Universität Stuttgart, Institut für Maschinelle Sprachverarbeitung
Pfaffenwaldring 5b
70569 Stuttgart

Fraunhofer Institut für Arbeitswirtschaft und Organisation
Nobelstraße 12
70569 Stuttgart

Masterarbeit

Automatisierte Generierung von Trainingsdaten für die Informationsextraktion aus deutschen Geschäftsdokumenten auf Basis von Sprachmodellen

Jannik Burkhardt

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Thang Vu
Betreuer/in:	Matthias Engelbach, M.Sc. Dr.-Ing. Maximilien Kintz
Beginn am:	7. Januar 2023
Beendet am:	7. Juli 2023

Kurzfassung

Generative KI hat seit der Veröffentlichung von ChatGPT im Dezember 2022 enorme Popularität erlangt. Ihr Potenzial ist immens und schon heute wird diese neue Technik in viele Produkte und Anwendungen integriert. In dieser Arbeit wird untersucht, welchen Einfluss automatisiert annotierte Trainingsdaten und von ChatGPT generierte Trainingsdaten auf das Finetuning von Sprachmodellen haben, wenn nur wenige handannotierte Daten vorhanden sind. Die mit den Methoden verbundenen Vorteile und Hindernisse werden am Beispiel der Relation Extraction aus deutschen Geschäftsdokumenten in Erfahrung gebracht. Es wird gezeigt, dass die Daten von ChatGPT von Fehlern bereinigt werden müssen, diese Daten dann jedoch die Leistung des Sprachmodells signifikant verbessern gegenüber einem Sprachmodell, das nur auf wenigen handannotierten Daten basiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

AI Artificial Intelligence.

API Application Programming Interface.

BERT Bidirectional Encoder Representations from Transformers.

CNN Convolutional Neural Network.

DARPA Defense Advanced Research Projects Agency.

DL Deep Learning.

DNN Deep Neural Networks.

DSGVO Datenschutz-Grundverordnung.

GAI General Artificial Intelligence.

GB Gigabyte.

GPT Generative Pretrained Transformer.

HTML Hypertext Markup Language.

IE Information Extraction.

KI Künstliche Intelligenz.

LLM Large Language Model.

MB Megabyte.

ML Machine Learning.

n. a. nicht angegeben.

NER Named Entity Recognition.

NLP Natural Language Processing.

NRE Neural Relation Extraction.

OCR Optical Character Recognition.

PLM Pretrained Language Model.

POS Part-Of-Speech.

RC Relation Classification.

RE Relation Extraction.

RI Relation Identification.

TB Terabyte.

XAI Explainable Artificial Intelligence.

1. Einführung & Motivation

Machine Learning (ML), Deep Learning (DL) und generative Künstliche Intelligenz sind die Schlagwörter, die seit längerem die Medien und die Fachliteratur dominieren. Spätestens seit der Veröffentlichung von ChatGPT[1] durch OpenAI weiß jeder Bescheid über die Versprechungen und Bedrohungen, die von künstlicher Intelligenz ausgehen. Die Überzeugung, KI würde den eigenen Job ersetzen oder mindestens stark verändern ist allgegenwärtig. Die Studie von OpenAI über das Ersetzen von Arbeitskräften durch Systeme wie GPT gilt als Auslöser dieser Ängste [EMMR23].

Immer wieder wird von Unternehmen berichtet, die neue Meilensteine mit der Hilfe von ChatGPT und dem zugrundeliegenden Sprachmodell GPT-4 erreicht haben. Über den verblüffenden Inhalt der Antworten wird regelmäßig berichtet, und auch im Familien- und Bekanntenkreis hat jeder eine Anekdote darüber, welche überraschende Fertigkeit ChatGPT in einem bestimmten Kontext beweisen konnte. Immer mehr Schülerinnen und Schüler nutzen ChatGPT für Hausarbeiten und Präsentationen. Auch für die Beantwortung von Fragen wenden sich viele an ChatGPT, ob das empfehlenswert ist oder nicht. Zwischen diesen Anwendungen und den Cutting-Edge-Forschungen von Tech-Riesen und Start-Ups liegt jedoch ein weites Feld, das zwar großen Nutzen aus generativer KI ziehen könnte, bisher aber noch kaum berücksichtigt wurde.

Um die Anwendbarkeit von ChatGPT für praktische Arbeit zu testen, wurde im Rahmen dieser Arbeit ein Fall untersucht, in dem durch Finetuning ein deutsches vortrainiertes Sprachmodell (Pretrained Language Model (PLM)) auf die Erkennung von Relationen zwischen Unternehmen und Personen angepasst wird. Wie bei vielen ML-Anwendungen ist auch hierbei der beschränkende Faktor die Verfügbarkeit von handannotierten Trainingsdaten. Durch die Arbeit auf deutschen Daten steht kein Trainingskorpus zur Verfügung und es sind andere Methoden notwendig, um möglichst hochqualitative Daten für das Training zu erhalten.

Bisher hätte man hierfür, abhängig vom Level des nötigen Datenschutzes, auf Trainingsdaten zurückgreifen können, die von einem Skript regelbasiert erzeugt wurden („Synthetische Trainingsdaten“). Um jedoch ein Skript zu verfassen, ist viel Arbeit und Anpassung nötig, bis die daraus resultierenden Daten mit positivem Ergebnis dazu verwendet werden können, Finetuning oder Training an einem Modell durchzuführen [GL19]. Da die Erkennung, bzw. Extraktion von Relationen ein textbasiertes Problem ist, wäre es denkbar, hier die umfangreichen Fähigkeiten von ChatGPT zum Einsatz zu bringen. Dabei kann ChatGPT dazu genutzt werden, bereits annotierte Daten zu generieren, in welchen die Eigenschaften enthalten sind, die von dem ML-Modell erlernt werden sollen. Die Tatsache ob, und wie gut dies gelingt, bietet Einblicke in die Leistungsfähigkeit von ChatGPT und dem zugrundeliegenden Sprachmodell GPT-4.

[1] https://chat.openai.com

Die generierten Daten werden dann für das Finetuning eines PLM genutzt. Auf PLMs basiert die Technik, die es ermöglicht, natürliche Sprache zu verarbeiten und zu erzeugen [ZZL+23b]. Um ein PLM zu erzeugen ist ausführliches, unüberwachtes Training auf großen Textmengen nötig. Auf diese Weise entsteht ein grundlegendes Verständnis für Sprache und Grammatik. Je nach Art des Trainings haben die resultierenden Modelle verschiedene Stärken. In dieser Arbeit wird das BERT-Modell eingesetzt, welches im Finetuning mit Trainingsdaten die spezifische Aufgabe der Relation Extraction zwischen Personen und Unternehmen vornehmen soll. BERT-Modelle lernen architekturbedingt mehr Informationen pro Token, was bei de Klassifikation hilfreich sein kann, während sich die GPT-Modelle besonders gut für die Generierung von Inhalten eignen [CLL+23]. Während BERT im Rahmen dieser Arbeit durch Finetuning mit verschiedenen Trainingsdaten an die Aufgabe der Relation Extraction angepasst wird, wurde das Finetuning von ChatGPT bereits von OpenAI durchgeführt, weshalb es in dieser Arbeit ohne weitere Anpassungen verwendet werden kann.

Die Ergebnisse des Finetunings mit Daten, die von ChatGPT generiert wurden, werden innerhalb dieser Arbeit mit einer Baseline und dem Finetuning mit Daten, die durch einen regelbasierten Annotator erstellt wurden, verglichen. In der Baseline wird ein regelbasierter Ansatz zur RE eingesetzt, der kein ML oder PLMs verwendet. Bevor ML für die breite Masse zugänglich war, wurden solche regelbasierten Methoden auf spezifische Probleme angewendet. In dieser Arbeit bieten sie einen guten Vergleich zu der Verbesserung, die ML im Allgemeinen und PLMs im Speziellen ermöglichen.

Dieser Arbeit liegt ein großer Datensatz zugrunde, der vom Fraunhofer IAO über Jahre aufgebaut wurde und um die 1.8 Millionen Pressemitteilungen umfasst [FKKK09]. Da in diesen Pressemitteilungen in großer Häufigkeit über Personalwechsel berichtet wird, also über die in dieser Arbeit untersuchten Relationen zwischen Personen und Unternehmen, ist es naheliegend, diese als Quelle für Trainingsdaten zu nutzen. Dazu wurden aus diesem Datensatz Pressemitteilungen von Hand annotiert, von welchen ein Teil für die Evaluation, und der andere Teil als Trainingsdaten genutzt werden konnte. Der große Aufwand, der für die Handannotation von Daten betrieben werden muss, hat jedoch zur Folge, dass nur ein sehr kleiner Teil des Datensatzes von Hand annotiert werden konnte.

Um auch den Rest des Datensatzes in das Finetuning einfließen zu lassen, wurde ein Ansatz gewählt, bei dem echte Daten über einen regelbasierten Prozess annotiert wurden. In bisherigen Arbeiten erreichen regelbasierte Daten nicht die gleiche Qualität wie handannotierte [GL19], sie ermöglichen aber die Nutzung von beträchtlich mehr Daten. Da diese Daten aus echten Pressemitteilungen stammen, wird erwartet, dass es von Vorteil ist, größere Mengen dieser Daten im Finetuning zu nutzen, da das Modell so besser auf den Anwendungsfall vorbereitet wird. Dieser Vorteil sollte auch die etwas niedrigere Qualität der regelbasiert annotierten Daten ausgleichen [FMY95].

Die Baseline und die regelbasiert annotierten Daten liefern wertvolle Vergleiche zu der untersuchten Forschungshypothese dieser Arbeit: „Generative KI ermöglicht die Synthese bereits annotierter Trainingsdaten, die zum Finetuning von LLMs zur Relation Classification verwendet werden können. So können in speziellen Anwendungsfällen, in denen wenige echte Trainingsdaten zur Verfügung stehen, mit weniger Aufwand vergleichbare oder bessere Ergebnisse erzielt werden als mit herkömmlichen Methoden wie der regelbasierten Synthese von Trainingsdaten.“

Diese Arbeit ist in folgende Kapitel gegliedert: Im Kapitel 2 wird der Stand der Technik der beiden dieser Arbeit zugrundeliegenden Forschungsfelder beleuchtet, nämlich synthetische Trainingsdaten und Relation Extraction.

In Kapitel 3 werden die in dieser Arbeit verwendeten Methoden und Werkzeuge erläutert. Dazu werden in Abschnitt 3.1 verfügbare Methoden beschrieben, Künstliche Intelligenz zu verwenden. In Abschnitt 3.2 wird erläutert, wie regelbasiert Trainingsdaten annotiert werden können, und wo ähnliche Ansätze verwendet werden.

Kapitel 4 beschreibt den Datensatz des Fraunhofer IAO, sowie OpenNRE, ein Tool, das beim Training des BERT-Modells zum Einsatz kommt.

In Kapitel 5 werden die Vorarbeiten, die bis zur Durchführung der Experimente notwendig sind, beschrieben. Zu Beginn wird dafür in Abschnitt 5.1 die Baseline beschrieben, mit der die anspruchsvolleren Methoden verglichen werden. Der Prozess der Handannotation von echten Daten wird in Abschnitt 5.2 beschrieben, in den folgenden Abschnitten 5.3 und 5.4 folgen die Schritte, die zum Erhalt von automatisiert annotierten Daten und von ChatGPT generierten Daten notwendig sind. In den weiteren Abschnitten wird noch die Datenvorverarbeitung vor dem Finetuning des BERT-Modells, sowie die Experimente selbst beschrieben.

Kapitel 6 präsentiert die Ergebnisse der Experimente und ordnet diese in den Abschnitten 6.2 und 6.3 ein. Die Diskussion in Abschnitt 6.3 umfasst mehrere Ansätze, wie die Ergebnisse zu interpretieren sind, sowie Erklärungsversuche des beobachteten Verhaltens.

Kapitel 7 bietet eine Zusammenfassung dieser Arbeit, sowie einen Ausblick auf zukünftige Arbeiten.

2. Stand der Technik

In diesem Kapitel wird der Stand der Technik der beiden grundlegenden Forschungsbereiche dieser Arbeit vorgestellt. Darüber hinaus werden wichtige Entwicklungen auf diesen Gebieten bis hin zum aktuellen Stand der Forschung beschrieben und somit die Basis dieser Arbeit erläutert. In Abschnitt 2.1 wird auf synthetische Trainingsdaten im Kontext von Statistik und Machine Learning eingegangen. Abschnitt 2.2 bietet eine Übersicht zu Relation Extraction (RE) und definiert die verwandten Begrifflichkeiten.

2.1. Synthetische Trainingsdaten

Unter synthetischen Daten versteht man Daten, die zwar auf der Grundlage von echten Daten erstellt wurden und auch die gleichen statistischen Eigenschaften aufweisen wie echte Daten, aber künstlich erzeugt wurden [Luc21]. Dadurch kann zum Beispiel Datenschutz gewährleistet werden, was besonders durch die DSGVO [1] zusätzliche Relevanz erhalten hat. Es ist schon lange bekannt, dass unter bestimmten Voraussetzungen die Nutzung von synthetischen Daten von Vorteil sein kann, Donald B. Rubin hat bereits 1993 zum ersten Mal den Einsatz von synthetischen Daten vorgeschlagen [Rub93]. Im damaligen Kontext lag der Fokus bei der Nutzung synthetischer Daten jedoch noch darauf, die statistischen Eigenschaften von Datensätzen bei Gewährleistung von Vertraulichkeit zu nutzen. Synthetische Daten konnten diesen Anspruch erfüllen. Dieser Einsatzbereich von synthetischen Daten besteht bis heute und wurde über die Jahre stetig optimiert [Rag21; Rub04; SM17].

Das gesteigerte Interesse an Machine Learning und der damit verbundene Bedarf an Trainingsdaten hat synthetischen Trainingsdaten auch in diesem Bereich zu großer Bedeutung verholfen. Es ist in vielen Bereichen schwierig, Zugriff auf echte Daten, geschweige denn annotierte echte Daten zu erhalten. Zugriff auf existierende Datensätze kann aufgrund von Datenschutz, Sicherheits- und regulatorischen Gründen untersagt werden Selbst wenn diese Hürden überwunden werden, enthalten viele Datensätze zu wenige Datenpunkte oder Daten von geringer Qualität [LWW23]. Damit auch in solchen Fällen Forschung an ML-Systemen möglich ist, oder ML sogar in vorhandene Systeme integriert werden kann, sind synthetische Trainingsdaten von Nutzen. Wie auch die synthetischen Daten von Rubin bilden sie die Eigenschaften von echten Daten ab, ohne jedoch echte Daten zu beinhalten. Die enthaltenen Eigenschaften werden dabei üblicherweise von einem Programmierer definiert und können losgelöst von den echten Eigenschaften von Trainingssets manipuliert werden. Das erlaubt es, Trainingsdaten in genau der Menge zu erzeugen, die für einen gewissen Anwendungsfall notwendig ist.

[1] https://dsgvo-gesetz.de

Viele Bereiche des Machine Learning, vor allem im Bereich der Computer Vision, nutzen synthetische Daten, um bessere Ergebnisse von Modellen zu erhalten, die auf Bildern basierend arbeiten [Nik19]. Ein Schwachpunkt vieler Modelle im medizinischen Bereich liegt in der geringen Anzahl an Beispielen, die von den Symptomen seltener Krankheiten existieren, was das Training stark erschwert [CLC+21]. Synthetische Daten sind in solchen Fällen zu einem unverzichtbaren Werkzeug für Firmen und Labore geworden [LWW23]. Spezielle Systeme, die bei der Synthese von Trainingsdaten unterstützen können, wie zum Beispiel „SynSys“, ein Tool von Dahmen et al. [DC19] können beim Generieren von Trainingsdaten unterstützen. Um synthetische Trainingsdaten jedoch in dem theoretisch möglichen Umfang nutzen zu können, sind noch bessere Systeme notwendig, da die hohen Datenschutzanforderungen durch aktuelle Implementierung nicht automatisch gewährleistet und durch unvorsichtige Nutzung zu noch größeren Problemen führen können [CLC+21].

Auch bei der Optical Character Recognition (OCR) kommen synthetische Daten zum Einsatz [MLK19]. Dabei werden auch Methoden der Manipulation der Daten wie Unschärfe und Verzerrung eingesetzt, um die synthetischen Daten den echten Daten ähnlicher zu machen.

Ein Beispiel für die Nutzung synthetischer Daten auf größerer Skala ist das Crowd Counting, die Schätzung der Größe von Menschenmassen. Auch hier sind der Datenschutz, aber auch der große Aufwand, den das Annotieren von Bildern per Hand kostet, Gründe, aus denen auf synthetische Daten zurückgegriffen wird. Um realitätsnahe Bilder von Menschenmassen zu erhalten wird dabei auf ein Computerspiel zurückgegriffen (Grand Theft Auto V von Rockstar Games[2] aus dem Jahr 2013), das sich, aufgrund fortgeschrittener Grafik und der Option, für nichtkommerzielle Zwecke Modifikationen am Code vorzunehmen, für diesen Zweck anbietet. So kann eine Pipeline eingerichtet werden, die an verschiedenen Settings eine unterschiedliche Anzahl Personen anzeigen kann, und diese bereits exakt zählen und für die Evaluation mit einer Maske versehen kann. Dieses Vorgehen ist deutlich kostengünstiger als das Erwerben existierender Datensätze oder das Annotieren eigener Aufnahmen, und verbessert die Ergebnisse der Modelle signifikant [WGLY19].

Generative KI hat das Potential, einen großen Einfluss auf das Feld der synthetischen Trainingsdaten zu nehmen. Für Situationen, in denen auf Bildern trainiert wird, können multimodale Modelle genutzt werden, die aus Textinput Bilder generieren. Da die Bilder dem Input entsprechen, sind auf diese Art direkt Annotationen verfügbar. Für Situationen, in denen Text zum Training verwendet wird, können unimodale Modelle aus Prompts neuen Text generieren.

Im medizinischen Bereich werden schon länger erfolgreich Trainingsdaten verwendet, die von generativer KI erzeugt wurden [KWT+18]. Neue Entwicklungen bei der Bildsynthese erlauben es, diesen Ansatz auf immer anspruchsvollere und diversere Anwendungsfelder anzuwenden [JK23]. Auch GPT-4 und ChatGPT wurden bereits zur Synthese von Trainingsdaten verwendet, was durch die breite Verfügbarkeit und die überlegenen Fähigkeiten der Systeme ebenfalls für ein immer breiteres Spektrum an Anwendungsfeldern attraktiv sein wird [MDPA23].

[2] https://www.rockstargames.com/de/

2.2. Relation Extraction

Das Internet besteht aus Texten verschiedenster Formate, wie Nachrichten, Reportagen, Forschungspublikationen, Blogs, interkommunikativen Foren und sozialen Medien. In diesen Texten sind Informationen enthalten, auf die allerdings aufgrund ihrer unstrukturierten Natur nur schwer zugegriffen werden kann [PPB17]. Um den Zugriff auf die Informationen zu erleichtern, ist es erstrebenswert, sie auf eine strukturierte Weise, z.B. in Datenbanken, zu hinterlegen.

Der Forschungsbereich, der sich mit dieser Problematik beschäftigt, ist die Informationsextraktion (Information Extraction). Da eine Vielzahl unstrukturierter Daten im Internet in der Form von natürlicher Sprache vorliegt, und in dieser auch die Informationen gefunden werden müssen, gehört Informationsextraktion zum Bereich des Natural Language Processing (NLP). Es können viele Arten von Informationen aus Texten extrahiert werden, die häufigsten drei sind jedoch:

- Entitäten wie Personen, Organisationen oder Orte
- Relationen zwischen diesen Entitäten, wie zum Beispiel `Kind-von`, `wohnt-in` oder `Hauptstadt-von`
- Events, zu denen typischerweise die Fragen „Wer, wann, wo, was, warum“ und „wie“ beantwortet werden müssen [XW19]

Im Rahmen dieser Arbeit liegt der Fokus auf dem Extrahieren von Relationen, der sogenannten Relation Extraction (RE). Sie ist ein Teilgebiet der Information Extraction und setzt sich aus mehreren Teilschritten zusammen. Diese sind im Allgemeinen die Named Entity Recognition (NER), die Relation Identification (RI) und Relation Classification (RC), wobei diese Struktur von Fall zu Fall unterschiedlich ist [BP22]. In Abbildung 2.1 wird dargestellt, in welchen Schritten die RE abläuft und was bei jedem Schritt passiert. Bei der NER werden die Entitäten im Text erkannt und markiert. Die Relation Identification (RI) stellt daraufhin fest, zwischen welchen Entitäten eine Relation besteht, und zwischen welchen nicht. Zuletzt wird bei der Relation Classification (RC) entschieden, welche Relation zwischen den von der RI bestimmten Entitäten besteht. Dies ist typischerweise eine Klassifikation mit mehreren Klassen („Multiclass Classification“) [BP22].

Um die Entitäten und letztlich Relationen aus natürlichem Text zu extrahieren, wurden über lange Zeit regelbasierte Ansätze genutzt, die vordefinierte Muster in den Sätzen erkennen und daraus Verhältnisse ableiten [AB08]. Dies ist mit viel Handarbeit verbunden, da die Muster in natürlicher Sprache sehr vielfältig sind, zusätzlich ist ein umfangreiches Regelwerk nötig, um einen signifikanten Anteil der im Text enthaltenen Entitäten und Relationen zu erkennen [Gri97]. Außerdem wird auch kritisiert, dass dieser Ansatz typischerweise zu relativ hoher Precision, aber dafür zu einem schlechten Recall führt. Die Relationen, für die Muster definiert sind, werden dann in der Regel korrekt erkannt, diese Muster kommen aber zu selten in Texten vor [CPSS05]. Der erste Ansatz, der auf dieser Technik basiert, wurde von 1992 von Marti A. Hearst vorgeschlagen [Hea92].

Eine grundlegende Abweichung von den Hearstschen Prinzipien erfolgte 2015 mit dem Vorschlag von Nguyen et al., Convolutional Neural Network (CNN), eine Methode des Deep Learning auf das Problem anzuwenden [NG15]. Eine umfangreiche Übersicht der auf Deep Neural Networks basierenden Methoden für RE bieten Wang et al. [WQZ+22]. Mit der Entwicklung von Pretrained Language Models (PLM) wie BERT werden auch diese für Relation Extraction angewendet [SFS+21]. Dies führt zu einer deutlichen Verbesserung in Bezug auf Testsets im Vergleich zu vorherigen Methoden wie CNNs [BP22].

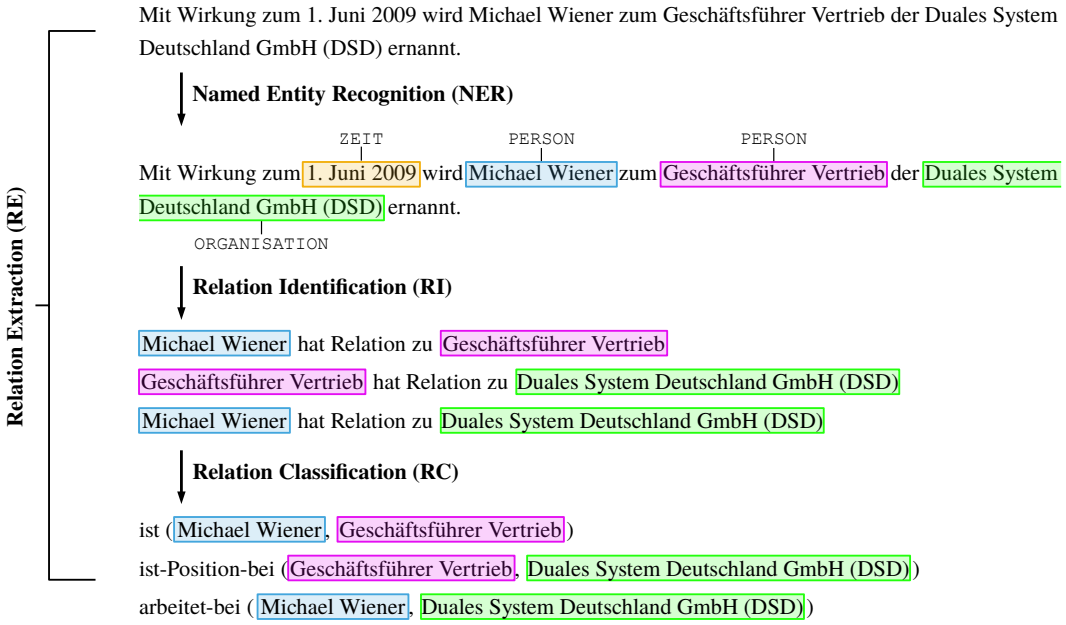

Abbildung 2.1.: Graphische Darstellung der Schritte, aus denen Relation Extraction besteht.

Um ein PLM in einem bestimmten Anwendungsgebiet nutzen zu können, ist jedoch vorher sogenanntes „Finetuning“ notwendig. Beim Finetuning wird das PLM mit Beispielen aus dem spezifischen Anwendungsbereich nachtrainiert. So kann das Modell das vorhandene Sprachverständnis auf das bestimmte Problem anwenden, was bei ausreichendem Tuning die deutliche Verbesserung der Leistung in Bezug auf Testsets ermöglicht.

3. Grundlagen dieser Arbeit

In diesem Kapitel werden die Methoden beschrieben, die als Basis für die in dieser Arbeit eingesetzten Techniken dienen. In Abschnitt 3.1 werden aktuelle Entwicklungen im Feld der KI beschrieben, und welchen Einfluss sie auf diese Arbeit haben. Ein besonderer Fokus liegt auf den verwendeten Modellen BERT und dem zugehörigen Feld der unterscheidenden KI (Unterabschnitt 3.1.1) und auf GPT und dem zugehörigen Feld der generativen KI (Unterabschnitt 3.1.2). In Abschnitt 3.2 wird beschrieben, wie regelbasiert annotierte Daten zu besseren Trainingsergebnissen führen können, sowie ein praktisches Tool, welches dabei unterstützen kann. Abschnitt 3.3 beschreibt den Aufbau der Trainingsdaten, die für das Finetuning von LLMs notwendig sind. Mit diesen Trainingsdaten lässt sich das Problem der RE, welches in dieser Arbeit behandelt wird, mit der Hilfe von KI in der Form von LLMs lösen.

3.1. Künstliche Intelligenz (KI) und Sprachmodelle

Künstliche Intelligenz ist schon seit den ersten Versuchen mit Computern um 1950 ein aktives Forschungsfeld [Tur50], das über die Jahre viele große Errungenschaften feiern konnte. Ein zentraler Bereich der KI ist das Machine Learning / Maschinelle Lernen (ML). Bei ML erstellt sich der Computer, ohne dass ein Mensch die Parameter vorgibt oder Zusammenhänge definiert, ein Modell der Realität. Um dieses zu erzeugen, werden ihm sogenannte Trainingsdaten gezeigt, die den Teil der Realität umfassen, für den der Computer ein Verständnis entwickeln soll. Das kann fast alles sein, von Bildern einer bestimmten Epoche über handgeschriebene Buchstaben bis hin zu sämtlichen Wikipedia-Einträgen. Basierend auf diesen Daten wird das Modell an diese Trainingsdaten angepasst, und es werden Muster erkannt. Je mehr Trainingsdaten vorliegen, desto höher ist die Wahrscheinlichkeit, dass diese Muster korrekt sind, sich also auch auf weitere Daten verallgemeinern lassen. Bei zu wenig vorliegenden Trainingsdaten können Probleme auftreten, indem das Modell Muster erkennt, die nur in dem speziellen Datensatz auftreten, sich aber nicht auf andere Daten der gleichen Art übertragen lassen. Das Problem ist also eine Überanpassung an den Trainingsdatensatz. Muster, die in den Daten vorhanden wären und für die Verallgemeinerung auf weitere Daten nötig wären, können so womöglich nicht erkannt werden. Daraus ergibt sich eine grundlegende Problematik des ML: Wie erhält man ausreichend viele Trainingsdaten, um sämtliche Nuancen des Beispielraumes abzudecken, Überanpassung zu vermeiden und so viele Muster und Variationen wie möglich zu erfassen?

Trainingsdaten können verschiedene Formen haben, welche von der Art des Trainings abhängen. Drei Ansätze werden dabei häufig verwendet:

- Überwachtes Lernen (Supervised Learning)
  Die Trainingsdaten müssen annotiert sein, sodass für die Daten sowohl die Eingabewerte, als auch die gewünschten Ausgabewerte bekannt sind. Dafür werden die Daten mit Labels

versehen, die die Eigenschaften beinhalten, die erlernt werden sollen. So kann das Modell seine Antworten direkt mit den gewünschten Antworten vergleichen und seine Struktur dementsprechend anpassen. Auf diese Art der Trainings werden die besten Modelle trainiert, allerdings ist es oft extrem teuer, genügend annotierte Trainingsdaten zu beschaffen.

- Unüberwachtes Lernen (Unsupervised Learning)
  Hierfür werden unannotierte Trainingsdaten verwendet, anhand derer das Modell Strukturen feststellt und die Daten anhand dieser Strukturen gruppieren kann. Aus dieser Gruppierung kann das Modell Informationen ableiten, wobei die Abbildung auf einen Anwendungsfall in der Regel nicht trivial ist.
- Semi-überwachtes Lernen (Semi-Supervised Learning)
  In diesem Ansatz werden die Ansätze des überwachten Lernens und des unüberwachten Lernens zusammengeführt, dementsprechend werden auch sowohl annotierte als auch unannotierte Trainingsdaten benötigt. Die Gruppierungen des unüberwachten Ansatzes können über die klaren Verknüpfungen des überwachten Lernens genutzt werden, was die Vorhersageleistung verbessern kann. Bei diesem Ansatz ist der Anteil unannotierter Trainingsdaten wesentlich größer als der annotierter Trainingsdaten.

Diese Methoden werden auch beim Deep Learning (DL) eingesetzt. Beim DL ist das zugrundeliegende Modell ein künstliches neuronales Netz. Je mehr Knotenpunkte (Neuronen) und Schichten dieses Netz hat, desto komplexere Sachverhalte können in ihm repräsentiert und hinterlegt werden. Dieser Ansatz wurde zuerst 1986 von Rumelhart, Hinton und Williams vorgestellt [RHW86] und seitdem durch leistungsfähigere Hardware und optimierte Prozessoren stets verbessert. Der letzte große Meilenstein, der auch zur Entwicklung von ChatGPT geführt hat, war die Einführung von Transformern durch Vaswani et al. [VSP+17]. Dies ist eine neue Architektur zur Verarbeitung von Texten, die herkömmliche Methoden bei weitem übertrifft. Sie hat es ermöglicht, dass vortrainierte Sprachmodelle (Pretrained Language Models, PLM) heutzutage über ein breites Verständnis natürlicher Sprache verfügen. Die Leistung dieser PLMs steigt mit der Anzahl der Parameter, die innerhalb der Modelle verwendet werden. Das Erhöhen dieser Parameteranzahl führte in den resultierenden Large Language Models (LLMs) jedoch nicht nur zu einer besseren Leistung, sondern auch zu neuen Fähigkeiten wie z.B. kontextbezogenem Lernen, die kleine PLMs nicht besitzen. Es gibt keine klare Definition, ab welcher Grenze ein PLM ein LLM ist, aber der Konsens liegt bei 10 bis 100 Milliarden Parametern [ZZL+23b].

Transformer sind auch die Basis zweier aktueller Modelle, BERT (Bidirectional Encoder Representations from Transformers) von Devlin et al. [DCLT19] sowie GPT (Generative Pretrained Transformer) von Radford et al. [RNS+18]. Diese beiden Modelle gehören trotz ihrer ähnlichen Architektur zu zwei verschiedenen Arten von ML-Modellen: BERT gehört zur unterscheidenden (discriminative) KI, GPT hingegen zur erzeugenden, generativen KI. Diese beiden Arten sind in Abbildung 3.1 grafisch dargestellt.

3.1.1. Unterscheidende KI

Die unterscheidende KI (discriminative Artificial Intelligence, discriminative AI) klassifiziert Datenpunkte, oder sagt Datenpunkte nach zuvor erlernten Kriterien voraus. Diese Kriterien kann das Modell verallgemeinern, sodass auch Datenpunkte, die während des Trainings nicht behandelt

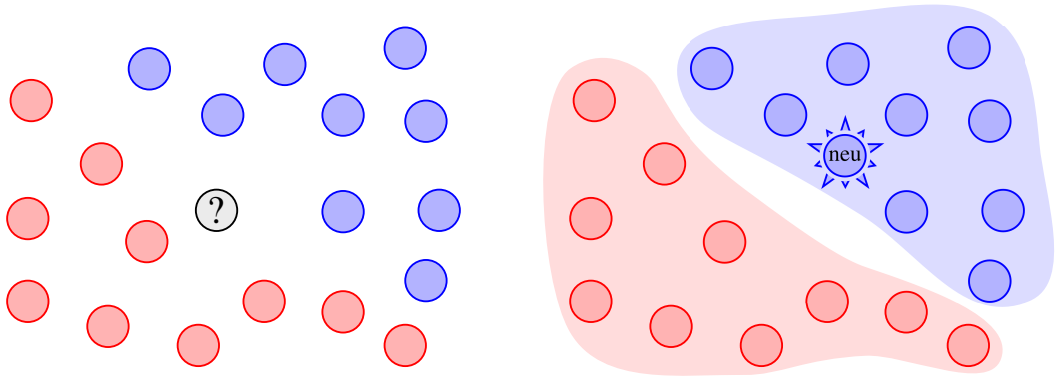

Abbildung 3.1.: Graphische Darstellung unterscheidender und generativer KI. Links unterscheidende (discriminative) KI, die Datenpunkte nach erlernten Kriterien einer Klasse zuordnet. Rechts generative KI, die basierend auf den erlernten Eigenschaften der Klassen neue Datenpunkte erstellt.

wurden, zugeordnet werden können. Insbesonders BERT, entwickelt von einem Team von Google-Ingenieuren, hat auf diesem Feld große Verbesserungen bewirkt. Für das Training von BERT wurde natürlicher Text aus dem BookCorpus [ZKZ+15] und der englischen Wikipedia verwendet. Um mit diesen Daten ein Modell zu trainieren, wurde ein Ansatz gewählt, den die Autoren „Masked Language Model“ nennen. Hierbei wird in Sätzen zufällig ein Wort maskiert, und das Modell damit beauftragt, dieses Wort vorherzusagen. Die Vorhersage des Modells wird dann anhand des echten Wortes bewertet. Dieser Ansatz unterscheidet sich von bisherigen Ansätzen, da er den kompletten Satz und damit den gesamten Kontext mit in den Entscheidungsprozess einbezieht [DCLT19].

Seit der Veröffentlichung von BERT, dessen Code Open-Source veröffentlicht wurde, wurden noch einige Verbesserungen vorgenommen. Liu et al. fanden heraus, dass BERT nicht lange genug trainiert wurde, und optimierten den Prozess für die neue Version des Modells: RoBERTa („Robustly Optimized BERT Pretraining Approach“). Seitdem wurden viele weitere verbesserte Ansätze und Modelle entwickelt, die auf BERT basieren [CKG+19; HBT+21; ZWYJ21].

Bevor nun ein solches LLM, welches über ein allgemeines Sprachverständnis verfügt, an einer konkreten Aufgabe angewendet wird, ist noch sogenanntes „Finetuning“ mit annotierten Trainingsdaten notwendig. Beim Finetuning wird dem Modell vermittelt, auf welche Eigenschaften in dem bestimmten Anwendungsfall zu achten ist. Erst hier kann das Modell das erlernte Wissen über natürliche Sprache einsetzen, und Verknüpfungen zwischen gewissen Satzkonstruktionen, Wörtern oder Phrasen schaffen. Diese Verknüpfungen können dann verwendet werden, um die beim Finetuning eingeführten Labels auf neue Beispiele anzuwenden.

3.1.2. Generative KI

Generative KI kann im Gegensatz zur unterscheidenden KI zu einer gegebenen Klasse, deren Eigenschaften im Training erlernt wurden, neue Datensätze erzeugen. Diese erfüllen dann alle Eigenschaften der anderen Datenpunkte dieser Klasse, sind aber keine einfache Kopie. Bekannte Beispiele dafür sind DALL-E, entwickelt von Ramesh et al. bei OpenAI [RPG+21], ein Modell, welches Bilder auf Basis von textbasierten Prompts generiert, sowie GPT (Generative Pretrained

Transformer), entwickelt von Radford et al. bei OpenAI [RNS+18], ein Modell, welches auf Basis von textbasierten Prompts textbasierte Antworten generiert. Im Rahmen dieser Arbeit liegt der Fokus auf Text-zu-Text Modellen, weshalb sich der Fokus auf GPT beschränkt. Seit 2018 wurde auch GPT stetig optimiert, mit einer größeren Menge Trainingsdaten, neuer Architektur und neuen Trainingsmethoden, die zu besseren Ergebnissen in verschiedenen Szenarien führen sollen. In Tabelle 3.1 sind die bekannten Unterschiede zwischen den Modellen aufgelistet. Die grundlegende Trainingsmethode ist jedoch bei allen Versionen von GPT gleich: Es wird auf der Basis großer Mengen von unannotierten Daten trainiert, und im Gegensatz zu BERT wurde autoregressive Sprachmodellierung genutzt. Das bedeutet, dass das Modell darauf trainiert wird, die Wahrscheinlichkeit des nächsten Token eines Satzes zu modellieren, basierend auf allen bisherigen Tokens. Dieser, von rechts nach links gehende Ansatz, eignet sich besser als Modelle, die mit dem „Masked Language Model"-Ansatz trainiert wurden für die Generierung von Daten.

Wie auch BERT wurde die erste Version von GPT unter anderem auf der Basis des BookCorpus [ZKZ+15] trainiert. Dies ist möglich, da für beide Modelle keine annotierten Trainingsdaten notwendig sind.

Modell	GPT	GPT-2	GPT-3	GPT-3.5	GPT-4
Publikation	[RNS+18]	[RWC+19]	[BMR+20]	—	[Ope23]
Parameter (in Millionen)	117	1 500	175 000	n. a.	n. a.
Trainingsdaten	> 40 GB	40 TB	45 TB	n. a.	n. a.

Tabelle 3.1.: Die Unterschiede zwischen GPT-Versionen. Werte für GPT-3.5 und GPT-4 sind nicht angegeben (n. a.).

Im Gegensatz zu BERT sind die GPT-Modelle nicht öffentlich zugänglich. Ausgewählte Entwickler haben jedoch die Möglichkeit, die Modelle über eine API (Application Programming Interface, eine Schnittstelle zu den Servern von OpenAI) zu nutzen. Bevor im November 2022 ChatGPT der Öffentlichkeit zugänglich gemacht wurde[1], war dies der einzige Weg, mit den Modellen zu interagieren.

Die Qualitäten von LLMs werden besonders im Schritt von GPT-2 zu GPT-3 deutlich. Während der Trainingsprozess derselbe bleibt, wird hauptsächlich die Modellgröße skaliert, von 1.5 Milliarden Parametern auf 175 Milliarden. So kann GPT-3 etwa durch kontextbezogenes Lernen Aufgaben mit wenigen erklärenden Beispielen lösen, eine Qualität von LLMs, die in GPT-2 nicht ausgeprägt ist. Bei der aktuellen Version, GPT-4, argumentieren einige Forscher dafür, sie würde deutliche Anzeichen von Intelligenz besitzen [BCE+23]. Ob sie tatsächlich Intelligenz besitzt oder nicht, die von GPT-4 erzeugten Texte lassen sich jedenfalls kaum von Texten unterscheiden, die von einem menschlichen Autor verfasst wurden.

Dies bietet eine interessante Chance für die natürliche Sprachverarbeitung (Natural Language Processing, NLP), besonders die Relation Extraction (RE): Um ein LLM für die Lösung eines speziellen Problems einsetzen zu können, ist, wie bereits erwähnt, noch Finetuning notwendig, wofür annotierte Trainingsdaten aus dem Anwendungsfall benötigt werden. Diese zu erhalten ist schwierig, da sie ein breites Feld natürlicher Sprache abdecken müssen und für ein gutes Ergebnis

[1] https://openai.com/blog/chatgpt

möglichst viele Trainingsdaten notwendig sind. Der Inhalt der Trainingsdaten ist dabei nicht relevant, stattdessen lernt das LLM die Eigenschaften relevanter Wörter, und in welchem Satzkontext sie auftreten.

Diese Unterscheidung ist wichtig, da Halluzinationen ein großes Problem von generativer KI darstellen. Darunter versteht man das Erfinden plausibel klingender Informationen durch LLMs. Halluzinationen entstehen typischerweise, wenn die Trainingsdaten das abgefragte Feld nicht ausreichend abdecken. Es ist für den Nutzer jedoch sehr schwierig Halluzinationen als solche zu erkennen, da LLMs nicht in der Lage sind festzustellen, wenn sie halluzinieren [BCL+23]. Somit müsste der Nutzer die Informationen eigenhändig überprüfen, oder Widersprüche zu bekannten Fakten feststellen, um Halluzinationen zu entlarven.

Wie bereits erwähnt spielen Halluzinationen aber im Kontext des Finetunings von LLMs für RE keine Rolle, da dort der Fokus auf den Satzstrukturen liegt. Satzstrukturen und natürliche Sprache werden von LLMs jedoch in der Regel fehlerfrei synthetisiert. Im Rahmen dieser Arbeit wird untersucht, wie gut sich generative KI am konkreten Beispiel von ChatGPT, basierend auf GPT-4, dazu eignet, annotierte Trainingsdaten für das Finetuning eines LLMs zu erstellen.

3.1.3. Erklärbare KI

Erklärbare KI (Explainable Artificial Intelligence, XAI) hat sich in den vergangenen Jahren zu einem sehr aktiven Forschungsgebiet entwickelt, da der stetig wachsende Einfluss der Entscheidungen von KI die Problematik der Nachvollziehbarkeit immer stärker in den Fokus rückt [IABB22]. Beschleunigt wurde dieser Prozess durch die Förderung durch große Wirtschaftsmächte. In den USA wurde durch die DARPA im April 2017 das „Explainable AI (XAI) Program“ („Erklärbare KI Programm“) finanziert, welches zum Ziel hat, die Entscheidungsfindung von KI besser nachvollziehbar zu machen [GA19]. China veröffentlichte im Juli 2017 den „Entwicklungsplan für die neue Generation KI“, mit dem Ziel, erklärbare und erweiterbare KI zu erforschen [WC18]. Die Europäische Union verabschiedete im Mai 2018 die DSGVO, ein Gesetz, welches jedem Bürger bei der Interaktion mit KI Anspruch „auf Erläuterung der nach einer entsprechenden Bewertung getroffenen Entscheidung“ (Erwägungsgrund 71, EU-DSGVO) zuspricht [WMR17]. Seit diesen Entwicklungen ist die Anzahl an Publikationen auf dem Gebiet der Explainable Artificial Intelligence rasant angestiegen [IABB22].

3.2. Regelbasierte Annotation von Trainingsdaten

Ein Ansatz, der zwischen rein synthetischen und handannotierten, echten Daten steht, ist der regelbasierte Ansatz. Hierbei werden echte, unannotierte Daten verwendet, die dann über einen regelbasierten Prozess annotiert werden. Dieser Prozess wird von einem Menschen definiert und programmiert, der ein tiefes Verständnis für die annotierten Daten hat. Somit kann er Regeln und Ausnahmen in diesem Prozess berücksichtigen, die über das, was aus den Beispielen enthalten ist, die dem Menschen vorliegen, hinausgeht. Um mit diesem Ansatz Daten zu generieren sind also nur sehr wenige handannotierte Datensätze als Vorlage für die regelbasiert erzeugten Resultate notwendig. Ein Beispiel dafür, was ein menschlicher Annotator aus einem einzigen annotierten Satz über das zugrundeliegende Modell und die zugrundeliegenden Daten schließen kann, ist in

Abbildung 3.2 dargestellt. Damit jedoch viele Annotationen regelbasiert erstellt werden können ist ein großer Korpus von Daten notwendig, die auf den Anwendungsbereich passen. Die Verwendung von Daten aus dem Anwendungsbereich führt dazu, dass das resultierende Modell auf genau die Daten angewendet wird, auf denen es trainiert wurde. Dies unterscheidet diesen Ansatz von rein synthetischen Ansätzen, da synthetische Daten kaum alle Besonderheiten und Eigenarten der echten Daten des Anwendungsgebiets widerspiegeln können.

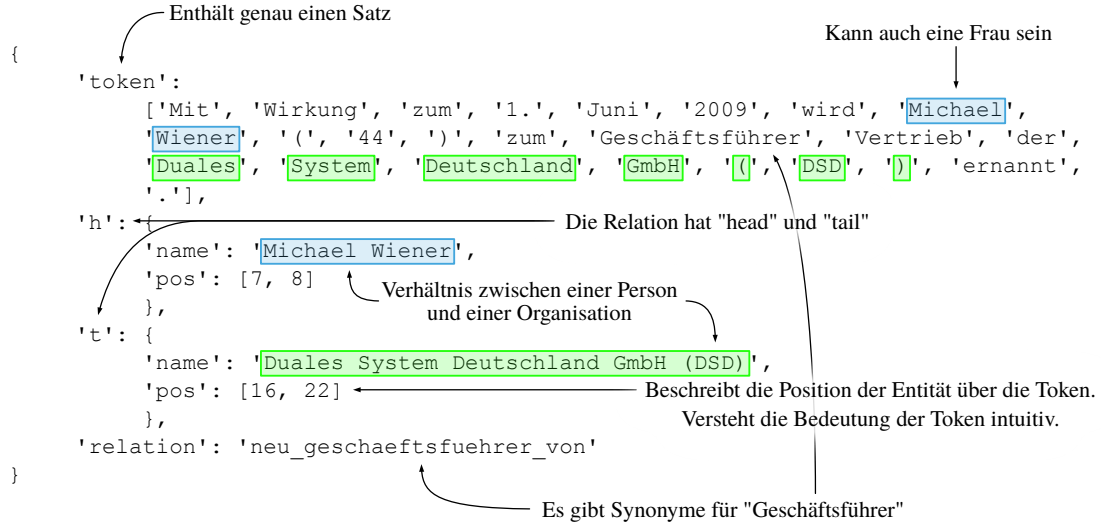

Abbildung 3.2.: Beispiele für Informationen, die ein menschlicher Annotator aus nur einem annotierten Satz ziehen kann.

In medizinischen Bereichen ist es teurer, handannotierte Trainingsdaten zu erlangen als in den meisten anderen Disziplinen, da hierzu Expertenwissen nötig ist. Gu und Leroy vergleichen in ihrer Arbeit zu Diagnosekriterien für das Autismusspektrum [GL19] einen regelbasierten Parser mit ML-Methoden der Annotation und stellen fest, dass der regelbasierte Parser in vielen Testsets bessere Ergebnisse erzielt. Außerdem konnten die Daten des regelbasierten Parsers in Kombination mit handannotierten Daten das resultierende Modell verbessern. Dies belegt, dass dieser Ansatz in Situationen, wo Daten auf andere Art nur schwer oder gar nicht zu erhalten sind, mit regelbasiert annotierten Daten Abhilfe geschaffen werden kann. Der verwendete regelbasierte NLP-Parser wurde im Rahmen einer früheren Arbeit von Leroy und Gu [LGP+18] entworfen und umfasst 104 Patterns (Muster zur Erkennung von Zeichensequenzen) und 94 Wörterbücher mit insgesamt 1 787 Einträgen, welche zum Zuordnen verschiedener Krankheitsbilder verwendet wurden. Mit dem Parser können zwölf psychische Krankheiten mit einer Genauigkeit (Precision) zwischen 76 % und 97 % und einem Trefferquote (Recall) zwischen 43 % und 57 % erkannt werden.

Es existieren jedoch auch offensichtliche Begrenzungen für diesen Ansatz. So können aus den somit erhaltenen Daten beim Training eines Modells keine Muster erkannt werden, die nicht bereits von dem Experten über Regeln in den Prozess der regelbasierten Annotation integriert wurden. Ein Training mit rein regelbasiert annotierten Daten sollte also zu keinem Mehrwert führen, es ist also in der Regel notwendig, dass auch Zugriff auf andere Trainingsdaten besteht. Außerdem kann die Gestaltung des Prozesses für die regelbasierte Annotation je nach Anwendungsfall sehr zeitintensiv sein, da ein tiefes Verständnis für die Daten vorausgesetzt wird, (korrekte) Annahmen

über die Daten getroffen werden müssen und diese dann noch mit sämtlichen Sonderfällen, die bei natürlicher Sprache auftauchen, implementiert werden müssen. Sofern keine andere Option besteht, in großem Stil annotierte Daten zu erlangen, kann das sinnvoll sein. Je nach Szenario muss aber evaluiert werden, ob dieser Aufwand verhältnismäßig ist.

3.2.1. SpaCy

SpaCy[2] ist eine Open-Source-Bibliothek für Natural Language Processing und Textverarbeitung. Es bietet umfangreiche Werkzeuge und Modelle für eine Vielzahl von Anwendungen: Unter anderem das Teilen von Paragraphen in Sätze („Sentencization“), das Teilen von Sätzen in Token („Tokenization“), das Benennen grammatikalischer Satzbestandteile pro Wort („Part-Of-Speech-Tagging (POS)“), das Zurückführen von Wörtern auf ihre Grundform („Lemmatization“) und die Klassifikation von Token in Entitäten („NER“). Anderen Open-Source NLP-Werkzeuge wie NLTK (Natural Language Toolkit)[3] legen den Fokus auf die Implementierung vieler verschiedener Ansätze, und wollen diese auch für die Lehre verständlich machen. Im Gegensatz dazu legt SpaCy den Fokus auf Leistung und Effizienz. Daher beinhaltet SpaCy nur die fortschrittlichsten Algorithmen und Modelle. SoZugriff sind bereits vortrainierte Modelle für über 24 Sprachen in SpaCy enthalten, wodurch es einen sehr kurzen Weg zu einer funktionierenden NLP-Pipeline bietet. Diese Pipeline erlaubt das schnelle und effiziente Verarbeiten von natürlichem Text, und bildet eine gute Basis, um mit verhältnismäßig wenig Aufwand gute Ergebnisse zu erlangen.

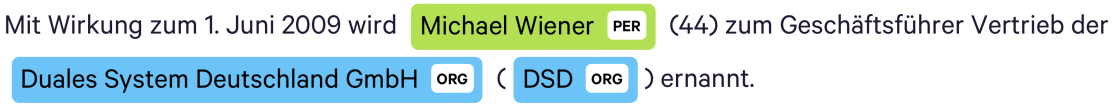

Abbildung 3.3.: Ein Beispielsatz, annotiert von der Named Entity Recognition von SpaCy. Visualisiert mit dem displayCy-Package.

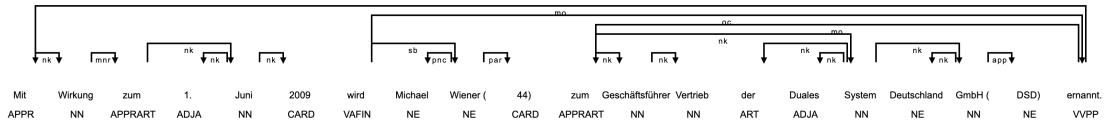

Abbildung 3.4.: Derselbe Beispielsatz wie in Abbildung 3.3, annotiert vom Part-Of-Speech-Tagging von SpaCy. Für jedes Wort im Satz wird die grammatikalische Rolle sowie der Bezug auf andere Wörter vorhergesagt. Visualisiert mit dem displayCy-Package.

Mit Hilfe dieser Optionen können leicht komplexe Aufgaben auf natürlichem Text ausgeführt werden, und die Intuitionen des menschlichen Annotators als Regeln umgesetzt werden. Über die NER von SpaCy werden Wörter aus dem Satz direkt Klassen wie Person (`PER`), Ort/Location (`LOC`), Organisation (`ORG`) oder Sonstige/„Miscellaneous“ (`MISC`) zugeordnet, wie auch im Beispiel in Abbildung 3.3 dargestellt. Im Rahmen dieser Arbeit sind lediglich die beiden Klassen `PER` und

[2] https://spacy.io
[3] https://www.nltk.org/index.html

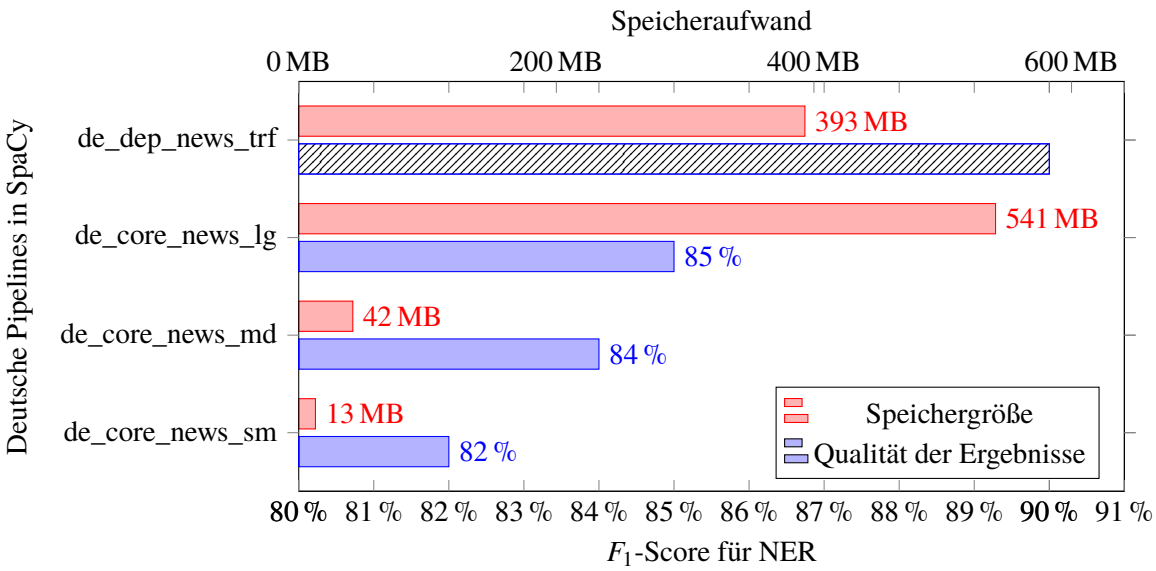

Abbildung 3.5.: Gegenüberstellung von Speicheraufwand in MB (Megabyte) und Qualität der deutschen SpaCy-Pipelines am Beispiel des F_1-Scores für NER. Die Pipeline `de_dep_news_trf` unterstützt zum Zeitpunkt der Experimente noch keine NER. Der schraffierte Balken, der für `de_dep_news_trf` einen F_1-Score von 90 % angibt ist aus der englischen Version des Modells übernommen, da mit einer ähnlichen Leistung zu rechnen ist, sobald NER auch auf dem deutschen Modell verfügbar ist.

`ORG` relevant, da zwischen diesen Klassen Relationen definiert werden. Durch das POS-Tagging lassen sich feingranularere Besonderheiten in der Sprache erkennen und auswerten. Dies kann zur besseren Erkennung von Satzstrukturen führen, die sich an bestimmte Muster halten.

SpaCy verfügt über mehrere Pipelines in deutscher Sprache, die auf verschiedenen Modellen ausgeführt werden. Ein typischer Tradeoff ist hier die Größe des Modells und die Qualität der Ergebnisse. Große Modelle brauchen länger in der Ausführung und benötigen mehr Speicher, erzielen aber auch etwas bessere Resultate. Im Falle der von SpaCy zur Verfügung gestellten Modelle sind die Unterschiede bezüglich benötigtem Speicherplatz zwischen der größten Pipeline (`de_core_news_lg`, 541 MB) und der kleinsten Pipeline (`de_core_news_sm`, 13 MB) sehr groß, die F_1-Scores der NER unterscheiden sich jedoch nur um wenige Prozentpunkte (85 % beim großen Modell, 82 % beim kleinen Modell). In Abbildung 3.5 wird das Verhältnis von Speichergröße zur Qualität der Ergebnisse dargestellt.

Vielversprechend ist der Ansatz, eine transformerbasierte Pipeline zu nutzen. Für die Englische Sprache existiert bereits ein solches Modell mit vollem Funktionsumfang und einem F_1-Score bei NER von 90 %. Leider unterstützt die deutsche transformerbasierte Pipeline `de_dep_news_trf` zum Zeitpunkt der Experimente keine NER, was jedoch von zentraler Bedeutung für diese Arbeit ist.

Aufgrund dieser Einschränkung wurde im Rahmen dieser Arbeit die Pipeline `de_core_news_lg` verwendet, um die bestmöglichen Features nutzen zu können. Das zugrundeliegende Modell dieser Pipeline wurde mit Daten aus dem TIGER Korpus [BDE+04] (aufbereitet im Rahmen von GRAIN-S

[FCJ+20] bereinigt), WikiNER [NRR+17], OSCAR [OSR19] und weiteren trainiert. In Tabelle 3.2 wird die von SpaCy für die `de_core_news_lg`-Pipeline angegebene Leistung in verschiedenen Bereichen des NLP angegeben. Basierend darauf ist SpaCy eine solide Basis für die Arbeit mit regelbasierter Annotation.

Komponente	Precision	Recall	F_1-Score	Accuracy
Tokenization	100 %	100 %	100 %	100 %
Sentencization	95 %	96 %	95 %	—
NER	85 %	84 %	85 %	—
POS	—	—	—	98 %
Lemmatization	—	—	—	98 %

Tabelle 3.2.: Die von den SpaCy-Entwicklern veröffentlichten Evaluationsdaten verschiedener SpaCy Komponenten der `de_core_news_lg`-Pipeline.

3.3. Daten für das Training von Relation Classification

„Relation Classification“ ist ein Feld des Natural Language Processing (NLP), welches sich mit dem Verhältnis zweier vorliegender Entitäten zueinander befasst. Sind also zwei Entitäten in ihrem Kontext gegeben, wird von Relation Classification die Relation beider Entitäten zueinander vorhergesagt. Diese Aufgabe zählt zu den klassischen ML-Aufgaben, doch wie in der Motivation beschrieben, wird hier versucht, die Stärken von Large Language Models (LLMs) auszunutzen. Um diese vortrainierten Modelle an das vorliegende Relation Classification Problem anzupassen sind dennoch annotierte Trainingsdaten notwendig, die sich der gleichen Grundsätze des klassischen ML bedienen. Diesen Schritt nennt man auch „Finetuning“. Hier lernt das Modell, welche sprachlichen Besonderheiten im Kontext der Entitäten auftauchen. Um ein Modell gut für Relation Classification anzupassen, sind möglichst viele und möglichst hochqualitative Trainingsdaten notwendig.

3.3.1. Inhalt

Die Trainingsdaten enthalten den Textabschnitt, aus dem sich eine Relation, wie sie von der Relation Classification erkannt werden soll, ableitet. Im Fall dieser Arbeit beschränken sich die Textabschnitte auf einzelne Sätze. Innerhalb dieser Sätze sind Personen und Institutionen als solche markiert. Zwischen diesen Entitäten werden die Relationen definiert. Diese Annotation eines Satzes wird in Abbildung 3.6 anhand eines Beispiels dargestellt.

Im Rahmen dieser Arbeit werden die möglichen Relationen auf eine vordefinierte und an den Daten orientierte Auswahl reduziert. Verhältnisse zwischen Personen und Institutionen, die nicht in diese vorausgewählten Relationen passen, werden unter „Sonstige“ zusammengefasst.

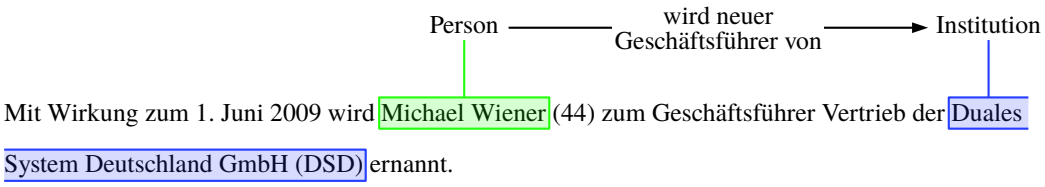

Abbildung 3.6.: Zusätzlich zu dem eigentlichen Satz sind diese Annotationen für das Finetuning eines vortrainierten Modells notwendig.

4. Daten & Tools

Im Folgenden wird der Datensatz sowie das Toolkit beschrieben, mit denen im Rahmen dieser Arbeit geforscht wurde. Der Datensatz stammt vom Fraunhofer IAO und besteht aus Pressemitteilungen, die von 2003 bis 2017 gesammelt wurden. Durch den großen Umfang des Datensatzes und ein gut gepflegtes Metadatensystem bietet sich der Korpus für vielerlei Analysen an. In Abschnitt 4.1 wird dieser Korpus beschrieben.

Die Arbeit mit den Daten, vor allem das Training von Machine Learning-Modellen wurde mit dem öffentlich zugängliche Toolkit OpenNRE durchgeführt. OpenNRE wurde speziell für Neural Relation Extraction entwickelt, weshalb es besonders gut zu den Anforderungen dieser Arbeit passt. In Abschnitt 4.2 wird genauer auf die Funktionsweise und die Besonderheiten von OpenNRE eingegangen.

4.1. Reale Pressemitteilungen

Der Anstoß, die Problematik der themenbezogenen Entity-Relation-Extraction am Beispiel von Personen-Geschäftsbeziehungen zu erforschen, kam von einem Datensatz des Fraunhofer IAO [FKKK09]. Dieser wurde bereits in mehreren Publikationen verwendet und stets mit neuen Pressemitteilungen gefüllt. In der hier verwendeten Version sind Mitteilungen vom 4. November 2003 bis zum 3. Juli 2017 enthalten. Diese wurden aus öffentlich zugänglichen Quellen gesammelt, der daraus resultierende Korpus hat einen Umfang von 1 854 951 Datensätzen. Die Pressemitteilungen sind überwiegend deutsch, gefolgt von englischen. In Abbildung 4.1 wird die Verteilung der Sprachen dargestellt. Im Zuge dieser Arbeit lag der Fokus auf den deutschen Pressemitteilungen, da diese speziell die Analyse von Dokumenten in deutscher Sprache untersucht.

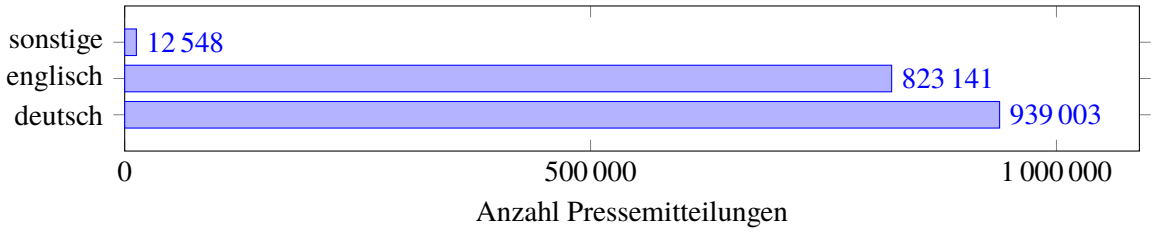

Abbildung 4.1.: Diagramm der im Fraunhofer-Korpus enthaltenden Pressemitteilungen, nach Sprache geordnet.

4.1.1. Struktur

Der Korpus bietet eine umfangreiche Kontextualisierung durch Metadaten an. Für jede Pressemitteilung können Informationen wie Quelle, Datum, Organisation etc. hinterlegt werden. Diese Metadaten sind in vielen Fällen vorhanden (Siehe Abbildung 4.2) und können eine sehr wertvolle Ressource sein.

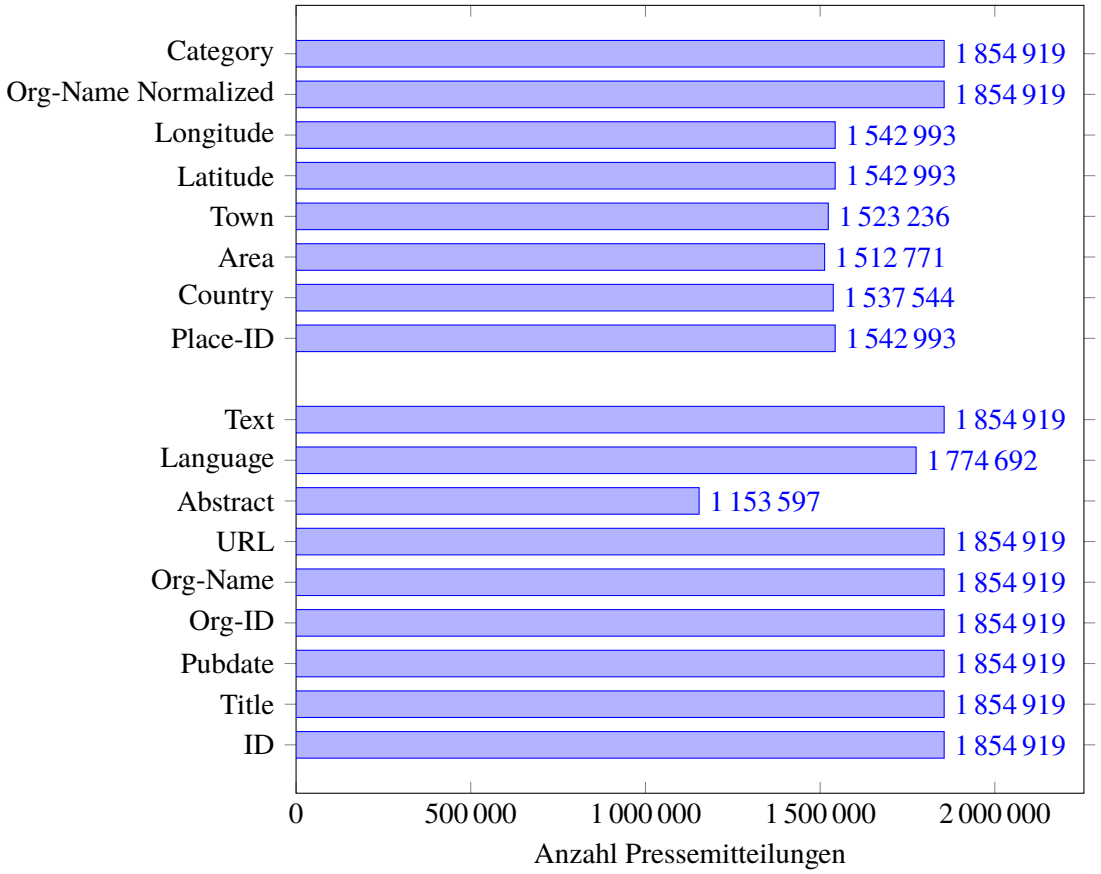

Abbildung 4.2.: Die Metadaten des Fraunhofer-Korpus mit der Anzahl der Pressemitteilungen, bei welchen der Datenpunkt hinterlegt ist. Auf 31 der 1 854 951 Pressemitteilungen kann über SQL nicht zugegriffen werden.

Um bei weiteren Arbeiten mit realen Pressemitteilungen sämtliche Informationen aus den Metadaten einzubeziehen, wurde jede Pressemitteilung in einen sogenannten „Artikel“ gebündelt. Dieser enthält alle Metadaten in einer einzigen Text-Datei, welche von Menschen lesbar ist und später auch für die Annotation verwendet werden kann. Der Artikel wird durch eine Aneinanderreihung der Metadaten erzeugt, was durch die Funktion in Listing 4.1 umgesetzt wird. Die Methode ist in Python[1] umgesetzt. Ein Beispiel für eine so formatierte Pressemitteilung ist in Listing 4.2 zu finden. Es ist zu beachten, dass alle vorhandenen Informationen hinterlegt und für die Weiterverarbeitung bereit sind.

[1] www.python.org

Um den Korpus möglichst gleichförmig und nah an natürlicher Sprache zu halten, wurden einige offensichtliche Formatierungsprobleme behoben. Diese haben ihren Ursprung vermutlich in den verschiedenen Quellen der Pressemitteilungen und den jeweiligen verschiedenen Codierungen, die von den Presseportalen genutzt wurden. Während über diese Korrekturen einige Unregelmäßigkeiten beseitigt wurden, ist es bei einem Korpus diesen Umfangs sehr aufwändig, sämtliche Daten zu bereinigen. Da dies den Rahmen der Arbeit gesprengt hätte und auch für die weiteren Schritte nicht unbedingt notwendig ist, wurden nur naheliegende Verbesserungen vorgenommen, die aus Tabelle 4.1 entnommen werden können.

```
def generateArticle(self):
    result = ""
    if self.pubdate != None: result += self.pubdate + " "
    if self.orgname != None: result += self.orgname + "\n"
    if self.publisher != None: result += self.publisher + ", "
    if self.country != None: result += self.country + ", "
    if self.area != None: result += self.area + ", "
    if self.town != None: result += self.town + ", "
    if self.language != None: result += self.language
    result += "\n"
    if self.title != None: result += self.title + "\n"
    if self.abstract != None: result += self.abstract + "\n"
    if self.text != None: result += self.text
    return result
```

Listing 4.1: Die `generateArticle`-Methode, durch die relevante Metadaten zu einer Textdatei verbunden werden. Als Input bekommt sie alle Daten und Metadaten eines SQL-Eintrags. Die Funktion ist in Python programmiert.

```
2009-05-11 14:45 TK Techniker Krankenkasse. PressePortal, DE, Hamburg, Hamburg,
 DE. Dr. Hans-Heinrich Gerth neuer Vorsitzender des TK-Verwaltungsrates.
Hamburg (ots) - Dr. Hans-Heinrich Gerth aus Meersburg ist vom Verwaltungsrat
der Techniker Krankenkasse (TK) zum alternierenden Vorsitzenden gewählt worden.
 Das Gremium hat ihn zugleich mit der Übernahme des Vorsitzes für das Jahr 2009
 beauftragt. Die Neuwahl war erforderlich geworden, nachdem Peter K. Thomsen
Ende der vergangenen Woche freiwillig auf sein Amt verzichtet hatte.
Pressekontakt: TK-Pressestelle Dorothee Meusch Tel. 040 - 6909 - 1783, Fax 040
- 6909 - 1353 E-Mail: dorothee.meusch@tk-online.de
```

Listing 4.2: Ein Artikel, der aus den Metadaten einer Pressemitteilung mit der `generateArticle`-Methode aus Listing 4.1 erstellt wurde.

4.1.2. Inhalt

Der Korpus des Fraunhofer IAO umfasst Pressemitteilungen von 24 Presseportalen [FKKK09] aus dem Zeitraum vom 4. November 2003 bis zum 3. Juli 2017. Im Rahmen dieser Arbeit sind hauptsächlich Pressemitteilungen relevant, die einen Personalwechsel verkünden und in deutscher Sprache verfasst sind. Die Methode, mit der diese Mitteilungen aus dem Korpus gefiltert werden, beruht auf einer einfachen Annahme: Handelt ein Artikel von der Einsetzung eines neuen Mitarbeiters in seine Position, so wird im Titel der Pressemitteilung immer der Name des Mitarbeiters, der Name seiner Position sowie ein Stichwort wie „neu“, „übernimmt“, „ernennt“ o. ä. in direkter Nähe zum

Vorher	Nachher	Beschreibung
`\n\n\n\n`	`\n`	Mehrere Zeilenumbrüche hintereinander werden zu einem Zeilenumbruch zusammengefasst.
`\t`	␣	Tabulatoren werden zu Leerzeichen geändert.
␣␣␣␣␣␣	␣	Mehrere Leerzeichen hintereinander werden zu einem Leerzeichen zusammengefasst.
` `		HTML-Artefakte entfernen

`&auml;`
`&ouml;`
`&uuml;`
`&quot;`
	ä
	ö
	ü
	”
		HTML-Kodierungen für Umlaute anpassen.

Ã¤ Ã¶ Ã¼	ä ö ü	Fehlinterpretationen von Umlauten anpassen.

Tabelle 4.1.: Eine Auflistung der Änderungen, die an den Pressemitteilungen des Fraunhofer-Datensatzes vorgenommen wurden.

Namen genannt. Diese Annahme ergibt sich aus der Betrachtung von zufällig gewählten Beispielen aus dem Korpus. Insbesonders die Notwendigkeit der Berücksichtigung von Stichwörtern, die einen Wechsel anzeigen, ging aus Beobachtungen des Korpus hervor. Bei Verzicht auf solche Stichwörter würden sämtliche Mitteilungen erkannt werden, bei denen von „CEO Max Mustermann“die Rede ist, unabhängig vom Kontext. Durch die Stichwörter wird der Anteil von Mitteilungen aus anderem Kontext minimiert. Die genutzten Stichworte, mit denen die Pressemitteilungen gefiltert wurden, lassen sich in Tabelle 4.2 ablesen.

Mithilfe von SpaCy[2] und der enthaltenen Named Entity Recognition (NER) lassen sich leicht Pressemitteilungen herausfiltern, bei denen eine Person, eine Position in einer Firma, sowie ein Verb, welches Personaländerung anzeigt, im Titel enthalten sind. NER erkennt die in einem Satz enthaltenen Entitäten und ordnet sie einer bestimmten Klasse zu. Diese Klassen umfassen bei SpaCy „PER“ für Personen und „ORG“ für Organisationen sowie weitere, die für diese Arbeit nicht relevant sind. Nach Angaben der Entwickler von SpaCy hat die hier verwendete, vortrainierte deutsche Pipeline (`de_core_news_lg`) für die Zuordnung von Entitäten einen F1-Score von 85%[3]. Das ist die beste deutsche Pipeline für NER, die SpaCy zur Verfügung stellt.

Auf diese Art können aus dem Korpus 3 235 deutsche Pressemitteilungen gefiltert werden, in denen eine Firma eine Neueinstellung verkündet. Dass auf diese Art eine gute Schnittmenge der tatsächlich enthaltenen Personalwechsel ergibt, lässt sich durch Stichproben überprüfen: Während nicht der Eindruck entsteht, es seien Pressemitteilungen übersehen worden, so scheint doch ca. ein Zehntel der gefilterten Nachrichten keine Neueinstellung zu verkünden. Fehlerfälle beinhalten häufig eine Produktpräsentation durch den CEO, ein Szenario, in dem sowohl Name, Position als auch ein Verb der vordefinierten Liste (Siehe Tabelle 4.2) wie z.B. „neu“ oder „vorgestellt“ vorkommen. Diese Daten eigenen sich dennoch gut für die Nutzung im Rahmen dieser Arbeit.

[2] https://spacy.io
[3] https://spacy.io/models/de

Stichwort	Abdeckung durch Stichwort
Geschäftsführ	Geschäftsführerin, Geschäftsführer, Geschäftsführung
CEO	CEO
CFO	CFO
Chef	Chefin, Chef
Leit	Leiterin, Leiter, Leitung, Abteilungsleiterin, Abteilungsleiter...
Führung	Führung
Präsident	Präsidentin, Präsident
President	President
Vorsitzende	Vorsitzende, Vorsitzender
Head of	Head of HR, Head of Engineering...
Partner	Partnerin, Partner
Intendant	Indentantin, Intendant

Stichwort
einsetzen
eingesetzt
beginnt
neu
ernennt
ernannt
benennt
benannt
übernommen
übernimmt
vorgestellt
leitet
verstärkt
verstärken

Tabelle 4.2.: Stichwörtertabellen für die Filterung von Pressemitteilungen nach Neueinstellungen. Links die Tabelle mit Bezeichnungen für die Position, die von einer Person besetzt werden kann. Sie ist als Liste in `ceo_synonyms.txt` hinterlegt. Rechts eine Tabelle mit den Verben, die andeuten, dass es sich um eine Neubesetzung der Position handelt. Diese ist unter `ceo_verbs_DE.txt` hinterlegt.

4.2. OpenNRE

OpenNRE ist ein von Han et al. entwickeltes Open-Source Toolkit, welches ein einheitliches Framework für Relation Extraction (RE) bereitstellt [HGY+19]. RE beschreibt das Fachgebiet, bei dem in natürlichem Text Fakten über Dinge (Relationen) gefunden (extrahiert) werden. OpenNRE fokussiert sich auf die Relation Classification (RC), hat also keine Methoden für Named Entity Recognition (NER), eine Interpretation der RE, die den Titel des Papers von Bassignana et al. „What Do You Mean by Relation Extraction?“ erklärt [BP22]. So geht OpenNRE davon aus, dass die Entitäten, zwischen denen eine Relation erkannt werden soll, bereits annotiert vorliegen, und nur noch die Art der Relation vorhergesagt werden muss. Dabei wird der Schritt der Relation Identification (RI) aufgeteilt, sodass Entitäten, zwischen denen offensichtlich keine Relation besteht (zum Beispiel aufgrund der Entitätstypen), im Voraus aussortiert werden. Entitäten, die keiner Relation entsprechen, aber trotzdem von OpenNRE klassifiziert werden, bekommen die Klasse „other“ zugeordnet.

OpenNRE legt den Fokus, wie der Name schon sagt, auf Neural Relation Extraction (NRE), also Relation Extraction mit Hilfe von Deep Learning. Das Ziel von OpenNRE ist es, die Fortschritte im Deep Learning weiter voranzutreiben, und als zuverlässiges Toolkit für die NRE zu dienen. OpenNRE basiert auf TensorFlow [MAP+15], einer von Google entwickelten, leistungsstarken und flexiblen ML-Bibliothek, die, wie OpenNRE auch, als Open-Source verfügbar ist.

Um die Trainingsdaten für das zugrundeliegende BERT-Modell [DCLT19] interpretierbar zu machen, nutzt OpenNRE die Lückentext-Methode. Diese wurde von Soares et al [SFLK19] vorgestellt und beinhaltet das Verdecken der Entitäten innerhalb der Beispielsätze. Das BERT-Modell sagt voraus, was in diesen Lücken stehen könnte. Da der eigentliche Inhalt bekannt ist, kann das Modell seine Vorhersagen basierend darauf verbessern. Es ist stets bekannt, welche die korrekten Wörter sind, die in diese Lücken gehören, weshalb man diese Form des Lernens auch überwacht („supervised“) nennt.

Die Leistungsfähigkeit von OpenNRE wurde von Han et al. [HGY+19] mit dem offiziellen Benchmark-Datensatz SemEval 2010 Task 8 unter Beweis gestellt, bei dem mit den gleichen Einstellungen wie bei Soares et al [SFLK19] vergleichbare Ergebnisse erzielt wurden. Auch mit dem inoffiziellen Benchmark Wiki80, welcher 80 Relationen und 56 000 Beispiele aus Wikipedia und Wikidata enthält, wurden Tests durchgeführt. Aus ihnen geht hervor, dass die BERT-Modelle bessere Ergebnisse liefern als die vorherige Generation von ML-Modellen, die sogenannten Convolutional Neural Networks (CNN).

Es werden einige verschiedene RE-Modi von OpenNRE unterstützt, darunter Sentence-Level RE, Bag-Level RE und Document-Level RE. Beim Entwickeln lag der Fokus laut der Autoren jedoch auf der Sentence-Level Relation Extraction, die auch in dieser Arbeit verwendet wird. Dabei bekommt OpenNRE ein Satz mit bereits markierten Entitäten vorgegeben, und basierend auf einem Sprachmodell, welches im Voraus finegetunt wurde, wird dann eine Relation zwischen den Entitäten vorausgesagt. Im Gegensatz dazu findet Bag-Level RE auf mehreren Sätzen statt, die alle die gleiche Relation beinhalten, und Document-Level auf ganzen Dokumenten, wo sich die Relation im Normalfall nicht aus einem einzelnen Satz ergeben würde.

Auch das Finetuning der Modelle kann von OpenNRE durchgeführt werden. Dafür kann ein beliebiges Sprachmodell zusammen mit Trainingsdaten und einigen Informationen über die in den Trainingsdaten enthaltenen Relationen verwendet werden. Tatsächlich bestehen diese Informationen, die OpenNRE über die Relationen benötigt, hauptsächlich darin, welche Relationen existieren, und mit welcher ganzzahligen Nummer sie codiert werden. Zusätzlich wird noch definiert, welche Relation als „other“ interpretiert werden soll. Diese Relation wird von OpenNRE insofern besonders behandelt, als dass sämtliche Beispiele, die nicht mit ausreichender Sicherheit einer der davor definierten Relationen zugeordnet werden können, unter diese „other“-Relation eingeordnet werden.

Beim Tuning mit OpenNRE kann auf einige Parameter Einfluss genommen werden. Am relevantesten sind hierbei Batch-Size und die Anzahl der Epochen. Die Batch-Größe beschreibt, wie viele Sätze gleichzeitig pro Iteration durch das neuronale Netz verarbeitet werden sollen.

Die Wahl der Batch-Größe kann einen Einfluss auf das Training haben. Sie bestimmt, wie viele Datenpunkte dem Modell beim Finetuning gleichzeitig zugeführt werden. Bei kleineren Batches wird das Modell häufiger aktualisiert und es findet eine schnellere Konvergenz statt. Allerdings kann dies zu längeren Trainingszeiten führen, da die Berechnungen für jeden einzelnen Batch durchgeführt werden müssen, und bei kleinen Batches dafür deren Anzahl größer ist. Eine größere Batch-Größe kann hingegen die Trainingszeit verkürzen, da die Berechnungen parallelisiert werden können. Allerdings kann dies zu einer geringeren Modellgenauigkeit führen, da das Modell möglicherweise einen geringeren Fokus auf die einzelnen Datenpunkte legt.

Die Anzahl der Epochen beschreibt, wie oft der Trainingsprozess mit dem ganzen Trainingsset wiederholt wird. Bei großen Batches kann dies die Modellgenauigkeit erhöhen, allerdings gehen viele Epochen auch mit einer hohen Laufzeit einher. Außerdem kann beim Training von ML-Modellen ein Idealpunkt existieren, ab dem weiteres Training zu Overfitting, dem Überanpassen des Modells an das Trainingsset, führt. Dadurch wird die Leistung des Modells auf dem Evaluationsdatensatz schlechter, da die Generalisierung des Modells verloren geht.

OpenNRE zeigt dem Nutzer nach jeder Epoche die Ergebnisse, die das Modell auf dem Evaluationsdatensatz erzielt an, und speichert den Zustand des Modells mit den besten Ergebnissen ab. Diese Ergebnisse sind dann auch das, was als Leistung des Modells mit Finetuning angegeben wird.

4.2.1. Datenstruktur

Damit OpenNRE die Trainings- und Evaluationsdaten auch verarbeiten kann, müssen sie in dem von OpenNRE vorgegebenen Format vorliegen. Beispiele hierfür sind auf der GitHub-Seite von OpenNRE hinterlegt[4]. Da kein anderes Tool für das Tuning von Modellen genutzt wird (Siehe Abschnitt 5.5), sind sämtliche Daten, von handannotierten über regelbasiert annotierte bis hin zu denen, die von ChatGPT generiert wurden, entweder direkt in diesem Format hinterlegt, oder in einem Nachverarbeitungsschritt in dieses Format übersetzt worden.

```
{
  'token':
    ['Mit', 'Wirkung', 'zum', '1.', 'Juni', '2009', 'wird', 'Michael',
    'Wiener', '(', '44', ')', 'zum', 'Geschäftsführer', 'Vertrieb', 'der',
    'Duales', 'System', 'Deutschland', 'GmbH', '(', 'DSD', ')', 'ernannt',
    '.'],
  'h': {
    'name': 'Michael Wiener',
    'pos': [7, 8]
    },
  't': {
    'name': 'Duales System Deutschland GmbH (DSD)',
    'pos': [16, 22]
    },
  'relation': 'neu_geschaeftsfuehrer_von'
}
```

Listing 4.3: Der Satz aus Abbildung 3.6, dargestellt in dem von OpenNRE genutzten (Pseudo-) JSON-Format.

Das Beispiel aus Abbildung 3.6 ist in Listing 4.3 im Format von OpenNRE abgebildet. In diesem Format wird der Satz in Token aufgeteilt und auch in der aufgeteilten Form hinterlegt. Die Entitäten werden im Verhältnis zur Relation als „`h`“ (Head) und „`t`“ (Tail) eingestuft, um die Direktionalität der Relation zu bewahren. Die Position dieser Entitäten im Satz wird in `pos` mit dem Start- und Endindex der betreffenden Token im Satz markiert. So können Entitäten, die aus mehreren Token bestehen, abgebildet werden. Die Relation selbst ist unter `relation` vermerkt. Durch ihre Form

[4] https://github.com/thunlp/OpenNRE

wird implizit festgelegt, dass im Head stets Personen, und im Tail stets Institutionen hinterlegt sein müssen (Eine Person kann `neu_geschaeftsfuehrer_von` einer Institution sein, aber nicht anders herum). Die Zeilenumbrüche werden für die Arbeit mit den annotierten Beispielen entfernt, und in jeder Zeile eines Dokuments kann ein Satz hinterlegt werden, weshalb es nicht den offiziellen Formatauflagen für eine JSON-Datei[5] entspricht (Bei einer korrekten JSON-Datei müssten die Sätze in einer Liste abgelegt werden, was sie hier aber nicht sind).

[5] https://www.json.org/json-de.html

5. Methodik & Umsetzung

In diesem Kapitel wird beschrieben, auf welche Art die verschiedenen Lösungen im Rahmen dieser Arbeit gefunden wurden. Dies umfasst Programmierung, die Arbeit mit ChatGPT sowie die Gestaltung des experimentellen Setups. In Abschnitt 5.1 wird beschrieben, was zu den Kriterien zu der Baseline führte und wie sie umgesetzt wurde. Anschließend geht es um die Erstellung der drei Trainingssets, beginnend mit den handannotierten Daten in Abschnitt 5.2. Im darauffolgenden Abschnitt 5.3 wird beschrieben, wie existierende Daten automatisiert annotiert werden konnten. In Abschnitt 5.4 wird erklärt, wie mit dem Einsatz von generativer KI in der Form von ChatGPT synthetische Trainingsdaten erstellt wurden. Zuletzt wird der Versuchsaufbau und die durchgeführten Experimente in Abschnitt 5.5 beschrieben.

5.1. Baseline

Eine Baseline stellt einen Ansatz dar, der auf einer bewährten Technik basiert, mit dem eine neue Methode verglichen werden kann. Anhand dieser Baseline können Stärken und Schwächen der neuen Methode festgestellt und diskutiert werden. Es ist sinnvoll, die Baseline im Kontext der Forschungsfrage zu definieren. Die Forschungshypothese dieser Arbeit lautet „Generative KI ermöglicht die Synthese bereits annotierter Trainingsdaten, die zum Finetuning von Large Language Models zur Relation Classification verwendet werden können. So können in speziellen Anwendungsfällen, in denen wenige echte Trainingsdaten zur Verfügung stehen, mit weniger Aufwand vergleichbare oder bessere Ergebnisse erzielt werden als mit herkömmlichen Methoden wie der regelbasierten Synthese von Trainingsdaten.“Es wird also untersucht, ob generative KI die Möglichkeit bietet, den Prozess der Trainingsdatenbeschaffung mit weniger Aufwand zu erledigen. Aufwand wird in diesem Kontext als Zeitaufwand interpretiert, da dieser leicht messbar und vergleichbar ist.

Um diese Forschungshypothese zu belegen, wurde folgende Begrenzung für die Baseline gewählt: Für Ihre Programmierung darf nicht mehr Zeit investiert werden, als für die Generierung der synthetischen Trainingsdaten durch ChatGPT. Diese Zeitspanne beträgt zum Zeitpunkt der Experimente zwei Tage. Die Begründung dieser Entscheidung ist in Unterabschnitt 5.4.1 nachzulesen. Somit darf also auch die Komplexität der Baseline in ihrer Programmierung einen Aufwand von zwei Tagen nicht überschreiten.

Ziel der Baseline ist es also, den Evaluationsdatensatz (Siehe Unterabschnitt 5.5.2) zu verarbeiten, und für jede Zeile anzugeben, zu welcher der 10 oder 4 Relationen sie gehört. Dazu dürfen die Entitäten `h` und `t` zwar genutzt werden, aber selbstverständlich nicht die in dem Feld `relation` hinterlegte Ground Truth, die von Hand definierte wahre Relation.

Die Baseline ist als regelbasierter Parser implementiert. Sie basiert auf Wörterbüchern, die für jeden zu erkennenden Aspekt definiert wurden und für die Stichwortprüfung in den Sätzen angewendet werden. Diese Wörterbücher sind in Tabelle 5.1 dargestellt. Bevor diese Stichwortprüfung jedoch

Wörterbuch	Inhalt
`woerter_fuer_neu`	neu, eingestellt, eingesetzt, einsetzen, ernennt, ernannt, benannt, beginnt, benennt, übernimmt, übernehmen, übernahm, begann, begonnen, benann, vorgestellt, verstärkt, verstärken
`woerter_fuer_ex`	ex, war, entlassen, entlässt, ruhestand, getrennt, hat, verlässt, verlassen, endet, abgelöst, ablösen, geleitet
`woerter_fuer_ceo`	ceo, geschäftsführer, geschäftsführerin, geschaeftsfuehrung, intendant, intendantin
`woerter_fuer_vorstand`	vorstand, vorstehende, vorsitz, vorsitzende, leitung, mitglied, präsident, präsidentin, president, admiral, minister, ministerin, sprecher, sprecherin
`woerter_fuer_abteilungsleiter`	leiter, leiterin, team, verantwortlich, vize, kapitän, kapitänin, captain, fernsehspielchef, fernsehspielchefin, fernsehdirektor, fernsehdirektorin, chefvolkswirt, chefvolkswirtin

Tabelle 5.1.: Stichwörtertabellen für die Klassifizierung von Relationen in der Baseline. Aus dem Namen der Wörterbücher geht hervor, welche Eigenschaft durch die Wörter in den Wörterbüchern erkannt werden soll. Im Fall, dass die Stichwortprüfung keine Position feststellen kann, wird auf die Klasse `sonstige` zurückgegriffen, und falls kein Tempus festgestellt werden kann auf die Zeitangabe `ist`.

durchgeführt wird, werden die annotierten Beispiele mit regulären Ausdrücken geparst, und die Bestandteile des annotierten Satzes der Baseline zugänglich gemacht. An dieser Stelle wird auch dafür gesorgt, dass die hinterlegte Ground Truth dem Programm nicht zugänglich gemacht wird.

Über die Stichwortprüfung werden dann die Sätze den Relationen zugewiesen. Sätze, die weder `ceo`, `vorstand` noch `abteilungsleiter` zugewiesen werden können, erhalten die Relation `sonstige`. Nachdem die Position abgefragt wird, wird der Tempus abgefragt. Was hier nicht zu `neu` oder zu `ex` passt, wird dem Tempus `ist` zugewiesen. Diese Entscheidung beruht auf Beobachtungen aus dem Fraunhofer-Korpus (Siehe Abschnitt 4.1).

5.2. Handannotierte Daten

Handannotierte Daten sind die wertvollsten Daten beim Finetuning von LLMs, da sie die „Ground Truth“, die als korrekt definierten Ergebnisse beinhalten. Da handannotierte Daten aber nur unter hohem Zeitaufwand zu erhalten sind, sind sie typischerweise ein Bottleneck.

5.2.1. Prozess

Um handannotierte Daten zu erlangen, wurde in dieser Arbeit ein Umweg gewählt. Da das OpenNRE-Format (Siehe Listing 4.3) aufwändige Formatierung und Redundanzen beinhaltet, bietet sich dieses Format nicht zur Annotation von Hand an. Ein einfacheres Format, wie in Listing 5.1 dargestellt,

verzichtet auf Redundanzen und benötigt auch nur ein Minimum an zusätzlicher Formatierung. Es basiert auf Dokumenten, nicht auf Sätzen, weshalb sich die Handannotationen theoretisch auch noch für das angrenzende Forschungsfeld der Document-Level RE nutzen ließen. Bei diesem einfacheren Format wird das Dokument in Anführungszeichen gestellt, im Anschluss daran sind die im Text enthaltenen Relationen aufgelistet. Die Entitäten im Text werden mit Tags (`<e1>` ... `</e1>`) markiert. Jeder Tag kann mehrmals im Text auftauchen und muss auch nicht identisch zu anderen Instanzen des gleichen Tags sein (An einer Stelle „Zweites Deutsches Fernsehen“, an einer anderen „ZDF“). Diese Tags werden von den Relationen unter dem Text nochmals referenziert, eine Relation ist hier wie folgt aufgebaut: `relationenname(head, tail)`, zum Beispiel `neu_geschaeftsfuehrer_von(e1,e2)`.

```
"<e1>Genotec</e1> gibt Führungswechsel bekannt - neuer CEO ist <e2>Roger
Hofstetter</e2> (pressebox) Allschwil/Basel, 06.01.2010, '<e3>Simon Jenny</e3>,
 langjähriger Partner und Mitbegründer des Internet Service Providers <e1>
Genotec AG</e1>, übergibt seine Position als Chief Executive Officer an COO <e2
>Roger Hofstetter</e2>' <e2>Roger Hofstetter</e2>, ebenfalls Mitbegründer der
ersten Stunde, übernimmt ab sofort die alleinige operative Leitung des
Unternehmens mit dem erklärten Ziel, die Führung auf dem Schweizer Hosting-
Markt weiter auszubauen. Die dazu erforderlichen Massnahmen wie Aufstockung des
 Personals und Ausbau der Infrastruktur wurden bereits im 2009 erfolgreich
umgesetzt. Durch die Veränderungen an der Spitze soll die nächste Runde der
Unternehmensexpansion eingeleitet werden. <e3>Simon Jenny</e3> wird weiter als
Verwaltungsratsmitglied tätig sein und sich zukünftig neuen Aufgabe zuwenden,
die hauptsächlich im strategischen Bereich positioniert sind. <e3>Simon Jenny</
e3> kommentiert: 'Künftig werde ich mich um den Auf- und Ausbau der neu durch
Zukäufe und Beteiligungen hinzugekommenen Standbeine kümmern'."
neu_ceo_von(e2,e1)
ex_ceo_von(e3,e1)
Comment:
```

Listing 5.1: Der vollständige Artikel einer Pressemitteilung, der mit der Methode aus Listing 4.1 generiert wurde, von Hand annotiert.

Dieses Format kann nun in mehreren Schritten in das OpenNRE-Format überführt werden, wie in Abbildung 5.1 grafisch dargestellt wurde. Das Skript hierzu wurde in Python[1] programmiert.

Im Schritt ① werden die durch Tags markierten Entitäten aus dem Text der Pressemitteilung extrahiert. Dazu werden Listen angelegt, welche die verschiedenen Schreibweisen der Entitäten hinterlegen („Genotec“ und „Genotec AG“).

Bei ② werden die Tags aus dem Text der Pressemitteilungen herausgefiltert, sodass der ursprüngliche Text wieder hergestellt wird. Die Position der Entitäten ist über die Buchstabenanzahl hinterlegt. Der aus mehreren Sätzen bestehende Text wird nun in einzelne Sätze zerlegt. Dazu wurde der SpaCy Sentencizer verwendet, welcher, wie auch die SpaCy NER, in der Pipeline `de_core_news_lg` enthalten ist. Der Sentencizer hat laut den Entwicklern einen F_1-Score von 95%[2]. Nun werden diese Sätze nach den Entitäten aus Relationen durchsucht. Befinden sich im gleichen Satz beide Entitäten einer Relation, kann dieser in das OpenNRE-Format übertragen werden.

[1] https://www.python.org
[2] https://spacy.io/models/de

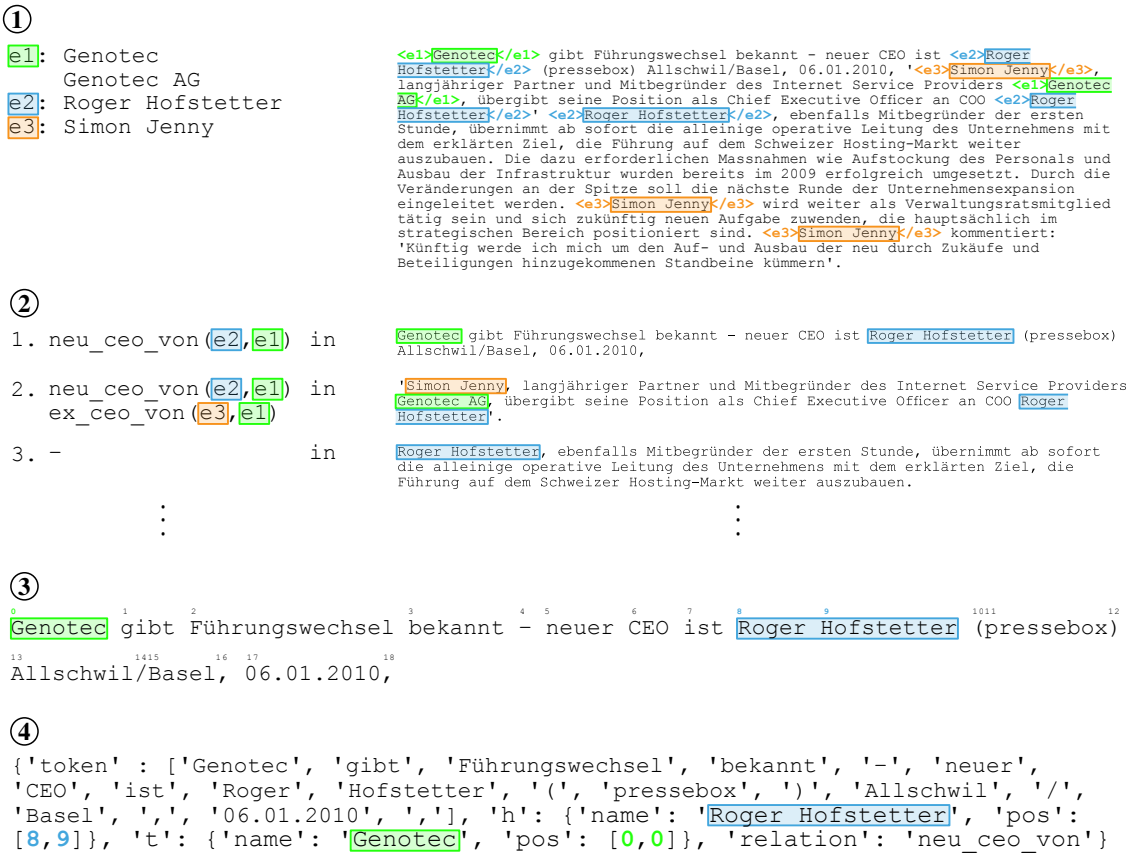

Abbildung 5.1.: Eine schematische Darstellung des Prozesses der Konvertierung vom handannotierten Format in das von OpenNRE nutzbare Format.

Dazu wird, wie in ③ dargestellt, der Satz in einzelne Token aufgeteilt. Der SpaCy Tokenizer übernimmt diese Tokenization mit sehr hoher Zuverlässigkeit (F_1-Score von 100 %). Da bekannt ist, welche Buchstaben zu welchen Entitäten gehören, kann nun jeder Entität ein oder mehrere Token zugeordnet werden.

Im letzten Schritt ④ werden Token, Entitäten samt zugehöriger Token-Positionen und die zugewiesene Relation in das Pseudo-JSON-Format von OpenNRE zusammengeführt. Durch den Aufbau der Relationen ist die Person immer die `h`-Entität, während die Institution stets in der `t`-Entität liegt. Nun können die Indizes der Entitäts-Token bestimmt und zusammen mit den Entitätsbezeichnungen in die `h` und `t`-Einträge übernommen werden.

5.2.2. Inhalt

Es wurden für diese Arbeit 424 Sätze aus 97 Pressemitteilungen von Hand annotiert. Alle Sätze stammen aus dem Fraunhofer Pressemitteilungs-Korpus (Siehe 4.1), wurden jedoch mit dem Ziel gewählt, eine möglichst ausgeglichene Verteilung der Datenpunkte auf die Relationen zu erreichen.Wie in Abbildung 5.2 dargestellt, enthält jede Relation eine zweistellige Anzahl Datenpunkte mit der Ausnahme von `ist_abteilungsleiter_von`. Dass diese Relation weniger Datenpunkte enthält,

liegt an der Natur von Pressemitteilungen: Während existierende CEOs und Vorstände häufig im Bezug auf neue Entwicklungen und Personalwechsel zitiert werden, wird kaum ein Abteilungsleiter befragt. Somit kommen die Relationen `ist_ceo_von` und `ist_vorstand_von` auf ein Vielfaches an Datenpunkten.

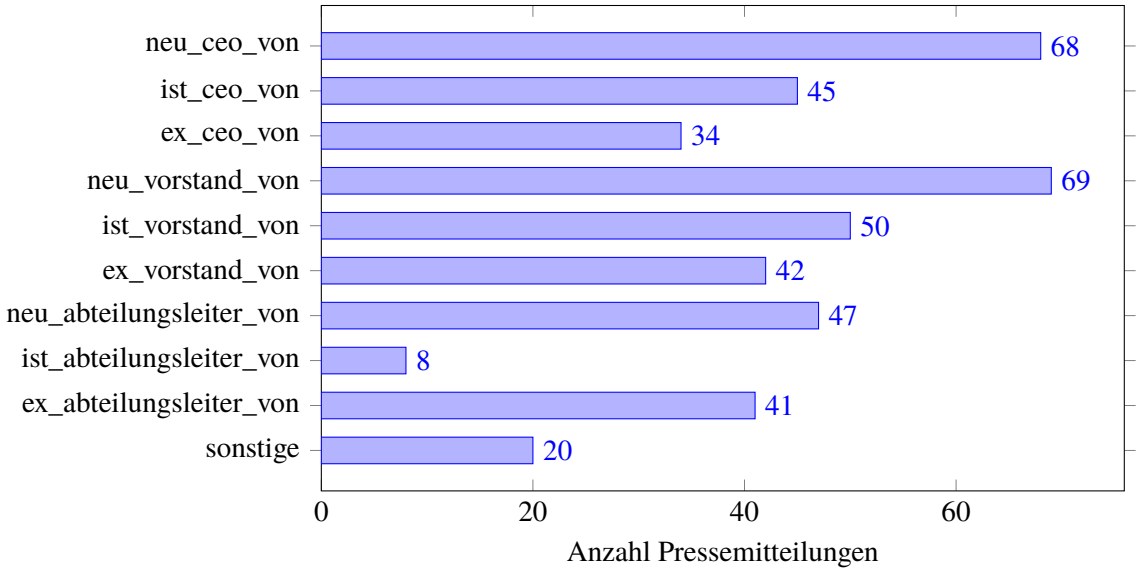

Abbildung 5.2.: Die Verteilung der 424 Datenpunkte im handannotierten Korpus auf die 10 Relationen.

Da die Relation `sonstige` ein sehr großes Feld umfasst, wurde bei den Handannotationen kein großer Fokus darauf gelegt, dieses Feld vollständig abzudecken. Stattdessen soll diese Klasse hauptsächlich durch eine sehr niedrige Konfidenz in alle anderen Relationsklassen von OpenNRE erschlossen werden, wie in Abschnitt 4.2 beschrieben.

5.3. Regelbasiert annotierte Daten

Regelbasiert annotierte Daten können hilfreich sein, wenn für das Trainieren oder Finetunen eines Modells nicht genügend Daten vorliegen. Die Qualität dieser regelbasiert annotierten Daten kann zwar üblicherweise nicht mit handannotierten Daten mithalten, aber dafür stehen sie in weit größerer Quantität zur Verfügung. Dennoch ist ein optimierter Prozess erforderlich, der möglichst hochqualitative Daten liefern kann. Um einen solchen Prozess einzurichten, sind umfangreiche Programmierarbeiten sowie tiefe Einblicke in die Problematik notwendig.

5.3.1. Prozess

Die regelbasierte Erstellung von Trainingsdaten wird in Abbildung 5.3 schematisch dargestellt. Ausgangspunkt für die regelbasierte Annotation sind die 3 235 realen Pressemitteilungen, die, wie in Unterabschnitt 4.1.2 beschrieben, aus den deutschen Nachrichten des Fraunhofer-Korpus gefiltert wurden (Siehe ①) und vorwiegend Personalwechsel beinhalten.

Von diesen Pressemitteilungen wird nun auf den Artikel (`article`) zugegriffen, welcher alle relevanten Daten der Mitteilung enthält (Siehe Listing 4.1). Dieser wird vom SpaCy Sentencizer in einzelne Sätze geteilt (Siehe ②).

In jedem Satz wird nun nach einer Relation gesucht (Siehe ③). Dafür wird in einem ersten Schritt mittels SpaCy NER nach Named Entities der Kategorie Person (`PER`) und Organisation (`ORG`) gesucht. Beinhaltet ein Satz sowohl eine Person als auch eine Organisation, so wird im nächsten Schritt nach einer Position gesucht, die diese Person in der Firma einnimmt. Falls mehrere Personen oder Organisationen gefunden werden, entscheidet der Abstand zwischen ihnen, ob eine Relation zwischen ihnen angenommen wird. Wenn zwischen einer Person und einer Organisation eine Relation angenommen wird, werden sie dem nächsten Schritt übergeben. Um nun die Position der Person innerhalb der Firma in die definierten Relationen einzuordnen, wurde eine Prüfung auf Stichwörter aus der Tabelle 5.2 durchgeführt. In Fällen, wo kein Stichwort im Satz gefunden werden konnte, wurde auf die Relation `sonstige` zurückgegriffen. Wenn die Stichwortprüfung jedoch erfolgreich verlief, wurde in einem weiteren Schritt noch die Art des Personalwechsels festgestellt. Auch hierzu wurde wieder eine Stichwortprüfung verwendet, um festzustellen, ob die Person in ihrer Position neu beginnt (`neu`), die Position verlässt (`ex`) oder die Person nach wie vor die Position innehat (`ist`). Falls die Stichwortprüfung keine Treffer liefert, wird hier auf die `ist`-Relation zurückgegriffen, da die Daten gezeigt haben dass das häufig bei Zitaten von Führungskräften der Fall ist, die sich z.B. zu einem Personalwechsel äußern.

Um die erkannten Relationen nun in das von OpenNRE interpretierbare Format zu bringen, wird analog vorgegangen wie bei der Konvertierung in das OpenNRE-Format. Erst wird der Satz vom SpaCy Tokenizer in seine Token geteilt, dann werden die Indizes der früher erkannten Entitäten Person und Organisation ermittelt (Siehe ④). Im letzten Schritt wird dann der Satz, gemeinsam mit den Annotationen in ein JSON-Objekt eingepflegt (Siehe ⑤) und einer Pseudo-JSON-Datei mit den anderen erkannten Relationen gesammelt.

Kategorie	**Indikatoren**
`ceo`	CEO, Geschäftsführ, Intendant
`vorstand`	Vorstand, Präsident, President, Admiral, Minister, Vorsitz, Sprecher
`abteilungs-leiter`	Leiter, General, Vize, Kapitän, Captain, Chefredakteur, Fernsehspielchef, Fernsehdirektor, Chefsvolkswirt

Kategorie	**Indikatoren**
`neu`	einsetzen, eingesetzt, beginnt, neu, ernennt, ernannt, benennt, benannt, übernommen, übernimmt, vorgestellt, verstärkt, verstärken
`ist`	leitet, derzeitig
`ex`	war, abgelöst, ablösen, Ruhestand, hat, geleitet

Tabelle 5.2.: Stichwörtertabellen für die Zuordnung zu den verwendeten Relationen. Links eine Tabelle mit Positionsbeschreibungen, rechts eine Tabelle, die festlegt, ob die Position neu besetzt wird, gleichbleibend ist oder verlassen wird. Wie auch in Tabelle 4.2 wurden generische Stichwörter verwendet, um mit einem Stichwort mehrere Varianten abzudecken („Geschäftsführ“ erkennt sowohl „Geschäftsführerin“, „Geschäftsführer“ als auch „Geschäftsführung“).

Abbildung 5.3.: Eine schematische Darstellung des Prozesses zur regelbasierten Generierung von Trainingsdaten.

5.3.2. Inhalt

Die 8 567 extrahierten und annotierten Sätze sind recht gleichmäßig auf die verschiedenen Relationen verteilt, wie in Abbildung 5.4 erkennbar. Deutliche Abweichungen von den Mengenverhältnissen aus den handannotierten Daten (Siehe Abbildung 5.2) sind hauptsächlich in den Klassen `ist_abteilungsleiter_von` und `sonstige` sichtbar.

Erklären lässt sich dies über das Zurückgreifen auf die Teilrelation `ist`, wenn die Stichwortprüfung der Zeitkomponente im Falle der Relation `ist_abeteilungsleiter_von`, keine Treffer erzielt. Das geschieht, da im Kontext der Teamleitung häufig andere Stichwörter verwendet werden als in dem Stichwortkatalog in Tabelle 5.2 hinterlegt. Bei der Relation `sonstige` wurde die regelbasierte Relationszuweisung absichtlich so gestaltet, dass eine möglichst große Menge an Beispielen zustande kommt. Da Personen und Institutionen in offenem Kontext häufig im gleichen Satz vorkommen, ohne dass die Person für die Institution arbeitet, kann es wichtig sein, ein Relation Classification

Modell auch mit möglichst vielfältigen Gegenbeispielen zu trainieren. Ansonsten ist es möglich, dass das Modell zu viele Relationen zwischen Personen und Institutionen in eine der vordefinierten Beschäftigungsverhältnisse einordnet.

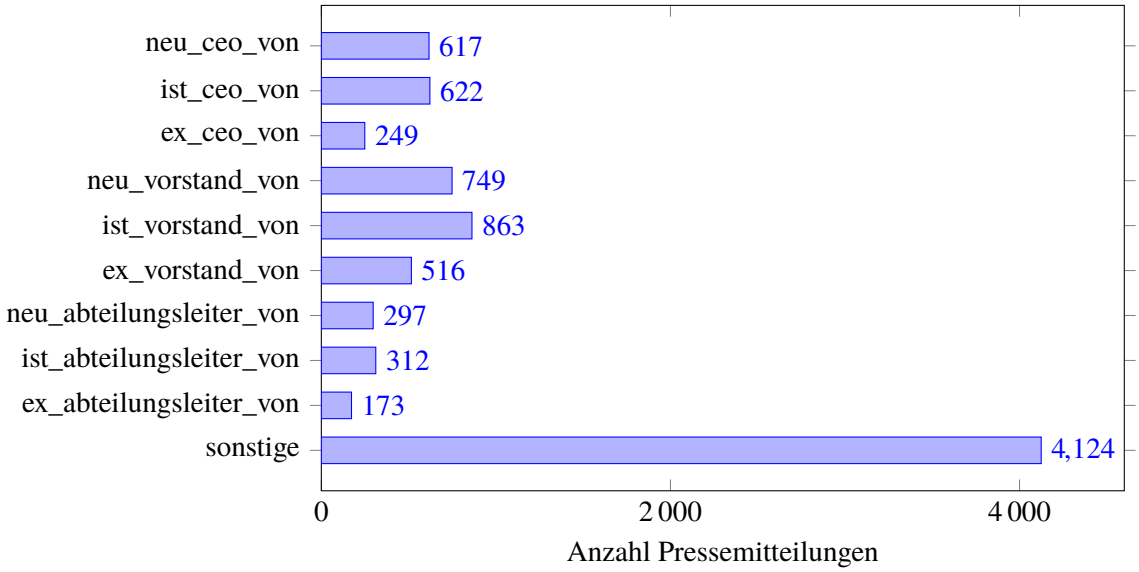

Abbildung 5.4.: Die Verteilung der 8 567 Daten im regelbasiert annotierten Korpus auf die 10 Relationen.

Bei dem Erstellen dieses Datensatzes wurde berücksichtigt, dass bei den verwendeten Sätzen keine Schnittmenge mit den handannotierten Daten existieren darf, da diese später für die Evaluation verwendet werden. Sollte ein Modell nämlich mit Daten trainiert werden, mit denen es auch später evaluiert wird, verzerrt dies stark das Ergebnis. Es wäre keine Verallgemeinerung oder Erlernen von Features notwendig, um diese Evaluationsdaten korrekt zuzuordnen. Somit ließe sich auch nicht die Qualität der erlernten Features oder Verallgemeinerungen des Modells bewerten.

5.4. Von generativer KI (ChatGPT) generierte Daten

Während der Experimente mit ChatGPT im Rahmen dieser Arbeit ist das Modell GPT-3.5 öffentlich zugänglich. ChatGPT Plus, ein Service für $ 20 im Monat[3] bietet Zugang zu GPT-4, einem Modell, welches die Qualität der Antworten auf jede messbare Art gegenüber GPT-3.5 verbessert [Ope23]. Dieses Abonnement umfasst allerdings keinen Zugriff auf die ChatGPT API. Es existiert die Möglichkeit sich auf eine Warteliste für einen API-Zugang für ChatGPT setzen zu lassen, die Wartezeit, bis dieser gewährt wird, hat allerdings den zeitlichen Rahmen dieser Arbeit überschritten.

[3] https://openai.com/product/gpt-4

5.4.1. Hindernisse

Zum Zeitpunkt der Experimente existieren einige Probleme, die die Arbeit mit ChatGPT erschweren. Lösungen für diese Probleme sind jedoch leicht vorstellbar und teilweise bereits von OpenAI angekündigt.

Geschwindigkeit

Wenn man Antworten mit dem überlegenen GPT-4 generieren lassen will, braucht das Modell für einen Output von 1 800 Zeichen, was ungefähr 4.5 annotierten Beispielen entspricht, etwa 100 s. Mit dieser Beschränkung allein benötigt die Generierung von 160 Beispieldaten eine Stunde.

Das GPT-3.5 Modell produziert Output des gleichen Umfangs jedoch in 10 s, sofern man über einen ChatGPT Plus Account zugreift. Basierend auf der Geschwindigkeit, mit der OpenAI diese effizientere und schnellere Version von GPT-3.5, „`chatgpt-3.5-turbo`“[4] verfügbar gemacht hat (ChatGPT mit GPT-3.5 wurde am 30. November 2022 öffentlich zugänglich gemacht[5], GPT-3.5 Turbo am 1. März 2023[6]). Daher lässt sich annehmen, dass auch von GPT-4 bald eine schnellere Version zugänglich sein wird.

Begrenzte Outputmenge & Formatierung

In der aktuellen Implementierung bricht ChatGPT nach den oben genannten 1 800 Zeichen den Output ab. Um ihn fortzusetzen muss der Nutzer ChatGPT über eine Eingabe auffordern, den Output weiterzuführen („weiter“, „continue“ oder ähnliches). Das erfordert die Anwesenheit und alle 100 s die Interaktion des Nutzers, während die Daten generiert werden.

Während der ChatGPT den Output meist da fortsetzt wo er unterbrochen wurde muss doch überprüft werden, ob angefangene Klammern oder Anführungszeichen tatsächlich in der Fortsetzung ein Gegenstück erhalten haben. Da die ChatGPT-Daten ohnehin über verschiedene Formatierungen verfügen (Mal mit Überschriften, mal nummeriert) ist eine simple Bereinigung der Daten von Hand ohnehin notwendig.

Bisher hat OpenAI für ChatGPT nur die Kontextlänge, also den möglichen Input, erweitert, nicht die Länge des Outputs. Dass in Zukunft, mit ausgebauter Serverkapazität und effizienteren Modellen auch Output von größerem Umfang generiert werden kann, ist dennoch nicht unwahrscheinlich, und über die API bereits möglich.

[4] https://platform.openai.com/docs/models/gpt-3-5
[5] https://openai.com/blog/chatgpt
[6] https://openai.com/blog/introducing-chatgpt-and-whisper-apis

Begrenzte Anfragen

Die hohe Nachfrage nach den ChatGPT-Services führt nicht nur dazu, dass regelmäßig der kostenlose Zugriff auf GPT-3.5 ausgesetzt wird, sondern auch zu einer Begrenzung der Anfragen, die man mit ChatGPT Plus an GPT-4 stellen darf. Zum Zeitpunkt der Experimente sind das 25 Anfragen pro Tag. Sofern man keine anderen Anfragen an das Modell stellt, entspricht das etwa 112 annotierten Sätzen.

Diese Limitation wurde seit den Experimenten bereits gelockert, und es ist naheliegend, dass sie auch eines Tages aufgehoben wird. Wann das soweit sein wird ist allerdings im Moment unklar.

Ergebnisse sind nicht reproduzierbar

ChatGPT ist nicht-deterministisch, das heißt bei gleichem Input kann verschiedener Output generiert werden. Dies erschwert die Arbeit mit ChatGPT, da Prompts, die ursprünglich zu den erwünschten Ergebnissen geführt haben, bei einem weiteren Versuch möglicherweise unerwünschte Ergebnisse produzieren.

Der Parameter, der die Varianz der Ergebnisse beeinflusst ist die sogenannte „Temperatur“(`temperature`). Je höher ihr Wert, desto stärker weichen die Ergebnisse bei gleicher Eingabe voneinander ab. Der `temperature`-Parameter kann Werte zwischen 0 und 2 annehmen, der Standard liegt bei 1. Die Ergebnisse bei einem Wert von 0 sind fokussiert und nahezu deterministisch, während am anderen Ende des Spektrums mit einem Wert von 2 zufälligere Ergebnisse generiert werden[7].

Eine Alternative zur Temperatur ist die Beeinflussung der Ergebnisvielfalt durch sogenanntes „nucleus sampling“. Über den `top_p`-Parameter wird hierfür angepasst, wieviel Prozent der wahrscheinlichsten Token für die Generierung berücksichtigt werden sollen. Ein Wert von 0,1 würde also bedeuten, dass nur die Top 10 % der wahrscheinlichsten Token berücksichtigt werden. Der Standard liegt auch hier bei 1, es werden also alle Tokens der Wahrscheinlichkeitsverteilung berücksichtigt.

Leider hat man nur über die API die Möglichkeit, diese Parameter zu beeinflussen, und wie bereits erwähnt hat zum Zeitpunkt der Experimente nicht jeder Zugriff auf diese API. Außerdem entstehen bei der Nutzung der API Kosten von $ 0,03 pro 1 000 Input-Tokens und $ 0,06 pro 1 000 Output-Tokens.

Es ist aber wichtig zu erwähnen, dass die API viele der bisher genannten Hindernisse beseitigen oder den Umgang mit ihnen erleichtern würde. So wäre es kein Problem, wenn die Generierung von annotierten Beispielen über die API mehrere Stunden in Anspruch nähme, sofern das ohne menschliche Mitwirkung abläuft. Für die API sind auch die Limits deutlich höher was den Umfang des Outputs angeht. Für GPT-4 sind hier 8 192 Tokens angegeben[8], das entspricht ungefähr 6 000 Wörtern. Auch die Begrenzung der Anfragen pro Tag auf 25 Stück existiert für die API nicht.

Die API ist also sehr vielversprechend, auch wenn im Rahmen dieser Arbeit starke Begrenzungen bei der Nutzung von ChatGPT existierten.

[7] https://platform.openai.com/docs/api-reference/completions/create
[8] https://platform.openai.com/docs/models/gpt-4

Fehlerhafte Ergebnisse

ChatGPT ist zwar in der Lage, fehlerfreie annotierte Daten zu erstellen, leider gelingt dies jedoch nicht immer. Da die Ergebnisse nicht-deterministisch sind, jede Antwort also anders ist, muss mit jeder Art von Fehler gerechnet werden. Einige Fehler treten jedoch häufiger auf als andere. Die beiden Spitzenreiter sind hierbei falsche Indizes oder Entitäten, die anders heißen als in den Token/im Satz (Siehe Listing 5.2). Falsche Indizes sind alles, wo der Startindex nicht genau den ersten Token der Entity beschreibt und der Endindex nicht genau den Letzten. Somit sind auch Indizes, die zwar die richtige Stelle im Satz beschreiben, aber zu viele oder zu wenige Folgetoken beinhalten falsch, von Indizes die nicht die richtige Stelle im Satz beschreiben ganz zu schweigen. Wenn die Entitäten anders benannt sind als jene, die im Text genannt sind, ist der Unterschied meistens nicht groß. Häufig unterscheiden sich die Namen durch grammatikalische Verschiedenheiten, oder unterschiedliche Satzzeichen, wie im Beispiel dargestellt. Auch dafür ist ein Beispiel in Listing 5.2 dargestellt.

Dass GPT-Modelle Probleme mit Indizes haben ist nicht verwunderlich, da sie keinerlei mathematisches Verständnis besitzen, weshalb keine Form von Rechnung möglich sein sollte. Korrekte Antworten sind in dem Kontext eher als eine Schätzung, basierend auf den verwendeten Wörtern und der Länge der Entitäten sowie des Satzes, als als tatsächliche Berechnung zu verstehen. Dennoch existieren große Verbesserungen zwischen GPT-3.5 und GPT-4, wie aus Abbildung 5.5 hervorgeht. Diese Entwicklung lässt auf weitere Verbesserungen in der Zukunft hoffen, wobei der Mangel an tatsächlicher Berechnung und Interpretation der Indizes bei LLMs wohl immer zu Fehlern führen wird.

Lightman et al. von OpenAI haben außerdem am 31. Mai 2023 ein Paper veröffentlicht, in welchem sie auf die Chancen von prozess-überwachtem Lernen gegenüber dem bisherigen ergebnisüberwachtem Lernen hinweisen [LKB+23]. Dies zeigt deutliche Vorteile beim Beantworten von mathematischen und logischen Problemen, für die von ChatGPT mehrere Lösungsschritte gemacht werden müssen bis eine Antwort vorliegt. Es ist laut den Autoren noch unklar, wie weitreichend sich diese Änderungen auf die verschiedenen Anwendungsfelder von ChatGPT auswirken, dennoch ist es nicht ausgeschlossen, dass die korrekte Interpretation und Bestimmung von Indizes davon profitiert. Die Forschung in diesem Bereich lässt vermuten, dass GPT in Zukunft auch bei logischen und mathematischen Problemen immer zuverlässiger antworten wird und Lösungen gefunden werden, die architekturbedingten Schwächen auszumerzen.

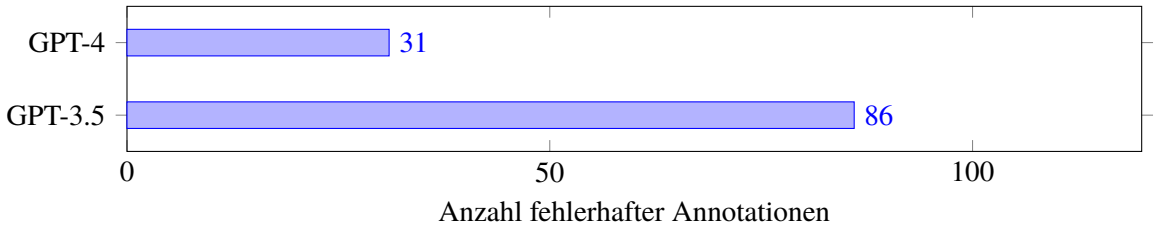

Abbildung 5.5.: Von ChatGPT fehlerhaft generierte Daten pro 110 Stück; beide Modelle wurden mit dem gleichen Prompt (Siehe Listing 5.3) ausgeführt.

5.4.2. Fehlerkorrektur

```
{
  'token':
    ['Mit', 'Wirkung', 'zum', '1. Juni 2009', 'wird', 'Michael',
    'Wiener', '(', '44', ')', 'zum', 'Geschäftsführer', 'Vertrieb', 'der',
    'Duales', 'System', 'Deutschland', 'GmbH', '(', 'DSD', ')', 'ernannt',
    '.'],
  'h': {
    'name': 'Michael Wiener',
    'pos': [4, 4]
    },
  't': {
    'name': 'Duales System Deutschland GmbH - DSD',
    'pos': [16, 22]
    },
  'relation': 'neu_vizevorstand_von'
}
```

Listing 5.2: Ein konstruiertes Beispiel einiger möglicher Fehler, die in einem generierten, annotierten Dokument enthalten sein können. Von oben nach unten: Falsche Tokenization, falsche Indizes, eine Entity, deren Name nicht mit den Token übereinstimmt, und eine Relation, die nicht vorgegeben war.

Bei der Generierung von Trainingsdaten mit ChatGPT werden auch fehlerhafte Daten erzeugt. Die möglichen Fehler sind vielseitig, in Listing 5.2 werden einige davon beispielhaft dargestellt. Durch Fehler verunreinigte Daten führen zu schlechteren Ergebnissen als bereinigte Daten [Kav09], auch wenn dieser Effekt erst bei einer ausreichenden Menge an Datenpunkten deutlich wird. Bei Tests mit den Daten dieser Arbeit (Siehe Abbildung 5.6) mit 10 Relationen tritt dieser Effekt nicht auf: Die korrigierten Daten erzielen einen F_1-Score, welcher mit 41.67 % über 3 %P (Prozentpunkte) unter dem der fehlerbehafteten, unbearbeiteten Daten liegt. Es ist daher naheliegend, dass die für die Tests verwendeten 220 Datenpunkte nicht ausreichen, um aussagekräftige Ergebnisse bei der Klassifikation von 10 Relationen zu erlangen. Daher entspricht auch das Verhalten bezüglich der fehlerbehafteten Daten bei 10 Relationen nicht den Erwartungen. Dass der F_1-Score der korrigierten Trainingsdaten, die auf 4 Relationen angewendet wurden höher ist als derjenige, der fehlerhaften Trainingsdaten entspricht den Beobachtungen von Kavzoglu [Kav09]. Bei 59 fehlerhaften Datenpunkten in dem 220 Datenpunkte umfassenden Korpus könnte die vergleichsweise geringe Differenz von 1.05 %P ebenfalls durch die geringe Menge der Datenpunkte beeinflusst sein und bei größeren Datenmengen höher ausfallen.

Sobald größere Datenmengen generiert werden, ist eine Korrektur der Daten erstrebenswert. Um abschätzen zu können, wie viel Aufwand die Implementierung eines Korrekturskripts mit sich bringt, und auch das volle Potenzial automatisch generierter Trainingsdaten zu erfassen, wurde im Rahmen dieser Arbeit ein Korrekturskript implementiert. Die zukünftigen Experimente dieser Arbeit finden auf den korrigierten Versionen der durch ChatGPT generierten Daten statt.

Ziel des Korrekturskripts ist es, die Fehler in den Daten nach Möglichkeit zu korrigieren, oder Beispiele, bei denen das nicht möglich ist aus dem Datensatz zu entfernen. Auch handannotierte und regelbasiert annotierte Daten wurden mit diesem Skript geprüft, um sicherzustellen, dass beim Training keine Fehler auftreten.

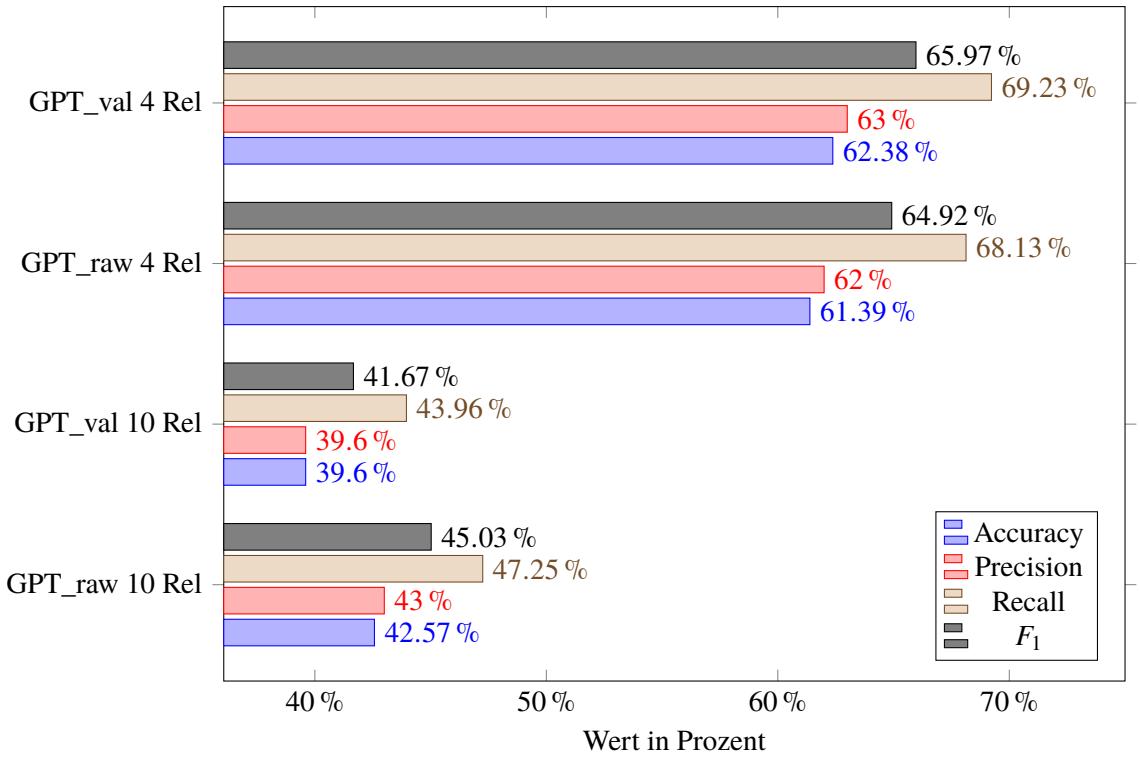

Abbildung 5.6.: Vergleich von fehlerbehafteten, unbearbeiteten ChatGPT-Daten (GPT_raw) mit korrigierten und validierten ChatGPT-Daten (GPT_val).

Das im Appendix Listing A.2 hinterlegte Korrekturskript prüft in mehreren Schritten, ob die vorgelegten Daten korrekt sind und wird, wenn nicht, versuchen sie zu reparieren. Dazu wird über einen regulären Ausdruck geprüft, ob das Format korrekt umgesetzt wurde, also alle Felder vorhanden sind und auch die Struktur stimmt (Klammern, Kommas und Anführungszeichen). Sollten hier Fehler auftreten ist eine manuelle Korrektur notwendig. Eine automatische Korrektur wäre über den Rahmen dieser Arbeit hinausgegangen. Dazu kann aus dem Konsolenoutput entnommen werden, in welcher Zeile sich der fehlerhafte Satz befindet. In einem Texteditor kann dieser dann korrigiert und das Skript im Anschluss erneut ausgeführt werden. Diese Form von Syntaxfehler ist in den Tests jedoch äußerst selten vorgekommen, weshalb die aufwändige, manuelle Korrektur kein großes Hindernis darstellt.

Im nächsten Schritt werden die Token extrahiert, und jedes Token an eventuell enthaltenen Leerzeichen getrennt. Dadurch wird das Problem der falschen Tokenization behoben, das zu den häufigeren Problemen bei den von ChatGPT synthetisierten Daten zählt. Dass durch diese Erhöhung der Tokenanzahl die späteren Indizes verändert werden, wird im nächsten Schritt adressiert.

Das Prüfen von Name und Indizes der Entitäten ist aufgrund der vielen möglichen Fehler nicht trivial: Bei einem Mismatch können Index oder Entitäts-Name fehlerhaft sein, es kann auch beides fehlerhaft sein. Da bei fehlerhaften Entitäts-Namen im Allgemeinen nur kleine Abweichungen zwischen dem im Text erwähnten und in der Entität genannten Namen vorkommen, lässt das die Annahme zu, dass beide maximal eine Levenshtein-Entfernung [Lev+66] von 2 zueinander haben. Sofern also das Token, welches am genannten Index im Satz eine Levenshtein-Distanz größer 2 zu dem ersten Token im Namen der Entität hat, wird in zunehmendem Abstand vor und nach dem

Ausgangspunkt nach einem Token mit der passenden Levenshtein-Distanz gesucht. Wenn nun das richtige Token gefunden wurde, kann der Rest des Entitäts-Namens einfach mit den folgenden Token (ebenfalls mit Levenshtein-Entfernung) abgeglichen werden. Da in dem von OpenNRE genutzten Format nicht vorgesehen ist, dass die Namen der Entitäten von den im Text genannten Namen abweichen, müssen diese angepasst werden, falls ein Match über die Levenshtein-Distanz zustande kam. Dazu werden die identifizierten Token aus dem Satz konkateniert, und als neuer Name der Entität eingesetzt.

5.4.3. Prompts

Für das Engineering des Prompts sind drei Bestandteile von Bedeutung:

1. Relationen, für die annotierte, synthetische Trainingsdaten erstellt werden sollen
2. Format, in der die Annotation stattfinden soll
3. Menge an Daten pro Relation

Diese Bestandteile müssen auf die aktuellen Fähigkeiten von ChatGPT angepasst werden. Durch das nicht-deterministische Verhalten ChatGPTs können auch innerhalb eines Modells große Unterschiede beim Output auftreten. Im Rahmen dieser Arbeit verfestigte sich jedoch der Eindruck, dass Prompts, welche mit wenigen Tagen Abstand zueinander ausgeführt werden, auch ähnliche Ergebnisse erzielen.

Relationen

Damit die von ChatGPT generierten Textbeispiele auch eine möglichst große Bandbreite an zugrundeliegenden Untergruppen abdeckt, werden synthetische Daten für 22 Relationsarten generiert. Diese setzen sich aus den Klassen „CEO“, „Geschäftsführer“, „Intendant“, „Vorstand“, „Vorsitz“, „Leiter“und „Abteilungsleiter“zusammen, die alle jeweils noch in den drei Varianten „neu“, „ist“ und „ex“ ausgeführt wurden, sowie der Relation „Sonstige“. Die Auswahl dieser Relationen als Basis für die Generierung von Trainingsdaten wurde aufgrund der Pressemitteilungen aus dem Fraunhofer-Datensatz getroffen. Wie diese Relationen dann zu den verwendeten, gröberen Klassen zusammengefasst wurden wird in Abbildung 5.7 dargestellt.

Format

Damit ChatGPT die annotierten Daten auch in dem gewünschten Format wiedergibt, muss das Prompt gute Beispiele enthalten. Der naheliegende Ansatz, ChatGPT mit so vielen handannotierten Daten wie möglich (100 annotierte Sätze) zu prompten, führt zu schlechten Ergebnissen. Das Problem hierbei ist nicht unbedingt die maximale Inputlänge, die hierdurch nicht erreicht wird, sondern ChatGPTs Fähigkeit, den Input dann noch wie gewünscht zu interpretieren: Die Informationen über Relationen und die Menge der gewünschten synthetischen Daten werden so nicht mehr berücksichtigt. Mehrere Tests und die stichprobenhafte Bewertung der Ergebnisse haben gezeigt, dass wenige ausgewählte Beispiele zu besseren Ergebnissen führen.

Dabei ist es wichtig, die folgenden Fälle bezüglich der Indizes abzudecken:

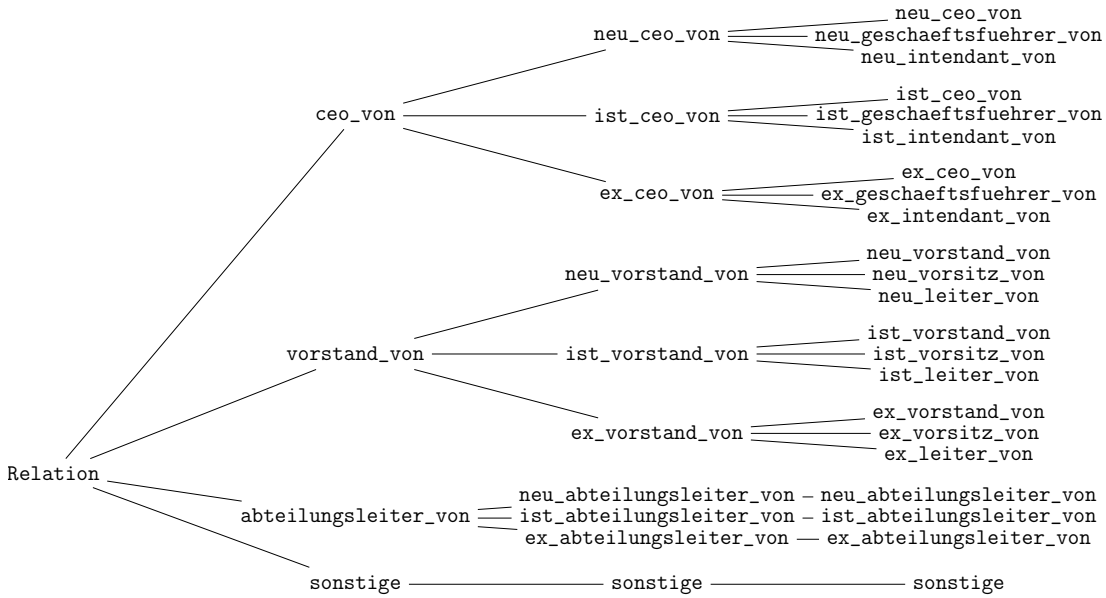

Abbildung 5.7.: Um ChatGPT eine weitere Bandbreite an Datenpunkten abzugewinnen, wurden erst Beispiele für feingranularere 22 Relationen erstellt. Diese wurden dann zu den Überrelationen zusammengefasst.

1. Der Startindex eines Wortes beträgt Null. Fehlt ein solcher Fall, ist es wahrscheinlich dass ChatGPT auch in Fällen, wo als Startindex Null korrekt wäre, einen Index von 1 angibt.
2. Eine Entität, die sich über mehr als zwei Token erstreckt. Fehlt ein solches Beispiel, ist es möglich dass ChatGPT standardmäßig Start- und Endindex entweder identisch oder mit einem Unterschied von 1 gestaltet, unabhängig von der Anzahl der Token in der Entität.
3. Eine Entität, die sich über nur ein Token erstreckt, aus dem selben Grund wie oben.

Zusammenfassend lässt sich sagen, dass am Besten jeder Fall, der sich in dem Ziel-Datenformat ereignen kann, abgedeckt werden sollte. Es darf nicht davon ausgegangen werden, dass ChatGPT erfolgreich verallgemeinert, wie das Format funktioniert.

Menge Daten pro Relation

Es ist zum Zeitpunkt der Experimente nicht möglich, mithilfe eines einzigen Promptes beliebige Mengen an Trainingsdaten zu erstellen. Sobald die Anzahl der Menge an Daten pro Relation fünf übersteigt, verliert ChatGPT (wie schon, wenn zu viele Beispiele vorgelegt wurden) Teilkonzepte wie die vorgegebenen Relationen, und auch die gewünschte Menge an Datenpunkten wird nicht mehr befolgt.

Deshalb wurden für die Experimente dieser Arbeit die synthetischen Daten mit ChatGPT durch die mehrfache Anwendung eines Prompts generiert, welches fünf Datenpunkte pro Relation erzeugt.

Das komplette Prompt

Die oben genannten Berücksichtigungen werden alle in einem Prompt umgesetzt. Es ist bei ChatGPT zwar explizit nicht notwendig, die komplette Anfrage in einem Prompt zu formulieren und oftmals können im Dialog mit ChatGPT auch Fehler, die durch ein unvollständiges Prompt entstehen, behoben werden. Das begrenzt die Reproduzierbarkeit der Ergebnisse allerdings stark, da durch jede nicht-deterministische Antwort ChatGPTs mehr Abweichungen auftauchen. Um diesen Einfluss zu minimieren, wird diese ebenfalls zielführende Methode der Generierung von Trainingsdaten in dieser Arbeit nicht genutzt.

```
Es gibt folgende Relationen zwischen einer Person und einem Unternehmen/
Institution:
neu_ceo_von
ist_ceo_von
ex_ceo_von
neu_geschaeftsfuehrer_von
ist_geschaeftsfuehrer_von
ex_geschaeftsfuehrer_von
neu_intendant_von
ist_intendant_von
ex_intendant_von
neu_vorstand_von
ist_vorstand_von
ex_vorstand_von
neu_vorsitz_von
ist_vorsitz_von
ex_vorsitz_von
neu_leiter_von
ist_leiter_von
ex_leiter_von
neu_abteilungsleiter_von
ist_abteilungsleiter_von
ex_abteilungsleiter_von
sonstige
Erstelle annotierte Beispielsätze zu diesen Relationen. Diese Beispielsätze
sollten folgende Formatierung haben:
{'token': ['Thomas', 'Müller', 'ist', 'ein', 'treuer', 'Kunde', 'der', 'XYZ', '
Bank', 'Münsingen', '.'], 'h': {'name': 'Thomas Müller', 'pos': [0, 1]}, 't': {
'name': 'XYZ Bank Münsingen', 'pos': [7, 9]}, 'relation': 'sonstige'}
{'token': ['Dr.', 'Hans-Heinrich', 'Gerth', 'neuer', 'Vorsitzender', 'des', 'TK
-Verwaltungsrates', '.'], 'h': {'name': 'Dr. Hans-Heinrich Gerth', 'pos': [0,
2]}, 't': {'name': 'TK-Verwaltungsrates', 'pos': [6, 6]}, 'relation': '
neu_vorsitz_von'}
Die Zahlen in 'pos' beschreiben die Position der Wörter in den Tokens,
beginnend bei 0.
Erstelle für jede Relation fünf annotierte, fiktive Beispielsätze. Die
Beispielsätze sollten abwechslungsreich sein und es soll bei jeder Relation ein
 Relativsatz und ein Satz mit Nebensatz enthalten sein.
```

Listing 5.3: Der Prompt, mit dem die synthetischen Daten in ChatGPT generiert wurden.

In Listing 5.3 ist das komplette Prompt dargestellt, ergänzt durch Details die das Ergebnis verbessern, aber keinen maßgeblichen Einfluss auf den Erfolg der Generierung haben. Durch die Anweisung, dass die Beispielsätze abwechslungsreich sein sollen, und ein Relativsatz sowie ein Satz mit Nebensatz pro Relation enthalten sein soll, führt zu Sätzen, die ein breiteres Feld der Sprache abdecken. Das Prompt sowie der komplette, durch ChatGPT generierte Output, lassen sich auf chat.openai.com[9] einsehen.

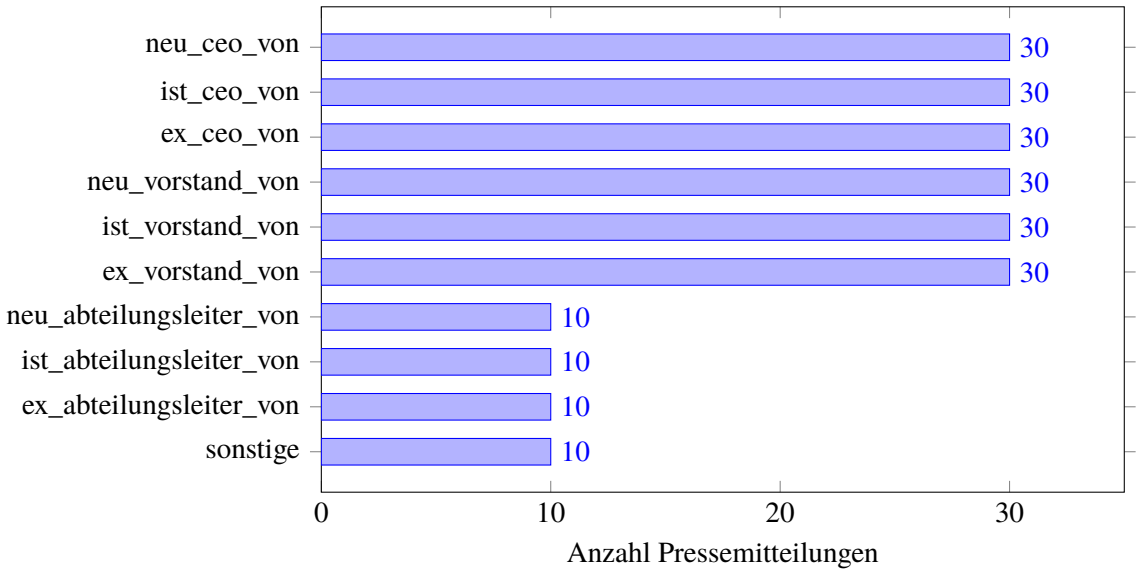

Abbildung 5.8.: Die Verteilung der 220 Datenpunkte im von ChatGPT generierten Korpus auf die 10 Relationen.

Somit besteht der mittels ChatGPT und GPT-4 generierte Korpus aus 220 synthetischen Trainingsdaten, die wie in Abbildung 5.8 auf die verschiedenen Relationen verteilt sind. Dass von den Relationen `neu_abteilungsleiter_von`, `ist_abteilungsleiter_von`, `ex_abteilungsleiter_von` sowie `sonstige` vergleichsweise wenige Beispiele existieren, liegt daran, dass diese Relationen nicht aus Unterrelationen zusammengesetzt sind. Wie sich die Relationen zusammensetzen ist in Abbildung 5.7.

5.5. Experimente

Für das Durchführen der Experimente existieren für 10 und für 4 Relationen je drei Trainingssets (handannotiert, regelbasiert erstellt und von ChatGPT generiert). Diese können in verschiedenen Kombinationen für das Training verwendet werden, um so maximale Einblicke zu ermöglichen. Vor dem Training werden die Datensätze noch durch das `validation.py`-Skript (Siehe Appendix Listing A.1) überprüft. Dieses bestätigt die Korrektheit des Datensatzes und listet Anzahl der Datenpunkte pro Relation auf. Nach diesem Schritt können die Daten in den Experimenten verwendet werden.

[9] https://chat.openai.com/share/d225d95b-eb62-45fa-8e35-78830c88f716

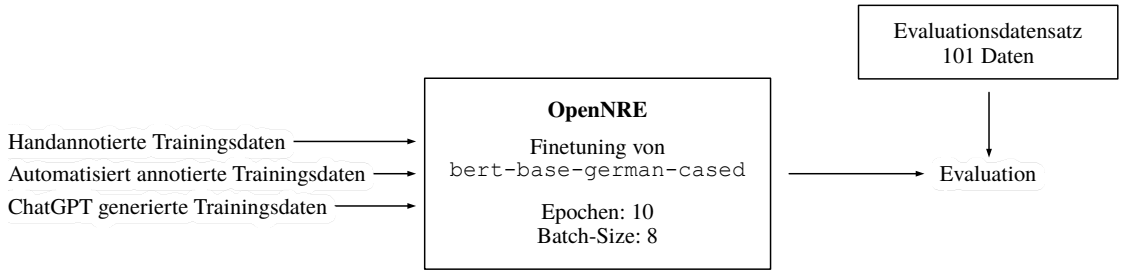

Abbildung 5.9.: Der grundlegende Aufbau der Experimente. Die verwendeten Trainingsdaten, die für das Finetuning verwendet werden, variieren von Modell zu Modell

In Abbildung 5.9 wird dargestellt, wie der grundlegende Aufbau der Experimente aussieht: Für das Finetuning des BERT-Modells werden verschiedene Kombinationen der vorhandenen Trainingsdatensätze verwendet. OpenNRE führt dann das Finetuning mit diesen Trainingsdaten am deutschen Modell `bert-base-german-cased` mit einer Batch-Size von 8 in 10 Epochen durch. Die Ergebnisse dieser Experimente werden dann mithilfe des Evaluationsdatensatzes ausgewertet.

5.5.1. Trainingsdaten

Die Trainingsdaten entsprechen direkt den in den obigen Abschnitten beschriebenen Daten, die entweder per Hand (Siehe Abschnitt 5.2), per regelbasiertem Prozess (Siehe Abschnitt 5.3) oder mittels ChatGPT (Siehe Abschnitt 5.4) erstellt wurden. In den Experimenten wurden diese teilweise zu größeren Datensets verbunden, indem alle Daten des einen Datensets dem anderen Datenset angefügt wurden.

Damit ein Modell nicht bereits mit den Daten trainiert wird, mit denen es auch evaluiert wird ist es wichtig, dass das handannotierte Trainingsset keine Daten beinhaltet, die auch das Evaluationsset beinhaltet. Die beiden Mengen müssen disjunkt sein. Dementsprechend verliert das handannotierte Trainingsset an Umfang. Zum Training konnten 323 handannotierte Sätze, mit der in Abbildung 5.10 dargestellten Verteilung auf die 10 Relationen, verwendet werden.

Damit OpenNRE die besondere Rolle der Relation `sonstige` berücksichtigt, muss sie in der OpenNRE-Datei `framework>data_loader.py` als gleichbedeutend mit der Relation „other“ vermerkt werden. So weiß OpenNRE, dass es in Fällen, wo keine Relation auf den Beispielsatz passt, auf die Relation `sonstige` zurückfallen soll.

5.5.2. Evaluationsdaten

Alle Trainingsdaten werden gegen ein Set aus 101 handannotierten Sätzen getestet. Diese Sätze wurden zufällig aus dem handannotierten Korpus extrahiert, jedoch wurde beachtet, dass jede Relation in dem Evaluationsdatensatz anteilig nur so oft vorkommt, wie sie auch im ganzen handannotierten Datensatz vorkommt. So kann gewährleistet werden, dass für jede Relation das Verhältnis zwischen Trainings- und Evaluationsdaten gleich ist, und auch jede Relation im Evaluationsdatensatz vorkommt. In einem Anwendungsfall, bei dem alle Relationen gleich oft vorkommen wäre ein

Verhalten erwünscht, in dem die korrekte Zuordnung jeder Relation den identischen Einfluss auf das Ergebnis hat. Da in den echten Daten jedoch nicht alle Relationen mit der gleichen Häufigkeit auftauchen, ist auch eine gleiche Gewichtung der Relationen nicht erwünscht. Um die Leistung eines Modells auf realistischen Daten abzubilden, kann es sinnvoll sein, die Relationen im Evaluationsset am Verhältnis der Relationen in natürlichen Sets auszurichten.

Damit das Evaluationsset allerdings nicht ganz unausgeglichen aufgebaut wird, wurde versucht, jede Relation ungefähr gleich oft im Datensatz zu repräsentieren. Dadurch wird bei der Auswertung jede Relation ungefähr gleich gewichtet, und das korrekte Identifizieren von einer Relation hat keinen unverhältnismäßig größeren Einfluss auf das Ergebnis als eine andere. Durch das Abwägen dieser beiden Ziele entsteht ein Testset, welches gleichwohl die Realität abbildet, aber auch für jede Relation erfasst, wie gut sie erkannt wird.

Um Ansätze und Modelle vergleichen zu können, ist es ohnehin am wichtigsten, dass das gewählte Evaluationsset in jedem Anwendungsfall verwendet wird.

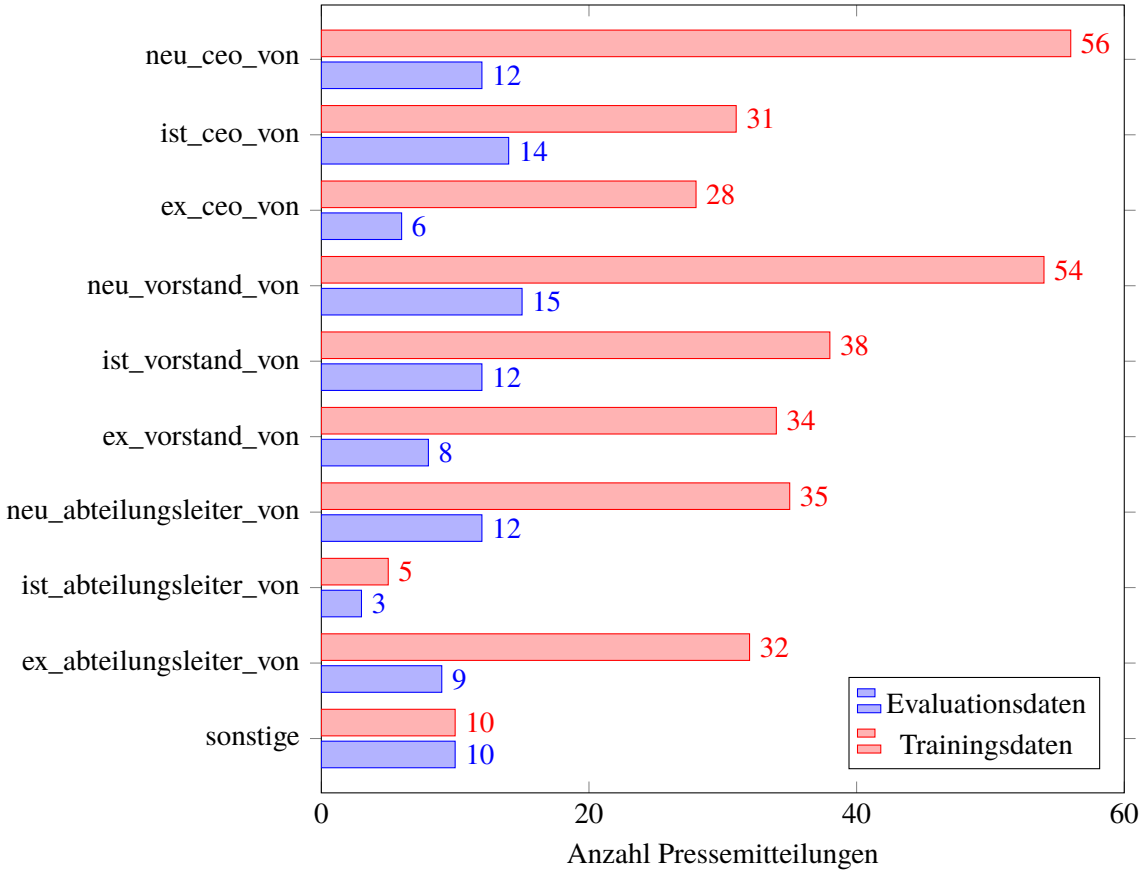

Abbildung 5.10.: Die Aufteilung der handannotierten Sätze pro Relation in Trainings- und Evaluationsdatensatz, so dass mit insgesamt 323 Sätzen trainiert, und mit 101 Sätzen evaluiert werden kann.

5.5.3. Finetuning

Für das eigentliche Training oder Tuning wird mit OpenNRE (Siehe Abschnitt 4.2) ein deutsches BERT-Modell (`bert-base-german-cased`[10]) getunt. Dieses Modell wurde bereits in vielen Publikationen verwendet und ist eines der wenigen deutschen Sprachmodelle, das über die Huggingface-Plattform[11] öffentlich zugänglich ist. Es besteht aus 110 Millionen Parametern. Jeder von ihnen wird mit einer 32-bit Fließkommazahl beschrieben, und wurde auf Daten aus der deutschen Wikipedia[12], von OpenLegalData[13] und Nachrichtenmeldungen trainiert. Im Vergleich zu LLMs in englischer Sprache ist das jedoch wenig.

Für das Tuning wurde eine Batch-Size von acht, und eine Anzahl von zehn Epochen gewählt. Die Batch-Size wurde gewählt, um dem Modell nicht zu viele Datenpunkte pro Iteration vorzuführen, aber gleichzeitig auch so viele Datenpunkte, dass mit jeder Iteration auch die 110 Millionen Parameter Verwendung finden. Wie bereits unter Abschnitt 4.2 erwähnt, merkt sich OpenNRE das beste Modell während des Trainings, auch wenn in späteren Epochen so Qualitätsverlust durch Overfitting entsteht wirkt sich das nicht negativ auf das Endergebnis aus. In Tests zeigte sich, dass die besten Ergebnisse ungefähr nach Epoche fünf zu erwarten sind, somit bieten die zehn Epochen einen deutlichen Puffer. Das Tuning kann zwischen wenigen Minuten und mehreren Stunden dauern, abhängig von der Anzahl der Beispiele im Trainingskorpus.

Über eine Abwandlung des Codes von OpenNRE kann außerdem eine Datei erstellt werden, in der genau aufgelistet wird, welches Beispiel des Evaluationssets als welche Relation eingestuft wurde. So können die Ergebnisse im Nachhinein feingranualar ausgewertet, und Verbesserungen am Trainingsset vorgenommen werden.

Im Finetuning wurde das deutsche BERT-Modell mit sechs verschiedenen Trainingssets angepasst:

1. Handannotierte Daten (Hand)
2. Regelbasiert annotierte Daten (Regel)
3. ChatGPT-generierte Daten (GPT)
4. Handannotierte Daten mit regelbasiert annotierten Daten (Hand+Regel)
5. Handannotierte Daten mit ChatGPT-generierten Daten (Hand+GPT)
6. Handannotierte Daten, regelbasiert annotierte Daten und von ChatGPT generierte Daten (Hand+Regel+GPT)

Diese Sets wurden jeweils mit 10 Relationen (Siehe Abbildung 5.7 Spalte 3) und mit 4 Relationen (Siehe Abbildung 5.7 Spalte 2) trainiert. Die Ergebnisse dieser Trainingsreihen sind für 10 Relationen in Abbildung 6.3, und für 4 Relationen in Abbildung 6.4 dargestellt.

Für einen tieferen Einblick in die Qualitätsunterschiede zwischen handannotierten, regelbasiert annotierten und von ChatGPT generierten Daten wurden zusätzliche Modelle trainiert. Dazu wurden zum Finetuning 220 Datenpunkten pro Relation verwendet, diese Anzahl wurde gewählt weil sie der

[10] https://huggingface.co/bert-base-german-cased
[11] https://huggingface.co
[12] https://dumps.wikimedia.org/dewiki/
[13] https://de.openlegaldata.io

Gesamtanzahl der von ChatGPT generierten Daten entspricht. Ergebnisse aus diesem Experiment ermöglichen es, den Einfluss der verschiedenen Arten von Trainingsdaten direkt miteinander zu vergleichen. Da die Trainingsdatenmengen der handannotierten und regelbasiert annotierten Daten mehr als 220 Datenpunkte umfassen, ist es möglich, dass zufällig gewählte Trainingssets besonders günstige oder ungünstige Datenpunkte fürs Finetuning enthalten. Die daraus resultierenden Modelle erzielen in der Folge untypisch gute oder schlechte Ergebnisse. Um das zu vermeiden werden die Tests mehrfach mit verschiedenen Teilmengen der Trainingsdaten durchgeführt, und der Mittelwert dieser Tests als Ergebnis verwendet.

5.5.4. Analyse-Skript

Um aus den feingranularen Daten, die wie in Unterabschnitt 5.5.3 beschrieben, aus OpenNRE extrahiert werden zusätzliche Erkenntnisse zu ziehen, wird ein hierfür konzipiertes Skript verwendet. Die Daten enthalten zu jedem Satz des Evaluationssets die Vorhersage, die das System gemacht hat, sowie das „Golden Label“, welches die korrekte Zuordnung beinhaltet. Damit können Standardmetriken wie Micro-Precision, Micro-Recall und Micro-F_1 berechnet werden, die Hinweise auf die Qualität der Daten geben. Ein Beispiel dafür, wie diese Daten aussehen, ist in Listing 5.4 abgebildet.

Im Rahmen dieser Arbeit ist von besonderem Interesse, ob es einen Unterschied gibt zwischen der Zuverlässigkeit, mit der die richtige Position (`ceo`, `vorstand`, `abteilungsleiter`) erkannt wird und der Zuverlässigkeit, mit der der richtige Tempus (`neu`, `ist`, `ex`) erkannt wird. Das lässt Einblicke in die Funktionsweise des Modells zu und könnte Hinweise auf die Vorteile liefern, die die Arbeit mit LLMs wie BERT hat. Außerdem kann es aufzeigen, welcher Aspekt der Trainingsdaten noch verbessert werden muss.

```
{
  "predicted_label": 1,
  "predicted_label_name": "neu_ceo_von",
  "golden_label": 1,
  "golden_label_name": "neu_ceo_von",
  "sample":
    {"token":
      ["Mit", "Wirkung", "zum", "1.", "Juni", "2009", "wird", "Michael", "
Wiener", "(", "44", ")", "zum", "Geschäftsführer", "Vertrieb", "der", "Duales",
 "System", "Deutschland", "GmbH", "(", "DSD", ")", "ernannt", "."],
      "h": {
        "name": "Michael Wiener",
        "pos": [7, 8]},
      "t": {
        "name": "Duales System Deutschland GmbH (DSD)",
        "pos": [16, 22]},
      "relation": "neu_ceo_von"
    }
}
```


Listing 5.4: Der annotierte Satz aus Listing 4.3, nach der Evaluation durch OpenNRE samt Angabe zu Vorhersage (`predicted_label` und `predicted_label_name`), und die Ground Truth, gegen die diese Vorhersage getestet wurde (`golden_label` und `golden_label_name`).

6. Evaluation und Fehleranalyse

In diesem Kapitel werden die Ergebnisse der Experimente genannt, bewertet und gedeutet. Die durchgeführten Experimente werden unter Abschnitt 5.5 beschrieben. In Abschnitt 6.1 werden die Ergebnisse der Experimente systematisch dargestellt. Abschnitt 6.2 beinhaltet eine Analyse der Ergebnisse und ordnet sie ein. Zuletzt wird in Abschnitt 6.3 versucht zu begründen, wie und weshalb die Ergebnisse so zustande gekommen sind.

6.1. Auflistung der Ergebnisse

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \qquad \text{Recall} = \frac{TP}{TP + FP}$$

$$\text{Precision} = \frac{TP}{TP + FN} \qquad F_1 = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

Abbildung 6.1.: Die klassische Definition von Accuracy, Precision, Recall und F_1. TN und TP stehen für „True Negative“ und „True Positive“, also Werte, die korrekt zugeordnet wurden. Im Gegensatz dazu stehen FN und FP für „False Negative“ und „False Positive“, Werte, die vom System falsch zugeordnet wurden.

Übliche Metriken sind bei der Bewertung der Ergebnisse von Klassifikationsaufgaben die Nutzung von Accuracy, Precision, Recall und F_1-Score, siehe Abbildung 6.1. Da diese Werte klassischerweise für ein binäres Klassifikationsproblem definiert sind, werden hier an den Problemfall angepasste Varianten verwendet. Obwohl die klassischen Bezeichnungen verwendet werden, werden stets die Werte für Micro-Accuracy, Micro-Precision, Micro-Recall und Micro-F_1-Score angegeben. Um eine Metrik mit „Micro“-Präfix zu erhalten, müssen erst die TN (und anschließend TP, FN und FP) aller Klassen addiert werden. Daraus lässt sich dann mit den bekannten Formeln der Micro-Wert ermitteln.

6.1.1. Baseline

In Abbildung 6.2 sind die Ergebnisse der Baseline auf den Testsets für 10 und für 4 Relationen abgebildet.

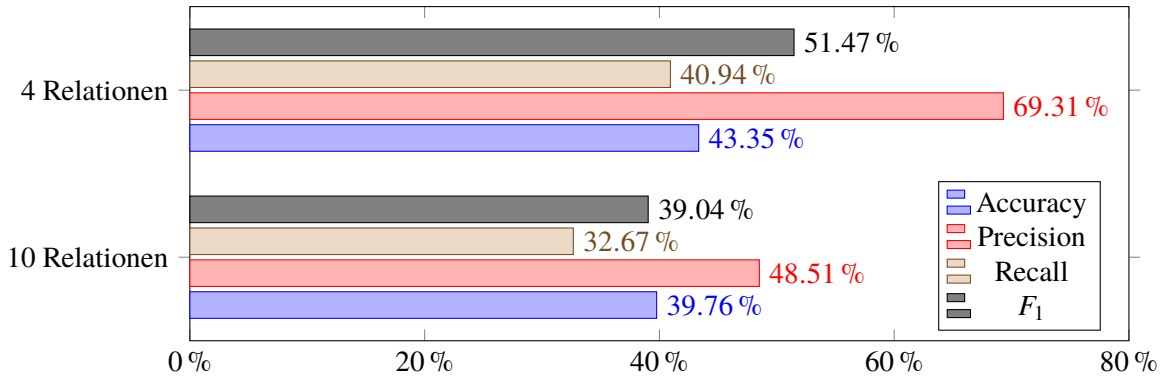

Abbildung 6.2.: Ergebnisse der Baseline, angewendet auf 4 Relationen und auf 10 Relationen.

Der F_1-Score bei 10 Relationen übertrifft den Wert, den ein zufälliger Klassifikator hätte (10 %) um 29.04 %. Im Gegensatz dazu übertrifft der F_1-Score bei dem Evaluationsset mit 4 Relationen den zufälligen Klassifikator (25 %) nur um 26.47 %. Es ist auffällig, dass die Precision bei beiden Evaluationssets deutlich höher ist als der Recall.

6.1.2. 10 Relationen

Bei Klassifikationsproblemen mit 10 Relationen würde ein Klassifikator, der die Relationen zufällig zuweist, durchschnittlich einen F_1-Score von 10 % erreichen. Die Ergebnisse der Experimente mit 10 Relationen sind in der Abbildung 6.3 dargestellt.

Die handannotierten Daten schneiden mit einem F_1-Score von 63.16 % schon alleine ziemlich gut ab. Regelbasiert annotierte Daten mit einem F_1 von 45.35 % , und die von ChatGPT generierten Daten mit einem F_1 von 41.67 % erreichen schlechtere Ergebnisse. Die Kombination der regelbasiert generierten Daten mit den Handannotierten verbessert das Ergebnis zwar gegenüber dem der regelbasiert annotierten Daten, erreicht mit einem F_1-Score von 56.18 % aber nicht das Niveau der handannotierten Daten von 63.16 %. Im Gegensatz dazu erreicht die Kombination aus von ChatGPT generierten Daten mit den Handannotierten einen Bestwert von 69.84 %. Alle drei Datenquellen zusammen erreichen einen F_1-Score von 61.78 %. Das ist zwar besser als das Ergebnis anderer Modelle, die mit regelbasierten Daten arbeiten, aber schlechter als das Ergebnis aus der Kombination von handannotierten und von ChatGPT generierten Daten (69.84 %).

6.1.3. 4 Relationen

Bei Klassifikationsproblemen mit 4 Relationen würde ein Klassifikator, der die Relationen zufällig zuweist, durchschnittlich einen F_1-Score von 0.25 % erreichen.

Auch bei 4 Relationen erzielt ein Modell, das allein durch handannotierte Daten getunet wurde gute Ergebnisse mit einem F_1-Score von 79.37 %. Die Modelle, die allein durch regelbasiert annotierte und durch ChatGPT generierte Daten getunt wurden kommen auf geringere F_1-Scores von 71.19 % für regelbasiert annotierte Daten und 65.97 % für von ChatGPT generierte Daten. Das Tuning mit handannotierten und regelbasiert annotierten Daten führt zu einem Modell mit 76.74 % F_1-Score, was nicht an das Ergebnis mit rein handannotierten Daten (79.37 %) herankommt. Erst

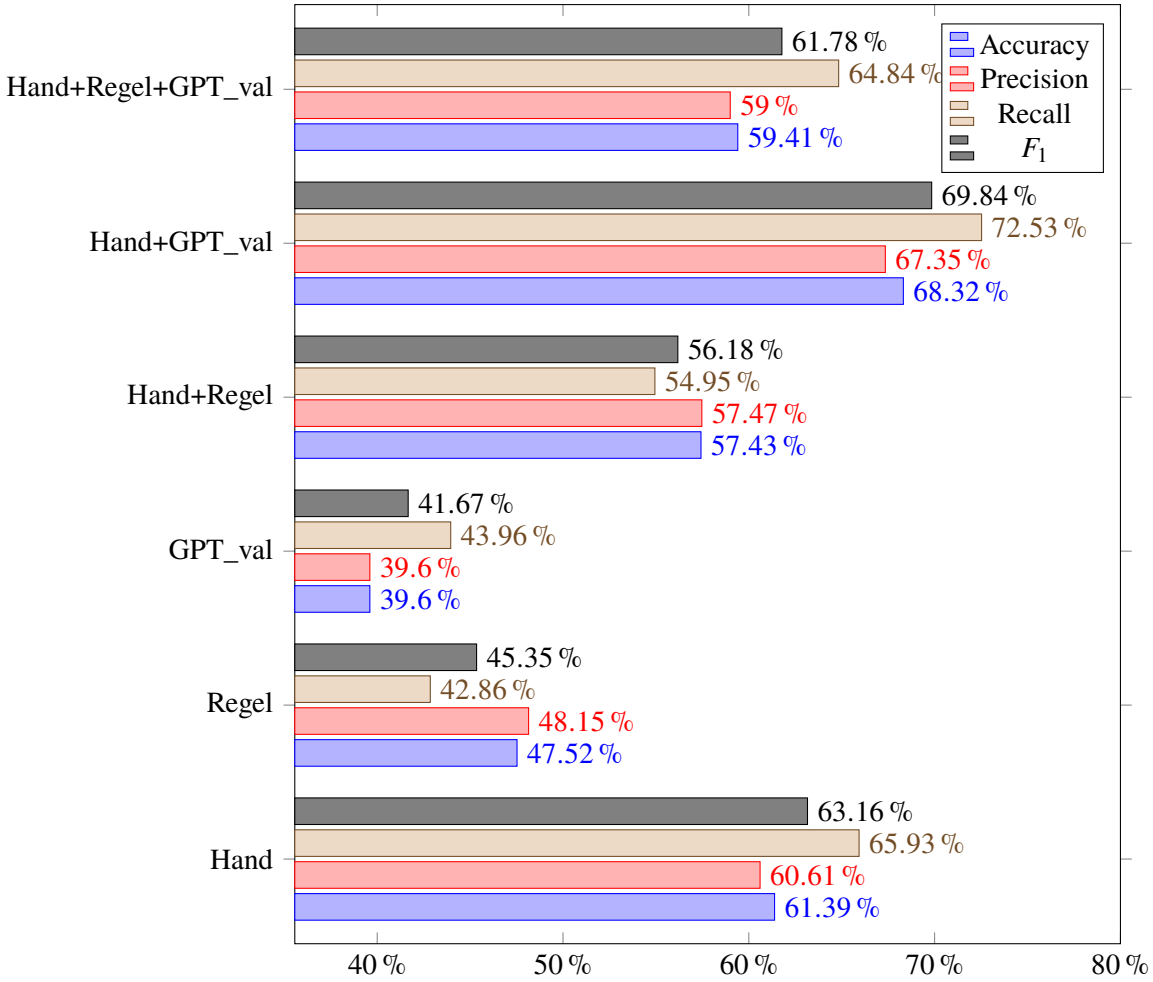

Abbildung 6.3.: Ergebnisse des Finetuning mit Trainingsdaten, die 10 Relationen beinhalten. Den besten F_1-Score von 69.84 % erzielt das Modell, welches mit Hand+GPT-Daten getuned wurde.

der F_1-Score des Modells, welches die handannotierten Daten mit den von ChatGPT generierten kombiniert, übertrifft den Score der handannotierten Daten mit einem Wert von 82.54 %. Die Kombination aller drei Datenquellen führt auch in diesem Fall zu keiner Verbesserung gegenüber den rein handannotierten Daten. Das resultierende Modell hat einen F_1-Score von 77.09 %.

6.1.4. Direkter Vergleich zwischen den Trainingsdaten

Um die Daten miteinander vergleichen zu können, ohne die Anzahl der Trainingsdaten als Faktor berücksichtigen zu müssen, wurde für jede Art Trainingsdaten auch ein Modell mit 220 Datenpunkten getunt. Die Ergebnisse sind in Abbildung 6.5 dargestellt. Der F_1-Score der handannotierten Daten liegt bei diesem verkleinerten Trainingsdatensatz im Fall der 10 Relationen bei 53.18 % , und erreicht im Fall der 4 Relationen 71.92 %. Die von ChatGPT generierten Daten erreichen bei 10 Relationen einen F_1-Score von 41.67 %, bei 4 Relationen sind es 65.97 %. Da nur 220 Datenpunkte von ChatGPT vorliegen sind diese Ergebnisse identisch zu den vorherigen Experimenten, die mit

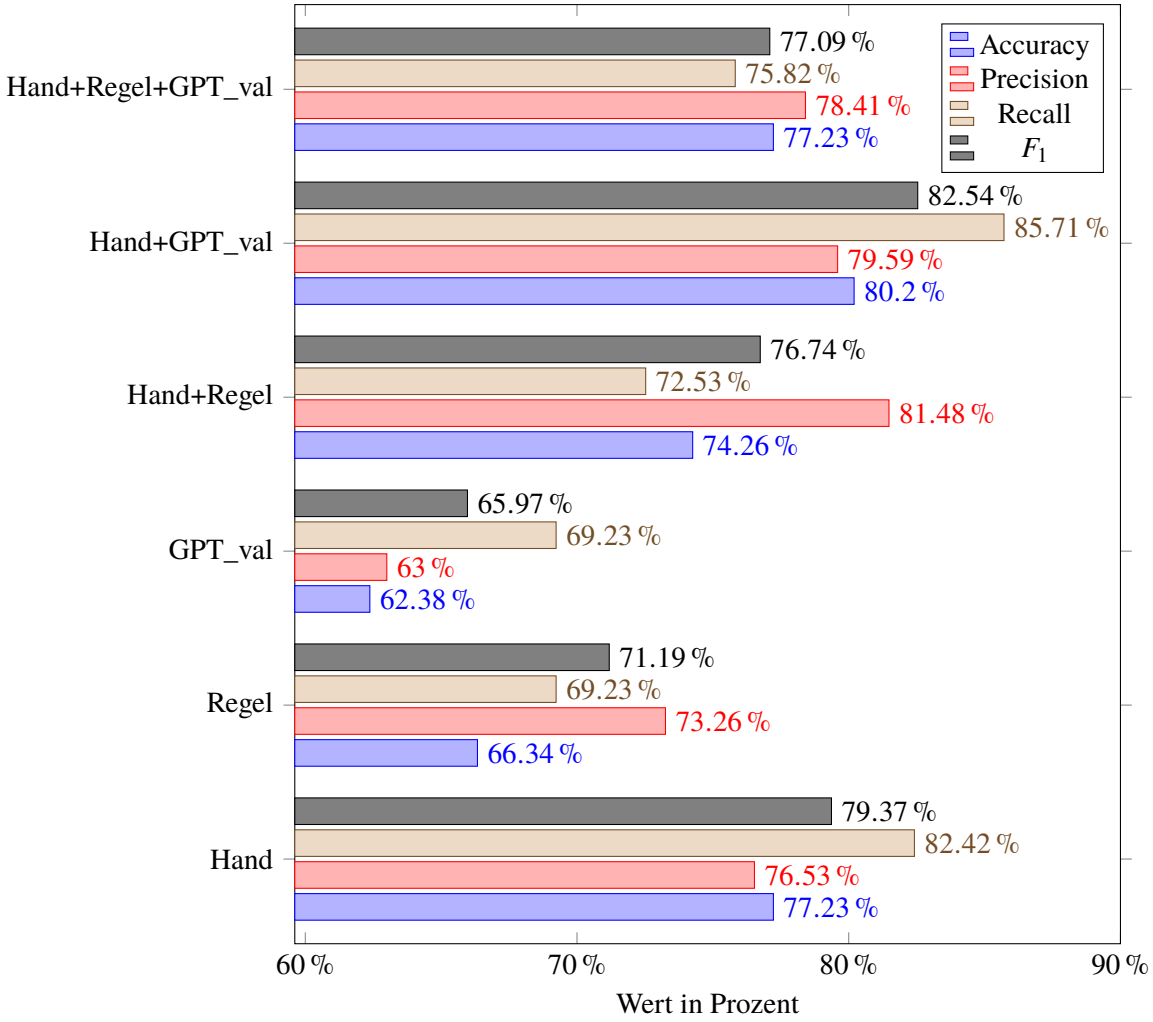

Abbildung 6.4.: Ergebnisse des Finetuning mit Trainingsdaten, die 4 Relationen beinhalten. Den besten F_1-Score von 82.54 % erzielt das Modell, welches mit handannotierten und durch ChatGPT generierten Daten getunt wurde.

ChatGPT durchgeführt wurden. Die größte Differenz zwischen den Ergebnissen für 10 und 4 Relationen erreichen die regelbasiert annotierten Trainingsdaten. Auf 10 Relationen erreichen sie den niedrigsten F_1-Score von 31.11 %, wohingegen sie mit 4 Relationen 69.35 % erreichen.

6.2. Vergleiche

In Abbildung 6.6 werden die erzielten F_1-Scores aller Methoden dieser Arbeit verglichen. Es fällt auf, dass das Einbeziehen von regelbasiert annotierten Daten in jedem Fall die Ergebnisse des Modells gegenüber eines Modells, welches ohne die regelbasiert annotierten Daten getunt wurde, verschlechtert (Hand+Regel: 76.74 % gegenüber Hand: 79.37 %, Hand+Regel+GPT_val: 77.09 % gegenüber Hand+GPT_val: 82.54 %). Auf die gleiche Art lässt sich beobachten, dass das Einbeziehen der von ChatGPT generierten Daten zu einer Verbesserung des Modells führt

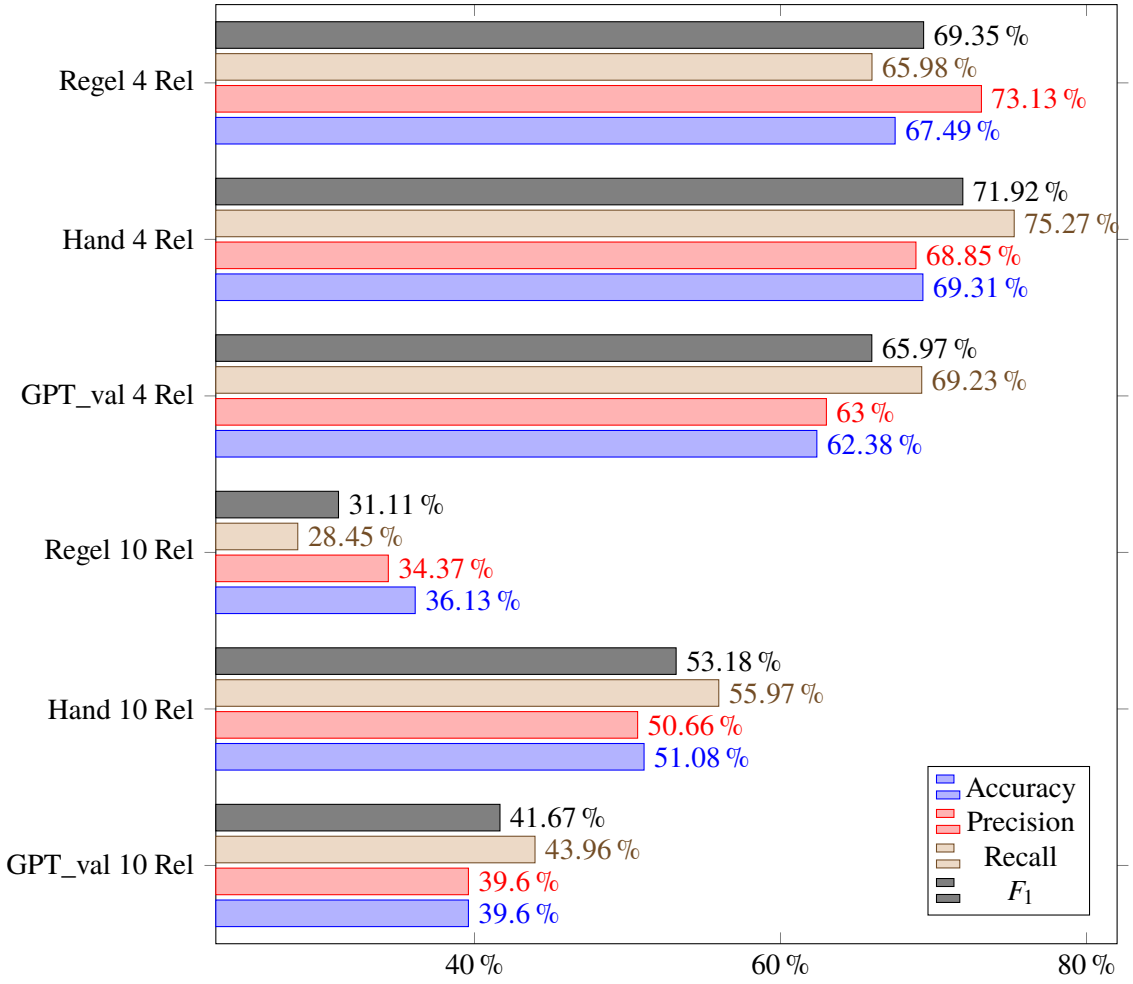

Abbildung 6.5.: Ergebnisse der Modelle, wenn jedes Modell mit nur 220 Datenpunkten trainiert wird.

(Hand+GPT_val: 82.54 % gegenüber Hand: 77.09 %, Hand+Regel+GPT_val: 77.09 % gegenüber Hand+Regel: 76.74 %). Dieses Muster besteht sowohl bei 10 als auch bei 4 Relationen. Trotz dieses verbessernden Einflusses der von ChatGPT generierten Daten und dem verschlechternden Einfluss der regelbasiert annotierten Daten, schneidet ein Modell, welches nur auf regelbasiert annotierten Daten trainiert wurde, deutlich besser ab, als das mit von ChatGPT generierten Daten (Regel: 71.19 % gegenüber GPT_val: 65.97 %). In Abschnitt 6.3 werden Erklärungsansätze für dieses Verhalten vorgeschlagen.

Um die Modelle, die 10 Relationen klassifizieren, aussagekräftig mit denen vergleichen zu können, die 4 Relationen klassifizieren, sind die Balken in Abbildung 6.6 mit dem Wert markiert, den ein durchschnittlicher zufälliger Klassifikator erzielen würde. Bei 10 Relationen erreicht ein Klassifikator, der die Relationen zufällig zuweist durchschnittlich einen F_1-Score von 10 %, bei 4 Relationen sind es 25 %. Durch diesen durschnittlichen, zufälligen Klassifikator lässt sich vergleichen, wie sehr eine Methode den F_1-Score verbessert, auch zwischen Klassifikationsproblemen mit verschieden vielen Relationen. Dabei wird angenommen, dass der Vorteil, der durch ein Modell entsteht, erst da beginnt, ab wo es Relationen besser zuordnet als der durchschnittliche zufäl-

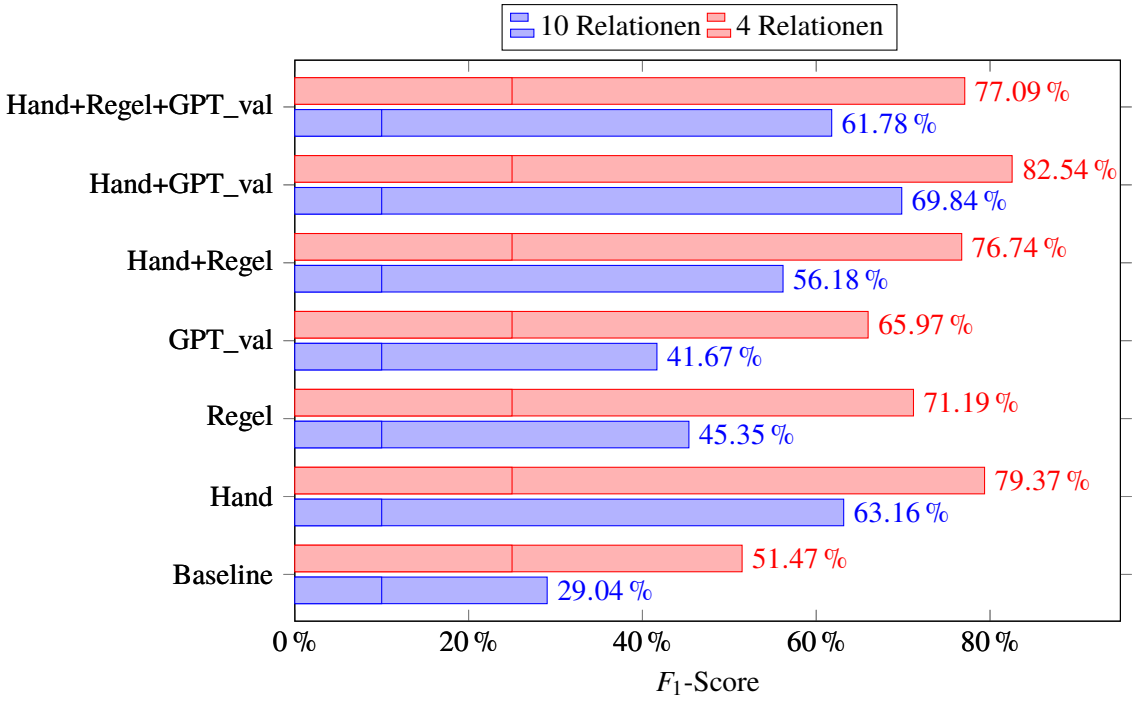

Abbildung 6.6.: Ein Vergleich der F_1-Scores aller Methoden. Durch Markierungen wird gezeigt, welchen Score ein durchschnittlicher, zufälliger Klassifikator erreichen würde. Bei 10 Relationen sind das 10 %, bei 4 Relationen 25 %.

lige Klassifikator. So ist ein Modell, welches bei einem binären Klassifikationsproblem einen F_1-Score von 70 % erreicht, zwar auf den ersten Blick besser als ein Modell, welches bei einem Klassifikationsproblem mit fünf Klassen 47 % erreicht. Wird aber berücksichtigt, wie groß die Verbesserung gegenüber dem zufälligen Klassifikator ist, so ist die Leistung des zweiten Klassifikators (47 %P (Prozentpunkte) – 20 %P = 27 %P) größer als die des Ersten (70 %P – 50 %P = 20 %P).

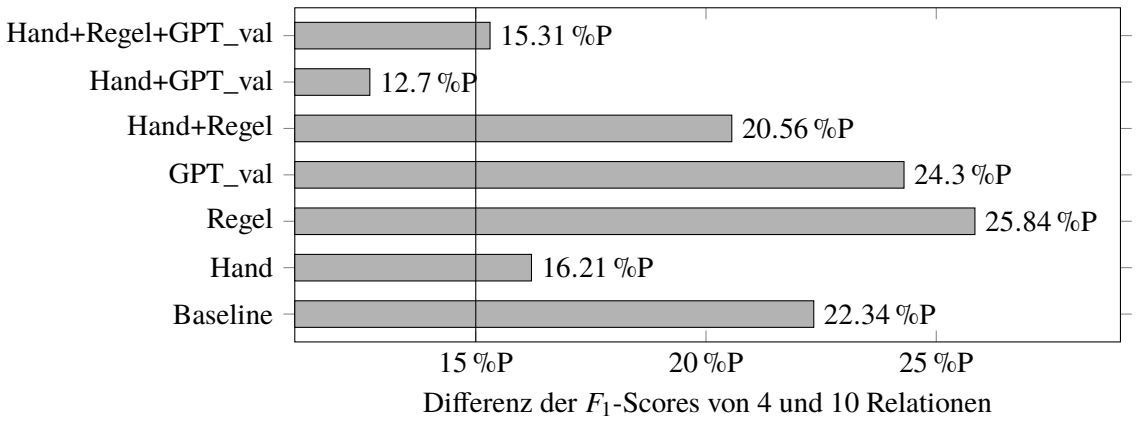

Abbildung 6.7.: Die Differenz der F_1-Scores zwischen 10 und 4 Relationen. Würden die Methoden die Klassifikation um den gleichen Wert verbessern, wäre die Differenz zwischen den F_1-Scores 15 Prozentpunkte (%P) (25 % – 10 %, die Werte der durchschnittlichen zufälligen Klassifikatoren). Die Linie markiert diese 15 %P.

Auf diese Art lässt sich auch beurteilen, ob die Modelle, wenn man ihre Leistungsfähigkeit unter diesem Aspekt betrachtet, verhältnismäßig besser 10 Relationen oder 4 Relationen klassifizieren können. In Abbildung 6.7 sind die Differenzen der F_1-Scores dargestellt. Wenn die Baseline, aufgrund ihrer rein regelbasierten Wirkungsweise außer acht gelassen wird, so fällt auf, dass ein Zusammenhang zwischen der absoluten Qualität der Modelle und dieser Differenz zwischen dem F_1-Score für 10 und für 4 Klassen besteht. Mit einem Pearsons-Korrelationskoeffizienten r [Pea20] von -0.886 besteht ein starker Zusammenhang zwischen beiden Eigenschaften: „Je höher der absolute F_1-Score eines Modells, desto niedriger die Differenz zwischen dem F_1-Score für 10 und dem für 4 Relationen“. Unterabschnitt 6.3.5 enthält Erklärungsansätze für dieses Verhalten.

Modell	Hand	Regel	GPT_val	Hand+Regel	Hand+ GPT_val	Hand+Regel+ GPT_val
F_1-Score 4 Relationen	79.37 %	71.19 %	65.97 %	76.74 %	82.54 %	77.09 %
F_1-Score 10 Relationen	63.16 %	45.35 %	41.67 %	56.18 %	69.84 %	61.78 %
Differenz F_1-Scores	16.21 %P	25.84 %P	24.30 %P	20.56 %P	12.70 %P	15.31 %P

Tabelle 6.1.: Die Werte der F_1-Scores für 10 und für 4 Relationen, zusammen mit der Differenz zwischen den Werten. Zwischen den Werten der F_1-Scores bei 4 Relationen und der Differenz besteht eine Korrelation mit $r = -0.886$. Zwischen den Werten der F_1-Scores bei 10 Relationen und der Differenz besteht eine Korrelation mit $r = -0.967$.

Aus Unterabschnitt 6.1.4 und Abbildung 6.5 geht hervor, dass die geringere Trainingsdatenmenge auf jedes Modell einen negativen Einfluss hat. Die Rangfolge der F_1-Scores verändert sich gegenüber den Experimenten mit den kompletten Datensätzen mit einer Ausnahme nicht. Diese Ausnahme ist jedoch bemerkenswert: Bei 10 Relationen schneiden die regelbasiert annotierten Daten mit 31.11 % nur etwas besser ab als es der durchschnittliche, zufällige Klassifikator tun würde. In diesem Szenario entsteht also nur ein sehr kleiner Vorteil durch die Nutzung regelbasiert annotierter Daten.

Abgesehen davon zeigt sich, dass die handannotierten Daten auch mit dieser kleineren Trainingsmenge den höchsten F_1-Score erzielen. Der auf diese Art von den handannotierten Daten erzielte F_1-Score ist dennoch deutlich niedriger (7.41 %P), als der, der erreicht wird, wenn mit dem 323 Datenpunkte umfassenden, kompletten handannotierten Trainingsdatensatz getunt wird. Die Differenz zwischen dem F_1-Score des ganzen Trainingsdatensatzes und der reduzierten ist bei den regelbasiert annotierten Daten mit 1.84 %P deutlich kleiner.

6.3. Diskussion

Wie in den Ergebnissen bei Abbildung 6.3 und Abbildung 6.4 deutlich wird haben Modelle, die mit regelbasiert annotierten Daten erstellt wurden niedrigere F_1-Scores als die gleichen Modelle ohne die regelbasiert annotierten Daten. Das ist der Fall, obwohl die Modelle, die nur mit regelbasiert annotierten Daten trainiert sind nicht den schlechtesten F_1-Score aufweisen (43.53 % bzw. 71.19 %), sondern von den Modellen, die nur auf von ChatGPT generierten Daten trainiert sind unterboten

werden (41.67 % bzw. 65.97 %). Es ist schwierig, das Verhalten von ML-Modellen und DL-Netzen im Speziellen mit Sicherheit klaren Ursachen zuzuordnen. DL-Modelle werden von Entwicklern und Nutzern als Black Box behandelt, und folgen typischerweise der Beobachtung, dass sich die Erklärbarkeit eines ML-Modells umgekehrt zu seiner Vorhersagegenauigkeit verhält – je höher die Vorhersagegenauigkeit, desto geringer die Erklärbarkeit [Rud19; XUD+19]. LLMs mit Finetuning gehören hier auf zu den Modellen mit hoher Vorhersagegenauigkeit, und sehr geringer Erklärbarkeit, weshalb über die Ursachen der Leistungsunterschiede nur Vermutungen angestellt werden können.

Wie bereits in Unterabschnitt 3.1.3 beschrieben, werden ML-Modelle häufig als Black Box behandelt, da sie keinen Rückschluss darauf zulassen, wie genau ein Ergebnis zustande kommt. Deswegen sind im Folgenden keine konkreten Erklärungen, sondern nur Vermutungen möglich, die das Verhalten der Modelle zu erklären versuchen.

6.3.1. Deutungshypothese 1: Abdeckung der Features

Eine Erklärung für die in Abschnitt 6.1 hinterlegten Testergebnisse wird durch diese Deutungshypothese geliefert. Sie geht davon aus, dass das Testset, welches repräsentativ für sämtliche Daten des Anwendungsbereichs stehen soll, eine gewisse Feature-Menge umfasst, die nötig ist, um alle Einträge des Testsets der korrekten Relation zuzuordnen. Jedes trainierte Modell deckt dabei einen Teil des Testsets ab, der dem F_1-Scores des Modells entspricht – je mehr abgedeckt ist, desto besser werden die Einträge den Relationen zugeordnet. Wie viele Features ein Modell erlernt, hängt von der Vielfältigkeit und Qualität der Trainingsdaten ab, da nur die darin enthaltenen Informationen während des Finetunings vom PLM übernommen werden können. Die Kombination von zwei Trainingsdatensätzen könnte dazu führen, dass sich auch die Features beider Trainingsdatensätze ergänzen, sofern beide Datenmengen nicht die gleichen oder sehr ähnliche Featuremengen abdecken.

Durch diese Hypothese kann erklärt werden, weshalb die von ChatGPT generierten Daten, wenn sie als einzige Trainingsdaten genutzt werden, nur einen sehr niedrige Accuracy erzielen. Die abgedeckte Featuremenge ist klein. Erklärt werden könnte das dadurch, dass ChatGPT scheinbar doch keine abwechslungsreichen und vielfältigen Beispiele generieren konnte, sondern nur ein kleines Feld von dem abdecken konnte, was in den Evaluationsdaten gefordert ist. Ähnlich ließe sich für die regelbasiert annotierten Daten argumentieren: Die durch den Prozess gefundenen Daten könnten aufgrund der regelbasierten Filterung nur einen kleinen Ausschnitt der kompletten Featuremenge abbilden.

Der entscheidende Unterschied könnte in der Menge der abgedeckten Features selbst liegen: Regelbasiert annotierte Daten greifen auf den Fraunhofer-Korpus zurück, aus dem auch die handannotierten Daten stammen. Der regelbasierte Filterprozess führt dazu, dass nur eine kleinere Teilmenge der Daten abgebildet wird als bei den handannotierten Daten, weshalb auch die Featuremenge der regelbasiert annotierten Daten eine Teilmenge der Featuremenge der handannotierten Daten umfasst. Da ChatGPT nur einen sehr kleinen Teil (zwei Beispiele, siehe Unterabschnitt 5.4.3) des Fraunhofer-Korpus kennt, und die Beispiele auf dem Textverständnis des LLMs basieren, könnten diese Beispiele Features beinhalten, die zwar zur Klassifikation der Relationen wichtig sind, aber nur schwer aus den Daten des Fraunhofer-Korpus zu extrahieren sind.

Vor diesem Hintergrund könnte gut erklärt werden, weshalb das Modell, welches auf handannotierten und von ChatGPT generierten Daten einen höheren F_1-Score erzielt als die handannotierten Daten alleine. Die regelbasiert annotierten Daten führen im Gegensatz dazu weder in der Kombination

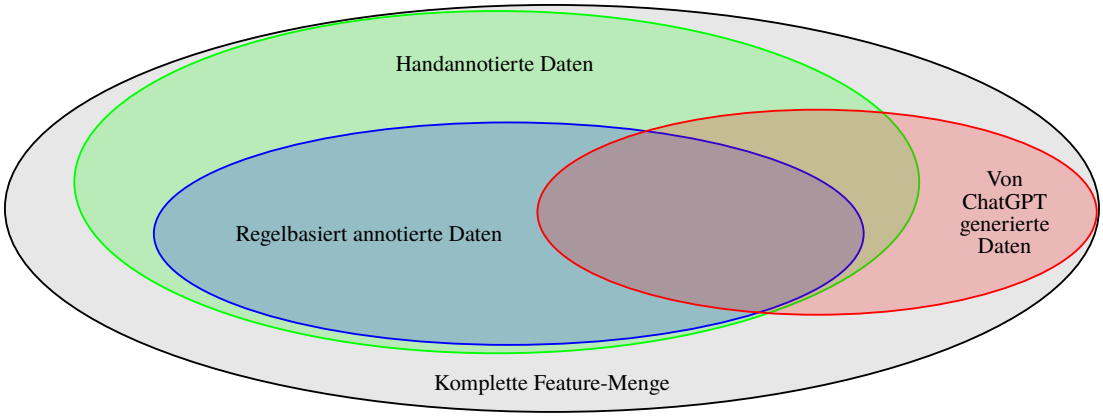

Abbildung 6.8.: Eine beispielhafte Darstellung der Menge der Features, die von den Modellen abgedeckt werden.

mit den handannotierten, noch in Kombination mit handannotierten und von ChatGPT generierten Daten zu einer Verbesserung des F_1-Scores. Eine schematische Abdeckung der Feature-Menge ist in Abbildung 6.8 dargestellt.

6.3.2. Deutungshypothese 2: Qualität und Quantität

Schon lange ist bekannt, dass zwei Faktoren über die Qualität eines ML-Modells entscheiden: Die Qualität [Kav09] und die Quantität [FMY95] der Trainingsdaten. Im Idealfall ist beides erfüllt: Ein Modell kann mit einer hohen Quantität von Trainingsdaten mit einer hohen Qualität trainiert werden. In der Praxis müssen Kompromisse eingegangen werden: Je höher die Qualität der Daten, desto aufwändiger die Beschaffung, was meistens zu einer geringeren Quantität führt. Um im Rahmen dieser Deutungshypothese konkrete Schlüsse ziehen zu können, wird dieser Zusammenhang zwischen Qualität und Quantität der Trainingsdaten über die Formel in Abbildung 6.9 modelliert. Dabei wird der natürliche Logarithmus des Faktors der Quantität verwendet, um den mit steigender Anzahl nachlassenden Einfluss der Trainingsdaten auf den F_1-Score abzubilden [HAP+23]. Die Qualität der Daten fasst hierbei viele Eigenschaften der Daten zusammen: Wie viele Muster und Varianten der Evaluationsdaten in den Trainingsdaten vorhanden sind, ob die Annotationen in den Daten korrekt sind, ob die Trainingsdaten Muster beinhalten die in den Evaluationsdaten nicht auftreten etc.

$$F_1 \mathrel{\widehat{=}} \ln(\text{Quantität}) \cdot \text{Qualität} \qquad \Longrightarrow \qquad \text{Qualität} \mathrel{\widehat{=}} \frac{F_1}{\ln(\text{Quantität})}$$

Abbildung 6.9.: Die zugrundeliegende Annahme der zweiten Deutungshypothese: F_1 ist ein Produkt aus Qualität und Quantität der Trainingsdaten. Da der F_1 Wert nicht linear mit der Anzahl der Trainingsdaten wächst, wird ihr Einfluss durch den natürlichen Logarithmus gedämpft.

Basierend auf der Formel aus Abbildung 6.9 lässt sich die Qualität der Trainingsdaten für die verschiedenen Trainingsdatensätze abschätzen, da die Anzahl der enthaltenen Daten sowie die über 4 und 10 Relationen gemittelten F_1-Scores bekannt sind. Für Qualität existiert in diesem Rahmen keine klare Metrik, die resultierenden Werte dienen lediglich dem Vergleich untereinander. In Tabelle 6.2 sind die Trainingsdatensätze zusammen mit F_1-Score, Quantität und der mit der Formel aus Abbildung 6.9 berechneten Qualität hinterlegt.

Diese Interpretation der Qualität der Daten erklärt, weshalb die Ergebnisse des Hand+Regel-Modells nicht nur keine Verbesserung gegenüber dem Modell, das auf rein handannotierten Daten trainiert wurde aufweisen, sondern sogar einen schlechteren F_1-Score erzielen. Der Grund wäre die, im Vergleich zu anderen Trainingsdaten, schlechte Qualität der regelbasiert annotierten Daten. Den gleichen Effekt kann man bei dem Hand+Regel+GPT-Modell beobachten. Diese Deutungshypothese erklärt ebenfalls, warum die von ChatGPT generierten Daten zusammen mit den handannotierten Daten zu einem F_1-Score führt, der besser ist als beide Trainingsdatensätze in seperaten Modellen. Durch die hohe Qualität der Daten kann durch eine höhere Quantität eine Verbesserung des F_1-Scores erzielt werden.

Nach Qualität geordnet ist die Reihenfolge der Trainingsdatensätze auch plausibel: Trainingsdatensätzen, die aus mehreren zusammengesetzt sind wird eine durchschnittliche Qualität zugeordnet, die zwischen denen der Teildatensätze liegt, und auch die durchschnittliche Qualität des Hand+Regel+GPT-Datensatzes liegt über der des Hand+Regel-Datensatzes. Die Werte selbst sind teilweise nicht so verteilt, wie intuitiv erwartet werden würde: Die errechnete Qualität des Hand+GPT-Datensatzes liegt weit näher an der des reinen handannotierten Datensatzes als an der des von ChatGPT generierten Datensatzes.

Die grundsätzliche Anwendbarkeit der Formel wird durch die Ergebnisse des direkten Vergleichs zwischen den handannotierten Daten und den durch ChatGPT generierten Daten bekräftigt. Der Mittelwert der in Abbildung 6.5 dargestellten F_1-Scores, zusammen mit der Menge der Datenpunkte errechnet für die handannotierten Daten eine sehr ähnliche Qualität. Da alle Daten des handannotierten Datensatzes mit der identischen Vorgehensweise erstellt wurden, kann davon ausgegangen werden, dass auch ihre Qualität gleich ist. Die Abweichung der berechneten Werte für Qualität um nur 0.1 unterstützt diese Intuition.

Bei den Werten für Qualität, die für den regelbasierten Ansatz mit 220 Datenpunkten ermittelt werden, wird die große Differenz zwischen 10 und 4 Relationen in der Tabelle nicht berücksichtigt, da ein Mittelwert beider F_1-Scores verwendet wird. Es existiert dennoch eine deutliche Abweichung zwischen der errechneten Qualität für 220 und 8 567 Daten, was im Konflikt mit dieser Deutungshypothese steht.

Die in Abbildung 6.9 aufgestellte Formel erklärt zwar viele Zusammenhänge dieser Arbeit, muss jedoch in zukünftigen Arbeiten noch validiert und ergänzt werden. Es ist laut der aktuellen Formel möglich, einen F_1-Score von über 100 % zu erreichen. Eine Normalisierung ist also ebenso notwendig, um konsistente Ergebnisse zu erhalten.

Diese Deutungshypothese legt nahe, dass von ChatGPT trainierte Daten nicht von höherer Qualität sind als handannotierte Daten, was von dem direkten Vergleich handannotierter und von ChatGPT generierter Daten in Unterabschnitt 6.1.4 bestätigt wird. Dennoch können die generierten Daten die schwer zu beschaffenden echten Trainingsdaten ergänzen und auf diese Art bessere Ergebnisse

Datensatz	F_1-Score	Quantität	Qualität
Hand	71.27 %	323	12.33
Hand (220 Datenpunkte)	65.98 %	220	12.23
Hand+GPT_val	76.19 %	614	11.86
GPT_val	53.82 %	220	9.97
Regel (220 Datenpunkte)	50.23 %	220	9.02
Hand+Regel+GPT_val	69.45 %	9 181	7.61
Hand+Regel	66.46 %	8 961	7.30
Regel	58.27 %	8 567	6.43

Tabelle 6.2.: Mit der Formel aus Abbildung 6.9 berechnete Werte für die durchschnittliche Qualität der Daten in den Trainingssets. Der Wert für Qualität hat hierbei keine direkte Bedeutung, sondern kann ausschließlich für den Vergleich von Trainingssets genutzt werden. Die Werte für den F_1-Score wurden aus den Werten für 10 und 4 Relationen gemittelt.

erzielen. Um die Qualität der generierten Daten zu verbessern können bessere Prompts ausprobiert werden, es können Grenzfälle gesucht und speziell dafür Trainingsdaten erzeugt werden. Auch die Generation neuer Trainingsdaten ist möglich, um höhere F_1-Scores zu erzielen.

6.3.3. Regelbasiert annotierte Daten

Es ist naheliegend, dass beide Forschungshypothesen, aber auch nicht berücksichtigte Faktoren, einen Einfluss auf die Ergebnisse haben. Unabhängig von der Interpretation der Ergebnisse lassen sich einige objektive Erkenntnisse aus den Experimenten ableiten:

Der Ansatz, die handannotierten Daten mit regelbasiert annotierten Daten zu unterstützten und so für bessere Ergebnisse zu sorgen hat nicht die gewünschten Ergebnisse erzielt. Auf die Frage, warum dieser Ansatz hier, anders als bei Gu und Leroy [GL19], zu schlechteren Ergebnissen führt, gibt es keine eindeutige Antwort. Beide Deutungshypothesen, die die Ergebnisse der Experimente zu erklären versuchen, können auch hierauf angewendet werden.

Nach Deutungshypothese 1, die sich mit der Abdeckung der Features beschäftigt, könnte das Featureset bei Gu und Leroy größer sein als in dieser Arbeit. Die handannotierten Daten decken nur Teile dieses Featuresets ab. Der Experte, der den regelbasierten Ansatz bei Gu und Leroy gestaltet hat, konnte umfassende Erkenntnisse einfließen lassen. Dadurch wird die Menge der Features, die in den von Hand annotierten Daten enthalten ist, übertroffen, und ein größeres Featureset wird abgedeckt. Werden die handannotierten Daten nun durch die regelbasiert annotierten Daten ergänzt, könnte sich also eine Verbesserung der Leistung des daraus resultierenden Modells ergeben.

In der zweiten Deutungshypothese sind die tatsächlichen Werte, die zu den Trainingsdaten bekannt sind, relevant. In Tabelle 6.3 wird dargestellt, dass sich die in Deutungshypothese 2 vorgeschlagene Formel zur Abschätzung der Qualität der Trainingsdaten auch auf die Daten von Gu und Leroy anwenden lässt.

Datensatz	F_1-Score	Quantität	Qualität
Hand	55.6 %	19 000	5.64
Regel	55.8 %	109 000	4.81
Hand+Regel	59.8 %	128 000	5.09

Tabelle 6.3.: Anwendung der Deutungshypothese 2 mit der Formel aus Abbildung 6.9 auf die Ergebnisse von Gu und Leroy [GL19]. Der Ansatz für regelbasiert annotierte Daten wird von Gu und Leroy RBP genannt, „Rule Based Parser“.

Eine mögliche Erklärung, weshalb die Verwendung regelbasiert annotierter Daten bei Gu und Leroy zu verbesserten Ergebnissen und in dieser Arbeit zu schlechteren Ergebnissen führt, ist die mangelhafte Qualität der regelbasiert annotierten Daten im Vergleich zu den handannotierten Daten. Während die Differenz der Qualität zwischen Hand und Regelbasiert annotierten Daten nach der Formel aus Abbildung 6.9 bei Gu und Leroy relativ niedrig ist (0.83 %P) ist sie in dieser Arbeit relativ hoch (5.5 %P). Bei Gu und Leroy kann diese Differenz durch die Quantität der Daten überwunden werden, in diesem Ansatz ist der Ausgleich dieser Differenz nicht möglich. Wäre die Qualität der handannotierten Daten in dieser Arbeit geringer, wäre mit einer Verbesserung durch die regelbasierten Daten zu rechnen.

Zu den Problemen, die die Qualität der Trainingsdaten des regelbasierten Ansatz dieser Arbeit einschränken könnten, gehören folgende:

1. Die NER von SpaCy, die dem regelbasierten Ansatz zugrunde liegt, erkennt 15 % aller Entitäten falsch oder unvollständig, siehe Tabelle 3.2.
2. Der Ansatz der Stichwortprüfung ist nicht in der Lage, die grammatikalischen Feinheiten natürlicher Sprache zu interpretieren.
3. Die Wörterbücher der Stichwortprüfung könnten nicht optimal gewählt sein.

Dass die regelbasiert annotierten Daten selbst bei der kleinen Trainingsdatenmenge von 220 Datenpunkten auf 10 Relationen Ergebnisse erzielen konnte, die nahe an denen von 8 567 Datenpunkten liegen, kann an der Verwendung von realen Daten liegen. Es ist plausibel, dass die echten Daten dem BERT-Modell ermöglichen, mit weniger Datenpunkten relevante Features zu erkennen. Dadurch kann schon durch diese kleine Teilmenge ein beträchtlicher Teil der in den regelbasierten Daten enthaltenen Features erlernt werden.

Im Gegensatz dazu sind die regelbasiert annotierten Daten nicht in der Lage, beim Finetuning relevante Features zu übermitteln. Dies kann an einer zu geringen Anzahl Trainingsdaten liegen, die zu wenige Features enthalten, die für die Klassifizierung relevant sind. Durch die geringe Menge Daten ist es auch naheliegend, dass das Modell falsche Features identifiziert, was dann zu schlechten Ergebnissen führt.

Dabei ist zu erwähnen, dass der Implementierungsaufwand des regelbasierten Ansatzes der Höchste im Rahmen dieser Arbeit war, den Aufwand der Konvertierung von handannotierten Daten oder der Korrektur der von ChatGPT generierten Daten also übersteigt. Es ist abzuwägen, ob die erwarteten Ergebnisse diesen Aufwand rechtfertigen, oder ob nicht andere Methoden wie z.B. die Generierung von Trainingsdaten mit ChatGPT höhere Qualität versprechen, siehe Unterabschnitt 6.3.4. Diese

Entscheidung hängt jedoch immer vom Anwendungsfall ab und kann viele Faktoren beinhalten, jedoch bieten sich NLP-Aufgaben aufgrund ihrer Komplexität und vieler Sonderfälle nicht an, mit einem regelbasierten Ansatz unterstützt zu werden.

Dass sich auch schon mit geringen Mengen regelbasiert Annotierter Trainingsdaten ein Modell finetunen lässt, das dem Modell der handannotierten Trainingsdaten nahekommt ist zwar beeindruckend, aber praktisch von keinem Vorteil. Die Zeit, die es benötigt, den regelbasierten Prozess zu implementieren könnte auch direkt in die Handannotation gesteckt werden. Es zeigt sich auch dass die große Menge an Daten, die sich so annotieren lässt, zu keinem großen Vorteil führt was den F_1-Score des damit trainierten Modells angeht.

6.3.4. Von ChatGPT generierte Daten

Die Generierung von Trainingsdaten hat sich als sehr vielversprechend erwiesen. Die Ergebnisse der Modelle, in denen die handannotierten Daten um die von ChatGPT generierten ergänzt wurden, sind besser, und durch zukünftige Verbesserungen können Trainingsdaten schneller, effizienter und umfangreicher generiert werden. In Unterabschnitt 5.4.1 wird ausführlich beschrieben, welche Verbesserungen in Zukunft die Arbeit mit ChatGPT erleichtern würden, allen voran jedoch ein Zugang zur OpenAI API.

Es wirkt naheliegend, dass eine höhere Anzahl an Trainingsdaten, die durch ChatGPT generiert wird, auch zu besseren Ergebnissen führen könnte, und mit der API und anderen Verbesserungen kann diese Vermutung untersucht werden. Auch die Variation in den Trainingsdaten sowie das Prompting mit Grenzfällen und dadurch resultierende, problembezogene Trainingsdaten lässt sich durch leichteren Zugriff auf ChatGPT noch ausführlicher erforschen.

Die Anwendungsfelder für Trainingsdaten, die von einer generativen KI erstellt wurden, sind vielfältig und es ist noch viel Forschung möglich, um eventuelle Stärken und Schwächen auf verschiedenen Gebieten herauszufinden. Die hohe Geschwindigkeit der Iterationen und Innovationen von OpenAI[1] erfordert außerdem das stetige Überprüfen und Hinterfragen bisheriger Grenzen und Probleme im Umgang mit ChatGPT und anderen generativen KIs. Ein Hindernis stellt dabei die Eigenschaft von LLMs dar, zu Halluzinieren, ein Problem welches bei dem Einsatz von ChatGPT und ähnlichen Systemen berücksichtigt werden muss. Anwendungsfelder wie NLP, für die der faktische Inhalt der generierten Trainingsdaten keine Rolle spielt, profitieren ohne große Beschränkung von dieser Art, Trainingsdaten zu erlangen.

6.3.5. Korrelation der Differenz der F_1-Scores

Eine verbesserte Leistung in der Klassifikation von 4 Relationen gegenüber 10 Relationen, auch über die durch den durchschnittlichen zufälligen Klassifikator implizierte Verbesserung hinaus ist nicht verwunderlich. Das Klassifikationsproblem ist auf 4 Relationen deutlich leichter, sofern die zeitliche Komponente aus der Problemstellung entfernt wurde (Siehe Abbildung 5.7). Die ML-Modelle scheinen die Eigenschaften der Relationen, für die im Trainingsset nun pro Relation mehr Beispiele enthalten sind, also besser zu erlernen.

[1] https://openai.com/blog

Die in Abschnitt 6.2 beobachtete Korrelation von hohen F_1-Scores mit niedrigeren Differenzen zwischen dem F_1-Score für 4 Relationen und dem für 10 kann mit der Natur von Klassifikationsproblemen zusammenhängen. Je höher die Leistung eines Modells, desto schwieriger wird es, sie weiter zu verbessern [SHK+23]. In dieser Arbeit führt das zu Ergebnissen mit handannotierten und von ChatGPT generierten Trainingsdaten, die zwar weniger gut sind als durch andere Versuche angedeutet, aber dennoch deutlich besser als die Baseline. Es gibt andere Fälle wo diese sogenannten „Diminishing Returns“ zu größeren Problemen führen: So können bei ausreichend guten Modellen selbst mit hohem Aufwand und modernsten Methoden nur noch Verbesserungen von ≤ 1 % erreicht werden [HAP+23]. Dadurch muss abgeschätzt werden, wie viel Wert diese geringen Verbesserungen haben, da insbesondere das Pre-Training, aber auch das Finetuning kosten- und arbeitsintensiv sein kann.

6.3.6. Anzahl der Relationen

Es wird schnell deutlich, dass sich das Klassifikationsproblem mit 10 Relationen im Rahmen dieser Arbeit nicht zufriedenstellend lösen lässt. Selbst das Modell mit dem höchsten F_1-Score (Hand+GPT) erreicht nur einen Wert von 69.84 % (Siehe Abbildung 6.3). Die Ursache hierfür kann in den teilweise sehr subtilen Unterschieden zwischen den Klassen liegen. Einzelne Wörter können den Unterschied machen zwischen `neu`, `ist` und `ex`, und auch die Einteilung in die Relationen `ceo`, `vorstand`, `abteilungsleiter` und `sonstiges` ist durch viele verschiedene Berufsbezeichnungen nicht trivial (wie die 4 Relationen mit F_1-Score von 82.54 % zeigen).

Um diese Nuancen zu erlernen reichen die wenigen Trainingsdaten, die im Rahmen dieser Arbeit zur Verfügung standen, nicht aus. Es kann zum Forschungsobjekt zukünftiger Arbeiten werden, herauszufinden, wie mit sehr wenigen Trainingsdaten auch Klassifikationsprobleme mit einer höheren Anzahl an Relationen zufriedenstellend gelöst werden. Die Generierung der Trainingsdaten durch ein LLM ist dabei zwar eine vielversprechende Methode, um die gewünschte Qualität zu erhalten sind jedoch weitere Verbesserungen über Prompt-Engineering sinnvoll.

7. Zusammenfassung und Ausblick

In diesem Kapitel werden die Ergebnisse der Arbeit auf den Punkt gebracht und gebündelt wiedergegeben. Diese Zusammenfassung findet sich in Abschnitt 7.1. Im Ausblick in Abschnitt 7.2 werden offene Themen vorgestellt, die in zukünftigen Arbeiten behandelt werden können.

7.1. Zusammenfassung

Im Rahmen dieser Arbeit wurde gezeigt, dass schon heute klassische ML-Anwendungen von generativer KI profitieren können. Es ist ohne nennenswerte Hürden für jeden möglich, diese Methoden anzuwenden, und zum Beispiel mit ChatGPT zu testen, wie gut sich ein bestimmter Anwendungsfall abbilden lässt. Durch präzises Prompting ist es möglich, hochwertige Trainingsdaten generieren zu lassen, die handannotierte Daten ergänzen. Besonders auf dem Feld des NLP können LLMs durch ihre Fähigkeit, natürliche Sprache zu erzeugen, einen großen Einfluss auf das Erstellen von Trainingskorpora haben.

Es ist jedoch wichtig zu beachten, dass dieser Ansatz aktuell noch einige Probleme birgt. So können die von LLMs generierten Trainingsdaten nicht ohne einen Korrekturschritt für das Finetuning von Modellen verwendet werden, da ein signifikanter Teil der Trainingsdaten Fehler aufweist. Die Programmierung eines Korrekturskripts kann sehr aufwändig sein, da die Fehler, mit denen LLMs die Trainingsdaten generieren, extrem vielseitig sind. Durch diese umfangreichen Fehlerquellen kann auch das Einrichten eines solchen Skripts zu einer hohen Komplexität führen.

Zusätzlich zu fehlerhaften Daten ist die Generierung von großen Datenmengen mit ChatGPT als Beispiel für ein LLM zum Zeitpunkt der Experimente noch sehr schwerfällig. Begrenzte Output-Länge, das langsame Generieren von Text und die zahlenmäßige Begrenzung der Anfragen pro Tag stellen erhebliche Hindernisse dar was den Umfang der Trainingsdaten angeht. Zukünftige Entwicklungen sowie Zugang zur OpenAI API werden diese Probleme stark reduzieren oder gänzlich beseitigen.

Die Forschungshypothese dieser Arbeit lautet: „Generative KI ermöglicht die Synthese bereits annotierter Trainingsdaten, die zum Finetuning von LLMs zur Relation Classification verwendet werden können. So können in speziellen Anwendungsfällen, in denen wenige echte Trainingsdaten zur Verfügung stehen, mit weniger Aufwand vergleichbare oder bessere Ergebnisse erzielt werden als mit herkömmlichen Methoden wie regelbasierter Synthese von Trainingsdaten.“

Es konnte gezeigt werden, dass generative KI in der Tat in der Lage ist, Trainingsdaten zu erzeugen mit denen Finetuning an einem PLM durchgeführt werden kann. Diese Daten können jedoch Fehler enthalten wie falsche Indizes oder fehlerhaften Syntax, und sollten vor der Nutzung erst von diesen

bereinigt werden. Die Qualität und der Inhalt der Daten hängt stark von dem verwendeten Prompt ab, es lohnt sich also vor und während der Erstellung eines Trainingsdatensatzes mit verschiedenen Formulierungen zu experimentieren.

Mit den resultierenden Daten lassen sich in der Tat bessere Ergebnisse erzielen als durch die regelbasiert annotierten Daten, zu deren Erstellung trotz des Korrekturskripts und der Probleme mit ChatGPT ein deutlich höherer Aufwand nötig war. Das zeigt, dass ChatGPT schon heute eine gute Alternative zu herkömmlichen Methoden der Trainingsdatengewinnung darstellt.

Die Experimente zeigen auch, dass die Nutzung regelbasiert annotierter Trainingsdaten einen kleineren Anwendungsbereich hat als ursprünglich vermutet. So scheint natürliche Sprache trotz einer klaren Aufgabenstellung zu komplex, um sie von einem regelbasierten System zuverlässig annotieren lassen zu können. In Bereichen der Medizin, wo die Synthese mit generativer KI womöglich erschwert ist, und große Mengen unannotierter Daten verfügbar sind, kann sich der Prozess des regelbasierten Annotierens dennoch auszahlen. Allgemein ist die Wahl der besten Methode, Trainingsdaten zu erhalten, stark von dem Anwendungsfall und den gegebenen Umständen abhängig und es kann keine allgemein gültige Lösung vorgeschlagen werden.

7.2. Ausblick

OpenAI hat mit ChatGPT und GPT-4 der breiten Bevölkerung das Potential von generativer KI und LLMs anschaulich gemacht. Das daraus resultierende öffentliche Interesse für die Forschung unterstützt die Arbeit, die auf diesem Feld geleistet wird, und bringt zahlreiche Forschende dazu, die Verschiedenen Aspekte von ChatGPT zu untersuchen [ZZL+23a]. Basierend auf den Erkenntnissen dieser Arbeit bieten die in Unterabschnitt 7.2.3 Themen noch viel Potential für die Bearbeitung in zukünftigen Veröffentlichungen. Auch auf dem Feld der Relation Classification und Relation Extraction kann noch weitere Forschung betrieben werden.

7.2.1. Document-Level Relation Extraction

Im Rahmen dieser Arbeit wurden Entitäten und Relationen extrahiert, die im gleichen Satz vorkommen. Dies lässt außer acht, dass sich in der Realität viele Relationen erst aus dem Kontext eines Abschnitts oder eines ganzen Dokuments ergeben. Laut Yao et al. lassen sich mehr als 40,7 % aller Fakten, die in einem Dokument enthalten sind, nur aus dem satzübergreifenden Kontext erkennen.

Die Dokumenten-basierte RE ist jedoch ein deutlich schwierigeres Problem als die RE auf Satzbasis [THBN22]. Regelbasierte Ansätze, die schon auf Satzbasis Schwierigkeiten haben kompetitive Ergebnisse zu erzielen, können kaum die Komplexität, die durch satzübergreifende Relationen entsteht, abbilden. Dafür können ML-Methoden und LLMs auf diesem Bereich punkten. Der Einsatz von synthetischen Trainingsdaten, die durch ChatGPT generiert werden könnten ist auch in diesem Anwendungsfall vielversprechend. Zusätzlich sind die handannotierten Daten dieser Arbeit, wie in Abschnitt 5.2 beschrieben, vor der Umwandlung in das OpenNRE-Format auf Dokumenten-Ebene annotiert worden, weshalb sie gut für den Einsatz in einer Arbeit geeignet sind, in der RE auf Dokumenten-Ebene untersucht wird.

7.2.2. Zero-Shot Prompts

Was LLMs von herkömmlichen ML-Modellen unterscheidet, ist die Fähigkeit, natürliche Sprache zu verstehen, und kohärente, passende Antworten zu geben. Dies lässt sich auch für klassische NLP-Aufgaben nutzen, die dann kein spezielles Modell mit Finetuning benötigen, sondern basierend auf einer Beschreibung der Aufgabe das Problem lösen können. „Zero-Shot“ bedeutet in diesem Kontext, dass das Modell im Voraus nie mit der Aufgabe konfrontiert wurde und keine Beispieldaten kennt, die die Lösung des Problems beschreiben. Das Finetunen eines Modells wird im Vergleich dazu als „Few-Shot-Learning“ bezeichnet, da es zum Beispiel bei Klassifikationsproblemen zu jeder Relation einige Beispiele gezeigt bekommt.

Die Ergebnisse dieser Zero-Shot Methode können häufig nicht mit denen von Modellen mithalten, die auch nur mit sehr kleinen Mengen Trainingsdaten finegetunt sind [ZHM+22]. Diesen Aspekt von LLMs zu nutzen, kann jedoch viel Zeit bei der Entwicklung von Lösungen in einer Vielzahl von Bereichen einsparen. Zukünftige LLMs werden bessere Leistungen bei der Lösung von Zero-Shot Prompts erzielen, und schon heute gibt es Möglichkeiten, die Ergebnisse von existierenden Modellen durch geschicktes Prompting zu verbessern. Kojima et al. haben bei der Forschung an LLMs herausgefunden, dass schon eine simple Ergänzung des Prompts um die Phrase „Let’s think step by step“ die Genauigkeit der Ergebnisse signifikant (in einem Fall von 17.7 % auf 78.7 %) verbessert [KGR+22]. Ergebnisse wie dieses legen nahe, dass es noch andere Möglichkeiten geben könnte, auf einfache Weise die Antworten von generativen Modellen deutlich zu verbessern. Dies sollte auch für den Umgang mit fortgeschritteneren Modellen umfassend erforscht werden.

7.2.3. Zukünftige Arbeiten mit ChatGPT

Die Arbeit mit ChatGPT ist zum Zeitpunkt der Experimente noch von vielen Hindernissen begleitet, wie in Unterabschnitt 5.4.1 beschrieben wird. Wenn in Zukunft der Umgang mit ChatGPT einfacher, oder der Zugang zur API unbeschränkt verfügbar sein wird, kann ein stärkerer Fokus auf die „eigentliche“ Arbeit mit ChatGPT gelenkt werden: Prompt Engineering. Wie auch schon von den Entwicklern bei OpenAI angemerkt[1], produzieren GPT-Modelle unter Umständen schon bei kleinen Änderungen im Wortlaut des Prompts stark unterschiedliche Lösungen. Dieses Verhalten zu verstehen und sich zu Nutzen zu machen, kann zu großen Verbesserungen der Ergebnisse führen. Aktuelle Forschung zeigt, dass über „Chain-of-Thought“-Reasoning LLMs dazu bewegt werden können, komplexe Aufgaben mit vielen Zwischenschritten zu lösen und so signifikant bessere Ergebnisse zu erzielen [LKB+23; WWS+22a; WWS+22b]. Prompt Engineering und One-Shot-Learning sind eng miteinander verwandte Themen, und die Methode von Kojima et al., „Let’s think step by step“ in das Prompt aufzunehmen, Teil des Ansatzes, Modelle zum Chain-of-Thought-Reasoning zu bewegen.

Das korrekte Prompting von ChatGPT ermöglicht es, bei der Generierung von Trainingsdaten gezielt Daten für Sonder- und Randfälle zu generieren. Dadurch werden die Modelle speziell auf diese Fälle vorbereitet, was besonders wertvoll für das Training sein kann und nur durch die Synthese von Trainingsdaten so gezielt möglich ist. Auf diese Art können die Daten dem Trainingsset auch iterativ und nach Bedarf zugeführt werden, was für viel Flexibilität sorgt.

[1] https://openai.com/blog/chatgpt

7.2.4. Verbesserungen an LLMs / ChatGPT

Large Language Models selbst, ChatGPT im Besonderen, bieten auch noch die Möglichkeit zur Verbesserung. Während die Modelle komplexe Prompts wie im Rahmen dieser Arbeit zwar teilweise richtig interpretieren, so erfüllen auch viele generierte Datenpunkte nicht die gewünschten Anforderungen. OpenAI arbeitet bereits an Verbesserungen [LKB+23] und wird die Systeme auch in Zukunft noch besser auf noch mehr Anwendungsfälle optimieren.

Grundlegende Probleme wie Halluzination müssen gelöst werden, bevor Anwendungen, die auf LLMs basieren, für Bereiche, in denen Faktualität wichtig ist, angewendet werden können [AM23]. Es gibt zwar bereits Ansätze, wie Halluzinationen unterdrückt oder erkannt werden können [ZZL+23b], dennoch sind auf dem Gebiet noch viele Fragen offen.

Schon heute sehen jedoch einige Forscher wie Zhang et al. in GPT-4 den Grundstein einer Allgemeinen Künstliche Intelligenz (General Artificial Intelligence, GAI) [ZZL+23a]. Bubeck et al. haben Experimente mit GPT-4 veröffentlicht, in denen Antworten gegeben wurden, die laut den Forschern Anzeichen für Intelligenz zeigen [BCE+23]. Unabhängig davon, wie man ChatGPT und das zugrundeliegende GPT-4 einordnet, werden zukünftige Entwicklungen nur zu noch besseren Antworten und Interaktionen führen, die sich auf immer größere Bereiche ausweiten.

Literaturverzeichnis

[AB08] A. Auger, C. Barrière. „Pattern-based approaches to semantic relation extraction: A state-of-the-art“. In: *Terminology* 14.1 (2008), S. 1 (zitiert auf S. 21).

[AM23] H. Alkaissi, S. I. McFarlane. „Artificial hallucinations in ChatGPT: implications in scientific writing“. In: *Cureus* 15.2 (2023) (zitiert auf S. 80).

[BCE+23] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg et al. „Sparks of artificial general intelligence: Early experiments with gpt-4“. In: *arXiv preprint arXiv:2303.12712* (2023) (zitiert auf S. 26, 80).

[BCL+23] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung et al. „A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity“. In: *arXiv preprint arXiv:2302.04023* (2023) (zitiert auf S. 27).

[BDE+04] S. Brants, S. Dipper, P. Eisenberg, S. Hansen-Schirra, E. König, W. Lezius, C. Rohrer, G. Smith, H. Uszkoreit. „TIGER: Linguistic interpretation of a German corpus“. In: *Research on language and computation* 2 (2004), S. 597–620 (zitiert auf S. 30).

[BMR+20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al. „Language models are few-shot learners“. In: *Advances in neural information processing systems* 33 (2020), S. 1877–1901 (zitiert auf S. 26).

[BP22] E. Bassignana, B. Plank. „What Do You Mean by Relation Extraction? A Survey on Datasets and Study on Scientific Relation Classification“. In: *arXiv preprint arXiv:2204.13516* (2022) (zitiert auf S. 21, 37).

[CKG+19] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov. „Unsupervised cross-lingual representation learning at scale“. In: *arXiv preprint arXiv:1911.02116* (2019) (zitiert auf S. 25).

[CLC+21] R. J. Chen, M. Y. Lu, T. Y. Chen, D. F. Williamson, F. Mahmood. „Synthetic data in machine learning for medicine and healthcare“. In: *Nature Biomedical Engineering* 5.6 (2021), S. 493–497 (zitiert auf S. 20).

[CLL+23] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, L. Sun. „A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt“. In: *arXiv preprint arXiv:2303.04226* (2023) (zitiert auf S. 16).

[CPSS05] P. Cimiano, A. Pivk, L. Schmidt-Thieme, S. Staab. „Learning taxonomic relations from heterogeneous sources of evidence“. In: *Ontology Learning from Text: Methods, evaluation and applications* 123 (2005), S. 59–73 (zitiert auf S. 21).

[DC19] J. Dahmen, D. Cook. „SynSys: A synthetic data generation system for healthcare applications“. In: *Sensors* 19.5 (2019), S. 1181 (zitiert auf S. 20).

[DCLT19] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Juni 2019, S. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423 (zitiert auf S. 24, 25, 38).

[EMMR23] T. Eloundou, S. Manning, P. Mishkin, D. Rock. *GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models*. 2023. arXiv: 2303.10130 [econ.GN] (zitiert auf S. 15).

[FCJ+20] A. Falenska, Z. Czesznak, K. Jung, M. Völkel, W. Seeker, J. Kuhn. „GRAIN-S: Manually Annotated Syntax for German Interviews“. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. 2020, S. 5169–5177 (zitiert auf S. 31).

[FKKK09] J. Finzen, M. Kintz, H. Kett, S. Koch. „Strategic Innovation Management on the Basis of Searching and Mining Press Releases.“ In: *WEBIST*. 2009, S. 347–353 (zitiert auf S. 16, 33, 35).

[FMY95] G. Foody, M. McCulloch, W. Yates. „The effect of training set size and composition on artificial neural network classification“. In: *International Journal of Remote Sensing* 16.9 (1995), S. 1707–1723 (zitiert auf S. 16, 71).

[GA19] D. Gunning, D. Aha. „DARPA's explainable artificial intelligence (XAI) program“. In: *AI magazine* 40.2 (2019), S. 44–58 (zitiert auf S. 27).

[GL19] Y. Gu, G. Leroy. „Mechanisms for automatic training data labeling for machine learning“. In: (2019) (zitiert auf S. 15, 16, 28, 73, 74).

[Gri97] 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, S. 10–27 (zitiert auf S. 21).

[HAP+23] Z. Hong, A. Ajith, G. Pauloski, E. Duede, K. Chard, I. Foster. „The Diminishing Returns of Masked Language Models to Science“. In: *arXiv preprint arXiv:2205.11342* (2023) (zitiert auf S. 71, 76).

[HBT+21] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, A. Mohamed. „Hubert: Self-supervised speech representation learning by masked prediction of hidden units“. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), S. 3451–3460 (zitiert auf S. 25).

[Hea92] M. A. Hearst. „Automatic acquisition of hyponyms from large text corpora“. In: *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*. 1992 (zitiert auf S. 21).

[HGY+19] X. Han, T. Gao, Y. Yao, D. Ye, Z. Liu, M. Sun. „OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction“. In: *Proceedings of EMNLP-IJCNLP: System Demonstrations*. 2019, S. 169–174. DOI: 10.18653/v1/D19-3029. URL: https://www.aclweb.org/anthology/D19-3029 (zitiert auf S. 37, 38).

[IABB22] M. R. Islam, M. U. Ahmed, S. Barua, S. Begum. „A systematic review of explainable artificial intelligence in terms of different application domains and tasks“. In: *Applied Sciences* 12.3 (2022), S. 1353 (zitiert auf S. 27).

[JK23] A. Jadon, S. Kumar. „Leveraging Generative AI Models for Synthetic Data Generation in Healthcare: Balancing Research and Privacy“. In: *arXiv preprint arXiv:2305.05247* (2023) (zitiert auf S. 20).

[Kav09] T. Kavzoglu. „Increasing the accuracy of neural network classification using refined training data“. In: *Environmental Modelling & Software* 24.7 (2009), S. 850–858 (zitiert auf S. 52, 71).

[KGR+22] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa. „Large language models are zero-shot reasoners“. In: *arXiv preprint arXiv:2205.11916* (2022) (zitiert auf S. 79).

[KWT+18] K. Kazuhiro, R. A. Werner, F. Toriumi, M. S. Javadi, M. G. Pomper, L. B. Solnes, F. Verde, T. Higuchi, S. P. Rowe. „Generative adversarial networks for the creation of realistic artificial brain magnetic resonance images“. In: *Tomography* 4.4 (2018), S. 159–163 (zitiert auf S. 20).

[Lev+66] V. I. Levenshtein et al. „Binary codes capable of correcting deletions, insertions, and reversals“. In: *Soviet physics doklady*. Bd. 10. 8. Soviet Union. 1966, S. 707–710 (zitiert auf S. 53).

[LGP+18] G. Leroy, Y. Gu, S. Pettygrove, M. K. Galindo, A. Arora, M. Kurzius-Spencer. „Automated Extraction of Diagnostic Criteria From Electronic Health Records for Autism Spectrum Disorders: Development, Evaluation, and Application“. In: *Journal of medical Internet research* 20.11 (Nov. 2018), e10497. ISSN: 1439-4456. DOI: 10.2196/10497. URL: https://europepmc.org/articles/PMC6249505 (zitiert auf S. 28).

[LKB+23] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, K. Cobbe. *Let's Verify Step by Step*. 2023. arXiv: 2305.20050 [cs.LG] (zitiert auf S. 51, 79, 80).

[Luc21] F. Lucini. „The real deal about synthetic data“. In: *MIT Sloan Management Review* 63.1 (2021), S. 1–4 (zitiert auf S. 19).

[LWW23] Y. Lu, H. Wang, W. Wei. „Machine Learning for Synthetic Data Generation: a Review“. In: *arXiv preprint arXiv:2302.04062* (2023) (zitiert auf S. 19, 20).

[MAP+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Watten-

berg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (zitiert auf S. 37).

[MDPA23] A. G. Møller, J. A. Dalsgaard, A. Pera, L. M. Aiello. „Is a prompt and a few samples all you need? Using GPT-4 for data augmentation in low-resource classification tasks“. In: *arXiv preprint arXiv:2304.13861* (2023) (zitiert auf S. 20).

[MLK19] J. Martínek, L. Lenc, P. Král. „Training strategies for OCR systems for historical documents“. In: *Artificial Intelligence Applications and Innovations: 15th IFIP WG 12.5 International Conference, AIAI 2019, Hersonissos, Crete, Greece, May 24–26, 2019, Proceedings 15*. Springer. 2019, S. 362–373 (zitiert auf S. 20).

[NG15] T. H. Nguyen, R. Grishman. „Relation extraction: Perspective from convolutional neural networks“. In: *Proceedings of the 1st workshop on vector space modeling for natural language processing*. 2015, S. 39–48 (zitiert auf S. 21).

[Nik19] S. I. Nikolenko. *Synthetic Data for Deep Learning*. 2019. arXiv: 1909.11512 [cs.LG] (zitiert auf S. 20).

[NRR+17] J. Nothman, N. Ringland, W. Radford, T. Murphy, J. R. Curran. „Learning multilingual named entity recognition from Wikipedia“. In: (Okt. 2017). DOI: 10.6084/m9.figshare.5462500.v1. URL: https://figshare.com/articles/dataset/Learning_multilingual_named_entity_recognition_from_Wikipedia/5462500 (zitiert auf S. 31).

[Ope23] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL] (zitiert auf S. 26, 48).

[OSR19] P. J. Ortiz Suárez, B. Sagot, L. Romary. „Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures“. en. In: Hrsg. von P. Bański, A. Barbaresi, H. Biber, E. Breiteneder, S. Clematide, M. Kupietz, H. Lüngen, C. Iliadi. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019. Mannheim: Leibniz-Institut für Deutsche Sprache, 2019, S. 9–16. DOI: 10.14618/ids-pub-9021. URL: http://nbn-resolving.de/urn:nbn:de:bsz:mh39-90215 (zitiert auf S. 31).

[Pea20] K. Pearson. „Notes on the history of correlation“. In: *Biometrika* 13.1 (1920), S. 25–45 (zitiert auf S. 69).

[PPB17] S. Pawar, G. K. Palshikar, P. Bhattacharyya. „Relation extraction: A survey“. In: *arXiv preprint arXiv:1712.05191* (2017) (zitiert auf S. 21).

[Rag21] T. E. Raghunathan. „Synthetic data“. In: *Annual review of statistics and its application* 8 (2021), S. 129–140 (zitiert auf S. 19).

[RHW86] D. E. Rumelhart, G. E. Hinton, R. J. Williams. „Learning representations by back-propagating errors“. In: *nature* 323.6088 (1986), S. 533–536 (zitiert auf S. 24).

[RNS+18] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever et al. „Improving language understanding by generative pre-training“. In: (2018) (zitiert auf S. 24, 26).

[RPG+21] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, I. Sutskever. *Zero-Shot Text-to-Image Generation*. 2021. arXiv: 2102.12092 [cs.CV] (zitiert auf S. 25).

[Rub04] D. B. Rubin. *Multiple imputation for nonresponse in surveys*. Bd. 81. John Wiley & Sons, 2004 (zitiert auf S. 19).

[Rub93] D. B. Rubin. „Statistical disclosure limitation“. In: *Journal of official Statistics* 9.2 (1993), S. 461–468 (zitiert auf S. 19).

[Rud19] C. Rudin. „Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead“. In: *Nature machine intelligence* 1.5 (2019), S. 206–215 (zitiert auf S. 70).

[RWC+19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al. „Language models are unsupervised multitask learners“. In: *OpenAI blog* 1.8 (2019), S. 9 (zitiert auf S. 26).

[SFLK19] L. B. Soares, N. FitzGerald, J. Ling, T. Kwiatkowski. „Matching the blanks: Distributional similarity for relation learning“. In: *arXiv preprint arXiv:1906.03158* (2019) (zitiert auf S. 38).

[SFS+21] A. Seganti, K. Firląg, H. Skowronska, M. Satława, P. Andruszkiewicz. „Multilingual entity and relation extraction dataset and model“. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, S. 1946–1955 (zitiert auf S. 21).

[SHK+23] S. Subramanian, P. Harrington, K. Keutzer, W. Bhimji, D. Morozov, M. Mahoney, A. Gholami. *Towards Foundation Models for Scientific Machine Learning: Characterizing Scaling and Transfer Behavior*. 2023. arXiv: 2306.00258 [cs.LG] (zitiert auf S. 76).

[SM17] H. Surendra, S. MohanH. „A Review Of Synthetic Data Generation Methods For Privacy Preserving Data Publishing“. In: *International Journal of Scientific & Technology Research* 6 (2017), S. 95–101 (zitiert auf S. 19).

[THBN22] Q. Tan, R. He, L. Bing, H. T. Ng. „Document-level relation extraction with adaptive focal loss and knowledge distillation“. In: *arXiv preprint arXiv:2203.10900* (2022) (zitiert auf S. 78).

[Tur50] A. Turing. „Computing machinery and intelligence.“ In: *Mind* (1950) (zitiert auf S. 23).

[VSP+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. „Attention is all you need“. In: *Advances in neural information processing systems* 30 (2017) (zitiert auf S. 24).

[WC18] Y. Wang, D. Chen. „Rising Sino-U.S. Competition in Artificial Intelligence“. In: *China Quarterly of International Strategic Studies* 04.02 (2018), S. 241–258. DOI: 10.1142/S2377740018500148. eprint: https://doi.org/10.1142/S2377740018500148. URL: https://doi.org/10.1142/S2377740018500148 (zitiert auf S. 27).

[WGLY19] Q. Wang, J. Gao, W. Lin, Y. Yuan. „Learning from synthetic data for crowd counting in the wild“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, S. 8198–8207 (zitiert auf S. 20).

[WMR17] S. Wachter, B. Mittelstadt, C. Russell. „Counterfactual explanations without opening the black box: Automated decisions and the GDPR“. In: *Harv. JL & Tech.* 31 (2017), S. 841 (zitiert auf S. 27).

[WQZ+22] H. Wang, K. Qin, R. Y. Zakari, G. Lu, J. Yin. „Deep neural network-based relation extraction: an overview“. In: *Neural Computing and Applications* (2022), S. 1–21 (zitiert auf S. 21).

[WWS+22a] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, D. Zhou. „Self-consistency improves chain of thought reasoning in language models“. In: *arXiv preprint arXiv:2203.11171* (2022) (zitiert auf S. 79).

[WWS+22b] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, D. Zhou. „Chain of thought prompting elicits reasoning in large language models“. In: *arXiv preprint arXiv:2201.11903* (2022) (zitiert auf S. 79).

[XUD+19] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, 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, S. 563–574 (zitiert auf S. 70).

[XW19] W. Xiang, B. Wang. „A survey of event extraction from text“. In: *IEEE Access* 7 (2019), S. 173111–173137 (zitiert auf S. 21).

[ZHM+22] C. Zhou, J. He, X. Ma, T. Berg-Kirkpatrick, G. Neubig. „Prompt consistency for zero-shot task generalization“. In: *arXiv preprint arXiv:2205.00049* (2022) (zitiert auf S. 79).

[ZKZ+15] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, S. Fidler. „Aligning books and movies: Towards story-like visual explanations by watching movies and reading books“. In: *Proceedings of the IEEE international conference on computer vision*. 2015, S. 19–27 (zitiert auf S. 25, 26).

[ZWYJ21] L. Zhuang, L. Wayne, S. Ya, Z. Jun. „A robustly optimized BERT pre-training approach with post-training“. In: *Proceedings of the 20th chinese national conference on computational linguistics*. 2021, S. 1218–1227 (zitiert auf S. 25).

[ZZL+23a] C. Zhang, C. Zhang, C. Li, Y. Qiao, S. Zheng, S. K. Dam, M. Zhang, J. U. Kim, S. T. Kim, J. Choi et al. „One small step for generative ai, one giant leap for agi: A complete survey on chatgpt in aigc era“. In: *arXiv preprint arXiv:2304.06488* (2023) (zitiert auf S. 78, 80).

[ZZL+23b] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong et al. „A survey of large language models“. In: *arXiv preprint arXiv:2303.18223* (2023) (zitiert auf S. 16, 24, 80).

Alle URLs wurden zuletzt am 2. 07. 2023 geprüft.

A. Programmcode

```
import re
import spacy

# Lade die verwendete Pipeline
nlp = spacy.load('de_core_news_lg')

# Die validate-Funktion, die als Argument den Speicherpfad der zu validierenden
 Datei benötigt
def validate(filepath):

    # Lese die Datei
    with open(filepath, "r") as file:
        file_contents = file.read()
    file.close()

  # Definiere Counter, für Werte, wie gemessen werden sollen
    total_entries = 0   # Gesamtzahl
    total_faults = 0    # Gesamte Fehler
    value_errors = 0    # Fehler aufgrund ungültiger Werte
    index_errors = 0    # Fehler aufgrund ungültiger Indizes
    attribute_errors = 0  # Fehler aufgrund ungültiger Attribute

    countingDict = {}   # Dictionary für die Anzahl der Relationen

  # Iteriere über jede Zeile der Datei
    for entry_index, entry in enumerate(file_contents.split("\n"), 1):

        total_entries += 1    # Erhöhe die Gesamtzahl der Einträge

        try:
          # Über den regulären Ausdruck wird die Zeile geparst
            extraction = re.search(r"\{('|\")token('|\"): ?\[(?P<content>('|\")
.+('|\")(, )?\s?)+\], ?('|\")h('|\"): ?\{('|\")name('|\"): ?('|\")(?P<head_name
>.+)('|\"), ?('|\")pos('|\"): ?\[(?P<head_pos_start>\d+), ?(?P<head_pos_end>\d
+)\]\}, ?('|\")t('|\"): ?\{('|\")name('|\"): ?('|\")(?P<tail_name>.+)('|\"),
?('|\")pos('|\"): ?\[(?P<tail_pos_start>\d+), ?(?P<tail_pos_end>\d+)\]\},
?('|\")relation('|\"): ?('|\")(?P<relation>.+)('|\")\}", entry)

      # Der ursprüngliche Satz wird in eine Variable gespeichert
            content = extraction.group("content").split("', '")

            try:
              # Überprüfe die Indizes der head-Entität
```

```
                check_indices(content, extraction.group('head_name'),
extraction.group('head_pos_start'), extraction.group('head_pos_end'))
                # Überprüfe die Indizes der tail-Entität
                check_indices(content, extraction.group('tail_name'),
extraction.group('tail_pos_start'), extraction.group('tail_pos_end'))
                # Bei gültiger Relation wird sie dem Dictionary hinzugefügt
                if extraction.group("relation") in countingDict:
                    countingDict[extraction.group("relation")] = countingDict[
extraction.group("relation")] + 1
                else:
                    countingDict[extraction.group("relation")] = 1

            # Falls bei der Index-Überprüfung ein Value-Error auftritt
            except ValueError as error:
                total_faults += 1   # Erhöhe Gesamtfehlerzahl
                value_errors += 1   # Erhöhe den Value-Error-Counter

            # Falls ein Index-Error auftritt
            except IndexError as error:
                total_faults += 1   # Erhöhe Gesamtfehlerzahl
                index_errors += 1   # Erhöhe den Index-Error-Counter

            # Andere Fehler auftritt werden in der Konsole ausgegeben
            except error:
                print(error)

        # Falls der reguläre Ausdruck die Zeile nicht interpretieren kann
        except AttributeError as error:
            total_faults += 1   # Erhöhe Gesamtfehlerzahl
            attribute_errors += 1 # Erhöhe den Attribute-Index-Counter

    # Ausgabe der Ergebnisse in die Konsole
    print(f"Total Entries: {total_entries}")
    print(f"Total Faults: {total_faults} ({attribute_errors} Attribute Errors /
 {value_errors} Value Errors / {index_errors} Index Errors)")

    for entry in countingDict:
        print(f"{entry}: {countingDict[entry]}")

# Die Methode zur Prüfung von Indizes
# Argumente sind der ursprüngliche Satz, der Name der Entität sowie die Indizes
def check_indices(content, vertexName, vertexPosStart, vertexPosEnd):
    index = int(vertexPosStart) # Initiiere den Index auf die Startposition

    # Iteriere über alle Teile des Namens der Entität
    for part in list(nlp(vertexName)):

      # Keine Aktion, wenn Entitäts-Teil mit Satz-Token übereinstimmt
        if str(part).lower() in content[index].lower():
            pass

        # Sonst Value-Error
        else:
```

```
            raise ValueError(f"{str(part)} not in {content[index]} at index {
index}")
        index += 1

    # Falls Endindexe nicht übereinstimmen: Value-Error
    if index - 1 != int(vertexPosEnd):
        raise ValueError
```

Listing A.1: Das `validate.py`-Skript

```
import re
import spacy
import validation as val
from Levenshtein import distance as levenshtein_distance

# Lade die verwendete Pipeline
nlp = spacy.load('de_core_news_lg')

# Hauptfunktion, die den Pfad der Ausgangsdatei sowie das Zielverzeichnis und
die Anzahl der verwendeten Relationen übergeben bekommt.
def correct(filepath, output_path, nr_relations=10):

    with open(filepath, "r") as file:
        # Read the contents of the file into a variable
        file_contents = file.read()
    file.close()

    if nr_relations == 10:
        relations = ["neu_ceo_von", "ist_ceo_von", "ex_ceo_von", "
neu_vorstand_von", "ist_vorstand_von", "ex_vorstand_von", "
neu_abteilungsleiter_von", "ist_abteilungsleiter_von", "ex_abteilungsleiter_von
", "sonstige"]
  elif nr_relations == 4:
    relations = ["ceo_von", "vorstand_von", "abteilungsleiter_von", "sonstige"]

    total_entries = 0
    total_faults = 0

  # Iteriere über jede Zeile der Ausgangsdatei
    for entry_index, entry in enumerate(file_contents.split("\n"), 1):
        total_entries += 1
        try:
            extraction = re.search(r"\{('|\")token('|\"): ?\[(?P<content>('|\")
.+('|\")(, )?\s?)+\], ?('|\")h('|\"): ?\{('|\")name('|\"): ?('|\")(?P<head_name
>.+)('|\"), ?('|\")pos('|\"): ?\[(?P<head_pos_start>\d+), ?(?P<head_pos_end>\d
+)\]\}, ?('|\")t('|\"): ?\{('|\")name('|\"): ?('|\")(?P<tail_name>.+)('|\"),
?('|\")pos('|\"): ?\[(?P<tail_pos_start>\d+), ?(?P<tail_pos_end>\d+)\]\},
?('|\")relation('|\"): ?('|\")(?P<relation>.+)('|\")\}", entry)

            content = []

            for token in extraction.group("content").split(", "):
                new_tokens = token.split(" ")
```

```
                for new_token in new_tokens:
                    content.append(new_token)

            entry = entry.replace(extraction.group("content"), (", ").join(
content))

            try:      # Prüfe die Indizes der head-Entität
                check_indices(content, extraction.group('head_name'),
extraction.group('head_pos_start'), extraction.group('head_pos_end'))

            except:     # Bei Fehler, leite Korrektur ein
                start_index_new, end_index_new, entity_new = testIndex(
extraction.group('head_name'), content, int(extraction.group('head_pos_start'))
, int(extraction.group('head_pos_end')))
                entry = entry.replace(f"[{extraction.group('head_pos_start')},
{extraction.group('head_pos_end')}]", f"[{start_index_new}, {end_index_new}]")
                entity_new = entity_new.strip("' ")
                entry = re.sub(rf"('|\")t('|\"): \{{('|\")name('|\"): ('|\"){re
.escape(extraction.group('head_name'))}('|\"),", f"'h': {{'name': '{entity_new
}',", entry)

            try:      # Prüfe die Indizes der tail-Entität
                check_indices(content, extraction.group('tail_name'),
extraction.group('tail_pos_start'), extraction.group('tail_pos_end'))

            except:     # Bei Fehler, leite Korrektur ein
                start_index_new, end_index_new, entity_new = testIndex(
extraction.group('tail_name'), content, int(extraction.group('tail_pos_start'))
, int(extraction.group('tail_pos_end')))
                entry = entry.replace(f"[{extraction.group('tail_pos_start')},
{extraction.group('tail_pos_end')}]", f"[{start_index_new}, {end_index_new}]")
                entity_new = entity_new.strip("' ")
                entry = re.sub(rf"('|\")t('|\"): \{{('|\")name('|\"): ('|\"){re
.escape(extraction.group('tail_name'))}('|\"),", f"'t': {{'name': '{entity_new
}',", entry)

            # Ist die angegebene Relation nicht vorgesehen, kann sie manuell ge
ändert werden
            if extraction.group("relation") not in relations:
                real_relation = input()

            with open(output_path, "a") as file:
                file.write(f"{entry}\n")
            file.close()

        # Falls der reguläre Ausdruck die Zeile nicht interpretieren kann
        except AttributeError as error:
            total_faults += 1
            print(f"ATTRIBUTE ERROR at {entry_index}")

        # Falls ein anderer Fehler auftritt
        except:
            total_faults += 1
```

```
            print(f"UNIDENTIFIABLE FAULT at {entry_index}")

# Suche die korrekten Indizes für die Entität. Argumente sind der ursprüngliche
 Satz, der Name der Entität, sowie die ursprünglichen, fehlerhaften Indizes
def testIndex(search, text, startIndex, endIndex):

    startIndexNeu = startIndex
    entity = search
    search_parts = list(nlp(search))

    if startIndex > len(text):
        startIndex = len(text)

    rnge = 0    # "Range", mit der um das ursprüngliche Wort gesucht wird
    found_flag = False    # Semaphor, der anzeigt ob die Suche erfolgreich war

    if str(text[startIndex]) in str(search_parts[0]) or str(search_parts[0]) in
 str(text[startIndex]):
        pass

    else:     # Wenn der Startindex nicht stimmt

      # Es wird bis zu 4 Token vor und nach dem ursprünglichen Token gesucht
        while rnge < 5 or not found_flag:

          # Wenn der Range nicht das Ende des Satzes überschreitet
            if (startIndex + rnge) < len(text):

              # Wenn ein Treffer existiert, erneuere Start- und Endindex
                if (str(text[startIndex + rnge]) in str(search_parts[0])
                  or str(search_parts[0]) in str(text[startIndex + rnge])):
                    startIndexNeu = startIndex + rnge
                    endIndex = startIndexNeu
                    found_flag = True

                # Wurde kein Treffer gefunden, prüfe mit Levenshtein-Distanz
                elif levenshtein_distance(str(text[startIndex + rnge]).strip("'
\""), str(search_parts[0])) <= 1:
                    startIndexNeu = startIndex + rnge
                    endIndex = startIndexNeu
                    found_flag = True

            # Analog für den Anfang des Satzes
            if (startIndex - rnge) >= 0:
                if (str(text[startIndex - rnge]) in str(search_parts[0])
                  or str(search_parts[0]) in str(text[startIndex - rnge])):
                    startIndexNeu = startIndex - rnge
                    endIndex = startIndexNeu
                    found_flag = True

                elif levenshtein_distance(str(text[startIndex - rnge]).strip("'
\""), str(search_parts[0])) <= 1:
                    startIndexNeu = startIndex - rnge
```

```
                    endIndex = startIndexNeu
                    found_flag = True
            rnge += 1

    entity = ""   # Zurücksetzen des Entitäts-Namens

    # Überprüfe die folgenden Token im Entität-Namen
    for index in range(0, len(search_parts)):

      # Prüfe, ob der Text direkt oder mit Levenshtein der Suche entspricht
        if (str(text[startIndexNeu + index]) in str(search_parts[index])
          or str(search_parts[index]) in str(text[startIndexNeu + index])
          or levenshtein_distance(str(text[startIndexNeu + index]).strip("' \""
), str(search_parts[index])) <= 1):
            endIndex = startIndexNeu + index    # Aktualisiere End-Index

            # Erstelle den Namen der Entität aus den Token im Text
            entity = entity.strip("' \"") + " " + str(text[startIndexNeu +
index]).strip("' \"")

        # Wenn das letzte Wort im Satz erreicht ist, breche nach der Aktion ab
        elif startIndexNeu + index > len(text) - 1:
            entity = entity.strip("' \"") + " " + str(text[startIndexNeu +
index]).strip("' \"")
            endIndex = startIndexNeu + index
            break
        else:
            endIndex = startIndexNeu + index

    return startIndexNeu, endIndex, entity

# Die Methode zur Prüfung von Indizes (analog in validate.py)
# Argumente sind der ursprüngliche Satz, der Name der Entität sowie die Indizes
def check_indices(content, vertexName, vertexPosStart, vertexPosEnd):

    index = int(vertexPosStart)

    for part in list(nlp(vertexName)):

        if str(part).lower() in content[index].lower():
            pass

        else:
            raise ValueError(f"{str(part)} not in {content[index]} at index {
index}")
        index += 1

    if index - 1 != int(vertexPosEnd):
        raise ValueError
```

Listing A.2: Das `correction.py`-Skript

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift