## Wikidata

Open Data fördert Transparenz, Zugänglichkeit und Wiederverwendbarkeit von Daten. Dies ermöglicht eine offene und nachhaltige Nutzung von Forschungsergebnissen. (vgl. Thiery/Schenk/Thiery 2024: 115). Die FAIR-Prinzipien spielen dabei eine zentrale Rolle, indem sie eine effektive Nutzung und Weitergabe von Forschungsdaten sicherstellen (vgl. Krefeld/Lücke 2019: 140–141). Das Semantic Web bzw. LOD ermöglichen die semantischen Verknüpfungen von öffentlich zugänglichen Daten. Dadurch entstehen offene, umfassende Wissensgraphen, die aus verschiedene wissensperspektiven abgefragt werden können (vgl. Schelbert/Müller 2023: 1).

Eine vollständige Realisierung dieser Vernetzung im kulturhistorischen Bereich steht jedoch noch aus. Hier setzen traditionelle Datenbanken bereits punktuelle Vernetzungen um. Die Nutzung von Normdaten stellt dabei den ersten Schritt in die Richtung einer Verknüpfung mit anderen Datenquellen (vgl. Schelbert/Müller 2023: 2). Normdaten gewährleisten dabei eine eindeutige und konstante Identifikation von gleichen Entitäten wie Personen, Orten oder Begriffen (vgl. Petz 2023: 219). Wird eine Person bspw. über [GND-Nummer](https://www.dnb.de/DE/Professionell/Standardisierung/GND/gnd_node.html) (Gemeinsame Normdaten) 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: 427). Die Entitäten in der GND sind nicht isoliert, sondern verknüpft miteinander, die durch zusätzliche Merkmale detailliert beschrieben sind (vgl. Menzel et al. 2021: 236).

Bei Normierung und Vernetzung von Daten spielt Wikidata eine wichtige Rolle. Es bietet eine strukturierte Plattform für die standardisierte Speicherung und Verknüpfung von Daten (vgl. Gupler 2022: 128). Wikidata ist eine freie und frei zugängliche Datenbank, die strukturierte Daten sammelt. Darüber hinaus bietet Wikidata eine emergente Ontologie, die aus den Beziehungen zwischen Datenobjekten entsteht und so eine komplexe Abbildung der Welt ermöglicht (vgl. Pintscher et al. 2021: 74). Abbildung 4.1 gibt die verschiedenen Komponenten eines Datenansatzes wieder.

Die Grundlage für die Verknüpfungen in Wikidata stellt das *Item* das zentrale Element dar. Jedes *Item* wird durch einen *Item-identifier* bezeichnet, der sich aus dem Präfix *Q* und eine Nummer, dem sog. eindeutigen Bezeichner, zusammensetzt (vgl. [Wikidata:Introduction](https://www.wikidata.org/wiki/Wikidata:Introduction/de)).

Ein Bild, das Text, Screenshot, parallel, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abb. 4.1 Ein Beispiel für die Strukturelemente eines [Wikidata-Datensatzes](https://www.wikidata.org/wiki/Q123494338)

Labels dienen als Bezeichner und machen die Datenobjekte für Menschen verständlich und identifizierbar. Im Gegensatz sind Labels nicht eindeutig und können mehrfach verwendet werden. Zusätzlich können Datenobjekte eine Beschreibung sowie alternative Namen (Aliases) enthalten (vgl. [Wikidata:Introduction](https://www.wikidata.org/wiki/Wikidata:Introduction/de)).

Ein Triple in Wikidata enthält Informationen, die über sämtliche Wikimedia-Portale erfasst wurden. Dies erfolgt durch die Verknüpfung einer Eigenschaft mit mindestens einem Wert. Die Aussagen können weiterentwickelt, mit Anmerkungen ergänzt oder durch zusätzliche Werte und optionale Qualifikationen erweitert sowie in einen spezifischen Kontext eingeordnet werden. Ein Datenobjekt enthält eine Aussage (Statement), die die Eigenschaften (property, *P*) und die dazu gehörigen Werte (values, *V*) eines Objekts beschreibt. Werte in Wikidata können Datenobjekte oder quantitative Werte, ebenso wie unbekannte oder kein Wert sein. Es ist auch möglich, dass ein Statement mehrere Werte für die gleiche Eigenschaft haben. Werte können durch Qualifikatoren weiter spezifiziert werden, um detaillierte Zusatzinformationen zu geben. Darüber hinaus können Statements durch Verweise auf externe Quellen ergänzt werden. (vgl. [Help:Statements](https://www.wikidata.org/wiki/Help:Statements/de)).

Die kollaborative Arbeit wird durch verschiedene Werkzeuge unterstützt: Diskussionsseiten, Versionshistorien, (vgl. Pintscher et al. 2021: 74), QuickStatements (QS) usw. QS ermöglich automatisch Wikidata-Einträgen zu bearbeiten. Damit können Aussagen, Bezeichnungen, Beschreibungen usw. hinzugefügt und entfernt werden. Dies kann entweder von externen Programmen oder Code durchgeführt werden (vgl. [Help:QuickStatements](https://www.wikidata.org/wiki/Help:QuickStatements)).

## Methode

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. 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. 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. 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: 68,74–75). In der vorlegenden Arbeit wird versucht, den Prozess so weit wie möglich zu automatisieren. Daher wird ein hybrider Ansatz gewählt: LLMs übernehmen die Extraktion von Entitäten und Relationen, während manuelle Eingriffe die inhaltliche Qualität sichern.

Vor diesem Hintergrund die Umsetzung erfolgt in verschiedene Verarbeitungsschritte, die den gesamten Prozess von der Texterkennung bis zur Erstellung von Wissensgraphen abbildet. Zunächst wird die OCR-Erkennung durchgeführt. Anschließend werden die erschlossenen Texte in einem TEI-konformen Modell strukturiert. Darauf aufbauend werden Verfahren der Informationsextraktion eingesetzt. Diese Punkt beinhaltet Named Entity Recognition (NER), Relationsextraktion (RE) und Entity Linking (EL) zur Ergänzung der erkannten Entitäten durch ein Wikidata-Datamodell. Schließlich wird ein RDF-Turtle erstellt. Die Umsetzung der Implementierung erfolgte in Python. Das Skript wurde mit der Unterstützung von ChatGPT-4o und ChatGPT-5 entwickelt. Um die methodische Vorgehensweise transparent zu gestalten, wurde der vollständige Chatverlauf im Anhang 1 gespeichert.

## 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

Die Digitalisate wurden per Python-Bibliothek *pytesseract* erfasst. *Tesseract* ist eine Open-Source-OCR-Engine von Google. Sie nutzt die Sprachmodelle zu der Textextraktion (vgl. Baierer/Zumstein 2016: 73) und unterstützt zahlreiche Bildformate (vgl. [Python-tesseract](https://pypi.org/project/pytesseract/)). Die Erkennungsqualität variiert je nach Verfahren[[2]](#footnote-2).

Um die Ergebnisse zu verbessern, 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 die originale JPG-Dateien invertiert, indem die Helligkeitswerte jedes Pixels umgekehrt wurden, 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 unterschiedlich wirksam, liefern jedoch 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)). Die Abbildung 5.2 (a–d) veranschaulicht den Prozess der Vorverarbeitung bis OCR und zeigen die Wirkung der einzelnen Schritte auf das Ausgansmaterial.

Ein Bild, das Text, Papier, Dokument, Brief enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Text, Handschrift, Karte Menü, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Text, Screenshot, Schrift, Schwarzweiß enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

KI-generierte Inhalte können fehlerhaft sein.

1. Originalscan (b) Invertiertes Bild (c)Binarisierung (d) OCR-Ausgabe

Abb. 5.2 (a – d) Prozess der OCR-Vorbereitung und -Erkennung

Danach kam OCR zum Einsatz (Abbildung 5.3). Dabei wurden zwei Sprachmodelle kombiniert: Deutsch und Fraktur.



Abb. 5.3 Screenshot des Python-Codeausschnitts zur OCR-Erkennung

Nach der OCR-Erschließung wurden OCR-Fehler analysiert, um mögliche Korrekturschritte abzuleiten. Dazu kam ein hybrides Verfahren zum Einsatz. Zunächst wurden wiederkehrende Fehlmuster mittels regulärer Ausdrücke (Regex) erkannt. 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. Die Eingriffe blieben bewusst minimal-invasiv, um das erkannte Layout nicht zu beeinträchtigen. Da auf dieser Grundlage die Grundstruktur des Dokuments automatisch in XML zu annotieren.

Trotzt sorgfältiger Bildvorverarbeitung traten mehrere Einschränkungen. Die Bildqualität war heterogen. Bspw. variierten Farbstiche und Kontraste von den Bildern innerhals desselben Bandes, sodass keine einheitliche Parametrisierung möglich war. Die Einstellungen der Bildvorverarbeitung mussten daher seitenweise angepasst werden, was mit einem erheblichen Zeitaufwand verbunden ist, der im Rahmen dieser Arbeit nur begrenzt zu leisten war. Zusätzlich erschwerten Verschmutzungen und Randabschattungen die OCR. Flecken wurden bspw. als Komma oder Punkt interpretiert. Darüber hinaus wurden einzelne Ziffern verwechselt (z.B. 1/I/|, 0/O). Diese Fehler mussten manuell korrigiert werden.

## TEI-XML- Modellierung

Für die vorlegende Arbeit wurde XML nach den Richtlinien 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 und zu jedem XML-Dokument zugewiesen.

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 selbstschließende <pb>-Element kodiert. Seitennummern wurden als n-Attribut zu dem <pb>-Element zugewiesen. Der Titel jeder Lebensbeschreibung wurde im <head>-Element eingeschlossen. Da ein Titel häufig Eigennamen enthaltet, erlaubt das Schema innerhalb von <head> die Verwendung von den TEI-Elementen für die Auszeichnung von Entitäten (<persName>-, <placeName>, <orgName>- und <date>-Elemente). In einigen Texten waren Kustoden vorhanden. Diese wurde als paratextuelle Formelemente im Element <fw> ausgezeichnet. Darüber hinaus wurde jeder Absatz in einem <p>-Element kodiert, indem die Absatzgrenzen aus dem OCR-Text (\n\n) abgeleitet wurden.

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. Zur Erkennung von Entitäten wurde ein automatischer NER-Verfahren eingesetzt (Kapitel 5.4.1). Die Abbildung 5.5 zeigt eine im XML ausgezeichnete Seite der Lebensbeschreibung von Johannes Fritsch:

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

KI-generierte Inhalte können fehlerhaft sein.

Abb. 5.5 Screenshot eines Ausschnitts des XML-Dokuments zur Lebensbeschreibung Johannes Fritsch im Oxygen-Editor

Deswegen wurde die XML-Modellierung 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 wurde.

Um erkannte Entitäten später mit Normdaten zu verknüpfen, wurde zunächst auf eine manuelle Normdatenzuordnung verzichtet. Aus diesem Grund wurden entsprechende Attribute wie @ref nicht definiert. Ziel war es, den aufwendigen händischen Abgleich zu vermeiden. Rückblicken war es jedoch ungünstig, die Datumsangaben während der manuellen Korrektur nicht zu normalisieren. Dies hätte anschließende Schritte vereinfacht und Bearbeitungszeit später eingespart.

## 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 aufeinanderfolgende 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 Entity Linking (EL) durch Zuordnung zu Normdaten (Wikidata) umgesetzt (Kapitel 5.4.3). Die Implementierung erfolgte in Python. Als zentrale Komponente kamen Bibliothek spaCy, Wikidata-API und LLM *llama3* 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). In der vorlegenden Arbeit wurden NER mithilfe NER-Tagger von *spaCy* erkannt.

*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 (TD) 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.

Große Sprachmodelle (LLM) können schon viele Aufgaben gut lösen und Daten erzeugen, aber ihre Anwendung für NER-Datenerweiterung wurden kaum benutzt (vgl. Santoso et al. 2024: 9652). Santogo (2024) nutzt LLMs zur Generierung synthetischer NER-Daten und zeigt, dass sich mit dieser Methode bereits mit wenigen annotierten Beispielen überzeugende Ergebnisse erzielen lassen. Anknüpfend an Santogos Ansatz wurde in der vorlegenden Arbeit die gleiche Idee verfolgt, allerdings mit einer abweichenden Methodik. 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 TD 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 bereitgestellten Tabelle. Diese beinhaltet Information zu über 1 000 Personen. Für die Ortsnamen wurden die Einträge aus den Spalten *Geburtsort* und *Sterbeort* 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 [*Name*] 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 [*Name*] und exakt diesen [*Ort*]. Gib ausschließlich den vollständigen Satz aus – keine Erklärungen, Kommentare oder Einleitungen.

Insgesamt wurden 1 977 Sätzen generiert, die diesem folgenden Format entsprechen: [„Satz“, [Startindex einer Entität, Endindex einer Entität, „Entitätstyp“]]

1. ["Am 27. August 1874 übernachten Johannes Chriſtian Adam in Richter.", [[31, 54, "PER"], [58, 65, "LOC"], [3, 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, "PER"], [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, "PER"], [65, 77, "LOC"], [47, 61, "DATE"]]]
4. ["Ich verbringe gemeinsam mit dem Wunderling in Nigel einen unvergesslichen Nachmittag bei der Entdeckung alter Schätze in einem überirdischen Arkonarium.", [[32, 42, "PER"], [46, 51, "LOC"]]]
5. ["Ich gehe mit Dober gemeinsam durch die Straßen von Uitenhage, um die neuesten Nachrichten über die Missionsstation zu besprechen.", [[13, 18, "PER"], [51, 60, "LOC"]]]
6. ["Ich unternehme gemeinsam mit Seifert eine Wanderung durch die sandigen Dünen von Calvinia, um an der Sonne warmen Nachmittags die Auswirkungen des Orkans auf die Landschaft zu studieren.", [[29, 36, "PER"], [81, 89, "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 es 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)). Dafür 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))

Die ersten Experimente zeigten, dass das domänenspezifische NER-Modell nicht die erwartete Leistung erbrachte. Um die Leistungsfähigkeit zu erhöhen, wurde entschieden, das Modell mit der Kombination aus synthetischen und authentischen Daten erneut zu trainieren. Dafür wurde aus annotierten XML-Dokumenten von Moravian Knowledge Network bei [Zenodo](https://zenodo.org/communities/mknetwork/records?q=&l=list&p=1&s=10&sort=newest) 96 Sätzen gewonnen sowie zusätzlich 149 annotierte Sätze aus zwei Lebensbeschreibungen von Johannes Friedrich Lemmerz und Adolph Bonatz extrahiert. Anschließend wurde das Modell erneuet trainiert und getestet.

Für die Testdaten wurden insgesamt 709 Sätzen aus den 18 Lebensbeschreibungen gewonnen. Sie entsprechen dem folgenden Format: {„text“: „Satz aus einer Lebensbeschreibung“, „entities“: [[Startindex einer Entität, Endindex einer Entität, „Entitätstyp“]]}

1. {"text": "Lebenslauf des am 25ſten Februar 1819 in Niesky heimgegangenen verwitweten Bruders Johann Gottfried Schulz.", "entities": [[83, 106, "PER"], [41, 47, "LOC"], [18, 37, "DATE"]]}
2. {"text": "Ich bin zu Görlig den 4ten April 1734 geboren.", "entities": [[11, 17, "LOC"], [22, 37, "DATE"]]}
3. {"text": "Mein Vater war Johann Chriſtoph Schulz Lehrer am daſigen Gymnaſium und meine Mutter Chriſtiane Dorothee Hellwig.", "entities": [[84, 111, "PER"], [15, 38, "PER"]]}

Anschließend wurden die Sätze in das erforderliche Format konvertiert und die Tests[[3]](#footnote-3) ausgeführt. Vergleichen wurden zwei domänenspezifische Modelle: NER-Modell mit gemischten TD und NER-Modell, die anhand rein synthetischen TD trainiert. Die Leistung jeweiligen Modells wurde Anhang der wichtigen Metriken *Precision* (P), *Recall* (R) und *F-Score* (F) bewertet. P zeigt der Anteil korrekt vorhergesagter Annotationen. R beschreib den Anteil der insgesamt relevanter Beispiele aus dem Korpus, die tatsächlich gefunden wurden (vgl. Mair Kupietz/Schmidt 2018: 12). F zeigt harmonisches Mittel aus P und R. Alle drei werten sollten nach dem Modelltraining steigern (vgl. [spaCy:Training Pipelines & Model](https://spacy.io/usage/training)).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NER | NER-Modell (gemischte TD) | | | NER-Modell (synthetisch TD) | | |
| P | R | F-Score | P | R | F-Score |
| LOC | 78,00 | 75,92 | 76,95 +19,17 | 59,50 | 56,14 | 57,78 |
| DATE | 72,97 | 68,69 | 70,77 +18,38 | 66,67 | 43,14 | 52,39 |
| PER | 47,99 | 34,79 | 40,34 +28,16 | 13,75 | 10,93 | 12,18 |
| ORG | 53,33 | 11,68 | 19,16 | 0,00 | 0,00 | 0,00 |

Tabelle. 5.1 Evaluierungsergebnisse zweier NER-Modelle (NER-Modell (gemischte TD) vs. NER-Modell (synthetisch TD))

Die Evaluierung (Tabelle 5.1) zeigt deutlich, dass NER-Modell mit gemischten TD dem anderen Modell in allen getesteten Kategorien übertreffend ist. Besonders wird dieser Unterschied bei der Erkennung von Orts- und Datumsangaben sichtbar. Dabei erreicht das gemischte NER-Modell F-Scores von 76,95 bzw. 70,77 und liefert damit einen robusten Wert. Bei der Erkennung von Personennamen fällt auf, dass auch das gemischte Modell nur mäßige Ergebnisse erzielt, jedoch immer noch besser als das synthetische Modell. Am problematischste bleibt die Kategorie ORG. Da Organisationsnamen im synthetischen TD von LLM nicht generiert wurden, war es wenig überraschend, dass das Modell in dieser Kategorie keine Erkennungsleistung zeigt (F=0). Bei dem gemischten Modell wird mit einem F von 19,16 sehr geringe Werte erreicht. Dies deutet darauf hin, dass die vorhandenen Beispiele für ORG nicht ausreichen, um die höhere Modellleistung zu erzielen. Insgesamt konnte durch den Einsatz von 245 authentische Sätzen die Modellleistung bei drei Entitätstypen LOC, DATE, PER auf 19,17, 18,38 und 28,16 Prozent erhöht werden. Damit zeigt sich, dass die gewählte Methode nicht nur leistungsfähig ist, sondern auch ein deutliches Potenzial für die weitere Verbesserung der Erkennung von NER in den historischen Texten ist.

Nach dem Training konnte das Modell bereits auf sämtliche XML-Dokumente angewendet werden. 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. Für die Sätzen, in deren die Entitäten erkannt wurden, wurde ein <s>-Element automatisch erzeugt, das die Elemente <persName>, <placeName> und <date> einschließt. Abschließend wurden die TEI-Dateien manuell überprüft und um den zusätzlichen Entitätstype *ORG* (Organisation) ergänzt. Dabei ist zu beachten, dass ORG (<orgName>) in diesem Kontext nicht Organisationsnamen in klassischen Sinn bezeichnet, sondern Herrnhuter Brüdergemeine in unterschiedlichen Regionen (z. B. Mission unter den Hottentotten oder Herrnhuter Brüdergemeine in Elim) oder Arbeitsplätze wie eine *Tischlerei.* 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.

## Relationsextraktion (RE)

Relationsextraktion verfolgt das Ziel, die erkannte Entitäten im Text miteinander zu verknüpfen (vg. Hamann Hinkelmann/ Hoppe/Humm 2025: 84). Relationen bilden die syntaktische und semantische Verbindung zwischen Entitäten. Die Komplexität ihrer Ermittlung führte zur Entwicklung unterschiedlicher Methoden (vgl. Tao/Wang/Bai 2024: 1), wie bspw. regelbasierte Verfahren (Dependency Tree), überwachte Ansätze (POS), semi-überwachtes Lernen (Supervised Learning) oder Active Learning mittels eines Klassifikators (vgl. Lanquillon/ Schacht 2023: 178–181). Seit kurzem kamen auch LLMs zum Einsatz (vgl. Tao/Wang/Bai 2024: 1). Mit der Einführung generativer Sprachmodelle wie GPT und Llama richtet sich die Forschung zur RE zunehmend auf deren Nutzung. Mit GPT-2 (2019) wurde die Idee von prompt-basierten RE populär und die ersten Ansätze konzentrierten sich auf die Suche nach optimalen Prompts für Pretrained Language Models (PLMs). Mit der Einführung von chatbasierten Modellen verschiebt sich der Fokus bei RE hin zu generativen Ansätzen, Entitäten und Relationen zusammen zu extrahieren (vgl. Swarup et al. 2025: 6670-6671). Darüber hinaus gab es bereits erfolgreiche Ansätze RE auf Dokumentebene durchzuführen (Xue et al. 2024), dennoch sind die meisten bestehenden Methoden für RE auf Satzebene (vgl. Xue et al. 2024: 211).

Die Forschung im Bereich der RE lässt sich in zwei Paradigmen einteilen: Relation Classifikation (RC) und Joint Relation Extraction (JRE). Bei RC wird eine Beziehung zwischen zwei bereits bekannten Entitäten bestimmt. JRE dagegen zielt darauf ab, komplette Triples aus Texten zu gewinnen. Dabei sind die Entitäten unbekannt (vgl. Swarup et al. 2025: 6672). Im Rahmen der vorlegenden Arbeit wurde zunächst JRE-Methode mit *llama3*, *qwen3*, *mistral-small3.1* und *deepseek-r1* getestet. Dabei zeigte sich, dass viele Entitäten entweder nicht erkannt oder falsch positiv identifiziert wurden, wodurch auch ihre Relationen fehlerhaft waren. In manchen Fällen trat ein weiteres Problem auf: Die Modelle halluzinierten und erfanden Relationen oder Entitäten, die im Text nicht vorhandelt waren, oder brachten die Analyse nach wenigen Sätzen ab. Deswegen wurde entschieden, die RE mit der RC-Methode durchzuführen.

Als Vorbereitung zu der RE 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 RE 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)).

Normalerweise werden die Beziehungen allgemein geschrieben: *ist Chef von, ist angestellt bei* (vgl. Lanquillon/ Schacht 2023: 165, 178–181). Au diese Weise lassen sich Tripeln direkt bilden (vg. Hamann Hinkelmann/ Hoppe/Humm 2025: 84). Aus diesem Grund wurden zuerst die möglichen Relationen definiert und in verschiedene Gruppen eingeteilt: *Biografische Basisdaten, Bildung und Beruf, Familie, Reise* usw. Außerdem wurde eine strukturierte Datenklasse festgelegt. Sie dient als Beispiel für Definition aller möglichen Entitätstypen (Abbildung 5.7).

Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

5.7 Screenshot des Python-Codeausschnitts von Humanprompt

Danach wurde zwei Prompts erstellt: Systemprompt und Humanprompt. Im Systemprompt (Abbildung 5.8) wurde dem KI-Modell die Rahmenbedingungen definiert. In diesem Fall muss das Modell aus biographischen, historischen und religiösen Texten strukturierte Informationen extrahieren: Relation zwischen den vorgegebenen Entitäten. Darüber hinaus wurde festgelegt, welche vordefinierten Entitäten erlaubt sind. Die Beziehungen wurden als mögliche Beispiele definiert. Dann soll das Ganze in vorgeschriebenes JSON-Format zurückgegeben werden.

Ein Bild, das Text, Schrift, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

5.8 Screenshot des Python-Codeausschnitts von Systemprompt

Das Humanprompt (Abbildung 5.9) stellt dem Modell konkreten Text und darin erkannten Entitäten zur Verfügung. Diese fordert das Modell auf den aktuellen Kontext zu fokussieren und die Relationen nur auf Grundlage der bereitgestellten Entitäten zu finden. Falls es zu einer Relation ein relevantes Datum gibt, soll dieses zusätzlich im Feld *zeit* vermerkt werden. Nach der Vorbereitung der RE wurde das Modell ausgeführt.

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

KI-generierte Inhalte können fehlerhaft sein.

5.9 Screenshot des Python-Codeausschnitts von Humanprompt

Für die Evaluierung der RE wurden P, R und F-1 verwendet. Berechnet werden sie nach den folgenden Formeln:

Wobei True Positive (TP) es alle korrekt erkannten Relationen sind, False Positive (FP) ist eine Relation, die vom Modell erkannt wurde, aber ihre Type oder beteiligte Entitäten nicht den Bedingungen des Goldstandards entspricht. False Negatives (FN) wurde bestimmt, indem die im Goldstandards Relationen gezählt wurden, die vom Modell nicht erkannt wurden (vgl. Li et al. 2017: 5–6).

**Evaluieren!!!**

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)
3. Zur Nachvollziehbarkeit der Ergebnisse wurden die ermittelten Metriken dokumentiert. Auf GitHub-Repositorium stehen zwei separate Dateien zur Verfügung, die die Auswertungen für das das synthetische Modell (metrics\_v1.json) sowie für das gemischte Modell (metrics.json) enthalten. Diese Dateien wurden automatisch beim Testlauf generiert. [↑](#footnote-ref-3)