## Methode

Mit KI hat es nicht geklappt, die WG gleich zu erstellen, deswegen zuerst NER im XML und dann wurde Relationsextraktion durchgeführt

Wissensgraphen können auf unterschiedliche Arte und Weise erstellt werden. In der Praxis sind die häufigste Variante: manuelle Erstellung, Erstellung aus (semi)strukturierten und unstrukturierten Quellen (vgl. Heist et al. Ege/Paschke 2021: 129). Außerdem gewährleisten neuronale Modelle des Textverstehens wie Large Language Models (LLM) ein großes Potential für die Wissenserfassung (vg. Hamann Hinkelmann/Hoppe/Humm 2025: 68).

Bei der manuellen Erstellung von Wissensgraphen werden die Fakten und Entitäten auf Grundlage von Erfahrung einer oder eines Experter:in definiert. Dies sichert die hohe Qualität des Wissens, aber dieser Ansatz ist zeitaufwendig und arbeitsintensiv (vg. Hamann Hinkelmann/Hoppe/Humm 2025: 74).

Eine Mischform aus manuellem Ansatz und KI-Modellen könnte ein optimaler Ansatz sein. Mithilfe von LLM wird das Korpus nach Entitäten und Fakten durchgesucht, während der manuelle Ansatz zu der Sicherstellung aller relevanten und qualitativ hochwertigen Informationen beiträgt (vg. Hamann Hinkelmann/Hoppe/Humm 2025: 74).

Im Vergleich zu dem manuellen Ansatz hat ein automatisierter Ansatz mehrere Vorteile. Er ist kosteneffizient, skalierbar, flexibel und aktuell. Ein Nachteil des automatisierten Ansatzes ist die Notwendigkeit von ausreichende annotierte Daten für das Modelltraining (vg. Hamann Hinkelmann/Hoppe/Humm 2025: 75).

Große Sprachmodelle (LLM) zeigen beeindruckende Fähigkeiten, Daten für verschiedene NLP-Aufgaben zu genieren, jedoch ist ihr Einsatz für Datenaugmentatation im NER-Bereich noch begrenzt (vgl. Santoso et al. 2024: 9652).

Die Umsetzung der Implementierung erfolgte in Python unter Verwendung der Standardbibliotheken SpaCy, ….

## Arbeitsmaterial

Das Arbeitsmaterial umfasst insgesamt 20 Lebensbeschreibungen[[1]](#footnote-1) aus Periodika der Herrnhuter Brüdergemeine. Die *Nachrichten aus der Brüdergemeine* (NBG) wurden zwischen 1819–1894 im Druck veröffentlicht und sind als Primärdigitalisate im digitalen Archiv der [Memorial University of Newfoundland](https://dai.mun.ca/digital/nachrichten/) zugänglich. Die Auswahl des geeigneten Materials beruhte auf mehreren Prämissen: Einerseits mussten die betreffenden Personen in Südafrika leben und wirken, andererseits sollten diese Menschen in ihren Lebensläufen andere Personen erwähnen. Die Digitalisate wurden mit Python-Tesseract für Optical Character Recognition (OCR) erschlossen. Darüber hinaus dient eine Tabelle als Arbeitsmaterial, die alle Personennamen sowie deren Geburts- und Sterbeorte enthält, basierend auf NBD. Diese Tabelle wurde von der SLUB bereitgestellt.

## OCR-Erkennung

Das gesamte Material wurde per Python-Bibliothek *pytesseract* erfasst. *Tesseract* ist eine Open-Source-OCR-Engine von Google. Sie nutzt die Sprachmodelle für die Textextraktion (vgl. Baierer/Zumstein 2016: 73) und unterstützt zahlreiche Bildformate (vgl. [Python-tesseract](https://pypi.org/project/pytesseract/)). Je nach verwendeter Methode[[2]](#footnote-2) liefert OCR-Erkennung unterschiedliche Ergebnisse.

Um die Verbesserung dieser Ergebnisse zu erzielen, wurden die Bilder zuerst vorbereitet (Abbildung 5.1). Dies erfolgt mithilfe einer Python-Bibliothek *OpenCV (CV2).*

Ein Bild, das Text, Schrift, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abb. 5.1 Screenshot des Python-Codeausschnitts zur Bildverbesserung mit der OpenCV-Bibliothek

Zunächst wurden alle JPG-Dateien invertiert, indem die Helligkeitswerte jedes Pixels umgekehrt wurde, sodass helle Pixel dunkel und dunkle Pixel hell werden. Dies erhöht den Kontrast zwischen dem Text und dem Hintergrund. Um verschiedene Grauabstufungen darzustellen, wurde auf die invertierten Bildern Grayscaling angewendet. Dieser Schritt reduziert die Komplexität des Bildes und erleichtert die Extraktion, da es den Fokus auf die Helligkeitswerte legt. Ein Computer kann normalerweise bis zu 256 Graustufen darstellen (vgl. Rawat/ Sharma/ Gusain 2021: 2590). Anschließend wurde das Graustufenbild in ein binäres Bild umgewandelt. Der Binarisierung erfolgt durch Festlegung eines Schwellenwerts. So kann ein hoher Wert bspw. Zeichenstriche entfernen, während ein zu niedriger Wert zu unscharfen Randbereichen führen kann (vgl. ebd.: 2590). Die [CV2](https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html)-Bibliothek bietet verschiedene Typen von Binarisierungsmethoden zur Auswahl. Nach verschiedenen Erprobungen mit unterschiedlichen Einstellungen wurde der am besten passenden Schwellenwert ermittelt und für die THRESH\_BINARY-Methode Entschieden. Bei dieser Methode wurden alle Pixel mit einer Helligkeit über 120 auf weiß 255 gesetzt, während Pixel mit niedriger Helligkeit schwarz 0 wurden. Diese Einstellungen erwiesen sich bei allen Bildern als besonders effektiv und liefern gute Resultate. Zum Schluss wurde der Kontrast verstärk und dadurch wurde Text deutlicher hervorgehoben (vgl. [OpenCV](https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59)). Danach kam OCR zum Einsatz (Abbildung 5.2). Dabei wurden zwei Sprachmodelle kombiniert: Deutsch und Fraktur.



Abb. 5.2 Screenshot des Python-Codeausschnitts zur OCR-Erkennung

Nach der OCR-Erschließung wurden OCR-Fehler analysiert, um mögliche Korrekturschritte abzuleiten. Es wurde festgestellt, dass der größte Teil von Fehler auf falsch erkannten Zeichen liegt, z.B.: *ic), auc), nad, cr, ſc<wer* usw.anstatt *ich, auch, nach, er, ſchwer*. Diese Fehler wurden automatisch durch die korrekten Zeichen ersetzt. Außerdem wurden Silbentrennung am Zeilenende und überflüssige Zeichnen im Text entfernt.

## TEI-XML- Modellierung

Für die vorlegende Arbeit wurde XML nach den Rechtlinien der TEI als semantischen Ebene eingesetzt. Dabei dient es zu der Erfassung der Metadaten von Primärdigitalisaten, zu der Abbildung der Textstruktur (Seitenumbrüche, Überschriften, Absätze, Sätze) und zu der inhaltlichen Annotation (Personen-, Orts-, Körperschaftsnamen, Datumsangaben).

Zu Beginn wurden die Primärdigitalisate analysiert, um festzustellen, welche Informationen in den Annotationen erforderlich oder optional sind. Dabei wurden strukturelle Elemente, ihre Relevanz ermittelt und in welcher Form sie erfasst werden können. Im Fokus stand die Sicherung der Metadaten und des Layouts jedes Digitalisats. Darüber hinaus wurde geprüft, in welchem Umfang die Entitäten erfasst werden sollten.

Im nächsten Schritt wurde das Vokabular des XML-Schemas definiert. Dazu dienten die Richtlinien der TEI, insbesondere [TEI-Header](https://tei-c.org/release/doc/tei-p5-doc/en/html/HD.html) und [Namen, Daten, Personen und Orten](https://www.tei-c.org/release/doc/tei-p5-doc/en/html/ND.html). Um eine standardisierte und einheitliche Darstellung aller Daten in den XML-Dokumenten sicherzustellen, wurden die Datentypen festgelegt und sämtliche Elemente und Attribute definiert, die in den XML-Dateien verwendet werden. Hierbei war entscheidend, nicht nur die ausgewählten Elemente zu berücksichtigen, sondern auch die exakte Reihenfolge der Elemente genau einzuhalten. Dabei wurden die Elemente mit festgelegter Reihenfolge und ohne unterschieden. Genauso wurde mit den Attributen vorgegangen. Im Schema wurde festgelegt, ob ein Attribut obligatorisch ist und ob ein Attribut ein festgelegter Wert hat. Schließlich wurde das Schema (rng-Datei) in *Oxygen* Editor auf Basis einer vormodellierten XML-Datei erstellt.

Jedes TEI-Dokument folgt einer bestimmten Grundstruktur: <teiHeader> und <text>. Im ersten Teil befindet sich die Information über die physische Transkription, die über das <teiHeader>-Element kodiert wurde. Das Element fasst Metadaten zur elektronischen Ressource um, wie die Information über die Verfasser:innen der Primärquelle, verantwortliche Personen usw. Der zweite Teil der XML-Datei fokussiert sich auf den Textbereich und nimmt Bezug auf die digitale Repräsentation. Dabei orientiert sich das <text>-Element an der Inhaltslogik des Textes.

Das oberste Element <teiHeader> enthält die Metadaten eines Digitalisats in einem Hauptbereich: <fileDesc> (Dateibeschreibung) (Abbildung 5.3). Das <fileDesc>-Element umfasst drei obligatorische Elemente: <titleStmt>, <publicationStmt> und <sourceDesc>, in denen die Angaben zum Titel, zur Publikation und Quelle kodiert wurden.

Ein Bild, das Text, Diagramm, Screenshot, Quittung enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abb. 5.3 Screenshot des XML-Schemas (teiHeader-Element) im Oxygen-Editor

Das oberste <text>-Element umschließt den gesamten Text eines Digitalisats und enthält eine Hierarchie von weiteren Elementen (Abbildung 5.4). Dadurch wird die Struktur und Semantik eines Textes definiert. Das untergeordnete <body>-Element enthält die eigentlichen Textabschnitte. Der Textteil einer XML-Datei ist relativ überschaubar, wobei das <div>-Element den gesamten Text umfasst.

Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abb. 5.4 Screenshot des XML-Schemas (text-Element) im Oxygen-Editor

Seiten wurden durch das <p>-Element kodiert. Seitennummern wurden als n-Attribut zu dem <p>-Element zugewiesen. Der Titel jeder Lebensbeschreibung wurde im <head>-Element eingeschlossen. Die Kustode wurde den TEI-Richtlinien als paratextuelle Formelemente im Element <fw> ausgezeichnet. Innerhalb eines <s>-Element wurde die Textsegmente befinden sich die identifizierten Entitäten mit den entsprechenden TEI-Tags: Für die Erfassung von Personen-, Orts-, Körperschaftsnamen wurde über das <persName>-, <placeName>, <orgName>- und <date>-Element kodiert.

Die XML-Modellierung wurde in zwei Etappen durchgeführt: (1) Aufbau der TEI-konformen Grundstruktur und (2) automatische Erkennung und Auszeichnung von Entitäten. Diese Zweiteilung spiegelt sich im GitHub-Repositorium wider, in dem der Code entsprechend in *Teil 1* und *Teil 2* gegliedert ist.

## Informationsextraktion aus Texten

In Bezug auf der Informationsextraktion aus unstrukturierten Daten können unterschiedliche Kategorie von Datentypen identifiziert und extrahiert werden. Man spricht dabei über Entitäten, Beziehungen, Adjektiven und ergänzende Informationen (vgl. Lanquillon/Schacht 2023: 163). Ziel der Extraktion ist es, diese Information zuverlässig zu erkennen, zu normalisieren und in eine auswertbare, strukturierte Repräsentation zu überführen.

Methodisch gliederte sich die Informationsextraktion in mehrere aufeinander folgende Teilschritten. Zu der Erkennung der Entitäten wurde zuerst Named Entity Recognition (NER) durchgeführt und die Ergebnisse in TEI-Struktur zurückgeschrieben (Kapitel 5.3). Danach wurden Relationsextraktion (RE) zur Identifikation semantischer Relationen zwischen den ermittelten Einheiten. Zum Ergänzend wurde eine Koreferenzauflösung und Named Entity Linking (NEL) durch Zuordnung zu Normdaten (Wikidata) umgesetzt (Kapitel 5.4.3). Die Implementierung erfolgte in Python. Als zentrale Komponente kamen Bibliothek spaCy und Wikidata-API zum Einsatz. Die extrahierte und angereicherten Information wurden abschließend in einer Hilfstabelle für die weitere Schritte im Rahmen von Semantic Web Technologie bereitgestellt.

### Named Entity Recognition

Entitäten sind im Kontext der Informationsextraktion jene Einheiten, die in einem Text als Subjekte und Objekte auftritt. Damit seien sie Träger semantischer Bedeutung. Am meinst bekanntesten sind Named Entities wie Personen, Orte, Firmen usw. Je nach Forschungsfrage können auch andere Subjekte wie Krankheiten, Medikamente usw. als Entitäten bezeichnet werden (vgl. Lanquillon/Schacht 2023: 163). Eine zentrale Komponente der Wissensextraktion im Text-Mining ist Named Entity Recognition (NER). NER-Tagger identifizieren und markieren Entitäten im laufenden Text und weisen dabei generische Typen wie *Person* oder *Organisation* zu (vg. Hamann Hinkelmann/Hoppe/Humm 2025: 81).

*SpaCy* ist eine leistungsfähige Python-Bibliothek für Natural Language Prozessing (NLP). Sie verfügt über ein schnelles, statistisches System zur Erkennung von NER. Dazu kann bei *spaCy* eigene, domänenspezifische Entitätsklassen hinzugefügt werden und das Modell mit neuen Beispielen trainiert werden. NER-Tagger erkennt die gängigen Entitätstypen wie *LOC* (Ort/ Region), *MISC* (sonstige Entitäten), *ORG* (Organisation) und *PER* (Person) (vgl. [spaCy](https://spacy.io/usage/linguistic-features#named-entities)).

Zur Vorprüfung wurde das deutschen *spaCy*-Modell (*de\_core\_news\_lg*) auf das Korpus angewandt. Die Ergebnisse blieben deutlich hinter den Erwartungen zurück. Insbesondere traten zahlreiche Fehlklassifikationen bei Personen-, Orts- und Organisationsnamen (Abbildung 5.4).

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

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abb. 5.4 Screenshot der Ergebnisse des *spaCy*-NER-Taggers

Außerdem ist es nicht vorgesehen, in der Standardkonfiguration von Entität-tagger die Datumsangaben zu erkennen. Vor diesem Hintergrund wurde entscheiden, das Modell domänenspezifisch zu trainieren. In diesem Zusammenhang kam die Frage, woher die Trainingsdaten stammen sollen. Man kann ein eigenes Korpus manuell annotiert und damit das Modell zu erweitern. Dieser Ansatz ist jedoch mit erheblichem zeitlichem Aufwand verbunden und in Rahmen dieser Arbeit nicht realisierbar. Deswegen wurde beschlossen, synthetische Trainingsdaten zu erstellen. Aus Kostengründen wurde für das kostenlose, Open-Source-Modell *Llama3* entschieden. *Llama3* ist ein Modell von Meta Inc. Momentan (Zustand 15.08.2025) sind die Modelle in zwei Parametergroßen erhältlich: 8B und 70B (vgl. [Llama](https://ollama.com/library/llama3)). Für diese Arbeit wurde das 8B Modell genutzt.

Das Grundprinzip der Erstellung von synthetischen Trainingsdaten bestand darin: (1) Sätze aus zufällig ausgewählten Entitäten aus unterschiedlichen Listen zu generieren: (2) den Satz teilweise annotieren; (3) die annotierten Sätze zur Erweiterung der *spaCy* NER-Modell zu nutzen. Für jeden Entitätstyp (Personen-, Ortsnamen, Datum) wurde eine separate Liste erstellt. Diese Entitäten basieren auf Daten aus der von SLUB zur Verfügung gestellten Tabelle. Für die Ortsnamen wurden Einträge aus den Spalten *Geburtsort* und *Sterbeort* dieser Tabelle verwendet. Zusätzlich wurde eine Liste für Verben erstellt, die typisch für eine Lebensbeschreibung sind. Da Lebensbeschreibungen eines Mitglieds der Brüdergemeine typischerweise entweder von Ich- oder Er-/Sie-Perspektive verfasst wurde (vgl. Roth 2020: 185–186), wurden aus diesem Grund zwei Prompt erstellt:

1. Erstelle einen deutschen Satz über die Person die sich in *Ort* aufhält oder dort etwas tut. Verwende exakt diesen *Namen* und exakt diesen *Ort*. Nutze genau eines der folgenden *Verben*. Der Satz muss das *Datum* enthalten, allerdings nicht am Satzanfang – es soll möglichst in der Satzmitte oder am Ende erscheinen. Gib ausschließlich den vollständigen Satz aus – keine Erklärungen, Kommentare oder Einleitungen.
2. Schreibe einen deutschen Satz aus der Ich-Perspektive. In diesem Satz beschreibe ich, wie ich zusammen mit *Namen* etwas in *Ort* tue. Der zeitliche Kontext liegt zwischen 1819 und 1894. Verwende exakt diesen *Namen* und exakt diesen *Ort*. Gib ausschließlich den vollständigen Satz aus – keine Erklärungen, Kommentare oder Einleitungen.

Insgesamt wurden 2 049 Sätzen generiert, die diesem folgenden Format entsprechen: [„Satz“, [Startindex einer Entität, Endindex einer Entität, „Entitätstype“]] **Beispiele anpassen**

1. ["Am 27. August 1874 übernachten Johannes Chriſtian Adam in Richter.", [[31, 54, "PERSON"], [58, 65, "LOC"], [14, 18, "DATE"]]]
2. ["Carl Friedr. Genge wird von seinem Aufenthalt in Herrnhut stark geprägt und hat bis zum 20 August 1860 die Gelegenheit, sich ein umfassendes Bild vom Stadtgeschehen zu erfahren.", [[0, 18, "PERSON"], [49, 57, "LOC"], [88, 102, "DATE"]]]
3. ["Johann Gottfried Ferdinand Groſſe vernehmen am 16 August 1850 in Berthelsdorf die Erinnerung an seine letzten Reiseerlebnisse, als er auf und ab ging zwischen den Dorfstraßen.", [[0, 33, "PERSON"], [65, 77, "LOC"], [47, 61, "DATE"]]]
4. ["Ich besuche gemeinsam mit Rodax die Botanische Sammlung in Vanderbijlpark, um neue Pflanzenspezies zu sammeln.", [[26, 31, "PERSON"], [59, 73, "LOC"]]]
5. ["Ich erinnere mich, wie ich mit Veil mitten in der Nacht durch die stillen Straßen von Knysna spazierte, um gemeinsam den Duft der Eukalyptusbäume zu genießen.", [[31, 35, "PERSON"], [86, 92, "LOC"]]]
6. ["Ich erfreue mich mit Reusner an einem sonnigen Nachmittag in Port Elizabeth an der Schönheit des Indian-Oceans-Ufers.", [[21, 28, "PERSON"], [61, 75, "LOC"]]]

Auf Grundlage dieser Sätze wurde das NER-Modell trainiert (Abbildung 5.5). Die Kernkomponenten von *spaCy* arbeitet auf Grundlage statistischer Modelle. D.h., dass jede Entscheidung des Modells eine Vorhersage ist, die auf den aktuellen Gewichtwerten des Modells basiert. Diese Gewichte werden aus den Trainingsbeispielen geschätzt. Das Training selbst erfolgt iterativ. Die Vorhersagen des Modells werden mit den Referenz-Label verglichen und dadurch wird der Gradienten des Fehlers zu berechnen. Der Gradientenabstieg passt die Gewichte an, sodass die Vorhersagen zunehmend den Referenzdaten entsprechen. Ziel des Trainings ist nicht die Beispiele *Auswendiglernen,* sondern Generalisierungsfähigkeiten des Modells zu fordern. Um die Leistung des Modells zu prüfen, sind auch Evaluierungsdaten erforderlich (vgl. [spaCy:Training Pipelines & Model](https://spacy.io/usage/training)). Für die Evaluierung des Modells wurden die während dieser Arbeit erstellten, annotierten Daten verwendet.

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

KI-generierte Inhalte können fehlerhaft sein.

Abb. 5.5 Modelltraining von *spaCy* ([spaCy:Training Pipelines & Model](https://spacy.io/usage/training))

Der nächste Schritt betraf die vormodellierte XML-Dokumente. Zunächst wurde der Text nach Sätzen segmentiert. Auf jeden einzelnen Satz wurde NER-Modell angewendet und die *PER*-, *LOC*- und DATE- Entitäten erkannt. Falls ein Satz keine vorgeschriebenen Entitäten enthält, blieb der Satz unverändert. In jedem Fall wird ein <s>-Element erzeugt. Abschließend wurden die TEI-Dateien manuell überprüft und um den zusätzlichen Entitätstype *ORG* (Organisation) ergänzt. Während der manuellen Korrektur wurden ausschließlich *aktive* Sätze beibehalten. Die Sätze, in deren ein Entitätstype lediglich erwähnt, aber nicht im eigentlichen Kontext verwendet wurde, wurden ignoriert.

**Validierung von ML**

## Relationsextraktion (RE)

Relationsextraktion verfolgt das Ziel, die erkannte Entitäten im Text miteinander zu verknüpfen (vg. Hamann Hinkelmann/ Hoppe/Humm 2025: 84). Beziehungen treten als Verben auf und bilden die syntaktische und semantische Verbindung zwischen Entitäten. Sie können mithilfe regelbasierter Verfahren identifiziert werden. Normalerweise werden die Beziehungen allgemein geschrieben: *ist Chef von, ist angestellt bei* (vgl. Lanquillon/ Schacht 2023: 165). Au diese Weise lassen sich Tripeln direkt bilden (vg. Hamann Hinkelmann/ Hoppe/Humm 2025: 84) **typische Verfahren für RE**

Als Vorbereitung zu der Relationsextraktion wurden zuerst die annotierten Sätze aus den <head>- und <s>-Elementen extrahiert. Außerdem wurden Informationen zum Namen des XML-Dokuments, zum Autor des Digitalisats sowie Link zu Ablageort von NBG erfasst. Die Ergebnisse wurden anschließend in einer JSON-Datei gespeichert:

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abb. 5.6 Screenshot des JSON-Datei-Ausschnitts

Für die Relationsextraktion wurden *Llama3* und ein Framework *LangChain* verwendet. *LangChain* bietet standardisierte Schnittstelle für beliebige LLMs. Damit können LLM-Basierte Anwendungen entwickelt, überwacht und schnell in Produktion gebracht werden. (vgl. [LangChain](https://www.langchain.com/langchain)). Moderne LLMs können eine Vielzahl von sprachbezogenen Aufgaben erledigen, wie Textgenerierung, Übersetzung oder Fragenbeantworten. Dafür wird ein Chat-Modell in der Regel genutzt. Es nimmt eine Liste von Nachrichten als Aufgabe und gibt eine Nachricht als Ausgabe zurück. Eine Nachricht an Chat-Modell ist üblicherweise mit einer Rolle versehen: system, human, assistent und erhalten einen oder mehrere Inhaltsblöcke, die Text oder multimodale Daten umfassen können. Das Chat-Modelle kann ihre Antwort in einer strukturieren Form liefern, z.B. in JSON (vgl. [Chat models](https://python.langchain.com/docs/concepts/chat_models/#interface)).

Im ersten Schritt wurden die möglichen Relationen definiert und in verschiedene Gruppen eingeteilt: *Biografische Basisdaten, Bildung und Beruf, Familie, Reise* usw. Außerdem wurde eine strukturierte Datenklasse festgelegt, die zur systematischen Entitätsextraktion dient. (Abbildung 5.7).

Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

5.7 Screenshot des Python-Codeausschnitts von Humanprompt

In diesem Hintergrund wurde zwei Prompts erstellt: Systemprompt und Humanprompt. Im Systemprompt (Abbildung 5.8) wurde dem KI-Modell die Rahmenbedingungen seiner Aufgabe erklärt. Es wurde festgelegt, welche vordefinierten Entitäten erlaubt sind. Darüber hinaus wurden allgemeine Regeln festgelegt: „KEINE Erklärungen oder zusätzlichen Texte!“.

Ein Bild, das Text, Schrift, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

5.7 Screenshot des Python-Codeausschnitts von Systemprompt

Der Humanprompt (Abbildung 5.9) stellt dem Modell konkreten Text und darin erkannten Entitäten zur Verfügung. Es fordert das Modell auf den aktuellen Kontext zu fokussieren und die Relationen nur auf Grundlage der bereitgestellten Entitäten zu erlauben.

Ein Bild, das Text, Quittung, Screenshot, Algebra enthält.

KI-generierte Inhalte können fehlerhaft sein.

5.9 Screenshot des Python-Codeausschnitts von Humanprompt

Damit Relationsklassifikation effektiv funktioniert, müssen die Entitäten in kurzen Textabschnitten gemeinsam erwähnt werden (vg. Hamann Hinkelmann/ Hoppe/Humm 2025: 85).

## 5.4.3 Datenabgleich

Kohärenzauflösung

Während der Datenvorverarbeiten wurden die Texte in Sätze untergeteilt.

Pronomen oder Umschreibungen von Entitäten können Informationen im Text verschleiern, wenn der Kontext nicht berücksichtigt wird.

Die automatische Zuordnung von Entitäten zu einem Graph erfolgt durch Named Entity Linking (NEL) (vg. Hamann Hinkelmann/Hoppe/Humm 2025: 83)

Da die Maschinen keine indirekten Bedeutungen erkennen können, daher 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. Trotzt der Bemühungen der Deutschen Nationalbibliothek kann die Eindeutigkeit von Normdaten nicht in allen Fällen gewährgeleistet werden. Regelmäßig auftretende Probleme, wie etwa fehlerhafte oder doppelt vorhandene GND-Einträge, lassen sich häufig auf die dezentrale Struktur der Datenerfassung zurückführen, die im Rahmen kooperativer Arbeitsprozesse. 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. Die Erstellung und Pflege von Normdaten stellt auch die Herausforderung im Bereich von Körperschaften, geographische Orte, Werte, Periodika usw. Ihre eindeutige Identifikation erweist sich oftmals als deutlich komplexer. Insbesondere bei Orten treten spezifische Schwierigkeiten auf: Während georeferenzierende Systeme (GIS) in der Regel lediglich punktuelle Koordinaten erfassen, bleiben flächenhafte Ausdehnungen sowie historische Veränderungen unberücksichtigt. Die historische Dimension eines Ortes – etwa sich verändernde politische Zugehörigkeiten, Ortsnamen oder Grenzverläufe – kann daher nur unzureichend über bestehende textbasierte Normdaten erfasst werden und bedarf ergänzender, oft nicht-textueller Quellen (vgl. Kamzelak 2018: 426-427).

## 5.5 RDF

## 5.6 OWL (Lanquillon/Schacht 2023, S. 187)

## 5.7 Turtle

## 5.8 Wikidata-Datenmodell

Laube-Artikel

Zur Auffindbarkeit und technischen Interoperabilität von Projektdaten ist es sinnvoll, etablierte Identifikatoren und Normdaten zu verwenden. Im Rahmen dieser Arbeit wurde Wikidata genutzt, um externe Konzepte und Realitäten referenziert und gemeinsamen Bezugsrahmen zu schaffen.

## 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

## Zusammenfassung

## Literaturverzeichnis

## Anhang

1. Anhang 2 [↑](#footnote-ref-1)
2. Kommerzielle OCR-Software wie *ABBYY* setzt auf einen Wörterbuchabgleich, während Open-Source-Programme wie *Tesseract* und *Ocropus* auf unterschiedliche Sprachmodelle oder neuronale Netzen zurückzugreifen (vgl. Baierer/Zumstein 2016: 73). [↑](#footnote-ref-2)