.uni-due.de, https://orcid.org/ 0009-0002-4823-7894
[2] Universität Duisburg-Essen, paluno – The Ruhr Institute for Software Technology, Gerlingstraße 16, 45127 Essen, rafael.schypula@paluno.uni-due.de
[3] Universität Duisburg-Essen, paluno – The Ruhr Institute for Software Technology, Gerlingstraße 16, 45127 Essen, michael.striewe@paluno.uni-due.de, https://orcid.org/0000-0001-8866-6971

doi:10.18420/abp2023-4

Lehrende wichtig zu wissen, mit welcher Wahrscheinlichkeit sie in einer Menge von Lösungen mit Fällen von hoher oder sehr hoher Übereinstimmung rechnen müssen.

Der vorliegende Beitrag vergleicht dazu konkret die Ergebnisse von zwei Werkzeugen zur Plagiatserkennung auf Lösungen von Aufgaben, die unter Aufsicht bearbeitet wurden und die somit keine Plagiate enthaltenen können, mit Ergebnissen auf Lösungen vergleichbarer Aufgaben, die nicht unter Aufsicht erstellt wurden und die somit Plagiate enthalten können. Damit soll herausgefunden werden, ob es eine nachweisbare Änderung in der Ähnlichkeitsverteilung gibt, die als Verdachtsmoment verwendet werden kann.

1.1 Verwandte Arbeiten

Strickroth [St21] vergleicht Werkzeuge zur Plagiatserkennung und verwendet dazu reale Daten, bei denen teilweise die Zahl der enthaltenen Plagiate bekannt ist. Die Untersuchung fokussiert die Übereinstimmung zwischen den verglichenen Werkzeugen und die Rate bei der Erkennung der Plagiate. Sie liefert keine Aussagen darüber, welche Ähnlichkeitswerte in Lösungen normal sind und bezeichnet den verwendeten Grenzwert von 80 % Ähnlichkeit als willkürlich. Ähnliches gilt für eine Studie von Modiba et al. [MPH16], in der die Ergebnisse von Werkzeugen zur Plagiatserkennung auf realen Daten mit einer manuellen Überprüfung auf Plagiate verglichen werden. Die Untersuchung legt den Fokus ebenfalls auf die Rate der Erkennung der Plagiate und liefert keine Erkenntnisse darüber, welche Ähnlichkeitswerte als (un)verdächtig angesehen werden können.

Pang und Vahid [PV23] gehen den umgekehrten Weg und nehmen die hohe Rate an vermeintlich nahezu identischen Lösungen zum Anlass, Programmieraufgaben gezielt so zu erweitern, dass die zu erwartende „natürliche“ Ähnlichkeit reduziert wird. Die Arbeit liefert damit zumindest einige Datenpunkte, welcher Grad an Ähnlichkeit in Abhängigkeit von der Komplexität der Aufgabe bzw. dem Umfang der Lösungen erwartet werden kann. In eine ähnliche Richtung geht eine Untersuchung von Haan und Striewe [HS21], die unter anderem den Einfluss der durchschnittlichen Länge einer Lösung auf die Ähnlichkeit untersucht, dabei jedoch zu uneindeutigen Ergebnissen kommt.

2 Datengrundlage und Methodik

Die verwendeten Daten stammen aus den Klausuren einer Programmierungsvorlesung in Java für Erstsemester. Hierbei wurden jeweils Haupt- und Nachtermin der Wintersemester 2020, 2021, 2022 untersucht. Im Jahr 2020 wurden bedingt durch die Corona-Pandemie die Klausuren zulassungsfrei, online und ohne Aufsicht als Freiversuch zu Hause geschrieben. In den Jahren 2021 und 2022 fanden die Klausuren in den Universitätsräumen unter Aufsicht statt und benötigten eine Zulassung, die durch vorherige Testate erworben wurde. Um Plagiaten bei der Onlineklausur im Jahr 2020 vorzubeugen, wurden vier unterschiedliche Versionen der Klausur erstellt. Alle Klausuren wurden mithilfe der Entwicklungsumgebung Eclipse durchgeführt. Die Studierenden erhielten Vorlagen in Form von Java-Quellcode-Dateien. Dieser Quellcode besaß Hilfsmethoden und zu

verwendende Objekte. Die Methodensignaturen der zu bearbeitenden Methoden waren vorgegeben. Die Lösungen wurden mittels eines E-Assessment-Systems eingereicht. Es gab für die Studierenden während der Klausur keinerlei Feedback zu ihren Einreichungen.

2.1 Auswahl der Aufgaben

Um einen sinnvollen Vergleich zu ermöglichen, wurden für dieses Paper thematisch ähnliche Aufgaben mit ausreichend großem Programmieranteil und Bearbeitungen ausgewählt. Zwar gab es in jeder Klausur eine Aufgabe zu Lambda-Ausdrücken, jedoch war diese mit fünf Zeilen Code zu lösen, welche für eine korrekte Lösung einen sehr eingeschränkten Lösungsraum aufwiesen. Bei einer Modellierungsaufgabe mussten lediglich Klassen mit Attributen und Getter-/Setter-Methoden mit vorgegebenen Bezeichnern erzeugt werden, was für die Plagiatsprüfung unbrauchbar war.

Alle Klausuren enthielten eine Aufgabe zu Arrays. Eine Aufgabe mit verketteten Listen war in allen Klausuren bis auf den Haupttermin 2022 vorhanden. In beiden Fällen war ein ausreichend großer Anteil von selbst verfasstem Quellcode gegeben, um einen Vergleich durchzuführen. Die Menge an studentischen Lösungen wurde um Leerabgaben bereinigt. Die sich insgesamt ergebenden Zahlen an verwertbaren Abgaben sind in Tab. 1 ersichtlich.

Tabelle 1: Anzahl der Abgaben je Aufgabe und Klausur. Aufgabenvarianten (nur im Jahr 2020 eingesetzt) sind zusammengefasst.

Abgaben	2020 HT	2020 NT	2021 HT	2021 NT	2022 HT	2022 NT
Array	321	175	165	93	173	69
Liste	196	125	172	93	---	69

2.2 Auswahl der Werkzeuge

Von den frei verfügbaren Softwarewerkzeuge zur Plagiatserkennung für Java-Quellcode wurde JPlag[4] aufgrund seiner großen Verbreitung und guten Erkennungsquote [WW12] ausgewählt. Zur Analyse des Quellcodes verwendet JPlag eine auf Token basierende Technik [PM02]. Plaggie[5] kann nur Java-Quellcode bis zur Version 1.5 verarbeiten und wurde daher ausgeschlossen. SIM[6] ließ sich auf den aktuellen Systemen nicht mit vertretbarem Aufwand installieren. MOSS[7] musste ausgeschlossen werden, da die Daten auf fremde Server hochzuladen waren. Sherlock[8] verwendet neben textbasierten auch eine auf Token basierende Technik zur Quellcodeanalyse [JL99] und wurde daher als weiteres Werkzeug verwendet. Beide Tools unterstützen die Verwendung von Vorlagen, so dass nur der Quellcode untersucht wird, welcher nicht in der Vorlage enthalten ist.

[4] https://github.com/jplag/JPlag
[5] https://www.cs.hut.fi/Software/Plaggie
[6] https://dickgrune.com/Programs/similarity_tester
[7] https://theory.stanford.edu/~aiken/moss/
[8] https://warwick.ac.uk/fac/sci/dcs/research/ias/software/sherlock/

2.3 Verwendete Methode

Zunächst wurden alle Einreichungen anonymisiert und für das Jahr 2020 nach Varianten getrennt, um den unterschiedlichen Bezeichnern der Varianten Rechnung zu tragen. Alle Codevorlagen wurden in den Tools hinterlegt. Jede Variante und Klausur wurde mit identischen Einstellungen der Tools verarbeitet. In JPlag wurde die vorausgewählte Methode der Durchschnittsberechnung und die Sensitivität von 9 verwendet. In Sherlock wurde die empfohlene Kombination aus „Original“, „No Comments and Normalised“ und „Tokenised“ verwendet. Das Resultat beider Tools war eine Menge an Paaren. Für jedes Paar gibt eine Kennzahl die Ähnlichkeit beider Einreichungen zueinander an. Sherlock gibt hierbei nur Paare mit einer Kennzahl über 0 aus, während JPlag alle Paare ausgibt.

3 Ergebnisse

Aufgrund der Technik des Paarvergleichs ergibt sich eine hohe Anzahl an Werten, die zudem aufgrund der unterschiedlichen Techniken nicht unmittelbar zwischen JPlag und Sherlock vergleichbar sind. Zur besseren Übersicht wurden die gemeldeten Paare anhand ihrer Werte in jeweils zehn Gruppen zusammengefasst. Damit die Ähnlichkeitsverteilung zwischen den Klausuren trotz der unterschiedlich hohen Zahl an Abgaben verglichen werden kann, wird jeweils das prozentuale Verhältnis zwischen der Anzahl der Paare je Gruppe und der Gesamtzahl der gefundenen Paare ermittelt und im Diagramm angegeben.

3.1 JPlag

In Tab. 2 und Abb. 1 ist die Ähnlichkeitsverteilung von JPlag für die Arrayaufgaben ersichtlich. Für die beiden Onlineklausuren im Jahr 2020 konnten nur die Einreichungen innerhalb einer Klausurvariante verglichen werden; dadurch ergibt sich eine geringere Paaranzahl. Einige Einreichungen konnten aufgrund von Syntaxfehler oder zu wenig Quellcode nicht von JPlag verarbeitet werden, daher gibt die zweite Spalte in Tab. 2 und Tab. 3 die Anzahl der von JPlag verarbeiteten Einreichungen an. Mit Ausnahme der Klausuren 2021 NT und 2022 HT weist der größte Teil der Abgaben kaum Ähnlichkeiten auf. Zwischen 20 und 50 % ergibt sich eine kleine Häufung. Im höchsten Bereich der Ähnlichkeit sind deutlich weniger Verdachtsfälle.

Die Ähnlichkeitsverteilung in der Klausur 2022 HT weicht von den anderen deutlich ab. Dies hängt mit den Aufgabenstellungen zusammen. Eine von drei Teilaufgaben bestand darin, drei mit unterschiedlich Werten gefüllte Arrays zu erstellen, um eine selbst erstellte Methode damit auf Korrektheit zu testen. Aufgrund der Token-Technik von JPlag werden unterschiedliche Zahlenwerte im Array als identisch erkannt. Auch die zweite Teilaufgabe erhöhte die Ähnlichkeitsquote, da sie die Implementierung eines ausführlich in Vorlesung und Übung behandelten Algorithmus erforderte.

Die Arrayaufgabe der Klausur 2021 NT weist als einzige einen zweistelligen relativen Ähnlichkeitswert von 13 % bzw. 560 Paaren im Bereich 90 bis 100 % auf. Die

Aufgabenstellung war, aus einem gegebenen eindimensionalen Array alle Strings, welche kürzer als ein übergegebener Parameter sind, in einem neuen Array zurückzugeben. Diese Aufgabe wurde zuvor nicht behandelt. Auch bei manueller Überprüfung ergibt sich eine sehr hohe Ähnlichkeit. Hier muss die Aufgabenstellung derart einengend sein, dass die Studierenden von selbst auf die gleiche Anweisungsabfolge gekommen sind, welche zugleich korrekt und minimal ist.

Bei den Aufgaben zum Thema verkettete Liste zeigt Tab. 3 deutlich geringere Ähnlichkeiten, so dass hier noch mehr Paarvergleiche eine geringere Ähnlichkeit als 10 % erreichen. Auch bei den Listenaufgaben ergibt sich eine zweite kleine Häufung zwischen 10 und 40 % welche bei 20 % den höchsten Wert erreicht.

Die Einreichungen zu Listen haben ähnlich viele Quellcodezeilen wie zu Arrays, somit kann dies nicht ausschlaggebend für die geringeren Ähnlichkeiten sein. Arrays lassen jedoch durch die spezifische Syntax möglicherweise weniger Lösungsvarianten zu.

Tabelle 2: JPlag Ähnlichkeitsverteilung mit Anzahl der Paare bei den Arrayaufgaben

Aufgabe Array	Einreich-ungen	Anzahl Paare	Ähnlichkeit in Prozentpunkten									
			0-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	90-100
2020 HT	300	11367	9461	0	37	235	630	539	223	47	25	170
2020 NT	138	2362	1963	5	67	113	124	62	17	4	3	4
2021 HT	154	11781	5209	288	2173	1849	1015	791	268	118	57	13
2021 NT	93	4278	2627	0	6	325	232	196	63	125	144	560
2022 HT	165	13530	1316	1160	1877	2759	1683	1993	1390	719	400	233
2022 NT	67	2211	1156	126	293	165	197	94	81	59	26	14

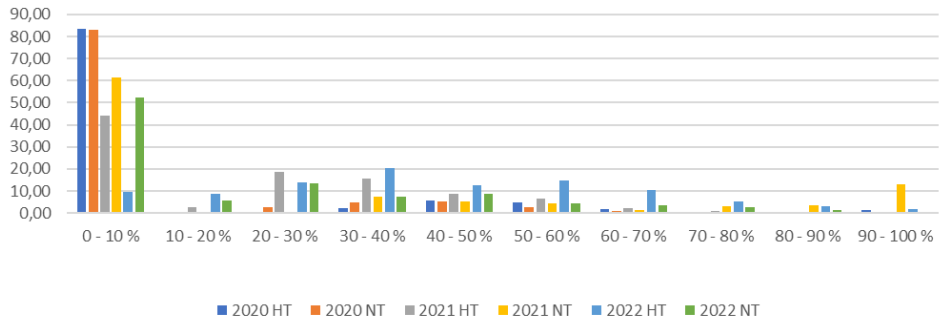

Abbildung 1: JPlag Ähnlichkeitsverteilung in Prozent bei den Arrayaufgaben

Tabelle 3: JPlag Ähnlichkeitsverteilung mit Anzahl der Paare bei den Listenaufgaben

Aufgabe Liste	Einreich-ungen	Anzahl Paare	Ähnlichkeit in Prozentpunkten									
			0-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	90-100
2020 HT	191	4050	3951	0	3	10	17	15	13	14	11	16
2020 NT	78	762	678	1	25	41	8	5	3	0	1	0
2021 HT	166	13695	8172	1050	2552	1163	471	180	72	27	7	1
2021 NT	92	4186	1781	507	1069	431	271	95	23	9	0	0
2022 NT	67	2211	2071	40	68	23	2	2	3	2	0	0

3.2 Sherlock

Da Sherlock keine Paare mit 0 Ähnlichkeit ausgibt und Werte unter 10 äußerst selten sind, wurden alle Fälle mit einer Kennzahl kleiner 20 zusammengefasst. Die Kennzahl ergibt sich aus der Summe der einzelnen verwendeten Methoden. Codezeilen, welche in der Vorlage vorhanden sind, werden nicht als identisch gewertet. Die Kennzahl einer einzelnen Methode ist die Menge an identischen Zeilen im Vergleich zur Gesamtmenge der Zeilen. Diese Gesamtmenge wird nicht um die Zeilen der Vorlage reduziert, womit sich bei größeren Vorlagen geringere Maximalwerte der Methoden ergeben. Somit lassen sich verschiedene Aufgaben nur bedingt mit absoluten Kennzahlen vergleichen. Das Augenmerk sollte auf der Verteilung im gegebenen Spektrum der einzelnen Klausur liegen. Tab. 4 und Abb. 2 zeigen die Menge an Verdachtsfällen und die Ähnlichkeits-verteilung von Sherlock für die Arrayaufgaben.

Bis auf 2020 HT ist bei allen Klausuren ein steter Abfall der Menge an Fällen ab 30 zu erkennen. Die größere Menge an Fällen im 30er Bereich der Klausur 2020 HT kommt zustande, da am Ende der Aufgabe ein Array mittels einer Schleife befüllt und zurückgegeben werden musste, was keine variablen Lösungen zulässt. Somit fallen korrekte Lösungen in diesen Bereich. Die wesentlich größere Gesamtmenge an Fällen in der Klausur 2022 HT wurde bereits in Abschnitt 3.1 erläutert. In der Klausur 2021 HT war die Methodensignatur in der Aufgabenstellung vorgegeben, allerdings nicht in der Vorlage vorhanden, weshalb Sherlock eine große Menge an Fällen im Bereich bis 30 aufweist. Die vermehrten Fälle im Bereich 70+ dieser Klausur wurden bereits in Abschnitt 3.1 dargelegt. In 2021 NT befand sich eine wesentlich größere Vorlage, als in allen anderen Klausuren, weswegen hier insgesamt niedrigere Werte erzielt wurden.

Tabelle 4: Sherlock Anzahl der Verdachtsfälle bei den Arrayaufgaben

Aufgabe Array	Einreich-ungen	Anzahl Fälle	Ähnlichkeit in Punkten									
			< 20	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99	100+
2020 HT	321	2522	40	659	1251	443	104	10	6	5	2	2
2020 NT	175	1759	237	658	402	235	128	63	24	8	2	2
2021 HT	165	6839	1477	2465	1356	711	405	217	136	45	22	5
2021 NT	93	1653	482	676	402	42	34	10	6	1	0	0
2022 HT	173	9380	4561	2040	1485	742	323	131	54	28	10	6
2022 NT	69	1350	292	444	280	132	105	50	26	13	6	2

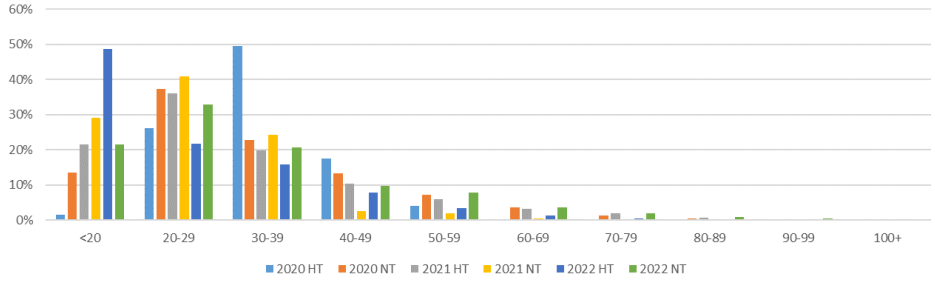

Abbildung 2: Sherlock Ähnlichkeitsverteilung in Prozent bei den Arrayaufgaben

Tabelle 5: Sherlock Anzahl der Verdachtsfälle bei den Listenaufgaben

Aufgabe Liste	Einreich-ungen	Anzahl Fälle	Ähnlichkeit in Punkten									
			< 20	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99	100+
2020 HT	196	2	1	0	1	0	0	0	0	0	0	0
2020 NT	125	15	14	0	0	0	0	1	0	0	0	0
2021 HT	172	884	846	36	2	0	0	0	0	0	0	0
2021 NT	93	225	221	3	1	0	0	0	0	0	0	0
2022 NT	69	4	4	0	0	0	0	0	0	0	0	0

Bei den Aufgaben zum Thema verkettete Liste zeigt Tab. 5, dass Sherlocks Umgang mit Vorlagen bei Listen zu Schwierigkeiten in der Erkennung von Verdachtsfällen führt. In 2020 HT musste gegebener Code korrigiert werden, womit die Zeilen ignoriert wurden. 2020 NT besaß sehr viel vorgegebenen Code und nur eine kleine Methode, die zu implementieren war. In 2021 HT und NT gab es eine schlankere Vorlage als in den anderen Jahren und die Studierenden mussten unter anderem eine printList()-Methode implementieren, deren eingeschränkter Lösungsraum in der großen Menge an Verdachtsfällen unter 20 resultiert.

4 Diskussion

JPlag und Sherlock weisen bei Listen dieselben Ergebnisse auf. Die Menge an Verdachtsfällen ist bei den drei Klausuren 2020 HT & NT und 2022 NT äußerst gering, während bei den Klausuren von 2021 größere Mengen mit leichten Verdachtsfällen bestehen. Sowohl bei den Array- wie Listenaufgaben, liegt die große Mehrheit aller Paare insgesamt im Bereich 0–10 % Ähnlichkeit. In Anbetracht der kurzen Lösungen von ca. 10 bis 20 Zeilen mit vorgegebenen Methodensignaturen und daraus resultierendem geringen Varianzbereich der möglichen Programabläufe ist dies beachtlich und deutet darauf hin, dass eine überdurchschnittliche Häufung hoher Ähnlichkeiten in der Tat ein geeignetes Verdachtsmoment darstellt. Allerding wurde in der Arrayaufgabe der Klausur 2021 NT bei JPlag eine solche Häufung im 90–100 % Bereich gefunden. Da diese Klausur unter Aufsicht stattfand zeigt dies, dass auch ein solch hoher Wert nicht zwangsläufig auf Plagiate zurückzuführen ist. Vielmehr lernen die Studierenden im Semester einen bestimmten Stil oder eine Denkweise, um die behandelten Aufgaben zu lösen. Dies schlägt sich auch unter Aufsicht in sehr ähnlichen Einreichungen nieder.

Da die Klausuren aus 2020 in vier Varianten aufgeteilt waren, ergeben sich weniger Paare. Allerdings waren aufgrund der Zahl der Studierenden (selbst ohne die Erwartung von Betrugsversuchen aufgrund der fehlenden Aufsicht) in etwa doppelt so viele Verdachtsfälle wie gefunden wurden zu erwarten. Ein Grund für die Abweichung kann der zulassungsfreie Freiversuch sein, der dafür sorgte, dass sich viele Studierende wesentlich schlechter auf die Klausur vorbereitet hatten. Viele Einreichungen erfüllten die Aufgabenstellung nicht einmal ansatzweise und wiesen weder Ähnlichkeiten zu korrekten Lösungen noch zu anderen Fehlversuchen auf. Die Arrayaufgabe von 2020 NT ist zudem wesentlich freier und komplexer als andere gestellte Aufgaben, was zu mehr Individuallösungen führte. In solchen Fällen ist es möglicherweise zielführender, nach charakteristischen,

ungewöhnlichen Fehlern oder stilistischen Eigenheiten zu suchen, die in der Normalisierung durch die Werkzeuge untergehen.

5 Fazit und Ausblick

Die Ergebnisse zeigen, dass die Ähnlichkeitsverteilung maßgeblich von der Aufgabenstellung abhängt. Aus den variierenden Verteilungen können keine allgemeingültigen Schlüsse gezogen werden. Eine besondere Auffälligkeit im Vergleich der überwachten Präsenzklausuren zu den nicht überwachten Prüfungen konnte nicht festgestellt werden.

Sherlock und JPlag können als Tools zur Plagiatserkennung lediglich einen ersten Verdacht liefern, dem manuell nachgegangen werden muss. Beide Tools benötigen spezifische Rahmenbedingungen, um eindeutige Ergebnisse liefern zu können, welche in den untersuchten Daten nur zum Teil vorlagen. Entsprechend fällt die Definition schwer, ab wann eine Einreichung ähnlich zu einer anderen Einreichung ist. Ob beide Tools mit anderen Einstellungen und Rahmenbedingungen vergleichbarer werden, kann an dieser Stelle nicht beantwortet und muss in weiteren Arbeiten untersucht werden.

Literaturverzeichnis

[PV23] Pang, A.; Vahid, F: Variability-Inducing Requirements for Programs: Increasing Solution Variability for Similarity Checking. In: Proc. 28th annual ACM conference on Innovation and Technology in Computer Science Education (ITiCSE), 2023.

[HS21] Haan, T.; Striewe, M.: On the Influence of Task Size and Template Provision on Solution Similarity. In: Proc. 5. Workshop Automatische Bewertung von Programmieraufgaben (ABP), S. 51–58, 2021.

[St21] Strickroth, S.: Plagiarism Detection Approaches for Simple Introductory Programming Assignments. In: Proc. 5. Workshop Automatische Bewertung von Programmieraufgaben (ABP), S. 43–50, 2021.

[MPH16] Modiba, P.; Pieterse, V.; Haskins, B.: Evaluating plagiarism detection software for introductory programming assignments. In: Proc. CSERC ’16. ACM, 2016.

[NJK19] Novak, M.; Joy, M.; Kermek, D.: Source-code Similarity Detection and Detection Tools Used in Academia. TOCE 19/3, S. 1–37, 2019

[WW12] Weber-Wulff, D.: Collusion Detection System Test Report 2012. URL: https://plagiat.htw-berlin.de/collusion-test-2012/ (besucht am 07.06.2023)

[PM02] Prechelt, L.; Malpohl, G.: Finding Plagiarisms among a Set of Programs with JPlag. Journal of Universal Computer Science. 8, S. 1016–1038, 2002

[JL99] Joy, M.; Luck, M.: Plagiarism in programming assignments, in *IEEE Transactions on Education*, vol. 42, no. 2, pp. 129–133, May 1999