

Thema:

Praktische Anwendung von Large Language Models im Bereich Software Engineering am Beispiel von ChatGPT

Ausarbeitung

im Rahmen des Moduls E541 Softwareprojekt

Themensteller/: Prof. Dr. Wolfgang Albrecht

Betreuer

Vorgelegt von: Jan Walldorf: 538992

Projektzeitraum: 01.10.2023 - 31.03.2023

Inhaltsverzeichnis

[1 Einleitung 1](#_Toc161574757)

[2 Aufgabenstellung und Ziele 1](#_Toc161574758)

[3 Technologien 1](#_Toc161574759)

[3.1 künstlicher Intelligenz in der Praxis 1](#_Toc161574760)

[3.1.1 ChatGPT von OpenAI 1](#_Toc161574761)

[3.1.2 Github Copilot 1](#_Toc161574762)

[3.1.3 Alternative Produkte wie Amazon Codewhisperer und Tabnine 1](#_Toc161574763)

[3.1.4 Vergleich ChatGPT und Github Copilot 1](#_Toc161574764)

[3.2 Dokumentationstool 2](#_Toc161574765)

[3.2.1 Jupyter Notebook 2](#_Toc161574766)

[3.2.2 Google Colabs 2](#_Toc161574767)

[3.2.3 Anaconda 2](#_Toc161574768)

[3.2.4 Visual Studio Code 2](#_Toc161574769)

[3.2.5 Vergleich zwischen Google Colabs, Anaconda und VS Code für Jupyter Notebooks 2](#_Toc161574770)

[4 Artefakte 2](#_Toc161574771)

[4.1 Code Refactoring mit KI 2](#_Toc161574772)

[4.2 Codeanalyse und Bewertung durch KI 3](#_Toc161574773)

[4.3 Tipps und Tricks für Prompt Engineering 3](#_Toc161574774)

[4.4 Anleitungen zu Github Copilot mit Erfahrungswerten 3](#_Toc161574775)

[5 Fazit 3](#_Toc161574776)

[6 Anhang 3](#_Toc161574777)

[6.1 Tipps\_und\_Tricks\_fuer\_ChatGPT.ipynb 3](#_Toc161574778)

[6.2 code\_Refactoring\_und\_Optimierung\_mit\_KI.ipynb 3](#_Toc161574779)

[6.3 Bewertung\_und\_konstruktive\_Kritik\_von\_Code\_durch\_KI.ipynb 3](#_Toc161574780)

[6.4 Anleitung Github Copilot Registrierung fuer Studenten.txt 3](#_Toc161574781)

[6.5 Anleitung Github Copilot Integration in QT.txt 3](#_Toc161574782)

[6.6 Anleitung Github Copilot Integration in VS Code.txt 3](#_Toc161574783)

[7 Literaturverzeichnis 4](#_Toc161574784)

# Einleitung

Die folgende Projektarbeit befasst sich mit der Anwendung künstlicher Intelligenz für die Softwareentwicklung am Beispiel von ChatGPT und Github Copilot.

Dabei wird ein Fokus daraufgelegt, wie man im Studium den Studierenden dieses Thema näherbringen und damit einen Mehrwert in der Ausbildung schaffen kann.

# Aufgabenstellung und Ziele

Aufgabenstellung:

* Technologieüberblick über den Einsatz von KI in der Softwareentwicklung
* Sammlung von Erfahrung im Umgang mit KI
* Testen verschiedener Tools und Evaluierung

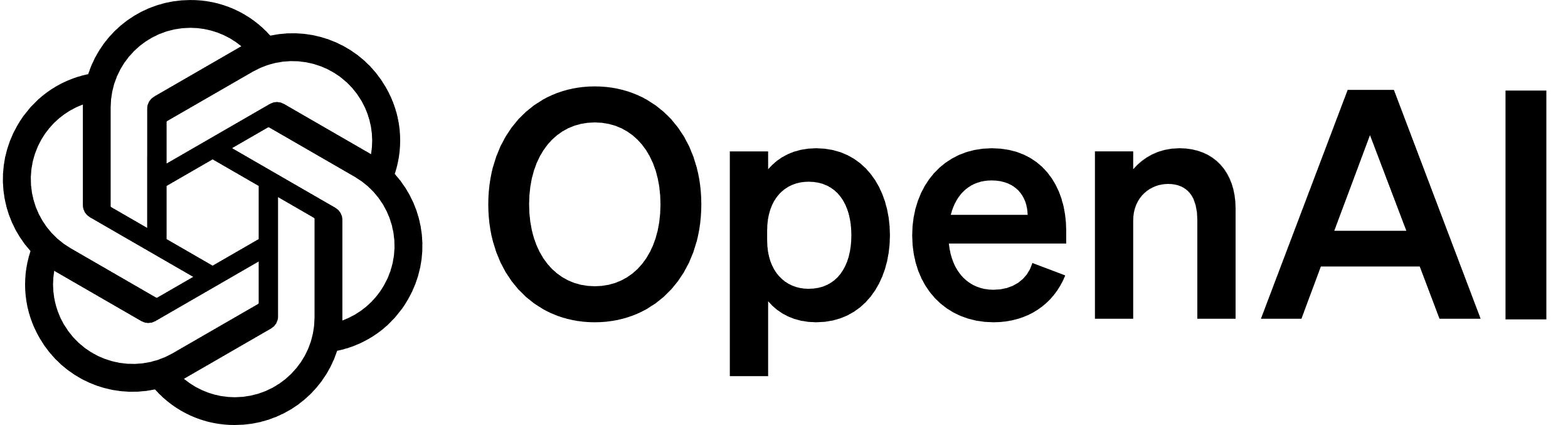
Ziele:

* Empfehlung für den Einsatz von KI im Studium
* Erstellung von praktischen Anwendungsbeispielen anhand von Praktikumsaufgaben für das Modul C++ Programmierung

# Technologien

## künstlicher Intelligenz in der Praxis

### ChatGPT von OpenAI



(OpenAI, 2024)

### Github Copilot

GitHub Copilot ist ein neuartiges Tool, das Entwicklern dabei hilft, Code zu schreiben. Es basiert auf der Technologie von OpenAI's GPT-3. GitHub Copilot analysiert den Kontext des Codes mithilfe von maschinellem Lernen und schlägt automatisch Vervollständigungen sowie ganze Codezeilen vor. Mit seiner Fähigkeit, verschiedene Programmiersprachen und Bibliotheken zu erkennen, ist es in der Lage, entsprechende Codeausschnitte zu generieren. Diese Funktionalitäten tragen dazu bei, die Produktivität von Entwicklern zu steigern und den Entwicklungsprozess zu beschleunigen, indem sie repetitive Aufgaben automatisieren und nützliche Codefragmente bereitstellen.



(Copilot, 2024)

### Alternative Produkte wie Amazon Codewhisperer und Tabnine

(Chornaya, 2024)

### Vergleich ChatGPT und Github Copilot

Github Copilot wurde dabei als Alternative zur Webversion von ChatGPT eingesetzt und lieferte dabei subjektiv betrachtet bessere Ergebnisse. Dies hat vor allem den Hintergrund, dass die KI speziell mit Codefragmenten trainiert wurde und daher für die Aufgabenstellung im Projekt – die Unterstützung bei der Erstellung von Programmcode – gut geeignet ist. Im Folgenden werden Vorteile von Github Copilot gegenüber dem klassischen ChatGPT aufgezählt:

1. Ausgelegt auf Codegenerierung
2. Integrationen in Entwicklungsumgebungen wie Visual Studio Code oder QT
3. Anpassung auf verschiedene Programmiersprachen und Bibliotheken

## Dokumentationstool

### Jupyter Notebook

Jupyter Notebooks sind interaktive Dokumente, die Code, Text und visuelle Elemente kombinieren und es Benutzern ermöglichen, Programmcode in Abschnitten auszuführen. Ihre Vorteile umfassen eine intuitive Benutzeroberfläche, die Zusammenarbeit erleichtert, und die Möglichkeit, Ergebnisse in Echtzeit zu visualisieren. Ein Nachteil ist unter anderem, dass als Standardprogrammiersprache Python festgelegt ist und andere Programmiersprachen nur in bestimmten Laufzeitumgebungen funktionieren. Jupyter Notebooks können mit verschiedenen Programmen erstellt werden, darunter VS Code (lokal), Google Colab (Cloud) und Anaconda (lokal und Cloud). Im Projekt wurden Jupyter Notebooks als Dokumentationstool verwendet, da das Feature Programmcode direkt auszuführen besonders für Studierende im Lernprozess vorteilhaft ist.

### Google Colabs

Google Colab ist eine Online-Plattform für interaktives Schreiben von Code, die es Benutzern ermöglicht, Jupyter Notebooks in der Cloud auszuführen. Die reine Zusätzlich bietet Google Colab die Möglichkeit, C++-Code in den Notebooks auszuführen, was die Flexibilität für Entwickler erhöht.

### Anaconda

Anaconda bietet eine umfassende Data-Science-Plattform, die eine Vielzahl von Tools und Bibliotheken für die Analyse und Visualisierung von Daten bereitstellt. Innerhalb der Anaconda-Umgebung sind Jupyter Notebooks eine zentrale Komponente wobei Anaconda sowohl lokal auf dem eigenen Computer wie auch in der Cloud betrieben werden kann.

### Visual Studio Code

### Vergleich Jupyter Notebook Programme

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Programm | Cloud | Lokale Ausführung | C++ Code ausführbar | ChatGPT Integration |
| Colab | Ja | Nein | Ja | Ja (mit Chrome Plugin) |
| Anaconda | Ja | Ja | Nein | Ja – nativ |
| VS Code | - | Nein | Nein | Ja – nativ über Plugin Store |

Grundsätzlich unterscheiden sich die einzelnen Programme nur in Details und für allgemeine Anwendungen ist es irrelevant, für welches Tool mach sich entscheidet.

# Use Cases

In der Softwareentwicklung gibt es eine Vielzahl von Anwendungsfällen für KI (künstliche Intelligenz), die die Effizienz verbessern und innovative Lösungen bieten können. Einige wichtige Use Cases sind:

* Code-Generierung und Autocompletion: KI-gestützte Tools können Entwicklern helfen, schneller und präziser Code zu schreiben, indem sie automatisch Codevorschläge generieren und beim Schreiben von Code unterstützen.
* Fehlererkennung und Debugging: KI-Algorithmen können eingesetzt werden, um Fehler im Code zu erkennen und zu beheben, indem sie Muster in den Daten analysieren und potenzielle Fehlerquellen identifizieren.
* Automatisierte Tests: KI kann verwendet werden, um automatisierte Tests zu erstellen und auszuführen, um die Qualität und Stabilität von Softwareprodukten zu verbessern.
* Natürliche Sprachverarbeitung (NLP): NLP-Algorithmen können in Entwicklertools integriert werden, um natürliche Sprache zu verstehen und Entwicklern bei der Kommunikation und Dokumentation zu helfen.
* Predictive Analytics: Durch die Analyse von Daten können KI-Modelle verwendet werden, um Vorhersagen über zukünftige Trends und Ereignisse in der Softwareentwicklung zu treffen, z. B. zur Schätzung von Entwicklungsaufwänden oder zur Identifizierung von potenziellen Engpässen im Projektmanagement.
* Automatische Code-Optimierung: KI-Algorithmen können verwendet werden, um vorhandenen Code zu analysieren und zu optimieren, um die Leistung zu verbessern, Speicherplatz zu sparen und die Wartbarkeit zu erhöhen.
* Sicherheit und Datenschutz: KI kann eingesetzt werden, um Sicherheitslücken und Datenschutzverletzungen in Softwareanwendungen zu erkennen und zu verhindern, indem sie verdächtiges Verhalten und Anomalien in den Daten identifiziert.

Diese Use Cases zeigen, wie KI-Technologien die Softwareentwicklung auf verschiedene Weise unterstützen können, indem sie Prozesse automatisieren, die Produktivität steigern und die Qualität von Softwareprodukten verbessern.

Im vorliegenden Projekt wurde sich auf die Punkte Code Optimierung und Code Analyse beschränkt.

# Artefakte

## Code Refactoring mit KI

Dieser Abschnitt behandelt das Code Refactoring mit künstlicher Intelligenz, dass im Jupyter Notebook „code\_Refactoring\_und\_Optimierung\_mit\_KI.ipynb“. Refactoring, der Prozess der Umstrukturierung von Code ohne Änderung seines Verhaltens, wird als Methode zur Verbesserung der nichtfunktionalen Attribute der Software erläutert. KI kann beim Refactoring helfen, indem sie den Code analysiert, dessen Funktionalität versteht und Verbesserungsvorschläge macht, wie die Vereinfachung von komplexem Code und die Optimierung der Code-Struktur. Das Ziel ist es, Beispiele für die praktische Anwendung des Refactorings mit KI zu geben. Dabei wird ein Beispielprogramm für einen Multiple-Choice-Test in C++ schrittweise refaktoriert und optimiert, wobei KI-basierte Anpassungen und Verbesserungen durchgeführt werden. Dabei werden auch wichtige Punkte wie die Strukturierung des Codes, die Verbesserung der Lesbarkeit und die Anpassung an bewährte Programmierstandards berücksichtigt. Abschließend wird diskutiert, wie KI als leistungsfähiges Werkzeug zur Verbesserung der Codequalität eingesetzt werden kann, jedoch stets mit der Notwendigkeit, den generierten Code zu testen und sicherzustellen, dass er wie erwartet funktioniert.

## Codeanalyse und Bewertung durch KI

## Tipps und Tricks für Prompt Engineering

Zuletzt wurden die aus dem Projekt gewonnen Erfahrungen zu guten und schlechten Prompts zur Kommunikation mit künstlicher Intelligenz in einem Dokument zusammengefasst und mit Beispielen erläutert. Dabei wird zunächst darauf eingegangen, welche allgemeinen Tipps es zu beachten gibt. Dazu zählen unter anderem Aspekte wie Ausdrucksweise, präzise Formulierungen oder auch Hinweise auf die zu verwendende Sprache. Anschließend wird auf verschiedene Integrationen von ChatGPT eingegangen, wo deren Vorteile liegen und was für Möglichkeiten dort bestehen. Im Hauptteil werden danach Prompts anhand von 5 Beispielen erläutert. Die Beispiele sind dabei immer so aufgebaut, dass zunächst ein schlechter Prompt gezeigt wird und danach eine verbesserte Version davon dargestellt ist. Die Antwort der KI liefert dabei stets Programmcode in C++, der direkt im Jupyter Notebook ausgeführt werden kann. Die Beispiele sind recht simpel gehalten, verdeutlichen jedoch gut wie groß der Einfluss der Wortwahl des Prompts auf das Ergebnis ist. Abgeschlossen wird das Jupyter Notebook von einer Zusammenstellung von guten Command Prompts in Form eines „Cheet Sheets“.

## Anleitungen zu Github Copilot mit Erfahrungswerten

# Fazit

# Anhang

## Tipps\_und\_Tricks\_fuer\_ChatGPT.ipynb

Siehe Datei

## code\_Refactoring\_und\_Optimierung\_mit\_KI.ipynb

Siehe Datei

## Bewertung\_und\_konstruktive\_Kritik\_von\_Code\_durch\_KI.ipynb

Siehe Datei

## Anleitung Github Copilot Registrierung fuer Studenten.txt

Siehe Datei

## Anleitung Github Copilot Integration in QT.txt

Siehe Datei

## Anleitung Github Copilot Integration in VS Code.txt

Siehe Datei

# Literaturverzeichnis

Chornaya, J. (09. 02 2024). *hubspot.de*. Von hubspot.de: https://blog.hubspot.de/website/ki-code abgerufen