BACHELORARBEIT

Entwicklung eines Datenanalyse-Tools zur Synthese von durch Google Maps gesammelten Nutzerdaten

vorgelegt am 13.02.2026 von
Isabelle Pryzwanski
Matrikelnr.: [redacted]

Erstprüferin: Prof. Dr. Sabine Schumann
Zweitprüferin: Prof. Dr. Marina Tropmann-Frick

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Fakultät Informatik und Digitale Gesellschaft
Finkenau 35
20081 Hamburg

Zusammenfassung

Diese Bachelorarbeit untersucht, inwiefern sich aus Google Maps-Aktivitätsdaten automatisiert auf Nutzerverhalten und geografische Lebensmittelpunkte rückgeschlossen werden kann. Unter Synthese wird die Zusammenführung einzelner Suchanfragen zu einem kohärenten Bild digitaler Nutzungsmuster verstanden.

Die zentrale Herausforderung liegt in der Verarbeitung heterogener, unstrukturierter Suchanfragen aus HTML-formatierten Daten. Es wurde ein Python-basiertes Analysetool entwickelt, das Google Maps-Aktivitätsdaten verarbeitet, analysiert und zu einem interpretierbaren Gesamtbild synthetisiert. Die Daten werden vor der Analyse mittels Parsing-Algorithmen, regelbasierter Adressnormalisierung sowie semantischer Anreicherung über die Google Maps Geocoding API aufbereitet. Die Implementierung folgt Privacy-by-Design-Prinzipien mit ausschließlich lokaler Datenverarbeitung.

Die Evaluation belegt mit vier realen Datensätzen, dass selbst ohne explizite Standortfreigabe präzise Rückschlüsse auf Lebensmittelpunkte und Routinen möglich sind.

Abstract

This bachelor thesis examines the extent to which Google Maps activity data can be used to automatically infer user behavior and geographical centers of life. Synthesis is understood as the combination of individual search queries into a coherent picture of digital usage patterns.

The central challenge lies in processing heterogeneous, unstructured search queries from HTML-formatted data. A Python-based analysis tool was developed that processes and analyzes Google Maps activity data and synthesizes it into an interpretable overall picture. Before analysis, the data is prepared using parsing algorithms, rule-based address normalization, and semantic enrichment via the Google Maps Geocoding API. The implementation follows privacy-by-design principles with exclusively local data processing.

The evaluation uses four real data sets to prove that even without explicit location sharing, it is possible to draw precise conclusions about people's centers of life and routines.

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CEST	Central European Summer Time
CSS	Cascading Style Sheets
CSV	Comma-separated Values
DSGVO	Datenschutz-Grundverordnung
GM	Google Maps
GPS	Global Positioning System
KI	Künstliche Intelligenz
JSON	JavaScript Object Notation
MESZ	Mitteleuropäische Sommerzeit
URL	Uniform Resource Locator
HTML	Hypertext Markup Language
XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Code-Verzeichnis

1 Einleitung

1.1 Motivation

Im März 2010 erklärte das Bundesverfassungsgericht die Vorratsdatenspeicherung in Deutschland für verfassungswidrig und bemängelte den unverhältnismäßigen Eingriff in das Grundrecht auf informationelle Selbstbestimmung *(Biermann, 2010)*. Der Grünen-Politiker Malte Spitz demonstrierte die Aussagekraft solcher Metadaten. Er verklagte die Deutsche Telekom auf Herausgabe seiner Verbindungsdaten und visualisierte diese über sechs Monate. Das Ergebnis war ein detailliertes Bewegungsprofil *(Biermann, 2015)*.

Während dieser Fall die Vorratsdatenspeicherung für staatliche Überwachung von Telekommunikationsdaten betraf, sammeln private Technologiekonzerne wie Google Nutzungsdaten auf Grundlage von Einwilligungserklärungen. Seit Inkrafttreten der DSGVO 2018 müssen diese Daten auf Anfrage in maschinenlesbarem Format bereitgestellt werden (vgl. DSGVO, Art. 20, Abs. 1). Google erfüllt dies über „Google Takeout“.

Bestehende Ansätze zur Analyse von Google Takeout-Daten fokussieren sich auf GPS-basierte Location-History-Daten mit kontinuierlich aufgezeichneten Koordinaten *(Ruktanonchai et al., 2018)*. Tools wie der *Location History Visualizer (DovarFalcone, o. D.)* verarbeiten diese strukturierten JSON-Dateien, die Zeitstempel und Geokoordinaten enthalten. Google Maps-Suchanfragen wurden bisher kaum untersucht, obwohl sie implizite Standortinformationen enthalten: Sie dokumentieren Absichten und Interessen, ohne dass Nutzer:innen aktiv ihre Position freigeben. Ein Tool zur systematischen Auswertung dieser impliziten Daten schließt eine methodische Lücke in der Analyse personenbezogener Geodaten.

Die vorliegende Arbeit untersucht die Forschungsfrage: Inwiefern lassen sich aus den Aktivitätsdaten von Google Maps, die über Google Takeout verfügbar sind, automatisiert Rückschlüsse auf das Nutzerverhalten und auf geografische Lebensmittelpunkte ziehen? Es wird ein Python-basiertes Analysetool entwickelt, das Google Maps-Aktivitätsdaten verarbeitet, analysiert und zu einem interpretierbaren Gesamtbild synthetisiert.

1.1 Zielsetzung

Das Analysetool soll Nutzer:innen ermöglichen, die über Google Takeout bereitgestellten Google Maps-Aktivitätsdaten systematisch auszuwerten und visuell aufzubereiten. Unter Synthese wird dabei die Zusammenführung einzelner Analyseergebnisse zu einem

kohärenten Bild digitaler Nutzungsmuster verstanden. So sollen konkrete Erkenntnisse über das eigene Aktivitäts- und Suchverhalten zugänglich werden. Dabei werden zeitliche und geografische Verteilungen und kategorische Schwerpunkte der Aktivitäten ausgewertet. Mithilfe des Tools können Nutzer:innen die eigene digitale Spur im Google-Ökosystem ohne technische Vorkenntnisse sichtbar machen und kritisch reflektieren.

Das Tool extrahiert die Informationen aus den Daten, normalisiert heterogene Suchanfragen und reichert diese mit semantischen Ortstypen über die Google Geocoding API an. Die Synthese der Analyseergebnisse ermöglicht es, implizite Informationen über Routinen, Interessen und Lebensmittelpunkte sichtbar zu machen, auch ohne explizite Standortdaten. Die Arbeit zeigt sowohl die technischen Möglichkeiten der Datenanalyse als auch die Herausforderungen bei der Normalisierung unstrukturierter Eingaben.

1.2 Aufbau der Arbeit

Die Arbeit gliedert sich in neun Kapitel. Es werden nötige Grundlagen erklärt, ein Konzept erstellt und dieses dann technisch umgesetzt und evaluiert und diskutiert.

Zunächst werden in Kapitel 2 die theoretischen Grundlagen der Arbeit dargelegt und Funktionsweisen von Google Maps und Google Takeout beschrieben. In Kapitel 3 wird die Datengrundlage der Arbeit beleuchtet. Die Struktur der Google Maps-Aktivitätsdaten (im folgenden GM-Aktivitätsdaten) wird analysiert, das HTML-Dateiformat dokumentiert und die Heterogenität der Suchanfragen untersucht. Weiterhin wird die Erhebung der Validierungsdatensätze unter Berücksichtigung der DSGVO-Anforderungen erläutert.

Auf dieser Basis entwickelt Kapitel 4 die Konzeption des Analysetools. Es begründet die Technologieauswahl, definiert Datenverarbeitungsstrategien für Parsing, Normalisierung und Kategorisierung und spezifiziert die Analysemethoden für Häufigkeits-, kategorische, geografische und zeitliche Auswertungen. Kapitel 5 beschreibt die Softwarearchitektur und den Datenfluss durch die Datenverarbeitungspipeline. In Kapitel 6 wird die technische Implementierung aller Komponenten dokumentiert, indem für jedes Modul die eingesetzten Bibliotheken, Algorithmen und Datenstrukturen beschrieben werden. Anschließend evaluiert Kapitel 7 das entwickelte Tool durch funktionale Tests, eine Performance-Analyse und Usability-Bewertungen mit Hilfe von Validierungsdatensätzen. Kapitel 8 diskutiert die Ergebnisse kritisch, reflektiert methodische Limitationen, erörtert Datenschutzimplikationen und zeigt Verbesserungspotenziale auf. Es erfolgt ein Ausblick auf zukünftige Weiterentwicklungsmöglichkeiten. Abschließend werden die Erkenntnisse der Arbeit zusammengefasst und die Forschungsfrage beantwortet.

2 Grundlagen

2.1 Google Maps

Google Maps (GM) ist ein webbasierter Kartendienst, der seit 2005 verfügbar ist und zu den meistgenutzten digitalen Navigationsdiensten weltweit zählt (*Taylor, 2005*). Der Dienst wird monatlich von über zwei Millarden Nutzer:innen verwendet *(Li, 2024)*. GM ist eine umfassende Plattform für Navigation, Standortsuche und die Beschaffung geografischer Informationen (*Jens, 2020*).

Die Kernfunktionalitäten liegen in der Anzeige von Karten, Standortsuche, Routenplanung für verschiedene Fortbewegungsarten und Navigation mit Echtzeit-Verkehrsinformationen. Der Dienst stellt Informationen zu Lokalitäten bereit, darunter Öffnungszeiten, Bewertungen und Auslastungsinformationen. Es wird eine Reihe weiterer Features zur Verfügung gestellt, wie z.B. 3D-Kartierung, Street View mit 360° Kartenansichten. Nutzer:innen können Listen von Orten erstellen und organisieren und auf eine persönliche Timeline der besuchten Orte zugreifen (*About – Google Maps, o. D.*).

GM bietet die Möglichkeit für Nutzer:innen Einstellungen über diverse Datenschutzmechanismen zu treffen, die in den Google-Konto-Einstellungen verwaltet und eingesehen werden können. So gibt es z.B. den Inkognito-Modus, der Suchanfragen und Navigationswege nicht im Nutzerkonto speichert. Die Standortverlaufsfunktion ist standardmäßig deaktiviert und kann optional aktiviert werden, um Echtzeit-Verkehrsprognosen zu erhalten (*Zusätzliche Nutzungsbedingungen für Google Maps/Earth – Google, o. D.*).

GM arbeitet eng mit anderen Google-Diensten zusammen. Die Plattform kann beispielsweise auf Informationen aus dem Google Kalender zugreifen, um Navigationsvorschläge zu passenden Zeiten vorzuschlagen. Außerdem integriert GM seit 2025 das KI-Sprachmodell Gemini. Die gesammelten Daten werden geräteübergreifend synchronisiert und stehen verschiedenen Google-Diensten zur Verfügung, sofern Nutzer:innen der entsprechenden Datenverarbeitung zugestimmt haben (*Google-Produkte Parallel Verwenden - Computer - Google Kalender-Hilfe*, o. D.).

GM ist über eine reine Kartenapplikation hinaus zu einem zentralen Werkzeug für die Erfassung, Verarbeitung und Visualisierung geografischer Daten entwickelt. Die dabei entstehenden Aktivitätsdaten bilden die Grundlage für das in dieser Arbeit entwickelte Analysetool.

2.2 Google Takeout

Im Jahr 2018 trat eine Reform der Datenschutzgrundverordnung in Kraft. Artikel 20 Absatz 1 dieser Verordnung räumt den Nutzer:innen das Recht ein, ihre personenbezogenen Daten „in einem strukturierten, gängigen und maschinenlesbaren Format zu erhalten“. Google stellt den kostenlosen Dienst „Google Takeout“ bereit, der es Nutzer:innen ermöglicht, die im Zusammenhang mit dem eigenen Google-Konto gespeicherten Daten herunterzuladen. Mithilfe dieses Tools können Daten aus mehr als 70 verschiedenen Google-Diensten angefragt werden, darunter auch GM *(Fair, 2020).* Voraussetzung hierfür ist ein Google-Konto. Nach Anmeldung bei Google Takeout wird der Nutzende dazu aufgefordert, die gewünschten Daten einzelner Google-Dienste auszuwählen, welche zum Export bereitgestellt werden sollen. Diese werden anschließend von Google zusammengestellt und in einer Archivdatei an den Anfragenden verschickt (*So Laden Sie Ihre Google-Daten Herunter - Google-Konto-Hilfe, o. D.*).

Für die vorliegende Arbeit sind die von Takeout unter „Meine Aktivitäten“ bereitgestellten Daten relevant. Nach Auswahl des Dienstes „Maps“ können die Übermittlungsmethode, das Exportformat und die Archivgröße festgelegt werden. Hat man den Export angefordert, wird ein Hinweis angezeigt, der über die Anzahl der exportierenden Dateien informiert und darauf aufmerksam macht, dass der Exportvorgang mehrere Stunden oder Tage in Anspruch nehmen kann. Nach Abschluss des Prozesses werden die Daten an den Anfragenden mittels der vorher ausgewählten Übermittlungsmethode in Form einer E-Mail versendet (*So Laden Sie Ihre Google-Daten Herunter - Google-Konto-Hilfe, o. D.*).

2.3 Google Maps-Aktivitätsdaten

2.3.1 Inhalt

Die GM-Aktivitätsdaten aus Google Takeout dokumentieren sämtliche Nutzerinteraktionen mit dem Kartendienst in chronologischer Reihenfolge.

Die Daten werden als HTML-Datei bereitgestellt, deren Beschreibungstexte in der vom Nutzer im Google-Konto ausgewählten Sprache vorliegen. Die Einträge umfassen manuell eingegebene Suchanfragen und systemgenerierte Einträge über Interaktionen mit dem Dienst (*Google Maps-Aktivitäten Verwalten - Computer - Google Maps-Hilfe, o. D.*).

Aktivitätseinträge können folgende Informationen beinhalten (vgl. Abb. 1):

1. **Interaktionsbeschreibung:** Eine Beschreibung der Nutzerinteraktion mit dem Kartendienst (z.B. „Gesucht nach:“, „Route nach“).

2. **Suchanfrage:** Ein konkreter, vom Nutzer eingegebener Suchbegriff. Z.B. eine Adresse, ein Ortsname („Stadtbäckerei“) oder eine kategoriale Suche („Bäckerei“).
3. **Google-Maps-Link:** Einen Hyperlink zum Suchergebnis. Dieser enthält darüber hinaus GPS-Koordinaten als URL-Parameter des Suchergebnisses.
4. **Zeitstempel:** Datum, Uhrzeit und Zeitzone der Aktivität.
5. **Nutzungsstandort:** Ein GM-Link zum Standort, an dem der Dienst verwendet wurde, sofern Google diesen erfasst hat.
6. **Rechtliche Grundlage:** Ein Verweis auf die Datenschutzeinstellungen des Google-Kontos mit Angabe der rechtlichen Grundlage für die Datenspeicherung.

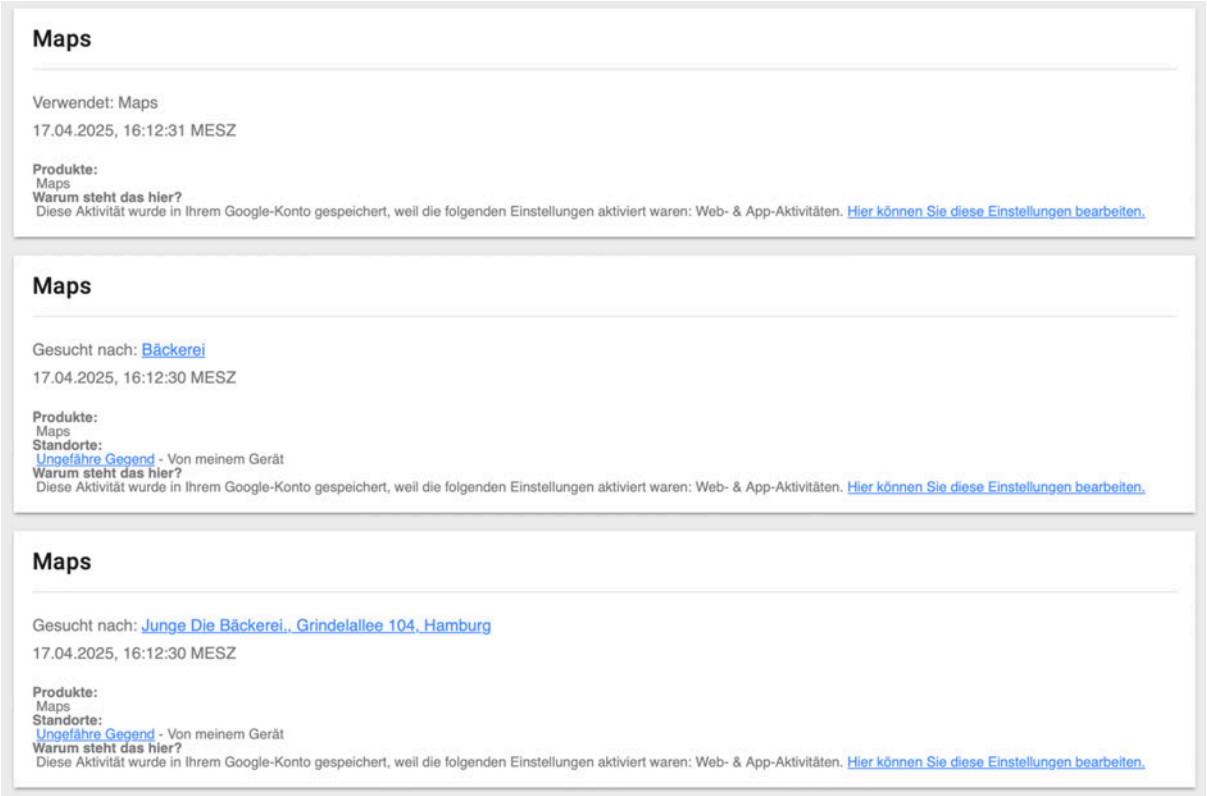

Abbildung 1: Auszug von drei Einträgen aus den GM-Aktivitätsdaten eines anonymen Probanden (Quelle: Eigene Abbildung)

2.3.2 Dateiformat

Google bietet die Daten in zwei Formaten an: als HTML-Datei und als JSON-Datei. *(Google Datenexport, o. D.)* Das JSON-Format bietet strukturierte Daten in einem verschachtelten JSON-Objekt, das sich mit Standard-JSON-Parsern verarbeiten lässt und für die automatisierte Verarbeitung geeignet ist. Das HTML-Format ist primär für die visuelle Darstellung im Browser konzipiert. Die Daten sind in verschachtelten HTML-Tags eingebettet, was die automatisierte Verarbeitung erheblich erschwert.

Für diese Arbeit wurde das HTML-Format als Datengrundlage verwendet, da dieses Format beim Projektstart als einzige Option bekannt war. Die Existenz des JSON-Formats wurde erst im späteren Projektverlauf festgestellt. Aufgrund des fortgeschrittenen Entwicklungsstands wurde entschieden, dieses beizubehalten.

Anders als Dateiformate, die den primären Zweck von Datenaustausch und Speicherung haben, müssen HTML-Dateien zunächst geparsed werden, bevor die Daten strukturiert weiterverarbeitet werden können. Die strukturierten Daten müssen aus einzelnen HTML-Elementen extrahiert und in ein maschinenlesbares Format umgewandelt werden *(Lotfi et al., 2021).* Wie dies technisch umgesetzt wird, wird im Kapitel 3.2.2 unter „Parsing-Strategie“ dargestellt.

2.3.3 Heterogenität der Suchanfragen

Eine Analyse der Beispieldatensätze zeigt, dass die in den Aktivitätsdaten erfassten Suchanfragen weder syntaktisch noch semantisch einheitlich strukturiert sind. Dies resultiert daraus, dass die Nutzer:innen die Daten manuell eingeben. Diese Eingaben müssen keinem festen Schema folgen.

Auf syntaktischer Ebene zeigt sich dadurch eine Heterogenität auf mehreren Ebenen. Identische Adressen erscheinen in unterschiedlichen Schreibweisen. Dies zeigt sich beispielsweise bei Eingaben wie „Alsterdorfer Str.“, „Alsterdorfer Straße“ oder „Alsterdorferstr“. Die Ergebnisse der Eingaben verweisen zwar auf denselben Ort, doch werden in den Aktivitätsdaten nicht die Suchergebnisse gespeichert, sondern die originalen Nutzereingaben. Aus demselben Grund weisen Adressen stark variierende Vollständigkeiten und Strukturen auf. GM findet Orte auch mit fragmentierten Adresseingaben, ohne dass eine vollständige Adresse notwendig ist. Dazu dient Googles „Place Autocomplete-Dienst“, der mehrdeutige oder unvollständige Adressen geocodiert (*Best Practices für das Geocoding von Adressen*, *o. D.*). So folgen viele der nutzerseitigen Adresseingaben keinem einheitlichen Schema sowohl in der Vollständigkeit einzelner Suchanfragen als auch in der strukturellen Reihenfolge einzelner Adresskomponenten. Auch Rechtschreibfehler sind nicht auszuschließen.

Auf semantischer Ebene beschreiben Nutzer:innen denselben Ort auf unterschiedliche Weise. So kann z.B. der Musikclub „Molotow“ in Hamburg auch ohne Adresse über seinen Namen gefunden werden. Darüber hinaus erlaubt GM auch nach Kategorien zu suchen *(About – Google Maps, o. D.*): So ist der Musikclub auch mit der Eingabe „Musikclub in Hamburg“ auffindbar. Folgende Sucheingaben führen zum identischen Suchergebnis

(„Molotow Musikclub, Reeperbahn 136, 20359 Hamburg“), variieren jedoch drastisch in ihrer syntaktischen und semantischen Struktur:

- **Ortsnamen-fokussiert:** „Molotow Hamburg“, „Molotow Reeperbahn“, „Molotow Musikclub“, „Hamburg Molotow“
- **Adress-fokussiert:** „Reeperbahn 136“, „Reeperbahn 136 Hamburg“, „Reeperbahn 136, 20359 Hamburg“, „Hamburg Reeperbahn 136“
- **Kombinierte Varianten:** „Molotow Reeperbahn 136“, „Reeperbahn 136 Molotow“, „Molotow Musikclub, Reeperbahn 136, 20359 Hamburg“
- **Kategoriale Suchen:** „Musikclubs in der Nähe“, „Musikclubs“, „Diskothek“
- **Koordinaten-basiert:** „53.54, 9.95“

Trotz dieser Heterogenität folgen viele Anfragen jedoch durch die Autovervollständigung von GM dem Schema „Ortsname/Beschreibung, Straße Hausnummer, Postleitzahl Stadt“.

Diese Heterogenität stellt eine zentrale Herausforderung für die automatisierte Datenverarbeitung dar. Ohne Normalisierung würden identische Orte als separate Entitäten behandelt; dies würde die Analyse verfälschen und die Aggregationen ungenau machen. Die methodische Konzeption der Normalisierungsstrategie wird in Kapitel 3.2.3 dargelegt.

2.4 Erhebung von Validierungsdaten

2.4.1 Rekrutierung und Einwilligung

Das entwickelte Analysetool soll mit realen GM-Aktivitätsdaten erprobt und validiert werden. Dazu wurden Daten von vier Proband:innen unter den Grundsätzen der Datenschutzgrundverordnung gesammelt. Die Teilnehmer:innen wurden aus dem persönlichen Umfeld der Autorin rekrutiert und teilten freiwillig persönliche Datensätze. Vor der Datenübermittlung wurden alle Proband:innen umfassend über den Zweck der Datenerhebung und über das Anonymisierungsverfahren informiert. Gemäß Artikel 6 Absatz 1 lit. a DSGVO willigten alle Teilnehmer:innen schriftlich ein, ihre GM-Aktivitätsdaten für wissenschaftliche Zwecke zu verwenden. Die Einwilligungserklärung stellt dabei transparent dar, dass die Daten ausschließlich zur Entwicklung und zum Test des Analysetools verwendet und nach Abschluss der Arbeit vollständig gelöscht werden. Die verwendete Mustervorlage der Einwilligungserklärung ist in Anhang 1 dokumentiert.

Die Proband:innen erhielten eine Anleitung zum Export ihrer GM-Aktivitätsdaten über den Google Takeout-Dienst. Dabei wurde ausschließlich die Kategorie „Meine Aktivitäten“ der GM-Daten exportiert.

2.4.2 Anonymisierung und Datenschutz

Um die Privatsphäre der Teilnehmenden zu schützen und den Datenschutz zu gewährleisten, wurden personenbezogene Daten anonymisiert, bevor sie verwendet und analysiert wurden. Ein Datensatz enthält keine direkten Identifikatoren wie Namen, E-Mail-Adressen oder Geräte-Kennungen. Jedem Datensatz wurde ein Pseudonym in Form von „Datensatz_00“ zugewiesen, um eine Zuordnung zwischen den verschiedenen Analysen zu ermöglichen, ohne dabei auf reale Personen rückschließen zu können.

Es ist jedoch anzumerken, dass eine vollständige Anonymisierung der Daten trotz dieser Maßnahme limitiert ist. Da die Teilnehmenden aus dem persönlichen Umfeld der Autorin stammen und häufig gesuchte Orte auf GM wie Wohnadressen oder Arbeitsstätten erfasst werden, ist eine Re-Identifikation denkbar. Dies unterstreicht die Sensibilität der Daten. Auch wenn es sich nicht um konkrete Standortdaten der Proband:innen handelt, lassen die Aktivitäten auf GM Rückschlüsse auf den Wohnort und häufig frequentierte Aufenthaltsorte zu. Selbst über formal anonymisierte Daten können Nutzer:innen mit hoher Wahrscheinlichkeit identifiziert werden.

Um das Risiko einer Re-Identifikation durch Dritte zu minimieren, werden die Daten ausschließlich lokal auf einem System ohne Cloud-Anbindung gespeichert. Die Daten werden nicht an Dritte weitergegeben und die Analyseergebnisse ausschließlich intern zu Forschungszwecken geteilt. Nach Abschluss der Arbeit werden die Daten vollständig und unwiderruflich gemäß der Einwilligungserklärung gelöscht. Die Autorin verwendet ausschließlich anonymisierte Visualisierungen, die keine Rückschlüsse auf individuelle Teilnehmer:innen zulassen.

2.4.3 Beschreibung des Datensatzes

Die vier Datensätze umfassen unterschiedliche Nutzungsintensitäten und Zeiträume, was eine robuste Validierung des Analysetools unter verschiedenen Bedingungen ermöglicht. Zwei Proband:innen weisen eine hohe Nutzungsfrequenz mit etwa 25.000 Einträgen auf, während andere Proband:innen eine geringere Frequenz mit 12.000 Einträgen zeigen (vgl. Tabelle 1). Im Durchschnitt decken die vier Datensätze 8 Jahre ab und bieten pro Person etwa 20.000 Einträge.

Die Sprache der Beschreibungen in den Daten des HTML-Dokuments ist abhängig von der im Google-Konto eingestellten Sprache des Nutzenden. So wurden drei der vier Datensätze in Deutsch und ein Datensatz in Englisch zur Verfügung gestellt. Zudem sind alle Proband:innen in Deutschland ansässig, sodass sich der geographische Raum der Suchanfragen und Aktivitäten auf GM zum großen Teil auf Deutschland und Zentraleuropa

beschränkt. Dies wird im Kapitel 3.2.3 zur Normalisierungsstrategie weiterhin relevant sein, da der geografische Schwerpunkt der Daten auf dem deutschen Adressraum liegt, wodurch deutsche Adresskonventionen (Postleitzahlsystem, typische Straßenbezeichnungen) dominieren. Suchanfragen außerhalb von Deutschland folgen möglicherweise abweichenden Strukturen, sind jedoch in der vorliegenden Datenmenge unterrepräsentiert.

Tabelle 1: Überblick über den Zeitraum, Anzahl von Einträgen und Sprache der GM-Aktivitätsdaten einzelner Proband:innen
(Quelle: Eigene Darstellung)

Datensatz	**Zeitraum** *(JJJJ-MM)*	**Anzahl der Einträge**	**Sprache**
01	2022-05 bis 2025-05	25.380	Deutsch
02	2011-06 bis 2025-04	26.878	Englisch
03	2013-03 bis 2025-05	18.418	Deutsch
04	2021-01 bis 2025-05	12.396	Deutsch

3 Konzeption

Die Konzeption des Analysetools basiert auf der Zielsetzung aus Kapitel 1.2 und den in Kapitel 2.3.2 identifizierten Herausforderungen der Datenstruktur und der im Kapitel 2.3.3 behandelten Heterogenität der Suchanfragen. Dieses Kapitel beschreibt die konzeptionellen Entscheidungen für Datenverarbeitung, Analysemethoden und Technologieauswahl. Die technische Implementierung wird im Kapitel 5 beschrieben.

3.1 Anforderungsanalyse

Die Anforderungen an das Analysetool ergeben sich aus der Forschungsfrage und den technischen Herausforderungen der GM-Aktivitätsdaten. Sie gliedern sich in funktionale und nicht-funktionale Anforderungen.

3.1.1 Funktionale Anforderungen

Datenverarbeitung

1. Die Aktivitätsdaten müssen in eine strukturierte, maschinenlesbare Form überführt werden. Dies umfasst die Extraktion von Interaktionstyp, Suchanfrage, GPS-Koordinaten und Zeitstempel aus der HTML-Struktur.

2. Die heterogenen Suchanfragen (vgl. Kapitel 2.3.3) erfordern eine Normalisierung, die syntaktische Variationen vereinheitlicht und Adresskomponenten extrahiert. Da die Suchanfragen keine standardisierte Struktur aufweisen, muss die Normalisierung mit verschiedenen Eingabeformaten umgehen können.
3. Für eine thematische Auswertung ist eine semantische Anreicherung durch Ortskategorisierung notwendig. Suchanfragen sollen mit Ortstypen versehen werden, um Aussagen über thematische Nutzungsschwerpunkte treffen zu können.

Analysefunktionen

1. Häufigkeitsanalyse: Identifikation der meistgesuchten Orte durch Aggregation über normalisierte Suchanfragen. Verschiedene syntaktische Varianten desselben Ortes müssen als identisch erkannt und zusammengefasst werden.
2. Kategorische Analyse: Thematische Gruppierung von Orten nach Kategorien wie Kultur, Bildung oder Gastronomie. Dies ermöglicht Aussagen über dominierende Interessensbereiche im Nutzungsverhalten.
3. Geografische Analyse: Visualisierung räumlicher Schwerpunkte der Aktivitäten. Nutzer:innen sollen geografische Lebensmittelpunkte visuell erfassen können.
4. Nutzungsmusteranalyse: Erkennung zeitlicher Nutzungsmuster durch Auswertung von Wochentagen und Uhrzeiten. Dies ermöglicht es Routien, wie z.B. Pendelzeiten zu identifizieren.

Visualisierung

Die Analyseergebnisse müssen in verständlicher interaktiver Form präsentiert werden. Dies umfasst:

1. Eine Heatmap für geografische Schwerpunkte
2. Interaktive Karte mit Markern für thematisch kategorisierte Orte
3. Statistische Diagramme für zeitliche Nutzungsmuster
4. Tabellarische Übersichten für die häufigsten Suchanfragen
5. Strukturiertes User-Interface mit Fortschrittsanzeigen während der Verarbeitung

3.1.2 Nicht-funktionale Anforderungen

Datenschutz

Im Gegensatz zu kommerziellen Diensten wie Google, die umfangreiche Nutzerdaten persistent speichern und für verschiedene Zwecke verwenden *(Schmidt, 2018)*, verfolgt das entwickelte Tool einen datenschutzorientierten Ansatz. Gemäß dem Prinzip „Data Protection by Design“ (DSGVO, Art. 25) wird der Schutz von Nutzerdaten sichergestellt, einhergehend

mit einer minimalen Erhebung und Verarbeitung.

1. Lokale und flüchtige Datenverarbeitung: Die Daten werden ausschließlich lokal auf dem Gerät der Nutzer:innen verarbeitet. Datenübertragung zu externen Servern oder Cloud-Services werden gemieden, um eine vollständige Kontrolle über sensible Standort- und Verhaltensdaten zu behalten. Die Daten werden im flüchtigen Arbeitsspeicher (RAM) verarbeitet. Caches, Temp-Dateien oder persistente Log-Dateien werden nicht genutzt.
2. Transparenz: Nutzer:innen werden transparent über die Verarbeitung und Löschung ihrer Daten informiert.

Performance

Die Verarbeitungszeit soll für Datensätze mit 10.000 bis 30.000 Einträgen im akzeptablen Bereich liegen. Da die GM-Aktivitätsdaten nur einmalig beim Import verarbeitet werden, können moderate Wartezeiten über etwa drei bis fünf Minuten toleriert werden, sofern Nutzer:innen über den Fortschritt und die Wartezeit informiert werden.

Usability

Das Tool muss sich ohne technische Vorkenntnisse bedienen lassen. Visualisierungen müssen auch für Laien verständlich bleiben. Fehlermeldungen sollen eine verständliche Formulierung aufweisen.

3.2 Lösungsansatz

3.2.1 Datenverarbeitungspipeline

Die Datenverarbeitung wird von einer mehrstufigen Pipeline orchestriert, die jeden Verarbeitungsschritt sequentiell durchläuft:

1. Extraktion: Parsing der HTML-Datei und Extraktion relevanter Attribute.
2. Normalisierung: Syntaktische Vereinheitlichung der Suchanfragen und Extraktion von Adresskomponenten.
3. Semantische Anreicherung: Erweiterung der Daten durch Ortskategorisierungen mit Hilfe der Google Geocoding API.
4. Analyse: Häufigkeitsanalyse, kategorische Gruppierung, geografische und zeitliche Musteranalyse.
5. Präsentation: Visualisierung der Ergebnisse in interaktiven Karten und Diagrammen.

Die Datenverarbeitungspipeline orchestriert den gesamten Prozess von der Datenextraktion bis zur Analyse und bildet die zentrale Schnittstelle zwischen Back- und Frontend. Alle Verarbeitungsschritte werden sequentiell koordiniert, ohne selbst die Datenverarbeitung durchzuführen. Für jahresübergreifende Vergleiche im Nutzerverhalten werden die Daten in jährliche Segmente unterteilt. Dies ermöglicht den Vergleich von Suchmustern zwischen verschiedenen Jahren.

Fehler in einzelnen Analyseschritten sollen durch Partial Results nicht zum Systemabsturz führen. Verfügbare Ergebnisse werden dargestellt und fehlende Ergebnisse entsprechend gekennzeichnet.

3.2.2 Parsing-Strategie

Die Extraktion strukturierter Daten aus HTML-Dokumenten werden mittels Parsing extrahiert. Für das HTML-Parsing wurde Beautiful Soup gewählt. Im Gegensatz zu vollständigen Web-Scarping-Frameworks wie z.B. Scrapy, die integrierte Fehlerbehandlung und asynchrone Verarbeitung bietet *(Scrapy Developers, 2025)*, genügt für statische HTML-Dateien eine Parsing-Bibliothek. Beautiful Soup bietet den notwendigen Funktionsumfang, der sich für einfache Parsing-Aufgaben eignet. Eine Integration der Bibliothek in die Python-Umgebung ist problemlos möglich, da der in der Standardbibliothek von Python enthaltene HTML-Parser von Beautiful Soup unterstützt wird *(Richardson, o. D.)*.

Für jeden Aktivitätseintrag in der HTML-Datei werden folgende Attribute extrahiert:

- Interaktionstyp (z.B. „Searched for“ bzw. „Gesucht nach:“)
- Suchanfrage
- GPS-Koordinaten
- Zeitstempel (Datum, Uhrzeit)
- Wochentag

Die Daten werden in tabellarischer Struktur gespeichert, wobei fehlende Werte explizit gekennzeichnet werden. Dies ermöglicht die weitere Verarbeitung unvollständiger Einträge.

Würde das JSON-Format verwendet werden, entfällt der HTML-Parsing-Schritt. In diesem Fall würden die JSON-Dateien mittels des in Python integrierten JSON-Moduls deserialisiert und die strukturieren Datenelemente direkt in tabellarischer Struktur überführt werden.

3.2.3 Normalisierungsstrategie

Aufgrund der Heterogenität der Suchanfragen (vgl. Kapitel 2.3.3), ist eine Normalisierung erforderlich. Diese erfolgt in zwei Schritten. Zunächst werden die Suchanfragen syntaktisch

vereinheitlicht, indem sie in Kleinschreibung konvertiert, überflüssige Leerzeichen entfernt und Abkürzungen standardisiert werden („Str.“ zur „straße“). Anschließend werden die Suchanfragen in einzelne Adresskomponenten (Straße, Hausnummer, PLZ, Stadt, Ortsbeschreibung) mittels eines Mustererkennungsverfahren zerlegt. Die Klassifizierung „Ortsbeschreibung“ soll, sofern keine nähere Beschreibung vorhanden ist, die Straßenadresse übernehmen. Dies ermöglicht es, in der Häufigkeitsanalyse zu zählen, wie oft bestimmte Orte oder Straßen gesucht wurden.

Die Extraktion erfolgt mit Hilfe regulärer Ausdrücke (RegEx). Adress-Parser wie `ez-address-parser` oder `usaddress` überführen Adresskomponenten in eine einheitliche Form. Diese Parser sind jedoch auf US-Amerikanische und Kanadische Adressstrukturen spezialisiert und erzielen bei deutschen Adressen nicht die gewünschte Genauigkeit *(Datamade, o. D., Zeng, o. D.)*. Im Rahmen der Recherche konnten keine vergleichbaren deutschsprachigen Alternativen gefunden werden. Es ist anzumerken, dass reguläre Ausdrücke ausschließlich eine syntaktische, aber keine semantische Normalisierung ermöglichen *(Chen et al., 2023)*.

Die in dieser Arbeit verwendeten RegEx-Patterns sind auf deutsche Adresstrukturen optimiert und berücksichtigen internationale Adressformate nur eingeschränkt, da das Tool sich primär auf den deutschen Adressraum konzentriert. Unerwartete Adressformate werden möglicherweise nicht korrekt extrahiert und liefern fehlende Werten oder ordnen Adresskomponenten falsch zu. Die Auswirkungen dieser Limitation auf die Analyseergebnisse werden in Kapitel 7.1 diskutiert.

3.2.4 Anreicherungsstrategie

Um Suchanfragen thematisch auswerten zu können, werden ihnen Ortstypen zugeordnet. Dieses sind von GM vergebene Kategorien, die Eigenschaften eines Ortes beschreiben. Dabei verwendet Google zwei Kategorisierungssysteme: Das erste umfasst thematische Kategorien wie Kultur, Bildung oder Unterhaltung und Freizeit, die beispielsweise die Tags „museum" oder „art_gallery" unter Kultur besitzen. Das zweite beschreibt administrative und funktionale Kategorien (z.B. „street_address", „point_of_interest", „establishment"). Ein Ort kann mehreren Kategorien angehören: So hat ein Restaurant beispielsweise folgende Ortstypen: „seafood_restaurant“, „restaurant“, „food“, „point_of_interest“, „establishment“ *(Google For Developers, o. D.).*

Die Kategorisierung erfolgt über die Google Geocoding API. Da die Aktivitätsdaten ebenfalls von Google stammen, wurde ein Google-Dienst als API gewählt, um die Datenkonsistenz zu gewährleisten. Alternative Geocoding-Dienste wie spaCy oder Nominatim verwenden

andere Datenquellen und Kategorisierungsschemata, was zu Inkonsistenzen bei der Zuordnung führen könnte. Studien zeigen, dass verschiedene Geocoding-Dienste unterschiedliche Genauigkeiten und Schwächen aufweisen. GM ist präziser, während Nominatim aufgrund seiner crowdsourced-Datenbasis Schwierigkeiten mit der Fehlertoleranz hat *(Serere et al., 2023)*.

Die API ist in der Nutzung beschränkt mit einem kostenlosen Kontingent von 10.000 Anfragen pro Monat *(Preisliste für die Hauptdienste der Google Maps Platform, o. D.)*. Es wurde beobachtet, dass die API-Anfragen als reine Adresssuche interpretiert werden. Dies führt zu unterschiedlichen Ergebnissen, abhängig davon, wie die Suchanfrage formuliert ist. Anfragen mit expliziten Ortsnamen liefern präzise Ortstypen. Anfragen mit reinen Straßenadressen ohne Ortsnamen werden hingegen als „street_address“ klassifiziert, da die API ohne Ortsnamen nicht auf POI-Metadaten (Points of Interest) zugreift. Dies bedeutet, dass die Anreicherung durch Ortstypen von der Formulierung der Benutzereingabe abhängig ist. Dies muss bei der Interpretation der Analyseergebnisse berücksichtigt werden.

3.2.5 User Interface

Das User Interface wird mit Streamlit realisiert. Die Bibliothek ermöglicht es, interaktive Applikationen schnell zu entwickeln ohne umfangreiche Frontend-Kenntnisse; sie ist speziell für Data-Science-Projekte konzipiert. Eine deklarative Syntax reduziert den Implementierungsaufwand, was es ermöglicht, sich auf die Analyselogik zu fokussieren. Darüber hinaus unterstützt Streamlit nativ Pandas DataFrames und bietet eigene Visualisierungsbibliotheken *(Streamlit, A Faster Way To Build And Share Data Apps, o. D.)*.

Vollständige Web-Frameworks würden zwar eine maximale Gestaltungsfreiheit bieten, setzen jedoch umfangreiche Webentwicklungskenntnisse und einen höheren Entwicklungsaufwand voraus, die über den Fokus dieser Arbeit hinausgehen.

3.3 Analysemethoden

Die Analysemethoden transformieren die Daten in interpretierbare Erkenntnisse über Nutzungsverhalten und geografische Schwerpunkte.

3.3.1 Häufigkeitsanalyse

Zur Identifikation der meistgesuchten Orte werden Suchanfragen der Spalte „Ortsbeschreibung“ aggregiert. Wie in Kapitel 3.2.3 zur Normalisierungsstrategie beschrieben, enthält diese Spalte den Ortsnamen, sofern dieser in der Suchanfrage

vorhanden ist. Falls kein Ortsname extrahiert werden kann, wird die Straßenadresse verwendet.

Dieses Schema ermöglicht es, verschiedene Suchanfragen-Varianten zu aggregieren:

- Anfragen mit Ortsnamen: „Molotow Music Club, Reeperbahn 136“ → „molotow music club“
- Anfragen nur mit Adresse: „Reeperbahn 136, Hamburg“ → „reeperbahn 136“
- Anfragen nur mit Ortsname: „Molotow Music Club“ → „molotow music club“

Systemgenerierte Einträge („Verwendet: Maps“) und Suchanfragen wie „Restaurants in der Nähe“ werden ausgeschlossen, da diese keine ortsspezifischen Informationen liefern. Für jede Gruppe wird die Häufigkeit ermittelt, und es werden repräsentative Attribute bestimmt: Die häufigste Suchanfrage dient als vollständige Adressangabe, die häufigsten GPS-Koordinaten als Referenzpunkt. Die Verwendung des häufigsten Wertes (statistisch: Modus) ist methodisch begründet, da es sich um kategoriale Daten handelt, für die ein Mittelwert nicht definiert ist.

Die Heterogenität der Suchanfragen führt jedoch zu inkonsistenten Beschreibungen. Während einige Einträge aussagekräftige Ortsnamen enthalten, besitzen andere lediglich Straßenadressen, obwohl diese denselben Ort beschreiben. Die aggregierten Ergebnisse sind somit nur eingeschränkt vergleichbar. Die Auswirkungen dieser Limitationen werden in Kapitel 9 diskutiert.

3.3.2 Kategorische Analyse

Die kategorische Analyse soll eine thematische Auswertung der Nutzer:innen-Interessen ermöglichen, indem verschiedene Ortstypen gruppiert werden. Die von der Google Geocoding API zurückgegebenen Ortstypen (vgl. Kapitel 3.2.4) werden zu 18 übergeordneten thematischen Gruppen zusammengefasst, die Google selbst definiert (z.B. Kultur, Bildung, Gastronomie, Sport, Transport). Die Gruppe „Culture“ enthält beispielsweise die Ortstypen „art_gallery“, „museum“ und „library“.

Die meistgesuchten Orte aus der Häufigkeitsanalyse werden thematisch kategorisiert. Da die kostenlosen API-Anfragen auf 10.000 Anfragen pro Monat begrenzt sind, werden lediglich die ersten 200 Einträge mit Ortstypen versehen. Dies ermöglicht repräsentative Aussagen über Nutzungsschwerpunkte, ohne das API-Kontingent zu überschreiten. Jedoch kann dies zu einer unvollständigen Abbildung des Nutzungsverhaltens führen.

Die Ortstypen werden durch Schnittmengenbildung den Ortsgruppen zugeordnet. Für jede thematische Kategorie wird geprüft, ob mindestens einer der Ortstypen des Eintrags in den vordefinierten Listen dieser Kategorie enthalten ist. Ein Ort mit dem Typ „restaurant“ wird beispielsweise der Gruppe „Food And Drink“ zugeordnet.

Die kategorisierten Orte werden auf einer interaktiven Karte visualisiert, wobei jede Kategorie durch eigene Farben und Symbole gekennzeichnet ist. Dies ermöglicht eine visuelle Analyse der geografischen Bereiche der relevantesten Orte des Nutzers im Zusammenhang mit den Ortskategorien. Dominante Kategorien identifizieren Interessenschwerpunkte. Die geografische Visualisierung zeigt räumliche Cluster auf. Die Kartendarstellung wurde mit der Python-Bibliothek Folium umgesetzt. Folium ermöglicht die Erstellung interaktiver Webkarten mit Python, ohne dass direkte JavaScript-Kenntnisse erforderlich sind *(Folium 0.20.0 Documentation, o. D.).*

3.3.3 Geografische Analyse

Ziel der geografischen Aufarbeitung der GPS-Koordinaten ist es, räumliche Schwerpunkte der Aktivitäten zu identifizieren und zu analysieren. Alle Suchanfragen mit GPS-Koordinaten werden durch eine Heatmap visualisiert. Räumlich nahe Koordinaten werden aggregiert und Cluster hoher Suchaktivität durch farbliche Intensität dargestellt (blau = niedrige Dichte, rot = hohe Dichte) (*Heatmap — Folium 0.20.0 Documentation*, *o. D.*). Die Heatmap verdichtet die Informationen zu visuell interpretierbaren Farbverläufen. Nutzer:innen können so geografische Schwerpunkte unmittelbar erkennen. Dies ermöglicht eine explorative Analyse geographischer Schwerpunkte.

Die Karte wird aus in Kapitel 3.3.2 erwähnten Gründen mit Folium erstellt. Es werden nur Einträge mit Koordinaten visualisiert, jene, die keine besitzen, werden exkludiert. Darüber hinaus korreliert die Suchhäufigkeit nicht zwingend mit der Aufenthaltsdauer oder -häufigkeit vor Ort.

3.3.4 Zeitliche Nutzungsmusteranalyse

Die zeitlichen Muster der Suchanfragen werden durch Aggregation nach Wochentagen und Uhrzeit sichtbar. Dies ermöglicht eine Identifikation von Routinen, wie z.B. Pendelzeiten. Die Visualisierung erfolgt durch zwei komplementäre Darstellungsformen.

Mittels eines Violin-Plots können Verteilungsdichten der Uhrzeiten pro Wochentag dargestellt werden. Im Gegensatz zu einfachen Balkendiagrammen zeigt ein Violin-Plot nicht nur Durchschnittswerte, sondern die gesamte Verteilung. Die Breite der Violine bildet ab, zu welchen Uhrzeiten die meisten Aktivitäten stattfanden *(Wilke, o. D.).* Ein Line-Plot zeigt die

durchschnittliche Anzahl der Suchanfragen pro Monat über ein Jahr hinweg. Dies ermöglicht es, Spitzen in der Nutzung des Dienstes zu erkennen.

Für die technische Umsetzung wurde die Bibliothek Plotly Express verwendet. Plotly Express bietet interaktive Visualisierungen mit minimalen Code-Aufwand. Durch Hover-Effekte können Nutzer:innen Details erkunden und Ausschnitte vergrößern. Zudem ist Plotly nahtlos in Streamlit integrierbar und unterstützt responsive Darstellungen (*Plotly Express in Python*, *o. D.)*.

Die Zeitstempel repräsentieren den Suchzeitpunkt, garantieren aber keine Ankunfts- oder Aufenthaltszeit am Zielort. Die Analyse zeigt Korrelationen und keine Kausalitäten.

3.4 Technologieauswahl

Für die Analyse und Visualisierung von Nutzerdaten existieren in der Praxis verschiedene etablierte Ansätze. Eine Auswertung mehrerer praxisorientierter Fachblogs zeigt, dass insbesondere die Programmiersprachen R und Python zu den zentralen Werkzeugen der Datenanalyse zählen. R wird vorrangig für statistische Analysen und komplexe Modellierungen eingesetzt, wohingegen Python als vielseitiger und besser integrierbar in unterschiedliche Anwendungsfelder der Datenverarbeitung gilt. Aufgrund der einfachen Syntax und der hohen Verfügbarkeit spezialisierter Bibliotheken, wie Pandas, NumPy und Matplotlib, bietet Python eine flexible Grundlage für datenanalytische Projekte *(Awan, 2024; Codelabs Academy, 2024; Pratt, 2024; GeeksforGeeks, 2025)*.

Für die vorliegende Arbeit wurde Python als Programmiersprache gewählt. Die Entscheidung basiert auf der nahtlosen Integration von Bibliotheken, die alle Projektphasen abdecken:

- Beautiful Soup 4 (mit lxml-Parser): HTML-Parsing der Takeout-Daten
- Pandas: Datenverarbeitung und -analyse
- RegEx: Datennormalisierung
- requests: HTTP-Anfragen an die Google Geocoding API
- Streamlit: Entwicklung des User Interfaces
- Folium: Geografische Visualisierungen
- Plotly Express: Statistische Diagramme

Diese Kombination ermöglicht eine durchgängige Pipeline von der Datenextraktion über die Normalisierung, Analyse bis zur Visualisierung. Die Entwicklung erfolgte in PyCharm auf macOS. Die Versionskontrolle wurde über Git/GitHub realisiert.

Bei der Implementierung wird das KI-Sprachmodell Claude (Anthropic) zur Unterstützung folgender Aufgaben genutzt: Codestrukturierung nach Best Practices, Debugging und Fehlerbehandlung, Dokumentation in Form von Docstrings und die Optimierung von RegEx-Patterns. Die finale Implementierung und die konzeptionellen Entscheidungen wurden von der Autorin eigenständig durchgeführt.

4 Entwurf

4.1 Systemarchitektur

Das Analysetool basiert auf einer modularen Architektur mit klarer Trennung zwischen Datenverarbeitung (Backend) und Benutzeroberfläche (Frontend). Die Architektur folgt dem Prinzip der losen Kopplung und hohen Kohäsion. Jedes Modul kapselt einen Verarbeitungsschritt gemäß dem Single-Responsibility-Prinzip und kommuniziert über definierte Schnittstellen. Dies ermöglicht eine unabhängige Erweiterung einzelner Komponenten ohne Auswirkungen auf andere Systemteile.

Die Architektur implementiert zentrale Software-Engineering-Prinzipien:

- **Separation of Concerns:** Jede Komponente hat eine klar definierte Verantwortung *(Smith, o. D.)*. Die Datenverarbeitung ist vollständig von der Präsentationslogik getrennt.
- **Orchestrator-Pattern:** Die Datenverarbeitungspipeline koordiniert Modulaufrufe ohne selbst Verarbeitungen durchzuführen. Dies zentralisiert die Ablaufsteuerung und vereinfacht Wartung und Debugging *(Gaur, 2023)*.
- **Fehlertoleranz:** Fehler in einzelnen Modulen führen nicht zum Systemabsturz. Die Pipeline fängt Fehler ab und ermöglicht Partial Results. Das bedeutet, dass das Frontend verfügbare Ergebnisse darstellen und fehlende Analysen entsprechend kennzeichnen kann.

Darüber hinaus gliedert sich das System in drei Hauptschichten, die unabhängig voneinander funktionieren:

- **Backend-Schicht:** Enthält alle Datenverarbeitungsmodule (Parsing, Normalisierung, Location Typisierung, Analyse).
- **Orchestrierungsschicht:** Über die Pipeline werden die sequentielle Verarbeitung koordiniert und Ergebnisse in einer Datenklasse gespeichert. Sie bildet die zentrale Schnittstelle zwischen Back- und Frontend.

- **Frontend-Schicht:** Besteht aus der Streamlit-Anwendung und dem Display-Modul. Sie greift ausschließlich auf aufbereitete Daten aus der Pipeline mittels der Datenklassen-Objektes zu und führt keine Datenverarbeitungen durch.

4.2 Modulstruktur

Die einzelnen Module des Systems haben eine klar definierte Verwendung.

Das **Utility-Modul** stellt zentrale Funktionen bereit, die von anderen Modulen genutzt werden. Es folgt dem Prinzip der Separation of Concerns und vermeidet Code-Duplizierung. Es verwaltet Umgebungsvariablen und API-Schlüssel und trennt so sensible Konfigurationen vom Programmcode. Zudem wird ein einheitliches Logging für alle Module implementiert, was die Fehlersuche und Nachvollziehbarkeit erleichtert.

Das **Parsing-Modul** extrahiert die Aktivitätsdaten aus HTML-Dateien und entkoppelt die Datenquelle von der weiteren Verarbeitung. Änderungen an der Parsing-Logik bleiben lokal begrenzt und haben keine Auswirkungen auf die Funktionalität nachgelagerter Module.

Das **Normalisierungs-Modul** adressiert die beschriebenen Herausforderungen heterogener Suchanfragen. Die Kapselung der Normalisierungslogik ermöglicht eine lose Kopplung zu anderen Modulen.

Das **LocationType-Modul** fungiert als Vermittlungsschicht zwischen normalisierten Daten und der Google Geocoding API. Die API-Kommunikation ist vollständig in diesem Modul gekapselt. Es werden Anfragen gesendet, Ergebnisse validiert und die Ortstypen gespeichert.

Das **Analyse-Modul** gliedert sich in vier Submodule für spezialisierte Auswertungen:

- Häufigkeitsanalyse und Aggregation
- Thematische Kategorisierung basierend auf Locationtypes und Darstellung auf einer Karte mittels Marker
- Heatmap-Erstellung aus GPS-Koordinaten
- Zeitliche Nutzungsmusteranalyse

Die modulare Trennung ermöglicht eine flexible Erweiterung einzelner Analysemethoden ohne Auswirkungen auf andere Module. Das Frontend folgt einer klaren Trennung zwischen Präsentationslogik und Verarbeitungslogik. Die Implementierung gliedert sich in zwei Hauptkomponenten: die zentrale Orchestrierung in der Hauptanwendung (`app.py`) und das **Display-Modul,** das alle Visualisierungsfunktionen kapselt und Analyseergebnisse in

Streamlit-kompatible Visualisierungen transformiert. Die Entkopplung erlaubt eine flexible Anpassung der Darstellungsformen ohne, dass sich die Auswertungslogik ändert.

4.3 Datenfluss

Der Datenfluss folgt einer Einbahnstraßen-Logik. Nutzer:innen laden über das Streamlit-Interface die HTML-Datei mit den GM-Aktivitätsdaten hoch. Diese wird von der Pipeline entgegengenommen und durchläuft sequentiell die Module. Zurückgegeben wird ein Dictionary mit ProcessedData-Objekten, indexiert nach Jahreszahlen (vgl. Abb. 2).

Eine nähere Beschreibung des Datenflusses innerhalb der Pipeline wird dargestellt (vgl. Abb. 3). Der Inhalt der HTML-Datei wird im Parsing-Modul in einem Pandas DataFrame gespeichert. Dieses durchläuft das Normalisierungs-Modul und wird anschließend nach Jahren segmentiert, sodass ein Dictionary mit jahresbasierten DataFrames entsteht.

Für jedes Jahres-DataFrame werden nacheinander Analysen durchgeführt:

1. Häufigkeitsanalyse: Das DataFrame wird nach Orten aggregiert. Das Ergebnis ist ein nach Häufigkeit sortiertes DataFrame.
2. Kategorische Analyse: Das DataFrame aus der Häufigkeitsanalyse durchläuft das LocationType-Modul, was ein erweitertes DataFrame zurückgibt. Aus diesem wird ein Dictionary mit thematischen Gruppierungen generiert. Dieses wird verwendet, um ein Map-Objekt mit Markern zu generieren. Das Map-Objekt und die Anzahl platzierter Marker werden als Tuple zurückgegeben.
3. Geografische Analyse: Mit den Daten aus dem DataFrame wird ein Map-Objekt erzeugt.
4. Nutzungsmuster-Analyse: Das DataFrame wird in zwei separate DataFrames transformiert, die die nötigen Werte für einen Violin-Plot und einen Line-Plot enthalten.

Alle generierten Datenobjekte werden in einem ProcessedData-Objekt gebündelt. Die jahresspezifischen ProcessedData-Objekte sowie ein Objekt für die Gesamtauswertung aller Jahre werden in einem Dictionary zusammengefasst und an das Frontend übergeben.

Datenfluss

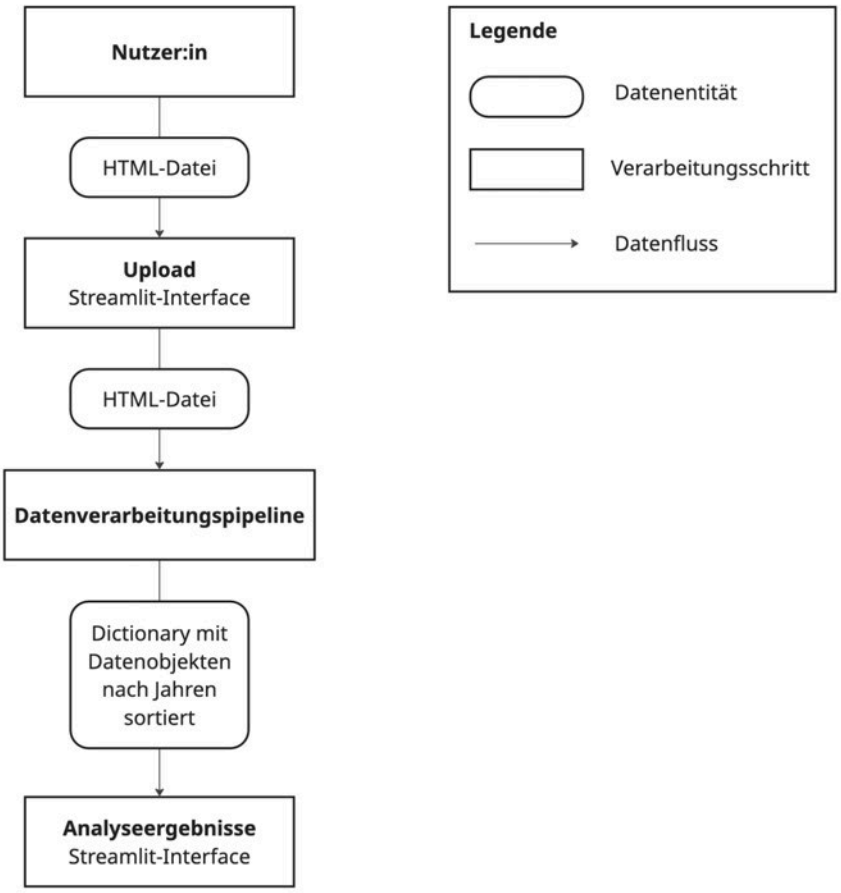

Abbildung 2: Datenflussdiagramm des Analysetools

(Quelle: Eigene Darstellung)

Datenfluss:
Pipeline

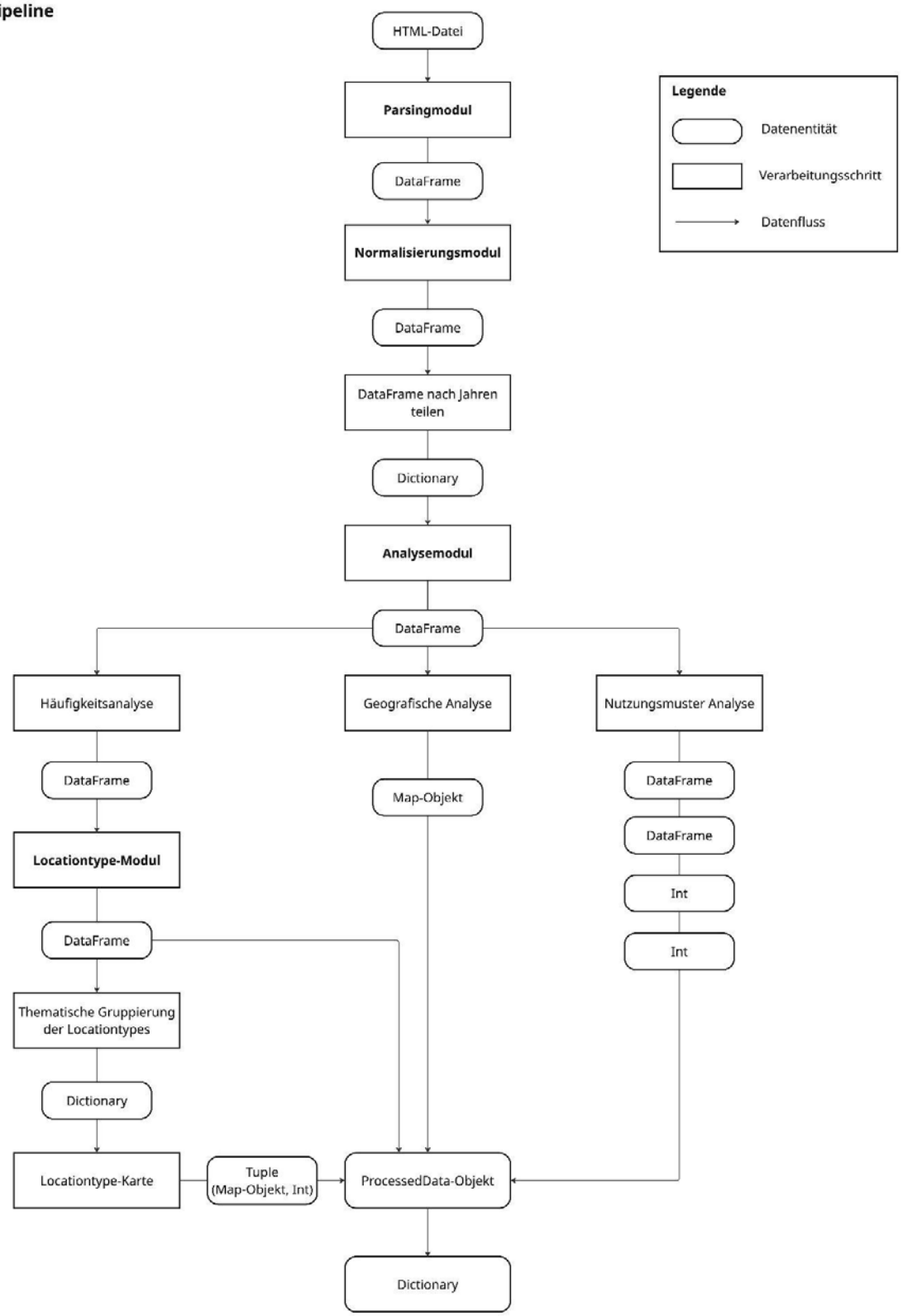

Abbildung 3: Datenflussdiagramm in der Datenverarbeitungspipeline
(Quelle: Eigene Darstellung)

5 Implementierung

5.1 Utility-Modul

Ein zentraler Aspekt des Utility-Moduls ist der sichere Umgang mit dem Google Geocoding API-Schlüssel. Dieser wird nicht direkt im Quellcode hinterlegt, sondern über Umgebungsvariablen einer `.env`-Datei verwaltet. Mithilfe der Bibliothek `python-dotenv` wird diese Datei eingelesen und stellt diesen dem LocationType-Modul zur Verfügung. So wird vermieden, dass vertrauliche Informationen in das Versionskontrollsystem GitHub gelangen.

Das Logging-System wird so konfiguriert, dass die Ausgaben in der Konsole erfolgen. Dadurch lassen sich Laufzeitinformationen und Fehler zentral erfassen. Das Logging ist in jeder einzelnen Funktion des Systems implementiert worden. Wird eine Funktion erfolgreich abgeschlossen oder schlägt sie fehl, wird dies entsprechend im Log ausgegeben.

5.2 Parsing-Modul

Input: HTML-Datei mit GM Aktivitätsdaten.
Output: Pandas DataFrame mit extrahierten Daten in der in Tabelle 2 aufgeführten Struktur.

Das Parsing-Modul implementiert die in Kapitel 3.2.2 konzipierte Parsing-Strategie. In der Funktion `parse_maps_activity` wird eine HTML-Datei mit `UTF-8`-Kodierung geöffnet und Beautiful Soup mit dem `lxml`-Parser initialisiert. Das lxml XML-Toolkit stellt Python-Bindings für die in C geschriebenen Bibliotheken `libxml2` und `libxslt` bereit. Durch diese Anbindung können die Performance-Vorteile der C-Implementierung mit einer pythonischen API kombiniert werden (*lxml - Processing XML And HTML With Python*, *o. D.*) .

Die Extraktion erfolgt durch verschachtelte Iteration über spezifische CSS-Klassen: Äußere Container namens `outer-cell` (vgl. Codeblock 1, Zeile 1) repräsentieren die einzelnen Aktivitätseinträge, innere Container namens `content-cell` (vgl. Codeblock 1, Zeile 7) enthalten die eigentlichen Metadaten. In dem inneren Container wird der gesamte Textinhalt erfasst, und es werden folgende Attribute identifiziert:

- Der **Interaktionstyp** wird durch einen Abgleich des Textes mit einer vordefinierten Liste von bekannten Aktionen in deutscher und englischer Sprache ermittelt und als String extrahiert (z.B. „Searched for“ bzw. „Gesucht nach:“).
- Anschließend wird der verbleibende Text, der nicht zur Aktionsbeschreibung gehört, als **Suchanfrage** als String übernommen.

- Die **zeitlichen Informationen** befinden sich immer in der letzten Textzeile. Diese werden durch ihre Positionierung identifiziert und mittels `dateutil.parser` in `DateTime`-Objekte konvertiert, damit sie strukturiert weiterverarbeitet werden können. Dabei wird die Angabe der Zeitzone von MESZ zu CEST normalisiert, um das europäische Datumsformat korrekt zu interpretieren.
- Die **GPS-Koordinaten** sind im Google-Maps-Link kodiert. Alle Link-Tags (`<a>`) innerhalb des Containers werden analysiert und die Koordinaten mittels eines regulären Ausdrucks aus dem URL-Parameter extrahiert. Die Wertepaare werden als String gespeichert.

Alle extrahierten Werte werden in einem Pandas DataFrame zusammengeführt, wobei fehlende Werte als `NaN` gespeichert werden. Tabelle 2 illustriert beispielhaft den Aufbau des DataFrames nach dem Parsing.

Codeblock 1: Auszug aus dem HTML-Code eines Eintrags aus den GM-Aktivitätsdaten eines anonymen Probanden mit in gelb hervorgehobenen Parametern
(Quelle: Datensatz_04)

```
<div class="outer-cell mdl-cell mdl-cell--12-col mdl-shadow--2dp">
   <div class="mdl-grid">
       <div class="header-cell mdl-cell mdl-cell--12-col">
           <p class="mdl-typography--title">Maps<br></p>
       </div>

<div class="content-cell mdl-cell mdl-cell--6-col mdl-typography--body-1">Searched for<a
href="https://www.google.de/maps/place/Butteldorf/@53.1902885,8.3785378,14z/data=!3m1!4b1!4m2!3m1!1s0x4
7b6d98a738d064b:0xaf83a4225f43acc4">Butteldorf, 26931 Elsfleth</a><br>11 Jun 2023, 16:31:05 CEST</div>
       <div class="content-cell mdl-cell mdl-cell--6-col mdl-typography--body-1
mdl-typography--text-right"></div>

       <div class="content-cell mdl-cell mdl-cell--12-col mdl-typography--caption">
           <b>Products:</b><br> Maps<br><b>Why is this here?</b><br> This activity was saved
to your Google Account because the following settings were on: Web &amp; App Activity. You
can control these settings  <a href="https://myaccount.google.com/activitycontrols">here</a>.
       </div>
   </div>
</div>
```

Tabelle 2: Beispielhafter Ausschnitt eines Pandas DataFrames nach Extraktion der Daten aus der HTML-Datei der GM-Aktivitätsdaten
(Quelle: Eigene Darstellung)

action	query	coordinates	date
Searched for	molotow music club, reeperbahn 136, 20359 hamburg	53.5772895, 10.0801031	2023-06-11 16:31:05+02:00

5.3 Normalisierungs-Modul

Input: Pandas DataFrame aus dem Parsing-Modul.

Output: Pandas DataFrame mit normalisierten Suchanfragen und zusätzlichen Spalten für einzelne Adresskomponenten.

Das Normalisierungs-Modul realisiert die im Kapitel 2.3.2 konzipierte Normalisierungsstrategie. Zunächst werden die Suchanfragen aus der Spalte `query` syntaktisch vereinheitlicht, indem sie mittels regulärer Ausdrücke in Kleinschreibung konvertiert und mehrfache Leerzeichen entfernt werden sowie Abkürzungen wie „str." zu „straße" angepasst werden. Weiterhin werden mittels spezialisierter RegEx-Muster die Anfragen in einzelne Adresskomponenten zerlegt.

Die Straßennamen werden über zwei komplementäre Patterns extrahiert, die primär deutsche Straßennamen mit unterschiedlichen Strukturen erfassen. Das erste Pattern identifiziert Adressformate mit Straßennamen und folgender Hausnummer (vgl. Codeblock 2, Zeile 1-5). Im Detail erkennt das Pattern Zeichenketten aus alphabetischen Zeichen, die optional durch Leerzeichen oder Bindestriche getrennt sind. Um als Straßenadresse erkannt zu werden, muss mindestens eine Ziffer im Anschluss der Zeichenkette stehen. Die Hausnummer erlaubt optionale Buchstaben und deckt Hausnummernbereiche ab („Karl-Marx-Straße 12a-d").

Das zweite Pattern identifiziert Straßennamen näher, in dem es typische deutsche Straßenendungen („Allee", „Chausee", „Kamp") und Straßennamen mit Präpositionen („An der Alster") erkennt (vgl. Codeblock 2, Zeile 7-15). Im Pattern sind eine Vielzahl an Präpositionen hinterlegt, wie z.B. „am", „an", „zur", „hinter", ebenso wie eine Liste typischer deutscher Straßenendungen. So wird auch eine Vielzahl an deutschen Straßennamen auch ohne Hausnummern erkannt.

Die Postleitzahlen werden über ein fünfstelliges Zahlenmuster extrahiert. Städtenamen werden durch ein kombiniertes Pattern identifiziert, das auf eine Postleitzahl folgende Zeichenketten erfasst, da Anfragen meist dem Schema „Postleitzahl Stadt" folgen.

Letztlich wird eine Ortsbeschreibung, sofern diese vorhanden ist, aus dem ersten Segment vor dem ersten Komma extrahiert (vgl. Codeblock 4), da autovervollständigte GM-Anfragen häufig dem Schema „Ortsname, Adresse" folgen. Ist keine Ortsbeschreibung vorhanden, wird mit dem Pattern die Straßenadresse extrahiert und als Beschreibung gespeichert. Dies erweist sich für die folgende Häufigkeitsanalyse als essentielle Spalte.

Alle RegEx-Patterns sind case-insensitive und berücksichtigen deutsche Sonderzeichen (ÄÖÜäöüß). Nicht-extrahierbare Komponenten werden als `None`-Werte gespeichert, um Datenintegrität zu wahren. Unerwartete oder unvollständige Adressformate führen zu fehlenden Werten oder fehlerhaften Zuordnungen. Die Ergebnisse des Moduls werden in einem erweiterten DataFrame gespeichert, das neben den bereits vorhandenen Attributen sowohl die normalisierte Suchanfrage als auch die extrahierten Adresskomponten enthält (vgl. Tabelle 3).

Codeblock 2: RegEx-Pattern zur Extraktion von deutschen Straßennamen
(Quelle: Eigene Implementierung)

```
STREET_PATTERN_1 = re.compile(
    r"[\wÄÖÜäöüß]+(?:[ -][\wÄÖÜäöüß]+)* \d+[a-zA-Z]?(?:-\d+[a-zA-Z]?)?",
    flags=re.IGNORECASE
)

STREET_PATTERN_2 = re.compile(
    r"(?:(?:am|an der|auf dem|in der|auf
der|zum|zur|im|bei|unter|hinter|vor|über|zwischen)\s+)?"
    r"[\wÄÖÜäöüß]+(?:[ -][\wÄÖÜäöüß]+)*\s?"

r"(?:straße|str|weg|allee|gasse|pfad|damm|steg|koppel|garten|chaussee|ufer|kamp|twiete|ring
|möhlen)\b",
    flags=re.IGNORECASE
)
```

Codeblock 3: RegEx-Pattern zur Extraktion von Postleitzahl und Stadt
(Quelle: Eigene Implementierung)

```
PLZ_PATTERN = re.compile(
    r"\d{5}",
    flags=re.IGNORECASE
)

CITY_PATTERN = re.compile(
    r"\d{5}\s+([A-Za-zÄÖÜäöüß\s\-]+)+",
    flags=re.IGNORECASE
)
```

Codeblock 4: RegEx-Pattern zur Extraktion der Ortsbeschreibung
(Quelle: Eigene Implementierung)

```
DESCRIPTION_PATTERN = re.compile(
    r"^[^,]+",
    flags=re.IGNORECASE
)
```

Tabelle 3: Beispielhafter Ausschnitt eines Pandas DataFrames nach der Normalisierung
(Quelle: Eigene Darstellung)

action	coordinates	date	normalized_query	street	plz	city	description
Searched for	53.5772895, 10.0801031	2023-06-11 16:31:05+02:00	molotow music club, reeperbahn 136, 20359 hamburg	reeperbahn 136	20359	hamburg	molotow music club

5.4 LocationType-Modul

Input: Pandas DataFrame nach der Häufigkeitsanalyse.

Output: Pandas DataFrame ergänzt mit einer Spalte, die Locationtypes für die ersten 200 Einträge enthält.

Das LocationType-Modul setzt die Ortskategorisierung um, indem es die normalisierten Daten mit semantischen Ortsinformationen anreichert. Es werden für die 200 häufigsten Suchanfragen Ortstypen über die Google Geocoding API abgerufen. Das Modul iteriert über die ersten 200 Einträge des DataFrames und stellt für jede Suchanfrage eine HTTP-Anfrage an die Geocoding API.

Die Suchanfrage aus der `query-Spalte` des DataFrames und der API-Schlüssel aus der `.env-Datei` werden in die URL für die HTTP-Anfrage eingebettet. Dabei werden bei der Suchanfrage Leerzeichen durch Pluszeichen ersetzt, um dem definierten URL-Schema der Geocoding API zu entsprechen (vgl. Codeblock 5).

Codeblock 5: Vorbereitung für die HTTP-Anfrage an die Geocoding API
(Quelle: Eigene Implementierung)

```
1   api_key = utils.config.get_gm_api_key()
2   address = query.replace(" ", "+")
3   url = f"https://maps.googleapis.com/maps/api/geocode/json?address={address}&key={api_key}"
```

Bei erfolgreicher Anfrage gibt die API ein JSON-Objekt mit sämtlichen Informationen zum Standort zurück. Relevant ist das Attribut `types`, das eine Liste der von Google zugewiesenen Ortstypen enthält *(Google for Developers, o. D.)*. Zugegriffen auf dieses Attribut wird über den Index: `results[0]["types"]`. Die zurückgegebene Liste von Ortstypen wird vollständig im DataFrame in einer neuen Spalte `type` übernommen. Die Sortierung innerhalb der Liste folgt keiner erkennbaren Priorisierung. So erhält beispielsweise das Fitnessstudio „FITNESSLAND Hamburg“ die Typen `["establishment", "gym", "health"]`, wobei der aussagekräftigste Typ „gym“ nicht an erster Stelle steht.

Die Spalten werden durch Vorbelegung aller DataFrame-Zeilen mit `None` initiiert, wobei die ersten 200 Einträge entsprechend überschrieben werden. Dies stellt sicher, dass alle Zeilen einen definierten Wert besitzen und nachfolgende Operationen nicht durch fehlende Werte fehlschlagen.

Die Implementierung verwendet `requests.exceptions.RequestException` für umfassendes Error Handling. Fehlerhafte Anfragen (z.B. bei Netzwerkproblemen oder ungültigen Adressen) werden protokolliert, führen jedoch nicht zum Abbruch des Gesamtprozesses. Für diese Einträge wird der Wert `None` gespeichert. Die Methode `raise_for_status()` der `Request`-Bibliothek prüft HTTP-Statuscodes und wirft bei Codes über 400 automatisch eine Exception. Um API-Rate-Limits einzuhalten, wird nach jeder erfolgreichen Anfrage eine Verzögerung von 0,2 Sekunden eingefügt.

Zur Effizienzsteigerung während der Entwicklungs- und Validierungsphase wurde ein lokaler Cache integriert. Dieser speicherte bereits angefragte URLs und deren Ergebnisse in einer JSON-Datei. Dies reduzierte API-Anfragen, um die Nutzungsbeschränkung des kostenlosen Kontingents von 10.000 Anfragen pro Monat einzuhalten (*Preisliste für die Hauptdienste der Google Maps Platform, o. D.)*. Der Cache verringerte die Anzahl notwendiger API-Anfragen erheblich und beschleunigte Test- und Validierungszyklen, da wiederholte Anfragen für dieselben Adressen direkt aus der lokalen Datei beantwortet wurden. Nach Abschluss der Implementierung wurde der Cache entfernt, um die datenschutzrechtlichen Vorgaben (vgl. Kapitel 3.1.2) einzuhalten. In der finalen Anwendung müssen personenbezogene oder ortsbezogene Daten nicht zwischengespeichert werden. Abgefragt wird nun ausschließlich direkt über die API ohne lokale Persistenz der Ergebnisse.

5.5 Analyse-Modul

5.5.1 Häufigkeitsanalyse

Input: Pandas DataFrame nach der Normalisierung.

Output: Pandas DataFrame mit nach Häufigkeit sortieren Suchanfragen.

Das Analyse-Modul aggregiert unter anderem die normalisierten Suchanfragen und ermittelt die meistgesuchten Orte. Vor der Aggregation werden systemgenerierte Einträge („angesehene Region“, „viewed area“) durch ein reguläres Ausdrucksmuster erkannt und herausgefiltert, um aussagekräftige Ergebnisse mit Adressbezug zu gewährleisten.

Während der Entwicklung wurde die Aggregation nach unterschiedlichen Spalten getestet, um herauszufinden, welche Methode die aussagekräftigsten Ergebnisse liefert. So

ermöglicht die Funktion `get_top_queries()` die Gruppierung nach verschiedenen Spalten (`street, query, description`). Die Spalte `description` erwies sich als optimal für die Häufigkeitsanalyse. Wie in Kapitel 3.2.3 zur Normalisierungsstrategie beschrieben, wird während der Normalisierung die Ortsbeschreibung in dieser Spalte gespeichert. Ist keine explizite Ortsbeschreibung vorhanden, wird die Straßenadresse als Beschreibung gespeichert. Diese Fallback-Logik stellt sicher, dass sowohl Einrichtungen mit Ortsnamen als auch Wohnadressen ohne spezifische Ortsbezeichnung erfasst werden. Bei der Gruppierung anderer Spalten des DataFrames würden entweder die Ortsbeschreibungen oder die Straßenadressen in der Auswertung verloren gehen. Die `description`-Spalte berücksichtigt sowohl Privatadressen als auch kommerzielle Orte.

Für jede Gruppierung werden drei Attribute berechnet: Die Anzahl der Vorkommen (über Pandas' `size`-Funktion) und die repräsentativste Suchanfrage mit den häufigsten Koordinaten, die mit der Suchanfrage auftreten. Der häufigste Wert wird mit der Lambda-Funktion im Codeblock 6 berechnet. Es werden drei Fälle behandelt:

- Eindeutiger Modus: Der häufigste Wert wird zurückgegeben.
- Mehrere Modi: Der erste Wert aus der Modus-Liste wird gewählt.
- Leere Gruppe: Dies sollte theoretisch nicht auftreten, wird aber durch das Fallback auf das erste Element der Gruppe abgefangen.

Nach der Aggregation wird ein Pandas Dataframe mit den Spalten „`description, count, query, coordinates`“ zurückgegeben (vgl. Tabelle 4).

Codeblock 6: Lambda-Funktion zur Modusberechnung
(Quelle: Eigene Implementierung)

```
1  lambda x: x.mode().iloc[0] if not x.mode().empty else x.iloc[0]
```

Tabelle 4: Beispielhafter Ausschnitt des Pandas DataFrames nach der Häufigkeitsanalyse
(Quelle: Eigene Darstellung)

description	count	query	coordinates
molotow music club	4	molotow music club, reeperbahn 136, 20359 hamburg	53.5772895, 10.0801031
reeperbahn 136	1	reeperbahn 136, 20359 hamburg	53.5500806, 9.9565278

5.5.2 Kategorische Analyse

Input: Pandas DataFrame nach Durchlaufen des LocationType-Moduls.

Output: Dictionary mit den Ortsgruppierungen als Schlüssel und einer Liste an zugehörigen Adressen und Koordinaten.

Die kategorische Analyse arbeitet mit dem DataFrame, welches die Häufigkeitsanalyse durchlief und anschließend für die ersten 200 Einträge API Anfragen zur Ermittlung des Ortstyps durchlief. Die Suchanfragen, die Ortstypen enthalten, sollen thematischen Gruppen zugeordnet werden. Da für jede Anfrage eine Liste von Ortstypen durch die API zurückgegeben wurde, kann ein Ort mehreren Gruppen angehören.

Die Grundlage bilden 18 von Google definierte Gruppen, denen jeweils mehrere detaillierte Ortstypen zugeordnet sind. Diese Listen wurden als statische Konstanten im Programm implementiert und bilden die Referenz für die Auswertung (vgl. Codeblock 7).

Codeblock 7: Ausschnitt der Listen zur Ortsgruppierung
(Quelle: Eigene Implementierung)

```
1    GOOGLE_PLACE_TYPES_BUSINESS = [
2        "corporate_office", "farm", "ranch"
3    ]
4
5    GOOGLE_PLACE_TYPES_CULTURE = [
6            "art_gallery", "art_studio", "auditorium", "cultural_landmark",
7            "historical_place", "museum", "performing_arts_theater"
8    ]
9
10   GOOGLE_PLACE_TYPES_EDUCATION = [
11       "library", "preschool", "primary_school", "secondary_school", "university"
12   ]
13   [...]
```

Die von der API zurückgegeben Listen mit Ortstypen werden durch Schnittmengenprüfung mit den Listen der Ortsgruppierungen nacheinander abgeglichen. Für jede Überschneidung der Ortstypenliste einer Suchanfrage mit einer Ortsgruppe wird der Ort mitsamt den Koordinaten in einem Dictionary gespeichert. Wurde einer Gruppe keine Suchanfrage zugeordnet, wird diese automatisch aus dem Dictionary entfernt. So entsteht ein hierarchisches Dictionary, das nach thematischen Gruppierungen von Orten geordnet ist und pro Gruppe sämtliche zugehörige Adressen enthält. Ein Eintrag besteht aus der ursprünglichen Suchanfrage und den Koordinaten (vgl. Codeblock 8). Diese Datenstruktur dient anschließend als Grundlage für eine geografische Visualisierung auf einer interaktiven Karte und einer statistischen Auswertung der Nutzerschwerpunkte.

Codeblock 8: Ausschnitt eines beispielhaften Dictionaries nach der Ortstypenanalyse (Quelle: Eigene Implementierung)

```
{
    "gym": {
        0: {
            "query": "FitX Oldenburg",
            "coordinates": (53.1385, 8.2146)
        }
    },
    "foodAndDrink": {
        1: {
            "query": "Edeka Center",
            "coordinates": (53.1502, 8.2234)
        }
    },
   [...]
}
```

Die geografischen Daten werden mit Hilfe der `Folium`-Bibliothek auf einer interaktiven Karte dargestellt. Für jede Ortsgruppe werden ein Farbschema und ein passendes Icon aus der Font-Awesome-Bibliothek für die Marker auf der Karte festgelegt. Als Kartenmittelpunkt dient die geografische Mitte Deutschlands. Die übergebenen Koordinaten aus dem Dictionary werden in ein Float-Tuple umgewandelt. Fehlerhafte oder fehlende Eingaben werden als `None` zurückgegeben und ungültige Strings bei der Umwandlung in ein Tuple abgefangen.

Um die Marker zu erzeugen, wird über das Dictionary iteriert; die Einträge aus jeder Gruppe zu einer eigenen `FeatureGroup` zugefügt und die definierten Styles (Farbe und Icon) übergeben. Für alle gültigen Standorte innerhalb einer Gruppe werden folgende Informationen für ein Pop-Up übergeben, welches erscheint, wenn auf den Marker geklickt wird: der Suchbegriff, die Koordinaten und die Ortsgruppierung. Die Marker werden der Karte hinzugefügt. Zusätzlich wird eine `LayerControl` erstellt, über welche gezielt die einzelnen `FeatureGroups` ein- und ausgeblendet werden können. Orte, die in mehreren Kategorien vorkommen, werden mit mehreren Markern unterschiedlicher `FeatureGroups` auf der Karte markiert (vgl. Abb. 4). Zurückgegeben wird ein Tupel mit dem Map-Objekt und der Anzahl der gesetzten Marker, die im Display-Modul weiterverarbeitet werden.

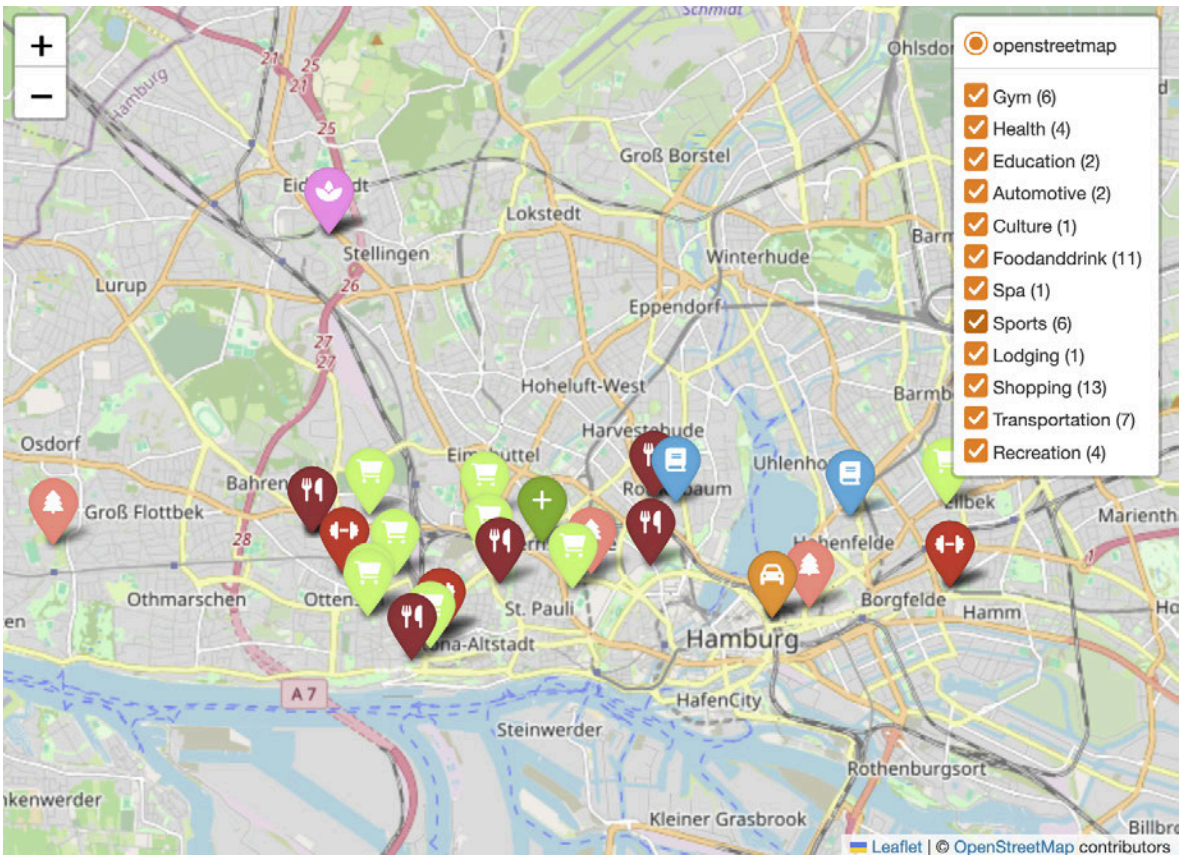

Abbildung 4: Screenshot einer mit Folium erstellten Karte mit Markern verschiedener Ortstypen (Quelle: Eigene Darstellung)

5.5.3 Nutzungsmusteranalyse

Input: Pandas DataFrame nach der Normalisierung.

Output 1: Pandas DataFrame mit der Anzahl der Aktivitäten pro Monat.

Output 2: Pandas DataFrame mit Uhrzeit und Wochentag von einzelnen Aktivitäten.

Die Nutzungsmusteranalyse wertet die zeitlichen Zusammenhänge der einzelnen GM Aktivitäten aus und bereitet die Daten für die Visualisierung vor.

Um Zusammenhänge der Wochentags-Uhrzeit-Korrelation herzustellen, werden für jeden Aktivitätseintrag der Name des Wochentags und die Uhrzeit benötigt. Die Spalte `date` wird mittels `pd.to_datetime()` in ein Pandas `DateTime`-Objekt konvertiert, wobei ungültige Datumsformate in `NaT` (Not a Time) umgewandelt werden. So wird auch eine Weiterverarbeitung bei fehlerhaften Einträgen garantiert. Aus dem `DateTime`-Objekt wird der Wochentagsname mit der Methode `.dt.day_name()` extrahiert. Für die Darstellung der Uhrzeit in einem Violin-Plot werden die Stunden als Dezimalzahl benötigt. Dazu werden aus dem `DateTime`-Objekt die Stunden und Minuten extrahiert. Die Minuten werden durch 60 dividiert und zur Stunde addiert (vgl. Codeblock 9). So werden Zeitangaben wie „15:30“ in

den Dezimalwert „15.5“ umgewandelt. Zurückgegeben wird letztlich ein DataFrame mit den Spalten `hour` und `weekday`. Diese Daten werden im Display-Modul (vgl. Kapitel 5.7.2) als Violin Plot visualisiert.

Codeblock 9: Umwandlung von Uhrzeiten in eine Dezimalzahl
(Quelle: Eigene Implementierung)

```
df_preparation["hour"] = (
        df_preparation["date"].dt.hour +
        df_preparation["date"].dt.minute / 60
)
```

Für eine monatliche Suchstatistik werden die Suchanfragen nach Kalendermonat gezählt. Nach der `DateTime`-Konvertierung wird mittels `.dt.to_period("M")` das Datum von bspw. `2024-02-24` zu `2024-02` transformiert. Die Einträge werden über Pandas' `groupby()` nach den neuen Datumsangaben aggregiert und anschließend mit der Methode `size()` gezählt. Das Ergebnis wird in einem DataFrame mit den Spalten `year_month` und `count` überführt. Mit dieser Struktur kann die monatliche Suchstatistik in einem Liniendiagramm dargestellt werden.

5.5.4 Geografische Analyse

Input: Pandas DataFrame nach der Normalisierung.
Output: Folium Map-Objekt

Die geografische Analyse erstellt eine Heatmap zur räumlichen Darstellung der einzelnen Aktivitätseinträge. Die Implementierung verarbeitet die GPS-Koordinaten aus dem DataFrame und visualisiert deren Dichte mittels des `Heatmap`-Plugins der `Folium`-Bibliothek.

Zunächst werden alle Einträge mit gültigen Koordinaten herausgefiltert, und es wird eine Kopie des gefilterten DataFrames angelegt. Falls keine Koordinaten vorhanden sind, wird die Verarbeitung abgebrochen und `None` zurückgegeben. Dadurch kann das Tool auch ohne Heatmap weiterarbeiten. Die Koordinaten liegen in der Spalte `coordinates` als String im Format (`"53,5777, 10.0801"`) vor. Sie werden mittels `str.split(",", expand=True)` in zwei Spalten aufgeteilt. Nachdem validiert wurde, dass zwei Spalten entstanden sind, werden die Koordinaten anschließend in `Float`-Werte konvertiert.

Um die Basis-Karte zu zentrieren, wird aus den Koordinaten ein Mittelwert ermittelt. Der Parameter `zoom_start=5` definiert die initiale Zoomstufe für die Karte. Der Wert fünf ermöglicht eine Ansicht von bspw. Deutschland und angrenzenden Ländern. Die Wahl der niedrigen Zoomstufe ermöglicht einen Gesamtüberblick für Aktivitätsdaten, die sich primär

auf Deutschland konzentrieren. Die Zoomstufe kann im Nachhinein von Nutzer:innen selbst angepasst werden.

Zu der Karte wird eine Heatmap-Layer hinzugefügt (vgl. Codeblock 10) . Der Parameter `data` erwartet ein NumPy-Array oder eine Liste von Koordinatenpaaren. Mit der Methode `.values` wird die Spalte `coordinates` in ein NumPy-Array konvertiert, sodass die erforderliche Struktur `[[lat1, lon1], [lat2, lon2], ...]` gegeben ist. Der Parameter `radius` definiert den Einflussbereich jedes Datenpunkts in Pixeln. Ein größerer Radius führt zu weitläufigen Heatmap-Bereichen, während ein kleinerer Radius präzise, aber möglicherweise fragmentierte Darstellungen erzeugt. Die Wahl von zehn Pixeln stellt einen Kompromiss zwischen Detailgrad und Übersichtlichkeit dar.

Codeblock 10: Erstellung der Heatmap-Layer mit dem Folium-Plugin HeatMap
(Quelle: Eigene Implementierung)

```
1   HeatMap(
2       data=coordinates[["lat", "lon"]].values,
3       radius=10
4   ).add_to(map)
```

Standardmäßig verwendet das Plugin einen Farbverlauf von blau (niedrige Dichte) über grün und gelb bis rot (hohe Dichte). Bereiche mit vielen überlagernden GPS-Daten erscheinen rot, während vereinzelte Suchanfragen blau dargestellt werden (vgl. Abb. 5). Durch die farbliche Abstufung werden verschiedene Dichten der Koordinaten sichtbar (*Heatmap — Folium 0.20.0 Documentation*, *o. D.*). Die Heatmap wird als Folium Map-Objekt zurückgegeben.

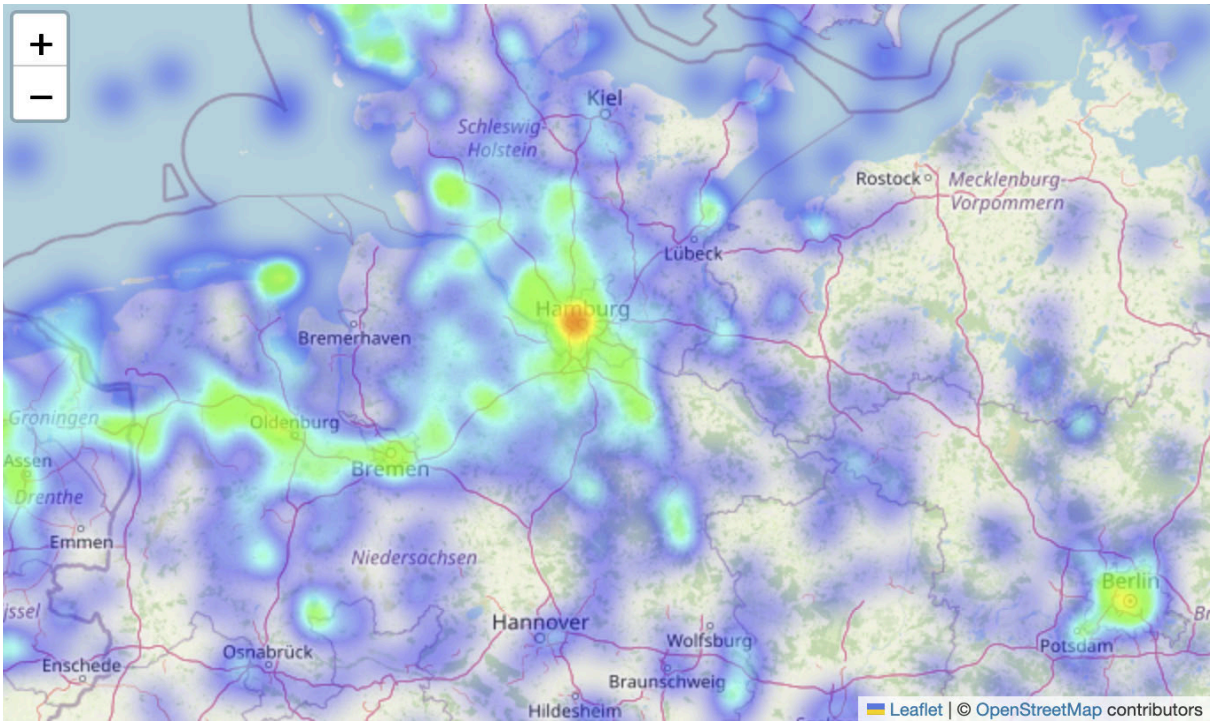

Abbildung 5: Screenshot einer mit Folium erstellten Heatmap mit den Daten eines anonymen Probanden (Quelle: Eigene Darstellung)

5.6 Pipeline

Input: HTML-Datei mit GM Aktivitätsdaten
Output: Datenobjekt mit Analyseergebnissen

Die Datenverarbeitungs-Pipeline orchestriert den gesamten Prozess von der Datenextraktion bis zur Analyse. Sie gibt ein Datenobjekt `ProcessedData` zurück, welches an das Frontend weitergegeben wird. Die Datenklasse enthält für jedes Analyseergebnis ein passendes Attribut.

Die Pipeline nimmt den temporären Pfad zu einer hochgeladenen HTML-Datei mit GM-Aktivitätsdaten entgegen. Hier wird über `Streamlit` eine Fortschrittsanzeige initialisiert, die den Verarbeitungsstatus in der Benutzeroberfläche visualisiert. Die Daten werden zunächst mittels Parsing extrahiert und normalisiert und als DataFrame ausgegeben. Anschließend wird der Gesamtdatensatz nach Jahren aufgeteilt, um jahresspezifische Analysen zu ermöglichen. Die nach Jahren aufgeteilten DataFrames werden in einem Dictionary mit Jahren als Schlüssel und den jeweiligen DataFrames als Werte gespeichert.

Über dieses Dictionary wird iteriert und so für jedes Jahr eine vollständige Analyse mittels des Analyse-Moduls durchgeführt. Die Fortschrittsanzeige wird entsprechend aktualisiert, was Nutzer:innen kontinuierliches Feedback über den Verarbeitungsstand gibt. Die

Ergebnisse der Analysen werden in der Datenklasse gespeichert. Die einzelnen Datenklassen-Objekte mit den Ergebnissen jedes einzelnen Jahres werden in einem Dictionary gespeichert, wobei der Schlüssel das Jahr identifiziert. Anschließend erfolgt eine Gesamtauswertung über alle Jahre hinweg. Dieses Datenobjekt wird unter dem Schlüssel „`Total`“ ebenfalls im Dictionary gespeichert.

5.7 Frontend

5.7.1 App

Die Komponente `app.py` koordiniert den gesamten Ablauf der Webanwendung. Zunächst werden den Nutzer:innen Instruktionen zur Datenbeschaffung der Aktivitätsdaten angezeigt. Anschließend ermöglicht eine Upload-Komponente das Hochladen der HTML-Datei, die temporär im Dateisystem gespeichert wird.

Sobald eine Datei hochgeladen wurde, wird die Datenverarbeitungspipeline ausgeführt. Nach der Verarbeitung wird die temporäre Datei gelöscht, um Speicherplatz freizugeben und den Datenschutz zu gewährleisten. Die Ergebnisse werden in einer Tab-Navigation dargestellt, wobei jedes Jahr sowie der Gesamtüberblick einen eigenen Tab erhalten. Innerhalb jedes Tabs werden die Analyseergebnisse in vier thematischen Abschnitten organisiert, die durch visuelle Trennelemente voneinander abgegrenzt sind. Das Rendering der Ergebnisse wird vom Display-Modul übernommen. Dieses Modul kapselt die Visualisierungslogik in Funktionen, die jeweils eine spezifische Darstellung übernehmen.

5.7.2 Display-Modul

Das Display-Modul wandelt die Ergebnisse aus der Datenverarbeitungspipeline in interaktive Darstellungen um und übernimmt die Fehlerbehandlung für die Frontend-Ausgabe.

Nutzungsmusteranalyse: Die Wochentags-Uhrzeit-Korrelation wird über einen Violin-Plot mit der Bibliothek `Plotly Express` visualisiert. Die Wochentage werden in chronologische Reihenfolge gebracht, indem eine explizite Sortierung über eine Liste Namens `weekday_order` festgelegt wird. Dies verhindert die alphabetische Standardsortierung von Plotly. Die Y-Achse zeigt Stunden im Dezimalformat (0.0 bis 24.0) und die X-Achse die sieben Wochentage. Im Violin-Plot werden mit dem Parameter `points='outliers'` nur statistische Ausreißer als einzelne Punkte im Plot angezeigt, während die Hauptverteilung durch die Violin-Form dargestellt wird. So werden Extrempunkte deutlich ohne eine visuelle Überladung.

Die monatliche Aktivitätsstatistik wird in einem Lineplot ebenfalls mit `Plotly Express` dargestellt. Die Y-Achse zeigt die Anzahl der Suchen und die X-Achse die Monatsnamen mit Jahreszahl. Die X-Achsenformatierung erfolgt über `xaxis_tickformat="%b %Y`, was Monatsnamen in abgekürzter Form mit Jahreszahl darstellt (z.B. „Feb 2024“). Dies verbessert die Lesbarkeit gegenüber der ursprünglichen Darstellung („2024-02“).

Geografische Analyse: Das Heatmap-Objekt aus der geografischen Analyse wird über `folium_static()` aus dem `streamlit-folium`-Package eingebettet, welches `Folium` Objekte in `Streamlit`-kompatible HTML-Komponenten konvertiert.

Kategorische Analyse: Die Funktion zur Visualisierung der kategorisierten Standortkarte erhält die Kartendaten als Tuple aus dem Analyse-Modul. Das Tuple enthält das Folium-Map-Objekt und die Anzahl der platzierten Marker. Das Rendering erfolgt ebenfalls über `folium_static()`. Falls das Tuple leer sein sollte, wird ein Hinweis für die Nutzer:innen angezeigt und `None` zurückgegeben. Zusätzlich werden die statistischen Metriken zur Anzahl verfügbarer Kategorien und die Gesamtzahl der kategorisierten Orte mit `st.metric()` dargestellt. Ein erweiterbares Panel (`st.expander()`) zeigt eine detaillierte Kategorieübersicht mit Anzahl der Orte pro Kategorie an. Diese Informationen werden aus dem entstandenen Dictionary der Kategorie-Analyse gezogen.

Häufigkeitsanalyse: Es werden die 20 ersten Ergebnisse der Häufigkeitsanalyse dargestellt. Die Verwendung von `st.dataframe()` ermöglicht die Darstellung des Dataframes in Form einer Tabelle. Weiterhin wird die Gesamtzahl der Aktivitäten als `st.metric()` dargestellt.

6 Evaluation

6.1 Funktionale Tests

6.1.1 Testmethodik

Die funktionalen Tests validieren die korrekte Funktionsweise aller Systemkomponenten anhand der vier Testdatensätze. Für jedes Modul wurden spezifische Testkriterien definiert und die Ergebnisse quantitativ erfasst.

Parsing- und Normalisierungs-Modul: Das DataFrame wird nach der Normalisierung einmalig als CSV-Datei exportiert und auf Richtigkeit überprüft. Dabei werden stichprobenartig extrahierte Werte mit den Originaleinträgen in der HTML-Datei abgeglichen.

Darüber hinaus wird die prozentuale Befüllung der einzelnen Spalten berechnet und auf zwei Nachkommastellen gerundet. Da nicht für jeden Eintrag alle Attribute in der Quelldatei vorhanden sind, ist eine lückenhafte Befüllung der Spalten erwartbar und kein Indikator für Fehlfunktionen.

LocationType-Modul: Das DataFrame aus der Häufigkeitsanalyse durchläuft das LocationType-Modul und wird anschließend als CSV-Datei gespeichert. Die Erfolgsquote der API-Anfragen wird durch die Befüllung der Spalte `location_type` und durch das Logging im System ausgewertet. Fehlgeschlagene Anfragen werden gezählt, prozentual ausgewertet und auf zwei Nachkommastellen gerundet.

Analyse-Module: Die Analyseergebnisse werden über einen End-to-End-Test validiert. Es wird überprüft, ob die Häufigkeitsanalyse korrekt sortiert ist. Bei der kategorischen Analyse werden die Anzahl erfolgreich zugeordneter Location und die Anzahl identifizierter thematischer Gruppen dokumentiert. Die geografische Analyse wird durch eine erfolgreiche Erstellung der Heatmap validiert. Die Nutzungsmusteranalyse wird auf plausible Werte überprüft (Uhrzeiten zwischen 0-24 Uhr, realistische Wochentags-Verteilungen).

Pipeline und Frontend: Die End-to-End Verarbeitung wird auf vollständige Durchführung ohne Systemabbruch getestet. Im Frontend wird überprüft, ob alle Visualisierungen korrekt gerendert und alle Jahre ausgewertet wurden.

6.1.2 Testergebnisse

Tabelle 5 zeigt die Ergebnisse der funktionalen Tests aller vier Testdatensätze, aus denen ein Mittelwert berechnet wurde.

Parsing-Modul: Die Spalte `date` weist eine vollständige Befüllung von 100% auf, da jeder Eintrag einen Zeitstempel besitzt. Die Spalte `query` erreicht mit durchschnittlich 91,02% und die Spalte `coordinates` mit 77,29% eine hohe Befüllungsrate. Diese betragen nicht 100%, da nicht alle Aktivitätseinträge eine Suchanfrage oder GPS-Koordinaten enthalten. Insbesondere allgemeine Ortssuchen ohne konkrete Adressen enthalten keine Koordinaten und systemgenerierte Aktivitätseinträge wie „Explored on Google Maps“ enthalten keine konkreten Suchanfragen. Die Spalte `action` weist mit 33,66% die niedrigste Befüllung auf. Dies ist erwartbar, da GM nicht für jede Aktivität einen expliziten Aktionstyp dokumentiert. Es gibt keine Hinweise auf systematische Extraktionsfehler.

Normalisierungs-Modul: Die Extraktion der Ortsbeschreibung in der Spalte `description` erreicht mit 91,02% die höchste Befüllung. Dieser Wert ist deckungsgleich mit der Befüllung der Spalte `query`, was darauf hindeutet, dass aus jeder Suchanfrage erfolgreich eine

Ortsbeschreibung oder mit der Fallback-Logik bei Straßenadressen die Straße extrahiert wurde. Die Extraktion einzelner Adresskomponenten zeigt niedrige Werte: Straßenadressen werden in 20,26% der Fälle erkannt, Postleitzahlen in 16,55% der Fälle und Städtenamen in 16,21%. Diese Werte reflektieren zum einen die Heterogenität der Suchanfragen, zum anderen weisen sie auf Lücken in den RegEx-Patterns hin.

LocationType-Modul: Die API-Anfragen erreichen eine Erfolgsquote von durchschnittlich 99,5%. Durch das Logging können Netzwerk-Timeouts ausgeschlossen werden. Für vereinzelte Adressen liefert die API keinen Ortstyp zurück. Nach einer Analyse der vereinzelten Adressen stellt sich heraus, dass die HTTP-Anfragen Sonderzeichen nicht korrekt interpretieren. Dies wurde in der finalen Implementierung behoben. Die implementierte Fehlerbehandlung verhindert Systemabstürze und kennzeichnet betroffene Einträge mit `None`.

Häufigkeitsanalyse: Alle vier Datensätze zeigen korrekt nach Häufigkeit sortierte Ergebnisse. Die ermittelten Top-Adressen sind plausibel und entsprechen den erwarteten Hauptaufenthaltsorten der jeweiligen Testpersonen.

Kategorische Analyse: Die thematische Zuordnung identifiziert durchschnittlich 77 Locations in 13 verschiedenen Ortsgruppen. Die Varianz spiegelt unterschiedliche Nutzungsprofile wider. Die Marker-Erstellung auf der interaktiven Karte verlief für alle Datensätze erfolgreich.

Nutzungsmuster-Analyse: Die zeitlichen Verteilungen zeigen plausible Werte mit erkennbaren Mustern. Eine Auffälligkeit ist die Skalierung der Y-Achse im Violin-Plot, die teilweise über den Rahmen von 0-24 Uhr hinausgeht. Dies ist auf die automatische Skalierung von Plotly zurückzuführen, beeinträchtigt jedoch nicht die Interpretierbarkeit.

Pipeline: Die End-to-End-Verarbeitung verlief für alle Datensätze erfolgreich. Alle vorhandenen Jahre wurden korrekt segmentiert und analysiert. Manche Jahres-Segmente wurden jedoch leer angezeigt; dies ist ein erwartetes Verhalten, wenn Nutzer:innen GM nicht regelmäßig nutzen.

Frontend: Alle Visualisierungen wurden erfolgreich dargestellt. Bei Jahren ohne Daten erscheinen entsprechende Info-Meldungen („No data available for this year“), was die implementierte Fehlerbehandlung validiert.

Die funktionalen Tests zeigen, dass alle Systemkomponenten funktionieren. Die niedrigen Befüllungsraten bei den extrahierten Adresskomponenten sind nicht auf Fehlfunktionen zurückzuführen, sondern reflektieren die heterogenen Nutzer-Suchanfragen und weisen auf

Lücken in den RegEx-Patterns hin. Die hohe Befüllung der `description`-Spalte zeigt, dass die Fallback-Mechanismen der Normalisierung greifen und auch bei fehlenden strukturierten Adressen sinnvolle Ortsbeschreibungen extrahiert werden.

Die nahezu vollständige Erfolgsquote der API-Anfragen und die fehlerfreie Verarbeitung durch die Pipeline demonstrieren die Robustheit des Systems. Die implementierte Fehlertoleranz ermöglicht Partial Results auch bei vereinzelten Fehlern.

Tabelle 5: Ergebnisse der funktionalen Tests
(Quelle: Eigene Darstellung)

Modul	**Test**	**Datensatz 01**	**Datensatz 02**	**Datensatz 03**	**Datensatz 04**	**Mittelwert**
Parsing-Modul	Spalte „action“	38,53%	17,75%	51,74%	26,61%	33,66%
	Spalte „query“	90,91%	96,06%	87,51%	89,60%	91,02%
	Spalte „coordinates“	73,68%	89,42%	75,21%	70,85%	77,29%
	Spalte „date“	100%	100%	100%	100%	100%
Normalisierungs-Modul	Straßenextraktion	22,43%	9,90%	34,75%	13,95%	20,26%
	PLZ-Extraktion	17,33%	8,28%	30,78%	9,79%	16,55%
	Städteextraktion	16,87%	8,11%	30,15%	9,70%	16,21%
	Beschreibungsextraktion	90,91%	96,06%	87,51%	89,60%	91,02%
LocationType-Modul	Erfolgreiche API-Anfragen	100%	99%	99%	100%	99,5%
Häufigkeitsanalyse	Aggregation	Korrekte Sortierung	Korrekte Sortierung	Korrekte Sortierung	Korrekte Sortierung	Korrekte Sortierung
Kategorische Analyse	Thematische Zuordnung	97 Locations in 14 Gruppen	58 Locations in 12 Gruppen	85 in 14 Gruppen	68 Locations in 13 Gruppen	77 Locations in 13 Gruppen
	Erstellung von Markern auf Heatmap	Erfolgreich	Erfolgreich	Erfolgreich	Erfolgreich	Erfolgreich
Geografische Analyse	Heatmap Erstellung	Erfolgreich	Erfolgreich	Erfolgreich	Erfolgreich	Erfolgreich
Nutzungsmuster	Zeitliche Verteilung	Plausible Werte	Plausible Werte	Plausible Werte	Plausible Werte	Plausible Werte
Pipeline	End-To-End	Alle Jahre verarbeitet	Alle Jahre verarbeitet	Alle Jahre verarbeitet	Alle Jahre verarbeitet	Alle Jahre verarbeitet
Frontend	Visualisierungen	Alle dargestellt	Alle dargestellt	Alle dargestellt	Alle dargestellt	Alle dargestellt

6.2 Performance-Analyse

6.2.1 Testmethodik

Die Performance des Systems wurde anhand der Logging-Timestamps während der Verarbeitung von Datensatz 04 analysiert (26.377 Einträge über einen Zeitraum von 15 Jahren). Jedes Modul protokolliert Start- und Endzeitpunkte seiner Verarbeitungsschritte, sodass die Verarbeitungsdauer in Millisekunden erfasst werden kann.

Die Messung erfolgte unter folgenden Bedingungen:

- **Testdatensatz:** Datensatz 04 mit 26.377 Einträgen (2011-2025, 15 Jahre)
- **Hardware:** MacBook Pro 2,3 GHz Quad-Core Intel Core i7, 32 GB RAM
- **Betriebssystem:** macOS 15.6.1
- **Python-Version:** 3.10
- **Netzwerkverbindung:** Stabile 2,4 gHz WLAN Verbindung
- **LocationType-Modul:** Ohne Cache-Implementierung (realistische Produktivumgebung)

Die Zeitstempel wurden extrahiert und Differenzen zwischen Start- und Endzeitpunkten berechnet. Die Pipeline durchlief das Analyse-Modul mit dem LocationType-Modul 16 mal (15 Jahre + Gesamtauswertung). Die Zeiten der einzelnen Analyseschritte wurden addiert. Für jede Verarbeitungsphase wurden die absolute Dauer und der prozentuale Anteil der Gesamtverarbeitungszeit ermittelt.

6.2.2 Testergebnisse

Die Tabelle 6 zeigt die detaillierte Aufschlüsselung der Verarbeitungszeiten nach Modulen und Phasen. Die Gesamtverarbeitungszeit beträgt 6 Minuten 33 Sekunden. Das LocationType-Modul dominiert mit 88,27%, was auf die externen API-Anfragen an die Google Geocoding API zurückzuführen ist. Die durchschnittliche Dauer pro API-Anfrage beträgt 0,28 Sekunden, die sich aus dem implementierten Rate-Limiting von 0,2 Sekunden (vgl. Kapitel 5.4) und mutmaßlichen Netzwerklatenzen zusammensetzt.

Die lokalen Verarbeitungskomponenten benötigen nur 42 Sekunden (10,7% der Gesamtzeit). Das Parsing-Modul verarbeitet mit 3.668 Einträgen pro Sekunde und die Normalisierung mit 1.869 Einträgen pro Sekunde die Daten effizient (vgl. Tabelle 6). Die Analyse-Module und das Frontend-Rendering sind mit zusammen unter 20 Sekunden vernachlässigbar.

Die Verarbeitungszeit der API-Phase bleibt auch bei größeren Datensätzen konstant durch die Limitierung auf maximal 200 Anfragen pro Jahr (vgl. Kapitel 5.4). Da die API-Phase mit 88,27% der Verarbeitungszeit dominiert und von der Eintragszahl nicht abhängt, lässt sich darauf schließen, dass die Gesamtverarbeitungszeit auch bei größeren Datensätzen im Bereich von 6-7 Minuten verbleibt.

Die Performance von etwa 6,5 Minuten für einen Datensatz mit 26.377 Einträgen liegt im akzeptablen Bereich für das Analyse-Tool. Der identifizierte Bottleneck im LocationType-Modul ist erwartungsgemäß und durch die externe API-Abhängigkeit bedingt. Die implementierte Fortschrittsanzeige ist essentiell, um Nutzer:innen über den Verarbeitungsstatus zu informieren. Für den Produktiveinsatz ist die Performance vertretbar, erfordert jedoch klare Nutzer-Kommunikation über die erwartete Wartezeit. Optional könnte die LocationType-Anreicherung deaktiviert werden, dadurch würde sich die Verarbeitungszeit auf etwa 30 Sekunden reduzieren.

Tabelle 6: Ergebnisse der Performance-Analyse des Analysetools mit Datensatz 04
(Quelle: Eigene Darstellung)

Phase	**Dauer (hh:mm:ss,000)**	**Anteil der Gesamtdauer**	**Anmerkung**
Backend			
HTML-Datei laden	00:00:00,001	<0,01%	Von Upload bis Pipeline-Start
Parsing-Modul	00:00:07,191	1,83%	HTML-Parsing und Datenextraktion
Normalisierungs-Modul	00:00:02,027	0,52%	Adresskomponenten extrahieren
Jahressegmentierung	00:00:11,519	2,93%	DataFrame in Jahre aufteilen
LocationType-Modul	00:05:46,750	88.27%	API-Anfragen an Google Geocoding
Analyse-Modul			
Häufigkeitsanalyse	00:00:00,979	0.25%	Aggregation nach Orten
Geografische Analyse	00:00:00,583	0.15%	Heatmap-Generierung
Kategorische Analyse	00:00:00,496	0.13%	Thematische Gruppierung der Suchanfragen
Nutzungsmusteranalyse	00:00:14,058	3.58%	Zeitliche Verteilungen
Analyse-Modul inkl. LocationType-Modul	00:06:02,866	92.37%	Inkludiert die API-Anfragen aus dem LocationType-Modul
Frontend			
Display-Modul	00:00:03,914	1,00%	Visualisierungen erstellen
Frontend bauen	00:00:01,407	0,36%	UI-Rendering
Pipeline gesamt	00:06:32,467	99,91%	Komplette Verarbeitung
Gesamtdauer	00:06:32,839	100%	Upload bis Darstellung

6.3 Anwendung und Usability

Die Evaluation der Anwendung und der Usability analysiert anhand der Implementierung, inwieweit das entwickelte Analysetool die in der Anforderungsanalyse definierten benutzerseitigen Anforderungen erfüllt (vgl. Kapitel 3.1.1). Die Bewertung erfolgt entlang des intendierten Nutzungsverlaufs vom ersten Öffnen der Anwendung bis zur finalen Datenanalyse.

Datenimport: Beim Öffnen der Streamlit-Anwendung im Webbrowser wird eine Aufforderung zum Upload der eigenen GM-Aktivitätsdaten angezeigt. Weiterhin wird eine Anleitung zum Ausklappen zur Beschaffung der Daten angezeigt (vgl. Abb. 6). Nach der Dateiauswahl beginnt die Verarbeitung automatisch, ohne dass weitere Konfigurationen nötig sind.

Das Tool akzeptiert die HTML-Datei in dem Format, wie Google sie bereitstellt. Es sind keine manuellen Umwandlungen oder Vorbereitungen der Daten notwendig. Dies entspricht dem Prinzip, dass die Beschaffung der Daten zwar nutzerseitig erfolgt, die technische Verarbeitung jedoch vollständig vom Tool übernommen wird.

Verarbeitung und Wartezeit: Nach dem Upload beginnt die Datenverarbeitung. Ein Fortschrittsbalken visualisiert die einzelnen Schritte. Eine Systemmeldung informiert über die voraussichtliche Verarbeitungsdauer von etwa sieben Minuten.

Ergebnisdarstellung: Nach Abschluss der Verarbeitung werden die Analyseergebnisse jahressweise sortiert angezeigt. Eine Tab-Navigation ermöglicht es, zwischen verschiedenen Jahren zu wechseln, ergänzt durch eine Gesamtübersicht. Für jedes Jahr präsentiert die Anwendung die Analyseergebnisse in mehreren Bereichen.

Nutzungsmusteranalyse: Die Oberfläche zeigt übersichtlich, wie viele Suchanfragen in diesem Jahr gestellt wurden. Ein Linien-Diagramm zeigt die monatliche Verteilung. Spitzen und ruhigere Phasen sind sofort erkennbar (vgl. Abb. 7). Weiterhin wird im Violinplot die Verteilung der Suchaktivitäten nach Wochentagen und Uhrzeiten dargestellt. Breite Ausbuchtungen zeigen Phasen intensiver Nutzung, während schmale Bereiche auf geringere Aktivität hindeuten (vgl. Abb. 8). Die Y-Achse geht aufgrund der automatischen Skalierung von Plotly teilweise über den Rahmen von 0-24 Uhr hinaus, was die Interpretierbarkeit jedoch nicht beeinträchtigt. So werden Routinen in der Nutzung deutlich, wie z.B. Pendelzeiten.

Häufigkeitsanalyse: In einer Tabelle werden die zwanzig meistgesuchten Begriffe angezeigt mit Informationen zur Häufigkeit und Adresse. Die Übersicht ermöglicht es

Nutzer:innen, ihre persönlichen „Top-Orte“ zu identifizieren. Die tabellarische Darstellung ist nach Häufigkeit sortiert, und es ist erkennbar, welche Orte die größte Rolle im Alltag des Nutzer:innen spielen (vgl. Abb. 9).

Geografische Analyse: Eine interaktive Karte zeigt die Verteilung der Aktivitäten auf einer Heatmap. Es werden die Schwerpunkte der eigenen geografischen Schwerpunkte deutlich. Nutzer:innen können in die Karte hineinzoomen, um Details zu erkunden, oder herauszoomen, um überregionale Muster zu erkennen (vgl. Abb. 10). So lassen sich Lebensschwerpunkte visuell identifizieren. Auch Urlaubsorte oder Umzüge können mit dem eigenen Wissen identifiziert werden.

Kategorische Analyse: Die Anwendung zeigt die Anzahl der verschiedenen besuchten Ortsgruppen sowie die dazugehörige Anzahl an Orten an. Eine detaillierte numerische Verteilung der Orte pro Gruppe wird in einem ausklappbaren Element angezeigt (vgl. Abb. 11). So können die beliebtesten Ortsgruppen abgelesen werden. Darunter befindet sich eine Karte, auf der die meistbesuchten Orte jeder Gruppe markiert sind. Ein Klick auf einen Marker zeigt Name und Adresse des Ortes (vgl. Abb. 12). Über ein Layer-Control, können bestimmte Gruppen ein- und ausgeblendet werden (vgl. Abb. 13).

Navigation und Vergleich: Die jahresweise Aufschlüsselung ermöglicht es, eigene zeitliche Entwicklungen nachzuvollziehen. Der Wechsel zwischen den Jahren zeigt, wo sich Suchverhalten und geografische Schwerpunkte verändert haben. Die Gesamtübersicht bietet einen aggregierten Blick über alle Jahre hinweg. Die Navigation ist flüssig. Alle Visualisierungen laden ohne spürbare Verzögerung. Die Interaktivität der Karten funktioniert erwartungsgemäß.

Die Anwendung bietet einen vollständigen funktionalen Workflow zur Synthese von GM-Aktivitätsdaten. Der Einstieg ist niedrigschwellig, die Ergebnisse sind aussagekräftig und visuell strukturiert aufbereitet. Die größte Herausforderung liegt in der Wartezeit während der Verarbeitung. Die Implementierung erfüllt alle definierten funktionalen und benutzerseitigen Anforderungen und ermöglicht Nutzer:innen ohne technische Expertise einen tiefen Einblick in die digitale Identität, wie sie sich in GM-Aktivitäten widerspiegelt.

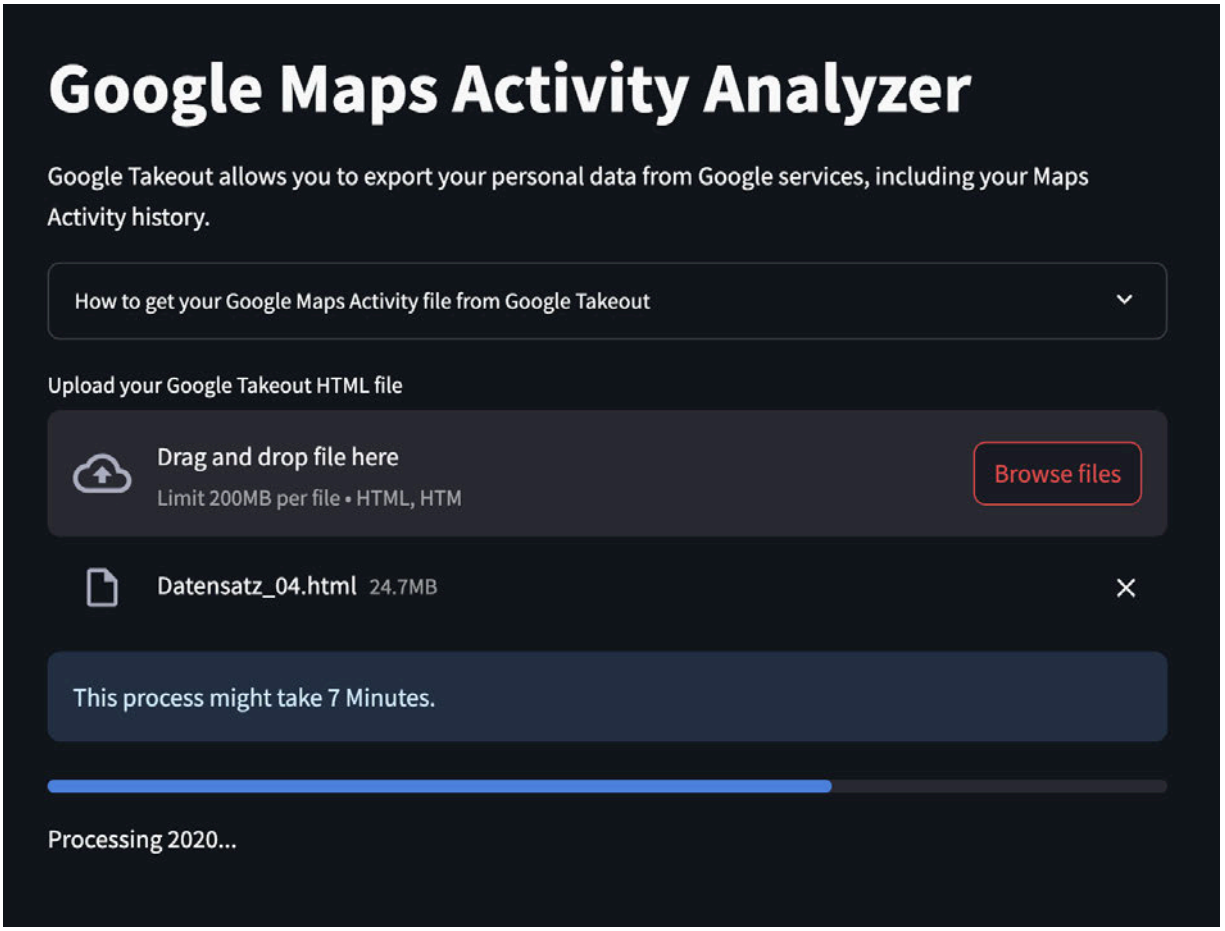

Abbildung 6: Benutzerinterface des Analyse-Tools nach Hochladen der GM-Aktivitätsdaten (Quelle: Eigene Darstellung)

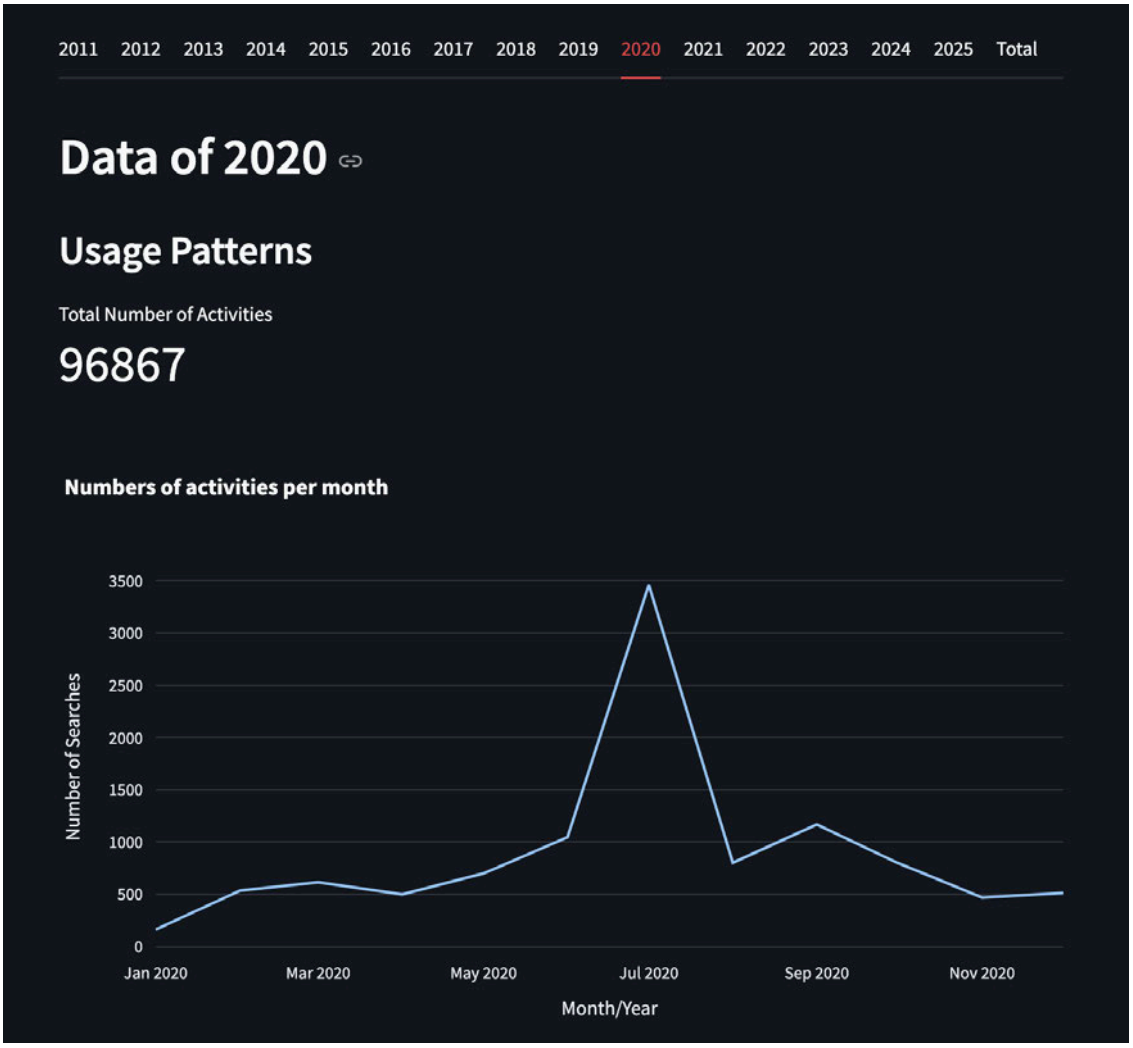

Abbildung 7: Screenshot des Analysetools: Nutzungsmusteranalyse, Liniendiagramm (Quelle: Eigene Darstellung)

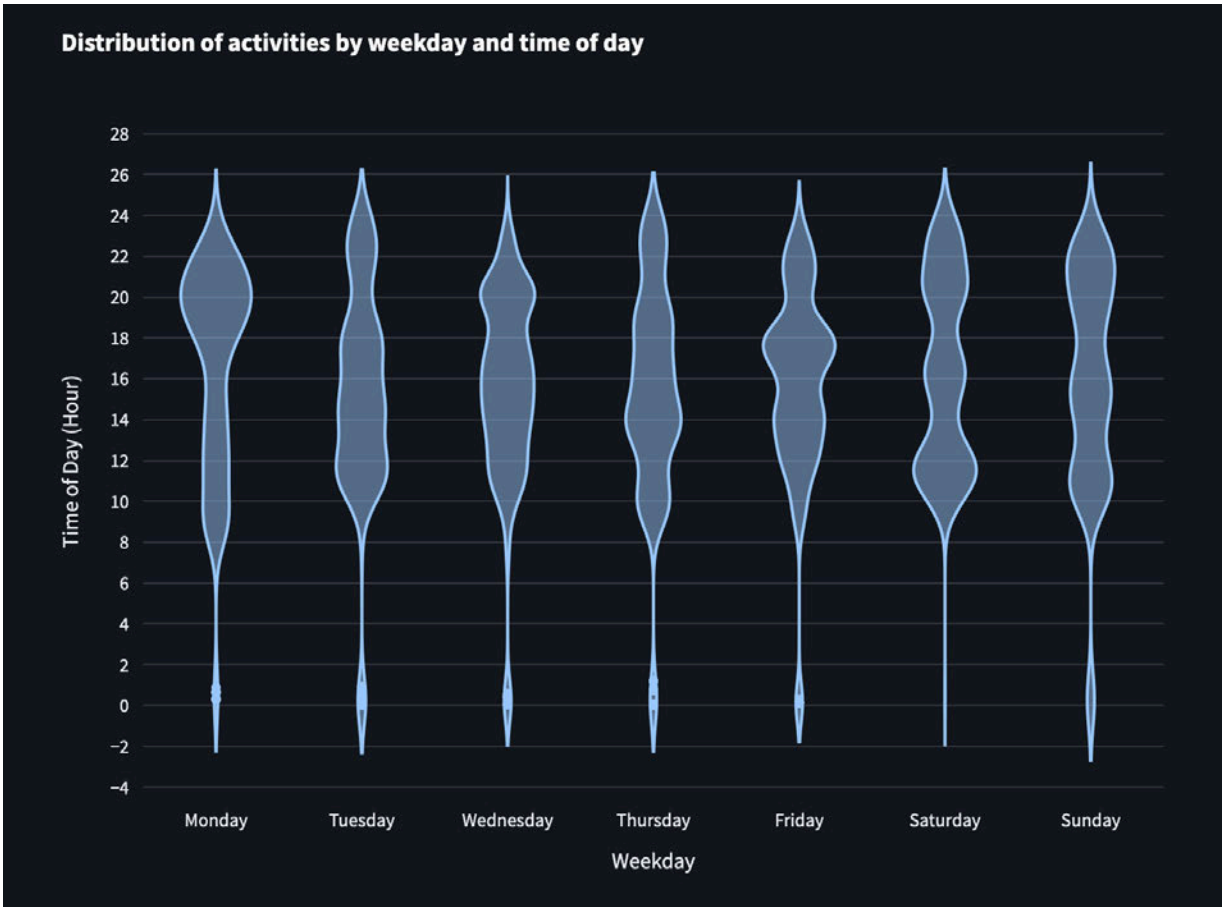

Abbildung 8: Screenshot des Analysetools: Nutzungsmusteranalyse, Violin-Plot
(Quelle: Eigene Darstellung)

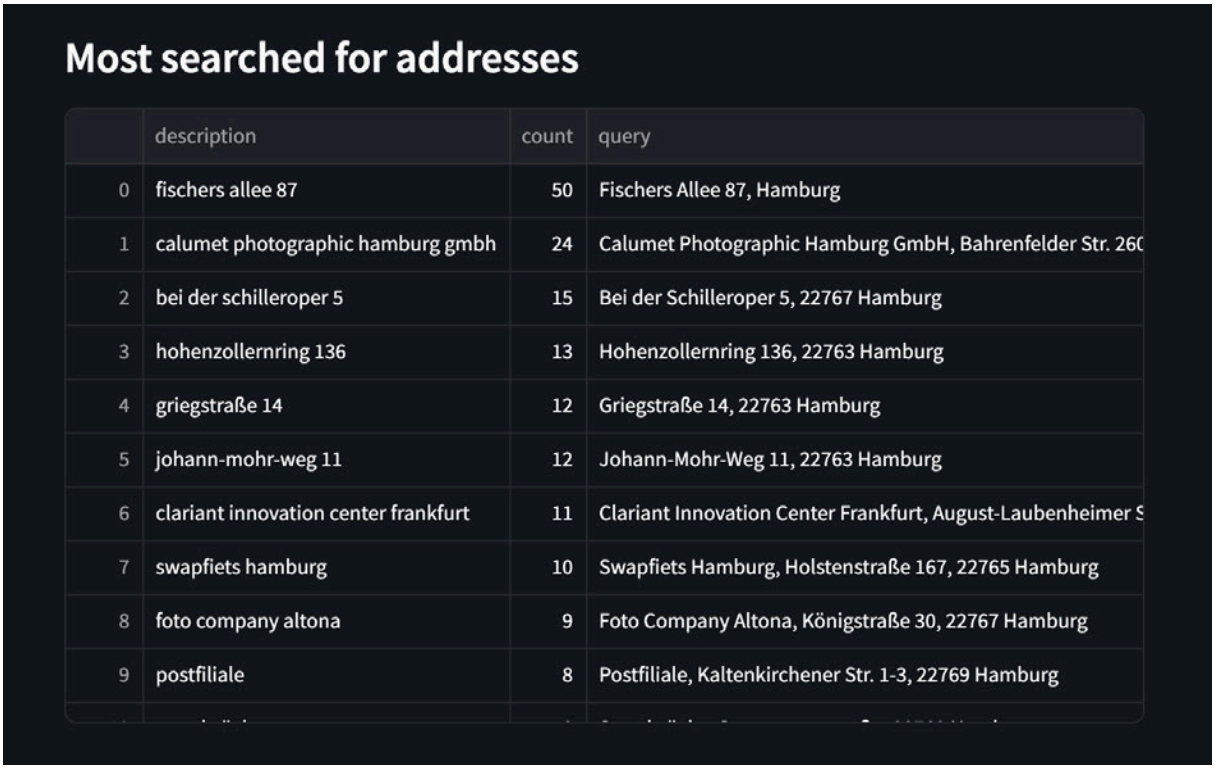

Abbildung 9: Screenshot des Analysetools: Häufigkeitsanalyse
(Quelle: Eigene Darstellung)

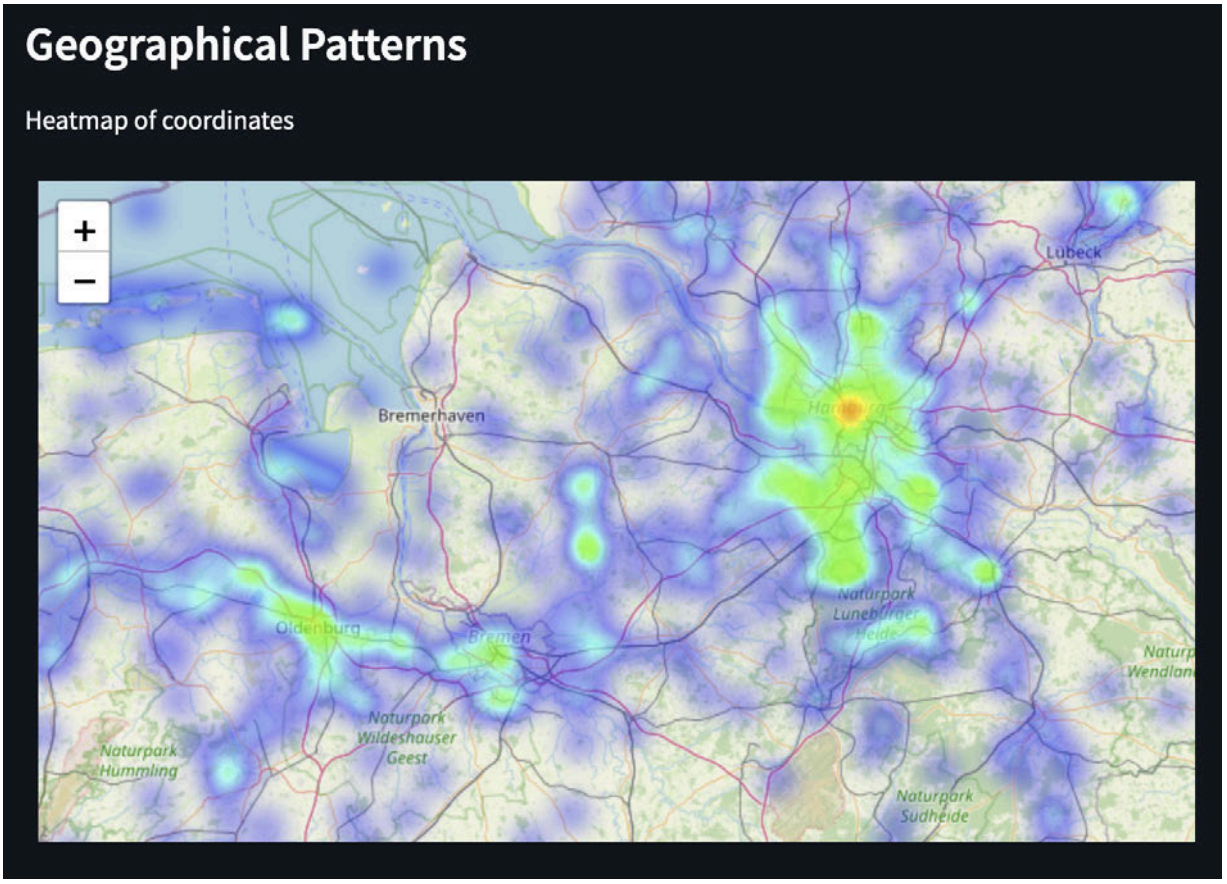

Abbildung 10: Screenshot des Analysetools: Geografische Analyse

(Quelle: Eigene Darstellung)

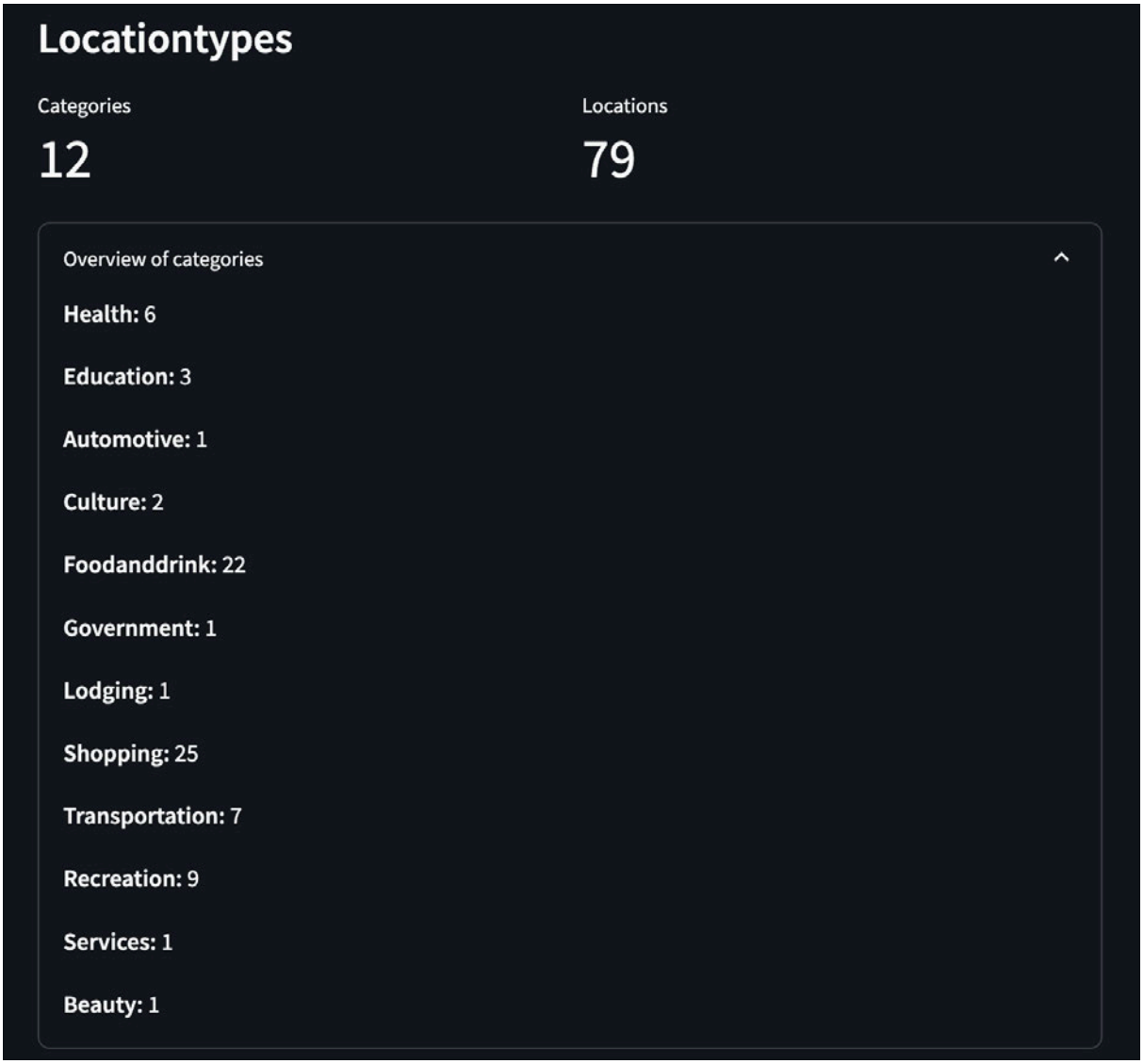

Abbildung 11: Screenshot des Analysetools: Kategorische Analyse

(Quelle: Eigene Darstellung)

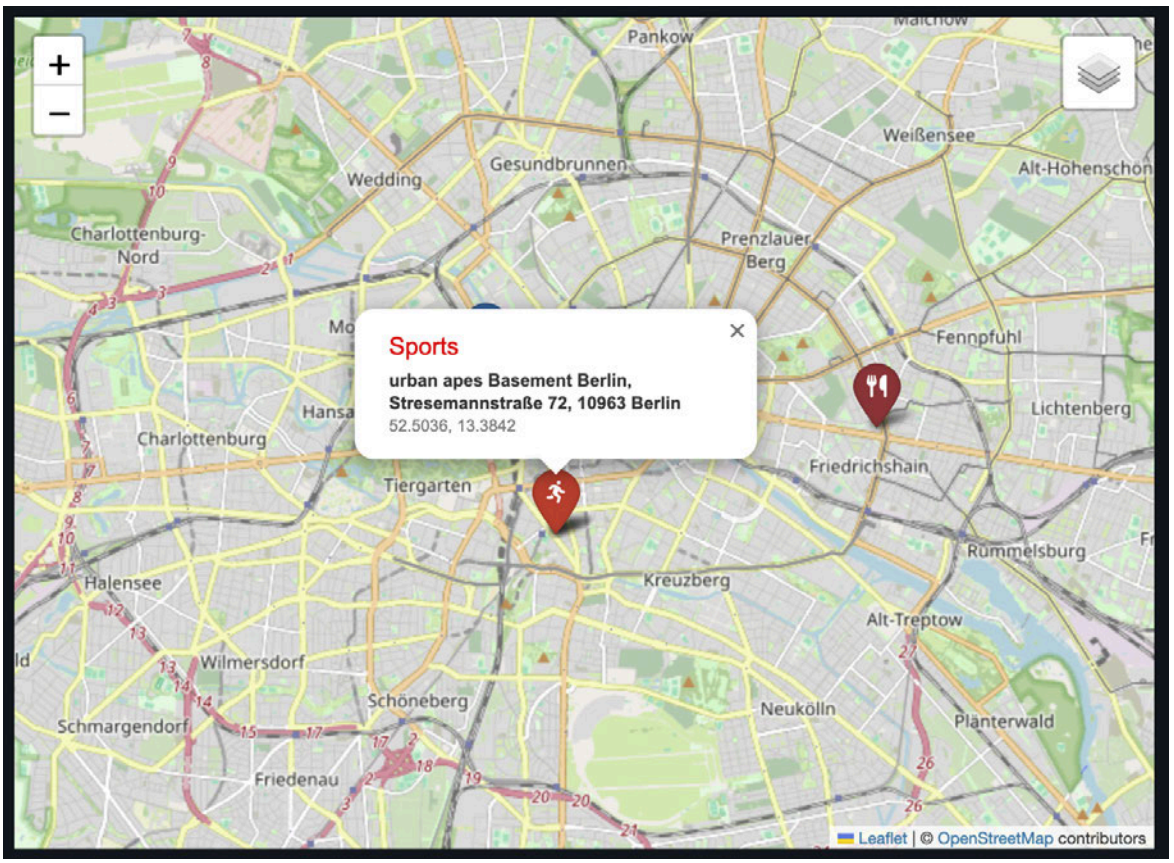

Abbildung 12: Screenshot des Analysetools: Kategorische Analyse, Marker Pop-Up

(Quelle: Eigene Darstellung)

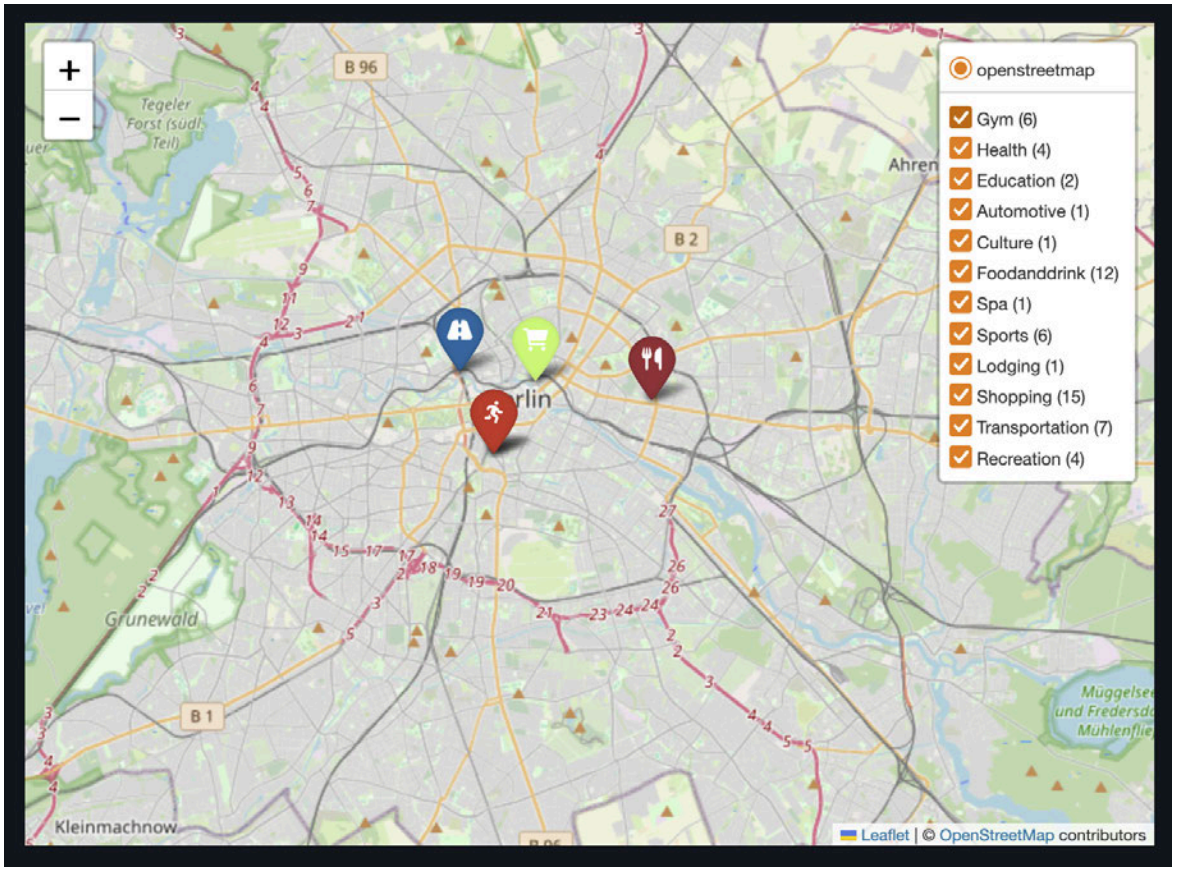

Abbildung 13: Screenshot des Analysetools: Kategorische Analyse, Layer-Control (Quelle: Eigene Darstellung)

7 Diskussion

Die vorliegende Arbeit demonstriert die Entwicklung eines funktionsfähigen Datenanalysetools zur Synthese von GM-Aktivitätsdaten. Die Implementierung erfüllt die in der Anforderungsanalyse definierten funktionalen Anforderungen und ermöglicht es Nutzer:innen, umfassende Einblicke in ihre eigenen Suchgewohnheiten zu gewinnen. Im Folgenden werden die Ergebnisse interpretiert, methodische Limitierungen diskutiert sowie Datenschutzimplikationen und Verbesserungspotentiale aufgezeigt.

7.1 Methodenkritik

Die gewählte Implementierung bietet mehrere methodische Vorteile, weist jedoch auch systematische Limitationen auf, die im Folgenden kritisch reflektiert werden.

Die modulare Architektur nach dem Seperation-Of-Concerns-Prinzip ermöglicht die unabhängige Entwicklung und Testbarkeit einzelner Komponenten. Durch die klare Trennung zwischen Datenverarbeitung und Präsentationslogik können Änderungen in einem

Modul vorgenommen werden, ohne andere Komponenten zu beeinträchtigen. Dies erhöht die Wartbarkeit und erleichtert zukünftige Erweiterung, etwa die Integration zusätzlicher Analysemethoden oder alternativer Visualisierungen.

Die Verwendung der Google Geocoding API für die Ortstypen-Kategorisierung gewährleistet konsistente Zuordnungen auf Basis der originalen Google-Daten. Da die Analysedaten ebenfalls aus dem Google-Ökosystem stammen, werden Orte mit denselben Kategorisierungslogiken klassifiziert, die auch bei ihrer ursprünglichen Erfassung verwendet wurden.

Das Privacy-by-Design-Prinzip mit ausschließlich lokaler Datenverarbeitung (mit Ausnahme der API-Anfragen) schützt die Privatsphäre der Nutzenden. Sensible Daten verlassen das lokale System nicht und werden nicht auf externen Servern gespeichert.

Die jahresweise Strukturierung der Analyseergebnisse ermöglicht zeitliche Vergleiche und macht Veränderungen im Nutzerverhalten erkennbar. Diese langfristige Perspektive erlaubt es, Umzüge, Jobwechsel oder Veränderungen in Nutzungsmustern zu identifizieren.

Die regelbasierte Adressnormalisierung mittels regulärer Ausdrücke ist auf deutsche Adressformate optimiert und weist bei internationalen Adressen möglicherweise systemische Einschränkungen auf. Die implementierten Muster erkennen deutsche Strukturen wie „Straße Hausnummer, PLZ Ort“ zuverlässig, scheitern jedoch bei abweichenden internationalen Konventionen. Beispielsweise werden britische Adressen nach dem Schema „Hausnummer Straße, Ort, Postcode“ oder US-amerikanische Adressen mit State-Abkürzungen nicht korrekt in ihre Komponenten zerlegt. Dies limitiert die Universalität des Tools erheblich. Eine semantische Analyse mittels Named Entity Recognition (NER) würde genauere Ergebnisse liefern, erforderte jedoch trainierte Sprachmodelle und würde das System erheblich komplexer machen.

Die Qualität der Ortstypzuordnung hängt stark von der Struktur der Suchanfrage ab. Bei Suchanfragen ohne expliziten Ortsnamen oder bei allgemeinen Straßenadressen liefert die API häufig nur generische Kategorien wie street_address. Diese Kategorisierung bildet die tatsächliche Nutzung nicht immer ab: Straßenadressen können sowohl reine Wohnadressen als auch spezifische Einrichtungen wie Restaurants oder Arztpraxen bezeichnen. Dies schränkt die thematische Auswertung ein und führt zu einer Unterrepräsentation bestimmter Ortstypen in der Analyse. Zudem gibt Google die Ortstypen als ungeordnete Liste zurück. Dies kann zu Inkonsistenzen führen. So ist beispielsweise die Boulderhalle „Urban Apes West“ in Hamburg auch als Café gelistet, da sie auch Heißgetränke anbietet.

Ein weiteres Problem stellt die Performance dar. Mit 88,3% Anteil an der Gesamtverarbeitungszeit ist das LocationType-Modul der klare Bottleneck. Jedoch ist dies für eine einmalige Analyse akzeptabel, da dies vor dem Hochladen der Datei den Nutzer:innen mittels eines Hinweises kommuniziert wird. Dennoch limitiert die Performance die Praktikabilität des Tools.

Die Validierung erfolgte auf Basis eines kleinen, selbst erhobenen Datensatzes von Freunden und Bekannten der Autorin. Diese Stichprobe ist weder repräsentativ für die Gesamtbevölkerung noch für verschiedene Altersgruppen, Regionen oder Nutzungsprofile. Die Validierung fand primär auf funktionaler Ebene statt. Es wurde nicht überprüft, ob die extrahierten Nutzungsmuster und die abgebildeten Lebensschwerpunkte mit der tatsächlichen Lebensrealität übereinstimmen. Die Validierung beschränkt sich auf informelle mündliche Rückmeldungen zur Plausibilität der Ergebnisse. Dies wäre methodisch durch strukturierte Interviews möglich gewesen, hätte jedoch den Rahmen dieser Arbeit überschritten.

Schließlich ist zu beachten, dass die Analyse ausschließlich Suchanfragen umfasst, nicht jedoch andere Formen der Standortdatenerfassung wie GPS-Tracking. Der rekonstruierte digitale Fußabdruck ist somit auf einen spezifischen Aspekt des Nutzungsverhaltens beschränkt. Zudem hängt die Datenqualität vom individuellen Nutzungsverhalten ab: Nutzer:innen, die GM selten verwenden, liefern deutlich weniger aussagekräftige Analysen.

7.2 Limitationen der Formatwahl

Die Verwendung des HTML-Formats anstelle des verfügbaren JSON-Formats führt zu erheblichem technischem Mehraufwand im Parsing-Modul. Eine Neuentwicklung mit JSON-Daten würde den Implementierungsaufwand reduzieren und die Robustheit des Systems erhöhen. Das HTML-Parsing ist anfälliger für Fehler bei eventuellen zukünftigen Strukturänderungen von Google. Der Parsing-Schritt würde bei der Nutzung der JSON-Daten nahezu vollständig entfallen.

Trotzdem sind die entwickelten Analysemethoden vom Format unabhängig. Die gewonnenen Erkenntnisse über die Machbarkeit automatisierter Analysen bleiben gültig. Eine sorgfältige Prüfung verfügbarer Datenformate vor Projektstart ist essentiell. Diese Arbeit demonstriert dennoch, dass auch bei unterdurchschnittlichen Ausgangsbedingungen eine erfolgreiche Implementierung möglich ist.

7.3 Datenschutz

Die Analyse demonstriert, welche Rückschlüsse aus GM-Aktivitätsdaten möglich sind. Aus den meistgesuchten Orten lassen sich mit hoher Wahrscheinlichkeit Wohnort, Arbeitsplatz, bevorzugte Einkaufsorte und Freizeitaktivitäten ableiten. Die zeitlichen Muster ermöglichen es, auf Tagesabläufe, Arbeitszeiten und Routinen zurückzuschließen. In Kombination ergibt sich ein detailliertes Profil der digitalen Identität einer Person.

Die Erkenntnisse unterstreichen die Notwendigkeit eines bewussten Umgangs mit digitalen Diensten. Während Google diese Daten nutzt, um Dienste zu personalisieren und zu verbessern, besteht das Risiko des Missbrauchs. Die DSGVO und das Recht auf Datenportabilität (DSGVO, Art. 20) geben Nutzer:innen zwar Kontrolle über ihre Daten, doch die wenigsten machen von diesem Recht Gebrauch oder sind sich der Aussagekraft ihrer Daten bewusst.

Das entwickelte Tool kann in diesem Kontext eine sensibilisierende Funktion erfüllen: durch die Visualisierung der eigenen Daten werden abstrakte Datenschutzrisiken konkret erfahrbar. Gleichzeitig zeigt das Tool, dass lokale und datenschutzkonforme Analysen möglich sind, ohne Daten an Dritte weiterzugeben.

7.4 Ausblick

Für eine Weiterentwicklung des Tools ergeben sich mehrere Ansatzpunkte.

Die API-Phase könnte durch Parallelisierung der Anfragen optimiert werden. Eine optionale Deaktivierung der Ortstypkategorisierung würde die Verarbeitungszeit auf etwa 30 Sekunden reduzieren und wäre für Schnellanalysen ausreichend.

Durch die Verbindung aufeinanderfolgender Suchanfragen könnten Bewegungsmuster und typische Routen rekonstruiert werden. Cross-Plattform-Analysen mit anderen Google-Diensten (Youtube, Search, Chrome) würden ein noch umfassenderes Bild der digitalen Identität ermöglichen. Predictive Analytics könnten zukünftige Suchmuster vorhersagen und personalisierte Empfehlungen generieren.

Eine Export-Funktion der Analyse sowie interaktive Filter für zeitliche und geografische Eingrenzungen würden die Nutzererfahrung interaktiver gestalten.

Die vorliegende Arbeit zeigt das Potential von Datenanalyse-Tools zur Selbstreflexion im digitalen Raum. Zukünftige Forschung könnte untersuchen, wie solche Tools zur Medienkompetenz und Datenschutzsensibilisierung beitragen.

Langfristig könnten standardisierte Schnittstellen für Datenportabilität und Open-Source-Analysetools dazu beitragen, die Kontrolle über persönliche Daten von großen Tech-Konzernen zurück zu den Nutzenden zu verlagern. Die technische Machbarkeit ist, wie diese Arbeit zeigt, gegeben. Die Herausforderung liegt darin, solche Tools in den gesellschaftlichen Diskurs über digitale Selbstbestimmung einzubetten.

8 Fazit

Die vorliegende Arbeit demonstriert die Entwicklung eines funktionsfähigen Analysetools zur Synthese von GM-Aktivitätsdaten aus Google Takeout. Unter Synthese wird dabei die Zusammenführung einzelner Aktivitäten, insbesondere Suchanfragen, zu einem kohärenten Bild digitaler Nutzungsmuster verstanden. Das entwickelte Python-basierte Tool extrahiert aus heterogenen Aktivitätseinträgen interpretierbare Informationen über Routinen, Interessen und geografische Lebensmittelpunkte.

Die Implementierung zeigt, dass die automatisierte Verarbeitung nutzergenerierter Daten erhebliche technische Herausforderungen mit sich bringt. Die Aufbereitung der Daten erwies sich als ressourcenintensiver als die eigentliche Analyse: Parsing, Normalisierung und semantische Anreicherung erfordern robuste Verarbeitungslogik, um die strukturelle Vielfalt der Suchanfragen zu bewältigen. Trotz dieser Komplexität liefert das Tool aussagekräftige Ergebnisse. Die Häufigkeitsanalyse identifiziert zuverlässig geografische Schwerpunkte, die kategorische Analyse erschließt thematische Nutzungsmuster, die zeitliche Auswertung macht Routinen sichtbar, und die geografische Heatmap visualisiert Aktivitätscluster. Die jahresweise Strukturierung der Ergebnisse erlaubt zeitliche Vergleiche und macht Veränderungen im Nutzungsverhalten erkennbar.

Die funktionalen Tests mit vier realen Datensätzen validieren das System über unterschiedliche Nutzungsprofile hinweg. Die modulare Architektur ermöglicht zukünftige Erweiterungen, etwa die Integration des JSON-Formats oder zusätzlicher Analysemethoden.

Die Arbeit zeigt auch systematische Grenzen automatisierter Datenverarbeitung. Sprachbarrieren bei internationalen Adressen, unterschiedliche Zeitzonen und semantische Mehrdeutigkeit von Suchanfragen erfordern kontextspezifische Anpassungen, die nicht vollständig automatisierbar sind. Eine manuelle Konfiguration würde die Genauigkeit erhöhen, widerspricht jedoch dem Anspruch eines universell einsetzbaren Tools ohne technische Vorkenntnisse. Die Abhängigkeit von der Google Geocoding API schafft einen Performance-Bottleneck und steht im Spannungsfeld zu den datenschutzorientierten Anforderungen. Einerseits ermöglicht die API semantische Ortskategorisierung, andererseits

werden Suchanfragen an Google übermittelt. Eine vollständig lokale Verarbeitung würde auf die kategorische Analyse verzichten müssen, oder sie würde umfangreiche lokale Geodatenbanken erfordern.

Die Umsetzung der Privacy-by-Design-Prinzipien zeigt, dass datenschutzkonforme Analysetools technisch realisierbar sind. Die ausschließlich lokale Verarbeitung im RAM, die automatische Datenlöschung nach Abschluss und der Verzicht auf persistente Speicherung demonstrieren alternative Ansätze zu datensammelnden kommerziellen Diensten. Nutzer:innen behalten vollständige Kontrolle über ihre Daten und können diese kritisch reflektieren, ohne weitere Spuren zu hinterlassen.

Die Arbeit unterstreicht die gesellschaftliche Relevanz transparenter Datenverarbeitung. Selbst bei deaktivierter Standortfreigabe bei Google-Diensten ermöglichen Aktivitätsdaten präzise Rückschlüsse auf persönliche Lebensmittelpunkte. Die Synthese der GM-Aktivitätsdaten zu einem interpretierbaren Gesamtbild macht abstrakte Datenschutzrisiken sichtbar und trägt zur digitalen Selbstbestimmung bei.

Das entwickelte Tool beweist die technische Machbarkeit lokaler, datenschutzkonformer Datenanalyse und leistet einen Beitrag zur kritischen Auseinandersetzung mit digitaler Selbstbestimmung. Die Herausforderung liegt nun darin, solche Tools in den gesellschaftlichen Diskurs über Datensouveränität einzubetten.

Literaturverzeichnis

About – Google Maps. (o. D.). Google. https://www.google.com/maps/about/ (zuletzt abgerufen am 05.12.2025)

Awan, A. A. (2024, 10. September). Die 10 wichtigsten Tools für Datenwissenschaft im Jahr 2024. Datacamp. https://www.datacamp.com/de/blog/top-data-science-tools (zuletzt abgerufen am 22.10.2025)

Best Practices für das Geocoding von Adressen. (o. D.). Google For Developers. https://developers.google.com/maps/documentation/geocoding/best-practices?hl=de (zuletzt abgerufen am 05.12.2025)

Biermann, K. (2010, 2. März). Speichern ja, aber nicht so. DIE ZEIT. https://www.zeit.de/digital/datenschutz/2010-03/vorratsdaten-bverfg-urteil (zuletzt abgerufen am 05.12.2025)

Biermann, K. (2015, 3. September). Was Vorratsdaten über uns verraten. DIE ZEIT. https://www.zeit.de/digital/datenschutz/2011-02/vorratsdaten-malte-spitz/komplettansicht (zuletzt abgerufen am 05.12.2025)

Chen, Q., Banerjee, A., Demiralp, Ç., Durrett, G. & Dillig, I. (2023). Data Extraction via Semantic Regular Expression Synthesis. Proceedings Of The ACM On Programming Languages, 7(OOPSLA2), 1848–1877. https://doi.org/10.1145/3622863

Codelabs Academy. (2024, 11. November). Die wichtigsten Data-Science-Tools und -Technologien, die Sie im Jahr 2025 kennen sollten. https://codelabsacademy.com/de/blog/top-data-science-tools-and-technologies-you-should-know (zuletzt abgerufen am 22.10.2025)

Datamade. (o. D.). GitHub - datamade/usaddress: :us: a python library for parsing unstructured United States address strings into address components. GitHub. https://github.com/datamade/usaddress (zuletzt abgerufen am 05.12.2025)

DovarFalcone. (o. D.). google-takeout-location-parser: Easily parse location .json files provided by the Google Takeout service. GitHub. https://github.com/DovarFalcone/google-takeout-location-parser (zuletzt abgerufen am 08.02.2026)

Folium 0.20.0 documentation. (o. D.). https://python-visualization.github.io/folium/latest/ (zuletzt abgerufen am 05.12.2025)

Gaur, G. (2023, 17. September). Orchestration pattern: managing distributed transactions. Code Thoughts. https://www.gaurgaurav.com/patterns/orchestration-pattern/ (zuletzt abgerufen am 05.12.2025)

GeeksforGeeks. (2025, 23. Juli). 7 Best tools and technologies for data science. GeeksforGeeks. https://www.geeksforgeeks.org/blogs/best-tools-and-technologies-for-data-science/ (zuletzt abgerufen am 22.10.2025)

Google Datenexport. (o. D.). Google Takeout. https://takeout.google.com/ (zuletzt abgerufen am 05.12.2025)

Google Maps-Aktivitäten verwalten - Computer - Google Maps-Hilfe. (o. D.). https://support.google.com/maps/answer/3137804?hl=de-DE&co=GENIE.Platform%3DDesktop (zuletzt abgerufen am 05.12.2025)

Google-Produkte parallel verwenden - Computer - Google Kalender-Hilfe. (o. D.). https://support.google.com/calendar/answer/106237?hl=de&co=GENIE.Platform%3DDesktop&sjid=3221316576907160187-EU (zuletzt abgerufen am 09.12.2025)

Heatmap — Folium 0.20.0 documentation. (o. D.). https://python-visualization.github.io/folium/latest/user_guide/plugins/heatmap.html (zuletzt abgerufen am 05.12.2025)

Jens. (2020, 10. Februar). 15 Jahre Google Maps: So hat sich die Kartenplattform verändert & die wichtigsten Meilensteine in Deutschland. GoogleWatchBlog. https://www.googlewatchblog.de/2020/02/jahre-google-maps-meilensteine/ (zuletzt abgerufen am 05.12.2025)

Li, A. (2024, 29. Oktober). Google Maps now has over 2 billion monthly users. 9to5Google. https://9to5google.com/2024/10/29/google-maps-2-billion/ (zuletzt abgerufen am 09.12.2025)

Lotfi, C., Srinivasan, S., Ertz, M. & Latrous, I. (2021). Web Scraping Techniques and Applications: A Literature Review. Soft Computing Research Society eBooks, 381–394. https://doi.org/10.52458/978-93-91842-08-6-38

Ortstypen (neu). (o. D.). Google For Developers. https://developers.google.com/maps/documentation/places/web-service/place-types?hl=de (zuletzt abgerufen am 05.12.2025)

Plotly Express in Python. (o. D.). Plotly Graphing Libraries. https://plotly.com/python/plotly-express/ (zuletzt abgerufen am 10.12.2025)

Pratt, M. K. (2024, 1. Februar). Diese 18 Tools sollten Datenwissenschaftler kennen. ComputerWeekly.de. https://www.computerweekly.com/de/tipp/Diese-18-Tools-sollten-Datenwissenschaftler-kennen (zuletzt abgerufen am 22.10.2025)

Richardson, L. (o. D.). Beautiful Soup Documentation — Beautiful Soup 4.13.0 documentation. Crummy. https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (zuletzt abgerufen am 05.12.2025)

Ruktanonchai, N. W., Ruktanonchai, C. W., Floyd, J. R. & Tatem, A. J. (2018). Using Google Location History data to quantify fine-scale human mobility. International Journal of Health Geographics, 17(1), Article 28. https://doi.org/10.1186/s12942-018-0150-z

Schmidt, D. C. (2018). Google Data Collection [Research Paper, Vanderbilt University]. https://digitalcontentnext.org/wp-content/uploads/2018/08/DCN-Google-Data-Collection-Paper.pdf (zuletzt abgerufen am 05.12.2025)

Scrapy Developers. (2025, 2. Juli). Scrapy at a glance — Scrapy 2.13.3 documentation. Scrapy 2.13 Documentation. https://docs.scrapy.org/en/latest/intro/overview.html (zuletzt abgerufen am 05.12.2025)

Serere, H. N., Resch, B. & Havas, C. R. (2023). Enhanced geocoding precision for location inference of tweet text using spaCy, Nominatim and Google Maps. A comparative analysis of the influence of data selection. PLoS ONE, 18(3), e0282942. https://doi.org/10.1371/journal.pone.0282942

Smith, S. (o. D.). Architectural principles - .NET. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles (zuletzt abgerufen am 05.12.2025)

So laden Sie Ihre Google-Daten herunter - Google-Konto-Hilfe. (o. D.). https://support.google.com/accounts/answer/3024190?hl=de (zuletzt abgerufen am 05.12.2025)

Streamlit, A faster way to build and share data apps. (o. D.). https://streamlit.io/ (zuletzt abgerufen am 05.12.2025)

Taylor, B. (2005, 8. Februar). Mapping your way. Official Google Blog. https://googleblog.blogspot.com/2005/02/mapping-your-way.html (zuletzt abgerufen am 05.12.2025)

Wilke, C. O. (o. D.). Fundamentals of Data Visualization. https://clauswilke.com/dataviz/boxplots-violins.html (zuletzt abgerufen am 05.12.2025)

Zeng, Z. (o. D.). GitHub - zehengl/ez-address-parser: A parser for Canadian postal addresses. GitHub. https://github.com/zehengl/ez-address-parser (zuletzt abgerufen am 05.12.2025)

Zusätzliche Nutzungsbedingungen für Google Maps/Earth – Google. (o. D.). Google. https://www.google.com/intl/de_de/help/terms_maps/ (zuletzt abgerufen am 05.12.2025)

KI-Verzeichnis

Die folgenden KI-Systeme wurden bei der Erstellung dieser Arbeit verwendet:

Claude (Anthropic)

- Modell: Claude Sonnet 4.5
- Zeitraum: Oktober bis Januar 2025
- URL: https://claude.ai
- Anbieter: Anthropic PBC

Die Systeme wurden zur Unterstützung folgender Aufgaben eingesetzt:

1. Sprachliche Optimierung und wissenschaftliches Schreiben
   Verbesserung der Formulierungen, Eliminierung von Redundanzen, Präzisierung der wissenschaftlichen Ausdrucksweise in den schriftlichen Kapiteln.
2. Strukturierung und Konsistenzprüfung
   Überprüfung der logischen Gliederung, Identifikation von Inkonsistenzen und Wiederholungen in der schriftlichen Ausarbeitung.
3. Literaturrecherche
   Unterstützung bei der Suche nach Fachliteratur und wissenschaftlichen Quellen. Alle identifizierten Quellen wurden eigenständig auf Relevanz, Seriosität und Aktualität geprüft und verifiziert.
4. Technische Implementierung
   a. Vorschläge zur Strukturierung und Modularisierung der Systemarchitektur
   b. Debugging und Fehleranalyse von Code
   c. Recherche von Best Practices in der Softwareentwicklung
   d. Recherche von Lösungsansätzen, insbesondere bei Parsing- und Normalisierungsoptionen sowie Datenstrukturierung
   e. Erstellung von Docstrings für die Code-Dokumentation

Die Konzeption des Analysetools, die algorithmischen Entscheidungen, die finale Implementierung des Python-Codes, die Entwicklung der Analysemethoden, die Durchführung und Auswertung der funktionalen Tests sowie alle methodischen und inhaltlichen Entscheidungen erfolgten vollständig eigenständig. Die KI-Unterstützung bei der technischen Implementierung beschränkt sich auf Vorschläge zur Code-Strukturierung, Debugging-Hilfe und Recherche von Lösungsansätzen. Alle vorgeschlagenen Implementierungen wurden kritisch geprüft, an die spezifischen Anforderungen angepasst und eigenverantwortlich in das System integriert. Die finale Entscheidung über Architektur,

Algorithmen und Implementierungsdetails lag bei der Autorin. Bei der schriftlichen Ausarbeitung beschränkte sich die KI-Unterstützung auf sprachliche Optimierung und strukturelle Verbesserungen bereits eigenständig verfasster Textabschnitte. Alle durch KI-Tools generierten oder modifizierten Inhalte wurden kritisch geprüft, eigenverantwortlich überarbeitet und auf Richtigkeit, Relevanz und wissenschaftliche Qualität kontrolliert. Die finale Verantwortung für den gesamten Inhalt dieser Arbeit liegt bei der Autorin.

Anhang 1: Muster Einwilligungserklärung

Einwilligungserklärung zur Verarbeitung personenbezogener Daten gemäß DSGVO

Im Rahmen meiner Bachelorarbeit „Entwicklung eines Datenanalyse-Tools zur Synthese von durch Google Maps gesammelten Nutzerdaten“ bitte ich um Ihre Einwilligung zur Verwendung Ihrer Google Maps Aktivitätsdaten.

Zweck

Die Daten dienen zum Test und zur Validierung eines Analyse-Tools, welches Suchmuster auf Google Maps hinsichtlich ihrer Frequenz, Uhrzeit und thematischen Kategorien analysiert.

Datenumfang

Google Takeout Export der Google Maps Aktivitäten.

Dauer

Die Daten werden während der Testphase benötigt. Die Löschung erfolgt nach Abschluss der Arbeit.

Ich wurde darüber informiert, dass:

☐ meine Teilnahme freiwillig ist
☐ ich die Einwilligung jederzeit widerrufen kann
☐ alle Daten anonymisiert werden
☐ die Daten nur für wissenschaftliche Zwecke verwendet werden
☐ keine Weitergabe an Dritte erfolgt

_____ _____

(Ort, Datum) (Unterschrift)

Anhang 2: Link zum Repository

https://github.com/panipaq/BA-GoogleMapsData

Installation nach Klonen des Repositories

1. Optional: Virtual Environment erstellen

```
python -m venv venv

source venv/bin/activate  # Linux/Mac

venv\Scripts\activate     # Windows
```

2. Dependencies installieren

```
pip install -r requirements.txt
```

3. Google Maps API-Key hinterlegen (siehe Anleitung: API Einrichten bei Google)
   a. Im Projekt eine `.env`-Datei anlegen
   b. API-Key einfügen
   `GM_API_KEY = 'deinAPIkey'`
4. App ausführen

```
streamlit run app.py
```

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel:

Entwicklung eines Datenanalyse-Tools zur Synthese von durch Google Maps gesammelten Nutzerdaten.

selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Datum

Unterschrift