## SPARQL

Laube-Artikel

## Visualisierung

Dies wurde mithilfe spezieller Web-Frameworks erreicht, die es ermöglichen, Python effektiv für die Webentwicklung zu nutzen.

Diese Frameworks lassen sich in zwei Typen unterteilen: Full-Stack und Non-Full-Stack. Erstere bieten eine umfassende Lösung für alle technischen Anforderungen für die Webentwicklung (z. B. Django, Pyromid), während letztere nur Grundfunktionen wie das Weiterleiten von HTTP-Anfragen an die relevanten Controller ermöglichen. Non-Full-Stack benötigen in der Regel Unterstützung durch weitere Application Programming Interfaces (APIs) und Tools. Da die älteren Frameworks wie *Flask* und *Django* die Kenntnisse in HTML und CSS erfordern (vgl. Khorasani 2022: [Getting Started with Streamlit](https://learning.oreilly.com/library/view/web-application-development/9781484281116/html/523242_1_En_1_Chapter.xhtml)), entschieden wir uns gegen deren Verwendung und wählten stattdessen das Framework *Streamlit*.

*Streamlit* ist ein Open-Source-Python-Framework, das im Oktober 2019 veröffentlicht wurde. Es wurde entwickelt, um Datenwissenschaftler:innen und Ingenieur:innen im Bereich maschinelles Lernen zu ermöglichen, Daten ohne Vorkenntnisse in der Webentwicklung direkt in einer Benutzeroberfläche zu präsentieren. Benutzer:innen können mit der Anwendung interagieren, indem sie Daten ändern oder hochladen, um sie zu visualisieren oder unterschiedliche Modelle anzuwenden (vgl. Raghavendra 2023: [Introduction to Streamlit](https://learning.oreilly.com/library/view/beginners-guide-to/9781484289839/html/532543_1_En_1_Chapter.xhtml)). Die Entwicklung erfolgt ausschließlich in Python und die Entwicklungszeit wird deutlich verkürzt, da die Änderungen am Webdokument direkt am Document Object Model (DOM)[[1]](#footnote-1) vorgenommen werden. Dadurch werden Seitenaktualisierungen für die Nutzer:innen in wenigen Sekunden sichtbar (vgl. Khorasani 2022: [Getting Started with Streamlit](https://learning.oreilly.com/library/view/web-application-development/9781484281116/html/523242_1_En_1_Chapter.xhtml)).

Mit *Streamlit* können sich Entwickler:innen vollständig auf die Implementierung der Backend-Logik konzentrieren, während die Frontend-Umsetzung weitgehend vom Framework übernommen wird. Die persönliche Gestaltung einer Webseite wird mithilfe von Sidebar, Spalten, Container usw. erfüllt. Zudem können dynamische Inhalte mithilfe von Platzhaltern angezeigt werden. Darüber hinaus können mit *Streamlit* beliebig viele Seiten und verschachtelte Unterseiten erstellt werden (vgl. Khorasani 2022: [Archietecting the User Interface](https://learning.oreilly.com/library/view/web-application-development/9781484281116/html/523242_1_En_3_Chapter.xhtml)).

Der erste Schritt zu der Konfiguration der Webseite erfolgte mithilfe der Python-Bibliothek *Streamlit*. Mit dem Befehl st.set\_page\_config() wurden der Titel und das Layout definiert. Bei der Erstellung der Webseite war es wichtig, eine durchdachte Struktur zu entwickeln. Um die interaktive digitale Darstellung des Friedhofs von der Analyse zu trennen, wurden zwei separate Seiten konzipiert: *Digitaler Gottesacker* und *Analyse*. Dies wurde durch Konfiguration einer Sidebar (st.sidebar) im linken Bereich der Webseite ermöglicht, was eine intuitive Navigation zwischen den beiden Seiten erlaubt.

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

1. DOM ist eine Programmier-Schnittstelle für HTML- und XML-Dokumente, die die logische Struktur und den Zugriff auf Dokumente definiert. Damit kann mit einigen Ausnahmen auf alle Elemente eines HTML- oder XML-Dokuments zugegriffen und diese verändert, entfernt oder ergänzt werden (vgl. World Wide Web Consortium (W3C): [What ist he Document Object Model?](https://www.w3.org/TR/WD-DOM/introduction.html)). [↑](#footnote-ref-1)