Möglichkeiten und Grenzen KI-unterstützter Softwareentwicklung im SAP-Umfeld

Konstantin Schatz
T.CON GmbH & Co. KG
OTH Regensburg
Straubingerstraße 2
94447 Plattling
konstantin.schatz@st.oth-regensburg.de

Norbert Kölbl
Solution Architect
T.CON GmbH & Co. KG
Straubingerstraße 2
94447 Plattling
norbert.koelbl@team-con.de

Prof. Dr. Frank Herrmann
Ostbayerische Technische Hochschule Regensburg
Labor für Informationstechnik und Produktionslogistik (LIP)
Galgenbergstraße 32
93053 Regensburg
frank.herrmann@oth-regensburg.de

ABSTRACT

Generative Künstliche Intelligenz wird zunehmend auch in der Softwareentwicklung eingesetzt. Verschiedene Anbieter, wie GitHub, eine renommierte Entwicklerplattform, haben Lösungen herausgebracht, die durch Codevervollständigung den Softwareentwicklungsprozess beschleunigen und fehlerfrei machen sollen. Doch der Einsatz dieser Lösungen birgt Risiken. Wem gehört der generierte Code? Was geschieht mit den Daten, anhand derer der Prompt Code generiert?
Ziel ist es, Bedürfnisse an Codegenerierungstools für die Firma T. CON zu definieren, und die drei Lösungen GitHub Copilot, Tabnine und Codeium anhand der Bedürfnisse miteinander zu vergleichen.

Keywords

Analyse, ABAP, Eclipse, Entwicklung, Generative KI, JavaScript, KI, LLM, ML, Python, SAP, Software, UI5, VS Code

1. EINLEITUNG

1.1 Einleitung

Laut Thomas Dohmke, CEO der weltweit größten Entwicklungsplattform GitHub, wird bereits (Stand Januar 2024) die Hälfte aller Software mit Unterstützung von KI-Copiloten entwickelt, mit der Prognose, dass dieser Anteil in den nächsten Jahren auf 80% ansteigt. Laut eigener Aussage können Entwickler einen Produktivitätsgewinn von 30 bis 50 Prozent verzeichnen. [1] Auch die SAP kündigte bei der TechEd 2023 an, jeden Entwickler, zum Entwickler für generative KI zu machen. Die Roadmap ist klar: Ohne den Einsatz generativer KI soll es bei der SAP keine Softwareentwicklung mehr geben.[2] Künstliche Intelligenz findet immer weiter Einzug, sowohl in unser Privatleben als auch in Unternehmen. Nach Prognose der International Data Corporation verfünffachen sich die Ausgaben für KI-Plattformen zwischen 2022 und 2026. [3] Deutsche Unternehmen betrachten laut Bitkom e. V. künstliche Intelligenz zu 65% als Chance, jedoch benutzen sie nur 18% des Mittelstands. Jedoch sind viele Erwartungen an KI und generative KI laut Gartner zur Zeit überzogen. [4]

Als technologieoffenes Unternehmen möchte die T. CON Möglichkeiten und Grenzen neuer Technologien erforschen, um sie sicher zu benutzen.[5]
Allerdings möchte die T. CON nicht ausschließlich abhängig von SAP-eigenen Tools sein. Daher soll in diesem Projekt erörtert werden, welche SAP-fernen generative KI-Tools sich für den Einsatz in der Softwareentwicklung eignen.

1.1 Problemstellung und Zielsetzung

Das Hauptziel und Kernergebnis des Projektes besteht darin, mit Hilfe einer wissenschaftlichen Evaluation festzustellen, ob der GitHub Copilot, Tabnine und Codeium sich für die Bedürfnisse der T. CON eignen.
Daher ist ein Unterziel mit Hilfe von Experteninterviews herauszufinden, welche Bedürfnisse die Softwareentwickler bei der T. CON haben. Dies beinhaltet unter anderem die Benutzung von IDEs und Programmiersprachen, sowie urheberrechtliche Aspekte. Auf den Ergebnissen der Experteninterviews aufbauend sollen dann in einer Nutzwertanalyse die drei Tools GitHub Copilot, Tabnine und Codeium analysiert werden. Dafür müssen aussagekräftige Vergleichskriterien gefunden werden. Die Evaluierung der Codequalität der Tools wird mit Hilfe von prototypischen Implementierungen durchgeführt.

2. T. CON

Die T. CON ist ein 1999 gegründetes SAP-Beratungsunternehmen mit Hauptsitz in Plattling und weiteren Standorten in Regensburg, Passau, Heilbronn, Hamburg und Berlin. Die 420 Mitarbeiter betreuen 240 Kunden durch Kompletteinführungen, Optimierungen, S/4-Transitionen, Managed Services und Digitalisierung, was ihnen bereits eine Auszeichnung für exzellente MILL-Expertise durch die SAP eingebracht hat. Es gilt als Familienfreundlich und ist regelmäßig unter den „Bayerns BEST 50". Das agile Unternehmen strebt neues Wachstum an, wirtschaftlich wie personell. [6]

3. PROBLEMANALYSE

Um die geeigneten Tools zu analysieren, ob sie sich für die T. CON eignen, müssen zunächst die Bedürfnisse der Firma erörtert werden. Hierfür wird die Methodik des Experteninterviews verwendet. Diese ist eine qualitative Forschungsmethode, bei der eine oder mehrere Personen interviewt werden, welche über spezifisches Fachwissen verfügen. [7] In diesem Projekt eignet sich das Experteninterview, da die Bedürfnisse der T. CON spezialisiertes Wissen sind, welches nicht in Literatur zu finden ist.
In der Problemstellung ist bereits aufgeführt, dass die drei Tools GitHub Copilot, Tabnine und Codeium analysiert werden. Dabei war der GitHub Copilot bereits von der Firma vorgegeben. Die anderen beiden Tools ergaben sich auf folgenden Gründen: Um zwischen mehreren Tools auswählen zu können reicht die Bewertung dieses Tools allein nicht. Jedoch ist auch die Bewertung von mehr als drei Tools aufgrund des zeitlichen Rahmens nicht sinnvoll, weswegen sich auf drei Tools geeinigt wurde.
Mindestanforderungen an ein Tool war, dass es direkt in VS Code integriert werden kann, das ergab sich aus den Experteninterviews. Die Entscheidung fiel in enger Absprache mit dem Firmenbetreuer auf zusätzlich Tabnine und Codeium, Konkurrenzprodukte des GitHub Copilot. Sobald die Bedürfnisse klar sind, müssen die Tools analysiert werden. Hierfür eignet sich die Nutzwertanalyse. Mit dieser Methodik können objektive Bewertungen hinsichtlich mehrerer unabhängiger Kriterien vorgenommen werden. Die Nutzwertanalyse ist eine strukturierte Methodik, die große Vorteile zur Entscheidungsfindung bietet. Die klar festgelegte Vorgehensweise gewährleistet eine bessere Nachvollziehbarkeit des Entscheidungsprozesses. Auch ihre quantitative Ausrichtung macht sie objektiver. Jedoch hängen die Genauigkeit der Ergebnisse von den verfügbaren Daten ab. Außerdem bietet die Gewichtung der Kriterien die Gefahr der Subjektivität. [8]

4. GRUNDLAGEN UND BEGRIFFSERKLÄRUNGEN

Bevor die Lösung erarbeitet wird, werden wichtige Begriffe definiert für das weitere Verständnis in der Abschlussarbeit.

4.1 SAPUI5

SAPUI5 ist ein von SAP entwickeltes Toolkit für die Webentwicklung, basierend auf JavaScript, HTML5 und CSS3. Es ermöglicht die einfache Erstellung von Fiori-Apps und dient als Grundlage für verschiedene Benutzeroberflächen in SAP-Lösungen. Entwickler nutzen Tools und Bibliotheken von SAPUI5 für die responsive Entwicklung von Webanwendungen, die sowohl auf Desktops als auch auf mobilen Geräten lauffähig sind [9]. Es ist der Nachfolger von SAP-UI-Entwicklungstechnologien wie BSP, PDK, Web Dynpro Java und Web Dynpro ABAP. Die Open-Source-Variante OpenUI5 ermöglicht auch Nicht-SAP-Kunden die Entwicklung plattformunabhängiger Anwendungen. Insgesamt erleichtert SAPUI5 die effiziente und benutzerfreundliche Webentwicklung für verschiedene Geräte und Plattformen [10].

4.2 Künstliche Intelligenz

Künstliche Intelligenz (KI) ist ein Teilgebiet der Informatik, das sich auf die Entwicklung von Maschinen konzentriert, die intelligentes Verhalten zeigen. KI umfasst alle Anstrengungen, deren Ziel es ist, Maschinen intelligent zu machen. Dabei wird Intelligenz verstanden als die Eigenschaft, die ein Wesen befähigt, angemessen und vorausschauend in seiner Umgebung zu agieren. Dies beinhaltet die Fähigkeit, Sinneseindrücke wahrzunehmen und darauf zu reagieren, Informationen aufzunehmen, zu verarbeiten und als Wissen zu speichern, Sprache zu verstehen und zu erzeugen, Probleme zu lösen und Ziele zu erreichen.[11]

Machine Learning
Machine Learning (ML) ist ein Teilbereich der Künstlichen Intelligenz (KI), der sich auf die Entwicklung von Algorithmen konzentriert, die es einem Computer ermöglichen, aus Erfahrungen (Daten) zu lernen und sich zu verbessern, ohne explizit programmiert zu sein. ML-Algorithmen bauen ein statistisches Modell auf, das auf Trainingsdaten beruht und welches gegen die Testdaten getestet wird. Es werden nicht einfach die Beispiele auswendig gelernt, sondern Muster und Gesetzmäßigkeiten in den Lerndaten erkannt.
Es gibt verschiedene Arten von maschinellem Lernen, darunter überwachtes Lernen, unüberwachtes Lernen, halbüberwachtes Lernen und bestärkendes Lernen. Beim überwachten Lernen wird das Modell mit Eingabe-Ausgabe-Paaren trainiert und das Ziel ist es, eine Funktion zu lernen, die Eingaben auf Ausgaben abbildet. Beim unüberwachten Lernen hingegen stehen keine Ausgabedaten zur Verfügung und das Ziel ist es, die Struktur in den Eingabedaten zu finden. Halbüberwachtes Lernen ist eine Kombination aus überwachtem und unüberwachtem Lernen, bei dem sowohl markierte als auch unmarkierte Daten für das Training verwendet werden. Beim bestärkenden Lernen lernt ein Agent, wie er sich in einer Umgebung verhalten soll, um eine Belohnung zu maximieren. [12]

Natural Language Processing
Natural Language Processing (NLP) bezieht sich auf den Zweig der Informatik – und insbesondere der KI – der sich damit befasst, Computern die Fähigkeit zu geben, Text und gesprochene Sprache auf dieselbe Art und Weise zu verstehen wie Menschen. NLP kombiniert Computerlinguistik — regelbasiertes Modellieren natürlicher Sprache — mit statistischem maschinellem Lernen und Deep-Learning-Modellen. Es umfasst eine Reihe von Techniken zur Interpretation menschlicher Sprache, oft mit dem Ziel, menschliche Sprache so zu verstehen und zu manipulieren, wie es Menschen tun. NLP umfasst sowohl die Verarbeitung (Verstehen) als auch die Generierung (Erzeugung) von natürlicher Sprache.[13]

Large Language Model
Large Language Models (LLMs) sind eine Art von maschinellem Lernen Modell, das darauf abzielt, menschenähnliche Texte zu erzeugen. Sie sind eine Unterklasse von Natural Language Processing (NLP) Modellen und verwenden Techniken aus dem Bereich des Deep Learning, um große Mengen an Textdaten zu verarbeiten und zu lernen, wie man menschenähnliche Texte erzeugt.
LLMs sind in der Regel neuronale Netzwerke, die auf einer Architektur namens Transformer basieren. Transformer-Modelle verwenden eine Technik namens "Aufmerksamkeit", um zu bestimmen, welche Teile eines Textes für die Vorhersage des nächsten Wortes relevant sind. Dies ermöglicht es ihnen, den Kontext über lange Distanzen hinweg zu berücksichtigen und komplexere Muster in den Daten zu erkennen. [14]

4.3 Datenschutz

SOC 2
Die SOC 2-Zertifizierung, entwickelt vom American Institute of Certified Public Accountants (AICPA), konzentriert sich auf die Sicherheit, Verfügbarkeit, Verarbeitungsintegrität, Vertraulichkeit und Datenschutz von Informationen in Serviceorganisationen. [15] Es definiert fünf Trust Service Criteria: Sicherheit, Verfügbarkeit, Integrität der Verarbeitung, Vertraulichkeit und Datenschutz. Diese Kriterien sind grundlegend für die Sicherheit und Integrität von Daten. SOC 2-Konformität dient nicht nur als Compliance-Label, sondern auch als Vertrauensanker für Kunden und als Wettbewerbsvorteil für Unternehmen, insbesondere in den Bereichen IT-Dienstleistungen und Cloud-Services. Es trägt auch wesentlich zum Risiko-management bei, indem es Unternehmen hilft, Sicherheitslücken zu identifizieren und zu schließen, das Risiko von Datenschutzverletzungen zu minimieren und gesetzliche Anforderungen zu erfüllen. Die SOC 2-Zertifizierung wird durch Certified Public Accountants in einem sogenannten SOC 2-Audit durchgeführt. [16]

DSGVO
Die Datenschutz-Grundverordnung (DSGVO) ist eine rechtliche Rahmenrichtlinie der Europäischen Union, die am 25. Mai 2018 in Kraft trat. Ihr Hauptziel ist der Schutz personenbezogener Daten innerhalb der EU und die Festlegung einheitlicher Standards für deren Verarbeitung. Die DSGVO ersetzt die vorherige Datenschutzrichtlinie von 1995 und berücksichtigt technologische Entwicklungen. Ein zentraler Aspekt ist die extraterritoriale Anwendbarkeit auf Organisationen, die Daten von EU-Bürgern verarbeiten, unabhängig vom Standort. Die Verordnung definiert klar personen-bezogene Daten und gewährt betroffenen Personen umfassende Rechte über ihre Daten. Rechtmäßigkeit, Fairness und Transparenz sind grundlegende Prinzipien, und die Verordnung betont Datenminimierung, Integrität und Vertraulichkeit. Verantwortliche und Auftrags-verarbeiter müssen strenge Anforderungen erfüllen, und die DSGVO sieht Geldbußen für Verstöße vor. Die Einführung der DSGVO markiert einen Paradigmenwechsel im Datenschutzrecht, indem sie den Schutz personenbezogener Daten als grundlegendes Menschenrecht anerkennt und Organisationen zu proaktiven Sicherheitsmaßnahmen verpflichtet. [17]

4.4 Herausforderung bei KI-unterstützter Softwareentwicklung

Vorgabe im Zuge der Projektarbeit war es auch, sich kritisch mit der Benutzung von generativen KI-Tools auseinanderzusetzen. Drei große Punkte sind ins-besondere bei der Verwendung von Codevervoll-ständigungstools zu beachten. Halluzinationen, die zur Erzeugung syntaktisch unkorrekten Codes führen kann, KI-Demenz, die das Sprachmodell zukünftiger Generationen verschlechtern kann, und urheberrechtliche Aspekte.

Halluzination
In der Künstlichen Intelligenz (KI) bezieht sich das Phänomen der Halluzination auf die Tendenz von Modellen, insbesondere von Large Language Models (LLMs), Informationen zu erzeugen, die nicht in den Eingabedaten vorhanden sind. Diese "halluzinierten" Informationen können in Form von zusätzlichen Details, falschen Behauptungen oder inkorrekten Interpretationen auftreten. [18]
Halluzinationen treten häufig auf, wenn ein Modell versucht, eine Aufgabe zu erfüllen, für die es nicht ausreichend trainiert wurde oder die über seine Fähigkeiten hinausgeht. Zum Beispiel könnte ein Modell, das darauf trainiert wurde, Texte zu generieren, "halluzinieren", indem es Details hinzufügt, die nicht in den ursprünglichen Eingabedaten vorhanden waren, oder indem es Behauptungen aufstellt, die nicht durch die Daten gestützt werden.
Das Phänomen der Halluzination ist ein aktives Forschungsgebiet in der KI. Forscher versuchen, die Ursachen für Halluzinationen zu verstehen und Methoden zu entwickeln, um sie zu verhindern oder zu minimieren. Einige Ansätze beinhalten die Verbesserung der Trainingsdaten, die Verwendung von Techniken zur Überwachung und Kontrolle der Modellausgabe und die Entwicklung von Methoden zur Erkennung und Korrektur von Halluzinationen. [19]
Halluzinationen können ein Zeichen dafür sein, dass ein Modell über seine Grenzen hinaus arbeitet. Sie können auch dazu führen, dass ein Modell ungenaue, irre-führende oder sogar schädliche Informationen erzeugt. Daher ist es wichtig, bei der Verwendung von KI-Modellen Vorsicht walten zu lassen und sicherzustellen, dass die Modelle ordnungsgemäß überwacht und kontrolliert werden. [20]

KI-Demenz

Ein weiterer bedeutender Aspekt bei der Anwendung künstlicher Intelligenz ist das Phänomen der KI-Demenz, das in KI-Modellen auftritt, die auf künstlich generiertem Text basieren. Eine wissenschaftliche Untersuchung von Stanford University und University of California, Berkeley, zeigte, dass GPT-4 im März 2023 bestimmte Aufgaben effektiver bewältigte als im Juni desselben Jahres. Beispielsweise erkannte GPT-4 im März 2023 97,6% aller Primzahlen richtig, doch dieser Anteil sank bis Juni 2023 auf 2,4%. Formatierungsfehler bei der Generierung von Programmcode nahmen ebenfalls zu, was darauf hinweist, dass das Verhalten eines Sprachmodells in kurzer Zeit erheblichen Veränderungen unterliegen kann. [21]

Die Qualität künstlicher Intelligenz hängt wesentlich von der Güte des zugrunde liegenden Datensatzes ab. Von Menschen erstellte Datensätze gewinnen an Bedeutung, wie Forscher von Oxford, Cambridge und London feststellten. Trotz vielversprechender Ergebnisse durch Training mit von Menschen generierten Daten besteht die Prognose, dass zukünftige Sprachmodelle auch mit Daten trainiert werden, die von vorherigen Modellen generiert wurden. Dies könnte zu irreversiblen Defekten führen, wie in einer Studie beobachtet, wo am Ende kein sinnvoller Text mehr generiert wurde.[22] Professor Arvind Narayanan von der Princeton University kritisiert die Studie, da die Verschlechterung der Ergebnisse möglicherweise auf die gestellten Aufgaben, insbesondere das Schreiben von Programmcodes und das Lösen von mathematischen Aufgaben, zurückzuführen sei. OpenAI selbst bestreitet einen Leistungsabfall und argumentiert, dass Anwender die Qualität der Sprachmodelle subjektiv und im Laufe der Zeit kritischer bewerten. Die mangelnde Offenlegung von Trainings- und Modifikationsdetails durch Open-AI erschwert jedoch Vorhersagen zur zukünftigen Entwicklung. [23]

Im Kontext der KI-unterstützten Softwareentwicklung bleibt zu beobachten, wie sich verschiedene Sprachmodelle entwickeln. Von der Verschlechterung des Modells GPT-4 wäre zunächst GitHub Copilot betroffen, Tabnine, Codeium oder SAP-Modelle könnten zukünftig auch betroffen sein. Insgesamt erfordert das Phänomen der KI-Demenz, ähnlich wie Halluzination, eine sorgfältige Überwachung der Modelle.

Urheberrechtsaspekte

Urheberrechtsaspekte sind in der KI-unterstützten Softwareentwicklung relevant in zwei Richtungen: Die generierten Inhalte können urheberrechtlich geschützt sein, jedoch können Programme, die hauptsächlich durch KI entwickelt wurden, möglicherweise keinen Schutz mehr genießen. [24]

Die Verwendung urheberrechtlich geschützter Trainingsdaten für KI-Modelle kann zu rechtlichen Problemen führen. Ein aktuelles Beispiel ist eine Klage der New York Times gegen OpenAI wegen der Verwendung urheberrechtlich geschützter Zeitungsartikel für das Training von Sprachmodellen. [25]

Es ist unklar, wie diese Fälle ausgehen werden und welche Auswirkungen sie auf den deutschen Rechtsraum haben werden. Darüber hinaus sind die generierten Inhalte von KI nicht automatisch urheberrechtlich geschützt, was zu weiteren rechtlichen Komplikationen führen kann, insbesondere wenn sie mit eigenen kreativen Werken vermischt werden. [26]

4.5 Analysierte Tools

GitHub Copilot, ein KI-basiertes Code-Completion- und Code-Generierungs-Tool, wurde in Kooperation zwischen GitHub und OpenAI entwickelt.[27] Es fungiert als Erweiterung für Visual Studio Code und analysiert öffentlich verfügbaren Code auf GitHub, um Entwicklern kontextbezogene Vorschläge für die Codevervollständigung zu bieten.[28] Basierend auf GPT-3 ermöglicht GitHub Copilot nicht nur einfache Code-Snippets, sondern auch komplexe Funktionen und Algorithmen. Es wird von über 27 Prozent aller Entwickler, insbesondere von Python-Entwicklern (40 Prozent), genutzt, wobei mehr als ein Viertel der Codevorschläge angenommen wird. [29]

Tabnine ist ein weiterer KI-basierter Codevervollständigungsassistent, der die allgemeine Produktivität steigern soll. Mit Funktionen wie automatischer Vervollständigung, Kommentieren und Datenschutz kann Tabnine in verschiedenen Programmiersprachen und IDEs eingesetzt werden.[30] Die KI-basierte Codevervollständigung kann lokal ausgeführt werden, ohne persönliche Daten oder Codes zu teilen. Tabnine respektiert laut eigenen Angaben den Datenschutz und Sicherheit und verwendet keine nicht erlaubten Quellcodes für sein Training.[31]

Codeium ist ein KI-basiertes Tool, das als Programmierassistent fungiert und intelligente Vorschläge für Codegenerierung und Navigation bietet. Es unterstützt über 70 Programmiersprachen und kann in bevorzugten Code-Editoren installiert werden. Mit Funktionen wie Autocomplete und Chat optimiert Codeium den Codierungsprozess. Es respektiert nach eigenen Angaben ebenfalls Privatsphäre und Sicherheit, speichert oder verkauft keine persönlichen Daten oder Codes und verwendet keine nicht erlaubten Quellcodes für sein Training. Codeium bietet kostenlose und kostenpflichtige Optionen für individuelle und Teamnutzung. [32]

5. EXPERTENINTERVIEWS

Mit den Experteninterviews wurden in diesem Projekt zwei konkrete Ziele verfolgt:
Das erste Ziel war von der Firma vorgegeben: Die Softwareentwickler bei der T. CON sollen für die neuen Technologien an denen erforscht werden sensibilisiert werden, und es soll erörtert werden, wie der Wissenstand über generative KI im Unternehmen ist. Dieses Wissen ist allerdings nicht relevant für die spätere Nutzwertanalyse und wird daher nicht näher erläutert.
Das zweite Ziel war es, konkrete Anforderungen an ein Tool für die spätere Nutzwertanalyse herauszufinden. Das sind zunächst die IDEs, welche verwendet werden.
Außerdem wird erörtert, mit welchen Programmiersprachen in welchen Umgebungen programmiert wird.

5.1 Interviewleitfaden

Der Interviewleitfaden ist mit dem Betreuer der Abschlussarbeit erstellt worden. Er enthält die zu stellenden Fragen, und mögliche Rückfragen. Er erleichtert später beim Interview auf verschiedene Antworten unterschiedlich zu reagieren, um dem Interview einen größeren Nutzen zu bieten. [7] Die ersten drei Fragen dienen dazu, Kenntnisse der Mitarbeiter bezüglich der Thematik der Abschlussarbeit zu erlangen. Anschließend wird nach den benutzten Programmiersprachen und Entwicklungsumgebungen gefragt. Danach sollen die Experten ihre Meinung zu Vor- und Nachteilen KI-unterstützter Softwareentwicklung nennen, sowie, welche Datenschutz- und Complianceanforderungen wichtig sind. Schließlich noch die bereits genannte Möglichkeit, Wünsche und Anregungen zu nennen. Es ist üblich, den Interviewleitfaden einem Pre-Test zu unterziehen, um sicherzustellen, dass Fragen verständlich formuliert sind. In regelmäßigen Abstimmungen mit dem Firmenbetreuer wurden die Fragen auf Verständlichkeit geprüft, jedoch wurde auf umfangreiche Pre-Tests verzichtet.

5.2 Kontaktierung der Interviewpartner

Die Auswahl der Interviewpartner hat Einfluss auf die Güte der Datenerhebung. [7] Sie erfolgte in Abstimmung mit dem Betreuer. Die interviewten Personen stellen einen Querschnitt der Softwareentwickler bei der T. CON dar. Sie sind im September 2023 über Microsoft Teams interviewt worden. Die 20-minütigen Interviews beinhalteten eine Präsentation des GitHub Copiloten, um einen ersten Eindruck der Experten zu bekommen.

5.3 Durchführung

Die Durchführung beinhaltete einen Dank für die Teilnahme, eine Einverständniserklärung zur Aufnahme des Interviews, sowie eine kurze Beschreibung des Ziels des Interviews. Am Schluss wird noch auf einen auszufüllenden Fragebogen hingewiesen.

5.4 Transkription und Kodierung

Die Aufzeichnungen wurden wörtlich transkribiert, wobei auf Füllwörter wie ”Äh“ verzichtet wurde. Das Transkript wurde nicht gekürzt, jedoch wurden Namen und andere Angaben anonymisiert, um die Vertraulichkeitsvereinbarung zu gewährleisten. Anschließend wird die unübersichtliche und umfangreiche Materialsammlung codiert. Dabei werden die Kernaussagen identifiziert. Die Codierung erfolgte pro Antwort in Tabellenform.

5.5 Ergebnis der Interviews

Die Ergebnisse der Experteninterviews beinhalten nicht nur die Kernaussagen, sondern bestimmen das weitere Vorgehen der Abschlussarbeit.

Programmiersprachen
Acht Experten bei der T. CON arbeiten mit ABAP, sieben mit JavaScript, vier mit TypeScript, drei mit Python, zwei mit HTML/XML, und je ein Experte mit C++ und C. JavaScript und TypeScript sind ausschließlich im UI5-Umfeld anzufinden.
Für die Integration von KI-Tools in die Softwareentwicklung liegt der Fokus auf JavaScript, TypeScript und Python. ABAP wurde in Absprache mit dem Firmenbetreuer bei der anschließenden Bewertung der KI-Tools außen vorgelassen, da die Hoffnung auf SAP interne Tools in der Zukunft besteht, und in einem anderen Projekt bereits ein firmeninternes generatives KI-Modell auf ABAP-Code gebaut wird.

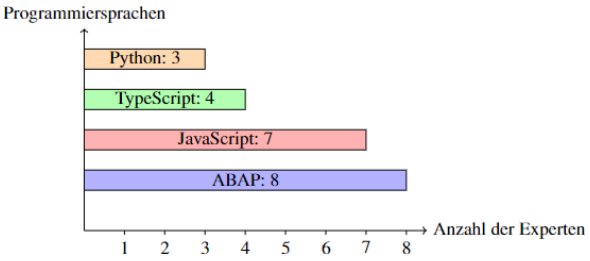

Abbildung 1: Häufigkeit berücksichtigter Programmiersprachen

IDEs
In Bezug auf die verwendeten Integrated Development Environments (IDEs) nutzen sechs Entwickler Eclipse, acht benutzen Visual Studio Code, drei arbeiten mit PyCharm, und ein Experte verwendet Jupyter Notebook. Daher stehen Eclipse, Visual Studio Code, PyCharm und Jupyter Notebook im Mittelpunkt der weiteren Untersuchungen zur KI-unterstützten Softwareentwicklung.

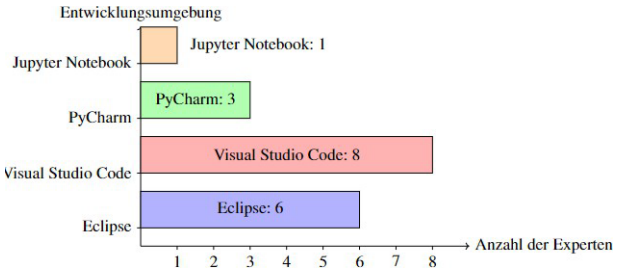

Abbildung 2: Verwendete Programmiersprachen

6. NUTZWERTANALYSE

Im Anschluss an die Experteninterviews wird die Nutzwertanalyse durchgeführt, welche auf den Ergebnissen der Experteninterviews aufbaut. Die Nutzwertanalyse, auch als „Multi-Criteria Decision Analysis (MCDA)“ bekannt, ist eine Methode, die in verschiedenen wissenschaftlichen Disziplinen und in der Praxis eingesetzt wird, um fundierte Entscheidungen zu treffen und alternative Lösungen zu bewerten. Sie ist ein strukturiertes Verfahren zur Bewertung von verschiedenen Alternativen anhand vordefinierter Kriterien. Die Nutzwertanalyse ermöglicht den Vergleich verschiedener Optionen, indem sie objektive Kriterien verwendet, um den Nutzen oder Wert jeder Option zu bestimmen. Dies geschieht durch die Zuweisung numerischer Werte zu den Kriterien und die Aggregation dieser Werte zu einer Gesamtbewertung. [33]
Die Nutzwertanalyse kann entweder als permanentes Tool für langfristige Überwachung oder als einmalige Analyse durchgeführt werden. Diese Abschlussarbeit wählt die zweite Variante, wobei beachtet wird, dass mit zunehmendem technologischem Fortschritt, beziehungsweise neuen Generationen der Vervollständigungstools, die Ergebnisse dieser Projektarbeit veraltet sein können. [8] Im Folgenden Abschnitt werden Teile der im Zuge der Abschlussarbeit durchgeführten Nutzwertanalyse erläutert.

6.1 Festlegung und Definition der Bewertungskriterien

Zunächst werden Kriterien festgelegt, um die Optionen zu bewerten. Diese Kriterien repräsentieren die Anforderungen der T. CON an ein Codevervollständigungstool und werden als Entscheidungsvariablen bezeichnet. Entscheidungsvariablen müssen sinnvoll unterscheidbar und vergleichbar sein. Die Kriterien wurden in Absprache mit dem Betreuer durch Brainstorming, Vorauswahl und Sortierung festgelegt. In dieser Arbeit wurden die Kriterien vom Verfasser definiert und vom Firmenbetreuer genehmigt. Es gibt in der Abschlussarbeit neun Bewertungskriterien:

Anwenderfreundlichkeit: Die Evaluierung der Anwenderfreundlichkeit von KI-Tools erfolgt anhand der Einschätzung ihrer Benutzerfreundlichkeit, welche sich auf die mühelose und intuitive Nutzbarkeit des Tools durch den Anwender bezieht. Hierbei wird geprüft, inwiefern die Softwareanwendung leicht erlernbar ist und ob die Notwendigkeit von Schulungen oder technischem Fachwissen eine Rolle spielt.

Community: Eine Community stellt eine Gruppe von Entwicklern, Benutzern und Experten dar, welche das KI-Tool nutzen oder unterstützen. Zwischen ihnen findet in einem Forum regelmäßiger Austausch statt, um Wissen und Ressourcen zu teilen. Die Community soll bei Fragen und allgemeinen Themen helfen können, zum Beispiel mit Hilfe eines Forums.

Datenschutz: Beim Datenschutzaspekt wird betrachtet, wie gut das zu testende Tool die Sicherheit von sensiblen Informationen gewährleistet. Dies schließt die Speicherung, Verarbeitung und den Zugriff auf die Daten mit ein. Es wird berücksichtigt, ob eingegebene Daten wieder zum Training des KI-Tools verwendet werden.

Geschwindigkeit und Leistung: Die Geschwindigkeit und Leistung eines KI-Tools bezieht sich auf die Reaktionszeit bei der Ausführung von Aufgaben. Es ist wichtig zu bewerten, ob das Tool in Echtzeit arbeitet und ob es komplexe Aufgaben effizient und mit hoher Qualität bewältigen kann, ohne den Programmierfluss zu stören.

Integration in benutzte IDE: Die Integration in benutzte integrierte Entwicklungsumgebungen (IDEs) betrifft die Fähigkeit des KI-Tools, nahtlos in vorhandene Entwicklungsworkflows integriert zu werden. Bei dieser Abschlussarbeit wird auch berücksichtigt, ob die IDEs, in die die Tools integriert werden können, bei der T. CON angewendet werden. Durch die Experteninterviews wissen wir, dass wir die Integration in VS Code, Eclipse, PyCharm sowie in Jupyter Notebook analysieren.

Kosten: Es werden die Kosten pro Zeitabschnitt, wie Monat, Jahr beziehungsweise pro Wort analysiert, die für jeden Benutzer aufkommen. Es wird auch analysiert, welche alternativen Kostenmodelle es gibt.
Qualität der Ergebnisse: Bei den Ergebnissen wird einerseits überprüft, welche Programmiersprachen die Tools unterstützen, aufgrund des Experteninterviews also insbesondere JavaScript und Python.
Des Weiteren ist es wichtig zu prüfen, ob das Tool syntaktische oder semantische Fehler macht und ob die Ergebnisse sinnvoll und konsistent sind. Es wird analysiert, inwiefern Ergebnisse aufeinander aufbauen, oder im Kontext generiert werden.

Support: Der Support für ein KI-Tool umfasst technischen Support, Dokumentationen und Schulungsmaterialien, die bei der Verwendung des Tools unterstützen.

Urheberrecht: Bei der Bewertung urheberrechtlicher Aspekte sind zwei Dinge zu beachten: Wem gehören die im Zuge der Softwareentwicklung generierten Inhalte und wurden beim Training des Sprachmodells urheberrechtliche Verstöße begangen.

6.2 Gewichtung der Kriterien

Jedes Kriterium erhält eine Gewichtung, um seine Bedeutung in der Gesamtbewertung widerzuspiegeln. Dies kann durch Expertenbefragungen oder mithilfe von Analysetechniken erfolgen. Die Gewichtungen zusammengenommen ergeben 100%. Die hohe Anzahl von Aspekten in dieser Nutzwertanalyse, neun an der Zahl, erschwert eine sinnvolle Gewichtung. Bei der sogenannten Nivellierungsverzerrung kommt es durch jedes weitere hinzugefügte Kriterium, welchem ein Gewicht zugeordnet werden muss, eine Bedeutungsverschiebung zuungunsten wichtiger Aspekte. [8] Dass es bei dieser Abschlussarbeit zu Nivellierungsverzerrung kam, zeigt die untenstehende Tabelle. Die niedrigste Gewichtung beträgt 8%, die höchste 14%. Das ist keine große Varianz.
Die Gewichtung der Kriterien ergab sich aus dem Fragebogen, den jeder Experte im Anschluss an das Experteninterview ausfüllen sollte. Bei diesem Fragebogen hat jeder Experte die Wichtigkeit der Aspekte von 0-5 pro Aspekt ausgefüllt. Die Tabelle zeigt, wie die Experten jeden Aspekt hinsichtlich der Wichtigkeit eingeschätzt haben. Diese Zahlen werden anschließend auf 1 (100%) normiert.

Kriterium	Summe	Gewichtet
Anwenderfreundlichkeit	48	11
Community	36	8
Datenschutz	58	13
Geschwindigkeit	46	11
IDE Integration	54	13
Kosten	37	9
Qualität	58	14
Support	42	10
Urheberrecht	49	11
Summe	481	100

Abbildung 3: Gewichtung der Kriterien

6.3 Bewertung der Alternativen

Anwenderfreundlichkeit
GitHub Copilot war das ursprünglich erschienene Tool. Abbildung 5 zeigt das Interface des GitHub Copiloten in einem UI5 Beispielprogramm. Links ist die Chatfunktion zu sehen. Das Tool erkennt, dass in dem Beispiel mit Java-Script programmiert wurde, und schlägt eine Frage hinsichtlich Unittests für JavaScript vor. Rechts ist die Codevervollständigungsfunktion zu sehen. Anhand des Funktionsnamen `onShowHappynewyear` wird berechnet, dass der Anwender einen MessageToast mit der Aufschrift „Happy new Year!“ wünscht. Anhand der Funktionsweisen von Tabnine und Codeium zeigt sich, dass diese Tools stark am GitHub Copilot angelehnt sind, und somit eine vermeintlich bessere Konkurrenz darstellen wollen.
Alle drei Tools zeichnen sich durch intuitive und leicht erlernbare Benutzeroberflächen aus. Der Installationsprozess verläuft für alle drei Tools zügig und unkompliziert. Nach der Installation erscheint jeweils ein Icon, das den Zugang zum entsprechenden Tool ermöglicht.
Den Benutzeroberflächen fehlen jeweils wichtige Informationen zur vollständigen Nutzung der Funktionen, insbesondere zur Navigation durch Codealternativen. Da diese Funktionen leicht erlernt werden können fällt das allerdings nicht weiter ins Gewicht, und alle drei Tools bekommen hinsichtlich Anwenderfreundlichkeit **fünf Punkte**.

Datenschutz
GitHub betont, dass der Copilot derzeit noch nicht zertifiziert ist. Es wird angestrebt, eine SOC 2-Zertifizierung bis Mai 2024 von einem Drittanbieter zertifiziert zu bekommen. Auch betont GitHub, dass der generierte Code nicht zum Training zukünftiger Modelle benutzt wird.[34]
Tabnine benutzt ebenfalls keinen vorgeschlagenen Code zum Training des Modells. Zusätzlich ist es hinsichtlich der Einhaltung der Datenschutzgrundverordnung und nach dem SOC 2-Standard zertifiziert. Das Audit und sämtliche Zertifikate können heruntergeladen werden.[35]
Codeium benutzt ebenfalls keine Codevorschläge, um das Model weiter zu trainieren. Außerdem ist Codeium ebenfalls SOC 2-Zertifiziert. Das Zertifikat ist veröffentlicht, das komplette Audit allerdings nicht. [36]
Zusammengefasst benutzt keines der drei Tools Codevorschläge, um das Model weiter zu trainieren. Insgesamt zeigt sich somit, dass Tabnine und Codeium bereits wichtige Schritte im Bereich Datenschutz vollzogen haben. Besonders positiv hervorzuheben ist, dass Tabnine alle relevanten Informationen hinsichtlich beider Zertifizierungen zur Verfügung stellt. GitHub muss für den Copiloten einiges aufholen, positiv sind allerdings die Bemühungen zur Zertifizierung. Daher wird der Copilot mit **zwei Punkten**, Tabnine mit **fünf Punkten** und Codeium mit **vier Punkten** hinsichtlich des Datenschutzes bewertet.

IDE Integration
Abbildung 4 zeigt, welches Tool in welche der berücksichtigten IDEs integrierbar sind. Codeium ist hier das flexibelste Tool, da es nicht nur in VS Code, Eclipse, PyCharm und Jupyter Notebook integrierbar ist, sondern man das Tool auch vorab im Browser testen kann. Tabnine und der GitHub Copilot sind jeweils in VS Code integrierbar. Der GitHub Copilot zusätzlich in PyCharm, Tabnine zusätzlich in Eclipse.

IDE	GitHub Copilot	Tabnine	Codeium
Browser	Nein	Nein	Ja
VS Code	Ja	Ja	Ja
Eclipse	Nein	Ja	Ja
PyCharm	Ja	Nein	Ja
Jupyter Notebook	Nein	Nein	Ja

Abbildung 4: Integrationsmöglichkeit der Tools in IDEs

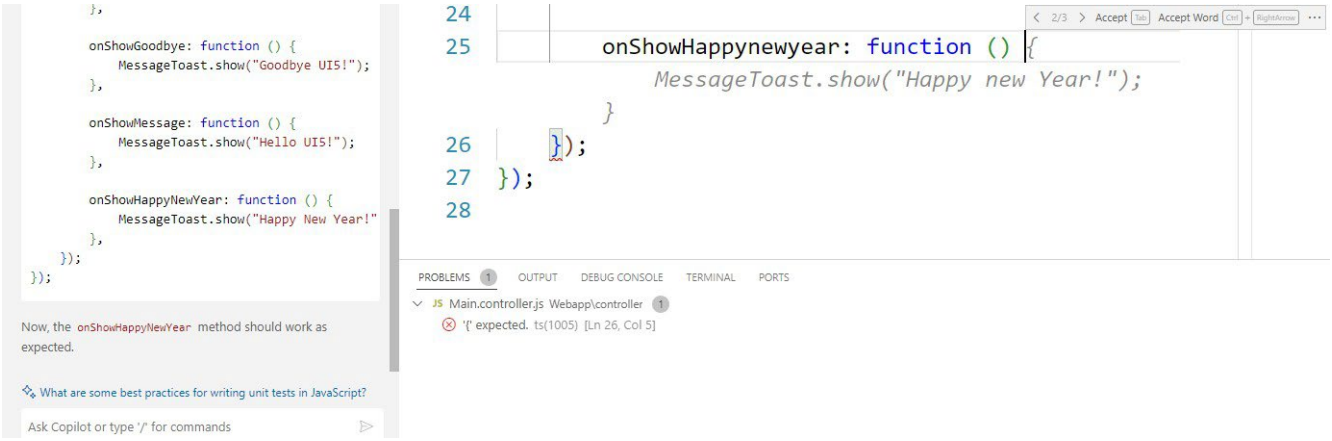

Abbildung 5: Interface des GitHub Copilot

Bei der Bewertung wird berücksichtigt, wie viele Experten welches Tool nutzten. Da VS Code von den meisten Experten benutzt wird, fällt es sehr positiv ins Gewicht, dass alle drei KI-Tools auch eine Integration in diese IDE unterstützen. Dass Tabnine zusätzlich in Eclipse integrierbar ist fällt positiver ins Gewicht als die Integrierbarkeit des GitHub Copilot in PyCharm, da Eclipse mit sechs Experten höher gewichtet ist als PyCharm mit zwei.
In der Gesamtschau erhält der GitHub Copilot daher in dieser Kategorie **drei Punkte**. Tabnine erhält **vier Punkte**, Codeium erhält **fünf Punkte**.

Kosten
Positiv zu bewerten ist bei allen drei Tools, dass es verschiedene Preismodelle gibt, welche den unterschiedlichen Anforderungen, verschiedener Kunden gerecht wird.
Dass das günstigste Preismodell des GitHub Copilot nicht für alle kostenlos ist, sondern 10 US-Dollar im Monat pro Nutzer kostet fällt negativ ins Gewicht. Auch ist negativ anzumerken, dass es „Copilot Enterprise“ nur für Unternehmen gibt, welche die GitHub Enterprise Cloud haben, was die tatsächlichen monatlichen Kosten auf 60 US-Dollar im Monat erhöht. Allerdings bietet das kostengünstigere Preismodell „Copilot Business“ mit 20 US-Dollarn im Monat bereits eine breite Produktpalette, welche für Firmen wichtig sind. Aufgrund der negativen Aspekte gibt es trotz des Modells „Copilot Business“ mit hervorragendem Preisleistungsverhältnis **vier Punkte** im Aspekt Kosten.
Bei Tabnine ist positiv hervorzuheben die kostenlose Version „Starter“, um das Tool auszuprobieren. Allerdings ist das nur ein Preismodell, um die grundlegende Funktion Tabnines zu testen, die 2-3 Wörter, welche vorgeschlagen werden, bieten keine Produktivitätssteigerung. Das zweite Modell „Pro“ für 12 US-Dollar im Monat pro Nutzer ist von den Funktionalitäten vergleichbar mit „Copilot Individual“ und damit teurer als der Copilot. Besonders negativ ins Gewicht fällt, dass Tabnine Unternehmen mit weniger als 100 Softwareentwicklern vom dritten Preismodell „Pro“ abrät, sodass es keine Funktionen wie VPC beziehungsweise finegetunte Modelle gibt.
Außerdem ist die Preisintransparenz, den Preis gibt es nur auf Anfrage, hinsichtlich des dritten Preismodells negativ anzumerken. Aufgrund der aufgezählten Aspekte wird Tabnine hinsichtlich der Kosten mit **drei Punkten** bewertet. Bei Codeium fällt besonders positiv ins Gewicht, dass es die kostenlose Version „Individual“ mit bereits breiter Produktpalette, wie einen chatbasierten KI-Assistenten und Codevervollständigung in Echtzeit gibt. Dass das zweite Preismodell „Teams“ für 12 US-Dollar pro Nutzer und Monat im Jahresabonnement bereits fortgeschrittene Funktionen wie Nutzermanagement und eine personalisierte Codebasis bietet fällt, äußerst positiv ins Gewicht. Codeium ist damit hinsichtlich Preisleistungsverhältnisses das günstigste Tool. Auch bei Codeium gibt es die Preisintransparenz hinsichtlich des Modells „Enterprise“. Da aber wie angemerkt bereits das „Pro“-Modell sehr viele Funktionen beinhaltet, ist die Bewertung von Codeium hinsichtlich der Kosten **fünf Punkte**.

Urheberrecht
Bei der Bewertung der einzelnen Tools hinsichtlich des Urheberrechts wird in zwei Subkategorien aufgeteilt: In der Kategorie **Training des Sprachmodells** wird analysiert, ob beim Training des zugrundeliegenden Modells Urheberrechtsverstöße begangen wurden. In der Kategorie **Rechte am generierten Code** wird darauf eingegangen, welche Rechte die Anbieter der Tools am generierten Code verlangen. In einem folgenden Kapitel wird genauer erläutert, inwiefern das Training des Sprachmodells und die Rechte am generierten Code problematisch sein können.

Training des Sprachmodells:
Tabnine behauptet, dass sein Sprachmodell ausschließlich auf Open-Source-Code trainiert wird, für den das Unternehmen rechtmäßige Lizenzen erworben hat. Welche Trainingsdatensätze benutzt wurden, einschließlich sämtlicher Lizenzen, ist öffentlich zugänglich. Durch diese transparente Vorgehensweise möchte Tabnine nachweisen, dass bei der Modellentwicklung keine Urheberrechtsverletzungen begangen wurden. Die Offenlegung der Trainingsets ermöglicht es einen detaillierten Einblick in die verwendeten Datenquellen zu bekommen, und trägt zur Schaffung von Vertrauen in den Prozess des Modelltrainings bei.

Durch die behauptete konsequente Einhaltung der Lizenzbestimmungen möchte Tabnine seine Verpflichtung zur rechtlich einwandfreien Nutzung von Datenquellen unterstreichen.
Diese Behauptungen sind seitens des Anbieters zwar plausibel vorgetragen worden, dennoch kann nicht mit letzter Sicherheit unabhängig geprüft werden, ob ausschließlich lizensierter Code benutzt wurde. Die Datensätze sind in 36 CSV-Dateien gelistet, wobei jede Datei mehrere 1000 Zeilen beinhaltet, weshalb eine Überprüfung in dieser Abschlussarbeit nicht möglich ist.

Rechte am generierten Code:

In den allgemeinen Geschäftsbedingungen unter Abschnitt 12 betont Tabnine, dass man durch das Erwerben einer Tabninelizenz keinerlei Rechte an den Sprachmodellen erlangt, welche für das Training benutzt wurden. Jedoch wird explizit betont, dass sowohl der entwickelte, als auch der generierte, akzeptierte Code ausschließlich dem Entwickler gehört, und Tabnine auf sämtliche Urheberrechtsansprüche an dem Code oder anderen geistigen Eigentumsansprüche verzichtet, und diese unbefristet selbst genutzt werden können.
In den allgemeinen Geschäftsbedingungen von Tabnine für das kostenpflichtige Preismodell „Teams“ überträgt die Firma im Abschnitt 7.2 dem Nutzer alle Rechte an alles Vorschlägen, die zur Verfügung gestellt, oder zurückgegeben werden. Allerdings betont Tabnine, dass die durch maschinelles Lernen generierten Vorschläge denen anderer Kunden ähneln können, oder sogar übereinstimmen können, sodass dann keine Rechte an den Codevorschlägen gewährt wird.
Das Training des Sprachmodells des GitHub Copilots ist nicht nur Geschäftsgeheimnis, sondern bereits Schauplatz juristischer Auseinandersetzungen geworden. Daher gibt es lediglich **zwei Punkte**. Da Tabnine die Lizenzen zum Training des Sprachmodells sehr detailliert offenlegt, bekommt es in dieser Subkategorie **fünf Punkte**. Codeium verspricht als Alternative zum GitHub Copilot das Modell ausschließlich auf lizensierten Code trainiert zu haben, macht allerdings keine genaueren Angaben hierzu. Daher wird es mit **vier Punkten** bewertet. Da alle drei Tools die Rechte am generierten Code abgeben, wird diese Subkategorie bei allen drei Tools mit **fünf Punkten bewertet**. Dies ergibt aufgerundet einen Gesamtnutzwert hinsichtlich des Urheberrechts von **drei Punkten** für den GitHub Copilot, **fünf Punkte** für Tabnine und **fünf Punkte** für Codeium.

Codequalität

Um die Qualität der drei Codevervollständigungstools zu testen, sind je Tool drei prototypische Anwendungsfälle implementiert worden, welche den Copiloten, Tabnine und Codeium auf Kontextsensitivität, Fähigkeiten im Umfeld von Python, und Fähigkeiten im UI5-Umfeld testen. Für jeden Anwendungsfall sind im Vorfeld konkrete Erwartungen an die Tools formuliert worden, wie sie den Code im Kontext vervollständigen sollen. Nach diesen Erwartungen sind die Tools bewertet worden. Am Ende der Fallstudie wird bewertet, ob die Erwartung erfüllt (e.) oder nicht erfüllt (n.e.) wurde. Alle Anwendungsfälle sind in der Entwicklungsumgebung VS Code durchgeführt worden. Die Prüfung auf Kontextsensitivität war Vorgabe, die Überprüfung auf Python und UI5 ergab sich aus den Experteninterviews.

Kontextsensitivität:

Die Ordnerstruktur des ersten Anwendungsfalls ist in Abbildung 6 zu finden. Bei der Prüfung der Kontextsensitivität wurden in einem Ordner zwei Pythonfiles angelegt.

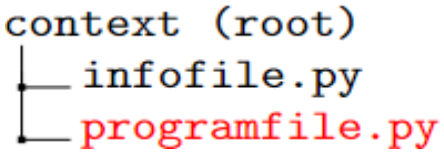

Abbildung 6: Ordnerstruktur Kontextsensitivität

In der Datei `infofile.py` stand Pythonsourcecode. Konkret eine Variable `cptadress`, mit dem Wert `chat.openai.com` initialisiert war. Außerdem eine Funktion, welche für einen Eingabeparameter berechnet, ob er eine Primzahl ist oder nicht.
In der zweiten Datei `programfile.py` wurde unter Benutzung jeweils eines Tools implementiert. Dabei wurde für das jeweilige getestete Tool folgende Erwartungen festgelegt:

1. Zunächst soll die Datei `infofile.py` importiert werden. Spätestens nach dem Tastenanschlag `import my` soll das Codevervollständigungstool erkennen, dass das Infofile zu importieren ist.
2. Anschließend soll eine Variable `my_number` erstellt und mit einem sinnvollen Wert initialisiert werden. Erwartung ist, dass das Tool spätestens nach dem Tastenanschlag `my_num` dies erkennt.
3. Mit Hilfe eines Kommentars, dass die initialisierte Nummer darauf geprüft werden soll, ob sie eine Primzahl ist, soll das Codevervollständigungstool erkennen, dass diese Funktion im Infofile implementiert ist, und diese anwenden.
4. Schließlich wird eine Variable `address` erstellt. Das Codevervollständigungstool soll den Kontext mit der Variable `gptaddress` erkennen, und `address` mit dem gleichen Wert initialisieren.

Anbei die Ergebnisse der drei Codevervollständigungstools:

GitHub Copilot:

1. Bereits nach dem Tastenanschlag von `imp` schlug der Git-Hub Copilot vor, das Infofile zu importieren. (e.)
2. Nach Eingabe von `my_num` vervollständigte der GitHub Copilot die Variable auf `my_number` und initialisierte diese auf den sinnvollen Integerwert 3. (e.)
3. Der Vorschlag, die initialisierte Nummer mit Hilfe der Primzahlfunktion im Infofile zu überprüfen kam automatisch. (e.)
4. Ebenfalls generierte der Copilot einen Printbefehl, in dem stand, dass die Webadresse von gpt `chat.openai.com` ist. (e.)

Dies zeigt, dass der GitHub Copilot hinsichtlich kontext-sensitivität alle Erwartungen erfüllt und übertrifft. Es ist dadurch bewiesen, dass das Tool mindestens den Kontext zwischen zwei Dateien erkennt.

Tabnine:

1. Während des Tastenanschlags wurde bis zum Schluss, als `import myinfofile` bereits eingegeben wurde, kein sinnvoller Vorschlag gemacht. (n.e.)
2. Nach Eingabe von `my_num` wurde auf `my_number` vervollständigt, und diese Variable mit 3 initialisiert, einem sinnvollen Integerwert. (e.)
3. Die Funktion `is_prime` wurde komplett außen vor gelassen. (n.e.)
4. Dagegen vervollständigte Tabnine die Variable `adress = infofile.cptadress(my_number)`. Die String-variable `cpt-adress` wird also aufgerufen. Dies ist syntaktisch unkorrekt, und wirft einen TypeError mit der Fehlermeldung „not callable“. (n.e.)

In der Fallstudie wurde demonstriert, dass Tabnine nicht nur den Kontext nicht korrekt erkennt, sondern auch syntaktisch inkorrekten Code generiert.

Codeium:

1. Codeium hat sofort erkannt, dass das Infofile importiert werden soll. (e.)
2. Die Variable `my_number` wurde per Methodenaufruf der Primzahlfunktion des Infofiles mit Wert 11 initialisiert. Da elf eine Primzahl ist, wurde `my_number` also mit `true` initialisiert. Das ist für eine Variable, deren Name einen Integer- oder Doublewert erwarten lässt also nicht sinnvoll, dennoch syntaktisch korrekt. (n.e.)
3. Damit ist diese Erwartung bereits erfüllt worden. (e.)
4. Der Variable `adress` ist der Wert der Variable `cptadress` des Infofiles zugewiesen. Anschließend wurde die Adresse auf der Konsole ausgegeben. (e.)

Codeium ist somit wie der GitHub Copilot grundsätzlich kontextsensitiv auf mindestens zwei Dateien. Jedoch wird der Kontext teilweise semantisch unkorrekt interpretiert.

Pythonprojekt:

Bei dem Pythonprojekt haben wir einen Ordner mit einer Datei `my_data.csv`. In dieser CSV-Datei sind viele Datensätze. Jeder Datensatz enthält die Variablen `age`, `gender`, `education_level`, `income`, `occupation` und die Zielvariable `target_clomn`. Die Codevervollständigungs-tools sollen anhand der gegebenen Kommentare ein Pythonprojekt generieren, welche Muster innerhalb des Datensatzes suchen. Der Anwendungsfall gliedert sich in zwei Phasen:
In der Datei `data_preprocessing.py` wird der Datensatz in Trainings- und Testdaten aufgeteilt. In der zweiten Datei `train_model.py` wird das Modell mit Hilfe des Trainingssatzes trainiert, evaluiert und gespeichert. Die Ordnerstruktur des Projekts ist in Abbildung 7 zu finden. In den roten Dateien wird implentiert, die blauen Dateien werden durch das geschriebene Pythonskipt generiert.

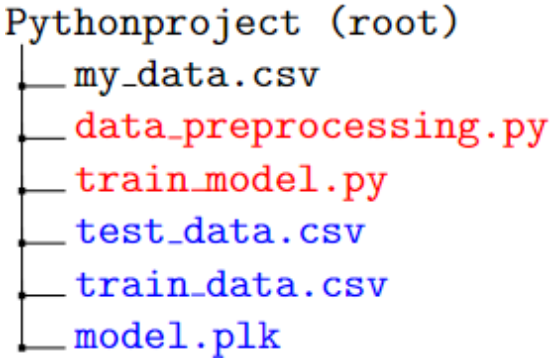

Abbildung 7: Ordnerstruktur des Pythonprojekts

Folgende Erwartungen wurde an das jeweilige Codevervollständigungstool in dem Data Science Projekt gestellt. Die nächsten Schritte ergeben sich aus Python-Kommentaren, die verfasst werden, die Art der Umsetzung aus den importierten Bibliotheken:

1. Da es sich um ein umfangreicheren Anwendungsfall handelt, soll das Codevervollständigungstool objekt-orientiert vorgehen. Es soll also die Funktionen in Methoden schreiben, insbesondere in eine Main-Funktion, und diese aufrufen.
2. Das Programm soll den Datensatz in der Datei `my_data.csv` in Trainings- und Testdaten aufsplitten. Es soll eine sinnvolle Testgröße genommen werden.
3. Bevor das Modell trainiert wird, sollen die nicht numerischen Variablen One-Hot-Codiert werden. Dafür soll das Tool erkennen, welche kategoriale Variablen es in der CSV-Datei gibt. Diese Codierung kategorialer Variablen wird in Data Science Projekten durchgeführt, da Modelle numerischen Input benötigen. [37]
4. Anschließend soll das Modell mit `LinearRegression` trainiert werden. `LinearRegression` ergibt sich aus den importierten Bibliotheken.
5. Die Evaluierung des Modells soll durch die Berechnung des mean squared errors erfolgen.
6. Die Speicherung des Modells soll erfolgen, indem die Funktion `dump` der Bibliothek `joblib` aufgerufen wird.

Hier die Ergebnisse der Tools:

GitHub Copilot:

1. Zu Beginn lieferte der GitHub Copilot Codevorschläge im Skriptstil. Jedoch erkannte das Tool nach der ersten implementierten Funktion, dass auch zukünftig in Funktionen implementiert werden soll. Die Main-Funktion wurde anhand der zuvor implementierten Funktionen in beiden Dateien korrekt umgesetzt. (e.)
2. Die Funktion zum Aufteilen der CSV-Datei in Trainings- und Testdaten ist korrekt umgesetzt worden. Als Testgröße wurde 0.2 gewählt, also 20% des CSV-Datensatzes wird als Testdaten genutzt. Dieser Wert ist Standard. [38] (e.)
3. Die One-Hot-Codierung konnte wurde syntaktisch korrekt umgesetzt. Jedoch der GitHub Copilot aus der CSV-Datei nicht die korrekten kategorialen Variablen auslesen. Diese mussten manuell eingetippt werden. (n.e.)
4. Das Training des Modells wurde anhand der importierten Bibliothek korrekt generiert. (e.)
5. Auch die Evaluierung des Modells wurde korrekt umgesetzt. (e.)
6. Auch die Funktion zum Speichern des Modells wurde korrekt generiert. (e.)

Der GitHub Copilot hat in diesem Data Science Projekt also viele Vorhersagen korrekt getroffen. Außerdem hat sich das Tool an die Bedürfnisse des Entwicklers angepasst, in dem es mit seinen Codevorschlägen zu einer objektorientierten Programmierweise gewechselt hat.

Tabnine:

1. Tabnine hat sich nicht an den objektorientierten Entwicklungsstil angepasst. Die Codevorschläge sind immer wieder in skriptart erstellt worden. Nach Tastenanschlag „def“ ist dann aber immer ein Vorschlag zur Implementierung einer Funktion generiert worden. Die Main-Funktionen konnte das Tool nicht anhand der zuvor implementierten Funktionen generieren. (n.e.)
2. Tabnine hat eine korrekte Funktion generiert, die die Daten in Test- und Trainingsdaten aufteilt. Auch dieses Tool hat die Testgröße auf den Wert 20% gesetzt. (e.)
3. Die One Hot-Codierung wurde korrekt umgesetzt. Jedoch hatte das Tool Probleme die korrekten kategorialen Variablen aus der CSV-Datei zu erkennen. Diese mussten manuell eingefügt werden. (n.e.)
4. Die Funktion zum Training des Modells wurde korrekt umgesetzt. (e.)
5. Auch wurde das Modell korrekt evaluiert. (e.)
6. Die Funktion zum Speichern des Modells wurde ebenfalls korrekt und fehlerfrei generiert. (e.)

Insgesamt zeigt sich, dass Tabnine viel in dem Anwendungsfall korrekt umsetzen konnte. Jedoch wurden die nächsten Schritte, wie Codieren, Trainieren, Evaluieren und Speichern nicht korrekt vorhergesagt, das Tool schlug immer vor, die Daten zu splitten, was, nachdem es bereits umgesetzt wurde nicht mehr sinnvoll war. Außerdem hat Tabnine sich immer an eine skriptweise Implementierung in Python orientiert. Als letztes Manko ist zu erwähnen, dass Tabnine aus den vorhandenen implementierten Funktionen keine fehlerfreie Main-Funktion erstellen konnte.

Codeium:

1. Codeium hat sich ebenfalls nicht an objektorientierter Programmierweise orientiert. Die Codevorschläge änderten sich auch nicht, nachdem bereits manueller Code entwickelt wurde, welcher objektorientierter Programmierweise entspricht. Auch konnte das Programm in der Datei `data_preprocessing.py` die Main-Funktion trotz implementierter Funktionen nicht korrekt umsetzten. In der Datei `train_model.py` wurde sie allerdings korrekt generiert. (e.)
2. Die Funktion zum Aufsplitten des Datensatzes in Trainings- und Testdaten wurde nicht korrekt umgesetzt. Einerseits wurde die Zielvariable nicht korrekt initialisiert. Außerdem wurde keine Testgröße angegeben. Dass die Testgröße 20% ist, musste manuell eingefügt werden. (n.e.)
3. Analog zu den anderen Codevervollständigungstools wurde die One-Hot-Codierung grundsätzlich korrekt umgesetzt. Jedoch hatte auch dieses Tool Probleme, kategoriale Variablen aus der CSV-Datei korrekt auszulesen, weshalb diese manuell eingefügt werden mussten. (n.e.)
4. Die Funktion zum Trainieren des Modells wurde korrekt umgesetzt. (e.)
5. Auch wurde die Funktion zum Evaluieren des Modells anhand des mean squared errors korrekt umgesetzt. (e.)
6. Auch das Speichern des Modells erfolgte fehlerfrei. (e.)

Codeium ist somit in der Lage Pythonprojekte korrekt umzusetzen. Jedoch macht das Tool keine Vorschläge kontextsensitiv zur vorhandenen CSV-Datei. Ebenfalls ändert das Tool seine Codevorschläge nicht hinsichtlich des Programmierstils des Entwicklers und verharrt auf nicht objektorientierter Programmierweise.

UI5-Projekt:

Der dritte Anwendungsfall war ein kleines prototypisches UI5-Projekt. Die Ordnerstruktur des Projektes ist in Abbildung 9 zu finden.
Die Datei `i18n.properties` wird verwendet, um Textressourcen zu speichern. Die Datei enthält Schlüssel-Wert-Paare. Der Schlüssel ist eindeutig, der Wert ist der übersetzte Text. Ein Beispiel in der Anwendung ist der Schlüssel `homePageTitel` mit dem Wert „UI5 Walkthrough“. In der Datei `manifest.json` wird die UI5 Anwendung zentral konfiguriert. Es werden vor allem Abhängigkeiten definiert. Abhängigkeiten in diesem Usecase waren konkret die Pfade zur Datei `i18n.properties`, den CSS, View und Controllerelementen. Die Tools wurden an den rot markierten Dateien `HelloPanel.controller.js` und `component.js` getestet. In der Datei `component.js` wird das UI5 Modell initialisiert. Dabei wird auf die Datei `manifest.json` verwiesen. In der Initialisierungsfunktion wird die Variable `oData`, ein JavaScript-Objekt mit UI5-Daten befüllt.
Die Datei `HelloPanel.controller.js` ist eine Controller-Datei, welche implementiert, wie die Webanwendung beim Klick auf verschiedene HTML-Elemente reagieren soll.

In der Datei `HelloPanel.view.xml` wurden drei Buttons implementiert, die beim Klick das ausführen sollen, was in den Funktionen `onShowHello`, `onOpenDialog` und `onCloseDialog` implementiert wird. Diese drei Funktionen werden in `HelloPanel.controller.js` implementiert. Daher ergeben sich folgende Erwartungen an die Codevervollständigungstools:

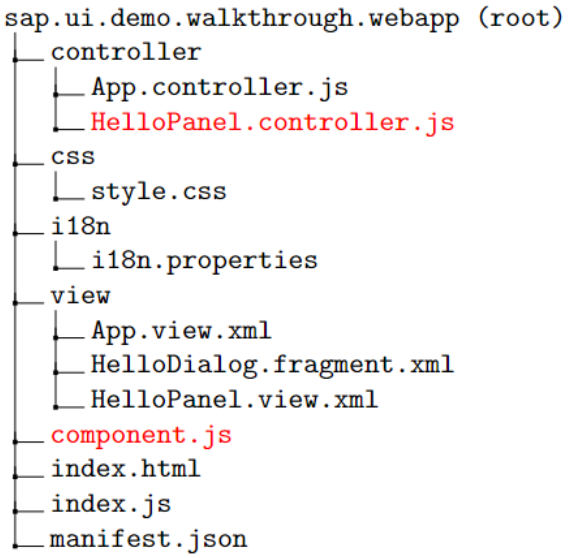

Abbildung 8: Ordnerstruktur des UI5-Projekts

1. In beiden Dateien soll der Modus „use strict“ verwendet werden. Dies ist eine Sicherheitsmaßnahme, die das Programm nur kompilieren lässt, wenn spezielle Programmierkonventionen, wie das nicht Verwenden globaler Variablen eingehalten wird. [39]
2. In der Datei `component.js` wird auf `manifest.json` verwiesen und in der Initialisierungsfunktion die Variable `oData` mit UI5-Daten befüllt.
3. In der Datei `HelloPanel.controller.js` wird erkannt, welche drei Funktionen sich aus dem zugehörigen View-Element `HelloPanel.View.xml` ableiten lassen.
4. In beiden Dateien wird dem Controller mitgeteilt, um welche Datei es sich handelt. Dabei muss der Pfad zur Datei korrekt angegeben werden.

GitHub Copilot:

1. "Use strict" ist korrekt generiert worden. (e.)
2. In der Datei `component.js` wurde die Datei `manifest.json` korrekt eingebunden. Außerdem wurde die `oData`-Variable korrekt mit UI5-Daten befüllt. (e.)
3. Der GitHub Copilot hat korrekt erkannt, welche Funktionen sich aus dem View-Element ableiten lassen. Bei der Funktion `onShowHello` wurde sogar das Key-Value-Paar `HelloMsg` aus der Datei `i18n.properties` benutzt, und sinnvoll in die Implementierung eingefügt. (e.)
4. In beiden Dateien wurde der Pfad zur Datei korrekt angegeben. (e.)

Der GitHub Copilot konnte also in dem Anwendungsfall mit Hilfe seiner Codevorschläge alle Erwartungen erfüllen. Besonders hervorzuheben ist, dass bei der Implementierung der Funktion `onShowHello` der semantisch sinnvolle Kontext zu einem Key-Value-Paar in den i18n-Properties hergestellt wurde.

Tabnine:

1. Tabnine hat "use strict“ in der ersten Datei `component.js` nach Tastenanschlag "u" vervollständigt. In der zweiten Datei wurde "use strict “ automatisch vervollständigt. (e.)
2. Tabnine hat die Notwendigkeit zur Implementierung einer Init-Funktion zwar erkannt, jedoch waren die Codevorschläge in mehrerer Hinsicht syntaktisch inkorrekt. Es wurde noch die Initialisierung einer oData-Variable korrekt vorgeschlagen. Jedoch wurde dieser nicht der Wert "UI“, sondern der sinnlose Wert "John “ zugewiesen. Außerdem erstellte das Programm kein neues JSONModel, mit der oData-Variable als Input. (n.e.)
3. Tabnine konnte nicht erkennen, welche Funktionen der zugehörigen View zu implementieren sind. Es wurden Funktionen vorgeschlagen wie `onAfterRendering`, welche in unserem Anwendungsfall nicht existieren. Nach manueller Vorgabe, die Funktion `onShowHello` zu implementieren kam ein Codevorschlag "HelloWorld“ auf der Konsole auszugeben. Das ist semantisch korrekt, jedoch nicht zielführend. Nach fertiger Implementierung der Funktion schlug Tabnine wiederholt vor die Funktion `onShowHello` wiederholt zu implementieren. (n.e.)
4. Der Pfad, auf den referenziert werden muss, wurde in den Dateien korrekt erkannt. (e.)

Tabnine machte bei der Initialisierung der UI-Komponente syntaktische Fehler. Außerdem konnte das Tool nicht aus dem Kontext heraus erkennen, welche Funktionen im Hello-Panel zu implementieren sind.

Codeium:

1. Codeium hat „use strict“ zweimal korrekt vorgeschlagen. (e.)
2. Codeium hat die Initfunktion generiert, jedoch einen syntaktischen Fehler eingebaut. Der oData-Variable wäre der sinnlose Wert „World“ statt „UI5“ zugewiesen worden. (n.e.)
3. Auch Codeium hat aus dem Viewelement heraus erkannt, welche drei Funktionen zu implementieren sind, und die Implementierung syntaktisch korrekt umgesetzt. (e.)
4. Codeium erkannte in beiden Dateien den Pfad, auf den zu verweisen war. (e.)

Codeium machte in diesem Anwendungsfall Fehler bei der Initialisierung der UI-Komponente. Dennoch konnte das Tools aus dem Kontext heraus erkennen, welche Funktionen zu implementieren sind, und um welche Dateienkomponente es sich jeweils handelt.

Für die Kontextsensitivität konnte ein Punkt erreicht werden. Im Pythonprojekt und im UI5-Projekt jeweils zwei. Anbei steht für jede Bewertung auch die Bewertung der einzelnen Unterkategorien. Der GitHub Copilot wurde mit fünf Punkten (1+2+2), Tabnine mit zwei Punkten (0+1+1) und Codeium mit vier Punkten (1+2+1) in Bezug auf Codequalität bewertet.

		GitHubCopilot		Tabnine		Codeium	
Kategorie	Gewicht	ung.	gew.	ung.	gew.	ung.	gew.
Anwenderfreundl.	11	5	55	5	55	5	55
Community	8	4	32	0	0	4	32
Datenschutz	13	2	26	5	65	4	52
Geschwindigkeit	11	5	55	2	22	5	55
IDE Integration	13	3	39	4	52	5	65
Kosten	9	4	36	3	27	5	45
Qualität	14	5	70	2	28	4	56
Support	10	3	30	5	50	5	50
Urheberrecht	11	4	44	5	55	3	33
Summe	100	35	387	31	354	40	443

Abbildung 9: Tabelle mit den Gesamtnutzwerten

6.4 Berechnung des Gesamtnutzwertes

Die Bewertungen der Einzelkriterien werden mit ihren Gewichtungen multipliziert und summiert (siehe Abbildung 9). Der höchstmögliche Nutzwert liegt bei 500 Punkten. Codeium kommt auf einen Nutzwert von 443 Punkten. Der Copilot auf 387 Punkte und Tabnine auf 354 Punkte. Der Copilot überzeugt durch hohe Qualität in den Ergebnissen der Codegenerierung und ist sehr anwenderfreundlich, muss jedoch im Bereich Datenschutzzertifizierung noch nachholen. Außerdem wäre es schön, wenn er in mehr IDEs integrierbar wäre. Tabnine hält sich zwar an hohe Urheberrechtsanforderungen, kann allerdings insbesondere nicht in der Kategorie Codequalität punkten. Codeium überzeugt mit guter Codequalität, wenn auch etwas schwächer als der GitHub Copilot. In der Kategorie IDE-Integration sticht er hervor.

6.5 Gewichtungsproblem

In dieser Nutzwertanalyse wurde die Gewichtung anhand der Einschätzungen der Experten in den Experteninterviews festgelegt. Jedoch kam es dabei zu Verzerrungen, da sich die Experten relativ uneinig waren. Daher wurde noch einmal mit Alternativen Gewichten berechnet: Anwenderfreundlichkeit 10%, Community 2,5%, Datenschutz 20%, Geschwindigkeit 15%, IDE Integration 10%, Kosten 5%, Qualität 25%, Support 2,5% und Urheberrecht 10%. Die Bewertungen bleiben gleich. Auch mit dieser sinnvolleren Gewichtung hat Codeium mit 435,5 Punkten das höchste Gewicht, gefolgt vom Copiloten mit 397,5 Punkten und Tabnine mit 347,5 Punkten. Das Ergebnis, dass Codeium also den größten Nutzwert hat hält einer sinnvolleren Gewichtung stand.

7. FAZIT

7.1 Kernergebnisse

Nach den Experteninterviews und der Nutzwertanalyse lautet das Kernergebnis, dass Codeium und der GitHub Copilot sich anhand der Bedürfnisse der T. CON eignen, im Softwareentwicklungsprozess eingesetzt zu werden.
Der GitHub Copilot viel besonders positiv auf im Aspekt der Codequalität. Er nahm bei allen drei Tests, mit denen die Qualität evaluiert wurde, Bezug auf den Gesamtkontext des Projekts, und machte semantisch und syntaktische Codevorschläge. Lediglich beim Auslesen einer CSV-Datei war der Codevorschlag nicht korrekt zur Laufzeit. GitHub Copilot ist in VS Code und PyCharm integrierbar. Der in den Prompt eingegebene Code wird nicht zum Training zukünftiger Generationen des Sprachmodells benutzt, und GitHub erhebt keinen urheberrechtlichen Anspruch auf den generierten Code. Es ist jedoch negativ anzumerken, dass unabhängige Datenschutzzertifizierungen noch fehlen.
Die größte Stärke des Tools Codeium ist seine flexible Einsetzbarkeit in alle bei der T. CON genutzten IDEs. Das sind VS Code, Eclipse, PyCharm und Jupyter Notebook. Die Codequalität konnte nicht an die des GitHub Copiloten heranreichen. Beim Testen der Qualität fiel er durch semantische Fehler negativ auf. So wollte er eine Variable mit dem Namen my_numer, was auf einen Integer oder Doublewert schließen lässt mit einem Booleanwert initialisieren. Auch dieses Tool erzeugte einen Laufzeitfehler beim Auslesen der CSV-Datei. Auch bei Codeium wird der in den Prompt eingegebene Code nicht zum Training des Modells verwendet. Im Gegensatz zum GitHub Copilot wurde Codeium SOC 2-Zertifiziert.

7.2 Güte der Lösung

Diese Projektarbeit hatte zum Ziel, drei Codevervollständigungstools anhand definierter Kriterien zu evaluieren. Folgende quantitative Kennzahlen bewerten die Lösung:
Der GitHub Copilot kostet 19 US-Dollar im Monat pro Benutzer. Codeium kostet 12 US-Dollar im Monat pro Benutzer.
Im Zuge der Qualitätstests wurden insgesamt 14 Erwartungen an die Tools definiert. Diese Erwartungen bezogen sich auf syntaktisch und zur Laufzeit korrekte, sowie semantisch sinnvolle Codevorschläge. Vier Erwartungen bei der Analyse der Kontextsensitivität, sechs bei einem kleinen Pythonprojekt und vier bei einem Projekt im UI5-Umfeld. Der GitHub Copilot konnte insgesamt 13 (4+5+4) dieser Erwartungen erfüllen. Codeium konnte zehn (3+4+3) der 14 Erwartungen erfüllen.
Bei der Nutzwertanalyse wurde der GitHub Copilot nach Gewichtung ohne Nivellierungsverzerrung mit 397,5 aus 500 Punkten bewertet, das sind 79,5%. Codeium kam auf 435,5 aus 500 Punkten, also 87,1%. Die Güte der Lösung kann auch anhand nicht quantitativer Kennzahlen erhoben werden. In der Abschlussarbeit gibt es Einschränkungen hinsichtlich der erarbeiteten Lösungen, dennoch konnten alle Ziele umgesetzt werden.
Die Experteninterviews waren sehr zeitaufwendig, daher konnte nur ein kleiner Querschnitt der T. CON interviewt werden. Die interviewten Experten haben ohne Ausnahme fachlich wertvolle Antworten gegeben, welche für die Nutzwertanalyse eine gute Basis bildeten.
Die Nutzwertanalyse konnte wie in der Zielsetzung formuliert umgesetzt werden. Die theoretischen Aspekte der Nutzwertanalyse konnten gut abgearbeitet werden, jedoch war man häufig auf Angaben der Unternehmen angewiesen. Die Qualität der Tools wurde durch die Implementierung umfangreicher praktischer Usecases bewertet, welche allerdings prototypischer Natur waren, und stark abstrahiert werden mussten. Es konnte begründet werden, warum sich der GitHub Copilot und Codeium eignen, Tabnine dagegen nicht. Es war ursprünglich geplant, die Gewichtung in der Nutzwertanalyse anhand der Expertenbefragung auszurichten. Dies stellte sich allerdings als nicht sinnvoll heraus, da es zur Nivellierungsverzerrung kam. Wichtige Muss-Aspekte wie Datenschutz und Codequalität waren dadurch nur unwesentlich höher gewichtet als Aspekte wie Community, welche eine Kann-Anforderung darstellen.

7.3 Handlungsempfehlung

Es wird ein offener Umgang der Firma mit den Tools Codedium und GitHub Copilot empfohlen. Der Copilot hat in den Analysen tendenziell eine höhere Codequalität als Codeium, jedoch entwickeln sich die Sprachmodelle beider Tools kontinuierlich weiter. Insbesondere die Entwickler, welche Interesse an den beiden Tools zeigen, sollten diese benutzen dürfen. So kann weitere Erfahrung gesammelt werden. Außerdem sollten, sobald die SAP eigene Tools veröffentlicht, auch diese erforscht werden.

7.4 Zukunftsausblick

Laut Gartner befindet sich generative KI derzeit auf dem Gipfel überhöhter Erwartungen im Hype Cycle for Emerging Technologies. Generative KI wird als Teil eines breiten Trends betrachtet, der neue Möglichkeiten für Innovationen eröffnet. Es wird prognostiziert, dass diese Technologien innerhalb der nächsten zwei bis fünf Jahren bedeutenden Nutzen bringen werden. Jedoch warnt das Institut alle Führungskräfte, dass ihre Aufmerksamkeit auch auf andere aufkommende Technologien wie Clouddienste, sowie Sicherheit und Datenschutz richten müssen. Technologien zu benutzen, welche sich in einem frühen Stadium des Hype Cycles befinden, bergen höhere Risiken, aber auch potenziell größere Vorteile [40].
Die SAP hat auf der TechEd 2023 viel angekündigt, jedoch bis dato wenig veröffentlicht. Jeder SAP-Entwickler soll laut dem Konzern in naher Zukunft mit generativer KI arbeiten. Die SAP stellte ein generatives KI-Modell vor, welches in SAP Build Code integriert ist, und stark dem Interface des GitHub Copilot ähnelt [41]. Außerdem soll die HANA Cloud an Vektordatenbanken angepasst werden, welche die Interaktion von LLMs und geschäftskritischen Daten verbessern soll. Außerdem baut die SAP Lernangebote aus, welche Einfluss auf die rollenbasierte Zertifizierung hat, was für den Partnerstatus der T. CON relevant ist [2]. Die SAP verspricht insgesamt in den nächsten zwei Jahren eine „dreistellige Anzahl von Neuerungen“ auf den Markt zu bringen. [42]

8 LITERATUR

[1] Holger Schmidt. KI macht Softwareentwickler 30 bis 50 Prozent produktiver, 2024. zuletzt besichtigt: 15. März 2024.
[2] Bernhard Luck. SAP macht jeden Entwickler zum Entwickler für generative KI, 2023. zuletzt besichtigt: 06. Januar 2024.
[3] IDC. IDC: Die ICT-Ausgaben in der DACH-Region werden in diesem Jahr 275 Milliarden US-Dollar erreichen, der Investitionsschwerpunkt liegt auf KI, 2023. zuletzt besichtigt: 06. Januar 2024.
[4] Hilker. Generative KI am Zenit: Gartners Hype-Zyklus 2023, 2023. zuletzt besichtigt: 06. Januar 2024.
[5] Andreas Streim. KI gilt in der deutschen Wirtschaft als Zukunftstechnologie – wird aber selten genutzt, 2023. zuletzt besichtigt: 06. Januar 2024.
[6] Tcon. Mehrwert durch Softwareerfahrung, Beratung und Service, 2023. zuletzt besichtigt: 15. Juni 2022.
[7] Robert Kaiser. Qualitative Experteninterviews: Konzeptionelle Grundlagen und praktische Durchführung. Springer Fachmedien Wiesbaden, 2014.
[8] Jörg B. Kühnapfel. Scoring und Nutzwertanalysen: Ein Leitfaden für die Praxis. Springer Fachmedien Wiesbaden, 2021.
[9] Johannes Behrndt. SAPUI5 OpenUI5, 2022. zuletzt besichtigt: 06. Januar 2024.
[10] Ingo Biermann. SAPUI5, 2021. zuletzt besichtigt: 06. Januar 2024.

[11] Jann Raveling. Was ist künstliche Intelligenz?, 2023. zuletzt besichtigt: 30. Januar 2024.
[12] Alexander Kästner, Maren Bührig, Janina Holm, Dominik Klee, Michael Löbbert, Marcel Scherbinek, and Vincent Schmid. SAP Data Intelligence. December 2020.
[13] Thimira Amaratunga. NLP Through the Ages, page 9–54. Apress, 2023.
[14] Stefan Luber. Was ist ein Large Language Model (LLM)?, 2023. zuletzt besichtigt: 30. Januar 2024.
[15] Data Guard. SOC 2 oder ISO 27001 Zertifizierung: Vergleich der InfoSec-Standards, 2023. zuletzt besichtigt: 06. Januar 2024.
[16] Andrew Williams. Was bedeutet SOC 2-Konformität?, 2022. zuletzt besichtigt: 06. Januar 2024.
[17] Sophie Suske, Sören Siebert. DSGVO: Was sollten Webseitenbetreiber und Unternehmer über die Datenschutz-Grundverordnung wissen?, 2021. zuletzt besichtigt: 06. Januar 2024.
[18] Craig Smith. Hallucinations could blunt ChatGPT's success, 2023. zuletzt besichtigt: 06. Januar 2024.
[19] Christian Meier. Warum die KI so gerne lügt, 2023. zuletzt besichtigt: 06. Januar 2024.
[20] Gerrit De Vynck. ChatGPT 'hallucinates': some researchers worry it isn't fixable., 2023. zuletzt besichtigt: 06. Januar 2024.
[21] Erich Moechel. Künftige KI-Modelle potenziell von Demenz bedroht, 2023. zuletzt besichtigt: 06. Januar 2024.
[22] Felix Holtermann. Wird ChatGPT dümmer? Das sagt eine Stanford-Studie, 2023. zuletzt besichtigt: 06. Januar 2024.
[23] Peter Zellinger. ChatGPT wird immer dümmer, doch niemand weiß, warum, 2023. zuletzt besichtigt: 06. Januar 2024.
[24] Holger Schmidt, Peter Buxmann. Matthias Orthwein: "Urheberrecht für KI-Inhalte wird ein Problem für die Softwareindustrie", 2024. zuletzt besichtigt: 06. Januar 2024.
[25] Charlotte Voß. New York Times klagt gegen Microsoft und OpenAI, 2023. zuletzt besichtigt: 06. Januar 2024.
[26] Julia Bald. Mithilfe künstlicher Intelligenz plötzlich Urheber?, 2023. zuletzt besichtigt: 06. Januar 2024.
[27] Assecor. GitHub Copilot, 2023. zuletzt besichtigt: 06. Januar 2024.
[28] IONOS. GitHub Copilot: Der Programmierassistent im Überblick, 2023. zuletzt besichtigt: 06. Januar 2024.
[29] Gedeon Rauch. Wie funktioniert GitHub Copilot?, 2023. zuletzt besichtigt: 06. Januar 2024.
[30] Lars Becker. Was ist Tabnine? Alle Infos über den Code-Assistenten, 2023. zuletzt besichtigt: 06. Januar 2024.
[31] Gedeon Rauch. Wie funktioniert Tabnine?, 2023. zuletzt besichtigt: 06. Januar 2024.
[32] Finn Hillebrandt. Codeium, 2023. zuletzt besichtigt: 06. Januar 2024.
[33] Philipp Steubel. Die Nutzwertanalyse: Definition, Anwendung und Beispiele!, 2023. zuletzt besichtigt: 06. Januar 2024.
[34] GitHub. Github Copilot Trust Center, 2024. zuletzt besichtigt: 30. Januar 2024.
[35] Tabnine. Trust Center, 2024. zuletzt besichtigt: 30. Januar 2024.
[36] Codeium. Codeium is SOC2 compliant, 2023. zuletzt besichtigt: 30. Januar 2024.
[37] Wojtek Fulmyk. One-Hot Encoding — A Brief Explanation, 2023. zuletzt besichtigt: 15. März 2024.
[38] Sajid Lhessani. What is the difference between training and test dataset?, 2022. zuletzt besichtigt: 15. März 2024.
[39] John Resig. ECMAScript 5 Strict Mode, JSON, and More, 2009. zuletzt besichtigt: 15. März 2024.
[40] Gartner. Gartner places generative ai on the peak of inflated expectations on the 2023 hype cycle for emerging technologies, 2023. zuletzt besichtigt: 06. Januar 2024.
[41] JG Chirapurath. Supercharging developer productivity with sap build code, 2023. zuletzt besichtigt: 06. Januar 2024.
[42] Christof Kerkmann. Warum SAP bisher mehr ankündigt als liefert, 2024. zuletzt besichtigt: 06. Januar 2024.