Hellmig, L. und Hennecke, M. (Hrsg.): Informatikunterricht zwischen Aktualität und Zeitlosigkeit,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 187

Large Language Models in der Berufsausbildung von IT-Fachkräften

Sven Jacobs[1], Steffen Jaschke[2]

Abstract: Die Auswirkungen von Large Langue Models (LLM) wie GPT-4 und darauf basierende Anwendungen (z.B. ChatGPT) in Bildungskontexten sind Gegenstand des wissenschaftlichen und gesellschaftlichen Diskurses. Zur Förderung von Programmierkompetenzen wurde eine Webanwendung entwickelt, die GPT-4 nutzt, um auf Basis von Aufgabenstellung, Programmcode und Compilerausgabe sowie automatisierten Testergebnissen Feedback für Lernende zu formulieren. Die mit dieser Webapplikation durchgeführte Vorstudie anhand von zwei Aufgaben für Programmieranfänger*innen liefert erste Ergebnisse zum Einsatz von LLMs in der Programmierausbildung, auch von IT-Fachkräften. Bei einem Großteil der hierzu generierten Feedbacks werden syntaktische und semantische Fehler bereits vollständig adressiert. Bei unvollständigem oder „falschem“ Feedback wird die Hypothese aufgestellt, dass in diesen Fällen präzisere Aufgabenstellungen sowie Testfälle im Kontext des LLMs notwendig sein könnten.

Keywords: Large Language Models; Feedback; Programmierausbildung; IT-Berufe

1 Einleitung

Die aktuellen Entwicklungen im Bereich *künstlicher Intelligenz (KI)* im Allgemeinen und der Einsatz von *Large Language Models (LLM)* im Speziellen haben unbestritten großen Einfluss auf die berufliche Bildung. Parallel zu den Entwicklungen einer durchdringenden Automatisierung von industriellen Prozessen und einem hohen Grad der Vernetzung durch *cyber-physische Systeme (CPS)* durchdringen Systeme der *Künstlichen Intelligenz* nun zahlreiche Arbeits- und Geschäftsprozesse in nahezu allen Branchen.

Eine besondere Stellung nimmt KI in den gewerblich-technischen IT-Berufen Fachinformatiker/-in (Fachrichtung Daten und Prozessanalyse sowie Digitale Vernetzung) und IT-Systemelektroniker/-in ein. Während die Teilgebiete der Informatik (Praktische-, Technische-, Theoretische Informatik) einzelnen IT-Berufsbildpositionen zugeordnet werden können, bildet die Künstliche Intelligenz (KI) einen Querschnittsbereich, welcher zumeist nicht explizit benannt wird [Gr20]. Ungeachtet dessen benötigen IT-Fachkräfte Kompetenzen zur reflektierten Nutzung künstlicher Intelligenzen als Werkzeug im eigenen Betrieb. [Ku19]. Beispielsweise unterstützen auf LLM basierende Produkte, wie *GitHub Copilot*, Anwendungsentwicklerinnen und Anwendungsentwickler bei der Implementierung von Softwaresystemen.

[1] Universität Siegen, Didaktik der Informatik, Hölderlinstr. 3, 57076 Siegen, DE sven.jacobs@uni-siegen.de
[2] Universität Siegen, Didaktik der Informatik, Hölderlinstr. 3, 57076 Siegen, DE steffen.jaschke@uni-siegen.de

doi:10.18420/infos2023-017

Künstliche Intelligenz tangiert die berufliche Bildung also auf zwei Ebenen. Als Bildungsgegenstand und außerdem durch den Einsatz von KI-basierten Werkzeugen im Bildungsbereich [Ja23]. Im Folgenden wird eine empirische Vorstudie vorgestellt, welche betrachtet, wie LLMs als lernunterstützendes Werkzeug auch in der Ausbildung von IT-Fachkräften, insbesondere beim Erlernen programmiersprachlicher Konzepte, eingesetzt werden können.

2 Large Language Models (LLMs)

Generative Pre-trained Transformer 4 (GPT-4) ist ein multimodales Sprachmodell, das die Architektur des Transformer-Decoders (sequentielles Lernen) verwendet [Va17]. GPT-4 lernt, die Wahrscheinlichkeit des Auftretens eines Wortes in einer Sequenz auf Basis aller vorherigen Wörter zu optimieren. Dabei nutzt es Aufmerksamkeitsmechanismen zur Gewichtung einzelner Elemente einer Eingabesequenz um die semantische und syntaktische Struktur von Sätzen (den Kontext) zu erfassen und sie in der generierten Ausgabe wiederzugeben [Op23]. Eine multimodale Nutzung der von OpenAI bereitgestellten API ist derzeit nicht möglich, da diese auf textuelle Ein- und Ausgabe beschränkt ist. Die Anfragen bestehen grundsätzlich aus Parametern und einem Prompt. Mithilfe der Parameter können die Länge und Zufälligkeit der Antwort beeinflusst werden. Der Prompt kann beispielsweise bei GPT-4 eine Kombination aus drei verschiedenen Nachrichtentypen sein:

- System-Nachricht: Zusätzliche Anweisungen, welche Entwickler dem LLM geben. Dies kann genutzt werden, um das Sprachmodell in eine bestimmte Rolle (z.B. Lehrperson) zu versetzen oder es nur auf bestimmte Nutzer-Nachrichten antworten zu lassen. Die System-Nachricht soll dabei am höchsten priorisiert werden, um zu verhindern, dass das Modell durch eine Nutzer-Nachricht seine Rolle verlässt oder Fragen außerhalb des Kontextes beantwortet.
- Nutzer-Nachricht: Typischerweise die Eingabe des Nutzers bei einem Chatbot. Die Nutzer-Nachricht kann jedoch auch automatisiert basierend auf anderem Kontext generiert werden.
- Assistent-Nachricht: Die Antwort des LLM auf den vorherigen Prompt.

2.1 LLM unterstützte Anwendungsentwicklung

In der betrieblichen Praxis können bereits heute z.B. mit *GitHub Copilot* oder *ChatGPT* Codefragmente erzeugt und erklärt und so die Implementierung effizienter gestaltet werden. Mit der Erweiterung *Code Interpreter* können Nutzer*innen Dateien hochladen, von dem Modell verändern, ergänzen oder korrigieren lassen und anschließend wieder herunterladen. Für das Lösen von Programmieraufgaben zeigen umfassende Evaluierungen eine hohe Leistungsfähigkeit von GPT-4 [Bu23]. Diese neuen Möglichkeiten müssen daher von allen

Bildungseinrichtungen bedacht werden. Um LLM basierte Anwendungen zu differenzieren, wird im Folgenden zwischen integrierter und externer LLM-Unterstützung unterschieden.

Externe LLM-Unterstützung umfasst Anwendungen, die LLMs nutzbar machen und nicht direkt in das Werkzeug, z.B. eine Programmierumgebung, integriert sind. Beispiele hierfür sind Chatbots wie *ChatGPT (OpenAI)*, *Bing (Microsoft)* oder *Bard (Google)*. Durch ihre Externalität können sie in Kombination mit jeder textbasierten Programmierumgebung eingesetzt werden, allerdings muss ihnen der Kontext in Form von vorhandenem Programmcode oder Dateien durch die Nutzenden zur Verfügung gestellt werden. Unter *integrierter LLM-Unterstützung* werden im Folgenden Anwendungen oder Anwendungserweiterungen verstanden, die LLMs nutzbar machen und direkt in das Werkzeug wie eine Programmierumgebung integriert oder integrierbar sind. Dazu gehören beispielsweise *GitHub Copilot* und *GitHub Copilot X* und weitere, die *Visual Studio* und andere Entwicklungsumgebungen erweitern können. Die integrierte unterscheidet sich von der externen LLM-Unterstützung zudem dadurch, dass Anfragen bestehend aus Prompt und Parametern nicht nur automatisiert im Hintergrund generiert, sondern auch ohne explizite Aufforderung durch den Nutzenden ausgeführt werden können. So werden in *GitHub Copilot* basierend auf vorhandenem Code und Kommentaren Vorschläge zur Vervollständigung gemacht oder bei Fehlern ein Kontextmenü mit Korrekturmöglichkeiten dargestellt. Auch das natürlichsprachige Erklären von vorhandenem Code oder das automatische Generieren von passenden Tests ist mit *GitHub Copilot Chat* mit wenigen Klicks ohne eigenen Prompt möglich.

2.2 LLM unterstützte Förderung von Programmierkompetenzen

Sowohl integrierte als auch externe LLM-Unterstützung können von Lernenden genutzt werden, um Aufgaben nicht oder weniger eigenständig zu lösen. Ein Verbot erscheint wenig zielführend, da Lernende auf die künftige Arbeitswelt vorbereitet werden müssen, in welcher der produktive Einsatz der beschriebenen Tools bereits Realität ist. Im Folgenden wird eine Alternative vorgestellt, welche Lernende anleitet, ohne ihnen die Lösung als Programmcode vorzugeben. Hierzu wurde eine integrierte LLM-Unterstützung konzipiert und implementiert. Dabei wird die System-Nachricht verwendet, um das LLM *GPT-4* in die Rolle eines Informatiklehrenden zu versetzen, welcher hilfreiches Feedback in einfacher Sprache formulieren soll.

2.3 Feedback beim Erwerb von Programmierkompetenzen

„In Instruktionskontexten bezeichnet man mit dem Begriff „Feedback“ alle Informationen, die einer Person nach dem oder auch beim Bearbeiten von Lernaufgaben über ihren aktuellen Lern- oder/und Leistungsstand angeboten werden, mit dem Ziel, dass diese Informationen für die Regulation des Lernprozesses in Richtung erwünschter Ziele genutzt werden.“ [Na18]

Bei Übungsplattformen für das Programmieren wird zwischen Ja/Nein-Feedback, Syntax-Feedback, Semantik-Feedback (basierend auf Absicht oder Programmcode), Layout-Feedback und Qualitäts-Feedback unterschieden [Le16]. Zur Generation von Feedback wurden für automatisierte Assessment Tools bereits diverse Möglichkeiten entwickelt [PLF22]. Der Vergleich zwischen automatisch generiertem und von Expert*innen formuliertem Feedback zeigt, dass bisher kein System in der Lage ist, Feedback auf der Basis von Kontext, Lernstrategie und möglichen Missverständnissen in gleichem Maße zu individualisieren wie Expert*innen [Je22].

3 Feedback Framework

Um aufgrund der rasanten Entwicklung bereits zu einem ersten wissenschaftlich fundierten Einsatz von LLMs in der beruflichen Bildung zu gelangen, wurde für Probanden der akademischen Bildung (N=51) eine integrierte LLM-Unterstützung als Webapplikation entwickelt. Diese ermöglicht das Bearbeiten von Programmieraufgaben, welche mit denen der beruflichen Bildung vergleichbar sind. Innerhalb dieser Applikation werden den Lernenden nach Anmeldung über einen CAS-Server ihrer Bildungseinrichtung die verfügbaren Aufgaben sowie ein Code-Editor anzeigt. Der dort formulierte Programmcode kann isoliert ausgeführt und getestet werden.

Kategorie	**Wert**
Anzahl der Lernenden	51
Anzahl der Aufgaben	26
Anzahl der Codeausführungen	1994
Anzahl der generierten Feedbacks	1099
Anzahl der bewerteten Feedbacks	768
Durchschnittliche Bewertung der Feedbacks	5,4844

Tab. 1: Summierte Datengrundlage

3.1 Prototypische Implementierung

Das Ergebnis, wie die Compilerausgabe und ein Test-Feedback, wird den Lernenden direkt unterhalb des Code-Editors angezeigt. Sofern weitere Hilfestellung benötigt wird, kann auf Anfrage der Lernenden ein Feedback erzeugt werden. Hierbei wird der sonstige Kontext bestehend aus Aufgabenstellung, Programmiersprache, Programmcode, Compilerausgabe und Testergebnis als Nutzer-Nachricht für das LLM formuliert. Aktuell wird die GPT-4 API von OpenAI verwendet. Hinzu kommt die statische System-Nachricht, welche GPT-4 in die Rolle des imaginären Informatiklehrenden *Kai* versetzt, welcher nicht die Lösung verrät, sondern nur zur selbstständigen Problemlösung anleiten soll.

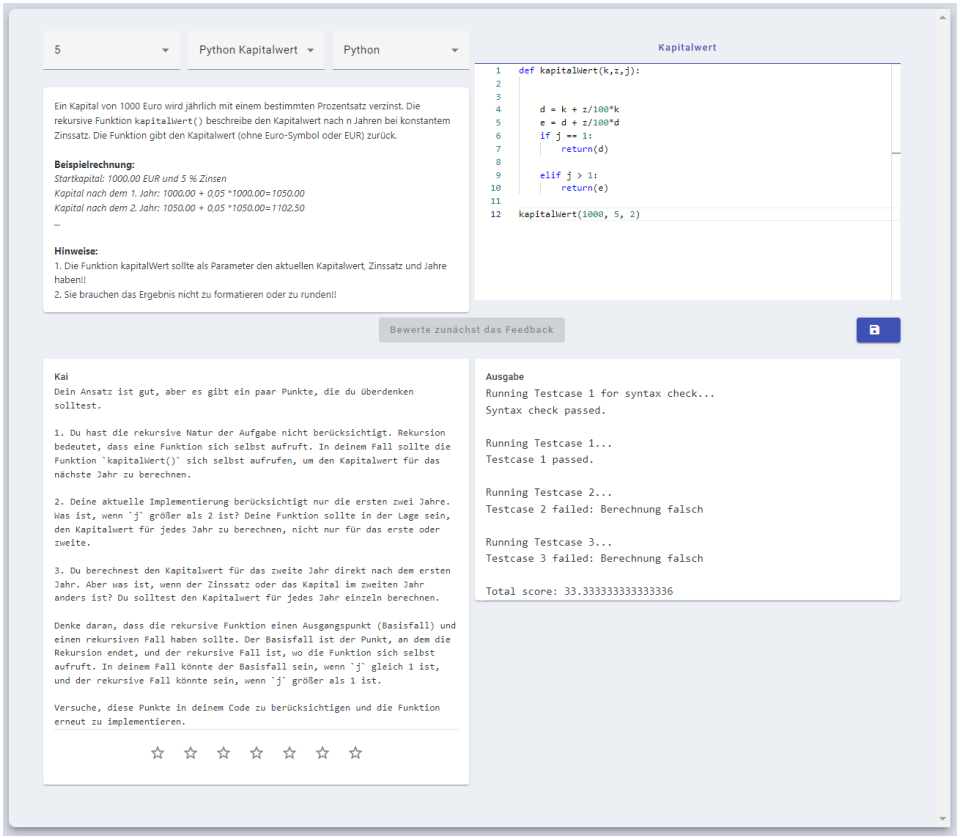

Abb. 1: Beispielhaftes Feedback in der Weboberfläche (Aufgabenstellung: oben links; Programmcode des Lernenden: oben rechts; KI-Feedback: unten links; Compiler-Output und Testergebnisse: unten rechts)

3.2 Evaluation

Bei dem beschriebenen System wurde von 51 Lernenden (davon 32 mit mehr als fünf Codeausführungen) zu 26 verschiedenen Aufgaben (18 Java und 8 Python) über 7 Wochen 1994 mal Programmcode ausgeführt. Zu diesen wurden 1099 Feedbacks generiert, welche insgesamt 768-mal auf einer Likert-Scala von sehr schlecht (ein Stern) bis sehr gut (sieben Sterne) bewertet wurden. Die durchschnittliche Bewertung liegt dabei bei 5,5. Obwohl die Nutzung des Tools optional war, blieb die Nachfrage auch über mehrere Wochen hoch. Die Tatsache, dass zu 55% nach einer Codeausführung auch ein Feedback angefragt wurde, kann als erstes Indiz für dessen Nützlichkeit für die Lernenden verstanden werden.

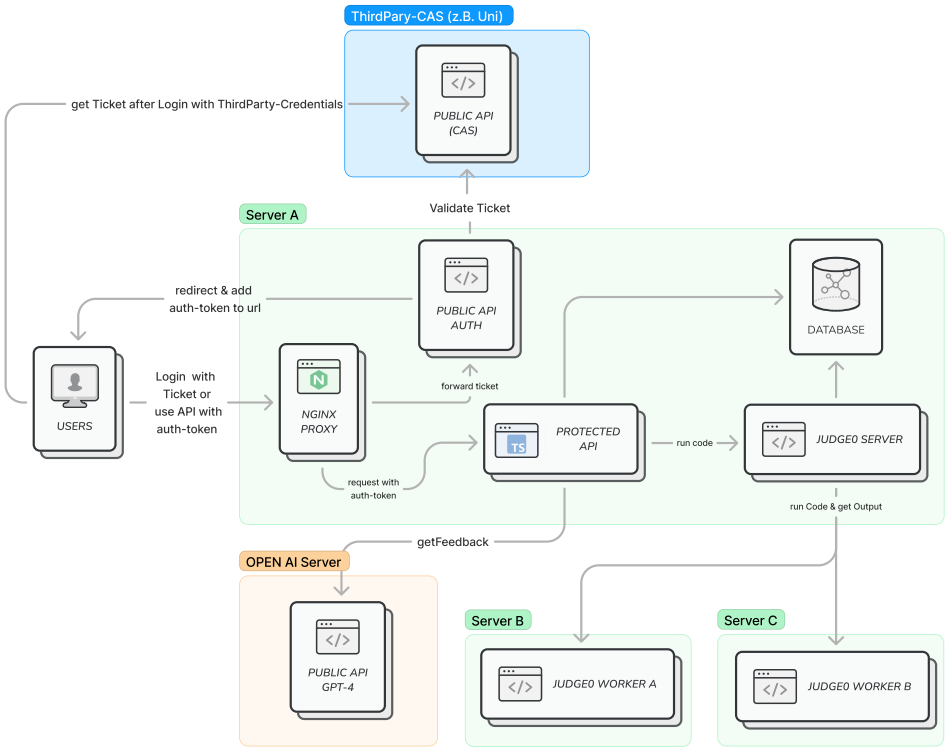

Abb. 2: Architektur der prototypischen Webapplikation

3.2.1 Ergebnisse

In Anlehnung an die Methodik von [He23] wurde anhand einer Python- und einer Java-Aufgabe (Tab. 2) untersucht, inwieweit das Feedback die in der Lösung vorhandenen Fehler adressiert. Die Aufgaben wurden so ausgewählt, dass der Abstand zwischen den jeweiligen Bearbeitungszeiträumen möglichst gering ist. So stammt die Python-Aufgabe aus der fünften und die Java-Aufgabe aus der sechsten Übungswoche. Es wurde betrachtet, ob falsche Vorschläge gemacht oder Fehler halluziniert wurden. Falsches Feedback bezieht sich auf Feedback, das auf vorhandene Fehler hinweist, dessen Umsetzung aber nicht zum richtigen Ergebnis führt. Bei halluzinierten Fehlern bezieht sich das Feedback auf nicht vorhandene Fehler im Programmcode. Darüber hinaus wurde untersucht, ob unnötige Verbesserungen vorgeschlagen wurden, wobei unter unnötigen Verbesserungen solche zu verstehen sind, die für die Lösung im Sinne der Aufgabenstellung nicht notwendig sind. Solche unnötigen Verbesserungen wurden insbesondere dann vorgeschlagen, wenn der Programmcode im Sinne der Tests bereits korrekt war. Sie beziehen sich meist auf kürzere Schreibweisen und stilistische Fragen.

Aufgabe	Sprache	Aufgabentext
Kapitalwert	Python	Ein Kapital von 1000 Euro wird jährlich mit einem bestimmten Prozentsatz verzinst. Die rekursive Funktion kapitalWert() beschreibt den Kapitalwert nach n Jahren bei konstantem Zinssatz. Die Funktion gibt den Kapitalwert (ohne Euro-Symbol oder EUR) zurück. Beispielrechnung: Startkapital: 1000.00 EUR und 5 % Zinsen Kapital nach dem 1. Jahr: 1000.00 + 0,05 *1000.00=1050.00 Kapital nach dem 2. Jahr: 1050.00 + 0,05 *1050.00=1102.50
Maximalwert	Java	Implementieren Sie die Methode max() der Klasse Starter im vorgegebenen Code-Gerüst (Starter.java). Die Methode max gibt die Größere der beiden natürlichen Zahlen a und b zurück. Wenn beide Zahlen gleich sind, wird dieser Wert zurückgegeben.

Tab. 2: Aufgabenbeschreibungen

Vergleicht man die Ergebnisse zwischen der Aufgabe Kapitalwert und der Aufgabe Maximalwert (Tab. 3), so fällt auf, dass die Ergebnisse für die Aufgabe Maximalwert besser sind. Die folgende qualitative Einzelbetrachtung soll Hypothesen zur Erklärung dieses Unterschiedes aufstellen.

3.2.2 Qualitative Einzelbetrachtung

Wie in Abb. 1 dargestellt, soll eine Funktion zur Berechnung eines Kapitalwertes basierend auf Startkapital, Prozentsatz und Dauer in Jahren erstellt werden. Im Kontext der anderen Aufgaben und der die Übung begleitenden Lehrveranstaltung bietet sich eine rekursive Herangehensweise an. Die Lösung des Lernenden (Abb. 1) ist zwar syntaktisch korrekt, jedoch entspricht sie semantisch nicht den Anforderungen. Daher ist hier auch nur einer der drei Testfälle korrekt. Der Studierende erhält durch diese Angaben jedoch noch keine Information darüber, weshalb dem so ist und wie er den Fehler beheben könnte. In diesem Beispiel wird richtigerweise auf die mögliche rekursive Umsetzung hingewiesen sowie der aktuelle Fehler benannt, dass die Lösung des Lernenden nur zwischen zwei Fällen unterscheidet und somit für $j = 1$ die richtige Lösung und für $j > 1$ immer das Ergebnis für eine Dauer von zwei Jahren ausgibt. Wie in der System-Nachricht gefordert, liefert Kai eine zusätzliche einfache Beschreibung der notwendigen informatischen Konzepte wie in diesem Falle des Basis- und Rekursionsfalls. Zudem bleibt GPT-4 in der beschriebenen Rolle des freundlichen und konstruktiven Informatiklehrers.

Bei der Evaluation des Feedbacks (Tab. 3) zeigte sich bei 12 der 80 generierten Feedbacks, dass die Testfälle 2 & 3 für die Aufgabe negativ ausfielen, obwohl die Lösung des Lernenden korrekt war. Testfall 2 erwartet für Input (500, 3, 7) den Output (614.936932712435) und Testfall 3 für Input (100, 2, 18) den Output (142.82462475762728). Wird der Kapitalwert

	Aufgaben	
Kategorien	Kapitalwert	Maximalwert
Anzahl der Codeausführungen	113	38
Anzahl der generierten Feedbacks	80	17
Durchschnittliche Bewertung der Feedbacks	5,9	6,3
Mindestens ein Fehler erkannt	57 (71%)	17 (100%)
Alle Fehler erkannt	40 (50%)	15 (88%)
Falsche Verbesserungsvorschläge	13 (16%)	1 (6%)
Halluzinierte Fehler	8 (10%)	0 (0%)
Unnötige Verbesserungen	5 (6%)	4 (23,5%)
Codeausgaben	4 (5%)	0 (0%)

Tab. 3: Evaluation des Feedbacks

im Rekursivfall mit kapitalWert(kapital * (1 + zinssatz / 100), zinssatz, jahre - 1) anstatt mit kapitalWert(kapital + kapital * zins / 100, zinssatz, jahre - 1) übergeben, so wird durch die geänderte Reihenfolge anders gerundet und es kommt zu unerwünschten Fehlern. Da dies an den gleichen Stellen wie die falschen Feedbacks und halluzinierten Fehler auftrat, lässt dies die Hypothese zu, dass diese im Zusammenhang mit den Testfällen stehen.

Des Weiteren wurde festgestellt, dass der Unterschied zwischen Zins und Zinssatz in den generierten Feedbacks teilweise nicht berücksichtigt wurde. Sowohl für GPT-4 als auch für die Lernenden ist aus der Aufgabenstellung nicht ersichtlich, ob der Funktionsaufruf in den Tests mittels Zinses oder Zinssatzes erfolgt und eine Division durch 100 erforderlich ist. Durch nachfolgende Tests konnte das Feedback in solchen Fällen verbessert werden, wenn nicht nur das Testergebnis, sondern auch die Testfälle in der generierten Nutzer-Nachricht als Kontext für GPT-4 zur Verfügung stehen.

4 Diskussion

Die Ergebnisse (Tab. 3) dieser ersten Vorstudie, die auf einem experimentellen Einsatz des hier vorgestellten Prototyps basiert, erscheinen vielversprechend. So konnten bei den für Programmieranfänger*innen konzipierten Aufgaben nicht nur die meisten semantischen und syntaktischen Fehler adressiert, sondern meist auch falsche und unnötige Hinweise vermieden werden. Diesbezügliche Verbesserungen scheinen basierend auf Einzelbetrachtungen durch eine Ausweitung des Kontextes sowie durch bessere Tests möglich. Darüber hinaus zeigen die Ergebnisse im Vergleich zu [He23], dass GPT-4 für diesen Anwendungsfall in allen untersuchten Kategorien besser abschneidet als GPT-3.5. Vor dem Hintergrund der generell besseren Ergebnisse von GPT-4 gegenüber GPT-3.5 [Bu23] [Op23] war dies zu erwarten. Insgesamt könnte eine Schwelle überschritten worden sein, da im Vergleich zu [He23] überproportional mehr Feedback von den Lernenden eingeholt wurde. Ein Grund dafür könnte sein, dass die Anzahl der falschen Verbesserungsvorschläge deutlich reduziert wurde.

Aus didaktischer Sicht ist bei den Ergebnissen hervorzuheben, dass GPT-4 durch die System-Nachricht nur selten unerwünschten Programmcode formuliert und damit die Lösung vorgegeben hat. Damit wurde ein in [He23] dargestelltes Teilproblem gelöst. Fraglich bleibt, inwiefern das formulierte natürlichsprachliche Feedback durch teilweise sehr detaillierte Hinweise selbstständige Problemlöseprozesse verhindert.

Es ist nicht auszuschließen, dass die Lernenden die Anwendung parallel zu einer externen Entwicklungsumgebung nutzen und somit der Anteil des KI-Feedbacks an allen Codeausführungen geringer ausfällt. Die hier präsentierten Ergebnisse beziehen sich auf beispielhafte Aufgaben für Programmieranfänger*innen. Somit bleibt fraglich, inwiefern die Komplexität und der Umfang der Aufgaben - beispielsweise mit mehreren Klassen - die Ergebnisse beeinflussen.

5 Fazit und Ausblick

Im Rahmen dieser Vorstudie hat sich gezeigt, dass Lernende das KI-generierte Feedback zu Programmieraufgaben insgesamt positiv bewerten. Dies und die rege Nutzung bei vorhandenen Alternativen belegen einen Mehrwert. Die Evaluation der generierten Feedbacks zeigt, dass diese bereits einen Großteil der vorhandenen Fehler im Programmcode der Lernenden adressieren. Darüber hinaus konnte durch Einzelbetrachtungen die Hypothese aufgestellt werden, dass zusätzlicher Kontext in Form von Testfällen hilfreich sein kann. Durchgeführt wurde die Studie in den Einführungsveranstaltungen der Bachelorstudiengänge Informatik, jedoch mit Fokus auf Inhalte, welche ebenso in der beruflichen Bildung adressiert werden, der Programmierung in Python und Java. Das Forschungsdesign und der Prototyp werden in weiteren Iterationen unter anderem hinsichtlich folgender Punkte angepasst:

- Die Testfälle werden als zusätzlicher Kontext als Nutzer-Nachricht aufgenommen.
- Auszubildende im Beruf Fachinformatiker/-in werden das Tool in der C# Programmierung erproben.
- Es wird ein differenzierteres Feedback von den Lernenden eingeholt (vor- und nach erfolgreicher Adaption der eigenen Implementierung).
- Es wird erprobt, welche Auswirkungen komplexere Aufgabenstellungen mit umfangreicheren Problemlösungen haben.
- Vorherige Codeausführungen und generierte Feedbacks werden als zusätzlicher Kontext aufgenommen.
- Möglichkeiten weiterentwickelter LLMs werden evaluiert.

Literatur

[Bu23] Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; Nori, H.; Palangi, H.; Ribeiro, M. T.; Zhang, Y.: Sparks of Artificial General Intelligence: Early experiments with GPT-4, 2023, URL: https://doi.org/10.48550/arXiv.2303.12712.

[Gr20] Grimm, A.: IT-Lehrkräfteausbildung: Zum Problem gewachsener Strukturen. lernen & lehren/139, S. 116–122, 2020.

[He23] Hellas, A.; Leinonen, J.; Sarsa, S.; Koutcheme, C.; Kujanpää, L.; Sorva, J.: Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. Proceedings of the 2023 ACM Conference on International Computing Education Research V.1/, 2023, URL: https://arxiv.org/abs/2306.05715.

[Ja23] Jaschke, S.; Klusch, M.; Krupka, D.; Losch, D.; Michaeli, T.; Opel, S.; Schmid, U.; Schwarz, R.; Seegerer, S.; Stechert, P.: Positionspapier der Gesellschaft für Informatik e.V. (GI): Künstliche Intelligenz in der Bildung, 2023.

[Je22] Jeuring, J.; Keuning, H.; Marwan, S.; Bouvier, D.; Izu, C.; Kiesler, N.; Lehtinen, T.; Lohr, D.; Peterson, A.; Sarsa, S.: Towards Giving Timely Formative Feedback and Hints to Novice Programmers. In: Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education. ITiCSE-WGR '22, Association for Computing Machinery, New York, S. 95–115, 2022, URL: https://doi.org/10.1145/3571785.3574124.

[Ku19] Kultusministerkonferenz: Rahmenlehrplan für die Ausbildungsberufe Fachinformatiker und Fachinformatikerin IT-System-Elektroniker und IT-System-Elektronikerin, Berlin, 2019, URL: https://www.kmk.org/fileadmin/Dateien/pdf/Bildung/BeruflicheBildung/rlp/Fachinformatiker_19-12-13_EL.pdf.

[Le16] Le, N.-T.: A Classification of Adaptive Feedback in Educational Systems for Programming. Systems 4/2, 2016, URL: https://doi.org/10.3390/systems4020022.

[Na18] Narciss, S.: Feedbackstrategien für interaktive Lernaufgaben. In (Kracht, S.; Niedostadek, A.; Sensburg, P., Hrsg.): Praxishandbuch Professionelle Mediation. Springer Reference Psychologie, Springer Berlin Heidelberg, S. 1–24, 2018.

[Op23] OpenAI: GPT-4 Technical Report, 2023, URL: https://doi.org/10.48550/arXiv.2303.08774.

[PLF22] Paiva, J. C.; Leal, J. P.; Figueira, Á.: Automated Assessment in Computer Science Education: A State-of-the-Art Review. ACM Transactions on Computing Education 22/3, S. 1–40, 2022, URL: https://doi.org/10.1145/3513140.

[Va17] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I.: Attention Is All You Need, 2017, URL: https://doi.org/10.48550/arXiv.1706.03762.