

Masterarbeit
über
**Ein Vergleich von
Informationsextraktionslösungen im Deutschen:
Open-Source vs. proprietär**

**Universität
Rostock**



Traditio et Innovatio

Eingereicht bei:

Fakultät für Informatik und Elektrotechnik
Universität Rostock

Eingereicht von:

Goergen, Léon Hermann

Matrikelnummer:

218204406

Studiengang:

Wirtschaftsinformatik (M.Sc.)

Erstbetreuer:

Prof. Dr. Kurt Sandkuhl

Zweitbetreuer:

Leon Griesch

Rostock,

22. Oktober 2024

Inhaltsverzeichnis

Abbildungsverzeichnis	4
Tabellenverzeichnis	4
1. Einleitung	5
1.1. Motivation	6
1.2. Anwendungsszenarien	7
1.3. Ziele der Arbeit	8
1.4. Struktur der Arbeit	9
2. Grundlagen und Begriffe	10
2.1. Konzept der Informationsextraktion	10
2.1.1. Übersicht über die Informationsextraktion	11
2.1.2. Prozess der automatisierten Informationsextraktion	13
2.1.3. Techniken der Informationsextraktion	14
2.2. Herausforderungen der Informationsextraktion	18
2.2.1. Ressourcen und Datenzugang	18
2.2.2. Datenaufbereitung	19
2.2.3. Merkmalsextraktion	20
2.3. Transformer-Architekturen	21
2.3.1. Konzept	21
2.3.2. Technische Architektur	24
3. Bestehende Ansätze	29
3.1. FOSS Modelle	29
3.1.1. Encoder Modelle	29
3.1.2. Decoder Modelle	32
3.1.3. Encoder-Decoder Modelle	33
3.2. Proprietäre Softwarelösungen	35
3.3. Stand der Forschung	37
4. Lösungsansatz	39
4.1. Methodik	39
4.2. Beschreibung des Datensatzes	40

4.3. Datenvorbereitung und -verarbeitung	43
4.4. Bewertungsmetriken	45
4.4.1. Leistungskennzahlen	46
4.4.2. Kostenvergleich	48
4.5. Domänenübergreifender Ansatz	50
5. Umsetzung	53
5.1. Implementation der Ansätze	53
5.2. Ergebnisse	56
5.2.1. Ergebnisse der FOSS-Modelle	57
5.2.2. Ergebnisse der proprietären Softwarelösungen	59
5.2.3. Kostenvergleich und -analyse	60
5.2.4. Ergebnisse im domänenübergreifenden Kontext	66
5.3. Diskussion der Resultate	67
6. Schlussbetrachtung	70
6.1. Zusammenfassung der Ergebnisse	70
6.2. Kritische Bewertung der Arbeit	72
6.3. Ausblick auf zukünftige Forschungsmöglichkeiten	73
Literatur	76
Anhang	85
Akronyme	91

Abbildungsverzeichnis

1.	Konzept	11
2.	Klassifizierung	12
3.	Informationsextraktion Workflow	13
4.	Techniken	17
5.	Self-Attention Mechanismus	23
6.	Encoder Schicht	24
7.	Decoder Architektur	27
8.	Binäre Konfusionsmatrix	46
9.	Amazon SageMaker Workflow	49
10.	Kostenvergleich	65

Tabellenverzeichnis

1.	FOSS Ergebnisse	58
2.	Proprietäre Ergebnisse	59
3.	FOSS Ergebnisse Domänenübergreifend	66

1. Einleitung

Die vorliegende Masterarbeit befasst sich mit der Informationsextraktion (IE) mithilfe von Deep-Learning Ansätzen, einem Bereich, der in den letzten Jahren zunehmend an Bedeutung gewonnen hat. Aufgrund der fortschreitenden technologischen Entwicklungen und der steigenden Anforderungen gibt es eine Vielzahl an Ansätzen, welche für diese Aufgabe genutzt werden können. Diese Arbeit zielt darauf ab, verschiedene Techniken und Ansätze in einem Experiment miteinander zu vergleichen. Im Folgenden wird in Kapitel 1.1 zunächst die Motivation für die Wahl des Themas genauer erläutert, gefolgt von einer Darstellung der relevanten Anwendungsszenarien in Abschnitt 1.2. Danach werden in Abschnitt 1.3 die konkreten Ziele der Arbeit beschrieben, bevor schließlich der Aufbau und die Struktur der Arbeit in Kapitel 1.4 vorgestellt werden.

1.1. Motivation

Unstrukturierte Daten sind Informationen, welche nicht in einem vordefinierten, strukturierten Format vorliegen, wie beispielsweise Textdokumente, Bilder, Videos, Social-Media-Beiträge oder E-Mails [1]. Prognosen zufolge wird die Menge dieser Daten in den kommenden Jahren signifikant zunehmen. Laut Schätzungen des Marktforschungsunternehmens International Data Corporation sollten bis zum Jahr 2020 etwa 95% der global gespeicherten Daten unstrukturiert sein. Darüber hinaus wurde ein jährliches Wachstum dieser Datenkategorie von 65% prognostiziert [1]. Diese Entwicklung verdeutlicht die rasche Expansion unstrukturierter Informationen in der digitalen Welt und stellt Unternehmen sowie Organisationen vor erhebliche Herausforderungen in Bezug auf die Verarbeitung und Analyse dieser Datenmengen. Der Prozess der IE dient dazu, nützliche und strukturierte Informationen aus unstrukturierten Daten zu gewinnen [2]. Diese Informationen können in Form von Entitäten, Beziehungen, Objekten, Ereignissen und anderen Arten vorliegen. Die extrahierten Informationen können anschließend dazu genutzt werden, um Daten für weitergehende Analysen vorzubereiten [2]. Daher trägt eine effiziente und präzise Umwandlung von unstrukturierten Daten im Rahmen des IE-Prozesses wesentlich dazu bei, die Qualität und Genauigkeit der Datenanalyse zu verbessern. Die IE ist eine anspruchsvolle Aufgabe, insbesondere wenn es um die Verarbeitung hochgradig heterogener Daten geht. Aktuelle Technologien im Bereich des Data-Mining stoßen bei der Verarbeitung von Informationen aus Textdaten an ihre Grenzen [3]. Um dennoch eine effektive Informationsgewinnung aus textbasierten Daten zu ermöglichen, wurden verschiedene IE-Techniken entwickelt. Diese Arbeit befasst sich mit einem Vergleich von verschiedenen proprietären und Free Open-Source Software (FOSS) Ansätzen im Bereich des Deep Learning zur Extraktion dieser Daten. Es ist zu beachten, dass diese Arbeit an [4] angelehnt ist, welche sich bereits mit verschiedenen Ansätzen zur Dokumentenklassifikation mithilfe des maschinellen Lernens auseinandergesetzt hat. Der methodische Ansatz sowie einige Inhalte können sich dementsprechend in einigen Punkten überschneiden.

1.2. Anwendungsszenarien

Systeme des Information Retrieval (IR) extrahieren nützliche Daten aus großen Mengen an Informationen. Die IE verarbeitet diese Daten weiter, isoliert relevante Fragmente und integriert sie in eine leicht analysierbare, kohärente Struktur. Dies macht komplexe Informationen zugänglicher und besser verwertbar [5]. So können beispielsweise Informationen, die in unstrukturiertem Text vorliegen, in herkömmliche Datenbanken übersetzt werden, welche die Benutzer durch Standardabfragen durchsuchen können [5], wodurch eine Vielzahl an Anwendungsbereichen entsteht.

Die Metastudien von [2] und [3] bieten eine umfassende Analyse der potenziellen Anwendungsbereiche von IE in verschiedenen Domänen. In der Medizin und Biomedizin ermöglicht IE beispielsweise die automatische Extraktion relevanter Informationen aus wissenschaftlichen Publikationen, klinischen Berichten und Patientendaten, was die Forschung beschleunigt und die klinische Entscheidungsfindung unterstützt [6, 7]. Im Bereich Social Media kann IE verwendet werden, um Trends und Stimmungen zu analysieren, indem große Mengen an Benutzerbeiträgen und Interaktionen ausgewertet werden. Dies ist besonders wertvoll für Unternehmen und Organisationen, die beispielsweise Inhaltspersonalisierung betreiben [8]. Nachrichtenzusammenfassung ist ein weiteres Anwendungsgebiet, bei dem IE hilft, wesentliche Informationen aus umfangreichen Texten herauszufiltern und in prägnanten Zusammenfassungen darzustellen. Dies ist nützlich für Nachrichtenagenturen und Informationsdienste, die ihren Nutzern schnell relevante Informationen bereitstellen müssen [7, 9]. In Frage-Antwort-Systemen wird IE eingesetzt, um präzise Antworten auf spezifische Fragen zu extrahieren, was die Entwicklung intelligenter Suchmaschinen und persönlicher Assistenten wie Chatbots unterstützt [10]. Im Finanzsektor können durch IE automatisch Berichte und Analysen von Markttrends, Unternehmensfinanzen und wirtschaftlichen Entwicklungen erstellt werden, was die Effizienz von Finanzanalysten und Beratern erhöht [7, 10]. Marketingstrategien profitieren von IE, indem Daten aus verschiedenen Quellen aggregiert und analysiert werden, um zielgerichtete Kampagnen zu entwickeln und die Marktleistung zu überwachen [10].

1.3. Ziele der Arbeit

Die vorliegende Masterarbeit verfolgt das Ziel, verschiedene Ansätze zur Umwandlung unstrukturierter Daten in strukturierte Informationen eingehend zu untersuchen. Die zentrale Fragestellung dieser Arbeit besteht darin, wie diese Daten systematisch analysiert und in ein strukturiertes Format überführt werden können, um ihre Weiterverwendbarkeit zu gewährleisten.

1. Ein Schwerpunkt der Untersuchung liegt auf der Analyse technischer Ansätze wie Named Entity Recognition (NER) und Question Answering (QA). Diese Verfahren haben sich in der automatischen Verarbeitung natürlicher Sprache als effektiv erwiesen und bieten vielversprechende Möglichkeiten zur Extraktion relevanter Informationen aus unstrukturierten Daten. Darüber hinaus werden sowohl FOSS Lösungen als auch proprietäre Softwarelösungen betrachtet, um ein umfassendes Bild der verfügbaren Technologien zu erhalten und deren Vor- und Nachteile zu vergleichen. Im Bereich der FOSS sollen zudem verschiedene Architekturen untersucht werden.
2. Die Evaluierung der unterschiedlichen Ansätze erfolgt sowohl hinsichtlich ihrer Performance als auch ihrer Effizienz. Dabei sollen verschiedene Metriken zur Anwendung kommen, um die Genauigkeit und Zuverlässigkeit der Ergebnisse zu messen. Zudem soll ein Kostenvergleich aufgestellt werden, um die wirtschaftliche Rentabilität der einzelnen Lösungen zu bewerten. Diese duale Betrachtung von Performance und Effizienz ermöglicht eine differenzierte Analyse der eingesetzten Methoden und deren Anwendbarkeit in realen Szenarien.
3. Ein weiteres Ziel dieser Arbeit ist es, wiederverwendbare Ansätze zu schaffen, die als Grundlage für zukünftige Vergleiche verschiedener Softwarelösungen dienen können. Durch die Entwicklung standardisierter Kriterien und Bewertungsmethoden sollen relevante Methoden zur Evaluierung und Auswahl von Tools zur Datenverarbeitung bereitgestellt werden. Dies trägt nicht nur zur wissenschaftlichen Diskussion bei, sondern unterstützt auch Praktiker in der Industrie fundierte Entscheidungen bei der Auswahl geeigneter Software zu treffen.

1.4. Struktur der Arbeit

Zunächst wird in Kapitel 2 eine umfassende Einführung in die benötigten Grundlagen und Begriffe gegeben, welche das grundlegende Konzept der Informationsextraktion behandelt. Es beginnt mit einer Einführung in die Informationsextraktion und beleuchtet anschließend den automatisierten Prozess der IE und die dabei verwendeten Techniken. Weiterhin werden die Herausforderungen, wie Ressourcen- und Datenzugang, die Datenaufbereitung erörtert, sowie die Merkmalsextraktion erörtert. Anschließend wird das Konzept der Transformer-Architekturen, welche die in dieser Arbeit verwendeten FOSS Modelle darstellen, detailliert beschrieben, wobei deren Konzept und technische Architektur im Fokus stehen. Kapitel 3 beschreibt bestehende Ansätze im Bereich der IE, wobei auf die in dieser Arbeit verwendeten FOSS Modelle, sowie die proprietären Softwarelösungen eingegangen wird. Anschließend folgt eine kritische Bewertung der aktuellen Ansätze im Bereich der IE. In Kapitel 4 wird die Methodik beschrieben, um die in Abschnitt 1.3 beschriebenen Ziele zu erreichen. Dabei wird die Vorgehensweise, der verwendete Datensatz und die genutzten Bewertungsmetriken, sowohl für die Leistung als auch den Kostenvergleich, diskutiert. In Kapitel 5 wird die Implementierung der in Abschnitt 4 beschriebenen Herangehensweise diskutiert, sowie die Ergebnisse des Experimentes vorgestellt. Das Kapitel endet anschließend mit einer Diskussion der Resultate. Das abschließende Kapitel 6 fasst die wesentlichen Erkenntnisse der Arbeit zusammen. Anschließend wird eine kritische Bewertung der Arbeit vorgenommen. Darüber hinaus werden Vorschläge für zukünftige Forschungsansätze gemacht und ein Framework skizziert, welches für die Wiederverwendbarkeit der Ergebnisse dieser Arbeit dienen soll.

2. Grundlagen und Begriffe

In den letzten Jahren hat die IE aus Textdaten durch Fortschritte im Bereich des Deep-Learnings erheblich an Bedeutung gewonnen [11]. Das Ziel dieses Kapitels ist es, einen umfassenden Überblick über die wesentlichen Konzepte und Grundlagen im Bereich der IE mithilfe von Deep-Learning-Techniken zu geben. Abschnitt 2.1 geht hierbei auf das grundlegende Konzept der IE ein, inklusive den Prozessen und Techniken der IE. Abschnitt 2.2 geht auf die unterschiedlichen Herausforderungen ein, welche bewältigt werden müssen um ein IE-System erfolgreich zu entwickeln. In dieser Arbeit wird eine Vielzahl an Transformermodellen untersucht, weshalb Abschnitt 2.3 das Konzept und die Architektur dieser erläutert. Des weiteren werden die untersuchten Modelle vorgestellt.

2.1. Konzept der Informationsextraktion

In den 1980er Jahren begannen intelligente, textbasierte Systeme damit, Texte so zu verarbeiten, dass sie automatisch relevante Informationen auf schnelle, effektive und hilfreiche Weise liefern. Generell lassen sich zwei Hauptbereiche dieser Systeme unterscheiden: Information Retrieval (IR) und Informationsextraktion (IE) [12, 13]. IR-Techniken werden verwendet, um aus einer Sammlung von Dokumenten diejenigen auszuwählen, die den Einschränkungen einer Abfrage, in der Regel einer Liste von Schlüsselwörtern, am ehesten entsprechen. Daher ermöglichen IR-Techniken die Bereitstellung relevanter Dokumente als Antwort auf eine Anfrage [12]. Die IE-Technologie beinhaltet eine tiefere Verständnisaufgabe. Während bei IR die Antwort auf eine Anfrage einfach eine Liste potenziell relevanter Dokumente ist, erfordert IE das Auffinden und Extrahieren des relevanten Inhaltes solcher Dokumente [12].

In diesem Abschnitt wird das Konzept der IE erläutert. Abschnitt 2.1.1 gibt einen Überblick über das grundlegende Konzept hinter IE, Abschnitt 2.1.2 erläutert die Funktionsweise von automatisierten IE-Systemen im Hinblick auf Machine Learning (ML) und Abschnitt 2.1.3 veranschaulicht die verschiedenen Techniken, welche in das Gebiet der IE fallen.

2.1.1. Übersicht über die Informationsextraktion

Die IE ist ein spezialisierter Bereich des Natural Language Processing (NLP), der entwickelt wurde, um mehrdeutige natürliche Texte zu verstehen [3, 14], um anschließend spezifische Entitäten oder Beziehungen in unstrukturierten oder halb-strukturierten Textdaten zu identifizieren, zu extrahieren und in ein strukturiertes Format wie eine Datenbank zu überführen [15, 16]. Diese extrahierten Informationen ermöglichen es Benutzern, Daten leichter zu handhaben und zu verstehen, was wiederum die Analyse, Suche und Visualisierung dieser Daten erleichtert [17]. Ein konkretes Beispiel für ein solches Ergebnis der IE zeigt sich in Abbildung 1, in der strukturierte Informationen in Form einer Tabelle aus einem natürlichsprachigen Text extrahiert wurden.

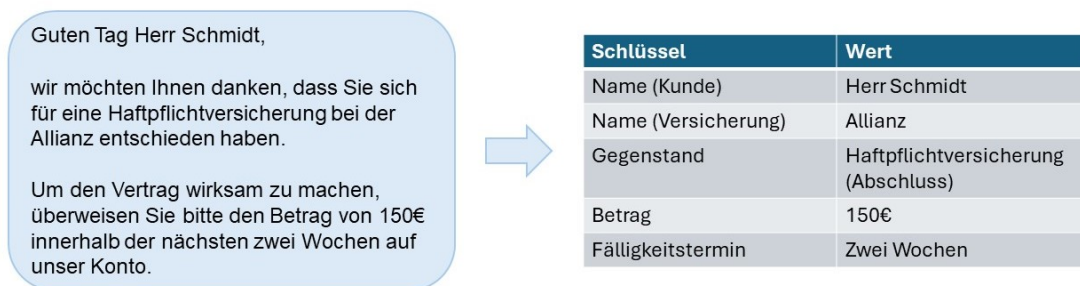


Abbildung 1: Konzept

IE kann in zwei Paradigmen unterteilt werden: „Open-Domain“ IE und „Closed-Domain“ IE. Open-Domain und Closed-Domain IE unterscheiden sich hauptsächlich in ihrer Anwendungsbreite. Open-Domain IE zielt darauf ab, Informationen aus Texten jeglicher Art und ohne thematische Einschränkung zu extrahieren. Es ist generalisiert und nicht auf eine bestimmte Domäne beschränkt, wodurch es in vielen verschiedenen Wissensbereichen angewendet werden kann [18]. Closed-Domain IE hingegen konzentriert sich auf spezifische Themengebiete oder Domänen und ist stark spezialisiert. Es wird für festgelegte Bereiche entwickelt, um präzise Informationen aus diesen Fachgebieten zu extrahieren [18]. Diese Arbeit konzentriert sich auf den Bereich der Closed-Domain IE, was in Abschnitt 4 genauer beschrieben wird. Des weiteren kann IE in zwei Strategien unterteilt werden: Ansätze im Bereich des Knowledge Engineering (KE) und Ansätze im Bereich des Machine Learning (ML) [11]. Die Unterteilung wird in Abbildung 2 veranschaulicht.

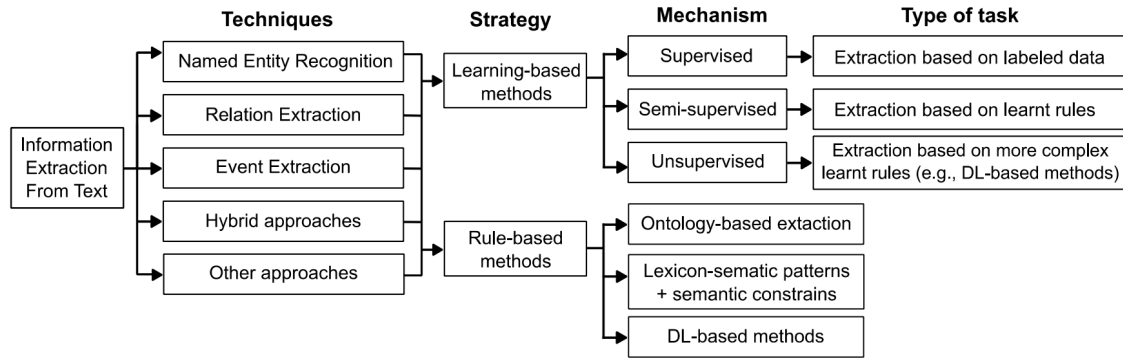


Abbildung 2: Klassifizierung [3]

Das Gebiet der KE wurde von [19] als „die technische Disziplin, bei der Wissen in Computersysteme integriert wird, um komplexe Probleme zu lösen, die normalerweise ein hohes Maß an menschlichem Fachwissen erfordern“ definiert. Typischerweise wird das benötigte Wissen durch Befragung von Experten erlangt, welche erläutern, wie bestimmte Aufgaben gelöst werden. Dieses Wissen wird in der Regel in Form von Produktionsregeln umgesetzt, welche dann von einem entsprechenden Regelinterpreter ausgeführt werden. Daher werden diese Systeme auch als regelbasierte Systeme bezeichnet [20]. Da diese Arbeit sich auf den maschinellen Lernaspekt der IE konzentriert, wird auf das Gebiet des KE nicht weiter eingegangen. Für weiterführende Informationen auf diesem Bereich wird dem Leser empfohlen, [20] zu konsultieren.

Der auf ML basierende Ansatz der IE beschäftigt sich mit der Anwendung von Lernmethoden und der Implementierung von ML-Algorithmen zur Erstellung von Regeln für das IE-System [11]. Somit ist es nicht erforderlich, dass ein Wissensexperte diese Regeln manuell formuliert. Stattdessen ist lediglich eine Person erforderlich, die mit der Domäne und den Systemfunktionalitäten vertraut ist [21]. Der ML-Ansatz kann in zwei Teilgebiete unterteilt werden: „supervised learning“ und „unsupervised learning“ [22]. Überwachtes Lernen bezieht sich auf die Erstellung eines Prädiktormodells aus einem Trainingsdatensatz, welches eine Reihe von Trainingsbeispielen mit Eingabedaten und den gewünschten Ausgabereaktionen enthält. Beim unüberwachten Lernen hingegen werden versteckte Strukturen in nicht gekennzeichneten Eingabedaten gesucht. Dabei lernt das Modell, Muster in den Daten zu erkennen, ohne dass die richtigen Antworten oder Ausgaben bekannt sind [11]. Darüber hinaus kann eine Kombination beider Ansätze genutzt werden: „semi-supervised Learning“, bei welchem eine Kombination aus beschrifteten und unbeschrifteten Daten verwendet wird [23].

2.1.2. Prozess der automatisierten Informationsextraktion

Der Prozess des überwachten ML folgt üblicherweise einem typischen Muster, wie in in Abbildung 3 dargestellt.

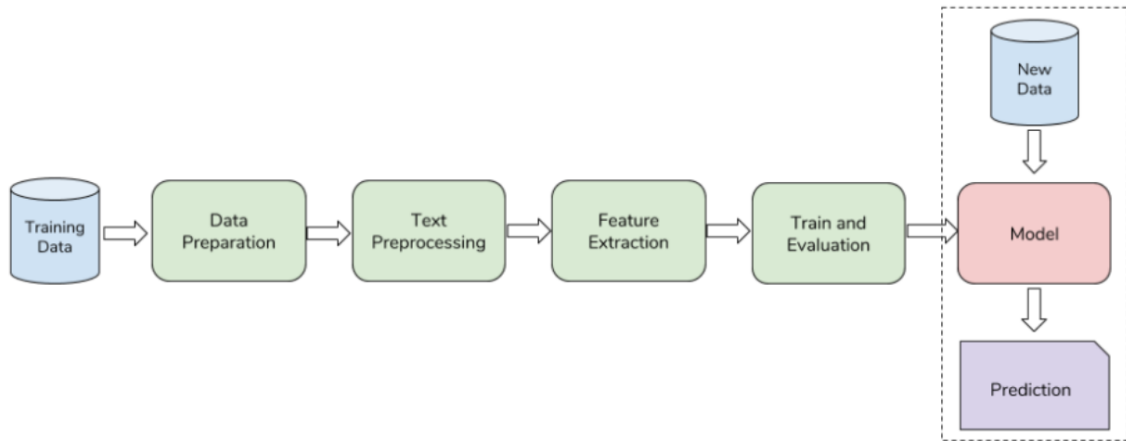


Abbildung 3: IE Workflow [24]

Bevor der Trainingsprozess beginnt, ist es notwendig, die gegebenen Trainingsdaten vorzubereiten [25]. Diese Vorbereitung umfasst in der Regel das Entfernen redundanter oder irrelevanter Proben oder Merkmale, das Erkennen und Beseitigen von Ausreißern, die das Ergebnis beeinträchtigen könnten, und das Ausbalancieren des Datensatzes, um eine Verzerrung zu vermeiden [25]. Diese Schritte sind nötig, um sicherzustellen, dass der Trainingsprozess zuverlässig und präzise verläuft [26]. Es zu erwähnen, dass die Trainingsdaten in unterschiedlichen Formaten, wie beispielsweise PDF-Dateien, bereitgestellt werden können, was den Einsatz von Techniken wie Optical Character Recognition (OCR) zur Extraktion des Inhalts erforderlich macht. Nach der Extraktion und der Formatierung der Trainingsdaten wird in den meisten Studien, wie beispielsweise [27], die Vorverarbeitung des Textinhalts mittels Techniken wie der Entfernung von Stoppwörtern, Rechtschreibkorrektur und Lemmatisierung empfohlen. Diese Vorverarbeitungsschritte können die Genauigkeit des IE-Systems [28] erheblich verbessern. Eine detaillierte Ausarbeitung dieser Schritte befindet sich in Abschnitt 2.2.2.

Der folgende Schritt, die „Merkmalsextraktion“, ist ein Prozess, bei dem die Daten auf ihre wichtigsten Merkmale reduziert werden, um die zugrunde liegende Struktur der Daten zu ermitteln. Anschließend kann das Modell die Daten interpretieren und als Eingabe verwenden, indem es den Text in numerische Darstellungen, wie beispielsweise Wordembeddings, umwandelt [29]. Häufig folgt auf die Merkmalsextraktion eine „Merkmalsauswahl“, bei der

eine Teilmenge der relevantesten und informativsten Merkmale aus dem Datensatz ausgewählt und als Eingabe für das Modell verwendet wird, während der Rest verworfen wird [30]. Die Merkmalsauswahl kann dazu beitragen die Dimensionalität des Merkmalsatzes zu reduzieren und somit die Effizienz und Leistung des Modells zu verbessern, vor allem bei der Arbeit mit großen Datensätzen [31]. Während die Merkmalsextraktion und -auswahl in vielen traditionellen ML-Pipelines von entscheidender Bedeutung sind, werden sie in Deep-Learning-Modellen häufig automatisiert oder implizit gehandhabt [32]. Deep-Learning-Modelle können Merkmale direkt aus den Daten lernen, ohne manuelles Feature-Engineering, wodurch sie komplexe Muster und Beziehungen in den Daten erfassen und die Leistung bei vielen Aufgaben verbessern können [32]. Der Prozess der Merkmalsextraktion ist genauer in Abschnitt 2.2.3 beschrieben.

Der letzte Schritt im Arbeitsablauf des maschinellen Lernens umfasst die Erstellung des gewünschten Modells, die Auswahl geeigneter Hyperparameter und das Training des Modells auf den vorverarbeiteten und mit Merkmalen versehenen Daten [33]. Nach der Erstellung des Modells ist es wichtig, dessen Leistung anhand der Testdaten zu bewerten und gegebenenfalls die Hyperparameter anzupassen, um seine Implementierung zu optimieren. Nach dem Training oder der Feinabstimmung des Modells kann es Informationen aus neuen, nicht gesehenen Daten in der Produktion extrahieren.

2.1.3. Techniken der Informationsextraktion

Die Message Understanding Conferences (MUC) spielten eine maßgebliche Rolle bei der Entwicklung und Verbesserung von Technologien des NLP, insbesondere im Bereich der IE. Diese Konferenzen fanden in den späten 1980er und 1990er Jahren statt und wurden von der Defense Advanced Research Projects Agency (DARPA) in den USA organisiert. Ihr Ziel war die Bewertung des Fortschritts und der Leistungsfähigkeit NLP-Systemen [10]. Mit der siebten und letzten Konferenz im Jahr 1998, der *MUC-7* [34], wurde eine Definition der IE in fünf Teilgebieten festgelegt.

Die grundlegenden Prinzipien dieser Teilgebiete werden hier kurz aufgeführt; für eine ausführliche Erklärung wird die Arbeit von [10] empfohlen:

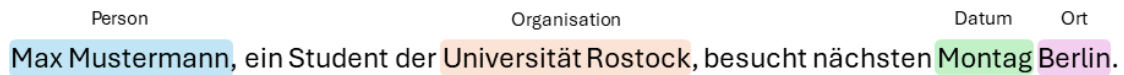
- **Named Entity Recognition** (NER) ist eine Technik, die darauf abzielt, Entitäten wie Personennamen, Ortsnamen, Organisationen und andere spezifische Begriffe in einem Text zu identifizieren und zu klassifizieren. Zum Beispiel erkennt NER in dem Satz „Angela Merkel ist die Bundeskanzlerin von Deutschland“ die Entitäten „Angela Merkel“ (Person) und „Deutschland“ (Ort).
- **Coreference resolution** (CO) ist die Aufgabe, festzustellen, wann verschiedene Ausdrücke im Text auf die gleiche Entität verweisen. Zum Beispiel in dem Satzpaar „Angela Merkel ist die Bundeskanzlerin von Deutschland. Sie wurde 1954 geboren.“, bezieht sich „Sie“ auf „Angela Merkel“. Die Technik hilft, Referenzen auf Entitäten innerhalb eines Textes zu verknüpfen.
- **Template element construction** (TE) bezieht sich auf das Extrahieren und Strukturieren von Informationen aus einem Text, um spezifische Elemente eines vordefinierten Schemas (Templates) zu füllen. Diese Technik wird häufig in der Informationsextraktion verwendet, um relevante Daten zu bestimmten Ereignissen oder Entitäten zu sammeln.
- **Template relation construction** (TR) ist der Prozess des Identifizierens und Strukturierens von Beziehungen zwischen den verschiedenen Elementen innerhalb eines Templates. Es geht darum, die Verbindungen und Interaktionen zwischen den extrahierten Informationen zu definieren und darzustellen, um ein vollständiges und kohärentes Bild zu erstellen.
- **Scenario template production** (ST) umfasst die Erstellung umfassender Szenarien auf der Basis von Templates, die verschiedene Ereignisse und ihre Beziehungen zueinander darstellen. Diese Technik wird oft verwendet, um komplexe Ereignisse zu modellieren und zu analysieren, indem alle relevanten Informationen und ihre Verknüpfungen in einem strukturierten Format dargestellt werden.

Die Aufgaben des MUC wurden vom Programm des Automatic Content Extraction (ACE) des U.S. National Institute of Standards and Technology (NIST) übernommen, wobei allgemeinere Typen von Entitäten, Beziehungen und Ereignissen definiert wurden [7]. Die endgültigen Techniken sind in Abbildung 2 zu finden. ACE umfasst die folgenden Aufgaben:

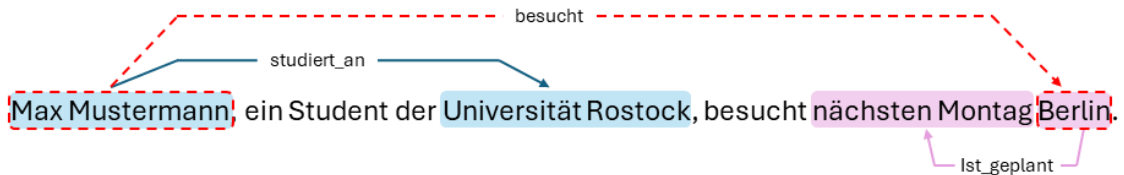
- **Entity Detection and Recognition** zielt darauf ab, bestimmte vordefinierte Entitäten in einem Text zu identifizieren und zu klassifizieren. Zu den häufig erkannten Entitäten gehören Personen, Organisationen, Orte und andere spezielle Begriffe [7, 35, 2]. Beispielsweise würde der Satz „Olaf Scholz ist der Bundeskanzler von Deutschland“ die Entitäten „Olaf Scholz“ (Person) und „Deutschland“ (Ort) enthalten.
- **Relation Detection and Recognition** ist eine Technik, die darauf abzielt, die Beziehungen zwischen den erkannten Entitäten in einem Text zu identifizieren und zu klassifizieren. Diese Technik geht über die einfache Erkennung von Entitäten hinaus und versucht, die Art der Beziehung zwischen ihnen zu verstehen [7, 35, 2]. Beispielsweise würde die Technik im Satz „Olaf Scholz ist der Bundeskanzler von Deutschland“ erkennen, dass „Olaf Scholz“ in einer „Führungsposition“ zu „Deutschland“ steht.
- **Event Detection and Recognition** ist ein Prozess, der darauf abzielt, Ereignisse in Texten zu identifizieren und zu klassifizieren. Ein Ereignis ist eine spezifische Handlung oder ein Vorkommnis, das von Entitäten ausgeführt wird oder diese betrifft. Diese Technik versucht, komplexe Informationen wie Wer, Was, Wann, Wo und Warum eines Ereignisses zu extrahieren [36, 2]. Zum Beispiel würde die Technik im Satz „Olaf Scholz hielt eine Rede auf dem G20-Gipfel“ das Ereignis „Rede halten“ erkennen und relevante Details wie handelnde Person (Olaf Scholz) und Veranstaltung (G20-Gipfel) extrahieren.

Ein Beispiel der zuvor beschriebenen Aufgaben der ACE findet sich in Abbildung 4. Die Grafik ist angelehnt an die Abbildung von [3].

Entity Detection and Recognition



Relation Detection and Recognition



Event Detection and Recognition

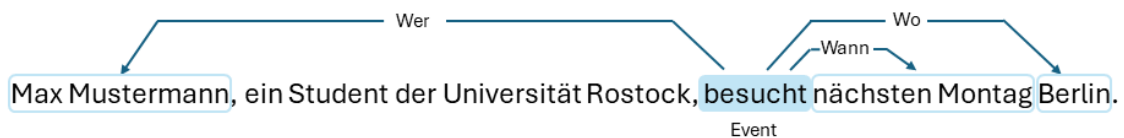


Abbildung 4: Techniken (angelehnt an [3])

Diese Arbeit konzentriert sich auf eine spezielle Form der NER. Die ACE definierte sieben Kategorien für Entitäten [35], die hier durch weitere Klassen in einer benutzerdefinierten NER erweitert werden. Darüber hinaus soll auch das Konzept des QA untersucht werden. QA ist ein Forschungsgebiet, welches verschiedene, aber verwandte Bereiche kombiniert, nämlich IR, IE und NLP. Ein typisches QA-System besteht aus drei verschiedenen Modulen, von denen jedes eine Kernkomponente neben anderen ergänzenden Komponenten hat: Das *Abfrageverarbeitungsmodul*, dessen Kern die Klassifizierung von Fragen ist, das *Dokumentverarbeitungsmodul*, dessen Kern das IR ist, und das *Antwortverarbeitungsmodul*, dessen Kern die IE ist [37]. QA Systeme sind deutlich komplexer als NER Systeme, da diese zuerst die Frage verstehen, die Dokumente verarbeiten und anschließend die passende Antwort formulieren müssen. Für die Erstellung eines QA-Systems sind zudem die Erkennung und Erfassung der Entitäten, Beziehungen, sowie Ereignissen erforderlich, während NER sich lediglich auf die Erkennung von Entitäten bezieht. Eine ausführliche Erläuterung zu QA-Systemen findet sich in [37].

2.2. Herausforderungen der Informationsextraktion

Wie in Abschnitt 1.2 ausführlich dargestellt, besitzt die IE eine Vielzahl von praktischen Anwendungsmöglichkeiten. Dennoch bestehen signifikante Herausforderungen darin, diese Strategien effektiv an die Anforderungen der realen Welt anzupassen. In der modernen Ära werden Daten in enormem Umfang und aus einer Vielzahl von Quellen gesammelt, was als „Big-Data-Problem“ bezeichnet wird. Diese Daten stammen aus vielfältigen Quellen, weisen zahlreiche Formate auf und sind in unterschiedlichen Sprachen kodiert [38, 3]. In diesem Abschnitt werden einige der am weitest verbreiteten Schwierigkeiten erörtert. Abschnitt 2.2.1 widmet sich den Herausforderungen in Bezug auf den Ressourcenaufwand und den Zugang zu Daten. Abschnitt 2.2.2 beleuchtet die Probleme der Datenqualität, während Abschnitt 2.2.3 die Schwierigkeiten bei der Verarbeitung menschlich verfasster Texte für maschinelle Systeme thematisiert.

2.2.1. Ressourcen und Datenzugang

Unstrukturierte Daten stellen eine reichhaltige Informationsquelle dar, aber die Extraktion relevanter Informationen bleibt eine große Herausforderungen [2]. Mit der wachsenden Menge an Informationen im Internet wird es immer schwieriger, den Überblick zu behalten und die Informationen effektiv zu nutzen. Die Herausforderung ist dabei, die gesuchten Informationen in einem nützlichen Format, einfach und schnell aufzufinden [17].

Zudem erfordert die Verarbeitung dieser Daten fortschrittliche Methoden und Techniken, um die relevanten Informationen effektiv zu extrahieren und zu nutzen [39]. Viele existierende Systeme wurden für die Verarbeitung kleiner Datensätze entwickelt und stoßen an ihre Grenzen, wenn sie auf große Mengen an Daten angewendet werden. Die Verarbeitung von Millionen von Dateneinträgen erfordert erhebliche Zeit- und Ressourcenaufwände, die weit über die Kapazitäten traditioneller Systeme hinausgehen [39]. Die hohe Dimensionalität, Vielfalt, Dynamik und Heterogenität der Daten verlangen nach fortschrittlichen Techniken zur Datenverarbeitung [40, 41]. Mit der zunehmenden Menge an Daten aus unterschiedlichsten Quellen steigt die Komplexität der Verarbeitung, da nicht nur die Zugänglichkeit, sondern auch die Nutzbarkeit der Daten sichergestellt werden muss.

Des Weiteren stellt die Spärlichkeit dieser Daten eine signifikante Herausforderung dar [42, 43]. Viele nützliche Informationen sind in großen Textkorpora nur selten vertreten. Das bedeutet, dass Modelle häufig mit einer begrenzten Anzahl von Beispielen für bestimmte Entitäten oder Ereignisse arbeiten müssen, was die Robustheit und Verallgemeinerungsfähigkeit der Modelle beeinträchtigen kann [2].

2.2.2. Datenaufbereitung

Die Implementierung von Modellen des maschinellen Lernens birgt verschiedene Herausforderungen, von denen die Gewährleistung einer angemessenen Datenqualität als eine der wichtigsten gilt, da die Leistung und der Erfolg des Modells in hohem Maße von dieser Qualität abhängen [44]. Eine Datenbereinigung ist oft notwendig, da fehlende Werte, Ausreißer und verrauschte Daten in den meisten realen Produktionsdatensätzen weit verbreitet sind, was die Leistung der Datenanalyse beeinträchtigt und die Effizienz verringert [25].

Das Trainieren von IE-Systemen aus unstrukturierten Textdaten stellt Forscher vor eine Reihe bedeutender Herausforderungen, da Qualitätsprobleme ein großes Hindernis bei der Extraktion nützlicher und relevanter Informationen sind, was den IE-Prozess mühselig macht [2]. Verschiedene Studien und Metaanalysen, unter anderem von [2, 3], haben eine Vielzahl dieser Probleme analysiert.

Ein Hauptproblem ist die Mehrdeutigkeit von Texten. Worte und Phrasen können in verschiedenen Kontexten unterschiedliche Bedeutungen haben, was die präzise Identifikation von Entitäten erschwert [45]. Beispielsweise kann das Wort „Bank“ sowohl ein Finanzinstitut als auch eine Sitzgelegenheit bezeichnen. Diese Ambiguität macht es schwierig die korrekte Bedeutung ohne zusätzliche Kontextinformationen zu bestimmen. Ein weiteres wesentliches Problem sind fehlende oder unvollständige Daten [39]. Oft sind die zur Verfügung stehenden Daten lückenhaft, was die Modelle vor Herausforderungen stellt, da sie auf unvollständigen Informationen basieren müssen. Dies kann zu ungenauen oder verzerrten Ergebnissen führen, insbesondere wenn wichtige Kontextinformationen fehlen, die für das Verständnis und die richtige Kategorisierung notwendig sind [39, 2]. Komplexe verschachtelte Entitäten sind ein weiteres Hindernis [40]. Texte enthalten oft ineinander verschachtelte Entitäten, wie etwa in dem Satz „Das Buch von Autor X über die Geschichte von Land Y“. Hier müssen mehrere Schichten von Informationen extrahiert und korrekt

miteinander in Beziehung gesetzt werden, was eine zusätzliche Komplexitätsebene darstellt. Die Identifizierung von Kontextinformationen ist ebenfalls entscheidend [45]. Ohne den richtigen Kontext können extrahierte Informationen falsch interpretiert werden. Kontext hilft dabei, die Bedeutung von Entitäten und ihren Beziehungen zu anderen Entitäten besser zu verstehen, was für die Präzision der Extraktion unerlässlich ist. Darüber hinaus tragen Rauschen und Sprachvariabilität zur Herausforderung bei [40]. Texte enthalten oft irrelevante Informationen (Rauschen) und weisen eine große Vielfalt an Ausdrucksweisen und Sprachstilen auf, was die Konsistenz und Genauigkeit der Modelle beeinträchtigen kann. Sprachvariabilität umfasst sowohl die Verwendung unterschiedlicher Synonyme als auch grammatikalische und stilistische Unterschiede, die von Autor zu Autor variieren [40].

2.2.3. Merkmalsextraktion

Die Merkmalsextraktion ermöglicht es Maschinen, Daten wie Menschen zu klassifizieren und zu verstehen. Dies beinhaltet die Umwandlung von Rohdaten in numerische Darstellungen, die von Modellen interpretiert und verarbeitet werden können [46]. Die Merkmalsextraktion stellt eine zentrale Herausforderung in der Datenverarbeitung dar, da sie entscheidend dafür ist, wie Maschinen Daten klassifizieren und verstehen. Ein Hauptproblem ist der syntaktische und semantische Informationsverlust, der bei der Umwandlung von Wörtern in numerische Werte auftreten kann [46].

Kategorische Wortdarstellungen, die Wörter als diskrete kategorische Variablen kodieren, leiden unter der Einschränkung, dass sie die syntaktischen und semantischen Nuancen von Wörtern nicht adäquat erfassen können. Dies führt zu einer hohen Dimensionalität, die die Effizienz und Genauigkeit von Modellen beeinträchtigen kann [46]. Gewichtete Wortdarstellungen, die die Häufigkeit von Wörtern im Text berücksichtigen, bieten eine Verbesserung, sind jedoch ebenfalls nicht in der Lage, die komplexen syntaktischen und semantischen Beziehungen zwischen Wörtern vollständig zu erfassen [46]. Diese Methoden führen oft zu einer redundanten und ineffizienten Repräsentation, die die Leistungsfähigkeit der daraus resultierenden Modelle beeinträchtigen kann.

Eine weitere Herausforderung besteht in der Notwendigkeit, qualitativ hochwertige Wortembeddings zu erzeugen. Während vortrainierte Modelle auf großen Korpora basieren und hochwertige Embeddings liefern können, ist es schwierig, diese Modelle an spezifische Anwendungsfälle anzupassen, ohne die ursprüngliche Qualität der Embeddings zu beeinträchtigen [46]. Moderne Ansätze wie die Transformer-Architektur adressieren einige dieser Herausforderungen durch die Verwendung von Selbstaufmerksamkeit und positionalen Embeddings, die es ermöglichen, kontextuelle Informationen effektiver zu integrieren. Dennoch bleibt die Komplexität der Modellarchitektur eine Herausforderung, da sie erhebliche Rechenressourcen erfordert und die Implementierung kompliziert ist [47, 48].

2.3. Transformer-Architekturen

Bei allen in dieser Arbeit getesteten FOSS Modelle handelt es sich um Transformermodelle. Transformermodelle zählen heute zu dem State-of-the-Art Modellen in vielen Anwendungsbereichen [46]. Abschnitt 2.3.1 geht dabei auf die Grundlegende Funktionsweise dieser Modelle ein. Abschnitt 2.3.2 beschäftigt sich mit der Architektur und Abschnitt 3.1 stellt die in dieser Arbeit genutzten Modelle kurz vor.

2.3.1. Konzept

Gewichtete Wortdarstellungen erstellen numerische Repräsentationen von Text auf der Grundlage der Häufigkeit von Wörtern im Text, indem sie jedem Wort ein Gewicht zuweisen, das ein Maß für die Wichtigkeit oder Relevanz des Wortes im Text darstellt [46]. Bekannte Techniken sind unter anderem „TF-IDF“ und „LSA“. Diese Ansätze haben jedoch, wie in Absatz 2.2.3 beschrieben, Schwierigkeiten die syntaktischen und semantischen Nuancen von Wörtern zu erfassen. Folglich haben Forscher die Verwendung verteilter Wortrepräsentationen in einem niedrigdimensionalen Raum erforscht und somit Wordembeddings entwickelt. Bei einer Wordembedding werden alphanumerische Zeichen in Vektorformen umgewandelt, wobei ein Vektor jedes Wort repräsentiert und eine bestimmte Dimension eines Punktes im Raum bezeichnet, was die Gruppierung von Wörtern mit gemeinsamen Merkmalen, wie kontextuelle oder semantische Ähnlichkeit, in unmittelbarer Nähe innerhalb des Raums ermöglicht [49, 50]. Während es möglich ist, benutzerdefinierte Wordembeddings zu trainieren, während ein maschinelles Lernmodell implementiert wird, wurden verschiedene Wordembeddingmodelle auf großen Korpora von Textdokumenten

vortrainiert. Ein bekanntes Beispiel eines solchen Worteinbettungsmodells ist „Word2Vec“, das mit Hilfe von zwei Schichten in einem neuronalen Netz einen Vektor für jedes Wort erstellt, der es ermöglicht, die semantischen und grammatikalischen Informationen eines Wortes zu erfassen, indem es sich auf lokales Kontextwissen konzentriert [46, 50]. Weitergehend gibt es die Technik der kontextualisierten Einbettungen, welche für jedes Wort eine dynamische Einbettung, die sich je nach Kontext im Satz ändert, erzeugt. Durch das Lernen aus bidirektionalen Sprachmodellen, die die umgebenden Wörter auf beiden Seiten des Zielworts berücksichtigen, wird diese Einbettung erreicht [46, 51]. Ein bekanntes Beispiel hierfür ist unter anderem „ELMo“.

Eine Forschungsteam von Google stellte 2017 in der Arbeit „Attention is all you need“ [52] die Transformerarchitektur vor. Transformer-basierte kontextuelle Worteinbettungen sind deutlich besser, da sie aufgrund der tief gestaffelten Modellarchitektur mehr Informationen in Wortvektoren kodieren [47]. Dies geschieht unter anderem durch „Self-Attention“ und positionale Einbettungen [47]. Das Grundprinzip der Transformerarchitektur besteht darin, dass diese eine Sequenz von Eingabewörtern gleichzeitig verarbeitet und dabei Beziehungen zwischen Wörtern unabhängig von ihrem Abstand in der Sequenz berücksichtigt. Die Grundidee hinter der Self-Attention ist, dass jedes Wort in einer Sequenz auf alle anderen Wörter in der gleichen Sequenz „aufmerksam“ sein kann, um seinen Kontext zu bestimmen. Anstatt eine feste Einbettung für jeden Token zu verwenden, wird ein gewichteter Durchschnitt der Einbettungen berechnet, wobei die Gewichte durch die Relevanz der anderen Wörter bestimmt werden [48]. Da durch die Self-Attention jedoch keine Informationen über die Position der Wörter in der Sequenz enthält, werden zudem positionsabhängige Muster von Werten zu den Token-Einbettungen hinzugefügt. Diese Muster ermöglichen es den Aufmerksamkeitsmechanismen und Feedforward-Schichten, Positionsinformationen in ihre Berechnungen einzubeziehen, wodurch die Reihenfolge der Wörter berücksichtigt wird [48]. Ein Beispiel für den Self-Attention Mechanismus findet sich in Abbildung 5, in welche dem Wort „flies“ je nach Kontext eine andere Bedeutung angerechnet wird, veranschaulicht durch die verschieden farbigen Blöcke.

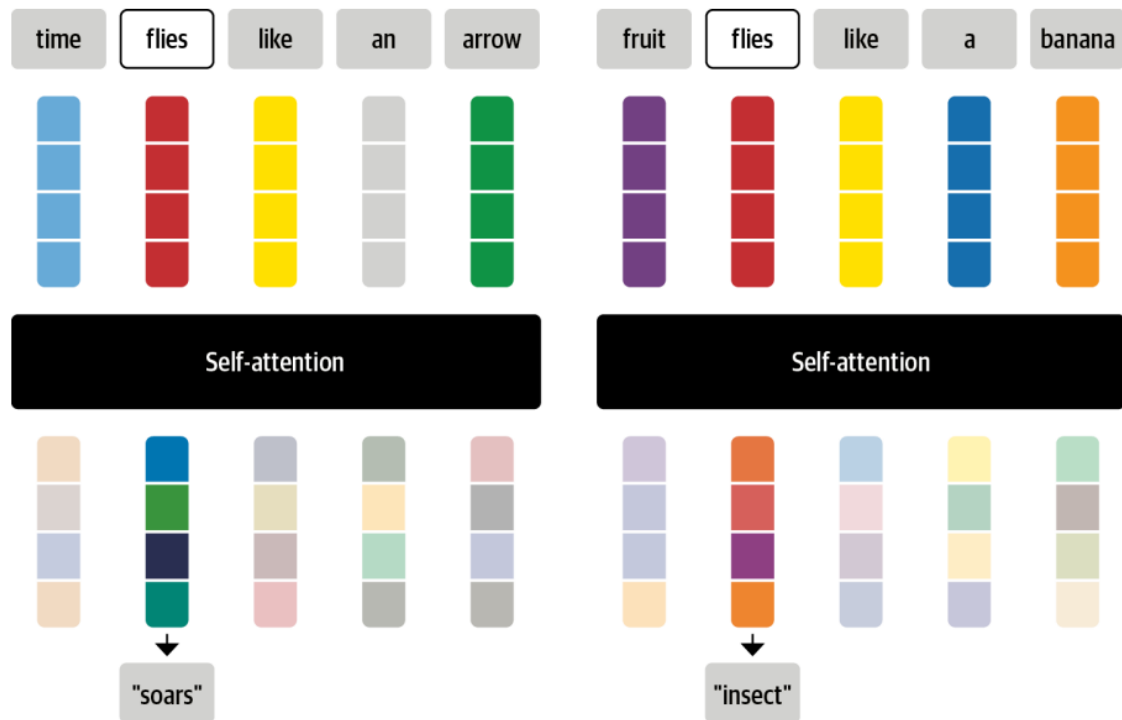


Abbildung 5: Self-Attention Mechanismus [48]

Ein weiteres zentrales Konzept ist das „transfer learning“, bei dem vortrainierte Modelle auf große Mengen von Daten auf spezifische Aufgaben angepasst werden. Ein Transformer-Modell kann auf einem umfangreichen Textkorpus vortrainiert und anschließend auf spezifische NLP-Aufgaben wie maschinelle Übersetzung oder Textklassifikation feinabgestimmt werden. Dies ermöglicht es, die in den großen Datenmengen gelernten allgemeinen Sprachstrukturen und -muster zu nutzen und die Leistung auf spezifischen Aufgaben erheblich zu verbessern [48]. [53] stellte mit „ULMFiT“ für diese Aufgabe einen allgemeinen Rahmen vor. Der Prozess besteht dabei aus bis zu drei Phasen: im *pretraining* wird versucht das nächste Wort basierend auf den vorherigen Wörtern vorauszusagen. Hierbei sind keine annotierten Daten notwendig, weshalb mit sehr großen Datensätzen wie beispielsweise der Wikipedia¹ Datenbank trainiert werden kann. Anschließend kann das Modell über eine *Domänenanpassung* an ein spezielles Umfeld angepasst werden. Im letzten Schritt, dem *fine-tuning*, wird das Modell auf die Zielaufgabe (beispielsweise NER) spezialisiert, womit das Training abgeschlossen ist [48].

¹<https://de.wikipedia.org/>

2.3.2. Technische Architektur

In diesem Abschnitt werden die Bestandteile der Transformerarchitektur vorgestellt. Die folgende Erläuterung über die Encoder- und Decoderarchitektur basiert auf [52, 48]. Für eine umfassendere Erklärung wird dem Leser empfohlen die originalen Arbeiten zu konsultieren.

Encoder

Ein Transformer-Encoder besteht aus mehreren identischen Schichten, wobei ein typischer Transformer sechs bis zwölf solcher Schichten enthält. Der Aufbau einer solchen Schicht ist in Abbildung 6 dargestellt. Jede dieser Schichten ist in zwei Hauptkomponenten unterteilt: den **Self-Attention-Mechanismus** und das **Feedforward-Netzwerk**. Der Self-Attention-Mechanismus ermöglicht es, dass jedes Wort beziehungsweise Token in einem Satz auf alle anderen Wörter „achtet“, um deren Bedeutung im gegebenen Kontext zu verstehen. Das Feedforward-Netzwerk transformiert die resultierenden Informationen und hilft dabei, komplexere, nicht-lineare Beziehungen zu modellieren. Beide Komponenten sind von Schicht-Normalisierung umgeben, und es gibt Residual-Verbindungen, die sicherstellen, dass Informationen aus früheren Schichten ungehindert fließen können.

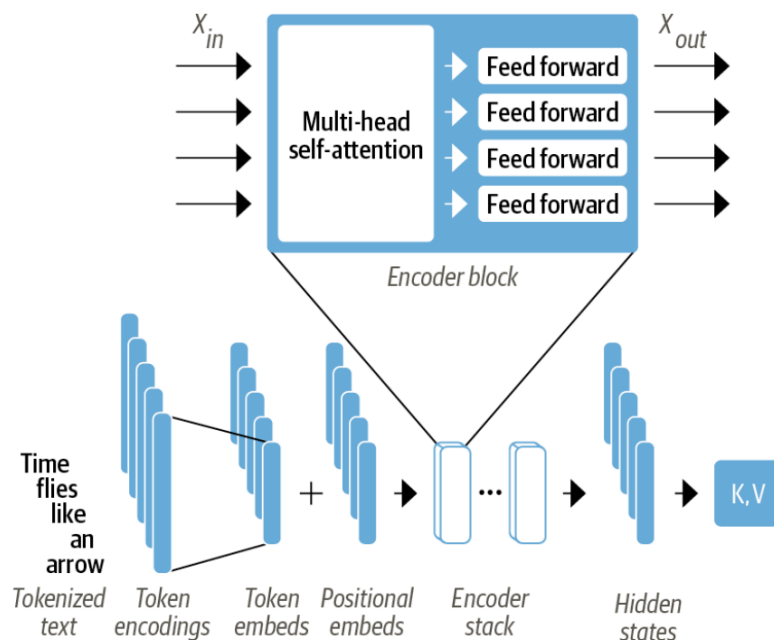


Abbildung 6: Encoder Schicht [48]

Im Self-Attention-Mechanismus werden für jedes Wort drei Vektoren berechnet: der *Query-Vektor* (Q), der *Key-Vektor* (K) und der *Value-Vektor* (V). Der Query-Vektor repräsentiert das aktuelle Wort, das gerade analysiert wird, um zu bestimmen, wie stark es auf die anderen Wörter im Satz achten sollte. Der Key-Vektor steht für die Information, auf die andere Wörter potenziell „aufmerksam“ sein könnten und der Value-Vektor enthält die eigentliche Information des Wortes, die weitergegeben wird, wenn es beachtet wird. Für jedes Token wird dann das Dot-Produkt zwischen seinem Query-Vektor und den Key-Vektoren aller anderen Tokens berechnet. Diese Berechnungen zeigen an wie stark ein bestimmtes Wort auf andere Wörter achten sollte. Anschließend werden diese Gewichte normalisiert und auf die Value-Vektoren angewendet. Dadurch kann jedes Token Informationen von den anderen Tokens sammeln, wobei stärker gewichtete Wörter mehr Einfluss haben. Ein entscheidender Vorteil des Self-Attention-Mechanismus ist seine Fähigkeit, Informationen unabhängig von der Entfernung zwischen den Wörtern im Satz zu verarbeiten. Dies steht im Gegensatz zu rekurrenten Modellen, die darauf angewiesen sind, Wörter sequentiell zu verarbeiten, was bei langen Sequenzen ineffizient ist. Durch diesen Mechanismus kann der Transformer-Encoder Beziehungen zwischen Wörtern effizient erfassen, unabhängig davon, wie weit sie voneinander entfernt sind.

Ein weiteres zentrales Konzept des Transformer-Encoders ist die **Multi-Head Self-Attention**. Anstatt nur einen einzigen Self-Attention-Prozess zu verwenden, wird dieser Prozess parallel in mehreren „Köpfen“ ausgeführt. Jeder Kopf arbeitet in einem anderen Vektorraum und kann so unterschiedliche Aspekte der Beziehungen zwischen den Wörtern erkennen. Nach der Berechnung der Self-Attention für jeden Kopf werden die Ergebnisse dieser Köpfe kombiniert. Dies ermöglicht es dem Modell, verschiedene semantische Perspektiven auf die Beziehung der Wörter zu gewinnen. Beispielsweise könnte ein Kopf die syntaktischen Abhängigkeiten zwischen Subjekt und Verb erkennen, während ein anderer Kopf die Beziehungen zwischen Adjektiven und Substantiven modelliert. Da der Transformer zudem keine rekurrente Struktur aufweist, fehlt ihm eine eingebaute Information über die Reihenfolge der Wörter im Satz. Diese Information wird durch **Positionskodierungen** ergänzt. Diese Positionscodierungen, die in der Regel auf sinus- und kosinusbasierten Funktionen beruhen, geben an, wo sich ein bestimmtes Wort im Satz befindet. Die Positionskodierungen werden zu den Eingaben addiert, sodass der Transformer-Encoder auch die Reihenfolge der Wörter berücksichtigen kann, wenn er deren Bedeutung im Kontext verarbeitet.

Nach dem Self-Attention-Prozess folgt eine Feedforward-Schicht. Diese besteht aus zwei vollständig verbundenen neuronalen Netzwerken und ist für jeden Token identisch. Diese Feedforward-Schicht fügt zusätzliche Rechenleistung hinzu, indem sie die durch Self-Attention gewonnenen Informationen nicht-linear weiterverarbeitet. Dies hilft dem Modell, komplexere Zusammenhänge zu erkennen. Neben den Hauptkomponenten des Self-Attention-Mechanismus und der Feedforward-Schicht spielen zudem **Residual-Verbindungen** und **Layer-Normalisierung** eine wichtige Rolle im Transformer-Encoder. Residual-Verbindungen ermöglichen es, dass die Ausgabe einer Schicht mit der Eingabe dieser Schicht kombiniert wird. Dadurch wird es leichter für das Modell, Informationen aus früheren Schichten zu nutzen, was insbesondere bei tiefen Netzwerken entscheidend ist. Layer-Normalisierung stabilisiert das Training, indem sie die Verteilung der Eingaben pro Schicht normalisiert, was zu einer schnelleren Konvergenz und besseren allgemeinen Leistung führt.

Die Architektur des Transformer-Encoders bietet einige entscheidende Vorteile. Zum einen ermöglicht die Parallelisierung durch den Self-Attention-Mechanismus eine schnellere Verarbeitung, da alle Wörter gleichzeitig betrachtet werden können, anstatt sie sequenziell zu verarbeiten, wie es bei rekurrenten neuronalen Netzwerken der Fall ist. Dadurch kann der Transformer wesentlich effizienter trainiert werden. Zum anderen kann der Self-Attention-Mechanismus auch langreichweitige Abhängigkeiten zwischen Wörtern erfassen, was bei der Modellierung von Kontextinformationen besonders wichtig ist. Schließlich macht die Flexibilität der Multi-Head Self-Attention das Modell besonders leistungsfähig, da es unterschiedliche Aspekte der Beziehungen zwischen den Wörtern gleichzeitig analysieren kann.

Decoder

Die Struktur des Transformer-Decoders ist der des Encoders ähnlich, jedoch mit einigen Unterschieden. Auch der Decoder besteht aus mehreren identischen Schichten, in der Regel sechs bis zwölf, die aus drei Hauptkomponenten bestehen: einem **Masked Self-Attention-Mechanismus**, einem **Encoder-Decoder-Attention-Mechanismus** und einem **Feedforward-Netzwerk**. Der Aufbau einer solchen Schicht ist dargestellt in Abbildung 7. Die erste Komponente, der Masked Self-Attention-Mechanismus, ermöglicht es dem Decoder kontextuelle Beziehungen zwischen den bereits generierten Tokens zu erkennen. Der entscheidende Unterschied zum Self-Attention-Mechanismus des Encoders be-

steht jedoch darin, dass der Decoder bei der Vorhersage eines Tokens keine Informationen aus zukünftigen Tokens verwenden darf, da er die Sequenz Schritt für Schritt generiert. Um dies zu erreichen, wird eine Maske auf die Self-Attention-Matrix angewendet, die verhindert, dass Informationen von späteren Positionen im Satz berücksichtigt werden. Dies stellt sicher, dass das Modell nur auf die bisher generierten Tokens zugreift und die Sequenz „autoregressiv“ erstellt wird.

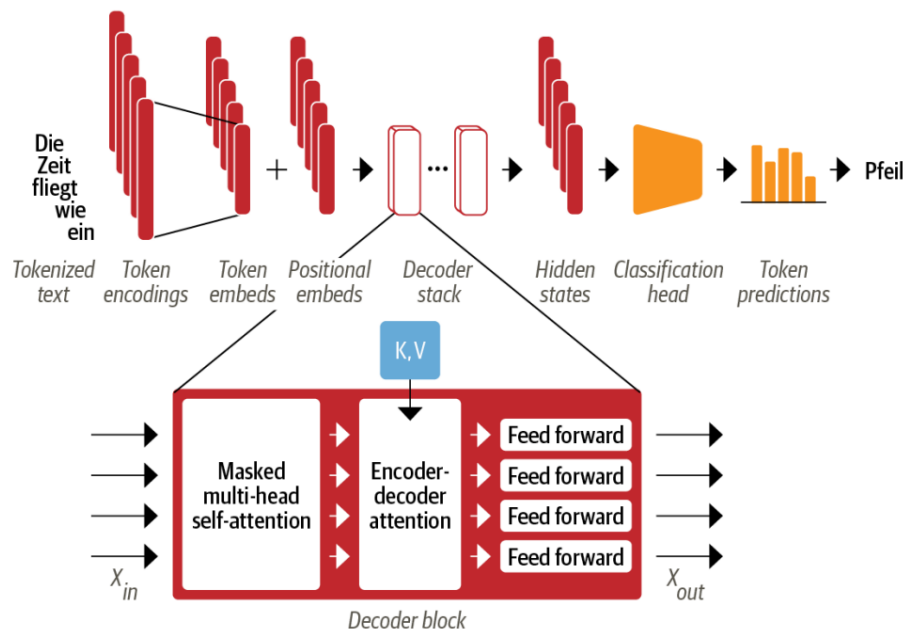


Abbildung 7: Decoder Architektur [48]

Die zweite Komponente des Decoders ist der Encoder-Decoder-Attention-Mechanismus, auch als „Cross-Attention“ bezeichnet. Dieser Mechanismus ermöglicht es dem Decoder, auf die Ausgabe des Encoders zu achten und Informationen aus der Eingabesequenz zu nutzen, um die Generierung der Zieldaten zu steuern. Technisch gesehen ähnelt dieser Prozess dem Self-Attention-Mechanismus, jedoch mit dem Unterschied, dass die Query-Vektoren aus dem Decoder und die Key- und Value-Vektoren aus dem Encoder stammen. Der Decoder lernt so welche Teile der vom Encoder verarbeiteten Eingabesequenz relevant sind, um den nächsten Token in der Zielsequenz korrekt zu erzeugen.

Nach diesen beiden Schritten folgt die Feedforward-Schicht, die auch im Decoder verwendet wird. Diese Schicht ist identisch zur Feedforward-Schicht im Encoder und besteht aus zwei linearen Transformationen, die durch eine nichtlineare Aktivierungsfunktion getrennt sind. Sie wird unabhängig für jeden Token ausgeführt und hilft, die durch den Attention-Prozess gewonnenen Informationen weiter zu verarbeiten und komplexere, nichtlineare Be-

ziehungen zu modellieren. Auch der Decoder verwendet **Residual-Verbindungen** und **Layer-Normalisierung**. Residual-Verbindungen stellen sicher, dass der Ausgang einer Schicht mit ihrem Eingang kombiniert wird, um zu verhindern, dass wichtige Informationen verloren gehen. Diese Verbindungen tragen zur Stabilität des Trainings bei, da sie es dem Modell ermöglichen, frühere Informationen leichter zu nutzen. Layer-Normalisierung hilft dabei, die Ausgabe jeder Schicht zu stabilisieren und das Training zu beschleunigen, indem sie die Eingaben pro Schicht normalisiert.

3. Bestehende Ansätze

Dieses Kapitel gibt einen Überblick über die in dieser Arbeit genutzten Ansätze zur IE. Dabie geht Kapitel 3.1 auf die verschiedenen FOSS Modelle ein, Kapitel 3.2 gibt einen Überblick über die proprietären Softwarelösungen und Kapitel 3.3 rundet das Kapitel mit einem Einblick in den aktuellen Forschungsstand auf dem Gebiet der IE ab.

3.1. FOSS Modelle

Seit der Vorstellung der Transformer-Architektur wurden eine Reihe von Modellen basierend auf dieser Architektur entwickelt. Diese können in drei Gruppen unterteilt werden: Encoder-Modelle, Decoder-Modelle und Encoder-Decoder-Modelle. Die in dieser Arbeit genutzten Modelle werden dafür kurz vorgestellt.

3.1.1. Encoder Modelle

Diese Modelle wandeln eine Eingabesequenz von Text in eine reichhaltige, numerische Darstellung um, die sich gut für Aufgaben wie die Textklassifizierung oder die Erkennung benannter Entitäten eignet. Die für ein bestimmten Token berechnete Repräsentation hängt in dieser Architektur sowohl vom vorherigen als auch vom rechten nachfolgenden Kontext ab. Dies wird als bidirektionale Aufmerksamkeit bezeichnet [48].

Bidirectional Encoder Representations from Transformers (BERT)

BERT [54] ist ein Durchbruch im NLP, das die Art und Weise revolutionierte, wie Sprachmodelle trainiert werden. BERT basiert auf der Transformer-Architektur, die auf Selbstaufmerksamkeit und voll verbundene Schichten setzt, um Beziehungen zwischen allen Wörtern eines Satzes gleichzeitig zu modellieren. Eine zentrale Innovation von BERT ist sein bidirektionaler Ansatz: Während frühere Modelle wie ELMo in der Regel unidirektional trainiert wurden, verwendet BERT ein bidirektionales Training. Dadurch wird ein tieferes Verständnis der Bedeutung eines Wortes innerhalb eines Satzes erreicht. Bei BERT stehen zwei Aufgaben im Mittelpunkt:

- Masked Language Modeling, bei welchem zufällig ausgewählte Wörter im Satz maskiert werden, während das Modell diese Wörter vorhersagen muss, indem es den bidirektionalen Kontext verwendet.
- Next Sentence Prediction, wobei Das Modell darauf trainiert wird, vorherzusagen, ob ein Satz auf einen anderen folgt, um das Verständnis von Satzbeziehungen zu verbessern.

BERT erzielte nach seiner Veröffentlichung herausragende Ergebnisse auf verschiedenen NLP-Benchmarks, wie dem GLUE-Benchmark [55], und etablierte sich als Standardmodell für viele NLP-Aufgaben.

Robustly optimized BERT approach (RoBERTa)

RoBERTa [56] ist eine optimierte Version von BERT, die mehrere Verbesserungen in den Trainingsmethoden implementiert, um die Performance von BERT zu steigern. Obwohl RoBERTa auf der gleichen Architektur wie BERT basiert, gibt es einige entscheidende Unterschiede in der Art und Weise, wie das Modell trainiert wurde:

- Verlängertes Training und größere Datenmengen: Während BERT auf einer relativ begrenzten Datenmenge trainiert wurde, nutzt RoBERTa ein viel größeres Korpus, das auch Web-Dokumente und andere textliche Quellen einschließt. Zudem wurde das Modell über einen längeren Zeitraum und mit größeren Batch-Größen trainiert, was die Fähigkeit des Modells verbesserte, komplexe Sprachmuster zu erfassen.
- Verfeinerung des Masked Language Modeling: RoBERTa verwendet eine dynamischere Maskierungsstrategie. Während BERT statische Maskierungen während des Vortrainings durchführt, ändert RoBERTa die maskierten Wörter in jedem Durchgang, was zu einer besseren Generalisierung des Modells führt.
- Abschaffung der Next Sentence Prediction: [56] fanden heraus, dass die Next Sentence Prediction in BERT nicht wesentlich zur Modellleistung beitrug. Daher wurde diese Aufgabe in RoBERTa entfernt, und das Modell fokussiert sich vollständig auf das Masked Language Modeling, um eine effizientere Nutzung der Trainingsressourcen zu ermöglichen.

- Anpassung der Hyperparameter: RoBERTa wurde mit optimierten Hyperparametern trainiert, insbesondere in Bezug auf Lernrate, Batch-Größe und die Anzahl der Trainingsschritte. Diese Anpassungen ermöglichten es dem Modell, das Potenzial der BERT-Architektur besser auszuschöpfen.

Insgesamt konnte RoBERTa durch diese Verbesserungen einige Leistungssteigerungen auf mehreren Benchmarks erzielen und übertraf BERT in vielen NLP-Aufgaben, ohne die Grundstruktur des ursprünglichen Modells zu verändern.

A Lite BERT (ALBERT)

ALBERT [57] ist ein Modell, das sich auf die Effizienzoptimierung von BERT konzentriert. BERT in seiner ursprünglichen Form ist sehr ressourcenintensiv, was es schwierig macht, solche Modelle in praktischen Anwendungen einzusetzen, welche auf begrenzte Speicher- und Rechenressourcen angewiesen sind. ALBERT versucht, diese Einschränkungen zu überwinden, indem es verschiedene Techniken zur Reduzierung der Modellgröße einführt, ohne dabei die Leistung erheblich zu beeinträchtigen:

- Parameterteilung: Eine der Schlüsselinnovationen von ALBERT ist die Wiederverwendung derselben Parameter über verschiedene Schichten hinweg. In BERT hat jede Schicht eigene Parameter, was zu einer großen Anzahl von Parametern im Modell führt. In ALBERT werden die gleichen Parameter in jeder Schicht geteilt, was die Gesamtanzahl der Parameter drastisch reduziert, ohne die Fähigkeit des Modells, komplexe Sprachmuster zu lernen, signifikant zu beeinflussen.
- Faktorisierung der Einbettungen: BERT verwendet sehr große Einbettungsvektoren, um jedes Wort zu repräsentieren, was die Anzahl der Parameter in den Eingabeschichten erhöht. ALBERT verwendet eine Faktorisierungstechnik, bei der die Einbettungen in zwei kleinere Matrizen aufgeteilt werden, was die Modellgröße weiter verringert, ohne die Genauigkeit stark zu beeinträchtigen.
- Verfeinerung der Next Sentence Prediction: Während RoBERTa die Next Sentence Prediction entfernt hat, behält ALBERT sie bei, jedoch in einer modifizierten Form. ALBERT führt dabei eine schwerere Version der Next Sentence Prediction ein, die das Modell zwingt, feinere Satzunterschiede zu erkennen, was zu einer besseren Leistung bei satzübergreifenden Aufgaben führt.

ALBERT bietet somit eine Möglichkeit, die Effizienz der BERT-Architektur zu verbessern und gleichzeitig eine ähnliche oder sogar bessere Leistung zu erzielen, insbesondere in Szenarien, in denen Rechenressourcen und Speicherkapazitäten limitiert sind. Es bietet eine flexible Lösung für industrielle Anwendungen, bei denen die Optimierung von Speicher und Geschwindigkeit entscheidend ist.

3.1.2. Decoder Modelle

Bei einer Aufforderung wie „Ich studiere an einer ...“ vervollständigen diese Modelle die Sequenz automatisch, indem sie iterativ das wahrscheinlichste nächste Wort vorhersagen. Die Repräsentation, die für ein bestimmten Token in dieser Architektur berechnet wird, hängt nur von dem vorherigen Kontext ab. Dies wird als kausale oder autoregressive Aufmerksamkeit bezeichnet [48].

Generative Pre-trained Transformer (GPT)-2

GPT-2 [58] ist ein autoregressives Sprachmodell, das auf der Decoder-Architektur basiert. Es generiert Text sequenziell, wobei es jedes neue Wort auf der Grundlage der zuvor erzeugten Wörter vorhersagt. Der Schlüsselansatz von GPT-2 ist die autoregressive Modellierung, bei der der nächste Token nacheinander vorhergesagt wird. Das Modell wurde auf einer riesigen Datenmenge von Texten aus dem Internet trainiert, was es dem Modell ermöglicht, sehr kohärente und zusammenhängende Texte zu erzeugen.

GPT-2 ist in verschiedenen Größen verfügbar, wobei die größte Version 1,5 Milliarden Parameter umfasst. Diese enorme Größe und die Breite des Trainingskorpus ermöglichen es dem Modell, eine Vielzahl von Aufgaben wie Textvervollständigung, Übersetzung, Zusammenfassung und sogar die Erstellung kreativer Texte zu bewältigen. Es zeichnet sich durch seine Fähigkeit aus, kohärente und inhaltlich konsistente Texte zu generieren, die oft sehr plausibel und menschenähnlich erscheinen. Jedoch ist GPT-2 rein unidirektional, da es die Wörter sequentiell vorhersagt, was den Kontext nur von den vorhergehenden Wörtern berücksichtigt und nicht den gesamten Satz wie bei bidirektionalen Modellen.

GPT-Neo

GPT-Neo [59] ist ein Open-Source-Projekt und wurde als Antwort auf die geschlossene Entwicklung von GPT-3 entwickelt. GPT-Neo ist ein Modell, das auf der GPT-2-Architektur basiert, aber darauf abzielt, eine offene Alternative zu bieten, die Entwicklern und Forschern zugänglich ist. GPT-Neo ist hinsichtlich der Architektur und der Funktionalität stark an GPT-2 angelehnt und verwendet ebenfalls eine autoregressive Methode zur Textgenerierung, wobei es den nächsten Token auf Basis der vorhergehenden Wörter vorhersagt.

Eine Besonderheit von GPT-Neo ist, dass es unter einer offenen Lizenz veröffentlicht wurde, was es Entwicklern ermöglicht, das Modell für verschiedene Anwendungen zu nutzen und es anzupassen, ohne an die Beschränkungen von proprietären Modellen gebunden zu sein. GPT-Neo bietet Modelle mit unterschiedlichen Parametern an, die bei textgenerativen Aufgaben wie der Erstellung von Inhalten, der Beantwortung von Fragen oder sogar beim Schreiben von Code verwendet werden können.

3.1.3. Encoder-Decoder Modelle

Diese Modelle kombinieren die Encoder und Decoder Architektur und werden für die Modellierung komplexer Zuordnungen von einer Textsequenz zu einer anderen verwendet. Sie eignen sich für maschinelle Übersetzungen und Zusammenfassungen [48].

Text-to-Text Transfer Transformer (T5)

T5 [60] ist ein besonders vielseitiges NLP-Modell, das einen neuen Ansatz verfolgt, indem es alle NLP-Aufgaben als Text-zu-Text-Probleme formuliert. Während Modelle wie BERT oder GPT sich auf spezifische Aufgaben wie Textklassifikation oder Textgenerierung konzentrieren, betrachtet T5 jede NLP-Aufgabe, wie beispielsweise Übersetzung, Zusammenfassung oder Sentimentanalyse, als eine Aufgabe der Texttransformation. Das bedeutet, dass sowohl Eingaben als auch Ausgaben in Form von Text vorliegen, unabhängig von der Art der Aufgabe.

T5 basiert auf der Transformer-Architektur, wobei es sowohl eine Encoder- als auch eine Decoder-Komponente verwendet. Das Modell wird auf große Mengen von Textdaten trainiert, die aus verschiedenen Quellen stammen, und kann durch seine flexible Architektur

auf eine Vielzahl von Aufgaben angepasst werden. Es wurde auf der sogenannten „C4“-Datenbank trainiert, einem großen, gereinigten Textkorpus. Der große Vorteil von T5 liegt in seiner Flexibilität und Generalisierungsfähigkeit, da es unterschiedliche NLP-Probleme ohne spezifische Anpassungen der Architektur bearbeiten kann. T5 war nach seiner Veröffentlichung eines der leistungsstärksten Modelle auf dem GLUE-Benchmark und zeigte eine herausragende Leistung bei Aufgaben wie Textübersetzung, QA und Textzusammenfassung.

Bidirectional and Auto-Regressive Transformer (BART)

BART [61] kombiniert die bidirektionale Textverarbeitung von BERT mit den autoregressiven Fähigkeiten von GPT. BART wird in einem zweistufigen Prozess trainiert, bei dem zunächst die Eingabe verzerrt oder beschädigt wird und das Modell dann versucht, die ursprüngliche Eingabe zu rekonstruieren. Dieser „Text-Denoising“-Ansatz macht BART besonders effektiv bei Aufgaben, die eine Textrekonstruktion erfordern, wie beispielsweise die Textzusammenfassung oder die maschinelle Übersetzung.

Das Modell verwendet einen bidirektionalen Encoder wie BERT, der den gesamten Kontext eines Satzes betrachtet, und einen autoregressiven Decoder wie GPT, der Text sequenziell generiert. Diese Kombination von bidirektionalen und autoregressiven Trainingsmethoden verleiht BART eine besondere Stärke bei der Generierung und Rekonstruktion von Texten. Das Modell wurde auf große Textkorpora vortrainiert und zeigt eine hervorragende Leistung bei einer Vielzahl von Aufgaben, darunter auch die Bearbeitung verzerrter Eingaben und die Generierung von zusammenhängenden Texten. BART hat sich besonders in den Bereichen Textzusammenfassung und maschinelle Übersetzung als erfolgreich erwiesen und bietet eine effiziente Methode zur Textgenerierung und -rekonstruktion.

XLNet

XLNet [62] ist ein weiteres Sprachmodell, das als Alternative zu BERT entwickelt wurde, um einige seiner Einschränkungen zu überwinden. Während BERT ein bidirektionales Training verwendet, um den Kontext eines Wortes aus beiden Richtungen zu verstehen, verwendet XLNet eine neue Trainingsmethode namens „Permutation-based Training“. Anstatt wie BERT den Kontext starr von links und rechts, beziehungsweise rechts nach links, zu betrachten, berechnet XLNet den Kontext durch zufällige Permutationen der Eingabe-

sequenzen. Das bedeutet, dass XLNet für jedes Wort die Wortreihenfolge flexibel variieren kann und somit alle möglichen Wortfolgen berücksichtigt.

Durch diesen Ansatz vermeidet XLNet das Problem der maskierten Tokens, die in BERT nicht auf sich selbst verweisen können, und verbessert die Modellleistung, indem es alle möglichen Wortreihenfolgen und deren Abhängigkeiten betrachtet. Dies ermöglicht es XLNet, bessere Ergebnisse bei verschiedenen NLP-Aufgaben wie Textklassifikation, QA und Textgenerierung zu erzielen. Darüber hinaus baut XLNet auf der Transformer-XL-Architektur auf, die es dem Modell ermöglicht, Langzeitabhängigkeiten über sehr lange Textsequenzen hinweg zu berücksichtigen, was es besonders nützlich für Aufgaben macht, die eine langfristige Kontextbeziehung erfordern.

3.2. Proprietäre Softwarelösungen

Mehrere Softwareanbieter bieten Tools an, die Unternehmen und Forscher bei der Informationsextraktion und anderen Aufgaben unterstützen. Dieser Abschnitt gibt einen kurzen Überblick über die in dieser Studie genutzten Anbieter.

- **Google** ist ein globaler Technologiekonzern, der für seine Suchmaschine bekannt ist, aber auch eine führende Rolle in der Forschung zu künstlicher Intelligenz und maschinellem Lernen spielt. Aufgrund seiner umfassenden Erfahrung in der Datenverarbeitung hat Google über Google AI stark in NLP Prozesse investiert. Googles Vertex AI¹ spiegelt dies wider und bietet ausgefeilte Tools für Aufgaben wie Sentiment-Analyse, Entity-Erkennung und Syntax-Parsing, welche zum Verstehen und Interpretieren großer Textmengen eingesetzt werden können.
- **Amazon** ist einer der größten Online-Einzelhändler und Anbieter von Cloud Computing über Amazon Web Services (AWS). Amazon Comprehend², sein NLP-Angebot, wurde entwickelt, um mithilfe von maschinellem Lernen Erkenntnisse aus Text zu gewinnen. Der Service lässt sich nahtlos in AWS integrieren und bietet Unternehmen eine skalierbare, zugängliche Plattform für die Extraktion von Informationen, die Analyse von Sentiment und die Erkennung von Entitäten, wodurch er sich ideal für Anwendungen eignet, die von Kundenfeedback bis hin zu umfangreichem Data Mining reichen.

¹<https://cloud.google.com/natural-language>

²<https://aws.amazon.com/de/comprehend/>

- **Microsoft** ist eines der größten Technologieunternehmen der Welt mit Schwerpunkt auf Unternehmenssoftware, Cloud Computing und KI-Forschung. Über seine Azure-Plattform bietet Microsoft eine breite Palette von KI-Diensten an, die darauf abzielen, Geschäftsprozesse zu automatisieren und zu verbessern. Azure Cognitive Services for Language¹ ist ein Teil dieses Netzwerkes, welches zur Dokumentenverarbeitung durch Extraktion von Daten aus unstrukturiertem Text genutzt werden kann.
- **SAP** ist einer der weltweit führenden Anbieter von Unternehmenssoftware und vor allem für seine ERP-Systeme bekannt, die Unternehmen bei der Verwaltung von Geschäftsabläufen und Kundenbeziehungen unterstützen. SAP Business Entity Recognition² spiegelt das Engagement der SAP bei der Automatisierung von Geschäftsprozessen wider, indem NLP-Funktionen direkt in das Software-Ökosystem integriert werden. Durch die Möglichkeit, wichtige Entitäten aus Texten zu extrahieren, hilft SAP Unternehmen, strukturierte Erkenntnisse aus unstrukturierten Datenquellen zu gewinnen und so die Abläufe in großen Unternehmen zu optimieren.
- **OpenAI**³ ist eine Forschungsorganisation und ein Unternehmen, welches für seine bahnbrechende Arbeit im Bereich des maschinellen Lernens und der natürlichsprachlichen Modelle bekannt ist. OpenAI hat einige der weltweit fortschrittlichsten KI-Modelle entwickelt, darunter die GPT-Serie. Die Modelle von OpenAI, wie beispielsweise GPT-4, sind in verschiedenen Anwendungen weit verbreitet, darunter in der Texterstellung, der Sprachübersetzung und der Informationsextraktion. Für diese Arbeit wurde das Modell „babbage-002“ verwendet.

Das Training und die Bereitstellung der Modelle der proprietären Software erfolgte nach der jeweiligen offiziellen Dokumentation der Anbieter. Die durchgeführten Tests sind in dem öffentlichen Notebook⁴ zu finden.

¹<https://language.cognitive.azure.com/>

²<https://www.sap.com/products/technology-platform/business-entity-recognition.html>

³<https://cookbook.openai.com/topic/text>

⁴<https://colab.research.google.com/drive/1AYPCyhwyxqx1ELD4Q72r20yKIIYpfsIo?usp=sharing>

3.3. Stand der Forschung

Obwohl durchgehend viel an neuen Methoden und Modellen gepfeilt wird, da der Bereich des IE vor einer Reihe technischer und methodischer Herausforderungen steht, welche die Entwicklung und den breiten Einsatz dieser Technologie erschweren. Ein wesentliches Problem besteht im Mangel an Trainingsdaten für viele domänenspezifische Anwendungen, da oftmals die notwendigen, qualitativ hochwertigen Datensätze fehlen, um leistungsstarke Modelle zu trainieren [17]. Zudem fehlen in einigen Datensätzen wichtige, relevante Beispiele, was die Generalisierbarkeit der Modelle weiter einschränkt, wodurch Modelle, die in einem bestimmten Bereich trainiert wurden, schwer auf andere Bereiche übertragbar sind. Dies limitiert die Flexibilität und Anwendungsmöglichkeiten solcher Modelle in interdisziplinären oder dynamischen Umgebungen [63]. Ein weiteres zentrales Problem stellt die Interpretierbarkeit von IE-Modellen dar, da diese als „Black Boxes“ konzipiert sind, wodurch deren Entscheidungsprozesse für Anwender nur schwer nachzuvollziehen sind [64]. Das Phänomen des katastrophischen Vergessens stellt eine weitere zentrale Herausforderung dar. Hierbei handelt es sich um das Problem, dass Modelle bei fortlaufendem Training auf neuen Daten dazu neigen, zuvor erlerntes Wissen zu „vergessen“. Dies ist besonders kritisch in Szenarien, in denen Modelle kontinuierlich neue Daten integrieren müssen, ohne dabei frühere Lerninhalte zu verlieren [63]. Ein weiteres Problem betrifft die Komprimierung von Modellen. Während viele Deep Learning-Modelle sehr leistungsfähig sind, erfordern sie in der Regel erhebliche Rechenressourcen und Speicherkapazitäten [64]. Ein weiteres wichtiges Forschungsfeld betrifft die Robustheit von Modellen, insbesondere im Umgang mit verrauschten oder inkonsistenten Daten. In vielen realen Anwendungen sind Daten nicht perfekt und enthalten Fehler oder Ungenauigkeiten. Modelle müssen in der Lage sein, auch unter solchen Bedingungen robuste und zuverlässige Vorhersagen zu treffen [2]. Auch die kontextuelle IE gewinnt zunehmend an Bedeutung. Es reicht oft nicht aus, isolierte Informationen zu extrahieren; vielmehr muss der breitere Kontext, in dem diese Informationen stehen, berücksichtigt werden, um ein vollständiges Verständnis zu gewährleisten [2]. Schließlich bleibt die multilinguale Modellierung eine Herausforderung. Viele Deep Learning-Modelle sind auf die englische Sprache beschränkt, was ihre Anwendung in mehrsprachigen oder nicht-englischen Kontexten einschränkt. Die Entwicklung multilingualer Modelle, die zuverlässig in verschiedenen Sprachen arbeiten, ist daher eine wichtige Forschungslücke, die geschlossen werden muss, um die globale Nutzbarkeit und Zugänglichkeit von Deep Learning zu verbessern [2].

Neben diesen Herausforderungen zeichnen sich jedoch auch einige zentrale Trends in der Weiterentwicklung der IE ab. Eine der größten Trends war das Transferlernen, welches durch Transformermodelle ermöglicht wurde. Hierbei werden Modelle auf einem sehr großen Datensatz über viele Domänen hinweg vortrainiert und später auf einem kleineren, domänenspezifischen Datensatz nach-trainiert [64]. Dieser Ansatz trägt dazu bei, den Mangel an Trainingsdaten in der Zieldomäne zu verringern. Darüber hinaus reduziert Transferlernen die zum Trainieren des Modells erforderlichen Rechenressourcen und trägt zu einer schnelleren Konvergenz bei [64]. Eine bemerkenswerte Entwicklung ist die zunehmende Nutzung von Cloud-basierten Plattformen für die Entwicklung und das Training von Deep Learning-Modellen [63]. Diese Plattformen bieten nicht nur skalierbare Rechenleistung und Speicherressourcen, sondern erleichtern auch die Zusammenarbeit und den Austausch großer Datenmengen. Einige dieser Anbieter sind in Abschnitt 3.2 beschrieben. Ein weiterer bedeutender Trend ist die multimodale Informationsextraktion, bei der Daten aus verschiedenen Modalitäten – wie Text, Bild und anderen Quellen – integriert und gemeinsam verarbeitet werden, um eine umfassendere und genauere Informationsgewinnung zu ermöglichen [65]. Dies eröffnet neue Möglichkeiten für die Analyse komplexer Datenquellen und verbessert die Modellleistung in vielen Anwendungsfeldern. „Few-shot“ und „Zero-shot“ Learning gewinnen ebenfalls zunehmend an Bedeutung [65]. Diese Methoden zielen darauf ab, Modelle zu entwickeln, die mit sehr wenigen (Few-shot) oder gar keinen (Zero-shot) domänenspezifischen Trainingsdaten auskommen. Dies könnte die Abhängigkeit von großen, annotierten Datensätzen verringern und die Anwendbarkeit von Modellen in Bereichen mit begrenzten Datenressourcen erheblich erweitern. Parallel dazu rückt die Entwicklung von erklärbarer künstlicher Intelligenz (Explainable AI) verstärkt in den Fokus, mit dem Ziel, die Interpretierbarkeit und Transparenz von IE-Modellen zu verbessern [66].

4. Lösungsansatz

In diesem Abschnitt wird der Lösungsansatz beschrieben, mit der verschiedene FOSS-Modelle mit kommerzieller Software verglichen wird. Nach einem Einblick in die Methodik in Abschnitt 4.1 wird in Abschnitt 4.2 der für diese Studie ausgewählte Datensatz beschrieben. Der Datensatz wird zunächst mehreren vorbereitenden Schritten unterzogen, wie in Abschnitt 4.3 beschrieben, um die Genauigkeit der Ergebnisse zu gewährleisten. Anschließend wird in Abschnitt 5.1 die Implementation der FOSS-Modelle beschrieben. Abschließend werden in Abschnitt 4.4.1 die zur Bewertung der Leistung und Kosteneffizienz der Modelle verwendeten Metriken erörtert. Das leistungsstärkste FOSS-Modell, das anhand der Bewertungsmetriken ermittelt wurde, wird dann mit der kommerziellen Software verglichen, um festzustellen, wie es im Vergleich zu diesen abschneidet.

4.1. Methodik

Die Methodik dieser Masterarbeit ist darauf ausgelegt, einen systematischen Ansatz zur Analyse und Evaluierung von Verfahren zur Umwandlung unstrukturierter Daten in strukturierte Informationen zu verfolgen. Der Prozess beginnt mit der Auswahl eines geeigneten Datensatzes. Der ausgewählte Datensatz dient als Grundlage für die weiteren Schritte und sollte eine Vielzahl von unstrukturierten Informationen enthalten, um die verschiedenen Ansätze umfassend zu testen. Im Anschluss an die Auswahl des Datensatzes erfolgt die Vorbereitung der Daten. Dieser Schritt beinhaltet die Bereinigung und Transformation der unstrukturierten Daten, um sie in ein Format zu bringen, das für die Modellentwicklung geeignet ist. Diese Phase ist entscheidend, um die Qualität der späteren Modelle und die Validität der Ergebnisse zu gewährleisten.

Nachdem die Daten vorbereitet sind, werden verschiedene Modelle auf dem Datensatz trainiert und getestet. Hierbei werden die Ansätze NER und QA untersucht. Die Performance der trainierten Modelle wird mithilfe ausgewählter Metriken bewertet, wobei der F1-Score eine zentrale Rolle spielt. Nach der Evaluierung der Modelle wird das leistungsfähigste FOSS, basierend auf dem höchsten F1-Score, ausgewählt. Dieses Modell wird anschließend in einem Kostenvergleich gegen proprietäre Softwarelösungen gestellt. Durch diesen Vergleich wird untersucht, inwiefern die FOSS-Modelle hinsichtlich ihrer Effizienz und Wirtschaftlichkeit mit kommerziellen Lösungen konkurrieren können.

Zusätzlich wird die Generalisierbarkeit des besten FOSS-Modells getestet, indem es auf verschiedenen Domänen angewendet wird. Dieser Schritt ist entscheidend, um festzustellen, ob die gewonnenen Ergebnisse über den ursprünglichen Datensatz hinaus gültig sind und ob das Modell in unterschiedlichen Anwendungsbereichen eingesetzt werden kann. Die Durchführung dieser Tests ermöglicht es, die Robustheit und Adaptierbarkeit des Modells zu bewerten und bietet wertvolle Erkenntnisse darüber, wie gut es sich in verschiedenen Kontexten bewährt.

4.2. Beschreibung des Datensatzes

Es existieren bereits zahlreiche Studien [55, 67], die sich mit der NER und der QA unter Verwendung bekannter Datensätze wie SQuAD, NewsQA, CoNLL-2003 und FiNER befassen. Diese Studien fokussieren sich oft auf gut erforschte Domänen wie Nachrichten, Wissensdatenbanken (z.B. Wikipedia), den Finanzsektor, Film- oder Seriensdatenbanken sowie soziale Netzwerke. Die dabei verwendeten Modelle werden häufig in englischer Sprache trainiert. Um sich von diesen Arbeiten abzuheben, wurde für die vorliegende Studie ein Datensatz¹ juristischer Dokumente in deutscher Sprache gewählt. Dieser Datensatz wurde in der Arbeit von [68] vorgestellt. Juristische Dokumente unterscheiden sich von Texten in anderen Bereichen sowohl in textinternen als auch in textexternen Kriterien, was erhebliche Auswirkungen auf die sprachliche und thematische Gestaltung sowie die Struktur hat [69]. Diese Studie zielt darauf ab, diese Nischendomäne zu adressieren und die Modelle durch diese besondere Herausforderung zu evaluieren. Der Datensatz besteht aus Entscheidungen verschiedener deutscher Bundesgerichte mit Annotationen von Entitäten, die auf Rechtsnormen, Gerichtsentscheidungen, juristischer Literatur und anderes verweisen. Auf den Datensatz wurde am 02.07.2024 zugegriffen.

¹<https://huggingface.co/datasets/elenanereiss/german-ler>

Der Datensatz besteht aus 750 deutschen Gerichtsentscheidungen von 2017 bis 2018, die online im Portal „Rechtsprechung im Internet“¹ veröffentlicht sind. Dabei handelt es sich um anonymisierte Gerichtsentscheidungen des Bundesverfassungsgerichts, Bundesverwaltungsgerichts, Bundesfinanzhofs, Bundesarbeitsgerichts, Bundessozialgerichts, Bundespatentgerichts und des Bundesgerichtshofs. Eine wesentliche Besonderheit der veröffentlichten Beschlüsse ist, dass alle personenbezogenen Daten aus Datenschutzgründen anonymisiert sind. Zudem wurde der Datensatz von der ursprünglichen Dokumentenstruktur in eine Satzstruktur überführt, womit nicht weiter ersichtlich ist, welcher Satz aus welchem Dokument stammt. Es liegen insgesamt 66.722 Sätze und 2.157.048 Tokens aus den verschiedenen Gerichten vor. Um ein besseres Verständnis für den Datensatz zu schaffen, wurde ein Beispielsatz aus dem Datensatz extrahiert:

„Die Angemessenheit der Verfahrensdauer ist dabei stets im Lichte der aus Art. 2 Abs. 1 iVm., Art. 20 Abs. 3 GG **GS** und Art. 19 Abs. 4 GG **GS** sowie Art. 6 Abs. 1 EMRK **EUN** folgenden Verpflichtung des Staates, Gerichtsverfahren in angemessener Zeit zum Abschluss zu bringen, zu beurteilen (BGH 13. Februar 2014 - III ZR 311/13 - Rn. 27 **RS** mwN).“

¹<https://www.rechtsprechung-im-internet.de/jportal/portal/page/bsjrsprod.psm1>

Der Datensatz kann zudem in einer feinkörnigen Variante mit 19 Entitätskategorien oder einer grobkörnigen Variante mit sieben Entitätskategorien verwendet werden. Um die Modelle möglichst genau zu analysieren, beschäftigt sich diese Studie mit der feinkörnigen Variante. Der Aufteilung der Entitäten in die verschiedenen Kategorien findet sich im folgenden:

Label	Anzahl
RS <i>Gerichtsentscheidung</i>	194.601
GS <i>Gesetz</i>	116.934
LIT <i>Juristische Literatur</i>	38.119
VT <i>Vertrag</i>	15.227
EUN <i>EU-Rechtsnorm</i>	12.520
INN <i>Institution</i>	6.872
GRT <i>Gericht</i>	5.981
VO <i>Verordnung</i>	5.238
VS <i>Regulierung</i>	4.702
ORG <i>Organisation</i>	2.771
UN <i>Unternehmen</i>	2.384
PER <i>Person</i>	2.262
LD <i>Land</i>	1.752
RR <i>Richter</i>	1.625
ST <i>Stadt</i>	772
MRK <i>Marke</i>	666
STR <i>Straße</i>	265
LDS <i>Landschaft</i>	231
AN <i>Anwalt</i>	160
Gesamt	2.157.048

4.3. Datenvorbereitung und -verarbeitung

Um die Bedingungen für die NER und die QA Systeme zu vereinheitlichen, war es erforderlich, den ursprünglichen Datensatz umfassend zu bearbeiten. Der gewählte Datensatz war ursprünglich für Anwendungen im Bereich der NER konzipiert, weshalb eine Anpassung erforderlich war, um ihn für QA-Systeme nutzbar zu machen. Der Prozess der Datenvorbereitung findet sich in diesem Notebook¹

Im Zuge der Aufarbeitung des Datensatzes wurden zunächst irrelevante, redundante und verrauschte Daten entfernt. Diese Maßnahmen dienten der Sicherstellung, dass die verbleibenden Dateneinträge den Anforderungen eines QA-Systems entsprechen. Da viele QA-Systeme bei der Beantwortung von Fragen zu einem bestimmten Kontext nur eine einzelne Antwort zurückgeben können (beispielsweise: „Wer ist die Person im Haus?“ – Antwort: „Max Müller“; im Gegensatz zu „Wer sind die Personen im Haus?“ – Antworten: „Max Müller, Anna Schmidt“), mussten sämtliche Einträge, in denen dieselbe Entität mehrfach vorkam, aus dem Datensatz entfernt werden. Diese Bereinigung führte zu einer Reduktion des Datensatzes um 13,58%.

Zusätzlich wurden alle Einträge, die keine Entitäten enthielten, aus dem Datensatz entfernt. Während solche Einträge für NER-Systeme möglicherweise nützlich sein könnten, da sie potenziell wertvolle Informationen zur Identifizierung von Entitäten enthalten, sind sie für QA-Systeme nicht von Relevanz. QA-Systeme benötigen spezifische Entitäten zur Beantwortung von Fragen im Kontext, und Einträge ohne Entitäten bieten keinen Mehrwert für die Beantwortung solcher Fragen. Daher war es notwendig, diese Einträge zu eliminieren, um die Homogenität der Trainingsdaten beider Ansätze zu gewährleisten. Dieser Schritt führte zu einer weiteren Reduktion des Datensatzes um 63,38%. Die Verteilung der Entitäten hat sich dadurch wie folgt verändert:

¹https://colab.research.google.com/drive/1gbNry7EzEigLZJ0tC7f0TnCEbm2P8irw?usp=drive_link

Label	Anzahl
RS <i>Gerichtsentscheidung</i>	71.552
GS <i>Gesetz</i>	55.558
LIT <i>Juristische Literatur</i>	12.487
VT <i>Vertrag</i>	7.965
EUN <i>EU-Rechtsnorm</i>	5.686
INN <i>Institution</i>	4.668
GRT <i>Gericht</i>	3.153
VO <i>Verordnung</i>	2.658
VS <i>Regulierung</i>	2.880
ORG <i>Organisation</i>	1.079
UN <i>Unternehmen</i>	1.358
PER <i>Person</i>	1.312
LD <i>Land</i>	975
RR <i>Richter</i>	1.217
ST <i>Stadt</i>	415
MRK <i>Marke</i>	422
STR <i>Straße</i>	153
LDS <i>Landschaft</i>	126
AN <i>Anwalt</i>	101
Gesamt	724.914

Für QA-Systeme müssen die Daten im Frage-Antwort-Format vorliegen. Zur Schaffung solcher Datensätze wurden die Entitäten als Antworten definiert und jeweils mit einer entsprechenden Frage versehen. Um eine Diversität [70] zu gewährleisten wurden die Fragen aus einem Pool von drei verschiedenen Fragen zu jeder Entität ausgewählt. Ein Beispiel für diesen Umwandlungsprozess ist wie folgt dargestellt:

NER

„Die **Polizei Bremen INN** stellte mehrere Speicherkarten im Haus sicher.“

QA

Kontext: „Die Polizei Bremen stellte mehrere Speicherkarten im Haus sicher.“

Frage: „Wie heißt die Institution, die erwähnt wird?“; Antwort: „Polizei Bremen“

Abschließend wurden die Daten in Trainings- und Testdatensätze aufgeteilt, wobei ein Verhältnis von 85 zu 15 gewählt wurde. Diese Unterteilung stellt sicher, dass jedes Modell mit denselben Trainingsdaten trainiert und mit denselben Testdaten evaluiert wird. Nach der allgemeinen Datenvorbereitung wurden ausschließlich die erforderlichen Schritte zur Anpassung der Daten an das spezifische Format des Modells vorgenommen, basierend auf der Dokumentation von Huggingface¹.

Wie erkennbar ist, bestehen erhebliche Unterschiede in der Anzahl der Entitäten pro Klasse. Diese Diskrepanz ermöglicht es zu untersuchen ob das Modell in der Lage ist Entitäten auch unter Bedingungen knapper Trainingsdaten zuverlässig zu identifizieren. Die Analyse der Modellergebnisse in Bezug auf Klassen mit geringer Anzahl kann daher wertvolle Rückschlüsse auf die Robustheit und Generalisierungsfähigkeit des Modells ziehen.

4.4. Bewertungsmetriken

In diesem Abschnitt werden die Bewertungsmetriken und -methoden untersucht, die zum Vergleich der Leistung der in Abschnitt 5.1 vorgestellten FOSS-Modelle und der in Abschnitt 3.2 vorgestellten proprietären Software verwendet werden. Die Evaluation ist ein entscheidender Aspekt eines jeden Projekts des maschinellen Lernens, da sie Aufschluss darüber gibt wie gut die Modelle funktionieren und ob sie die gewünschten Leistungskriterien erfüllen. Zunächst werden die grundlegenden Konzepte der Leistungskennzahlen und wie diese für die Bewertung der Modelle genutzt werden in Abschnitt 4.4.1 erörtert. In Abschnitt 4.4.2 werden die Metriken und Berechnungen beschrieben, die für den Kostenvergleich der Modelle verwendet werden. In diesem Abschnitt wird die Bedeutung dieser Metriken für die Auswahl des am besten geeigneten Modells für die jeweilige IE-Aufgabe diskutiert. Die Erörterung sowohl der leistungsbezogenen als auch der kostenbezogenen Bewertungsmethoden bietet einen umfassenden Rahmen für die Bewertung und den Vergleich der Modelle, der es den Unternehmen ermöglicht, fundierte Entscheidungen über das für die jeweilige Aufgabe am besten geeignete Modell zu treffen.

¹<https://huggingface.co/docs/datasets/index>

4.4.1. Leistungskennzahlen

Bei der Bewertung von maschinellen IE-Modellen geben verschiedene Metriken Aufschluss über die Effektivität des Modells. Der folgende Abschnitt bietet einen kurzen Überblick über die verwendeten kritischen Metriken. Die Diskussion bezieht sich auf [71] und [72]; dem Leser wird empfohlen für ein umfassenderes Verständnis die Originalarbeiten zu konsultieren.

In diesem Abschnitt wird zunächst ein Verständnis von True Positive (TP), True Negative (TN), False Positive (FP) und False Negative (FN) geschaffen. Bei der binären Klassifizierung der Antworten ist die Berechnung dieser Metriken relativ trivial:

- **TP:** Die Anzahl der positiven Instanzen, die das Modell korrekt als positiv klassifiziert hat
- **TN:** Die Anzahl der negativen Instanzen, die das Modell korrekt als negativ klassifiziert hat
- **FP:** Die Anzahl der negativen Instanzen, die das Modell fälschlicherweise als positiv klassifiziert hat
- **FN:** Die Anzahl der positiven Instanzen, die das Modell fälschlicherweise als negativ klassifiziert hat.

Eine Konfusionsmatrix, eine Kreuztabelle, die die Anzahl der Übereinstimmungen zwischen zwei Bewertern aufzeichnet, visualisiert diese Metriken, wie in Abbildung 8 gezeigt.

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Abbildung 8: Binäre Konfusionsmatrix [73]

NER kann als Problem der Tokenklassifizierung betrachtet werden, bei welcher jedem Token eine „Klasse“, wie beispielsweise „Person“ oder „Organisation“, zugeordnet wird. Bei dieser Mehrklassenklassifizierung werden die genannten Metriken erweitert, indem die TP, TN, FP und FN für jede Klasse einzeln berechnet werden und diese Werte dann aggregiert werden, um ein einziges Maß für die Vorhersageleistung des Modells zu erhalten. Eine Standardmethode besteht darin, jede Klasse als ein binäres Klassifizierungsproblem zu betrachten, wobei die Instanzen dieser Klasse positiv und die Instanzen aller anderen Klassen negativ sind. Dieser Ansatz ermöglicht die Berechnung von TP, TN, FP und FN für jede Klasse und die anschließende Mittelung dieser Werte über alle Klassen, um die durchschnittliche Vorhersageleistung des Modells zu messen.

Auf der Grundlage dieser Konfusionsmatrixmetriken kann eine breite Palette von Vorhersageleistungsmaßen abgeleitet werden, die eine umfassende Bewertung der Vorhersageleistung eines Klassifikators ermöglichen. In dieser Arbeit werden mehrere dieser Maße zur Bewertung der Modelle verwendet. Im Folgenden wird ein kurzer Überblick über die binären Maße gegeben:

- **Präzision:** Die Genauigkeit gibt an, wie sehr man dem Modell vertrauen kann, wenn es eine Instanz als positiv vorhersagt:

$$\text{Präzision} = \frac{TP}{TP + FP} \quad (1)$$

- **Recall:** Recall misst die Vorhersagegenauigkeit des Modells für die positive Klasse: intuitiv misst es die Fähigkeit des Modells, alle positiven Instanzen im Datensatz zu finden:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

- **F1-Score:** Der F1-Score kann als gewichteter Durchschnitt von Precision und Recall betrachtet werden. Der harmonische Mittelwert kann zur Bestimmung des optimalen Kompromisses verwendet werden, um die relativen Beiträge von Precision und Recall zum F1-Score auszugleichen:

$$F1\text{-Score} = 2 * \left(\frac{\text{Präzision} * \text{Recall}}{\text{Präzision} + \text{Recall}} \right) \quad (3)$$

Desweiteren ist zu beachten wie der F1-Score über die verschiedenen Klassen berechnet wird. Der „Microaverage“ F1-Score wird durch Aggregation der TP, FP und FN über alle Klassen hinweg berechnet. Dies bedeutet, dass alle Klassifizierungsfehler und -erfolge global summiert werden, bevor die F1-Score-Berechnung erfolgt. Dieser Ansatz eignet sich besonders gut für Szenarien, in denen die Klassengrößen stark variieren, da er das globale Leistungsmaß unabhängig von der Anzahl der Klassen berücksichtigt. Im Gegensatz zum Microaverage F1-Score wird beim „Makroaverage“ F1-Score der F1-Score für jede Klasse einzeln berechnet und anschließend der Durchschnitt dieser Scores ermittelt. Die Makroaverage-Berechnung gewichtet jede Klasse gleich, unabhängig von deren Häufigkeit im Datensatz. Dies ist besonders nützlich, wenn das Ziel darin besteht die Leistung des Modells auf allen Klassen gleichmäßig zu betrachten, was zu einer höheren Sensibilität für seltene Klassen führt. Der „Weighted Average“ F1-Score berücksichtigt die relative Häufigkeit jeder Klasse und gewichtet die F1-Scores entsprechend. Dieser Ansatz ist besonders nützlich wenn die Klassenverteilung ungleich ist, da er die Leistung des Modells entsprechend der Verteilung der Klassen gewichtet und somit ein besseres Gesamtbild der Modellleistung in Bezug auf die tatsächliche Verteilung der Daten liefert. Für die Bewertung der Modelle wird in der Arbeit der Weighted Average F1-Score angegeben. In den entsprechenden Notebooks kann jedoch der Microaverage F1-Score und der Makroaverage F1-Score, sowie die Metriken der einzelnen Klassen vorgefunden werden. Zusätzlich zu den zuvor genannten Leistungskennzahlen für die Vorhersage wird die Dauer des Modelltrainings und der Inferenz erfasst, um einen Einblick in die Recheneffizienz zu erhalten.

4.4.2. Kostenvergleich

Dieser Abschnitt gibt einen umfassenden Überblick über die Vergleichsmethode, die zur Bewertung der Kosteneffizienz verschiedener Produkte zur IE verwendet wird. Der Vergleichsprozess beinhaltet die Bewertung der drei Hauptkosten, die mit den Produkten verbunden sind, insbesondere die Trainingskosten, die Hosting-Kosten und die Inferenzkosten, die in den meisten Fällen den Gesamtpreis der Produkte ausmachen. In dieser Arbeit wird ein Zeitraum von 30 Tagen simuliert, in dem das IE-Modell am ersten Tag trainiert, während der gesamten Dauer gehostet wird und die Inferenzkosten für unterschiedliche Dokumentenmengen berechnet werden. Darüber hinaus werden die Hosting-Kosten für den gesamten Zeitraum zu den Trainingskosten am Anfang addiert, um einen besseren

Überblick zu erhalten. Das Hauptziel dieses Vergleichs besteht darin das kosteneffizienteste Produkt für die IE zu ermitteln, wobei unterschiedliche Dokumentenmengen für einen Monat berücksichtigt werden. Die Preise für Training, Hosting und Inferenz wurden den Websites der jeweiligen Anbieter entnommen, welche in Abschnitt 3.2 zu finden sind.

Darüber hinaus wird in dieser Arbeit gemäß der in Abschnitt 4 vorgestellten Methodik die proprietäre Software mit dem leistungsstärksten Modell der FOSS-Modelle verglichen. Dieser Vergleich birgt jedoch einige Herausforderungen, da die Preisgestaltung des FOSS-Modells von der Implementierungsmethode abhängt. Daher konzentriert sich diese Studie auf eine weit verbreitete Variante - Amazon SageMaker. Amazon SageMaker ist ein vollständig verwalteter Service für maschinelles Lernen, der von AWS bereitgestellt wird und die Entwicklung, das Training und den Einsatz von maschinellen Lernmodellen in großem Maßstab erleichtert. Der Service umfasst verschiedene vorgefertigte Algorithmen und Frameworks, die ihn zu einer umfassenden Plattform für die Verwaltung des gesamten maschinellen Lernprozesses machen. Das zuvor auf Google Colab trainierte FOSS-Modell wird hierbei in den in Abbildung 9 dargestellten Workflow integriert. Der Workflow umfasst die Kosten für die Speicherung der FOSS-Modellartefakte in einem S3-Bucket, die Erstellung eines Elastic Container Registry (ECR)-Container-Images und das Hosting des Endpunkts für das Modell. Anhang C und Abschnitt 5.2.3 präsentieren eine detaillierte Analyse aller Kosten basierend auf dem AWS Preiskalkulator.

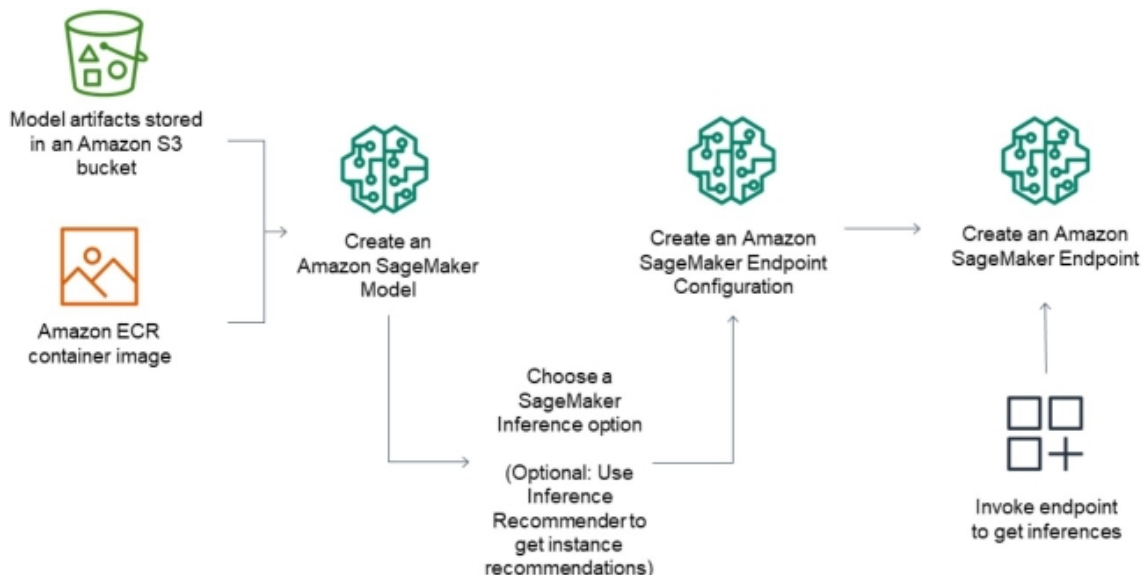


Abbildung 9: Amazon SageMaker Workflow [74]

Es muss jedoch beachtet werden, dass ein Kostenvergleich zwischen verschiedenen kommerziellen Anbietern von IE-Systemen möglicherweise nicht ganz gerecht ist, da bei diesem

Vergleich zusätzliche Faktoren, die über die Grundkosten des Dienstes hinausgehen, nicht berücksichtigt werden, wie beispielsweise die Preise für Zusatzdienste wie Cloud-Speicher, der Aufwand, den der Benutzer betreiben muss, um das Modell zu trainieren und an seine spezifischen Anforderungen anzupassen, und etwaige Verbindungspunkte, die in der Preisstruktur enthalten sein können. Diese zusätzlichen Überlegungen können eine wichtige Rolle bei den Gesamtbetriebskosten und dem allgemeinen Nutzenversprechen eines bestimmten Anbieters spielen. Dasselbe gilt für die Implementierung des FOSS-Modells, das keine zusätzlichen Kosten für Faktoren vorsieht, die über die Grundkosten des Dienstes hinausgehen, wie beispielsweise lokale/cloudbasierte Datenspeicherung, Modellpflege oder Personalkosten.

4.5. Domänenübergreifender Ansatz

Um die FOSS-Modelle umfassend zu testen, wurden diese neben dem Standard-Datensatz, beschrieben in 4.2, zusätzlich mit drei weiteren Datensätzen trainiert und evaluiert. Dabei wurden Datensätze aus verschiedenen Domänen und unterschiedlicher Größe gewählt, um eine breite Palette an Texttypen und Herausforderungen abzudecken. Die ausgewählten Domänen umfassen die öffentliche Verwaltung, die Medizinbranche und die Mobilitätsbranche. Ziel dieser zusätzlichen Untersuchung ist es, die Stärken und Schwächen der Modelle in verschiedenen Kontexten zu analysieren. Durch die Verwendung dieser unterschiedlichen Datensätze aus verschiedenen Domänen wird eine umfassendere Evaluation der Modelle ermöglicht, wobei sowohl die sprachlichen als auch die domänenspezifischen Besonderheiten berücksichtigt werden. Da das Hauptaugenmerk dieser Arbeit auf der NER liegt, wurden die FOSS Modelle nur auf diesem Gebiet mit den weiteren Datensätzen untersucht. Der Leser kann jedoch, aufgrund des öffentlichen Zugangs zu den Modellen und dem Quellcode, selbst unterschiedliche Aufgaben mit verschiedenen Datensätzen kombinieren und eigene Tests durchführen.

Der erste Datensatz stammt von [75] aus dem Jahr 2023 und umfasst medizinische Dokumente. Diese zeichnen sich durch eine stark technische Sprache aus, die vor allem durch die Nennung von Medikamenten, Anweisungen zur Verabreichung und medizinischen Abkürzungen geprägt ist. Die Dokumente beschreiben in der Regel, welches Medikament in welcher Dosierung, Form und Häufigkeit einem Patienten verabreicht werden soll. Die relevanten Entitäten in diesem Datensatz sind: *Dosage*, *Drug*, *Duration*, *Form*, *Frequency*,

Route und Strength. Der Datensatz besteht aus 7309 Trainingsdokumenten und 1209 Testdokumenten. Die Implementierung der Modelle zu diesem Datensatz findet sich in dem entsprechenden Notebook¹. Ein Beispiel für einen typischen Satz lautet:

„Er kann auch wiederkehrende Anfälle haben, die mit **Ativan Drug** **IV Route** oder **IM Route** behandelt werden sollten und nicht notwendigerweise darauf hindeuten, dass der Patient ins Krankenhaus zurückkehren muss, es sei denn, sie dauern länger als 5 Minuten an oder er hat mehrere wiederkehrende Anfälle oder Komplikationen wie Aspiration.“

Der zweite Datensatz stammt von [76] aus dem Jahr 2021 und umfasst Tweets, die sich auf die Deutsche Bahn beziehen. Tweets sind Kurznachrichten von der Social Media Plattform X² (ehemals Twitter), welche sich durch ihre Kürze, informelle Sprache sowie die Verwendung von Hashtags und Abkürzungen charakterisiert. Die Entitäten in diesem Datensatz umfassen unter anderem *organization-company*, *location-route*, *trigger*, *location-stop*, *date* und *time*. Der Datensatz besteht aus 2327 Trainings- und 411 Testdokumenten und ist damit der kleinste der untersuchten Datensätze. Die Implementierung der Modelle zu diesem Datensatz findet sich in dem entsprechenden Notebook³. Ein Beispiel für einen solchen Tweet lautet:

„**#S4 location-route** **#RegionDS organization-company** **#Teilausfall trigger** **#Mellendorf location-stop** (**23.03 date**) > **#Bennemühlen location-city** (**23.07 date**). Grund: **technische Störung event-cause** an der Strecke. Bitte nutzen Sie **#RB38 location-route** nach **Soltau location-city** über **Bennemühlen location-stop** Abfahrt: **23:08 time** Uhr vom Gleis **2 number**.“

¹<https://colab.research.google.com/drive/1FCphf1VW1wFxr-NjKS1BHwAeKXJfP8Ri?usp=sharing>

²<https://x.com/>

³https://colab.research.google.com/drive/1sY809nV9RbXbzRf4z0Hx3rWgjh20_SJ-?usp=sharing

Der dritte Datensatz von [77] aus dem Jahr 2024 enthält Dokumente der öffentlichen Verwaltung. Diese sind ähnlich formell wie Rechtsdokumente, jedoch zeichnen sie sich durch häufige Handlungsaufforderungen und längere Satzstrukturen aus. Die Entitäten in diesem Datensatz umfassen *Bedingung*, *Handlungsgrundlage*, *Aktion*, *Signalwort*, *Dokument*, *Frist*, *Datenfeld*, *Ergebnisempfänger*, *Mitwirkender* und *Hauptakteur*. Der Datensatz besteht aus 18.490 Trainings- und 3.264 Testdokumenten. Die Implementierung der Modelle zu diesem Datensatz findet sich in dem entsprechenden Notebook¹. Ein beispielhafter Satz aus diesem Datensatz lautet:

„Die Prüfung muss **Signalwort** vor einem Prüfungsausschuss **Hauptakteur** abgelegt **Aktion** werden, der bei der für die Finanzverwaltung zuständigen obersten Landesbehörde **Mitwirkender** zu bilden **Aktion** ist **Signalwort**.“

¹https://colab.research.google.com/drive/1oj_tKcN9XS1IaChW-9Za0BR1wAp8SVBq?usp=sharing

5. Umsetzung

Nachdem Kapitel 4 beleuchtet hat, wie die verschiedenen Ansätze miteinander verglichen werden, bietet dieses Kapitel einen Einblick in die Implementation des Lösungsansatzes. Dabei geht Kapitel 5.1 auf die technische Implementation der verschiedenen Ansätze ein, Kapitel 5 zeigt die erzielten Ergebnisse auf und Kapitel 5.3 diskutiert die Resultate.

5.1. Implementation der Ansätze

In der vorliegenden Arbeit wurden Modelle zur Informationsextraktion unter Verwendung von Python-Notebooks in der Google Colab Umgebung implementiert. Die Wahl von Google Colab erwies sich als besonders vorteilhaft, da die Plattform eine nahtlose Integration mit Google Drive ermöglicht, wodurch Datensätze effizient geladen, Modelle gespeichert und Ergebnisse zuverlässig verglichen werden konnten. Die direkte Verknüpfung der Notebooks mit dieser Arbeit gewährleistete zudem eine umfassende Dokumentation und Nachvollziehbarkeit des gesamten Implementierungsprozesses. Alle Modelle wurden auf der in Google Colab verfügbaren T4-GPU trainiert, um eine einheitliche Vergleichbarkeit sicherzustellen.

Zur Abdeckung unterschiedlicher Ansätze im Bereich des NLP wurden verschiedenen FOSS Modelle für NER sowie für das QA trainiert. Die Untersuchung konzentrierte sich dabei auf drei Modellgruppen: Encoder-Modelle, darunter BERT, RoBERTa und ALBERT, Decoder-Modelle wie GPT-2 und GPT-Neo, sowie Encoder-Decoder-Modelle, einschließlich T5, XLNet und BART (letzteres ausschließlich für das QA). Diese Modelle wurden auf einem im Abschnitt 4.2 beschriebenen, annotierten Datensatz trainiert, der gemäß den im selben Abschnitt dargestellten Schritten vorbereitet wurde.

Die Implementierung erfolgte unter Verwendung der Hugging Face Transformers-Bibliothek, einer Open-Source-Lösung, die auf PyTorch und TensorFlow basiert und herausragende Funktionalitäten für NLP-Anwendungen bietet. Diese Bibliothek stellt eine breite Auswahl vortrainierter Modelle sowie zahlreiche Werkzeuge zur Verfügung, die die Entwicklung und Feinabstimmung der Modelle erheblich erleichtern. Der gesamte Implementierungsprozess wurde ausführlich in den entsprechenden Notebooks dokumentiert.

Die Modelle wurden unter Nutzung der Trainer-Klasse von Hugging Face implementiert, wobei die Implementierung der offiziellen Dokumentationen¹ folgte.

Zudem wurde ein klassenbasierter Ansatz gewählt, um die Wiederverwendung für weitere Modelle und Datensätze so einfach wie möglich zu gestalten. Dabei initialisiert man die Trainerklasse mit dem Modell- sowie Tokenizernamen des entsprechenden Huggingface Modells:

```
# initialize the model name and the tokenizer name from huggingface
model_name = "bert-base-german-cased"
tokenizer_name = "bert-base-german-cased"

# initialize the ModelTrainer class
model = ModelTrainer(model_name, tokenizer_name, ids_to_labels, labels_to_ids)
```

Und definiert daraufhin die Hyperparameter, welche für das Training des Modells genutzt werden sollen. Dabei kann eine beliebige Anzahl an Hyperparametern gewählt werden, eine Übersicht aller Hyperparameter findet sich in der entsprechenden Dokumentation. Anschließend kann das Modell mit dem Trainingsdatensatz trainiert werden:

```
# define the hyperparameters used for training the model
training_args = TrainingArguments(
    num_train_epochs = 3,
    learning_rate = 2e-5,
    per_device_train_batch_size = 32,
    per_device_eval_batch_size = 32,
    gradient_accumulation_steps = 2,
    weight_decay = 0.01,
    output_dir = model_name,
    eval_strategy = "epoch",
    save_strategy = "epoch",
    load_best_model_at_end = True,
)

# train the model with the train dataset and the defined hyperparameters
model.train(train_dataset, training_args)
```

¹https://huggingface.co/docs/transformers/main_classes/trainer

Obwohl in dieser Arbeit spezifische Hyperparameter für jedes Modell festgelegt wurden, lag der Fokus auf einer allgemeinen Darstellung der Modelle, anstatt die optimierte Implementierung in den Vordergrund zu stellen. Es sei darauf hingewiesen, dass eine weitergehende Hyperparameter-Optimierung das Leistungspotenzial der Modelle steigern könnte. Zusätzlich zu den in den jeweiligen Modelldokumentationen enthaltenen Standard-Hyperparametern wurden spezifische Hyperparameter festgelegt, darunter:

- **Lernrate** (`learning_rate`): $2e-5$

Die Lernrate bestimmt wie groß die Schritte sind, die das Modell bei der Anpassung der Gewichte in Richtung des Optimierungsziels macht.

- **Batch-Größe für das Training** (`per_device_train_batch_size`): 32

Die Batch-Größe gibt an wie viele Trainingsbeispiele das Modell pro Schritt (Iteration) verarbeitet. Eine Batch-Größe von 32 bedeutet, dass das Modell 32 Datenpunkte gleichzeitig verarbeitet bevor es die Gewichte aktualisiert.

- **Batch-Größe für die Evaluation** (`per_device_eval_batch_size`): 32

Ähnlich wie bei der Trainings-Batch-Größe gibt dies die Anzahl der Datenpunkte an, die während der Auswertung pro Schritt verwendet werden. Auch hier wird mit 32 Datenpunkten gleichzeitig gearbeitet.

- **Anzahl der Trainingsepochen** (`num_train_epochs`): 3

Eine Epoche entspricht einem vollständigen Durchlauf durch den gesamten Trainingsdatensatz. Wenn `num_train_epochs` auf 3 gesetzt ist, wird der gesamte Datensatz drei Mal durchlaufen.

- **Gewichtsabfall** (`weight_decay`): 0,01

Der Gewichtsabfall ist eine Regularisierungstechnik, die verhindert, dass die Gewichte des Modells zu groß werden und damit Überanpassung (Overfitting) vermieden wird.

- **Schritte zur Gradientenakkumulation** (`gradient_accumulation_steps`): 2

Gradient Accumulation erlaubt es effektive größere Batches zu simulieren indem die Gradienten über mehrere Schritte akkumuliert werden bevor eine Gewichtsaktualisierung erfolgt. Ein Wert von 2 bedeutet, dass nach jedem zweiten Schritt die Gewichte aktualisiert werden.

In einigen Fällen wurde die Batch-Größe für das Training und die Evaluation reduziert, um den begrenzten RAM-Speicher der T4-GPU nicht zu überlasten. Die Implementation der FOSS Modelle ist in dem öffentlichen NER-Notebook¹ beziehungsweise QA-Notebook² einsehbar.

Die Evaluation der Modelle erfolgt über eine Modelpipeline-Klasse, welche mit dem zuvor trainierten Modell sowie Tokenzizer initialisiert wird. Anschließend kann man die Inferenz mit einem Testdatensatz durchführen:

```
# initialize the ModelPipe class for testing
pipe = ModelPipe(model_name, tokenizer_name)

# test the model on the test dataset
pipe.evaluate(test_dataset)
```

Um die Ergebnisse der Inferenz zu erhalten, kann man sich einerseits den Klassifikationsreport, welcher Aufschluss über die in Abschnitt 4.4 beschriebenen Metriken gibt, sowie eine Konfusionsmatrix ausgeben lassen:

```
# print the classification report, resulting in the different metrics
pipe.print_classification_report()

# print the corresponding confusion matrix to the test results
pipe.print_confusion_matrix()
```

5.2. Ergebnisse

In diesem Abschnitt werden die Ergebnisse der Experimente mit verschiedenen IE-Modellen vorgestellt und die Leistung verschiedener FOSS-Modelle aus Abschnitt 3.1 verglichen. In Abschnitt 5.2.1 wird die Leistung der entwickelten FOSS-Modelle erörtert, gemessen an dem in Abschnitt 4.4.1 beschriebenen F1-Score. Darüber hinaus gibt dieser Abschnitt einen Überblick über die Effizienz der Modelle, gemessen an Trainingszeit und Inferenzzeit. In Abschnitt 5.2.2 wird das in Bezug auf F1-Score beste FOSS-Modell aus

¹<https://colab.research.google.com/drive/1DLtCSozSNSRpTD7s7Mj5TWZ80GA-dPFB?usp=sharing>

²<https://colab.research.google.com/drive/1kt3-AKyxjwr9815pGGCFUWeeOpF5sVKT?usp=sharing>

Abschnitt 5.2.1 als Basis für den Vergleich mit proprietärer Software, beschrieben in Abschnitt 3.2, ausgewählt und die Leistung dieser bewertet. Die Ergebnisse werden dann mit dem FOSS-Modell verglichen, um Unterschiede in der Leistung und Effizienz zu ermitteln. Abschnitt 5.2.3 vergleicht anschließend die Kosten der kommerziellen Softwareanbieter und des ausgewählten FOSS-Modells. In diesem Abschnitt werden verschiedene Faktoren, wie Training, Hosting und Inferenz, untersucht, um einen umfassenden Überblick über die Kostenunterschiede zwischen den Modellen zu geben.

5.2.1. Ergebnisse der FOSS-Modelle

Die in Tabelle 1 vorliegenden Ergebnisse bieten eine detaillierte Gegenüberstellung von den in Abschnitt 5.1 vorgestellten Modellen im Bereich der NER und des QA anhand von drei Evaluationskriterien: F1-Score, Trainingszeit und Inferenzzeit. Diese Evaluationskriterien sind für eine fundierte Analyse der Modellleistung entscheidend, da sie nicht nur die Effektivität eines Modells in Bezug auf die Erkennungs- und Antwortgenauigkeit messen, sondern auch die zeitliche Effizienz, die bei praktischen Anwendungen eine zentrale Rolle spielt, berücksichtigen. Zudem liegen in den entsprechenden Notebooks noch weitere detaillierte Ergebnisse vor, unter anderem Präzision, Recall und F1-Score pro Entität. Zu jedem Modell kann für die Aufgabe des NER zudem eine Konfusionsmatrix in Anhang A gefunden werden.

BERT erzielt mit einem herausragenden F1-Score von 97,8 in der NER-Aufgabe das beste Ergebnis. Auch die Zeit zum Trainieren des Modells ist mit 14:29 Minuten die Geringste. Im Bereich der QA-Aufgabe benötigt BERT jedoch deutlich mehr Trainingszeit (1:01:00 Stunden) und erreicht dabei einen vergleichsweise geringeren F1-Score von 79,0%. Im Vergleich zu Modellen wie GPT-2 (83,0%) und BART (85,7%) schneidet BERT hier schwächer ab. RoBERTa, eine Weiterentwicklung von BERT, zeigt ebenfalls eine starke Leistung in der NER-Aufgabe mit einem F1-Score von 95,2%, jedoch fällt der Wert in der QA-Aufgabe auf 76,2% ab. Diese Ergebnisse sind insofern überraschend, als RoBERTa als komplexere Version von BERT normalerweise bessere Resultate liefert. Es scheint jedoch, dass die Optimierungen von RoBERTa nicht in allen Aufgabenbereichen zu Leistungssteigerungen führen obwohl sie zusätzliche Trainingsressourcen erfordern. ALBERT, eine kompaktere Variante von BERT, erzielt in beiden Aufgaben geringere F1-Scores (92,7% bei NER und

Model	Aufgabe	F1-Score	Trainingszeit	Inferenzzeit
BERT	NER	97,7	14:29	00:48
	QA	79,0	1:01:00	01:10
RoBERTa	NER	95,2	20:03	00:52
	QA	76,2	1:09:23	00:54
ALBERT	NER	92,7	26:20	00:56
	QA	76,9	1:12:14	00:55
GPT-2	NER	82,6	15:50	00:41
	QA	83,0	1:05:11	00:41
GPT-Neo	NER	84,8	30:39	01:56
	QA	17,1	1:33:03	01:03
T5	NER	66,9	25:11	01:02
	QA	69,3	3:31:17	02:31
XLNet	NER	96,9	39:00	01:27
	QA	73,6	2:12:10	01:26
BART	NER	-	-	-
	QA	85,7	4:47:02	02:00

Tabelle 1: FOSS Ergebnisse

76,9% bei QA). Die längeren Trainingszeiten könnten durch die geringere Anzahl an Parametern und die vereinfachte Architektur erklärt werden, was zu Leistungseinbußen führt, aber gleichzeitig die Modellgröße reduziert.

GPT-2 übertrifft viele Modelle in der QA-Aufgabe mit einem F1-Score von 83,0%, was auf seine Architektur zurückzuführen, die auf Textgenerierung optimiert ist. In der NER-Aufgabe schneidet GPT-2 jedoch mit einem F1-Score von 82,6% schwächer ab, was darauf hinweist, dass das Modell weniger auf die präzise Erkennung von Entitäten ausgelegt ist. Dies lässt sich durch die Verwendung der Decoder-Architektur erklären, wie in Abschnitt 2.3.2 erläutert. T5 erzielt in beiden Aufgaben schwächere Ergebnisse. Insbesondere in der NER-Aufgabe mit einem F1-Score von 66,9%. Diese Resultate deuten darauf hin, dass T5 trotz seiner flexiblen Text-zu-Text-Architektur für NER und QA weniger geeignet ist, da diese Aufgaben spezifischere Modellarchitekturen erfordern. Ein interessantes Ergebnis erzielt zudem GPT-Neo, welches mit 84,8% in der NER-Aufgabe besser als GPT-2 abschneidet, jedoch in der QA-Aufgabe lediglich 17,1% erreicht, was den niedrigsten Wert in dieser Kategorie unter allen Modellen markiert.

Ein bemerkenswertes Ergebnis zeigt XLNet, das mit einem F1-Score von 96,9% das zweithöchste Ergebnis in der NER-Aufgabe erreicht. Die langen Trainingszeiten deuten auf die hohe Komplexität der Encoder-Decoder-Architektur hin. Auch BART weist lange Trainingszeiten auf, was ebenfalls auf die Komplexität des Modells hinweist. Es erzielt jedoch den höchsten F1-Score von 85,7% in der QA-Aufgabe, was seine Stärken bei Aufgaben der Textzusammenfassung und -transformation unterstreicht.

5.2.2. Ergebnisse der proprietären Softwarelösungen

In Tabelle 2 werden die Ergebnisse der in Abschnitt 3.2 vorgestellten proprietären Software gegenübergestellt. Die Evaluationskriterien sind hierbei dieselben wie in Tabelle 1: F1-Score, Trainingszeit und Inferenzzeit. Es liegen zu jedem Modell ebenfalls detaillierte Ergebnisse in dem entsprechenden Notebook vor. Zudem können die Konfusionsmatrizen dieser Modelle in Anhang B gefunden werden. Das beste Modell, basierend auf dem F1-Score, aus Abschnitt 5.2.1 „BERT“ dient hierbei als Vergleich und befindet sich am Ende der Tabelle.

Produkt	F1-Score	Trainingszeit	Inferenzzeit
Google Vertex AI	89,4	3:49:13	12:08
Amazon Comprehend	97,9	40:28	19:35
Microsoft Azure	98,5	1:08:13	4:38:04
SAP	90,6 ¹	2:49:16	-
OpenAI	92,8	11:27	19:27
<i>BERT</i>	97,7	14:29	00:48*

Tabelle 2: Proprietäre Ergebnisse

In Bezug auf die F1-Scores liegt Microsoft Azure mit 98,5% an der Spitze, was auf eine äußerst hohe Präzision bei der Erkennung von Entitäten hinweist. Amazon Comprehend folgt mit einem F1-Score von 97,9% knapp dahinter und zeigt ebenfalls eine exzellente Performance. Beide Modelle übertreffen den F1-Score von BERT (97,7%) nur minimal, was die Konkurrenzfähigkeit von FOSS-Modellen wie BERT deutlich macht. Google und SAP fallen hingegen mit F1-Scores von 89,4% und 90,6% ab, was darauf hindeutet, dass diese Modelle in der Praxis weniger zuverlässig sein könnten. OpenAI erreicht mit einem

¹SAP hat diese Metrik selbst bereitgestellt, da aufgrund von Budgetbeschränkungen kein unabhängiger Test wie bei den anderen Modellen durchgeführt werden konnte.

F1-Score von 92,8% ein mittleres Leistungsniveau. Diese Wertung ist jedoch kritisch zu betrachten, da der OpenAI-Service für NER eigentlich nicht vorgesehen ist. OpenAI verhält sich eher wie ein Question-Answering-System und liefert nicht das spezifizierte Format zurück, was zu Fehlinterpretationen führen kann. Ein Beispiel dafür findet sich in dem entsprechenden Notebook. Diese Einschränkungen erklären möglicherweise auch die geringere Genauigkeit von OpenAI im Vergleich zu spezialisierten Systemen wie Azure und Comprehend.

Ein weiterer wesentlicher Punkt der Analyse betrifft die Effizienz in Bezug auf Trainings- und Inferenzzeiten. OpenAI erreicht mit einer Trainingszeit von nur 11:27 Minuten die kürzeste Trainingszeit. Amazon Comprehend zeichnet sich ebenfalls durch eine relativ kurze Trainingszeit von nur 40:28 Minuten aus, was es zu einer besonders attraktiven Option für Szenarien macht, die schnelles Modelltraining erfordern. Microsoft Azure benötigt mit 1:08:13 Stunden ebenfalls eine moderate Trainingszeit, während Google mit 3:49:13 und SAP mit 2:49:16 deutlich längere Trainingszeiten aufweisen. BERT zeigt sich hier mit einer Trainingszeit von nur 14:29 Minuten äußerst effizient, jedoch etwas langsamer als OpenAI. Besondere Unterschiede werden bei der Inferenzzeit sichtbar. BERT erweist sich mit einer Inferenzzeit von nur 48 Sekunden als herausragend schnell. Im Vergleich dazu sind die proprietären Modelle deutlich langsamer: Google benötigt 12:08 Minuten, OpenAI 19:27 Minuten und AWS 19:35 Minuten. Azure, obwohl es den höchsten F1-Score erzielt, weist mit einer Inferenzzeit von 4:38:04 Stunden die mit Abstand langsamste Verarbeitung auf. Hierbei ist jedoch zu beachten, dass diese signifikanten Unterschiede darin begründet sind, dass die proprietären Modelle über APIs und Serverstrukturen bereitgestellt werden, was die Inferenzzeiten verlängern kann. Im Gegensatz dazu läuft BERT lokal und effizienter, was die enormen Diskrepanzen bei der Inferenzzeit erklärt.

5.2.3. Kostenvergleich und -analyse

Um einen möglichst einheitlichen Kostenvergleich darzustellen wurde überprüft, wie viele Kosten abhängig von der Dokumentenmenge entstehen. Grundsätzlich entstehen für den Einsatz eines Modells drei Kostenarten: die Kosten für das Training des Modells, die Kosten für die Bereitstellung (Hosting) des Modells und Kosten für die Inferenz. In diesem Vergleich wird eine 30-Tage Periode simuliert, in welcher ein Modell am Anfang der Periode trainiert, über den gesamten Zeitraum bereitgestellt wird und anschließend eine variable

Menge an Dokumenten analysiert werden. Die genannten Kostenarten sind in der Regel abhängig von dem genutzten Datensatz. Der in dieser Arbeit genutzte Datensatz enthält im Schnitt 234,74 Zeichen pro Dokument. Die Kosten für das Training der Modelle ist in der Regel abhängig von der Trainingszeit, welche in Tabelle 2 zu finden ist.

Wie in Abschnitt 4.4.2 erwähnt, wurde BERT über AWS SageMaker bereitgestellt. AWS SageMaker umfasst die folgenden Kostenpunkte:

- **Modell-Artefakte:** Nach dem kostenfreien Training des BERT-Modells über Google Colab müssen die Modellartefakte in einen Amazon S3-Bucket hochgeladen werden, um sie für Amazon SageMaker verfügbar zu machen. Da das Modell etwa 384,5MB groß ist und für 30 Tage mit bis zu 100.000 Anfragen gespeichert werden muss, berechnen sich die Kosten wie folgt:

$$\text{S3 Kosten: } 0,3845GB \cdot \$0,0245/GB + 100.000 \cdot \$0,0000054/request = \$0.55 \quad (4)$$

- **ECR-Container-Image** Das Modell muss in einem ECR-Container-Image enthalten sein, um es für Amazon SageMaker verfügbar zu machen. Die Kosten hierfür berechnen sich wie folgt:

$$\text{ECR Kosten: } 0,3845GB \cdot \$0,1 = \$0.0384 \quad (5)$$

Wenn man diese Kosten zur Speicherung der Modellartefakte hinzurechnet, ergeben sich Trainingskosten von \$0,59.

- **Endpunkt-Hosting** Für das Hosting des Modells wurde eine „ml.c5.large“-Instanz gewählt. Die Kosten für das Hosten dieser Instanz für 24 Stunden am Tag für 30 Tage mit einem Modell und einem Endpunkt berechnen sich wie folgt:

$$\text{Hosting Kosten: } 24h \cdot 30d \cdot \$0,116 = \$83.52 \quad (6)$$

- **Inferenz** Da die Dokumente über UTF-8 kodiert sind, welches pro Zeichen 1-4 Bytes Speicher benötigt, sind die Dokumente im großzügigsten Fall $234,74 \cdot 4$ Bytes, also ungefähr 1KB groß. Hierbei fallen Kosten für Eingabedaten und Ausgabedaten an.

$$\text{Inferenz Kosten: } x \cdot 1KB \cdot \$0,016/GB + x \cdot 1KB \cdot \$0,016/GB \quad (7)$$

Eine umfassendere Kalkulation der Kosten findet sich in über den Kostenrechner¹ von AWS. Ein Ausschnitt davon befindet sich in Anhang C. Im folgenden werden nun die Kosten der kommerziellen Anbieter berechnet.

Google Vertex AI²

Google Vertex AI berechnet \$3,30 pro Stunde für das Training des Modells, \$0,05 pro Stunde für das Hosting des Modells und \$0,005 pro 1000-Zeichen-Textdatensatz. Da der genutzte Datensatz im Schnitt 234,74 Zeichen enthält, denotiert b_t mit einem Wert von 0,23474 einen Teil dieses Textdatensatzes.

$$\begin{aligned} \text{Training: } & \$3,30/h \cdot 3,8203h = \$12,607 \\ \text{Hosting: } & \$0,05/h \cdot 720h = \$36 \\ \text{Inferenz: } & \$0,005 \cdot x \cdot b_t = \$0,0011737 \cdot x \\ \text{Gesamt: } & \$12,607 + \$36 + \$0,0011737 \cdot x = \$48,607 + \$0,0011737x \end{aligned} \quad (8)$$

¹<https://calculator.aws/#/estimate?id=6bd5a6b81513e2d91233a44e1f0310e01edc0931>

²<https://cloud.google.com/vertex-ai/pricing>

Amazon Comprehend¹

Amazon Comprehend berechnet \$3 pro Stunde für das Training des Modells, \$0,05 pro Monat für das Hosting des Modells und \$0,005 pro 100-Zeichen-Textdatensatz. b_h denotiert mit einem Wert von 2,3474 einen Multiplikator dieses Textdatensatzes. Zudem arbeitet Amazon mit Inferenzeinheiten, welche pro Einheit einen Durchsatz von einem 100-Zeichen-Textdatensatz pro Sekunde bietet. Damit kann skaliert werden wie schnell das Modell Inferenzanfragen bearbeitet. Für diese Arbeit wurden 10 Inferenzeinheiten verwendet, weswegen der Preis für die Inferenz entsprechend angepasst werden muss.

$$\begin{aligned}\text{Training: } & \$3/h \cdot 0,6744h = \$2,023 \\ \text{Hosting: } & \$0,5 \\ \text{Inferenz: } & \$0,0005 \cdot x \cdot b_h \cdot 10 = \$0,011737 \cdot x \\ \text{Gesamt: } & \$2,023 + \$0,5 + \$0,011737 \cdot x = \$2,523 + \$0,011737x\end{aligned}\tag{9}$$

Azure Language Service²

Azure Language Service berechnet ebenfalls \$3 pro Stunde für das Training des Modells und \$0,05 pro Monat für das Hosting des Modells. Das Inferenzmodell ist ähnlich dem von Google Vertex AI mit \$0,005 pro 1000-Zeichen-Textdatensatz. b_t denotiert mit einem Wert von 0,23474 dementsprechend einen Teil dieses Textdatensatzes.

$$\begin{aligned}\text{Training: } & \$3/h \cdot 1,304h = \$3,911 \\ \text{Hosting: } & \$0,5 \\ \text{Inferenz: } & \$0,005 \cdot x \cdot b_t = \$0,0011737 \cdot x \\ \text{Gesamt: } & \$3,911 + \$0,5 + \$0,0011737 \cdot x = \$4,4108 + \$0,0011737x\end{aligned}\tag{10}$$

SAP Business Entity Recognition³

SAP berechnet für das Training keine Kosten, während für das Hosting 1,034€ pro Stunde anfallen. Die Inferenz wird hierbei in 10.000-Zeichen-Blöcken gemessen, wodurch b_X mit einem Wert von 0,023474 einen Teil dieses Textdatensatzes denotiert. Zudem senken sich die Kosten stufenweise, je nachdem wie viele 10.000-Zeichen-Blöcke für die Inferenz genutzt werden. Die Abstufung wurde hierbei b_X angepasst. Desweiteren gibt SAP die

¹<https://aws.amazon.com/de/comprehend/pricing/>

²<https://azure.microsoft.com/en-us/pricing/details/cognitive-services/language-service/>

³https://discovery-center.cloud.sap/serviceCatalog/business-entity-recognition?region=all&tab=service_plan

Preise in Euro an. Um einen einheitlichen Vergleich zu schaffen, wurden die Preise mit dem aktuellen Umrechnungskurs von EUR 1 = USD 1.1155 (26.09.2024) der europäischen Zentralbank¹ umgewandelt.

$$\begin{aligned}
&\text{Training: } \$0 \\
&\text{Hosting: } \$1,153/h \cdot 720h = \$830,46 \\
&\text{Inferenz: } \$11,71 \cdot x \cdot b_X = \$0,274945 \cdot x \text{ für } x \leq 2.130,017 \\
&\quad \$0,23 \cdot x \cdot b_X = \$0,005499 \cdot x \text{ für } x \leq 213.001,619 \\
&\quad \$0,15 \cdot x \cdot b_X = \$0,003404 \cdot x \text{ für } x \leq 639.004,856 \\
&\quad \$0,1 \cdot x \cdot b_X = \$0,002357 \cdot x \text{ für } x > 639.004,856
\end{aligned} \tag{11}$$

Open AI²

Open AI rechnet die Nutzung in der Anzahl der Tokens ab. Um eine möglichst genaue Bestimmung der Tokens für den genutzten Datensatz zu erhalten, wurde das von Open AI bereitgestellte Tool³ genutzt. Dies ergab eine durchschnittliche Menge von 60,27 Tokens pro Dokument. Für das Modell „babbage-02“ werden \$0.400/1M Tokens für das Training des Modells berechnet, während das Hosting kostenlos ist. Die Abrechnung der Inferenz erfolgt über Eingabe-Tokens und Ausgabe-Tokens, mit jeweils \$1.600/1M Tokens. Die Ausgabe Tokens betragen hierbei im Schnitt 17,82. Die Anzahl der Trainingsdokumente betrug 17.946.

$$\begin{aligned}
&\text{Training: } \$0,0000004 \cdot 17.946 \cdot 60,27 = \$0.43 \\
&\text{Hosting: } \$0 \\
&\text{Inferenz: } \$0,0000016 \cdot x \cdot 60,27 + \$0,0000016 \cdot x \cdot 17,82 = \$0,000124944 \cdot x \\
&\text{Gesamt: } \$0.43 + \$0 + \$0,000124944 \cdot x = \$0.43 + \$0,000124944x
\end{aligned} \tag{12}$$

In Abbildung 10 sind die oben aufgeführten Kostenfunktionen in einem Diagramm aufgeführt. Hierbei gibt die x-Achse eine Menge von Dokumenten für die Inferenz in einer Spanne von 0 bis 100.000 an. Die Trainings- und Hostingkosten wurden bei $x = 0$ addiert. In Anhang D sind bei Bedarf zudem weitere Diagramme für Dokumentenmengen von ≤ 10.000 und $\leq 1.000.000$ dargestellt.

¹<https://www.ecb.europa.eu/>

²<https://openai.com/api/pricing/>

³<https://platform.openai.com/tokenizer>

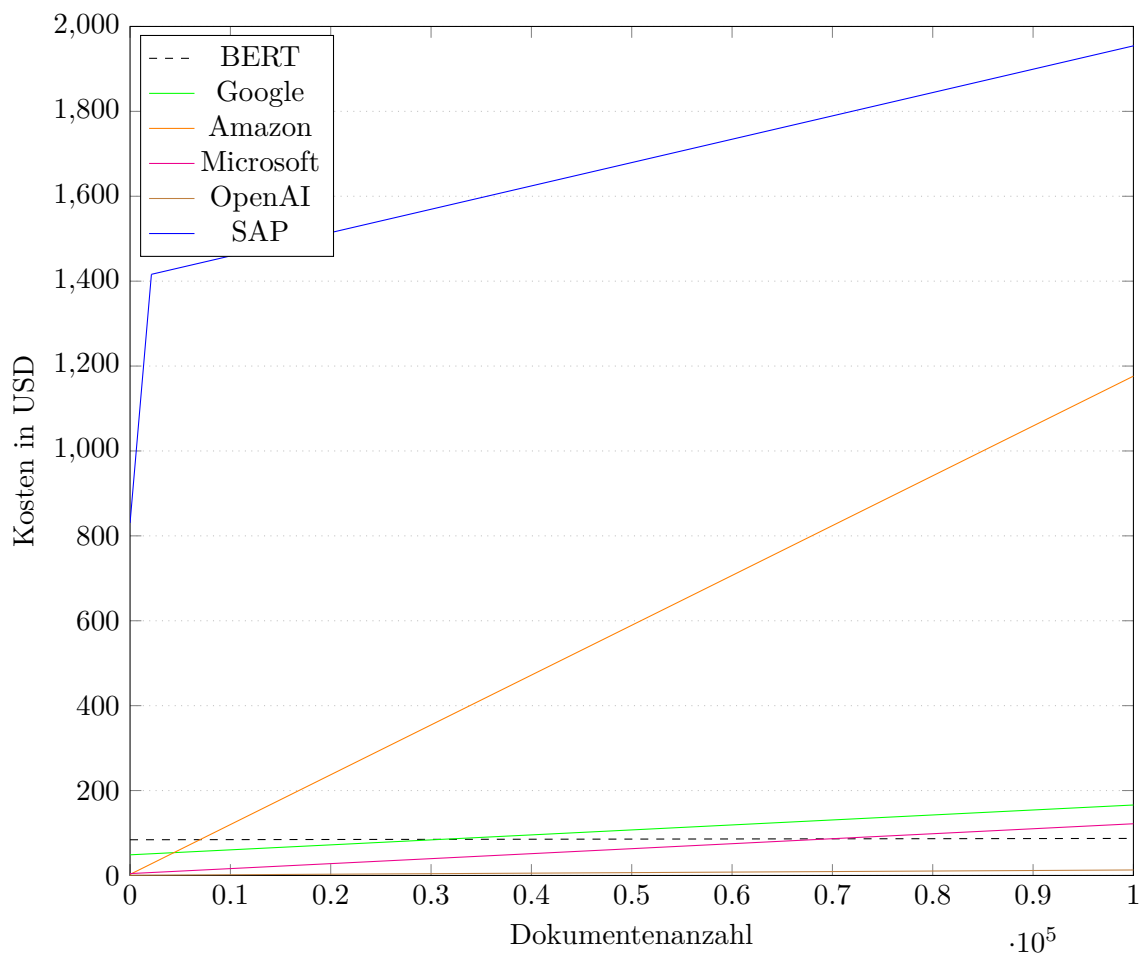


Abbildung 10: Kostenvergleich

Wie dem Graphen entnommen werden kann, ist das Produkt von SAP das kostenintensivste. Dies resultiert einerseits aus den vergleichsweise hohen Hostingkosten, andererseits aus den hohen Kosten für die ersten 50 Blöcke à 10.000 Zeichen. Amazon Comprehend weist trotz relativ niedriger Einstiegskosten ebenfalls hohe Inferenzkosten auf, was es nach 100.000 Dokumenten zum zweit-kostspieligsten Produkt macht. Diese Kostensteigerung ist auf die Wahl der Anzahl der genutzten Inference Unit (IU) zurückzuführen, welche die Inferenzkosten um den Faktor 10 erhöht hat. Würde das Produkt nur mit einer IU betrieben, entsprächen die Inferenzkosten denen von Google Vertex AI und Microsoft Azure. Google Vertex AI weist leicht höhere Hostingkosten auf, was es im Vergleich zu Microsoft Azure teurer macht, unabhängig von der verarbeiteten Dokumentenmenge. OpenAI hingegen bietet die niedrigsten Vorlauf- sowie Inferenzkosten pro Dokument, wodurch es das kostengünstigste kommerzielle Produkt darstellt. Dabei ist auch die Wahl des Modells von OpenAI „babbage-02“ zu beachten, welches das günstigste Modell darstellt. Im Ver-

gleich dazu zeigt das Open-Source-Modell BERT signifikant geringere Inferenzkosten im Vergleich zu allen anderen Anbietern. Allerdings führen die relativ hohen Hostingkosten der „ml.c5.large“-Instanz zu vergleichsweise hohen Vorlaufkosten, wodurch die Implementierung bei einer Dokumentenmenge von 100.000 teurer ist als die von OpenAI. Es ist jedoch zu beachten, dass die Wahl der Instanz den größten Teil der Kosten ausmacht und durch die Wahl einer anderen Instanz die Gesamtkosten erheblich gesenkt oder erhöht werden können. Die Kosten für Produkte, die das Hosting stündlich abrechnen (Google Vertex AI, SAP Business Entity Recognition und BERT über AWS SageMaker), können weiter reduziert werden, indem das Hosting beispielsweise nachts oder an Wochenenden deaktiviert wird.

5.2.4. Ergebnisse im domänenübergreifenden Kontext

Tabelle 3 vergleicht die Leistung der FOSS im Bereich der NER über mehrere Domänen hinweg. Hierbei wird der gewichtete F1-Score wie in den Tabellen 1 und 2 dargestellt. Weitere Metriken, wie beispielsweise Trainingszeit, Inferenzzeit oder Konfusionsmatrizen, können in den entsprechenden Notebooks zu den Domänen, beschrieben in Abschnitt 4.5, gefunden werden.

Model	Recht	Medizin	Mobilität	Verwaltung
BERT	97,7	90,2	78,7	80,5
RoBERTa	95,2	89,9	83,6	79,1
ALBERT	92,7	88,4	81,3	74,8
GPT-2	82,6	82,9	62,4	76,4
GPT-Neo	84,8	85,9	70,1	74,2
T5	66,9	47,3	43,0	64,4
XLNet	96,9	90,8	86,8	78,4

Tabelle 3: FOSS Ergebnisse Domänenübergreifend

Insbesondere die Encoder-Modelle BERT und RoBERTa zeigen durchweg hohe F1-Scores. Diese Ergebnisse sind bemerkenswert da sie nicht nur im Rechtsbereich, sondern auch in anderen Fachdomänen konsistent gut abschneiden. Dies deutet darauf hin, dass diese Modelle eine bemerkenswerte Fähigkeit zur domänenübergreifenden Generalisierung besitzen. Darüber hinaus hat auch ALBERT, welches eine komprimierte Version von BERT darstellt, relativ gute Ergebnisse erzielt. Trotz der Reduzierung der Modellgröße und der

Anzahl der Parameter bleibt die Leistungsfähigkeit in NER-Aufgaben stark. Dies lässt darauf schließen, dass die grundlegenden architektonischen Vorteile von BERT auch in seiner komprimierten Form erhalten bleiben. Ein interessanter Punkt ist zudem die Architektur von XLNet, welche ebenfalls als starke Alternative zu BERT und RoBERTa angesehen werden kann. XLNet erreicht F1-Scores, die mit den besten Ergebnissen von BERT und RoBERTa vergleichbar sind. In zwei Domänen erzielt XLNet sogar die besten Resultate, was seine Vielseitigkeit und Leistungsfähigkeit unterstreicht.

Während Encoder-Modelle in der Regel hervorragende Ergebnisse liefern, zeigen die Decoder-Modelle, zu denen GPT-2, GPT-Neo und T5 zählen, eine deutlich schwächere Leistung. Dabei ist die Leistung von T5 besonders auffällig: Dieses Modell erzielt in allen getesteten Domänen die schwächsten Ergebnisse. Mögliche Gründe hierfür könnten in den unterschiedlichen Trainingsstrategien und der Art der Datenverarbeitung liegen, die für T5 verwendet wurden. Die F1-Scores der GPT-Modelle fallen sowohl im Rechtsbereich als auch in anderen Domänen in der Regel hinter denen von BERT und RoBERTa zurück.

5.3. Diskussion der Resultate

Die Ergebnisse der verschiedenen Modelltypen für das NER und das QA unterscheiden sich signifikant. Hinsichtlich der F1-Scores zeigt das BERT-Modell mit einem Wert von 97,8% die beste Performance unter den getesteten Modellen. Selbst das schwächste Encoder-Modell, ALBERT, erzielt mit 92,7% noch immer vergleichsweise gute Ergebnisse. Im Gegensatz dazu schneiden die Decoder-Modelle deutlich schlechter ab. Hier erzielt GPT-Neo als bestes Decoder-Modell lediglich 84,8%, während T5 mit 66,9% das schlechteste Ergebnis aufweist. Das Encoder-Decoder-Modell XLNet zeigt mit 96,9% im NER ebenfalls gute Resultate.

In Bezug auf die Trainingszeiten lassen sich zwischen Encoder- und Decoder-Modellen keine signifikanten Unterschiede feststellen; diese hängen eher von der Modellgröße als von der Architektur ab. Encoder-Decoder-Modelle benötigen allerdings eine längere Trainingszeit, wohingegen die Encoder-Modelle die schnellsten Inferenzzeiten aufweisen. Dies könnte auf die Architektur zurückzuführen sein, da Encoder-Modelle auf eine effizientere Verarbeitung optimiert sind. Bei den Inferenz- und Trainingszeiten kommerzieller Anbieter existieren sehr große Unterschiede. Während OpenAI das Training nach 11:27 Minuten beendet hat, wodurch es schneller als alle FOSS Modelle war, benötigte Google

Vertex AI 3:49:13 Stunden. Dennoch erzielen alle Modelle in dem F1-Score Vergleich eine relativ hohe Performance mit einem Mindestwert von 89,4%. Interessanterweise übertreffen die kommerziellen Modelle von AWS und Azure sogar das BERT-Modell wobei Azure mit 98,5% und AWS mit 98,9% leicht besser abschneiden. Die Unterschiede zwischen NER und QA sind ebenfalls deutlich zu erkennen. Im Durchschnitt zeigen die Modelle im Hinblick auf den F1-Score für NER eine deutlich bessere Performance als bei QA. Zudem sind die Trainings- und Inferenzzeiten für QA signifikant länger. Während für NER die Encoder-Modelle besonders gut abschneiden, performen beim QA die Decoder- und Encoder-Decoder-Modelle besser. Dieser Unterschied in der Performance könnte auf die unterschiedlichen Anforderungen der beiden Aufgaben zurückzuführen sein: NER erfordert eine präzise Identifizierung von Entitäten, was Encoder-Modelle aufgrund ihrer Architektur effizienter leisten können. QA hingegen erfordert eine komplexere Generierung von Antworten, bei der die Decoder- und Encoder-Decoder-Modelle besser abschneiden.

Zudem lässt sich feststellen, dass diese Ergebnisse unabhängig von der Domäne sind. Diese Ergebnisse legen nahe, dass die architektonische Gestaltung der Decoder-Modelle möglicherweise weniger geeignet ist, um die spezifischen Anforderungen von NER-Aufgaben zu erfüllen. XLNet erreicht zudem sehr gute Ergebnisse in allen Domänen, was weiter verdeutlicht wie wichtig der Encoder Teil in der Modellarchitektur ist, wenn man Modelle für die IE nutzen möchte.

Es ist jedoch zu beachten, dass mögliche Fehlerquellen in der Implementierung zu den beobachteten Unterschieden beitragen. Beispielsweise könnten bestimmte Modelle nicht optimal auf die Aufgabe abgestimmt oder die Hyperparameter falsch gewählt worden sein. Ein weiterer Faktor könnten Unterschiede in der Trainingsstrategie sein, insbesondere im Hinblick auf die spezifische Nutzung der Testversionen von proprietären Modellen, die zu Performance-Einbußen führen können. So ist nicht auszuschließen, dass Unternehmen nur eine abgeschwächte Version ihrer Modelle für einen kostenlosen Testzeitraum zur Verfügung stellen. Ein Beispiel hierfür ist der Natural Language Understanding Free Plan von IBM: „The Natural Language Understanding Free plan limits the size and performance of your custom model. To test a custom model to its full extent, use it with the Natural Language Understanding Standard plan.“¹.

Des Weiteren gibt es sehr große Kostenunterschiede, doch wie bereits in Abschnitt 5.2.3

¹<https://cloud.ibm.com/docs/natural-language-understanding?topic=natural-language-understanding-entities-and-relations>

diskutiert, können verschiedene Maßnahmen ergriffen werden, um die Kosten anzupassen, darunter die Wahl der IU, die Instanz über AWS SageMaker, der Hostingzeitraum oder das gewählte Modell von OpenAI. Zudem könnten FOSS Modelle über eigene Bereitstellungen statt über kommerzielle Anbieter wie AWS genutzt werden, um Kosten zu senken. In dieser Arbeit wurden zudem keine Zusatzfaktoren betrachtet wie Verfügbarkeit, Benutzerfreundlichkeit, Skalierbarkeit, Datenverschlüsselung, Integrationsmöglichkeiten oder andere Dienste.

Der gewählte Datensatz hat zudem einige Schwierigkeiten geborgen, darunter beispielsweise die Anonymisierung von Namen oder Unternehmen, welche lediglich durch einen Buchstaben angegeben werden. Viele Modelle hatten Probleme diese Entitäten zu Erkennen, darunter „Landschaft“, „Straße“, „Stadt“, „Land“, „Person“, „Anwalt“, „Unternehmen“ und „Marke“. Obwohl der verwendete Datensatz eher anspruchsvoll ist, wurden dennoch gute Ergebnisse erzielt. Zu beachten ist jedoch, dass der verwendete Datensatz öffentlich verfügbar ist, was die Möglichkeit nicht ausschließt, dass einige der verwendeten Modelle diesen Datensatz oder Teile davon bereits in ihrem Pre-Training gesehen haben könnten, was die Ergebnisse verfälschen würde.

6. Schlussbetrachtung

Um die Resultate der Arbeit abzuschließen fasst Abschnitt 6.1 die Ergebnisse dieser Arbeit zusammen, Kapitel 6.2 gibt eine kritische Bewertung über die Einhaltung der Ziele der Arbeit ab und Kapitel 6.3 gibt einen Ausblick auf mögliche, zukünftige Forschungsrichtungen auf diesem Gebiet.

6.1. Zusammenfassung der Ergebnisse

Um die verschiedenen Ansätze zur Verarbeitung und Analyse zu vergleichen, wurde ein umfangreicher deutscher Datensatz aus dem Bereich der Rechtsliteratur ausgewählt. Dieser Datensatz wurde zunächst einer gründlichen Vorverarbeitung unterzogen, um die Daten für die anschließende Modellierung und Analyse aufzubereiten. Die Vorverarbeitung umfasste Schritte wie Datenbereinigung, Normalisierung und das Entfernen irrelevanter Informationen, um sicherzustellen, dass die Modelle auf qualitativ hochwertigen und strukturierten Daten trainiert werden. Im nächsten Schritt wurden verschiedene FOSS Modelle implementiert. Diese Modelle wurden zunächst auf einem Teil des vorverarbeiteten Datensatzes trainiert, um ihre Lernfähigkeiten und Genauigkeit zu verbessern. Der restliche Teil des Datensatzes diente als Testdatensatz zur Evaluierung der Modelle. Zwei unterschiedliche Ansätze standen im Fokus: Zum einen wurde die NER eingesetzt, um benannte Entitäten zu identifizieren. Zum anderen wurde der QA-Ansatz angewendet, bei welchem die Modelle auf die Beantwortung spezifischer Fragen zu rechtlichen Texten trainiert wurden. Die Leistungsfähigkeit der implementierten Modelle wurde anhand verschiedener Metriken evaluiert, darunter Präzision, Recall und insbesondere der F1-Score, um eine ausgewogene Betrachtung zwischen Genauigkeit und Vollständigkeit der Vorhersagen zu gewährleisten. Das Modell, welches im Hinblick auf den F1-Score die besten Ergebnisse erzielte, wurde anschließend mit proprietären Softwarelösungen kommerzieller Anbieter verglichen. Dabei wurde derselbe Datensatz verwendet, um eine faire Vergleichbarkeit sicherzustellen. Auch hier kamen dieselben Metriken zum Einsatz, um die Modelle objektiv gegenüberzustellen. Zusätzlich zu diesem Vergleich wurde ein Kostenvergleich zwischen den Open-Source-Lösungen und den kommerziellen Produkten durchgeführt. Ziel war es nicht nur die technische Leistungsfähigkeit sondern auch die ökonomische Effizienz der Ansätze zu bewerten. Um die Generalisierbarkeit der FOSS-Modelle über den Bereich der Rechtsliteratur hinaus zu untersuchen, wurden diese Modelle in einem weiteren Schritt mit drei

zusätzlichen Datensätzen aus anderen Domänen getestet: dem medizinischen Bereich, der Mobilitätsbranche und der öffentlichen Verwaltung. Diese Domänen wurden ausgewählt, um die Robustheit und Anpassungsfähigkeit der Modelle in verschiedenen Anwendungsfeldern zu bewerten. Auch hier kamen die zuvor verwendeten Metriken zum Einsatz, um eine einheitliche und transparente Bewertung der Modellleistungen in den unterschiedlichen Bereichen zu ermöglichen.

Die Encoder-Architektur ist für Aufgaben wie die Informationsextraktion, insbesondere für NER, oft besser geeignet als die Decoder-Architektur. Ein Hauptgrund dafür ist, dass Encoder-Modelle wie BERT die gesamte Eingabesequenz gleichzeitig betrachten können. Dadurch sind sie in der Lage, Kontextinformationen von beiden Seiten eines Tokens zu berücksichtigen, was entscheidend für das Verständnis der Bedeutung von Wörtern im jeweiligen Kontext ist. Diese bidirektionale Betrachtung hilft enorm bei der Erkennung von Entitäten. Darüber hinaus sind Encoder-Modelle oft speziell für Aufgaben der Informationsverarbeitung entwickelt. Sie verwenden Techniken wie die Maskierung von Tokens, um sicherzustellen, dass das Modell lernt, die Bedeutung von Wörtern im Kontext zu verstehen. Decoder-Modelle hingegen sind auf die Generierung von Text ausgelegt, was sie weniger geeignet macht für Aufgaben, die sich auf die Extraktion von Informationen konzentrieren. Zusätzlich zeigt diese Arbeit, dass es ausreichend ist, die Encoder-Architektur in Modellen wie den Encoder-Decoder-Architekturen, wie beispielsweise XLNet, zu integrieren. Diese Modelle erzielten Ergebnisse, die in vielerlei Hinsicht mit denen von BERT vergleichbar sind, obwohl sie eine längere Trainingszeit erfordern und oft eine größere Modellgröße aufweisen.

Die Modelle haben zudem im Bereich des NER in der Regel bessere Ergebnisse erzielt als im QA. Dies könnte daran liegen, dass die Zielsetzung von NER klar definiert ist: Es geht darum, spezifische Entitäten wie Namen, Orte und Organisationen aus dem Text zu identifizieren und zu klassifizieren. Diese Fokussierung ermöglicht es den Modellen gezielte Lernstrategien zu entwickeln. Im Gegensatz dazu erfordert QA ein komplexeres Verständnis, da Modelle relevante Informationen finden und diese in Antworten umsetzen müssen, was die Komplexität erhöht. Die Fragen im QA-Bereich können unterschiedlich interpretiert werden, was zu mehr Unsicherheit in den Ergebnissen führen könnte. Eine weitere Beobachtung während dieser Arbeit ist die Herausforderung geeignete Modelle für die deutsche Sprache zu finden. Viele der verfügbaren Modelle sind überwiegend auf englische Texte spezialisiert. Dies stellt eine signifikante Hürde dar, da die Anwendbarkeit der Model-

le auf andere Sprachen, insbesondere auf Deutsch, eingeschränkt ist. Multilinguale Modelle stellen in diesem Kontext eine potenzielle Lösung dar, da sie darauf ausgelegt sind mehrere Sprachen zu unterstützen. Dennoch bleibt die Verfügbarkeit hochqualitativer, speziell für die deutsche Sprache optimierter Modelle begrenzt.

Die Untersuchung zeigt auch, dass das selbstständige Training und die Implementierung von Modellen kostengünstiger sein können. Je nach Implementierung können hierbei erhebliche Einsparungen erzielt werden. Auch wenn im Vergleich zu den proprietären Lösungen leicht schlechtere Ergebnisse erzielt wurden, könnte ein gezieltes Finetuning der Modellparameter die Ergebnisse möglicherweise verbessern. Allerdings ist der Prozess der Optimierung sowohl der Modellparameter als auch des Datensatzes zeitaufwändig, da kontinuierliche Anpassungen und Tests erforderlich sind, um die bestmögliche Leistung zu erreichen. Für die proprietären No-Code-Anwendungen zeigt die Untersuchung, dass die Ergebnisse sehr zufriedenstellend sind. Dennoch erfordert die Anbindung in der Regel den Einsatz von Code, was die Zugänglichkeit für Nutzer ohne Programmierkenntnisse einschränken kann. In der heutigen Zeit, in der der Bedarf an benutzerfreundlichen Lösungen wächst, stellt dies eine bedeutende Herausforderung dar. OpenAI wäre in diesem Zusammenhang die günstigste kommerzielle Alternative doch der Service wird bedauerlicherweise ab dem 28. Oktober 2024 nicht mehr angeboten¹.

6.2. Kritische Bewertung der Arbeit

Im Rahmen dieser Arbeit wurden die gesetzten Ziele weitgehend erreicht und die zentralen Fragestellungen konnten zufriedenstellend beantwortet werden. Ein primäres Ziel war die Untersuchung verschiedener Ansätze zur Umwandlung unstrukturierter Daten in strukturierte Informationen, wobei sowohl technische Verfahren als auch Softwarelösungen betrachtet wurden. Diese Zielsetzung wurde erfolgreich umgesetzt indem die Ansätze der NER und des QA detailliert analysiert und evaluiert wurden. Dabei wurden erfolgreich verschiedene Architekturstile, wie Encoder-, Decoder- oder Encoder-Decoder-Modelle untersucht. Ebenso erfolgte der Vergleich zwischen Open-Source- und proprietären Softwarelösungen auf Grundlage klar definierter Metriken.

Durch die Performance-Analyse der Modelle, die vor allem durch den F1-Score gemessen wurde, konnte ein umfassender Einblick in die Stärken und Schwächen der verschiedenen

¹<https://platform.openai.com/docs/guides/fine-tuning>

Ansätze gewonnen werden. Dies ermöglichte es das leistungsfähigste Open-Source-Modell zu identifizieren und dieses in einem anschließenden Kostenvergleich mit proprietären Lösungen zu bewerten. Auch dieser Aspekt der Arbeit wurde erfolgreich umgesetzt und die Untersuchung zeigte auf, dass Open-Source-Lösungen in bestimmten Szenarien eine ökonomisch sinnvolle Alternative zu kommerziellen Softwareprodukten darstellen können.

Darüber hinaus wurde das Ziel der wiederverwendbare Ansätze für künftige Vergleiche zu schaffen, teilweise erreicht. Die gewählte klassenbasierte Struktur zur Modellierung ermöglicht eine hohe Flexibilität bei der Entwicklung und Anwendung verschiedener Modelle. Durch diese modulare Architektur können unterschiedliche Modelle mit geringem Aufwand implementiert und evaluiert werden, was sowohl die Effizienz als auch die Reproduzierbarkeit der Analyseprozesse erheblich steigert. Ein weiterer Vorteil dieser Struktur liegt in ihrer Anpassungsfähigkeit an verschiedene Datensätze. Diese Flexibilität in der Datenvorbereitung und Modellierung ermöglicht eine umfassende Erprobung der Modelle in verschiedenen Kontexten und unter verschiedenen Bedingungen, was die Validität und Robustheit der Ergebnisse weiter erhöht. Auf diese Weise können die entwickelten Methoden und Modelle langfristig als wertvolle Grundlage für weitere Forschung und Anwendungen in diesem Bereich dienen.

6.3. Ausblick auf zukünftige Forschungsmöglichkeiten

Um einen nachhaltigen Mehrwert zu gewährleisten, wurde der gesamte Code zur Erstellung und Evaluierung der Modelle in einer klassenbasierten Struktur implementiert. Dieses Design ermöglicht es die Datensätze, die verwendeten Transformer-Modelle sowie die Hyperparameter flexibel und unkompliziert auszutauschen. Dadurch wird sichergestellt, dass auch zukünftige, neu entwickelte Modelle nahtlos in den bestehenden Vergleich integriert oder auf neuen Datensätzen getestet werden können. Diese Modularität ist entscheidend, um die langfristige Anpassungsfähigkeit und Erweiterbarkeit des Systems sicherzustellen.

Der einzige manuelle Eingriff, den die Nutzer vornehmen müssen, besteht darin die Trainings- und Testdaten in das vorgegebene Format zu überführen. Das erwartete Format entspricht der JSON-Syntax, die beispielsweise in den Frameworks spaCy und Hugging-face Anwendung findet. Ein Beispiel für die Struktur dieser JSON-Dateien ist nachfolgend dargestellt:

```
[
  {
    'tokens': ['Dies', 'ist', 'ein', 'Objekt', '.'],
    'ner_tags': ['O', 'O', 'O', 'B-Entity', 'O']
  },
  {
    'tokens': ['Ein', 'langes', 'Objekt', '.'],
    'ner_tags': ['O', 'B-Entity', 'I-Entity', 'O']
  }
]
```

In diesem Format kennzeichnet das Tag *O* Tokens, die keine Relevanz für die Entitätserkennung besitzen. Das Präfix *B-* markiert den Beginn einer benannten Entität während das Präfix *I-* auf Tokens hinweist, die zu einer mehrteiligen Entität gehören.

Um die Nutzerfreundlichkeit des Systems weiter zu verbessern könnte darüber hinaus eine grafische Benutzeroberfläche entwickelt werden. Diese würde es ermöglichen die verschiedenen Trainings- und Testkomponenten noch intuitiver zu bedienen und nahtlos mit der bestehenden Klassenstruktur zu interagieren. Ein Prototyp für eine solche Oberfläche ist im Anhang E dieser Arbeit zu finden. Durch diese Maßnahmen wird nicht nur die Zukunftsfähigkeit des Systems gesichert sondern auch der Entwicklungsaufwand für die Implementierung neuer Modelle und deren Tests erheblich reduziert.

Ein zentraler Punkt zur Verbesserung der aktuellen Modelle wäre ein optimiertes und umfangreiches Hyperparameter-Tuning. Derzeit besteht die Möglichkeit, dass die Genauigkeit der implementierten Modelle durch ein nicht ausreichend durchgeführtes Tuning beeinträchtigt ist. Ein intensiveres und zeitaufwendigeres Hyperparameter-Tuning würde es ermöglichen die tatsächliche Leistungsfähigkeit der Modelle genauer zu erfassen. Ein weiterer Vorschlag für zukünftige Forschung auf diesem Gebiet zielt auf den Einsatz eines Datensatzes ab, der nicht öffentlich zugänglich ist. Wenn die Modelle auf einem solchen,

bislang unbekannten Datensatz trainiert und getestet werden, kann ausgeschlossen werden, dass diese bereits vorab auf diesem Datensatz trainiert wurden. Dies würde die Validität und Aussagekraft der Evaluationsergebnisse erheblich steigern, da so eine authentischere Bewertung der Modelleistung möglich wäre. Zusätzlich könnten neue methodische Ansätze dazu beitragen die Effizienz der IE zu verbessern. Eine interessante Möglichkeit bestünde darin die IE zweistufig zu gestalten: Zunächst könnten grobkörnige Klassen extrahiert werden, um dann innerhalb dieser Klassen feinere Subklassen zu bestimmen. Dieser Ansatz könnte die Granularität und Präzision der Extraktion erhöhen. Zudem könnten moderne Modelle wie LayoutLM, die neben dem Textinhalt auch die räumliche Struktur von Dokumenten berücksichtigen, innovative und vielversprechende Resultate liefern. Diese Modelle wären besonders für Aufgaben geeignet, bei denen die räumliche Anordnung von Informationen eine wichtige Rolle spielt, wie etwa bei der Verarbeitung von Rechnungen oder Formularen.

Ein weiterer vielversprechender Forschungsansatz wäre die Entwicklung eines eigenen neuronalen Netzwerks oder eines speziell angepassten Transformer-Modells. Diese eigens entwickelten Modelle könnten dann mit etablierten Standardmodellen wie BERT verglichen werden. Durch diesen Vergleich könnte untersucht werden, ob maßgeschneiderte Modelle, die gezielt für den jeweiligen Anwendungsfall entwickelt wurden, eine bessere Leistung erbringen als allgemeinere, vortrainierte Modelle wie BERT. Dies wäre insbesondere in spezialisierten Anwendungsbereichen von großem Interesse, da hier möglicherweise spezifische Merkmale oder Anforderungen existieren, die durch angepasste Modelle besser abgedeckt werden könnten. Abschließend wäre es von großem Nutzen, eine Studie zur Praxistauglichkeit solcher Informationsextraktionsmodelle durchzuführen. In dieser Studie könnte untersucht werden, ob eine bestimmte Genauigkeit, beispielsweise 97,7%, für Unternehmen ausreicht, um diese Modelle produktiv einzusetzen. Zusätzlich könnte die Einführung eines Confidence Scores, der die Zuverlässigkeit der Modellergebnisse anzeigt, sowie der Einsatz eines „Human-in-the-loop“-Ansatzes erprobt werden. Letzterer würde es ermöglichen menschliche Überprüfungen in den Prozess zu integrieren, um so die Akzeptanz und den praktischen Nutzen der Modelle in realen Anwendungsszenarien weiter zu steigern. Diese Kombination aus automatisierter Informationsextraktion und menschlicher Kontrolle könnte dazu beitragen die Effizienz und Genauigkeit der Modelle in der Praxis zu maximieren.

Literatur

- [1] J. Gantz and D. Reinsel, “The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east,” *IDC iView: IDC Analyze the future*, vol. 2007, no. 2012, pp. 1–16, 2012.
- [2] K. Adnan and R. Akbar, “An analytical study of information extraction from unstructured and multidimensional big data,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–38, 2019.
- [3] M. H. A. Abdullah, N. Aziz, S. J. Abdulkadir, H. S. A. Alhussian, and N. Talpur, “Systematic literature review of information extraction from textual data: recent methods, applications, trends, and challenges,” *IEEE Access*, vol. 11, pp. 10 535–10 562, 2023.
- [4] L. Görgen, L. Griesch, and K. Sandkuhl, “Comparison of ai-based document classification platforms,” in *International Conference on Business Informatics Research*. Springer, 2024, pp. 68–84.
- [5] J. Cowie and W. Lehnert, “Information extraction,” *Communications of the ACM*, vol. 39, no. 1, pp. 80–91, 1996.
- [6] Y. Chen, C. Zhou, T. Li, H. Wu, X. Zhao, K. Ye, and J. Liao, “Named entity recognition from chinese adverse drug event reports with lexical feature based bilstm-crf and tri-training,” *Journal of biomedical informatics*, vol. 96, p. 103252, 2019.
- [7] H. Ji, *Information Extraction*. Boston, MA: Springer US, 2009, pp. 1476–1481. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_204
- [8] C. Suman, S. M. Reddy, S. Saha, and P. Bhattacharyya, “Why pay more? a simple and efficient named entity recognition system for tweets,” *Expert Systems with Applications*, vol. 167, p. 114101, 2021.
- [9] D. Feng and H. Chen, “A small samples training framework for deep learning-based automatic information extraction: Case study of construction accident news reports analysis,” *Advanced Engineering Informatics*, vol. 47, p. 101256, 2021.

- [10] H. Cunningham *et al.*, “Information extraction, automatic,” *Encyclopedia of language and linguistics*, vol. 3, no. 8, p. 10, 2005.
- [11] M. Mannai, W. B. A. Karâa, and H. H. B. Ghezala, “Information extraction approaches: A survey,” in *Information and Communication Technology: Proceedings of ICICT 2016*. Springer, 2018, pp. 289–297.
- [12] J. Turmo, A. Ageno, and N. Catala, “Adaptive information extraction,” *ACM Computing Surveys (CSUR)*, vol. 38, no. 2, pp. 4–es, 2006.
- [13] P. S. Jacobs, *Text-based intelligent systems: Current research and practice in information extraction and retrieval*. Psychology Press, 2014.
- [14] D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson, “Fastus: A finite-state processor for information extraction from real-world text.” in *IJCAI*, 01 1993, pp. 1172–1178.
- [15] R. Grishman, “Information extraction: Techniques and challenges,” in *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology: International Summer School, SCIE-97 Frascati, Italy, July 14–18, 1997*. Springer, 1997, pp. 10–27.
- [16] L. Eikvil, “Information extraction from world wide web-a survey,” Technical Report 945, Norweigan Computing Center, Tech. Rep., 1999.
- [17] S. Singh, “Natural language processing for information extraction,” *arXiv preprint arXiv:1807.02383*, 2018.
- [18] O. Etzioni, A. Fader, J. Christensen, S. Soderland *et al.*, “Open information extraction: The second generation,” in *Twenty-Second International Joint Conference on Artificial Intelligence*. Citeseer, 2011.
- [19] J. Fox, “Formalizing knowledge and expertise: where have we been and where are we going?” *The Knowledge Engineering Review*, vol. 26, no. 1, pp. 5–10, 2011.
- [20] R. Studer, V. R. Benjamins, and D. Fensel, “Knowledge engineering: Principles and methods,” *Data & knowledge engineering*, vol. 25, no. 1-2, pp. 161–197, 1998.

- [21] K. Kaiser and S. Miksch, “Information extraction. a survey.” E188 - Institute of Software Technology and Interactive Systems; Vienna University of Technology, Tech. Rep., 2005. [Online]. Available: http://publik.tuwien.ac.at/files/pub-inf_2999.pdf
- [22] M.-F. Moens, *Information extraction: algorithms and prospects in a retrieval context*. Springer, 2006, vol. 1.
- [23] J. Han, J. Pei, and H. Tong, *Data mining: concepts and techniques*. Morgan kaufmann, 2022.
- [24] B. Bofin, “text classification workflow,” <https://cloudsek.com/hierarchical-attention-on-neural-networks-beyond-the-traditional-approaches-for-text-classification/>, accessed: 2024-05-27.
- [25] M. Frye and R. H. Schmitt, “Structured data preparation pipeline for machine learning-applications in pro-duction,” *17th IMEKO TC*, vol. 10, pp. 241–246, 2020.
- [26] J. Brownlee, “Data preparation for machine learning,” 2022.
- [27] E. Haddi, X. Liu, and Y. Shi, “The role of text pre-processing in sentiment analysis,” *Procedia computer science*, vol. 17, pp. 26–32, 2013.
- [28] L. Hickman, S. Thapa, L. Tay, M. Cao, and P. Srinivasan, “Text preprocessing for text mining in organizational research: Review and recommendations,” *Organizational Research Methods*, vol. 25, no. 1, pp. 114–146, 2022.
- [29] S. Wang, W. Zhou, and C. Jiang, “A survey of word embeddings based on deep learning,” *Computing*, vol. 102, no. 3, pp. 717–740, 2020.
- [30] I. H. Sarker, “Machine learning: Algorithms, real-world applications and research directions,” *SN Computer Science*, vol. 2, no. 3, pp. 1–21, 2021.
- [31] R. Ramya, K. Venugopal, S. Iyengar, and L. Patnaik, “Feature extraction and duplicate detection for text mining: A survey,” *Global Journal of Computer Science and Technology*, 2017.
- [32] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, “A survey on text classification: From shallow to deep learning,” *arXiv preprint arXiv:2008.00364*, 2020.

- [33] P. C. Sen, M. Hajra, and M. Ghosh, “Supervised classification algorithms in machine learning: A survey and review,” in *Emerging technology in modelling and graphics*. Springer, 2020, pp. 99–111.
- [34] N. Chinchor, “Overview of muc-7,” in *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*, 1998.
- [35] E. Nismi Mol and M. Santosh Kumar, “Review on knowledge extraction from text and scope in agriculture domain,” *Artificial Intelligence Review*, vol. 56, no. 5, pp. 4403–4445, 2023.
- [36] W. Xiang and B. Wang, “A survey of event extraction from text,” *IEEE Access*, vol. 7, pp. 173 111–173 137, 2019.
- [37] A. M. N. Allam and M. H. Haggag, “The question answering systems: A survey,” *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, vol. 2, no. 3, 2012.
- [38] K. Hussaina, M. N. M. Salleha, S. Talpura, and N. Talpura, “Big data and machine learning in construction: a review,” *International Journal of Soft Computing and Metaheuristics*, pp. 2–4, 2018.
- [39] C. Napoli, E. Tramontana, and G. Verga, “Extracting location names from unstructured italian texts using grammar rules and mapreduce,” in *Information and Software Technologies: 22nd International Conference, ICIST 2016, Druskininkai, Lithuania, October 13-15, 2016, Proceedings 22*. Springer, 2016, pp. 593–601.
- [40] K. Feldman, L. Faust, X. Wu, C. Huang, and N. V. Chawla, “Beyond volume: The impact of complex healthcare data on the machine learning pipeline,” in *Towards Integrative Machine Learning and Knowledge Extraction: BIRS Workshop, Banff, AB, Canada, July 24-26, 2015, Revised Selected Papers*. Springer, 2017, pp. 150–169.
- [41] P. Li and K. Mao, “Knowledge-oriented convolutional neural network for causal relation extraction from natural language texts,” *Expert Systems with Applications*, vol. 115, pp. 512–523, 2019.

- [42] U. Roll, R. A. Correia, and O. Berger-Tal, “Using machine learning to disentangle homonyms in large text corpora,” *Conservation Biology*, vol. 32, no. 3, pp. 716–724, 2018.
- [43] A. Khosla, R. Hamid, C.-J. Lin, and N. Sundaresan, “Large-scale video summarization using web-image priors,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2698–2705.
- [44] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, “Data preprocessing for supervised leaning,” *International journal of computer science*, vol. 1, no. 2, pp. 111–117, 2006.
- [45] A. Goyal, V. Gupta, and M. Kumar, “Recent named entity recognition and classification techniques: a systematic review,” *Computer Science Review*, vol. 29, pp. 21–43, 2018.
- [46] U. Naseem, I. Razzak, S. K. Khan, and M. Prasad, “A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models,” *Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 5, pp. 1–35, 2021.
- [47] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, “Ammus: A survey of transformer-based pretrained models in natural language processing,” *arXiv preprint arXiv:2108.05542*, 2021.
- [48] L. Tunstall, L. Von Werra, and T. Wolf, *Natural language processing with transformers*. Ö'Reilly Media, Inc.", 2022.
- [49] D. Dessì, D. R. Recupero, and H. Sack, “An assessment of deep learning models and word embeddings for toxicity detection within online textual comments,” *Electronics*, vol. 10, no. 7, p. 779, 2021.
- [50] E. M. Dharma, F. L. Gaol, H. Warnars, and B. Soewito, “The accuracy comparison among word2vec, glove, and fasttext towards convolution neural network (cnn) text classification,” *Journal of Theoretical and Applied Information Technology*, vol. 100, no. 2, p. 31, 2022.

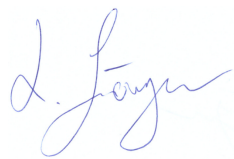
- [51] N. A. Smith, “Contextual word representations: A contextual introduction,” *arXiv preprint arXiv:1902.06006*, 2019.
- [52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [53] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” *arXiv preprint arXiv:1801.06146*, 2018.
- [54] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [55] A. Wang, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” *arXiv preprint arXiv:1804.07461*, 2018.
- [56] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.
- [57] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” *arXiv preprint arXiv:1909.11942*, 2019.
- [58] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, “Improving language understanding by generative pre-training,” 2018.
- [59] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang *et al.*, “Gpt-neox-20b: An open-source autoregressive language model,” *arXiv preprint arXiv:2204.06745*, 2022.
- [60] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.
- [61] M. Lewis, “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” *arXiv preprint arXiv:1910.13461*, 2019.

- [62] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” *Advances in neural information processing systems*, vol. 32, 2019.
- [63] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, “Review of deep learning: concepts, cnn architectures, challenges, applications, future directions,” *Journal of big Data*, vol. 8, pp. 1–74, 2021.
- [64] M. H. M. Noor and A. O. Ige, “A survey on deep learning and state-of-the-arts applications,” *arXiv preprint arXiv:2403.17561*, 2024.
- [65] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent trends in deep learning based natural language processing,” *ieee Computational intelligence magazine*, vol. 13, no. 3, pp. 55–75, 2018.
- [66] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, “Explainable ai: A brief survey on history, research areas, approaches and challenges,” in *Natural language processing and Chinese computing: 8th cCF international conference, NLPCC 2019, dunhuang, China, October 9–14, 2019, proceedings, part II 8*. Springer, 2019, pp. 563–574.
- [67] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *Journal of machine learning research*, vol. 12, pp. 2493–2537, 2011.
- [68] E. Leitner, G. Rehm, and J. Moreno-Schneider, “Fine-grained named entity recognition in legal documents,” in *International conference on semantic systems*. Springer, 2019, pp. 272–287.
- [69] I. Horvatić Bilić and S. Husinec, “Fachspezifische texte im rechtswesen am beispiel der textsorte vertrag,” in *IDT 2022-XVII. Internationale Tagung der Deutschlehrerinnen und Deutschlehrer, Beč, Austrija, 15. 8.–20. 8. 2022.*, 2023, pp. 113–128.
- [70] L. Vikraman, W. B. Croft, and B. O’Connor, “Exploring diversification in non-factoid question answering,” in *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, 2018, pp. 223–226.

- [71] M. Hossin and M. N. Sulaiman, “A review on evaluation metrics for data classification evaluations,” *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.
- [72] M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: an overview,” *arXiv preprint arXiv:2008.05756*, 2020.
- [73] J. Mohajon, “confusion matrix,” <https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826>, accessed: 2023-05-27.
- [74] “sagemaker workflow,” <https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html#deploy-model-steps-byom>, accessed: 2024-05-27.
- [75] J. Frei and F. Kramer, “German medical named entity recognition model and data set creation using machine translation and word alignment: Algorithm development and validation,” *JMIR Formative Research*, vol. 7, p. e39077, 2023.
- [76] L. Hennig, P. T. Truong, and A. Gabryszak, “Mobie: A german dataset for named entity recognition, entity linking and relation extraction in the mobility domain,” *arXiv preprint arXiv:2108.06955*, 2021.
- [77] L. Feddoul, S. T. Bachinger, C. Lachenmaier, S. Apel, P. Karg, N. Klewer, D. Forshayt, R. Erd, and M. Mauch, “Gerps-ner: A dataset for named entity recognition to support public service process creation in germany,” 2024.

Eidesstattliche Versicherung

Ich versichere eidesstattlich durch eigenhändige Unterschrift, dass ich die Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht und ist in gleicher oder ähnlicher Weise noch nicht als Studienleistung zur Anerkennung oder Bewertung vorgelegt worden. Ich weiß, dass bei Abgabe einer falschen Versicherung die Prüfung als nicht bestanden zu gelten hat.



Rostock, den 22. Oktober 2024

Anhang

A. Konfusionsmatrizen FOSS

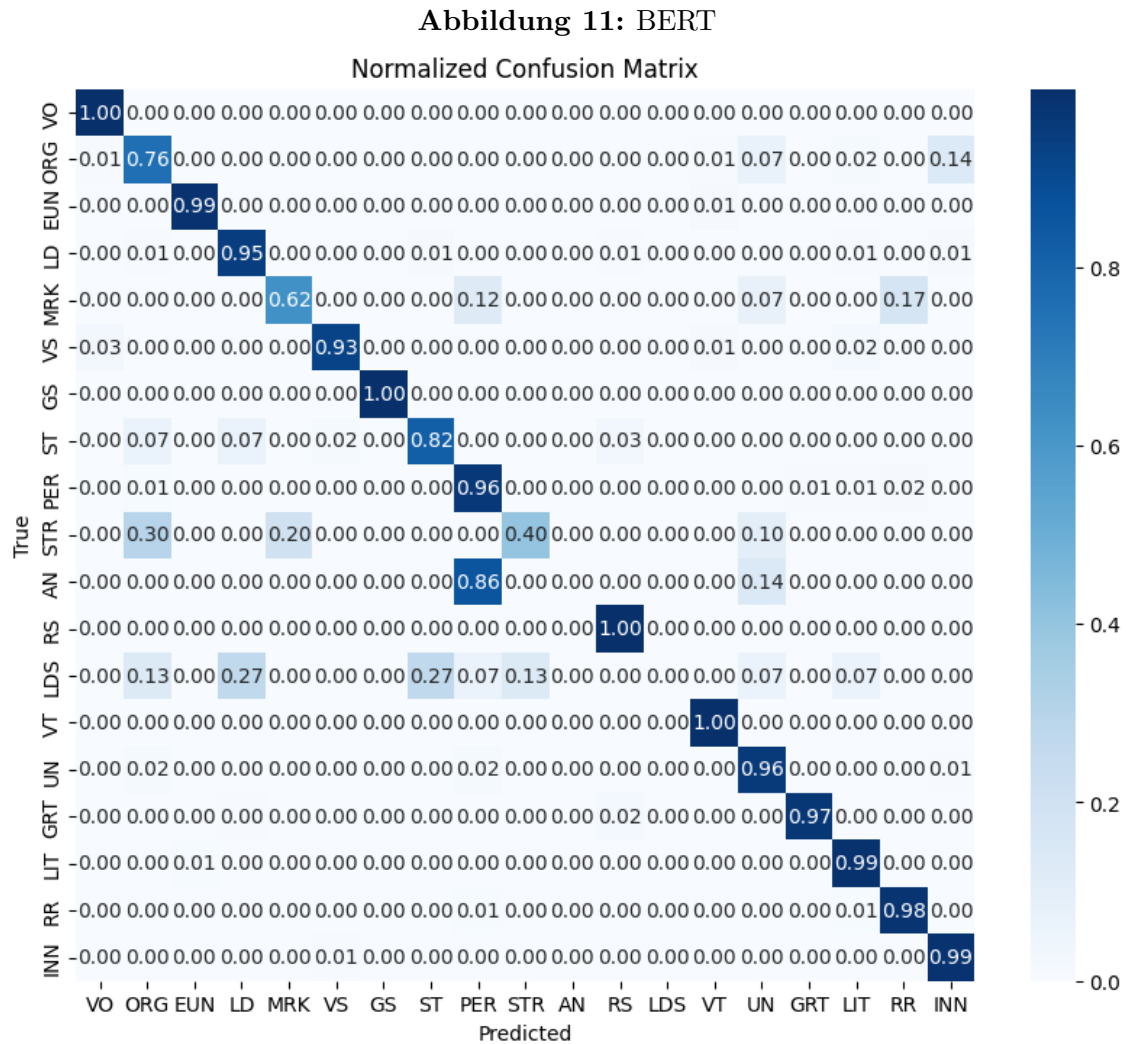


Abbildung 12: RoBERTa

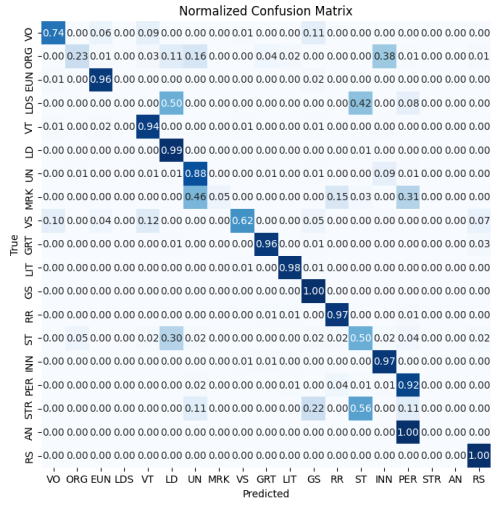


Abbildung 15: GPT-Neo

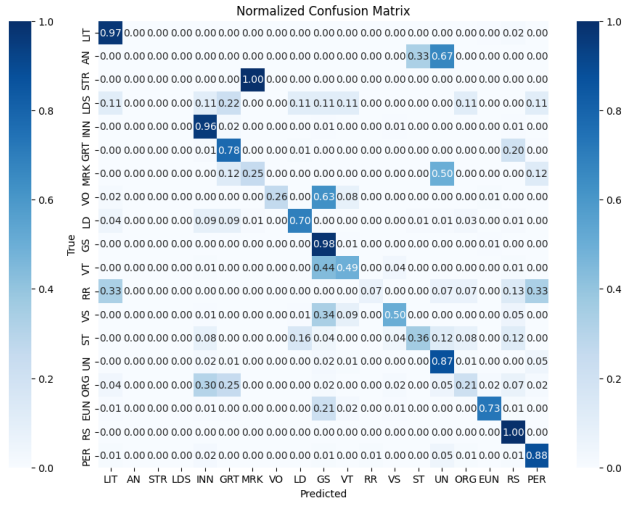


Abbildung 13: Albert

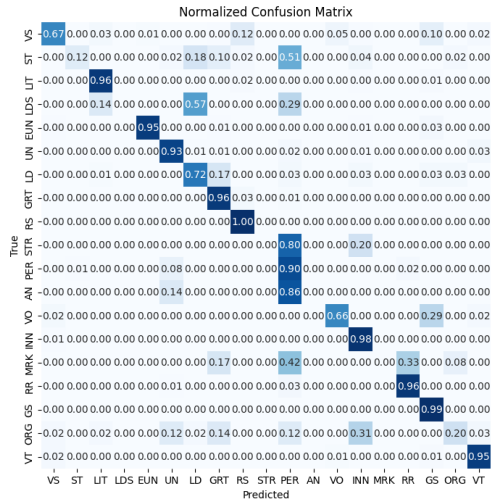


Abbildung 16: T5

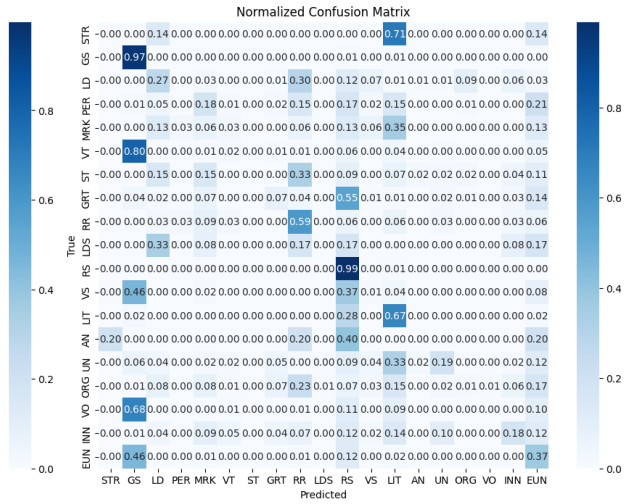


Abbildung 14: GPT-2

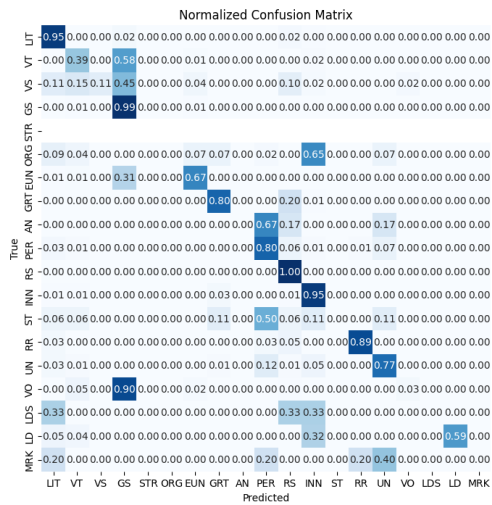
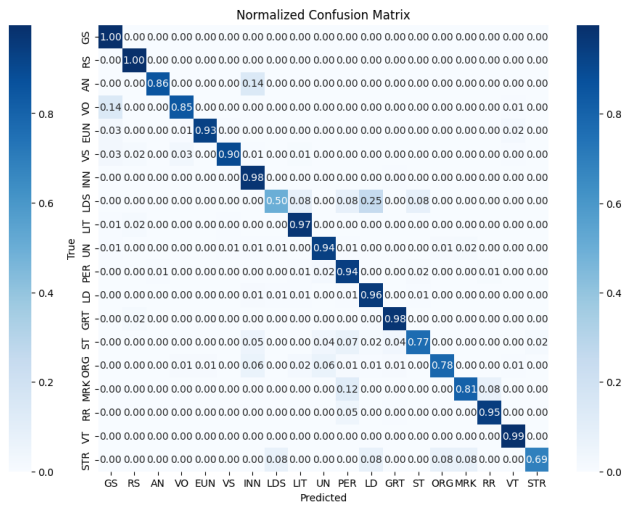


Abbildung 17: XLNet



B. Konfusionsmatrizen Kommerziell

Abbildung 18: Vertex AI

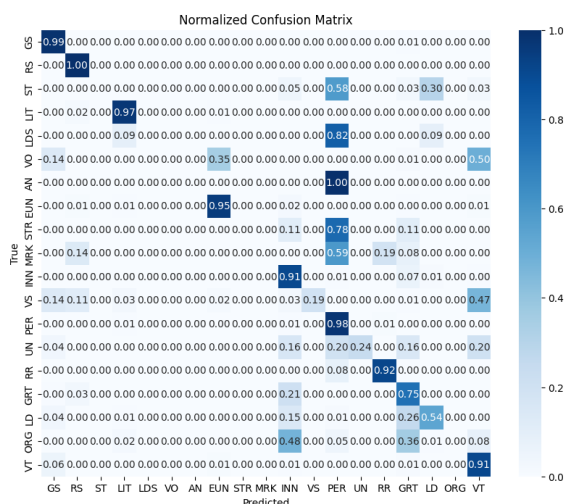


Abbildung 20: Azure

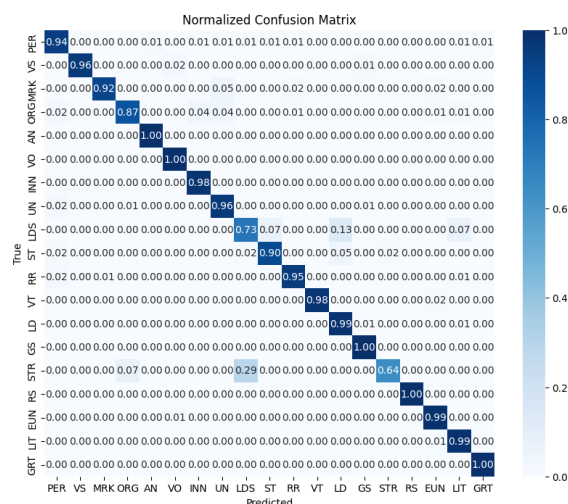


Abbildung 19: AWS

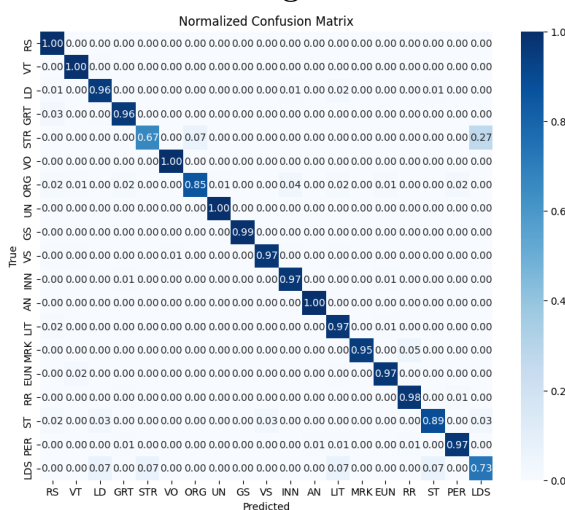
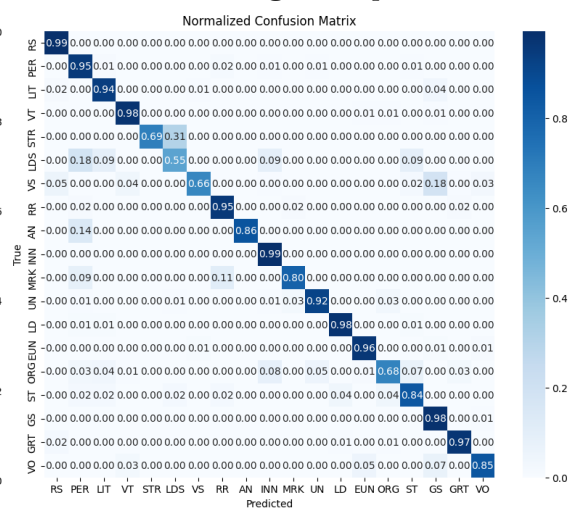



Abbildung 21: OpenAI



C. AWS SageMaker

27.09.24, 12:13

My Estimate – AWS Pricing Calculator



An Ihren AWS-Mitarbeiter wenden: [Vertrieb kontaktieren](#)

Exportdatum: 27.9.2024

Sprache: Deutsch

URL der Schätzung: <https://calculator.aws/#/estimate?id=6bd5a6b81513e2d91233a44e1f0310e01edc0931>

Übersicht über die Schätzung

Vorabkosten	Monatliche Kosten	Gesamtkosten 12 months
0,00 USD	84,11 USD	1.009,32 USD
		Einschließlich Vorabkosten

Detaillierte Schätzung

Name	Gruppe	Region	Vorabkosten	Monatliche Kosten
Amazon Simple Storage Service (S3)	Keine Gruppe angewendet	EU (Frankfurt)	0,00 USD	0,55 USD
Status: - Beschreibung: Config summary: S3-Standardspeicher (0.3845 GB pro Monat), PUT-, COPY-, POST-, LIST-Anfragen an S3 Standard (100000), GET, SELECT und alle anderen Anforderungen von S3 Standard ()				
Amazon Elastic Container Registry	Keine Gruppe angewendet	EU (Frankfurt)	0,00 USD	0,04 USD
Status: - Beschreibung: Config summary: Menge der gespeicherten Daten (0.3845 GB pro Monat)				
Amazon SageMaker	Keine Gruppe angewendet	EU (Frankfurt)	0,00 USD	83,52 USD
Status: - Beschreibung: Config summary: Anzahl der Modelle pro Endpunkt (1), Speicher (Allzweck-SSD (gp2)), Instance-Name (ml.c5.large), Anzahl der bereitgestellten Modelle (1), Anzahl der Instances pro Endpunkt (1), Endpunktstunde(n) pro Tag (24), Endpunkttag(e) pro Monat (30), Verarbeitete eingehende Daten (100 MB), Verarbeitete ausgehende Daten (100 MB)				

<https://calculator.aws/#/estimate>

1/2

Abbildung 22: AWS Kostenkalkulation

D. Kostenvergleiche

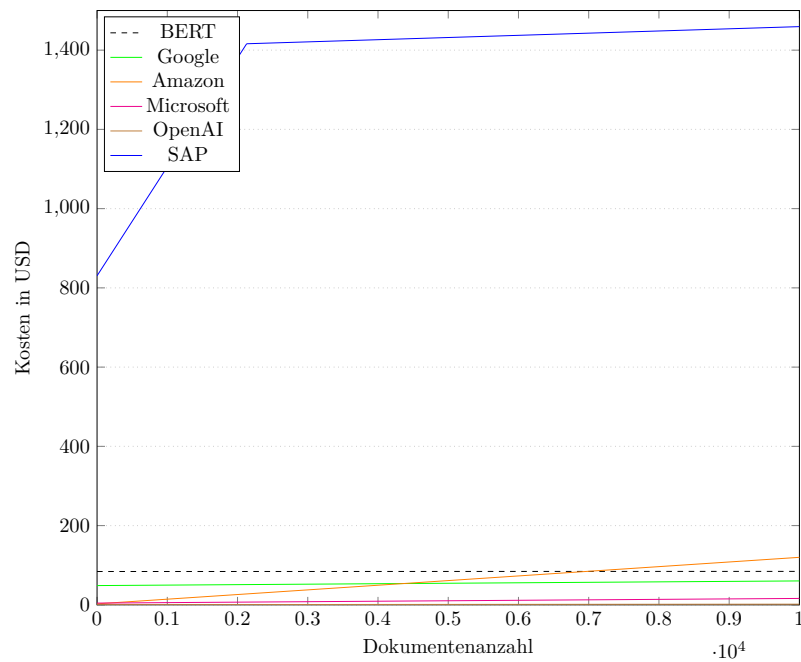


Abbildung 23: Kostenvergleich - Dokumentenanzahl ≤ 10.000

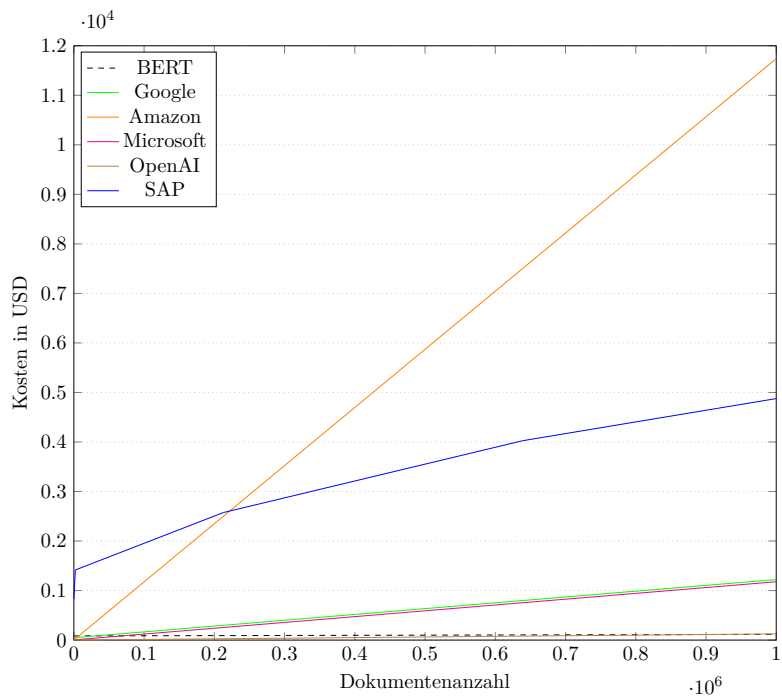


Abbildung 24: Kostenvergleich - Dokumentenanzahl $\leq 1.000.000$

E. Prototyp Benutzeroberfläche

Modelltrainer

Aufgabenbereich

Wählen Sie einen Aufgabenbereich aus, den Sie trainieren möchten.

Aufgabenbereich

Named Entity Recognition

Wählen Sie die Sprache, in der Sie trainieren möchten.

Sprache

Deutsch

Modellauswahl

Wählen Sie ein oder mehrere Modelle aus, die Sie trainieren möchten.

Modelle

BERT, RoBERTa, ALBERT, XLNet

Oder importieren Sie ein Modell über eine Huggingface URL.

URL

Trainingsdaten

Laden Sie Ihre Trainingsdaten hoch. Die Daten sollten Im JSON-Format vorliegen.

Datei auswählen

Format der Trainingsdaten:

{

"text": ['Test', 'Text', 'for',
'training', 'the', 'model'],

"tokens": ['0', 'B-LOC', 'I-LOC',
'0', '0', '0'],

},

Modellparameter

Wählen Sie die Parameter für das Training aus.

Batch Size

32

Epochs

3

Learning Rate

0.002

Gewichtsabfall

0.001

Gradientenakkumulation

2

Abbildung 25: Prototyp Benutzeroberfläche

90

Akronyme

ACE Automatic Content Extraction. 16, 17

ALBERT A Lite BERT. 31, 32, 53, 57, 66, 67

AWS Amazon Web Services. 35, 49, 61, 62, 66, 68, 69, 88

BART Bidirectional and Auto-Regressive Transformer. 34, 53, 57, 59

BERT Bidirectional Encoder Representations from Transformers. 29–35, 53, 57, 59–61, 66–68, 71, 75

CO Coreference resolution. 15

DARPA Defense Advanced Research Projects Agency. 14

ECR Elastic Container Registry. 49, 61

FN False Negative. 46–48

FOSS Free Open-Source Software. 6, 8, 9, 21, 29, 39, 40, 45, 49, 50, 53, 56–59, 66, 67, 69, 70

FP False Positive. 46–48

GPT Generative Pre-trained Transformer. 32–34, 36, 53, 57, 58, 67

IE Informationsextraktion. 5–7, 9–14, 17–19, 29, 37, 38, 45, 46, 48, 49, 56, 68, 75

IR Information Retrieval. 7, 10, 17

IU Inference Unit. 65, 69

KE Knowledge Engineering. 11, 12

ML Machine Learning. 10–14

MUC Message Understanding Conferences. 14, 16

NER Named Entity Recognition. 8, 15, 17, 23, 39, 40, 43, 47, 50, 53, 56–60, 66–68, 70–72

NIST National Institute of Standards and Technology. 16

NLP Natural Language Processing. 11, 14, 17, 23, 29–31, 33–36, 53

OCR Optical Character Recognition. 13

QA Question Answering. 8, 17, 34, 35, 39, 40, 43, 44, 53, 56–59, 67, 68, 70–72

RoBERTa Robustly optimized BERT approach. 30, 31, 53, 57, 66, 67

ST Scenario template production. 15

T5 Text-to-Text Transfer Transformer. 33, 34, 53, 58, 67

TE Template element construction. 15

TN True Negative. 46, 47

TP True Positive. 46–48

TR Template relation construction. 15