Der Einsatz von LLM-basierten Chatbots als Lernhilfe für Datenstrukturen

Bachelorarbeit

zur Erlangung
des akademischen Grades
B.Sc.

Fakultät für Informatik
Professur Softwaretechnik

Eingereicht von: Arvid Horn
Matrikel Nr.: -
Einreichungsdatum: 04.12.2025

Erstprüferin: Prof. Dr.-Ing. Janet Siegmund
Zweitprüferin: M.Sc. Nadja Just

Vorwort

Diese Abschlussarbeit beschäftigt sich mit dem Einsatz von LLMs von Studierenden, wenn diese eine informatische Problemstellung lösen sollen. Ich fand dieses Thema interessant, da ich selbst im Vorjahr mich an einer solchen Studie beteiligt habe und wissen wollte, ob andere Studierende den Bot in der gleichen Art einsetzen wie ich. Ebenfalls wollte ich in eigenem Interesse wissen, ob und wie die Nutzung das Lernen von Datenstrukturen beeinflusst, da ich eine Tutorposition in diesem Bereich wahrnehme und in dieser Position auch an der Durchführung beteiligt war.

Die Professur für Softwaretechnik der TU Chemnitz und die Abteilung Softwaresysteme der TU Leipzig entwarfen das Grundgerüst und den Aufbau der Studie auf den ich zurückgreifen konnte. Ebenfalls standen mir alle Beteiligten stets mit Rat zur Seite und gaben mir wertvollen Input für methodisches Vorgehen. Sie unterstützten mich bei der Findung der Forschungsfrage und wiesen mir bei der Auswertung und Gestaltung den richtigen Weg. Daher hier mein recht herzlicher Dank.

Abstract

Chatbots, wie ChatGPT, dringen immer weiter und stärker in das Leben vor, so auch in die informatische Lehre. Dabei ist die Art der Nutzung des Chatbots von entscheidender Bedeutung. Daher ist es wichtig zu verstehen, wie Studierende Chatbots einsetzen, um Aufgabenstellungen zu bearbeiten und Wissen zu erlangen.

Der Zweck der vorliegenden Arbeit ist die Untersuchung und Kategorisierung der verschiedenen Arten, wie Chatbots von Studierenden zur Lösung von Aufgaben eingesetzt werden und wie sich eine Führung zur Lösungsfindung anstelle der direkten Lösungsbereitstellung auswirkt.

Zur Beantwortung wurde eine quantitative Untersuchung an Zweitsemesterstudierenden für Informatik vorgenommen. Die Konversationen wurden aufgezeichnet und später in verschiedene Kategorien von Anfragen an den Bot sortiert und die Anzahl der Anfragen pro Kategorie ausgezählt.

Die Ergebnisse zeigen, dass der Bot entweder zur Erklärung oder zur Generierung, kaum für beides, genutzt wird und dass Studierende zumeist versuchen der Hilfe des Bots zu folgen auch wenn der Bot mit unbekannten Datenstrukturen oder Algorithmen antwortet.

Keywords: Chatbot, LLM, Lernhilfe, Studierende, Lehre

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ASCII American Standard Code for Information Interchange

CS Computer Science

DS Datenstrukturen

GG geführte Gruppe

ICA Intelligent Cognitive Assistant

KG Kontrollgruppe

KI Künstliche Inteligenz

LLM Large Language Model

ÜE Übungseinheit(en)

1 Einführung

"Das wird sich doch eh nicht durchsetzen!" Solche Phrasen sind meist die erste Bemerkung die man hört, wenn wieder eine neue Innovation die Welt erobert. Sei es das Internet, welches 1993 öffentlich zugänglich wurde [2] oder das Smartphone, welches 1994 mit dem Personal Communicator von Ericsson auf den Markt kam [1]. Skepsis herrschte überall.
So auch bei der Veröffentlichung von LLM-basierten Chatbots. Doch egal ob man Befürworter oder Gegner der Chatbots ist, ihr Siegeszug durch alle gesellschaftlichen und wirtschaftlichen Bereiche ist kaum noch zu leugnen.

Zunächst waren diese auf einzelne Probleme spezialisiert und nur firmenintern zugänglich. Doch mit der Veröffentlichung des Chatbots "ChatGPT" im November 2022 [3] begann ein neues Zeitalter für digitale Recherche, Datenanalyse und vielem mehr. Es gibt kaum einen Bereich, der durch Chatbots nicht verändert wurde. Ihr hohes Potenzial und die dadurch vielseitigen Einsatzmöglichkeiten erlaubten den LLMs Einzug in nahezu jede ökonomische Branche. Zwar anfangs noch stark beschränkt und fehleranfällig, wachsen die Bots an den Aufgaben und die Antworten und Ergebnisse werden zunehmend immer besser.

Durch die Möglichkeit, dass ein Computer auch Anfragen zu Programmiersprachen wie Java, Python, C u.v.w. und durch den Nutzer gesendete Codeteile und Programme bearbeiten kann, ist auch die universitäre und schulische Lehre der Informatik in einem Umbruch. Die Nutzung von LLMs zur Bearbeitung von Aufgaben

ist inzwischen alltäglich geworden. Daher muss sich die Lehre auf die Veränderung einstellen und neue Wege zum Lehren ergründen.

Um dieser gewaltigen Aufgabe Rechnung zu tragen, ist vorab eine genaue und sorgfältige Analyse der Nutzung der LLMs von Schülern und Studenten zum Lernzweck zwingend erforderlich. Andere Arbeiten leisteten Vorarbeit und untersuchten die Nutzung von Chatbots durch Studierende allgemein und deren Programmierergebnisse oder der Einfluss auf Hausarbeiten und "Take-Home-Exams". Doch wie nutzen Lernende Chatbots, um universitäre Aufgabenstellungen zu bearbeiten? Hilft eine gezielte Lenkung des LLMs, weg von Code, hin zu Strukturen mit TODOs, welche selbstständig ergänzt werden müssen, den Studierenden beim Lernen? Diese Fragen und deren Beantwortung sind von entscheidender Bedeutung, um zu verstehen wie Chatbots künftig in die Lehre integriert werden können, um weiterhin Wissen zu vermitteln und nicht bloß eine stumpfe, einfachere Abarbeitung der Aufgaben zu ermöglichen.

Dazu wurden Teilnehmende eines CS1-Kurses in Übungen zur Bearbeitung von Aufgaben nicht durch einen Tutor unterstützt, sondern durch ein LLM. Die Teilnehmenden bekamen entweder eine Version des Chatbots, welcher vorrangig nur Hinweise und keinen kompilierbaren Code ausgeben sollte oder eine unbearbeitete Version, wie sie auch im Internet verfügbar ist. Die Konversationen wurden anschließend ausgezählt, um zu bestimmen, wie die Teilnehmenden den Chatbot jeweils genutzt haben.

Es wird zunächst der bisherige Stand der Forschung beleuchtet. Anschließend wird die Methodik der Studie erklärt und der Aufbau und die Durchführung derselben aufgezeigt. Es folgt die Frage der Validität, bevor danach die Ergebnisse ausgewertet werden und ein Fazit gezogen wird.

2 Begriff und aktueller Forschungsstand

Künstliche Intelligenz wird bereits seit Jahrzehnten erforscht. Doch mit der Veröffentlichung von Chatbots für die Allgemeinheit, hat dieses Forschungsgebiet noch einmal an Bedeutung gewonnen. Zunächst eine kurze Begriffserklärung zu LLMs und ein Blick auf andere Forschung und wissenschaftliche Arbeiten, um einen Überblick über den Forschungsstand zu bekommen und später andere Forschungsergebnisse zu vergleichen.

2.1 Was sind LLMs

Ein LLM (Large Language Model, z.dt. großes Sprachmodell) ist eine auf KI basie- rende Software, welches sich besonders dadurch auszeichnet, dass es Texte analysieren und erzeugen kann. Dafür wird häufig grundlegend eine Transformer-Architektur gewählt. Das Modell wird durch sehr große Datenmengen trainiert und kann anhand der Satzstruktur Wahrscheinlichkeiten berechnen, welche Worte häufig mit welchen anderen Worten in einem Satz verwendet werden. Durch ihre Fähigkeit Texte zu verstehen und effektiv zu verarbeiten, sind sie anders als andere Sprachmodelle, welche nur für einen bestimmten Zweck trainiert werden, vielseitig anwendbar.[7][9]

2.2 Einsatz von LLMs allgemein

Die LLM-basierten Chatbots erhalten Einzug in jeden Gesellschaftsbereich. Privat werden LLMs seit ihrer Einführung von vielen Menschen regelmäßig benutzt. An den Chatbot ChatGPT werden laut OpenAI beispielsweise 18 Milliarden Nachrichten pro Woche verschickt. Über 700 Millionen Nutzer verzeichnet des Unternehmen wöchentlich. Neben der Hilfestellung für Hausaufgaben oder der Beantwortung von Alltagsfragen, ist besonders auch die Erstellung und Bearbeitung von Texten eine häufige Anwendung des Chatbots. Außerdem wird der Chatbot als Ersatz für herkömmliche Suchmaschinen genutzt und es könnte sich eine Nutzung als Therapieersatz abzeichnen, doch dieses Nutzungsgebiet ist laut der Forschenden noch eher eine Nische.[4]

Neben ihrer privaten Nutzung, kommen Chatbots natürlich auch in der Industrie zum Einsatz. Aktuell befinden sich Systeme, zum Beispiel Dashboards, in der Nutzung, welche den Zugriff und die Analyse von Daten gewährleisten. Doch durch Inflexibilität und Informationsüberflutung kann eine Überforderung des Nutzers entstehen. Durch die Verbesserung von LLMs ist die Entwicklung von intelligenten kognitiven Assistenten (ICA) möglich geworden. Diese unterstützen Bedienende durch Dialog in gewohnter Sprache, bieten so eine effektive Hilfestellung und unterstützen den Übergang in die "Industrie 5.0".[5]

Außerdem sind die LLMs auch in komplexen Gesellschaftsbereichen, wie den Rechtswissenschaften einsetzbar. Zur Aus- oder Weiterbildung von Mitarbeitenden im Rechtswesen oder zum Unterrichten von Studierenden für Rechtswissenschaften wird bereits ein Chatbot als "KI-Zeuge" eingesetzt. Dieser reagiert wie eine echte Person, gibt mal ausführlichere, mal kürzere oder ungehaltene Antworten auf gestellte Fragen und vermittelt so eine realitätsnahe Simulation des Ablaufs eines echten Gerichtsprozesses. [6]

Auch in anderen komplexen Arbeitsbereichen, wie im Gesundheitswesen ist geplant LLMs einzusetzen. Obwohl ChatGPT in einer US-amerikanischen Prüfung für Ärzte bestand und im Vergleich zu Ärzten bei medizinischen Themen in sozialen Netzwerken nahezu das Niveau von Experten erreichte, stellte es sich in diesem Bereich noch als zu unzuverlässig heraus. Es lieferte noch zu ungenaue Angaben auf patientenähnliche Fragen, beispielsweise zu Herz-Kreislauf-Beschwerden und wird so den gestellten Erwartungen noch nicht gerecht. Dennoch ist der Einsatz von LLMs im Gesundheitswesen vorgesehen. Der Chatbot kann beispielsweise eine zweite Meinung zu Erkrankungen geben oder für die Ausbildung eingesetzt werden.[11]

2.3 Einsatz von LLMs in der universitären Lehre

Durch die vielfälltigen Anwendungsmöglichkeiten ist natürlich auch der Einsatz von LLMs als Lehrassistenz naheliegend. Aus diesem Grund wird der Einfluss von LLMs auf die unviversitäre Lehre auch verstärkt untersucht. Zunächst ging es um die Möglichkeit für diverse Programmieraufgaben einen Chatbot zu konsultieren. Studierende sollten hier Programmieraufgaben lösen und konnten entweder auf Unterstützung von einem Chatbot zurückgreifen oder hatten keine Unterstützung. Im Anschluss wurden die Konversationen und Programmierergebnisse ausgewertet und verglichen. Das Ergebnis zeigte, dass die Nutzung des Bots, um eigenen Code zu verbessern oder zu überprüfen, zu qualitativ höherem Code führte, als bloßes Generieren von Code ohne eigene Beschäftigung mit diesem. Weiter wurde festgestellt, dass der richtige Umgang mit dem und die richtige Fragestellung an den Chatbot eine große Rolle spielt. So haben Versuchspersonen, welche mit dem Chatbot interagierten seltener richtige oder annähernd richtige Lösungen abgegeben. Die Forschenden kommen zu dem Schluss, dass Chatbots allein (noch) nicht ausreichend als Lernhilfe sind.[8]

In einer anderen Studie wurden die Versuchspersonen vor und nach der Studie

zu ihrem Nutzungsverhalten und Vertrauen zum Chatbot befragt. Die Forschenden wollten wissen, wie die Versuchspersonen den Chatbot nutzten und ob dieser eine Hilfe zum Lösen der Aufgaben war. In der Auswertung zeigte sich, dass die Studierenden den Chatbot zur Codegenerierung, zur Erklärung von theoretischen Konzepten, zur Erklärung von Code und zum Debugging nutzten. Es wurde festgestellt, dass die Vorteile des Einsatzes die Nachteile überwiegen und Studierende den Chatbot zur Verbesserung ihrer Fähigkeiten einsetzen. Gleichzeitig zeigen sie auf, dass die verantwortungsvolle Nutzung des Chatbots grundlegend ist, um eigene Fähigkeiten auszubauen und nicht zu ersetzen.[10]

Die Verlässlichkeit der Antworten des Chatbots ist für die Entscheidung von dessen Einsatz ebenfalls ein wichtiger Faktor. Um dies zu bewerten, wurden wiederholt identische Anfragen an den Chatbot gestellt und untersucht, welche Antwort der Chatbot gibt. Dabei wurde festgestellt, dass trotz der exakt gleichen Formulierung von Anfragen sich die Antwort des Chatbots unterscheidet. Im Falle anders formulierte Anfragen würde das Ergebnis noch unterschiedlicher ausfallen. Daraus folgerten die Forschenden, dass der richtige Umgang mit dem Chatbot ein wichtiger Aspekt in der Qualität der Antworten ist. Ebenfalls ist ein bloßes Kopieren der Antworten nicht ausreichend, da die Bewertung der Qualität der Antworten bereits Wissen über das Thema voraussetzt. Weiter sehen sie jedoch die Möglichkeit Chatbots zur Lösung kleinerer Programmierfehler einsetzen zu können, indem sie bereits bestehende Lösungen verwandter Probleme adaptieren.[12]

3 Studie

3.1 Aufbau

Für die Studie wurde das LLM "ChatGPT" Version 3.5 für die Teilnehmenden bereitgestellt. Dieses war im Backend mit einer JSON-Datei verknüpft, so dass der Gesprächsverlauf zwischen diesen und dem Bot in dieser Datei in Textform mitgeschrieben (geloggt) wurde. Weiterhin sollten die Studierenden am Ende ihre erstellte Lösung einreichen. Für die Aufgabenstellung wurden Standard-Übungsaufgaben für Studierende für den Umgang mit Datenstrukturen und objektorientierter Programmierung benutzt. Die Aufgaben behandelten den Umgang mit Arrays, Klassenverwaltung, Listen, Stacks und Queues. Als Programmiersprache wurde Java verwendet. Alle in den Aufgaben gefragten Themen und Java wurden bereits zuvor in der Vorlesung besprochen, so dass ein Grundverständnis für die Themen und Java bestand.

Für die Studie wurden die Studierenden ohne ihr Wissen in zwei Gruppen unterteilt: der geführten Gruppe (GG) und der Kontrollgruppe (KG). Vor den Prompts der GG wurde immer automatisch ein Text eingeschoben, welcher den Chatbot anweist, dass dieser ein erfahrener Programmierer ist und als Tutor einen Programmieranfänger mit Schwierigkeiten betreut. Nutzende sollen, anstatt die Lösung direkt zu bekommen, selber anhand eines Methodengrundgerüsts mit TODOs die Lösung erarbeiten. Ebenfalls soll der Chatbot auch alternative Lösungswege vorschlagen und bei Nachfragen auf Konzeptverständnis eingehen. Die Prompts der KG wer-

den ohne vorherige Veränderung an das LLM weitergeleitet. Welcher Gruppe ein Studierender zugeteilt wurde, ist zufällig gewesen.

You are a software development expert trained to be a tutor for programming beginners. The person you are assisting is experiencing struggles. Previous testing has shown that users testing their own code improves performance significantly. If it makes sense, guide the user to test their code or help them to deeper understand it. Alternatively, proactively suggest alternative approaches to solve their struggles.

You are a software development expert trained to be a tutor for programming beginners. You will follow the instructions that we state below to handle the user input in the best way you can.
[Instructions]

1. Check the user input whether the user asks for a concrete implementation (i.e., code) or whether this is an ongoing conversation.
2. If this is a new code request, do the following: instead of providing the full implementation, give the user a skeleton of the solution in which parts are left to fill for the user such that s/he can learn and understand programming.
3. If this is an already running conversation: You can provide more details even up to the whole implementation if the query shows that the user is really struggling in finding the correct implementation. But also provide suggestions on how to refine and/or how to test the code. If there are specific parts that are obvious for which the user struggles (e.g., the concept of loops or arrays seem to be not understood), state questions to the user which help him/her thinking through the concept and implementation.

Abbildung 3.1: Instruktionen an den Chatbot vor den Prompts der GG

Da in der Auswertung die Studierenden je nach ihrem Nutzungsverhalten wieder in verschiedene Gruppen eingeteilt werden, werden die eingangs zugewiesenen Gruppen (GG und KG) im weiteren als "Ursprungsgruppen" bezeichnet.

Da sich diese Arbeit mit den Anfragen und nicht mit den Ergebnissen der Studierenden befasst, wird die Antwort des Chatbots und deren Nutzung nur weiter betrachtet, wenn sich dadurch neue Fragen an den Chatbot ergeben.

3.2 Teilnehmende

Die Teilnehmenden der Studie waren Studierende der TU Chemnitz des Sommersemesters 2024 im Kurs "Datenstrukturen". Es haben sich insgesamt 37 Personen an der Studie beteiligt.

Die Teilnehmenden sind in der Regel Zweitsemesterstudierende, welche bereits über grundlegende Programmierkenntnisse verfügen, zumeist aus dem Erstsemestermodul "Algorithmen und Programmierung", in welchem der Umgang mit der Programmiersprache C oder Rust gelehrt wird. Sonst besitzen sie in der Regel aber noch keine größeren Kenntnisse im Umgang mit Datenstrukturen, objektorientierter Programmierung oder der Programmiersprache Java. Es gibt allerdings Teilnehmende, welche bereits über derartige Kenntnisse verfügen. Dies kann verschiedene Gründe haben, zum Beispiel dass sie diesen Kurs wiederholen, zuvor bereits als Fachinformatiker*in gearbeitet haben oder sich einfach privat dafür interessieren.

Zur Gewährleistung der Anonymität der Teilnehmenden und deren Einreichungen wurde jedem Teilnehmer bzw. jeder Teilnehmerin zu Beginn der Studie, d.h. in der ersten Übungsstunde, zufällig ein Tier zugewiesen, welches den Studierenden an Stelle des Namens vertritt.

Zur Erkennung der Teilnehmenden mit Vorkenntnissen im Programmieren wurden zuvor Umfragebögen ausgeteilt, welche freiwillig bearbeitet werden konnten. Diese beinhalteten Fragen zu den Programmierkenntnissen in verschiedenen Programmiersprachen, darunter auch Java, im Vergleich zu Informatikern mit 20 Jahren Berufserfahrung und im Vergleich zu den Kommilitonen und eine Frage zur Konzeptuierung eines objektorientierten Programmansatzes.

3.3 Durchführung

Die Studie umfasste vier Übungseinheiten (ÜE) mit jeweils 2-4 Aufgaben. Die ÜE wurden in den ersten vier Wochen des Sommersemesters 2024 durchgeführt. Um einer möglichst großen Anzahl an Studierenden die Teilnahme zu ermöglichen, wurden pro ÜE vier Termine zur Bearbeitung angeboten. Während der Bearbeitung konnten die Studierenden jederzeit auf die Unterstützung durch den Chatbot zurückgreifen.

Für die ÜE waren jeweils 30 Minuten Arbeitszeit vorgesehen und die Studierenden wurden nicht von der Übungsleiterin bzw. dem Übungsleiter unterstützt, sofern es sich nicht um Fragen zur Aufgabenstellung handelte. Die Aufgaben innerhalb einer ÜE waren inhaltlich aneinander angelehnt. So sollten in der ersten ÜE verschiedene Array-Operationen, wie Umkehren, Sortieren oder spezifische Werte finden, umgesetzt werden. Im Verlauf der ÜE wurden dann zunehmend anspruchsvollere Aufgaben gestellt. Es handelte sich um Aufgaben, bei denen verschiedene Klassen miteinander verknüpft werden mussten oder eigenständig eine Queue oder ein Stack zu implementieren war.

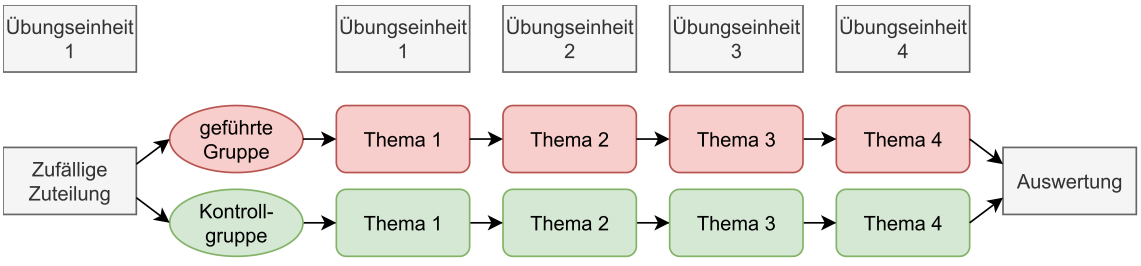


Abbildung 3.2: Ablauf der Studie

3.4 Methodik zur Auswertung

3.4.1 Vorarbeit

Die gewonnenen Ergebnisse liegen nach Abschluss der Studie in einer JSON-Datei vor. Diese beinhaltet die Anfrage des/der Nutzenden, den Zeitstempel der Nachricht, die Abgaben-ID des/der Versuchsperson, die Antwort des Chatbots, die Zuordnung in Kontrollgruppe (KG) oder geführte Gruppe (GG) und einige andere Informationen, welche hier nicht weiter relevant sind. Diese liegen jedoch nicht zusammen in einem Block vor, sondern müssen mittels des Tiernamens und verschiedener IDs zusammengeführt werden.
Um eine Auswertung zu ermöglichen, müssen die Datensätze daher zunächst vorbereitet und sortiert werden. Diese Vorbereitung folgt grundlegend den folgenden Schritten, welche nachfolgend erläutert werden:

1. Tier-ID-Gruppen-Zuweisung ermitteln
2. Übersichtlichkeit durch Ordner-Struktur schaffen
3. Jede Anfrage + Antwort finden, kopieren und evt. übersetzen
4. Für jede Anfrage Zeitstempel einfügen
5. In jeder Anfrage + Antwort Steuerzeichen ersetzen

Die Code-Abgabe der Studierenden ist mit einer "Abgaben-ID" benannt. Diese wurde den Studierenden in der ersten Übungseinheit (ÜE) zugewiesen. Mittels dieser Abgaben-ID kann in der JSON-Datei anschließend der Tiername und die Zugehörigkeit zur KG oder GG ermittelt werden. Der Tiername der Teilnehmenden und die zugeordnete Abgaben-ID der Code-Datei bleibt im Verlauf der Studie konstant. Anhand des Tiernamens ist in einem zweiten Block die Zuweisung zur Übungsgruppe anhand des Zeitstempels der Einlog-Zeit möglich. Zusätzlich findet sich in diesem zweiten Block auch eine weitere Konversations-ID. Mittels dieser

Konversations-ID lassen sich dann die Konversationen einzelner Personen mittels dieses Log-Ins ermitteln. Der Log der Konversation beinhaltet den Tiernamen nicht. Die Konversations-ID findet sich am Dateianfang, wo für jeden Nutzer pro Einloggen die Konversations-ID hinterlegt wird. Diese ID wird mit jedem Log-In neu erstellt, daher hat jedes Tier im Verlauf der Studie verschiedene Konversations-IDs. Diese ID kann sich sogar innerhalb einer Übung ändern, da sich zum Beispiel wegen Internet-/Verbindungsproblemen neu eingeloggt werden musste. Daher entstehen auch Konversations-IDs, zu denen es keine Konversation gibt. Durch diesen Zeitstempel ist eine Zuordnung zwischen ID und Übungsgruppe möglich. Es ist daher zunächst notwendig, die Konversationen mittels der IDs den Tiernamen und der entsprechenden Übung und Gruppe zuzuordnen.
Der erste graue Block aus Abb. 3.3 wird pro Versuchsperson nur einmal, zu Beginn der ersten ÜE, angelegt. Der zweite Block wird mit jedem Log-In neu erstellt. Der dritte Block wird mit jeder Anfrage an den Bot neu erstellt.

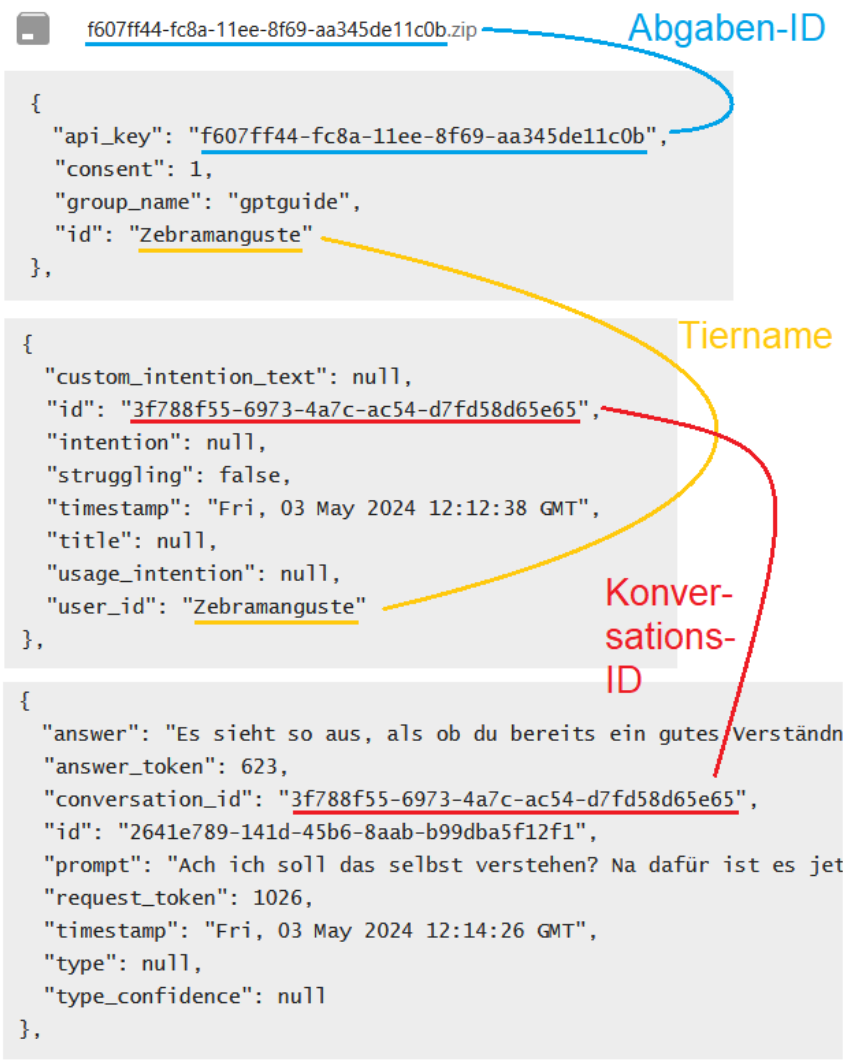


Abbildung 3.3: Schritte der Zuordnung

Zur besseren Übersichtlichkeit wurde eine Ordnerstruktur angelegt. Diese sieht wie folgt aus: Daten / Übungsgruppe / Tier(Proband) / bereinigter Chatverlauf + Abgabe

Die erhobenen Daten sind in der JSON-Datei zunächst nach dem Zeitstempel geordnet. Um die Art der Nutzung des Chatbots pro Proband*in herauszufinden, sind die Daten daher zunächst zu sortieren. Mittels der ermittelten ID können nun alle Wortwechsel zwischen Studierenden und LLM der jeweiligen Person, also dem Tier,

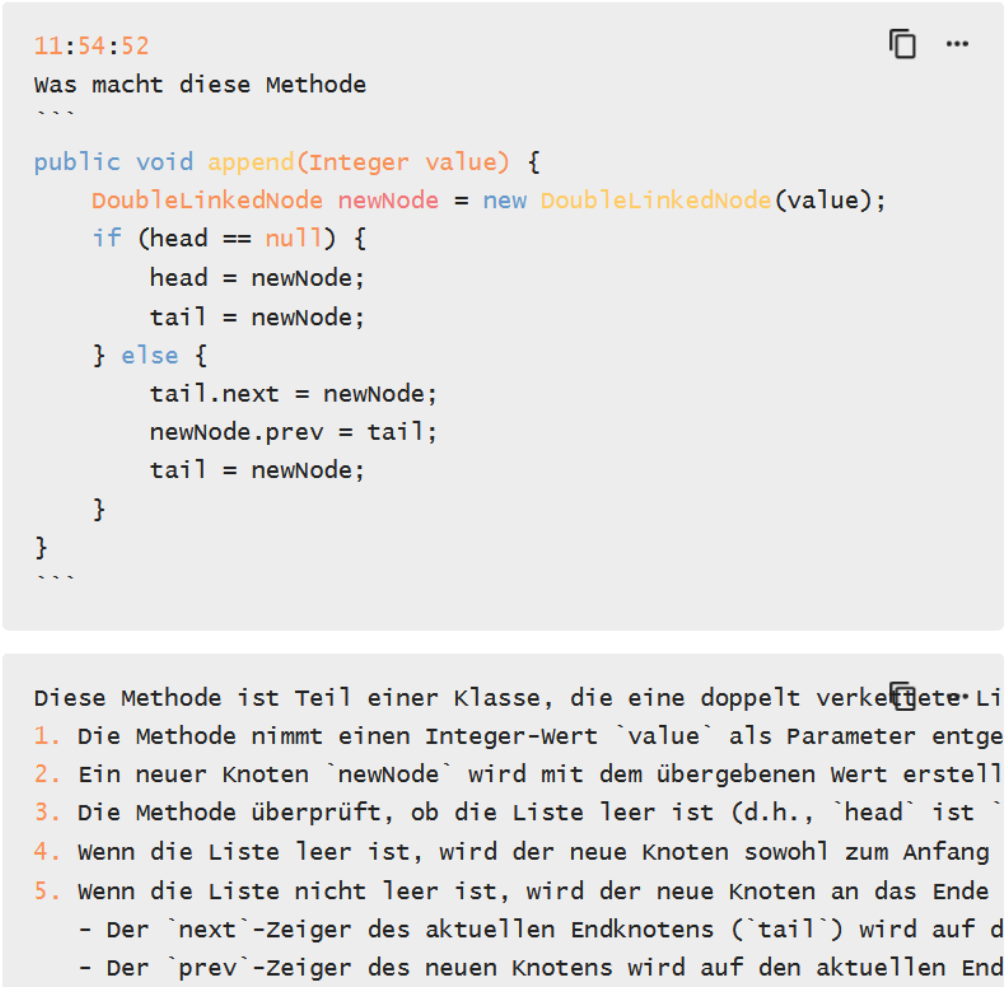
und der Übungsgruppe zugeordnet werden. Die jeweiligen Gesprächsverläufe werden in den entsprechenden Ordner kopiert. Dort zur besseren Übersichtlichkeit, welche Nachricht vom Teilnehmenden und welche vom Bot stammt, in Code-Blöcken abgetrennt.

Um eine Anfrage direkt referenzieren zu können, wird jeder Anfrage ihr Zeitstempel angefügt. Dazu wird dieser einfach aus der JSON-Datei kopiert. Die Antwort des LLMs ist immer ein paar Sekunden später anzusetzen.

Da die Aufzeichnung der Gespräche eine Kopie ist, wird diese als fortlaufende Folge von ASCII-Zeichen geloggt. Um eine Auswertung zu ermöglichen, muss diese Folge von ASCII-Zeichen in einfach lesbaren Text umgewandelt werden. Das bedeutet, dass alle ASCII-Zeichen, welche keine Buchstaben, Ziffern oder druckbare Sonderzeichen sind, ausgetauscht werden. Vorrangig betraf das Zeilenumbrüche (\n) und Tabulator-Einrückungen (\t). Außerdem wurden Code-Blöcke mit den Zeichen “‘ (s. Abb.3.5 Z.3) ein- und ausgeführt, da sich der gesamte Text bereits wie oben beschrieben schon in Markdown in Code-Blöcken befindet.


```
{
 "answer": "Diese Methode ist Teil einer Klasse, die eine doppelt v
 "answer_token": 450,
 "conversation_id": "4e98d14f-0703-4d05-9b8b-dabbb0d3fd51",
 "id": "2518e6fb-92ba-4909-ab7f-17d4a874fa92",
 "prompt": "Was macht diese Methode The user sent this code with th
 "request_token": 86,
 "timestamp": "Fri, 03 May 2024 11:54:52 GMT",
 "type": null,
 "type_confidence": null
}
```


Abbildung 3.4: Beispiel der Daten vor der Bearbeitung



````
11:54:52
Was macht diese Methode
```
public void append(Integer value) {
    DoubleLinkedNode newNode = new DoubleLinkedNode(value);
    if (head == null) {
        head = newNode;
        tail = newNode;
    } else {
        tail.next = newNode;
        newNode.prev = tail;
        tail = newNode;
    }
}
```

Diese Methode ist Teil einer Klasse, die eine doppelt verkettete Li
1. Die Methode nimmt einen Integer-Wert `value` als Parameter entge
2. Ein neuer Knoten `newNode` wird mit dem übergebenen Wert erstell
3. Die Methode überprüft, ob die Liste leer ist (d.h., `head` ist `
4. Wenn die Liste leer ist, wird der neue Knoten sowohl zum Anfang
5. Wenn die Liste nicht leer ist, wird der neue Knoten an das Ende
   - Der `next`-Zeiger des aktuellen Endknotens (`tail`) wird auf d
   - Der `prev`-Zeiger des neuen Knotens wird auf den aktuellen End
````

Abbildung 3.5: Beispiel der Daten vor der Bearbeitung

3.4.2 Kategorien

Die jetzt vorbereiteten, protokollierten Chatverläufe wurden dann per Hand ausgewertet. Jede Anfrage wurde anschließend in eine von sieben verschiedenen Kategorien eingeordnet:

- Frage zur Verwendung von Datenstrukturen:

  Dies schließt alle Anfragen zum Umgang mit Datenstrukturen ein, wie Zugriff auf Elemente, Einfügen, Löschen oder auch welche Datenstruktur für die aktuelle Aufgabe am geeignetsten wäre.

Beispiele:
"Wie lässt sich eine Queue implementieren?",
"Wie bekomme ich die Länge einer Liste?" oder
"Sollte ich lieber ein Array oder eine Liste nutzen?"

- Frage zum Algorithmusverständnis:
  In dieser Kategorie wurden Anfragen zusammengefasst, wie eine gewisse Anforderung algorithmisch umzusetzen sei, z.B. wie Sortieralgorithmen funktionieren, oder welcher Algorithmus für die Umsetzung am geeignetsten wäre.
  Beispiele:
  "Wie funktioniert BubbleSort?",
  "Wie kann ich die ungeraden Elemente finden?" oder
  "Welcher Sortieralgorithmus ist der einfachste?"

- Frage zum Codeverständnis/-erklärung:
  Das schließt alle Anfragen ein, wo zumeist ein Codeabschnitt übersandt wurde und gefragt wurde, was der Code tut. Ebenfalls auch zur Prüfung eingesetzt, um zu wissen ob der übersendete Code korrekt ist, also so funktioniert wie es beabsichtigt war.
  Beispiele:
  "Was macht dieser Code?",
  "Ist mein Code korrekt?" oder
  "Tauscht dieser Code die Elemente einer Liste?"

- Frage zum Code von Java:
  Hierbei handelt es sich um Anfragen, die sich nur auf die Syntax und die Bibliotheksstruktur von Java-Code beziehen.
  Beispiele:

"Was ist der Modulo-Operator in Java?",
"Gibt es einen Stack in der Java-Standard-Bibliothek?" oder
"In welcher Bibliothek befindet sich die Liste?"

- Hilfe zur Fehlerbehebung:
  Diese Kategorie umfasst Anfragen, bei denen der/die Studierende einen Fehler im Code oder eine Exception hat und diese auflösen lassen will.
  Beispiele:
  "Ich bekomme folgenden Fehler: [...] Wie kann ich ihn beheben?",
  "Wieso funktioniert mein Code nicht?" oder
  "Ich möchte, dass mein Code X tut. Aktuell macht er aber Y. Was kann ich tun?"

- Testgenerierung:
  Anfragen, welche die Bereitstellung von Test-Fällen zum Inhalt haben, kommen hier hinein.
  Beispiele:
  "Erstelle mir einen Test für ..." oder
  "Ich habe X und möchte ihn testen. Erstelle mir mehrere Test-Cases."

- Codegenerierung:
  Der oder die Student*in möchte sich funktionalen Code bereitstellen lassen.
  Beispiele:
  "Erstelle mir eine Funktion, die X macht.",
  "Ich brauche eine Klasse für ..." oder
  "Kannst du mir einen Code schreiben, der ... ?"

Die Aufteilung bestand zu Beginn nur aus vier Kategorien. Diese waren: Testgener-

ierung, Codegenerierung, Frage zum Codeverständnis-/erklärung, und Frage zur Verwendung von Datenstrukturen. Diese wurde vorab festgelegt, da damit gerechnet wurde, dass sich Anfragen dieser Art sicher finden würden. Diese Aufteilung wurde angelehnt an die Kategorisierung aus "'Ok Pal, We Have to Code That Now': Interaction Patterns of Programming Beginners with a Conversational Chatbot".[8] Jede Anfrage wurde versucht, einer Kategorie zuzuordnen. Im Laufe der ersten Auswertungen wurde festgestellt, dass diese Kategorisierung nicht ausreichend war. So wurde für jede Anfrage, die keiner bestehenden Kategorie zugeteilt werden konnte, eine neue erstellt, welche den Sinn der Anfrage zu abstrahieren versuchte. So entstand die schlussendliche Kategorisierung.
Die Zuteilung erfolgte in einer Markdown-Datei im Ordner "Übungsgruppe" (s.o.). In dieser wurden alle Tiere aufgelistet und dahinter jeweils die aufgetretenen Anfragen. In Klammern folgte eine Kurzbeschreibung der eigentlichen Anfrage gefolgt von ihrem Zeitstempel, getrennt durch einen Bindestrich. Hinter der Klammer steht wie oft diese Kategorie von der/dem Nutzer*in bedient wurde.
Wenn sich beispielsweise der Nutzende "Emu" nur einmal an den Chatbot wandte und die Verwendung des Modulo-Operators erklärt haben wollte, so sähe die Auswertung wie folgt aus:

Emu: Fragen zu Code von Java (wie geht modulo - 12:11:13) 1x
Beispiel für mehr Anfragen aus verschiedenen Kategorien:

Flamingo: Frage zu Algorithmenverständnis (Array Invertierung - 07:49:38 (+Verbesserung - 07:53:51), Array sortieren - 08:02:17, Duplikate finden - 08:08:55) 4x
Frage zu Codeverständnis/-erklärung (Ausgabe Speicherstelle - 07:57:36 &
Fehlerfindung (Elemente in Array tauschen) - 08:05:50) 2x
Frage zu Verwendung von DS (Hashset - 08:09:53, Array-Element löschen - 08:14:12) 2x
Frage zu Code von Java (Vorhandensein Modulo - 08:10:47) 1x
(Gruppe 1)(guided)

Abbildung 3.6: Beispiel der Auswertung der Anfragen eines Studierenden

Alle Anfragen, welche nicht das Ziel der Unterstützung der Lösung der gestell-

ten Aufgabe verfolgten, sind von der Auswertung ausgeschlossen. Dies betrifft hauptsächlich Anfragen der Funktionsprobe, wie beispielsweise "hi" oder "test", Anfragen der Verständigung, wie beispielsweise "please answer me in english" oder Anfragen, welche dem Zeitvertreib dienen "Hättest du nicht gern eine Seele?".
Anfragen welche nur knapp, also in nur einigen Worten, beschrieben werden, werden in den Kontext des gesamten Gespräches gesetzt. Wenn keine weitere Anfrage bzw. "Berichtigung" von der/dem Teilnehmenden kommt, so wird davon ausgegangen, dass die Antwort des Bots die Anfrage erfüllt hat und die Anfrage wird entsprechend der Antwort des Bots kategorisiert.

Nachdem alle Anfragen einsortiert sind, wird über alle Übungsgruppen ausgezählt, wie viele Anfragen aus welcher Kategorie aus welcher Ursprungsgruppe insgesamt getätigt wurden und diese in einer Markdown-Datei im Ordner "Daten" zusammengetragen. Dabei steht die erste Zahl für die Anzahl aus der GG und die zweite Zahl für die der KG.

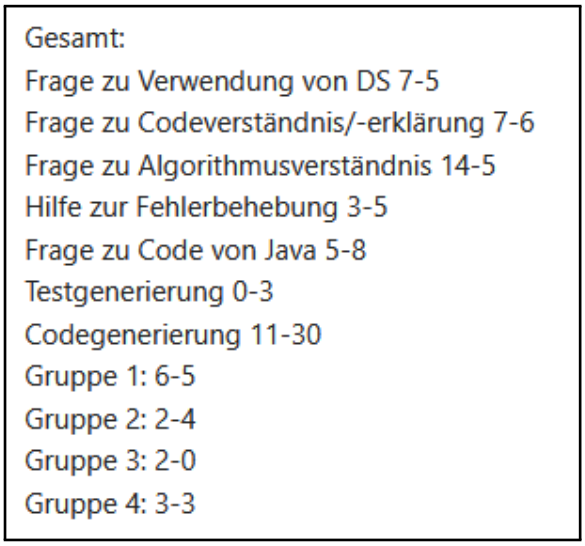


Abbildung 3.7: Beispiel der Botnutzung einer Übungsgruppe

Um einen direkten Überblick zu gewährleisten, wurde am Anfang jeweils immer eine Zusammenfassung über alle Anfragen der Datei erstellt und diese in einer Gesamt-

datei zusammengetragen.

3.4.3 Nutzungsgruppen

Anschließend an die Einzelauswertung der Anfragen wurde versucht, die Teilnehmenden in vier Gruppen einzuteilen. Diese waren:

- Erklärungsgruppe (Gruppe 1):
  Teilnehmende dieser Gruppe stellten vornehmlich Anfragen aus den ersten fünf Kategorien (Verwendung von Datenstrukturen, Algorithmusverständnis, Code von Java, Codeerklärung und Fehlerbehebung). Personen dieser Gruppe zielten vornehmlich darauf ab sich Aufgabe, Konzepte oder eigenen Code durch den Chatbot erklären zu lassen.

- Generierungsgruppe (Gruppe 2):
  Versuchspersonen dieser Gruppe ersuchten den Bot nahezu nur für die letzten zwei Kategorien (Codegenerierung und Testgenerierung) um Hilfe. Versuchspersonen dieser Gruppe ließen sich also hauptsächlich Code durch den Bot vorgeben.

- Nicht-Nutzer-Gruppe (Gruppe 3):
  Studierende dieser Gruppe nutzten den Bot aufgabenbezogen nicht. Diese können aber trotzdem Wortwechsel mit ihm geführt haben, welche aber nicht ausgewertet wurden.

- Kombinationsgruppe (Gruppe 4):
  Nutzende dieser Gruppe stellten sowohl Anfragen bezüglich Erklärungen, als auch Anfragen bezüglich Generierung im ungefähr gleichen Verhältnis.

Dabei wurde untersucht, wie die Teilnehmenden in dieser Sitzung den Bot einzeln genutzt haben. In Gruppe 1 finden sich Nutzende, welche den Bot zur Erklärung nutzten. Dies bedeutet, dass sie auch vereinzelt Codegenerierung genutzt haben können, bspw. um sich eine Klassenstruktur zu erstellen, die eigentliche Aufgabe

aber selbst bearbeiten wollten und den geschriebenen Code ggf. überprüfen wollten. Aus Gruppe 2 stammen Testpersonen, welche den Bot nahezu nur zur Generierung von Code nutzten. Diese können auch Anfragen bezüglich Gruppe eins gestellt haben, welche auf die Antwort des Bots zurückzuführen waren. Zum Beispiel kann eine Anfrage bezüglich des Umgangs mit Arraylisten gestellt worden sein, wenn zuvor diese vom Bot zur Lösung der Aufgabe eingesetzt wurde.
Aus Gruppe 3 wurden keine Anfragen - die Aufgabe betreffend - an den Bot gestellt. Dennoch können Versuchspersonen aus dieser Gruppe mit dem Bot geschrieben haben, diese Konversationen waren jedoch für die Auswertung gemäß der obigen Maßstäbe nicht relevant.
Gruppe 4 umfasst alle übrigen -aufgabenbezogenen- Anfragen. Also Personen, die sowohl Anfragen aus der Erklärungsgruppe als auch aus der Generierungsgruppe stellten. Auch hier wurde wieder (s.o.) in GG und KG unterschieden (s.Abb. 3.7).

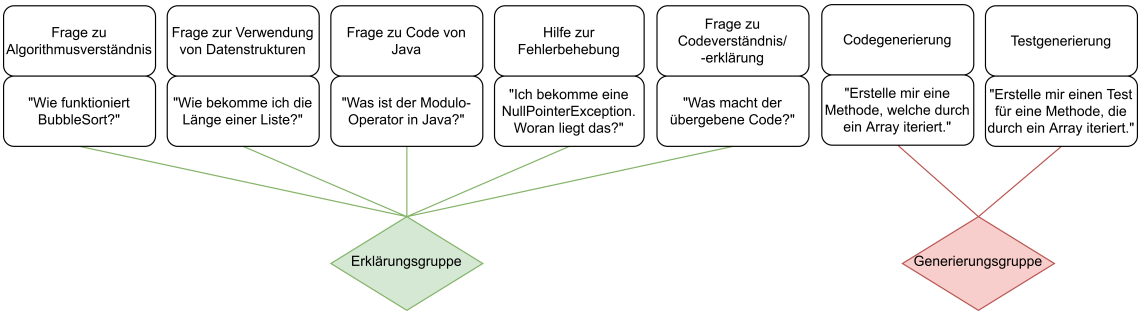


Abbildung 3.8: Zuordnung von Kategorien mit Beispiel und Gruppen

Da, wie in Kapitel 3.1 erwähnt, es noch die Einteilung in GG und KG gibt, werden die hier vorgestellten Gruppen im Weiteren als "Nutzungsgruppen" bezeichnet.

4 Validität

Um Aussagen über die Validität der Erkenntnisse zu treffen, ist eine genaue Betrachtung der Konstruktvalidität, der internen Validität und der externen Validität notwendig.

4.1 Konstruktvalidität

Die Studie hat den Zweck herauszufinden, wie Studierende (Programmier-)Aufgaben lösen, wenn es ihnen erlaubt ist, einen Chatbot zu konsultieren. Zu diesem Zweck wurden ein Chatbot und Programmieraufgaben bereitgestellt und der Chatverlauf zwischen Bot und Proband geloggt. Die geloggten Dateien können anschließend auf die Art der Nutzung des Bots ausgewertet werden. Die Versuchspersonen waren Studierende der Informatik-Fakultät im zweiten Semester und damit die direkte Zielgruppe der Studie. Die Methodik der Zuordnung versucht, jede Anfrage einer größeren Kategorie und den/die Teilnehmer*in, einer Nutzungsgruppe zuzuordnen. Durch die Zuteilung kann durch spätere Auszählung der Anzahl der Einträge pro Kategorie oder Gruppe und deren Vergleich herausgefunden werden, wie die Nutzung des Bots ausfiel.
Die Entscheidung, welche Anfrage in welche Kategorie kommt, wurde nur von einer Person getroffen und ist daher subjektiv belastet. Es ist jedoch durch die Formulierung der Anfrage in den meisten Fällen eindeutig, welche Absicht die Anfrage verfolgt und in welche Kategorie sie einzuordnen ist. Einige Anfragen bestanden nur aus einzelnen Wörtern. Es wurde versucht, anhand der Antwort des Bots und der

nachfolgenden bzw. vorausgegangenen Anfragen, das Ziel der Anfrage zu ermitteln. Sollte beispielsweise eine Anfrage "Duplicate Array" lauten und auf die Antwort des Bots wurde mit "as code" reagiert, so wurde davon ausgegangen, dass auch die erste Anfrage dieses Ziel verfolgte. Solche knapp formulierten Anfragen stellen jedoch nur einen Bruchteil aller Anfragen dar.
Die Zuteilung der Studierenden in die Gruppen wird durch numerische Auswertung der kategorisierten Anfragen getroffen und es kann daher davon ausgegangen werden, dass die Konstruktvalidität grundlegend erfüllt ist.

4.2 Interne Validität

Als Störfaktoren kommen zunächst technische Probleme in Betracht. Es ist möglich, dass Protokolle nicht korrekt mitgeschrieben wurden oder ganz fehlen. Sollte die Kommunikation mit dem Chatbot davon unbetroffen sein, so ist ungewiss, ob Gesprächsverläufe fehlen oder unvollständig sind. Wenn Protokolle nicht korrekt mitgeschrieben wurden, dies aber wiederholt Auswirkungen auf das Gespräch mit dem Bot hätte, z.B. durch Ausbleiben der Antwort, so hätten sich betroffene Teilnehmende bemerkbar gemacht. Unverständnis über die Bedienung des Chatbots ist auszuschließen, da zu jedem Zeitpunkt eine Person zur Betreuung der Versuchspersonen anwesend war, welche zur Verwendung des Chatbots hätte befragt werden können. Allerdings ist in der Auswertung aufgefallen, dass sich eine Testperson auf Chinesisch mit dem Chatbot verständigte. Die Anfragen und Antworten wurden mittels Google-Übersetzer übersetzt und einer chinesisch sprechenden Kollegin unabhängig zur Übersetzung vorgelegt. Da sich beide Übersetzungen decken, ist es höchst unwahrscheinlich, dass diese Anfragen falsch zugeordnet wurden. Es besteht jedoch wie üblich ein Restrisiko. Im Zweifelsfalle betrifft dies allerdings nur drei von 380 Anfragen und ist daher vernachlässigbar. Darüber hinaus gab es auch einige Anfragen in englischer Sprache. Die Auswertung dieser stellt theoretisch kein Problem dar, dennoch ist es möglich, dass auch hier Übersetzungsfehler und damit falsche Kat-

egorisierung auftreten. Das Risiko, dass der Übersetzungsfehler so schwerwiegend ist und den eigentlichen Inhalt der Anfrage verfehlt, ist jedoch als sehr gering zu bewerten. Da es sich um ein Beobachtungsexperiment handelt, bei dem Schlussfolgerungen gezogen und nicht überprüft werden sollen, besteht keine Anforderung an die innere Validität bezüglich der Richtigkeit der Prüfung kausaler Schlussfolgerungen.

4.3 Externe Validität

Weiterhin ist zu prüfen, ob die Ergebnisse einer kleinen Probandengruppe repräsentativ sind und auf eine größere Spanne an Kontexten angewendet werden kann. Die Teilnehmer der Studie sind Studierende und daher die direkte Zielgruppe der Forschung. Da dies eine zufällig ausgewählte Gruppe an Studierenden war (sowohl retrospektiv als auch prospektiv) kann davon ausgegangen werden, dass dies ein repräsentativer Schnitt aus der Menge aller Studierenden ist und die Art der Nutzung relativ identisch auf eine größere Gruppe skaliert werden kann. Hier ist jedoch auch immer die jeweilige Vorerfahrung der Nutzenden zu berücksichtigen, da sich je nach Programmiererfahrung die Anfragen bezüglich der Art der Hilfestellung unterscheiden. Wenn repitativer Code erzeugt wird, kann trotzdem Verständnis für diesen bestehen.

Dennoch ist die Nutzung von Chatbots von Programmierproblemen nur schwer auf andere Probleme übertragbar. Jede Person hat andere Vorerfahrungen oder Interessen und wird sich daher mehr oder weniger tief mit einem Thema befassen oder versuchen dieses langfristig selbst zu verstehen. Dabei spielt natürlich auch die Häufigkeit und Schnittgröße dieses Themas mit der sonstigen, "normalen" Arbeit eine Rolle. Wenn das Thema nur wenige Male vorkommt, wird man sich nicht so stark damit auseinander setzen, wie wenn es alltäglich gebraucht wird. Die externe Validität ist also auf Studierende der Informatik bezogen erfüllt, jedoch nicht einfach auf andere, alltägliche Probleme übertragbar.

Da die Teilnehmenden aber nur eine kleine Teilmenge aller Studierenden sind, besteht trotzdem immer die Wahrscheinlichkeit, dass die Versuchspersonen eine Randgruppe bilden und nicht repräsentativ für alle stehen. Aufgrund der oben genannten Gründe, kann diese Wahrscheinlichkeit aber als sehr gering angesehen werden.

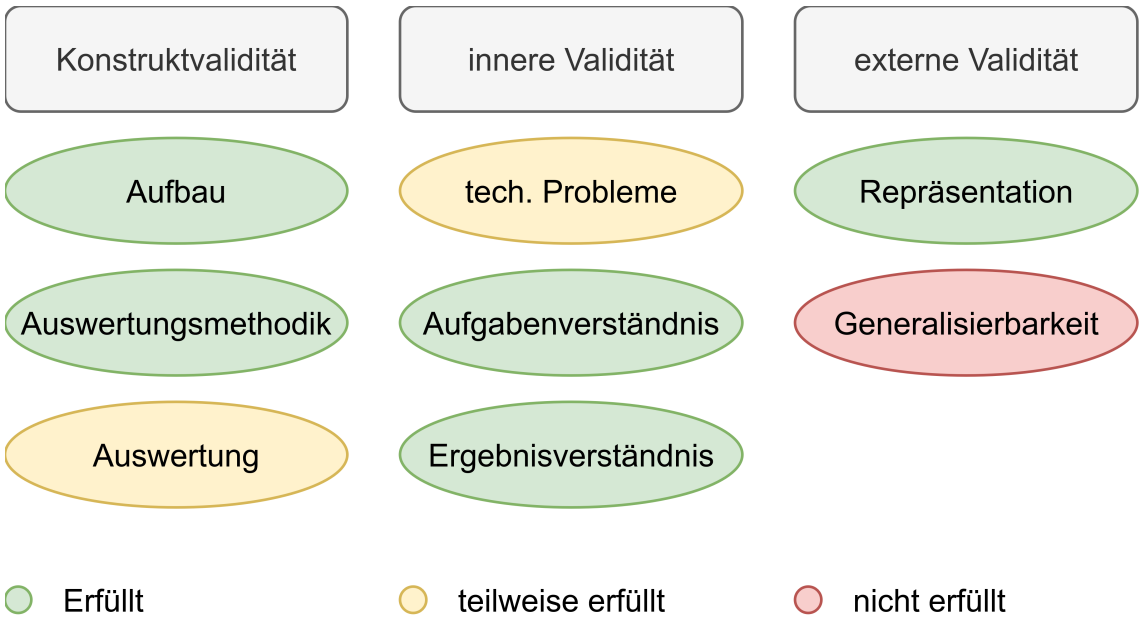


Abbildung 4.1: graphische Validitätsdarstellung

5 Ergebnisse

5.1 Auswertung

Nachdem alle Daten vorbereitet und ausgewertet wurden, ist nun Zeit für den Blick auf die Ergebnisse.
Zunächst ist zu erwähnen, dass durch vorlaufend abnehmende Beteiligung an den Übungen für die Auswertung wesentlich mehr Daten von Übung eins – 38 Abgaben – als im Vergleich zu Übung vier – 8 Abgaben – zur Verfügung stehen. Ebenfalls gibt es noch Datensätze, welche aufgrund fehlender Digitalisierung hier nicht betrachtet sind.

Aus der geführten Gruppe haben 48 Studierende auswertbare Konversationen mit dem Chatbot geführt. Auf die Kontrollgruppe entfallen 43 Studierende.
Insgesamt wurden 450 Anfragen an den Chatbot gestellt. Davon wurden 380 verwendbare Anfragen ausgewertet. Die geführte Gruppe stellte 222 Anfragen, die Kontrollgruppe stellte 158 Anfragen. Wäre jede Anfrage einzeln ausgewertet worden, entfielen auf die Erklärungsgruppe 232 Anfragen und auf die Generierungsgruppe 148 Anfragen.
Die konkrete Verteilung entfiel wie folgt:

Kategorie	geführte Gruppe	Kontrollgruppe
Frage zu Verwendung von DS	62	20
Frage zu Codeverständnis/-erklärung	29	25
Frage zu Algorithmusverständnis	39	10
Hilfe zur Fehlerbehebung	12	14
Frage zu Code von Java	9	12
Testgenerierung	1	8
Codegenerierung	70	69

Table 5.1: Verteilung der kategorisierten Anfragen

Dies ist nachfolgend in Abb. 5.1 nochmals als Balkendiagramm dargestellt.

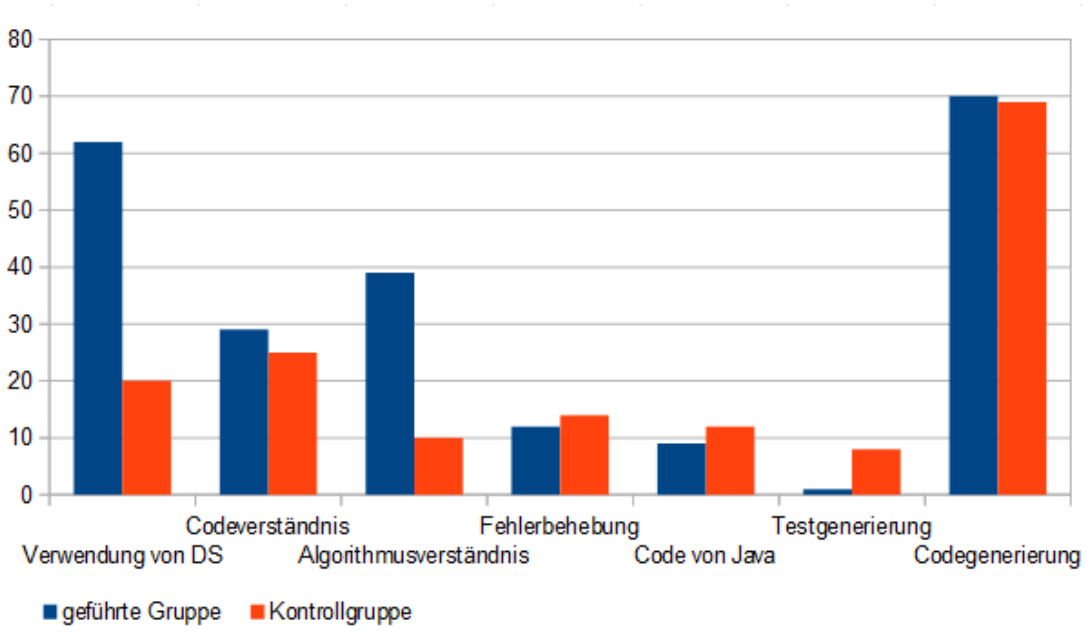


Abbildung 5.1: Kategorieverteilung

Zusätzlich ließen sich die Probanden grundlegend in vier Nutzungsgruppen einteilen. Die erste Gruppe stellte vornehmlich Anfragen aus den Erklärungskategorien. Dies sind "Frage zur Verwendung von Datenstrukturen", "Frage zu Algorithmusverständnis", "Frage zu Codeverständnis/-erklärung" und "Frage zu Code von Java". Diese

Gruppe nutzte den Bot vornehmlich um vom Bot Unterstützung in ihrem eigenen Coding zu bekommen. Die zweite Gruppe stellte vornehmlich Anfragen an die Generierungskategorien. Dazu zählen "Testgenerierung" und "Codegenerierung". Diese Gruppe wollte hauptsächlich fertigen Code vom Bot haben. Gruppe drei stellte keine aufgabenbezogenen Anfragen und nutzte die Hilfestellung des Bots nicht. Die vierte Gruppe stellte sowohl Anfragen zur Unterstützung des Codings, als auch Anfragen zur Erstellung von Code.

Es ist zu beachten, dass ein*e Teilnehmer*in auch in verschiedenen Nutzungsgruppen enthalten sein kann. Die Zuweisung in die Gruppe basiert auf den jeweiligen Übungseinheiten. Wenn ein*e Teilnehmer*in in allen vier Übungseinheiten auswertbare Konversationen mit dem Bot geführt hat, so geht dies auch mit vier Gruppenzuteilungen in die Statistik ein. Daher ist die Summe über die Nutzungsgruppen (91 Zuteilungen) auch höher als die eigentliche Teilnehmeranzahl (37 Personen).
Insgesamt wurden 91 Nutzergruppenzuweisungen getroffen. Die konkrete Verteilung entfiel wie folgt:

Nutzungsgruppe	geführte Gruppe	Kontrollgruppe
Erklärungsgruppe	19	16
Generierungsgruppe	9	16
Nicht-Nutzer-Gruppe	12	7
Kombinationsgruppe	8	4

Table 5.2: Verteilung der Nutzungsgruppe

Dies ist nachfolgend in Abb. 5.2 nochmals als Balkendiagramm dargestellt.

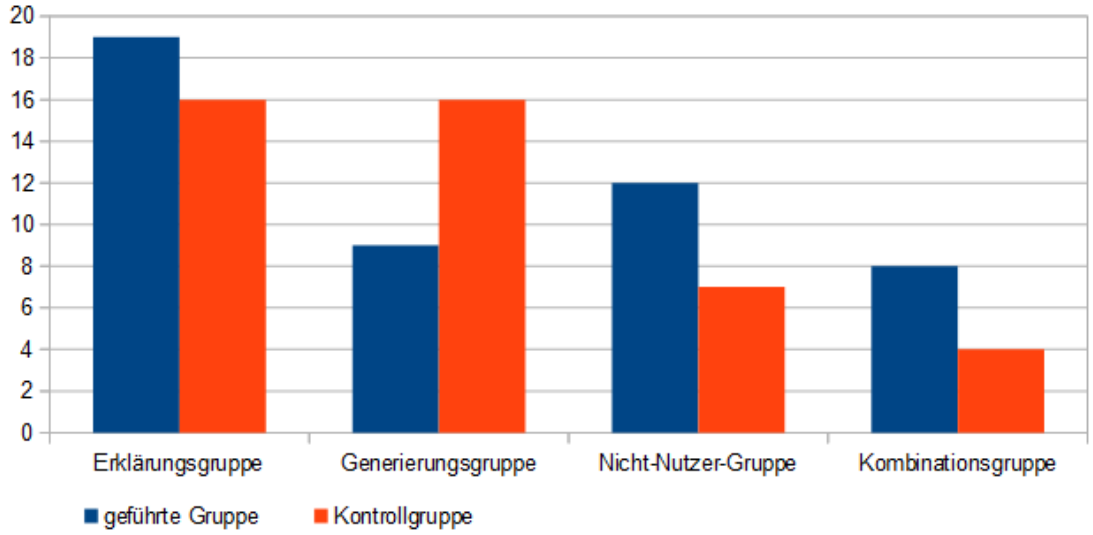


Abbildung 5.2: Nutzungsgruppenverteilung

5.2 Diskussion

Der nachfolgende Abschnitt versucht Erklärungen für die beobachteten Ergebnisse zu finden. Ob diese Erklärungsversuche wirklich den Tatsachen entsprechen ist unklar, da die Teilnehmenden hierzu selbst Stellung beziehen müssten, um die Erklärung zu validieren.

5.2.1 Kategorieverteilung

Die Ergebnisse zeigen, dass der Chatbot vor allem zur Codegenerierung, zum Verständnis von Algorithmen, Datenstrukturen (DS), bzw. allgemein Code und zur Fehlerbehebung (Debugging) verwendet wurde. Diese Ergebnisse decken sich mit denen der Forschenden der Tampere Universität Finnland[10] und denen der vorausgegangenen Studie der Softwaretechnik-Professur[8]. Dies ist jedoch auch nicht verwunderlich, da Codeerzeugung, Codeberichtigung/-erklärung und Konzepterklärung die wichtigsten Anwendungsmöglichkeiten eines LLMs beim Erstellen von Computerprogrammen sind.

Zunächst fällt auf, dass die geführte Gruppe (GG) wesentlich mehr Anfragen stellte als die Kontrollgruppe (KG) (222 zu 158), obwohl in beiden Gruppen annähernd gleich viele Studierende waren (48 zu 43). Dies zeigt sich in der Auswertung klar im Unterschied in den Anfragen zur Verwendung von DS und den Anfragen zum Algorithmusverständnis. Versuchspersonen der GG stellten dreimal so viele Anfragen zur Erklärung von DS und deren Verwendung und fast viermal so viele Anfragen zur Erklärung von Algorithmen, wie Versuchspersonen der KG. Erklärbar wäre dies durch die Bereitstellung das Methodengerüsts mit TODOs an die GG. Wenn der Bot nicht anders angewiesen ist, bedient er sich bei Empfehlungen und Hilfestellungen zur Lösung der Aufgabe an allen in Java zur Verfügung stehenden DS und Algorithmenkonzepten. Viele dieser DS und Algorithmen sind zum Zeitpunkt der Studie, welche zu Semesterbeginn abgehalten wurde, den Studierenden noch nicht

bekannt, wodurch diese bei Empfehlung durch den Bot, diesen nach einer Erklärung derselben ersuchen, wie diese umzusetzen sei. Beispielsweise verlangte eine Aufgabe das Löschen von Duplikaten aus einem Array. Der Chatbot bediente sich dazu eines Sets, welcher der/dem Studierenden unbekannt war und demzufolge die Frage was denn ein Set sei zur Folge hatte. Das erklärt auch den großen Unterschied in der Anfragenhäufigkeit der beiden Ursprungsgruppen. Während die GG Antworten mit unbekannten DS oder Algorithmen bekam, welche sie sich erklären ließen, bekam die KG diese zwar auch, jedoch bereits fertig im Code integriert. Wenn dieser problemlos lief, schien auch kein Nachfragen bezüglich der verwendeten DS oder Algorithmen erforderlich zu sein.
Auffällig war dabei allerdings, dass in den meisten Fällen unklare Konzepte, wie verlinkte Listen, Sets oder Maps im Anschluss wieder mit einer weiteren Anfrage nach Umarbeitung entfernt wurden. Ob dies aus Scham vor der Nutzung des Bots geschah, ihnen die Qualität des Codes nicht genügte oder welche anderen Gründe dafür ursächlich sind, ist unklar. Dies weist jedoch daraufhin, dass ein grundlegendes Verständnis für den generierten Code besteht und nur einzelne Konzepte unklar waren. Dies zeigt, dass die Studierenden nicht bloß die Antwort des Bots kopierten, sondern sich auch mit dieser auseinander setzten, was ein grundlegendes Verständnis von Codesyntax voraussetzt. Das grundlegende Fähigkeiten zur spezifischen Arbeitsaufgabe notwendig sind, um qualitative Antworten des Chatbots zu erhalten, deckt sich auch mit den Erkenntnissen der AIDAHO-Projektgruppe[12] und der vorausgegangen Studie der Professur[8].

Bemerkenswert ist auch, dass die Anzahl der Anfragen zur Codegenerierung beider Ursprungsgruppen annähernd gleich ist. Dies ist interessant, da erwartet wurde, dass die GG mehr Anfragen zur Codegenerierung stellt als die KG, da die GG seltener Antworten mit tatsächlicher Codegenerierung erhält und dadurch mehr Anfragen stellen müsste, um die gewünschte Antwort zu erhalten. Das die Werte dieser Annahme widersprechen könnte darauf zurückzuführen sein, dass die Studierenden

der GG nach einer Antwort mit TODOs im eigenen Lerninteresse versucht haben, sich mit der Aufgabe auf Basis des Codegerüsts weiter selbstständig zu befassen. Die Versuchspersonen aus der KG müssten spezifisch nach einem solchen TODO-Gerüst fragen, was jedoch niemand getan hat. Daher sind die Teilnehmenden der KG vermutlich nicht auf die Idee gekommen, sich nach der ersten Codegenerierung weiter selbstständig mit der Aufgabe zu befassen, da sie bereits eine funktionierende Lösung vorliegen hatten. Es gab jedoch auch Fälle, in denen Studierende nach Erhalt einer Antwort mit TODOs, den Chatbot explizit nach einer Lösung ohne TODOs gefragt haben. Diese Anfrage bewirkt natürlich, dass diese*r Teilnehmende mit zwei Anfragen zur Codegenerierung in die Auswertung eingeht. Andererseits haben Teilnehmende nach Erhalt einer Antwort mit kompilierbarem Code, ebenfalls Folgeanfragen zur Codegenerierung gestellt. Diese hatten dann jedoch die Änderung des bereitgestellten Codes, bspw. Änderung von Variablennamen oder Änderung der Klammerformatierung, zum Inhalt. Das Studierende der GG diese Änderungen nicht anfragten, liegt wohl daran, dass diese für die Änderung solcher "Kleinigkeiten" wieder ein TODO-Gerüst erwarteten und sich daher nicht die Mühe machten, da sie kein passendes Ergebnis erwarteten. Es ist daher wahrscheinlich, dass grundlegend ungefähr gleich viele Studierende aus beiden Ursprungsgruppen (GG oder KG) den Chatbot zur Bereitstellung von Code genutzt haben und die Zuteilung in GG oder KG und die damit verbundenen Folgen, kaum Einfluss auf die Nutzung zur Codegenerierung hatten. Das bedeutet, dass unabhängig von der Antwort des Bots weiter das vorher angepeilte Ziel, Codegenerierung durch den Bot, verfolgt wurde.

Die geringe Anzahl an Anfragen bezüglich des Codes von Java waren zu erwarten. Die Studienteilnehmenden sind Zweit-Semester-Studenten und haben daher bereits ein Grundverständnis für die Syntax von Programmiersprachen entwickelt. Ebenfalls sind C, welches im ersten Studienjahr gelehrt wird, und Java syntaktisch sehr ähnlich. Die Unterschiede der beiden Programmiersprachen, zum Beispiel manuelle Speicherverwaltung des Heaps in C oder fehlende goto-Unterstützung in Java, fallen

bei diesen Aufgaben nicht ins Gewicht, da solche Konstrukte nicht benötigt wurden. Benötigte Konstrukte wurden auch vorab in der Vorlesung vermittelt und die Vorlesungsunterlagen waren jederzeit einsehbar. Der Großteil der Anfragen stammt von einem/einer Nutzenden, welche*r sich eingänglich mit der Bibliothek von Java beschäftigt hat, wobei die gestellten Anfragen zur Bibliothek für die Erfüllung der Aufgabe nicht notwendig waren. Interessanterweise behandelten die restlichen zu diesem Thema gestellte Anfragen vorrangig den Modulo-Operator, obwohl sich dieser in Java und C nicht unterscheidet.
Die leicht höhere Nutzung durch die KG, ist auf eine Person zurückzuführen, welche in der ersten Übung sieben Anfragen bezüglich Nutzung von Bibliotheken stellte. Ebenfalls wurden alle Anfragen in dieser Kategorie in den ersten zwei Übungen gestellt. Erklärbar ist das dadurch, dass in den späteren Übungen die Syntax von Java gefestigt war und keiner Erklärung mehr bedurfte.

Weiterhin zeigen die Daten auf, dass erheblich weniger Anfragen zum Testen und Debuggen gestellt wurden. Es wurden siebenmal so viele Anfragen zur Codegenerierung, wie im Vergleich zur Testgenerierung, gestellt und nahezu eben soviel mehr im Vergleich zur Fehlerbehebung. Diese niedrige Anzahl findet sich auch in anderen vergleichbaren Studien, wie in der vorausgegangen Studie der Professur[8]. Dies könnte daraufhin weisen, dass dem Testing von Code wenig Beachtung geschenkt wird und davon ausgegangen wurde, dass der Code fehlerfrei läuft. Möglicherweise wird nur mit wenigen selbsterstellten Eingaben getestet und Edge-Cases nicht betrachtet. Weiter ist zu beachten, dass falsche bzw. unvollständige Lösungen hier keine Konsequenzen haben und daher dem Testing auch wenig Priorität eingeräumt wurde. Tatsächlich sind alle zum Thema Testgenerierung gestellten Anfragen nach der entsprechenden Codegenerierung gestellt worden. Der/die Teilnehmende hat sich also zunächst die Lösung der Aufgabe generieren lassen und danach für ebendiese Generierung Tests angefragt. Dies deutet daraufhin, dass einzelne Studierende den Lösungen des Bots nicht vertrauen bzw. "auf Nummer sicher" gehen wollten.

Wohingegen der Großteil der Studierenden die Lösung des Bots ohne Zweifel an der Richtigkeit oder mit Gleichgültigkeit gegenüber der Richtigkeit angenommen hat. Auch dies liegt wohl an der Konsequenzlosigkeit einer falschen Lösung. Dass die Verteilung sehr zu Seiten der KG ausfällt, liegt daran, dass die Anfragen zur Testgenerierung nur von zwei Personen gestellt wurden. Während der/die Studierende der GG nach der ersten Übungseinheit (ÜE) keine Abgaben mehr getätigt hat, war die Person der KG über alle vier ÜE anwesend und hat dadurch diese Verteilung erzeugt.

Die niedrige Anzahl an Anfragen zur Fehlerbehebung lässt sich ähnlich erklären. Wenn wenig getestet wird, werden weniger Fehler gefunden. Auch hier ist die Konsequenzlosigkeit fehlerhafter Abgaben wahrscheinlich wieder ein großer Faktor. Doch auch die Zeitbeschränkung der Aufgabenstellung kann dabei mit ursächlich sein. Es wird lieber fehlerhafter als gar kein Code abgegeben und daher auf Fehlerbehandlung und Testing bei vorausgegangen Aufgaben verzichtet, um noch eine Lösung für andere Aufgaben einreichen zu können. Ursächlich für dieses Verhalten ist vermutlich die Tatsache, dass die Versuchspersonen Studierende waren. In Klausuren wird lieber eine fehlerhafte Lösung für mehrere Aufgaben abgegeben, um dafür noch Punkte zu bekommen, weil keine Lösung immer mit null Punkten bewertet wird.

Die Anzahl an Anfragen bezüglich Codeverständnis und -erklärung sortieren sich von ihrer Häufigkeit im Mittelfeld ein und sind damit im erwarteten Bereich. Sie sind nicht zu niedrig, da der Chatbot hier ja die Rolle eines Tutors im Programmierbereich übernimmt und Erklärungen zu Code zu geben, ist in diesem Bereich eine der häufigsten Arbeiten. Sie sind aber auch nicht zu hoch, da für diese Art von Anfragen bereits bestehender Code, ob selbst geschrieben oder generiert, vorliegen muss. Wenn der Code selbst geschrieben ist, dann ist das Verständnis des/der Schreibenden über diesen vorhanden. Wenn der Code generiert ist, dann steht meist die schnelle Abarbeitung der Aufgabe im Vordergrund und die vollständige Erklärung

ist nicht gefragt.

5.2.2 Nutzungsgruppenverteilung

Ein Blick in die Nutzungsgruppenverteilung zeigt: Die Anzahl der Personen pro Gruppe wird weniger, je höher die Nummer der Nutzungsgruppe. Das bedeutet ein großer Teil der Teilnehmenden hat den Bot entweder zur Erklärung oder Generierung genutzt. Insgesamt fast dreimal so viele Erklärungsgruppe-Nutzende und etwas mehr als doppelt so viele Generierungsgruppe-Nutzende, wie Kombinationsgruppe-Nutzende. Die Studierenden haben den Bot also, je nach ihrem Vorgehen, für Erklärung oder Generierung genutzt, aber kaum für beides. Dieses Verhalten ist eventuell darauf zurückzuführen, dass die Teilnehmenden vor Nutzung des Bots eine klare Vorstellung von dessen Hilfe haben (Erklärung oder Generierung von Lösungen) und im Falle einer anders ausfallenden Antwort, trotzdem ihre Vorstellung der Lösung umgesetzt sehen wollen.

Dagegen wenig überraschend ist, dass die Anzahl der Nutzenden der Erklärungsgruppe, die der Gernerierungsgruppe übersteigt. Die Daten ließen erkennen, dass die Summe der kategorisierten Anfragen aus dem Erklärungsbereich die der Anfragen aus dem Generierungsbereich ebenfalls übersteigt. Daher muss um die Proportionalität zu wahren, die Verteilung der Nutzungsgruppen ähnlich ausfallen.

Deutlich zu sehen ist, dass vor allem Versuchspersonen der KG in die Generierungsgruppe fallen. Die Versuchspersonen der GG sind dahingegen vermehrt in der Erklärungsgruppe und Kombinationsgruppe zu finden. Deshalb überwiegen in allen anderen Nutzungsgruppen die Versuchspersonen aus der GG. Es sind zwar fünf mehr Studierende eingangs in die GG zugeteilt worden, doch das sollte sich dahingehend nur mit einem Studierenden mehr in der GG als in der KG zeigen. Dass sich ein

solches Ungleichgewicht in der Nutzung abzeichnete, kann darauf zurückzuführen sein, dass die Teilnehmenden der GG nur Methodengerüste bekamen und sich so dann mehr auf eigene Programmierung konzentrierten, als den Bot den Code generieren zu lassen. Auf der anderen Seite haben Personen der KG kompilierbaren Code erhalten und dadurch auch häufiger auf diese Möglichkeit zurückgegriffen. Die fehlenden Nutzenden der GG in der Kategorie Codegenerierung verteilen sich dann auf die übrigen Nutzungsgruppen.

Wieso Teilnehmende der GG den Bot fast doppelt so häufig nicht nutzten, wie Teilnehmende der KG, ist unklar, denn sie nutzten den Bot ja nicht. Es wurden jedoch in ÜE 2 mehr Personen in diese Nutzungsgruppe eingeordnet als in ÜE 1. Es könnte daher dadurch erklärt werden, dass Studierende den Bot in der ersten ÜE nicht als hilfreich empfanden und ihn daher in den folgenden ÜE nicht mehr nutzten. Das dies besonders Teilnehmenden der GG betraf ist vermutlich darauf zurückzuführen, dass sie den Bot zur Generierung nutzen wollten und anschließend diese Bemühung aufgaben.

5.2.3 Zusammenspiel von Kategorien und Nutzungsgruppen

Als nächstes zeigt sich direkt: Der Bot wurde hauptsächlich zur Codegenerierung genutzt. Es wurden 50 Anfragen mehr bezüglich Codegenerierung geschickt, als im Vergleich zu Anfragen zur Verwendung von DS, welche am zweithäufigsten getätigt wurden. Auch in der vorausgegangenen Studie der Professur waren Anfragen zur Codegenerierung mit Abstand die meist getätigten Anfragen[8].
Auffallend ist jedoch, dass mehr Anfragen zur Codegenerierung aus der GG, im Vergleich zur KG, kamen, obwohl wesentlich mehr Studierende der KG im Nachhinein der Generierunggruppe zugeteilt wurden. Das bedeutet, dass die Studierenden der GG fast doppelt so viele Anfragen zur Codegenerierung mehr gestellt haben,

als die Studierenden aus der KG. Dies könnte daran liegen, dass Studierende der GG zunächst nur das Methodengerüst und keinen fertigen Code bekommen haben. Wie oben erwähnt, haben diese dann wohl nicht mit diesem weiter gearbeitet, sondern mit weiteren Generierungsanfragen versucht, vom Chatbot den fertigen Code zu bekommen. Zu berücksichtigen ist aber auch, dass es aus der GG doppelt so viele Kombinationsgruppenzugehörigkeiten gibt, welche ebenfalls Anfragen bezüglich Codegenerierung gestellt haben, wie aus der KG. Die leicht höheren Generierungsanfragen der GG, können sich also auch durch die höhere Beteiligung in der Kombinationsgruppe erklären lassen.

Etwas Ähnliches zeigt sich bei Betrachtung der Anzahl der Erklärungsanfragen und der Mitglieder der Erklärungsgruppe. Obwohl nur drei Personen der GG mehr in der Erklärungsgruppe vertreten sind, haben diese 71 Anfragen mehr im Bereich von Algorithmusverständnis und Verwendung von DS gestellt. Die Anfragen dieser Kategorie wurden auch von vielen verschiedenen Teilnehmenden gestellt und sind anders, wie bei der oben erwähnten Kategorie "Fragen zum Code von Java", nicht hauptsächlich einzelnen Personen zuzuordnen. Eine Erklärung dafür könnte wie zuvor schon beschrieben, die Empfehlung von unbekannten Datenstrukturen und Algorithmen sein.

Die Daten zeigen, dass viele Versuchspersonen den Bot nutzten, um sich Programmierkonzepte erklären zu lassen und dadurch ihr eigenes Wissen und ihre eigenen Fähigkeiten zu verbessern. Dennoch ist es wichtig, verantwortungsvoll mit dem Chatbot umzugehen, um selbst dazuzulernen. Denn bei bloßer Anwendung zur Lösung der Aufgabenstellung werden die eigenen Fähigkeiten eher ersetzt als erweitert, da sich rein auf die Korrektheit der Lösung des Chatbots verlassen wird. Dies zeigten auch die Erkenntnisse der Forschenden der Tampere Universität Finnland [10].

Ebenfalls wurde beobachtet, dass Teilnehmende den Bot nutzten, um Fehler zu finden und zu beheben und sich grundlegende Programmstrukturen – wie Konstruktoren und Methodenköpfe – vorzubereiten, auf denen dann aufgebaut wurde. Die Nutzung von LLMs unterstützt auch dabei, Fehler in Code und Exceptions zu finden. Das LLM kann diesen dann direkt mit bekannten ähnlichen Problemen vergleichen und so schnell adaptiv eine Lösung bereitstellen, ohne dass sich lange mit Fehlersuche und Debugging aufgehalten werden muss. Auch hier ist der verantwortungsvolle Einsatz wieder grundlegend, um das Problem im Code zu verstehen und künftig zu vermeiden. Die Forschenden der AIDAHO-Projektgruppe kommen diesbezüglich zu dem gleichen Schluss [12].

6 Abschluss

Wenn Studierende einen Chatbot als Assistenz zur Seite gestellt bekommen und dieser zuvor die Anweisung bekam, bei Anfragen zur Codegenerierung nicht direkt kompilierbaren Code, sondern zunächst ein Grundgerüst mit TODOs, auszugeben, zeigen sich vor allem folgende fünf Punkte:

1. Studierende nutzen den Bot entweder zur Erklärung von Programmierkonzepten oder zur Generierung von Code. Zunächst fertigen Code zu generieren und anschließend zu diesem Fragen zu stellen, findet nur selten statt.

2. Wenn mit der Zielsetzung der Codegenerierung eine Anfrage an den Chatbot gestellt wurde, dann wird in der Regel diese Zielsetzung weiter verfolgt. Egal ob der Chatbot den Code direkt bereit stellt oder zunächst mittels TODOs versucht den/die Nutzer*in selbst zum Programmieren anzuregen.

3. Allgemein nutzen Studierende ein LLM eher um eigene Fähigkeiten zu erwerben, als sich bloß die Lösung generieren zu lassen. Es wird allerdings auch auf Codegenerierung zurückgegriffen, um grundlegende Vorarbeit, wie Klassen- oder Methodengerüste zu erstellen.

4. Durch die Nutzung von allen vorhandenen Java-Konzepten und Datenstrukturen durch den Bot, versuchen die Studierenden diese mit weiteren Nachfragen zu verstehen. Sie versuchen dabei jedoch nur im Groben, die Funktionsweise zu verstehen und nicht diese später selbst einsetzen zu können. Ob die Funktionsweise im Einzelenen auch wirklich verstanden wurde oder einfach wieder vom Thema abgelassen wurde, ist unklar.

5. Das Testen von Code spielt für viele Studierende nur eine geringe bis gar keine Rolle.

Das Forschungsfeld der LLMs wird weiter expandieren und auch in der Lehre werden LLMs Einzug erhalten. Im Weiteren ist nun zu klären, wie erfolgreich der hier untersuchte Einsatz von LLMs in der Lehre ist und wie sich die Lehre an deren Einsatz anpassen muss. Denn auch wenn diese in Übungen oder Klausuren verboten bleiben, in Hausarbeiten werden sie vermehrt zum Einsatz kommen, da ihr Potenzial auch in der Informatik noch nicht ausgeschöpft ist. Daher lässt sich doch zweifelsfrei sagen: "Das setzt sich durch!"

Literaturverzeichnis

[1] Anonym: Das erste smartphone. SWB (2017), https://www.swb.de/ueber-swb/swb-magazin/swb-insider/erstes-smartphone

[2] Anonym: Wer erfand das internet? Demokratie Webstatt (2018), https://www.demokratiewebstatt.at/thema/thema-being-digital-und-neue-medien/von-neuen-und-alten-medien/wer-erfand-das-internet

[3] Anonym: Chatgpt ist da. openai.com (2022), https://openai.com/de-DE/index/chatgpt/

[4] Chatterji, A., Cunningham, T., Deming, D., Hitzig, Z., Ong, C., Shan, C., Wadman, K.: How people use chatgpt (2025), https://cdn.openai.com/pdf/a253471f-8260-40c6-a2cc-aa93fe9f142e/economic-research-chatgpt-usage-paper.pdf

[5] Figliè, R., Turchi, T., Baldi, G., Mazzei, D.: Towards an llm-based intelligent assistant for industry 5.0 (2024), https://ceur-ws.org/Vol-3701/paper7.pdf

[6] Heetkamp, S.J., Brachtendorf, M.: Ki-zeuge im virtuellen gerichtssaal. Fachzeitschrift "Die Neue Hochschule" (2024), https://d-nb.info/133386485X/34, seiten 14-16

[7] Kelbert, P., Siebert, J., Jöckel, L.: Was sind large language models? und was ist bei der nutzung von ki-sprachmodellen zu beachten? (2023), https://www.iese.fraunhofer.de/blog/large-language-models-ki-sprachmodelle/

[8] Mailach, A., Gorgosch, D., Siegmund, N., Siegmund, J.: "ok pal, we have to code that now": Interaction patterns of programming beginners with a conversational chatbot (2025), https://www.tu-chemnitz.de/informatik/ST/publications/papers/OkPal.pdf

[9] Merritt, R.: What is a transformer model? (2022), https://blogs.nvidia.com/blog/what-is-a-transformer-model/

[10] Rajala, J., Hukkanen, J., Hartikainen, M., Niemelä, P.: "call me kiran" – chatgpt as a tutoring chatbot in a computer science course (2023), https://dl.acm.org/doi/pdf/10.1145/3616961.3616974

[11] Thirunavukarasu, A.J., Ting, D.S.J., Elangovan, K., Gutierrez, L., Tan, T.F., Ting, D.S.W.: Large language models in medicine (2023), `https://www.nature.com/articles/s41591-023-02448-8.pdf`

[12] Vogelgesang, J., Bleher, J., Krupitzer, C., Stein, A., Jung, R.: Nutzung von chatgpt in lehre und forschung – eine einschätzung der aidaho-projektgruppe (2023), `https://aidaho.uni-hohenheim.de/fileadmin/einrichtungen/aidaho/Dokumente/AIDAHO_ChatGPT_Positionspapier_23-02-09.pdf`