universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

"Evaluierung eines Natural Language Processing-gestützten Web-GIS für die visuelle Datenanalyse als Alternative zu traditionellen Interaktionsmöglichkeiten"

verfasst von / submitted by

Anna Strobl, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 856
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Kartographie und Geoinformation
Betreut von / Supervisor:	Ass.-Prof. Mag. Dr. Andreas Riedl

Danksagung

Ich möchte mich bei Ass.-Prof. Mag. Dr. Andreas Riedl für die Betreuung meiner Masterarbeit bedanken, sowie bei Gernot Pucher, meinem firmeninternen Betreuer.

Ein großer Dank gebührt auch der Firma Trafficon, die mir die Umsetzung dieser Arbeit ermöglicht haben. Insbesondere möchte ich mich bei den Mitarbeiter:innen bedanken, die die Implementierung des Web-GIS umgesetzt haben. Auch bedanke ich mich bei allen Teilnehmer:innen der Studie, die durch ihre Mitarbeit zu spannenden Ergebnissen beigetragen haben.

Ganz besonders möchte ich mich bei meinen Eltern bedanken, die mir das Studium ermöglicht haben, bei allem unterstützt haben und mit Rat und Tat beiseite gestanden sind.

Und abschließend bedanke ich mich auch besonders bei meinem Freund Johannes, der mich durch alle Phasen des Studiums und der Masterarbeit begleitet hat.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BERT	Bidirectional Encoder Representation
DL	Deep Learning
DTM	Dynamic Traffic Monitor
EDA	Explorative Datenanalyse
FCD	Floating Car Data
GIS	Geoinformationssystem
HCI	Human Computer Interaction
IE	Information Extraction
IR	Information Retrieval
KI	Künstliche Intelligenz
LOS	Level of Service
ML	Machine Learning
NEM	Named Entity Matching
NER	Named Entity Recognition
NLP	Natural Language Processing
NN	Neuronale Netzwerke
OSM	Open Street Map
PTM	Pre-Trained Transformer Model
RoBERTa	Robustly Optimized BERT Pre-Training Approach
UI	User Interface

Kurzfassung

Immer mehr Anwendungen zur Analyse von räumlichen Daten werden im öffentlichen und privaten Bereich eingesetzt. Dadurch wird auch die Gruppe an User:innen immer diverser. Es handelt sich hier in vielen Fällen um Personen, die Fachexperten einer anderen Richtung sind, aber nur wenig oder keine Erfahrung im Umgang mit Geoinformationssystemen haben. Bei den Anwendungen handelt es sich oft um Web-GIS, die eine moderne und flexible Weiterentwicklung des Desktop-GIS darstellen. Diese Anwendungen können komplex in der Benutzung sein. Hier sollen Web-GIS, die durch Natural Language Processing gestützt sind, eine innovative und intuitive Alternative bieten.

Ziel dieser Masterarbeit ist es, zu evaluieren, ob ein textgesteuertes Web-GIS eine Alternative zu einem traditionellen, buttongesteuerten Web-GIS darstellt. Dazu wird folgende Forschungsfrage gestellt: „Sind signifikante Unterschiede in identifizierbaren Metriken zwischen einem buttongesteuerten und einem textgesteuerten Interface eines Web-GIS erkennbar?“. Die Forschungsfrage und weitere Arbeitsfragen werden durch das folgende Vorgehen beantwortet: Anfangs wird eine umfassende Literaturrecherche durchgeführt, auf die alle weiteren Vorgehensweisen gestützt werden. Weiters werden zwei Deep Learning Modelle trainiert, durch die die Verarbeitung des Input-Textes im für die Arbeit verwendete Web-GIS vorgenommen wird. Das Web-GIS, das primär an Experten in der Verkehrsplanung gerichtet ist, ermöglicht die Analyse von Verkehrsdaten in München. Durch das abschließend durchgeführte Usability Testing soll herausgefunden werden, ob das textgesteuerte Web-GIS für User:innen eine Alternative zum buttongesteuerten Interface darstellt. Durch die Erörterung kann gezeigt werden, dass, während noch weiterer Forschungsbedarf besteht und eine Weiterentwicklung des Web-GIS notwendig ist, dieses für User:innen eine mögliche Alternative darstellt.

Abstract

More and more applications for the analysis of spatial data are used in the public and private sector. As a result, the group of users is becoming more diverse. These are often people, who are experts in a different field, but have little to no experience in geoinformationsystems. The applications are often web GIS, which are a modern and flexible evolution of desktop GIS. These applications can often be complex to use. Here, web GIS powered by Natural Language Processing can be an innovative and intuitive alternative.

The goal of this master thesis is to evaluate, if a text-driven web GIS is an alternative to a traditional, button-driven web GIS. Therefore, the following research question is posed: „Are there significant differences in identifiable metrics between a button-driven and text-driven interface of a web GIS?“. The research question and other working questions are answered by the following procedure: In the beginning, a comprehensive literature review is conducted, on which all further procedure is being based. Furthermore, two deep learning models are trained, through which the processing of the input text in the Web-GIS used for the work is performed. The web GIS enables the analysis of traffic data in Munich, which is primarily addressed to experts in traffic planning. The usability testing, that is conducted finally, aims to find out whether the text-driven Web-GIS is an alternative to the button-driven interface for users. Through the discussion it can be shown that while further research and development of the web GIS is needed, it is a possible alternative for users.

1. Einleitung

Sowohl im privaten als auch im öffentlichen Sektor ist der Stellenwert der Analyse und Visualisierung von räumlichen Daten in den vergangenen Jahren gestiegen, und somit auch die Verwendung von Geoinformationssystemen (GIS). Heutzutage kommen hierbei oft keine traditionellen Desktop-basierten Systeme zum Einsatz, sondern Web-GIS, die über das Internet abrufbar sind. Diese ermöglichen, wie auch ein Desktop-GIS die Analyse von Daten, und bieten weiters auch die Möglichkeit einer unkomplizierten Verbreitung der Ergebnisse, sowie einen flexiblen Zugriff auf die Anwendung. Web-GIS stellen somit eine moderne und zukunftsorientierte Form von Geoinformationssystemen dar. (vgl. ZHAODI/DAN 2022: 2)

Die Verwendung von Web-GIS und die Durchführung von Analysen kann jedoch oft komplex, und dadurch mit einem hohen Zeit- und Kostenaufwand verbunden sein. Dies kann den Zugang zu diesen Systemen für fachfremde Personen und Experten aus anderen Fachrichtungen erschweren. Aufgrund dieser Herausforderungen werden Alternativen zu traditionellen buttongesteuerten Web-GIS gesucht. Eine Möglichkeit sind hier textbasierte Anwendungen, die auf Natural Language Processing (NLP) beruhen. (vgl. CALÌ ET AL. 2011: 921f)

Natural Language Processing (dt.: natürliche Sprachprozessierung) kann als die Interpretation von Text und Gesprochenem mittels Computer definiert werden. (vgl. ZHANG/TENG 2021: 3)

Die textbasierte Interaktion zwischen Menschen und Computer durch Natural Language Processing kann in der Handhabung intuitiver als die Interaktion mit traditionellen, buttongesteuerte Web-GIS sein. Durch die Verwendung von Sprache kann die Kommunikation transparenter, natürlicher und dialogähnlich stattfinden, und Anwendungen somit zugänglicher gestaltet werden. Weiterführend können dadurch Herausforderungen wie die Komplexität der Verwendung von Web-GIS reduziert, und somit etwa kostenschwere Schulungen vermieden werden. Dies stellt nur einen Ausschnitt aus den möglichen Vorzügen eines textgesteuerten Web-GIS dar. (vgl. CALÌ ET AL. 2011: 921f, 925)

1.1. Stand der Forschung

Das Feld des Natural Language Processing zur Verarbeitung von räumlichen Informationen ist durch seine Relevanz und mögliche Potenziale ein aktives Forschungsfeld. Es bestehen diverse Studien, die sich mit den Ausprägungen dieses wissenschaftlichen Feldes beschäftigen. (vgl. LAMPOLTSHAMMER/HEISTRACHER 2012: 81f)

In einigen Artikeln wird die Kombination von Natural Language Processing und räumlichen Informationen im Allgemeinen untersucht, so auch bei MCKENZIE/ADAMS 2021. Dabei wird

primär auf die Grundlagen und die unterschiedlichen Felder und Anwendungen von NLP in Verbindung mit der Geoinformatik eingegangen. Ein großer Teil der bestehenden Studien behandelt hier einzelne Teilgebiete der Fachrichtung. Diese Studien sind von Bedeutung, da sie die Grundlagen und die Herausforderungen des Natural Language Processings von raumbezogenen Informationen aufarbeiten und somit einen wichtigen Beitrag zu Forschung leisten, auf den auch in der Umsetzung dieser Arbeit zurückgegriffen wird.

In Bezug auf diese Masterarbeit sind insbesondere Studien relevant, die sich mit der Integration von Natural Language Processing in ein GIS oder Web-GIS beschäftigen.

Eine dieser Studien stammt von CALÌ ET AL. 2011. Es wurde untersucht, inwiefern ein textgesteuertes Interface eines Geoinformationssystems eine Alternative zu einem buttongesteuerten Interface in Hinblick auf die Interaktion zwischen Menschen und Computer darstellt. Das textgesteuerte GIS gleicht hier einem Chatbot, der räumliche Fragen beantwortet. In einer User-Studie wurden verschiedene Aufgaben von Testpersonen in einem buttongesteuerten und dem textgesteuerten GIS gelöst werden. Die Performance wurde basierend auf Parametern wie der Umsetzungsdauer der gegebenen Aufgaben und der Korrektheit der Ergebnisse evaluiert. Die Ergebnisse der User-Studie zeigten positive Resultate, und CALÌ ET AL. 2011 konnten somit das Potenzial von textbasierten GIS darstellen.

Eine weitere Studie stammt von YAQOT/ALBASEER 2018. Darin wurde ein textbasiertes Web-GIS entwickelt. Hier handelt es sich um eine sehr simple Anwendung, die es ermöglicht, durch eine Texteingabe Tweets, die mit deren Inhalt korreliert, zu finden. Diese werden wiederum in einer Karte dargestellt, und deren Polarität analysiert.

SETLUR ET AL. 2016 entwickelten ein textgesteuertes Web-GIS, das für die visuelle Analyse von Geodaten verwendet werden kann, und somit auch dem in dieser Arbeit verwendeten Web-GIS ähnelt. In der Anwendung ist eine anfängliche Eingabe mittels Text erforderlich, anschließend kann diese jedoch durch Buttons konkretisiert oder verändert werden. Die Autorinnen und Autoren des Artikels sehen Natural Language Processing nicht als eine ganzheitliche Lösung für ein Web-GIS und verfolgen einen kombinierten Ansatz zwischen einem textbasierten und buttongesteuerten Interface.

Aus den bestehenden Studien ergibt sich eine Forschungslücke, in der mit dieser Masterarbeit angesetzt werden soll. Die Versuche, ein textbasiertes Interface für ein Desktop-GIS umzusetzen bestehen bereits, wie CALÌ ET AL. 2011 zeigt. Ebenso existieren Versuche, Natural Language Processing in ein Web-GIS zu integrieren. Die Anwendungen sind hier jedoch primär für die Analyse von Tweets oder andere Aufgaben geeignet. In der Studie von SETLUR ET AL. 2016 ist das

Web-GIS zwar für die visuelle Datenanalyse konzipiert, jedoch ist das Interface zur Eingabe der Fragestellung nicht gänzlich textbasiert. Während die Anzahl an Studien im Forschungsfeld im Gesamten noch nicht hoch ist, fokussieren sich die bestehenden Studien zusammengefasst entweder auf Desktop-GIS, andere Aufgabenstellungen als die visuelle Datenanalyse oder multimodale Interfaces.

Zum Zeitpunkt der Erstellung der Disposition bestehen keine Studien, in denen ein textbasiertes Web-GIS zur visuellen Analyse von räumlichen Daten umgesetzt wurde. Dies stellt folglich den innovativen Beitrag dieser Masterarbeit zum heutigen Stand der Forschung dar, wodurch dieser auch erweitert werden soll.

1.2. Zielsetzung

Die übergeordnete Zielsetzung der Masterarbeit ist es, ein für den Anwendungsfall geeignetes Deep Learning (DL) Modell zu identifizieren, dieses in ein bestehendes Web-GIS für Verkehrsmonitoring einzubinden und das Potenzial der textgesteuerten Anwendung für die visuelle Datenanalyse durch ein Usability Testing zu evaluieren. Folgend sind die konkreten Ziele dieser Masterarbeit aufgelistet:

- Darstellen des Forschungsstands in der natürlichen Sprachprozessierung, insbesondere im Zusammenhang mit räumlichen Daten
- Tuning und Training eines bestehenden DL-Modells für ein Web-GIS zur visuellen Datenanalyse
- Evaluation der Performance des DL-Modells
- Implementierung des DL-Modells in ein bestehendes Web-GIS
- Identifikation geeigneter Metriken der Human Computer Interaction (HCI) für ein Usability Testing des Web-GIS
- Evaluierung des textgesteuerten Web-GIS durch ein Usability Testing im Vergleich zu einem traditionellen, buttongesteuerten Interface durch Methoden der Human Computer Interaction

1.3. Forschungsfragen

Die Forschungsfrage dieser Masterarbeit lautet:

„Sind signifikante Unterschiede in identifizierbaren Metriken zwischen einem buttongesteuerten und einem textgesteuerten Interface eines Web-GIS erkennbar?“.

Weiters wurden folgende Arbeitsfragen formuliert, um die Beantwortung der Forschungsfrage zu unterstützen:

1. „Was ist der aktuelle Forschungsstand im Natural Language Processing, besonders im Zusammenhang mit räumlichen textbasierten Daten, und wo bestehen Herausforderungen und Mängel in der Technologie?“
2. „Welche bestehenden Deep Learning Modelle eignen sich zur Verarbeitung von textbasierten Informationen mit Raumbezug in einem Web-GIS, bezogen auf den definierten Use-Case?
3. „Welche etablierten Metriken der Human Computer Interaction eignen sich zur Evaluierung der Ergebnisse des Usability Testing von textbasierten Anfragen in einem Web-GIS?“

1.4. Methodik

Um die Ziele der Masterarbeit zu erreichen, werden diverse theoretische und praktische Methodiken gewählt. Am Anfang steht eine umfassende Literaturrecherche, um den Stand der Forschung zu erörtern und die weitere praxisbezogene Vorgehensweise darauf stützen zu können. Im nächsten Schritt werden für den Use-Case passende Deep Learning Modelle verglichen, um das für den Anwendungsfall am besten geeignete Modell selektieren zu können. Dieses wird anschließend trainiert und getuned, und somit auf den Use-Case abgestimmt. Das angelernte Modell wird in ein bestehendes Web-GIS der Firma Trafficon implementiert, und weiters auf ein textgesteuertes Interface umgebaut. Dabei wird das User-Interface selbst konzipiert, die Umsetzung des Backends jedoch vom Entwicklerteam der Anwendung übernommen. Um das Modell zu evaluieren, wird ein Usability Testing mit dem textbasierten Interface im Vergleich zu dem ursprünglichen Interface durchgeführt, bei dem verschiedene Aufgaben in beiden Anwendungen von den Testpersonen umgesetzt werden müssen. Die Studie wird auf geeignete Metriken der Human Computer Interaction gestützt, die in der Literaturrecherche identifiziert werden. Die Studie soll in einem überwachten Setting durchgeführt werden. Es werden je nach Möglichkeit fünf bis zehn Testpersonen herangezogen, die Experten in der Verkehrsplanung sind. Generell ist das Tool ein Experten-Tool, das nicht für User:innen der Öffentlichkeit, sondern speziell für Fachpersonen der Verkehrsplanung und des Verkehrsmanagements vorgesehen ist.

2. Stand der Forschung

Das folgende Kapitel soll einen Überblick über sämtliche theoretische Grundlagen dieser Masterarbeit geben. Anfangs wird auf die Grundlagen der Geoinformatik und die Definition, Aufbau und Vor- und Nachteile von Web-GIS eingegangen. Darauffolgend werden die Human Computer Interaction und Methodiken des Usability Testings erläutert. Weiters werden die theoretischen Konzepte des Natural Language Processing dargestellt, und im spezielleren auch der Fokus auf die Verbindung von Natural Language Processing und räumlichen Informationen gelegt. Abschließend wird in die Grundlagen des Machine Learning (ML) und Deep Learning eingeführt, im spezielleren auch auf die Anwendung im Natural Language Processing.

2.1. Geoinformatik und Web-GIS

Das folgende Kapitel soll die Grundlagen und Funktionen von Web-GIS erläutern, und deren Vor- und Nachteile gegenüber Desktop-GIS herausarbeiten.

Das wissenschaftliche Feld, in dem Web-GIS verortet werden können, ist die Geoinformatik. Die Geoinformatik ist ein interdisziplinäres Feld, das zwischen der Informatik, den Geowissenschaften und anderen raumbezogenen Wissenschaften wie den Umweltwissenschaften und der Geografie angesiedelt werden kann. (vgl. DE LANGE 2020: 1)

Definiert werden kann die Geoinformatik wie folgt:

> *Die Geoinformatik widmet sich der Entwicklung und Anwendung von Methoden und Konzepten der Informatik zur Lösung raumbezogener Fragestellungen unter besonderer Berücksichtigung des räumlichen Bezugs von Informationen. Die Geoinformatik beschäftigt sich mit der Erhebung oder Beschaffung, mit der Modellierung, mit der Aufbereitung und vor allem mit der Analyse sowie mit der Präsentation und der Verbreitung von Geodaten. (DE LANGE 2020: 4)*

Der Stellenwert der Geoinformatik ist im privaten sowie im öffentlichen Bereich stetig wachsend. Das Feld ermöglicht die Darstellung der räumlichen Beziehungen zwischen Menschen, der Umwelt, Wirtschaft und Kultur. Dadurch können Entscheidungen, die Raumbezug haben, informiert getroffen werden. (vgl. MCHAFFIE 2019: 1)

2.1.1. Grundlagen von Geoinformationssystemen

Das zentrale Werkzeug der Geoinformatik ist das Geoinformationssystem (GIS). (vgl. DE LANGE 2020: 1) Dieses kann wie folgt definiert werden:

> *Ein Geoinformationssystem ist ein rechnergestütztes System, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst, gespeichert, verwaltet, aktualisiert, analysiert und modelliert sowie alphanumerisch und graphisch präsentiert werden. (DE LANGE 2020: 375)*

Das Geoinformationssystem kann mit den Vier-Komponenten-Modellen dargestellt werden. Es bestehen zwei Modelle: das HSDA-Modell, und das EVAP-Modell. (vgl. DE LANGE 2020: 373ff)

Ersteres, das HSDA-Modell, beschreibt die strukturellen Komponenten eines Geoinformationssystems:

- Hardware

- Software
- Daten
- Anwender

(vgl. ebd.)

Das EVAP-Modell definiert die funktionalen Komponenten eines GIS. Es ermöglicht die:

- Erfassung
- Verwaltung
- Analyse
- Präsentation

von Daten und Informationen. Je nach Anwendung und System verschwimmen die genannten Funktionen, und sind unterschiedliche stark ausgeprägt und gewichtet. Nach strenger Definition muss jedes Geoinformationssystem alle diese Komponenten und Funktionen abdecken, um als solches bezeichnet werden zu können. Fehlt beispielsweise die Möglichkeit zur Datenaufnahme, kann das System als Auskunftssystem bezeichnet werden. (vgl. ebd.)

Die Anfänge der Geoinformationssysteme liegen in den 1960ern. Zu dieser Zeit wurden die ersten Systeme entwickelt. Bis etwa 1990 übernahmen Behörden und Firmen Geoinformationssysteme in deren Prozesse, bis 1988 auch Anwender:innen in die Entwicklung miteinbezogen wurden. Ende der 1990er begann der Austausch von Daten und Produkten am offenen Markt. Die letzte und auch aktuelle Phase der Entwicklung von Geoinformationssystemen hat 2010 begonnen. Seitdem sind die Produkte auch ubiquitär und auf Smartphones und Tablets vorhanden. (vgl. DE LANGE 2020: 2)

2.1.2. Web-GIS und Web-Mapping

Ein weiteres und bei weitem neueres Werkzeug der Geoinformatik ist das Web-GIS. Einfach definiert ist ein Web-GIS ein Geoinformationssystem, das das Internet als Plattform verwendet. Streng gesehen muss auch ein Web-GIS alle Funktionen des EVAP-Modells abdecken, um als solches bezeichnet werden zu können. In der Praxis ist dies jedoch oft weniger eindeutig abgegrenzt. (vgl. DE LANGE 2020: 381f)

Weiters bestehen sogenannte Web-Mapping Systeme. Oft sind diese Systeme rein zur Visualisierung von Geodaten gedacht. Diese Anwendungen sind meist einfachere und kostengünstigere Alternativen zu einem Web-GIS. (vgl. DE LANGE 2020: 383f)

Web-Mapping Anwendungen unterscheiden sich grundsätzlich durch ihre Funktionen und Möglichkeiten von einem Web-GIS. Web-GIS beinhalten meist viel umfangreichere Funktionen und ermöglichen ebenso die Abfrage von Datenbanken. Weiters muss in einem Web-GIS mindestens eine zusätzliche Analyse-Funktion integriert sein, wie die Verschneidung von Geometrien oder etwa die Möglichkeit zum Routing. (vgl. DE LANGE 2020: 381ff)

Die Unterscheidung zwischen Web-GIS und Web-Mapping verschwimmen in der Praxis meist, und sind nicht klar voneinander abgetrennt. Oft werden auch Systeme, die rein zur Visualisierung von Geodaten im Internet dienen als Web-GIS bezeichnet. (vgl. DE LANGE 2020: 383)

2.1.2.1. Entwicklung von Web-GIS

Die Geschichte des Web-GIS beginnt 1989 mit der Erfindung des Internets. Im Jahr 1993 wurden die ersten statischen Karten online veröffentlicht, 1996 wurde das erste Web-GIS entwickelt. Seitdem unterliegen Web-GIS stetiger Veränderung und Weiterentwicklung. Über die Zeit wurden die Anwendungen immer dynamischer und interaktiver. Zwischen 2000 und 2010 gewannen kollaborative Web-GIS wie „Open Street Map" an Beliebtheit, und Ende der 2010er wurde der Fokus auf die Verwendung von Mobilgeräten gelegt. Bis etwa 2015 wurden besonders Cloud-Softwares weiterentwickelt, wodurch die Anwendungen zugänglicher und skalierbarer wurden. In der heutigen Zeit liegt der Fokus bei der Weiterentwicklung von Web-GIS darauf, die Interaktion zwischen Menschen und Computer zu verbessern. Es sollen maßgeschneiderte Anwendungen entwickelt werden, die einen möglichst großen Mehrwert für User:innen generieren. (vgl. VEENENDAAL ET AL.: 2017: 7-15)

2.1.2.2. Architekturen von Web-GIS

Die Architektur einer Anwendung beschreibt das Grundgerüst und die Struktur dieser und die Mechanismen, wie dieses funktioniert. Web-GIS können auf verschiedenen Architekturen aufgebaut sein. (vgl. SHOAB ET AL. 2017: 10f)

Die grundsätzliche Architektur von Web-GIS basiert auf einer Client-Server Architektur. Client-Server-Architekturen haben zwei Bestandteile: Server und Clients. Ersterer sind Rechner, die Dienstleistungen anbieten, zweiteres Rechner, die wiederum diese Leistungen anfragen. (vgl. DE LANGE 2020: 42f)

Hier kann weiterfolgend zwischen der „Thin-Client" und „Thick-Client" Architektur unterschieden werden. (vgl. KULAWIAK ET AL. 2019: 31) Abbildung 1 zeigt beide Architekturen.

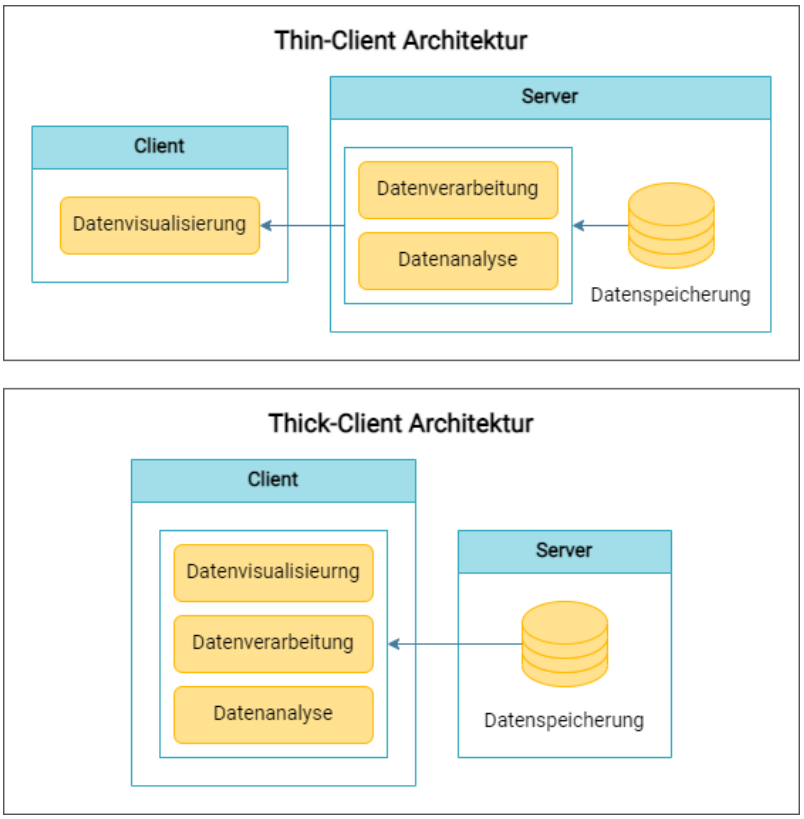

Abbildung 1: Thin-Client Architektur (Oben) und Thick-Client Architektur (Unten) eines Web-GIS (Basierend auf: KULAWIAK ET AL. 2019: 31)

Typischerweise erfolgt die Geoprozessierung, also die Verarbeitung und Analyse der Daten bei einem Web-GIS auf der Serverseite. Bei der Thin-Client Architektur erfolgen auf dem Client selbst nur die Kommunikation mit dem Server, und die visuelle Darstellung der Ergebnisse. Diese Web-GIS sind vor allem für einfache Analysen und Anwendungen geeignet. Genauso kommuniziert der Client bei der Thick-Client Architektur mit dem Server, jedoch werden durch Erweiterungen wie Plug-ins oder JavaScript die Geoprozessierung auf der Client-Seite ermöglicht. Dadurch können auch umfangreichere Daten verwendet und komplexere Analysen durchgeführt werden. Eine Zwischenlösung zwischen den beiden Architekturen ist die Medium-Client Architektur, bei der die Prozessierung auf Client und Server umgesetzt wird. (vgl. DE LANGE 2020: 382; SHOAB ET AL. 2017: 11ff)

2.1.2.3. Vorteile von Web-GIS

Die Verwendung eines Web-GIS hat gegenüber einem Desktop-GIS diverse Vorteile. Einer der positiven Aspekte ist, dass keine eigene Software auf dem Computer der User:innen installiert und aufgesetzt werden, und weiters konfiguriert werden muss. Die Anwendung ist also grundsätzlich unabhängig von Geräten, und kann dadurch auch leichter auf Tablets oder Ähnlichem verwendet werden. (vgl. KERSKI/BAKER 2019: 4ff)

Ein weiterer Vorteil eines Web-GIS ist, dass es einfach über den Browser abgerufen werden kann, was grundsätzlich eine für viele Personen bekannte Umgebung darstellt. Wenn Personen eine für sie neue Anwendung verwenden, kann es einen positiven Einfluss haben, wenn dies auf einer bereits geläufigen Plattform möglich ist. (vgl. KERSKI/BAKER 2019: 7)

Auch erleichtert ein Web-GIS die Einbindung von Daten. Anstatt, dass User:innen die Daten selbst in die Anwendung laden müssen, wie bei einem Desktop-GIS, sind diese einfach durch eine Datenbankanbindung abrufbar. Ein weiterer, wenn auch nur weniger relevant erscheinender Vorteil ist die Möglichkeit von voreingestellten Basemaps. Dadurch kann direkt zu Beginn der Verwendung des Web-GIS den User:innen ein Überblick über den räumlichen Rahmen gegeben werden. (vgl. ebd.)

2.1.3. Visuelle Analyse von räumlichen Daten

Ein weiterer relevanter Bereich in der Geoinformatik für diese Masterarbeit sind visuelle Analysen von Daten, sowie die Werkzeuge zur Erstellung dieser Analysen. Dieses Feld kann auch „Visual Analytics" genannt werden. Ziel der Visual Analytics ist, Tools zur Analyse und Interpretation von räumlichen Daten zu erstellen. Dadurch soll User:innen ermöglicht werden, ein besseres Verständnis der vorliegenden Daten zu erlangen, und dadurch auch neue Ergebnisse zu erhalten. Die Methoden sind dabei interaktive Visualisierungen und Datenanalysen, deren Durchführung automatisiert funktioniert. Die Werkzeuge können dabei beispielsweise GIS und Web-GIS sein. (vgl. DRANSCH ET AL. 2019: 21f)

Die grundlegende Methode der Visual Analytics ist die explorative Datenanalyse (EDA). Zusätzlich zur deskriptiven Datenanalyse, die sich auf die Beschreibung von Daten mittels statistischer Kennwerte fokussiert, sucht die explorative Datenanalyse nach Mustern und Zusammenhängen. Dadurch können weiterführend Hypothesen und Fragestellung zur weiteren Untersuchung der Daten formuliert werden. Werden die Ergebnisse dieser Analysen graphisch durch Karten und Diagramme aufbereitet, kann ein tiefergehendes und umfassendes Verständnis für die Daten und Ergebnisse der Analysen ermöglicht werden. Weiterführend hilft dies bei der Konzeption und Umsetzung von Maßnahmen, basierend auf der Analyse. Dies stellt die Vorteile der visuellen Analyse dar. (vgl. DRANSCH ET AL. 2019: 24)

2.2. Human Computer Interaction

Die Human Computer Interaction (HCI) ist ein wissenschaftliches Feld, das sich auf das Design von Technologie wie Softwares und Computern, und die Interaktion zwischen Menschen und diesen Systemen fokussiert. Dabei wird vor allem auch die Erfahrung, die User:innen bei dieser Interaktion haben, die sogenannte „User Experience", beachtet. Ziel der Human Computer Interaction ist es, Technologien von Menschen für Menschen zu designen, um diese möglichst wertvoll und angenehm für User:innen zu gestalten. Die Human Computer Interaction ist ein interdisziplinäres Feld, das in einer Vielzahl von unterschiedlichen Wissenschaften zu verorten ist. So ist sie ein Teilgebiet der Computer Sciences, von Design und Media, der Verhaltenspsychologie und vielen weiteren wissenschaftlichen Feldern. (vgl. BECKER 2020: 10ff)

Das folgende Kapitel geht auf die Grundkonzepte der Human Computer Interaction ein. Konkreter werden Methodiken der Usability und des Usability Testings vorgestellt und erläutert.

2.2.1. Human Computer Interaction und Usability

Menschen und Technologie sind keine abgetrennten Systeme, die ohne jeglichen Einfluss aufeinander existieren, sondern sind eng miteinander verbunden. Menschen sind durch Gewohnheiten geprägt. Diese Gewohnheiten steuern das Verhalten, welches sich wiederum in den Softwares abbildet, die entwickelt werden. Gleichzeitig verändern sich Technologien, und dadurch auch unser Verhalten. Dabei kann dieser Prozess reaktiv oder proaktiv passieren. Einerseits müssen HCI-Designer mit den sich schnell ändernden Technologie mithalten, andererseits können die Designer jedoch auch mit neuen Anwendungen, die sie erfinden, die Technologie weiterentwickeln. Dies stellt die Grundlage der Human Computer Interaction dar. (vgl. BECKER 2020: 50)

Der Prozess des Designs und der Erstellung einer Software ist kein statischer. Werden User:innen im Prozess mitbedacht, ist dieser dynamisch. Es muss mit Entwicklungen, wie neuen Modalitäten wie Tablets oder Smartphones, aber auch beispielsweise neuen Interaktionsmöglichkeiten wie der Stimme oder Gestik, mithalten. Die Prozessschritte der Human Computer Interaction sind:

1. Designen (design)
2. Erstellen (build)
3. Testen (test)

Um eine gute Software oder Anwendung zu erstellen, muss dieser Prozess iterativ erfolgen. In jedem der Schritte kann es zu Problemen kommen. Menschen machen Fehler oder treffen falsche Entscheidungen. Wiederholt man diese Schritte, lernt man aus Fehlern und kann diese ausbessern. Dies führt zu einem dynamischen Prozess. (vgl. BECKER 2020: 35-39)

Eines der Grundkonzepte der Human Computer Interaction ist die „Usability". Es bestehen auch weitere Konzepte, welche jedoch für diese Arbeit nicht relevant sind. Die Usability kann laut ISO 9241-11 definiert werden als:

> *„extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"* (ISO 2018)

In der Definition werden drei Stichworte erwähnt:

- Effektivität (effectivness)
- Effizienz (efficiency)
- Zufriedenheit (satisfaction)

(vgl. DE BLEECKER/OKOROJI 2018: 5f)

Die ersten beiden Stichwörter beschreiben simpel gesagt wie gut User:innen ihre Ziele in einer Anwendung erreichen können, und mit wie viel Aufwand sie diese erreichen. Hierzu zählen vor allem die Korrektheit, Genauigkeit und Geschwindigkeit der Lösung der Aufgaben. Es geht primär darum, ob die Anwendung für User:innen einen Mehrwert generiert. Ein Konzept, das über die Effektivität und Effizienz hinaus geht, ist die Zufriedenheit. Auch wenn die anderen Faktoren einen Einfluss auf diese haben, steht hier die Wahrnehmung von User:innen im Fokus. Wichtig ist dabei, ob die Anwendung für User:innen angenehm zu verwenden ist, und ob diese eine positive Erfahrung bei der Verwendung haben. Dies ist auch der Grund, wieso die Usability nicht nur auf objektiven, sondern auch subjektiven Faktoren basiert. (vgl. BARNUM 2011: 11f; DE BLEECKER/OKOROJI 2018: 5f)

Zentral bei der Usability von Softwares ist das „User Interface" (UI). Dieses stellt die Oberfläche und alle Komponenten dar, über die User:innen mit einem System kommunizieren und dieses kontrollieren können, um Aufgaben mit interaktiven System zu lösen. (vgl. ISO 2020)

2.2.2. Usability Testing

Um Software geeignet für User:innen zu entwickeln, müssen diese auch in den Prozess miteinbezogen werden. Das Ziel sollte nicht sein, dass Personen denken wie die Software, um damit Aufgabenstellungen zu lösen, sondern Softwares so zu gestalten, dass User:innen die Aufgabenstellungen damit möglichst einfach und schnell lösen können. (vgl. BECKER 2020: 10ff)

Eine Möglichkeit, dies umzusetzen ist ein „Usability Test" bzw. das „Usability Testing". Dies ist eine formale Art, um zu überprüfen, ob die drei Faktoren der Usability (Effektivität, Effizienz und Zufriedenheit) in der Anwendung erfolgreich umgesetzt wurden. Beim Usability Testing werden

meist User:innen bei der Lösung von relevanten Aufgabenstellungen in der Anwendung beobachtet. Die User:innen repräsentieren grundsätzlich die gewünschte Zielgruppe. Durch die Überwachung und Beobachtung können diverse Daten und Informationen gesammelt werden. (vgl. BARNUM 2011: 13ff; DE BLEECKER/OKOROJI 2018: 5f)

Es bestehen zwei grundsätzliche Formen des Usability Testings:

- Formatives Testing
- Summatives Testing

Ersteres sind meist kleine Studien, die während der Produktentwicklung durchgeführt werden, um Fehler zu diagnostizieren und ausbessern zu können, wobei hier oft nur wenige Personen miteinbezogen werden. Die zweite Möglichkeit wird nach der Fertigstellung eines Produktes gemacht, um zu evaluieren, dass dieses den bestehenden Voraussetzungen entspricht. Hier werden meist mehr Testpersonen verwendet. (vgl. BARNUM 2011: 14)

Dieses Kapitel soll die Bestandteile und den Ablauf der Planung eines Usability Testing beschreiben. In der Planung eines Usability Tests müssen diverse Aspekte beachtet werden:

- Studienziel
- Studienform
- Studienteilnehmer:innen
- Tasks
- Metriken

(vgl. BARNUM 2011: 107)

2.2.2.1. Studienziel

In der Planung der als erstes zu beachtende Punkt sind die Studienziele. Diese müssen für eine erfolgreiche Durchführung einer Studie festgelegt werden. Meist möchte man so viele Informationen wie möglich aus der Studie erlangen, jedoch ist dies nur selten möglich. Deswegen müssen anfangs konkrete Ziele festgelegt werden, auf die man sich fokussieren kann. (vgl. BARNUM 2011: 107)

2.2.2.2. Studienform

Es bestehen unterschiedliche Methoden des Usability Testings, bezogen auf das Setting und die Form der Studie. Die Formen des Testings können auf mehrere Arten unterschieden werden. Eine

der Unterscheidungen ist zwischen den Formen „Remote“ und „In-Person“ zu treffen. (vgl. DE BLEECKER/OKOROJI 2018: 6f)

Bei der In-Person Studie wird in einem dafür vorgesehenen Raum getestet, wie etwa einem Labor. Es können jedoch auch informale Räume wie ein Büroraum verwendet werden. Eine Studie in einem dedizierten Raum erfordert eine Ausrüstung. Vorhanden sein muss mindestens ein Laptop oder Computer, an dem Studienteilnehmer:innen die Aufgabenstellungen durchführen können. Weiteres Equipment können beispielsweise Kameras und Mikrophone sein. In einem informalen Raum kann ein Studiensetting mit relativ wenig Kosten- und Zeitaufwand durchgeführt werden. Der Vorteil der Verwendung eines dedizierten Raums ist, dass hier optimale Bedingungen geschaffen werden können. (vgl. BARNUM 2011: 27ff)

Die zweite Möglichkeit des In-Person Testings ist eine Feldstudie. Dafür muss je nach Bedarf nur eine kleine Ausstattung von einem Laptop für sich selbst, und teilweise für die Studienteilnehmer:innen, sowie eine Kamera und ein Mikrofon mitgenommen werden. Der Vorteil dieser Art der Studie ist, dass Personen in einem natürlichen Umfeld beobachtet können, dass nicht perfekt ist, sondern real. Dadurch kann der Einfluss des Umfelds wie Störfaktoren in die Entwicklung miteinbezogen werden. Gleichzeitig kann dies jedoch ein Nachteil sein, da Faktoren wie Lärm auch einen negativen Einfluss auf eine Studie haben können. (vgl. BARNUM 2011: 38ff)

Eine weitere Form des Usability Testings ist die Remote Studie. Diese kann in moderierter und unmoderierter Form durchgeführt werden. Das Remote Testing hat sich besonders in den letzten Jahren an wachsender Beliebtheit erfreut. Dieser Aufschwung ist vor allem neuen Techniken und Tools zu verdanken. (vgl. BARNUM 2011: 38ff; DE BLEECKER/OKOROJI 2018: 7f)

Das Remote Usability Testing hat einige Vor- und Nachteile. Es ermöglicht den Studienleiter:innen, Einsicht darin zu bekommen, wie die Studienteilnehmer:innen das Produkt in ihrer natürlichen Umgebung verwenden. Die Personen lösen die Aufgabenstellungen hier mit dem eigenen Computer oder Laptop. Weiters ermöglicht es die Remote Studie, Personen zu erreichen, die ansonsten durch hohen Kosten- und Zeitaufwand, der durch eine Anreise zu einer In-Person Studie entstehen kann, nicht miteinbezogen werden könnten. Dadurch kann auch eine höhere Anzahl an Studienteilnehmer:innen rekrutiert werden. Auch ist diese Form des Testings von Vorteil, wenn sich die Termine über mehrere Tage ziehen. (vgl. BARNUM 2011: 42; DE BLEECKER/ OKOROJI 2018: 8-11)

Genauso bestehen jedoch Nachteile bei der Durchführung einer Remote Studie. Eine der Herausforderungen kann die Installation und das Aufsetzen der Anwendung auf dem Computer der Studienteilnehmer:innen sein. Hier können Schwierigkeiten abhängig von unterschiedlichen

Geräten oder Einstellungen entstehen, die die Durchführung der Studie erschweren. Genauso können Probleme entstehen, wenn eine schlechte Internetverbindung besteht. Besonders bei der Aufzeichnung von Online-Meetings im Zuge der Studie kann dies eine Herausforderung sein. Das natürliche Umfeld, in dem sich Studienteilnehmer:innen bei der Studie befinden, kann ein Vorteil sein, genauso können die Personen jedoch auch durch Smartphones oder Ähnliches abgelenkt werden. (vgl. BARNUM 2011: 42f; DE BLEECKER/OKOROJI 2018: 11f)

Beim Remote Usability Testing kann zwischen einer moderierten und unmoderierten Studie differenziert werden. Bei einer unmoderierten Studie führen Teilnehmer:innen die Aufgabenstellungen ohne Beobachtung durch, bei der moderierten Form ist eine moderierende Person anwesend. Moderator:innen können dadurch zum einen das Verhalten der Studienteilnehmer:innen beobachten, zum anderen auch den Personen Fragen stellen oder beantworten. (vgl. DE BLEECKER/OKOROJI 2018: 12ff)

Die Vorteile der unmoderierten Studie sind, dass sich User:innen bei der Verwendung der Anwendung natürlich verhalten, und die Beobachtung durch Moderator:innen keinen Einfluss ausübt. Weiters kann ein unmoderiertes Usability Testing unabhängig von Zeitzonen durchgeführt werden, ebenso ist die Umsetzung generell weniger Aufwand für alle beteiligten Personen. (vgl. DE BLEECKER/OKOROJI 2018: 14f)

Die Form der unmoderierten Remote Studie bringt jedoch auch einige Herausforderungen mit sich. Moderator:innen haben keine Möglichkeit, mit den Studienteilnehmer:innen zu kommunizieren. Das bedeutet, dass den Personen weder Fragen zu ihrer Vorgehensweise gestellt werden können noch bei Problemen oder Fragen eingegriffen werden kann. Dadurch muss darauf geachtet werden, keine zu komplexen Aufgabenstellungen zu formulieren. Wird ein Skript erstellt, um den Teilnehmer:innen möglichst viel Information zukommen zu lassen, stellt dies ebenso einen hohen Aufwand dar, und es besteht keine Garantie, dass die Personen dieses korrekt verstehen. (vgl. BARNUM 2011: 45; DE BLEECKER/OKOROJI 2018: 15f)

2.2.2.3. Studienteilnehmer:innen

Weiters muss ein Anforderungsprofil für Teilnehmer:innen erstellt werden, die auch die Zielgruppe der Anwendung repräsentieren. Dies kann eine Herausforderung sein, wenn die Gruppe eine große Menge an unterschiedlichen Personen miteinbezieht. Es muss eine Liste an Charakteristiken formuliert werden. Die Charakteristika können sich unterscheiden von technischen Skills und Jobprofil, bis hin zu demographischen Faktoren. Bei den technischen Fähigkeiten können beispielsweise Hardware- oder Softwares Skills miteinbezogen werden. Alter, Geschlecht, Ausbildung und ökonomische Faktoren können Teil der demographischen Aspekte sein. Stehen

die technischen Aspekte im Vordergrund, sollte versucht werden, die Merkmale wie Geschlecht oder Alter zu vermischen, um eine ausgeglichene Gruppe an Teilnehmer:innen zu erlangen. (vgl. BARNUM 2011: 116-119)

Weiters muss die Anzahl der Studienteilnehmer:innen gewählt werden. DE BLEECKER/OKOROJI 2018 empfehlen für moderierte Studien je nach möglichem Kosten- und Zeitaufwand mindestens drei bis fünf Teilnehmer:innen, und wenn möglich bis zu 15 Personen. Dies gilt vor allem für quantitative Studien. Eine Studie von Jakob Nielsen, die in diverser Literatur zitiert wird zeigt, dass 80% - 85% der Informationen bei bereits fünf User:innen aufgedeckt werden. Diese kleinen Studien werden oft bei einem formativen Testing durchgeführt. (vgl. BARNUM 2011: 16f; DE BLEECKER/OKOROJI 2018: 14f)

2.2.2.4. Tasks

Ein weiterer Schritt ist das Erstellen und Formulieren von Tasks und Aufgabenstellungen. BARNUM 2011 formuliert einige Voraussetzungen dafür. Anfangs werden die Aufgaben selbst erdacht. Diese Aufgaben werden anschließend in Szenarien für die User:innen verpackt. Wichtig ist, die Tasks und Aufgabenstellungen basierend auf den Zielen der Studie zu erstellen. Bei der Formulierung der Aufgaben ist besonders darauf zu achten, dass diese keine Schritt-für-Schritt Anleitungen, sondern reale Szenarien darstellen. Auch sollen die Aufgabenstellungen nicht zu detailliert, oder zu unkonkret formuliert sein. Bei der Reihenfolge der Szenarios sollte ein möglichst einfaches Szenario am Anfang stehen, sodass die User:innen sich im Interface zurechtfinden können. Weiters muss definiert sein, an welchem Punkt ein Szenario abgeschlossen ist. (vgl. BARNUM 2011: 128-130f)

2.2.2.5. Metriken

Der nächste Schritt ist zu entscheiden, welche Metriken bei der Studie evaluiert, und welche Daten gesammelt werden sollen, um die Performance der Anwendung und die User Experience messen zu können. Das Sammeln von Daten und Informationen ist ein zentraler Schritt, um Informationen über die Produkte und die Verwendung durch Personen evaluieren zu können. Das Verhalten von Menschen ist kaum vorhersehbar, weswegen versucht wird, es mittels Daten näher zu verstehen. Die gesammelten Informationen können in allen Schritten des Design-Prozesses wiederverwendet werden. Sie helfen uns zu verstehen, wie User:innen denken und handeln, und wo mögliche Problemstellen liegen. (vgl. BECKER 2020: 121ff)

Grundsätzlich kann zwischen qualitativen und quantitativen Daten unterschieden werden. Daten sind nicht-numerische Daten. Sie ermöglichen es, die Motivationen, Probleme und Erfahrungen der User:innen offenzulegen. Diese Daten erhält man meist durch Beobachtung, Interviews und

Aufnahmen mit und von User:innen. Die zweite Art von Daten, quantitative Daten, sind numerische Daten. Es sind Daten, die gezählt werden und durch Statistik analysiert werden können. Damit können Beweise gesammelt werden, ob ein Konzept funktioniert oder nicht. Zum Sammeln quantitativer Daten können beispielsweise Umfragen gemacht werden. (vgl. BECKER 2020: 126, 141f)

Die beschriebenen Daten können mit sogenannten Metriken gesammelt werden. Eine Metrik ist eine Art, Phänomene zu messen und zu evaluieren. In der Usability besteht eine Vielzahl an Metriken. Diese Metriken sollen die Interaktion zwischen Mensch und Computer messen, genauer gesagt die Effizienz, Effektivität und die Zufriedenheit, wie sie in Kapitel 2.2.1 erläutert wurden. Besonders, wenn Probleme nicht so offensichtlich sind, dass sie alle User:innen betreffen, helfen Metriken, diese Probleme offenzulegen und im nächsten Schritt auszubessern. (vgl. DE BLEECKER/OKOROJI 2018: 6ff)

Task Success

Der „Task Success" ist eine der beliebtesten und meist-genutzten Usability Metriken. Dies stammt daher, dass er einerseits auf beinahe alle Phänomene anwendbar ist, von der Verwendung einer Maschine hin bis zur Lösung von Aufgabenstellungen in einem Web-GIS. Auch ist die Metrik verständlich und definierbar. Wird der Task erfolgreich gelöst, funktioniert die Anwendung. Wird er nicht gelöst, besteht ein Problem. Um den Task Success zu messen, muss erst festgelegt werden, inwiefern dieser definiert wird. Deswegen müssen auch die Aufgabenstellungen so formuliert werden, dass ein konkretes Ergebnis derer ersichtlich ist. (vgl. TULLIS/BILL 2013: 65-68)

Es wird zwischen unterschiedlichen Arten des Task Success unterschieden. Die einfachste Art ist der Binary Success. Dabei wird eine Aufgabe entweder vollständig korrekt und alleine von den User:innen gelöst, ansonsten besteht ein Misserfolg. Es wird die Aufzeichnung des Erfolgs durch numerische Werte wie Null und Eins präferiert. Dadurch können die Werte geordnet und statistisch analysiert werden. Weiters kann so die durchschnittliche Erfolgsrate pro Task angegeben werden. (vgl. ebd.)

Es ist nicht immer sinnvoll, den Task Success in einer zweiteiligen Skala wie Erfolg und Misserfolg zu bewerten, da oft Graubereiche bestehen. Es kann beispielsweise unterschieden werden, ob User:innen den Task allein oder mit Hilfe erfolgreich abschließen konnten. Es können sogenannte „Levels of Success" definiert werden, die auf verschiedenste Arten festgelegt werden können. Die Levels of Success beschreiben unterschiedliche Zwischenstufen zwischen Erfolg und Misserfolg. (vgl. TULLIS/BILL 2013: 70ff)

Eine Herausforderung bei der Durchführung der Studie und des Task Success ist festzulegen, wann User:innen bei der Lösung einer Aufgabe gestoppt werden sollen. Es bestehen drei unterschiedliche Punkte an denen es sinnvoll sein kann zu stoppen: es kann nach einer bestimmten Anzahl von versuchen gestoppt werden, nach einer konkreten Zeitspanne oder sobald User:innen von selbst aufgeben. Zusätzlich muss jedoch auch hier beachtet werden, dass es vorkommen kann, dass der Task von den Moderator:innen frühzeitig gestoppt werden muss, auch wenn das definierte Limit noch nicht erreicht ist. Eine solcher Situation wäre, wenn Personen ungeduldig oder unzufrieden werden, oder nicht mehr weiterwissen. (vgl. TULLIS/BILL 2013: 73f)

Time on Task

Die Effizienz einer Anwendung kann durch die Zeit, die die Lösung einer Aufgabenstellung erfordert, evaluiert werden. Diese Zeit kann auch als „Time on Task" bezeichnet werden. Kernkonzept der Effizienz ist es, wie lange es dauert, bis User:innen in der Anwendung der Software brauchen, bis sie ihr Ziel erreicht haben. Die Effizienz setzt sich aus der Zeit, die die Lösung eines Tasks erfordert zusammen, und dem Task selbst. Beispielsweise erfordern manche Tasks einen hohen Zeitaufwand, sind aber schlussendlich von hoher Effektivität. Je schneller User:innen eine Aufgabenstellung fertigstellen können, desto zufriedener sind sie meist auch. Besonders bei Tasks, die im Alltag wiederholt oder regelmäßig durchgeführt werden müssen ist eine möglichst kurze Time on Task von Bedeutung. (vgl. TULLIS/BILL 2013: 74f; BECKER 2020: 225f)

Grundsätzlich kann die Zeit, die für eine Aufgabenstellung aufgewendet wird, vom Start der Aufgabe bis hin zum Ende gemessen werden. Die Herausforderung ist hierbei festzulegen, wann genau das Ende eines Tasks ist, und wann die Zeitaufzeichnung gestoppt wird. Es kann beispielsweise die Zeit erst gestoppt werden, wenn das Ergebnis der Aufgabe vorhanden ist, oder wenn User:innen selbst der Meinung sind, eine Aufgabe fertig gelöst zu haben. (vgl. TULLIS/BILL 2013: 75f)

Weitere Faktoren, die in Erwägung gezogen werden müssen ist, ob auch Tasks, die nicht erfolgreich gelöst wurden in die Berechnung miteinbezogen werden. Grundsätzlich legen TULLIS/BILL 2013: 81 fest, dass von Moderator:innen abgebrochen Tasks nicht in die Evaluierung miteinbezogen werden.

Learnability

Die „Learnability" einer Anwendung beschreibt, wie lange es dauert, bis User:innen diese effizient verwenden können. Das Lernen des Verwendens einer Anwendung kann ein kurzer Prozess von wenigen Minuten sein, aber teilweise auch über mehrere Stunden, Tage und Wochen andauern. Ist

die Learnability einer Software hoch, lernen User:innen die Verwendung dieser meist schnell und können auch bald damit produktiv sein. Die Software hat hier eine flache Lernkurve. Die Learnability zeigt auch, ob die Funktion eines Produkts einfach wiederholt werden können. (vgl. BECKER 2020: 225)

Die Learnability kann in verschiedenen anderen Metriken gemessen werden, wie dem Task Success und der Time on Task. Im Laufe des Lernens ist das Ziel, eine Verbesserung der Effizienz zu sehen. Die Learnability kann auf drei unterschiedliche Arten überprüft werden. Zum einen können einfach alle Tasks der Reihenfolge von User:innen abgearbeitet werden. Die zweite Methode ist, eine Pause oder Ablenkung zwischen den unterschiedlichen Aufgaben einzubauen, wodurch User:innen möglicherweise Teile der Verwendung wieder vergessen können. Die letzte Möglichkeit ist es, mehrere Sessions durchzuführen. Um die Learnability messen zu können, sollten mindestens drei oder vier Aufgabenstellungen den User:innen vorgelegt werden. (vgl. DE BLEECKER/2018: 14f)

User Experience und Zufriedenheit

Alle der zuvor beschriebenen Metriken fokussieren sich auf objektive Aspekte. Ebenso wichtig ist jedoch die subjektive Erfahrung der User:innen, die User Experience. Informationen darüber sind von großer Bedeutung, da wie in Kapitel 2.2.1 bereits erwähnt eine Anwendung noch so effizient und effektiv sein kann, die Zufriedenheit von User:innen jedoch etwas anderes widerspiegeln kann. Die Zufriedenheit von User:innen ergibt sich vor allem daraus, ob die Anwendung für sie einen Mehrwert darstellt. (vgl. BECKER 2020: 227f; DE BLEECKER/OKOROJI 2018: 122f)

Die Zufriedenheit der User:innen wird meist durch Fragebögen evaluiert. Die Fragen können von Skalen bis hin zu offenen Fragen reichen. Es bestehen im Usability Testing unterschiedliche Arten von Bewertungsskalen. (vgl. DE BLEECKER/OKOROJI 2018: 123f)

Eine der bekanntesten ist die Likert-Skala. Diese besteht aus mehreren Leveln der Zustimmung. Bewertet werden Aussagen wie: „Das Interface ist klar strukturiert“. Eine typische Skala mit Antwortmöglichkeiten sieht wie folgt aus:

1. Stimme gar nicht zu
2. Stimme eher nicht zu
3. Stimme weder zu noch nicht zu
4. Stimme eher zu
5. Stimme zu

In der Praxis bestehen jedoch heutzutage unterschiedlichste Arten und Ausprägungen der Likert-Skala. (vgl. ebd.)

Eine weiter Skala ist die semantische differenzielle Skala. Dabei werden Paare von bipolaren oder gegenteiligen Adjektiven auf einer Skala präsentiert. Paare können beispielsweise „Stark" und „Schwach" sein. (vgl. DE BLEECKER/OKOROJI 2018: 124)

Bei der Wahl und der Gestaltung der Skala sind unterschiedliche Faktoren zu beachten. Zwei der relevantesten sind die Anzahl an Levels, und ob eine gerade oder ungerade Zahl gewählt wird. Bei zweiterem Punkt wird oft eine Skala mit einem neutralen Punkt gewählt, da davon ausgegangen wird, dass eine neutrale Antwort ein valides Ergebnis ist. In der Praxis werden jedoch genauso Skalen ohne neutralen Punkt verwendet. Bei der gesamten Anzahl an Levels werden meist fünf bis sieben Punkte gewählt. (vgl. DE BLEECKER/OKOROJI 2018: 126ff)

Ein weiterer Aspekt, der bei der Erstellung eines Fragebogens miteinbezogen werden muss, ist die Formulierung der Fragen oder Aussagen. Hier muss darüber entschieden werden, ob diese positiv, negativ oder variierend formuliert werden. Während grundsätzlich die Abwechslung der Polarität der Fragen Vorteile haben kann, wie auf den ersten Blick die Vermeidung von Vorurteilen, können ebenso negative Effekte dadurch entstehen. Eines der größten Probleme ist, dass User:innen dadurch verwirrt werden können und schlussendlich für sie nicht mehr klar ist, ob die Frage nun negativ oder positiv formuliert ist, und dadurch ungewollt gegenteilige Aussagen getroffen werden. SAURO/LEWIS 2016 konnten ebenso keine negativen Einflüsse basierend auf rein positiv formulierten Fragebögen auf die Korrektheit derer finden. (vgl. SAURO/LEWIS 2016: 206f)

Es besteht eine Vielzahl an Fragebögen, die die Zufriedenheit von User:innen überprüfen. Hier können beispielsweise standardisierte Fragebögen verwendet werden. Dies sind bereits erstellte und vorgefertigte Fragebögen. Der Vorteil dieser standardisierten Fragebögen ist, dass sie objektiv und replizierbar sind. Weiters sind sie quantifizierbar, ökonomisch und ermöglichen eine wissenschaftliche Generalisierung. (vgl. SAURO/LEWIS 2016: 185f)

Einer der bekanntesten standardisierten Fragebögen ist der „SUS" (System Usability Scale). Der Fragebogen beinhaltet zehn Fragen, die teils positiv und teils negativ formuliert sind, und mit einer fünfteiligen Skala bewertet werden können. Dieser ist in Abbildung 2 dargestellt. (vgl. DE BLEECKER/OKOROJI 2018: 138f)

	The System Usability Scale Standard Version		Strongly disagree				Strongly agree
			1	2	3	4	5
1	I think that I would like to use this system frequently.		O	O	O	O	O
2	I found the system unnecessarily complex.		O	O	O	O	O
3	I thought the system was easy to use.		O	O	O	O	O
4	I think that I would need the support of a technical person to be able to use this system.		O	O	O	O	O
5	I found the various functions in the system were well integrated.		O	O	O	O	O
6	I thought there was too much inconsistency in this system.		O	O	O	O	O
7	I would imagine that most people would learn to use this system very quickly.		O	O	O	O	O
8	I found the system very awkward to use.		O	O	O	O	O
9	I felt very confident using the system.		O	O	O	O	O
10	I needed to learn a lot of things before I could get going with this system.		O	O	O	O	O

Abbildung 2: Fragebogen „System Usability Scale“ (Quelle: SAURO/LEWIS 2016: 198)

Es besteht noch eine Vielzahl an weiteren standardisierten Fragebögen, die unterschiedliche Aspekte der Usability behandeln, mit verschiedenen Skalen und Ausprägungen bezogen auf die Anzahl und Art der Fragen.

2.3. Grundlagen des Natural Language Processing

Dieses Kapitel stellt die theoretischen Grundlagen des Natural Language Processing dar. Der Fokus liegt dabei auf dem Workflow von Natural Language Processing und den Anwendungsmöglichkeiten.

Natural Language Processing kann vereinfacht definiert werden als Methode zur Verarbeitung und der Interpretation von Text oder Gesprochenem mittels Computer. Die ersten Versuche, Sprache in Form von Text und Gesprochenem zu verarbeiten und zu verstehen, liegen bereits über ein halbes Jahrhundert zurück. Heute ist Natural Language Processing Teil unseres alltäglichen und auch geschäftlichen Lebens. Die Methodiken dieses wissenschaftlichen Feldes finden Anwendung in Online-Übersetzern, Mail-Programmen als Spam-Filter oder Chatbots. Natural Language Processing ist ein interdisziplinäres Feld. Wie Abbildung 3 zeigt, kann NLP primär in den Computer Sciences und der Linguistik verortet werden. Die künstliche Intelligenz (KI) ist der methodische Rahmen, in den NLP eingebettet werden kann. (vgl. ZHANG/TENG 2021: 3)

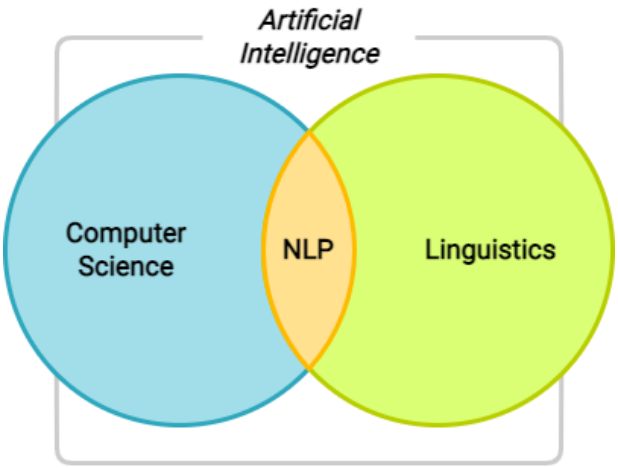

Abbildung 3: Wissenschaftliche Verortung von Natural Language Processing (Basierend auf: CLEVERTAP 2019)

2.3.1. Ablauf von Natural Language Processing

In diesem Kapitel werden die einzelnen Schritte des Natural Language Processing dargestellt.

In den Anfängen und früheren Jahren des Natural Language Processing wurde die Analyse von Text und Gesprochenem in drei theoretische Konzepte unterteilt:

- die Syntax
- die Semantik
- die Pragmatik

(vgl. DALE 2010: 5)

Anfangs wird die Syntax bzw. grammatikalische Zusammensetzung eines Satzes analysiert. Anschließend folgt eine Untersuchung der Semantik, in der die wörtliche Bedeutung und Struktur des Satzes untersucht wird, und zuletzt folgt die Pragmatik, in der die tiefere Bedeutung eines Satzes freigelegt werden soll. Praktisch gesehen ist eine strikte Abtrennung der Konzepte wie diese kaum möglich, in der Theorie ist die Trennung jedoch ein wichtiger Anhaltspunkt. (vgl. ebd.)

Die Unterteilung der Schritte des Natural Language Processing, wie sie in der Realität oft verwendet wird, ist in Abbildung 4 dargestellt.

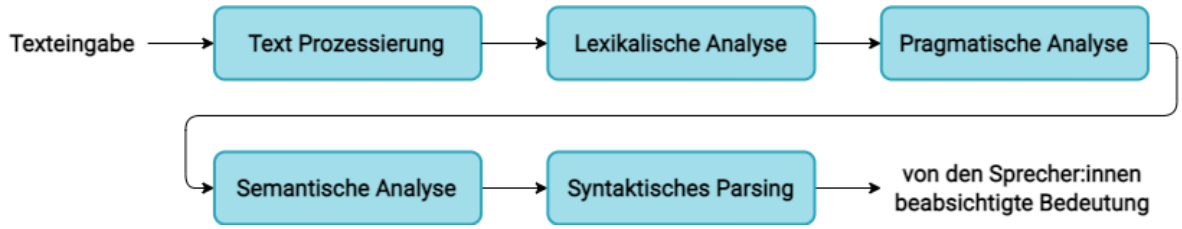

Abbildung 4: Workflow des Natural Language Processing (Basierend auf: DALE 2010: 4)

Folgend werden alle Schritte des Natural Language Processing beschrieben.

2.3.1.1. Text-Prozessierung

Der erste und mitunter auch einer der relevantesten Schritte des Natural Language Processing ist die „Text-Prozessierung". Ziel ist dabei, den unverarbeiteten Input-Text in abgegrenzte Einheiten, geeignet zur Weiterverarbeitung, umzuwandeln. Solche Einheiten können beispielsweise Sätze und Wörter sein. Diese bilden die Grundlage und den Input für alle weiteren Analyse-Schritte. (vgl. PALMER 2010: 9f)

Die Text-Prozessierung kann in zwei weitere Stufen unterteilt werden. Zum einen gibt es die „Text Segmentation", deren Ziel es ist, Texte in einzelne Sätze zu teilen, zum anderen die „Word Segmentation", bei der die Sätze weiter in Wörter unterteilt werden. Diese kann auch als „Tokenisierung" bezeichnet werden, da im Natural Language Processing einzelne Wörter auch „Tokens" genannt werden. Bei beiden Schritten geht es darum, die Begrenzung eines Satzes oder Wortes zu finden, weswegen sie gezwungenermaßen auch zusammengehören. (vgl. PALMER 2010: 9f)

In vielen Sprachen werden Wörter durch Leerzeichen, und Sätze durch Satzzeichen getrennt. Dazu gehören alle Sprachen, die das lateinische Alphabet verwenden. So zählt auch Deutsch dazu. Der Vorteil solcher Sprachen ist, dass durch die in der Theorie eindeutigen Strukturen Sätze und Wörter eindeutig voneinander abgetrennt werden können. Trotz der klaren Regeln gibt es ein großes Maß an Herausforderung wie verschiedene Unregelmäßigkeiten. Eine der größten Problematiken der Text-Prozessierung sind Mehrdeutigkeiten. (vgl. PALMER 2010: 16)

So können beispielsweise nicht nur Sätze durch Satzzeichen getrennt werden, sondern auch Abkürzungen und Datumsangaben Punkte beinhalten. Weitere Probleme sind mehrteilige Wörter oder Multi-Wort Expressionen. Ersteres bezeichnet Wörter, die aus mehreren einzelnen Wörtern zusammengesetzt sind, aber nicht durch ein Leerzeichen getrennt sind, wie etwa „Lebensversicherung". Um dieses Wort etwa korrekt verarbeiten zu können, müssen beide Wortteile als solche erkannt werden. Da aber kein Leerzeichen inkludiert ist, stellt dies eine Herausforderung dar. Bei Multi-Wort Expressionen ergibt sich hingegen die Schwierigkeit daraus, dass ein Leerzeichen zwischen Wörtern, die inhaltlich zusammengehören besteht, wie etwa bei „22. März". (vgl. PALMER 2010: 18f)

2.3.1.2. Lexikalische Analyse

Ist der Input-Text in einzelne, und weiter verarbeitbare Einheiten unterteilt, folgt die „lexikalische Analyse". Hier wird ein Text auf dem Level von einzelnen Worten untersucht. Wörter sind keine undynamischen Textteile, sondern morphologisch komplex. Auf diese Morphologie hin müssen die Wörter analysiert werden. Zwei der häufigsten Methoden der lexikalischen Analyse sind die „Lemmatisierung" und das „Stemming". Beide können aufeinanderfolgend angewandt werden. Als erstes kann das Stemming eingesetzt werden. Dabei werden alle Affixe, also alle Vorsilben (Präfixe) und Nachsilben (Suffixe) vom Grundwort entfernt. Im nächsten Schritt, der Lemmatisierung, wird versucht, eine bedeutungsvolle und korrekte Grundform des unverarbeiteten Worts zu finden. (vgl. HIPPISLEY 2010: 31f)

2.3.1.3. Syntaktisches Parsing

Im Schritt des „syntaktischen Parsings" soll die grammatikalische Struktur des Satzes untersucht werden, sodass dieser anschließend semantisch interpretiert werden kann. (vgl. ZHANG/TENG 2021: 6)

Einer der nützlichsten Tasks ist hierbei das Part-of-Speech (POS) Tagging. Dabei wird jedem Wort die Rolle, die es im Satz spielt, zugewiesen. Tags sind beispielsweise „NN" für ein Nomen im Singular, oder „VB" für ein Verb in seiner Grundform. Wenn ein Wort mehrere Rollen hat, können diesem auch mehrere Tags zugewiesen werden. Eine Möglichkeit, das POS-Tagging umzusetzen, ist beispielsweise der „Syntax Tree", der in Abbildung 5 dargestellt ist. Dadurch kann die hierarchische Struktur des Satzes analysiert werden. (vgl. ebd.)

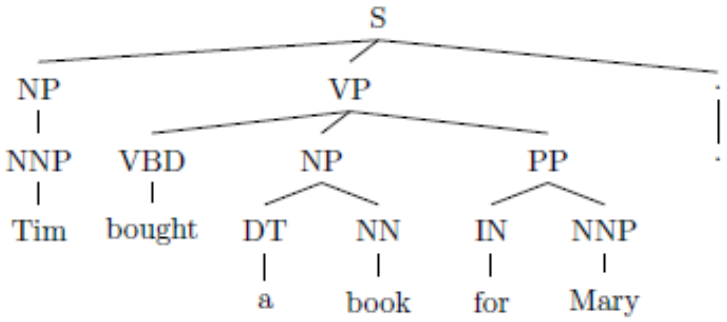

Abbildung 5: Syntax Tree für syntaktisches Parsing (Quelle: ZHANG/TENG 2021: 6)

Es bestehen noch weitere Methoden für das syntakische Parsing, die jedoch alle denselben Output haben: einen grammatikalisch analysierten, strukturierten und hierarchischen Satz, der als Input für die semantische Analyse verwendet werden kann. (vgl. ZHANG/TENG 2021: 6-9)

2.3.1.4. Semantische Analyse

Ziel der „semantischen Analyse" ist es, die wortwörtliche Bedeutung eines Textes zu finden. Die semantische Analyse ermöglicht es, die Intuition der Aussage einer Person zu verstehen. Dieser Schritt ist mitunter einer der schwerste des Natural Language Processings. Die semantische Analyse kann grundsätzlich auf Wort-Level (lexikalische Semantik) und auf Satz-Level (supralexikalische Semantik) durchgeführt werden. Beide sind miteinander verbunden. (vgl. GODDARD/SCHALLEY 2010: 93f)

Möglichkeiten der Analyse der lexikalischen Semantik beinhalten beispielsweise die Untersuchung der Relation der Bedeutung von Wörtern, wie etwa von Wortpaaren wie „gut-schlecht" oder „Baum-Blatt". Auf Satz-Level kann die Bedeutung eines Satzes analysiert werden, in dem die Beziehung von Wörtern zueinander untersucht wird. (vgl. ZHANG/TENG 2021: 6-9)

Während bei der semantischen Analyse die eher oberflächliche Bedeutung eines Wortes oder Satzes analysiert wird, soll bei der „pragmatischen Analyse" die tiefere Bedeutung eines Satzes oder Textes freigelegt werden, die aus den Wörtern und Aufbau eines Satzes nicht direkt herausgelesen werden kann. Hier bestehen noch große Herausforderungen und Forschungslücken, weswegen die pragmatische Analyse hier nicht detaillierter beschrieben wird. Auch ist sie für diese Masterarbeit nicht weiter relevant. (vgl. DALE 2010: 7)

2.3.2. Anwendungen und Methoden des Natural Language Processing

In den vorhergehenden Kapiteln wurden bereits einige Anwendungen und Tasks des Natural Language Processing wie die Lemmatisierung und das POS-Tagging beschrieben. In den folgenden Kapiteln werden konkrete Tasks, die für die praktische Umsetzung der Masterarbeit relevant sind, beschrieben.

2.3.2.1. Text Klassifizierung

Ein bekannter Task ist die „Text Klassifizierung“, auch „Sequence Classification“ genannt. Dieser Task kann als simpler Klassifikations-Task bezeichnet werden. Es wird versucht, einem Satz eine von mehreren definierten Klassen zuzuweisen. Es bestehen mehrere Anwendungsfelder dieser Methode, wie die „Document Categorization“, bei der das Thema eines Dokuments erörtert wird, oder die Klassifikation eines Satzes zur Generierung einer passenden Antwort darauf. (vgl. LAURIOLA ET AL. 2022: 444)

Eine der häufigsten Anwendungen der Text Klassifizierung ist die „Sentiment Analyse“. Hier wird dem Satz eine Polarität oder eine Emotion zugewiesen. Oft werden die einfachsten Formen des Sentiments verwendet, wie positiv, negativ oder neutral, es kann allerdings auch eine viel detailliertere Analyse durchgeführt werden. (vgl. ZHANG/ZHIYANG 2021: 17)

2.3.2.2. Named Entity Recognition

Ein bekannter NLP-Task, der zur „Information Extraction“ (IE) gezählt werden kann, ist die „Named Entity Recognition“ (NER). Die Grundlage der Information Extraction, sowie auch der Named Entity Recognition ist es, strukturierte Informationen aus unstrukturiertem Text zu extrahieren. Extrahiert werden können unterschiedlichste Informationen, wie Events oder Emotionen, aber auch konkrete „Entitäten“. Letztere werden bei NER gesucht. (vgl. ZHANG/ ZHIYANG 2021: 14)

Dazu werden die Wörter bzw. Tokens eines Satzes basierend auf einem Wörterbuch oder deren Struktur und Kontext gelabelt und kategorisiert. Die meist verwendeten Entitäten sind Personen, Orte, und Organisationen, diese können jedoch nach Belieben erweitert werden, beispielsweise durch Datum oder Zeit. (vgl. MCKENZIE/ADAMS 2021: 3f)

Für das Taggen der Entitäten bestehen unterschiedliche Tagging-Schemes, die von sehr einfach bis komplex reichen. Das einfachste Schema ist das „IO-Schema“, wobei das „I-Tag“ für „Inside“, und das „O-Tag“ für „Outside“ steht. Es werden die Tags „I“ oder „O“ der Abkürzung der Entität hinzugefügt. Wird ein Wort beispielsweise als Person erkannt, kann diese als „I-PER“ gelabelt werden. Ein Wort, das keine Entität darstellt, wird als „O“ getagged. (vgl. ASHAMMARI/ALANAZI 2021: 1f, ZHANG/ZHIYANG 2021: 14)

Eines der bekanntesten Tagging-Schemes ist das „IOB-Schema“, auch „BIO-Schema“ genannt. Es werden die Tags „I“, „O“ und „“ verwendet. Das „I-Tag“ und „O-Tag“ funktionieren gleich wie beim vorherigen Schema, das „B-Tag“ wird hier jedoch am Beginn einer Entität verwendet, die aus mehreren Wörtern oder Tokens bestehen. (vgl. ASHAMMARI/ALANAZI 2021: 1f)

Bei der Named Entity Recognition bestehen einige Herausforderungen. Schwierigkeiten können beispielsweise komplexen Wörter oder Token-Kombinationen, wie Wörter, die aus mehreren Tokens bestehen, sein. (vgl. ASHAMMARI/ALANAZI 2021: 1)

Ein weiteres Problem sind Mehrdeutigkeiten. So kann etwa das Wort „Mercedes" entweder als Person oder aber auch Organisation getagged werden. (vgl. MCKENZIE/ ADAMS 2021: 4)

Eine zusätzliche Herausforderung bei der Umsetzung der Named Entity Recognition ist, dass grundsätzlich nur ein kleiner Teil der Trainingsdaten die eigentlichen Entitäten sind, denn ein Satz beinhält oft nur wenige der Wörter, die zu taggen sind, und hauptsächlich Wörter, die dem „O-Tag" zuzuweisen sind. Deshalb wird hier oft eine große Menge an Trainingsdaten benötigt. (vgl. CHO ET AL. 2022: 2)

2.4. Natural Language Processing von räumlichen Informationen

Wie bereits erwähnt, ist Natural Language Processing ein interdisziplinäres Feld. Jedoch kann die Wissenschaft nicht nur in der Linguistik, den Computer Sciences und der KI verortet werden, sondern auch als Teil der Geografie und Geoinformatik. Dies hat mehrere Gründe. Zum einen hat ein beachtlicher Teil der Sprache und Kommunikation Inhalt, der auf den geographischen Raum bezogen ist. Durch die Anwendung von NLP in der Geografie kann dies dazu beitragen, geographische Phänomene durch die Identifikation und Extraktion von Orten, Events und Weiterem besser verstehen und analysieren zu können als mit traditioneller räumlicher und zeitlicher Analyse. Im Gesamten hat die Kombination von Natural Language Processing und der Geoinformatik zu vielen wertvollen Applikationen heutzutage positiv beigetragen. (vgl. MCKENZIE/ ADAMS 2021: 1f)

LAMPOLTSHAMMER/HEISTRACHER 2012: 82 zeigen den Fokus der Forschung im Bereich von Natural Language Processing in Verbindung mit Geoinformatik und Geografie auf. Die Schwerpunkte liegen auf drei Feldern:

- Human Computer Interaction von textgesteuerten GIS
- Geoparsing
- Location-Based Services

Im Folgenden Kapitel wird vor allem auf das Geoparsing und die Kombination mit der HCI eingegangen.

2.4.1. Grundlagen des Geoparsings

Bei der Verarbeitung von räumlichen Informationen mittels Natural Language Processing ist der Ablauf grundsätzlich derselbe wie bei der Verarbeitung anderer Informationen. Jedoch gibt es einen zusätzlichen Schritt, der für die Verarbeitung von rein räumlichen Informationen verantwortlich ist: das Geoparsing. Ziel des Geoparsing ist es, textuelle räumliche Informationen in geographische Koordinaten umzuwandeln. (vgl. GRITTA ET AL. 2017: 604)

Das Geoparsing ist eine Art von Information Extraction, diese kann wiederum zur Methode des Information Retrieval (IR), also dem Extrahieren von relevanten Informationen gezählt werden. (vgl. MIDDLETON ET AL. 2018: 2)

Das Geoparsing besteht aus den folgenden zwei Schritten: dem „Geotagging“, und dem „Geocoding“, wie in Abbildung 6 dargestellt.

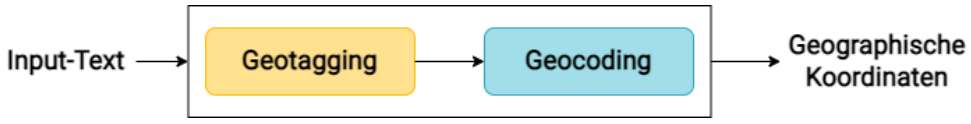

Abbildung 6: Worklow des Geoparsing (Basierend auf: GRITTA ET AL. 2017: 604)

Das Geotagging wird auch „Toponym Extraction“ oder „Location Extraction“ genannt. Dabei werden „Toponyme“ aus Texten und Sätzen extrahiert. Toponyme sind einfach beschrieben Ortsangaben, die genauere Definition wird jedoch in Kapitel 2.4.1.1 erläutert. Beim Geotagging bestehen unterschiedliche Definitionen in der Literatur, beispielsweise als Synonym für das Geoparsing (vgl. LEIDNER/LIEBERMAN 2011: 1). Für diese Arbeit wird jedoch die Definition von GRITTA ET AL. 2017: 604 verwendet. Hier wird das Geotagging als Prozess der Extraktion der räumlichen Informationen aus Text beschrieben. Das Geotagging kann auch mit der Named Entity Recognition gleichgesetzt werden, die in Kapitel 2.3.2.2 beschrieben wurde, da auch hier basierend auf linguistischen Eigenschaften die relevanten Informationen extrahiert werden. (vgl. MIDDLETON ET AL. 2018: 5)

Der zweite Schritt, das Geocoding, wird auch „Toponym Resolution“ genannt. Ziel dieses Schrittes ist es, das gemeinte Toponym korrekt zu identifizieren, und die passenden räumlichen Koordinaten zuzuweisen. (vgl. GRITTA ET AL. 2019: 684)

2.4.1.1. Toponyme

Toponyme sind, in einfachster Übersetzung, Ortsnamen. Die tatsächliche Definition ist viel umfangreicher, worauf später in diesem Kapitel eingegangen wird. Es bestehen differenziertere und detailliertere Unterscheidungen von Toponymen, auf die in diesem Kapitel kurz eingegangen wird. GRITTA ET AL. 2019 definiert Toponyme wie folgt:

> *A location is any of the potentially infinite physical points on Earth identifiable by coordinates. With that in mind, a toponym is any named entity that labels a particular location. (GRITTA ET AL. 2019: 690)*

GRITTA ET AL. 2019: 691-694 unterscheidet zwischen „Non-Toponymen“, wörtlichen Toponymen und assoziativen Toponymen.

Non-Toponyme sind Worte, die beim regulären Geoparsing nicht als Orte extrahiert werden, wie beispielsweise der „British Grand Prix“. Da diese Wörter jedoch tatsächlich, wenn auch indirekt, einen Ort bezeichnen, führt dies zu Informationsverlust, wenn diese im Geoparsing übergangen werden. Wörtliche Toponyme beschreiben „reguläre“ Ortsnamen wie Ländernamen, Städtenamen und Straßenamen, die eindeutig physische Orte beschreiben. Als assoziative Toponyme werden Toponyme bezeichnet, die nicht eindeutig als Ortsnamen extrahierbar sind. Dies macht sie

teilweise nur schwer erkennbar im Geoparsing. Beispiele dafür sind „Homonyme". Diese beschreiben Namen für Personen, die eigentlich auch Ortsnamen sind, wie „Paris". Eine weitere Herausforderung sind etwa Bezeichnungen für Sprachen, wie Italienisch. Diese bezeichnen grundsätzlich keine Orte, sind aber dennoch räumliche Informationen, die nützlich sein können. (vgl. GRITTA ET AL. 2019: 691-694)

GRITTA ET AL. 2019 unterscheidet noch zwischen unterschiedlichen weiteren Differenzierungen von Toponymen, dies ist jedoch für diese Arbeit nicht weiter von Relevanz.

2.4.1.2. Umsetzung von Geoparsing

Das Geoparsing kann mittels verschiedener Methoden umgesetzt werden. Die drei meistverwendeten Methoden sind:

- „Gazetteer Lookup"
- „Regelbasiert"
- „Machine Learning"

(vgl. LEIDNER/LIEBERMAN 2011: 7)

Der erste Ansatz ist auf einem Gazetteer basiert. Ein Gazetteer ist eine Datenbank für Ortsnamen (vgl. MCKENZIE/ADAMS 2021: 1). Es werden dabei im unstrukturierten Text im Gazetteer vorkommende Ortsnamen gesucht. Je nach Umfang und Qualität des Gazetteers kann dieser Ansatz zu Schwierigkeiten führen, wenn beispielsweise nicht vorhandene Toponyme verarbeitet werden sollen. Eine weitere Herausforderung sind sich ändernde Benennungen von Städten, oder die Veränderung von Grenzen, da es schwer ist, den Gazetteer aktuell zu halten. (vgl. LEIDNER/LIEBERMAN 2011: 7) Dieser Prozess kann auch als Named Entity Matching (NEM) bezeichnet werden (vgl. MIDDLETON ET AL. 2018: 5).

Bei der regelbasierten Methode werden Toponyme basierend auf zuvor definierten Regeln extrahiert und als räumliche Informationen identifiziert. (vgl. LEIDNER/LIEBERMAN 2011: 7)

Der dritte Ansatz ist mittels Machine Learning, dessen Grundlagen in Kapitel 2.5 konkreter beschrieben werden. Hier werden die räumlichen Informationen basierend auf erlernten Mustern extrahiert. Diese werden anhand von einer großen Menge an Trainingsdaten erlernt. (vgl. ebd.)

2.4.1.3. Herausforderungen im Geoparsing

Im Geoparsing bestehen mehrere Herausforderungen. Eine der größten ist die Mehrdeutigkeit von Toponymen, wie es auch generell im Natural Language Processing eine Herausforderung ist. Wie bereits in Kapitel 2.4.1.1 erläutert bestehen viele verschiedene Ausprägungen von Toponymen.

Herkömmliche NER-Tagger sind laut GRITTA ET AL. 2019: 684 zwar für die Extraktion von wortwörtlichen Toponymen gut geeignet, aber nicht darüber hinaus für die Extraktion von pragmatischen oder semantischen Toponymen. In folgendem Satz: „Frau Schmidt geht mit ihrem deutschen Schäferhund in einem Wiener Park spazieren ", würden zwar „deutsche" und „Wiener", als raumbezogene Wörter erkannt werden, jedoch ist nur eines dieser Wörter hier als Ortsname gemeint. (vgl. GRITTA ET AL. 2019: 684)

Es kann hier grundsätzlich zwischen zwei Arten von Mehrdeutigkeit unterschieden werden. Eine dieser ist das Unwissen, ob ein Wort überhaupt ein Toponym ist. Ein Beispiel dafür ist das Wort „Paris". Hier kann je nach Kontext das Wort als die Stadt Paris, oder als der Name identifiziert werden. Diese Entscheidung ist teils eine Herausforderung, da sie nicht immer eindeutig zu treffen ist. (vgl. LEIDNER/LIEBERMAN 2011: 5f) Nachdem entschieden wurde, ob das Wort ein Toponym ist oder nicht, muss wiederum der korrekte Ort identifiziert werden. Beispielsweise gibt es ein Paris in Frankreich, oder eine Stadt in den USA mit demselben Namen. (vgl. MCKENZIE/ADAMS 2021: 4)

Generell gilt auch: je niedriger das Level, auf dem Toponyme unterschieden werden, desto höher die Chance an Mehrdeutigkeit. Beispielsweise bestehet weniger Mehrdeutigkeit, wenn nur zwischen Ländernamen unterschieden wird. (vgl. LEIDNER/LIEBERMAN 2011: 5)

Auch wenn die bisher genannten Herausforderungen noch bestehen, sind hier teilweise schon einige Fortschritte gemacht worden. Laut GRITTA ET AL. 2017: 621f ist die Anforderung für eine Weiterentwicklung von Geoparsing die Beachtung von räumlichen Informationen nicht nur auf der syntaktischen Ebene, sondern insbesondere auch der semantischen Ebene. Bei dem Satz „Ich fahre mit dem Auto von Salzburg aus 20 km in Richtung Norden…" werden komplexe Systeme mit künstlicher Intelligenz benötigt, um auch unkonkrete Angaben wie Distanzen oder Himmelsrichtungen korrekt und effizient zu verarbeiten. Dies stellt die Herausforderungen zukünftiger Geoparsing-Systeme dar.

2.4.2. Geoinformationssysteme und Natural Language Processing

Wie LAMPOLTSHAMMER/ HEISTRACHER 2012 erwähnen, ist auch die Verbindung von der Human Computer Interaction und GIS im Kontext des Natural Language Processing von großer Bedeutung in der Forschung. Bei Studien zu dieser Kombination geht es primär darum, mit textgesteuerten Interfaces für Web-GIS oder Desktop-GIS eine Alternative zu traditionellen buttongesteuerten Interfaces zu schaffen. Folgend werden drei unterschiedliche Studien vorgestellt, die ein textgesteuertes Interface für GIS und Web-GIS entwickelt haben, und somit relevant für diese Masterarbeit sind.

SETLUR ET AL. 2016 haben ein textgesteuertes User Interface für ein Web-GIS entwickelt. Dieses wurde speziell für die visuelle Analyse entwickelt, wobei hier Daten wie Erdbeben und Temperaturaufzeichnungen abgefragt werden können. Abbildung 7 zeigt die Anwendung „Eviza" die entwickelt wurde.

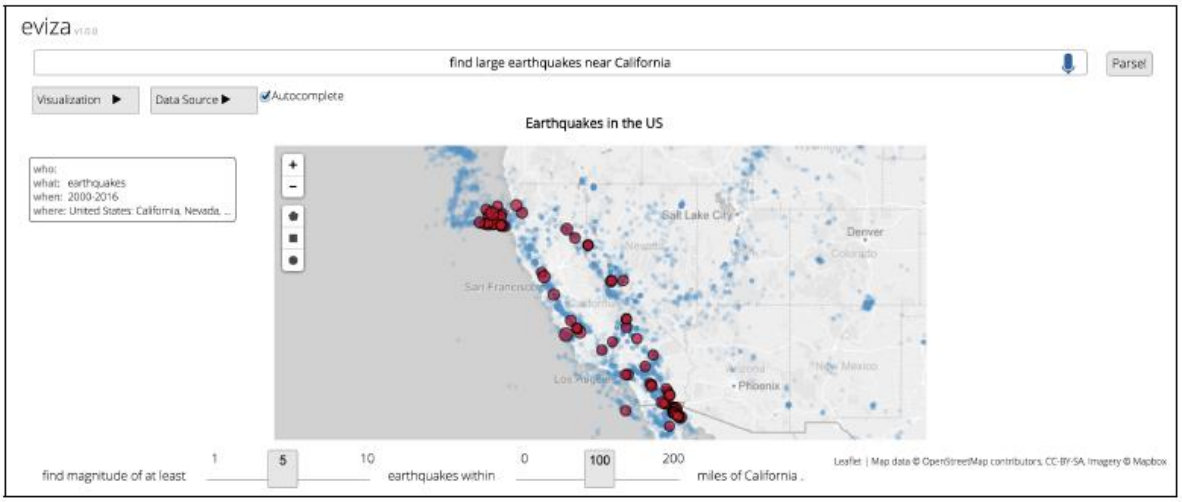

Abbildung 7: User Interface des Web-GIS „Eviza" (Quelle: SETLUR ET AL. 2016: 365)

Dabei können Anfragen in Form von natürlich formulierten Fragen mittels Textfeldes gestellt werden. Das Ergebnis wird in Karten und Diagrammen angezeigt. Anschließend kann das Ergebnis mittels Buttons und Schiebereglern manuell verändert werden. Laut SETLUR ET AL. 2016 liegt die Stärke des textgesteuerten Interfaces besonders darin, dass User:innen kein Wissen über die analytischen Funktionen und die Umsetzung haben müssen, und die Anwendung ohne vorherige Einschulung verwenden können. Ebenso zeigt das Interface für ein Web-GIS großes Potenzial für die Weiterentwicklung von Anwendungen für visuelle räumliche Analysen. Ebenso wurde bei der Studie ein Usability Testing durchgeführt. Die Ergebnisse sind vielversprechend. Bei der Time on Task waren Teilnehmer:innen mit dem textgesteuerten Interface teils doppelt so schnell wie beim buttongesteuerten Interface. Auch fand das Interface bei den Teilnehmer:innen persönlichen Anklang. Jedoch bestanden auch Zweifel, dass das Interface eher für Anfänger:innen geeignet ist.

Eine sehr simple Anwendung von einem textgesteuerten Interface für ein Web-GIS wurde von ALABSEER/YAQOT 2018 entwickelt. Die Anwendung ermöglicht das visualisieren von Tweets, basierend auf der Eingabe von Stichwörtern durch User:innen. Tweets, die zum Stichwort passen werden auf einer Karte verortet, und weiters auch die Polarität des Tweets angezeigt. Diese Studie bildet den Stand der Forschung gut ab, da der Fokus bei den Anwendungen oft auf Twitter liegt, und nur sehr wenige Interfaces zur räumlichen Analyse entwickelt wurden.

Eine weitere Studie wurde von CALÌ ET AL. 2011 durchgeführt. Auch wenn diese zeitlich schon weiter zurückliegt, ist sie trotzdem noch immer von Bedeutung. Die Anwendung „i-Tour", die dafür erstellt wurde, ist für das Routing von multi-modaler Mobilität erstellt. User:innen können dabei entscheiden, welche der Eingabemöglichkeiten sie wählen: textgesteuert, sprachgesteuert oder buttongesteuert. Das Interface, das durch Text gesteuert wird, ist wie ein Chat-Bot aufgebaut. Weiters wurde auch von CALÌ ET AL. 2011 ein Usability Testing durchgeführt. Auch hier konnten teils sehr positive Ergebnisse erzielt werden.

Aus der Literatur geht hervor, dass unterschiedliche Ansätze bestehen, textgesteuerte Interfaces für GIS oder Web-GIS umzusetzen. Nun soll erläutert werden, welche Vorteile die Entwicklung eines solchen Interfaces mit sich bringt.

Die Verwendung von GIS und Web-GIS zur visuellen Analyse haben im privaten und öffentlichen Bereich über die Jahre an Bedeutung gewonnen. (vgl. CALÌ ET AL. 2011: 920f; SETLUR ET AL. 2016: 365)

Es ist jedoch nicht nur die Anzahl an Web-GIS und GIS in Verwendung gestiegen, denn gleichzeitig ist auch die Gruppe an User:innen größer und diverser geworden. Es arbeiten nicht mehr nur GIS-Experten mit Geoinformationssystemen verschiedener Arten und Ausprägungen, sondern auch Personen, die Experten in anderen fachlichen Richtungen sind, jedoch keine oder kaum Erfahrung im Umgang mit GIS haben. Textbasierte Interfaces ermöglichen Experten aus diversen Fachrichtungen, räumliche Anwendungen zu verwende, ohne konkret die Funktionen einer räumlichen Analyse oder Ähnliches erlernt haben zu müssen. Es können durch unpräzise Statements Ergebnisse erhalten werden. (vgl. LAMPOLTSHAMMER/HEISTRACHER 2012: 82)

Eine weiter Herausforderung bei der Verwendung von GIS und Web-GIS mit herkömmlichen buttongesteuerten Interfaces ist, dass viele unterschiedliche Softwares bestehen, deren Steuerung sich in den meisten Fällen leicht bis stark unterscheidet. Das Wissen der Verwendung einer Software, das User:innen erlernt haben, ist in vielen Fällen nicht auf eine andere anwendbar. Diese Faktoren führen dazu, dass es teilweise zeitaufwendig ist, Personal in die Verwendung dieser Softwares einzuschulen, was auch zu einem höheren Kostenaufwand führen kann. Ein

textgesteuertes Interface ermöglicht hier eine Anwendungs-übergreifende Verwendung, da die Texteingabe grundsätzlich immer gleich funktioniert. (vgl. CALÌ ET AL. 2011: 920f)

Weiters kann es für User:innen herausfordern sein, ihre Bedürfnisse in konkrete Analysen umzusetzen. Statt die Fragen in eine Abfolge von Befehlen oder Button-Klicks umzusetzen, können die Anfragen einfach in natürlicher Sprache, wie die Bedürfnisse auch erdacht werden, an das System gestellt werden. (vgl. SETLUR ET AL. 2016: 366)

Im Gesamten stellen textgesteuerte Interfaces eine moderne Alternative zu traditionellen, buttongesteuerten Interfaces von GIS und Web-GIS dar. Die Kommunikation gestaltet sich natürlicher und flexibler, und die Interaktion ist transparenter. (vgl. CALÌ ET AL. 2011: 920-925)

2.5. Deep Learning als Methode für Natural Language Processing

Natural Language Processing kann durch verschiedene Methoden umgesetzt werden. Diese sind über die Jahre einer stetigen Veränderung und Weiterentwicklung unterlegen. (vgl. ZHANG/TENG 2021: 3f)

In den 1950ern, den Anfängen des Natural Language Processing versuchte man, die Aufgaben und Probleme durch das Modellieren von linguistischen Regeln zu lösen. Schnell wurde jedoch klar, dass dieser Regel-basierte Ansatz die Komplexität der Sprache nicht widerspiegelt. Dies führte zu einem Einbruch der Forschung in der Thematik Ende der 1960er. In den späten 1980ern wurde die Forschung aufgrund von neuen und modernen Methoden, wie dem Machine Learning und statistischen Ansätzen wieder aufgenommen. Diese ermöglichen, dass Algorithmen statistische und linguistische Muster lernen, wie etwa, welche Wortarten normalerweise aufeinanderfolgen. Schnell wurden mit diesen neuen Ansätzen bessere Ergebnisse als mit traditionellen und regelbasierten Methoden erzielt. Mit dieser Entwicklung fing das Feld des Natural Language Processing auch an, sich weiter von der Linguistik zu entfernen. Ende der 2000er war einer der aktuellsten Aufschwünge in der Thematik. Hier kam die Methode des Deep Learnings auf. Auch hier konnten schnell bessere Ergebnisse als durch die bisherigen Methoden, wie auch dem Machine Learning, erzielt werden. Das Deep Learning ist auch heute noch die beliebteste Methode zur Umsetzung von Natural Language Processing. (vgl. ebd.)

Dieses Kapitel soll die Grundlagen des Deep Learning im Generellen und als Methode für Natural Language Processing darstellen. Zuerst wird jedoch eine kurze Erläuterung der Darstellung von Wörtern im Machine Learning und Deep Learning gegeben, da diese Grundlagen für die Textverarbeitung mittels Computer sind.

2.5.1. Verarbeitung von Text durch Computer

Texte stellen den Input für jegliche Umsetzung von Natural Language Processing durch ML und DL dar. Um die Algorithmen jedoch trainieren zu können, müssen die Textdaten in numerische Daten umgewandelt und nummeriert werden. (vgl. HIRSCHLE 2022: 8)

Die Umsetzung dieser Darstellung wird durch die Tokenisierung erreicht, die bereits in Kapitel 2.3.1.1 definiert wurde. Sätze und Wörter werden dabei in atomare Einheiten zerteilt, die auch Tokens genannt werden. Es bestehen mehrere Möglichkeiten zur Umsetzung der Tokenisierung:

- „Character"-Tokenisierung
- Wort-Tokenisierung
- Subwort-Tokenisierung

(vgl. TUNSTALL ET AL. 2023: 29-34)

Bei der Character-Tokenisierung wird jedes Wort in alle einzelnen Buchstaben unterteilt, welche anschließend nummeriert werden und eine Art Wörterbuch darstellen. Dies ist die einfachste Form der Tokenisierung. Zweitere Methode trennt alle Sätze in die vorkommenden Wörter, welche auch wiederum nummeriert werden. (vgl. ebd.)

Ein Zwischenansatz ist die Subwort-Tokenisierung. Dabei werden einerseits häufig vorkommende Wörter als einzelne Entitäten behandelt, und seltene und komplexe Wörter in kleinere Einheiten aufgeteilt. Die Subwort-Tokenisierung wurde auch basierend auf Algorithmen erlernt. Ein bekannter Tokenizer ist „WordPiece“. Abbildung 8 zeigt den tokenisierten Output des Satzes “Tokenizing text is a core task of NLP.”. (vgl. ebd.)

```
{'input_ids': [101, 19204, 6026, 3793, 2003, 1037, 4563, 4708, 1997, 17953,
2361, 1012, 102], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

```
['[CLS]', 'token', '##izing', 'text', 'is', 'a', 'core', 'task', 'of', 'nl',
'##p', '.', '[SEP]']
```

Abbildung 8: Tokenisierter Output eines Satzes (Quelle: TUNSTALL ET AL. 2023: 34)

Der Satz wurde in Subwörter aufgeteilt, und mit je einer einzigartigen Zahl bzw. numerischen ID versehen. Dies ist in „input_ids“ dargestellt. „attention_mask“ gibt an, zu welchem Satz die IDs gehören. Zusätzlich wurden zu den tokenisierten Wörtern die Tags „[CLS]“ und „[SEP]“ hinzugefügt. Dadurch werden Start und Ende eines Satzes angezeigt. Die Symbole „##“ beschreiben, dass zwischen den Tokens kein Leerzeichen besteht. Diese Form der Tokenisierung wird bei einem auf der BERT-Architektur basierenden Modell für den gesamten Datensatz umgesetzt, das später in Kapitel 2.5.4.2 beschrieben wird. (vgl. ebd.)

2.5.2. Einführung in das Machine Learning

Um Deep Learning als Methode verstehen zu können, muss zuerst auf das Machine Learning eingegangen werden, auf das DL aufgebaut ist. Machine Learning ist ein Teilgebiet der KI Diese kann wie folgt definiert werden:

> *„Künstliche Intelligenz ist die Fähigkeit einer Maschine, menschliche Fähigkeiten wie logisches Denken, Lernen, Planen und Kreativität zu imitieren.“ (EUROPÄISCHES PARLAMENT 2023)*

Das Machine Learning selbst kann wiederum so definiert werden:

> *„Maschinelles Lernen ist ein Teilgebiet der künstlichen Intelligenz (KI, engl. Artificial Intelligence, AI) (KI), bei der Computer aus Daten lernen - üblicherweise, um ihre*

Performance für einen eng definierte Aufgabe zu verbessern -, ohne explizit dafür programmiert zu werden." (PATEL 2020: XII)

Einfach gesagt werden sogenannte Daten in einen Lernalgorithmus eingefügt, und dieser lernt von den bereits gesammelten Erfahrungen. ML-Systeme sind einfache Input-Output Systeme. Bei einem Übersetzer ist der Input beispielsweise ein Satz auf Deutsch, und der Output derselbe Satz auf Englisch. Weiters besteht der Prozess des Machine Learnings aus drei einfachen Schritten, die in Abbildung 9 dargestellt sind. (vgl. JIANG 2021: 1-6)

Abbildung 9: Prozess des Machine Learnings (Basierend auf: JIANG 2021: 3)

Der erste Schritt ist das Sammeln von Daten, sogenannten Trainingsdaten, die in NLP auch als „Corpus" bezeichnet werden können. Ebenso können sie als „x-Werte" beschrieben werden. Oft müssen Datensätze manuell gesammelt und bearbeitet werden, was diesen Schritt sehr aufwendig gestaltet. Weiters ist die Performance eines Machine Learning Systems stark von der Menge und Qualität der Daten abhängig. Meist gilt: je mehr Daten desto besser. Der zweite Schritt ist die Extraktion von „Features", also den für das System wichtige Informationen aus dem gesamten Datensatz. Diese werden auch als „y-Werte" bezeichnet. Im dritten Schritt wird ein passender Lernalgorithmus gesucht, um mathematische Modelle zu erstellen, mit denen Features aus den Trainingsdaten, wie vorhin manuell gemacht, extrahiert werden können. Lernalgorithmen können auch Modelle genannt werden. (vgl. ebd.)

Es kann im ML zwischen zwei unterschiedlichen „Problemen" unterschieden werden: der Regression und der Klassifikation. Bei der Regression sind die Output-Werte kontinuierlich, bei der Klassifikation kann der Output nur einer von mehreren diskreten, vordefinierten Werten sein. (vgl. ebd.)

Weiters kann zwischen unterschiedlichen Methoden des Machine Learnings unterschieden werden. Diese sind das überwachte und das unüberwachte Lernen. Beim überwachten Lernen sind bereits Input und Output, und deren Verbindung bekannt. Für jeden Input in den Trainingsdaten ist der passende Output bekannt, und dadurch kann auch das Anlernen des Modells gelenkt werden. Bei einer passenden Datenmenge ist diese Art des Machine Learnings sehr verlässlich und zeigt gute Ergebnisse. Hier werden jedoch auch oft im Vorhinein Personen benötigt, die den Output der Trainingsdaten manuell festlegen. Beim unüberwachten Lernen hingegen ist nur der Input bekannt. Hier werden die Trainingsdaten anhand von ähnlichen Merkmalen gruppiert und gelabelt. Es besteht die Schwierigkeit für den Algorithmus, tatsächlich zu lernen, welche Daten eine Ähnlichkeit aufweisen. Das Sammeln von Daten ist hier kostengünstig, da nur Input-Daten gesammelt werden

müssen. Das unüberwachte Lernen ist jedoch eine komplexe Methodik, die teilweise noch nicht tiefergehend erforscht ist. (vgl. JIANG 2021: 5)

Im Natural Language Processing werden meist überwachte Methoden angewendet, wie beispielsweise bei der Text Klassifizierung. Hier sind die Input-Daten sowie die zugewiesenen Klassen im Lernprozess bekannt. (vgl. LAURIOLA ET AL. 2022: 445)

Weitere Methoden und Grundlagen, die gleicherweise für das Machine Learning als auch das Deep Learning angewendet werden können, werden in den folgenden Kapiteln speziell auf das Deep Learning bezogen erläutert.

2.5.3. Einführung in das Deep Learning

Das Verfahren des Deep Learnings ist ein Teil des Machine Learnings, übertrifft die Leistung dessen jedoch in vielen Bereichen. Besonders bei der Anwendung im Natural Language Processing zeigt Deep Learning deutlich bessere Ergebnisse.

Aktuell erfreuen sich Deep Learning Methoden großer Beliebtheit in NLP. Dafür gibt es mehrere Gründe. Einer der Faktoren, die ihre hervorragende Leistung begründen ist die Fähigkeit, große Mengen an Daten zu verarbeiten. Weiters verfügen DL-Modelle über hochkomplexe Lernarchitekturen, die eine Lösung von anspruchsvollen Aufgaben ermöglichen. Ein weiterer Vorteil ist, dass DL-Modelle befähigt sind, sich sehr genau an gegebene Aufgaben und Trainingsdaten anzupassen. (vgl. HIRSCHLE 2022: 57; LAURIOLA ET AL. 2022: 443)

2.5.3.1. Aufbau von Deep Learning Modellen

Die Methoden des Deep Learning basieren auf sogenannten „neuronalen Netzen“ (NN), welche wiederum aus Neuronen aufgebaut sind. Diese können auch als Lernzellen bezeichnet werden. Eine Lernzelle eines neuronalen Netzes funktioniert wie folgt: Kommt ein Input (x-Werte) in die Lernzelle, aktiviert sie dieser. Je nach Input reagiert die Zelle. Dies geschieht mithilfe einer Aktivierungsfunktion, die den Input in eine gewisse Form bringt. Weiters verfügt jede Lernzelle über ein sogenanntes Gewicht, auch Parameter genannt. Dieses bestimmt, wie die Input-Daten von dieser beeinflusst werden. Je nach Eingabedaten und Gewichten reagieren die Neuronen anders. Gewichte beschreiben das, was bereits in der Vergangenheit gelernt wurde, und der Input die aktuelle Situation. Ein Neuron wird in Abbildung 10 dargestellt. (vgl. HIRSCHLE 2022: 58f)

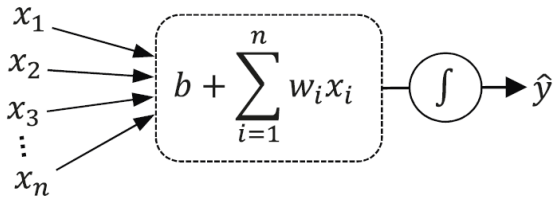

Abbildung 10: Schematische Darstellung eines Neurons (Quelle: HIRSCHLE 2022: 58)

Um von einem Neuron zu einem neuronalen Netz zu kommen, werden eine Vielzahl von Neuronen zu einem Verbund zusammengeschlossen. Die einfachste Form eines neuronalen Netzes ist in Abbildung 11 zu sehen. (vgl. HIRSCHLE 2022: 58)

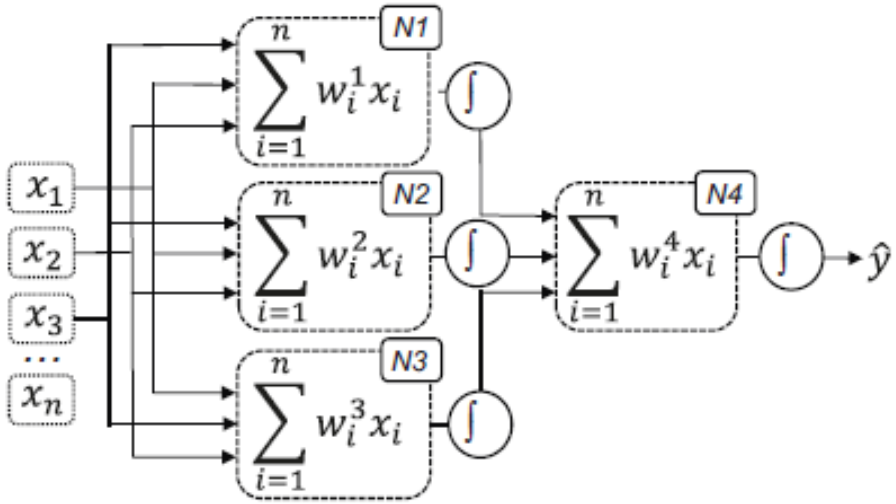

Abbildung 11: Schematische Darstellung eines neuronalen Netzes (Quelle: HIRSCHLE 2022: 59)

Ein solches Netz wird auch als ein „vollständig verbundenes Netz“ oder „Feed-Forward-Netz“ bezeichnet. Dieses Netz besteht aus drei unterschiedlichen Schichten:

- Eingabeschicht (Input Layer)
- Verdeckte Schicht (Hidden Layer)
- Ausgabeschicht (Output Layer)

(vgl. HIRSCHLE 2022: 59ff)

Aufgabe der Eingabeschicht ist das Entgegennehmen und Weitergeben der Eingabewerte an die verdeckte Schicht. Die verdeckte Schicht besteht in diesem Fall aus drei einzelnen Neuronen. Jedes Neuron erhält jeden der Eingabewerte, was es auch zu einem vollständig verbundenen Netz macht. Die Ausgabeschicht ist dafür zuständig, dass das Ergebnis der Neuronen der verdeckten Schicht empfängt und als Output ausgibt. Der Output sind sogenannte Schätzwerte, die den korrekten Wert der Input-Daten annähernd schätzen. (vgl. ebd.)

Durch das Verbinden mehrerer Neuronen wird es möglich, komplexe Aufgabenstellungen zu lösen. Ein Neuron kann trainiert werden, um je auf eine bestimmte Art von Werten zu reagieren, beispielsweise auf hohe Werte. Wenn man jedoch auch möchte, dass das neuronale Netz reagiert, wenn niedrige Werte eingefügt werden, braucht man mindestens ein zweites Neuron für mittlere Werte und ein drittes für niedrige. In der Praxis verfügen neuronale Netze über eine viel höhere Anzahl an Neuronen. (vgl. ebd.)

Der grundsätzliche Ablauf der Output-Werte ist wie folgt: Drei x-Werte werden von der Eingabeschicht in die verdeckte Schicht weitergeleitet, die in diesem Fall mit drei Neuronen angenommen wird. Jedes dieser Neuronen bekommt je alle der drei Input-Werte. Jeder dieser wird in dem Neuron mit dessen Gewicht versehen, die dadurch den Einfluss des Inputs auf die Ausgabe bestimmen. Jedes der Neuronen der verdeckten Schicht hat nun einen Output produziert. Dieser werden mit einer Aktivierungsfunktion wieder in eine konkrete Form gebracht. Anschließend kommen alle Outputs in die Ausgabeschicht, in diesem Fall mit einem Neuron. Auch hier wird jeder Wert mit einem Gewicht versehen, dass seinen Einfluss auf die Ausgabe beeinflusst. Diese Werte stellen nun die y-Werte, also Zielvariablen dar. Wieder werden die Werte mittels Aktivierungsfunktion in eine Form gebracht, die als schlussendlicher Output geeignet ist. (vgl. ebd.)

2.5.3.2. Lernvorgang eines neuronalen Netzes

Das Anlernen eines Deep Learning Modells erfolgt in zwei Phasen: der Trainingsphase und der Anwendungsphase. In ersterer wird das Modell mittels Input-Daten und bekannten Zielwerten eingestellt und angelernt, sodass die Soll-Werte der Zieldaten vom Modell vorhergesagt werden können. In der Anwendungsphase werden anschließend unbekannte Daten in das Modell eingefüttert, welches dann die ebenso unbekannten Zielwerte vorhersagen muss. Diese Phase beruht auf der „Forward-Propagation", also dem einfachen Durchleiten von Werten durch das Modell. (vgl. HIRSCHLE 2022: 61ff)

Hier wird folgend konkreter auf die Trainingsphase eingegangen. Um ein DL-Modell anzulernen, wird das sogenannte „Gradientenabstiegsverfahren" verwendet, und der Grundsatz des Trainings ist die Optimierung des Verlustes und der Gewichte. Das Verfahren wird iterativ in mehreren Wiederholungen durchgeführt. (vgl. ebd.)

Anfangs muss noch der Begriff der Verlustfunktion oder dem „Loss" oder „Verlust" geklärt werden. Der Verlust beschreibt, wie gut oder schlecht ein Modell Zielwerte vorhersagt, und wie hoch der Fehler dieser Vorhersage ist. Je höher der Verlustwert, desto schlechter ist das Modell in der Vorhersage. Die Verlustfunktion bestimmt, wie dieser Wert berechnet wird. Die meistverwendeten Verlustfunktionen der „Mean Squared Error" (MSE), bei der der mittlere Fehler

aller Iterationen berechnet wird, und die „Cross Entropy“, die eine komplexere Berechnung beschreibt, deren Erläuterung hier jedoch nicht weiter für diese Masterarbeit benötigt wird. (vgl. HIRSCHLE 2022: 61ff, 65; KROHN 2020: 111f)

Ein weiterer Bestandteil der Trainingsphase ist der Optimierer bzw. die Optimierungsfunktion. Diese ist dafür verantwortlich, dass die Gewichte in die vorherig durch die Verlustfunktion bestimmte Richtung gedreht werden. Durch die Optimierungsfunktion fangen die Modelle erst das tatsächliche Lernen an. (vgl. HIRSCHLE 2022: 66)

Das Ziel beim Anlernprozess eines Deep Learning Modells ist es, den minimalen Verlust zu erreichen, der auch durch das Minimum der Verlustfunktion beschrieben wird. Dazu wird das bereits erwähnte Gradientenabstiegsverfahren verwendet. Der Gradient beschreibt die Steigung der Funktion, die verwendet wird. Das Anlernen eines Deep Learning Modells basiert auf dem Iterativen Bewegen von Werten zum Minimum der Verlustfunktion. HIRSCHLE 2022: 61ff beschreibt diese Verlustfunktion als ein Tal, an dessen tiefsten Punkt man gelangen möchte. Beim Gradientenabstiegsverfahren startet man an einem beliebigen Punkt der Funktion. Jedes Gewicht der Neuronen bekommt einen zufälligen Wert zugewiesen. Bei jedem Lernschritt des iterativen Verfahrens, auch Epoche genannt, wird versucht, tiefer in das Tal zu gelangen. Dies wird durch das Einstellen der Gewichte des Modells erreicht. Am Ende eines Lernschritts wird der Verlust gespeichert und dem Modell zurückgeführt. Dieser wird als Referenz für die nächste Epoche verwendet. Dies ist die sogenannte „Back-Propagation“, also das Zurückführen von Daten, das Gegenteil der Forward-Propagation. So wird im iterativen Verfahren das Minimum an Verlust angestrebt. (vgl. HIRSCHLE 2022: 61ff)

2.5.3.3. Hyperparameter eines Deep Learning Models

Beim Anlernprozess eines Deep Learning Modells gibt es unterschiedliche Einstellungen, die verändert werden können, um zum minimalen Verlust zu gelangen, und so die Qualität eines Modells zu beeinflussen. Diese Einstellungen werden auch „Hyperparameter“ genannt. Die Suche nach den bestmöglichen Kombinationen von Hyperparametern wird „Hyperparameter Tuning“ oder „Hyperparameter Optimization“ genannt. (vgl. BARTZ ET AL. 2023: 11-15) Diese Begriffe können getrennt definiert, aber auch synonym verwendet werden.

Beim Deep Learning werden die Hyperparameter des Models vor dem Trainingsprozess selbst bestimmt. Es besteht eine Vielzahl von Hyperparametern, die festgelegt werden können:

- Anzahl der Schichten
- Aktivierungsfunktion
- Dropout
- Lernrate (Learning Rate)
- Anzahl der Epochen
- Optimierungsfunktion
- Verlustfunktion
- Batch Size

(vgl. BARTZ ET AL. 2023: 58-66)

Folgend wird eine Auswahl von Parametern genauer beschrieben, die auch für die praktische Umsetzung der Arbeit relevant sind.

Einer der essenziellen Hyperparameter bei einem Deep Learning Model ist die Lernrate. Diese bestimmt, wie groß die Schritte sind, mit denen versucht werden soll, das Minimum der Verlustfunktion zu erreichen. Hier muss ein Mittelmaß zwischen einer zu hohen und zu niedrigen Lernrate gefunden werden. Ist die Lernrate zu niedrig, ist das Anlernen besonders ressourcen- und zeitaufwendig. Ist die Lernrate hingegen zu hoch, kann das Minimum des Verlusts leicht übersehen werden. Die Werte liegen typischerweise zwischen 1e-2 (0,01) und 1e-6 (0,000001). (vgl. BARTZ ET AL. 2023: 63; KROHN 2020: 118f)

Die Epochen bestimmen die Anzahl der Iterationen des Lernprozesses. Eine Epoche beschreibt die Erneuerung der Gewichte. Die Epochen beeinflussen primär, wie lange der Anlernprozess dauert. Grundsätzlich sind höhere Werte besser, jedoch kann hier auch oft eine Überanpassung, die im folgenden Kapitel beschrieben wird, eintreten. (vgl. BARTZ ET AL. 2023: 64)

Die Funktion des Optimierers (Optimizer) bzw. die Optimierungsfunktion wurde bereits im vorherigen Kapitel beschrieben. Es existieren mehrere vorgefertigte Optimierer. Die beliebtesten sind „SGD“ (Stochastic Gradient Descent), von dem auch „RMSProp“ (Root Mean Square Propagation) und der Optimierer „Adam“ (Adaptive Moment Optimization) abgeleitet werden. (vgl. BARTZ ET AL. 2023: 58-66)

Die Definition des Verlustes wurde bereits in Kapitel 2.5.3.2 erläutert. Bei den Hyperparametern muss eine passende Verlustfunktion gefunden werden die bestimmt, wie der Verlust berechnet

werden soll. Hier gibt es die folgenden Funktionen, die je nach Deep Learning Task (Regression oder Klassifikation) gewählt werden:

- „Mean Squared Error“ (MSE)
- „Binäre Kreuzentropy“ (Binary Crossentropy)
- „Kategoriale Kreuzentropy“ (Categorical Crossentropy)

Erstere wird bei Regressionen verwendet, die zweite und dritte Funktion bei der Klassifikation. Dabei wird die binäre Kreuzentropy bei Klassifizierungen mit zwei Klassen, und die kategoriale Kreuzentropie bei mehreren Klassen angewendet. (vgl. HIRSCHLE 2022: 65)

Der letzte Hyperparameter, der besprochen wird, ist die „Batch Size“. Batches sind kleine Teildatensätze aus dem gesamten Datensatz. Auch hier muss zwischen zu großen und zu kleine Batch Sizes abgewogen werden. Ist die Batch Size zu groß, kann das Modell einerseits in einem lokalen Minimum gefangen bleiben, und somit nicht das globale Minimum erreichen. Ist sie jedoch zu klein, ist es schwerer, jemals zu einem Minimum zu gelangen. Außerdem ist zu bedenken, dass das Modell länger lernt, je kleiner die Batch Size ist. Dadurch ist das Training auch ressourcenaufwendiger. Typische Batch Sizes liegen zwischen 16 und 128 Batches. (vgl. KROHN 2020: 119-124)

Es bestehen mehrere Optionen, das Hyperparameter Tuning umzusetzen. Drei der beliebtesten sind:

- Manuelles Suchen
- „Grid Search“
- „Random Search“

Der erste Ansatz ist das manuelle Suchen der optimalen Kombination von Hyperparametern. Dabei wird vor allem „Trial-and-Error“ als Methode eingesetzt. Dieser Ansatz kann bei guter Recherche auch zu guten Ergebnissen führen. Jedoch kann dies auch eine sehr zeitaufwendige Methode sein, und kann bei falscher Interpretation der Ergebnisse auch zu Problemen führen. Bei der Grid Search wird der Raum, in dem nach Kombinationen gesucht wird, durch ein Gitter abgebildet. Jeder Punkt dieses Gitters wird evaluiert. Bei der Random Search hingegen werden Werte dem Zufall nach ausgewählt. Der Vorteil der Random Search gegenüber der Grid Search ist vor allem, dass sie weniger Ressourcen und Zeit benötigt, da nicht der ganze Raum abgesucht wird. (vgl. BARTZ ET AL. 2023: 76f)

2.5.3.4. Evaluierung von Deep Learning Modellen

Um die Performance von Machine Learning und Deep Learning Modellen evaluieren zu können bestehen mehrere Metriken.

Die Grundlage dieser Metriken sind die vier Klassen der Performance. Diese werden in einer „Confusion Matrix" dargestellt, wie sie in Tabelle 1 dargestellt ist. Dabei werden die geschätzten Zielwerte des Modells den tatsächlichen Zielwerten gegenübergestellt. Hier werden die Klassen 0 (negativ) und 1 (positiv) vorhergesagt.

Tabelle 1: Grundkonzept der Confusion Matrix (Basierend auf: HIRSCHLE 2022: 32)

		Vom Modell geschätzte Zuordnung	
		0	**1**
Wahre Zuordnung	**0**	True negative	False positive
	1	False negative	True positive

Die erste Klasse ist „True Positive". Dies schließt alle Werte ein, die den Wert von 1 haben und auch als solche vorhergesagt wurden. Bei „True Negative" wurden die Werte als 0 vorhergesagt, und sind tatsächlich auch die Klasse 0. „False Positive" beschreibt alle Werte, bei denen die Klasse 1 vorhergesagt wurde, die jedoch eigentlich der Klasse 0 angehören. Abschließend beschreiben die „False Negatives" alle Vorhersagen, die als 0 klassifiziert wurden, jedoch Klasse 1 sind. (vgl. HIRSCHLE 2022: 32)

Aus den nun eingeordneten Werten kann die „Accuracy" berechnet werden. Sie ist die einfachste Metrik und beschreibt die Häufigkeit der korrekten Klassifikationen, in dem diese durch die gesamte Anzahl an Beispielen geteilt wird. Folgend ist die Formel dargestellt:

$$accuracy = \frac{true\ positives + true\ negatives}{Gesamtzahl\ der\ Fälle}$$

(vgl. HIRSCHLE 2022: 32; KUBAT 2015: 213f)

Da die Accuracy jedoch oft die Performance eines Modells nicht korrekt schätzt, werden in der Praxis zwei weitere Metriken in Betracht gezogen: die „Precision" und der „Recall". Beide haben einen Wertebereich von Null bis Eins, wobei Null schlecht und Eins gut ist. Die Precision beachtet die Anzahl der als positiv klassifizierten Werte. Je mehr dieser positiven Werte eigentlich der negativen Klasse angehören, desto niedriger die Precision. Sie kann auch beschrieben werden als die Wahrscheinlichkeit, dass ein tatsächlich positives Beispiel auch als solches klassifiziert wurde.

Die Formel der Precision lautet wie folgt:

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

(vgl. ebd.)

Der Recall beschreibt hingegen den Anteil der tatsächlich korrekt als positiv klassifizierten Fälle. Es ist also der Anteil der True Positives an allen als positiv vorhergesagten Fälle. Die Formel ist die folgende:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

(vgl. HIRSCHLE 2022: 33; KUBAT 2015: 216f)

Eine weitere Metrik besteht, die die Precision und den Recall kombiniert: der F1-Score. Dabei werden beide Metriken in einem Wert ausgeglichen dargestellt. Der F1-Score ist wichtig, wenn bei der Evaluierung eine hohe Precision und ein hoher Recall von Bedeutung sind. Auch hier gehen die Werte von Null bis Eins, und je höher der Ergebniswert ist, desto besser ist die Performance des Modells. (vgl. KUBAT 2015: 218ff)

2.5.3.5. Herausforderungen beim Anlernen von Deep Learning Modellen

Während die Anwendung von neuronalen Netzwerken große Vorteile mit sich bringt, bestehen jedoch auch einige Herausforderungen, die bedacht werden müssen. So ist der Fakt, dass neuronale Netzwerke komplexe Beziehungen lernen können, gleichzeitig einer der größten Vorteile und größten Problematiken. Netze fangen dadurch mit der Zeit an, sich an die Trainingsdaten zu sehr anzupassen. Das Phänomen ist die „Überanpassung", auch „Overfitting" genannt. Hat sich ein Modell überangepasst, ist es nicht mehr nutzbar, da es keine neuen, und unbekannten Daten mehr erkennen kann. (vgl. HIRSCHLE 2022: 83-87)

Die Überanpassung entsteht durch zu kleine Datenmengen oder Daten, die nicht repräsentativ genug sind. Deshalb ist es wichtig, dass die Input-Daten eine möglichst große Diversität aufweisen, sodass eine große Menge an Wörtern und Kombinationen erlernt werden kann, und zwischen Mustern und Ausreißern unterschieden werden kann. (vgl. ebd.)

Auch die Herausforderung des Gegenteils, das „Underfitting" oder auch „Unteranpassung" genannt, besteht. Diese tritt auf, wenn die Architektur eines Modells nicht komplex genug ist, oder zu wenig Schichten hat. Dadurch können die komplexen Beziehungen der Daten nicht mehr erlernt werden. (vgl. ebd.) Abbildung 12 zeigt die zeigt die Phänomene.

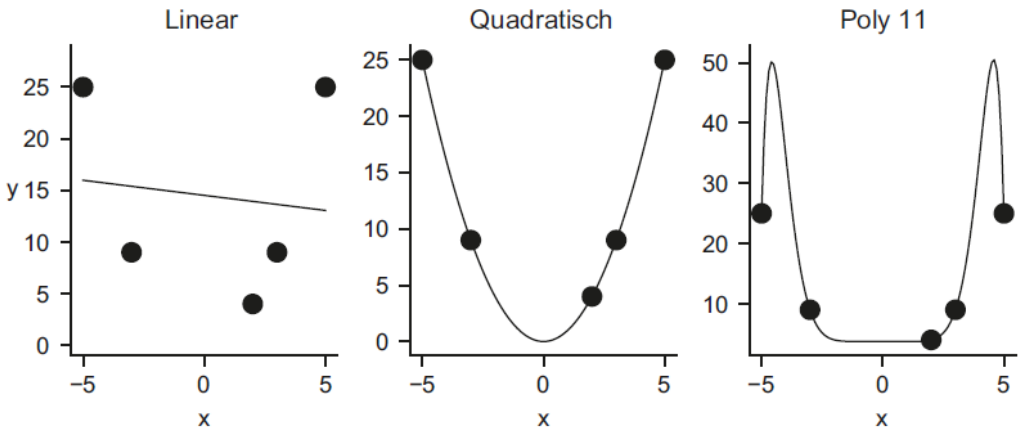

Abbildung 12: Unteranpassung (Links), lernendes Modell (Mitte) und Überanpassung (Rechts) (Quelle: HIRSCHLE 2022: 83-87)

Die Punkte in der Abbildung beschreiben die Datenpunkte. Links ist die Überanpassung zu sehen, bei dem sich das Modell nicht an die Daten anpasst. In der Mitte bildet das Modell die Daten passend ab. Rechts ist die Überanpassung zu sehen, bei der sich das Modell zu gut anpasst. (vgl. ebd.)

Ziel bei der Wahl der Architektur eines funktionierenden Models, das weder anfällig für Überanpassung noch Unteranpassung ist, ist einen Kompromiss zwischen zu komplexen und zu einfachen Modellen zu finden. Dies wird auch „Bias-Variance Tradeoff“ genannt. „Bias“ beschreibt den Fehler, der durch ein zu einfaches Modell entsteht, und „Variance“ ist der Fehler, der aus zu komplexen Modellen und zu wenigen Daten resultiert. Hier muss eine Balance zwischen den beiden Phänomenen gefunden werden, was eine große Herausforderung ist, da die Komplexität der Daten meist nicht bekannt ist. Ebenso ist ein in der Theorie passendes Modell nicht unbedingt eines, das in der Praxis funktioniert. (vgl. ebd.)

Um die Überanpassung oder Unteranpassung überprüfen zu können werden Testdaten benötigt, die im Vergleich zu den Trainingsdaten nicht im Anlernprozess des Modells verwendet wurden und somit unbekannte Daten sind. (vgl. ebd.)

Es bestehen unterschiedliche Möglichkeiten, Überanpassung oder Unteranpassung zu verhindern. Eine dieser ist das „Early Stopping“. Dabei wird der Anlernprozess vor seinem eigentlichen Ende gestoppt. Dies kann basierend auf verschiedenen Werten wie beispielsweise der Accuracy gemacht werden. Es kann festgelegt werden, bei welcher Bedingung der Prozess stoppt. Steigt beispielsweise der ausgesuchte Wert innerhalb einer definierten Anzahl von Epochen nicht, wird das Training gestoppt. So können negative Entwicklungen vermieden werden, und generalisierbare Beziehungen im Modell jedoch bereits erlernt sein. Schlussendlich kann das bestmögliche Modell für den Anwendungszweck verwendet werden. (vgl. ebd.)

Eine weitere Möglichkeit, zu wenig oder zu gleiche Trainingsdaten auszugleichen und somit das Overfitting oder Underfitting zu vermeiden ist die „Data Augmentation". Ziel der Methode ist es, die Menge an Daten durch die Erstellung leicht abweichender Kopien der bestehenden Daten synthetisch zu erstellen. Der Fokus liegt aber ebenso darauf, mehr Diversität in die Trainingsdaten zu bringen, um das Erlernen von generalisierbaren Informationen zu ermöglichen. Es bestehen unterschiedlichste Methoden der Data Augmentation, wie das „Paraphrasing", bei dem die Wörter und Grammatik eines Satzes verändert werden, und das „Noising", bei dem beispielsweise die inhaltliche Aussage eines Satzes leicht verändert wird. (vgl. LI ET AL. 2022: 4ff)

2.5.4. Deep Learning Modelle

Es bestehen unterschiedliche Arten an neuronalen Netzen. Folgend werden drei verschiedene Netze erläutert, um herauszuarbeiten, welche der Arten für die eigene Masterarbeit verwendet werden soll.

2.5.4.1. Rekurrente und Konvolutionale Netze

Zwei der bekanntesten und meistverwendeten neuronale Netze sind das rekurrente neuronale Netz (RNN) und das konvolutionale neuronale Netz (CNN).

RNN sind besonders für die Verarbeitung von Textdaten geeignet. Sie beachten, dass zwischen den Wörtern eines Satzes Beziehungen und Abhängigkeiten bestehen, und die Bedeutung der Wörter selbst oft erst aus der Position im Satz oder dem restlichen Kontext hervorgeht. Auch die Architektur der rekurrenten Netze ist auf die Eigenschaften von Text- und Sequenzdaten zugeschnitten. Die Modelle sind schlank, lernen generalisierbare Beziehungen, und merken sich signifikante Wörter, wie Verneinungen, auch über mehrere Positionen und in Kombination mit anderen Wörtern. Ein weiterer Vorteil von rekurrenten Netzen ist, dass jedes Neuron über dasselbe Gewicht verfügt, wodurch das Netz ressourcensparend arbeitet, da dieses nur einmal bestimmt werden muss. Ebenso wird bei jedem Lernschritt der aktuelle Zustand der Lernzelle gespeichert, der beim nächsten Lernschritt als zusätzlicher Faktor eingebracht werden kann. (vgl. HIRSCHLE 2022: 93-98)

Eine weitere Form der neuronalen Netze sind CNN. Diese sind grundsätzlich auf Bildmaterial zugeschnitten, sind aber trotzdem auch gut anwendbar für Textdaten. Die Vorteile der konvolutionalen Netze sind, dass sie die Zusammenhänge in den Input-Daten finden, und weiters verhältnismäßig wenige Parameter verwenden. Hierbei lernt die Lernzelle den Gewichtungsfaktor für einen kleinen Ausschnitt. Diese Lernzelle wird schlussendlich für den gesamten Datensatz angewandt. (vgl. HIRSCHLE 2022: 121-125)

2.5.4.2. Transformers

Eine der aktuell beliebtesten Arten von Deep Learning Modellen sind „Transformer Modelle". Diese werden auch Pre-Trained Transformers Models (PTM) genannt. Diese haben gegenüber den anderen neuronalen Netzen einige Vorteile. Zum einen verfügen sie über mehrere Milliarden unterschiedliche Parameter, die es ermöglichen, mit riesigen Datenmengen umzugehen. Ein weiterer Vorteil ist die Architektur des „Self-Attention Mechanismus", auf den im folgenden Kapiteln noch konkreter eingegangen wird. Weiters sind die Transformer Modelle vortrainiert. Dadurch sind sie auf universelle Sprache bereits angelernt, wodurch sie über ein allgemeines Sprachverständnis verfügen. Anschließend sind sie mit verhältnismäßig wenig Ressourcen auf eigene Anwendungen und Aufgabenstellungen trainierbar. (vgl. HIRSCHLE 2022: 207f)

Funktion von Transformer Models

Die Basis von Transformer Models ist die Encoder-Decoder Architektur, die in Abbildung 13 dargestellt ist.

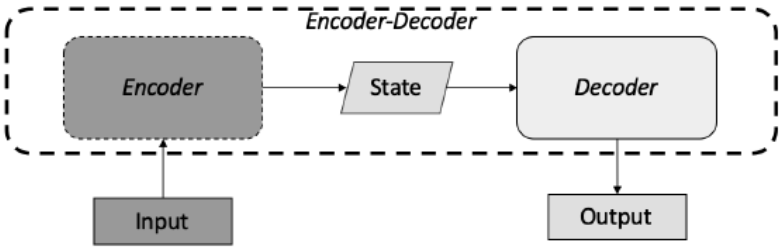

Abbildung 13: Encoder-Decoder Architektur eines Transformer-Models (Quelle: KAMARATH et al. 2022: 12)

Der Encoder bekommt eine Sequenz als Input, die in unterschiedlichen Zeichenlängen geliefert werden kann. Die Sequenz wird transformiert und in eine fixierter Länge gebracht. Der Decoder nimmt die Daten mit fixierter Länge und formt sie in einen Output mit wiederum variabler Länge um. (vgl. KAMATH ET AL. 2022: 11f)

Das zentrale Konzept der Transformer Modelle ist die „Self-Attention". Dabei wird für jedes Wort in einem Satz bestimmt, welche Verbindung dieses zu allen anderen Wörtern des Satzes hat. Bei Transformern werden mehrere Self-Attention-Schichten zu einer „Multi-Head-Attention-Schicht" zusammengefügt, wodurch gleichzeitig mehrere Merkmale eines Wortes, wie beispielsweise Konjugationen, Zeiten etc. erlernt werden können. (vgl. KAMATH ET AL. 2022: 22f)

Ebenso gibt es im Transformer Modell „Normalization Layers", die zur Stabilisierung des Trainingsprozesses beitragen. (vgl. KAMATH ET AL. 2022: 26)

Die gesamte Architektur eines Transformer Models ist in Abbildung 14 dargestellt. Die Erläuterung der exakten Funktion dieses Models würde für diese Masterarbeit zu tief greifen, deswegen wird versucht, die grundsätzlichen Funktionen und Vorteile zusammenzufassen.

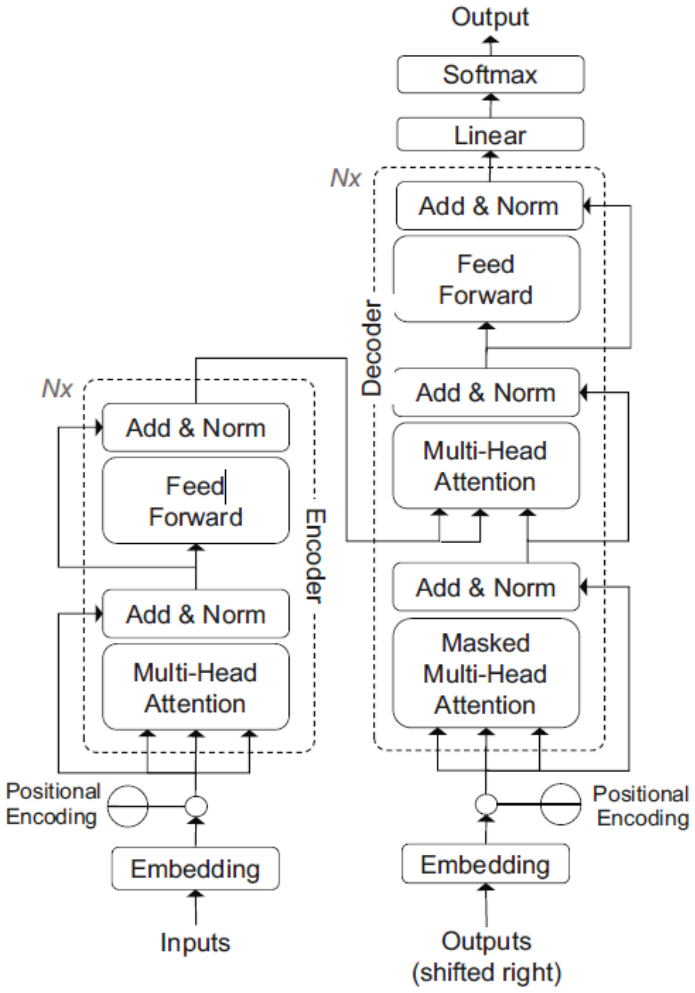

Abbildung 14: Architektur eines Transformer-Modells (Quelle: HIRSCHLE 2022: 211)

Teil des Transformer Models, und auch der erste Schritt sind das „Embedding“ und „Positional Encoding“. Das Embedding ist eine Möglichkeit, um Wörter und deren semantische Bedeutung numerisch zu modellieren. Dadurch werden die Positionen der Wörter mitbeachtet. Da die Reihenfolge von Wörtern in NLP eine große Rolle spielt, muss diese Information auch weitergegeben werden. (vgl. KAMATH ET AL. 2022: 20f)

Anschließend folgt der bereits beschriebene Encoder-Block. Dieser enthält den Multi-Head Attention Layer, und einen Normalization Layer. Danach folgen vollständig verbundene Schichten (Feed-Forward Netz), und wieder ein Normalization Layer. In der Praxis können mehrere Encoder-Blöcke hintereinander geschalten sein. Der Output aus dem Encoder wird schlussendlich in den Decoder-Block eingespeist, und die Daten in einem ähnlichen Prozess wie im Encoder verarbeitet. (vgl. HIRSCHLE 2022: 211f)

Eine weitere Besonderheit von Transformer-Modellen ist, dass diese über eine maximale Länge des Inputs verfügen. Das bedeutet, dass nur Input-Texte unter einer Länge, wie beispielswiese 512 Tokens, also wenige Absätze, von diesen verarbeitet werden kann. Dieser Faktor wird in der Tokenisierung durch das „Padding" und die „Truncation" umgesetzt. Mit letzterem werden alle Sequenzen, die Länger als die Mindestlänge eines Modells auf diese gekürzt. Durch das Padding werden wiederum Sequenzen die kürzer als diese sind mit dem numerischen Wert „0" auf die maximale Länge aufgefüllt. (vgl. TUNSTALL et al. 2023: 28)

Transfer Learning

Wie bereits erwähnt werden die Modelle durch große Datensätze vortrainiert. Dies ermöglicht es Anwender:innen, diese komplexe Modelle auf eigene Anwendungen zu tunen. Dabei wird das zuvor antrainierte Wissen von universeller Sprache wie etwa Deutsch genutzt, und auf die eigene Aufgabe angewendet. Der Prozess dieses Anlernens wird auch „Transfer Learning" genannt. Abbildung 15 zeigt diesen Prozess. (vgl. TUNSTALL ET AL. 2023: 28f)

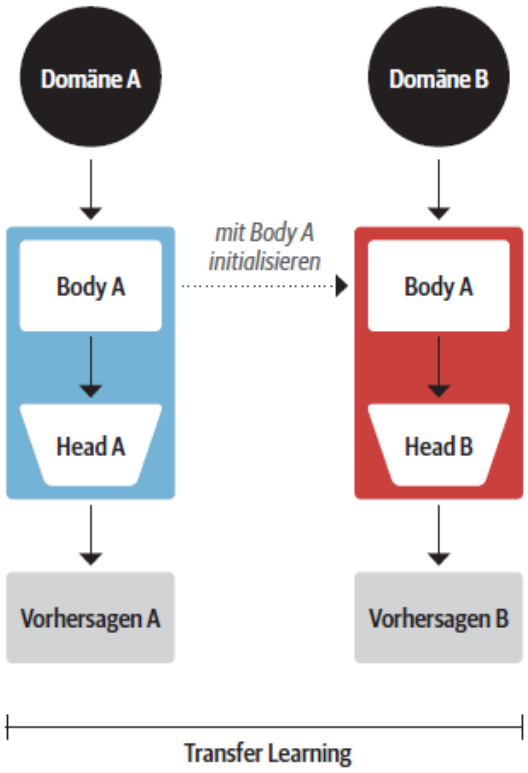

Abbildung 15: Darstellung des Transfer Learnings (Quelle: TUNSTALL et al. 2023: 28)

Das Transformer Modell wird dabei in den „Body" (Körper) und den „Head" (Kopf) geteilt. Der Body ist der bereits vortrainierte Teil und lernt mit seinen bereits eingestellten Gewichten. Damit wird ein neues Modell für die eigene Aufgabe initialisiert. Der Head stellt ein auf die Aufgabe angepasstes neuronales Netz dar, der einfach ausgetauscht werden kann. Dies ermöglicht die

effiziente Verwendung der Modelle für eigene, hochqualitative Anwendungen. (vgl. TUNSTALL et al. 2023: 28f)

Beispiele von Transformer Modellen

Es bestehen unterschiedlichste Transformer Modelle, die für verschiedene Anwendungen geeignet sind. Wie bereits in diesem Kapitel erwähnt, bestehen Transformer Modelle aus Encodern und Decodern. Es bestehen Modelle, die rein aus dem Encoder-Teil, rein aus dem Decoder-Teil, oder beiden der Teile aufgebaut sind. Die Aufteilung der unterschiedlichen Modelle ist in Abbildung 16 zu sehen.

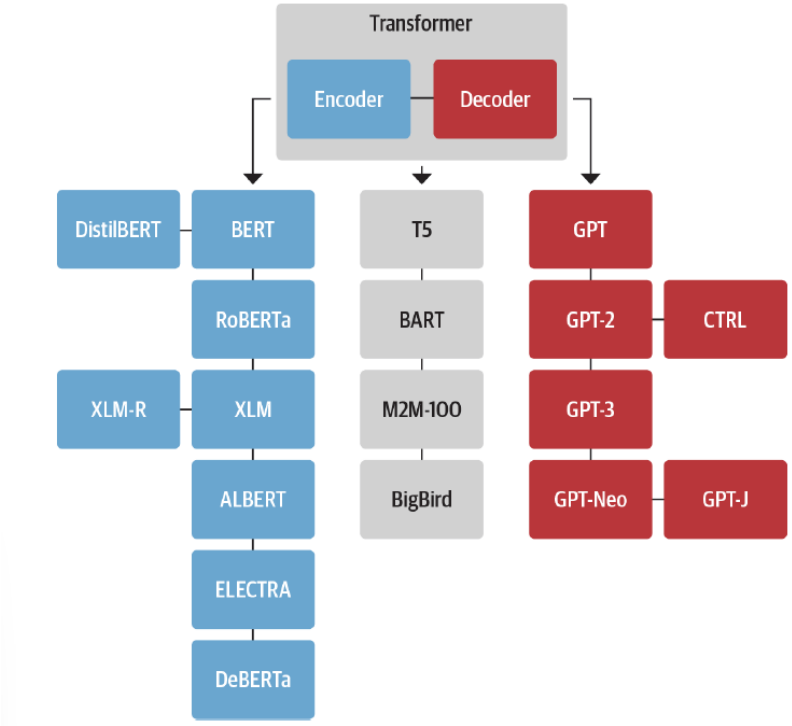

Abbildung 16: Arten von Transformer-Modellen (Quelle: TUNSTALL ET AL. 2023: 79)

Encoder-Modelle sind grundsätzlich dafür geeignet, Sprache zu verstehen, die Decoder-Modelle generieren hingegen Sprache. Die anderen Modelle sind fähig, beide Tasks durchzuführen. (vgl. TUNSTALL ET AL. 2023: 79-84)

Eines der bekanntesten Modelle und das erste rein Encoder-basierte Modell ist „BERT“ (Bidirectional Encoder Representation). Dieses ist durch sein einfaches Design der Architektur gekennzeichnet. Dadurch ist es auch auf viele verschiedene Aufgabenstellungen anwendbar. Das Fine-Tuning des BERT-Modells auf eine spezifische Aufgabe ist ein einfaches Vorhaben, da grundsätzlich nur der Input und Output korrekt miteinander in Verbindung gebracht werden müssen. Dadurch ist BERT für viele verschiedene Anwendungen mit einem Satz als Input, wie die Text Klassifizierung und Named Entity Recognition, geeignet. (vgl. KAMATH ET AL. 2022: 43-48)

Mit der Zeit wurden unterschiedliche Abwandlungen und Verbesserungen des BERT-Models entwickelt. Eine dieser Verbesserungen ist „RoBERTa“ (Robustly Optimized BERT Pre-training Approach). Dieses Modell unterscheidet sich durch mehrere Faktoren zu BERT. Bei RoBERTa wurden beispielsweise andere Methoden zum Vortrainieren des Modells verwendet. Weiters wurden größere Mengen an Input-Daten eingespeist. Dadurch zeigt das Modell teils eine bessere Performance als das herkömmliche BERT-Modell. (vgl. KAMATH ET AL. 2022: 48)

3. Web-GIS für Implementierung

Ein zentraler Bestandteil der Praxis dieser Masterarbeit ist das Web-GIS, das von einem buttongesteuerten auf ein textgesteuertes Interface durch NLP umgebaut werden soll.

Dieses Kapitel soll einen Einblick in die Anwendung, die verwendet und adaptiert wird, genauso wie einen Überblick über dessen Interface und Funktionen geben. Anfangs wird ebenso das Projekt, im Rahmen dessen die Anwendung entwickelt wurde, vorgestellt.

3.1. Projekt TEMPUS

Das Projekt, in dessen Rahmen das Web-GIS erstellt wird, ist das Projekt „TEMPUS“. Dieses ist ein gesponsortes Forschungsprojekt des deutschen Bundesministeriums für Digitales und Verkehr (BMDV). Die Hauptakteure sind unteranderem die Landeshauptstadt München und der Freistaat Bayern, weitere Projektpartner sind Firmen wie die BMW-Group und Trafficon, die auch Auftraggeber dieser Masterarbeit sind. Das Untersuchungsgebiet des Projektes ist die Stadt München. Die Projektlaufzeit ist von Jänner 2021 bis Juni 2023. (vgl. TRAFFICON 2023)

Ziel des Projektes ist eine Erprobung des automatisierten Fahrens im Individualverkehr und auch Personennahverkehr. Dies ist gekoppelt an eine zunehmende Digitalisierung des Straßenverkehrs, die hier ebenso erweitert werden soll. Es sollen innovative Technologien in Real-Situationen getestet werden. Gleichzeitig wird auch eine Akzeptanzforschung in der Bevölkerung gegenüber dem automatisierten Fahren durchgeführt werden. Dies soll zu einer sichereren und effizienteren Zukunft im Verkehr beitragen. Abbildung 17 zeigt unterschiedlichste Komponenten und Ziele des TEMPUS-Projekts. (vgl. MOBILITÄTSREFERAT 2023)

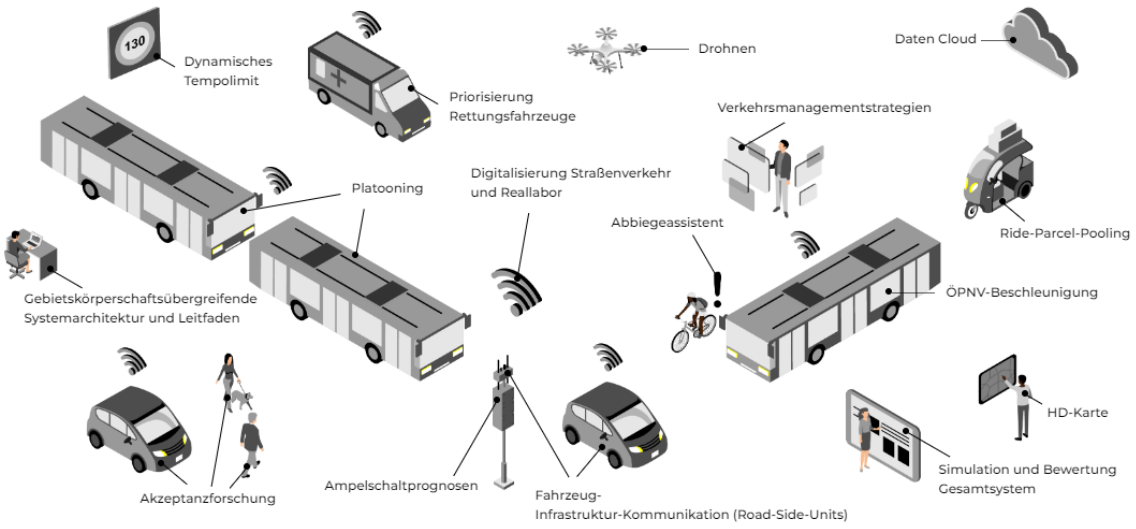

Abbildung 17: Projektziele und Bestandteile von TEMPUS (Quelle: Mobilitätsreferat 2023)

Die Aufgabe von Trafficon ist es, Verkehrsmanagementstrategien zu planen und erstellen. Dies wird durch die Evaluierung aktueller Verkehrsdaten und der Analyse deren Status Quo erreicht. Dadurch können Autoritäten unteranderem Stauzonen, Vorrangschaltungen von Ampeln, und auch bevorzugte Routen von Verkehrsteilnehmern basierend auf Real-Time Verkehrsdaten analysiert werden. Durch diese Analysen können anschließend Verkehrsmanagementstrategien erstellt, und vorhersagbare und auch spontane Verkehrsevents kommuniziert werden. Dies ermöglicht eine frühzeitige Vermeidung oder Auflösung von Staus und Verkehrsüberlastung, und Emissionen können reduziert werden. (vgl. TRAFFICON 2023)

3.2. Dynamic Traffic Monitor

Für die Umsetzung der Anwendung wird eine bestehende Anwendung der Firma Trafficon verwendet, die im Rahmen des bereits vorgestellten TEMPUS-Projektes entwickelt wird. Diese Anwendung wird auch „DTM“ (Dynamic Traffic Monitor) genannt. Konkreter definiert ist die Anwendung ein Web-GIS für das Verkehrsmonitoring und die Verkehrsplanung. Auch kann dieses als Tool zur visuellen Analyse, wie in Kapitel 2.1.3 beschrieben, bezeichnet werden. Auch hier ist das Untersuchungsgebiet, wie beim gesamten Projekt, die Stadt München. Analysiert werden können Real-Time Daten, die anschließend auch historisiert abrufbar sind. Es werden Ergebnisse zur visuellen Analyse durch Karten und Diagramme dargestellt. Durch verschiedene Analysen soll ermöglicht werden, Verkehrssituationen analysieren und somit Strategien für eine effizienteren und sichereren Verkehr entwickeln zu können.

Die Zielgruppe der Anwendung sind Personen von Autoritäten im öffentlichen Bereich, die sich mit der Verkehrsplanung und dem Verkehrsmanagement beschäftigen. Die User:innen haben

einen Hintergrund in der Verkehrsplanung, mit Fokus auf die IT oder das Consulting. Die Personen sind also Experten in einer fachlichen Richtung, die jedoch keine oder nur wenig Erfahrung im Umgang mit GIS haben.

3.2.1. Funktionen und Design des DTM

Wie bereits erwähnt stellt das Web-GIS eine Plattform für Verkehrsmonitoring dar. User:innen können unterschiedliche Analysen und Abfragen in der Anwendung durchführen. Folgend werden alle Möglichkeiten der Analyse erläutert und dargestellt. Im selben Zug wird das Interface der ursprünglichen Anwendung vorgestellt. Alle Informationen und Abbildungen stellen den Stand der Entwicklung zur Zeit der Erstellung der Masterarbeit dar.

In der Anwendung ist es möglich, verschiedene Informationen zur aktuellen und vergangenen Verkehrssituation in München zu bekommen. Dabei können drei unterschiedliche Arten von Anfragen gestellt werden:

- Monitoring Live
- Monitoring Historisch
- FCD (Floating Car Data) -Verkehrsanalyse

In Tabelle 2 ist ein Überblick über alle drei Anfragearten und die dabei benötigten Informationen dargestellt. Die Funktion und die Abfragen im Generellen werden in späterer Folge genauer beschrieben.

Tabelle 2: Mögliche Anfragearten an das Web-GIS und die dafür benötigten Angaben

	Monitoring Live	***Monitoring Historisch***	***FCD-Verkehrsanalyse***
Zeit	0	1	2
Datum	0	1	2
Straße	0	0	2 (pro Route)
Intervall	0	0	1
Kennwert	0	0	1

Bei der Live-Anfrage werden weder Straßen noch Datum und Zeit benötigt. Die Abfrage der historischen Daten erfordert je eine Zeitangabe, und eine Datumsangabe. Die komplexeste Abfrage ist die der FCD-Verkehrsanalyse. Diese erfordert auch die meisten Informationen. Es werden dazu je zwei Zeitangaben und Datumsangaben benötigt, um daraus einen Zeitraum, begrenzt durch Datum und Zeit zu erstellen. Weiters werden pro Route, die analysiert werden soll, je ein Start- und ein Endpunkt in Form einer Straße benötigt. Zusätzlich muss hier je ein Intervall und ein Kennwert angegeben werden. Mehr dazu in den folgenden Kapiteln.

3.2.1.1. Monitoring Live und Monitoring Historisch

Die ersten beiden Abfragemöglichkeiten sind die Analyse der Live-Daten und der historisierten Daten. Die Live-Abfrage zeigt die aktuelle Verkehrssituation in München. Das Ergebnis wird in einer kartographischen Darstellung gezeigt, wobei die Straßen, die im Netz enthalten sind, mit Werten versehen werden. Es kann zwischen den folgenden Kennwerten gewählt werden

- Level of Service (LOS)
- Vertrauenskennwerte
- Geschwindigkeitsüberschreitungen
- Geschwindigkeitsunterschreitungen

Der Wert, der initial angezeigt wird, ist das LOS. Dieses beschreibt die Qualität an Service, die die Straßenverkehrsinfrastruktur den Verkehrsteilnehmer:innen bietet. (vgl. PANDEY/BISWAS 2022: 49) Alle weiteren Kennwerte können anschließend ausgewählt werden. Die Anzeige erfolgt auf Basis von Real-Time Daten. Diese Daten werden anschließend historisiert. Dadurch sind sie für die zweite Art der Abfrage, die historische Abfrage, bereitgestellt. Dabei kann die Verkehrslage zu einem Zeitpunkt an einem Tag in der Vergangenheit abgefragt werden. Auch hier können dieselben Kennwerte wie bei der Live-Abfrage angezeigt werden.

Zusätzlich zu beiden Abfragearten können Verkehrsmeldungen und bestehende Ampelanlagen angezeigt werden.

Abbildung 18 zeigt das Interface, in dem die Live-Analyse und Historische Analyse durchgeführt werden können.

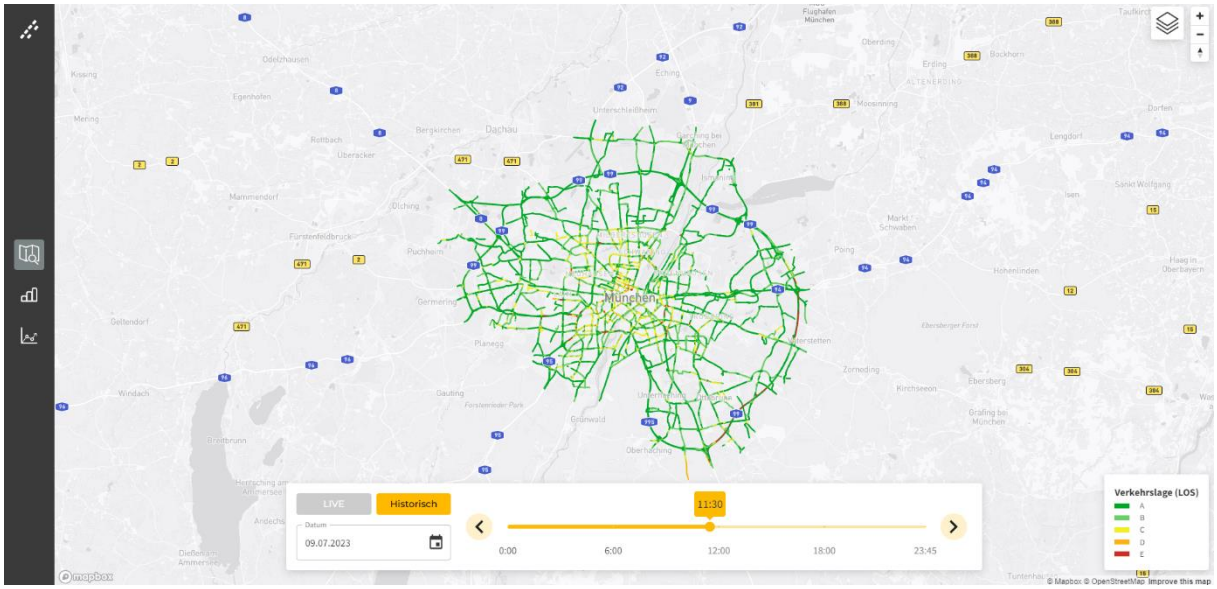

Abbildung 18: Menü der Ansichten „Monitoring Live“ und „Monitoring Historisch“

Beide Abfragen können über dasselbe Menü gemacht werden. Links ist ein Button zu sehen, durch den die Live Analyse angewählt werden kann, rechts kann die Historische Analyse ausgewählt werden. Für die Historische Abfrage kann ebenfalls das Datum in einem Kalender, und die Uhrzeit am Zeitstrahl eingegrenzt werden. Rechts oben kann zwischen den Layern der unterschiedlichen Kennwerte, genauso wie zwischen unterschiedlichen Basemaps gewählt werden.

Zusätzlich zur gesamten Verkehrslage können ebenso konkretere Informationen für jede der im Netz enthaltenen Straßen abgerufen werden. Diese werden in einem Pop-up, das in Abbildung 19 dargestellt ist, durch Klicken auf das Segment angezeigt werden.

Abbildung 19: Pop-Up für detaillierte Informationen auf jedem Netz-Segment

Hier werden wiederum die Kennwerte wie das LOS, die mittlere Geschwindigkeit und Weitere speziell für das ausgewählte Straßensegment angezeigt.

3.2.1.2. FCD-Verkehrsanalyse

Die FCD-Verkehrsanalyse ist eine weitere Möglichkeit der Analyse im DTM. Dabei können verschiedene Kennwerte auf konkreten Routen dargestellt werden. Das Auswahl-Menü für die FCD-Verkehrsanalyse im Web-GIS wird in Abbildung 20 dargestellt.

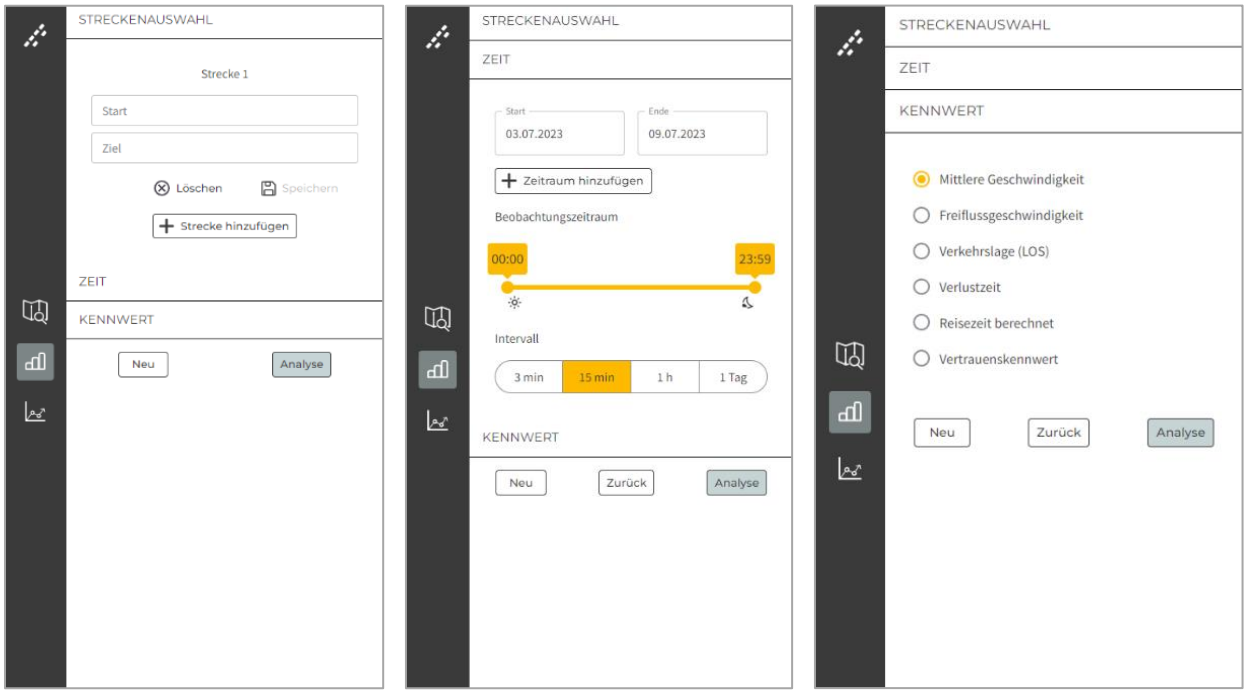

Abbildung 20: Auswahl-Menü für die FCD-Verkehrsanalyse

Es können eine oder mehrere Routen ausgewählt werden, die je einen Start- und Endpunkt in Form einer Straße erfordern. Bei Bedarf kann auch ein Via-Punkt, also ein Punkt zwischen Start und Ende, über den die Route verlaufen soll, gewählt werden. Für diese Masterarbeit werden jedoch nur Analysen zwischen zwei Punkten in Betracht bezogen. Das Ergebnis ist immer die kürzeste Route zwischen den Punkten. Ebenso müssen, wie bereits erwähnt zwei Daten und zwei Zeiten ausgewählt werden. Das Datum gibt an, an welchen Tagen das Ergebnis im Diagramm gezeigt werden soll, während die Zeitspanne eingrenzt, zwischen welchen Uhrzeiten die Werte an den Tagen angezeigt werden sollen. Auch müssen je ein Intervall und ein Kennwert ausgewählt werden. Die Kennwerte sind die folgenden:

- Mittlere Geschwindigkeit
- Freiflussgeschwindigkeit
- LOS
- Verlustzeit
- Reisezeit
- Vertrauenskennwert

Es können die Intervalle von drei oder fünfzehn Minuten, einer Stunde oder einem Tag gewählt werden. Basierend auf diesem Intervall wird der berechnete Kennwert auf dieses aggregiert.

Die Ergebnisse der Analyse werden in zwei Formen dargestellt. Zum einen werden die ausgewählten Routen auf der Karte angezeigt, inklusive Start- und Endpunkt. Ebenso werden die berechneten Kennwerte in einem Diagramm dargestellt. Beide Visualisierungen sind in Abbildung 21 abgebildet. Es ist zu sehen, dass die Analyse über eine andere Maske als die der Abfrage der Live- und Historischen Daten gemacht wird.

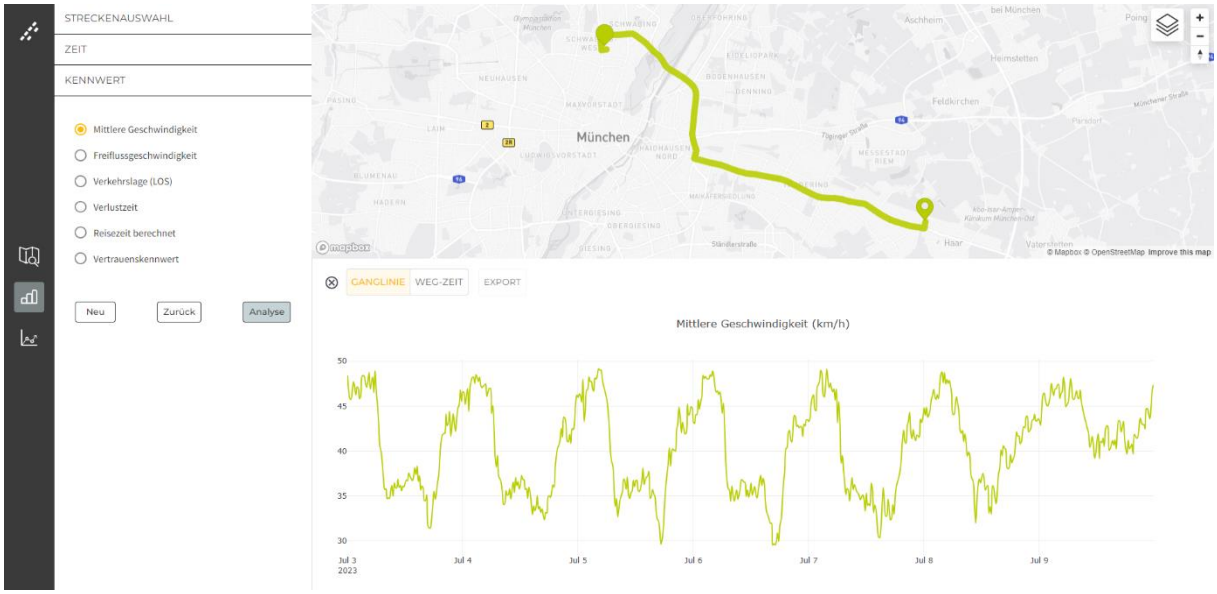

Abbildung 21: Ergebnis einer FCD-Verkehrsanalyse

Links ist das Auswahl-Menü zu sehen, während oben die Karte, und unten das Diagramm angezeigt wird. Dieses wird in Abbildung 22 noch einmal hervorgehoben.

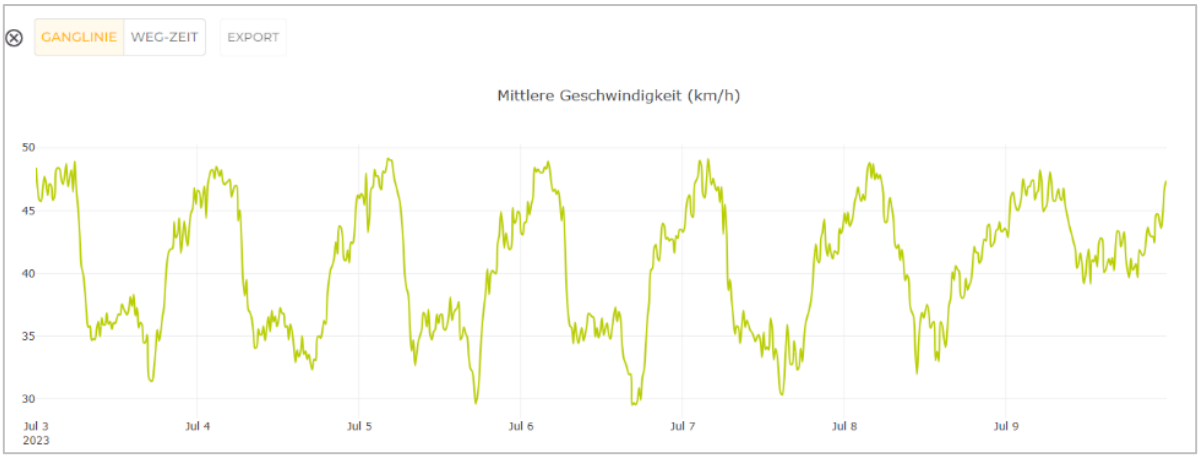

Abbildung 22: Diagramm der FCD-Verkehrsanalyse

Es ist ersichtlich, dass das Diagramm die mittlere Geschwindigkeit zwischen 3. Juli und 10. Juli darstellt. Es können hier die Trends zu den Tageszeiten und Wochentagen analysiert werden. Zu beachten ist, dass die berechneten Werte im gewählten Intervall aufsummiert werden.

Eine weitere Möglichkeit ist die Analyse der Reisezeit im Weg-Zeit-Diagramm. Dabei wird auf jeder der für die Route verwendeten Kanten die Reisezeit zu den ausgewählten Zeitangaben angegeben. Die Reisezeit beschreibt die Zeit die gebraucht wird, um von Anfang bis Ende dieses Segments zu reisen. Dieses Diagramm zeigt Abbildung 23.

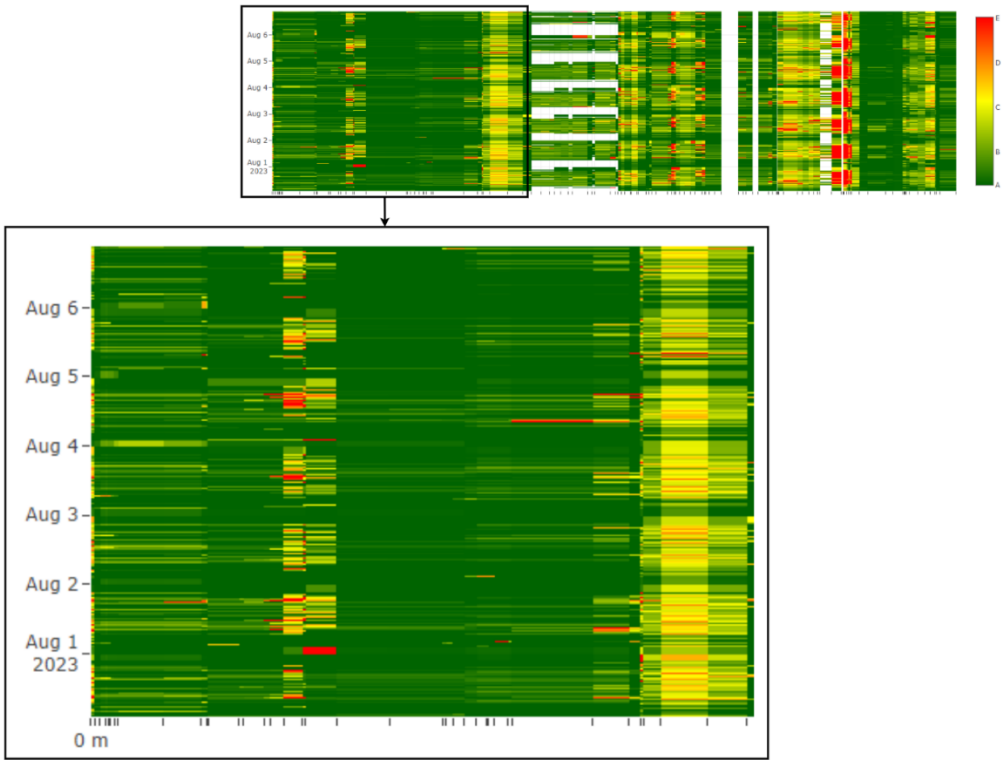

Abbildung 23: Weg-Zeit-Diagramm zur Analyse der Reisezeit auf einer Route

Auf der X-Achse sind die Segmente bzw. Features der Route zwischen Start- und Endpunkt zu sehen. Je länger das Segment, desto breiter auch dessen Ausweitung auf der X-Achse. Die Y-Achse zeigt das in der FCD-Analyse angegebene Datum, in diesem Fall von 1. bis 6. August. Im Diagramm selbst ist der Kennwert der Reisezeit auf jedem Segment zu jeder Tages- und Uhrzeit, die ausgewählt wurde auf der Route zu sehen. Rote und Gelbe Farben zeigen eine erhöhte Reisezeit, und somit ein hohes Verkehrsaufkommen an, während Grün ein schnelles Vorankommen der Verkehrsteilnehmer:innen auf diesem Segment zeigt.

3.2.2. Architektur des DTM

Folgend wird die Architektur des originalen Web-GIS erläutert, wie sie auch in Abbildung 24 dargestellt wird.

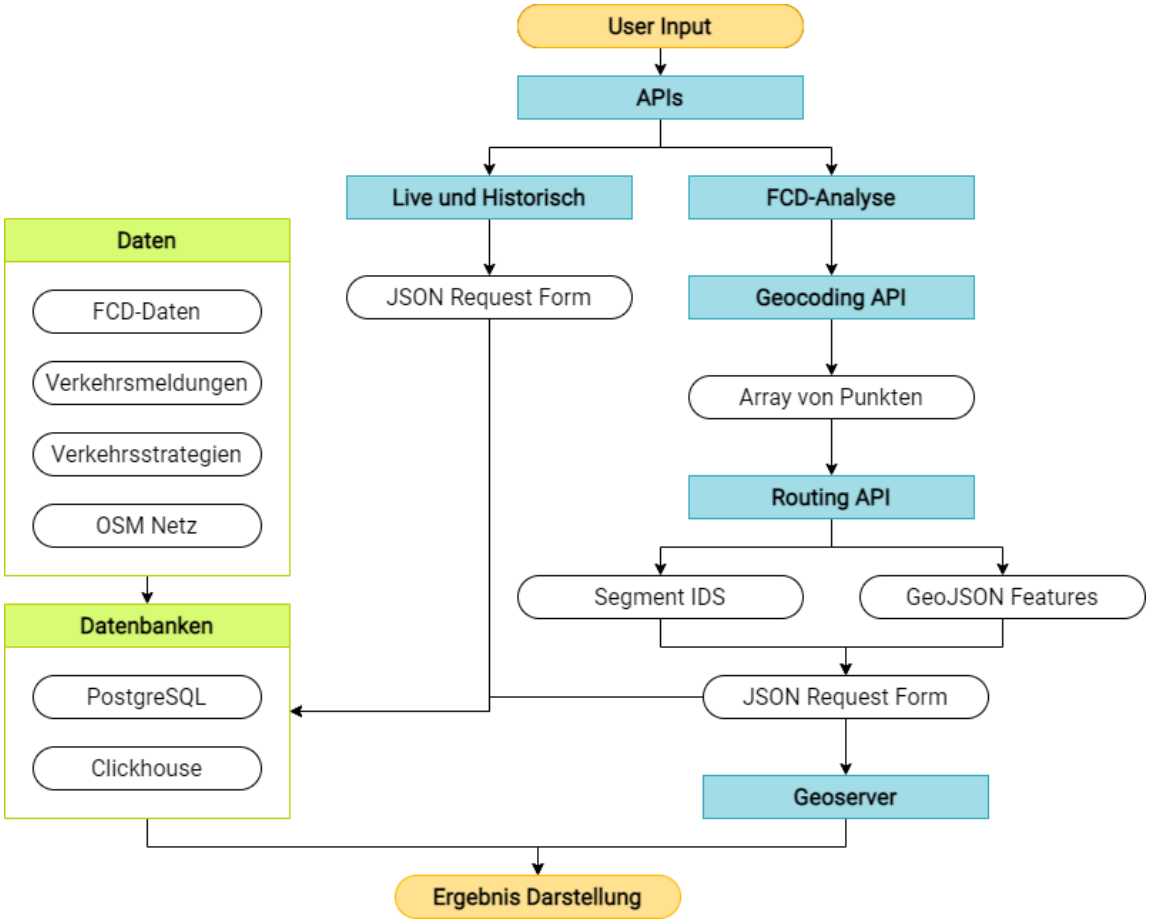

Abbildung 24: Backend des buttongesteuerten Web-GIS

Die Anwendung ist browserbasiert und kann über eine URL abgerufen werden. Das Web-GIS kann als eine Thin-Client Architektur beschrieben werden, wie sie in Kapitel 2.1.2.2 erläutert wurde, da die Prozessierung und Geoprozessierung auf Serverseite ablaufen.

Geben User:innen den Input für die diversen Abfragen über die Buttons ein, wird dieser als erstes basierend auf der Art der Abfrage zu einer API weitergeleitet. Es besteht eine API für die Live und Historische Abfrage, und eine für die FCD-Verkehrsanalyse. Wird die erstere Anfrage gestellt, wird ein JSON-File befüllt. Mithilfe des JSON-Files wird eine Abfrage an die Datenbank gestellt.

Für die Anwendung werden drei unterschiedliche Typen von Daten verwendet, wie sie auch in Abbildung 24 gezeigt sind. Die FCD-Verkehrsdaten, auf denen die Berechnungen der Kennwerte primär basieren, sind von der Firma „HERE“. Anfangs gehen die Daten als Real-Time Daten in eine Clickhouse-Datenbank ein, die besonders für große Datenmengen geeignet ist. Anschließend werden diese historisiert und in eine PostgreSQL-Datenbank überführt und für die Historische

Analyse und FCD-Verkehrsanalyse angewendet. Zusätzlich werden Verkehrsmeldungen und Verkehrsstrategien miteinbezogen, die von „MDM“ stammen. Das Straßennetz basiert auf „Open Street Map“ (OSM). Je nach Anfrage werden die Daten von der PostgreSQL oder Clickhouse Datenbank abgefragt. Alle Daten gehen bis Jänner 2023 zurück.

Wird eine FCD-Verkehrsanalyse von den User:innen angefordert, wird der Input nicht direkt in das JSON-File eingefügt. Zuvor werden die ausgewählten Straßen an die Geocoding API weitergegeben, um diese wie in Kapitel 2.4.1 beschrieben, in verarbeitbare Koordinaten umzuwandeln. Dieses Array von Koordinaten wird anschließend an die Routing API weitergeleitet. Durch diese wird die kürzeste Route zwischen den Punkten berechnet, und zum einen die IDs der Straßensegmente zurückgegeben, zum anderen die GeoJSON-Features der Straßen. Die Segment IDs werden benötigt, um die Kennwerte im Diagramm berechnen zu können, und durch die GeoJSON-Features wird die Geometrie in der Karte dargestellt. Wurden alle diese Daten berechnet, werden auch diese in ein JSON-File eingefügt, mit dem die Datenbank abgefragt wird.

Nach der Abfrage werden die Ergebnisse der Analysen in der Anwendung dargestellt. Mit einem Geoserver wird die Anzeige aller Layer im Web-GIS ermöglicht.

Der letzte Bestandteil ist der Geoserver, der die Anzeige aller Layer in der Anwendung ermöglicht.

3.3. Textgesteuerter DTM

Wie bereits erwähnt basiert die Anwendung, die für die Implementierung der Modelle und die Studie erstellt werden soll auf dem eben beschriebenen Web-GIS der Firma Trafficon. Dieses wird auf ein textgesteuertes Interface umgebaut. Das Konzept und die Gestaltung wurden selbst erdacht, ebenso wie die Datenverarbeitung der Input-Daten und die DL-Modelle selbst implementiert wurden. Die Umsetzung des Backends und Frontends der Anwendung wurde vom Entwicklerteam der Firma übernommen, da diese nicht zu den Zielen der Arbeit zählen. Die Anwendung soll kein fertiges Produktivsystem für den Alltag von Experten darstellen. Das Web-GIS, das hier mit NLP realisiert wird, ist ein erster Prototyp, der für die Verwendung in der Masterarbeit umgesetzt wird.

Die Funktionen und Analysemöglichkeiten bleiben auch in der textgesteuerten Anwendung dieselben. Auch ändert sich am Backend selbst kaum etwas, außer der Befüllung der JSON-Files, die nun basierend auf einem Textfeld und der Verarbeitung durch die DL-Modelle gemacht wird. Genaueres dazu in Kapitel 4.4.

Abbildung 25 zeigt das textgesteuerte Interface der Anwendung. Dieses wurde möglichst einfach gehalten.

Abbildung 25: Interface des testgesteuerten Web-GIS

Alle Abfragen werden hier über das Textfeld oben links getätigt. Dies schließt die Live- und Historische Abfrage, genauso wie die FCD-Verkehrsanalyse mit ein. Es wird somit vermieden, dass User:innen je nach Anfrage das Menü wechseln müssen. Ebenso wurde das komplexe und umfangreiche Auswahl-Menü der FCD-Verkehrsanalyse eliminiert. Um die Analyse zu starten kann dies mit der Eingabetaste oder dem Start-Button gemacht werden. Das textgesteuerte Interface soll so eine starke Vereinfachung für User:innen darstellen, und einen Einblick in die Performance eines rein textgesteuerten Web-GIS ermöglichen.

4. Modell für Natural Language Processing

Um das Natural Language Processing in die Anwendung zu integrieren, soll ein Deep Learning Modell trainiert werden, mit dem die textuellen Input-Daten verarbeitet werden können. Dieses Kapitel beschreibt den Prozess der Auswahl des Modells, des Trainingsprozesses und der Implementierung dessen in die Anwendung.

4.1. Konzept der Modelle

Die Auswahl des Modells, das für die Datenverarbeitung trainiert werden soll, basiert auf den in Kapitel 2 beschriebenen theoretischen Grundlagen.

Durch die vorhin genannten Grundlagen wurde offensichtlich, dass ein Modell benötigt wird, dass für die Named Entity Recognition geeignet ist. Diese ermöglicht es, wie bereits in Kapitel 2.3.2.2 beschrieben, strukturierte und relevante Informationen aus unstrukturiertem Input-Text zu extrahieren. Dies ist auch bei der Implementierung in der Anwendung von hoher Wichtigkeit, da die Informationen aus der Named Entity Recognition benötigt werden, um eine Datenbankabfrage zu ermöglichen.

Weiters muss eine Möglichkeit gefunden werden, um über die Anfrageart des User-Inputs zu entscheiden, die die drei Analysemöglichkeiten der Anwendung widerspiegeln. Diese Unterscheidung der Anfrage ist nötig, um die unterschiedlichen APIs der Anfragearten anzusprechen. Für die Erkennung der Anfrageart bestehen zwei Ansätze: ein dynamischer oder ein starrer. Bei ersterem würde die Art der Anfrage basierend auf der Menge und Art der durch die Named Entity Recognition extrahierten Informationen ausgewählt werden. Dies würde bedeuten, dass bei zwei Datumsangaben automatisch vermutet wird, dass User:innen eine FCD-Verkehrsanalyse machen möchten. Es ist nur schwer möglich, alle bestehenden Anfragearten inklusive möglicher Fehler starr auszuformulieren und abzudecken. Dieser Ansatz ist unflexibel und geht kaum auf Fehler und Abweichungen ein. Eine mögliche Alternative bildet hier ein dynamischer Ansatz: die Analyse der Anfrageart mittels Natural Language Processing. Durch die in Kapitel 2.3.2.1 beschriebene Text Klassifizierung ist es möglich, einem Text eine Klasse basierend auf dem Inhalt dessen zuzuweisen. In diesem Fall bedeutet das, das dem Input-Text die Klassen Monitoring Live, Monitoring Historisch oder Analyse (FCD-Verkehrsanalyse) zugewiesen werden würde.

Daraus folgt das Konzept, dass zwei Modelle trainiert und implementiert werden. Eines dieser Modelle ist für die Named Entity Recognition geeignet, und eines für die Text Klassifizierung. Die Modelle werden folgend als „Token Model“ und „Sequence Model“ bezeichnet. Ersteres stammt

von dem Synonym für Named Entity Recognition, der Token Classification, zweiteres wird nach der Sequence Classification, dem Synonym für Text Klassifizierung benannt.

4.2. Vergleich und Auswahl der möglichen Modelle

Wie in Kapitel 2.5.4.2 bereits erwähnt bieten Transformer-Modelle die Möglichkeit, auf universelle Sprache trainierte Modelle für eigene Anwendungen mit verhältnismäßig wenig Ressourcen zu trainieren. Weiters sind Transformer-Modelle für ihre herausragende Performance bekannt. Deshalb werden verschiedene vortrainierte Modelle gewählt, um eines dieser selbst weiter trainieren zu können.

Wie bereits in Kapitel 2.5.4.2 erwähnt, sind Modelle der BERT-Architektur für viele Tasks, wie auch die Named Entity Recognition und die Text Klassifizierung geeignet. Auch Modelle mit der auf BERT basierenden RoBERTA-Architektur werden zur Auswahl gestellt, da diese ebenso vielseitig sind, und in vielen Hinsichten das ursprüngliche Modell verbessern sollen. Eine weitere Voraussetzung der Modelle ist deren Fähigkeit, Deutsch zu verarbeiten. Das bedeutet, dass das Modell entweder nur für die deutsche Sprache vortrainiert ist oder multilingual trainiert wurde.

Auch, wenn das Token Model und Sequence Model auf unterschiedliche Aufgaben trainiert wird, wird dasselbe vortrainierte Modell für sie ausgewählt, da sich die beiden Aufgabenstellungen stark ähneln.

Tabelle 3 zeigt die nähere Auswahl an Modellen, aus denen schlussendlich ein Modell auf die eigenen Anwendungen trainiert werden soll. Die Auswahl basiert auf eigenen Recherchen und der Evaluierung von SCHEIBLE et al. 2020.

Tabelle 3: Eigenschaften der ausgewählten Modelle (Basierend auf: SCHEIBLE et al. 2020: 6)

Model	*Architektur*	*Sprachen*	*Datengröße*	*Datenquellen*
GottBERT	RoBERTa	1	145 GB	OSCAR, Wikipedia, EU Bookshop corpus
dbmz BERT	BERT	1	16 GB	Open SUbtitles, Common-, Para-, NewsCrawl
mBERT$_{cased}$	BERT	104	Unbekannt	Wikipedia
German BERT	BERT	1	12 GB	News articles, Open Legal Data, Wikipedia
XLM RoBERTa	RoBERTa	100	2.5 TB (66.6 GB Deutsch)	CommonCrawl, Wikipedia

Zwei der Modelle basieren auf einem RoBERTa-Modell, die anderen drei Modelle auf dem BERT-Modell. Ebenso wurden zwei Modelle gewählt, die multilingual trainiert wurden, eines davon auf 100 Sprachen, das zweite auf insgesamt 104. Die Größe des Corpus, mit dem die Modelle vortrainiert wurden, ist sehr unterschiedlich verteilt. Die Modelle „dmbz BERT“ und „german BERT“ wurden je mit 16 GB und 12 GB trainiert. Der deutsche Teil des „XLM RoBERTa“ wurde

mit rund 66 GB trainiert, und das Modell „GottBERT“ mit 145 GB. Die Trainingsdaten selbst unterscheiden sich je nach Modell.

Um jedoch ein passendes Modell wählen zu können, ist besonders die Performance dessen von Bedeutung. Basierend auf SCHEIBLE ET AL. 2020 wurden verschiedene Performance-Benchmarks für Natural Language Processing Tasks gesammelt. Die Aufzeichnungen sind in Tabelle 4 dargestellt. Wie die Auflistung zeigt, werden die Modelle basierend auf Benchmarks in der Named Entity Recognition und der Text Klassifizierung verglichen. Abschließend wird die Performance pro Task und ebenso im Gesamten aufsummiert.

Tabelle 4: Bewertungsmatrix ausgewählter Modelle von diversen Benchmarks (Basierend auf: SCHEIBLE et al. 2020: 6)

	XLM RoBERTa	***German BERT***	***GottBERT***	***mBERT***	***dbmz BERT***
Named Entity Recognition					
CoNLL 2003	81,4	81,2	83,6	81,2	82,3
GermEval 2014	85,4	85,0	86,8	85,4	85,8
Sequence Classification					
GermEval 2018 coarse	75,2	77,2	76,4	72,9	77,3
GermEval 2018 fine	45,6	49,3	51,3	44,8	51,0
10kGNAD	89,3	90,7	89,2	88,7	90,9
Gesamt-Performance					
NER Performance	166,8	166,2	170,4	166,6	168,1
Sequence Performance	210,1	217,2	216,8	206,4	219,2
Overall Performance	376,9	383,4	387,3	373,0	387,3

Die in der Tabelle angegebenen Werte sind je Modell der beste F1-Score im angegebenen Task, aus zehn durchgeführten Runs. Der F1-Score wurde in Kapitel 2.5.3.4. bereits erläutert und geht bis zum maximalen und besten Wert Eins.

Bereits bei der ersten Betrachtung wird offensichtlich, dass das Modell „GottBERT“ und „dbmz BERT“ in der Performance deutlich an der Spitze stehen, und nur das Modell „German BERT“ eine annähernd gleichgute Performance aufweist. In der Performance in der Named Entity Recognition liegt GottBERT in allen Benchmarks an erster Stelle, und zeigt die beste Performance auf. Hier steht dbmz BERT immer an zweiter Stelle. In der Sequence Classification hingegen liegt dbmz BERT in zwei von drei Benchmarks, und im aufsummierten Sequence Classifiction Benchmark vorne. German BERT liegt bei zwei der Benchmarks, sowie beim aufsummierten Wert der Sequence Classification direkt hinter dbmz BERT. GottBERT belegt hier bei einem der Benchmarks den ersten Platz. In der gesamten Summe der Performance stehen GottBERT und dbmz BERT beide mit der höchsten Performance am ersten Platz. Im direkten Vergleich ist besonders zu beachten, dass zwischen den Werten, insbesondere bei den beiden am besten

performenden Modellen, teils nur minimale Unterschiede sichtbar sind. Da die NER die komplexere Aufgabenstellung der in dieser Arbeit verwendeten ist, wurde final das Modell GottBERT gewählt, da es hier besser performt. Genauso weist es auch eine sehr gute Performance in der Text Klassifizierung auf.

Das Modell ist ein Transformer-Modell mit der RoBERTa-Architektur. Es wurde von SCHEIBLE ET AL. 2020 mit dem deutschsprachigen Teil des OSCAR-Datensatzes (Open Super-large Crawled ALMAnaCH coRpus) trainiert. Dieser Datensatz hat eine Größe von 145 GB, und besteht aus etwa 21.5 Milliarden Wörtern, und 459 Dokumenten. Das Modell wurde mit den in Tabelle 4 gezeigten Benchmarks in Named Entity Recognition und Sequence Classification getestet, und hat dabei eine zufriedenstellende Performance erreicht. Das GottBERT ist das erste rein deutschsprachige Modell, dass auf RoBERTa basiert. (vgl. SCHEIBLE et al. 2020: 2-6)

Nun konkreter zu der Architektur des Modells. Das Modell wurde je einmal für die Text Klassifizierung und die Named Entity Recognition geladen. Abbildung 26 zeigt die oberflächliche Zusammensetzung der Modelle.

```
Model: "tf_roberta_for_sequence_classification"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 roberta (TFRobertaMainLayer  multiple                 125394432
 )

 classifier (TFRobertaClassi  multiple                 592130
 ficationHead)

=================================================================
Total params: 125,986,562
Trainable params: 125,986,562
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "tf_roberta_for_token_classification"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 roberta (TFRobertaMainLayer  multiple                 125394432
 )

 dropout_75 (Dropout)        multiple                  0

 classifier (Dense)          multiple                  1538

=================================================================
Total params: 125,395,970
Trainable params: 125,395,970
Non-trainable params: 0
_________________________________________________________________
```

Abbildung 26: Layer des GottBERT-Models für die Text Klassifizierung (Oben) und die Named Entity Recognition (Unten)

Oben ist das Modell für die Text Klassifizierung dargestellt, unten das für die Named Entity Recognition. Die Architektur stellt auch die Theorie des Transfer Learnings, das in Kapitel 2.5.4.2 beschrieben wurde, dar. Der erste Layer ist der „TFRobertaMainLayer“. Dies ist der Body, der vortrainiert wurde, und universelle Sprache verstehen kann. Dieser Layer ist bei beiden Modellen derselbe. Der Layer verfügt über eine Vielzahl an Gewichten (Parametern), die trainierbar sind. Parameter beschreiben in diesem Fall die Gewichte der Neuronen. Ebenso wurde für beide Modelle ein „Classifier“ geladen. Dies ist der Head, der selbst antrainiert werden kann. Dabei wurde jedoch für jedes der Modelle der für den Task spezifischer Head geladen. Beim Token Model wurde ebenso ein Dropout Layer geladen, der jedoch über keine trainierbaren Parameter verfügt und, und für diese Anwendung nicht von Bedeutung ist. Auch jeder der Heads verfügt über Gewichte, wobei die Menge der Gewichte im Token Model niedriger ist. Das Modell verfügt über eine maximale Länger der Input-Sequenzen von 514 Tokens. Diese ist später für die Truncation und das Padding relevant, wie es in Kapitel 2.5.4.2 beschrieben wurde.

4.3. Training der Modelle

Im folgenden Kapitel wird der Trainingsprozess des Token Models und Sequence Models detailliert erläutert. Anfangs werden die Libraries, die verwendet wurden beschrieben, und anschließend wird je Model dessen Trainingsdaten und Trainingsprozess, sowie das finale Model beschrieben.

4.3.1. Libraries für Training

Für die Umsetzung des Trainings und Tunings wurde die Programmiersprache Python gewählt. Für Python bestehen unterschiedlichste Libraries, mit denen Natural Language Processing und das Training eines Modells umgesetzt werden kann. Es wurden Libraries ausgewählt, die die Umsetzung von NLP mittels Deep Learning ermöglichen.

Eine der zentralen Libraries dafür ist „Transformers“. Die Library wurde von „Hugging Face“ entwickelt. Sie bildet eine standardisierte Schnittstelle zwischen den Transformer Modellen, Datensätzen zum Training und Tools, die die Umsetzung erleichtern. Unterstützt werden von Transformers mehrere Deep Learning Libraries:

- PyTorch
- TensorFlow
- JAX

Dies bietet für User:innen umfangreiche Möglichkeiten, performante Transformer Modelle für eigene Anwendungen trainieren zu können. Die Transformer-Library ist grundsätzlich für die

Computer Vision, Audio und auch Natural Language Processing geeignet. (vgl. TUNSTALL ET AL. 2023: 32f)

Ein weiterer relevanter Bestandteil der Arbeit ist auch die Plattform Hugging Face, in der die Library Transformers integriert ist. Auf der Plattform werden eine Vielzahl von PTM als einfach herunterladbare Modelle bereitgestellt. Auch können die eigens trainierten und feingetunten Modelle hochgeladen, einfach abgerufen und bei Bedarf auch veröffentlicht werden. Weiters werden Tokenizer, passend zu den Modellarchitekturen wie BERT oder RoBERTA hochgeladen. Ebenso bestehen weitere sogenannte „Pipelines". Diese ermöglichen es, mit den trainierten Modellen einfach Vorhersagen, für verschiedenste Tasks wie Named Entity Recognition und Text Klassifizierung, durchzuführen. (vgl. TUNSTALL ET AL. 2023: 38-42) Das ausgewählte Modell GottBERT wird ebenfalls von Hugging Face bezogen.

Weiters müssen die Modelle mit einer Library, die für das Deep Learning geeignet ist trainiert werden. Wie bereits erwähnt stehen hier PyTorch, TensorFlow und JAX zur Verfügung. Alle der Libraries sind interoperabel, und können in wenigen Schritten gegeneinander ausgetauscht werden. Jedes der Modelle auf Hugging Face ist entweder für PyTorch oder TensorFlow erstellt, kann aber auch für die anderen Library einfach angewendet werden. (vgl. TUNSTALL ET AL. 2023: 9)

Für die Umsetzung des Trainigsprozesses wurde die Library TensorFlow schlussendlich gewählt. Diese ist eine der beliebtesten Plattformen für Machine Learning und Deep Learning Frameworks und wurde von Google entwickelt. Ebenso wurde die mit TensorFlow gekoppelte Library „Keras" verwendet. Diese ist eine high-level API, auf die in der Verbindung mit TensorFlow zugegriffen werden kann, die eine intuitive Möglichkeit zum Aufbau von eigenen Modellen bieten. (vgl. VASILEV ET AL. 2019: 82ff)

4.3.2. Trainingsprozess

Folgend wird der Prozess des Trainings und Hyperparameter Tunings der beiden Modelle beschrieben. Je Modell wird der gesamte Trainingsprozess und die Evaluierung der finalen Modelle erläutert. Ebenso werden die Trainingsdaten und deren Erstellung vorgestellt. Alle Entscheidungen im Trainingsprozess basieren auf der in Kapitel 2.5 dargestellten theoretischen Grundlagen getroffen.

Beide Modelle werden auf einen Klassifikationstask trainiert, bei dem, wie in Kapitel 2.5.2 beschrieben, den Daten je eine Klasse zugewiesen wird. Ebenso können beide Trainingsprozesse als überwachtes Lernen definiert werden, da die Zielwerte beim Training bekannt sind. Auch dies wurde in Kapitel 2.5.2 erläutert.

Ziel des Prozesses ist das Finden der optimalen Hyperparametereinstellungen. Für den Trainingsprozess werden das manuelle Einstellen der Hyperparameter und auch die automatisierten Methoden der Grid Search und Random Search ausprobiert. Optimiert wurden die folgenden Hyperparameter:

- Lernrate
- Batch Size
- Anzahl der Epochen

Zwei weitere Hyperparameter werden nicht während des Anlernprozesses verändert, sondern zu Beginn einmalig festgelegt. Diese Parameter sind der Optimizer und die Verlustfunktion. Bei diesen gibt es bestehende Limitationen und Empfehlungen, die eingehalten werden sollten.

Anschließend wird das Modell mit der besten Performance exportiert. Die Performance wird mit den folgenden Metriken evaluiert:

- Accuracy
- Loss
- Precision
- Recall
- F1-Score

Auch diese wurden in Kapitel 2.5 erläutert.

4.3.2.1. Trainingsdaten der Modelle

Folgend werden die Trainingsdaten und deren Erstellung und Auswahl für das Sequence Model und das Token Model erläutert.

Sequence Model

Für das Anlernen des Sequence Models wurde ein eigens erstellter Datensatz verwendet. Für die Text Klassifizierung muss ein solcher Datensatz aus einem Satz und einer ihm zugewiesenen Klasse bestehen. Die Sätze stellen dabei später den Input (x-Werte) dar, und die Klasse das Label (y-Werte), dass dem Satz zugewiesen werden muss.

In diesem Fall sind die Klassen keine Emotionen oder Polarität, wie sonst häufig bei der Sentiment Analyse, sondern die Anfragearten, die von User:innen an das Web-GIS gestellt werden können. Diese lauten: Monitoring Live, Monitoring Historisch und FCD-Verkehrsanalyse.

Die Input-Sätze, mit denen das Modell trainiert werden soll, sind Beispielanfragen, wie sie später auch an die Anwendung gestellt werden können. Bei der Erstellung der Sätze lag der Fokus darauf, grammatikalisch korrekte und sinnvolle Sätze, und eine möglichst große Vielfalt an Sätzen zu erstellen. Dazu wurde versucht, Sätze basierend auf deren Satzteile und Grammatik in kleinere, abgegrenzte Wortgruppen, aber auch einzelne Worte zu teilen.

Um beispielsweise die Verwendung von unterschiedlichen Straßen zu ermöglichen, wurden hier die Straßennamen der OSM-Daten von München geladen. Dieser umfasst etwa 9000 Straßen. Diese Straßen werden später bei der Erstellung der Sätze randomisiert verwendet.

Im nächsten Schritt wurde eine Liste von Datumsangaben von Jänner 1990 bis Dezember 2010 erstellt. Es wurden unterschiedliche Datumsangaben erstellt wie: „1.1.1990“, oder aber auch „Montag, 1.1.1990“, und noch weitere unterschiedliche Formaten. Insgesamt wurden 30652 Datumsangaben erstellt. Dadurch soll ermöglicht werden, dass eine von bestimmen Formaten abweichende Datumsform keinen negativen Einfluss auf die Klassifizierung von Sätzen hat. Ebenso werden unterschiedlichste Zeitangaben erstellt, für alle möglichen Zeiten eines Tages. Es werden Formate wie „00:06 Uhr“ oder „7 Uhr“ erstellt. Das Resultat sind 4332 Zeitangaben.

Ebenso wurden Wortgruppen mit Intervallen erstellt, wie: 2 Stunden, oder 3 Tage. Es wurden 295 Intervalle erstellt.

Weiters wurden Synonyme für das Wort „Verkehrssituation“ und „Route“ gesammelt, wobei diese für Fragesätze und Antwortsätze dekliniert wurden. Ebenso wurden unterschiedliche Verben zur Anweisung, Präpositionen und Fragewörter gesammelt. Auch wurden die Kennwerte, die abgefragt werden können, und mögliche Synonyme verwendet.

Nachdem alle Wörter und Wortgruppen gesammelt wurden, wurden Satzteile definiert. Satzteile sind beispielsweise Formulierungen wie „die Route von der Prinzregentenstraße zur Verdistraße“. Innerhalb dieser Satzteile werden die vorhin definierten Entitäten zufällig zusammengefügt, sodass sich Formulierung und Inhalt ändern, jedoch die Teile immer noch grammatikalisch und inhaltlich korrekt sind. Im nächsten Schritt werden die erstellten Satzteile gespeichert, und zu Sätzen zusammengefügt. Dabei werden für jede Anfrageart Fragesätze und Aussagessätze erstellt. Jedem Satz wird hier gleichzeitig die passende Klasse hinzugefügt. Insgesamt wurden 45 000 Sätze erstellt, und als CSV-File exportiert.

Tabelle 5 zeigt einen der erstellten Sätze für jede der Klassen.

Tabelle 5: Beispielsätze der erstellen Trainingsdaten für das Sequence Modell

Klasse	***Satz***
Monitoring Live	Kannst du mir zeigen, wie das momentane Verkehrsaufkommen ist?
Monitoring Historisch	Wie war die Verkehrssituation zwischen 12.10.2007 und Mittwoch, 11.01.2006 von 15:34 Uhr bis 23:27?
Analyse	Was war das LOS aggregiert auf 1 Tag auf der Route von Sickingerstraße nach Kufsteiner Platz zwischen 13.12.1991 und Donnerstag, 13.12.2001 zwischen 17:31 Uhr und 01:11?

Der eigens erstellte CSV-Datensatz wurde eingelesen. Der Datensatz wurde durchgemischt, um eine zufällige Verteilung der Klassen zu erreichen. Anschließend wurden die Daten in einen Trainingsdatensatz und einen Testdatensatz zur Überprüfung geteilt. 80 % der Daten befinden sich im Trainingsdatensatz, 20 % der Daten wurden den Testdaten zugewiesen. Abbildung 27 zeigt die Verteilung der Daten auf die unterschiedlichen Datensätze, sowie die Verteilung der Klassen.

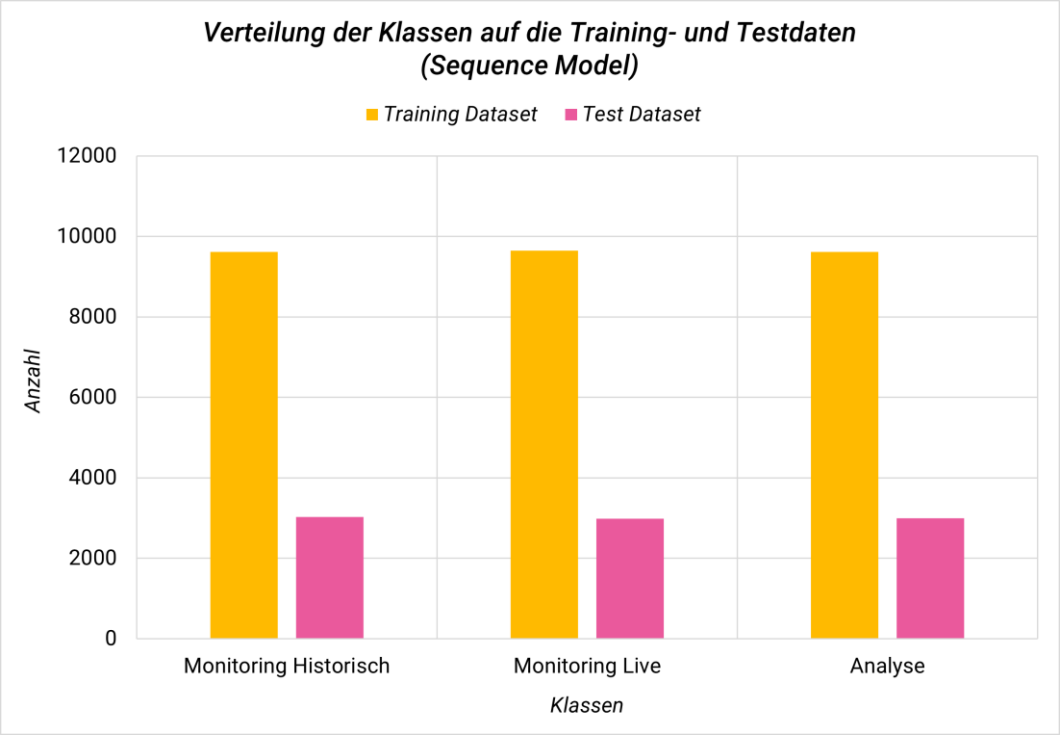

Abbildung 27: Verteilung der Trainingsdaten des Sequence Models auf die Klassen und das Train- und Test Dataset

Jede der Klassen Live, Historisch und Analyse sind in allen drei Datensätzen beinahe gleichverteil, was eine positive Voraussetzung für den Anlernprozess darstellt.

Anschließend werden die Sätze tokenisiert. In diesem Fall kann dies ohne weitere Anpassungen mit dem geladenen Tokenizer gemacht werden. Dieser wurde für RoBERTa-Modelle geladen. Das

Padding und die Truncation wurden aktiviere. Als maximale Länge für die Sätze wurden 50 Tokens ausgewählt, da keine längeren Sätze erstellt wurden.

Token Model

Für das Token Model wurden zwei unterschiedliche Datensätze für das Training vorbereitet. Einer der Datensätze wurde selbst erstellt, der andere ist ein bereits bestehender Datensatz.

Der eigene Datensatz wurde basierend auf den Sätzen, die für das Trainieren des Sequence Model erstellt wurden, umgesetzt. Zusätzlich zu den Sätzen wurden die Wörter, die darin enthalten sind, mit Entity Labels getagged. Es wurde das IOB-Tagging Scheme, das bereits in Kapitel 2.3.2.2 erläutert wurde, angewendet. Insgesamt wurden fünf verschiedene Entitäten für das Tagging ausgewählt:

- „Date“
- „Time“
- „Location“
- „Kennwert“
- „O“

Die Tags wurden zu allen Wörtern automatisiert hinzugefügt. Um die Aufteilung auf das „I-“, „O-“ und „B-Tag“ zu ermöglichen wurden die Sätze erst mit dem Tokenizer, der später auch beim Training verwendet wird, im bereits tokenisierten Zustand gelabelt, da die Tags schlussendlich auch in diesem Zustand zugewiesen werden. Bei der Named Entity Recognition ist dies ein größerer Aufwand als bei der Text Klassifikation, da nicht nur ein Label pro Satz, sondern für jedes einzelne Wort zugewiesen werden muss.

Die Daten wurden ebenfalls als CSV-File gespeichert und eingelesen. Auch hier wurden die Daten mit dem RoBERTa-Tokenizer von Transformers tokenisiert. Anschließend müssen die Wörter und Tags jedoch in einem etwas komplexeren Verfahren parallelisiert werden. Der Datensatz in mit je 80% und 20% in Training- und Testdaten aufgeteilt. Abbildung 28 zeigt die Verteilung der Aufteilung der Daten und der einzelnen Entitäten.

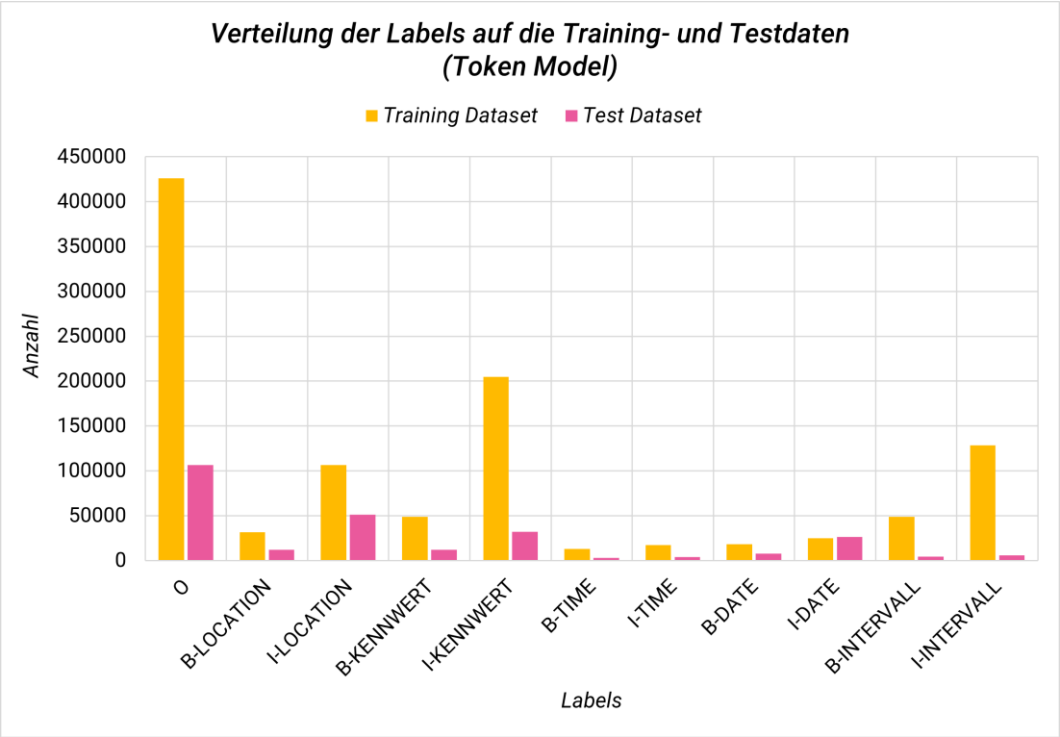

Abbildung 28: Verteilung der Trainingsdaten des Token Models auf die Labels und das Train- und Test Dataset

Hier ist offensichtlich, dass die Daten sehr unterschiedlich verteilt sind. Die Kategorie „O“ hat die meisten Daten, da dies die Wörter darstellt, die keine der Entitäten sind. Andere Entitäten haben hingegen eine sehr niedrige Anzahl. Dies ist beispielsweise bei dem Label für die Kennwerte problematisch. Hier können nur die Kennwerte selbst, so wie sie in der Anwendung genannt werden als Input-Daten verwendet werden, sowie einige wenige Synonyme. Hier stellt sich die Frage, ob dies überhaupt in einem Modell als generalisierbare Eigenschaft erlernbar ist. Ziel ist, diesen selbst erstellten Datensatz zu verwenden, jedoch sind hier von Anfang an große Schwierigkeiten gegeben. Deswegen wird ein Ersatz-Datensatz ausgewählt, der verwendet wird, wenn das Training mit dem eigenen Datensatz keinen Erfolg zeigt.

Der zweite Datensatz wurde vom deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) erstellt und wird „smartdata Corpus“ genannt. Der Datensatz ist deutschsprachig und ist für das Training von Modellen für die Named Entity Recognition für Themen mit Verkehrsbezug geeignet. Der Corpus wurde aus beinahe 4 Millionen Tweets, 400 000 RSS Feeds und 900 000 News-Dokumenten erstellt. In diesen wurden die Entitäten, die in Abbildung 29 gezeigt werden, getagged.

Entity / Concept	Description	Examples
Location (LOC)	General locations	Bayern, Zugspitze, Norden
Location-City (LOC-CIT)	Municipalities, e.g. cities, towns, villages	Berlin, Berlin-Buch, Hof
Location-Street (LOC-STR)	Named streets, highways, roads	Hauptstrasse, A1
Location-Route (LOC-ROU)	Named (public) transit routes	U1, ICE 557, Nürnberg – Hof
Location-Stop (LOC-STO)	Public transit stops, e.g. train stations, bus stops	S+U Pankow, Berlin-Buch
Organization (ORG)	General organizations	Greenpeace, Borussia Dortmund
Organization-Company (ORG-COM)	The subset of organizations that are businesses	Siemens AG, BMW
Person (PER)	Persons	Angela Merkel
OrgPosition (POS)	A person's position within an organization	CEO, Vizepräsident
Date (DAT)	Point in time, date	1. September 2017, gestern
Time (TIM)	Time of day	8:30, 5 Uhr früh
Duration (DUR)	Time periods	mehr als eine halbe Stunde
Distance (DIS)	Distances with unit	5 Kilometer
Number (NUM)	Other numeric entities, e.g. money, percentages	3%, 4 Millionen Euro
Disaster-Type (DIS-TYP)	Man-made and natural disaster types	Erdbeben, Überschwemmung
Trigger (TRI)	Trigger terms or phrases for events	Stau, Streik, Entlassungen

Abbildung 29: Entitäten des smartdata Corpus (Quelle: SCHIERSCH ET AL. 2018: 2)

Für die eigene Anwendung werden nur die Entitäten Location-Street (LOC-STR), Date (DAT) und Time (TIM) verwendet. Dieser Datensatz ist auf dem Hugging Face Hub als Datensatz hochgeladen, weswegen er auch als solcher eingebunden wurde. Die Daten sind bereits in Training- und Testdaten aufgeteilt. Auch diese Daten wurden tokenisiert.

Während der Vorteil des eigenen Datensatzes ist, dass er auf die eigene Anwendung spezialisiert ist, ist dies ebenso der Nachteil. Die Daten weisen in einigen Labels eine niedrige Diversität auf. Es werden folgend im Trainingsprozess beide der Datensätze getestet, und schlussendlich einer dieser für das finale Training ausgewählt.

4.3.2.2. Sequence Model

Das Modell wird für die Text Klassifizierung trainiert. Es wurden diverse Kombinationen von Hyperparameter ausprobiert, und das Modell trainiert. Hier aufgeführt und weiter erläutert werden jedoch nur die Ergebnisse der Iterationen, die einen relevanten Beitrag zum finalen Modell oder ein interessantes Ergebnis zeigen. Die hier beschriebenen Durchläufe werden als „Run“ mit fortlaufender Nummerierung benannt.

Anfangs wird das Training mit einer Hyperparameter Optimization durchgeführt. Es werden die Grid Search und die Random Search ausprobiert, wie sie in Kapitel 2.5.3.3 erläutert wurden. Anfangs wurde die Grid Search durchgeführt. Diese wurde jedoch nach kurzer Zeit abgebrochen, da die Verarbeitung von 500 Samples des Trainingsdatensatzes bereits über 13 Minuten in Anspruch nahm. Da auch in der Literatur erwähnt wird, dass die Grid Search zwar ressourcen- und zeitaufwendig ist, jedoch oft ähnliche oder auch schlechtere Ergebnisse erreichbar sind, wurde die Random Search verwendet.

Es wurden die folgenden Werte basierend auf Empfehlungen in Kapitel 2.5.3.3 getestet:

- Lernrate: 1e-2, 1e-3, 1e-4
- Batch Size: 8, 12, 16
- Epochen: 4, 8, 12

Das Ergebnis der Random Search waren die folgenden Werte:

- Lernrate: 1e-2
- Batch Size: 16
- Epochen: 4

Vor den ersten Runs wurde ein Early Stopping eingerichtet, wie es in Kapitel 2.5.3.5 bereits erläutert wurde. Dadurch werden Zeit und Ressourcen gespart, da bei negativen Lernprozessen frühzeitig und automatisch gestoppt werden kann. Als Abbruchswert wurde die Accuracy ausgewählt, und die „Patience" auf 1 gestellt. Das bedeutet, dass nach einer Epoche gestoppt wird, wenn die Accuracy sinkt. Schnell wurde jedoch klar, dass dies zu kurz ist, weswegen die Patience auf zwei Epochen hochgesetzt wurde.

Der Optimizer und die Verlustfunktion wurden basierend auf den Erläuterungen in Kapitel 2.5.3.3 ausgesucht. Der Optimizer ist „Adam", einer der beliebtesten und meist verwendeten Optimizer. Als Verlustfunktion wurde die „Sparse Categorical Crossentropy" gewählt, da diese für Klassifizierungen mit mehr als zwei Klassen, wie sie hier gemacht wird, geeignet ist.

Die Werte aus der Random Search wurden weiterverwendet und das Modell damit trainiert. Abbildung 30 zeigt den Verlauf der Accuracy und des Verlustes über alle Epochen.

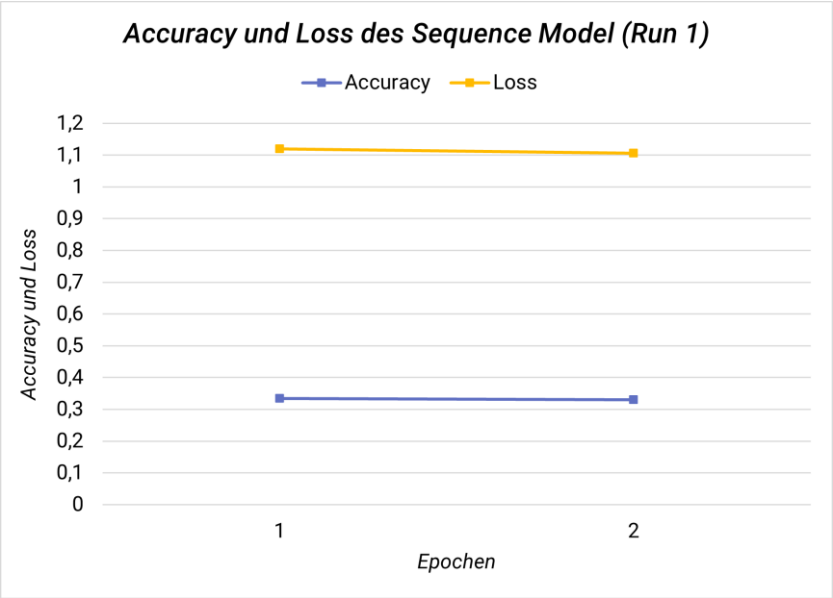

Abbildung 30: Accuracy und Loss des Sequence Models (Run 1)

Es ist ersichtlich, dass der Verlust in allen Epochen mit Werten einem stetigen Verlust von Eins sehr hoch ist. Ebenso ist die Accuracy mit Werten unter Null nicht nur gering, sondern bleibt ebenso über alle Epochen gleich. Genauso wurde das Training nach zwei Epochen gestoppt, da die Accuracy bereits nach einer Iteration nicht gestiegen, sondern gleichgeblieben oder gesunken ist. Hier war die Patience des Early Stoppings noch auf eine Epoche eingestellt. Laut KROHN 2020 weisen diese Werte darauf hin, dass das Modell nicht lernt. Um dies zu vermeiden ist eine der ersten Vorgehensweisen, die Lernrate zu erhöhen. Weiters wurde die Performance des Modells auch mit einer Matrix der Metriken Accuracy, Recall, Precision und F1-Score evaluiert.

Diese Werte werden in einem Classification Report, der alle dieser Metriken darstellt, in Abbildung 31 gezeigt.

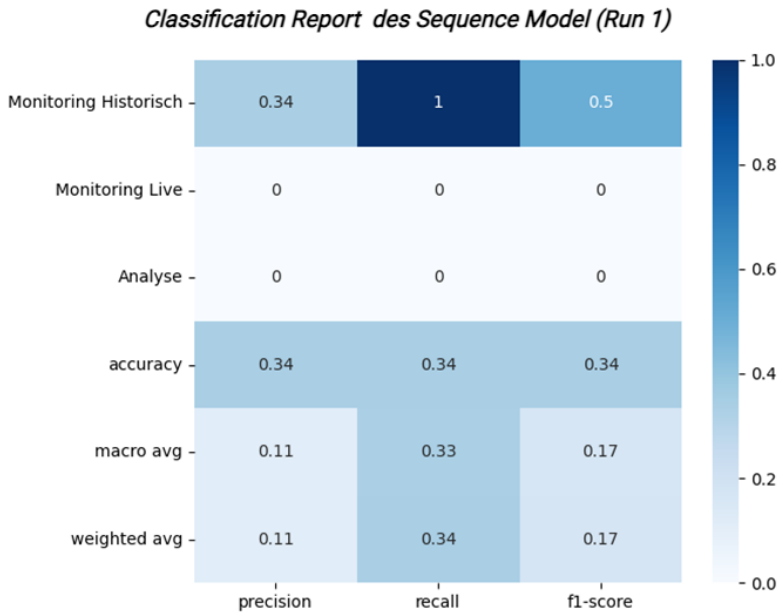

Abbildung 31: Classification Report Sequence Model (Run 1)

Die Werte des Classification Reports werden grundsätzlich für jede Klasse berechnet und angezeigt. Zusätzlich sind die Werte „macro avg“ und „weighted avg“ inkludiert. Ersterer ist der ungewichtete Durchschnitt pro Label, und zweiterer Wert ist nach der Größe der Klasse gewichtet. (vgl. PEDREGOSA ET AL. 2011)

Die Accuracy liegt bei allen Klassen bei einem Wert von 0,34. Die anderen Metriken zeigen das Muster auf, dass nur die Klasse „Monitoring Historisch“ mittel bis gut gelernt wurde. Bei den anderen beiden Klassen wurde jeweils nur ein Ergebnis von Null erzielt. Bei allen der Metriken wird klar, dass im Gesamten keine zufriedenstellenden Ergebnisse erreicht werden konnten, was basierend auf Verlust und Accuracy zu erwarten war. Abgesehen von der Klasse Monitoring Historisch konnte das Modell keine generalisierbaren Eigenschaften anlernen.

Die Werte der Hyperparameter Optimization wurden weiterverwendet, jedoch wurde die Lernrate basierend auf Empfehlungen in KROHN 2020: 118f erhöht. Ebenso wurde das Early Stopping auf eine Patience von zwei Epochen hinaufgesetzt. Die Zahl der Epochen wurde auf fünf erhöh um einen längeren Lernprozess zu ermöglichen. Die Werte der Hyperparameter sind für diesen Run die folgenden:

- Lernrate: 1e-5
- Batch Size: 16
- Epochen: 5

Abbildung 32 zeigt den Verlauf von Accuracy und Verlust über alle Epochen.

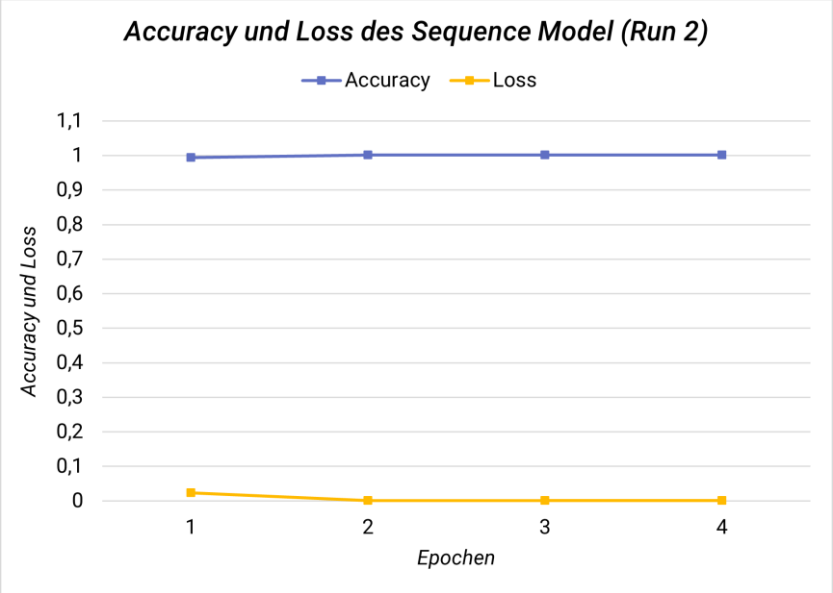

Abbildung 32: Accuracy und Loss des Sequence Models (Run 2)

Der Trainingsprozess wurde nach vier Epochen vom Early Stopping abgebrochen. Es wird offensichtlich, dass die Erhöhung der Lernrate einen eindeutigen Erfolg zeigt. Die Accuracy liegt nun in jeder Epoche bei einem Wert von etwa Eins, während der Verlust gegen Null geht. Die Ergebnisse dieses Runs werden ebenso durch die Berechnung eines Classification Reports berechnet. Dieser ist in Abbildung 33 zu sehen.

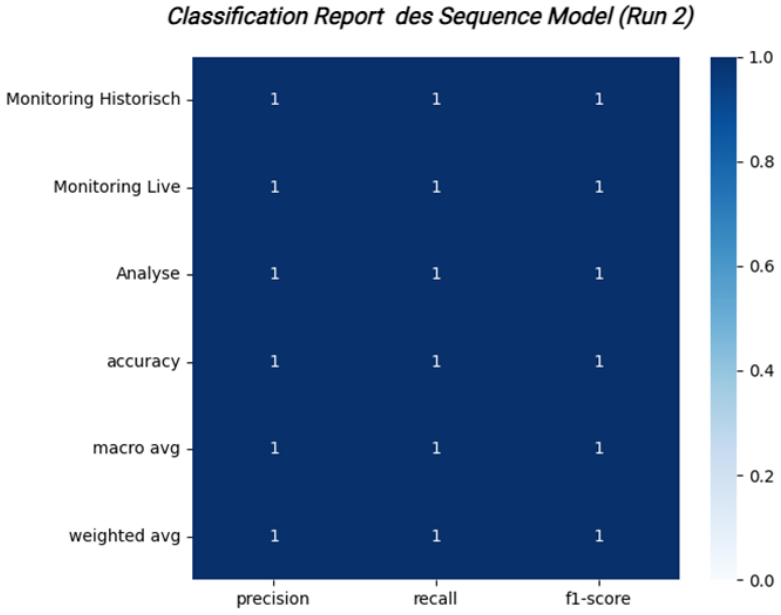

Abbildung 33: Classification Report Sequence Model (Run 2)

Alle Metriken weisen hier bei allen Labels einen Wert von Eins auf. Grundsätzlich zeigt die Evaluierung durch alle Metriken ein sehr gutes Ergebnis. Dies kann jedoch auch zu gut sein, und

ein Zeichen für eine Überanpassung an die Trainingsdaten sein. Da der Classification Report jedoch auch sehr gute Ergebnisse zeigt, obwohl dieser mit den Testdaten berechnet wird, kann der Verdacht auf ein Overfitting nicht bestätigt werden.

Da jedoch aufgrund der Herstellung des Trainings- und Testdatensatzes die Möglichkeit bestehet, dass diese nicht divers genug sind, wurde ein Versuch einer Gegenmaßnahme zum Overfitting durchgeführt. Dazu wurde eine Data Augmentation, wie sie in Kapitel 2.5.3.5 beschrieben wurde, umgesetzt. Es wurde eine Methode des Paraphrasings gewählt, genauer gesagt die „Machine Translation", also eine Übersetzung. Dabei wird ein Satz auf eine andere Sprache übersetzt, und schlussendlich in die Ausgangssprache wieder zurückübersetzt. Vom Übersetzer unterschiedliche Wörter und grammatikalische Strukturen verwendet, und dadurch eine höhere Diversität der Daten ermöglicht. (vgl. LI ET AL. 2022: 11)

Es wurde dafür die Python Library „deep-Translator" verwendet, und die Sätze von Deutsch auf Englisch hin, und wieder zurück übersetzt. Es wurden insgesamt 8000 Sätze damit verändert. Dadurch verschlechterte sich jedoch die Performance des Modells deutlich, und die Ergebnisse waren nicht mehr zufriedenstellend. Die Data Augumentation wurde deswegen beiseitegelegt.

Deshalb wurden die originalen Trainingsdaten verwendet und die finale Kombination der Hyperparameter basierend auf dem vorherigen Testing ausgewählt. Es wurden die folgenden Parameter verwendet:

- Lernrate: 1e-5
- Batch Size: 16
- Epochen: 5

Es wurde die erhöhte Lernrate verwendet, und die Batch Size und die Anzahl der Epochen wurden aufgrund von den positiven Ergebnissen so belassen. Das fertig angelernte Modell wurde auf den Hugging Face Hub exportiert, und somit für das spätere Abrufen bereitgestellt.

4.3.2.3. Token Model

Das zweite Model, das trainiert werden soll, ist das Modell für die Named Entity Recognition. Wie bereits in den vorhergehenden Kapiteln erwähnt wurden für das Training dieses Modells zwei unterschiedliche Datensätze erstellt und akquiriert. Zu Beginn wurde der selbst erstellte Datensatz für das Anlernen ausgewählt.

Ebenso wurde wiederum ein Early Stopping eingerichtet, bei dem wie auch zuvor die Accuracy als Überwachungs-Wert und eine Patience von zwei Epochen gewählt wurde. Auch wurden der

Optimizer und die Verlustfunktion wieder vor dem Training final ausgewählt. Gewählt wurde der Optimizer Adam und die Verlustfunktion Sparse Categorical Crossentropy, da diese auch bei der Named Entity Recognition empfohlen werden. Verändert werden wieder die Lernrate, die Batch Size und Anzahl der Epochen.

Die Erkenntnisse aus dem Training des anderen Modells wurde in den Prozess dieses Modells miteinbezogen. Dies wurde vor allem in zwei Aspekten beachtet. Zum einen wurde, da die Werte der Random Search schlechtere Ergebnisse als die eigens gewählten Hyperparameter-Werte erbracht haben, diese nicht eingesetzt, sondern von Anfang an manuell nach den passenden Werten gesucht. Ebenso wurden die Anfangswerte basierend auf den Ergebnissen und funktionierenden Werten des vorherigen Trainings gewählt. Weiters werden auch hier nur die Runs gezeigt, die wichtige Erkenntnisse im Training zeigen.

Die Einstellungen der Hyperparameter im ersten Run wurden wie folgt gewählt:

- Lernrate: 1e-3
- Batch Size: 16
- Epochen: 5

Die Anzahl der Epochen und die Batch Size wurden wie beim finalen Modell des vorherigen Trainingsprozesses gewählt. Die Lernrate wurde anfangs niedriger gesetzt, um zu überprüfen, wie sich diese hier auswirkt. In Abbildung 34 sind die Accuracy und der Verlust des Trainings des ersten Runs zu sehen.

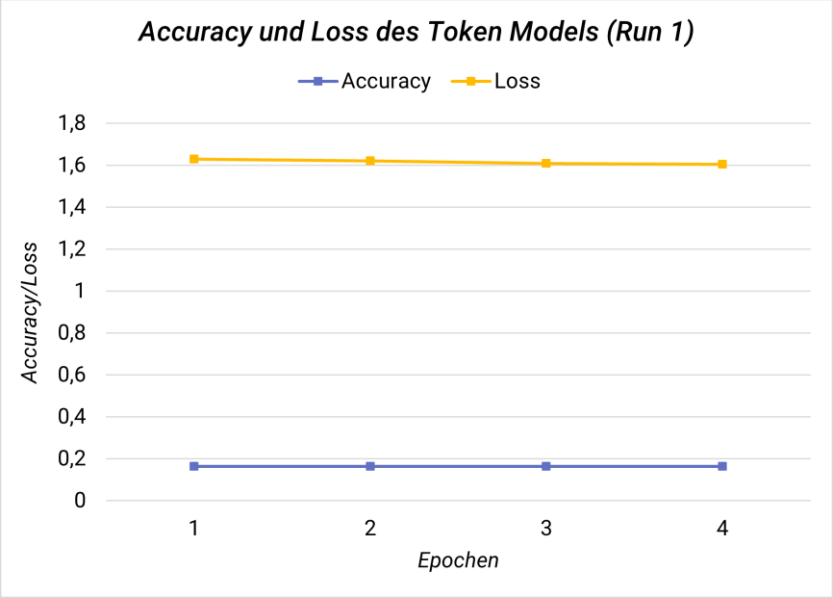

Abbildung 34: Accuracy und Loss des Token Models (Run 1)

Auch hier wurde der Trainingsprozess durch das Early Stopping abgebrochen, sicherheitshalber wird die Anzahl der Epochen jedoch bei fünf belassen. Die Accuracy ist in diesem Run mit einem stetigen Wert unter 0,2 sehr niedrig. Genauso ist der Verlust mit Werten über 1,6 ungewöhnlich hoch. Zusätzlich scheint das Modell auch nicht lernen zu können, da sich die Werte über die Epochen kaum verändern. Auch bei einer Erhöhung der Lernrate wurde keine Verbesserung erzielt, diese war folglich als Fehleerquelle auszuschließen.

Auch für die Evaluierung der Performance des Token Models wurde je ein Classification Report berechnet. Dieser ist in Abbildung 35 dargestellt.

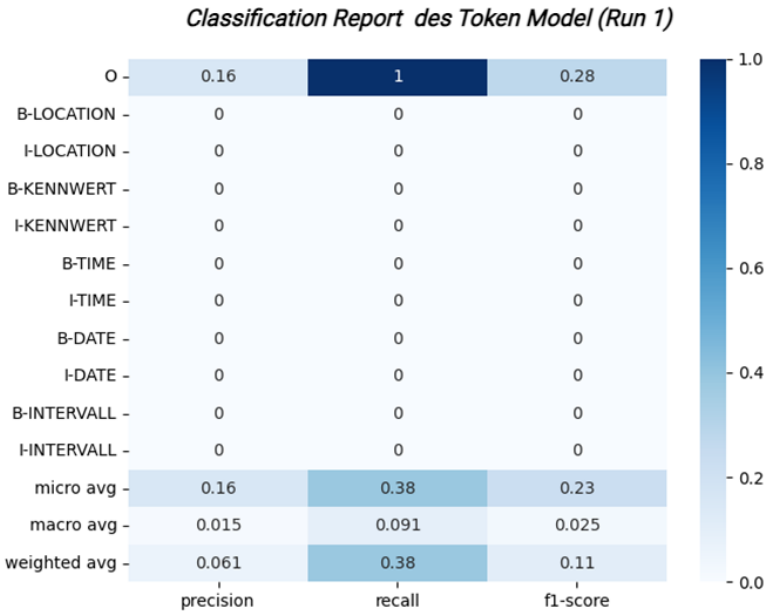

Abbildung 35: Classification Report Token Model (Run 1)

Auch hier spiegelt sich die festgestellte schlechte Performance in beinahe allen Klassen wider. Nur das Tag „O“, dass Wörter beschreibt, denen keine Entität zugewiesen werden konnte, zeigen als einziges ein annähernd angelerntes Muster. Die Werte wie macro avg und weighted avg sind relativ hoch, da das Tag O den Durchschnitt stark anhebt. Alle anderen Labels haben jedoch durchgehend den Wert Null. Daraufhin wurden unterschiedliche Konfigurationen der Hyperparameter getestet, jedoch wurde bei keiner der Kombinationen ein besseres Ergebnis erzielt. Bei einer Überprüfung des Codes wurde offensichtlich, dass die Truncation und das Padding im Tokenizer nicht aktiviert waren. Die beiden Konzepte wurden bereits in Kapitel 2.5.4.2 erwähnt, und stellen einen relevanten Teil der Transformer Modelle dar. Deswegen wurde das Modell noch einmal mit den zuerst gewählten Parametern trainiert, wobei hier die Lernrate hochgesetzt wurde:

- Lernrate: 1e-5
- Batch Size: 16

- Epochen: 5

Abbildung 36 zeigt den Verlust und die Accuracy des Runs.

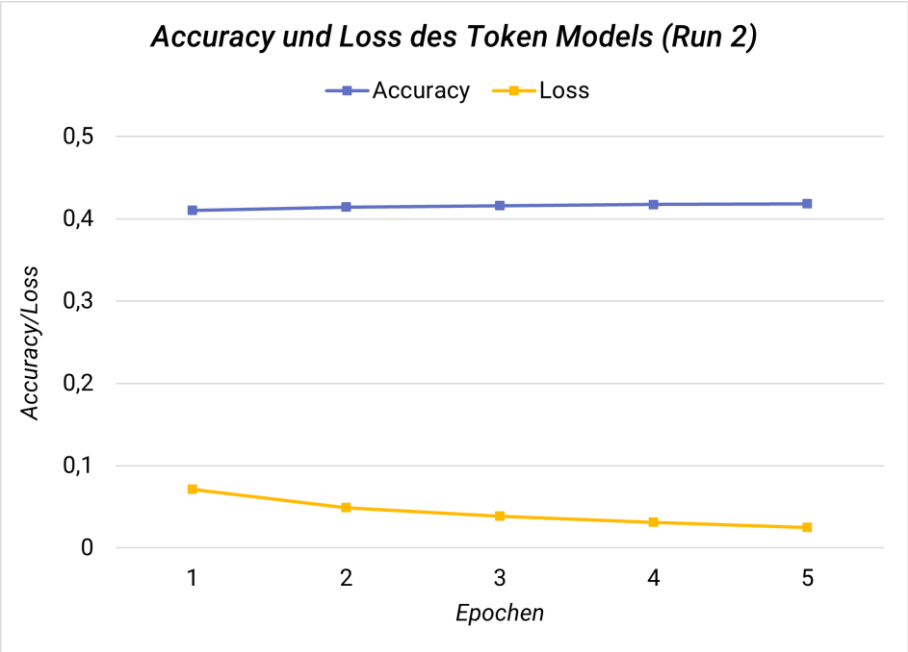

Abbildung 36: Accuracy und Loss des Token Models (Run 2)

Hier wurde der Trainingsprozess durch das Early Stopping nicht abgebrochen, und Die Ergebnisse sind deutlich besser als zuvor. Die Werte des Verlustes liegen stets unter Eins, die Accuracy hingegen ist zwar besser als zuvor, jedoch mit Werten um 0,4 noch nicht vollständig zufriedenstellend.

Da jedoch die Accuracy nicht die einzige aussagekräftige Metrik ist wird wiederum ein Classification Report berechnet, der in Abbildung 37 gezeigt wird.

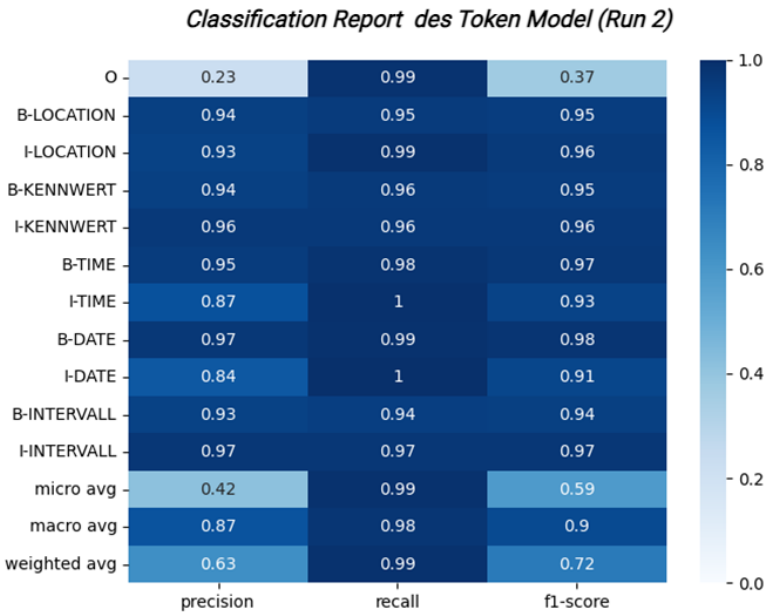

Abbildung 37: Classification Report Token Model (Run 2)

Im Vergleich zum Loss und der Accuracy der Epochen wurden hier sehr gute Ergebnisse in beinahe allen Klassen erzielt. Besonders der Recall, also die Anzahl der korrekt vorhergesagten Labels ist in jeder Klasse sehr positiv. Hier sind alle der Werte über 0,9. Auch die Precision und der F1-Score weisen gute Ergebnisse auf, wobei bei diesen das „O-Tag“ schlechter erkannt wurde. Im Vergleich zu der weniger zufriedenstellenden Accuracy zeigt der Classification Report in jedem der Labels positive Ergebnisse.

Trotz dieser Ergebnisse wurde entschieden, das Training des finalen Modells mit den Trainingsdaten des smartdata Corpus durchzuführen. Diese Entscheidung basiert auf mehrere Faktoren. Einer dieser ist, dass die zwei Variablen Intervall und Kennwert über eine zu kleine Klassengröße verfügen, die aufgrund von einem Fehlen an unterschiedlichen Formulierungen entsteht, als das hier generalisierbare Muster erlernt werden können. Das Modell hat sich scheinbar zwar darauf angepasst, jedoch können hier von User:innen neu eingegeben Synonyme vermutlich nicht erkannt werden. Ebenso weisen die Daten des smartdata Corpus im Gesamten eine größere Diversität in den Daten auf, da diese basierend auf echten Informationen gesammelt wurden. Die Wahl des Corpus soll dazu führen, dass diverse Formulierungen der User:innen erkannt werden können.

Zum Training des Token Models mit den smartdata Daten wurden anfangs die bereits erprobten Hyperparameter wiederverwendet:

- Lernrate: 1e-5
- Batch Size: 16
- Epochen: 5

Das Ergebnis des Anlernprozesses ist in Abbildung 38 zu sehen.

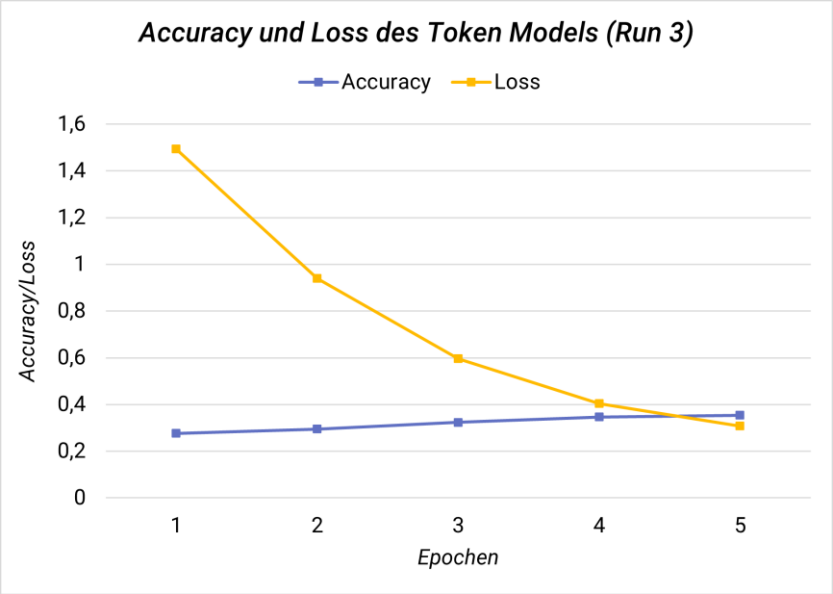

Abbildung 38: Accuracy und Loss des Token Models (Run 3)

Der Verlust sinkt über alle Epochen stark, während die Accuracy auf einen Wert von 0,4 steigt. Der Verlauf der Accuracy ähnelt hier stark dem des vorherigen Runs mit den eigenen Trainingsdaten.

Auch hier wird wiederum der Classification Report berechnet. Dieser ist in Abbildung 39 dargestellt.

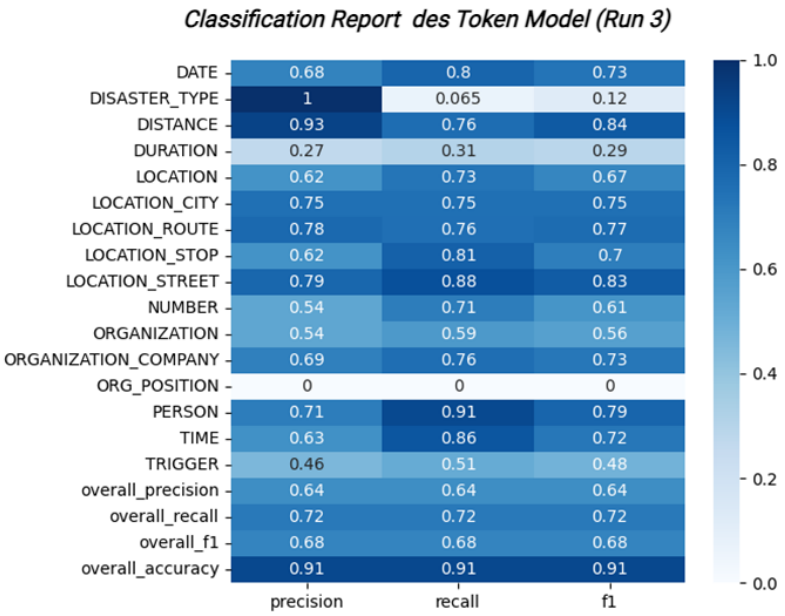

Abbildung 39: Classification Report Token Model (Run 3)

Hier ist nur die Performance der Klassen DATE, TIME und LOCATION_STREET von Relevanz, da nur diese weiterverwendet werden. Die Precision liegt überall über 0,8. Alle anderen Metriken liegen bei den drei Labels mindestens über 0,6 und oft auch über 0,7. Die Klasse LOCATION_STREET wurde im Gesamten am besten erkannt. Auch, wenn die Ergebnisse hier schlechter als bei den eigenen Daten sind, kann dies positiv sein, da das Modell dadurch empfänglicher für vom Standard abweichende Angaben ist.

Es wurden folgend noch unterschiedliche Hyperparameter-Kombinationen probiert, um eine bessere Performance zu erreichen, jedoch wurde das beste Ergebnis mit dem vorherigen Run 3 erzielt. Deswegen wurde das finale Modell mit diesen Hyperparameter-Werten trainiert:

- Lernrate: 1e-5
- Batch Size: 16
- Epochen: 5

Das Modell wurde auf den Hugging Face Hub zur Weiterverwendung hochgeladen.

4.4. Implementierung in das Web-GIS

In folgendem Kapitel wird die Implementierung der trainierten Modelle in die neue Anwendung beschrieben. Nach dem Anlernen der Modelle müssen diese in das Web-GIS implementiert werden, und ein Framework für die Verarbeitung der Daten geschaffen werden. Das Framework

der Datenverarbeitung und Integration der Modelle wird mittels Docker bereitgestellt und so in die Anwendung eingebaut. Das Framework für die gesamte Verarbeitung sowie der Docker wurden selbst erstellt, und das Interface und das gesamte Backend wurde vom Entwicklerteam der Firma entwickelt.

Der erste Schritt ist das Einbinden der Modelle für die Text Klassifizierung und die Named Entity Recognition. Wie bereits erwähnt wurden beide der Modelle nach dem Trainingsprozess auf den Hugging Face Hub hochgeladen. So werden sie im Framework für die Implementierung auch von dort heruntergeladen. Das Modell wird im Docker gespeichert und nur einmal am Anfang geladen. Ebenso wurden die Tokenizer eingebunden, um den Input-Text für die Analyse im Modell aufzubereiten.

Abbildung 40 zeigt die gesamte Struktur des Frameworks. Hier werden nur die für die Verarbeitung der Input-Daten relevanten Schritte miteingebaut, ein Flowchart des gesamten Backends des Web-GIS wird in Kapitel 3.2.2 konkreter erläutert und dargestellt.

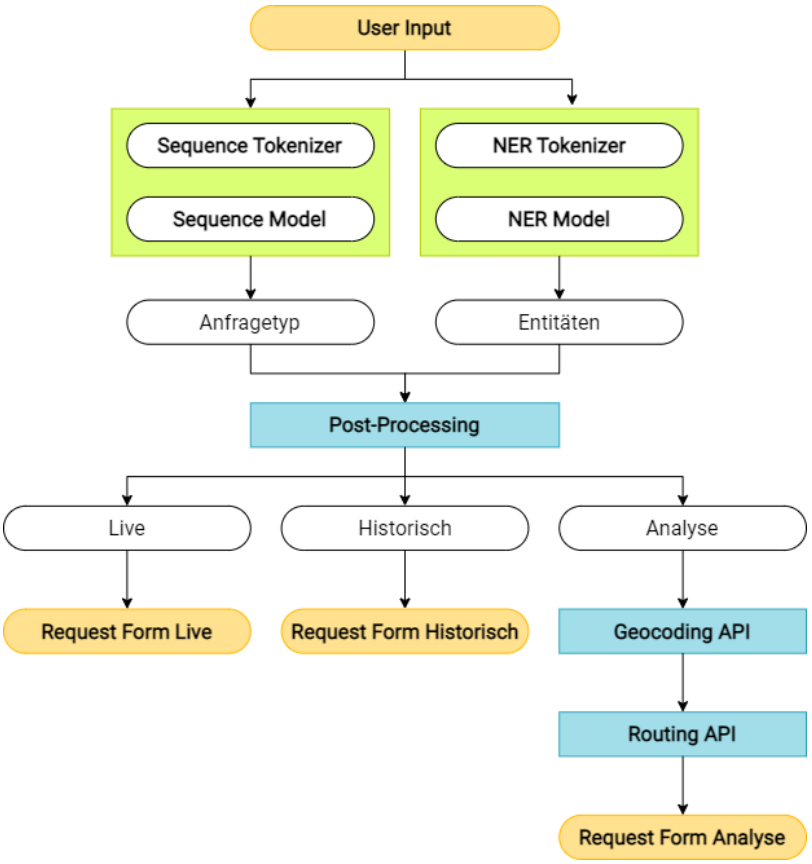

Abbildung 40: Backend des textgesteuerten Web-GIS

Wie bereits erwähnt wurde, werden die Daten an die Datenbank als JSON-Datei übermittelt, die durch den Input-Text befüllt wird. Hierzu sind die Ergebnisse der Modelle verantwortlich. Im ersten Schritt wird der von User:innen eingegebene Text mit dem Sequence Model analysiert. Das

Ergebnis ist eine Klasse: Monitoring Live, Monitoring Historisch und Analyse. Jede dieser Anfragearten hat ein eigenes JSON-File das benötigt wird. Basierend auf den Ergebnissen der Text Klassifizierung wird das passende JSON ausgesucht und geladen. Alle JSON-Files sind im Anhang 10.2 zu finden.

Im nächsten Schritt wird die Analyse mit dem zweiten Modell, dem Token Model gestartet. Ergebnis ist eine Extraktion der Entitäten Datum, Zeit und Location bzw. Straßen. Die extrahierten Daten müssen noch eine weitere Verarbeitung durchlaufen, um ins JSON-File eingefügt werden zu können und passend für die Datenbankabfrage zu sein.

Vor der weiteren Verarbeitung der Daten wurde festgelegt, inwiefern Fehler von User:innen wie falsche Angaben oder fehlende Angaben behandelt werden sollen. Es wurde anfangs überlegt, bei fehlenden Angaben wie einer fehlenden Datumsangabe zusätzlich zu dem angegebenen Datum den Zeitraum von einer Woche anzugeben. Hier wäre die Wahrscheinlichkeit jedoch sehr hoch, dass das Ergebnis nicht das ist, was die User:innen wollten. Es könnte zusätzlich verwirren, wenn das Ergebnis anders als gewünscht wäre. Deswegen wurde entschieden, fehlende Angaben nicht auszubessern, sondern mit einer Fehlermeldung abzufangen. Die weitere Frage ist, wie mit falschen Angaben wie mit einem Datum, das außerhalb der verfügbaren Daten liegt, umgegangen werden soll. Hier wurde ebenfalls entschieden, keine Fehler auszubessern, sondern es den User:innen mittels Fehlermeldungen mitzuteilen. Die Möglichkeiten von Fehlern sind so vielfältig, dass nicht alle behandelt werden können. Deshalb wurde ein ganzheitlicher Ansatz gewählt, in alle Fehler mit einer Fehlermessage gemeldet werden, und keiner dieser behandelt wird.

Folgend wird der Prozess der Datenverarbeitung erläutert. Der gesamte Ablauf ist in Abbildung 41 dargestellt.

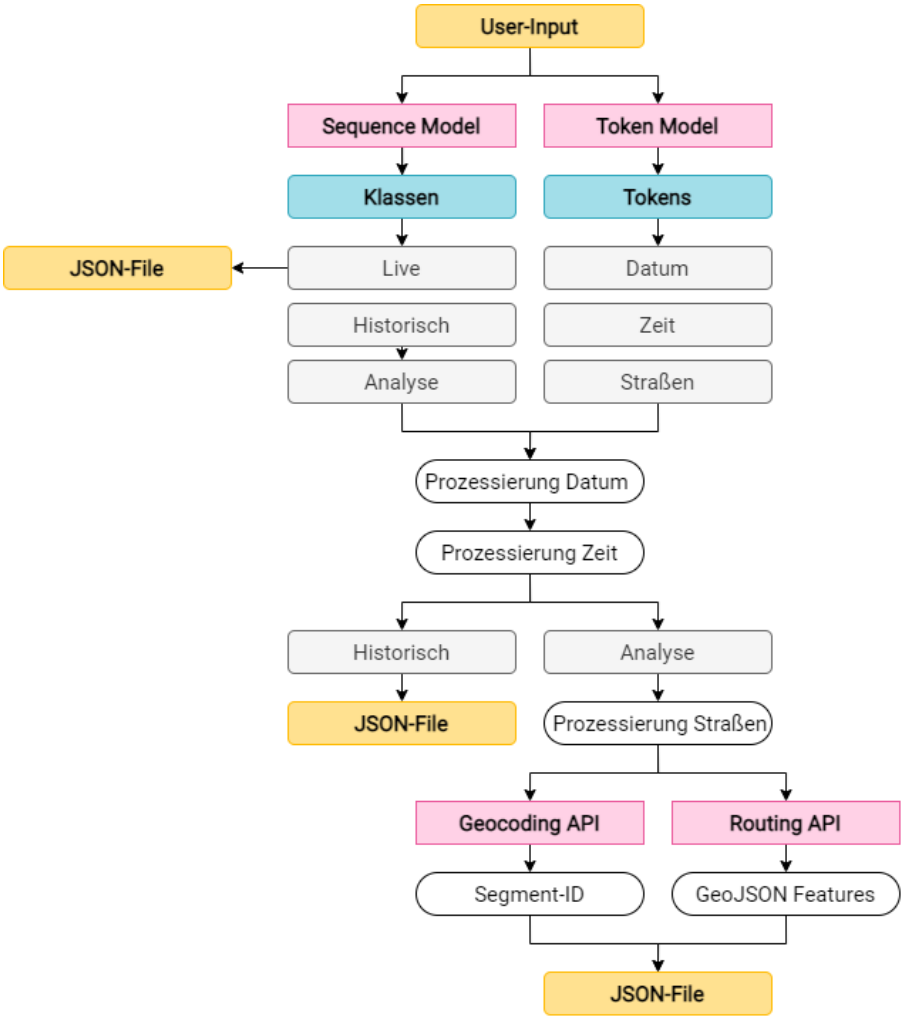

Abbildung 41: Prozessierung des Input-Textes durch den Docker

Nach der Analyse mit beiden Modellen starten die weitere Verarbeitung und Befüllung des ausgewählten JSON-Files. Da die Anfrage für die Live-Ansicht keine Informationen oder Entitäten benötigt, wird nach der Feststellung der Klasse das fertige JSON-File weitergegeben. Ist die Abfragemöglichkeit jedoch Monitoring Historisch oder die Analyse, werden die Daten noch weiterverarbeitet.

Im ersten Schritt werden die extrahierten Datums- und Zeitangaben bereinigt, in Auflistungen gegeben und in ein verarbeitbares Format umgewandelt. Sind die Daten aus Gründen wie falschen Angaben von User:innen nicht lesbar, wird eine Fehlermeldung zurückgegeben. War der Vorgang erfolgreich, werden Datum und Zeit zu Timestamps und Datestamps weiterverarbeitet. Da User:innen die Angaben in unterschiedlichsten Formaten angeben können wird versucht, das korrekte Datum- und Zeitformat zu finden. Mit den originalen Formaten können die Informationen zu standardisierten Formaten, in diesem Fall dem UTC-Format umgewandelt

werden. Hier werden weiters Angaben, die ein falsches Jahr oder nicht bestehende Zeiten abgeben herausgefiltert und bei Bedarf eine Fehlermeldung gesendet. Ist die Anfrage Monitoring Historisch, wird der Prozess an dieser Stelle abgeschlossen. Es wurden alle benötigten Angaben extrahiert und verarbeitet.

Soll jedoch eine FCD-Verkehrsanalyse durchgeführt werden, müssen zusätzlich die Straßen für das Routing verarbeitet werden. Ein Teil dieser Verarbeitung ist auch das Geoparsing, das in Kapitel 2.4.1 bereits beschrieben wurde. Die vorherige Analyse des Token Models übernimmt das Geotagging, bei dem die Straßen aus dem unstrukturierten Input-Text extrahiert werden. Um die Straßen zu verarbeiten, muss die Liste an extrahierten Straßen in Start- und Endpunkt-Paare verarbeitet werden. Anfangs wurde überlegt, die Präpositionen vor den Straßen als Kennzeichnung dafür zu nehmen, in welche Richtung die Route umgesetzt werden soll. Jedoch wurde hier schnell klar, dass dieser Ansatz zu Unsicherheiten und Fehlern führen kann. Deswegen wird angenommen, dass der Startpunkt die erste Straße der Paare ist, und zweitere der Endpunkt.

Der nächste Schritt ist das Geocoding und das Routing, das je über eine API durchgeführt wird. Das Geocoding ist, wie bereits erwähnt, der zweite Schritt des Geoparsings, bei dem den extrahierten räumlichen Informationen die passenden Koordinaten zugewiesen werden.

Die API für das Geocoding basiert auf den OSM-Daten und ist bereits eingeschränkt auf die Stadt München und die Straßenkategorien, die im Netz der Anwendung enthalten sind. Je Start- und Endstraße werden die Longitude und Latitude aus dem Geocoder zurückgegeben. Bei einer Rückgabe von mehreren Ergebnissen des Geocoders wird das erste gewählt. Da in dieser Arbeit nur ein Geoparsing im Generellen umgesetzt werden muss, aber die Erstellung eines sehr differenzierten Geoparsings nicht zu den Zielen der Arbeit gehört, wird kein detaillierteres Geoparsing umgesetzt.

Weiters folgt das Routing. Die API nimmt die Start- und Endpunkte und gibt eine Route zurück. Es wird ein Array der Routen, die verwendet werden, zurückgegeben. Das Ergebnis kann entweder in der Form von IDs, die in der Datenbank für jedes Segment festgelegt wurde, zurückgegeben werden, oder als GeoJSON-Feature. Beides wird für die Analyse benötigt und eingefügt, da das Array von GeoJSON-Features für die Darstellung der Route, und die Segment-IDs für die Abfrage des Kennwerts auf den Segmenten und somit die Darstellung in des Diagramms verantwortlich ist. Kann hier entweder das Geocoding oder das Routing für eine Route nicht umgesetzt werden, bekommen die User:innen eine Fehlermeldung.

Die restlichen Informationen, die für die FCD-Verkehrsanalyse benötigt werden, sind der Kennwert und die Intervalle. Wie in Kapitel 4.3.2.1 bereits erwähnt war es bei beiden Entitäten

aufgrund der zu kleinen Datenmenge nicht sinnvoll, das Modell damit zu trainieren. Da es ohne das Natural Language Processing kaum möglich ist, die Angaben zu verarbeiten, wurde davon abgelassen und fixe Werte eingestellt.

Die Daten sind somit fertig verarbeitet und in das passende JSON-File eingefügt, mit dem nun die Datenbank abgefragt wird, und das Ergebnis in Karte und Diagramm angezeigt werden kann.

5. Usability Testing

Um die Performance des textgesteuerten Web-GIS in der Verwendung durch User:innen zu evaluieren, wird ein Usability Testing konzipiert und durchgeführt. Um dies in Verhältnis zur Performance der buttongesteuerten Anwendung setzen zu können, werden beide miteinander verglichen. Das folgende Kapitel umfasst eine Beschreibung des Konzeptes der Studie sowie alle zugehörigen Aspekte, basierend auf Kapitel 2.2.. Abschließend wird auf die Durchführung der Studie sowie die Ergebnisse eingegangen.

5.1. Studienkonzept

Die Studie wird basierend auf den Prinzipien und Grundkonzepten der Human Computer Interaction und dem Usability Testing erstellt. Es werden die folgenden Aspekte festgelegt:

- Studienziel
- Art der Studie
- Studienteilnehmer:innen
- Tasks
- Metriken

Anfangs werden die Rahmenbedingungen der Studie ausgewählt, und anschließend die Metriken, mit denen im Usability Testing die Performance evaluiert werden soll.

5.1.1. Studienziele

Im ersten Schritt werden die Ziele der Studie festgelegt, um eine zielgerichtete Konzeption und Durchführung dieser zu ermöglichen. Durch das Usability Testing sollen Erkenntnisse darüber erlangt werden:

- ob das textgesteuerte Web-GIS funktioniert und unterschiedlichste Aufgabenstellungen damit lösbar sind
- wo Fehlerquellen liegen und welche Arten von Fehlern bei der Verwendung des Web-GIS entstehen
- welche Verbesserungen in der Anwendung implementiert und umgesetzt werden können
- ob das textgesteuerte Web-GIS für die User:innen eine Alternative zum buttongesteuerten Web-GIS ist.

Dies stellt die Grundlage aller weiteren Aspekte der Studienplanung dar. Die Ziele sollen durch die gesammelten Informationen und Daten erreicht werden.

5.1.2. Art der Studie

Die Grundlagen für die Art der Studie sind in Kapitel 2.2.2.2 zu finden. Für die Durchführung der Studie wurde die Form eine Remote Studie gewählt. Das bedeutet, dass alle Einheiten über Online-Meetings abgehalten werden. Grund dafür ist, dass Personen, die als Studienteilnehmer:innen ausgewählt werden sollen in verschiedenen Städten und Ländern ansässig sind. Dadurch ist es weder vom Kosten- noch vom Zeitaufwand sinnvoll, eine Studie vor Ort durchzuführen. Weiters wird so ermöglicht, mehr Teilnehmer:innen zu rekrutieren.

Die Durchführung einer Remote Studie bringt jedoch auch weitere Vorteile mit sich. So verwenden die User:innen das Web-GIS in ihrem natürlichen Umfeld, und auf dem eigenen Gerät. Da die Anwendung weiters einfach über einen URL abgerufen werden kann, stellt hier die Installation der Software kein Problem dar. Die mögliche Problematik einer schlechten Internetverbindung muss aufgrund der anderen Vorteile und der Notwendigkeit einer Remote Studie in Kauf genommen werden.

Ebenso muss zwischen einer moderierten oder unmoderierten Studie gewählt werden. Umgesetzt wird eine moderierte Studie. Im Vergleich zur unmoderierten Studie ist es dadurch möglich, Studienteilnehmer:innen einzuweisen, Fragen zu beantworten und bei Problemen einzugreifen. Weiters kann abgesichert werden, dass die Personen die Aufgabenstellungen korrekt verstehen.

Dieser Usability Test kann als eine Mischung zwischen einer formativen und summativen Studie bezeichnet werden. Das Testing hat den Charakter einer summativen Studie, da diese nach Fertigstellung des Web-GIS für diese Masterarbeit durchgeführt wird. Es wird jedoch nur eine kleine Anzahl an Studienteilnehmer:innen zugezogen, wie in den folgenden Kapiteln konkreter erläutert wird, was so für eine formative Studie spricht. Weiters ist das Web-GIS der finale Stand für diese Arbeit, sollte jedoch für ein Produktivsystem noch weiterentwickelt werden. So gesehen könnte die Studie auch als summativ bezeichnet werden, da sie nur für einen Zwischenstand der Anwendung durchgeführt. Die Studienform ist folgend eine Mischung aus formativer und summativer Studie.

5.1.3. Profil der Studienteilnehmer:innen

Folgend wird das Profil der Studienteilnehmer:innen, die rekrutiert werden sollen, beschrieben. Dieses soll die Zielgruppe der User:innen, die das Web-GIS später auch produktiv verwenden, wiederspiegeln. Die Auswahl des Profils wird basierend auf der Literatur in Kapitel 2.2.2.3 getroffen.

Es werden insgesamt 10 Teilnehmer:innen rekrutiert. Laut Studien sind bei der Verwendung von fünf Personen bereits 80-85% aller Erkenntnisse einer Studie aufgedeckt. Weiters wird bei einer

moderierten Studie eine Mindestanzahl von drei bis fünf Personen, und eine maximale Anzahl von 15 Personen empfohlen werden. Dies ist zwar besonders für qualitative Studien gültig, da diese jedoch auch in dieser Arbeit von besonderer Relevanz sind und auch gesammelt werden, wird dieser Referenzwert verwendet.

Um User:innen zu rekrutieren müssen die Charakteristiken dieser formuliert werden. Für diese Studie wurde die berufliche Tätigkeit und Erfahrungen als zentraler Aspekt der Personenauswahl gewählt. Weitere Faktoren wie demographische Charakteristiken werden zwar beachtet, bestimmen allerdings nicht, welche Personen ausgewählt werden. Alle der ausgewählten Personen arbeiten im Consulting in der Verkehrsplanung, entweder mit Fokus auf das Consulting oder auf die Informatik. Dies ist auch die Zielgruppe der Personen, die das Web-GIS schlussendlich produktiv verwenden, da dieses ein Experten-Tool für Personen mit Kenntnissen in der Verkehrsplanung ist. Alle der Studienteilnehmer:innen sind Mitarbeiter der Firma Trafficon. Es wurde darauf geachtet, dass keine der Personen bei der Entwicklung der Anwendung oder dem originalen DTM mitgewirkt hat.

Es werden keine speziellen technischen Fähigkeiten von den Studienteilnehmer:innen gefordert. Jedoch wird aufgrund des Berufsbildes eine grundsätzliche Erfahrung im Umgang mit dem Computer und dem Internet vorausgesetzt.

Weiters gibt es bei den Teilnehmer:innen grundsätzlich keine Anforderungen an die demographischen Charakteristiken dieser. Allerdings wurde darauf geachtet, eine Mischung dieser zu erreichen. So werden beispielsweise Personen aus unterschiedlichen Altersgruppen rekrutiert. Weiters wurden für die Studie je fünf männliche und weibliche Personen ausgewählt.

Beim Usability Testing wird in vielen Fällen ein „Screener“ erstellt, der für die Rekrutierung von Teilnehmer:innen verwendet wird. Dies ist ein Fragebogen, in dem unterschiedlichste Fragen zur Demographie, technischen Fähigkeiten, Gewohnheiten und weitere Eigenschaften von User:innen enthalten sind. (vgl. BARNUM et al. 2011: 124). In diesem Fall werden durch den Screener die technischen Fähigkeiten und demographischen Charakteristiken der Teilnehmer:innen abgefragt.

Abbildung 42 zeigt das Ergebnis des Screeners für die Verteilung der Geschlechter und der Altersgruppen. Hinzuzufügen ist, dass nur neun der Studienteilnehmer:innen den Screener ausgefüllt haben. Aufgrund der Anonymität konnte auch nach mehrmaligem Aufruf keine zehnte Abstimmung erhalten werden.

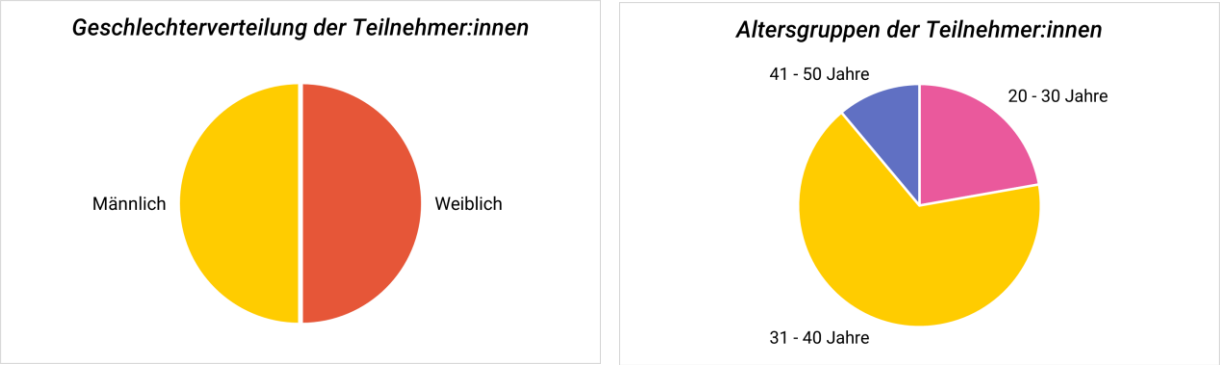

Abbildung 42: Angaben zur Geschlechterverteilung (Links) und Altersgruppen (Rechts) der User:innen

Rechts ist die Geschlechterverteilung der Teilnehmer:innen aufgezeigt, wie sie im Screener aufgezeichnet wurde. Links ist die Verteilung der Altersgruppen gezeigt. Hier konnte zwar kein ausgeglichenes Ergebnis zwischen allen der Gruppen erreicht werden, jedoch konnten zumindest Personen aus drei verschiedenen Gruppen rekrutiert werden.

Ebenso wurden die Personen nach ihren Kenntnissen im Umgang mit Computer und Internet abgefragt. Abbildung 43 zeigt die Verteilung.

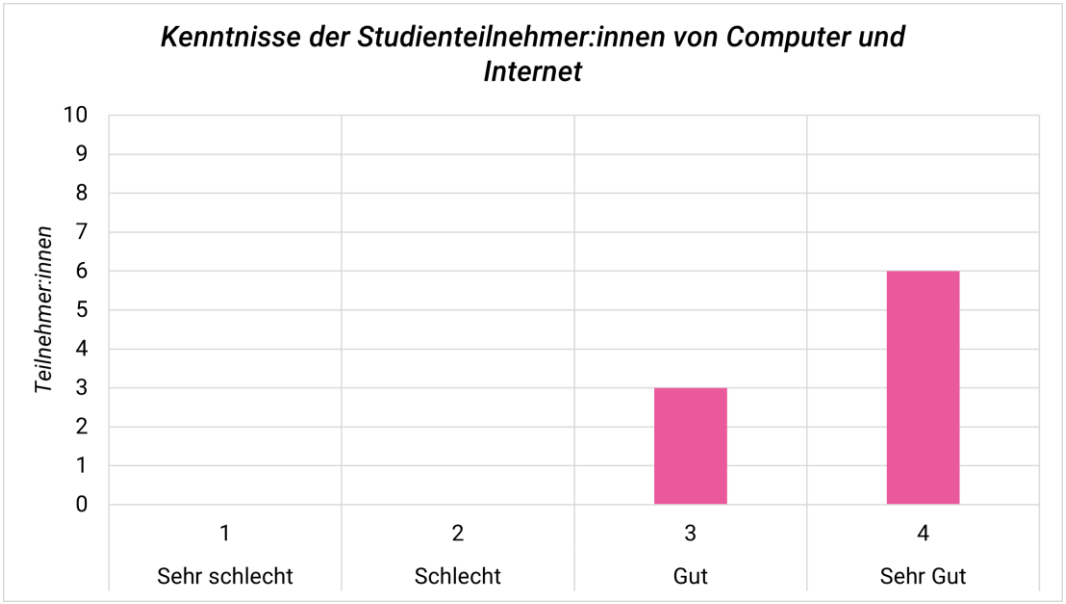

Abbildung 43: Bewertung der Studienteilnehmer:innen der eigenen Erfahrungen im Umgang mit Computern und dem Internet

Alle Studienteilnehmer:innen gaben an, gute oder sehr gute Kenntnisse im Umgang mit Computern und dem Internet zu haben. Dies ist für die Studie grundsätzlich von Vorteil, da somit sichergestellt ist, dass zu mindestens die grundsätzlichen Funktionen von Websiten bekannt sind.

5.1.4. Szenarios und Aufgabenstellungen

Im nächsten Schritt werden die Szenarios und Aufgabenstellungen beschrieben, die in der Studie von den Teilnehmer:innen gelöst werden sollen. Diese wurden basierend auf den Grundlagen in Kapitel 2.2.2.4 realisiert.

Insgesamt wurden sechs Aufgaben erstellt. Diese werden von den Teilnehmer:innen je in der buttongesteuerten und der textgesteuerten Anwendung gelöst. Anfangs wurde überlegt, keine vollständigen Szenarios oder Aufgabenstellungen zu formulieren, sondern nur die relevanten Informationen in Stichpunkten anzugeben. Basierend auf der Literatur wurde jedoch entschieden, vollständige Sätze und Szenarios den User:innen vorzulegen. Es wird auch darauf geachtet, dass die Tasks keine Anleitung der Lösung darstellen. Ebenso wurde versucht, die Formulierungen pro Task unterschiedlich zu gestalten, wie bei die Form von Datums- und Zeitangaben. Die Tasks bilden die drei Abfragearten Live, Historisch und FCD-Verkehrsanalyse wieder ab. Alle Aufgabenstellungen sind in Tabelle 6 zu sehen.

Tabelle 6: Aufgabenstellungen der Studie und deren geplante Dauer

	Szenario	***Dauer***
Task 1	Finde heraus, wie die Verkehrslage am 3. Mai 2023 um 10 Uhr war.	3 min
Task 2	Versuche herauszufinden, wie die momentane Verkehrslage in München ist.	2 min
Task 3	Lasse dir die durchschnittliche Geschwindigkeit auf der Route von der Herzogstraße zu Maximilianstraße zwischen 17.3.2023 und 21.3.2023 zwischen 05:00 Uhr und 21:00 Uhr anzeigen.	4 min
Task 4	Finde heraus, wie die durchschnittliche Geschwindigkeit auf der Route von der Verdistraße zur Prinzregentenstraße von 22. April 2023 bis 27. April 2023 zwischen 6 Uhr und 22 Uhr war.	4 min
Task 5	Lasse dir anzeigen, wie die durchschnittliche Geschwindigkeit auf der Route von der Frauenstraße zu Oberweg , und auf der Route von Würmtalstraße zu Triebstraße zwischen 16.5.2023 bis 19 .5.2023 zwischen 7:00 Uhr und 21:00 Uhr ist.	5 min
Task 6	Vergleiche die durchschnittliche Geschwindigkeit auf den Routen von der Lindwurmstraße zur Sonnenstraße und von Blumenstraße bis Falkenstraße zwischen 8. Juni 2023 und 11. Juni 2023 und zwischen 6 Uhr und 22 Uhr.	5 min

Am Anfang werden die einfachsten Szenarios abgefragt. Die User:innen müssen dazu eine Abfrage der historischen Daten machen. Weiters wird die Umsetzung einer Live-Anfrage gefordert. In allen weiteren Szenarios soll eine FCD-Verkehrsanalyse durchgeführt werden. Dabei werden anfangs zwei Tasks mit nur einer Route, und anschließend zwei Aufgaben mit je zwei Routen abgefragt. In der Tabelle ist ebenso die maximale Dauer, die für die Tasks eingeplant wird, angezeigt. Diese wurde je nach Komplexität des Tasks gewählt.

Hier fällt bei der Formulierung auf, dass die Entitäten Kennwert und Intervall nicht inkludiert. Wie in Kapitel 4.4 erwähnt, wurden diese aufgrund zu wenig Diversität in Synonymen und Formulierungen nicht dynamisch, sondern mit einem fixierten Wert angegeben. Deswegen werden diese auch nicht in die Szenarios miteinbezogen. Beim textgesteuerten Web-GIS müssen die Studienteilnehmer:innen hier keine Informationen angeben, und können die Formulierungen auslassen. Beim buttongesteuerten Web-GIS werden die Personen angewiesen, die Default-Werte der Entitäten zu belassen.

Die Szenarios werden den Teilnehmer:innen zum Start der Studie übermittelt. Es besteht auch die Möglichkeit, erst nach Abschluss eines Szenarios das nächste freizugeben, dies kann jedoch zu nicht nötigen Unterbrechungen der Studie führen.

Weiters wurde festgelegt, wann eine Aufgabenstellung als gelöst bezeichnet wird. In diesem Fall wird, nachdem die User:innen ein korrektes Ergebnis erreicht haben, dieses gemeinsam mit der Moderatorin überprüft. Wurde ein korrektes Ergebnis erzielt, wird der Task als gelöst und abgeschlossen bezeichnet.

Wie bereits erwähnt werden beide Anwendungen getestet, und dadurch auch alle sechs Aufgabenstellungen in beiden Interfaces gelöst. Dabei ist laut BARNUM 2011: 133 zu beachten dass die Hälfte der Teilnehmer:innen mit dem textgesteuerten Web-GIS, die andere Hälfte mit dem buttongesteuerten Web-GIS startet, um ausgeglichene Ergebnisse zu erhalten.

5.1.5. Metriken

Durch das Usability Testing werden unterschiedliche Daten gesammelt, und die Performance der Interfaces mit mehrere Metriken evaluiert, die bereits in Kapitel 2.2.2.5 beschrieben wurden. Es werden die folgenden Metriken evaluiert:

- Task Success
- Time on Task
- Learnability
- User Satisfaction

Folgend wird beschrieben, in welcher Form die ausgewählten Metriken in der Evaluierung des Web-GIS eingesetzt werden.

Task Success

Der Task Success beschreibt wie bereits erwähnt, ob eine Aufgabenstellung korrekt gelöst wurde. Für diese Studie wurde eine Skala für die Level des Success mit vier Stufen erstellt:

- 0: Misserfolg
- 1: Erfolg mit Problem und Lösung durch Moderatorin
- 2: Erfolg mit Problem Lösung User:in
- 3: Erfolg

Als Level 0 oder Misserfolg werden Tasks bewertet, die nicht korrekt gelöst werden konnten. Grund hierfür kann sein, dass die User:innen kein erfolgreiches Ergebnis bekommen bzw. keine erfolgreiche Anfrage stellen können. Weiters kann hierfür ein Fehler in der Anwendung selbst ein Auslöser sein. Wird die Lösung eines Tasks als Level 1 oder Level 2 eingestuft, sind Probleme aufgetreten. Diese konnten entweder nur mithilfe der Moderatorin, wie bei Level 1, oder von den User:innen selbst gelöst werden, wie es Level 2 beschreibt. Level 3 beschreibt die erfolgreiche Lösung einer Aufgabe.

Bei Fehlern oder Problemen wird den Testpersonen grundsätzlich die Chance gegeben, diese selbstständig auszubessern und zu lösen. Weiters soll individuell entschieden werden, ob den Teilnehmer:innen Hilfestellung geboten wird. Beim Auftreten eines Fehlers, der durch die Anwendung selbst entsteht, wird eingegriffen und versucht, diesen gemeinsam zu lösen.

Auch wurde ein „Stopping Point" festgelegt der definiert, wann die Moderatorin die Aufgabenlösung von selbst stoppt. Es wird grundsätzlich festgelegt, dass bei Erreichen des Zeitlimits eines Tasks dieser gestoppt wird. Wenn offensichtlich ist, dass User:innen frustriert sind oder keinen richtigen Lösungsansatz finden, wird die Durchführung ebenso frühzeitig gestoppt.

Time on Task

Die Time on Task beschreibt die Zeit, die benötigt wird, um eine Aufgabenstellung zu lösen. Ebenso beschreibt die Metrik die Effizienz. Diese wird durch die Messung von den Zeiten evaluiert.

Es muss ein Start und ein Stopp für die Messung festgelegt werden. In diesem Fall wird die Zeit ab dem Moment, ab dem User:innen die Eingabe in das Textfeld oder mittels Buttons beginnt, gestartet, und wird beendet, sobald das Eingabeformular durch den Klick auf den Start-Button oder der Eingabetaste abgeschickt wird. Da Faktoren wie die Dauer einer Datenbankabfrage durch die Internetverbindung beeinflusst werden können, wird nicht das Erscheinen einer Visualisierung im Web-GIS als Endpunkt verwendet.

Learnability

Bei der Learnability wird überprüft, wie lange es dauert, bis User:innen eine Anwendung effizient und produktiv verwenden können. Dies kann beispielsweise durch die Time on Task und den Task Success untersucht werden, in dem diese Metriken über einen Zeitraum oder die Iteration von Aufgabenstellungen gemessen werden.

Für dieses Usability Testing wird für die Learnability die Time on Task und der Task Success überprüft. Es werden die Tasks 3-6 zur Evaluierung herangezogen, da diese alle dieselbe Anfrage, die FCD-Verkehrsanalyse, mit steigender Komplexität beinhalten. Dadurch soll die Learnability über die Wiederholung der ähnlichen Aufgaben abgebildet werden.

User Satisfaction

Die User Experience und Zufriedenheit der User:innen wird durch einen Fragebogen abgefragt. Es werden zwei verschiedene Arten von Fragen eingebaut. Der Großteil des Fragebogens besteht aus Aussagen über das textgesteuerte Interface, die auf der Likert-Skala bewertet werden müssen. Die Skala wurde basierend auf der Recherche in Kapitel 2.2.2.5 erstellt. Es wurden die folgenden Skalen-Level gewählt:

- Stimme zu
- Stimme eher zu
- Stimme eher nicht zu
- Stimme nicht zu

Basierend auf der Literatur wurden nur vier Level gewählt. Häufig wird empfohlen, eine neutrale Stufe zur Auswahl bereitzustellen. Dies ist jedoch auch umstritten, da ebenso Studien ohne neutrales Level umgesetzt wurden. Während eine neutrale Meinung grundsätzlich valide ist, soll diese Studie eindeutig positive oder negative Ergebnisse zeigen, weswegen kein neutrales Level eingebaut wurde.

Anfangs bestanden Überlegungen, einen vorgefertigten, standardisierten Fragebogen zu verwenden, da hier beispielsweise die Reproduzierbarkeit und Objektivität versichert werden kann. Jedoch konnte kein bestehender Fragebogen identifiziert werden, bei dem die Formulierung sowie die Kombination der Fragen für das eigene Usability Testing geeignet waren. Deshalb wurde der Fragebogen selbst erstellt. Dazu wurden Fragen aus einer Sammlung diverser standardisierter Fragebögen gewählt, die zu den Studienzielen passen. Diese wurden weiters auf die eigene Anwendung umformuliert.

Bei der Formulierung der Fragen sind diverse Aspekte wie die Polarität dieser zu beachten. Da in der Literatur, die in Kapitel 2.2.2.5 vorgestellt wurde, keine negativen Effekte durch eine durchgehend positive oder negative Polarität der Fragen bewiesen werden konnten, wurden alle der Fragen mit einer positiven Neigung formuliert. Dadurch soll vermieden werden, dass die Teilnehmer:innen durch einen Wechsel der Polarität verwirrt werden.

Zusätzlich wurden dem Fragebogen vier offene Fragen hinzugefügt. Diese sollen den Studienteilnehmer:innen die Möglichkeit geben, ihre Gedanken und Erfahrungen mit dem Web-GIS ausführlicher und mit eigenen Worten auszudrücken. Weiters sollen dadurch zusätzliche Informationen, die durch die vorgefertigten Fragen nicht offengelegt werden können, gezeigt werden.

Der vollständige Fragebogen befindet sich im Anhang 10.3.1.2.

5.2. Ablauf der Studieneinheiten

Für die Studie wurden zehn Personen der Firma, die den Anforderungen der Zielgruppe entsprechen, ausgewählt und angeschrieben. Von allen der Personen konnte eine Zusage erreicht werden. Es wurden Termine innerhalb einer Woche angesetzt, wobei für die Teilnehmer:innen an diesen Tagen jeder Zeitpunkt freigelegt wurde. Für jeden der Studiendurchläufe wurde ein einstündiges Meeting angesetzt. Vor dem Meeting wurden eine rechtliche Datenschutzerklärung zur Weiterverwendung der Daten, sowie der Screener den Personen zugesandt.

Anfangs wurde eine kurze Präsentation zur Einführung in die Thematik der Masterarbeit und die Studie selbst gemacht, und ebenso Fragen der Teilnehmer:innen beantwortet. Es wurde erklärt, welche Abfragen gestellt werden können, und welche Elemente diese beinhalten. Ebenso wurde darauf hingewiesen, dass die Aufgabenstellungen nicht kopiert, sondern basierend auf den Tasks die Anfrage selbst formuliert werden sollen. Jedoch wurden möglichst keine Angaben über die Formulierung gemacht, um User:innen nicht zu beeinflussen und „echte“ Ergebnisse zu bekommen, um herauszufinden, wie Anfragen real formuliert und gestellt werden.

Anschließend wurden alle Vorbereitungen zur Durchführung der Studie gemeinsam mit den Teilnehmer:innen gemacht. Dazu wurde die Videoaufnahme mit Einverständnis der Person gestartet und diese ebenfalls gebeten, den Bildschirm zu teilen. Zuletzt wurden den Teilnehmer:innen die Zugangsdaten und Links zu den beiden Anwendungen, sowie ein PDF mit den Aufgabenstellungen übermittelt.

Wie in den vorherigen Kapiteln bereits erwähnt, begann die Hälfte der Personen mit der buttongesteuerten, und die andere Hälfte mit der textgesteuerten Anwendung. Die Personen

wurden grundsätzlich zufällig gewählt, wobei jedoch auf eine ausgeglichene Geschlechterverteilung geachtet wurde.

Die Personen führten anschließend alle Tasks im ersten, und anschließend im zweiten Interface durch. Nebenbei wurde die Zeit gestoppt und aufgezeichnet, während die Zeitwerte jedoch erst durch die Videoauswertung final festgelegt wurden. Nach Erledigung einer Aufgabenstellung wurde gemeinsam durch Anklicken der Karte und der Diagramme die Korrektheit des Ergebnisses überprüft.

Für die Studie wurde ein Zeitplan erstellt, der in Tabelle 7 gezeigt ist. Dieser Zeitplan sollte versichern, dass die Dauer des Online-Meetings nicht eine Stunde übersteigt.

Tabelle 7: Zeitplan der Studie

Minuten	***Abschnitt der Studie***
10	Einführung, Einrichtung, Fragen
23	Tasks Interface 1
23	Task Interface 2
5	Fragen und Verabschiedung

Für die Präsentation am Anfang, sowie die Beantwortung von Fragen und die Einrichtung der Web-GIS wurden 10 Minuten eingeplant. Für die Lösung der Aufgabenstellung in jedem der beiden Web-GIS wurden 23 Minuten eingeplant, die auf den maximalen Zeiten der Tasks basieren. Die Verabschiedung, Danksagung und die Beantwortung von Fragen waren für fünf Minuten angesetzt. Bei einem Testlauf wurde offensichtlich, dass der Zeitplan passend ist und die Studie in der vorgegebenen Zeit durchgeführt werden kann.

Der Fragebogen wurde den Teilnehmer:innen nach der Studie zugeschickt und diese wurden noch im Meeting selbst gebeten, diesen direkt auszufüllen. Dadurch sollte ermöglicht werden, so viel Zeit wie möglich für die Lösung der Tasks selbst einzuplanen, da der Fragebogen keine Anwesenheit der Moderatorin benötigt.

5.3. Auswertung der Studienergebnisse

Im folgenden Kapitel werden sämtliche im Laufe des Usability Testings gesammelte Daten ausgewertet. Auszuwerten sind dabei die Videoaufnahmen aus den Meetings und die Ergebnisse des Fragebogens. Die Interpretation der Ergebnisse folgt später in Kapitel 6.

5.3.1.1. Task Success und Fehlerauswertung

Während der Online-Meetings konnte eine erste Evaluierung des Task Success bereits aufgezeichnet werden. Durch die Analyse der Videoaufnahmen konnten die Fehler konkreter untersucht und eingeordnet werden. Der Task Success wurde basierend auf den definierten Levels of Success analysiert. Abbildung 44 zeigt die Auswertung dieser in einem Diagramm.

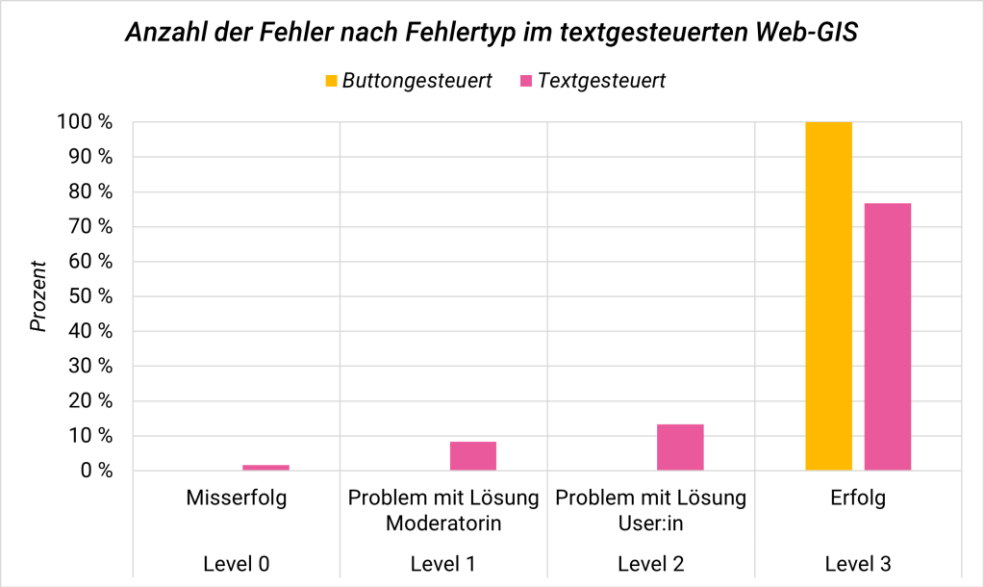

Abbildung 44: Prozent der „Levels of Success“ im Vergleich im textgesteuerten und buttongesteuerten Web-GIS

Insgesamt wurden durch die sechs Tasks pro Anwendung und mit zehn Studienteilnehmer:innen eine Anzahl von 60 Lösungsversuchen ausgewertet. Die Auswertung wird in Prozent vorgenommen, um diese vergleichbarer zu gestalten. Bei der buttongesteuerten Anwendung wurden alle der Aufgabenstellungen erfolgreich durchgeführt. Das textgesteuerte Web-GIS weist eine Erfolgsrate von 77 % auf. Die Erfolgsrate beschreibt in diesem Fall den Prozentsatz an Aufgaben, die ohne Fehler und falsche Angaben gemacht, und als Level 3 eingestuft wurde. Bei der Lösung der Tasks im textgesteuerten Interface sind bei 13 % der Versuche Probleme aufgetreten, die von den Teilnehmer:innen selbst gelöst werden konnten. Bei 8 % wurde eine Hilfe der Moderatorin benötigt. 2 % der Aufgabenstellungen konnten nicht gelöst werden. Tabelle 8 zeigt noch einmal die genaue Anzahl der Fehler pro Fehlertyp und Anwendung.

Tabelle 8: Anzahl der Fehlertypen je „Level of Success“ im buttongesteuerten und textgesteuerten Web-GIS

Level of Success	*Definition*	*Buttongesteuert*	*Textgesteuert*
3	Success	60	46
2	Problem mit Lösung User	0	8
1	Problem mit Lösung Moderatorin	0	5
0	Failure	0	1

Hier ist noch einmal konkreter ersichtlich, wie oft die Fehler tatsächlich aufgetreten sind. Die abgeschlossenen Tasks mit Level 3 liegen beim buttongesteuerten Web-GIS bei 60, die des textgesteuerten bei 46. Nicht erfolgreich abgeschlossen wurde nur ein Versuch nicht.

Bis jetzt wurden alle Fehler in den Leveln des Success, die für diese Arbeit festgelegt wurden, analysiert. Nun werden die Fehler und Probleme konkret aufgeschlüsselt und analysiert, wo die Fehlerquellen lagen. Dies ist in Tabelle 9 dargestellt.

Tabelle 9: Fehlerquellen des textgesteuerten Web-GIS

Fehlerart	***Anzahl***
Ungenaue Angabe (Tippfehler, kein Abstand, …)	5
Komplexe Analyse in zwei Sätzen	3
München dazu	2
Problem von Anwendung	2
Falsche Satzstellung Datum und Zeit	2
Fehlende Angabe	1

Das meist aufgetretene Problem sind zu ungenaue Angaben im textgesteuerten Interface. Dies betrifft im speziellen die Angabe von Datums- und Zeitangaben. Beispiele sind hier einfache Tippfehler, oder das Fehlen eines Worts zwischen Datum und Zeit, was dazu führt dass diese Angaben durch die Named Entity Recognition nicht erkannt werden können.

In drei Fällen ist aufgetreten, dass User:innen die Aufgabenstellungen 5 und 6 lösen wollten, indem sie zwei vollständige Sätze hintereinander schreiben. Dies kann von der Datenprozessierung jedoch nicht erkannt werden. Diese setzt voraus, dass die Anfrage in einem Satz geschrieben wird und beispielsweise die Datumsangabe so nicht doppelt vorkommt. Bestehen zu viele Angaben einer Art, wird eine Fehlermeldung herausgegeben.

Zwei der Teilnehmer:innen haben bei der FCD-Verkehrsanalyse zusätzlich zu der Straße den Zusatz „München“ in Klammern oder durch Abtrennung mit Beistrichen hinzugefügt, um so die Straßen auf die gewollte Stadt einzugrenzen. Dies ist jedoch nicht vorgehsehen, da München als zusätzliche Örtlichkeit erkannt wird und die Prozessierung so eine Fehlermeldung generiert.

Ein weiteres Problem trat zwei Mal auf. Dabei wurde die Anfrage in falscher Satzstellung gestellt. Nicht miteingeschlossen ist hier die Reihenfolge der Datumsangaben und Straßenangaben, da hier alle Stellungen möglich sind. In diesen Fällen wurde jedoch die die Anfrage in der Reihenfolge „1. Datum, 1. Zeit, 2. Datum, 2. Zeit“ gemacht. Dies ist in der Prozessierung nicht verarbeitbar, da alle Angaben einer Art hintereinander geschrieben werden müssen.

In zwei Fällen traten unbekannte Probleme auf. Einmal konnte eine Straße nicht gelesen werden, was in einer Fehlermeldung resultierte. Genauso konnte einmal das Datum nicht erkannt, und kein korrektes Ergebnis erzielt werden. Hier wurde mithilfe der Moderatorin am Problem gearbeitet. Ersteres konnte gelöst werden, das zweite stellt die einzige unerfolgreiche Anfrage aus der vorherigen Tabelle dar.

5.3.1.2. Time on Task Auswertung

Zur Auswertung der Time on Task werden nur die Lösungen von Aufgabenstellungen miteinbezogen, die auch erfolgreich gelöst werden konnten. Dies bedeutet, dass hier eine Durchführung des Task 6 nicht miteinbezogen wird.

Die Zeit wird durch statistische Lagemaße und Streuungsmaße analysiert. Lagemaße beschreiben typische und zentrale Werte einer Verteilung. Das bekannteste Lagemaß ist das arithmetische Mittel, das der Mittelwert einer Verteilung und der Quotient der Summe aller Einzelwerte und der Anzahl der Elemente ist. Für diese Auswertung wird jedoch der Median verwendet. Dieser teilt die Verteilung aller Werte in je 50%. Der Vorteil des Medians gegenüber dem Mittelwert ist, dass er im Gegensatz zu ihm nicht von Ausreißern beeinflusst wird. (vgl. ZWERENZ 2015: 80-97) Dies ist auch in dieser Auswertung von Vorteil, da die Vermutung besteht, dass möglicherweise mehrere Ausreißer vorhanden sind. Weiters werden das Minimum und Maximum analysiert.

Streuungsmaße zeigen die Verteilung der Werte und umschreiben, ob ein Lagemaß typisch für die Verteilung ist. Diese können in Form eines Boxplots dargestellt werden. Dabei werden der Median, sowie das Minimum, Maximum und bestehende Ausreißer gezeigt. Weiters werden das 25%-Quantil und 75%-Quantil dargestellt. Quantile beschreiben Bereiche einer Verteilung, wie beispielsweise die oberen 25%. Dies wäre das 75%-Quartil. (vgl. ZWERENZ 2015: 98-111)

Die Auswertung der Zeit wird hier in zwei Schritten vorgenommen. Anfangs wird die Zeit für die initiale Eingabe ausgewertet. Damit wird die Zeit vom Start der Eingabe bis hin zum ersten Abschicken des Eingabeformulars gemessen. Erst im nächsten Schritt wird die Zeit inklusive der Fehlerbehandlung verglichen. Während die gesamte Zeit inklusive der Fehler von größerer Bedeutung ist, hat auch die Zeit der initialen Eingabe eine Relevanz. Dadurch kann verglichen werden, wie die Zeitauswertungen der beiden Interfaces ohne das Aufkommen von Fehlern aussehen. Deswegen werden beide Zeiten evaluiert.

Initiale Zeit

Als erstes wird die Zeit der initialen Eingabe verglichen. Abbildung 45 stellt diese in einem Boxplot dar.

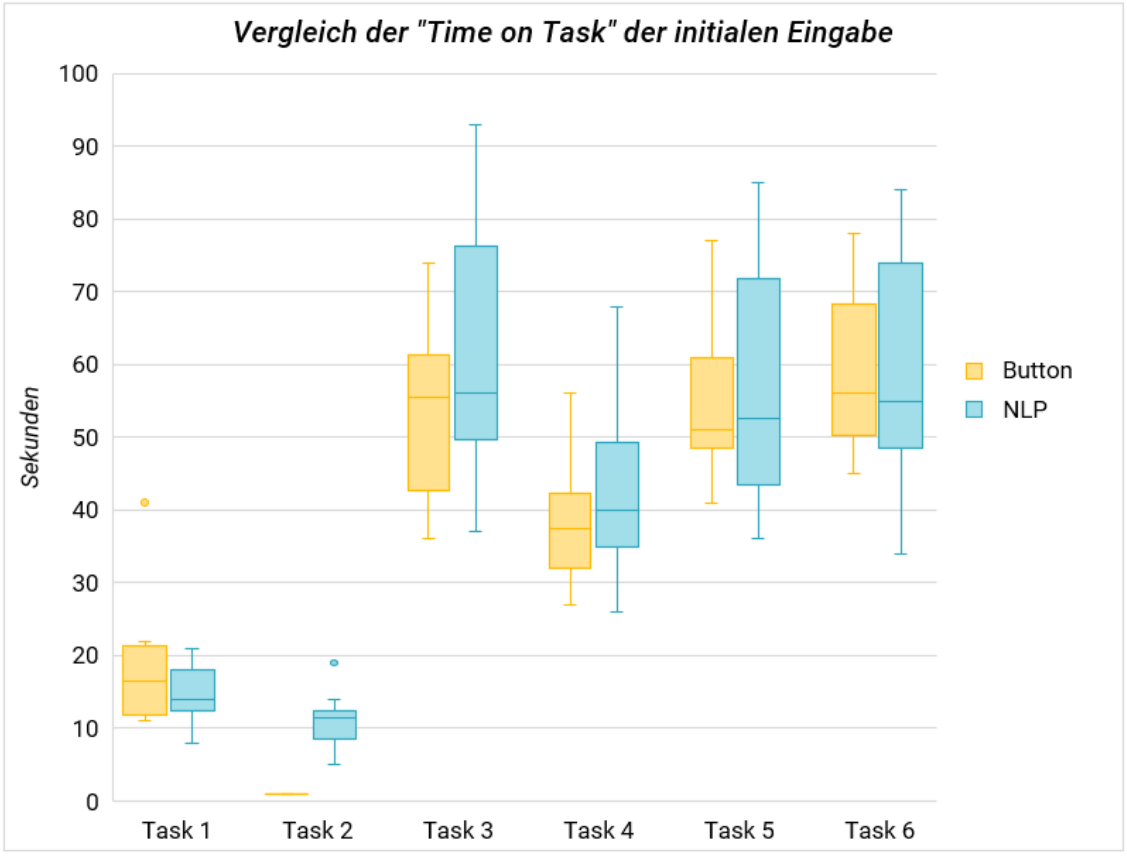

Abbildung 45: Vergleich der „Time on Task“ der initialen Eingabe im buttongesteuerten und textgesteuerten Web-GIS im Boxplot

Der Median wird durch den langen horizontalen Strich jedes Plots dargestellt. Bei Task 1 und 6 wurde die initiale Eingabe im textgesteuerten Web-GIS schneller durchgeführt als im buttongesteuerten. Die anderen Tasks wurden im buttongesteuerten Web-GIS schneller gelöst. Der größte Unterschied zwischen den Zeiten besteht bei Task 2. Die Lösung dieses Tasks in beiden Anwendungen ist nur schwer vergleichbar, da das buttongesteuerte Web-GIS nur das Klicken eines einzigen Buttons erfordert. Zwar ist zu erwähnen, dass dies im textgesteuerten Interface auch nur durch ein Wort wie „Live“ lösbar wäre, dies aber von keinem der User:innen umgesetzt wurde. Deswegen besteht hier ein sehr offensichtlicher Unterschied. Bei der Lösung aller anderer Tasks bestehen nur Unterschiede von wenigen Sekunden.

Das Minimum und Maximum werden durch die kleinen, horizontalen Striche an Anfang und Ende der Plots dargestellt. Bei den Tasks 1, 4, 5 und 6 wurde die minimale Zeit der initialen Eingabe im textgesteuerten Web-GIS erreicht. Ebenso wurde die maximale Zeit bei allen Aufgaben außer Aufgabe 1 in der textgesteuerten Anwendung erreicht.

Bei Task 1 besteht beim buttongesteuerten Interface ein Ausreißer, im textgesteuerten Interface wurde dieser bei Task 2 festgestellt. Gezeigt werden beide je durch einen Punkt.

Abgesehen von Task 1 weißen alle der Quantile, dargestellt durch die Boxen, bei der textgesteuerten Anwendung eine größere Streuung der Werte auf.

Zeit inklusive Fehlerbehandlung

Weiters wurde die Time on Task inklusive der Zeit, die für die Lösung von Problemen oder Fehlern benötigt wurde, analysiert. Hier werden dieselben Metriken angewendet wie bei der initialen Zeit, wie es in Abbildung 46 gezeigt ist.

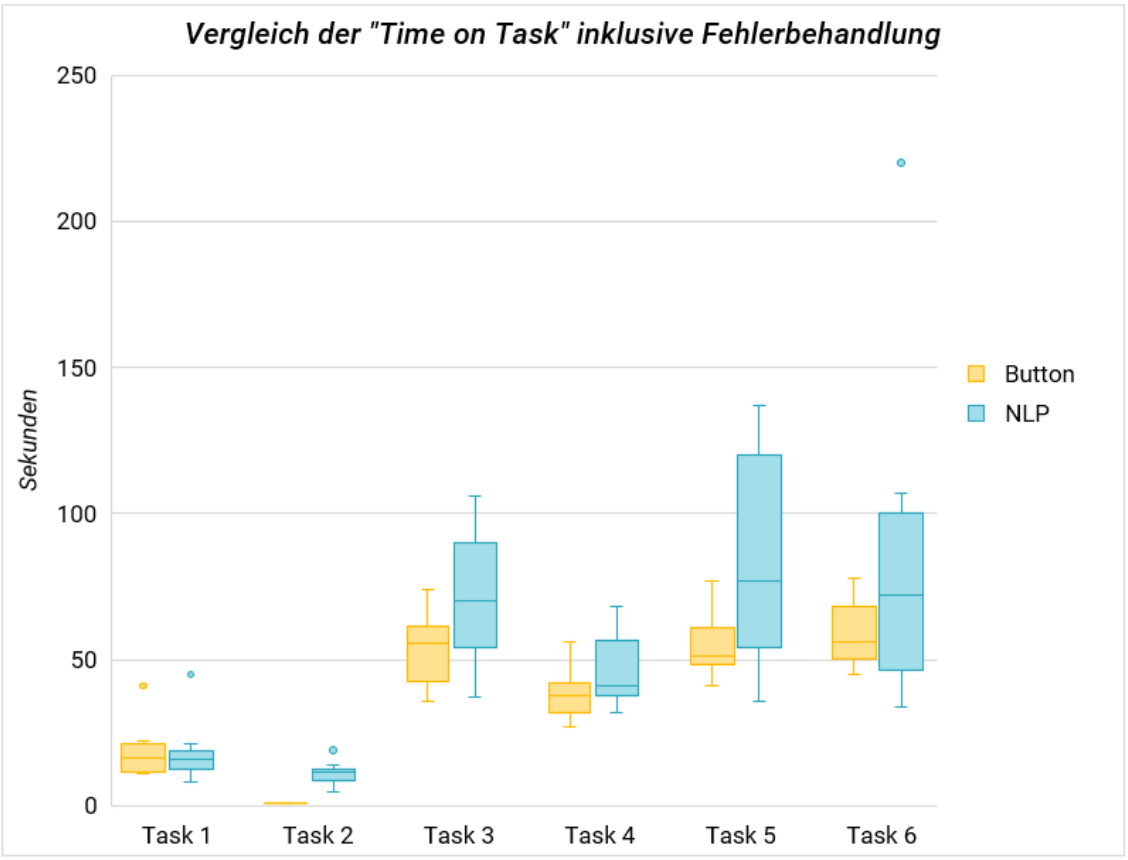

Abbildung 46: Vergleich der „Time on Task“ der Eingabe inklusive Fehlerbehandlung im buttongesteuerten und textgesteuerten Web-GIS im Boxplot

Hier ist der Median bei einigen Aufgaben, abgesehen von der ersten Aufgabe, in der textgesteuerten Anwendung höher als in der buttongesteuerten. Bei Task 4 besteht nur eine kleine Differenz der Zeit.

Das Minimum hat sich hier nicht geändert, da das Einbeziehen der Fehlerbehandlung sich nicht auf dieses auswirkt. Die maximale Zeit ist besonders bei den Tasks 3-6 stark gestiegen, da hier Fehler aufgetreten sind.

Auch die gesamten Werte sind, wie es in den Quantil-Boxen deutlich wird, viel weiter gestreut.

Die in der Analyse der initialen Zeit bereits beschriebenen Ausreißer wiederholen sich auch hier, wobei bei Task 1 in der textgesteuerten Anwendung ein Ausreißer hinzugekommen ist.

5.3.1.3. Learnability Auswertung

Die Learnability wird durch die Time on Task und den Task Success evaluiert. Die ersten beiden Tasks werden hier bei der Berechnung ausgeschlossen, da sie eine andere Aufgabenstellung umfassen, während die Tasks 3-6 eine FCD-Verkehrsanalyse erfordern. Die ersten beiden Tasks umfassen die Analyse einer Route, die beiden weiteren jeweils zwei Routen. Die Komplexität ist dadurch ansteigend. In Kapitel 2.2.2.5 wird basierend auf der Literatur bereits beschrieben, dass mindestens drei Tasks zum Vergleich der Learnability benötigt werden. Hier werden insgesamt vier Tasks verglichen.

Die Metrik wird über den Verlauf der Aufgabenstellungen kontinuierlich verglichen. Abbildung 47 zeigt die Auswertung.

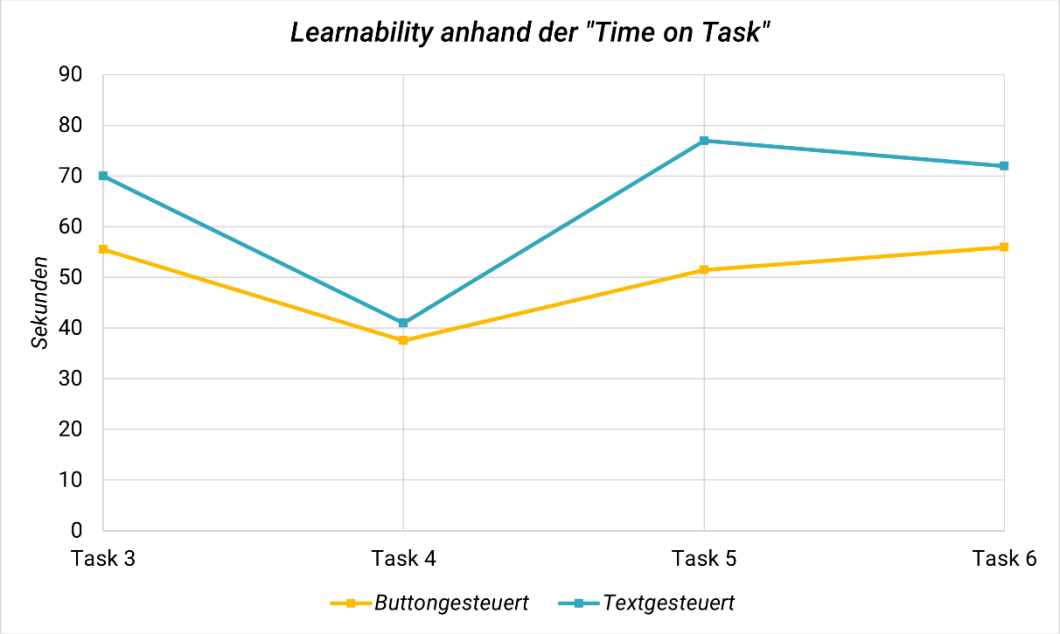

Abbildung 47: Learnability analysiert anhand der „Time on Task“ im textgesteuerten Web-GIS

Die Learnability wird im buttongesteuerten und textgesteuerten Web-GIS verglichen. Der Fokus liegt hier nicht auf den konkreten Zeitwerten, sondern dem Verlauf der Zeit über die Aufgabenstellungen. Zwischen der Aufgabe 3 und 4 ist in beiden Anwendungen ein deutlicher Lerneffekt sichtbar. Bei beiden sinkt die Time on Task. Beim textgesteuerten Interface ist hier durchaus ein noch größerer Effekt sichtbar, da die Zeit bis zu 30 Sekunden sinkt. Beim buttongesteuerten Interface sinkt sie um etwa 20 Sekunden. Zu Task 5 steigt die Time on Task in

beiden Interfaces deutlich, wobei die Steigerung bei der buttongesteuerten Anwendung höher ist. Allerdings steigt sie nur etwas über die Zeit der ersten Aufgabe, obwohl diese um einiges komplexer ist und mehr Informationen enthält, die angegeben werden müssen. Zu Task 6 hin sinkt die Time on Task wiederum im textgesteuerten Web-GIS, während sie im buttongesteuerten leicht ansteigt.

Weiters wird der Verlauf der Fehlerrate über die Tasks 3-6 evaluiert, wie es in Abbildung 48 dargestellt ist.

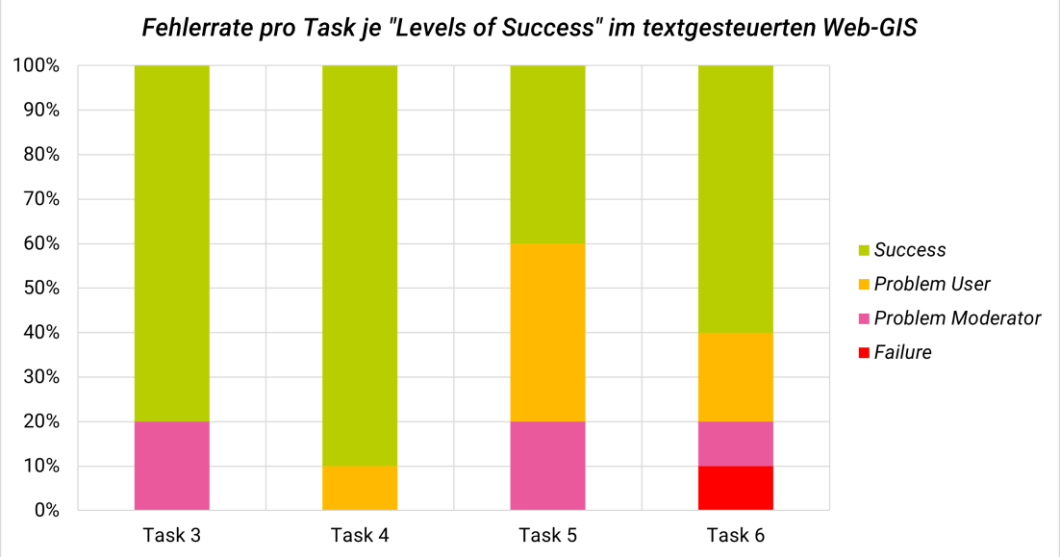

Abbildung 48: Learnability analysiert anhand des „Task Success“ pro Task im textgesteuerten Web-GIS

Bei Task 3 gab es bei 20 % der Versuche Probleme, die nur mithilfe der Moderatorin gelöst werden konnten. Im Vergleich dazu ist bei Task 4 die Fehlerrate niedriger, und die Fehler konnten von den User:innen selbst gelöst werden. Dasselbe Muster ist zwischen Task 5 und Task 6 zu sehen. Task 5 ist die erste der beiden komplexen Analysen. Hier tritt eine wiederum höhere Rate an Fehlern auf. Zu Task 6 sinkt die Fehlerrate, wobei hier ein Versuch überbleibt, der aufgrund von Problemen der Anwendung selbst nicht gelöst werden konnte.

5.3.1.4. Zufriedenheit und User Experience Auswertung

Folgend wird die Zufriedenheit und Erfahrung der Studienteilnehmer:innen mit den Web-GIS ausgewertet. Dies geschieht anhand des Fragebogens.

Fragen mit Skalen

Als erstes werden die Aussagen, die durch die Likert-Skala bewertet werden konnten, ausgewertet, wie es in Abbildung 49 dargestellt ist.

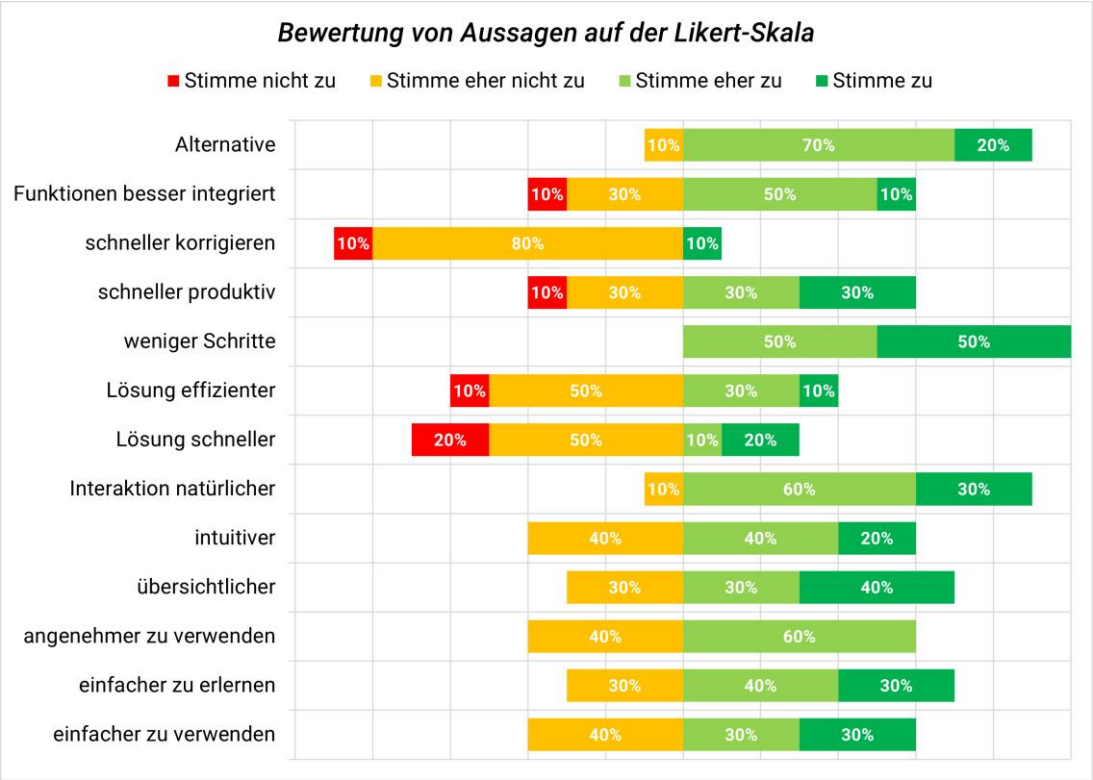

Abbildung 49: Auswertung der Bewertungen von Aussagen zum textgesteuerten Web-GIS auf der Likert-Skala

Auf den ersten Blick ist zu sehen, dass die Tendenz der Bewertungen positiv ist. Die Aussagen wurden tendenziell mehr mit „Stimme eher zu“ und „Stimme zu“ bewertet. Am schlechtesten wurden die Fragen 2, 3, 4, 5 und 6 bewertet. Hier wurde als einziges ebenso mit „Stimme nicht zu“ abgestimmt, das sind insgesamt 5 von 13 Fragen. Die Aussagen, dass die Fehler mit dem textgesteuerten Web-GIS zu korrigieren sind, sowie dass die Funktionen besser integriert sind, man schneller produktiv sein kann, und die Lösung schneller oder effizienter ist, wurden am schlechtesten bewertet. Die beste Bewertung wurde bei der Aussage, dass weniger Schritte zur Lösung nötig sind, gemacht. Ebenso wurden besonders die Aussagen, dass die Interaktion natürlicher und das Interface eine Alternative für die User:innen ist, positiv bewertet.

Offene Fragen

Weiters werden die offenen Fragen ausgewertet. Fragen 1 und 2 wurden in der Auswertung zusammengefasst, da teils ähnliche bzw. zusammenpassenden Antworten erhalten wurden, aufgrund der ebenso ähnlichen Fragestellung. Die Studienteilnehmer:innen mussten hier angeben, ob ihnen das textgesteuerte Web-GIS gefällt oder nicht, und was die Gründe für ihre Meinung sind.

Es wurde versucht, je positive und negative Erwähnungen zusammenzufassen, und statistisch auszuwerten. Hier wurden teils Erwähnungen umformuliert. Wurde beispielsweise erwähnt, dass das buttongesteuerte Interface schneller ist, wurde dies umgeformt darauf, dass das textgesteuerte

Interface langsamer ist, um eine einheitliche Auswertung zu ermöglichen. Weiters wurden hier nur grundsätzliche Erwähnungen der Stichwörter zusammengefasst, eine differenziertere Darstellung relevanter Aussagen erfolgt später.

In Abbildung 50 sind die positiven Stichwörter, die im Zuge der offenen Fragen 1 und 2 erwähnt wurden, statistisch zusammengefasst.

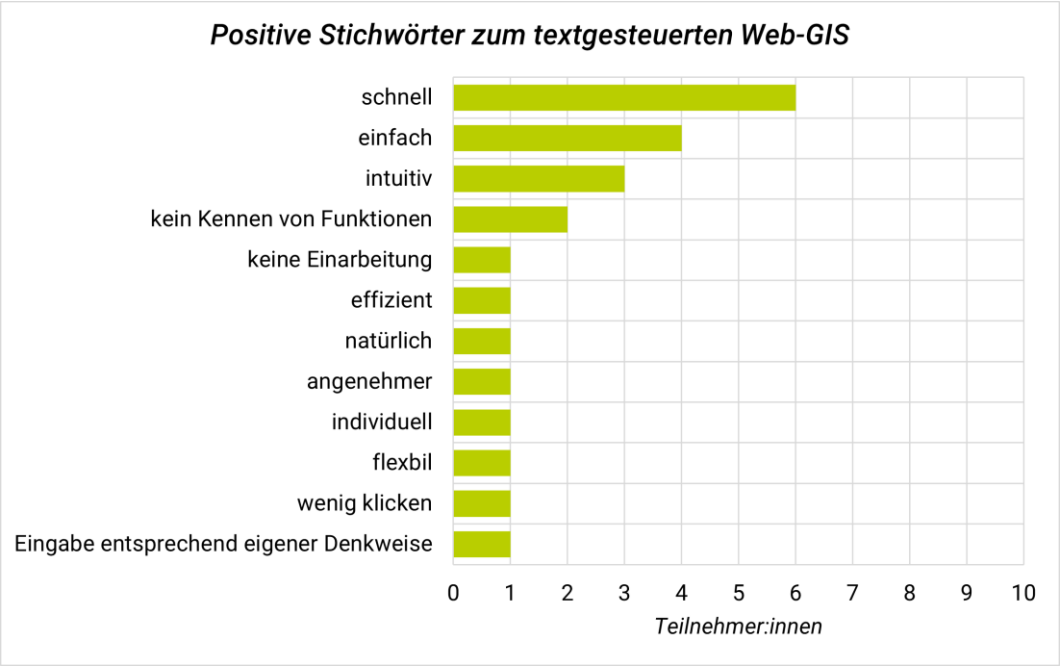

Abbildung 50: Positive Stichwörter der Studienteilnehmer:innen zum textgesteuerten Web-GIS

Das meisterwähnte Stichwort ist „schnell“, im Kontext, dass das textgesteuerte Web-GIS schnell oder schneller als das buttongesteuerte Web-GIS in der Anwendung ist. Ebenso wurde der Ansatz von drei und vier Personen als einfach und intuitiv bezeichnet. Zweimal wurde positiv hervorgehoben, dass die Funktionen des Web-GIS nicht bekannt sein müssen, um dieses verwenden zu können. Je einmal wurden weiter Stichworte wie effizient, natürlich und angenehm erwähnt.

Abbildung 51 zeigt die negativen Stichwörter, die über das textgesteuerten Web-GIS geschrieben wurden.

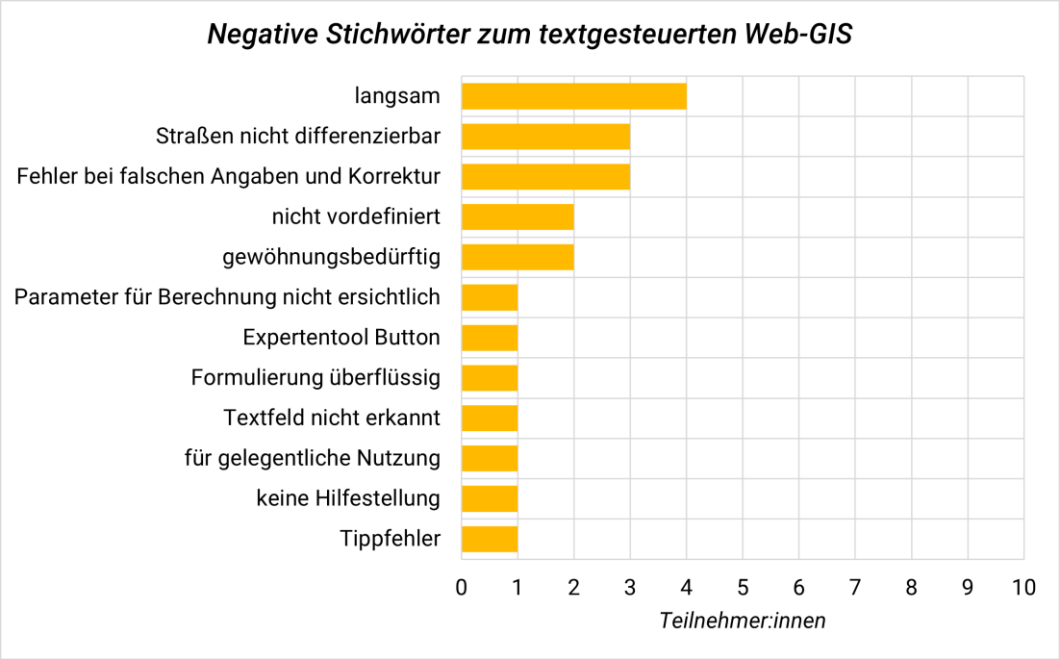

Abbildung 51: Negative Stichwörter der Studienteilnehmer:innen zum textgesteuerten Web-GIS

Während bei den positiven Worten mit sechs Erwähnungen die textgesteuerte Anwendung als schneller erwähnt wurde, wurde ebenso viermal erwähnt, dass diese langsamer ist. Weiters wurde von je drei Personen angemerkt, dass die Straßen in der textgesteuerten Anwendung nicht differenzierbar sind, und dass Fehler entstehen. Ebenso wurde erwähnt, dass das Interface unklarer und gewöhnungsbedürftig ist. Von zwei Personen wurde ausgesagt, dass das Web-GIS für sie nicht als Experten-Tool, sondern für die gelegentliche Nutzung geeignet ist. Weiterhin wurde unteranderem einmal erwähnt, dass Tippfehler entstehen, keine Hilfestellung gegeben ist, und die Formulierung von Texten überflüssig ist. Weitere Aussagen sind der Abbildung zu entnehmen.

Es ist ersichtlich, dass verschiedenste Meinungen über das textgesteuerte Web-GIS bestehen, wobei diese teils übereinstimmen, teils aber auch gegensätzlich zueinander sind. Weiteres dazu in Kapitel 6.

Bei Frage 4 wurde den Studienteilnehmer:innen die Möglichkeit gegeben, abschließend noch etwas zu sagen oder hinzuzufügen.

Alle der Antworten auf den Fragebogen sind im Anhang 10.3.2.8 zu finden.

5.3.1.5. Auswertung der Texteingaben

Zusätzlich zu den Metriken wurden die Texteingaben und Arten der Texteingabe analysiert. Hier wird konkret auf interessante und unterschiedliche Formen der Eingabe näher eingegangen.

Wie bereits erwähnt, wurden die Teilnehmer:innen gebeten, die Formulierungen nicht direkt von der Aufgabenstellung zu übernehmen, sondern basierend darauf die Anfragen eigen und frei zu formulieren. Dabei hielten sich einige der Personen mehr, andere weniger an die Formulierungen.

Die Formulierungen unterscheiden sich je nach User:in vor allem dadurch, ob die Anfrage als vollständiger und grammatikalisch korrekter Satz formuliert ist, oder nur die wichtigsten Stichpunkte erwähnt werden.

Folgend werden die Eingaben von vier Studienteilnehmer:innen bei Task 4 verglichen. In Abbildung 52 sind diese dargestellt. Alle der gezeigten Anfragen stellen erfolgreiche Lösungen der Aufgabenstellung dar. Es wurde weiters versucht, „extreme" Beispiele bzw. eindeutig als kurze oder lange identifizierbare Anfragen zu verwenden.

Abbildung 52: Ausgewählte Anfragen der Studienteilnehmer:innen an das textgesteuerte Web-GIS

Teilnehmer:in 9 und Teilnehmer:in 4 haben beide komplett ausformulierte Sätze als Anfragen gestellt. Es wurden hier sämtliche Füllwörter, die für einen vollständigen deutschen Satz nötig sind, eingefügt. Es ist offensichtlich, dass die Personen auf eine grammatikalisch korrekte Eingabe geachtet haben. Die Personen haben ebenso auf Groß- und Kleinschreibung geachtet, und haben alle Informationen ganz ausgeschrieben. Auch wurde versucht, Anfragen ohne Tippfehler zu stellen.

Im Gegensatz dazu haben Teilnehmer:in 1 und 3 die Anfragen anders gestaltet. Sie haben weder auf die Grammatik des Satzes noch die Groß- und Kleinschreibung geachtet. Es wurden nur die wichtigsten Stichwörter und Informationen eingefügt, und auf jegliche Füllwörter, die für den Informationsinhalt von Bedeutung sind, verzichtet. Auch Tippfehler wurden in Kauf genommen. Weiters wurde sogar versucht, die Straßennamen abzukürzen, um möglichst wenig Aufwand zu betreiben. Dazu wurde das Wort „Straße" teilweise als „str" abgekürzt. Hier konnte trotz dessen ein korrektes Ergebnis erzielt werden.

Es wurden weiters die Zeiten der vorher gezeigten Anfragen herausgesucht, um zu vergleichen, ob Personen, die einen vollständigen Satz eingeben langsamer sind, und Personen, die möglichst kurze Sätze schreiben, schneller. Die Auswertung der Time on Task von Task 4 der Teilnehmer:innen 1, 3, 4 und 9 ist in Tabelle 10 dargestellt.

Tabelle 10: Vergleich der „Time on Task" von Task 4 der vier hervorgehobenen Teilnehmer:innen und Median

Teilnehmer:in	***Sekunden***
1	26
3	36
4	41
9	47
Median	38,5

Teilnehmer:innen 1 und 3 haben die Anfragen möglichst kurz, Teilnehmer:innen 4 und 9 möglichst vollständig geschrieben. Aus der Tabelle ist schwer feststellbar, ob tatsächlich ein direkter Einfluss zwischen der Länge des Satzes und der Time on Task besteht. Es ist jedoch erkennbar, dass zumindest kein großer positiver Einfluss sichtbar wird. Dies spiegelt sich auch bei den anderen Tasks und Teilnehmer:innen wieder.

Teilweise unterscheidet sich jedoch auch nicht nur die Formulierung des Satzes voneinander, sondern auch die Art der Eingabe. Beispielsweise wurden von drei Personen nur die relevanten Informationen wie Straßen, Datum und Zeit ausgetauscht, während die restliche Anfrage stehen gelassen wurde. Die im Gesamten schnellste Person hat diese Methodik verwendet.

6. Diskussion

Im Rahmen dieser Masterarbeit wurden zwei Deep Learning Modelle ausgewählt und trainiert. Sie wurden in ein Web-GIS implementiert, genauso wie ein Skript zur Prozessierung der Input-Daten entwickelt wurde. Weiters wurde eine Studie konzipiert und durchgeführt, um die Funktionen des Web-GIS in der Interaktion mit User:innen zu evaluieren und herauszufinden, ob das textgesteuerte Web-GIS für User:innen eine Alternative zu einem buttongesteuerten Web-GIS darstellt.

Alle Vorgehensweisen wurden auf einer umfassenden Literaturrecherche gewählt. Dabei wurden die Grundlagen der Geoinformatik und von Web-GIS, sowie der Human Computer Interaction dargestellt. Weiters wurde auf die theoretischen Konzepte des Natural Language Processings, besonders in der Anwendung von räumlichen Informationen beschrieben, sowie die praktische Umsetzung mit Deep Learning und Transformers.

Abbildung 53 zeigt den Workflow der Erstellung dieser Arbeit.

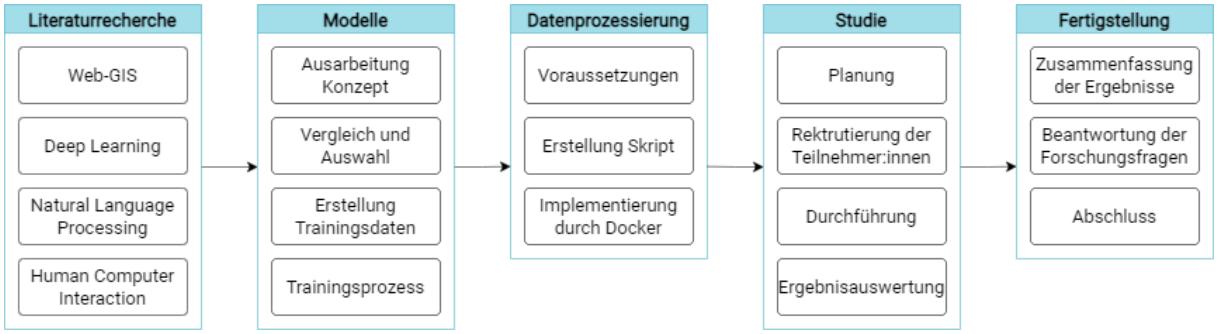

Abbildung 53: Ablauf Masterarbeit

Im folgenden Kapitel werden die Ergebnisse zusammengefasst und interpretiert. Weiters werden Herausforderungen und Limitierungen in der Umsetzung der Arbeit und der Technik erläutert, sowie ein Ausblick für zukünftige Forschungen und Entwicklungen gegeben wird.

6.1. Zusammenfassung und Interpretation der Ergebnisse

Folgend werden alle Ergebnisse, die durch die Umsetzung dieser Arbeit entstanden sind zusammengefasst und interpretiert, um die Arbeitsfragen und Forschungsfragen, die in Kapitel 0 definiert wurden zu beantworten.

Die erste Arbeitsfrage lautet:

- „Was ist der aktuelle Forschungsstand im Natural Language Processing, besonders im Zusammenhang mit räumlichen textbasierten Daten, und wo bestehen Herausforderungen und Mängel in der Technologie?“

Diese Frage konnte durch eine umfassende Literaturrecherche beantwortet werden, die in Kapitel 2.3 vorzufinden ist.

Natural Language Processing beschreibt die Verarbeitung und Interpretation von Text oder Gesprochenem mittels Computer. Während die Anfänge bis in die 1950er zurückreichen, ist NLP heute Teil des alltäglichen Lebens. Von Mail-Programmen bis hin zu Online-Übersetzern basiert eine Vielzahl von Anwendungen auf dieser Methode. NLP ist ein interdisziplinäres Feld, das in den Computer Sciences und der Linguistik verortet werden kann, während die künstliche Intelligenz den methodischen Rahmen bildet. (vgl. ZHANG/TENG 2021: 3)

Der Prozess des Natural Language Processings kann in insgesamt fünf Schritte unterteilt werden. In der Text-Prozessierung wird ein unstrukturierter Input-Text verarbeitet und in strukturierte, verarbeitbare Einheiten wie Sätze und Wörter unterteilt. In der lexikalischen Analyse werden Wörter auf ihre Grammatik und Struktur hin untersucht. Das syntaktische Parsing ermöglicht weiters die Analyse der Struktur ganzer Sätze. Sind diese Schritte abgeschlossen kann versucht werden, eine tiefere Bedeutung eines Textes zu erörtern. Dies erfolgt in der semantischen und der pragmatischen Analyse. (vgl. DALE 2010: 5)

Es bestehen unterschiedliche Aufgaben, die durch Natural Language Processing gelöst werden können. Zwei der bekanntesten Tasks sind die Text Klassifizierung und die Named Entity Recognition. Bei ersterer wird einem Satz, basierend auf dessen Inhalt, eine Klasse zugewiesen. Die Named Entity Recognition geht einen Schritt weiter. Dabei werden aus einem unstrukturierten Input-Text strukturierte Informationen extrahiert. Es werden allen Wörtern, die bestimmten Entitäten wie Ort, Datum oder Zeit angehören, die passende Klasse zugewiesen. (vgl. LAURIOLA ET AL. 2022: 444; ZHANG/ ZHIYANG 2021: 14)

Natural Language Processing kann auch für die Prozessierung von räumlichen Informationen angewendet werden. Dadurch können unteranderem geographische Phänomene durch die Extraktion von diesen räumlichen Informationen besser verstanden werden. Die Kombination der beiden Felder hat zu vielen wertvollen Anwendungen beigetragen. (vgl. MCKENZIE/ ADAMS 2021: 1f)

Laut LAMPOLTSHAMMER/HEISTRACHER 2012: 82 liegen die primären Forschungsfelder in der Verarbeitung von textlichen räumlichen Informationen im Geoparsing und der Integration von Natural Language Processing in GIS und Web-GIS.

Das Geoparsing kann als die Extraktion von räumlichen Informationen aus unstrukturiertem Text, und die Zuweisung von Koordinaten zu diesen definiert werden. Es kann aufgrund dieser Definition auch zur Named Entity Recognition gezählt werden. Der Prozess ist in zwei Schritte zu teilen:

- Geotagging
- Geocoding

(vgl. GRITTA ET AL. 2017: 604)

Im Geotagging werden die Informationen extrahiert, während ihnen im Geocoding die korrekten Koordinaten zugewiesen werden. (vgl. ebd.)

Das Geoparsing kann grundsätzlich auf drei verschiedene Arten umgesetzt werden:

- Gazetteer-Lookup
- Regelbasiert
- Machine Learning

(vgl. LEIDNER/LIEBERMAN 2011: 7)

Beim Gazetteer-Lookup wird basierend auf einem Wörterbuch mit Toponymen (Gazetteer) nach den enthaltenen Wörtern in Texten gesucht. Beim regelbasierten Ansatz werden die Toponyme aus dem Satz nach definierten Regeln extrahiert. Der dritte Ansatz wird mittels Machine Learning durchgeführt und gleicht der Named Entity Recognition. Dieser ist im Vergleich zu den anderen Ansätzen flexibler und ist nicht für Problematiken wie die Aktualität von Toponymen anfällig, wie es ein Gazetter ist. (vgl. ebd.)

Beim Geoparsing bestehen auch heute noch zahlreiche Herausforderungen und Probleme. Diese sind nicht nur bei der Verarbeitung von räumlichen, textuellen Informationen anwendbar, sondern betreffen teils auch Natural Language Processing im generellen Kontext. Eines der größten offenen Probleme ist hier die Mehrdeutigkeit von Toponymen, die sich auf mehrere Arten auswirkt. Zum einen müssen Toponyme im Satz korrekt identifiziert werden. Beispielsweise muss die Stadt Paris von dem gleichklingenden Namen abgegrenzt werden. Wurde dies korrekt erkannt, muss das Toponym zum anderen auch von weiteren gleichnamigen Städten differenziert werden. ((vgl. LEIDNER/LIEBERMAN 2011: 5f; vgl. MCKENZIE/ADAMS 2021: 4)

Eine weitere Problematik ist die Analyse der Semantik und Pragmatik von Toponymen. Es stellt eine Herausforderung dar, tiefere Bedeutungen von Toponymen zu extrahieren und zu verarbeiten. Dies betrifft beispielsweise die Extraktion und effiziente Verarbeitung von Himmelsrichtungen und Distanzen. Grundsätzlich ist dies bis zu einem gewissen Grad schon möglich, jedoch müssen hier noch komplexere und potentere Geoparser entwickelt werden. (vgl. GRITTA ET AL. 2017: 621f)

Beide erwähnten Punkte beschreiben die momentan größten Herausforderungen und Mängel für moderne Geoparsing-Systeme. Auch, wenn schon viele Fortschritte gemacht wurden, besteht immer noch ein hoher Bedarf an Forschung und Weiterentwicklung.

Wie bereits erwähnt ist ein weiteres der aktiven Forschungsfelder die Steuerung von Geoinformationssystemen durch Text-Input. Hier ist das Ziel, eine Alternative zu traditionellen buttongesteuerten GIS und Web-GIS zu finden.

CALÌ ET AL. 2011 hat eine Anwendung, die einem Chatbot ähnelt, umgesetzt, das Antworten auf Fragen zu räumlichen Daten beantwortet. In einer Studie, in der die Anwendung mit ArcGIS verglichen wird, werden positive Ergebnisse in verschiedenen Metriken erzielt. Besonders bei der Time on Task konnten Aufgaben teils doppelt so schnell gelöst werden, wie im Desktop-GIS.

SETLUR ET AL. 2016 hat beispielsweise ein textgesteuertes Web-GIS entwickelt, das die visuelle Analyse von räumlichen Daten ermöglicht. Es wurde dabei ein dualer Ansatz gewählt, bei dem Anfragen mittels Textfeldes gestellt, und durch Buttons nachträglich verändert werden können. Es wird hier besonders hervorgehoben, dass solche Interfaces ohne weiteres Wissen über die analytischen Funktionen der Anwendung verwendet werden können. Das Web-GIS fand bei Studienteilnehmer:innen großen Anklang, und zeigte in verschiedenen Metriken der Usability positive Ergebnisse.

Laut SETLUR ET AL. 2016 liegt eine der größten Herausforderungen hier darin, Systeme zu schaffen, die eine große Menge an unterschiedlichen Anfragen verarbeiten kann. Generell sind viele der Anwendungen vor allem Prototypen, ein wirkliches Produktivsystem fehlt noch. Ein weiterer Schritt ist auch die Ausweitung der unterschiedlichen Visualisierungen. Der Fokus liegt meist nur auf einzelnen Art der Visualisierung, wie Karten oder Diagrammen, während nur wenige kombinierte und komplexere Darstellung bestehen. Weiters ist auch für die Entwicklung von textgesteuerten Interfaces für Web-GIS die Entwicklung besserer Geoparser von Bedeutung, da sie die bereits genannte nötige Entwicklung komplexerer Systeme unterstützt und ermöglicht.

Die zweite Arbeitsfrage ist die folgende:

- „Welche bestehenden Deep Learning Modelle eignen sich zur Verarbeitung von textbasierten Informationen mit Raumbezug in einem Web-GIS, bezogen auf den definierten Use-Case?

Wie bereits erwähnt ist eine der beliebtesten Methoden der Umsetzung des Natural Language Processing das Deep Learning. Hier bestehen verschiedenste Modelle bzw. Algorithmen, die trainiert werden können. Wie HIRSCHLE 2022 und LAURIOLA ET AL. 2022 beschreiben ermöglichen diese Modelle das Anlernen von komplexen Beziehungen und können durch große Mengen von Daten trainiert werden. Dies ist auch ihren komplexen Lernarchitekturen zu verdanken. Auch können sich diese Modelle sehr gut an die Trainingsdaten anpassen. So können weit bessere Performances erreicht werden als durch andere Methoden. (vgl. HIRSCHLE 2022: 57; LAURIOLA ET AL. 2022: 443)

Es bestehen verschiedene Arten von Deep Learning Modellen. Zwei der bekanntesten sind die rekurrenten und konvolutionalen Netze, wie sie von HIRSCHLE 2022 erwähnt werden, die beide für die Verarbeitung von Textdaten geeignet sind und zufriedenstellende Performances bringen. (vgl. HIRSCHLE 2022: 93-125)

Seit wenigen Jahren gibt es jedoch eine neue Art von Modellen, die vortrainierten Transformer Modelle. Diese bringen im Vergleich zu den bisher bekannten neuronalen Netzen einige Vorteile mit sich. HIRSCHLE 2022 fasst diese zusammen. Einer der größten Vorteile ist, dass diese Modelle von Firmen mit hohen Ressourcen auf universelle Sprachen angelernt werden. Die Modelle werden folglich bereitgestellt und können von Anwender:innen mit verhältnismäßig wenig Zeit- und Kostenaufwand auf eigene Aufgaben angelernt werden. Das Verständnis der universellen Sprache behalten die Modelle dabei und verfügen deswegen über sehr potente Performances. Die Eigenschaften der Transformer-Modelle waren für diese Masterarbeit von großem Vorteil, da sie die Möglichkeit einer guten Performance bei der eigenen Aufgabenstellung bieten. (vgl. HIRSCHLE 2022: 207f)

In Kapitel 4 wurde deswegen eine nähere Auswahl aus fünf bestehenden vortrainierten Transformer-Modellen, die je für die deutsche Sprache oder multilinguales Lernen geeignet sind, erstellt. Aus diesen wurde das Modell GottBERT aufgrund seiner hervorragenden Performance in den in der Arbeit benötigten NLP- Tasks gewählt wurde. Das Modell basiert auf der RoBERTa-Architektur und ist rein deutschsprachig trainiert.

Um die Textverarbeitung im Web-GIS zu ermöglichen, wird ein Modell für die Text Klassifizierung und die Named Entity Recognition benötigt. Ersteres analysiert die Art der Anfrage, wobei zwischen den Klassen „Monitoring Live“, „Monitoring Historisch“ und „Analyse“ unterschieden werden kann. Basierend auf diesen Klassen wird ein passendes JSON-File zur Datenbankabfrage ausgewählt. Dadurch wird wiederum die korrekte Verarbeitung durch die NER ermöglicht, um die relevanten Informationen von Straßen, Zeit und Datum zu extrahieren. Das GottBERT-Modell ist für beide dieser Aufgaben geeignet.

Für das Training der Modelle wurde je ein Datensatz eigens erstellt. Für das Modell der Text Klassifizierung umfasst dieser 45 000 Sätze, die möglichst divers, sowie grammatikalisch korrekt randomisiert erstellt wurde. Für die Named Entity Recognition wurden die relevanten Entitäten in den erstellten Sätzen gelabelt. Da hier jedoch eine zu kleine und homogene Menge an Daten erreicht wurde, wurde der Corpus von „smartdata“ verwendet, der viele verschiedene und auch reale Daten für die Entitäten Straßen, Zeit und Datum aufweist.

Die Modelle wurden in mehreren Iterationen trainiert, und verschiedene Kombinationen von Hyperparametern getestet. Anfangs wurde die Suche nach den Parametern durch eine Random Search durchgeführt, schlussendlich wurde die Suche jedoch manuell umgesetzt, da hier bessere Ergebnisse erzielt werden konnten. Die besten Parameter für das Modell für die Text Klassifizierung sind die folgenden:

- Optimizer: Adam
- Verlustfunktion: Sparse Categorical Entropy
- Lernrate: 1e-5
- Batch Size: 16
- Epochen: 4

Es wurden sehr positive Ergebnisse in allen Metriken im Anlernprozess dieses Modells erzielt. Für alle Klassen konnte hier eine Accuracy, Precision, Recall und F1-Score von Eins erreicht werden. Dabei entstand der Verdacht auf eine Überanpassung des Modells auf die Daten, wie sie in Kapitel 2.5.3.5 beschrieben wurde. Um dies auszuschließen, wurde eine Data Augmentation durchgeführt. Diese zeigte jedoch keine positiven Effekte. Der Trainingsdatensatz wurde hier scheinbar schnell zu inhomogen, sodass das Modell wiederum keine generalisierbaren Eigenschaften erlernen konnte. Auch durch eine Literaturrecherche konnten keine Hinweise auf die Überanpassung gefunden werden. Der Grund für die hohen Werte liegt vermutlich darin, dass die Input-Sätze, die zum Training des Modells verwendet wurden, sehr homogen sind, auch wenn sie über verschiedene Formulierungen und Satzstellungen verfügen. Trotz der homogenen Daten hat das Modell jedoch

generalisierbare Eigenschaften der Daten gelernt. Wie in Kapitel 5.3.1.5 gezeigt wurde, haben die Studienteilnehmer:innen teils Anfragen gestellt, die sich stark von der Form der Trainingsdaten unterschieden hat. Auch diese wurden korrekt erkannt. So konnte, trotz homogenen Daten, die basierend auf den eigenen Einschätzung erstellt wurden ein Modell trainiert werden, dass für unterschiedlichste Arten an Anfragen funktioniert. Deswegen wurde das Modell mit den ausgewählten Parametern trainiert und auf den Hugging Face Hub hochgeladen.

Weiters wurde das Modell für die Named Entity Recognition trainiert. Hier war der Anlernprozess komplexer als beim vorherigen Modell. Erst wurde das Training mit den eigens erstellten Daten versucht. Trotz zufriedenstellender Ergebnisse wurde klar, dass manche der Klassen, wie Intervalle und Kennwerte zu klein für das Erlernen von generalisierbaren Eigenschaften waren. Es konnten beispielsweise nur wenige Synonyme für die Kennwerte gefunden werden. Bei kleinen Abweichungen könnten diese vom Modell nicht mehr erkannt werden. Auch ist das Training eines DL-Modells für so wenige Daten in einer Klasse nicht mehr sinnvoll, da hier möglicherweise ein starrer Ansatz besser funktionieren könnte.

Deshalb wurde das finale Modell mit den Trainingsdaten von „smartdata“ angelernt. Es wurden die folgenden Hyperparameter-Einstellungen verwendet:

- Optimizer: Adam
- Verlustfunktion: Sparse Categorical Entropy
- Lernrate: 1e-5
- Batch Size: 16
- Epochen: 5

Das Modell für die Named Entity Recognition weist im Gegenteil dazu eine auf den ersten Blick eine nicht zufriedenstellende Performance auf. Bei der Accuracy konnten grundsätzlich nur Werte um 0,4 erreicht werden. Bei der Berechnung des Classification Reports, der mit den Testdaten berechnet wird, konnten jedoch positivere Ergebnisse gezeigt werden. Hier liegt die Accuracy über alle Klassen bei über 0,9. Bei den verwendeten Klassen Date, Time und Location_Street konnten in allen Metriken Werte über 0,6, und Großteils von 0,7-0,9 erreicht werden. Bei einem Testing des Modells konnten hier gute Ergebnisse erzielt werden, weswegen dieses ebenso exportiert und bereitgestellt wurde.

Beide Modelle wurden in eine Anwendung implementiert. Diese wurde in Kapitel 3 genauer beschrieben, die Implementierung der Modelle sowie die Prozessierung der Input-Daten werden in Kapitel 4.4 detailliert erläutert. Die Anwendung ist ein Web-GIS, das von der Firma Trafficon

im Rahmen des Projektes TEMPUS entwickelt wurde. Das Web-GIS ermöglicht die Visualisierung und Analyse von Verkehrsdaten in München. Das Tool ist im speziellen für Experten im Verkehrsmanagement und der Verkehrsplanung geeignet. Das Web-GIS ermöglicht die Analyse von Live-Verkehrsdaten, der Verkehrssituation in der Vergangenheit, sowie die Analyse verschiedener Kennwerte auf einer oder mehreren Routen. Die Ergebnisse werden in Karten und Diagrammen dargestellt. Die Modelle werden je vom Hugging Face Hub heruntergeladen. Die Input-Daten der User:innen werden gesäubert, prozessiert und schlussendlich in ein JSON-File gefüllt, um somit eine Datenbankabfrage zu ermöglichen.

Die dritte Arbeitsfrage ist die folgende:

- „Welche etablierten Metriken der Human Computer Interaction eignen sich zur Evaluierung der Ergebnisse des Usability Testing von textbasierten Anfragen in einem Web-GIS?“

In der Literaturrecherche in Kapitel 2.2 wurden unterschiedlichste Metriken der Human Computer Interaction gefunden, die anschließend in einer Studie praktisch angewendet und angepasst wurden. In den Studien von SETLUR ET AL. 2016 und CALÌ ET AL. 2011 wurden ebenso ähnliche Metriken verwendet.

Diese Metriken sollen laut DE BLEECKER/OKOROJI 2018 grundsätzlich die Usability einer Anwendung evaluieren, diedie folgenden Aspekte miteinschließt:

- Effektivität
- Effizienz
- Zufriedenheit

Dabei geht es bei den ersten beiden Stichwörtern primär um objektive Faktoren wie die Geschwindigkeit und Korrektheit der Lösung von Aufgabenstellungen, während die Zufriedenheit subjektive Faktoren, wie die Erfahrung, die User:innen bei der Verwendung einer Anwendung haben, betrifft. (vgl. BARNUM 2011: 11f; DE BLEECKER/OKOROJI 2018: 5f)

Um diese drei Aspekte zu überprüfen, wurden die folgenden Metriken der Human Computer Interaction ausgewählt:

- Task Success
- Time on Task
- Learnability
- User Experience/Zufriedenheit

Laut TULLIS/BILL 2013 ist der Task Success eine der beliebtesten Metriken, da er auf verschiedenste Softwares, wie auch ein Web-GIS anwendbar ist. Dadurch wird die Korrektheit der Lösung von Aufgabenstellungen überprüft. Für die praktische Umsetzung können unterschiedliche Levels of Success festgelegt werden, die beschreiben, ob ein Task erfolgreich gelöst wurde, und ob dies beispielsweise mit oder ohne Hilfe durch eine Moderation erreicht wurde. (vgl. TULLIS/BILL 2013: 65-74)

Für diese Arbeit wurden vier Levels of Success festgelegt:

- Misserfolg
- Problem mit Lösung Moderatorin
- Problem mit Lösung User:in
- Erfolg

Diese wurden durch eine Auswertung aller Versuche der Lösung der Aufgabenstellungen ausgewertet.

Weiters kann die Metrik Time on Task evaluiert werden. Evaluiert wird hierbei, wie lange die Verwendung einer Anwendung dauert, bis User:innen das gewünschte Ergebnis erhalten. Die Zeit wird von Anfang bis Ende einer Aufgabenstellung gemessen, wobei hierbei der Start- und Endpunkt dafür definiert werden muss. Dies wurde durch das Stoppen der Zeit basierend auf den Videoaufzeichnungen ausgewertet. (vgl. TULLIS/BILL 2013: 74ff)

Die dritte Metrik, mit der die Usability eines Web-GIS gemessen wird, ist die Learnability. Diese beschreibt, wie lange es dauert, bis Personen in einer Anwendung produktiv sein können. Eine hohe Learnability zeigt, dass die Anwendung schnell und einfach erlernbar und verwendbar ist. Um die Learnability zu messen können unterschiedliche Metriken wie der Task Success und die Time on Task verwendet werden. Dabei wird der Verlauf dieser über mehrere Aufgaben hin beobachtet. Entwickeln sich diese Werte positiv, spricht das für eine hohe Learnability, und die Anwendung ist schnell erlernbar. In dieser Arbeit wurde die Learnability durch beide der bekannten Metriken über den Verlauf der Aufgaben erörtert. (vgl. BECKER 2020: 225; DE BLEECKER/OKOROJI 2018: 14f))

Die ersten drei Metriken stellen quantitative Daten dar. Sie werden durch Videoaufnahmen sowie das Stoppen der Zeit und Aufzeichnungen über die Korrektheit von Lösungen evaluiert.

Eine der wichtigsten Metriken für die Evaluierung der Usability eines Web-GIS ist die Zufriedenheit bzw. die User Experience. Während sich alle anderen Metriken auf objektive

Aspekte fokussieren, steht hier vor allem die subjektive Erfahrung der User:innen im Vordergrund. (vgl. BARNUM 2011: 11f; DE BLEECKER/OKOROJI 2018: 5f)

Meist wird die Zufriedenheit durch Fragebögen mit unterschiedlichen Arten von Fragen evaluiert. Es können beispielsweise Aussagen formuliert werden, die auf Skalen bewertet werden müssen, sowie auch offene Fragen Teil der Evaluierung sein können (vgl. DE BLEECKER/OKOROJI 2018: 123f). Es können hier auch standardisierte Fragebögen verwendet werden, um die Objektivität und Reproduzierbarkeit der Studie zu versichern (vgl. SAURO/LEWIS 2016: 185f). Für diese Studie wurde ein Fragebogen mit Aussagen, die auf der Likert-Skala zu bewerten sind, erstellt. Die Aussagen beziehen sich auf die persönliche Erfahrung der User:innen. Es wurde die folgende Likert-Skala mit vier Stufen der Zustimmung verwendet:

- Stimme zu
- Stimme eher zu
- Stimme eher nicht zu
- Stimme nicht zu

Es wurde kein neutrales Antwortfeld in die Skala eingebaut, um eindeutig positive oder negative Ergebnisse zu erhalten. Während meist ein neutrales Feld verwendet wird, ist auch das Auslassen dessen laut DE BLEECKER/OKOROJI 2018: 126ff eine valide Möglichkeit. Weiters wurden basierend auf Empfehlungen von SAURO/LEWIS 2016: 206f alle Aussagen mit derselben, in diesem Fall positiven Polarität formuliert, um die Studienteilnehmer:innen nicht zu verwirren.

Genauso wurden drei offene Fragen in den Fragebogen eingebaut. Der Fragebogen befindet sich im Anhang 10.3.1.2. Anfangs sollte ein standardisierter Fragebogen verwendet werden, jedoch wurde hier keiner gefunden, der alle Anforderungen erfüllt hat. Deswegen wurden Fragen aus unterschiedlichen standardisierten Fragebögen verwendet und selbst zusammengefügt.

Durch die Ausarbeitung und Beantwortung der drei Arbeitsfrage soll nun die übergeordnete Forschungsfrage der Arbeit beantwortet werden können die wie folgt lautet:

- „Sind signifikante Unterschiede in identifizierbaren Metriken zwischen einem buttongesteuerten und einem textgesteuerten Interface eines Web-GIS erkennbar“

Beantwortet werden kann diese Frage primär durch die Darstellung und Interpretation der Ergebnisse des Usability Testings.

Nach der Fertigstellung der Modelle und des Web-GIS wurde ein Usability Testing geplant. Sämtliche Planungen, die Durchführung sowie die Ergebnisse werden in Kapitel 5 genauer

erläutert. Damit sollten Erkenntnisse darüber erlangt werden, ob das textgesteuerte Web-GIS eine Alternative zum buttongesteuerten Web-GIS ist. Ebenso soll überprüft werden, wie User:innen die Anfragen stellen, ob die Anwendung funktioniert, und an welchen Stellen Probleme bestehen, die ausgebessert werden sollten.

Es wurde entschieden, eine moderierte Remote-Studie durchzuführen, da sich die Studienteilnehmer:innen in verschiedenen Ländern befinden. Weiters wurde den User:innen dadurch ermöglicht, die Studie in ihrem natürlichen Umfeld zu absolvieren. Durch die Moderation konnte bei Problemen und Fragen geholfen und eingegriffen werden. (vgl. BARNUM 2011: 38-44; DE BLEECKER/OKOROJI 2018: 7-16)

Die Studienteilnehmer:innen sind Mitarbeiter:innen der Firma Trafficon, die einen Hintergrund in der Verkehrsplanung und dem Verkehrsmanagement mit Fokus auf Consulting und IT haben. Es wurden insgesamt 10 Personen rekrutiert. Die Anzahl der Personen wurde basierend auf der Literaturrecherche gewählt. Hauptcharakteristika der Personen ist ihr beruflicher Hintergrund, während demographische Merkmale als weniger wichtig erachtet wurden. Jedoch wurde basierend auf Empfehlungen von BARNUM 2011: 117ff darauf geachtet, eine ausgeglichene Geschlechterverteilung zu erhalten, sowie verschiedene Altersgruppen zu inkludieren.

Um Vergleichswerte zwischen dem buttongesteuerten und textgesteuerten Web-GIS zu erhalten wurden sechs Tasks erstellt, die von den Studienteilnehmer:innen in beiden der Anwendungen zu lösen waren.

Folgend werden die Ergebnisse der Studie zusammengefasst und interpretiert, um weiterfolgend die Forschungsfrage beantworten zu können.

Task Success

Die Auswertung des Task Success hat gezeigt, dass in 77 % der Versuche eine erfolgreiche Durchführung der Aufgabe im textgesteuerten Web-GIS erzielt werden konnte, während diese Anzahl beim buttongesteuerten Web-GIS bei 100 % liegt. Ein Task konnte in der textgesteuerten Anwendung nicht gelöst werden. 13 % der Versuche konnten nach einem Problem von den User:innen selbst gelöst werden, wobei in weiteren 8 % der Problemfälle nur mithilfe der Moderatorin eine Lösung erzielt werden konnte.

Ein großer Teil der Fehlerarten sind ungenaue Angaben von relevanten Informationen, wie dem Datum oder der Zeit. Ebenso zählen dazu falsche Satzstellungen. Weiters waren die Fehlerquellen fehlende Angaben, Schreibfehler und Tippfehler. Nur in zwei Fällen lag das Problem tatsächlich an der Anwendung selbst. Im ersten Fall konnte keine Route erstellt werden. Hier lag das Problem

vermutlich an der Positionierung von Absätzen im Text, die so das erste Mal auftraten und in der Datenprozessierung so nicht lösbar war. Das zweite Problem war, dass kein Datum für die Angabe ausgewertet wurde, und die Zeitreihe mit Anfang Jänner startete. Bei erneutem eigenem Ausprobieren trat das Problem nicht mehr auf. Grund dafür war vermutlich, dass vorhergehend der Satz in der Studie schon unterschiedlich eingegeben wurde, und in der Prozessierung schlussendlich kein Ergebnis mehr erstellt werden konnte. Die genauen Ursprünge der Fehler konnten hier nicht festgestellt werden. Grundsätzlich ist jedoch zusammenzufassen, dass die meisten Fehler auf die Unwissenheit der User:innen über die Art der Formulierung und die zu simple Fehlerbehandlung zurückzuführen ist.

Grund, für den im Vergleich zum buttongesteuerten Interface niedriger Erfolg im Task Success kann der Faktor sein, dass das textgesteuerte Interface ungewohnt ist. Systeme wie ChatGPT, in denen rein über die eigene, natürliche Sprache kommuniziert wird, sind erst seit kurzem weitverbreitet. Personen sind dadurch eine solche Eingabe noch nicht gewöhnt. Weiters wurden im Vorhinein kaum Informationen zur Formulierung von Sätzen und Angaben wie etwa dem Datum gegeben. Dies wurde bewusst auf diese Art umgesetzt. Dadurch sollten reale Ergebnisse erzielt werden. So sollten Informationen darüber gesammelt werden, inwieweit Personen Hilfestellungen brauchen, um hier Verbesserungen und Hilfestellungen abzuleiten können. Ein weiterer Grund, wieso Fehler hier zustande kamen, sind Limitierungen der erstellten Anwendung und vor allem der Datenverarbeitung. In dieser Masterarbeit wurden, da es kein Ziel war eine, umfassende Datenprozessierung umzusetzen, alle inkorrekten Anfragen durch Fehlermessages abgefangen.

Das buttongesteuerte Interface hat im Vergleich zum textgesteuerten keine Kapazität hat, Fehler zu erzeugen, da hier schon klar vorgegeben ist, wie Informationen angegeben werden müssen. Dies ist grundsätzlich ein Nachteil des textgesteuerten Interfaces. Es ist jedoch auch zu beachten, dass hier verschiedenste Ansätze bestehen, um Fehlerquellen zu eliminieren und dadurch die Fehlermenge zu reduzieren, was später in Kapitel 6.3 diskutiert wird.

Dieses Ergebnis spiegelt auch die eigenen Erwartungen wider. Durch die neue und ungewohnte Art des Web-GIS war zu erwarten, dass die Verwendung dessen für einige Personen mit Fehlern verbunden ist. Genauso war vorherzusehen, dass das buttongesteuerte Web-GIS kaum bis keine Fehler in der Anwendung aufweisen wird, da die Anwendung hier eine bekannte und strukturierte Vorgehensweise ermöglicht. +

Time on Task

Weiters wurde die Time on Task ausgewertet. Es wurde die Zeit der initialen Eingabe, ohne die Fehlerbehandlung untersucht. Ebenso wurde die Time on Task inklusive der Zeit, die für die Behebung von Problemen benötigt wurde, analysiert. Bei beiden Auswertungen ergeben sich ähnliche Muster. Während in Kapitel 5.3.1.2 die Zeiten durch die Kombination aller statistischen Metriken in einem Boxplot ausgewertet wurden, werden hier noch einmal die wichtigsten Statistiken dargestellt und interpretiert.

Für die Auswertung inklusive der Fehlerbehandlung werden das Minimum und das Maximum noch einmal genauer dargestellt. Abbildung 54 zeigt das Maximum der Zeiten.

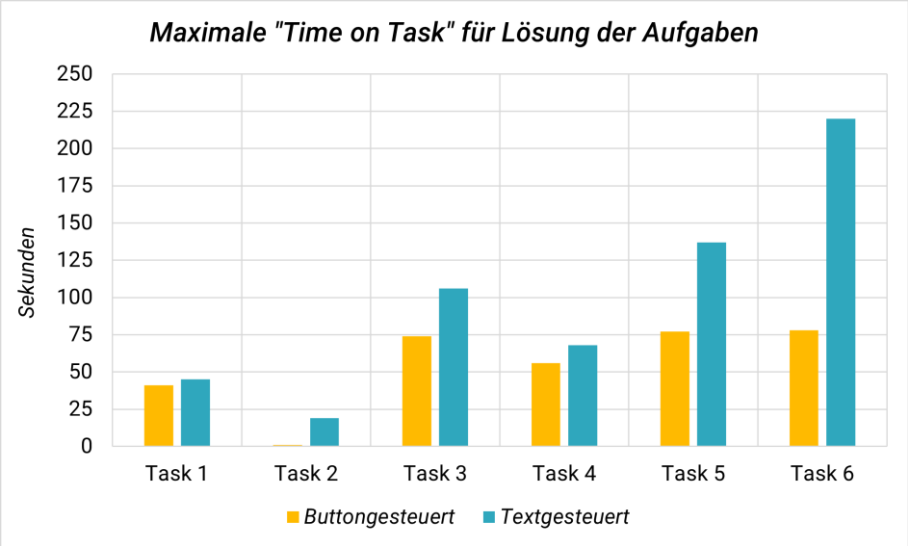

Abbildung 54: Maximale „Time on Task“ für die Lösung der Aufgaben, im Vergleich zwischen dem textgesteuerten und buttongesteuerten Web-GIS (inklusive Fehlerbehandlung)

Durch die verschiedenen Fehler, die entstanden sind, ist die maximale Time on Task in allen Fällen im textgesteuerten Interface vorzufinden. Die langen Zeiten sind bei den Aufgabenstellungen mit der höchsten Fehlerrate zu finden, und sind im Gesamten auf die Fehler zurückzuführen. Es ist jedoch auch bei Task 1 und 4 zu sehen, dass es sich auch hier wie beim Median um nur wenige Sekunden Unterschied handelt.

Gleichzeitig soll auch die minimale Zeit für alle Task überprüft werden. Diese ist in dargestellt Abbildung 55 dargestellt.

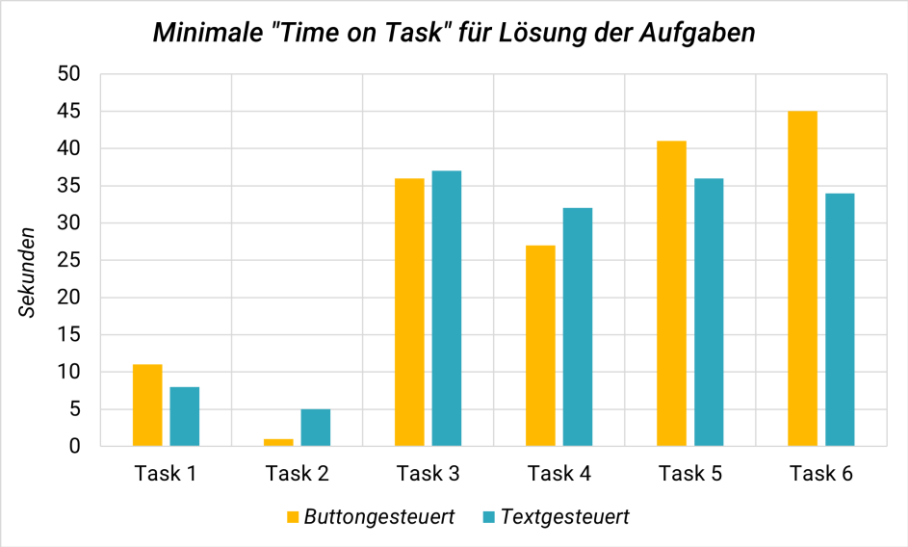

Abbildung 55: Minimale „Time on Task“ für die Lösung der Aufgaben, im Vergleich zwischen dem textgesteuerten und buttongesteuerten Web-GIS (inklusive Fehlerbehandlung)

Die minimale Zeit ist bei Task 1, 5 und 6 im textgesteuerten Interface niedriger als im buttongesteuerten Interface. Bei den anderen Tasks ist die minimale Zeit höher, wobei es sich jedoch bei Task 3 nur um wenige Sekunden handelt. Ansonsten sind hier ebenso wie bei den maximalen Zeiten etwas größere Zeitspannen Unterschied, wie etwa bei Task 6.

Aus der Auswertung des Maximums und des Minimums wird klar, dass die Werte der Time on Task im textgesteuerten Web-GIS stark gestreut sind und sich in beide Richtungen weit ausbreiten.

In Abbildung 56 ist der Median der initialen Eingabe noch einmal genauer dargestellt, um auszuwerten, wie sich das Minimum und Maximum darauf auswirken.

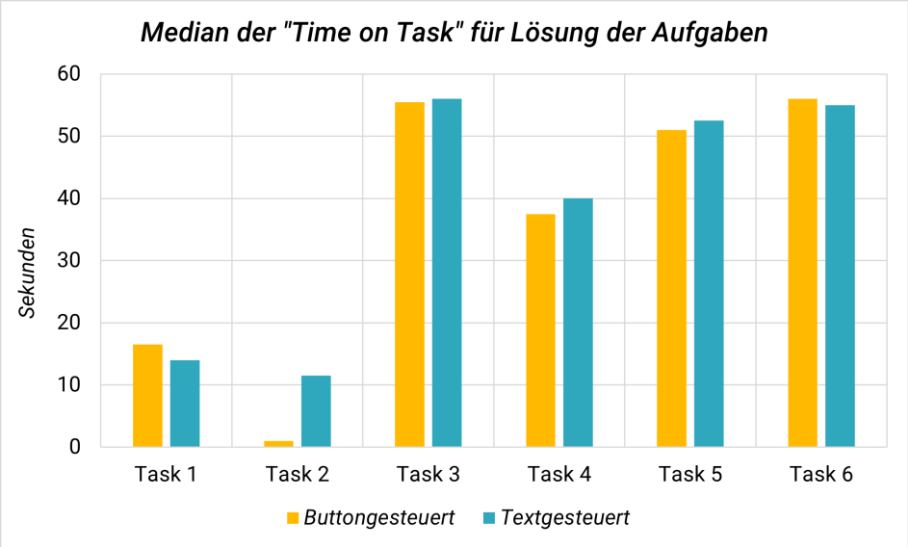

Abbildung 56: Median der „Time on Task“ für die Lösung der Aufgaben, im Vergleich zwischen dem textgesteuerten und buttongesteuerten Web-GIS (initiale Eingabe-Zeit)

In Task 1 und 6 liegt der Median in der textgesteuerten Anwendung niedriger, bei allen anderen Tasks ist das buttongesteuerte Web-GIS schneller. Jedoch ist zu beachten, dass es sich abgesehen von Task 2 immer nur um einen Unterschied im Sekundenbereich handelt. Hier ist interessant zu sehen, dass im Median grundsätzlich keine großen Differenzen zwischen dem textgesteuerten und buttongesteuerten Web-GIS bestehen.

Die Time on Task zwischen den beiden Anwendungen unterscheidet sich folgend vor allem dadurch, dass die Werte stärker gestreut sind, während der Median zeigt, dass im Mittel zwischen textgesteuert und buttongesteuert die Unterschiede so gut wie ausgeglichen sind. Die extremen Zeiten sind wie auch der Task Success darauf zurückzuführen, dass die Anwendung für manche Personen sehr ungewohnt ist, während anderen Personen die Verwendung dessen zu liegen scheint.

Learnability

Weiters wurde die Learnability interpretiert. Dies sollte zeigen, wie schnell User:innen im Web-GIS produktiv werden können. Es wurden vier Tasks (3-6) mit derselben Aufgabe und ansteigender Schwierigkeit evaluiert. Die Learnability wird durch den Verlauf der Time on Task über die Aufgabenstellungen bewertet. Von Task 3 auf 4 sinkt die Zeit deutlich. Zum komplexeren Task hin steigt sie wieder, jedoch zur zusätzlichen Menge an Informationen und Komplexität verhältnismäßig wenig. Zu Task 6 hin bleibt die Time on Task beinahe gleich und sinkt nur minimal. Dasselbe Muster wird bei der Betrachtung der Fehlerrate pro Task sichtbar. Auch hier

steigt die Menge der Fehler grundsätzlich bei Lösung eines komplexeren Tasks an, aber genauso sinkt diese wieder nach dem zweiten Durchlauf.

Die Interpretation der Ergebnisse zeigt, dass die Verwendung des Interfaces durchaus schnell erlernbar ist, da bei der Lösung der zweiten Aufgabe derselben Art die Time on Task und der Task Success besser werden. Während es durch die Auswertung der beiden Metriken so wirken kann, wie wenn das Web-GIS teilweise nur wenig Erfolg zeigt, wird bei der Auswertung dieser im Rahmen der Learnability sichtbar, dass die Verwendung der Anwendung sich schnell verbessert.

User Experience/Zufriedenheit

Die Zufriedenheit und persönliche Erfahrung der Studienteilnehmer:innen wurde wie bereits erwähnt durch einen Fragebogen evaluiert. Inhalt waren Aussagen über das textgesteuerte Interface, denen zugestimmt oder nicht zugestimmt werden konnte. Die Auswertung hat hier gezeigt, dass die Tendenz der Erfahrungen deutlich positiv ist. Nur fünf von 13 Aussagen wurden tendenziell negativ beantwortet. Dies betrifft vor allem die Aussagen, dass sich Fehler schneller korrigieren lassen, und die Lösung schneller und effizienter ist. Besonders positiv wurde bewertet, dass weniger Schritte zur Lösung benötigt werden, die Interaktion natürlicher ist und die Anwendung einfacher zu erlernen ist. Diese Ergebnisse stimmen auch mit der Evaluierung der Time on Task und des Task Success überein.

Weiters empfinden User:innen das textgesteuerte Interface beispielsweise als intuitiver, natürlicher und einfacher zu erlernen. Besonders hervorzuheben für die Beantwortung der Forschungsfrage ist, dass für 90 % der Befragten das textgesteuerte Web-GIS als eine Alternative zum buttongesteuerten ist.

Diese Auswertung stellt teilweise eine gegensätzliche Situation zu den objektiven Metriken dar, da diese gezeigt haben, dass das Interface zwar Potenzial hat, die Verwendung manchen Studienteilnehmer:innen jedoch schwer gefallen ist. Die positive Tendenz der Antworten im Fragebogen kann möglicherweise auf die Aussage von DE BLEECKER/OKOROJI 2018 zurückgeführt werden. Die positiven Erfahrungen von User:innen mit einer Anwendung hängen nicht nur von den objektiven Faktoren ab, sondern sehr stark auch von subjektiven Aspekten. Es kann vermutet werden, dass die Teilnehmer:innen durchaus interessiert an der Studie waren. Der Ansatz ist neu und modern, und eine intuitive und natürliche Möglichkeit der Interaktion, was für Personen ein durchaus positives Erlebnis ermöglichen kann.

Weiters ist auch zu beachten, dass auch, wenn bei einigen Personen Fehler aufgetreten sind und die Lösung der Tasks dadurch auch länger gedauert hat, die Studie bei andere Personen sehr positiv

verlaufen ist, und durchaus bessere Ergebnisse im textgesteuerten Web-GIS erzielt wurden, was sich auch auf die Antworten ausgewirkt haben kann.

Durch die Antworten auf die offenen Fragen konnte identifiziert werden, welche positiven oder negativen Stichwörter oft erwähnt wurden. So konnten auch sehr persönliche Empfindungen der Personen gesammelt werden und so ein besseres Verständnis über ihre Meinung und Erfahrung erhalten werden. Mit positiver Tendenz wurden hier besonders oft „schnell", „einfach", und „intuitiv" erwähnt. Kritisiert wurde hingegen, dass das Interface langsamer, unklarer, und gewöhnungsbedürftig ist. Die gesamten Aussagen sind im Anhang zu finden.

In der Auswertung der offenen Fragen wurden teils gegensätzliche Aussagen von unterschiedlichen Teilnehmer:innen getroffen. Die vollständigen Antworten aller Personen befinden sich im Anhang 10.3.2.7. Während manche der Personen hervorheben, dass das neue Interface flexibel, intuitiv und natürlicher ist, sagen andere, dass ihnen im Gegensatz dazu das buttongesteuerte Interface die Vorgehensweise klar definiert und strukturiert. Es ist hier festzustellen, dass die Präferenzen der Personen sich durchaus unterscheiden.

Genauso erwähnt ein Teil der Teilnehmer:innen, dass das textgesteuerte Web-GIS das effiziente und schnelle Arbeiten ohne das Kennen von Funktionen und Einlernen ermöglicht, während wiederum von anderen Personen gesagt wird, dass die Eingabe von Texten mühsam und unnötig ist.

Ein weiterer Gegensatz ergibt sich in der Aussage, für welche Arten von Anfragen das Web-GIS geeignet ist. Manche erwähnen, dass es besonders für komplexe Anfragen angenehm ist. Im Gegensatz dazu sagen Personen, dass das Interface vor allem für die Eingabe von kurzen Anfragen praktisch ist.

Eine Sache, die zwei Personen erwähnt haben ist, dass sie die Anwendung eher als Tool für gelegentliche und unerfahrene User:innen sehen, und als Experten-Tool das buttongesteuerte Interface vorziehen würden. Dies kann auf das Auftreten der Fehler und die dadurch entstehenden längeren Zeiten zurückgeführt werden, da diese bei Experten-Tools verständlicherweise nicht erwünscht ist.

Aus der Studie ist vor allem herauszulesen, dass starke Differenzen zwischen den Meinungen, aber auch der Funktion des Web-GIS bestehen. Bei manchen Leuten hat die Anwendung sehr gut funktioniert, es sind wenige bis keine Fehler aufgetreten und die Lösung der Aufgaben war schnell durchführbar. Für diese Personen handelt es sich um eine positive Erfahrung mit dem Web-GIS.

Im Gegensatz kam es bei anderen Studienteilnehmer:innen zu Problemen. Es entstanden Fehler und die Lösung der Tasks war langsamer.

Das textgesteuerte Interface ist ein neuer, moderner und innovativer Ansatz, der die momentane Entwicklung in der Technik widerspiegelt. Die Interaktion durch natürliche Sprache mit einer Anwendung wurde vor kurzem erst durch Webistes wie ChatGPT allgemein bekannt. Manchen Personen liegt diese Interaktionsform, und sie haben solche Systeme auch schon häufig verwendet, während andere Personen die Interaktion als ungewohnt und nicht angenehm empfinden. Die Präferenzen von Personen unterscheiden sich grundsätzlich auch. Folgendes Zitat aus den offenen Fragen (Anhang 10.3.2.8) zeigt jedoch auch die durchaus positiven Meinungen der Teilnehmer:innen:

„Innovativer und niedrigschwelliger Ansatz, um Menschen mit Ablehnung von klassischen UI-Elementen zur Bedienung komplexer Software zu bringen!"

Nun zu einer zusammenfassenden Beantwortung der Forschungsfrage. Basierend auf dem Ergebnis des Fragebogens stellt das textgesteuerte Web-GIS eine Alternative zu einem buttongesteuerten Web-GIS dar. Es bestehen Differenzen und Präferenzen bei den User:innen, wobei sich hier eine große Streuung zwischen positiven und negativen Ergebnissen und Meinungen zeigt. Weiters besteht noch Bedarf für Forschung und einer Weiterentwicklung des Web-GIS.

6.2. Beschränkungen der Forschung

Eine der größten Limitierungen des textgesteuerten Web-GIS ist das eingeschränkte Geoparsing. Das Geoparsing, dass für diese Anwendung eingesetzt wurde, kann keine detaillierten Informationen verarbeiten. Das Problem liegt grundsätzlich nicht in der Funktion des Geotaggings und Geocodings. Durch die Named Entity Recognition können ein großer Teil an räumlichen Informationen extrahiert werden, und der Geocoder kann diesen wiederum auch die passenden Koordinaten zuweisen. Die Beschränkung liegt hierbei primär in der Datenprozessierung selbst. Es wird automatisiert immer das erste Ergebnis aus dem Geocoder verwendet. Dabei entsteht das Problem, dass das zurückgegebene Ergebnis nicht die Straße am gewünschten Ort zurückgibt. Für diese Arbeit wurde das Geoparsing bewusst so einfach wie möglich gehalten, da ein funktionsfähiges Geoparsing-System zwar ein notwendiger Teil der Anwendung ist, jedoch kein Ziel oder Teil der Arbeit war, sich auf ein möglichst detailliertes System zu konzentrieren.

Ähnliches ist auch für die gesamte implementierte Datenverarbeitung der Input-Daten zu behaupten. Auch diese stellt eine Beschränkung der Arbeit dar. Die Verarbeitung wurde ebenfalls möglichst einfach gehalten, um auch hier den Fokus auf der Beantwortung der Fragestellung zu belassen. Ein System zur Datenverarbeitung war grundsätzlich nötig und wurde auch umgesetzt,

Ziel der Arbeit war es jedoch nicht, alle Fehler, die durch User-Eingaben entstehen zu behandeln. Dies ist vor allem für ein zukünftiges Produktivsystem von Bedeutung.

Beide der nun genannten Beschränkungen spiegeln sich auch in der folgenden, und letzten Limitierung wider. Das entwickelte System stellt einen Prototyp dar, mit dem die Forschungsfragen beantwortet werden konnten. Um jedoch als Produktivsystem im Alltag von Experten verwenden zu können, muss dieses noch weiterentwickelt werden.

6.3. Empfehlungen und Ausblick für weiterführende Forschung

Das textgesteuerte Web-GIS zeigt Potenzial und kann für zukünftige technische Entwicklungen einen wichtigen Schritt darstellen. Es kann zur weiteren Verbreitung von Web-GIS im öffentlichen und privaten Bereich beitragen. Hier besteht jedoch auch noch ein Bedarf an Forschung und Weiterentwicklung. Folgend sollen eine Empfehlung und ein Ausblick für die weiterführende Forschung gegeben werden. Diese basieren auf den Ergebnissen der Masterarbeit und den dadurch identifizierten Fehlerquellen und Problemstellen. Die Empfehlungen sind ebenso auf den Verbesserungsvorschlägen der User:innen ausgewählt worden.

Bezogen auf den Anlernprozess der Modelle müssen umfangreichere Trainingsdaten erstellt werden. Der hier verwendete Satz an Trainingsdaten wurde basierend auf Vermutungen, wie Anfragen von User:innen aussehen könnten erstellt. Nun wurde Erfahrung gesammelt, wie diese tatsächlich aussehen. Der nächste Schritt ist, diese Erfahrungen herzunehmen und daraus bessere Trainingsdaten, vor allem auch für die Named Entity Recognition zu erstellen. Weiters muss untersucht werden, inwiefern mit Klassen wie hier mit Intervallen und Kennwerten umgegangen wird, bei denen nur eine sehr kleine Menge an Daten besteht. Ebenso besteht die Möglichkeit und Empfehlung, noch bessere Modelle für die Verarbeitung der Trainingsdaten zu trainieren.

Eine der relevantesten Verbesserungen besteht in der Implementierung eines detaillierten Geoparsing-Systems. Dadurch sollen komplexere Anfragen verarbeitet und beantwortet werden können. Dies beinhaltet vor allem die Möglichkeit, zusätzlich zu den Straßen den Suchbereich des Geoparsers mit der Angabe von Örtlichkeiten wie etwa Bezirken und Gemeinden eingrenzen zu können.

Auch müssen die Datenprozessierung des User-Inputs und die Fehlermessages ausgeweitet und verbessert werden. Während im momentanen Web-GIS noch bei Abweichungen von „korrekten“ Anfragen Fehler auftreten, muss dies für ein Produktivsystem verändert werden. Hier muss einerseits die Verarbeitung der Daten ausgeweitet werden, sodass die Anfragen auf mehr unterschiedliche Arten gestellt werden können und beispielsweise die Satzstruktur und Datumsangaben flexibler umsetzbar sind. Genauso muss eine Fehlerbehandlung implementiert

werden, sodass mögliche Fehler vom System auch schon vom System selbst behandelt werden können. Weiters müssen konkretere Fehlermessages implementiert werden.

Bezogen auf die Anwendung ist eine Autocomplete-Funktion von Straßennamen, Datumsangaben und Zeitangaben ein Vorschlag von den Studienteilnehmer:innen. Ein weiterer Wunsch der Teilnehmer:innen ist es, Hilfestellungen zur Formulierung von Anfragen, sowie Textvorschläge in das Interface einzubauen. Dies wäre besonders für neue User:innen eine hilfreiche Verbesserung.

Die weiteren Empfehlungen und Vorschläge stimmen auch mit den Empfehlungen, die SETLUR ET AL. 2016 erwähnt, überein. So wurde von den Teilnehmer:innen der eigenen Studie erwähnt, beispielsweise eine Kombination des textgesteuerten und buttongesteuerten Web-GIS zu erstellen. Dies ist definitiv ein interessanter Ansatz für zukünftige Forschungen. SETLUR ET AL. 2016 hat hier durchaus schon positive Ergebnisse gezeigt, jedoch handelt es sich auch hier nur um einen Prototypen. Ebenso schlägt SETLUR ET AL. 2016 die Erstellung von multimodalen Interfaces vor. Auch in der eigens durchgeführten Befragung im Rahmen der Studie wurde erwähnt, dass beispielsweise die Spracheingabe eine interessante Erweiterung wäre. Zusammenfassend ist zu sagen, dass die Untersuchung der multimodalen Interaktion eine spannende Forschungsrichtung ist, um die Vorteile aller Interaktionsmöglichkeiten nutzen zu können.

Abschließend ist zu sagen, dass generell noch Forschungsbedarf besteht und eine Weiterentwicklung nötig ist, um das erstellte textgesteuerte Web-GIS in ein Produktivsystem umzuwandeln. So kann auch die Akzeptanz dieser Form der Interaktion zwischen Mensch und Anwendung gesteigert, und zu weiteren positiven Entwicklungen beitragen werden.

7. Fazit

Es wurden alle der Ziele, die zu Beginn der Masterarbeit definiert wurden, erfüllt und alle Forschungsfragen beantwortet. Der Stand der Forschung der Grundkonzepte des Natural Language Processing, sowie in der Kombination mit räumlichen Daten wurde dargestellt und erläutert. Weiters wurden zwei Deep Learning Modelle für die Einbindung in ein Web-GIS trainiert und die Performance dieser evaluiert. Es wurden ebenfalls geeignete Metriken der Human Computer Interaction identifiziert, die für ein Usability Testing des Web-GIS geeignet sind. Weiters wurde das Usability Testing geplant, durchgeführt und die Ergebnisse evaluiert.

Durch die Studie konnte gezeigt werden, dass das textgesteuerte Web-GIS, das in der Masterarbeit umgesetzt wurde, für die Studienteilnehmer:innen eine Alternative zum traditionellen, buttongesteuerten Web-GIS darstellt. Es konnten Differenzen in der Verwendung festgestellt werden. Für einen Teil der Personen stellte die Benutzung des textgesteuerten Interfaces eine Herausforderung dar. Die Verwendung war durch das Auftreten von Fehlern und eine langsamere Lösung von Aufgaben gekennzeichnet. Ebenso konnten jedoch auch positive Erkenntnisse aus der Studie gezogen werden, da für andere Personen die Benutzung des textgesteuerten Web-GIS erfolgreich war. Sie konnten es schneller und effizienter als das buttongesteuerte Interface verwenden. Hier konnte auch eine persönliche positive Erfahrung erzielt werden. Zusammenfassend zeigt die Studie, dass das textgesteuerte Web-GIS für User:innen eine Alternative darstellt. Es kann tatsächlich zu einer einfacheren Lösung von Aufgaben im Berufsleben beitragen, und so Experten aus anderen Fachrichtungen ermöglichen, die Anwendung produktiv zu verwenden.

Das textgesteuerte Web-GIS stellt einen innovativen, modernen und niederschwelligen Ansatz für eine Alternative zum buttongesteuerten Web-GIS dar und zeigt Potenzial. Durch weitere Forschungen und eine Weiterentwicklung der Anwendung kann eine größere Akzeptanz dieser Interaktionsform geschaffen werden, und so ein größerer Mehrwert in der Verwendung von Web-GIS in verschiedensten Bereichen erreicht werden.

8. Literaturverzeichnis

ALBASEER, ABDULLATIF/YAQOT, MOHAMMED (2018): Integration Mapping Application: (Web-GIS as Geo-Located Tweets and Sentiment Analysis). In: Journal of Computer Engineering and Information Technology, 2018/7/4-7

TULLIS, TOM/ALBERT, BILL (2013). Measuring the user experience: collecting, analyszing and presenting usability metrics. 2. Auflage

BARNUM CAROL M. (2011): Usability Testing Essentials : Ready, Set… Test!. Amsterdam, Boston: Morgan Kaufmann Publishers

BARTZ, EVA/BARTZ-BEIELSTEIN, T/ZAEFFERER, MARTIN/MERSMANN, OLAF (Hrsg.) (2023): Hyperparamter Tuning for Machine and Deep Learning with R, Singapore: Springer Nature

BECKER, CRISTOPHER REID (2020): Learn Human-Computer Interaction: solve human problems and focus on rapid prototyping and validating solutions through user testing. Birmingham: PACKT Publishing Limited

CALÌ, DAVIDE/CONDORELLI, ANTONIO/PAPA, SANTO/RATA, MARIUS/ZAGARELLA, LUCA (2011): Improving intelligence through use of Natural Language Processing. A comparison between NLP interfaces and traditional visual GIS interfaces. In: Procedia Computer Science, 2011/5/920-925

CHO, HAN-CHEOL/OKAZAKI, NAOAKI/MIWA, MAKOTO/TSUJI, JUN`ICHI (2013): Named entity recognition with multiple segment representations. In: Information Processing & Management, 49/4/954-965

DALE, ROBERT (2010): Classical Approaches to Natural Language Processing. In: INDURKHYA, NITIN/DAMERAU, FRED J. (Hrsg.): Handbook of Natural Language Processing. Boca Raton: Taylor & Francis Group, 3-8

DE BLEECKER, INGE/OKOROJI, REBECCA (2018): Remote Usability Testing: Actionable Insights in User Behaviour Across Geographies and Times Zones. Birmingham: PACKT Publishing Limited

DE LANGE, NORBERT (Hrsg.) (2020): Geoinformatik in Theorie und Praxis. Grundlagen von Geoinformationssystemen, Fernerkundung und digitaler Bildverarbeitung. 4. Auflage, Berlin Heidelberg: Springer Berlin Heidelberg

DRANSCH, DORIS/SIPS, MIKE/UNGER, ANDREA (2019): GeoVisual Analytics. In: SESTER, MONIKA (Hrsg.): Geoinformatik. 1. Auflage, Berlin Heidelberg: Springer Berlin Heidelberg: Imprint: Springer Spektrum, 22-41

GODDARD, CLIFF/SCHALLEY, ANDREA C. (2010): Semantic Analysis. In: INDURKHYA, NITIN/DAMERAU, FRED J. (Hrsg.): Handbook of Natural Language Processing. Boca Raton: Taylor & Francis Group, 93-120

GRITTA, MILAN/TAHER PILHEVAR, MOHAMMED/COLLIER, NIGEL (2019): A pragmatic guide to geoparsing evaluation. Toponyms, Named Entity Recognition and pragmatics. In: Lang Resources & Evaluation, 2020/54/683-712

GRITTA, MILAN/TAHER PILHEVAR, MOHAMMED/ LIMSOPATHAM, NUT/ COLLIER, NIGEL (2017): What's missing in geographical parsing?. In: Lang Resources & Evaluation, 2018/52/603-623

HIPPISLEY, ANDREW (2010): LEXICAL ANALYSIS. In: INDURKHYA, NITIN/DAMERAU, FRED J. (Hrsg.): Handbook of Natural Language Processing. Boca Raton: Taylor & Francis Group, 31-58

HIRSCHLE, JOCHEN (2022): Deep Natural Language Processing. Einstieg in Word Embedding, Sequence-to-Sequence-Modelle und Transformer mit Python. München: Carl Hanser Verlag München

JIANG, HUI (2021): Machine Learning Fundamentals. A Concise Introduction. Cambridge: Cambridge University Press

KAMATH, UDAY/GRAHAM, KENNETH L./EMARA, WAEL (2022): Tranformers for Machine Learning. A Deep Dive. 1. Auflage, Boca Raton: Chapman and Hall/CRC

KERSKI, JOSEPH J./BAKER, THOMAS R. (2019): Infusing Educational Practice with Web GIS. In: DE MIGUEL, GONZÀLEZ, RAFAEL/DONERT, KARL/KOUTSOPOLOUS, KOSTIS (Hrsg.): Geospatial technologies in geography eduction. Cham: Singer Eurogeo, 3-20

KROHN, JOHN/BEYLEVELD, GRANT/BASSENS, AGLAÉ (2022): Deep Learing Illustrated: A Visual, Interactive Guide to Artificial Intelligence. Addison-Wesley Data &Analytics

KUBAT, MIROSLAV (2015): An Introduction to machine Learning. 1. Auflage. Springer

KULAWIAK, MARCIN/DAWIDOWICZ, AGNIESKA/ PACHOLCZYK, MAREK EMANUEL (2019): Analysis of server-side and client-side Web-GIS data processing methods on the example of JTS and JSTS using open data from OSM and geoportal. In: Computers & Geosciences, 129/8/26-37

LAMPOLTSHAMMER, THOMAS J./HEISTRACHER, THOMAS (2012): Natural Language Processing in Geographic Information Systems – Some Trends and Open Issues. In: In. J. Comp Sci. Emerging Tech, 3/3/81-88

LAURIOLA, IVANO/ LAVELLI, ALBERTO/AIOLLO, FABIO (2022): An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools. In: Neurocomputing, 2022/470/443-456

LEIDNER, JOCHEN/LIEBERMAN, MICHAEL D. (2011): Detecting geographical references in the form of place names and associated spatial natural language. In: SIGSPATIAL, 3/2/5-11

LI, BOHAN/HOU, YUTAI/CHE, WANXIANG (2022): Data augumentation approaches in natural language processing: A survey. In: AI Open, 3/71-90f

MCKENZIE, GRANT/ADAMS, BENJAMIN (2021): Natural Language Processing in GIScience Applications. Bezogen unter: https://www.researchgate.net/publication /232250058_Natural_Language_Processing_in_Geographic_Information_Systems__Some_Trends_and_Open_Issues_- (Zugriff: 01.09.2023)

MIDDLETON, STUART E./KORDKOPATIS-ZILOS, GIORGOS /PAPADOPOLOUS, SYMEON/KOMPATSIARIS, YIANNIS (2018): Location Extraction from Social Media: Geoparsing, Location Disambiguation, and Geotagging. In: ACM Transactions on Information Systems, 36/4/1-27

PALMER, DAVID D. (2010): Text Processing. In: INDURKHYA, NITIN/DAMERAU, FRED J. (Hrsg.): Handbook of Natural Language Processing. Boca Raton: Taylor & Francis Group, 9-30

PANDEY, ASHUTOSH/BISWAS, SUBHADIP (2022): Assesment of Level of Service on urban roads: a revisit to past studies. In: Advances in transportation studies, 57/49-70

PATEL, ANKUR A. (2020): Praxisbuch Unsupervised Learning. Machine-Learning-Anwendungen für ungelabelte Daten mit Python programmieren. 1. Auflage: O`Reilly

SAURO, JEFF/LEWIS, JAMES R. (2016): Quantifying the User Experience: Practical Statistics for User Research. 2. Auflage, San Fransisco: Elsevier Science

SCHEIBLE, RAPHAEL/THOMCZYK, FABIAN/TIPPMANN, PATRIC/JARAVINE, VICTOR/BOEKER, MARTIN (2020): GottBERT: a pure German Language Model. Bezogen unter: https://www.researchgate.net/publication/346614679_GottBERT_a_pure_German_Language_Model (Zugriff: 02.09.2023)

SCHIERSCH, MARTIN/MIRONOVA, VESELINA/SCHMITT, MAXIMILIAN/THOMAS, PHILLIPPE/GABRYSZAK, ALEKSANDRA/HENNIG, LEONHARD (2018): A German Corpus for Fine-Grained Named Entity Recognition and Relation Extraction of Traffic and Industry Events. Bezogen unter: https://aclanthology.org/L18-1703.pdf (Zugriff: 02.09.2023)

SETLUR, VIDYA/BATTERSBY, SARAH/TORY, MELANIE/GOSSWEILER, RICH/CHANG, ANGEL X. (2016): Eviza: A Natural Language Interface for Visual Analysis. Bezogen unter: https://www.researchgate.net/publication/309222635_Eviza_A_Natural_Language_Interface_for_Visual_Analysis (Zugriff: 02.09.2023)

SHOAB, MOHD/TIWARI, ANUJ/WU, DEREK/BENEDIX, FRANK (2017): The Evolution and Emerging Trends of Web GIS with a Special Emphasis on Assets Management and Navigation Services for the Indian Railway. – In: TIWARI, ANUJ/JAIN, KAMAL (Hrsg.): Concepts and Applications of Web GIS. New York: Nova Science Publishers, Inc.

TUNSTALL, LEWIS/VON WERRA, LEANDRO/WOLF, THOMAS/FRAAß, MARKUS (2023): Natural Language Processing mit Transformers. Sprachanwendungen mit Hugging Face erstellen. 1. Auflage, Heidelberg: dpunkt.verlag

VASILEV, IVAN (2019): Python Deep learning: exploring deep learning techniques and neural network architectures wit PyTorch, Keras, and TensorFlow. 2 Auflage, Brimingham: Packt Publishing

VEENENDAAL, BERT/BROVELLI, MARIA ANTONIA/LI, SONGNIAN (2016): Review of web mapping: Eras, trends and driections. In: ISPRS International Journal of Geo-Information, 6/10/317-348

ZHANG, YUE/TENG, ZHIYANG (2021): Natural Language Processing: a machine learning perspective. Cambridge: Cambridge University Press

ZHAODI, LI/DAN, LI (2022): Data Visualization and Interaction of Urban Traffic Logistics Management System Using WebGIS. In: Computational Intelligence and Neuroscience, 1-12

ZWERENZ, KARLHEINZ (2015): Statistik: Einführung in die computergestützte Datenanalyse. 6. Auflage, Berlin/München/Boston: De Gruyter Oldenbourg

9. Quellenverzeichnis

CHOLLET, FRANCOIS ET AL. (2015): Keras. Dokumentation und Tutorials von Keras. Online: https://keras.io/guides/keras_tuner/ (zugriff: 24.06.2023)

CLEVERTAP (2019): Introduction to Natural Language Processing. Blogseite „CleverTap“. Online: https://clevertap.com/blog/natural-language-processing/ (Zugriff: 04.09.2023)

EUROPÄISCHES PARLAMENT (2023): Was ist künstliche Intelligenz und wie wird sie genutzt?. Online: https://www.europarl.europa.eu/news/de/headlines/society/20200827STO85804/was-ist-kunstliche-intelligenz-und-wie-wird-sie-genutzt (Zugriff: 01.09.2023)

INTERNATIONAL ORGANISATION FOR STANDARDIZATION (ISO) (2018): ISO 9241-11. Ergonomics of human-system-interation – Part 11: Usability Definitions and concepts.

INTERNATIONAL ORGANISATION FOR STANDARDIZATION (ISO) (2020): ISO 9241-110. Ergonomics of human-system-interation – Part 110: Interaction Principles.

MOBILITÄTSREFERAT (2023): TEMPUS. Testfeld für automatisiertes Fahren in München. Projekthomepage. Online: https://tempus-muenchen.de/ (Zugriffsdatum: 01.09.2023)

PEDREGOSA ET AL. 2011: Scikit-learn: Machine Learning in Python. Homepage der Python-Library „scikit-learn“. Online: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

TRAFFICON (2023): TEMPUS – A Pilot Test for Urban Automated Road Traffic. A Field Experiment in Munich. Homepage der Firma Trafficon – Traffic Consultants GmbH. Online: https://www.trafficon.eu/en/projekte/tempus-urban-automated-road-traffic/ (Zugriff: 03.09.2023)

WOLF, THOMAS ET AL. (2020): Transformers: State-of-the-Art Natural Language Processing. Dokumentation und Tutorials der Library Hugging Face Transformers. Online: https://github.com/huggingface/transformers (Zugriff: 22.07.2023)

10. Anhang

10.1. Skripte

10.1.1. Skript Trainingsdaten

```
#!/usr/bin/env python
# coding: utf-8
@ author: Anna Strobl 2023

### IMPORT & DIR

import geopandas as gpd
import pandas as pd
import os
import itertools
import random
import matplotlib as plt
import datetime as dt
import locale
import ast

from transformers import AutoTokenizer
from sklearn.model_selection import train_test_split

path = os.getcwd()
data = os.path.join(path, 'data')

grenzen_file = os.path.join(data, 'grenzen_clip.gpkg')
osm_file = os.path.join(data, 'roads_clip.gpkg')
csv_file = os.path.join(path, 'dataset_sequence_token.csv')
csv_new = os.path.join(path, 'dataset_newnew.csv')
csv_train = os.path.join(path, 'dataset_train.csv')
csv_val = os.path.join(path, 'dataset_val.csv')
csv_test = os.path.join(path, 'dataset_test.csv')
csv_gottbert = os.path.join(path, 'dataset_gottbert.csv')

roads_gdf = gpd.read_file(osm_file)
print(roads_gdf.head(10))

#### DATA LISTS

##### Straßen

roads_gdf = roads_gdf.dropna(subset=['name'])
road_names = roads_gdf['name'].values.tolist()

for i in road_names:
    length = len(i)
    if length < 2:
        road_names.remove(i)

roads_lst = list(set(road_names))

locale.setlocale(locale.LC_TIME, "de_DE")

##### Datum

date_range = pd.date_range(start = '1990/1/1', end = '2010/12/24', freq='D')

date_base = date_range.strftime('%d.%m.%Y').tolist()

date_zero = date_range.strftime('%#d.%#m.%Y').tolist()

date_weekday = date_range.strftime('%A, %d.%m.%Y').tolist()

date_month = date_range.strftime('%d. %B %Y').tolist()

date_lst = date_base + date_zero + date_weekday + date_month

##### Uhrzeit

time_range = pd.date_range(start = '1/1/2000', periods = 1440, freq='T')

time_base = time_range.strftime('%H:%M').tolist()

time_uhr = time_range.strftime('%H:%M Uhr').tolist()

time_uhr_simple = time_range.strftime('%#I Uhr').tolist()
time_uhr_simple = list(set(time_uhr_simple))
```

```
time_uhr_zero = time_range.strftime('%#I:%M Uhr').tolist()

time_lst = time_base + time_uhr + time_uhr_simple + time_uhr_zero

##### Intervalle
intervall_dauer = ['Tage', 'Stunden', 'Std.', 'Minuten', 'Min.']

intervall_werte = list(range(1, 60))

intervall_lst = []

for i in intervall_werte:

    for j in intervall_dauer:

        if i == 1 and (j == 'Tage' or j == 'Stunden'):

            j_lst = list(j)
            j_single = ''.join(j_lst[:-1])
            intervall_lst.append(str(i) + ' ' + j_single)

        else:

            intervall_lst.append(str(i) + ' ' + j)

#### Verkehrssynonyme

verkehr_frage = ['die Verkehrssituation', 'das Verkehrsaufkommen', 'die Verkehrsbedingung',
                 'die Verkehrsverhältnisse', 'der Straßenverkehr', 'der Verkehrszustand',
'die Verkehrslage']

verkehr_aufforderung = ['die Verkehrssituation', 'das Verkehrsaufkommen', 'die
Verkehrsbedingung',
                 'die Verkehrsverhältnisse', 'den Straßenverkehr', 'den Verkehrszustand',
'die Verkehrslage']

route_lst = ['der Route', 'dem Weg', 'der Strecke', 'dem Routenverlauf',  'dem Verlauf',
'der Fahrstrecke']

#### Füllwörter

verben_lst = ['Zeige', 'Weise', 'Demonstriere', 'Visualisiere', 'Präsentiere']

preposition_start = ['ab', 'ausgehend von', 'entlang', 'über', 'von', 'ausgangs', 'von',
'zwischen']
nomen_start = ['Start', 'Startpunkt', 'Anfang', 'Ausgangspunkt', 'Beginn']

preposition_via = ['auf', 'via', 'entlang', 'nach', 'über', 'und']
nomen_via = ['Via-Punkt', 'Zwischenstopp']

preposition_ende = ['an', 'nach', 'zu']
nomen_ende = ['Abschluss', 'Endpunkt', 'Ziel', 'Zielpunkt', 'Schluss', 'Schlusspunkt',
'Endstation']

fragewort_situation = ['Wie', 'Was']
fragewort_route = ['Wo', 'Was', 'Welche']

aktuell_lst = ['aktuelle', 'gegenwärtige', 'augenblickliche', 'momentane', 'derzeitge']

gut_lst = ['günstigste', 'optimalste', 'beste', 'schnellste', 'bestmögliche']

#### Kennwerte

durchschnittsgeschwindigkeit = ['die Durchschnittsgeschwindigkeit', 'die
Mittelgeschwindigkeit',
                                'die durchschnittliche Geschwindigkeit', 'die mittlere
Geschwindigkeit'
                                'das Durchschnittstempo', 'das Mitteltempo', 'das
durchschnittliche Tempo', 'das mittlere Tempo',]

freiflussgeschwindigkeit = ['die Freiflussgeschwindigkeit']

verkehrslage = ['die Verkehrslage', 'das LOS', 'die Verkehrslage (LOS)', 'das Level of
Service']

verlustzeit = ['die Verlustzeit', 'den Zeitverlust']

reisezeit = ['die Reisezeit']

vertrauenskennwert = ['den Vertrauenskennwert']

kennwerte = durchschnittsgeschwindigkeit + freiflussgeschwindigkeit + verkehrslage +
verlustzeit + reisezeit + vertrauenskennwert
```

```
kennwerte_ner = []

for i in kennwerte:
    i_new = i.split(' ')[-1]
    kennwerte_ner.append(i_new)

kennwerte_lst = list(set(kennwerte))

### Satzerstellung

def generate_satzteile(*satzteile):
        satz = []
        satz_struktur_lst = []

        satzteile_lst = list(satzteile)
        permutations = list(itertools.permutations(satzteile_lst))

        for permutation in permutations:

            permutation = list(permutation)
            satz.append("".join(permutation))
            satz_struktur_lst = [i.strip("['',].") for i in satz]

        return satz_struktur_lst

def generate_sentences(amount):

    historisch_frage, historisch_forderung, historisch_konjunktiv, live_frage,
live_forderung, live_konjunktiv, analyse_frage, analyse_forderung, analyse_konjunktiv = ([]
for i in range(9))

    for i in range(amount):

        verkehr_fragewort = random.choice(verkehr_frage).split()
        verkehr_aufforderungswort = random.choice(verkehr_aufforderung).split()

        verb = random.choice(verben_lst)
        route = random.choice(route_lst)
        frage_monitoring = random.choice(fragewort_situation)
        frage_analyse = random.choice(fragewort_route)
        aktuell = random.choice(aktuell_lst)
        gut = random.choice(gut_lst)
        straße = random.sample(roads_lst, 3)
        start = random.choice(preposition_start)
        via = random.choice(preposition_via)
        ende = random.choice(preposition_ende)
        route_struktur_lst = [f' auf {route} {start} {straße[0]} {ende} {straße[2]}', f'
auf {route} {start} {straße[0]} {via} {straße[1]} {ende} {straße[2]}']
        route_struktur = random.choice(route_struktur_lst)

        datum = random.sample(date_lst, 2)
        zeit = random.sample(time_lst, 2)
        datum_struktur_lst = [f' von {datum[0]} bis {datum[1]}', f' zwischen {datum[0]} und
{datum[1]}']
        datum_struktur = random.choice(datum_struktur_lst)
        zeit_struktur_lst = [f' von {zeit[0]} bis {zeit[1]}', f' zwischen {zeit[0]} und
{zeit[1]}']
        zeit_struktur = random.choice(zeit_struktur_lst)
        intervall = random.choice(intervall_lst)
        intervall_struktur_lst = [f' im Intervall von {intervall}', f' aggregiert auf
{intervall}']
        intervall_struktur = random.choice(intervall_struktur_lst)

        kennwert = random.choice(kennwerte_lst)

        verkehr_frage_struktur = f'{verkehr_fragewort[0]} {verkehr_fragewort[1]}'
        verkehr_forderung_struktur = f'{verkehr_aufforderungswort[0]}
{verkehr_aufforderungswort[1]}'
        satzteile_frage_historisch = generate_satzteile(f' {verkehr_frage_struktur}',
f'{datum_struktur}', f'{zeit_struktur}')
        satzteile_forderung_historisch = generate_satzteile(f'
{verkehr_forderung_struktur}', f'{datum_struktur}', f'{zeit_struktur}')
        historisch_frage_struktur = random.sample(satzteile_frage_historisch, 1)
        historisch_forderung_struktur = random.sample(satzteile_forderung_historisch, 1)

        verkehr_live_frage_struktur = f'{verkehr_fragewort[0]} {aktuell}
{verkehr_fragewort[1]}'
        verkehr_live_forderung_struktur = f'{verkehr_aufforderungswort[0]} {aktuell}
{verkehr_aufforderungswort[1]}'
        satzteile_frage_live_struktur =
generate_satzteile(f'{verkehr_live_frage_struktur}')
        satzteile_forderung_live_struktur =
generate_satzteile(f'{verkehr_live_forderung_struktur}')
```

```
        satzteile_analyse = generate_satzteile(f' {kennwert}', f'{route_struktur}',
f'{datum_struktur}', f'{zeit_struktur}', f'{intervall_struktur}')
        analyse_struktur = random.sample(satzteile_analyse, 1)

        historisch_frage.append(f'{frage_monitoring} war{historisch_frage_struktur[0]}?')
        historisch_forderung.append(f'{verb} mir{historisch_forderung_struktur[0]}.')
        historisch_konjunktiv.append(f'Kannst du mir {verb.lower()}n
{frage_monitoring.lower()}{historisch_frage_struktur[0]} war?')

        live_frage.append(f'{frage_monitoring} ist {satzteile_frage_live_struktur[0]}?')
        live_forderung.append(f'{verb} mir {satzteile_forderung_live_struktur[0]}.')
        live_konjunktiv.append(f'Kannst du mir {verb.lower()}n {frage_monitoring.lower()}
{satzteile_frage_live_struktur[0]} ist?')

        analyse_frage.append(f'{frage_monitoring} war{analyse_struktur[0]}?')
        analyse_forderung.append(f'Zeige mir{analyse_struktur[0]}.')
        analyse_konjunktiv.append(f'Kannst du mir {verb.lower()}n
{frage_monitoring.lower()}{analyse_struktur[0]} war?')

    historisch = historisch_frage + historisch_forderung + historisch_konjunktiv
    live = live_frage + live_forderung + live_konjunktiv
    analyse = analyse_frage + analyse_forderung + analyse_konjunktiv

    return historisch, live, analyse

historisch_lst, live_lst, analyse_lst = generate_sentences(5000)

anfragen_df = pd.DataFrame(list(zip(historisch_lst, live_lst, analyse_lst)), columns
=['historisch', 'live', 'analyse'])

anfragen_stacked = pd.melt(anfragen_df)

anfragen_stacked['label'] = None
anfragen_stacked['target'] = None

anfragen_stacked.loc[anfragen_stacked['variable'] == 'historisch', 'label'] = 'Monitoring
Historisch'
anfragen_stacked.loc[anfragen_stacked['variable'] == 'live', 'label'] = 'Monitoring Live'
anfragen_stacked.loc[anfragen_stacked['variable'] == 'analyse', 'label'] = 'Analyse'

anfragen_stacked.loc[anfragen_stacked['label'] == 'Monitoring Historisch', 'target'] = 0
anfragen_stacked.loc[anfragen_stacked['label'] == 'Monitoring Live', 'target'] = 1
anfragen_stacked.loc[anfragen_stacked['label'] == 'Analyse', 'target'] = 2

### NER Daten
#### Tokenizer

model_name = 'bert-base-german-cased'
tokenizer = AutoTokenizer.from_pretrained(model_name, padding=True, truncation=True)
tokenizer_gottbert = AutoTokenizer.from_pretrained('uklfr/gottbert-base', padding=True,
truncation=True, from_pt = True)

tokens = []

for index, row in anfragen_stacked.iterrows():

    sentence = list(row['value'])
    last_char = sentence.pop(-1)

    sentence_new = ''.join(sentence)

    wort_lst = time_lst + date_lst + roads_lst + kennwerte_ner + intervall_lst

    contained_lst = [wort for wort in wort_lst if(wort in sentence_new)]

    sentence_lst = sentence_new.split()

    for elem in contained_lst:
        if ' ' in elem:
            elem_split = elem.split()
            idx_lst = []

            for splitted in elem_split:
                if splitted in sentence_lst:
                    idx_lst.append(sentence_lst.index(splitted))

            if len(idx_lst) == len(elem_split):
                sentence_lst[idx_lst[0]] = elem
                idx_lst.pop(0)

                for idx in sorted(idx_lst, reverse=True):
                    sentence_lst.pop(idx)
```

```
    sentence_lst.append(last_char)

    tokens.append(sentence_lst)

tokens_cleaned = []
for sentence in tokens:

    for elem in sentence:
        if elem[-1] == ',':
            elem_split = elem.rpartition(',')
            elem_new = list(elem_split[:-1])
            idx_elem = sentence.index(elem)
            sentence[idx_elem] = elem_new
            sentence = [word for sublist in sentence for word in (sublist if
isinstance(sublist, list) else [sublist])]
    tokens_cleaned.append(sentence)

anfragen_stacked['tokens'] = tokens

ner_label = []
ner_id = []

for index, row in anfragen_stacked.iterrows():

    elem_lst = row['tokens']

    tag_lst = []
    id_lst =[]
    for i in elem_lst:


        if (i in date_lst) or (i in time_lst) or (i in roads_lst) or (i in kennwerte_ner)
or (i in intervall_lst):

            tokenized_word = tokenizer_gottbert(i)
            token = tokenizer.convert_ids_to_tokens(tokenized_word["input_ids"])
            word_id = tokenized_word.word_ids()

            while(None in word_id):
                word_id.remove(None)

            length = len(word_id)

            if i in roads_lst:

                tag_lst.append('B-LOCATION')
                id_lst.append(1)

                for i in range(length - 1):
                    tag_lst.append('I-LOCATION')
                    id_lst.append(2)

            elif i in kennwerte_ner:

                tag_lst.append('B-KENNWERT')
                id_lst.append(3)

                for i in range(length - 1):
                    tag_lst.append('I-KENNWERT')
                    id_lst.append(4)

            elif i in time_lst:

                tag_lst.append('B-TIME')
                id_lst.append(5)

                for i in range(length - 1):
                    tag_lst.append('I-TIME')
                    id_lst.append(6)

            elif i in date_lst:

                tag_lst.append('B-DATE')
                id_lst.append(7)

                for i in range(length - 1):
                    tag_lst.append('I-DATE')
                    id_lst.append(8)

            elif i in intervall_lst:

                tag_lst.append('B-INTERVALL')
                id_lst.append(9)
```

```
                for i in range(length - 1):
                    tag_lst.append('I-INTERVALL')
                    id_lst.append(10)

        else:
            id_lst.append(0)
            tag_lst.append('O')

    ner_label.append(tag_lst)
    ner_id.append(id_lst)

anfragen_stacked['ner_tags'] = ner_id
anfragen_stacked['ner_label'] = ner_label

### EXPORT
anfragen_stacked.to_csv(csv_gottbert, sep = ';', index=False, encoding = 'utf-8')
```

10.1.2. Skripte Trainingsprozess

10.1.2.1. Training Sequence Model

```
# -*- coding: utf-8 -*-
@author: Anna Strobl 2023

## Imports
import tensorflow as tf
import pandas as pd
import os
import matplotlib.pyplot as plt
from tensorflow import keras
import datetime

from transformers import RobertaTokenizerFast, TFAutoModel,
TFAutoModelForSequenceClassification, TFRobertaForSequenceClassification, AutoTokenizer
from sklearn.model_selection import train_test_split
from keras.optimizers import Adam
from keras.losses import SparseCategoricalCrossentropy
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import RandomizedSearchCV
from keras.callbacks import History
#from keras_tuner import HyperModel, RandomSearch, GridSearch
#from keras_tuner.engine.hyperparameters import HyperParameters
#from keras_tuner import Objective
import seqeval
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score
import numpy as np
import evaluate
from sklearn.metrics import classification_report
import seqeval
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from itertools import chain
from collections import Counter
import seaborn as sns
from keras.callbacks import EarlyStopping
from deep_translator import GoogleTranslator

## Load Data
path = os.getcwd()
data = os.path.join(path, 'data')

csv_gottbert = os.path.join(path, 'dataset_gottbert.csv')

from google.colab import drive
drive.mount('/content/drive')

df = pd.read_csv('/content/drive/My Drive/data/dataset_gottbert.csv', sep = ';')
df = df.sample(frac=1, random_state = 32)
df.head()

df.tail()

## Load Model
model_checkpoint = 'uklfr/gottbert-base'

label2id = {'Monitoring Historisch': 0, 'Monitoring Live': 1, 'Analyse': 2}
id2label = {0: 'Monitoring Historisch', 1: 'Montitoring Live', 2: 'Analyse'}

sequence_model = TFRobertaForSequenceClassification.from_pretrained(model_checkpoint,
num_labels = 3, id2label=id2label, label2id=label2id, from_pt = True)
```

```
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint, padding=True, truncation=True,
from_pt = True)

run_number = 22

## Data Augumentation
#df_small = df.head(9000)

augument_data = df['value'].head(8000)
shuffle_data = df['value'].tail(8000)

translated_lst = []

for i in augument_data:

  translated = GoogleTranslator(source='auto', target='en').translate(text = i)
  translated_new = GoogleTranslator(source='auto', target='de').translate(text =
translated)

  translated_lst.append(translated_new)

shuffled_lst = []

for i in shuffle_data:
  sentence = i.split()
  random.shuffle(sentence)
  shuffled_sentence = ' '.join(sentence)

  shuffled_lst.append(shuffled_sentence)

value_lst = df['value'].values.tolist()

normal_data = value_lst[8000:]

translated_df = pd.DataFrame(translated_lst)

translated_df.to_csv('/content/drive/My Drive/data/translated_data.csv', sep = ';',
index=False, encoding = 'utf-8')

shuffled_df = pd.DataFrame(shuffled_lst)

shuffled_df.to_csv('/content/drive/My Drive/data/shuffled_data.csv', sep = ';',
index=False, encoding = 'utf-8')

#new_value_lst = translated_lst + normal_data + shuffled_lst
new_value_lst = translated_lst + normal_data

target_lst = df['target'].values.tolist()

new_df = pd.DataFrame()

new_df['value'] = new_value_lst
new_df['target'] = target_lst

new_df.tail()

#new_df = new_df.sample(frac=1, random_state = 32)

## Training and Test Data

x = df['value'].values
y = df['target'].values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state =
32)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2,
random_state=32)

### Plot Distribution

label_names = ['Monitoring Historisch', 'Monitoring Live', 'Analyse']

def show_categories(array):

    categories = np.unique(array, return_counts=True)
    dict_keys = categories[0]
    dict_values = categories[1]
    category_counts = dict(zip(dict_keys, dict_values))

    return category_counts

train_categories = show_categories(y_train)
test_categories = show_categories(y_test)
val_categories = show_categories(y_val)
```

```
fig, axes = plt.subplots(1, 2, figsize = (15, 5))
ax1, ax2 = axes.flatten()

ax1.bar(train_categories.keys(), train_categories.values(), tick_label = label_names, color
= 'lightskyblue', width = 0.5)
ax2.bar(test_categories.keys(), test_categories.values(), tick_label = label_names, color =
'lightskyblue', width = 0.5)
#ax3.bar(val_categories.keys(), val_categories.values(), tick_label = label_names, color =
'lightskyblue', width = 0.5)

ax1.set_title('Train Dataset')
ax2.set_title('Test Dataset')
#ax3.set_title('Validation Dataset')

ax1.set_xticklabels(label_names, rotation=45)
ax2.set_xticklabels(label_names, rotation=45)
#ax3.set_xticklabels(label_names, rotation=45)

plt.tight_layout()
fig.savefig('/content/drive/My Drive/results_sequence/distribution_' + str(run_number) +
'.png', dpi=fig.dpi)
plt.show()

## Tokenizer

x_train_tok = dict(tokenizer(x_train.tolist(), padding = 'max_length', truncation = True,
max_length = 50, return_tensors = 'tf'))
x_test_tok = dict(tokenizer(x_test.tolist(), padding = 'max_length', truncation = True,
max_length = 50, return_tensors = 'tf'))
x_val_tok = dict(tokenizer(x_val.tolist(), padding = 'max_length', truncation = True,
max_length = 50, return_tensors = 'tf'))

## Early Stopping

callback = EarlyStopping(monitor = 'accuracy', patience=2, mode='max',
restore_best_weights=True)


*******************************
## Hyperparameter Optimization
(Quelle: Chollet 2015)
*******************************
hyperparams = {
    'learning_rate': [1e-2, 1e-3, 1e-6],
    'batch_size': [16, 32, 48],
    'num_epochs': [4, 8, 12]
}

class MyHyperModel(HyperModel):
    def __init__(self, model):
        self.model = model

    def build(self, hp):

        learning_rate = hp.Choice('learning_rate', hyperparams['learning_rate'])
        batch_size = hp.Choice('batch_size', hyperparams['batch_size'])
        num_epochs = hp.Choice('num_epochs', hyperparams['num_epochs'])

        model_copy = self.model

        model_copy.compile(optimizer=Adam(learning_rate = learning_rate), loss=
SparseCategoricalCrossentropy(from_logits = True), metrics=['accuracy'])

        return model_copy

objective = Objective('val_accuracy', direction='max')

tuner_hp = HyperParameters()

hypermodel = MyHyperModel(sequence_model)

*******************************
### Random Search
(Quelle: Chollet 2015)
*******************************
tuner_random = RandomSearch(objective=objective, max_trials=10, hypermodel=hypermodel,
overwrite = True)

tuner_random.search(x_train_tok, y_train, validation_data=(x_val_tok, y_val))

random_hps = tuner_random.get_best_hyperparameters(num_trials=1)[0]
random_learning_rate = random_hps.get('learning_rate')
random_batch_size = random_hps.get('batch_size')
```

```
random_num_epochs = random_hps.get('num_epochs')

random_model = hypermodel.build(random_hps)

random_param_dict = {}
random_param_dict['learning_rate'] = float(random_learning_rate)
random_param_dict['batch_size'] = int(random_batch_size)
random_param_dict['num_epochs'] = int(random_num_epochs)
random_param_dict

random_param_df = pd.DataFrame(random_param_dict, index = [0])
random_param_df.to_csv('results_sequence/random_params_' + str(run_number) + '.csv', sep =
';', index=False, encoding = 'utf-8')
random_param_df
******************************
## Training

callback = EarlyStopping(monitor = 'accuracy', patience=1, mode='max',
restore_best_weights=True)

sequence_model.compile(optimizer=Adam(learning_rate = 1e-5), loss=
SparseCategoricalCrossentropy(from_logits = True), metrics=['accuracy'])

sequence_history = sequence_model.fit(x_train_tok, y_train, epochs =5, batch_size= 16,
validation_data = (x_val_tok, y_val), callbacks = [callback])

## Metrics
******************************
### Whole Model
(Quelle: HIRSCHLE 2022: 77)
******************************
fig, axes = plt.subplots(figsize =(7, 5))

train_acc = sequence_history.history['accuracy']
val_acc = sequence_history.history['val_accuracy']
train_loss = sequence_history.history['loss']
val_loss = sequence_history.history['val_loss']

epochs = range(1, len(train_acc) + 1)
plt.plot(epochs, train_loss, label='loss', color = 'gold')
plt.plot(epochs, train_acc, label='accuracy', color = 'lightskyblue')
plt.plot(epochs, val_acc, label='val_accuracy', color = 'cornflowerblue')
plt.plot(epochs, val_loss, label='val_loss', color = 'tomato')
plt.legend()
plt.tight_layout()
plt.xlabel('Epochs')
fig.savefig('/content/drive/My Drive/results_sequence/loss_accuracy_' + str(run_number) +
'.png', dpi=fig.dpi)

plt.show()
******************************
loss, accuracy = sequence_model.evaluate(x_test_tok, y_test)

evaluate_dict = {}
evaluate_dict['loss'] = loss
evaluate_dict['accuracy'] = accuracy

evaluate_df = pd.DataFrame(evaluate_dict, index = [0])
evaluate_df.to_csv('/content/drive/My Drive/results_sequence/evaluate_' + str(run_number) +
'.csv', sep = ';', index=False, encoding = 'utf-8')
evaluate_df

pred = sequence_model.predict(x_test_tok)
y_pred = np.argmax(pred['logits'], axis = -1)
y_logits = pred['logits']

precision = precision_score(y_test, y_pred, average=None, zero_division = 0)
recall = recall_score(y_test, y_pred, average=None, zero_division = 0)
f1 = f1_score(y_test, y_pred, average=None, zero_division = 0)

print(precision, recall, f1)

#### Label Metrics

target_names = ['Monitoring Historisch', 'Monitoring Live', 'Analyse']
labels = [0, 1, 2]

cl = classification_report(y_test, y_pred,target_names=target_names, labels = labels,
zero_division = 0, output_dict = True)
cl

cl_df = pd.DataFrame(cl)
cl_df_final = cl_df.head(3).transpose()
```

```
cl_df_final.to_csv('/content/drive/My Drive/results_sequence/cl_df_' + str(run_number) +
'.csv', sep = ';', index=False, encoding = 'utf-8')
cl_df_final

fig, ax = plt.subplots(figsize =(7, 5))
sns.heatmap(cl_df_final, annot=True, vmin = 0, vmax = 1, cmap= 'Blues', ax = ax)
plt.tight_layout()
fig.savefig('/content/drive/My Drive/results_sequence/cl_' + str(run_number) + '.png',
dpi=fig.dpi)
plt.show()

cm = confusion_matrix(y_test, y_pred, labels = labels, normalize = 'pred')
cm_df = pd.DataFrame(cm, columns = target_names)
cm_df.to_csv('/content/drive/My Drive/results_sequence/cm_' + str(run_number) + '.csv', sep
= ';', index=False, encoding = 'utf-8')

fig, ax = plt.subplots(figsize=(7, 5))

sns.heatmap(cm_df, annot=True, vmin = 0, vmax = 1, cmap= 'Blues', ax = ax,
xticklabels=target_names, yticklabels=target_names)
plt.yticks(rotation=0)
plt.tight_layout()
fig.savefig('/content/drive/My Drive/results_sequence/cm_' + str(run_number) + '.png',
dpi=fig.dpi)
plt.show()

## Export
tokenizer.save_pretrained('roberta_sequence' + str(run_number))
sequence_model.save_pretrained('roberta_sequence' + str(run_number))
```

10.1.2.2. Trainingsprozess Token Model

```
#!/usr/bin/env python
# coding: utf-8
@author: Anna Strobl 2023

# ## Imports

import tensorflow as tf
import pandas as pd
import os
import random
import keras
import ast
import matplotlib.pyplot as plt
import numpy as np

from transformers import AutoTokenizer, TFAutoModel, TFAutoModelForSequenceClassification,
TFAutoModelForTokenClassification, RobertaTokenizerFast, TFRobertaForTokenClassification
from sklearn.model_selection import train_test_split
from keras.optimizers import Adam
from keras.losses import SparseCategoricalCrossentropy
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import RandomizedSearchCV
from keras.callbacks import History
#from keras_tuner import HyperModel, RandomSearch
#from keras_tuner.engine.hyperparameters import HyperParameters
#from keras_tuner import Objective
import seqeval
import evaluate
#from seqeval.metrics import classification_report
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score,
classification_report
from itertools import chain
from collections import Counter
import seaborn as sns
from transformers import pipeline
from keras.callbacks import EarlyStopping

path = os.getcwd()
data = os.path.join(path, 'data')

csv_gottbert = os.path.join(path, 'dataset_gottbert.csv')

df = pd.read_csv(csv_gottbert, sep = ";")
df = df.sample(frac=1, random_state = 32)
df.head()

model_checkpoint = 'uklfr/gottbert-base'
```

```
label2id = {'O': 0, 'B-LOCATION': 1, 'I-LOCATION': 2, 'B-KENNWERT': 3, 'I-KENNWERT': 4,
            'B-TIME': 5, 'I-TIME': 6, 'B-DATE': 7, 'I-DATE': 8, 'B-INTERVALL': 9, 'I-
INTERVALL': 10}
id2label = {0: 'O', 1: 'B-LOCATION', 2: 'I-LOCATION', 3: 'B-KENNWERT', 4: 'I-KENNWERT',
            5: 'B-TIME', 6: 'I-TIME', 7: 'B-DATE', 8: 'I-DATE', 9: 'B-INTERVALL', 10: 'I-
INTERVALL'}

label_names = list(label2id.keys())

tokenizer = AutoTokenizer.from_pretrained(model_checkpoint, from_pt = True,
add_prefix_space=True)
#tokenizer = AutoTokenizer.from_pretrained(model_checkpoint, padding=True, truncation=True,
from_pt = True, add_prefix_space=True)

token_model = TFRobertaForTokenClassification.from_pretrained(model_checkpoint,
num_labels=11, id2label=id2label, label2id=label2id, from_pt = True)

### Create Train and Test Data

tokens_nested = [ast.literal_eval(cell) for cell in df['tokens']]
tags_nested = [ast.literal_eval(cell) for cell in df['ner_tags']]
labels_nested = [ast.literal_eval(cell) for cell in df['ner_label']]

df['tokens_new'] = tokens_nested
df['tags'] = tags_nested
df['label_categories'] = labels_nested

train, test = train_test_split(df, test_size=0.2, random_state = 32)

def show_categories(dataframe):

    categories = dataframe['label_categories'].values.tolist()
    category_count = dict(Counter(chain.from_iterable(categories)))

    return category_count

train_categories = show_categories(train)
test_categories = show_categories(test)

### Plot Data

fig, axes = plt.subplots(1, 2, figsize = (15, 5))
ax1, ax2 = axes.flatten()

ax1.bar(train_categories.keys(), train_categories.values(), color = 'lightskyblue')
ax2.bar(test_categories.keys(), test_categories.values(), color = 'lightskyblue')

ax1.set_title('Train Dataset')
ax2.set_title('Test  Dataset')

ax1.set_xticklabels(label_names, rotation = 90)
ax2.set_xticklabels(label_names, rotation = 90)

plt.tight_layout()
fig.savefig('results_token/distribution_' + str(run_number) + '.png', dpi=fig.dpi)
plt.show()

*****************************
### Tokenizer
(Quelle: WOLF ET AL. 2020)
*****************************
def align_labels_with_tokens(labels, word_ids):
    new_labels = []
    current_word = None
    for word_id in word_ids:
        if word_id != current_word:

            current_word = word_id
            label = -100 if word_id is None else labels[word_id]
            new_labels.append(label)
        elif word_id is None:

            new_labels.append(-100)
        else:

            label = labels[word_id]

            if label % 2 == 1:
                label += 1
            new_labels.append(label)

    return new_labels

def tokenizing_and_aligning(dataframe):
```

```
    tokenized_inputs = tokenizer(dataframe["tokens_new"].values.tolist(), truncation=True,
padding = True, max_length = 512, is_split_into_words=True, return_tensors = 'tf')

    all_labels = dataframe["tags"]
    new_labels = []
    for i, labels in enumerate(all_labels):
        word_ids = tokenized_inputs.word_ids(i)
        new_labels.append(align_labels_with_tokens(labels, word_ids))

    tokenized_inputs["labels"] = new_labels
    return tokenized_inputs

tokenized_train = tokenizing_and_aligning(train)
tokenized_test = tokenizing_and_aligning(test)

### Tokenize and Batch Datasets (vgl. Chollet 2015)
train_x = tf.data.Dataset.from_tensor_slices({
    "input_ids": tokenized_train["input_ids"],
    "attention_mask": tokenized_train["attention_mask"],
})

train_y = tf.data.Dataset.from_tensor_slices(tokenized_train["labels"])

train_dataset = tf.data.Dataset.zip((train_x, train_y))

test_x = tf.data.Dataset.from_tensor_slices({
    "input_ids": tokenized_test["input_ids"],
    "attention_mask": tokenized_test["attention_mask"],
})

test_y = tf.data.Dataset.from_tensor_slices(tokenized_test["labels"])

test_dataset = tf.data.Dataset.zip((test_x, test_y))

train_dataset_batched = train_dataset.batch(32)
test_dataset_batched = test_dataset.batch(32)

******************************

# ## Training

callback = EarlyStopping(monitor = 'accuracy', patience=2, mode='max',
restore_best_weights=True)

token_model.compile(optimizer = Adam(learning_rate = 5e-5), loss =
sparse_categorical_crossentropy(from_logits = True, ignore_class = -100), metrics =
'accuracy')

token_history = token_model.fit(train_dataset_batched, epochs=1, callbacks = [callback])

******************************
### Metrics
(Quelle: Wolf et al. 2020;)
******************************
******************************
##### Whole Model
(Quelle: Hirschle 2022)
******************************

fig, axes = plt.subplots(figsize =(7, 5))

train_acc = token_history.history['accuracy']

train_loss = token_history.history['loss']

epochs = range(1, len(train_acc)+1)
plt.plot(epochs, train_loss, label='loss', color = 'gold')
plt.plot(epochs, train_acc, label='accuracy', color = 'lightskyblue')
plt.legend()
plt.tight_layout()
plt.xlabel('Epochs')
fig.savefig('results_token/loss_accuracy_' + str(run_number) + '.png', dpi=fig.dpi)

plt.show()

******************************
### Metrics
(Quelle: Wolf et al. 2020;)
******************************
loss, accuracy = token_model.evaluate(test_dataset_batched)

evaluate_dict = {}
evaluate_dict['loss'] = loss
evaluate_dict['accuracy'] = accuracy
```

```
evaluate_df = pd.DataFrame(evaluate_dict, index = [0])
evaluate_df.to_csv('results_token/evaluate_' + str(run_number) + '1.csv', sep = ';',
index=False, encoding = 'utf-8')
evaluate_df

metric = evaluate.load("seqeval")

label_names = ['O', 'B-LOCATION', 'I-LOCATION', 'B-KENNWERT', 'I-KENNWERT',
            'B-TIME', 'I-TIME', 'B-DATE', 'I-DATE', 'B-INTERVALL', 'I-INTERVALL']
label_number = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

all_predictions = []
all_labels = []

pred = token_model.predict(test_dataset_batched)
logits = pred['logits']
labels = tokenized_test['labels']

predictions = np.argmax(logits, axis=-1)

for prediction, label in zip(predictions, labels):

    for predicted_idx, label_idx in zip(prediction, label):
        if label_idx == -100:
            continue
        all_predictions.append(label_names[predicted_idx])
        all_labels.append(label_names[label_idx])

metric.compute(predictions=[all_predictions], references=[all_labels], zero_division = 0)

#******************************
#### Label Metrics
(Quelle: WOLF ET AL. 2020;)
******************************
flat_labels = [item for sublist in labels for item in sublist]

flat_pred = [item for sublist in predictions for item in sublist]

#cl = classification_report(flat_labels,  flat_pred,target_names=label_names, labels =
label_number, zero_division = 0, output_dict = True)
cl = classification_report(flat_labels,  flat_pred,target_names=label_names, labels =
label_number, zero_division = 0, output_dict = True)

cl_df = pd.DataFrame(cl)
cl_df_final = cl_df.head(3).transpose()
cl_df_final.to_csv('results_token/cl_df_' + str(run_number) + '.csv', sep = ';',
index=False, encoding = 'utf-8')
cl_df_final

fig, ax = plt.subplots(figsize =(7, 5))
sns.heatmap(cl_df_final, annot=True, vmin = 0, vmax = 1, cmap= 'Blues', ax = ax)
plt.tight_layout()
fig.savefig('results_token/cl_' + str(run_number) + '.png', dpi=fig.dpi)
plt.show()

cm = confusion_matrix(flat_labels, flat_pred, labels = label_number, normalize = 'pred')
cm_df = pd.DataFrame(cm, columns = label_names)
cm_df.to_csv('results_token/cm_' + str(run_number) + '.csv', sep = ';', index=False,
encoding = 'utf-8')

fig, ax = plt.subplots(figsize=(7, 5))

sns.heatmap(cm_df, annot=True, vmin = 0, vmax = 1, cmap= 'Blues', ax = ax,
xticklabels=label_names, yticklabels=label_names)
plt.yticks(rotation=0)
plt.xticks(rotation=90)
plt.tight_layout()
fig.savefig('results_token/cm_' + str(run_number) + '.png', dpi=fig.dpi)
plt.show()
******************************
# ## Export

tokenizer.save_pretrained('roberta_token_' + str(run_number))
token_model.save_pretrained('roberta_token_' + str(run_number))
```

10.1.2.3. Training Token Model smartdata Corpus

```
# -*- coding: utf-8 -*-
```

```
******************************
# Gesamtes Skript
(Quelle: Wolf et al. 2020;)
******************************

"""## Imports"""

import tensorflow as tf
import pandas as pd
import os
import random
import keras
import ast
import matplotlib.pyplot as plt
import numpy as np

from transformers import AutoTokenizer, TFAutoModel, TFAutoModelForSequenceClassification,
TFAutoModelForTokenClassification, RobertaTokenizerFast, TFRobertaForTokenClassification,
AutoTokenizer
from sklearn.model_selection import train_test_split
from keras.optimizers import Adam
from keras.losses import SparseCategoricalCrossentropy
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import RandomizedSearchCV
from keras.callbacks import History
from keras_tuner import HyperModel, RandomSearch
from keras_tuner.engine.hyperparameters import HyperParameters
from keras_tuner import Objective
import seqeval
import evaluate
#from seqeval.metrics import classification_report
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score,
classification_report
from itertools import chain
from collections import Counter
import seaborn as sns
from datasets import load_dataset
from transformers import DataCollatorForTokenClassification
from transformers import pipeline
from keras.callbacks import EarlyStopping
from transformers import create_optimizer

dataset = load_dataset('smartdata')

model_checkpoint = 'uklfr/gottbert-base'

ner_feature = dataset['train'].features['ner_tags']
label_names = ner_feature.feature.names

id2label = {i: label for i, label in enumerate(label_names)}
label2id = {v: k for k, v in id2label.items()}

tokenizer = AutoTokenizer.from_pretrained(model_checkpoint, from_pt = True,
add_prefix_space=True)

token_model = TFRobertaForTokenClassification.from_pretrained(model_checkpoint,
num_labels=33, id2label=id2label, label2id=label2id, from_pt = True)

"""## Tokenizer"""

def align_labels_with_tokens(labels, word_ids):
    new_labels = []
    current_word = None
    for word_id in word_ids:
        if word_id != current_word:

            current_word = word_id
            label = -100 if word_id is None else labels[word_id]
            new_labels.append(label)
        elif word_id is None:

            new_labels.append(-100)
        else:

            label = labels[word_id]

            if label % 2 == 1:
                label += 1
            new_labels.append(label)

    return new_labels

def tokenize_and_align_labels(examples):
```

```
    tokenized_inputs = tokenizer(
        examples["tokens"], truncation=True, is_split_into_words=True
    )
    all_labels = examples["ner_tags"]
    new_labels = []
    for i, labels in enumerate(all_labels):
        word_ids = tokenized_inputs.word_ids(i)
        new_labels.append(align_labels_with_tokens(labels, word_ids))

    tokenized_inputs["labels"] = new_labels
    return tokenized_inputs

tokenized_datasets = dataset.map(
    tokenize_and_align_labels,
    batched=True,
    remove_columns=dataset["train"].column_names,
)

"""## Create Dataset"""

data_collator = DataCollatorForTokenClassification(
    tokenizer=tokenizer, return_tensors="tf"
)

tf_train_dataset = tokenized_datasets["train"].to_tf_dataset(
    columns=["attention_mask", "input_ids", "labels"],
    collate_fn=data_collator,
    shuffle=True,
    batch_size=16,
)

tf_eval_dataset = tokenized_datasets["validation"].to_tf_dataset(
    columns=["attention_mask", "input_ids", "labels"],
    collate_fn=data_collator,
    shuffle=False,
    batch_size=16,
)

tf_test_dataset = tokenized_datasets["test"].to_tf_dataset(
    columns=["attention_mask", "input_ids", "labels"],
    collate_fn=data_collator,
    shuffle=False,
    batch_size=16,
)

"""## Model"""

callback = EarlyStopping(monitor = 'accuracy', patience=2, mode='max',
restore_best_weights=True)

token_model.compile(optimizer=Adam(learning_rate = 1e-5), loss=
SparseCategoricalCrossentropy(from_logits = True, ignore_class = -100),
metrics=['accuracy'])

token_history = token_model.fit(tf_train_dataset, epochs=5, batch_size = 16,
validation_data = (tf_eval_dataset), callbacks = [callback])

*****************************
## Evaluation
(Quelle: WOLF ET AL. 2020;)
*****************************

fig, axes = plt.subplots(figsize =(7, 5))

train_acc = token_history.history['accuracy']
val_acc = token_history.history['val_accuracy']
train_loss = token_history.history['loss']
val_loss = token_history.history['val_loss']

epochs = range(1, len(train_acc) + 1)
plt.plot(epochs, train_loss, label='loss', color = 'gold')
plt.plot(epochs, train_acc, label='accuracy', color = 'lightskyblue')
plt.plot(epochs, val_acc, label='val_accuracy', color = 'cornflowerblue')
plt.plot(epochs, val_loss, label='val_loss', color = 'tomato')
plt.legend()
plt.tight_layout()
plt.xlabel('Epochs')
fig.savefig('/content/drive/My Drive/results_token/loss_accuracy_' + str(run_number) +
'.png', dpi=fig.dpi)

plt.show()

fig, axes = plt.subplots(figsize =(7, 5))
```

```
train_acc = token_history.history['accuracy']

train_loss = token_history.history['loss']

epochs = range(1, len(train_acc)+1)
plt.plot(epochs, train_loss, label='loss', color = 'gold')
plt.plot(epochs, train_acc, label='accuracy', color = 'lightskyblue')
plt.legend()
plt.tight_layout()
plt.xlabel('Epochs')
fig.savefig('/content/drive/My Drive/results_token/loss_accuracy_' + str(run_number) +
'.png', dpi=fig.dpi)

plt.show()

loss, accuracy = token_model.evaluate(tf_test_dataset)

evaluate_dict = {}
evaluate_dict['loss'] = loss
evaluate_dict['accuracy'] = accuracy

evaluate_df = pd.DataFrame(evaluate_dict, index = [0])
evaluate_df.to_csv('/content/drive/My Drive/results_token/evaluate_' + str(run_number) +
'1.csv', sep = ';', index=False, encoding = 'utf-8')
evaluate_df

metric = evaluate.load("seqeval")

labels = dataset["train"][0]["ner_tags"]
labels = [label_names[i] for i in labels]

predictions = labels.copy()
predictions[2] = "O"
metric.compute(predictions=[predictions], references=[labels])

all_predictions = []
all_labels = []
for batch in tf_eval_dataset:
    logits = token_model.predict_on_batch(batch)["logits"]
    labels = batch["labels"]
    predictions = np.argmax(logits, axis=-1)
    for prediction, label in zip(predictions, labels):
        for predicted_idx, label_idx in zip(prediction, label):
            if label_idx == -100:
                continue
            all_predictions.append(label_names[predicted_idx])
            all_labels.append(label_names[label_idx])

cl = metric.compute(predictions=[all_predictions], references=[all_labels])

cl

cl_df = pd.DataFrame(cl)
cl_df_final = cl_df.head(3).transpose()
cl_df_final.to_csv('/content/drive/My Drive/results_token/cl_df_' + str(run_number) +
'.csv', sep = ';', index=False, encoding = 'utf-8')
cl_df_final

fig, ax = plt.subplots(figsize =(7, 5))
sns.heatmap(cl_df_final, annot=True, vmin = 0, vmax = 1, cmap= 'Blues', ax = ax)
plt.tight_layout()
fig.savefig('/content/drive/My Drive/results_token/cl_' + str(run_number) + '.png',
dpi=fig.dpi)
plt.show()
*****************************
## Saving

from google.colab import drive
drive.mount("/content/gdrive")

tokenizer.save_pretrained('/content/gdrive/MyDrive/roberta_token_new')
token_model.save_pretrained('/content/gdrive/MyDrive/roberta_token_new')

model = TFRobertaForTokenClassification.from_pretrained('/content/drive/My
Drive/roberta_token_new')
```

10.1.3. Skripte Datenprozessierung Input-Daten

10.1.3.1. main.py

```
import apigateway.processing as processing
from fastapi import Body, FastAPI
from fastapi.encoders import jsonable_encoder
from fastapi.middleware.cors import CORSMiddleware
from fastapi.responses import JSONResponse

app = FastAPI()

# Enable CORS
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# root
@app.get("/")
def read_root() -> JSONResponse:
    return JSONResponse(status_code=200, content={"test": "system is running 😊"})

# input
@app.post("/input")
async def input(body: dict = Body(...)) -> JSONResponse:
    result = processing.analyse_text(body["request"])
    response: str = result[0]
    msg: str = result[1]['msg']
    status_code: int = result[1]['status_code']

    if status_code == 201:

        return JSONResponse(status_code=status_code, content=jsonable_encoder({"response":
response}))

    else:
        return JSONResponse(status_code=status_code, content=jsonable_encoder({"response":
msg}))
```

10.1.3.2. processing.py

```
'''
@author: Anna Strobl 2023
'''

import apigateway.post_processing as post_processing

def analyse_text(input: str) -> str:
    form_result, error_message = post_processing.process_data(input)

    return form_result, error_message
```

10.1.3.3. post_processing.py

```
'''
@author: Anna Strobl 2023
'''

import apigateway.functions as functions
import apigateway.model_analysis as model_analysis
from apigateway.model_analysis import forms_dict

status = {
            "status_code": 0,
            "msg": ""
        }

def process_data(text):

    # Get request form, form type, and entites
    form_type, form_result, dates, times, locations = model_analysis.analyse_text(text)

    form_result = forms_dict.get(form_type)
```

```
    # Check if Live
    # If no dates or times automatically Live View
    if (len(dates) == 0 and len(times) == 0) and form_type == 'Montitoring Live':

        form_result_final = forms_dict.get('Monitoring Live')
        form_type = 'Monitoring Live'

        status["status_code"] = 201
        status["msg"] = "Success"

        return form_result_final, status

    else:
        pass

    # Try to clean dates and time
    try:
        date_output = functions.get_dates(dates)
        time_output = functions.get_times(times)

    # If Error, Error Message
    except:

        # Error Message if Monitoring Historisch
        if form_type == 'Monitoring Historisch':

            form_result_final = None

            status["status_code"] = 422
            status["msg"] = "Incorrect number of date and time values or unreadable date
and time format provided. Please check the entered date and time and ensure that the
correct number and format of date and time values are provided."

            return form_result_final, status

        # Error Message if Analyse
        elif form_type == 'Analyse':

            form_result_final = None

            status["status_code"] = 422
            status["msg"] = "Incorrect number of date and time values or unreadable date
and time format provided. Please check the entered date and time and ensure that the
correct number and format of date and time values are provided."

            return form_result_final, status

    # Create usable Dates and Times
    date_ranges, dates = functions.date_stamps(date_output, form_type)
    time_ranges, times = functions.time_stamps(time_output)

    if len(date_ranges) == 0 or len(time_ranges) == 0:

        form_result_final = None

        status["status_code"] = 422
        status["msg"] = "Incorrect number of date and time values or unreadable date and
time format provided. Please check the entered date and time and ensure that the correct
number and format of date and time values are provided."

        return form_result_final, status

    elif (len(date_ranges) != 2 or len(time_ranges) != 2) and form_type == 'Analyse':

        form_result_final = None

        status["status_code"] = 422
        status["msg"] = "Incorrect number of date and time values or unreadable date and
time format provided. Please check the entered date and time and ensure that the correct
number and format of date and time values are provided."

        return form_result_final, status

    elif (len(date_ranges) < 1 or len(time_ranges) < 1) or (len(date_ranges) > 2 or
len(time_ranges) > 2)and form_type == 'Monitoring Historisch':

        form_result_final = None

        status["status_code"] = 422
        status["msg"] = "Incorrect number of date and time values or unreadable date and
time format provided. Please check the entered date and time and ensure that the correct
number and format of date and time values are provided."
```

```
        return form_result_final, status

    # Create Timestamps
    timestamps = functions.create_timestamps(date_ranges, form_type)

    # Fill request form with Dates and Times
    form_result = functions.fill_request_timestamp(timestamps, dates, times, form_result,
form_type)

    # Add Weekdays
    form_result = functions.set_weekdays(dates, form_result, form_type)

    # If one date and one time and form type is Monitoring Historisch successful
    if (len(date_output) == 1 and len(time_output) == 1) and form_type == 'Monitoring
Historisch':

        form_result_final = form_result

        status["status_code"] = 201
        status["msg"] = "Success"

        return form_result_final, status

    # If not fitting, continue
    else:

        pass

    # Try to get Locations
    try:

        location_output = functions.get_locations(locations)

    # If not enough resolvable Locations, Error Message
    except:

        pass

    if len(location_output) == 0 and form_type == 'Analyse':

        form_result_final = None

        status["status_code"] = 422
        status["msg"] = "Incorrect number of locations provided. Please check the entered
locations and ensure that the correct number of location values are provided."

        return form_result_final, status

    # If more or less than two dates and form type is Analyse Error Message
    if (len(date_output) != 2 or len(time_output) != 2) and form_type == 'Analyse':

        form_result_final = None

        status["status_code"] = 422
        status["msg"] = "Incorrect number of date and time values or unreadable date and
time format provided. Please check the entered date and time and ensure that the correct
number and format of date and time values are provided."

        return form_result_final, status

    # If enough pass
    else:

        pass

    # If Number of Locations not dividable by two, Error Message
    if len(location_output) % 2 !=0 or len(location_output) == 0:

        form_result_final = None

        status["status_code"] = 422
        status["msg"] = "Incorrect number of locations provided. Please check the entered
locations and ensure that the correct number of location values are provided."

        return form_result_final, status

    # If dividable by two, continue
    else:

        # Create iseful Locations
        location_lst = functions.prepare_locations(location_output, text)

        # Geocoding and Routing
        try:
```

```
            routing_id_results, routing_request_results =
functions.geocoding_routing(location_lst)

        except:
            routing_id_results = []
            routing_request_results = []

        # If Routing or Geocoding possible, Successful Analysis
        if routing_id_results != [] or routing_request_results != []:
            form_result_final = functions.fill_request_routes(form_result,
routing_id_results, routing_request_results)

            status["status_code"] = 201
            status["msg"] = "Success"

            return form_result_final, status

        # If Routing or Geocoding not possible, Error Message
        else:

            form_result_final = None

            status["status_code"] = 422
            status["msg"] = "Provided Route could not be resolved. Please check if the
provided locations are spelled correctly or exist in Munich."

            return form_result_final, status
```

10.1.3.4. model_analysis.py

```
'''
@author: Anna Strobl 2023
'''

#### IMPORTS ####
import os
import json

from dotenv import load_dotenv

from transformers import pipeline
from transformers import TFRobertaForTokenClassification,
TFRobertaForSequenceClassification
from transformers import AutoTokenizer, TFAutoModelForTokenClassification

load_dotenv()

# Load Models
path = os.getcwd()

sequence_checkpoint = 'ann-stro/roberta_sequence_3'

token_checkpoint = 'ann-stro/roberta_token_new'

# Request Paths
request_analyse = os.path.join(path, 'apigateway/analyse_request.json')

request_historisch = os.path.join(path, 'apigateway/historisch_request.json')

request_live = os.path.join(path, 'apigateway/live_request.json')

# Load Request Forms
form_analyse = open(request_analyse)
form_historisch = open(request_historisch)
form_live = open(request_live)

monitoring_historisch = json.load(form_historisch)
monitoring_live = json.load(form_live)
analyse =  json.load(form_analyse)

forms_dict = {'Monitoring Historisch': monitoring_historisch, 'Monitoring Live':
monitoring_live, 'Analyse': analyse}

#### MODELS and TOKENIZERS ####

access_token = os.getenv('HUGGING_TOKEN')

# Text Classification
tokenizer_sequence = AutoTokenizer.from_pretrained(sequence_checkpoint, padding = True,
Truncation = True, use_auth_token=access_token)
```

```
sequence_model = TFRobertaForSequenceClassification.from_pretrained(sequence_checkpoint,
use_auth_token=access_token)

pipe_sequence = pipeline('text-classification', model = sequence_model, tokenizer =
tokenizer_sequence)

# Token Classification
tokenizer_token = AutoTokenizer.from_pretrained(token_checkpoint, padding = True,
truncation = True, use_auth_token=access_token)

token_model = TFRobertaForTokenClassification.from_pretrained(token_checkpoint,
use_auth_token=access_token)

pipe_token = pipeline('ner', model = token_model, tokenizer = tokenizer_token)

#### ANALYSE ####
def analyse_text(text):
    # Text Classification
    sequence_result = pipe_sequence(text)

    for i in sequence_result:
        form_type = i.get('label')

    form_result = forms_dict.get(form_type)

    # Token Classification
    token_result = pipe_token(text)

    date_Entitäten = []
    time_Entitäten = []
    location_Entitäten = []

    for i in token_result:

        if 'DATE' in i['entity']:
            date_Entitäten.append(i)
        elif 'TIME' in i['entity']:
            time_Entitäten.append(i)
        elif 'LOCATION' in i['entity']:
            location_Entitäten.append(i)

    return form_type, form_result, date_Entitäten, time_Entitäten, location_Entitäten

form_analyse.close()
form_historisch.close()
form_live.close()
```

10.1.3.5. functions.py

```
'''
@author: Anna Strobl 2023
'''
import re
import pandas as pd
import os

from dotenv import load_dotenv

import requests
from supabase import create_client

from datetime import datetime
from datetime import timedelta
import locale

from apigateway.model_analysis import forms_dict

load_dotenv()

# Create Lookup for Months to transform "April" to 4
month_abk_str = {'Jän': '01', 'Jan': '01', 'Feb': '02', 'Mär': '03', 'Apr': '04', 'Mai':
'05', 'Jun': '06', 'Jul': '07', 'Aug': '08', 'Sep': '09', 'Okt': '10', 'Nov': '11', 'Dez':
'12'}
month_full_str = {'Jänner': '01', 'Januar': '01', 'Februar': '02', 'März': '03', 'April':
'04', 'Mai': '05', 'Juni': '06', 'Juli': '07', 'August': '08', 'September': '09',
'Oktober': '10', 'November': '11', 'Dezember': '12'}
month_str = month_full_str | month_abk_str

# Get Dates from Date Entitäten
def get_dates(date_input):
```

```
    date_result = []
    date_lst = []
    entity_lst = []

    for i in date_input:

        result_lst = []

        date = i['word']
        entity = i['entity']

        date_lst.append(date)
        entity_lst.append(entity)

        for dat, ent in zip(date_lst, entity_lst):

            dat_clean = dat.replace('Ġ', '').replace('Ãł', 'ß').replace('Ã„',
'Ä').replace('Ãż', 'ä').replace('Ã–', 'Ö').replace('Ã¶', 'ö').replace('Ãœ',
'Ü').replace('Ã¼', 'ü').replace('Ġ', '')

            if ((not dat_clean.isdigit()) and (dat_clean in month_str)) or
(dat_clean.isdigit()):

                month_trans = month_str.get(dat_clean)

                if not month_trans == None:
                    dat_clean = month_trans + '.'

                else:
                    pass

            dat_strip = re.sub(r'[a-zA-Z\s]+', '', dat_clean)

            if ent.startswith('B'):
                result_lst.append(dat_strip)

            else:
                if result_lst:
                    result_lst[-1] += '' + dat_strip

    for i in result_lst:
        i_new = re.sub(r'[a-zA-Z\s]+', '', i)
        date_result.append(i_new)

    while '' in date_result:
        date_result.remove('')

    return date_result

def get_times(time_input):
    time_result = []
    time_lst = []
    entity_lst = []

    for i in time_input:

        result_lst = []

        time = i['word']
        entity = i['entity']

        time_lst.append(time)
        entity_lst.append(entity)

        for tim, ent in zip(time_lst, entity_lst):

            tim_clean = re.sub(r'[^\d\s]', '', tim)

            tim_strip = tim_clean.replace('Ġ', '')

            if ent.startswith('B'):
                result_lst.append(tim_strip)

            else:
                if result_lst:
                    result_lst[-1] += '' + tim_strip

    for j in result_lst:
        time_result.append(j)

    return time_result

def get_locations(location_input):
    location_result = []
```

```
    location_lst = []
    entity_lst = []

    for i in location_input:

        result_lst = []

        location = i['word']
        entity = i['entity']

        location_lst.append(location)
        entity_lst.append(entity)

        for loc, ent in zip(location_lst, entity_lst):

            if ent.startswith('B'):
                result_lst.append(loc)

            else:
                if result_lst:
                    result_lst[-1] += ' ' + loc

    for j in result_lst:
        j_clean = j.replace('Ã‡', 'ß').replace('Ã„', 'Ä').replace('Ã¤', 'ä').replace('Ã–',
'Ö').replace('Ã¶', 'ö').replace('Ãœ', 'Ü').replace('Ã¼', 'ü').replace('Ġ', '')
        j_final = re.sub(r"(\w)([A-Z])", r"\1 \2", j_clean)
        location_result.append(j_final)

    return location_result

date_formats_y = ['%y', '%#y', '%Y']
date_formats_m = ['%m', '%#m']
date_formats_d = ['%d', '%#d']

date_combinations = []

for y in date_formats_y:
    for m in date_formats_m:
        for d in date_formats_d:
            date_combinations.append((d + '.' + m + '.' + y))

for m in date_formats_m:
    for d in date_formats_d:
        date_combinations.append((d + '.' + m))
        date_combinations.append((d + '.' + m + '.'))

def date_stamps(date_input, form_type):

    date_ranges = []
    date_lst = []
    dates = []

    today = datetime.now()

    for i in date_input:

    # Find form of Date Input

        for com in date_combinations:

            if len(date_lst) <= 2 and form_type == 'Analyse':

                try:

                    date_format = datetime.strptime(i, com)

                    if date_format.year >= 1900:

                        date_lst.append(date_format)

                    else:
                        pass

                except ValueError:
                    pass

            elif len(date_lst) <= 1 and form_type == 'Monitoring Historisch':
                try:

                    date_format = datetime.strptime(i, com)
                    if date_format.year >= 1900:
                        date_lst.append(date_format)

                    else:
```

```
                        pass

                except ValueError:
                    pass

            else:

                pass

        # If Date was not readable return with empty list
        if len(date_lst) == 0:

            date_ranges = []
            dates = []

            return date_ranges, dates

        else:
            pass

        # If Date under 1900 or Date after today, return with empty list
        for m in date_lst:

            if m.year < 1900 or m > today:

                date_ranges = []
                dates = []

                return date_ranges, dates

            else:
                pass

        # If one year or both are wrong or not provided change them to 2023
        if len(date_lst) == 2 and form_type == 'Analyse':

            if (date_lst[0].year == 1900) and (date_lst[1].year == 1900):

                current_year = datetime.now().year
                date_lst[0] = datetime(current_year, date_lst[0].month, date_lst[0].day,
tzinfo=date_lst[0].tzinfo)
                date_lst[1] = datetime(current_year, date_lst[1].month, date_lst[1].day,
tzinfo=date_lst[1].tzinfo)

            elif (date_lst[0].year == 1900) or (date_lst[1].year == 1900):

                for i, dt in enumerate(date_lst):
                    if dt.year != 1900:
                        year = dt.year

                        try:

                            date_lst[i + 1] = dt.replace(year=year)

                        except:
                            date_lst[i - 1] = dt.replace(year=year)

                        break

            else:

                pass

        # If no year or wrong year change it to current year
        if form_type == 'Monitoring Historisch':

            if (date_lst[0].year == 1900):

                current_year = datetime.now().year
                date_lst[0] = datetime(current_year, date_lst[0].month, date_lst[0].day,
tzinfo=date_lst[0].tzinfo)

            else:

                pass


            dates.append(date_lst[0])
            formatted_date = date_lst[0].strftime('%Y-%m-%dT')

            date_ranges.append(formatted_date)


    for i in date_lst:
```

```
        formatted_date = i.strftime('%Y-%m-%dT')
        dates.append(i)
        date_ranges.append(formatted_date)

    date_ranges = sorted(date_ranges, key=lambda x: datetime.strptime(x, '%Y-%m-%dT'))

    return date_ranges, dates

time_formats_h = ['%I', '%#I', '%H', '%#H']
time_formats_m = ['%M', '%#M']
time_formats_s = ['%S', '%#S']

time_combinations = []

for h in time_formats_h:
    for m in time_formats_m:
        for s in time_formats_s:
            time_combinations.append((h + m + s))

for h in time_formats_h:
    for m in time_formats_m:
        time_combinations.append((h + m))

for h in time_formats_h:
    time_combinations.append((h))

def time_stamps(time_input):

    time_ranges = []
    times = []
    time_formats = []

    for i in time_input:

        for com in time_combinations:

            try:

                time_format = datetime.strptime(i, com)
                time_str = time_format.strftime('%H:%M:%S')
                time_formats.append(time_format)

            except ValueError:
                pass

        if len(time_formats) == 0:

            time_ranges = []
            times = []

            return time_ranges, times

        else:
            pass

        # If Time was not readable return with empty list
        formatted_time = time_format.strftime('%H:%M:%S.%f')[:-3] + 'Z'

        time_ranges.append(formatted_time)
        times.append(time_str)

    return time_ranges, times

def create_timestamps(dates, form_type):
    time_parameters = []

    start = '00:00:00'
    end = '23:59:59'

    start_time = datetime.strptime(start, '%H:%M:%S')
    end_time = datetime.strptime(end, '%H:%M:%S')

    start_str = start_time.strftime('%H:%M:%S.%f')[:-3] + 'Z'
    end_str = end_time.strftime('%H:%M:%S.%f')[:-3] + 'Z'

    if form_type == 'Monitoring Historisch':
        gesamt_date = dates[0]

        start_stamp = gesamt_date + start_str
        end_stamp = gesamt_date + end_str

    elif form_type == 'Analyse':
        start_date = dates[0]
        end_date = dates[1]
```

```
        start_stamp = start_date + start_str
        end_stamp = end_date + end_str

    time_parameters.append(start_stamp)
    time_parameters.append(end_stamp)

    return time_parameters

def fill_request_timestamp(time_parameters, dates, times, form_result, form_type):

    # fill time parameters
    if form_type == 'Monitoring Historisch':

        form_result['timeParameters'][0]['start'] = time_parameters[0]
        form_result['timeParameters'][0]['end'] = time_parameters[1]

    elif form_type == 'Analyse':

        form_result['analysisRequest']['timeParameters'][0]['start'] = time_parameters[0]
        form_result['analysisRequest']['timeParameters'][0]['end'] = time_parameters[1]

    # fill times
    if (len(times) == 1) and (form_type == 'Monitoring Historisch'):

        form_result['timeParameters'][0]['timeRanges'][0]['from'] = times[0]
        form_result['timeParameters'][0]['timeRanges'][0]['to'] = times[0]

    elif (len(times) == 2) and (form_type == 'Analyse'):

        form_result['analysisRequest']['timeParameters'][0]['timeRanges'][0]['from'] =
times[0]
        form_result['analysisRequest']['timeParameters'][0]['timeRanges'][0]['to'] =
times[1]

    return form_result

def set_weekdays(dates, form_result, form_type):
    weekdays_num = []

    if len(dates) == 2:
        dates.sort()
        start = dates[0]
        end = dates[1]

        delta = end - start

        for i in range(delta.days + 1):
            day = start + timedelta(days=i)

            weekdays_num.append(day.weekday())

        weekdays_num.sort()
        weekdays_num = set(weekdays_num)

    elif len(dates) == 1:

        weekdays_num.append(dates[0].weekday())

    weekday_dict = {0: 'MONDAY', 1: 'TUESDAY', 2: 'WEDNESDAY', 3: 'THURSDAY',
                    4: 'FRIDAY', 5: 'SATURDAY', 6: 'SUNDAY'}

    weekdays = [weekday_dict.get(item) for item in weekdays_num]

    if form_type == 'Monitoring Historisch':
        form_result['timeParameters'][0]['dayCategories'] = weekdays

    elif form_type == 'Analyse':
        form_result['analysisRequest']['timeParameters'][0]['dayCategories'] = weekdays

    return form_result

def prepare_locations(location_result, text):

    location_tuples = list(zip(location_result[::2], location_result[1::2]))

    point_lst = []

    for tuple in location_tuples:

        pairs = list(tuple)

        point_lst.append(pairs)
```

```
    return point_lst

def geocoding_routing(point_lst):

    url_supabase = os.getenv('SUPABASE_URL')

    key_supabase = os.getenv('SUPABASE_KEY')

    email_supabase = os.getenv('SUPABASE_EMAIL')

    password_supabase = os.getenv('SUPABASE_PASSWORD')

    supabase = create_client(url_supabase, key_supabase)

    response = supabase.auth.sign_in_with_password({'email': email_supabase, 'password':
password_supabase})

    token = response.session.access_token

    routing_id_url = os.getenv('ROUTING_ID_URL')
    routing_geojson_url = os.getenv('ROUTING_GEOJSON_URL')
    geocoding_url = os.getenv('GEOCODING_URL')

    headers = {
        'Authorization': 'Bearer' + ' ' + token
    }

    routing_id_results = []
    routing_request_results = []

    for i in point_lst:

        routing_request = {
            "routingPoints": [
                {
                    "lng": 0,
                    "lat": 0
                },
                {
                    "lng": 0,
                    "lat": 0
                }
            ]
        }

        routes_id = []

        request_start = geocoding_url + 'api?q=' + i[0] + '&limit=1&layer=street'
        response_start = requests.get(request_start)
        response_start_json = response_start.json()

        request_end = geocoding_url + 'api?q=' + i[1] + '&limit=1&layer=street'
        response_end = requests.get(request_end)
        response_end_json = response_end.json()

        longitude_start = response_start_json['features'][0]['geometry']['coordinates'][0]
        latitude_start = response_start_json['features'][0]['geometry']['coordinates'][1]

        longitude_end = response_end_json['features'][0]['geometry']['coordinates'][0]
        latitude_end = response_end_json['features'][0]['geometry']['coordinates'][1]

        routing_request['routingPoints'][0]['lng'] = longitude_start
        routing_request['routingPoints'][0]['lat'] = latitude_start
        routing_request['routingPoints'][1]['lng'] = longitude_end
        routing_request['routingPoints'][1]['lat'] = latitude_end

        routing_id = requests.post(routing_id_url, json = routing_request, headers =
headers)
        routing_id_json = routing_id.json()

        for i in routing_id_json['results']:
                routes_id.append(i.get('id'))

        routing_id_results.append(routes_id)
        routing_request_results.append(routing_request)

    return routing_id_results, routing_request_results

def fill_request_routes(form_result, routing_id_results, routing_request_results):

    form_result['analysisRequest']['routes'] = []

    for i in routing_id_results:
```

```
        form_result['analysisRequest']['routes'].append(i)

    form_result['routesRequest'] = []

    for i in routing_request_results:
        form_result['routesRequest'].append(i)

    return form_result
```

10.2. Request Forms (JSON-Files)

10.2.1. Monitoring Live

```
{
  "requestType": "Live"
}
```

10.2.2. Monitoring Historisch

```
{
  "requestType": "Historic",
  "timeParameters": [
    {
      "start": "",
      "end": "",
      "timeRanges": [
        {
          "from": "",
          "to": ""
        }
      ],
      "dayCategories": [
      ]
    }
  ]
}
```

10.2.3. FCD-Verkehrsanalyse

```
{
  "requestType": "Analysis",
  "routesRequest": [],
  "analysisRequest": {
    "analysisType": "fcdAnalysis",
    "valueTypes": [
      "v"
    ],
    "mode": "avg",
    "interval": "min_15",
    "aggregate": false,
    "timeParameters": [
      {
        "start": "",
        "end": "",
        "timeRanges": [
          {
            "from": "",
            "to": ""
          }
        ],
        "dayCategories": [
        ]
      }
    ],
    "routes": [
    ]
  }
}
```

10.3. Studie

10.3.1. Fragebögen

10.3.1.1. Screener

Teilnehmer Fragebogen

Durch das Ausfüllen dieses Fragebogens erhalten wir wichtige Informationen über Sie als User*in.

* Indicates required question

Demographische Fragen

1. In welcher Altersgruppe befinden Sie sich? *

   Mark only one oval.

   - ◯ unter 20 Jahren
   - ◯ 20 - 30 Jahre
   - ◯ 31 - 40 Jahre
   - ◯ 41 - 50 Jahre
   - ◯ 51 - 60 Jahre
   - ◯ über 60 Jahren

2. Welchem Geschlecht fühlen Sie sich zugeordnet? *

   Mark only one oval.

   - ◯ Männlich
   - ◯ Weiblich
   - ◯ Divers
   - ◯ Keine Angabe

Fragen zur Experience

3. Wie gut sind Ihre Ortskenntnisse der Stadt München? *

Mark only one oval.

Sehr schlecht

1 ◯

2 ◯

3 ◯

4 ◯

Sehr gut

4. Wie erfahren sind Sie im Umgang mit Computern und dem Internet? *

Mark only one oval.

Sehr wenig erfahren

1 ◯

2 ◯

3 ◯

4 ◯

Sehr erfahren

10.3.1.2. Fragebogen zur Studie

Studie Fragebogen

Bitte beantworten Sie die folgenden Fragen zum Vergleich der beiden getesteten Systeme.

* Indicates required question

1. Das textgesteuerte Interface ist _____ als das Button-gesteuerte Interface. *

Mark only one oval per row.

	Stimme nicht zu	Stimme eher nicht zu	Stimme eher zu	Stimme zu
einfacher zu verwenden	◯	◯	◯	◯
einfacher zu erlernen	◯	◯	◯	◯
angenehmer zu verwenden	◯	◯	◯	◯
übersichtlicher	◯	◯	◯	◯
intuitiver	◯	◯	◯	◯

2. Mit dem textgesteuerten Interface _____ als mit dem Button-gesteuerten Interface. *

Mark only one oval per row.

	Stimme nicht zu	Stimme eher nicht zu	Stimme eher zu	Stimme zu
ist die Interaktion natürlicher	◯	◯	◯	◯
empfinde ich die Lösung der Tasks schneller	◯	◯	◯	◯
ist die Lösung der Tasks effizienter	◯	◯	◯	◯
ist die Lösung der Tasks in weniger Schritten möglich	◯	◯	◯	◯
könnte ich schneller produktiv sein	◯	◯	◯	◯

3. Im textgesteuerten Interface _____ als im Button-gesteuerten Interface. *

Mark only one oval per row.

	Stimme nicht zu	Stimme eher nicht zu	Stimme eher zu	Stimme zu
konnte ich falsche Ergebnisse schneller korrigieren	◯	◯	◯	◯
sind die Funktionen besser integriert	◯	◯	◯	◯

4. Das textgesteuerte Interface ist für mich eine Alternative zum Button-gesteuerten Interface. *

Mark only one oval.

Stimme nicht zu

1 ◯

2 ◯

3 ◯

4 ◯

Stimme zu

5. Wenn das textgesteuerte Interface eine Alternative zum Button-gesteuerten Interface für Sie ist, begründen Sie, wieso. Falls dies nicht zutrifft, begründen Sie ebenfalls. *

6. Was hat Ihnen bei der Verwendung des textgesteuertes Interface im Vergleich zum Button-gesteuerten Interface gefallen, und was nicht? *

7. Haben Ihnen bei der Verwendung des textgesteuerten Interfaces Funktionen gefehlt, oder haben Sie Verbesserungsvorschläge? Wenn ja, schreiben Sie diese hier auf. *

8. Wenn Sie noch weitere Anmerkungen haben, können Sie diese hier mitteilen.

10.3.2. Ergebnisse

10.3.2.1. Screener

Teilnehmer:innen	*In welcher Altersgruppe befinden Sie sich?*	*Welchem Geschlecht fühlen Sie sich zugeordnet?*	*Wie gut sind Ihre Ortskenntnisse der Stadt München?*	*Wie erfahren sind Sie im Umgang mit Computern und dem Internet?*
1	31 - 40 Jahre	Männlich	4	4
2	31 - 40 Jahre	Weiblich	3	4
3	20 - 30 Jahre	Weiblich	4	3
4	41 - 50 Jahre	Männlich	1	4
5	20 - 30 Jahre	Weiblich	4	4
6	31 - 40 Jahre	Männlich	3	3
7	31 - 40 Jahre	Weiblich	1	3
8	31 - 40 Jahre	Weiblich	1	4
9	31 - 40 Jahre	Männlich	1	4

10.3.2.2. Ergebnisse „Time on Task“ des buttongesteuerten Web-GIS (initial und mit Fehlern)

Teilnehmer:innen	*Task 1*	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*
1	12	1	56	33	61	72
2	15	1	68	39	61	64
3	11	1	50	37	52	59
4	11	1	42	27	44	52
5	41	1	59	56	50	51
6	12	1	57	29	41	45
7	19	1	74	37	51	48
8	18	1	43	38	51	78
9	21	1	55	49	77	67
10	22	1	36	40	51	53

10.3.2.3. Ergebnisse „Time on Task“ des textgesteuerten Web-GIS (initial)

Teilnehmer:innen	*Task 1*	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*
1	18	10	50	26	36	72
2	8	12	49	32	67	34
3	13	9	55	36	57	72
4	11	12	65	41	85	41
5	14	7	37	41	48	56
6	18	19	56	38	36	52
7	14	11	80	56	69	80
8	14	14	75	68	80	84
9	21	12	93	47	46	54
10	18	5	56	39	46	51

10.3.2.4. Ergebnisse „Time on Task“ des textgesteuerten Web-GIS (mit Fehlern)

Teilnehmer:innen	*Task 1*	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*
1	18	10	89	59	137	107
2	8	12	49	32	103	34
3	13	9	106	36	57	72
4	11	12	65	41	136	41
5	45	7	37	41	59	56
6	18	19	56	38	36	52
7	14	11	80	56	69	80
8	14	14	75	68	85	93
9	21	12	93	47	115	182
10	18	5	56	39	46	220

10.3.2.5. Ergebnisse „Task Success“ des buttongesteuerten Web-GIS

Teilnehmer:innen	*Task 1*	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*
1	4	4	4	4	4	3
2	4	4	3	4	4	4
3	4	4	4	4	4	4
4	4	4	4	4	4	3
5	4	4	4	4	4	3
6	4	4	4	4	4	3
7	4	4	4	4	4	3
8	4	4	4	4	3	3
9	4	4	4	4	4	3
10	4	4	4	4	4	3

10.3.2.6. Ergebnisse „Task Success“ des textgesteuerten Web-GIS

Teilnehmer:innen	*Task 1*	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*
1	4	4	1	2	1	2
2	4	4	3	4	2	4
3	4	4	1	4	4	4
4	4	4	4	4	1	4
5	2	4	4	4	2	4
6	4	4	4	4	4	3
7	4	4	4	4	4	4
8	4	4	4	4	2	2
9	4	4	4	4	2	0
10	4	4	4	4	3	1

10.3.2.7. Ergebnisse des Fragebogens von der Bewertung der Aussagen auf der Likert-Skala

	Frage 1	Frage 2	Frage 3	Frage 4	Frage 5	Frage 6	Frage 7	Frage 8	Frage 9	Frage 10	Frage 11	Frage 12	Frage 13
Teilnehmer:innen	**einfacher zu verwenden**	**einfacher zu erlernen**	**angenehmer zu verwenden**	**übersichtlicher**	**intuitiver**	**Interaktion natürlicher**	**Lösung schneller**	**Lösung effizienter**	**weniger Schritte**	**schneller produktiv**	**schneller korrigieren**	**Funktionen besser integrier**	**Alternative**
1	Stimme zu	Stimme zu	Stimme eher zu	Stimme eher zu	Stimme zu	Stimme eher zu	Stimme eher nicht	Stimme eher nicht	Stimme zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher zu	Stimme zu
2	Stimme eher zu	Stimme eher nicht zu	Stimme eher zu	Stimme eher zu	Stimme eher zu	Stimme eher zu	Stimme eher nicht	Stimme eher nicht	Stimme eher zu	Stimme eher zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher zu
3	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher nicht	Stimme zu	Stimme eher nicht	Stimme eher nicht	Stimme zu	Stimme eher zu	Stimme eher nicht zu	Stimme eher zu	Stimme zu
4	Stimme eher zu	Stimme eher zu	Stimme eher nicht zu	Stimme zu	Stimme eher zu	Stimme eher zu	Stimme nicht zu	Stimme nicht zu	Stimme eher zu	Stimme nicht zu	Stimme nicht zu	Stimme eher nicht zu	Stimem eher nicht zu
5	Stimme zu	Stimme zu	Stimme eher zu	Stimme zu	Stimme eher	Stimme zu	Stimme zu	Stimme eher	Stimme zu	Stimme zu	Stimme zu	Stimme zu	Stimme eher zu
6	Stimme zu	Stimme eher zu	Stimme eher zu	Stimme eher zu	Stimme eher nicht	Stimme eher zu	Stimme eher nicht	Stimme eher zu	Stimme eher zu	Stimme eher zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher zu
7	Stimme eher nicht zu	Stimme eher zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme zu	Stimme eher zu	Stimme eher nicht	Stimme eher zu	Stimme zu	Stimme zu	Stimme eher nicht zu	Stimme eher zu	Stimme eher zu
8	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme zu	Stimme eher nicht	Stimme eher nicht	Stimme nicht zu	Stimme eher nicht	Stimme eher zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher zu	Stimme eher zu
9	Stimme eher zu	Stimme zu	Stimme eher zu	Stimme eher nicht zu	Stimme eher zu	Stimme zu	Stimme zu	Stimme zu	Stimme zu	Stimme zu	Stimme eher nicht zu	Stimme nicht zu	Stimme eher zu
10	Stimme eher nicht zu	Stimme eher zu	Stimme eher zu	Stimme zu	Stimme eher nicht	Stimme eher zu	Stimme eher zu	Stimme eher nicht	Stimme eher zu	Stimme eher nicht zu	Stimme eher nicht zu	Stimme eher zu	Stimme eher zu

10.3.2.8. Fragebogen offene Fragen

Offene Frage 1
Wenn das textgesteuerte Interface eine Alternative zum Button-gesteuerten Interface für Sie ist, begründen Sie, wieso. Falls dies nicht zutrifft, begründen Sie ebenfalls.
schnelle Eingabe ohne Funktionen zu suchen.
Nur ein Textfeld, daher sehr einfach; Eingaben entsprechend auch der eigenen Denkweise, man muss weniger rumklicken
Die Eingabe einer Anfrage ist flexibler und individuell gestaltbar.
Für eine gelegentliche Nutzung (d.h. nicht erfahrener User) ist das textgesteuerte Interface eine tolle Alternative. Sobald man aber das Button-gesteuerte Interface gewohnt ist, sind die Eingaben der gewünschten Bereiche/Zeiten wesentlich schneller durchzuführen als im textgesteuerten Interface (außer vielleicht, wenn man sich Textblöcke für späteres Copy-Paste speichert). Nachteil im textgesteuerten Interface: gleichnamige Straßen in verschiedenen Gemeinden können nicht differenziert werden.
Ich war erst kurzzeitig hilflos, da nur das Textfeld eingeblendet war und ich es als solches erst nicht erkannt habe. Die Buttons haben mir beim ersten Task "schneller gesagt" was ich wo tun muss, auch wenn die Nutzung des Textfeldes nach dem ersten Task natürlich schneller ging
Intuitive Eingabe der Suche im Freitextfeld. Dennoch steht der Nutzer:in beim Freitext vor der Herausforderung, in welche Form (Frage, Aussage) die Suchanfrage beschrieben wird. Bei dem Button-gesteuerten Interface sind die Eingabemöglichkeiten bereits vordefiniert (Strecke, Zeit, Kennwert). Somit ist eine eigenständige Formulierung im textgesteuerten Interface überflüssig und daher die Führung durch das Interface bereits vordefiniert und strukturiert.
Das Textgesteuuerte Interface ist einfach und schnell zu bedienen
wenn Spraheingaben ermöglich werden wäre das deutlich schneller und effizienter. Aktuell Dauer die Texteingabe ein wenig
Das textgesteuerte Interface ermöglicht auf jeden Fall effizientes, schnelles Arbeiten. Allerdings ist die Überprüfbarkeit und Fehlererkennung (zB bei der Auswahl der Straßen) nicht so gegeben wie bei der Button gesteuerten Variante. Deshabl würd ich als Experten-tool eher die Button gesteuerte Variante vorziehen.
Die Eingabe im textgesteuerten Interface wirkt natürlicher und trägt bei entsprechendem Wissen zum Funktionsumfang zu schnelleren Ergebnissen bei.
Offene Frage 2
Was hat Ihnen bei der Verwendung des textgesteuertes Interface im Vergleich zum Button-gesteuerten Interface gefallen, und was nicht?
gefallen: 1. einfache Eingabe ohne die Funktionen (Messkenngrößen) zu kennen nicht gefallen: 1. Fehler wenn Straßen oder Datumsformat nicht richtig eingegeben wurde. 2. Unsicherheit, ob Straßenname dem richtigen Ort zugeteilt ist (z.B. Falkenstraße in Karlsfeld, Gräfeling, Aschheim...)
S.o.
S.o. die flexiblere Eingabe meiner Anfrage. Allerdings vertippt man sich auch schnell und überfliegt deshalb den Fließtext vor der Eingabe zur Sicherheit noch einmal. Außerdem hat man keine "Hilfestellung" in Form von z.B. der Kalenderanzeige. Insgesamt ist das textgesteuerte Interface gewöhnungsbedürftiger, bei regelmäßiger Nutzung aber mit Sicherheit intuitiver und angenehmer, je nach Suchanfrage.
Gefallen: Intuitive Herangehensweise (leider hat das Studiensetting die Formulierung quasi schon vorgegeben, es war somit nicht eruierbar, ob ein freier Text zu den gewünschten Ergebnissen geführt hätte). Nicht gefallen: Keine Hinweise im User Interface, wie man Zeiträume mit Uhrzeit korrekt formulieren muss. Ohne Hilfestellung der Studiendurchführenden hätte es durch Trial and Error sehr lange dauern können, bis zum gewünschten Ergebnis zu kommen.
siehe Antwort der Frage davor

Bei dem Button-gesteuerten Interface sind die Eingabemöglichkeiten bereits vordefiniert (Strecke, Zeit, Kennwert). Somit ist eine eigenständige Formulierung im textgesteuerten Interface überflüssig und daher die Führung durch das Interface bereits vordefiniert und strukturiert.
Die Auswahl im Buttongesteuerten Interface geht gefühlt schneller, die eingabe langer texte ist nicht so praktisch, für kurze Einagben aber sehr einfach
Einfache eingabe, teilweise dann Korrekturen erforderlich damit Ergebnisse dargestellt werden.
Das textgesteuerte Interface ermöglicht schnelles Arbeiten, ohne vorherige Einarbeitung in die Steuerung. Nicht so gut gefallen hat mir, dass nicht so gut ersichtlich war, welche Parameter wirklich von der KI als Parameter für die Berechnung übernommen wurden.
Vor allem für die Erstellung von Vergleichen bei mehreren Strecken wirkte die textgesteuerte Eingabe schneller. Für einfache Abfragen hat das Button-gesteuerte Interface klare Vorteile.
Offene Frage 3
Haben Ihnen bei der Verwendung des textgesteuerten Interfaces Funktionen gefehlt, oder haben Sie Verbesserungsvorschläge? Wenn ja, schreiben Sie diese hier auf.
automatische Vervollständigung der Adressen oder Straßennamen. automatische Vervollständigung der richtigen Datums oder Uhrzeitangabe.
Es fehlte die Information, was möglich ist und was nicht. Ein einfaches Prompten ohne die Taskaufgabe wäre viel schwieriger, weil man nicht weiß, wie man seine Eingabe gestalten soll. z. B. könnte man dies durch einen grauen Vorschlag im Prompfenster vorschreiben.
ggf. Textvorschläge bei Eingabe zum schnelleren Tippen bzw. rotes Unterkringeln bei falschen Wörtern (wobei ich gerade nicht weiß, ob das nicht schon integriert war)
Hinweise im User Interface, wie man Zeiträume mit Uhrzeit korrekt formulieren muss. Beispieltexte, wie man zu den diversen Ergebnissen kommt. Copy-Paste-Vorlagen sozusagen. Sprachfunktion, die gesprochenen Text direkt in das UI-Form überträgt.
Nur ein deutlicher visueller Hinweis dass alle Abfragen über das Textfeld gemacht werden müssen
Kurze Informationen zur Art der Texteingabe sind hilfreich. Z. B. immer Angabe des Datums, kein Copy n Paste. Aber auch ob es eine Frage sein muss oder lediglich die Angabe des zu Suchenden (z. B. Geschwindigkeit auf Maximilianstraße, 12.10.2022, 05:00 - 10:00 Uhr).
Textvorschläge für den Text würde die Eingabezeit verkürzen und es damit angehemer machen zu benutzen
Es wäre bei der Antwort bei einem Fehler gut, wenn der Art des Fehlers übermittelt würde (z.B. Straßenname falsch geschrieben, fehlende Eingabe z.B was berechnet werden soll (durchschnittliche Geschweingkeit
Automatisches auflösen der Adressdaten wäre super. Außerdem wäre es vielleicht schön, als Ergebnis von der Textbasierten Anfrage auch die ausgewählten Parameter zu sehen zB nicht nur einblenden des Analyse Ergebnis sondern auch der gewählten Optionen im Button Interface.
Hilfreich wäre eine Ergänzung von Keywords für unterstützte Funktionen. Entweder als Handout für Einschulungszwecke oder z.B. mittels interaktiven Textvorschlägen während der Eingabe.
Offene Frage 4
Wenn Sie noch weitere Anmerkungen haben, können Sie diese hier mitteilen.
Innovativer und niedrigschwelliger Ansatz, um Menschen mit Ablehnung von klassichen UI-Elementen zur Bedienung komplexer Software zu bringen!
Super Studie :-)
super Idee - wird mit Sicherheit bei noch komplexeren Abfragen (Kontextauswahl) einen Mehrwert haben. Vielleicht wäre eine Kopplung auch denkbar - Eingabefeld obern und daraus wird die Button-orientierte Eingabe dann korrekt gesetzt - bzw. mann könnte da noch feinjustieren
Super Ansatz!

Ich versichere:

- dass ich die Masterarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- dass alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Publikationen entnommen sind, als solche kenntlich gemacht sind.
- dass ich dieses Masterarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

12.09.2023

Datum

Unterschrift