Programmierung im Dialog: Einsatz von KI-Chatbots im Hochschulstudium der Bibliotheks- und Informationswissenschaft

Matthias Finck

Einleitung

Studierende im Bereich Bibliotheks- und Informationswissenschaften sehen sich durch die digitale Transformation in der Gesellschaft mit Herausforderungen im späteren beruflichen Alltag konfrontiert, für die der Erwerb breiter digitaler Kompetenzen notwendig ist[1]. Das reicht z.B. von der automatischen Inhaltserschließung digitalisierter Werke über die prototypische Gestaltung innovativer Such-Interfaces bis hin zur Vermittlung digitaler Kompetenzen im Bereich der Nutzung von KI-Tools.

Die Studierenden benötigen dazu z.B. Fähigkeiten im Umgang mit technischen Metadatenstrukturen wie XML und den Prozessen und Tools zur Verarbeitung dieser Daten. Sie brauchen Grundlagenwissen zu webbasierten Informationsdiensten, Schnittstellen und Architekturen sowie die Fähigkeit, die Integration und Interoperabilität technischer Systeme beurteilen zu können. Sie müssen KI-basierte Systeme nutzen und einschätzen können.

Für all diese Aufgaben müssen die Studierenden ein Verständnis für die Systeme, die Software und den Programmcode mit dem Ziel aufbauen, später im beruflichen Alltag IT-Projekte bzw. IT-gestützte Prozesse planen und umsetzen zu können bzw. diese Prozesse gegebenenfalls zu steuern. Diese Herausforderung gilt natürlich vor allem, aber nicht mehr nur für den Bereich der wissenschaftlichen Bibliotheken[2], auch die Öffentlichen Bibliotheken sehen z.B. die Vermittlung digitaler Kompetenzen als ein wichtiges Handlungsfeld[3].

Die Hochschulen, die bibliotheks- und informationswissenschaftliche Studiengänge anbieten, haben auf diesen Bedarf i.d.R. reagiert und Studieninhalte in Bereichen wie Data Science, Informationssysteme, KI und Programmierung in die Curricula aufgenommen.

Abstract

Daten anreichern, Prototypen für neue Such-Interfaces bauen. Bei dieser Art von Aufgaben könnten sich Bibliothekar:innen inzwischen von KI-Chatbots unterstützen lassen – wenn entsprechende Kompetenzen im Umgang mit diesen Tools erworben wurden. Es gibt erste Erfahrungen mit der Integration dieser Tools in der bibliothekarischen Ausbildung im Studiengang Bibliotheks- und Informationsmanagement der HAW Hamburg. Dort ist diese Form der dialogischen Unterstützung in zwei Modulen zum Einsatz gekommen und die Evaluationsergebnisse zeigen, wie sich der Zugang und Umgang zur Programmierung durch diese dialogische Form verändert. Vor allem die stärker wahrgenommene Selbstwirksamkeit kann z.B. dazu beitragen, dass sich die Kluft zwischen fachlichen und technischen Rollen in der Berufspraxis verringern und Bibliothekar:innen stärker als Vermittler:innen zwischen fachlichen Anforderungen und technischen Lösungen agieren könnten.

Enriching data and building prototypes for new search interfaces – these are the kinds of tasks for which librarians can now be supported by AI chatbots, provided they have acquired the necessary skills to use these tools. Initial experiences with integrating such tools into library education have been made in the Library and Information Management degree program at HAW Hamburg. There, this form of dialogical support has been used in two modules, and the evaluation results show how this dialogical approach changes access to and engagement with programming. In particular, the increased sense of self-efficacy may help reduce the gap between professional and technical roles in practice, enabling librarians to act more strongly as mediators between domain-specific requirements and technical solutions.

Code Literacy als Ausbildungsziel

In diesem Beitrag soll ein Blick auf die Vermittlung programmiersprachlicher Kompetenzen in informationswissenschaftlichen Studiengängen geworfen werden, die bislang eine besondere Herausforderung darstellen. Eine aktuelle Studie zu den akkreditierten Online-Masterstudiengängen der American Library Association zeigt, dass 60% Kurse mit Programmieranteil anbieten[4]. Allerdings sind diese Kurse häufiger Wahlfächer (66,7 %) als Pflichtfächer (33,3 %). Der Autor der Studie, Austin Stroud, weist deshalb auf eine mögliche Lücke in der Vermittlung grundlegender Kompetenzen für angehende Bibliothekar:innen hin, da seiner Ansicht nach der Bedarf an Programmierkenntnissen im Bibliothekswesen weiter steigen wird.

Um programmiersprachliche Kompetenz in höheren Programmiersprachen wie Python und PHP zu erwerben,

1 Vgl. Schneemann 2023
2 Vgl. Georgy 2019
3 Vgl. Lücht 2019
4 Vgl. Stroud 2025, S.309

die im beruflichen Alltag des Bibliothekswesens benötigt werden, braucht es bislang ein nicht unerhebliches technisches Vorwissen, das Verständnis für algorithmische Ablaufmuster und eine hohe syntaktische Präzision beim Schreiben des Codes.

Der Kompetenzerwerb in dieser Art von Programmiersprachen ist gekennzeichnet durch eine steile Lernkurve und hohes Frustrationspotenzial für die Studierenden, weil schon die Erstellung einfachster Programmierbeispiele oder das Nachvollziehen rudimentärer Programmcodes mit genannten hohen Einstiegshürden versehen ist. Dies führt dazu, dass in der Didaktik der Programmiersprachen oftmals mit vereinfachten Sprachen wie z.B. Scratch gearbeitet wird, um die Komplexität zu reduzieren und den Einstieg zu erleichtern[5].

Auf der anderen Seite geht es in der bibliothekarischen Ausbildung explizit nicht darum, vollwertige Programmierer:innen auszubilden. Das Ziel der Ausbildung sollte vielmehr ein Kompetenzprofil sein, das mit „Code-Literacy" beschrieben werden kann. Angelehnt an den Begriff der Data-Literacy (Datenkompetenz), der die Fähigkeit beschreibt, Daten kritisch zu erfassen, zu managen, zu bewerten, zu analysieren und anzuwenden[6], umfasst der Begriff Code-Literacy demnach die Fähigkeit, Programmcode kritisch erfassen, bewerten, analysieren und anwenden zu können.

Studierende der Bibliotheks- und Informationswissenschaften sollen also in die Lage versetzt werden, Code lesen, über Code kommunizieren und prototypisch Code generieren zu können, um auf Augenhöhe aus der fachlichen Expertise heraus mit Entwickler:innen zu kommunizieren – nicht um selbst Softwareentwickler:innen zu werden.

Bislang waren die Hürden der Vermittlung von Code-Literacy allerdings ähnlich hoch wie die einer programmiersprachlichen Ausbildung, weil die Möglichkeiten der Code-Produktion in höheren Programmiersprachen ohne entsprechende algorithmische und technische Kompetenz begrenzt waren.

Erschwerend kommt sogar noch hinzu, dass die möglichst praxisnahen Anwendungsbeispiele in informationswissenschaftlichen Studiengängen in der Regel deutlich komplexer sind als in einer klassischen Einführungsveranstaltung eines informatischen Studiengangs. Denn das Lernziel besteht nicht primär darin, grundlegende Algorithmen und Strukturen zu verstehen, sondern die Anwendung dieser auf den spezifischen Kontext. Das heißt, es ist z.B. nicht das Ziel, auf einer schlichten exemplarischen Datenstruktur einfache Operationen auszuführen, um die abstrakte Funktionsweise von Programmiersprachen zu durchdringen, sondern ihre Anwendung auf komplexen, möglichst realitätsnahen Daten nachzuvollziehen – wie z.B. einem komplexen XML-Dokument zur Beschreibung von Metadaten eines digitalen Objekts auf der Basis des METS/MODS-Formats.

Das setzt allerdings die Implementierung deutlich komplexerer Programme voraus, an denen die Studierenden fast zwangsläufig scheitern müssen, wenn sie diese selbst implementieren müssen. Und der didaktisch empfohlene Zwischenschritt über vereinfachte Sprachen und Sprachkonstrukte ist in der Ausbildung der Informationswissenschaften weder vom zeitlichen Umfang zu leisten noch im Hinblick auf das Kompetenzziel der Code-Literacy in Bezug auf die Programmiersprachen wie PHP oder Python zielführend.

Coding im Dialog mit der KI – am Beispiel des GitHub-Copilot

Die beschriebenen Einstiegshürden und steilen Lernkurven werden aktuell durch die immer stärkere Durchdringung von KI-Technologien massiv gesenkt. Mit der Verfügbarkeit der Large-Language-Models (LLMs) halten KI-Assistenten auch im Bereich der Programmierung Einzug. Mit den KI-Assistenten an der Seite ist eine Art Pair-Programming[7] möglich, bei dem die Studierenden den Assistenten in ihrer Entwicklungsumgebung fragen können, Code zu erklären, Fehler im Code zu finden oder auch eigenen Code zu schreiben.

Ein Beispiel für einen solchen Assistenten ist der GitHub-Copilot. GitHub ist eine der weltweit größten Entwicklungsplattformen, auf der Programmcode mittels GIT verwaltet und bereitgestellt wird[8]. Die Plattform, die 2008 gegründet wurde und lange die meistgenutzte Plattform für Open-Source-Projekt war, wird aktuell von einem Tochterunternehmen von Microsoft bereitgestellt, das die Plattform 2018 übernommen hat. Die Large-Language-Modelle des GitHub-Copilots konnten aufgrund der langen Historie und der großen Popularität der Plattform auf über 159 GB Daten von Programmcode aus über 54 Millionen frei zugänglicher Code-Repositorien trainiert werden. Die Modelle verfügen so über einen sehr spezifischen Trainingsdatenbestand und sind damit als programmiersprachliche Assistenz sehr präzise und spezifisch in ihrem Antwortverhalten.

Der Github-Copilot kann als Erweiterung in verschiedenen Programmierumgebungen wie Eclipse, JetBrains oder Visual Studio Code integriert werden (siehe Abbil-

5 Vgl. z.B. Kelleher & Pausch 2005

6 Vgl. Gläser 2020

7 Vgl. Hanks et al. 2011

8 https://en.wikipedia.org/wiki/GitHub

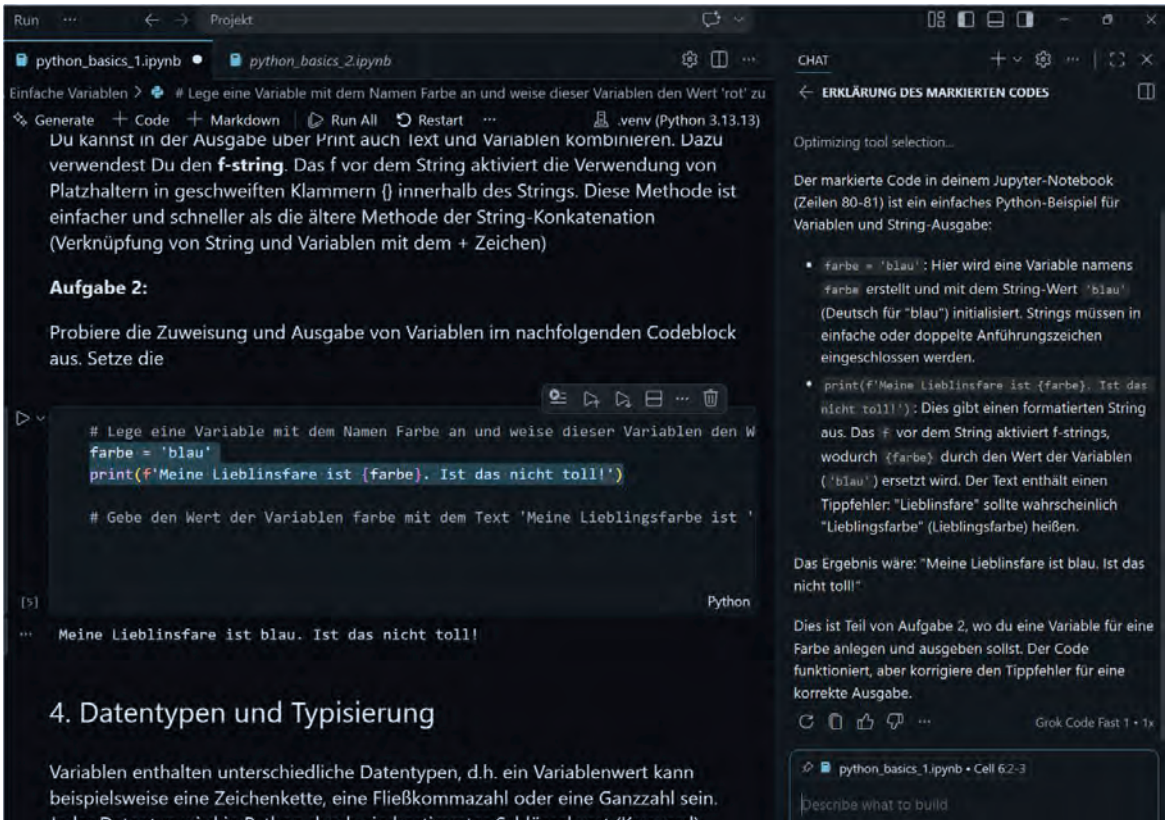

Abbildung 1: Ausschnitt des Programmiersprachen Editor Visual Studio Code mit dem Copilot-Chat in der rechten Spalte

dung 1). Ursprünglich auf dem Sprachmodell GPT-4 entwickelt ist inzwischen eine Vielzahl an Sprachmodellen verschiedener Anbieter verfügbar wie neben GPT auch Claude oder Gemini.

Der Copilot kann nach der Installation innerhalb der Programmierumgebung während des Entwicklungsprozesses verwendet werden, um z.B. Passagen des Programmcode zu erklären, indem der Code in dem Editor markiert wird und um eine Erläuterung gebeten wird (siehe Abbildung 1). Mit Unterstützung des Copilots lassen sich auch in einer geöffneten Programmdatei Fehler analysieren, Korrekturvorschläge machen und diese unmittelbar umsetzen.

Beide Möglichkeiten unterstützen vor allem Anfänger:innen, da ohne größere Programmiererfahrung gerade das Nachvollziehen von Code-Fragmenten oder das Finden teils kleiner syntaktischer Fehler, die den Ablauf des Programms verhindern, sehr zeitintensiv sind. Darüber hinaus kann der KI-Assistent aber auch Code-Vorschläge generieren, die auf der Basis des Wissens über den lokalen Projektstand (Kontext) und der sehr spezifischen Trainingsdaten zu vergleichsweise guter Qualität und wenig Halluzinationen führen.

Einsatz des Copilots im Studiengang der HAW

Es gibt bereits erste Erfahrungen mit dem Einsatz von LLMs in der Programmierausbildung, die Potenziale und Risiken beschreiben. Diese reichen von der Beschleunigung in der Bearbeitung von Aufgaben, bis zur unreflektierten Übernahme von falschem Code.[9] Diese bisherigen Erfahrungen basieren aber alle auf didaktischen Zielen der informatischen Programmierausbildung und nicht der Vermittlung von Code-Literacy. Aus diesem Grund wurde für diesen Beitrag der Einsatz eines LLMs in programmiersprachlichen Veranstaltungen eines informationswissenschaftlichen Studiengangs untersucht.

Die empirische Grundlage dieses Beitrags bilden Lehrveranstaltungen im Studiengang Bibliotheks- und Informationsmanagement an der Hochschule für Angewandte Wissenschaften in Hamburg (HAW)[10], der konsequent auch auf den digitalen und programmiersprachlichen Kompetenzaufbau setzt. Programmierbezogene Inhalte sind dort curricular fest verankert und erstrecken sich über mehrere Semester. In den ersten beiden Semestern werden grundlegende Kenntnisse in IT und Programmierung vermittelt, während in späteren Semestern Anwendungen in den Bereichen Data Science und automatische Erschließung im Vordergrund der IT-Ausbildung stehen.

Das Modul „IT-Grundlagen und Coding“ im Umfang von sechs LVS im ersten und zweiten Semester versetzt die Studierenden in die Lage, „Grundkonzepte der Programmierung wie Kontrollstrukturen, Datentypen, Programmabläufe, Algorithmen und Objektorientierung exemplarisch an Hand einer im Umfeld der Webprogrammierung

9 Vgl. Wienkop & Burk 2025

10 https://www.haw-hamburg.de/bachelor-bibliotheks-und-informationsmanagement/

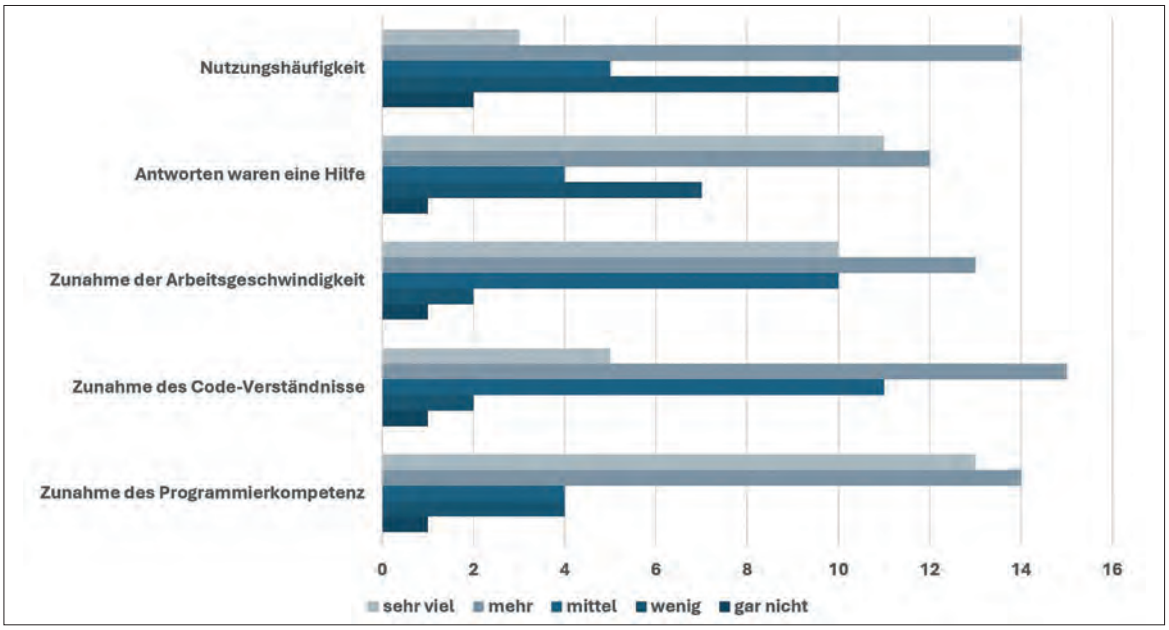

Abbildung 2: Auswirkungen des Einsatzes von Copilot in der Lehre

geeigneten Programmiersprache, wie z.B. Python oder JavaScript, unter Anleitung nutzen“[11] zu können. Neben Veranstaltungen zu Datenbanken oder Suchmaschinentechnologie folgen dann im vierten und fünften Semester die aufeinander aufbauenden Veranstaltungen „Data Science / Computerlinguistik“ und „Anwendungen der maschinellen Erschließung“ im Rahmen des Moduls „Datenanalyse und Anwendungen“, in denen mittels Python „die grundlegenden Verfahrensweisen des maschinellen Lernens (überwachtes Lernen, unüberwachtes Lernen) inklusive ‚Deep Learning‘ und der symbolischen KI“[12] vermittelt und im Rahmen maschineller Erschließung erprobt werden.

Sowohl im Grundlagenmodul als auch in dem Modul zu Data Science und KI geht es um den programmiersprachlichen Kompetenzaufbau im Sinne der Code-Literacy. Die Studierenden sollen den in der Veranstaltung entwickelten Code also in erster Linie nachvollziehen, anpassen und geeignet anwenden können.

Bis zum Sommersemester 2024 wurde dabei auf eine klassische programmiersprachliche Ausbildung ohne die Unterstützung eines KI-Assistenten gesetzt, während im Wintersemester 2024/2025 und im Sommersemester 2025 zum ersten Mal in beiden Modulen die Lehre mit Hilfe des Copilots durchgeführt und systematisch erprobt wurde. Die Studierenden waren aufgefordert, die KI-Assistenz zu nutzen, um Code zu generieren, bestehende Programme zu verstehen und Lösungen für konkrete Aufgaben zu entwickeln. Die Interaktion erfolgte dabei überwiegend in natürlicher Sprache, wodurch Programmierung für die Studierenden als dialogischer Prozess erfahrbar wurde.

Sowohl in dem Jahr vor dem Einsatz des Copilots als auch in dem Jahr des Experiments mit der Assistenz durch KI wurden die Veranstaltungen von demselben Dozenten und mit derselben Programmierumgebung – Visual Studio Code – durchgeführt. In allen Veranstaltungen gab es zwei parallele Gruppen mit ca. 20 bis 25 Studierenden – also ca. 40 bis 50 Studierenden pro Jahrgang. Dabei hatten die Studierenden im vierten und fünften Semester auch den Vergleich zu der zuvor stattgefundenen Grundlagenveranstaltung ohne KI-Unterstützung. Und in der Grundlagenveranstaltung gab es im Untersuchungszeitraum zwei vergleichbare Gruppen aus zwei Jahrgängen im ersten und zweiten Semester, die die Grundlagen einmal mit und einmal ohne den Co-Piloten erlernt hatten.

Die Auswertung der systematischen Erprobung basiert auf quantitativen und qualitativen Daten aus Lehrveranstaltungsevaluationen im Hinblick auf den KI-Einsatz sowie auf konkreten Rückmeldungen der Studierenden.

Erfahrungen im Umgang mit dem Codepilot

Die Analyse der Evaluationen zeigt zunächst, dass der Einsatz von KI-Chatbots zu einer signifikanten Verbesserung der wahrgenommenen Programmierkompetenzen führt. Knapp die Hälfte der Studierenden (47%) schätzt, dass der Einsatz des Copilots zu einer Steigerung der Programmierkompetenz geführt hat.

Diese Einschätzungen werden auch durch die qualitativen Daten der Studierendenbefragung unterstützt:

„Insgesamt bin ich begeistert von der Unterstützung des Copiloten. Ich hatte teilweise das Gefühl etwas langsamer voran zukommen, aber ich kam mit den Aufgaben besser zurecht. Vor allem erlangte ich mehr Sicherheit in dem, was

11 HAW Hamburg 2021, S. 6,

12 HAW Hamburg 2021, S. 27

ich schrieb. Mein Verständnis wurde durch den Copilot unterstützt und Fragen wurden verständlich beantwortet." (Zitat aus der Studierendenbefragung)

Interessanterweise werden auch die Zunahme der Arbeitsgeschwindigkeit und die Zunahme des Code-Verständnisses zwar mehrheitlich positiv bewertet, aber insgesamt deutlich differenzierter betrachtet – wie neben der statistischen Auswertung auch das nachfolgende Zitat verdeutlicht:
„Für das eigentliche Lernen ist es aus meiner Sicht maßgeblich, sich den Code natürlich auch erklären zu lassen, um diesen zu verstehen. Bedarf also schon Eigenverantwortung, nicht den Code nur zu copy pasten." (Zitat aus der Studierendenbefragung)

Aus der Evaluation lässt sich auch erkennen, dass der Copilot in allen Bereichen der Code-Literacy zum Einsatz kam. Er wurde zur Erklärung von Codezeilen genauso genutzt wie zur Fehlersuche oder zur Generierung von Code. Interessanterweise im höheren Semester deutlich weniger zur Erklärung von Code, was vermutlich an der bereits vorhandenen Grundkompetenz aus den ersten Semestern liegt (siehe Abbildung 3)

von der programmiersprachlichen Ausbildung. Das zeigt auch das folgende Zitat:
„Es ist eine große Vereinfachung, aber ich persönlich finde es besser, die Aufgaben selbstständig zu erarbeiten. So ist ein besseres Verständnis dafür da, wie der Code funktioniert, wie man ihn schreibt, etc. Es ist aber gut zu wissen, wie Copilot funktioniert, wenn man mal feststeckt und allein nicht weiterkommt. Es ist gut, das zu kennen und zu wissen, dass Copilot da ist, aber richtig lernen und verstehen kann ich besser ohne Copilot." (Zitat aus der Studierendenbefragung)

Nicht nur an dieser studentischen Einschätzung lässt sich feststellen, dass die Studierenden den Copilot nicht einfach nur einsetzen, sondern sich der Konsequenzen des Einsatzes im Hinblick auf den eigenen Lernprozess sehr bewusst sind. Mehrere Studierende weisen darauf hin, dass die Nutzung des Copilots dazu verleiten kann, Lösungen zu übernehmen, ohne sie vollständig zu verstehen. Dies deutet auf eine mögliche Verschiebung von tiefem zu oberflächlichem Lernen hin. Insbesondere das Verständnis algorithmischer Zusammenhänge scheint eventuell nicht in gleichem Maße zu wachsen wie die Fähigkeit, funktionierenden Code zu erzeugen – was die

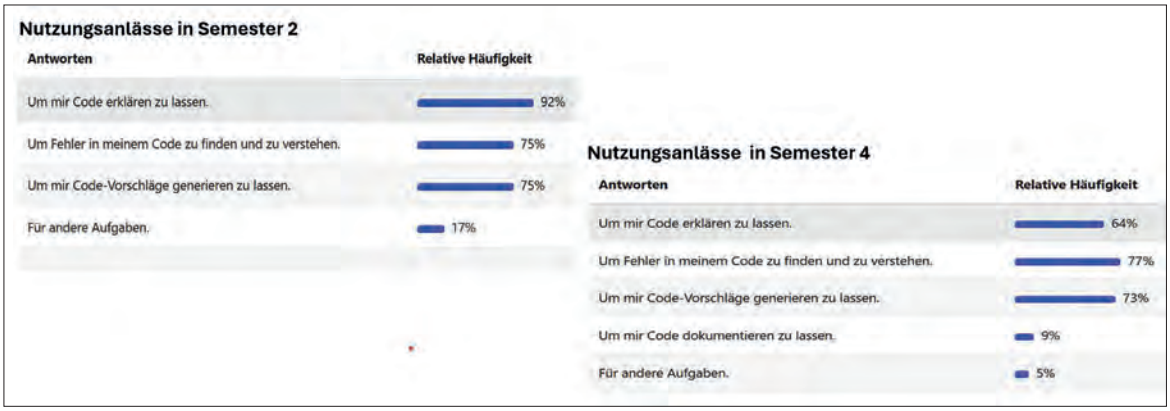

Abbildung 3: Nutzungsanlässe für den Einsatz von Copilot

Die Studierenden berichten insbesondere von einer erhöhten Fähigkeit, Code zu lesen und zu verstehen. Gleichzeitig sinkt die Hemmschwelle, selbst Code zu schreiben oder bestehende Programme zu modifizieren.
Grundsätzlich bewerten über 73% der Studierenden im zweiten Semester den Einsatz des Copiloten als sehr sinnvoll und immerhin noch 9% als sinnvoll. Bei den Studierenden des vierten Semesters ist die Zustimmung ebenfalls bei über 80%, allerdings verschiebt sich das Bild, weil hier nur noch 42% es für sehr sinnvoll und 38% es nur für sinnvoll halten. Das mag ebenfalls an der bereits vorhandenen Grundkompetenz liegen oder aber an einer gestiegenen Reflexionsfähigkeit in Bezug auf den unterstützenden Einsatz von KI-Technologie – unabhängig

Einschätzung von Wienkop und Burk (2025) in Bezug auf eine der möglichen Auswirkungen in Bezug auf Nutzung von LLMs in der Programmierausbildung bestätigt.[13]
Trotz dieses möglichen Risikos in Bezug auf einen Kompetenzverlust im tiefergehenden algorithmischen Verständnis zeigt z.B. der Notendurchschnitt im zweiten Semester mit 2,19 im Semester mit Copilot im Vergleich zum Semester davor mit 2,39 ohne Copilot, dass die in der Veranstaltung geforderten Kompetenzziele mindestens im gleichen Maß erreicht wurden.

Diskussion und Fazit

Die Evaluationsergebnisse der Lehrveranstaltungen legen nahe, dass KI-gestützte Chatbots die Programmierdi-

13 Vgl. Wienkop & Burk 2025

daktik auch in Nicht-Informatik-Studiengängen grundlegend verändern. Die Studierenden werden viel schneller ermächtigt, auch komplexere Programme zu schreiben, und sind damit in der Lage, fachlich motivierte Fragestellungen z.B. in der automatisierten Erschließung selbstständig und mit vertretbarem technischem Aufwand bearbeiten zu können.

Auffällig ist dabei die gesteigerte Selbstwirksamkeit. Viele Studierende geben an, sich durch die Unterstützung der KI sicherer im Umgang mit Programmierung zu fühlen und Aufgaben eigenständiger bewältigen zu können. Dabei wird vor allem das dialogische Moment als besonders hilfreich beschrieben.

Mit dem Chatbot an ihrer Seite müssen Studierende weniger lernen, Code selbstständig zu schreiben, sondern vor allem, die Qualität und Angemessenheit von KI-generierten Lösungen kritisch zu beurteilen. Durch das Pairprogramming mit dem Chatbot wird die Programmierung weniger als rein technische Fertigkeit vermittelt, sondern zunehmend als kommunikative Praxis gelebt. Der Fokus verschiebt sich von der korrekten Anwendung von Syntax hin zur Formulierung von Problemen und zur Bewertung von Lösungen. Das ist für Fächer, die die Programmierung als Mittel zum Zweck betrachten, von Vorteil.

Mittelfristig kann das Erlernen des Einsatzes solcher Technologien im Studium dazu beitragen, die Kluft zwischen fachlichen und technischen Rollen in der Berufspraxis zu verringern. Bibliothekar:innen könnten stärker als Vermittler:innen zwischen fachlichen Anforderungen und technischen Lösungen agieren und so neue Formen der Zusammenarbeit etablieren.

Wenn es für die Bibliothekar:innen kein Problem mehr darstellt, im Dialog mit der KI z.B. einen lauffähigen Prototyp als Demonstration der fachlichen Anforderungen gegenüber den eigentlichen Entwickler:innen zu bauen, dann trägt das zu einer Kommunikation auf Augenhöhe in bibliothekarischen Entwicklungsprojekten bei. Und das kann für den Projekterfolg nur von Vorteil sein.

Die vorliegenden Ergebnisse stellen natürlich nur einen ersten empirischen Beitrag zu einem sehr dynamischen Untersuchungsgegenstand dar. Sowohl die Technologie als auch der Umgang mit ihr ist aktuell einem sehr schnellen Wandel unterworfen. Weitere Untersuchungen in anderen Studiengängen und auch Wiederholungen im Verlauf der Zeit sind wünschenswert, um langfristige Effekte zu analysieren und belastbare Empfehlungen für die Ausbildung zu entwickeln.

Quellen

Alle Internetquellen wurden zuletzt am 28.04.2026 geprüft.

Georgy, U. (2019). Digitale Kompetenzen – dringend gesucht. Eine Stellungnahme und Positionierung zu den Empfehlungen des RfII – Rat für Informationsinfrastrukturen. In: b.i.t.online, 22(5), 405-411.

Gläser, C. (2020) Wer spricht die Sprache der Daten? – Data Literacy in der Lehre am Department Information. In: API Magazin 1(2). DOI10.15460/apimagazin.2020.1.2.48

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., Zander, C. (2011). Pair programming in education: a literature review. In: Computer Science Education, 21(2), 135-173. https://doi.org/10.1080/08993408.2011.579808

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. In: ACM computing surveys (CSUR), 37(2), 83-137.

Lücht, M. (2019). Lernraum Öffentliche Bibliotheken – Impulse zur Entwicklung Digitaler Kompetenz. Bachelorarbeit. HAW-Hamburg.

HAW Hamburg (2021). Modulhandbuch des Bachelor-Studiengangs Bibliotheks- und Informationsmanagement. Online verfügbar: https://www.haw-hamburg.de/fileadmin/zentrale_PDF/Modulhandb%C3%BCcher/Fakult%C3%A4t_Informatik_und_Digitale_Gesellschaft/BA_Bibliotheks%E2%80%93_und_Informationsmanagement/Modulhandbuch_BIM_2021-02-24.pdf

Schneemann, C. (2023). Digital qualifiziert – Kompetenzbereiche für die Zukunft. In: Nachhaltige Information – Information für Nachhaltigkeit (pp. 485-494). Verlag Werner Hülsbusch.

Stroud, A. T. (2025). Code literacy for information professionals: A critical evaluation of online MLIS programs. In: Journal of Education for Library and Information Science, 66(4), 309-319.

Wienkop, U., & Burk, L. (2025). LLMS in der Hochschullehre – Chancen und Herausforderungen für die Programmierausbildung. In: Tagungsband zum 6. Mint Symposium: Zukunft MINT Lehre: Was bleibt? Was kommt? Was wirkt? (S. 285-292). BayZiel.

Matthias Finck

studierte bis 2003 an der Universiät Hamburg Informatik und promovierte dort anschließend zum Thema „Usability Engineering in der Open Source Softwareentwicklung". 2007 gründete er die Firma effective WEBWORK GmbH, die sich der Beratung und Entwicklung von Open-Source-Lösungen für Bibliotheken, Bildungseinrichtungen und weiteren kulturellen Gedächtniseinrichtungen verschrieben hat. Matthias Finck hatte bis 2024 die Geschäftsführung inne und begleitet die effective WEBWORK seit 2026 als freiberuflicher Berater. Parallel dazu hat er von 2008–2013 an der Staats- und Universitätsbibliothek Hamburg gearbeitet und dort in zahlreichen IT-Projekten mitgewirkt. 2014 wechselte er an die Nordakademie, wo er eine Professur für Angewandte Informatik innehat. Seit 2022 arbeitet er an der HAW Hamburg als Professor für Informationstechnologie mit dem Schwerpunkt Webtechnologien und beschäftigt sich u.a. mit Fragen der KI-Unterstützung in der technischen Ausbildung der Bibliothekar:innen.
Matthias.Finck@haw-hamburg.de