**Wikidata (Ontologie)**

Wikidata verfügt keine spezielle formale Syntax für Klassen. Stattdessen verwendet die Community Items und Properties wie Klasse, Entität und Unterklasse, um die Ontologie zu strukturieren. Zusätzlich werden Items genutzt, die die Ordnung von Klassen ausdrücken: *Klasse erster Ordnung*, *Metaklasse zweiter, dritter, vierter Ordnung*. Die Klasse erster Ordnung ist die Metaklasse aller Klassen zweiter Ordnung. Sie liefern wertvolle Information über die Art der Instanzen einer Klasse. Dabei unterscheiden sich Klassen in Wikidata nicht grundsätzlich von anderen Elementen. Ihre Rolle als Klassen wird nur durch Beziehung erkennbar, die ausschließlich für Klassen vorgesehen sind (P31, P279) (vgl. [Wikidata:Ontologie](https://www.wikidata.org/wiki/Wikidata:WikiProject_Ontology/Modelling)).

Das RDF-Format ist eng mit dem Wikidata-Datenmodell verknüpft und hat RDF-Struktur: Subjekt – Prädikat – Objekt. Einige Standardeigenschaften in Wikidata sind dabei direkt den Prädikaten von RDF bzws. RDFS zugeordnet. So wird bspw. ein Wikidata-Label in einer bestimmten Sprache im Dump zu rdfs:label, skos:prefLabel und schema:name. Ein Alias (alternatives Label) wird als skos:altLabel, während Beschreibungen in schema:description-Tripeln überfürt werden (vgl. [RDF und Wikidata](https://www.wikidata.org/wiki/Wikidata:Relation_between_properties_in_RDF_and_in_Wikidata)).

**Prompt-Engine**

Der Prompt ist eine zentrale Schnittstelle zwischen Menschen und Maschine. Er muss nicht auf eine Aufgabezeile reduziert werden, sondern kann mehrseitige Texte, Bilder oder gesprochene Eingaben enthalten. Schlüsselwörter ermöglichen semantische Bedeutung und Zusammenhänge im Text zu erkennen. So kann eine konsistente und präzise Antwort erzielt werden (vgl. Bünnagel 2024: 177).

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abb. 5.8 Ein effektiver Prompt ([Caelen/Blete 2004: Chapter 4](https://learning.oreilly.com/library/view/developing-apps-with/9781098168094/ch04.html))

Prompts dienen zu der Wahrscheinlichkeit bestimmter Antworten zu steuern. Sie fungieren als Leitfaden, der das Modell zu einer bestimmten Antworttype lenkt. Obwohl es keine standardisierte Struktur für Prompts existiert, bietet die Kombination aus Rolle, Kontext und Aufgabe ein hilfreiches Orientierungsschema (Abbildung 5.8). Zuweisung einer Rolle ist eine Möglichkeit, ein LLM gezielt zu steuern. Der Kontext bildet dabei das wichtigste Element eines Prompts. Bei der Formulierung eines Eingabetexts für LLM sollte der Kontext möglich präzise beschrieben werden. Dies erhöht die Chance, eine relevante Antwort zu bekommen. Die Erstellung eines Prompts ist ein iterativer Prozess, der oft mehrere Versuche erfordert. Zudem muss die Aufgabe klar und spezifisch definiert werden, wobei die relevanten Informationen klar und eindeutig bereitgestellt werden müssen (vgl. [Caelen/Blete 2004: Chapter 4](https://learning.oreilly.com/library/view/developing-apps-with/9781098168094/ch04.html)). Aufgrund dessen wurden im Rahmen dieser Arbeit verschiedene Prompts erprob, um die Effektivität der Ergebnisse zu erhöhen. So wurden bspw. die unterschiedlichen Beispiele im Kontextteil definiert und mehrere Aufgaben integriert (u.a. RE und die Normalisierung von Datumsangaben). Diese Vorgehensweise führte jedoch zu einer deutlichen Belastung des Modells, wodurch sich die Bearbeitungszeit erheblich verlängerte und sie zwischen 8 und 11 Stunden lag. Darüber hinaus wurde bei der Steuerung des Grads an Zufälligkeit in den Modellantworten sog. Temperatur experimentiert und am besten passender Wert von 0,7 rausgesucht. Die Werte dieses Parameters liegen zwischen 0 und 1: Je höher der Wert ist (1,0), desto abwechslungsreicher und kreativer sind die Ergebnisse (vgl. [LangChain](https://python.langchain.com/docs/concepts/chat_models/#standard-parameters)).

**Evaluieren!!!**

Nach der RE wurden die Tripel für jeden Satz überprüft und ggf. manuell korrigiert. Da Wikidata als Zielrepositorium dienen sollte, war es erforderlich, bestimmte Relationen in umgekehrter Form zu modellieren. Dies betraf insbesondere asymmetrische Relationen wie die Beziehungen Vater/Mutter – Kind, Ehrpartner:innen. Bei der manuellen Korrektur traten einige Probleme auf. Während der Datenvorverarbeiten wurden die Texte in einzelne Sätze untergeteilt, wodurch das Problem mit der Kohärenz entstanden. Ein Hauptgrund dafür war, dass die Entity Linking nur später durchgeführt wurde und nicht parallel mit NER. Dies wurde als Fehler im Nachhinein herausgestellt: Während die Verwendung des Personalpronomens *ich* unproblematisch war, da es eindeutig zu der Autorin oder dem Autor zugeordnet werden konnte, führte ein Wechsel der Erzählperspektive von Ich- zu Er-/Sie-Form zu Mehrdeutigkeit. Auch beim Gebrauch der ersten Person Plural (*wir*) musste jeweils geprüft werden, auf wen sich die Referenz bezog, wenn dieser nicht aus dem Kontext erkennbar war. Darüber hinaus wurden die vorgegebenen Relationen bei der RE ignoriert. Stattdessen erfolgte die Extraktion entweder aus dem Text direkt oder durch Paraphrasierung. Dies führte zu den unterschiedlichen Formulierungen für dieselbe Relation. Ein weiteres Problem war nur eine erkannte oder fälschlicherweise duplizierte Relation wie *X und Y sind Kinder von Z*. Dieses Problem musste auch manuell behoben werden.

**FP (Tätigkeit) wurde übernommen**

## 5.4.3 Entity Linking

Entity Linking (EL) auch bekannt als Named Entity Disambiguation (NED) oder Entity Disambiguation etablierte sich im Forschungsfeld der Information Retrieval, natürlicher Sprache, semantisches Web und von den Datenbanken (vgl. Mulang et al. 2020: 1–2). Wissensgraphen wie Wikidata speichern strukturierte Informationen über Personen, Orte oder Konzepte. Jede Entität besitzt definierte Eigenschaften und Relationen. Diese Daten erleichtern semantische Anwendungen, z.B. Suchmaschine, die nach einer bestimmten Person suchen können. Das Hauptproblem dabei ist die Mehrdeutigkeit von Entitäten (vgl. Sevgili et al. 2022: 528). Die Erkennung dieser Entitäten ist nicht trivial. Gleiche Entitäten können vielen Name haben (Synonyme, Abkürzungen, Sprachvarianten) und umgekehrt sein, ein Name steht für mehrere Entitäten (z. B. Neustadt) (vgl. Ege/Paschke 2021: 103). Durch EL wird diese Mehrdeutigkeit aufgelöst (vgl. Sevgili et al. 2022: 528). Grundsätzlich umfasst EL zwei Aufgaben: die Erkennung von Entitäten im Text und die Disambiguierung. Bei der Disambiguierung werden Entitäten mit Strukturen und Wissensbasen wie Wikidata verknüpft werden (vgl. Mulang et al. 2020: 1–2).

Da die Maschinen keine indirekten Bedeutungen erkennen können, müssen Relationen zwischen Informationen eindeutig sein. Normdaten wie [GND-Nummer](https://www.dnb.de/DE/Professionell/Standardisierung/GND/gnd_node.html) (Gemeinsame Normdaten) ermöglichen diese Eindeutigkeit. Wird eine Person über GND-Nummer identifiziert, können Inhalte automatisch mit bspw. [Wikipedia](https://www.wikipedia.de/)-Eintrag oder [Deutschen Biographie](https://www.deutsche-biographie.de/) verknüpft werden (vgl. Kamzelak 2018: 426-427). Normdatensätze sind meistens über persistente Identifikatoren referenzierbar. Im Verknüpfungsprozess wird zunächst durch Disambiguierung der passende Normdatensatz aus den wahrscheinlisten Kandidaten bestimmt. Danach erfolgt die Verknüpfung der Entitätszeichenkette (vgl. Menzel et al. Franke-Maier et al. 2021: 231).

Für diese Arbeit wurde Wikidata als Referenzsystem gewählt. Wikidata erweist sich besonders gut, da es eine etablierte und frei zugängliche Wissensbasis ist, die über APIs abgefragt und direkt in Datenverarbeitungsprozesse integriert werden kann. Darüber hinaus war für das Projekt entscheidend, dass bereits zahlreiche Personen der Herrnhuter Brüdergemeine durch die Vorarbeiten der SLUB und anderer Einrichtungen in Wikidata erfasst wurden. Diese Daten wurden im Projekt *Moravian Knowledge Network Research* zusammengeführt. Auf diese Weise konnten diese bestehenden Daten zurückgegriffen und durch weitere Informationen ergänzt und angereichert werden.

Die Disambiguierung wurde mithilfe der Wikidata-API umgesetzt. Über API können diese IDs gezielt abgefragt werden. Dies erleichtert die Integration von Wikidata in semantische Anwendungen und Datenverarbeitungsprozesse (vgl. [Wikidata-API](https://www.mediawiki.org/wiki/API:Action_API)). Dafür wurde die bereinigte JSON-Datei mit den Tripeln in eine Excel-Tabelle konvertiert. Zunächst wurden unterschiedliche Formulierungen derselben Relation auf ein einheitliches Wikidata-Prädikat abgebildet. Danach erfolgte mithilfe Wikidata-API die automatische Zuordnung der gewonnenen Subjekte, Prädikate und Objekte zu den entsprechenden Q- und P-IDs. Informationen, die nicht identifiziert wurden, blieben leer und wurden händig ergänzt. Dabei zeigten sich zwei Fälle: Entweder ließ sich der Name einer Entität aufgrund von Mehrdeutigkeit nicht eindeutig identifizieren, oder die betreffende Person war in Wikidata noch nicht vorhanden. Die Entitäten, die nicht zugeordnet werden konnten, wurden aus dem Hilfstabelle entfernt; falls erforderlich, wurden neue Entitäten in Wikidata angelegt.

Für die Erstellung eines neuen Items für eine Person wurde ein Mindestkriterium definiert: Jedes neue Item muss den vollständigen Namen der Person sowie mindestens eine zusätzliche Information enthalten, wie bspw. geographisches Attribut oder eine familiäre Relation. Dies dient zugleich der Vermeidung von Redundanzen und der Sicherung der Datenqualität. In der Praxis wird aus Effizienzgründen oder unter Zeitdruck nicht selten darauf verzichtet, bereits bestehende Datensätze auf ihre Richtigkeit und Vollständigkeit hin zu prüfen. Stattdessen wird ein neuer Eintrag erzeugt, was wiederum zu Redundanzen sowie zu einer Verringerung der Datenkonsistenz und -qualität im Gesamtsystem führen kann (vgl. Kamzelak 2018: 426-427). So wurden einige verdoppelte Q-IDs entdeckt, die in Zukunft noch korrigiert werden sollten (z.B. Q94755477, Q123463892,

Ein wesentlicher Vorteil des gewählten Ansatzes bestand darin, dass ein großer Teil der Zuordnungen automatisiert erfolgen konnte. Dennoch zeigte sich eine klare Grenze: nicht alle Geodaten ließen sich identifizieren: Entweder die genannten Ortsnamen im Laufe der Zeit ihre historischen Bezeichnungen geänderten und in den Texten nicht genügend Information gab, um die Orte in Wikidata sicher zu erkennen, oder war die Schreibweise in den Primärquellen abweichend, sodass die betreffenden Orte nicht eindeutig erkannt werden konnten. Ähnlich verhielt es sich bei Personennamen. Häufig wurden die Bezeichnungen *Bruder XY*, *Schwester XY* oder *Geschwister XY* genannt, ohne dass weitere Anhaltspunkte zur eindeutigen Identifizierung vorhanden waren. Bei solchen Fällen musste überprüft werden, welche zusätzlichen Informationen extrahiert wurde, um eine bestehende Q-ID korrekt zuzuordnen oder ggf. eine neue anzulegen. Insgesamt war eine manuelle Nachbearbeitung in diesem Ansatz unverzichtbar.

## Wikidata-Datenmodell

Wie bereits im Kapitel 4 gesagt wurde, ist Wikidata eine multilinguale Graph-Datenbank, in der sowohl der Inhalt als auch Struktur des Graphen bearbeitet werden können. Es wird einem streng semantischen Prinzip durch Propertiesgefolgt. Klassen und Instanzen können dabei frei erzeugt werden. Da die Daten in Wikidata als RDF-Tripeln gespeichert werden, lässt sich das Datenmodell von Wikidata auf einfacher Ebene gut mit RDF abbilden: mit Item (rdf:subject), Property (rdf:predicate) und Wert (rdf:object) (vgl. [RDF und Wikidata](https://www.wikidata.org/wiki/Wikidata:Relation_between_properties_in_RDF_and_in_Wikidata)). Im Rahmen dieser Arbeit wurde zwei Typen von Items verwendet: Person mit biographischer Datensätze und Weg zu Missionsorten.

Da sich die Informationen über die Personen nicht als homogen zeigten, wurde sämtliches verfügbares Material im Datenmodell aufgenommen. Die Tabelle 5.2 zeig die Modellierung aller denkbaren Relationen zu einer Person. Wikidata-Elementen wurden mithilfe der Wikidata-Eigenschaft P31 (Instanz) ein Typ Q5 zugewiesen. Es gibt die typische Information für eine Person wie Geschlecht, Verwandtschaften, Ehrpartner:in, die Ausbildung, wichtige Datenangaben. Ein ergänzendes Element des Datenmodell ist die Verwendung von Qualifizierern. Sie ermöglichen es, Relationen mit zusätzlichen Kontextinformationen anzureichern: mit Datumsangaben (pq:580, pq:582 und pq:585) oder mit Quellenangaben zu Primärdigitalisat (pq:584).

|  |  |  |
| --- | --- | --- |
| Wikidata-Property | Beschreibung | Beispiel |
| rdfs:label (@de) | *Label* des Items (Deutsch, Literal) | Johann Christian Breutel (wd:Q21415674) |
| wdt:P31 | *Instanz* von (verlinkt) | Mensch (wd:Q5) |
| wdt:P19 | *Geburtsort* (verlinkt) | Weißenburg in Bayern (wd:Q44500) |
| wdt:P20 | *Sterbeort* (verlinkt) | Berthelsdorf (wd:Q502553) |
| wdt:P21 | *Geschlecht* (verlinkt) | männlich (wd:Q6581097) |
| wdt:P22 | *Vater* (verlinkt) | Johann Christian Breutel (wd:Q21415674) |
| wdt:P25 | *Mutter* (verlinkt) | Sophie Breutel (wd:Q125843987) |
| wdt:P26 | *Ehrpartner(in)* (verlinkt) | Sophie Breutel (wd:Q125843987) |
| wdt:P40 | *Kind(er)* (verlinkt) | Johann Christian Breutel (wd:Q21415674) |
| wdt:P69 | *besuchte Bildungseinrichtung* | Universität Leipzig (wd:Q154804) |
| wdt:P106 | *Tätigkeit* (verlinkt) | Botaniker (wd:Q2374149) |
| wdt:P463 | *Mitglied von* (verlinkt) | Herrnhuter Brüdergemeine (wd:Q159318) |
| wdt:P551 | *Wohnsitz* (verlinkt) | Berthelsdorf (wd:Q502553) |
| wdt:P569 | *Geburtsdatum* (Literal), ISO: YYYY-MM-DD/ YYYY-MM/ YYYY | 1788-01-21 |
| wdt:P570 | *Sterbedatum* (Literal), ISO: YYYY-MM-DD/ YYYY-MM/ YYYY | 1875-02-18 |
| wdt:P793 | *Schlüsselereignis* (verlinkt) | Weg zum Missionsort (Südafrika) (wd:Q135874728)/ Konfirmation (wd:Q214802) |
| wdt:P937 | *Wirkungsort* (verlinkt) | Südafrika (wd:Q258) |
| wdt:P1066 | *Schüler von* (verlinkt) | Johann Jürgen Stein (wd:Q123456933) |
| Wdt:P1971 | *Anzahl der Kind(er),* (Literal) | 7 |
| wdt:P3373 | *Geschwister* (verlinkt) | Johann Christian Breutel (wd:Q21415674) |
| pq:P580 | *Startzeitpunkt (Literal), ISO: YYYY-MM-DD/YYYY-MM/YYYY* | 1822-06-08 |
| pq:P582 | *Endzeitpunkt (Literal), ISO: YYYY-MM-DD/YYYY-MM/YYYY* | 1822-06-08 |
| pq:P585 | *Zeitpunkt/Stand (Literal), ISO: YYYY-MM-DD/YYYY-MM/YYYY* | 1822-06-08 |
| pq:P854 | *URL der Fundstelle (URL)* | URL |

Der zweite Typ des Datenmodells umfasst die Wege zu Missionsorten. In den untersuchten Lebensläufen wurden neben biographischen Angaben auch die reisen von Missionaren beschrieben: sowohl die Anreise zu einer Missionsstelle als auch Reisen zwischen verschiedenen Missionsorten (Elim, Silo usw.). Letztere waren jedoch nicht konsistent dokumentiert und die Beschreibungen dieser Reisen waren häufig unterbrochen. Dies machte die chronologische Rekonstruktion nicht möglich. Daher wurde entschieden, sich ausschließlich auf die Wege bis zur jeweiligen Missionsstelle zu konzentrieren. Dabei wurden drei solche Routen identifiziert: Westindien, Labrador und Südafrika.

Es wurde drei Items im Wikidata hinterlegt: Weg zum Missionsort (Westindien) (wd:Q135902282), Weg zum Missionsort (Südafrika) (wd:Q135874728), Weg zum Missionsort (Labrador) (wd:Q135875239). Das Modell erlaubt einige Eigenschaften zu beschreiben (Tabelle 5.3). Über wdt:P31 (Instanz von) wurde die Klasse Route definiert. Zentrale Relationen bspw. wie relevante Person (wdt:P3342) und das angestrebte Ziel (wdt:3712) beschrienen die Personen, die diese Reise machten und Ziel dieser Reise. Besonders wichtig ist auch hier der Einsatz von Qualifizierern. Sie ermöglichen es, die Route jeder Person zu spezifizieren. Dadurch wurden das Ziel jeweiliger Reise, Zwischenstationen und ggf. Datumsangaben definiert. Darüber hinaus wurden die Quellennachweise (pq:854) verknüpft.

|  |  |  |
| --- | --- | --- |
| Wikidata-Property | Beschreibung | Beispiel |
| rdfs:label (@de) | *Label* des Items (Deutsch, Literal) | Weg zum Missonsort (Südafrika) (Q135874728) |
| wdt:P31 | *Instanz* von (verlinkt) | Route (Q1322323) |
| wdt:P3342 | *relevante Person* (verlinkt) | Johann Gottlieb Bonatz (1773-1827) (wd:Q123396429) |
| wdt:P3712 | *angestrebtes Ziel* (verlinkt) | Südafrika (wd:Q258) |
| pq:P580 | *Startzeitpunkt* (Literal), ISO: YYYY-MM-DD/YYYY-MM/YYYY | 1822-06-08 |
| pq:P582 | *Endzeitpunkt* (Literal), ISO: YYYY-MM-DD/YYYY-MM/YYYY | 1822-06-08 |
| pq:P585 | *Zeitpunkt/Stand* (Literal), ISO: YYYY-MM-DD/YYYY-MM/YYYY | 1822-06-08 |
| pq:P854 | *URL der Fundstelle* (URL) | URL |
| pq:1427 | *geografischer Startpunkt* (verlinkt) | Herrnhut (wd:Q165140) |
| pq:P1444 | *geografischer Endpunkt* (verlinkt) | Genadendal (wd:Q1501295) |
| pq:P2825 | *Zwischenpunkt einer Reise* (verlinkt) | Herrnhut (wd:Q165140) |
| pq:P3712 | *angestrebtes Ziel* (verlinkt) | Südafrika (wd:Q258) |

Die Verknüpfung zwischen Personen und Reisen erfolgt wechselseitig. Die jeweilige Route wurde als Schlüsselereignis (wdt:P793) in den Personendaten erfasst. Die Personen wurden in den Routendaten als relevante Person ausgewiesen. Die Modellierung der Missionswege wurde als besonders relevant erachtet, da sie ermöglichen, den Verlauf der Reise zu nachvollziehen und dabei die zentrale Umsteigpunkte zu identifizieren. Dadurch lassen sich die Knotenpunkte rekonstruieren, die für die Missionsarbeit eine wichtige Rolle spielen.

Eine große Herausforderung bei der Datenmodellierung bestand in der Zuordnung von Relationen zwischen Hilfstabelle und Wikidata. Besonders anspruchsvoll war die Entscheidung, welche Informationen als eigenständige Q-IDs und welche als Qualifizierer modelliert werden sollten. Beispielhaft betraf dies die Relationen *tätig\_in* (Ort), *tätig\_bei* (Organisation), *tätig\_als* (Beruf) und *tätig\_mit* (Person). Im Modell wurde festgelegt *tätig\_in* als Wirkungsort zu erfassen, während *tätig\_bei* und *tätig\_als* als Tätigkeit markiert wurden. Aufenthalte von Personen in Städten, die etwa dem Erlenen einer Fremdsprache oder dem langzeitlichen Besuch dienten, wurden als Wohnsitz modelliert. Als Wirkungsorte wurden alle Städte markiert, in denen die jeweilige Person länger- oder kurztätig war. Diese Beispiele verdeutlichen nur einen Teil der Herausforderungen bei der Datenmodellierung.

Mit dem Tool [QuickStatements](https://quickstatements.toolforge.org/) wurden die tabellarisch aufbereiten Daten in Wikidata hochgeladen. Die Tabelle bildete RDF-Tripel ab: Subjekt-ID, Property-ID, Objekt-ID und Qualifizierer.

## Semantic Web Technologie

<https://www.wikidata.org/wiki/EntitySchema:E49>

## SPARQL

Laube-Artikel

## Visualisierung

Netzwerkanalyse biete neue Möglichkeit zur Visualisierung und Analyse von Sammlungswissen. Ihre zentralen Ziele sind die Identifikation wichtiger Verbindungen und Strukturen. (vgl. Werner 2020: 248–249).

Für geistes- und sozialwissenschaftliche Forschung ist die Verbindung zwischen Netzwerken und Visualisierung von großer Bedeutung. Die graphische Visualisierung von abstrakten oder komplexen Strukturen macht die Zusammenhänge, die sich auf rein sprachlichem Wege nur schwer oder unzureichend vermitteln lassen, greifbar. Digitale Werkzeuge bieten hier neue Möglichkeiten, große Datenbestände nicht nur zu verarbeiten, sondern auch visuell aussagekräftige Formen überführen können. Daraus ergibt sich einen Anknüpfungspunk für die Verwendung netzwerkbasierter Modelle. Für die biographische Forschung eröffnet der Netzwerkansatz die Perspektive, individuelle Lebensläufe nicht isoliert zu betrachten, sondern als Teil größer Beziehungsgeflechte (vgl. Dörpinghaus 2022).

## Auswertung / Perspektive

**OCR**: Zur Verbesserung der Ergebnisse könnte mehrstufige Vorgehensweise: Mehrere OCR-Durchläufe mit variierenden Parametern von Schwellenwerten bspw. und ggf. deren spätere Konsolidierung. Auf diese Weise lassen sich Fehlertypen gegenseitig kompensieren. Darüber hinaus konnte auch adaptive Vorverarbeitung helfen. Man könnte die Bilder sauber/ verschmutz, hell/dunkel annotieren und pro Klasse geeignete Parameter wählen. Als Nachkorrektur kann ein LLM eingesetzt werden.

**XML**: mehr Attributen zu definieren, Versionierung zeigen, Personen und Körperschaft gleich mit Wikidata Normdaten kodieren, Datumsangaben zu normieren,

**Relation**: viele Information ist verloren gegangen, z.B. Zweck der Reise (Fremdsprache lernen, eine neue Kirche bauen oder einen passenden Ort für weitere Mission zu finden)

**Programmieren mit ChatGPT**:

Im Rahmen dieser Arbeit wurde das Programmieren mit der Unterstützung von ChatGTP eingesetzt. Insgesamt erwiest sich dieser Ansatz als gut geeignet. Besonders hilfreich war die Möglichkeit, den eigenen Code überprüfen zu lassen. Außerdem zeigte sich das Modell nützlich bei der Beschreibung der benötigten Datenformat.

Herausfordernd war hingegen die Arbeit mir komplexeren Problemen. ChatGPT lieferte selten sofort eine vollständige und funktionierende Lösung für größere Aufgaben. Problematisch war zudem, dass der generierte Code bei Nachfragen oder Fehlerbehebungen teilweise vollständig umgeschrieben wurde. Dabei kam es vor, dass alternative Python-Bibliotheken verwendet oder Variablennamen verändert wurden. Dies erschwerte die Nachvollziehbarkeit und Konsistenz des Codes. Dieses Problem wurde aber durch eine Anpassung des Prompts gelöst, indem bspw. bestimmte Bibliotheken festgelegt wurden.

Insgesamt zeigt die Bewertung, dass ChatGPT als nützliches Werkzeug zur Code-Erstellung und -Optimierung eingesetzt werden kann, gleichzeitig aber ein kritisches Nacharbeiten und eigenständiges Verständnis des Codes unverzichtbar bleibt.

## Zusammenfassung

## Literaturverzeichnis

## Anhang