Hochschule
Bonn-Rhein-Sieg
University
of Applied Sciences

Fachbereich Informatik
Department of Computer Sciences

Master Thesis

Erklärbare Künstliche Intelligenz durch die Kombination von LLMs und Ontologien

von
Richard Bernhard Franken

Studiengang Master Informatik

Erstprüfer: Prof. Dr. Andreas Hense
Zweitprüfer: Prof. Dr. Matthias Bertram

11. September 2025

1 Vorwort

1.1 Eidesstattliche Erklärung

Name: _____

Adresse: _____

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbst angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Sankt Augustin, den 11. September 2025

Unterschrift: _____

1.2 Erklärung zur Nutzung von geschlechtergerechter Sprache

In dieser Arbeit erfolgt die Verwendung von geschlechtergerechter Sprache durch die Nutzung von geschlechtsunabhängigen Plural-Formen, oder der Verwendung von sowohl der maskulinen, als auch der femininen Plural-Form. So werden Begriffe wie „Mitarbeiter“ durch „Mitarbeitende“ oder „Mitarbeiter und Mitarbeiterinnen“ ersetzt. In dem Fall, dass das generische Maskulin erscheint, ist dies auf die Tatsache zurückzuführen, dass die angesprochenen Personengruppen lediglich aus männlichen Mitgliedern bestehen. Auf die Verwendung von Sonderzeichen zur Kennzeichnung von geschlechtergerechter Sprache, wie „*“ oder „:“ wird zugunsten des Leseflusses bewusst verzichtet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AI/KI Artificial Intelligence/Künstliche Intelligenz

CLM Causal Language Modelling

CoT Chain-of-Thought

CWA Closed World Assumption

DL Description Logic

F1 F1-Score (Harmonisches Mittel aus Precision und Recall)

FZQ Fehlerzahlquote (Anteil fehlerhafter OWL-Aussagen an allen generierten Aussagen)

ICD-10 Internationale statistische Klassifikation der Krankheiten

JSON JavaScript Object Notation

JSONL JSON Lines

LLM Large Language Model

ML Machine Learning

MLM Masked Language Modelling

NLP Natural Language Processing

OWA Open World Assumption

OWL Web Ontology Language

RAG Retrieval-Augmented Generation

RDF Resource Description Framework

RDFS RDF Schema

SD Standardabweichung

SPARQL SPARQL Protocol and RDF Query Language

TTL Turtle

XAI Explainable AI

2

Abstract

Diese Arbeit untersucht, inwiefern sich die Erklärbarkeit von Large Language Models (LLMs) durch die Kombination mit Ontologien verbessern lässt. Im Kontext von wachsenden Anforderungen an erklärbare KI-Systeme wird dabei analysiert, ob und wie LLMs in der Lage sind, auf Basis natürlichsprachiger Eingaben komplexe, konforme Axiome in Web Ontology Language (OWL) Turtle Syntax zu generieren. Hierzu wurden drei domänenspezifische Ontologien unterschiedlicher Komplexität entwickelt: Eine Tierklassifikation, eine historische Familienontologie zur Dynastie der Habsburger sowie eine medizinische Ontologie.

Die Arbeit verfolgt einen experimentellen Ansatz mit insgesamt vier verschiedenen Prompt Engineering Strategien: Zero-Shot, Few-Shot, Retrieval-Augmented Generation (RAG) und Finetuning. Für jede Ontologie wurden zehn natürlichsprachliche Fakten-Dokumente manuell zu OWL-Referenzdokumenten konvertiert und anschließend mit LLM-Ausgaben auf Basis der gleichen natürlichsprachlichen Dokumente und den gegebenen Ontologien verglichen. Die Ergebnisse wurden syntaktisch über OWL-Validatoren, semantisch quantitativ anhand einer manuellen Prüfung und etablierten Metriken (Precision, Recall, F1-Score, Jaccard-Index) sowie einer qualitativen Fehlerklassifikation evaluiert.

Die Ergebnisse der Arbeit zeigen, dass LLMs grundsätzlich auch komplexere OWL-Axiom-Mengen erzeugen können, die semantisch und syntaktisch weitgehend korrekt sind. Insbesondere unter Zugabe von Kontextinformationen, vor allem durch RAG, werden hier signifikante Verbesserungen in syntaktischer, semantischer und logischer Konsistenz identifiziert. Zudem konnten durch Anwendung von RAG bestimmte Fehlerklassen vollkommen vermieden werden. Dennoch bleiben systematische Fehler bestehen, etwa bei der Zuordnung korrekter Property-Werte oder der Generierung sämtlicher relevanter Fakten.

Die Arbeit belegt, dass LLMs unter Verwendung der richtigen Strategie das Potenzial bieten, strukturiert in Ontologien eingebettet zu werden, um durch eine Kombination der beiden Systeme die Erklärbarkeit von LLM-Ausgaben zu steigern und Bedienbarkeit sowie Geschwindigkeit in der Ontologie-Erstellung durch LLMs zu verbessern. Dies schafft eine Grundlage für zukünftige hybride, nachvollziehbare und effiziente KI-Systeme.

3
Einleitung

3.1 Hintergrund und Motivation

Die Digitalisierung von Organisationen und Unternehmen wird mit steigendem Digitalisierungsindex auch in Deutschland zu einem immer wichtigeren Thema [7]. Dazu gehört auch die Integration von künstlicher Intelligenz (KI) in Prozesse und alltägliche Arbeitsabläufe in Unternehmen. In den letzten Jahren haben künstliche Intelligenz und darunter vor allem Large Language Models (LLMs) wie OpenAIs GPT, LLaMA oder DeepSeek große Fortschritte in ihrer Entwicklung und ihrer Popularität gemacht, was sich auch in der stetig wachsenden Größe des Marktes dieser Modelle widerspiegelt [31]. Sie werden dabei in verschiedensten Anwendungsfällen eingesetzt, wie die automatisierte Generierung von Texten, der Entscheidungsunterstützung, der Validierung von Arbeitspaketen oder weiteren Disziplinen im Bereich Natural Language Processing (NLP) [5].

Aufgrund ihrer mittlerweile enormen Anzahl von Parametern und der Datenmengen, auf denen die Modelle trainiert worden sind, erzielen diese bereits beeindruckende und größtenteils auch präzise Ergebnisse. So ist GPT-4o beispielsweise bereits in der Lage, Prüfungen aus unterschiedlichen akademischen Themengebieten in verschiedenen Sprachen mit durchschnittlich über 80 Prozent korrekten Antworten zu bearbeiten [52]. Was aber trotz dieser Weiterentwicklungen ein zentrales Problem darstellt, ist die mangelnde Erklärbarkeit dieser Modelle. Klassische symbolische KI-Ansätze verwenden regelbasiertes NLP und Wörterbücher zur Verarbeitung von natürlichsprachlichen Texten, wodurch die getroffenen Entscheidungen nachvollziehbar sind [30]. Moderne Ansätze, wie LLMs, die durch neuronale Netze über mehrere Schichten (Deep Learning) Parameter in gewichteten Matrizen speichern [74] und mithilfe ihrer Transformer-Modelle die natürlichsprachigen Eingaben verarbeiten, sind allerdings als Black-Box-Modelle strukturiert [27]. Die genauen Abläufe der internen Methoden dieser Modelle, die zu den Ergebnissen führen, lassen sich also nicht nachvollziehen.

Dieses Verfahren ermöglicht zwar, mit bereits hoher Vorhersagegenauigkeit Texte zu verarbeiten, für die keine konkreten Regeln definiert sein müssen [23], die interne Entscheidungslogik bleibt dabei aber nicht nachvollziehbar. Dadurch erhalten Nutzer und Nutzerinnen zwar Antworten auf ihre Fragen, jedoch ohne dabei zu wissen, wie das Sprachmodell zu der Schlussfolgerung gelangt ist. Besonders in sicherheitskritischen Sektoren, wie Medizin, Recht oder Finanzen, kann dies zu drastischen Problemen führen, wenn keine Rückschlüsse auf die Entscheidungsfindung getroffen werden und auch verantwortliche Personen oder Systeme nicht identifiziert werden können [47].

3.2 Problemstellung

Vor diesem Hintergrund steigt der Bedarf nach interpretierbaren und erklärbaren Methoden. Der stetig steigende Einsatz von LLMs für reale, teils kritische Anwendungsfälle und Entscheidungen in Unternehmen aufgrund ihrer breitgefächerten Leistungsfähigkeit wird daher zunehmend auf technischer und ethischer Ebene kritisiert [19]. Das nicht bestehende Verständnis für die Entscheidungsmechanismen von LLMs birgt vor allem in sensiblen Bereichen Risiken für falsche oder irreführende Antworten [66]. Hinzu kommen zudem weitere Probleme, die bei aktuellen Sprachmodellen auftreten, wie das Prinzip des ”Model collapse”. Durch das Veröffentlichen von KI-gestützten Inhalten im Internet und das rekursive Verwenden dieser als Trainingsdaten für neuere Modelle werden Inkonsistenzen und Fehler älterer Modelle fortschreitend integriert. Diese Fehler werden so zu inhärenten Bestandteilen der Wissensbasis, sind mit jeder Modelliteration schwerer zu erkennen für Nutzende und noch schwerer aus dem Modell zu entfernen [69].

Eine Möglichkeit, um die Erklärbarkeit von Schlussfolgerungen zu gewährleisten, ist die Verwendung von Ontologien. Durch ihre Wissensrepräsentation anhand von Klassen, Relationen und Regeln lassen sich Schlussfolgerungen in Ontologien auf Basis von logischer Inferenz durchführen [25]. Durch eine Kombination von LLMs und Ontologien mithilfe einer formalen Sprache wie der Web Ontology Language (OWL) könnten die Sprachmodelle in ihrer Schlussfolgerung unterstützt und so transparenter gestaltet werden. Durch eine Integration von Ontologien ließen sich Anfragen an LLMs durch die formalen Axiome stützen. So kann über die Generierung von OWL-Fakten durch ein LLM basierend auf einer gegebenen Ontologie eine nachvollziehbare und überprüfbare Wissensrepräsentation aufgesetzt werden. Dabei könnte zugleich eine der großen Hürden von Ontologien, der manuelle zeitliche Aufwand der Faktenerzeugung, der die Skalierbarkeit und Anwendbarkeit von Ontologien einschränkt, drastisch reduziert werden. Das potenzielle Verständnis von Sprachmodellen, Ontologien zu interpretieren und konsistente Fakten zu erzeugen, könnte so die Geschwindigkeit der Faktenerzeugung durch LLMs mit der Erklärbarkeit von Ontologie-basierten Systemen kombinieren. Dadurch lassen sich hybride, erklärbare Anwendungen für Entscheidungsunterstützung in kritischen Domänen ermöglichen.

Allerdings bestehen noch offene Fragen bezüglich der praktischen Umsetzung einer solchen Kombination. Es ist unklar, ob LLMs tatsächlich in der Lage sind, solche OWL-Axiome einer gegebenen Ontologie korrekt zu interpretieren und basierend auf diesen natürlichsprachige Informationen präzise in formale OWL-Fakten für diese Ontologie zu übersetzen. Dabei ist zudem sicherzustellen, dass überprüft wurde, inwieweit die generierten Fakten syntaktisch und semantisch korrekt sind. Bei erfolgreicher Erstellung von OWL-Fakten ließen sich diese in die bestehende Ontologie integrieren und die Grundlage für die Kombination von LLMs und Ontologien zur Verbesserung von erklärbarer künstlicher Intelligenz wäre gegeben.

3.3 Fragestellung

Um diesen offenen Punkten auf den Grund zu gehen, stellt sich innerhalb dieser Arbeit die zentrale Forschungsfrage: Inwiefern kann die Integration von Ontologien mit LLMs die Erklärbarkeit von großen Sprachmodellen verbessern? Darauf basierend ergeben sich Unterfragen für die verschiedenen offenen Punkte, die zur Überprüfung der Funktionalität beantwortet werden müssen.

1. Interpretationsfähigkeit von LLMs für Ontologien
    - Inwieweit sind LLMs in der Lage, formale Axiome einer gegebenen Ontologie in OWL zu interpretieren und auch korrekt zu verarbeiten?
    - Welche Herausforderungen bestehen bei der semantischen und syntaktischen Verarbeitung von Ontologien durch LLMs?
2. Extraktion von formalen Fakten aus Textkorpora
    - Können LLMs natürlichsprachliche Informationen aus Textkorpora ableiten und in OWL-Fakten umwandeln?
    - In welchem Maße sind die generierten Fakten syntaktisch und semantisch korrekt, sowie vollständig?
    - Welche unterschiedlichen Ergebnisse ergeben sich durch das Anwenden verschiedener Prompt Engineering Strategien?
3. Limitierungen und Herausforderungen
    - Welche grundlegenden Einschränkungen bestehen bei der Kombination von LLMs mit Ontologien?
    - Wie unterscheidet sich die Leistung der LLMs in Abhängigkeit zu verschiedenen Komplexitätsstufen von Ontologien?
    - Gibt es systematische Fehler und Verzerrungen der Ergebnisse?
    - Welche Verbesserungsmöglichkeiten oder Alternativen lassen sich aus den Ergebnissen ableiten?

Eine Beantwortung dieser Fragen ermöglicht eine fundierte Analyse der Kombination von LLMs und Ontologien und leistet so einen Beitrag zur Forschung im Bereich Explainable AI (XAI). Die gewonnenen Erkenntnisse können dabei unterstützen, zukünftige Methoden zu implementieren und die praktische Anwendbarkeit der Kombination zu verbessern oder durch technische Implementierungen umzusetzen.

3.4 Ziele und Aufbau der Arbeit

Das Hauptziel dieser Arbeit ist es, die Erklärbarkeit von LLMs durch die Kombination mit Ontologien zu untersuchen. Dabei gilt es, mit entwickelten Ontologien systematisch zu analysieren, ob und inwiefern Ontologien dazu beitragen können, die Entscheidungsprozesse von LLMs transparenter und nachvollziehbar zu gestalten. Daraus leiten sich die folgenden Unterziele ab.

1. Eine Analyse des aktuellen Forschungsstands im Bereich XAI und der Kombination von Ontologien und LLMs.
2. Die standardisierte Untersuchung der Interpretationsmöglichkeiten von OWL-Korpora und der Umwandlung von natürlichsprachigen Texten für unterschiedlich komplexe Ontologien in konsistente OWL-Fakten.
3. Eine Diskussion der Ergebnisse und eine Ausarbeitung aktueller Grenzen und Herausforderungen, um den Rahmen für zukünftige Forschung und Entwicklung zu erarbeiten.

Der Aufbau der Arbeit teilt sich dabei in mehrere Kapitel, die sowohl theoretische als auch praktische Aspekte der Zielsetzung abdecken. Zunächst folgen die theoretischen Grundlagen dieser Arbeit, die die Basis für das thematische Verständnis und die weitere Durchführung bilden. Zu diesen Grundlagen zählen die Konzepte von künstlicher Intelligenz, XAI, LLMs und Methoden im Prompt Engineering. Zudem zählen zu den Grundlagen auch Ontologien, OWL sowie das in dieser Arbeit verwendete Ontologie-Framework „Protégé“. Nach diesen konzeptionellen Grundlagen folgt eine Einordnung in den aktuellen Forschungsstand zu bestehenden Arbeiten über die Erklärbarkeit von LLMs sowie zu Ansätzen in Theorie und Praxis, die sich mit der Kombination von Sprachmodellen und Ontologien auseinandersetzen.

Danach gilt es, das methodische Vorgehen zu konkretisieren und zu erläutern, welche Ontologien als Basis für die Evaluierung erstellt werden sowie die konkrete Interaktion dieser mit den LLMs. Zudem werden die verwendeten Prompt Engineering Strategien genauer erläutert und es werden auch die verwendeten Evaluierungsmethoden konkretisiert und verständlich gemacht. Auf dieser Basis kann die Durchführung der Experimente erfolgen. Konkret werden die Ontologien ausgearbeitet und es werden auf Basis dieser Ontologien manuell zwei verschiedene Arten von OWL-Dokumenten erstellt. Einerseits Dokumente, die als Trainingskorpora für die Anwendung im Prompt Engineering dienen und andererseits Vergleichsdokumente für die Evaluation der generierten Artefakte.

Sind diese Dokumente und die Ontologien aufgesetzt, kann die Durchführung der Generierung von OWL-Dokumenten für die verschieden komplexen Ontologien unter Verwendung unterschiedlicher Prompt Engineering Strategien erfolgen. Die erzielten Ergebnisse dieser Experimente können dann unter Verwendung der festgelegten Evaluationsmethoden ausgewertet und anhand dieser die Qualität der LLM-Ontologie-Interaktionen anschließend bewertet werden.

Im Anschluss erfolgt eine Diskussion, in der die Ergebnisse dieser Arbeit und der Evaluation kritisch betrachtet werden, abgeleitete Herausforderungen ausgearbeitet werden und die Richtung für mögliche zukünftige Forschung in dem Gebiet der Kombination von LLMs und Ontologien abgeleitet wird. Abschließend werden die zentralen Erkenntnisse der Arbeit zusammengefasst und es wird ein Fazit gezogen, inwiefern mit den Ergebnissen dieser Arbeit die Forschungsfragen beantwortet werden konnten.

4

Grundlagen

Um die Erklärbarkeit von LLMs durch eine Kombination mit Ontologien zu untersuchen, ist es wichtig, zunächst ein grundlegendes Verständnis der zugrunde liegenden Konzepte dieser Arbeit bereitzustellen. Dieses Kapitel gibt einen Überblick über die essenziellen theoretischen Grundlagen, die für die weitere Arbeit relevant sind. Hierzu wird die Künstliche Intelligenz als Forschungsbereich eingeführt und anschließend folgen Einführungen in Teilbereiche von KI mit LLMs und XAI, die eine zentrale Bedeutung für diese Arbeit haben. Schlussendlich folgen Ontologien und ihre Rolle in der Wissensrepräsentation sowie Grundlagen zu den Konzepten OWL und dem Ontologie-Framework Protégé.

4.1 Künstliche Intelligenz

Künstliche Intelligenz ist ein interdisziplinäres Forschungsgebiet aus den Bereichen der Informatik und der Philosophie, das sich mit der Entwicklung von technischen Systemen und Algorithmen befasst, die in der Lage sind, kognitive Fähigkeiten nachzuahmen. Der Begriff ist in den 1950er Jahren durch John McCarthy geprägt worden, der Logik als zentrale Methode zur Repräsentation von Informationen im Speicher präsentiert hat [44]. Das Forschungsgebiet umfasst dabei mehrere Teilbereiche und auch die Ansätze für KI-Systeme haben sich historisch über Phasen hinweg zu den modernen Methoden hin entwickelt.

In den frühen Jahren der Künstlichen Intelligenz bis in die 1970er sind zunächst Wörterbuch- und regelbasierte Systeme entwickelt worden. Wörterbuch-basierte Systeme wie das von dem Informatiker Joseph Weizenbaum am Massachusetts Institute of Technology entwickelte Dialog-System ELIZA, verwenden einen Thesaurus (Synonymdatenbank), um bestimmte Kernbegriffe in Nutzereingaben zu erkennen und passende Antworten zu liefern [80]. So imitiert ELIZA beispielsweise eine Psychotherapeutin, die nach bekannten Behandlungsmethoden Fragen stellt und auf Nutzereingaben eingeht, um den Dialog dynamisch in eine zielführende Richtung zu leiten. Die regelbasierten Systeme sind Systeme, die aus einer Datenbank von Fakten aufgebaut sind. Dazu kommt auch eine Menge von Wenn-Dann-Regeln, die hinterlegt werden und ergänzt, entfernt und bearbeitet werden können, sowie ein Interpreter [24]. Durch diesen Aufbau können die Systeme über die Regeln (wenn vorhanden) zu den Nutzereingaben passende Antworten liefern. Diese beiden Ansätze von KI-Systemen sind vor allem für sogenannte Expertensysteme verwendet worden. Das sind Anwendungen, die innerhalb einer bestimmten Domäne bei der Problemlösung unterstützen, indem sie Handlungsempfehlungen ableiten. Auch erklären diese Systeme die Schlussfolgerung basierend auf den Regeln. So unterstützen diese Systeme bereits seit Jahrzehnten in domänenspezifischen Problemstellungen, wie beispielsweise der Diagnose von Infektionskrankheiten [45].

Das Aufsetzen und Pflegen von solchen Expertensystemen ist allerdings vor allem bei komplexeren Domänen und Anwendungsfällen eine aufwändige Arbeit, was mit zu einem Rückgang von Investitionen und dem ”AI-Winter”, also einer Durststrecke, geführt hat [20]. Die darauffolgende vertiefte Forschung im Bereich Machine Learning ermöglicht die Unterstützung bei Problemlösungen ohne den Bedarf des manuellen Programmierens fester Regeln. Stattdessen werden Lernalgorithmen mit markierten Datensätzen trainiert, um nach dem Durchlaufen der Datensätze neue Eingaben auf die markierten Merkmale zu untersuchen (Supervised Learning) oder die Algorithmen untersuchen nicht-markierte Datensätze, um Muster oder Cluster zu erkennen (Unsupervised Learning) [63].

Als eine Unterkategorie des Machine Learning bzw. des Supervised Learning hat sich Deep Learning herauskristallisiert. Diese arbeitet mit mehreren Schichten zur Eingabe, Verarbeitung und Ausgabe von Daten. Dabei verarbeiten ”Neuronen” innerhalb einer Schicht als Knoten im Netz Eingaben in Form von Vektoren, beispielsweise durch Wortvektoren oder Bildpixel und multiplizieren diese mit einer Gewichtungsmatrix, die im Rahmen des überwachten Lernens entstanden ist und stetig angepasst wird. Wenn ein Modell trainiert wird, wird das Ergebnis mit dem erwarteten Ergebnis abgeglichen und das entstandene Delta (Loss) wird verwendet, um die Gewichtung für Knoten anzupassen. Damit wird der Verlust minimiert. Das Ergebnis von Berechnungen eines Knotens im neuronalen Netz wird an die nächste Schicht weitergegeben [23]. Von diesem Framework aus haben sich Kategorien von Deep Learning in neuronalen Netzen für unterschiedliche Anwendungsfälle entwickelt. Neben Natural Language Processing (NLP), also dem Verarbeiten von natürlicher Sprache, wird es auch im Bereich Computer Vision für die Bild- und Videoverarbeitung und auch im Bereich Robotik eingesetzt [72]. Auf diese weiteren Bereiche neben NLP wird innerhalb dieser Arbeit allerdings nicht weiter eingegangen.

Besonders im Bereich NLP haben neuronale Netze aber noch Schwachstellen, wie eine langsame Berechnung, da die Wörter sequenziell verarbeitet werden, und Schwierigkeiten bei der Verarbeitung von langen Texten. Um diese Probleme zu beheben, haben Informatiker bei Google 2017 die sogenannten ”Transformermodelle” präsentiert, die parallele Verarbeitung ermöglichen und durch den Self-Attention-Mechanismus den Kontext zwischen zwei Wörtern unabhängig von ihrer Position im Text miteinander abgleichen [74]. Diese Verlagerung von manuellen Wörterbuch- und regelbasierten Ansätzen über maschinelles Lernen hin zu neuronalen Netzen und den effizienteren Transformer-Modellen ermöglicht die Entwicklung von LLMs, allerdings ist durch die zugrundeliegenden mathematischen Algorithmen neuerer Modelle der Aspekt der Erklärbarkeit verloren gegangen.

4.2 Explainable AI

Explainable AI (XAI) ist ein Forschungsbereich der Künstlichen Intelligenz, der sich mit der Nachvollziehbarkeit und der Steigerung der Transparenz von KI-Modellen beschäftigt. Ziel ist, die bislang intransparenten Entscheidungsprozesse von modernen KI-Modellen für Menschen verständlicher und nachvollziehbarer zu gestalten, um das Vertrauen in KI-gestützte Entscheidungsprozesse zu verbessern. Das ist vor allem für sicherheitskritische Sektoren, wie medizinische oder finanzielle Bereiche von Relevanz [27]. Der Bedarf nach diesem Konzept ist über die letzten Jahre durch die steigende Verwendung von LLMs für Entscheidungen verstärkt worden [47].

Die Gründe für den Bedarf nach Erklärbarkeit sind dabei vielschichtig und lassen sich kategorisch in vier Dimensionen einteilen. Rechtfertigung ist die erste Dimension. Auch moderne LLMs machen Fehler in der Generierung von Texten und wenn fehlerhafte Ergebnisse zur Entscheidungsfindung verwendet werden, besteht der Bedarf daran, die getroffene Entscheidung rechtfertigen zu können. Auch präventiv spielt Erklärbarkeit bei der Kontrolle in frühen Phasen eine Rolle. So können Fehler im Modell bereits während einer Testphase von Anwendenden durch Debugging identifiziert werden. Schlussfolgerungen nachvollziehen zu können, ermöglicht zudem auch das kontinuierliche Verbessern des Modells als dritte Dimension. Wenn Anwendende wissen, weshalb ein bestimmtes Ergebnis aufgetreten ist, kann das Modell leichter angepasst werden, um die Ergebnisse zu verbessern. Die letzte Dimension ist das Entdecken. Wenn ein Modell in der Lage ist den Benutzenden den Lösungsweg zu erläutern, unterstützt dies den Menschen im Lernprozess für verschiedenste Problemstellungen [1].

Zur Verbesserung der Erklärbarkeit von KI-Modellen gibt es zwei Ansätze. Intrinsische Methoden und Post-hoc-Methoden. Intrinsische Modelle sind Methoden für KI-Modelle, die eine im Modell integrierte Erklärbarkeit ermöglichen. Zu diesen Methoden zählen unter anderem symbolische KI-Ansätze, die mit logischer Inferenz Schlussfolgerungen treffen, oder auch Modelle, die Methoden wie lineare Regression zur intrinsischen Nachvollziehbarkeit verwenden [1]. Die andere Art sind die Post-hoc-Methoden, welche die Ergebnisse von einem Modell im Nachhinein analysieren, um eine Erklärung zu liefern. Darunter fallen Methoden wie Feature Prediction durch Frameworks wie SHAP, die die Wichtigkeit bestimmter Merkmale eines Modells für Entscheidungen anzeigen können [38]. Andere Post-hoc-Methoden verwenden visuelle Interpretation, um so beispielsweise in der Bildgenerierung anzuzeigen, welche Abschnitte des neuronalen Netzes für welche Bildbereiche maßgeblich ausschlaggebend sind [70]. Auch ohne direkten Einblick in die Funktionsweise kann die Erklärbarkeit für Anwendende durch promptbezogene Erläuterungen auf fachlicher Flughöhe durch Post-hoc-Methoden ermöglicht werden. Dabei werden Erklärungen generiert, die zeigen, welche Ergebnisse bei leichter Abänderung des originären Prompts entstehen würden, um so kausale Zusammenhänge aufzuzeigen [77].

Ein vielversprechender Ansatz für XAI ist die Integration von Ontologien. Hier ist unter anderem zu klären, wie komplex die durch eine Ontologie beschriebenen Sachverhalte sein können, um dennoch präzise Ergebnisse zu liefern. Um herauszufinden, ob und in welchem Maße LLMs mit Ontologien kombiniert werden können, ist es wichtig ein Verständnis dafür zu haben, was LLMs sind und wie diese funktionieren.

4.3 Large Language Models

Large Language Models sind eine Art von Deep Neural Network, die für die Verarbeitung natürlicher Sprache verwendet werden. LLMs basieren auf der zuvor genannten Transformer-Architektur [74]. Sie können große Textmengen verarbeiten und darauf basierend menschenähnliche Texte zu generieren. Bekannte Beispiele für große Sprachmodelle sind GPT von OpenAI [53], LLaMA von Meta [36], Gemini von Google [22] oder auch DeepSeek [13]. Durch die Entwicklung kommerzieller und auch Open Source-basierter Sprachmodelle sind LLMs über die letzten Jahre öffentlich zugänglich geworden. So können diese Modelle sowohl von Endnutzenden als auch über einen API-Zugang verwendet werden. Die Modelle haben für gewöhnlich eine Dialog-basierte Oberfläche.

Neben der Transformer-Architektur basieren die Sprachmodelle auf einer Reihe von Schlüsseltechnologien, die den Umfang und die Zuverlässigkeit in ihrer Ausführung ermöglichen. Sie verwenden Tokenisierung. Dabei handelt es sich um ein Konzept aus dem Bereich NLP, bei dem zu verarbeitende Texte in kleinere Einheiten zerlegt und daraufhin vom Modell verarbeitet werden [81]. Dabei unterteilen verschiedene Modelle die Textbestandteile in unterschiedlich viele Tokens, was bei einer Erhöhung der Token-Zahl zu genaueren Ergebnissen führt, sich aber auch in einer vergrößerten benötigten Rechenleistung abbildet. Daher wird die Bepreisung für die Verwendung der APIs dieser Modelle auch sowohl für Trainingsvorgänge als auch für die Generierung von Output anhand der verbrauchten Tokens festgesetzt [59]. Das Trainieren dieser Modelle erfolgt durch Pre-Training, bei dem die LLMs mit einer großen Menge an Textkorpora trainiert werden [14], gefolgt von Finetuning, einem zusätzlichen Training eines Modells durch anwendungsspezifische Daten, um es für bestimmte Anwendungsfälle zuzuschneiden.

Um durch das Pre-training trainiert zu werden, verwenden LLMs dabei nicht Supervised Learning, da die Menge an natürlichsprachigen Daten, die für das Training verwendet wird, zu groß ist, als dass ein manuelles Klassifizieren der Daten rentabel wäre. Daher verwenden die Modelle Self Supervised Learning, eine Machine Learning Methode, bei der das Modell unbeschriftete Daten als Input erhält und basierend auf den Daten Aufgaben ausführt, um selbstständig Labels für die Daten zu generieren, wodurch kein manuelles Klassifizieren der Daten mehr nötig ist [15]. Konkret verwenden LLMs dabei verschiedene Techniken des Self Supervised Learning. Das GPT-Modell und LLaMA verwenden beispielsweise Causal Language Modelling (CLM). Bei dieser Technik werden die Texte in Abschnitten als Input gegeben. Das Modell muss daraufhin das nächste Wort im Satz generieren, woraufhin überprüft wird, ob das generierte Wort übereinstimmt. Dieses Vorgehen wird für den restlichen Text und eine große Menge weiterer Textkorpora fortgeführt. Systeme wie BERT verwenden stattdessen Masked Language Modelling (MLM), eine ähnliche Technik, bei der aber nicht das nächste Wort im Text das Label ist, sondern innerhalb eines Textes bestehende Wörter mit einer Maske versehen werden. Das Modell muss dann die korrekten Begriffe für die maskierten Textstellen generieren [46]. Beide Techniken haben ihre Vor- und Nachteile und ermöglichen beide das Trainieren von Sprachmodellen zur Generierung von kontextverständlichen Texten.

LLMs finden dabei Verwendung in verschiedenen Anwendungsfällen. So ermöglichen sie neben der Generierung von Texten für Artikel und Geschichten mittlerweile auch die Generierung von Source Code. Sie können verwendet werden, um maschinell lange Texte von einer Sprache in eine andere zu übersetzen [3], oder auch um automatisierte Chatbots oder virtuelle Assistenten aufzusetzen, die kontextbezogen Fragen beantworten können. Sie werden zudem auch nicht nur zur reinen Textgenerierung oder Kommunikation eingesetzt, sondern auch zur Textklassifikation oder auch Textanalyse. So können sie beispielsweise in der Biomedizin darauf trainiert werden, Fragen zu einem bestimmten Fachgebiet zu beantworten oder auf Anfrage Diagnosevorschläge für Krankheitsbilder zu generieren [40]. Damit erfüllen sie eine Aufgabe, für die auch die klassischen Expertensysteme zum Einsatz gekommen sind.

Die Qualität der Ergebnisse eines LLMs hängt allerdings nicht ausschließlich von den für das Training verwendeten Daten, dem Pre-training selbst und der verwendeten Technik

hierfür ab, sondern auch von anderen Parametern. Wie zuvor erwähnt, sind LLMs für Endnutzende über ein Dialog-UI dargestellt. Ein wichtiger Aspekt für die Qualität ist daher auch der Aufbau, der Inhalt und der Detailgrad der Anfragen, die über diese Oberfläche an das Modell übergeben werden. Diese Nutzer-Anfragen werden auch als Nutzer-Prompt oder Userprompt bezeichnet. Um die durch den Prompt gelieferten Ergebnisse zu optimieren, gibt es verschiedene Strategien für das Aufsetzen dieser. Diese Techniken fallen unter das Konzept des Prompt Engineering.

4.4 Prompt Engineering

Prompt Engineering umfasst Techniken zur Gestaltung von Eingabeaufforderungen für dialogbasierte Systeme, wie LLMs, um so die Ausgabe in eine gezielte Richtung zu lenken oder zu optimieren. Diese Techniken gelten nicht nur für Text-zu-Text basierte Systeme, wie LLMs, sondern auch für Text-zu-Bild oder auch Text-zu-Video basierte Systeme. Das bedeutet, dass diese Techniken auch für andere dialogbasierte Modelle im Bereich generative KI verwendet werden können. Da die Qualität der Ausgaben von Sprachmodellen mit der Formulierung der Anfragen variiert, ist es wichtig diese möglichst präzise zu formulieren. Die Popularisierung von LLMs durch moderne und leistungsfähige Systeme hat Prompt Engineering damit zu einem zentralen Bestandteil in der Verwendung dieser Modelle gemacht. Über die letzten Jahre haben sich so verschiedene Frameworks und Techniken im Bereich Prompt Engineering verbreitet. Grundsätzlich haben sich für Sprachmodelle über die Zeit Best Practices zur Formulierung von möglichst effektiven Prompts ergeben. So empfiehlt die offizielle Microsoft-Dokumentation für das Erstellen von Prompts in den verschiedenen Versionen des GPT-Sprachmodells, dass diese aus grundlegenden Bestandteilen zusammengesetzt sein sollten. Bevor die eigentliche Aufgabe beschrieben wird, sollte dem Sprachmodell ein Kontext gegeben werden. Dieser kann aus mehreren Elementen bestehen und sollte für bestmögliche Ergebnisse den thematischen Kontext, eine Rollenbeschreibung und die Tonalität beinhalten. Die Rollenbeschreibung schreibt dem Modell vor, dass es eine bestimmte Rolle, wie die eines Mathematikprofessors oder einer Mathematikprofessorin einnimmt. Durch die Tonalität kann nun beschrieben werden, dass die Antworten, die generiert werden sollen, einen für einen Erstsemesterstudierenden verständlichen Ton haben sollen. Der thematische Kontext beschreibt nun Ziel und Hintergrund der folgenden Aufgabe. Nach der beschriebenen Aufgabe empfiehlt es sich das Format, in dem die Antworten generiert werden sollen, kurz zu beschreiben [48]. Zusätzlich zu diesen Kernelementen existieren auch weitere Techniken zur Optimierung von Antworten durch Dialog-basierte KI-Systeme.

Die simpelste Form der Formulierung von Prompts ist das Zero-Shot Prompting. Bei dieser Technik wird die Eingabeaufforderung direkt formuliert, ohne zusätzlichen Kontext zur Rückgabe durch die Verwendung von Beispielen. [6] Diese Technik kann vor allem für einfache Anfragen, wie Zusammenfassungen, Wissensfragen und Paraphrasieren verwendet werden, bei denen eine genaue Formatierung der Ergebnisse keine Rolle spielt. So fällt die folgende Anfrage beispielsweise in die Kategorie des Zero-Shot Prompting.

Schreibe eine Zusammenfassung des folgenden Artikels: ”Artikelinhalt”

Möchte der Endnutzer oder die Endnutzerin nun, dass das Ergebnis nach einem bestimmten Format aufgebaut ist, oder auch lediglich, dass die Qualität des Ergebnisses gesteigert wird, kann der Prompt durch Beispiele von Anfrage-Ergebnis Kombinationen

ergänzt werden. Dies kann durch n-viele Beispiele erfolgen. Für n = 1 wird diese Technik als One-Shot Prompting bezeichnet und für n $>$1 als Few-Shot Prompting [6]. So wird die Genauigkeit der Rückgabewerte, vor allem für spezifische Aufgaben unterstützt. Der folgende Prompt fällt beispielsweise unter One-Shot Prompting.

> Beispiel: Übersetze die folgenden Sätze ins Französische: ”Hallo, wie geht es dir? $\rightarrow$ Bonjour, comment ça va?”. Jetzt übersetze folgenden Satz analog zum Beispiel: ”Mir geht es gut.“

Möchten die Nutzenden nicht lediglich die Antwort auf eine Lösung erhalten, sondern zusätzlich eine schrittweise Darlegung des Vorgehens zum Erreichen einer Antwort, kann Chain-of-Thought (CoT) Prompting verwendet werden. Eine solche schrittweise Generierung unterstützt die Problemlösungskompetenz bei der Generierung und erhöht zudem die Nachvollziehbarkeit bei Nutzenden [79]. Es ist allerdings nicht zu vergleichen mit tatsächlicher Erklärbarkeit, da es zum einen nicht den tatsächlichen Rechenprozess des Modells widerspiegelt und auch innerhalb der Schritte einer Antwort Fehler generiert werden können.

Eine weitere Methode zur Fehlerminimierung bei der Generierung von Antworten ist das Self-Consistency Prompting. LLMs werden zwar auf großen Datenmengen trainiert, sind allerdings dennoch fehleranfällig. Bei der Generierung einer Antwort besteht die Möglichkeit, dass eine objektiv fehlerhafte Aussage generiert wird, die sich nicht durch die trainierten Daten oder eine fehlerhafte Anfrage durch die Endnutzenden erklären lässt, aber von dem Sprachmodell als faktisch korrekt dargestellt wird. Ein solcher Fehler wird auch als Halluzination bezeichnet und kann fatale Folgen in kritischen Domänen haben, wenn dieser nicht erkannt wird [2]. Bei Self-Consistency Prompting wird nicht nur eine Antwort generiert, sondern mehrere, wobei die konsistenteste Antwort final ausgewählt wird. Dies kann durch eine Aufforderung im Prompt, mehrere Antworten zu generieren, geschehen oder durch die mehrfache Ausführung eines Prompts [78].

Reichen diese Methoden nicht aus, kann der Prompt auch zusätzlich durch externe Informationen unterstützt werden. So können dem Sprachmodell Informationen aus einer externen Datenbank oder Informationen aus einem beigefügten Dokument gegeben werden. Diese Informationen können daraufhin für die Generierung von Antworten abgerufen werden [35]. Auch durch diese Technik kann die Anzahl von Halluzinationen verringert werden und die faktische Genauigkeit wird verbessert. Zudem können so auch wesentlich komplexere und auch mehr Beispiele beigefügt werden als durch reines Few-Shot Prompting.

Prompt Engineering spielt damit auch in dieser Arbeit eine zentrale Rolle. Durch den Einsatz verschiedener Prompt Engineering Strategien wird überprüft, ob und wie effektiv sich OWL-Fakten aus natürlichsprachlichen Texten extrahieren lassen und welche Schwachstellen sich identifizieren lassen. Durch diesen Einsatz soll untersucht werden, wie sich LLMs aktuell optimal mit Ontologien kombinieren lassen, um nachvollziehbare und erklärbare Schlussfolgerungen zu ermöglichen. Hierfür müssen zunächst noch die fachlichen Grundlagen für Ontologien und OWL-basierte Konzepte aufgesetzt werden.

4.5 Ontologien

Ontologie als Begriff stammt ursprünglich aus dem Themenbereich der Philosophie und bezieht sich auf die Lehre des Seins und die damit einhergehende Kategorisierung der Existenz sowie den Beziehungen zwischen Entitäten [8]. Bereits im römischen Reich haben sich Philosophen mit diesem Konzept befasst. So finden sich bereits konzeptionelle Themen hierzu im Buch des römischen Philosophen Lukrez ”de rerum natura”(Deutsch: Über die Natur der Dinge), in dem bereits über den Hintergrund und die Zusammenhänge zwischen weltlichen Konzepten geschrieben wird [37]. In der Informatik werden Ontologien abgrenzend zur Philosophie definiert. In den 1980er bis 1990er Jahren werden sie als eine Methode zur Wissensrepräsentation populär [26]. So sind zu dieser Zeit erste computergestützte Ontologien entwickelt worden, um Wissensrepräsentation durch eine Form der Objektrelationen im großen Stil aufzusetzen. Darunter fällt auch eines der bekanntesten Projekte, die Ontologie CYC, die über 10^6 Axiome im Bereich des Allgemeinwissens beinhaltet, welche eine für die Zeit große, verknüpfte Wissensdatenbank bilden [33].

Durch das Konzept des ”Semantic Web” ist diese Idee der verknüpften Wissensrepräsentation schließlich noch weiterentwickelt worden. Vorangetrieben durch Tim Berners Lee und das World Wide Web Consortium (W3C), das offene Standards für das World Wide Web (WWW) erarbeitet und publiziert, ist so ein Konzept zur Erweiterung des WWW entstanden. Dieses soll ermöglichen, Daten im Web mit zusätzlicher Semantik zu erweitern, um sie über das Web hinweg nicht nur für Menschen verständlich zu machen, sondern auch maschineninterpretierbare Verknüpfungen zu ermöglichen [29]. Um eine solche über das Web verteilte Interpretierbarkeit zu ermöglichen, ist es notwendig, Standards zu setzen, damit verteilte Systeme untereinander kommunizieren können. Hierfür schlägt das W3C das Remote Description Framework (RDF) als Grundgerüst für semantische Daten und darauf aufbauend OWL vor [55]. Mittlerweile verwendet bereits ein beachtlicher Teil von Wissensbasen diesen Standard und Ontologien sind so zu einem zentralen Konzept in semantischen Online-Suchmaschinen geworden. OWL und RDF werden im folgenden Grundlagenkapitel weitergehend erläutert. Um diese Frameworks zu verstehen, bedarf es neben der Herkunft von Ontologien aber auch der dort enthaltenen Grundkonzepte und dem Verständnis der Struktur einer Ontologie.

Ontologien sind also im Kontext dieser Arbeit formale Wissensrepräsentationen, die es ermöglichen, in einer gegebenen Domäne eine strukturierte Beschreibung von Konzepten, Relationen und Regeln unter Verwendung einer einheitlichen Terminologie darzustellen. Ontologien bestehen dabei aus verschiedenen Komponenten [71].

- Konzepte beschreiben Entitäten oder Abstraktionen von solchen innerhalb einer gegebenen Domäne. Sie werden auch als Klasse bezeichnet.
- Instanzen sind Konkretisierungen von Konzepten. Sie werden auch als Individuals / Individuen bezeichnet.
- Relationen, auch Properties genannt, beschreiben die Beziehungen zwischen Instanzen und zwischen Konzepten.
- Axiome sind Regeln, die Einschränkungen innerhalb der Ontologie definieren und so Konsistenz gewährleisten

Zudem lassen sich in einer Ontologie auch taxonomische Konzepte, wie Ober- und Unterklassen definieren und basierend auf den aufgesetzten Komponenten innerhalb einer Ontologie lassen sich zudem Regeln festlegen und so durch logische Inferenz weitere Aussagen ableiten.

So lassen sich in der medizinischen Domäne in einer Ontologie beispielsweise die Klassen ”Patient”, ”Krankheit”, ”Symptom” und ”Behandlung” aufsetzen. Instanzen dieser Klassen können sein ”Typ-1-Diabetes”, ”Hyperglykämie”(hoher Blutzuckerspiegel) und Ïnsulintherapie”. Zwischen diesen Instanzen und Klassen werden dann Relationen festgelegt. ”hat_Symptom” und ”hat_Behandlung” könnten hier passende Relationen sein. Damit diese Relationen auch einen semantischen Sinn erhalten, können durch Axiome nun logische Regeln definiert werden, die festlegen, dass ein Patient mit Hyperglykämie an Diabetes leiden kann und dass Insulintherapie eine mögliche Behandlungsform ist. Ein Expertensystem, welches eine solche Ontologie verwendet, könnte nun durch logische Inferenz, klassischerweise durch den Mechanismus Modus Ponens oder komplexeren logischen Kalkülen[68], empfehlen, dass ein Patient A mit Hyperglykämie mit Insulintherapie behandelt werden könnte. Als logische formale Struktur würde diese Inferenz wie folgt aussehen.

$$\frac{\text{hat\_Symptom}(\text{PatientA},\text{Hyperglykämie}) \quad \text{ist\_Symptom\_von}(\text{Hyperglykämie},\text{Diabetes})}{\text{hat\_Krankheit}(\text{PatientA},\text{Diabetes})} \quad (\text{Modus Ponens})$$

$$\frac{\text{hat\_Krankheit}(\text{PatientA},\text{Diabetes}) \quad \text{hat\_Behandlung}(\text{Diabetes},\text{Insulintherapie})}{\text{empfohlene\_Behandlung}(\text{PatientA},\text{Insulintherapie})} \quad (\text{Modus Ponens})$$

Ontologien ermöglichen also objekt-relationale Wissensrepräsentationen und durch logische Inferenz können Schlussfolgerungen getroffen werden. Durch OWL ist hierfür ein standardisiertes Sprach-Framework gegeben, das die Formulierung von Ontologien und die Grundlagen Konzepte zur Ableitung von Aussagen durch Inferenz ermöglicht. Zum Verständnis von OWL ist nun noch zu erläutern, wie die Elemente einer Ontologie in OWL dargestellt werden und wie OWL-Dateien aufgebaut sind.

4.6 Web Ontology Language

Die Web Ontology Language ist eine von dem W3C im Rahmen des Semantic Web entwickelte formale Sprache zur Erstellung von Ontologien und Verarbeitung dieser durch Maschinen. In ihrer konzeptionellen Grundlage basiert OWL auf der Beschreibungslogik (engl. Description Logic) und ermöglicht hierdurch komplexe logische Schlussfolgerungen. OWL baut auf RDF auf, einem Framework zur Repräsentation von Informationen im Web und erweitert dieses um eine formale Grundlage zur Erstellung von Ontologien und zum Ermöglichen von logischen Schlussfolgerungen [56].

RDF beschreibt Daten im Web durch die Verwendung von Tripeln. Ein solches Tripel besteht aus einem Subjekt, welches das beschriebene Konzept repräsentiert und mit der Zeile einer relationalen Datenbanktabelle verglichen werden kann, einem Objekt, das die verknüpfte Entität darstellt und einem Prädikat, welches die Relation zwischen Subjekt und Objekt beschreibt. Es gibt verschiedene Notationen zur Darstellung von RDF-Dokumenten, wobei die wichtigsten aktuellen Notationen wohl die N3-Notation und

die daraus entstandene Turtle-Notation sind, welche vom W3C entwickelt und empfohlen werden. Die JSON-LD-Notation, die auf JSON als Struktur basiert, und RDF/XML, die XML als Grundlage verwendet, sind ebenfalls gängig [62]. In der beispielhaften Aussage: ”Ein Symptom von Diabetes kann Hyperglykämie sein.” wäre ”Diabetes” das Subjekt eines solchen Tripels, ”hatSymptom” das Prädikat und ”Hyperglykämie” ein Objekt. Ein RDF-Dokument, welches dieses Tripel abbildet, könnte wie folgt aussehen.

```
    @prefix ex: <http://example.org/> .

    ex:Diabetes ex:hatSymptom ex:Hyperglykaemie .
```

Listing 4.1: Einfaches RDF Beispiel in Turtle

Wie auch in anderen Sprachen, müssen in RDF-Dokumenten Namensräume definiert werden, welche einen Uniform Resource Identifier (URI) zur eindeutigen Identifikation der Ressourcen enthalten. RDF allein bietet allerdings nicht die Möglichkeit, komplexere Zusammenhänge zwischen Entitäten zu modellieren oder logische Schlussfolgerungen durchzuführen, weshalb RDF-Schema (RDFS) und OWL dieses erweitern. RDFS ermöglicht zusätzlich die Klassifikation von Entitäten, sowie die Definition von Klassenhierarchien. Dadurch ließe sich das vorherige Beispiel um Klassendefinitionen und Konzepte, wie Ober-/Unterklassen sowie Definitions- und Wertebereiche für Prädikate erweitern.

```
@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

# Definition von Klassen
ex:Krankheit a rdfs:Class .
ex:Symptom a rdfs:Class .

# Hierarchie von Krankheiten
ex:ChronischeKrankheit a rdfs:Class ;
    rdfs:subClassOf ex:Krankheit .

ex:Diabetes a ex:ChronischeKrankheit .

# Definition der Beziehung "hatSymptom"
ex:hatSymptom a rdf:Property ;
    rdfs:domain ex:Krankheit ;
    rdfs:range ex:Symptom ;

# Anwendung der Beziehungen auf Instanzen
ex:Diabetes ex:hatSymptom ex:Hyperglykaemie .
```

Listing 4.2: Erweitertes RDFS-Beispiel mit Ober-/Unterklassen

Die durch RDFS ermöglichten Konzepte lassen sich durch den Namensraum ”rdfs” erkennen. Mit OWL lassen sich nun auch weitere Konzepte aus der Logik repräsentieren. So kann beispielsweise angegeben werden, dass Entitäten nicht einer Klasse angehören, Gleichheit und Ungleichheit von Klassen können angegeben werden, und auch lassen sich Disjunktheit oder Quantorrestriktion definieren [56]. So kann in einem OWL-Dokument angegeben werden, dass eine Entität nicht zugleich eine Krankheit und ein Symptom sein kann, oder dass eine Krankheit mindestens ein Symptom haben muss.

```
@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

# Definition von Klassen
ex:Krankheit a owl:Class .
ex:Symptom a owl:Class .

# Disjunkt: Eine Entitaet kann nicht Krankheit und Symptom sein
ex:Krankheit owl:disjointWith ex:Symptom .

# Hierarchie von Krankheiten
ex:ChronischeKrankheit a owl:Class ;
    rdfs:subClassOf ex:Krankheit .

ex:Diabetes a ex:ChronischeKrankheit .

# Definition der Beziehung "hatSymptom"
ex:hatSymptom a owl:ObjectProperty ;
    rdfs:domain ex:Krankheit ;
    rdfs:range ex:Symptom .

# Existenzrestriktion: Jede Krankheit muss min. ein Symptom haben
ex:Krankheit rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty ex:hatSymptom ;
    owl:someValuesFrom ex:Symptom
] .

# Anwendung der Beziehungen auf Instanzen
ex:Diabetes ex:hatSymptom ex:Hyperglykaemie .
```

Listing 4.3: OWL-Erweiterung mit Disjunktheit und Existenzrestriktion

Die angegebenen Beispiele sind alle in Turtle formuliert, da Turtle eine sehr gute menschliche Lesbarkeit aufweist, hätten aber auch mit jeder beliebigen Syntax verfasst werden können. Ein weiteres wichtiges Konzept von OWL-basierten Ontologien ist, dass dort die sogenannte ”Open World Assumption” (OWA) gilt. Im Gegensatz zur ”Closed World Assumption” (CWA) besagt dieses Konzept, dass fehlende Informationen in einer Wissensrepräsentation nicht als falsch deklariert werden, sondern stattdessen als unbekannt. Neben unterschiedlicher Syntax ist OWL auch in verschiedenen Varianten ausgeprägt, die sich in der Stärke ihrer Ausdrucksweise unterscheiden. Dazu gehören OWL Lite für weniger komplexe Ontologien, OWL DL und OWL Full [56].

Die Fähigkeit logische Inferenzen durchführen zu können, lässt sich durch den Einsatz eines Reasoners unterstützen, der diese Ableitungen automatisiert durchführt. Die Begründungen für Schlussfolgerungen solcher Reasoner lassen sich visualisieren, weshalb sie für den Bereich XAI von Interesse sind, auch da ein manuelles Durchführen solcher Inferenzen, besonders in komplexen Ontologien, nicht verhältnismäßig wäre. Zudem ist auch das manuelle Aufsetzen von Ontologien aufwändig. Daher gibt es Anwendungen,

die Ontologie-Erstellung ohne das Schreiben von OWL-Dokumenten erlauben und zudem auch eingebaute Reasoner verwenden. Eine dieser Anwendungen ist der von der Stanford University entwickelte Ontologie-Editor Protégé [60], der in dieser Arbeit verwendet wird.

4.7 Protégé

Protégé ist ein Open-Source Ontologie-Editor, der die Erstellung, Bearbeitung und Verwaltung von Ontologien ermöglicht. Entwickelt worden ist der Editor an der Stanford University in den 1980er Jahren mit dem Ziel, Ontologien über eine Benutzeroberfläche verwalten zu können, ohne die zu dieser Zeit entwickelten Sprachen des Semantic Web verwenden zu müssen, diese aber zu unterstützen. Originär ist Protégé für die medizinische Domäne entwickelt worden, da zu diesem Zeitpunkt medizinische Diagnosesysteme beliebt waren [51]. Über die Jahrtausendwende hinweg ist Protégé stetig weiterentwickelt worden und unterstützt mittlerweile auch RDF, OWL und DL. Hinzu kommt, dass das Java-basierte Programm architektonisch modular aufgebaut ist und sich durch Plugins flexibel erweitern lässt.

Aktuell gibt es zwei verschiedene unterstützte Versionen von Protégé. Protégé Desktop ist die klassische Desktopanwendung zur Verwaltung von Ontologien und zweitens Web-Protégé, das die Bearbeitung über eine Weboberfläche ermöglicht und zudem auch kollaborative Arbeit an einem Projekt erlaubt [73]. In dieser Arbeit wird die Desktop-Variante verwendet. Der standardmäßige Funktionsumfang des Editors umfasst neben der bereits erwähnten Unterstützung für OWL und RDF-basierte Ontologien auch die grafische Modellierung von Ontologien. Zudem besitzt Protégé auch verschiedene integrierte Reasoner zur logischen Ableitung und je nach verwendetem Plugin auch andere Funktionalitäten, wie eine integrierte Validierung von Ontologien. Der Aufbau einer Ontologie in Protégé orientiert sich an der allgemeinen Struktur einer OWL-Ontologie und besteht daher aus äquivalenten Komponenten. Die Konzepte einer Ontologie werden in Protégé als ”Classes” oder Klassen definiert. Die Instanzen eines Konzeptes werden als Individuals bezeichnet. Protégé ermöglicht auch das Definieren von Relationen. Diese dort als Properties bezeichneten Eigenschaften lassen sich in drei Kategorien einteilen.

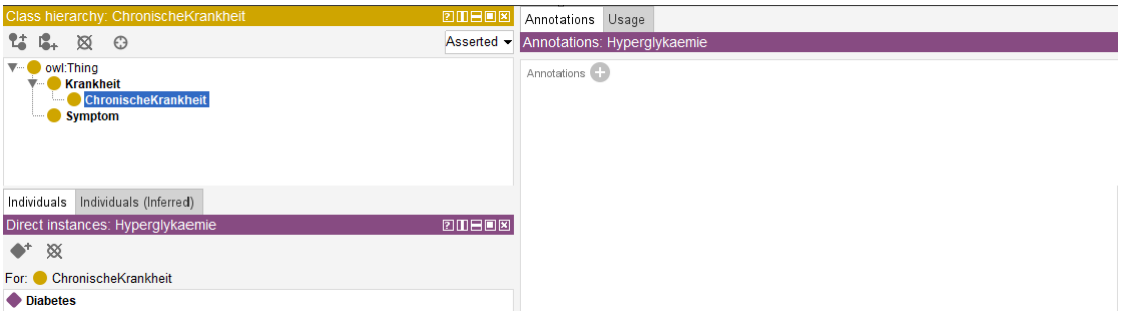
1. Object Properties sind Relationen zwischen zwei Individuals. ”Diabetes hatSymptom Hyperglykämie” wäre beispielsweise eine Object Property.

2. Data Properties verknüpfen Individuals stattdessen mit einem Datenwert. So könnte im medizinischen Beispiel einem Individual ”Blutzuckerwert” der Wert 180 zugewiesen werden. Protégé bietet hierbei eine Reihe von Datentypen. wie Integer, Strings, Datum, und auch Weitere. Zudem können eigene Datentypen spezifiziert werden.

3. Die letzte Art der Relationen sind die Annotation Properties. Dabei handelt es sich um Metadaten für Objekte in der Ontologie.

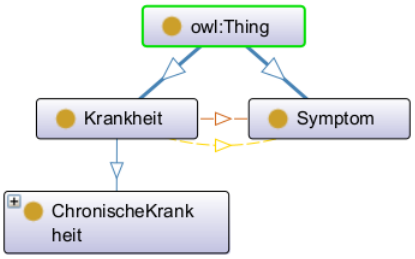
Über den Reiter ”Description” lassen sich Axiome und Einschränkungen für Klassen definieren. Ober-/Unterklassen, Disjunktheit und Quantor-basierte Restriktionen fallen unter anderem darunter. RDF-Elemente, wie Subklassen werden indirekt über eine hierarchische Struktur in der Benutzeroberfläche definiert [61]. Eine weitere Funktionalität, die auch für diese Arbeit von Relevanz ist, ist die Möglichkeit OWL-Dateien zu importieren und erstellte Ontologien auch in verschiedenen Notationen zu exportieren. So lässt sich auch die bereits präsentierte Beispiel-Ontologie in Protégé importieren.

Abbildung 4.1: Klassenhierarchie der Beispielontologie in Protégé



So lässt sich die Klassenhierarchie erkennen, die in der OWL-Datei definiert worden ist. Auch ist zu sehen, dass den Klassen korrekte Individuals zugewiesen worden sind. In diesem Fall ist ”Diabetes” eine Instanz der Klasse ”ChronischeKrankheit”. Auch definierte Einschränkungen, dass eine Krankheit mindestens ein Symptom haben muss und dass der Definitions- und Wertebereich der Object Property ”hatSymptom” eine Krankheit und ein Symptom sind, sind übernommen worden. Nur ist das Symptom ”Hyperglykämie” noch keine Instanz der Klasse ”Symptom”, da dies nicht in der OWL-Datei explizit definiert worden ist. Beim Ausführen eines Reasoners würde dies aber durch die definierten Axiome abgeleitet werden. Die hierarchische Struktur lässt sich auch grafisch darstellen, was im Verlauf der Arbeit zur Verwendung kommt, um die entwickelten Ontologien zu präsentieren.

Abbildung 4.2: Klassenhierarchie der Beispielontologie als Graph



Anhand dieses Graphen lassen sich auch die definierten Einschränkungen und Object Properties einsehen. So zeigt der rote Pfeil in der obigen Abbildung die Einschränkung des Definitions- und Wertebereichs und der gelbe Pfeil die Existenzeinschränkung. Nun da die wichtigsten theoretischen Grundlagen dargelegt worden sind, gilt es, den aktuellen Stand der Forschung zu ermitteln, um den Inhalt dieser Arbeit wissenschaftlich einzuordnen und zu zeigen, welche Fortschritte bereits in diesem Gebiet erfolgt sind.

5
State of the Art

Die stetigen Fortschritte im Bereich KI haben zu leistungsstarken Modellen geführt und LLMs wie ChatGPT haben für große Erfolge im Bereich NLP gesorgt. Diese Erfolge in den letzten Jahren haben das Forschungsfeld von XAI stetig vorangetrieben, damit Ansätze entwickelt werden können, um die Entscheidungen dieser Modelle nachvollziehbarer zu gestalten. Während klassische XAI-Ansätze bereits etabliert sind, erhalten neue hybride Methoden, bei denen andere Methoden der Wissensrepräsentation mit neuronalen Netzen kombiniert werden, mehr Bedeutung. Für diese hybriden Ansätze können auch Ontologien eine wichtige Rolle spielen, da diese durch ihre strukturierte Repräsentation von Wissen eine gute Möglichkeit zur Verbesserung der Erklärbarkeit bieten.

Dieses Kapitel soll einen Überblick über den aktuellen Stand der Forschung im Bereich Explainable AI und die Verwendung von kombinatorischen Ansätzen zwischen Ontologien und auf neuronalen Netzen basierenden KI-Systemen geben. Dabei sollen bestehende Konzepte präsentiert und relevante Forschungsarbeiten analysiert werden, sodass bestehende Lücken und Herausforderungen identifiziert werden können und sich der Kontext dieser Arbeit in das Forschungsfeld einordnen lässt. Zunächst wird auf aktuelle Ansätze im Bereich XAI eingegangen, gefolgt von einer Analyse bestehender Forschungsarbeiten in Bezug auf die Kombination von Ontologien und LLMs. Im Anschluss werden die bestehenden Lücken und Herausforderungen dargestellt, um die Arbeit in den Forschungsstand einzuordnen.

5.1 Aktueller Forschungsstand zu XAI und der Erklärbarkeit von KI-Modellen

Die zunehmende Verbreitung von LLMs in produktiven Anwendungsgebieten erfordert die Entwicklung neuer Transparenz-Ansätze für Verständlichkeit und Nachvollziehbarkeit. Dabei gilt dies vor allem für Systeme, die sicherheitskritisch sind oder ethisch sensibel. Daher hat sich XAI in den letzten Jahren als ein zentrales Konzept etabliert, um Antworten auf diese Anforderungen zu finden.

5.1.1 XAI als Schlüsseltechnologie in der Praxis

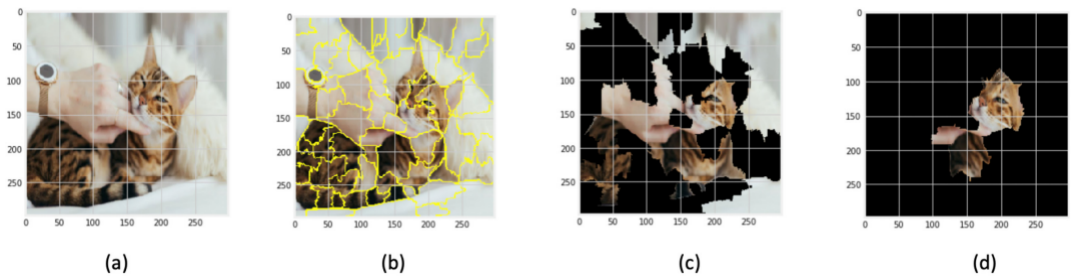
XAI ist ein aktuelles und mittlerweile praxisrelevantes Forschungsfeld, das für Unternehmen in verschiedensten Domänen und auch Organisationen, wie behördliche Institutionen oder internationale Organisationen von Bedeutung ist, was sich auch in regulatorischen Entscheidungen auf internationaler Ebene, wie dem ”AI Act” zur Einschränkung von risikoreichen Anwendungen, zeigt [17]. Eine umfassende Analyse von Dwivedi et al. aus dem Jahr 2023 präsentiert, dass Ansätze aus dem Bereich XAI zunehmend in der Praxis in Unternehmen integriert werden. Dabei untersuchen die Autorinnen und Autoren unter anderem Techniken aus der Praxis, zur Modell-Validierung, Entscheidungsunterstützung und

Nachprüfbarkeit von Modellen [16]. Die Forschenden klassifizieren dabei die XAI-Ansätze in verschiedene Techniken und erarbeiten die Unterschiede der Ansätze in Modelleigenschaften, Verfügbarkeit von nutzbaren Toolkits und Zielgruppen. Zudem unterscheiden die Autorinnen und Autoren die Modelle in ihrem Interpretationsraum, der entweder global ist, also den Entscheidungsprozess im Generellen analysiert oder lokal, was bedeutet, dass einzelne Entscheidungen des Modells analysiert werden. Dabei heben die Forschenden besonders Frameworks wie SHAP, LIME, PDPbox und Skater hervor, die sich bereits in verschiedenen Unternehmenskontexten bewährt haben [16].

SHAP (SHapely Additive exPlanations) ist ein Framework, das 2017 von Lundberg und Lee veröffentlicht worden ist und auf den Prinzipien der Spieltheorie basiert. Es geht vor allem auf die Shapley Values aus dem Bereich der kooperativen Spieltheorie ein. Shapley Values bezeichnen in der Spieltheorie eine Bewertungsmethodik, die den Beitrag jedes Spielenden am Gesamtergebnis des Spiels bestimmt. Das SHAP-Framework überträgt dieses Konzept auf Vorhersagen in Machine Learning basierten KI-Modellen. Dabei werden den Features, also den Merkmalen, die ein Machine Learning Modell zur Vorhersage verwendet, Wahrscheinlichkeitswerte zugeordnet, die deren anteiligen Beitrag an der Gesamtwahrscheinlichkeit für eine Vorhersage repräsentieren [39]. Dieses Framework kann durch diesen Ansatz für verschiedene Modelle verwendet werden, um so bei Modellen, die strukturierte Tabellen als Datengrundlage verwenden, den Spalten Shapley-Werte zuzuweisen. Auch den Pixelbereichen eines Bilderkennungsmodells, um beispielsweise die Bereiche eines Röntgenbilds zu bewerten, die zur automatischen Diagnose den größten Beitrag geleistet haben, können diese Werte zugewiesen werden. Durch diese Architektur fällt es unter die Post-Hoc Erklärungsverfahren.

LIME (Local Interpretable Model-agnostic Explanations) ist ein Framework, das Erklärungen für Vorhersagen von Modellen generiert, indem es das Verhalten eines Modells durch Störungen einzelner Parameter für einen bestimmten Beispieloutput annähert. Es ist 2016 von Ribeiro et al. präsentiert worden und funktioniert ähnlich wie SHAP, da es ebenfalls den Features eines Modells Werte zuordnet. Im Gegensatz zu SHAP wird hier ein

Abbildung 5.1: LIME Vorgehen zur Feature Gewichtung einer ägyptischen Katze [16]



Beispieloutput gewählt und dieser wird auf Veränderung überprüft, indem mehrmals leichte Anpassungen am Input vorgenommen werden und so leicht unterschiedliche Vorhersagen generiert werden. Diese erhalten eine Gewichtung basierend auf ihrer Ähnlichkeit zum Original und basierend auf diesen gewichteten Datensätzen wird ein kleines Modell trainiert. Die Gewichte dieses Modells zeigen nun, welche Features den größten Einfluss auf das Ergebnis hatten und erklären so die Vorhersage [64]. Auch LIME fällt damit genau wie

SHAP unter die Post-Hoc-Methoden und beide Frameworks interpretieren lokal. Beide fallen zudem unter das XAI-Konzept Feature Importance.

PDPbox (Partial Dependence Plot toolbox) ist ein Python Toolkit zur Visualisierung, das sogenannte Partial Dependence Plots verwendet, um den Einfluss von Features auf Vorhersagen darzustellen. Im Gegensatz zu SHAP und LIME erzeugt es globale Erklärungen, da es über den gesamten Datensatz hinweg den Einfluss der Features auf einem übergeordneten Level darstellt. Das letzte fokussierte Framework Skater ist eine Python Bibliothek, die verschiedene XAI-Methoden ermöglicht, wobei sowohl lokale als auch globale Interpretationen unterstützt werden und verschiedene Modelltypen unterstützt werden. Mit diesen und weiteren Methoden präsentieren die Autoren und Autorinnen aktuell eingesetzte Methoden zur Erklärung von Machine Learning Methoden in der Praxis. Zugleich untermauern sie ihre Wichtigkeit für bestehende Branchen und Anwendungsfälle, vor allem für kritische und ethisch sensible Domänen.

5.1.2 Vertrauen als eine fundamentale Herausforderung

Es gibt aktuell verschiedene Ansätze zur Verbesserung der Erklärbarkeit von KI-Modellen durch die Integration von XAI-Frameworks. Die Frage, wie das Vertrauen in KI gesteigert werden kann, ist dabei von zentralem Wert. Die präsentierten und häufig lokalen Methoden im Bereich XAI, wie SHAP oder LIME, ermöglichen zwar eine lokale Erklärbarkeit, um Aufschluss über Entscheidungsprozesse von Modellen zu geben, jedoch reicht dies nicht aus, um eine Vertrauensbasis für Anwender und Anwenderinnen zu schaffen. Narteni et al. haben in ihrer Forschung Methoden entwickelt, um das Vertrauen in KI-Systeme zur Entscheidungsunterstützung zu verbessern. Dabei kombinieren sie Erklärbarkeit und Verlässlichkeit zu einem Ansatz namens ”Reliable AI” [49]. Die Autoren schlagen für diese Entwicklung sogenannte ”Safety Regions” vor. Dabei handelt es sich um von dem Modell angegebene Bereiche für die Eingaben eines Modells, in denen die Vorhersagen mit hoher Wahrscheinlichkeit korrekt sind. Ziel dieser Regionen ist es, nur Vorhersagen zu erlauben, die innerhalb dieser Regionen liegen, um so die Fehlerwahrscheinlichkeit vor allem für risikobehaftete Domänen zu reduzieren.

Die Identifikation der Safety Regions basiert auf verschiedenen Klassifikationsmethoden, um transparente Wenn-Dann-Regeln aus den Trainingsdaten abzuleiten. Diese Regeln ermöglichen eine semantische Interpretation des Modells und zugleich auch eine Überprüfung durch Experten und Expertinnen. Die Forschenden validieren ihr Framework durch eine Fallstudie am Beispiel Cybersecurity in der Vernetzung von Fahrzeugen. Durch das Framework lässt sich das Systemverhalten während eines Cyberangriffs klassifizieren. Durch diese Klassifizierung lassen sich Angriffsmuster identifizieren, wodurch eine Risikoeinschätzung in Echtzeit ermöglicht wird [49]. Durch die Kombination von diesen Safety Regions und der Etablierung von regelmäßigen Audits zur Erkennung von Mängeln im KI-Modell sowie der Einhaltung von rechtlich-ethischen Standards wird sowohl ein Maß an Verlässlichkeit, als auch Erklärbarkeit ermöglicht. Dadurch wird ein Schritt in Richtung ”Reliable AI”gemacht.

5.1.3 Erklärbarkeit von LLMs

Auch in Hinblick auf LLMs im Speziellen wird im Bereich XAI aktuell geforscht. Trotz der großen Leistungsfähigkeit von Sprachmodellen stellt die Erklärbarkeit weiterhin eine Herausforderung dar. Cambria et al. zeigen in einer systematischen Studie, dass es in der Forschung aktuell noch keine einheitliche und standardisierte Methode gibt, um die Entscheidungsmechanismen von LLMs zu erklären. In ihrer Studie teilen die Autoren und Autorinnen die aktuellen Ansätze in zwei verschiedene Kategorien ein. Diese lauten ”To Explain” und ”As Feature” [9].

To Explain sind Ansätze und Methoden, bei denen versucht wird, die tatsächliche Verarbeitung von LLMs und die internen Prozesse transparenter zu entwickeln und zu gestalten. In diese Kategorie fallen unter anderem Visualisierungsmethoden, wie das durch Jesse Vig präsentierte Framework zur Visualisierung des tatsächlichen Attention-Mechanismus von Transformer-Modell-basierten Systemen wie BERT oder GPT. Durch solche Frameworks lassen sich einzelne Neuronen, aber auch Schichten im Modell hervorheben, Fehlerquellen identifizieren und Modellverhalten besser interpretieren [76]. Allerdings hängt die tatsächliche Interpretierbarkeit von den technischen Kenntnissen der Anwendenden über Transformer-Modelle ab. Andere Ansätze wie das ”Framework Boundless DAS” erkennen kausale Strukturen in LLMs während der Generierung von Antworten zwischen Ausschnitten aus Antworten und Neuronengruppen. Dies wird ermöglicht durch die Analyse von Neuronengruppen bei der Maskierung von Eingaben. So ist dieses Tool bereits in der Lage gewesen zu verdeutlichen, wie ein Sprachmodell bestimmte numerische Aufgaben durch die Verwendung von Operatoren gelöst hat [82]. Auch Frameworks, die externe Wissensdatenbanken verwenden, wie Knowledge Graphs, werden präsentiert. So beispielsweise der LMExplainer, der semantische Zusammenhänge zu einer Frage mithilfe von externen Knowledge Graphs aufbereitet und die relevantesten Knoten in die Antwort des verwendeten Sprachmodells mit einbaut, um so eine verständliche Erklärbarkeit zu ermöglichen. So bedienen sich die Autoren und Autorinnen an dem externen Knowledge Graph ConceptNet. Das Sprachmodell erhält in diesem Framework eine Eingabe in Form von einer Frage. Aus relevanten Begriffen der Eingabe wird in dem externen Knowledge Graph eine Teilmenge erstellt, die relevante semantische Zusammenhänge darstellt und die wichtigsten Knoten im LLM verarbeitet, um diese zur Erklärung zu verwenden [11].

Konzepte im Bereich ”As Feature” beziehen sich auf Arbeiten, die versuchen, erklärende Ausgaben zu generieren, ohne dabei die tatsächliche Funktionsweise in der Verarbeitung der LLMs zu präsentieren. Hierzu stellen die Autorinnen und Autoren Konzepte dar, die LLMs in bestehende Systeme zur Handlungsempfehlung integrieren und diese erweitern. Sie präsentieren auch Frameworks, die Chain-Of-Thought Prompting verwenden, welches zwar die Leistung des Modells und auch die Nachvollziehbarkeit verbessern kann, aber alleine auch zu verzerrten Erklärungen führen kann [9].

Bei der Analyse der bestehenden Methoden ist eines der Probleme, die von den Forschenden identifiziert werden, dass die Methodik in der Forschung im Bezug auf die Validierung von Frameworks sich zu sehr ähnelt. Studien beziehen sich übermäßig auf Benchmarking oder konkrete linguistische Aufgaben für Sprachmodelle. Anwendungsfälle, die auf längeren dialogbasierten Interaktionen basieren, werden weniger behandelt, was

zu Lücken in der Forschung führt. Abschließend formulieren die Autoren zwei Handlungsempfehlungen für die zukünftige Forschung im Bereich XAI im Zusammenhang mit großen Sprachmodellen [9].

- Erklärbarkeit sollte nicht im Nachhinein an entwickelte Systeme angehängt werden, sondern von Beginn an in das Systemdesign integriert werden.
- Die Darstellung von Erklärungen muss besser auf Zielgruppen zugeschnitten werden, die tatsächlichen Endanwendern und Endanwenderinnen entsprechen und ggfs. nicht die nötige technische Expertise über die internen Prozesse von LLMs besitzen.

Die bisherige Analyse des Forschungsstandes zeigt zusammenfassend, dass XAI als Forschungsbereich und die Erklärbarkeit von künstlicher Intelligenz, insbesondere von LLMs, eine technische Herausforderung darstellt, die durch unterschiedlichste Ansätze versucht wird zu verbessern. Zugleich ist es auch eine gesellschaftliche und regulatorische Notwendigkeit, welche durch Institutionen und die Arbeit in sicherheitskritischen Bereichen gefordert ist. Während XAI-Frameworks, wie SHAP oder LIME der Praxis in Anwendungsbereichen von Machine Learning und weiterführenden KI-Modellen Anwendung finden, bleibt das Sicherstellen von Vertrauen in KI-Systeme eine Herausforderung. Im Kontext von LLMs zeigen sich, dass Erklärbarkeit auch in der Arbeit mit großen Sprachmodellen thematisiert wird, allerdings eher sekundär zu tragen kommt. Zudem legen aktuelle Arbeiten in dem Bereich den Fokus vermehrt auf spezielle Anwendungsfälle von LLMs, während der Einbezug von zielgruppengerechter Evaluation nach hinten fällt. Ein steigendes Interesse in der Forschung richtet sich auf hybride Frameworks, in denen externe Wissenstrukturen oder symbolische Ansätze in die Arbeit mit LLMs eingebunden werden, wie beispielsweise der LMExplainer zeigt [11]. Auch Ontologien können eine solche externe Ressource darstellen, mit deren Kopplung sich die Erklärbarkeit durch logikbasierte Inferenzmechanismen potenziell verbessern lässt. Im folgenden Kapitel wird daher der Stand der Forschung in der Integration von Ontologien und LLMs detailliert betrachtet.

5.2 Stand der Forschung zur Integration von LLMs und Ontologien

Es gibt im Rahmen der Forschung Studien, die sich bereits damit beschäftigen, einzelne Aussagen in eine formal ontologische Sprache zu übersetzen, um Ontologien durch natürliche Sprache mithilfe älterer GPT-Modelle um Elemente zu erweitern [43]. Auch die Extraktion von Kausalzusammenhängen aus natürlichsprachlichen Dokumenten mit LLMs wird bereits domänenspezifisch untersucht, um beispielsweise Ontologien für Zusammenhänge zwischen Krankheitsbildern, deren Symptomen und Behandlungsmöglichkeiten zu modellieren [32]. Arbeiten darüber, ob Ontologien in einer Zeit von LLMs noch notwendig sind und ob sie als unterstützende Ansätze gemeinsam verwendet werden können, zeigen ebenfalls eine behandelte Forschungsrichtung [50].

5.2.1 Automatisierte Ontologieerweiterung in Protégé durch GPT

Der Artikel von Mateiu und Groza zeigt einen solchen Ansatz zur Anwendung von großen Sprachmodellen in der Entwicklung von Ontologien [43]. Dabei hat das Autorenpaar ein mittlerweile älteres GPT-3-Modell durch Finetuning trainiert, um Sätze in natürlicher Sprache in OWL-Syntax zu übersetzen und so Axiome für eine gegebene Ontologie zu erhalten. Ziel der Forschenden ist dabei die Erstellung von Ontologien effizienter

zu gestalten und dies durch eine Integration ihres Modells in Protégé über ein Plugin auch praktisch umzusetzen.

Die beiden Forschenden haben insbesondere zwei Herausforderungen im Ontology-Engineering (OE) also der klassischen Entwicklung von Ontologien identifiziert.

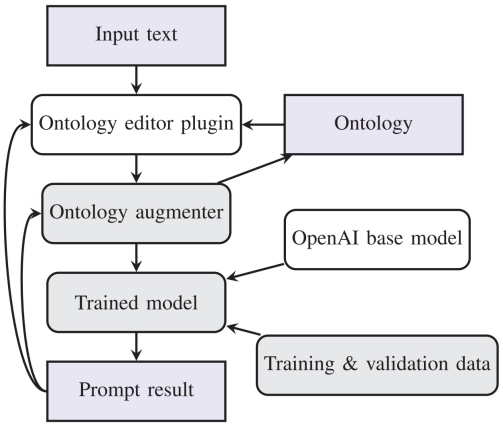
1. Der technische Aufwand für die formelle Modellierung von komplexeren Ontologien ist hoch.

2. Bei komplexeren Domänen besteht im OE eine Abhängigkeit zu Domänenexperten und Expertinnen, was Zeit und Ressourcen kostet.

In der Arbeit ist erwähnt, dass zwar etablierte Methoden im Bereich OE existieren, die den Prozess effektiver gestalten sollen, wie das ONTOCOM (Ontology Cost Model) [57], allerdings sind diese bereits lange etabliert, wodurch neue technologische Möglichkeiten nicht beachtet werden. Sie sind zudem nicht auf jedes industrielle Szenario übertragbar. LLMs sind ohnehin für die Übersetzung von Texten optimiert und können zudem mit domänenspezifischen Informationen angereichert werden, womit sie die beiden Herausforderungen adressieren und eine Integration sinnvoll erscheint.

Die Umsetzung der Integration erfolgt in dem Artikel über die Entwicklung eines Plugins für den Ontologie Editor Protégé. Das Plugin ermöglicht sowohl die Eingabe von Informationen zur Erstellung einer neuen Ontologie, als auch die Erweiterung einer bestehenden Ontologie durch neue Klassen, Individuals und Relationen. Die Spracheingabe in natürlicher Sprache erfolgt über einen Prompt an das trainierte GPT-Modell. Dieses übersetzt die Eingabe in OWL Functional Syntax, einer Syntax für OWL, die von APIs und für logik-basierte Anwendungen gerne verwendet wird und sowohl maschinenlesbar, als auch relativ einfach für Menschen lesbar ist. Über die OWL-API werden die generierten Ausgaben in die Ontologie integriert [43].

Abbildung 5.2: Datenfluss zur Übersetzung einfacher Aussagen in Protégé [43]



Der Eingabetext wird über ein Eingabefeld im Plugin eingetragen, durch den Augmenter, der auf dem trainierten GPT-Modell basiert, übersetzt und auch über den Augmenter über die OWL-API wieder an die Ontologie und das Plugin übergeben. Visuell wird das Plugin durch ein skalierbares Fenster innerhalb der Protégé-Anwendung dargestellt, das über ein

darüber liegendes Eingabefeld und ein darunterliegendes Ausgabefeld mit der generierten OWL-Syntax verfügt. Über die Eingabe des beispielhaften Satzes

> ”Anna is a girl.”

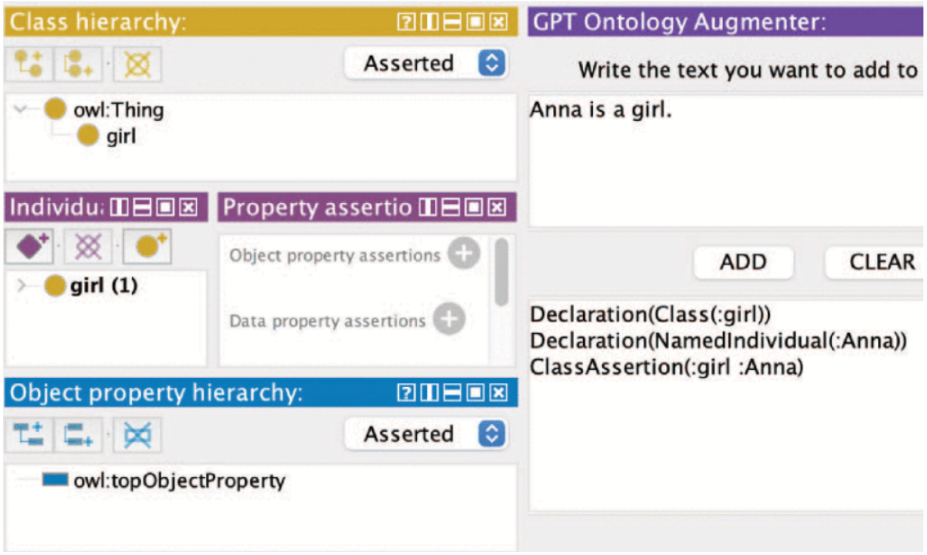
erstellt das Plugin die folgende formale, maschinenlesbare Ausgabe in DL.

> ”Declaration(Class(:girl))
> Declaration(NamedIndividual(:Anna))
> ClassAssertion(:girl :Anna)”

Die erstellten Axiome decken dabei unterschiedliche Aspekte von Ontologien ab. Darunter fallen unter anderem Klassen- und Instanzdeklarationen, Subklassen- und Disjunktheitsbeziehungen, Domänen und Wertebereiche und Kardinalitätsrestriktionen. Zum Finetuning des GPT-Modells haben die Forschenden einen Datensatz mit insgesamt 150 Prompt-Resultat-Paaren erstellt, die ein breites Spektrum von semantischen Mustern abdecken, wie taxonomische Beziehungen oder auch Eigenschaften zwischen Individuen. Zusätzlich ist auch ein Validierungsdatensatz mit 50 Einträgen eingesetzt worden, um zusätzliche Parameter zu optimieren [43].

Insgesamt haben Mateiu und Groza drei verschiedene Strategien verwendet, um die Ergebnisse zu verbessern. Beim Zero-Shot Learning ohne die Verwendung von Beispielen sind die Ergebnisse ungenau und teils zwar formal gültig, aber semantisch falsch ausgefallen, wodurch die Ergebnisse eine eher instabile Qualität aufgewiesen haben. Few-Shot Learning mit vereinzelten Beispielen hat ebenfalls zu inkonsistenten Antworten geführt und bei der wiederholten Eingabe des selben Prompts sind variierende Antworten entstanden. Durch die Verwendung des Datensatzes und Finetuning ist das Modell in der Lage gewesen, konsistente, formal gültige Antworten zu generieren [43].

Abbildung 5.3: Plugin zur OWL-Syntax Generierung in Protégé [43]



Evaluiert haben Autorin und Autor die Ergebnisse anhand einer von ihnen bezeichneten ”Family Ontology” also einem Familienstammbaum, der um zusätzliche Relationen

erweitert ist. Insgesamt sind durch die Verwendung von Finetuning im Test mit unterschiedlichen Szenarien konsistente Ergebnisse entstanden, aber gelegentlich hat das Modell auch nicht zutreffende oder auch überflüssige Axiome erstellt, was laut den beiden Forschenden auf die inhärente Unsicherheit solcher Sprachmodelle zurückzuführen ist. Um erstellte Inkonsistenzen nicht automatisch in die Ontologie zu übernehmen, können Anwendende im Plugin das Ergebnis überprüfen, bevor es manuell über einen ”ADD” Button in die Ontologie übernommen wird.

5.2.2 Extraktion von kausalen Relationen mit LLMs in der Medizin

Neben der Erweiterung bestehender Ontologien gibt es auch Ansätze, die Ontologien und LLMs kombinieren, um Informationen in praktischen Anwendungen in tatsächlichen Domänen zu verarbeiten. Lecu et al. zeigen in ihrer Arbeit ein erstelltes System zur Extraktion von kausalen Relationen aus medizinischen Abstracts. Dies erfolgt durch ein per Finetuning trainiertes GPT-Modell und eine domänenspezifische Ontologie, die für diesen Anwendungsfall entwickelt worden ist, die CausalAMD-Ontologie [32]. Diese Ontologie bildet die Grundlage für die Extraktion und auch für die Strukturierung von Aussagen über altersbedingte Makuladegeneration (AMD), daher auch der Name CausalAMD-Ontologie. Bei AMD handelt es sich um eine chronische Netzhauterkrankung, die über die Zeit zum Verlust des Sehvermögens führt. Die Ontologie definiert dabei 12 verschiedene Entitäten, wie Krankheit, Symptom, Behandlung oder Körperteil, sowie auch acht Relationen.

Zu Beginn haben die Autorinnen und Autoren 70 medizinische Abstracts aus der Datenbank ”Dimensions” unter Anwendung von verschiedenen Inklusionskriterien ausgewählt. Die wichtigen kausalen Relationen innerhalb der Abstracts sind von jeweils zwei Beteiligten annotiert worden, anhand einer zuvor definierten Guideline. Die Annotationen dienen als Trainingsdaten für das Finetuning des GPT-Modells. Dieses trainierte Modell überführt die Relationen aus neuen nicht-annotierten Abstracts daraufhin in ein RDF-Format und importiert diese in die grafenbasierte Datenbank GraphDB. Die Ergebnisse der erstellten Relationen erreichten eine Precision von 76,8 Prozent im Vergleich zu den manuell annotierten Relationen, wobei aber auch Relationen entdeckt werden konnten, die im manuellen Prozess übersehen worden sind [32].

Abbildung 5.4: Generierte Relationen durch das trainierte GPT-Modell [32]

```
[{"entity1_name": "Age-related macular degeneration",     "entity1_type": "disease",
    "entity2_name": "blindness",                          "entity2_type": "symptom",
    "relation_type": "cause"},
 {"entity1_name": "Age-related macular degeneration",     "entity1_type": "disease",
    "entity2_name": "eye",                                "entity2_type": "body_part",
    "relation_type": "affect"},
 {"entity1_name": "smoking",                              "entity1_type": "risk_factor",
    "entity2_name": "Age-related macular degeneration",   "entity2_type": "disease",
    "relation_type": "present"}]
```

Auf die in GraphDB geladenen Relationen ist ein Reasoner angewendet worden, der sieben weitere Axiome und Relationen definiert. Um Fehler durch das KI-Modell zu vermeiden, erlaubt das Framework auch die manuelle Korrektur von Relationen durch Endanwendende. Die entwickelte Ontologie ermöglicht zudem das Eingeben von SPARQL-Anfragen, um so Nutzeranfragen im Hinblick auf AMD stellen zu können [32].

Das entwickelte Framework berücksichtigt allerdings lediglich positive kausale Relationen. Es trifft also keine negativen Aussagen über kausale Relationen, wie ”Behandelt eine Krankheit, aber heilt sie nicht vollständig”, was von den Forschenden auch als Nachteil erkannt wird und auf die Problematik von LLMs hinweist, mehrdeutige, implizite und unklare Informationen zu extrahieren. Die Studie zeigt, wie Ontologien und große Sprachmodelle aktuell in der Praxis im medizinischen Bereich kombiniert werden können. So lässt sich semantisches Wissen über Zusammenhänge von Krankheitsbildern in Ontologien schneller, nachvollziehbarer und effizienter abbilden.

5.2.3 Ontologien im Zeitalter von LLMs - Konkurrenz oder Ergänzung

Sollten große Sprachmodelle in den kommenden Jahren fähiger werden und verbesserte Fehlertoleranzen entwickeln, stellt sich die Frage, ob Ontologien im Zeitalter von generativen Sprachmodellen nach und nach obsolet werden. Dieser Frage ist Neuhaus mithilfe einer fundierten Analyse auf den Grund gegangen. Während LLMs bereits in der Lage sind, präzise Antworten zu generieren, Sprache zu verarbeiten und auch weniger komplexe OWL-Ausgaben zu erzeugen, mangelt es ihnen aktuell noch an formaler Konsistenz und semantischer Eindeutigkeit [50].

Neuhaus betrachtet in der Analyse unter anderem, inwiefern LLMs in der Lage sind, kleinere Ontologien über Zero-Shot Prompting, also lediglich über das Verständnis des KI-Modells in der Interpretation von beispiellosen Prompts, zu generieren. Dabei fällt in manuellen Überprüfungen auf, dass qualitative Mängel vorliegen, weil Instanzen beispielsweise fälschlich als Klassen definiert werden, oder Subklassen nicht korrekt interpretiert werden. Zudem sind LLMs zwar in der Lage, diese mit zusätzlichen Prompt Engineering Strategien zu generieren, ohne dabei grundlegende Konsistenzfehler aufzuweisen, allerdings werden die Ergebnisse mit steigender Komplexität auch schwieriger zu überprüfen und der Mangel an symbolischer Überprüfbarkeit in der Generierung von Ergebnissen erschwert zusätzlich die Konsistenzprüfung [50].

Eine weitere Herausforderung in der Kombination von Ontologien und LLMs, die Neuhaus identifiziert, ist die Ambiguität von großen Sprachmodellen. Die Transformer-Architektur und der Self-Attention-Mechanismus von LLMs sorgen dafür, dass Sprachmodelle Begriffe in natürlicher Sprache kontextabhängig interpretieren. So kann ein ”Blatt” sowohl für eine DIN A4-Seite eines Druckerpapiers stehen, als auch für das Blatt eines Kastanienbaums, abhängig von dem umliegenden Kontext. Für Ontologien ist diese Ambiguität aber ein Nachteil. Ontologien benötigen eine klare und in sich konsistente Definition von Begriffen. Durch diese Eigenschaft lassen sich ggf. entstehende Inkonsistenzen bei der Kombination beider Methoden nicht vermeiden [50].

Trotz dieser Einschränkungen sieht Neuhaus dennoch Potenzial für beide Technologien. So können LLMs Ontologien zukünftig bei Routinearbeiten unterstützen, wie die Definition einzelner Axiome, dem Verständnis von Konzepten oder dem Bilden von Vorkonstrukten für zukünftige Ontologien aus freien Texten heraus. Anwendungen, wie die durch Mateiu und Groza durchgeführte Implementierung [43] zeigen bereits Beispiele für solche Anwendungen. Auch hebt Neuhaus den Aspekt der Qualitätssicherung hervor. So können LLMs dabei unterstützen, durch den Vergleich zwischen natürlichsprachigen Definitionen

und OWL-Modellen Inkonsistenzen zu erkennen und Verbesserungsvorschläge zu treffen. Voraussetzung hierfür ist aber stets noch eine manuelle Überprüfung durch einen Experten oder eine Expertin. Damit kommt Neuhaus insgesamt bei dem Ergebnis an, dass Ontologien insbesondere in Bereichen mit einem Bedarf nach hoher semantischer Präzision sowie Nachvollziehbarkeit unverzichtbar bleiben. LLMs können aber als komplementäre Werkzeuge dienen und den Prozess der Ontologieerstellung beschleunigen. Dabei ist allerdings zu beachten, wie komplex die Ontologien sein können, ohne dass zu große Inkonsistenzen in der LLM-gestützten Entwicklung entstehen [50].

5.2.4 Lücken und Forschungsbedarf

Die vorherigen Abschnitte verdeutlichen, dass die Integration großer Sprachmodelle und Ontologien zwar in der aktuellen Forschung behandelt wird, aber dennoch wesentliche Herausforderungen bestehen. Zum einen zeigt sich, dass zwar eine Vielzahl von Ansätzen zur Verbesserung der Erklärbarkeit existiert, insbesondere durch Frameworks, wie SHAP, LIME oder Abwandlungen verschiedener Prompt Engineering Strategien, jedoch bleibt die Frage offen, wie zuverlässig und zielgruppengerecht diese Ansätze sind. Die Erklärbarkeit von KI-Modellen wird lediglich als additiver Zusatz gesehen und nicht als integrierter Bestandteil der Modelle, wodurch die Verständlichkeit der Erklärbarkeit von dem technischen Know-how der Anwendenden abhängt. Die dargestellten Arbeiten zeigen das steigende Interesse an hybriden Ansätzen. Zwar konnten bereits erste Erfolge durch die Generierung von OWL-Aussagen wie durch Mateiu und Groza [43] präsentiert werden, jedoch bleiben zentrale Lücken bestehen.

- Bisherige Arbeiten adressieren einzelne isolierte Aspekte der Integration, wie die Generierung einzelner OWL-Aussagen in einem spezifischen Anwendungsfall. Eine vergleichende Untersuchung von verschieden komplexen Anwendungsfällen fehlt.
- Unklare Validität entsteht durch den Mangel an ausführlicher Validierung der Semantik generierter Aussagen.
- Ein Mangel an Generalisierbarkeit und Skalierbarkeit entsteht durch eine fehlende Evaluierung in der Generierung größerer OWL-Dokumente.

Diese offenen Forschungslücken bilden den Ausgangspunkt für den methodischen Rahmen dieser Arbeit. Ziel ist es, systematisch zu untersuchen, inwiefern aktuelle Versionen eines Sprachmodells durch gezieltes Prompting in der Lage sind, große, OWL-konforme Aussagen zu generieren, sowie basierend auf den erstellten Ontologien dabei ein gewisses Maß an semantischer Konsistenz zu halten. Dabei wird besonders auf syntaktische Korrektheit, logische Konsistenz und basierend auf der jeweiligen Ontologie auch auf fachliche Konsistenz geachtet. Im folgenden Kapitel wird die geplante Methodik konkretisiert durch die Beschreibung der zu entwickelnden Ontologien sowie das Design der Prompt Engineering Strategien und der verwendeten Evaluationsmetriken.

6
Methodik

Ziel dieses Kapitels ist es, die methodische Vorgehensweise der Arbeit detailliert zu beschreiben. Dabei orientiert sich die Vorgehensweise an der zentralen Fragestellung der Arbeit und den sich daraus abbildenden Unterfragen, ob LLMs in der Lage sind, OWL-Axiome zu interpretieren, darauf basierend Fakten zu erzeugen und auch in Kombination mit den zugrundeliegenden Ontologien konsistente Aussagen zu generieren, um damit zur Erklärbarkeit von KI-Modellen beizutragen. Die Arbeit verfolgt dabei einen experimentellen Ansatz zur Untersuchung. Dabei erfolgt die Umsetzung in drei aufeinanderfolgenden Schritten.

- Zu Beginn werden drei Ontologien von verschieden hoher Komplexität entwickelt, die als Grundlage für die nachfolgenden Experimente fungieren.
- Anschließend werden mehrere Prompt Engineering Strategien angewendet, um OWL-Aussagen zu den jeweiligen Ontologien auf Basis natürlichsprachiger Texte zu generieren. Die verwendeten Strategien sind Zero-Shot Prompting, Few-Shot Prompting, Retrieval Augmented Generation (RAG) und Finetuning. Das verwendete Sprachmodell für diese Arbeit ist dabei GPT in der Version 4o.
- Zuletzt wird die Qualität der generierten Ergebnisse anhand von verschiedenen syntaktischen, semantischen und logischen Eigenschaften systematisch evaluiert. Hierfür werden standardisierte Metriken, wie Precision und Recall verwendet. Die identifizierten Fehler werden zudem geclustert und Fehlerklassen zugeordnet, um eine Bewertung der Schwachstellen des Modells durchzuführen.

Neben diesen Bewertungsmetriken werden auch andere Methoden eingesetzt. Methoden, wie die Jaccard-Ähnlichkeit, können zur semantischen Übereinstimmung der Ergebnisse mit manuell generierten Dokumenten verwendet werden. Zudem lassen sich die Ergebnisse der manuell hinzugefügten Fakten und der generierten Fakten in Protégé nach Anwendung eines Reasoners vergleichen, um logische Verstöße gegen Restriktionen der jeweiligen Ontologie zu erkennen.

6.1 Ontologieentwicklung

Der erste Schritt ist die Erstellung der drei Ontologien, die als Grundlage für die Durchführung dienen. Ziel hierbei ist es, durch die unterschiedlichen Komplexitätsstufen zu evaluieren, ob syntaktische Korrektheit sowie semantische Tiefe und logische Konsistenz der durch die LLMs generierten Ergebnisse gegeben sind. Die drei Ontologien bestehen aus einer taxonomischen Tierklassifikation und einer komplexeren Ontologie, die die Habsburger Familiendynastie abbildet. Zuletzt auch einer komplexen domänenspezifischen Ontologie, die medizinische Zusammenhänge zwischen Krankheiten, Symptomen und

Therapiemöglichkeiten darstellt und Patienten mit ihren Krankheitsbildern speichert, um Therapievorschläge zu treffen.

Die Tierklassifikation wird eine hierarchische Struktur repräsentieren. Diese Ontologie hat die geringste Komplexität und dient als Möglichkeit, um eine Basis zu schaffen, die grundsätzliche Fähigkeit des Sprachmodells, taxonomische Relationen korrekt zu verstehen und neue Fakten zu generieren, zu testen. Die Ontologie enthält daher bewusst nur Klassen, Data Properties, sowie Unterklassen- und Disjunktheitsaxiome, um den Fokus auf das taxonomische Verständnis zu legen. So lässt sich über diese Ontologie evaluieren, ob das LLM in der Lage ist, Subklassenhierarchien zu erkennen und diese sowie disjunkte Klassen korrekt darzustellen, ohne dabei logisch widersprüchliche Strukturen zu generieren.

Die Familienstruktur modelliert eine komplexere Beziehungsstruktur als nur eine Taxonomie. Diese Ontologie beinhaltet neben Klassen nun auch Instanzen, sowie auch Object- und Data Properties. Ziel ist es zu überprüfen, ob das KI-Modell komplexe relationale Axiome interpretieren und basierend darauf Fakten generieren kann. Um einen gewissen anspruchsvollen Umfang einer solchen Familienstruktur zu gewährleisten, wird eine Teilmenge der Habsburger Familiendynastie abgebildet, wodurch stark verzweigte Familien- und Heiratsbeziehungen abgebildet werden können. So lassen sich weitere Fähigkeiten überprüfen, wie das Interpretieren von inversen Eigenschaften, Kardinalitäten oder auch bidirektionalen Beziehungen (z.B. Ehe).

Die dritte und komplexeste Ontologie basiert auf medizinischen Daten über Krankheiten und Krankheitsbilder, sowie zugehörige Symptome, Therapiemöglichkeiten und Medikamente. Sie ist dabei inspiriert von den bisher verwendeten Beispielen im Grundlagenteil und auch von präsentierten Ontologien, wie CausalAMD, ist jedoch generalisiert. Als Fakten, die das Sprachmodell in OWL übertragen muss, bieten sich in dieser Ontologie beispielsweise Krankheiten an, die bisherige Symptome und Beschwerden, sowie Behandlungsmöglichkeiten und Diagnoseverfahren beschreiben. Durch eine Ontologie solcher Komplexität sollen die Grenzen im Verständnis und in der Generierung in einem domänenspezifischen Anwendungsfall durch das LLM geprüft werden. Konkret bedeutet das ein Verständnis von medizinischem Wissen im Kontext der Ontologie, sowie ein Verständnis multipler Klassen und Relationen mit verschiedenen Restriktionen, als auch eine potenzielle fachliche Diagnostikunterstützung. Durch die Anwendung verschiedener Prompt Engineering Strategien sowie drei Ontologien von unterschiedlicher Komplexität kann so eine ausführliche Evaluation der Fähigkeiten von LLMs erfolgen, um systematisch die Möglichkeiten und aktuellen Grenzen aufzuzeigen.

6.2 Prompt Engineering und OWL-Generierung

Nachdem die Ontologien entwickelt sind, gilt es verschiedene Prompt Engineering Strategien einzusetzen, um auf Basis der gegebenen Ontologien syntaktisch und semantisch korrekte OWL-Fakten aus natürlichsprachlichen Texten zu generieren. Die Hypothese ist hierbei, dass sich durch Prompt Engineering mithilfe von LLMs auf Basis von gegebenen Ontologien syntaktisch korrekte und logisch konsistente Fakten in OWL ableiten lassen. Um diese Hypothese zu überprüfen, werden die zuvor erstellten Ontologien als Grundlage verwendet, um den Prozess der Evaluation in fünf Schritten durchzuführen.

1. Zuerst werden für jede der drei Ontologien 10 repräsentative Dokumente mit natürlichsprachlichen Aussagen erstellt (Einträge über bekannte oder unbekannte Tiere, biografische Einträge der Habsburger Familiendynastie oder Beschreibungen von Krankheiten und Krankheitseigenschaften). Basierend auf diesen Aussagen werden manuell OWL-Dokumente aufgesetzt, die formal validiert werden, um als Vergleich zu dienen.

2. Unter Anwendung der verschiedenen Prompt Engineering Strategien werden diese insgesamt 30 Dokumente mit jeder dieser Strategien an das Sprachmodell übergeben, um Antworten in OWL-Syntax zu generieren. Für bestimmte Prompt Engineering Strategien müssen hierfür zuvor noch zusätzliche Dokumente erstellt werden, die den Prompts als Beispiele für das Sprachmodell beigelegt werden.

3. Die generierten Dokumente werden manuell und unter Einsatz eines Validators automatisch auf syntaktische Konsistenz geprüft.

4. Die Ergebnisse werden daraufhin auf Basis der in den nächsten Abschnitten beschriebenen Metriken semantisch evaluiert.

5. Zuletzt werden, basierend auf den entstandenen Fehlern in der Generierung, Fehlerklassen identifiziert, denen die fehlerhaften Aussagen zugeordnet werden. So wird ein Überblick über die Schwachstellen der OWL-Axiomerstellung gewonnen und die Fehlerklassen werden bewertet.

Für die Experimente wird die GPT-4-Architektur verwendet, genauer gesagt die Version 4o. Diese soll laut OpenAI bessere Leistungen in den Bereichen erbringen, die strukturelles Verständnis und konsistente Textgenerierung benötigen. Zum Zeitpunkt dieser Arbeit ist es zudem das Kernmodell des Konzerns.

Die erste verwendete Strategie ist das Zero-Shot Prompting. Diese Methode ermöglicht das Stellen der Minimalanforderung an das Modell, da hierbei lediglich die Anforderung an das KI-Modell gestellt wird, ohne dabei Erläuterungen oder kontextbezogene Beispiele in den Prompt zu integrieren. Diese Strategie wird verwendet, um zu prüfen, ob das Modell bereits ohne zusätzliche Unterstützung OWL-Strukturen erzeugen kann. Da diese Strategie aber auch am wahrscheinlichsten zu syntaktischen und semantischen Fehlern in der Interpretation führt, dienen die Ergebnisse auch als Vergleichswert für die anderen Strategien. Zur Verbesserung des kontextuellen Verständnisses wird daraufhin Few-Shot Prompting eingesetzt. Hierbei werden den Prompts für jede der drei Ontologien drei zusätzliche Anfrage-Antwort-Kombinationen beigelegt, die das korrekte erwartete Format zeigen. So entsteht ein kontextueller Rahmen für Stil und Struktur der gewünschten Antworten [6]. Es wird daher erwartet, dass die Verwendung dieser Strategie zu einer erhöhten Konsistenz und engeren Einhaltung des gewünschten Formats führt.

Für eine zusätzliche Verbesserung des Kontextes wird als dritte Strategie Retrieval Augmented Generation eingesetzt. Dem Modell werden nun, anstatt im Prompt integrierte Beispiele zu verwenden, unterstützende Dokumente neben dem Prompt beigefügt, auf die das Sprachmodell zugreifen kann, um die Antworten zu erzeugen. In diesem Fall werden dem Modell das OWL-Dokument der Ontologie mit allen enthaltenen Axiomen, sowie ein ausführliches Dokument mit mehreren Seiten und insgesamt fünf Anfrage-Antwort-Paaren

beigefügt [35]. Dabei wird sich von den Ergebnissen eine noch höhere syntaktische Konsistenz und eine geringere Anzahl von Halluzinationen erhofft.

Als letzte Strategie wird Finetuning eingesetzt. Ähnlich zu den bisherigen Strategien werden dem Modell beim Finetuning Anfrage-Antwort-Paare beigefügt. Anstatt diese Dokumente aber als eine Wissensbasis dem Modell beizufügen, werden die Paare beim Finetuning zum Trainieren des Modells verwendet. Bei dieser Strategie werden im Rahmen dieser Arbeit JSONL-Dateien erstellt (zeilenbasiertes JSON-Format), die diese Paare beinhalten, und einem 4o-Modell als Trainingsdaten gegeben werden. Das Modell steht nach erfolgtem Training über die Oberfläche oder über die OpenAI-API zur Verfügung [53]. Die Anfragen werden bei Bedarf auch zusätzlich zur angewendeten Strategie durch Chain-of-Thought (CoT) Prompting unterstützt, um das Verständnis über die Ergebnisse zu verbessern [79]. Neben den Prompt Engineering Strategien folgen die Prompts auch der durch Microsoft empfohlenen Struktur für die Formulierung von Prompts [48]. Diese ist im Grundlagenteil bereits erläutert worden. Insgesamt entstehen so für vier Prompt Engineering Strategien und 10 Dokumente bei drei verschiedenen Ontologien 30 * 4 OWL-Dokumente, die im Anschluss evaluiert werden. In diesem Fall also 120 generierte Dokumente.

6.3 Methodik zur Evaluation der Ergebnisse

Um die Qualität der Ausgaben und die Eignung der durch ein LLM erzeugten OWL-Fakten systematisch bewerten zu können, werden die Ergebnisse der verschieden komplexen Ontologien unter Verwendung der Strategien in mehreren Schritten anhand von standardisierten Metriken bewertet. Es werden alle Ergebnisse, die durch das LLM erzeugt werden, zuerst auf syntaktische Korrektheit geprüft. Die Prüfung erfolgt manuell und über einen OWL-Validator. Fehlerhafte Dokumente werden gesondert zu den restlichen Ergebnissen protokolliert, die Fehlerhäufigkeit wird dokumentiert und sie gelten als syntaktisch ungültige Ergebnisse. Abhängig davon, wie viele solcher Fehler auftreten, wird den Ergebnissen eine Zahl FZQ (Fehlerzahlquote) zugewiesen, wobei FZQ für die Anzahl der auftretenden syntaktischen und semantischen Fehler im Verhältnis zur Anzahl der generierten Aussagen im Dokument steht. Zu den dokumentierten Fehlerarten für syntaktische Fehler gelten unter anderem

- Ungültige Schlüsselwörter oder fehlerhafte Strukturen im Dokument.
- Verwendung von ungültigen Zeichen oder Zeichenketten.
- Formatierungsfehler in der Verwendung von Klammern und Zeichensetzung.
- Auslassen von notwendigen Aussagen wie vorherige Präfix-Deklarationen.

Zusätzlich zu der syntaktischen Konsistenz werden auch standardisierte Metriken zur semantischen Evaluation verwendet. So erfolgt zunächst für alle gültigen Ergebnisse ein Vergleich mit den manuell erstellten OWL-Fakten-Dokumenten. Jedes der 30 * n durch das LLM generierten Dokumente wird mit einem der jeweils zugehörigen 30 manuell erstellten Dokumente verglichen. Zur Bewertung der faktischen Übereinstimmung zwischen den Ergebnissen und den Referenzdateien werden zum einen die Metriken Precision, Recall und der F1-Score eingesetzt. Dabei handelt es sich um Metriken aus dem Bereich des maschinellen Lernens und des Information Retrievals, die die Relevanz von Rückgabewerten

bewerten. Für das Verständnis dieser Metriken im Rahmen der Arbeit müssen zunächst drei Kategorien von Rückgabewerten des Sprachmodells definiert werden.

- True Positives (TP) beschreiben richtig generierte OWL-Aussagen, also Aussagen, die durch das LLM erzeugt worden sind und auch in dem manuell erstellten Referenzdokument enthalten sind.
- False Positives (FP) beschreiben Aussagen, die zwar generiert worden sind, aber nicht in den Referenzdokumenten auftreten
- False Negatives (FN) sind Aussagen, die in der Referenz zwar vorhanden sind, aber nicht von dem Sprachmodell erzeugt wurden.

Einträge gelten hierbei als Fehler, wenn nicht in der Ontologie vorhandene Klassen oder Properties verwendet werden, Data Properties angegeben werden, deren Informationen nicht im Referenzdokument stehen oder falsche Zuordnungen von Individuen zu Klassen stattfinden. Auch das Fehlen von Aussagen gilt als Fehler. Da für eine potenzielle spätere Möglichkeit der automatisierten Integration von LLMs und Ontologien in späteren Arbeiten die genaue Bezeichnung von Elementen eine wichtige Rolle spielt, gelten auch Unterschiede in der Namenskonvention von Klassen als Fehler. Sind in der zugrundeliegenden Ontologie alle Klassen im Plural formuliert und das Sprachmodell verwendet die Klasse ”Braunbär” im generierten Dokument anstatt der Klasse ”Braunbären”, wie es im Referenzdokument der Fall ist, gilt dies als FN und FP zugleich. Es fehlt hier die korrekte Klasse und zugleich ist auch eine falsche Klasse deklariert worden. Auf Grundlage dieser Kategorien von Rückgabewerten lassen sich nun die drei Metriken berechnen.

Precision beschreibt den Anteil der relevanten Aussagen im Verhältnis zu allen generierten Aussagen des Sprachmodells. Es gibt also an, wie viele der erzeugten Aussagen im Verhältnis zu allen generierten Aussagen TP sind. Formal ausgedrückt ist dies

$$Precision(P) = \frac{TP}{TP + FP}$$

Recall (R) berechnet sich durch den Anteil der vom Modell korrekt generierten Aussagen im Verhältnis zu allen Aussagen, die im Referenzdokument vorkommen. Diese Metrik misst also, wie vollständig das Sprachmodell relevante Aussagen generieren kann.

$$Recall(R) = \frac{TP}{TP + FN}$$

Der F1-Score ist eine kombinierte Metrik der vorherigen beiden Werte, die das harmonische Mittel von Precision und Recall angibt. Der F1-Score repräsentiert ein Verhältnis zwischen der präzisen und der vollständigen Erzeugung korrekter Aussagen [10].

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

Die beschriebenen Metriken werden sowohl auf einzelner Ebene der 30 * n generierten Ergebnisse berechnet, als auch als Durchschnittswerte für alle 10 OWL-Dokumente, die pro Ontologie durch eine Prompt Engineering Strategie erzeugt worden sind.

Zusätzlich zu diesen drei Metriken lassen sich die generierten Dokumente und die Vergleichsdokumente auf ihre Mengenähnlichkeit überprüfen. Hierfür wird die Jaccard-Ähnlichkeit angewendet. Dabei handelt es sich um eine Metrik zur quantitativen Bewertung der Übereinstimmung zweier Mengen [65]. Wenn also A die Menge aller Aussagen im Referenzdokument ist und B die Menge aller Aussagen im generierten OWL-Dokument, dann beschreibt die Jaccard-Ähnlichkeit das folgende Verhältnis.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Ein Wert von 1 beschreibt also eine vollständige Übereinstimmung beider Mengen und ein Wert von 0 eine leere Schnittmenge. Im Vergleich zu den vorherigen Metriken ist die Jaccard-Ähnlichkeit weniger anfällig gegenüber leicht unterschiedlichen, aber dennoch semantisch ähnlichen Aussagen, wodurch eine zusätzliche Perspektive durch diese Metrik geboten wird. Es ist dabei allerdings festzulegen, welche Aussagen als TP gelten und welche nicht.

- Generiert das LLM eine Aussage zu einer Object Property, die invers zu der ist, die im Referenzdokument steht, gilt die Aussage als TP, obwohl es sich nicht um dieselbe Aussage handelt. Ist im Referenzdokument der Familienontologie also beispielsweise die Rede von ”A hatKind B” und das Sprachmodell generiert die Aussage ”B istKindVon A”, wobei istKindVon und hatKind als inverse Properties definiert sind, so gilt dies nicht als Fehler und die Aussage gilt als TP. Dies gilt aber nur wenn die inverse Eigenschaft auch in der Ontologie als solche definiert ist.
- Redundante Aussagen gelten auch nicht als Fehler. Dabei gelten redundante Aussagen in den meisten Fällen als solche, wenn diese durch die Anwendung eines Reasoners ohnehin abgeleitet worden wären. Generiert das Sprachmodell beispielsweise Aussagen für beide Properties, die in einer inversen Beziehung stehen, oder definiert das LLM ein Individuum als Instanz einer Klasse und dessen Oberklasse, ist dies kein Fehler. Zu redundanten Aussagen zählen auch Data Properties, wie Beschreibungen, die zwar nicht in dem Referenzdokument vorhanden sind, sich aber dennoch aus dem Text entnehmen lassen und lediglich nicht in das Referenzdokument übernommen worden sind.
- Kommentare im generierten Dokument gelten ebenfalls nicht als Fehler auch wenn diese nicht im Referenzdokument vorhanden sind.

Neben solchen Metriken werden die generierten OWL-Fakten auch auf ihre logische Konsistenz geprüft. Durch die Verwendung von Reasoning Engines in Protégé werden die Ergebnisse auf logische Konsistenz überprüft, um potenzielle Widersprüche zu identifizieren. Dabei geht es um Konsistenz im eigentlichen Sinne, also der Überprüfung, ob keine widersprüchlichen Fakten generiert worden sind und um Inferenzprüfung, also der Überprüfung, ob die Reasoner auf Basis der neuen Fakten korrekt ableiten können. Fehler hier treten auf, wenn das Sprachmodell Fakten generiert, die den in der Ontologie definierten Restriktionen und Eigenschaften von Klassen und Properties widersprechen. Abweichungen und Inkonsistenzen werden festgehalten und als solche dokumentiert.

Nachdem mithilfe einer Prompt Engineering Strategie zu einer Ontologie alle Ergebnisse erzeugt und evaluiert worden sind, werden die gefundenen Fehler zudem inhaltlichen

Fehlerklassen zugeordnet, um zu evaluieren, ob bestimmte Fehlerklassen öfter auftreten als andere. Diese häufigen Fehlerklassen zu identifizieren soll Anwender und Anwenderinnen dabei unterstützen zu wissen, auf welche möglichen Fehler bei generierten Dokumenten besonders zu achten ist. Alle Ergebnisse in der Experimentreihe werden dokumentiert, die numerischen Metriken der Evaluationen werden tabellarisch dargestellt und die erzeugten graphischen Darstellungen und Evaluationsdokumente der Ontologien werden zuletzt auch im elektronischen Anhang der Arbeit hinterlegt.

Das in diesem Kapitel vorgestellte methodische Vorgehen bildet damit die Grundlage für die folgende experimentelle Untersuchung der Integrationsmöglichkeit von LLMs und Ontologien im Kontext von XAI. Durch die Evaluation anhand mehrerer Prompt Engineering-Strategien und zunehmend komplexer Testontologien wird ein Testfeld für LLMs in diesem Kontext geschaffen, das differenziert und systematisch ihre Leistungsfähigkeit unter verschiedenen Anforderungen prüft. Durch die verschiedenen Prompt Engineering-Strategien werden nicht nur die Ergebnisse bei unterschiedlichen Anforderungen geprüft, sondern auch der Einfluss der Eingabemethodik auf die Qualität der generierten OWL-Aussagen analysiert.

Durch umfassende Evaluationsmetriken, die von standardisierten Kennzahlen, wie Precision, Recall oder F1-Score, sowie Konsistenzprüfungen der Dokumente und qualitativen Auswertungen, wird gewährleistet, dass die Ergebnisse syntaktisch, semantisch und auch logisch evaluiert werden. Insgesamt legt das Kapitel damit die Basis für eine systematische Beantwortung der zentralen Forschungsfrage, inwiefern die Integration von LLMs und Ontologien verwendet werden kann, um die Erklärbarkeit von Sprachmodellen zu verbessern. Zudem wird auch beantwortet, inwiefern die Grundlage hierfür durch die korrekte Modellierung von OWL-Dokumenten durch Sprachmodelle gegeben ist. Nun gilt es mit der tatsächlichen Durchführung zu beginnen.

7 Durchführung

Nun da das methodische Vorgehen erläutert ist, kann die eigentliche Durchführung der experimentellen Evaluation erfolgen. Ziel dieses Kapitels ist die Umsetzung der zuvor beschriebenen Methodik. Aufbauend auf der Hypothese, dass große Sprachmodelle durch den gezielten Einsatz von Prompt Engineering-Strategien in der Lage sind unterschiedlich komplexe OWL-Aussagen zu erzeugen und somit die Entwicklung von Ontologien durch die Integration von LLMs zu unterstützen, wird das Vorgehen detailliert dokumentiert. Die Durchführung gliedert sich dabei in drei Abschnitte.

1. Erstellung der Testontologien
2. Erstellung der Referenzdokumente für die jeweiligen Ontologien
3. Anwendung der Prompt Engineering Strategien zur OWL-Generierung

Die Abschnitte werden jeweils noch weiter unterteilt, um das schrittweise Vorgehen in der Durchführung verständlich und übersichtlich abzubilden. So wird der erste Abschnitt zusätzlich in eine Sektion für jede Testontologie eingeteilt. Die Evaluation der Ergebnisse unter Verwendung der beschriebenen Metriken erfolgt im Anschluss im darauffolgenden Kapitel der Ergebnispräsentation. Die Durchführung beginnt mit der Erstellung der ersten Ontologie, der Taxonomie zur Klassifikation von Tierarten.

7.1 Erstellung von Test-Ontologien

In dieser Sektion geht es darum, die Struktur, den Aufbau und die Charakteristika der drei Test-Ontologien systematisch und nachvollziehbar zu beschreiben, um so eine Basis für die späteren Evaluationsschritte zu legen. Jede der drei Ontologien wird zunächst einleitend beschrieben. Daraufhin wird der Aufbau der einzelnen Ontologien geschildert, sprich welche Klassen, Relationen und Axiome verwendet werden. Es wird aufgelistet, welche Komplexitätsmerkmale in den Ontologien auftreten, also zusätzliche Konstrukte, wie Kardinalitäten oder Restriktionen. Zusätzlich werden repräsentative Ausschnitte in OWL-Syntax zur Verdeutlichung hinzugezogen. Eine grafische Darstellung jeder Ontologie wird im Anhang beigefügt und zuletzt wird die Relevanz der Ontologien für die spätere Evaluation eingeordnet. Die Ontologien werden in Reihenfolge steigender Komplexität erstellt. Die erste Ontologie ist die Taxonomie zur Klassifikation von Tierarten.

7.1.1 Einfache Taxonomie

Die erste Testontologie für die Evaluation ist eine taxonomische Einordnung von tierischen Lebewesen. Ziel dieser Ontologie ist es, eine formal korrekte hierarchische Klassenstruktur abzubilden, die durch ihre konzeptionelle Einfachheit als Einstiegspunkt

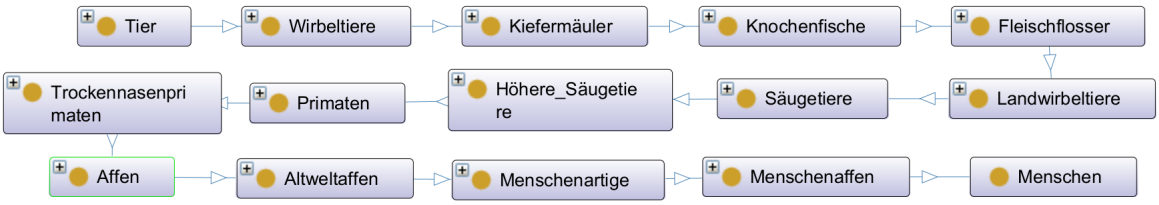
für die Analyse dient. Das bedeutet, dass die Hauptelemente dieser Ontologie Klassen mit Subklassenbeziehungen sind. Die Ontologie beinhaltet dabei auch keine konkreten Individuen, sondern lediglich eine taxonomische Klassenstruktur. Durch die geringe Komplexität der Beziehungen der Elemente untereinander, aber eine ausführliche hierarchische Unterklassenstruktur, eignet sie sich für eine erste Überprüfung. Der Fokus liegt auf der Fähigkeit des Sprachmodells, generierte Aussagen korrekt in die Struktur einer Ontologie einzuordnen und ähnlich bezeichnete Klassen voneinander zu unterscheiden.

Wichtig hierfür ist, dass die Tierklassifikation im bestmöglichen Maß fachlich korrekt aufgestellt ist. Dies ist relevant, um zu verhindern, dass das Sprachmodell während der Testreihe Aussagen aufgrund von einer abweichenden Klassifikation durch die Ontologie und der Vorkenntnis des Modells durch im Internet zugängliche Informationen falsch einordnet. Dies bedeutet, dass bestimmte Familien und Ordnungen von Tierarten ausgelassen werden können, aber keine nicht existierenden oder fälschlich zugeordneten Klassen in der Ontologie eingeordnet werden. Zudem wird vermieden allzu viele Subklassen zu überspringen und beispielsweise den Grauwolf nicht fälschlich als direkte Subklasse von Säugetieren zu modellieren.

Inhaltlich basiert die Ontologie auf einer öffentlich zugänglichen Sammlung von Taxonomien der Plattform ”Catalogue of Life”, auf der Inhalte durch verschiedene Expertendatenbanken und Fachwissende stetig ergänzt und überarbeitet werden [12]. Neben Pflanzen und Pilzen besteht dort eben auch eine taxonomische Auflistung der Tierwelt. Zum Zwecke der Übersichtlichkeit ist die taxonomische Struktur für diese Arbeit vereinfacht worden. Dies erfolgt, da die Modellierung von den in dieser Wissensdatenbank vorhandenen 1,5 Millionen tierischen Spezies nicht zeitlich umsetzbar sowie unübersichtlich ist und für das Sprachmodell eine zu große Dokumentengröße darstellen würde. Damit bildet die Ontologie eine übersichtliche, aber dennoch strukturierte Teilmenge der tatsächlichen Menge des Tierreichs ab, die sich auf den Unterstamm der Wirbeltiere fokussiert und damit die bekannten tierischen Großgruppen der Fische, Amphibien, Reptilien, Vögel und Säugetiere sowie weitere Gruppen beinhaltet. Wirbellose Tiere (eigentlich Gliederfüßer) werden als Oberklasse definiert, aber nicht weiter in Subklassen unterteilt. Zudem ist zu erwähnen, dass für diese Ontologie die moderne Betrachtung der Taxonomie des Tierreichs verwendet worden ist, welche Tiere nach ihren evolutionstheoretischen Abstammungen einteilt. Dies führt vor allem dazu, dass Reptilien, Amphibien und auch Säugetiere unter die Knochenfische fallen und auch Vögel den Reptilien zugeordnet sind, da diese direkte Nachfahren sind. Alle Klassen werden, wenn möglich, mit ihrer deutschen Bezeichnung betitelt.

Insgesamt umfasst die Ontologie 521 Klassen mit einer Oberklasse ”Tier”. Zusätzlich zu den 521 Klassen kommen fünf Data Properties mit zusätzlich jeweils einer Definition einer Domain und einer Definition einer Range, sowie 520 Subklassen-Deklarationen und 36 disjunkte Klassendefinitionen. Insgesamt werden damit 1092 Axiome in der Ontologie definiert. Die Oberklasse ”Tier” unterteilt sich in zwei Unterklassen ”Wirbeltiere” und ”Gliederfüßer” wobei lediglich die Klasse der Wirbeltiere weiter ausdifferenziert wird. Von dort aus erfolgt eine sukzessive Unterteilung bis hin zu spezifischen Tiergruppen. Insgesamt werden in dieser Ontologie damit eine große Reihe von Tieren abgedeckt, die von Fischen über Reptilien, Vögeln, Amphibien und Säugetieren reichen.

Abbildung 7.1: Ausschnitt Tierklassifikation: taxonomische Struktur des Menschen



So lässt sich beispielsweise die Einordnung des Menschen als eine Art der Menschenaffen in der Testontologie über 14 Ebenen hinweg darstellen, wie in der oberen Abbildung zu erkennen. Zusätzlich zu der Klassenstruktur beinhaltet die Ontologie auch die Data Properties geschlecht, lebt_an_Ort, lebt_in_Kontinent, lebt_in_Land und name, welche alle als Strings definiert sind und Individuen der Klasse Tier zugewiesen werden können. Eine ausführlichere Abbildung der Ontologie, die einen größeren Bereich der modellierten Klassen abbildet, findet sich im Anhang in Abbildung A.1.

Trotz der Betitelung als einfachste der drei Testontologien beinhaltet diese strukturelle Eigenschaften, die sie zu einem ersten Prüfstein für die Experimentreihe machen.

- Durch die tiefe Hierarchie der Unterklassen-Strukturen über teils zweistellig viele Ebenen hinweg, wird die Fähigkeit des Sprachmodells geprüft, die generierten Aussagen in dieser hierarchischen Struktur korrekt einzuordnen
- Die Ontologie enthält disjunkte Klassen zur Sicherung der logischen Konsistenz, die es für das Sprachmodell zu interpretieren gilt.
- Durch einfache Data Properties mit Einschränkungen auf Domain und Range lassen sich erste Überprüfungen der syntaktischen Validität von generierten Properties vornehmen.

Neben diesen Eigenschaften der Ontologie, gibt es noch weitere Aspekte, die durch das Experiment geprüft werden müssen. Die Subklassen der Klasse Tier in der Ontologie werden aus Gründen der semantischen Konsistenz immer nach dem deutschen Namen der Familie oder Ordnung des Tiers benannt und im Plural formuliert. Es ist zu prüfen, ob das LLM in der Lage ist, diese Konsistenz beizubehalten. Zudem kommt auch noch ein zweiter Aspekt hinzu. Elemente in OWL-Dokumenten werden über einen ”Internationalized Resource Identifier” (IRI) gekennzeichnet. In Protégé wird für eine Ontologie ein bestimmter Grundstamm für diesen definiert, dem jedes Element der Ontologie folgt. Der IRI-Grundstamm für diese Ontologie ist folgender.

http://www.semanticweb.org/Thesis/Ontologien/Tierklassifikation

So hat jedes definierte Element dieser Ontologie einen IRI, der sich aus diesem Grundstamm zusammensetzt. Die Klasse der Raubtiere hat basierend auf diesem Grundstamm so beispielsweise die folgende erweiterte Version eines Identifiers.

http://www.semanticweb.org/Thesis/Ontologien/Tierklassifikation/Raubtiere

Das LLM muss in der Lage sein, diese korrekt zu interpretieren und bei der Referenzierung von bestehenden Elementen der Ontologie in den generierten Antworten diese korrekt

zu verwenden. Zudem muss es bei der Erstellung neuer Elemente diese auch anhand der Vorlage korrekt erzeugen, um zu gewährleisten, dass Protégé auch erkennt, dass gleiche Elemente auch als solche erkannt werden und es nicht zu Verwechslungen durch das Programm kommt. Diese Fähigkeit gilt es allerdings nicht nur für diese Ontologie zu überprüfen, sondern für alle drei Testontologien. Damit erfüllt die erste Ontologie mehrere Funktionen im Rahmen der experimentellen Evaluation.

- Die Ontologie dient als ein Basistest für syntaktische Fähigkeiten des Sprachmodells, vor allem zum Testen der Fähigkeiten für Few-Shot und Zero-Shot Prompts
- Durch die tiefgehende hierarchische Struktur der vielen Subklassen lässt sich das Strukturverständnis des Modells unter Verwendung der verschiedenen Prompt Engineering Strategien überprüfen.
- Die große Masse an Klassen und Subklassenbeziehungen bietet die Grundlage für semantische Fehleinschätzungen bei der Zuordnung von Fakten zu korrekten Klassen durch das Sprachmodell.

Somit bietet diese Ontologie eine solide Grundlage zur Validierung elementarer Fähigkeiten in der Generierung von OWL-Dokumenten und dient als Ausgangspunkt für den Vergleich mit komplexeren Ontologietypen. Als wird die zweite Ontologie modelliert. Dabei ist der nächste Schritt eine Familienontologie zu erstellen, die komplexere Relationen zwischen Klassen und weitere Einschränkungen abbildet als die Tierklassifikation. Somit bringt die Habsburger-Ontologie eine weitere Ebene in die Evaluation der Fähigkeiten des Sprachmodells in der Generierung von OWL-Fakten.

7.1.2 Familienontologie der Habsburger

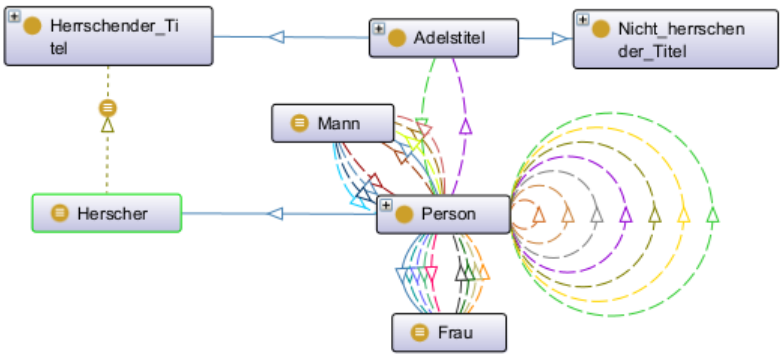
Die zweite Ontologie ist eine Familienontologie, genauer gesagt eine Ontologie der Habsburger-Dynastie. Ziel dieser Ontologie ist die modellhafte Abbildung der Strukturen der Habsburger. Die Habsburger zählen zu den bedeutendsten und verbreitetsten Adelsgeschlechtern der europäischen Geschichte, die über mehrere Jahrhunderte hinweg ihren politischen und kulturellen Einfluss ausgeweitet und über viele Länder verbreitet haben. Damit eignet sie sich für die Entwicklung einer Familienontologie, die nicht nur Verwandtschaftsbeziehungen, sondern vielfältige und komplexe Aspekte abdeckt. Die Ontologie integriert dabei neben Verwandtschaftsbeziehungen auch Adelstitel, Regierungszeiten, historische Epochen mitsamt Ereignissen, Orten und Herrschaftsverhältnissen sowie Todesursachen der Familienmitglieder. Diese vielfältige semantische Struktur erzeugt eine komplexe Testumgebung für das Verständnis des Sprachmodells von relationalen und logischen Zusammenhängen in Ontologien.

Die Grundlage für die benötigten strukturellen Elemente ist das Informationsportal ”Die Welt der Habsburger”. Dabei handelt es sich um ein Projekt der Schloss Schönbrunn Kultur- und Betriebsgesellschaft, die vom österreichischen Staat verwaltet wird und verschiedene Kulturstätten in Österreich zum Zwecke der Bildung und des Kulturerhalts betreibt [28]. Die Gesellschaft stellt über diese Plattform strukturierte Informationen zu Mitgliedern der Dynastie im historischen Kontext bereit und kann im Anschluss an die Ontologie-Erstellung auch als Datengrundlage für die natürlichsprachlichen Fakten verwendet werden.

Insgesamt umfasst die Ontologie 311 Axiome. Damit liegt die Anzahl der Elemente unter der Elementanzahl der Tierklassifikation, allerdings verteilen sich diese 311 Axiome auf lediglich 27 Klassen, wodurch die Ontologie eine wesentlich höhere strukturelle Komplexität aufweist. Von diesen 311 Elementen sind 85 Deklarationen, darunter 27 Klassen, 47 Object Properties und 11 Data Properties. 226 der Elemente bestehen aus logischen Axiomen. Hierzu gehört die Definition von Subklassen, äquivalenten Klassen, Subproperties, inversen Properties, logischen Einschränkungen wie funktionalen, asymmetrischen oder irreflexiven Properties sowie Kardinalitäts- und Quantorrestriktionen, als auch disjunkten Klassen und Restriktionen von Domain und Range für Properties.

So bildet die Klasse ”Person” die zentralste Klasse der Ontologie. Sie steht mit den meisten anderen Klassen in Beziehung und ist Domain oder Range für einen großen Teil der Object Properties. Zur Veranschaulichung und zum Verständnis der Struktur dieser Ontologie werden Teilmengen davon für die verschiedenen Aspekte der Ontologie nacheinander grafisch abgebildet. Eine vollständige Abbildung ist im Anhang unter Abbildung A.2 hinterlegt.

Abbildung 7.2: Ausschnitt Habsburger-Ontologie: Fokus auf familiäre Beziehungen und Titel

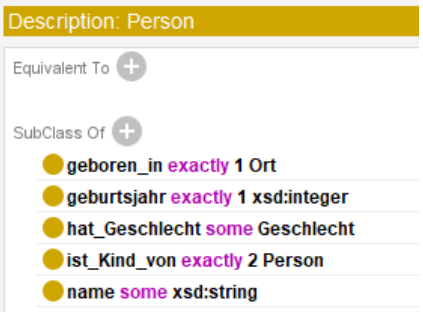


In der oberen Darstellung sind die Klassen abgebildet, die die familiären Beziehungen und die zugehörigen Adelstitel der Habsburger zeigen. Wie auch in der Tierklassifikation zeigen die Pfeile mit durchgezogenen Linien die Subklassen-Beziehungen und die gestrichelten Pfeile zeigen Object Properties, sowie Restriktionen und äquivalente Klassen. Eine Person wird in drei Subklassen unterteilt. Diese Subklassen sind Mann, Frau und Herrscher. Die Klassen Mann, Frau und Person sind über verschiedene Object Properties miteinander verbunden. Durch Object Properties, wie ”ist_Elternteil _von”, ”hat_Geschwister” und ”hat_Ehepartner” werden grundlegende familiäre Beziehungen modelliert. Die Beziehung für Ehepartner und Geschwister ist dabei symmetrisch modelliert und die Elternteilbeziehung hat eine inverse Property, die eine Kindbeziehung modelliert. Inverse Property bedeutet, dass es für eine Object Property zwischen Klasse A und Klasse B eine andere Property gibt, die eine gleichbedeutende Beziehung zwischen B und A darstellt. Für jede Beziehung in dieser Ontologie ist eine inverse Property definiert. Sie bilden also Gegenbeziehungen ab. Jede der Beziehungen verfügt zudem über Subproperties, die die geschlechtsspezifische Version, wie Schwester, Bruder, Mutter oder Ehefrau, beschreiben. Personen können einen Adelstitel haben, der sich in herrschende Titel und nicht herrschende Titel unterteilt. Diese Subklassen werden wiederum in Subklassen un-

terteilt. Zu den herrschenden Titeln zählen Adelstitel, wie König, Kaiser, Graf oder Kurfürst, die eine regierende Position in der Gesellschaft widerspiegeln. Nicht herrschende Titel repräsentieren keine direkte politische Macht, wie beispielsweise der Titel eines Prinzen. Durch diese Unterteilung entsteht auch die Beziehung zwischen der Klasse Herrscher und Adelstitel, da ein Herrscher in dieser Ontologie auch als eine Person definiert ist, die einen herrschenden Adelstitel besitzt. Diese Verknüpfungen sorgen für eine höhere Komplexität als die Tierklassifikation und bieten so eine weitere Perspektive in der Experimentreihe.

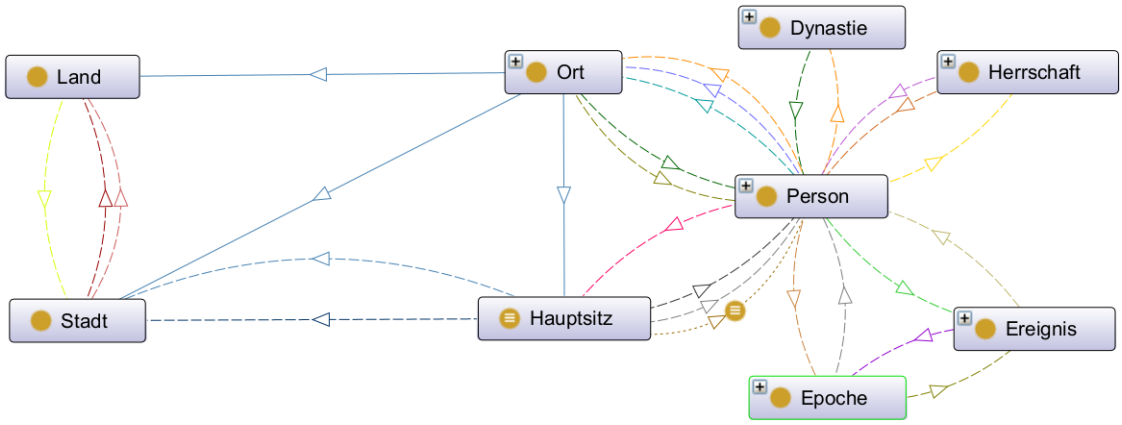
Neben den abgebildeten Object Properties sind auch Data Properties und Restriktionen, also Eigenschaften von Individuen einer Klasse, die mit einem Datenwert anstatt eines anderen Individuals versehen sind, und zusätzliche Einschränkungen, wie verpflichtende Properties, in der Familienontologie hinterlegt.

Abbildung 7.3: Einschränkungen der Klasse Person in der Familienontologie



So erhält beispielsweise die Klasse Person zusätzliche Einschränkungen innerhalb der Ontologie, auf deren Verständnis das Sprachmodell ebenfalls getestet werden kann. Eine Person muss demnach an genau einem Ort in einem spezifischen Jahr geboren sein, muss ein Geschlecht sowie einen Namen haben und muss ein Kind von 2 Personen sein. Auch sind in der Ontologie weitere Data Properties modelliert, die aber teils nicht verpflichtend sind, wie der Spitzname einer Person oder ein Sterbedatum. Solche Einschränkungen werden in der Ontologie auch auf andere Klassen angewandt. Die Klasse Epoche hat demnach verpflichtend ein Anfangsdatum und ein Enddatum und die Klasse Ereignis hat ebenfalls verpflichtende Data Properties mit einer Ereignisbeschreibung und einem Ereignisdatum. Die Klasse Herrschaft hat ebenfalls ein Beginndatum und ein Enddatum, sowie eine über die Object Property ”ausgeübt_durch” verbundene Person, die diese Herrschaft ausübt.

Abbildung 7.4: Ausschnitt Habsburger-Ontologie: Fokus auf anderweitige Beziehungen



Neben den zuvor gezeigten Klassen zur Modellierung von familiären Beziehungen und Adelstiteln sind auch weitere Beziehungsaspekte in der Ontologie abgedeckt, die für die Abbildung der Mitglieder einer Adelsfamilie von Relevanz sind. Personen haben, wie zuvor bereits erwähnt, einen Geburtsort und gegebenenfalls (im Falle der Habsburger in jedem Fall) einen Sterbeort. Diese sind über Object Properties zu der Klasse ”Ort” definiert. Die Klasse Ort teilt sich in die Subklassen ”Land”, ”Stadt” und ”Hauptsitz”. Die Subklassen sind jeweils noch über Object Properties miteinander verbunden. Ein Hauptsitz repräsentiert das Anwesen des jeweiligen Familienmitglieds, was auch durch zwei inverse Object Properties zwischen einer Person und einem Hauptsitz modelliert ist. Der Hauptsitz steht in exakt einer Stadt und die Stadt wiederum liegt in exakt einem Land. Damit sind ortsbezogene Elemente für die Ontologie modelliert.

Zudem werden auch weitere Aspekte in die Ontologie integriert, wie die Mitgliedschaft der Familienmitglieder zu einer bestimmten Dynastie, da sich die Habsburger über mehrere Epochen und verschiedene Länder Europas erstreckten. Des Weiteren kann einer Person eine Herrschaft zugeschrieben werden, die über einen gewissen Zeitraum hinweg ausgeübt worden ist. Auch erfolgt in der Ontologie eine Einordnung der Personen in den historischen Kontext. So leben Personen beispielsweise in einer Epoche. Innerhalb dieser Epochen können Ereignisse auftreten und an diesen Ereignissen sind wiederum gewisse Personen beteiligt. Eine vollständige graphische Abbildung der Ontologie ist im Anhang unter A.2 zu finden. Ein Eintrag für die Erzherzogin Maria Theresia von Österreich könnte in der Ontologie dann beispielsweise wie folgt aussehen.

```
:Maria_Theresia rdf:type :Frau ;
                :name ''Maria Theresia'' ;
                :geburtsjahr 1717 ;
                :ist_Kind_von :Karl_VI , :Elisabeth_Christine ;
                :hat_Ehepartner :Franz_I ;
                :gehoert_zu_Dynastie :Habsburg ;
                :hat_Titel :Erzherzogin_von_Oestereich , :
                   Koenigin_von_Boehmen ;
                :lebt_in_Epoche :Barock .
```

Listing 7.1: OWL-Eintrag: Maria Theresia

Damit enthält die Familienontologie eine Vielzahl an strukturellen Merkmalen, die für eine höhere Komplexität gegenüber der Tierklassifikation sorgen und die Überprüfung anderer Fähigkeiten des Sprachmodells im Rahmen des Experiments ermöglichen.

- Durch die vielen abgedeckten Aspekte finden sich multimodale Beziehungen in der Familienontologie. Personen können zur gleichen Zeit Ehepartner, Elternteile, Kinder, Herrscher, Titelinhaber, Dynastiemitglieder, Epochenbewohner und beteiligte an Ereignissen sein.
- Die komplexen logischen Verknüpfungen schaffen den Raum für logische Inferenzen innerhalb der Ontologie.
- Dennoch ist in einigen Klassen wie auch in der Tierklassifikation eine zumindest leichte hierarchische Klassenstruktur durch Subklassen gegeben.

Dadurch eignet sich die Ontologie für die Bewertung verschiedenster Aspekte im Zusammenspiel mit dem LLM. Zum einen erfordert die Erzeugung von OWL-Fakten aus natürlichsprachlichen historischen Einträgen die Generierung verschiedener Properties unter zugleich logischer Konsistenz. Durch die temporalen Zusammenhänge und die Namensüberschneidungen von Mitgliedern der Adelsfamilien in der Ontologie erlaubt dies zudem das Testen der semantischen Interpretation des Sprachmodells aus natürlichsprachlichen Texten heraus in Bezug auf Ereignisse im zeitlichen Kontext und auf Elemente mit ähnlicher Bezeichnung. Die größere Vielfalt an OWL-Elementen testet zudem die Fähigkeit des Sprachmodells OWL-Dokumente mit komplexerer Syntax zu erstellen. Diese Aspekte machen die Familienontologie zu einer geeigneten zweiten Umgebung für die Evaluation, um zu überprüfen, ob das Sprachmodell komplexere Zusammenhänge verstehen und daraus korrekt abgeleitete OWL-Aussagen erzeugen kann.

Während die Familienontologie bereits ein höheres Maß an Komplexität aufweist, basiert sie fachlich auf dem Kontext einer Familienstruktur, und damit auf einfachen sozialen und familiären Beziehungen. Um die methodische Struktur der Evaluation weiter zu differenzieren, indem die Fähigkeiten des LLMs im Umgang mit spezifischen Fachdomänen und der Kombination von relationaler und taxonomischer Komplexität getestet werden, folgt im nächsten Abschnitt die dritte und letzte Ontologie. Diese konzentriert sich auf die Repräsentation von Krankheitsbildern, Symptomen sowie darauf basierenden Therapieempfehlungen und Diagnosen.

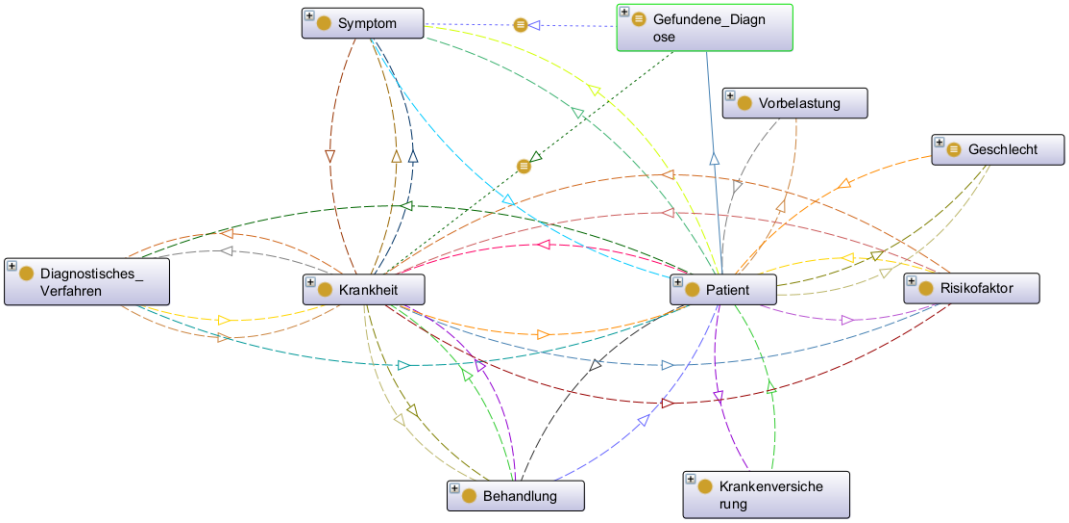
7.1.3 Medizinische Ontologie

Die dritte und letzte der entwickelten Testontologien befasst sich mit medizinischem Wissen und insbesondere der Darstellung von Krankheitsbildern mitsamt potenzieller Symptome, zugehöriger Diagnoseverfahren und Behandlungsmöglichkeiten sowie deren Beziehungen zu Patienten und Patientinnen. Ziel dieser Ontologie ist es, durch die Modellierung einer domänenspezifischen medizinischen Ontologie die Komplexitätsmerkmale der vorherigen beiden Ontologien zu vereinen und somit taxonomische Informationen und auch komplexe logische Zusammenhänge zwischen Klassen zu modellieren. Zudem ist die medizinische Ontologie noch um zusätzliche Komplexitätsmerkmale, wie Property Chains, erweitert.

Fachlich ist die Ontologie so aufgebaut, dass die Klassen ”Krankheit” und ”Patient” an zentraler Stelle sind. Krankheiten sind taxonomisch in Subklassen unterteilt und stehen in Verbindung mit zugehörigen Symptomen, Risikofaktoren, die sich negativ auf den Krankheitsverlauf auswirken, sowie Behandlungsmöglichkeiten und zugehörigen Diagnoseverfahren. Patienten werden Vorbelastungen und ebenfalls Symptome zugeordnet, wodurch in der Ontologie Diagnosen und Empfehlungen für Behandlungsmöglichkeiten sowie zu vermeidende Risikofaktoren ermittelt werden können. Die taxonomische Struktur von Behandlungen, Krankheiten und Symptomen basiert dabei auf der internationalen statistischen Klassifikation der Krankheiten und verwandter Gesundheitsprobleme, der ICD-10. Dabei handelt es sich um die offizielle Klassifikation von Krankheiten nach der Weltgesundheitsorganisation. Genauer gesagt basiert die Struktur auf der durch das Bundesinstitut für Arzneimittel und Medizinprodukte veröffentlichten deutschen Modifikation, der ICD-10-GM [4].

Insgesamt besteht die Ontologie aus 1073 Elementen, die sich aufteilen in 292 Deklarationen von 179 Klassen, 60 Object Properties und 35 Data Properties, sowie 635 logischen Axiomen. Hierzu gehören neben Subklassen, äquivalenten Klassen und disjunkten Klassen auch Einschränkungen auf Object Properties, Kardinalitäts- und Quantorrestriktionen sowie Property Chains (also die transitive Zusammensetzung mehrerer Object Properties zur Zusammensetzung einer übergeordneten Object Property). Zudem sind in dieser Ontologie erstmals auch Annotation Properties zur Definition von beschreibenden Metadaten verwendet worden. Die Ontologie bietet also eine taxonomische Subklassenstruktur, ähnlich der Tierklassifikation und komplexe Relationen zwischen den Klassen, ähnlich der Familienontologie. Des Weiteren bietet sie eine zusätzliche fachliche Domäne und zusätzliche OWL-Elemente, wie Property Chains, Annotation Properties und auch beispielhafte Individuen für eine Krankheit und einen Patienten.

Abbildung 7.5: Ausschnitt der Medizinontologie mit zentralen Klassen und Object Properties



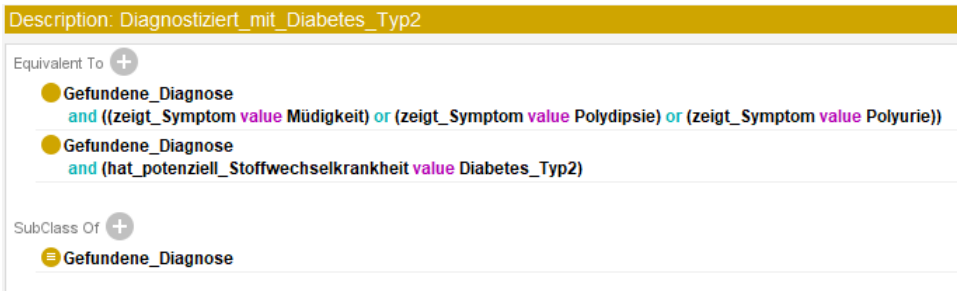
In der oben abgebildeten grafischen Darstellung sind die obersten Klassen mitsamt ihrer Relationen zu anderen Klassen dargestellt. Anhand dieser Struktur lassen sich die Komplexitätsmerkmale der Ontologie veranschaulichen. Die Subklassenbeziehungen würden die grafische Übersichtlichkeit der Ontologie weiter einschränken und sind im Anhang der Arbeit unter Abbildung A.3 abgebildet. Eine Krankheit ist in der Ontologie noch zusätzlich unterteilt in ihre nach ICD-10 eingeordneten Subklassen. Hierzu gehören insgesamt 16 direkte Subklassen, darunter Hautkrankheiten, Infektionskrankheiten wie bakterielle, virenbezogene oder parasitäre Infektionen, Neubildungen wie Tumore, Krankheiten des Verdauungs-, Atmungs- oder Nervensystems und noch weitere. In der Evaluation werden als natürlichsprachliche Dokumente Einträge zu Krankheitsbildern mitsamt Symptomen, Diagnoseverfahren und Behandlungsmöglichkeiten verwendet. Wie bereits in der Tierklassifikation stellt die korrekte Interpretation der taxonomischen Einordnung von Individuen eine zu überprüfende Herausforderung für das Sprachmodell dar. Die Klasse Krankheit ist mit verschiedenen Data Properties versehen. So haben Krankheiten einen String, der ihre ICD-10 Identifikationsnummer repräsentiert, eine Beschreibung, sowie eine Prävalenz und die Mortalitätsrate der Krankheit, als auch einen Booleanwert, mit dem angegeben werden kann, ob eine Krankheit vererblich ist.

Neben Data Properties ist die Klasse Krankheit noch mit weiteren Klassen über Object Properties verbunden, wodurch auch das erste zentrale Komplexitätsmerkmal der vielschichtigen Beziehungen ausgeprägt wird. Eine Krankheit wird über n-viele Symptome ausgeprägt. Diese Symptome sind wiederum in 11 Subklassen unterteilt, die sich durch die zugehörigen Körperelemente klassifizieren, wie psychologische, dermatologische oder neurologische Symptome. Symptome sind zudem mit Data Properties für eine Beschreibung des Symptoms und einer Auftrittshäufigkeit bei der verbundenen Krankheit versehen. Neben Symptomen werden Krankheiten auch der Klasse diagnostisches Verfahren über Object Properties zugeordnet. Ein diagnostisches Verfahren ist ein Verfahren zur fortführenden Diagnose einer Krankheit, das sich unterteilt in die Subklassen Anamnese, bildgebende Verfahren, körperliche Untersuchungen und Labordiagnosen. Des Weiteren sind Krankheiten verknüpft mit einer zugehörigen Behandlung, die über Subklassen medikamentös, medikamentlos, operativ oder psychotherapeutisch ausgeprägt ist und eine Beschreibung, eine Behandlungsdauer, ein Behandlungsziel sowie Behandlungskosten als Data Properties besitzt. Weiter haben Krankheiten auch Relationen zu der Klasse der Risikofaktoren, was Faktoren sind, die eine negative Auswirkung auf den Krankheitsverlauf haben oder bei vermuteter Krankheit vermieden werden sollten, da sie die Entstehung der Krankheit wahrscheinlicher machen. Rauchen wäre beispielsweise ein Risikofaktor für Lungenkrebs. Für die Experimentreihe gilt es an dieser Stelle, dass bei der Angabe von natürlichsprachlichen Fakten für eine Krankheit das Sprachmodell auch die notwendigen Individuen in den verbundenen Klassen erstellen und diese auch taxonomisch den korrekten Unterklassen zuordnen muss.

Die Klasse ”Patient” bildet die zweite zentrale Oberklasse. Diese ist unterteilt in Mann, Frau und die Subklasse Gefundene_Diagnose. Zudem hat die Klasse Data Properties, wie ein Alter, eine Anschrift, ein Gewicht, Körpergröße und einen Namen. Ein Patient ist zudem direkt über Object Properties mit einer Krankenversicherung verbunden, er zeigt n-viele Symptome und er hat ggfs. eine Vorbelastung, die wiederum über Object Properties mit einer Behandlung verbunden ist. Dies ist modelliert, um anzugeben, dass

bestimmte Behandlungen bei bestimmten Vorbelastungen von Patienten oder Patientinnen nicht durchgeführt werden sollen. Es wird dem Individuum eines Patienten zudem die ObjectProperty ”hat_Potenziell” zugeordnet, welche den Patienten oder die Patientin schließlich mit einer Krankheit verbindet. Zuletzt wird die Klasse ”Patient” über die Property Chains auch mit den Klassen, Diagnostisches Verfahren, Behandlung und Risikofaktor verbunden. Durch dieses Komplexitätsmerkmal hebt sich die medizinische Ontologie weiter von den anderen beiden ab. Zur Verständlichkeit wird an dieser Stelle eine Property Chain beispielhaft erläutert.

Abbildung 7.6: Ableitung der Object Property ”hat_Potenziell” über Subklasse von Patient



Die Subklasse ”Gefundene_Diagnose” von der Klasse ”Patient” wird für jede Krankheit um eine eigene Subklasse erweitert. Diese Subklassen ermöglichen eine Vorhersage für eine Krankheit in der Ontologie. In der oberen Abbildung ist eine Klassenbeschreibung für eine Subklasse von ”Gefundene_Diagnose”, die sich auf Diabetes Typ 2 bezieht. Hier wird für die Krankheit eine Äquivalenzbeziehung durch Properties eines potenziellen Patienten angegeben. In diesem Fall wird ein Patient oder eine Patientin dieser Subklasse zugewiesen, wenn er oder sie entweder das Symptom Müdigkeit, Polydipsie oder Polyurie aufweist. Auch können andere Properties für diese Äquivalenzbeziehung verwendet werden, wie das Geschlecht, um geschlechtsspezifische Diagnosen, wie eine Schwangerschaft, abzuleiten. Diese Zuordnung geschieht, sobald ein Reasoner angewandt ist. Zudem wird über eine zweite Äquivalenzbeziehung dem oder der Patientin die Object Property ”hat_Potenziell” zugewiesen. Sie ist zudem die Grundlage für die Property Chains und verbindet damit die Klasse ”Patient” zeitgleich mit den Klassen ”Behandlung”, ”Risikofaktor” und ”Diagnostisches Verfahren”. Diese Ableitung funktioniert, da die Klasse ”Krankheit” ja bereits mit diesen Klassen über Object Properties verbunden ist, und nun ”Patient” mit einer Krankheit.

```
:kann_fortfuehrend_untersucht_werden_durch owl:propertyChainAxiom (
    :hat_potenziell
    :diagnostiziert_durch
) .
```

Listing 7.2: OWL-Eintrag: Property Chain

Hierfür wird eine neue Object Property definiert, die als Domain die Klasse ”Patient” annimmt und als Range die gewünschte Klasse. Im Falle der oberen Abbildung also ”Diagnostisches_Verfahren”. Dann kann über die Property Chain eine transitive Relation zwischen Domain und Range aufgebaut werden, indem eine Kette von Object Properties angegeben wird, bei der die Range der vorherigen Property immer mit der Domain der

folgenden Property übereinstimmt. Für die Verbindung der Klasse ”Patient” mit ”Diagnostisches_Verfahren” kann diese Verbindung über die Properties ”hat_Potenziell”, welche den Patienten mit der Krankheit verknüpft, und ”diagnostiziert durch”, welche die Krankheit mit dem diagnostischen Verfahren verknüpft, geschaffen werden. Diese Ketten sind auch für die Klassen Behandlung und Risikofaktor definiert und jede dieser Property Chains hat auch eine zugehörige inverse Property.

Zusätzlich sind über Annotation Properties allen Klassen und Object Properties beschreibende Metadaten hinzugefügt worden, unter der Annahme, dass diese möglicherweise die Ergebnisse des LLMs unterstützen, da sie im Verständnis der Klassen und durch die natürlichsprachige Beschreibung unterstützen können. Vor allem in der Anwendung von Zero-Shot Prompting, bei dem das Sprachmodell keine zusätzlichen Beispiele erhält, ist die Annahme, dass zusätzliche beschreibende Angaben innerhalb der Ontologie selbst unterstützen können. Dieselbe Annahme gilt auch für das Einfügen von einem Individuum der beispielhaften Krankheit ”Diabetes Typ 2” mitsamt Symptomen, Diagnoseverfahren und Risikofaktoren sowie eines Individuums für einen männlichen Patienten mit Symptomen, die mit dieser Krankheit übereinstimmen. Diese Annotation Properties und Beispielindividuen bilden ein weiteres Komplexitätsmerkmal, das nicht durch die anderen Ontologien abgedeckt wird. Eine ausführliche Abbildung der medizinischen Ontologie, die aus Gründen der Übersichtlichkeit nicht alle Subklassen der Krankheiten anzeigt, ist im Anhang unter Abbildung A.3 zu finden. Die vollständige Ontologie ist im elektronischen Anhang zu finden. Die zugrundeliegende, durch ICD-10 vorgegebene Struktur kann auf der Internetseite des Bundesinstituts für Arzneimittel und Medizinprodukte (BfArM) genauer betrachtet werden [4].

Damit sind nun alle drei Ontologien erstellt, die durch verschiedene Komplexitätsmerkmale jeweils eine eigene Relevanz für die Experimentreihe bieten. Durch die Subklassenbeziehungen in verschiedensten Klassen der medizinischen Ontologie ist eine Überprüfung der Fähigkeit des Sprachmodells gegeben, Klassenbezeichnungen korrekt anzugeben, zu unterscheiden und Individuen bei der Generierung von Fakten korrekt einzuordnen. Durch die komplexen und vielen Object- und Data Properties wird das Verständnis getestet, Zusammenhänge zwischen Klassen zu verstehen und bei der Erzeugung von Fakten notwendige Individuen von verbundenen Klassen ebenfalls zu deklarieren, falls diese nicht bereits in der Ontologie vorhanden sind.

Zusätzlich testet die medizinische Ontologie das Verständnis des Sprachmodells von zusätzlichen OWL-Elementen, wie Property Chains und das Verständnis des LLMs von integrierten beschreibenden Elementen, wie Annotation Properties für beschreibende Metadaten der Elemente, als auch beispielhafte Individuen innerhalb der Ontologie. Damit bietet diese Ontologie eine anspruchsvolle Umgebung für die folgende Testreihe. Als Nächstes gilt es, für jede der drei Ontologien zehn natürlichsprachliche Referenzdokumente für die Experimentreihe zu erstellen und diese manuell in OWL zu überführen, um so Vergleichswerte für die durch das LLM generierten Fakten zu erhalten.

7.2 Erstellung von Referenzdokumenten

Ziel dieses Abschnitts ist es, die Erstellung der Referenzdokumente in natürlicher Sprache sowie deren Überführung in OWL-Dateien, welche für die Durchführung der

Evaluation benötigt werden, zu dokumentieren. Diese Referenzdokumente dienen als Abgleich zur Evaluation der durch das LLM erzeugten OWL-Dokumente. Für jede der im vorherigen Abschnitt beschriebenen Ontologien sind zehn natürlichsprachliche Texte erfasst worden. Diese sind so verfasst worden, dass sie unterschiedliche Anwendungsfälle der jeweiligen Ontologie abdecken und unterschiedliche Anforderungen an die OWL-Generierung stellen. Alle Referenzdokumente lassen sich im elektronischen Anhang einsehen.

- Der Fokus der Tierklassifikation liegt auf der korrekten Klassenzuordnung innerhalb der taxonomischen Struktur mit einfachen Data Properties.
- Die Texte der Habsburger-Ontologie testen das Generieren von mehreren, teils ähnlich bezeichneten Individuen und der Verknüpfung dieser über Object Properties.
- Die medizinische Ontologie verknüpft beide vorherigen Testaspekte.

Für jede der insgesamt 30 natürlichsprachlichen Dokumente sind anschließend manuell OWL-Dateien im Turtle-Format erstellt worden. Diese Dateien dienen als Referenz zur Evaluation hinsichtlich der verschiedenen Evaluationsaspekte zur Ermittlung von TP, FP und FN. In den folgenden Abschnitten wird die Erstellung der Referenzdokumente für jede einzelne Ontologie getrennt voneinander beleuchtet.

7.2.1 Referenzdokumente für die Tierklassifikation

Für die Tierklassifikation sind zehn inhaltlich differenzierte Texte über Tierindividuen erstellt worden. Diese umfassen teils berühmte Tiere und teils weniger bekannte Tiere, um zu überprüfen, ob das Sprachmodell ggf. Informationen für die beschriebenen Individuen halluziniert. Die beschriebenen Individuen beinhalten Säugetiere, wie den Bären Poldi, einem europäischen Braunbären, der im Schwarzwälder Bärenpark aufgenommen worden ist, oder die Delfine, die am Set der Serie ”Flipper” den gleichnamigen TV-Delfin gespielt haben. Auch Tiere wie Hachiko der berühmte Straßenhund, für den in Japan eine Gedenkstatue errichtet worden ist, Washoe der Schimpanse, der Teil eines Forschungsprojektes war, in dem Primaten Gebärdensprache erlernt haben, oder Cecil der Löwe sowie Charles Darwin sind textuell abgebildet worden. Auch sind Texte erstellt worden über Nicht-Säugetiere, darunter ein Kakadu, Kermit der Frosch, Lonesome George (die letzte lebende Galapagos-Riesenschildkröte) und auch der älteste Koifisch der Welt.

Die Texte behandeln bewusst nicht nur unterschiedliche Tierarten, sondern unterscheiden sich auch in anderen Aspekten, um zu überprüfen, wie gut das Sprachmodell mit unterschiedlichen Informationen umgehen kann.

- Einige der in den Texten benannten Tierarten existieren bereits als Klasse in der Ontologie und einige müssen durch das LLM selbst erstellt und korrekt deklariert werden.
- Es werden nicht nur einzelne, sondern teils auch mehrere Individuen benannt.
- In den Texten werden vereinzelt auch Informationen angegeben, die nicht in der Ontologie durch Data Properties angegeben, und damit vom Sprachmodell zu ignorieren sind

Die allgemeine Struktur der Texte ist allerdings in allen Fällen gleich aufgesetzt. Der Umfang der Dokumente entspricht etwa einer halben DIN A4-Seite und die Texte gliedern sich in zwei Abschnitte. Im ersten Abschnitt wird die Tierart benannt, sowie die nächste übergeordnete taxonomische Familie oder Ordnung der Tierart. Im zweiten Abschnitt folgen die Informationen über die Individuen, welche immer den Namen und das Geschlecht beinhalten, sowie den Ort, an dem diese leben. Zusätzlich sind noch weitere Informationen gegeben, wie die Geschichte der Tiere, wie die zugehörige Tierart sich ernährt oder Altersinformationen über die Individuen.

Die Grundlage für die Texte bilden öffentlich zugängliche Einträge über die Individuen. Im Fall der bekannten Tiere basieren diese auf gekürzten Abschnitten der jeweiligen Wikipedia-Artikel und im Fall des Textes über den Bären Poldi basiert dieser auf einem Eintrag über den Bären im Portal des Tierparks auf der Seite der deutschen Stiftung für Bären [58]. Das natürlichsprachige Referenzdokument für den Braunbären Poldi sieht beispielsweise wie folgt aus:

> ”Die Braunbären (Ursus arctos) gehören zur Familie der Bären innerhalb der Ordnung der Hundeartigen. Innerhalb dieser Art existieren mehrere Unterarten, darunter der Europäische Braunbär, der Sibirische Braunbär und der Grizzlybär. Braunbären sind Allesfresser und ernähren sich sowohl von Fleisch, etwa kleinen Säugetieren oder Aas, als auch von Fisch sowie von Pflanzen wie Beeren, Wurzeln und Gräsern.
>
> Poldi ist ein männliches Exemplar eines Europäischen Braunbären, der 2010 gerettet worden ist und im Alternativen Bärenpark Schwarzwald in Deutschland untergekommen ist.”

Dieses ist wie folgt manuell in ein OWL-Dokument in Turtle-Format übersetzt worden. Dabei werden alle benannten taxonomischen Klassen in der Tierklassifikation zusätzlich deklariert, die nicht bereits in der Ontologie vorhanden sind.

```
:Europaeische_Braunbaeren rdf:type owl:Class ;
                          rdfs:subClassOf :Braunbaer .

:Grizzlybaeren rdf:type owl:Class ;
               rdfs:subClassOf :Braunbaer .

:Sibirische_Braunbaeren rdf:type owl:Class ;
                        rdfs:subClassOf :Braunbaer .

:Poldi rdf:type :Europaeische_Braunbaeren ;
       :geschlecht ''maennlich'' ;
       :lebt_an_Ort ''Alternativer Baerenpark Schwarzwald'' ;
       :lebt_in_Kontinent ''Europa'' ;
       :lebt_in_Land ''Deutschland'' ;
       :name ''Poldi'' .
```

Listing 7.3: OWL-Eintrag: Poldi

Die Referenzdokumente bilden eine Grundlage für die erste strukturelle Bewertung der Fähigkeiten des LLMs in der Generierung von OWL-Dokumenten.

7.2.2 Referenzdokumente für die Habsburger Familienontologie

Die zehn Referenzdokumente für die Habsburger Familienontologie basieren auf den biografischen Einträgen der Familienmitglieder der Habsburger Dynastie, sowie Einträgen über Epochen und Ereignisse aus dem Habsburger Portal der österreichischen Schönbrunn Gruppe, das bereits für die Struktur der Ontologie verwendet worden ist [28]. Die Inhalte der Texte basieren hierbei jeweils auf einem biografischen oder epochenbezogenen Eintrag, der gekürzt worden ist, um eine Übersichtlichkeit zu behalten. Zudem sind drei Aspekte der Texte angepasst worden. Wenn der Name einer Person genannt worden ist, zu dem eigentlich noch ein Zusatz gehört, wie Karl IV., ist dieser Zusatz hinzugefügt worden. Wenn eine Person im Text einen Titel besitzt und die genaue Bezeichnung des Titels nicht gegeben ist, ist diese ebenfalls hinzugefügt worden. Ist ein Ort im Text benannt mit einer Stadt, ist zusätzlich das Land, in dem der Ort sich befindet, ergänzt worden.

Insgesamt ist die Struktur der Texte damit möglichst ähnlich gehalten. Die Texte enthalten Informationen über Geburten und Todeszeitpunkte, Informationen über Personen sowie familiäre, Heirats- und politische Beziehungen zwischen diesen, als auch Ereignisse mit Zugehörigkeit zu den jeweiligen Epochen. Die Texte sind mit durchschnittlich 1,5 DIN-A4 Seiten an Länge umfangreicher als die Tierklassifikation und enthalten komplexere Informationen. In den insgesamt zehn Dokumenten werden sechs verschiedene Themen behandelt.

- Rudolf I., mit dem die herrschaftliche Dynastie der Habsburger begonnen hat, ist Inhalt von insgesamt drei Referenzdokumenten, die allesamt Ausschnitte seines biografischen Eintrags auf der Seite sind [67] und sich jeweils gesondert auf seine Existenz, die durch ihn beeinflussten Ereignisse und seine familiären Beziehungen fokussieren.
- Die Dokumente vier bis sechs behandeln ebenfalls diese drei Aspekte, aber für eine andere Person aus der Dynastie, Maria Theresia und die Geschehnisse im Habsburger Erbfolgekrieg [41].
- Drei der Dokumente behandeln Herrscher der Dynastie aus verschiedenen Epochen, und vereinen Informationen über Ereignisse, Familie und die Persönlichkeiten selbst in einzelnen Dokumenten. Diese thematisieren Franz Joseph I. vor allem bekannt als Vorlage für die Geschichten über ihn und seine Ehefrau Elisabeth auch bekannt als ”Sissi” [21], Kaiser Leopold I. [34] und den letzten Vertreter der Dynastie vor ihrer Machtabgabe Otto Habsburg Lothringen [54].
- Eins der Dokumente beschäftigt sich mit der Epoche der französischen Revolution und den dort gehäuften Ereignissen mit Bezug zu Maria Antonia, auch bekannt unter ihrem französischen Beinamen ”Marie Antoinette” [42].

Die Texte beinhalten damit einfache, als auch komplexe Beziehungen über Familienbeziehungen, Thronfolgen, Datumsangaben und Ereignisse, sowie die durch diese Personen geprägten Epochen und historische Orte. Damit bieten die Habsburger Referenzdokumente die Grundlage für eine Evaluation von relationalen Modellierungsfähigkeiten durch das Sprachmodell mithilfe von komplexen Object Properties, sowie die korrekte Zuordnung von Data Properties zu menschlichen Individuen, die im Rahmen der Habsburger Dynastie des Öfteren ähnliche Namen besitzen.

7.2.3 Referenzdokumente für die medizinische Ontologie

Für die dritte Ontologie sind zehn Referenzdokumente erstellt worden, die sich jeweils mit einer Krankheit befassen. Die Texte für diese Ontologie kombinieren die relationalen Informationen durch medizinische Fachinformationen über Beschreibungen, Prävalenz, Symptome, Diagnoseverfahren, Risikofaktoren und Behandlungsmöglichkeiten einer Krankheit mit den taxonomischen Einstufungen der Krankheiten in Subklassen. Die Texte sind übernommen aus dem Gesundheitsportal des Bundesministeriums für Gesundheit "gesund.bund" [75]. Die Einträge in diesem Portal enthalten Informationen über Krankheiten, die auch in der medizinischen Ontologie enthalten sind und strukturieren diese auch in verschiedene Abschnitte je nach Fachinformation. Da das Portal vom Bundesministerium für Gesundheit mit dem Ziel entwickelt worden ist, Bürgern eine erste Anlaufstelle für die Information über Krankheiten und gesundheitsbezogene Themen zu bieten, sind die Einträge zugleich verständlich genug geschrieben, um auch von Menschen ohne medizinische Fachausbildung verstanden zu werden oder in diesem Fall ohne, dass ein LLM zuvor zusätzliche fachliche Informationen erhalten müsste.

Die Texte sind direkt aus dem Portal übernommen und daraufhin in zwei Aspekten angepasst worden. Zu Beginn jedes Textes wird die Krankheit in die nach ICD-10 klassifizierte zugehörige Klasse eingeordnet durch eine einführende Beschreibung. Des Weiteren sind zwei Abschnitte, die im Gesundheitsportal vorhanden sind und sich mit dem Krankheitsverlauf und dem Leben mit dieser Krankheit beschäftigen, aus den Texten entfernt worden, da diese Aspekte nicht Teil der Ontologie sind. Wie auch in der Tierklassifikation gehören einige der Krankheiten zu einer Subklasse, die bereits in der Ontologie vorhanden ist und manche Subklassen müssen zuvor von dem LLM selbst eingefügt werden. Die Texte decken folgende Krankheiten ab: Asthma, Migräne, Magen-Darm-Infektion durch Noroviren, Depressionen, chronisch obstruktive Lungenerkrankung, Multiple Sklerose, Schlaganfall, Alzheimer-Demenz, Blasenentzündung und zuletzt Schuppenflechte.

Inhaltlich enthalten die Texte eigene Abschnitte für die Themen Klassifikation, typische Symptome mit Angaben wie Häufigkeit oder Intensität, mögliche Risikofaktoren, geeignete diagnostische Verfahren und Behandlungsmöglichkeiten mit gegebenenfalls angegebener Medikation und Therapiezielen. Des Weiteren thematisieren sie auch statistische Angaben zur Häufigkeit der Krankheit unter der Bevölkerung. Dieser Umfang in Kombination mit der taxonomischen Struktur der Krankheiten sorgt für mehrere Herausforderungen und Prüfaspekte für das Sprachmodell.

- Das korrekte Einordnen der Krankheit in die taxonomische Struktur der Ontologie.
- Die korrekte Zuordnung von Symptomen, Risikofaktoren, Diagnosen und Behandlungen.
- Die Beachtung von Kardinalitäten und anderen Einschränkungen.
- Die semantische Interpretation von Strukturen in der medizinischen Domäne.

Die Referenzdokumente der medizinischen Ontologie ermöglichen damit eine fundierte Evaluation des LLMs darüber, wie sich strukturierte, mit Object Properties verbundene, fachspezifische Informationen in Kombination mit einer taxonomischen Struktur semantisch erfassen und in OWL-Dokumente überführen lassen.

7.3 Anwendung der Prompt Engineering Strategien

Im vorherigen Abschnitt sind die verwendeten Referenzdokumente vorgestellt worden. Nun erfolgt die praktische Anwendung der verschiedenen Prompt Engineering Strategien, um herauszufinden, inwiefern sich aus den natürlichsprachigen Texten korrekte OWL-Fakten generieren lassen. Die Durchführung ist dabei gezielt gewählt, um die unterschiedlichen Aspekte der Leistungsfähigkeit des Sprachmodells mit steigender Komplexität der Strategien zu evaluieren. Beginnend mit beispiellosen Prompts, dem Zero-Shot Prompting, über die Verwendung von angereicherten Beispielen durch Few-Shot Prompting bis hin zu dokumentengestütztem Prompting mit ausführlichen Anfrage-Antwort-Paaren (RAG), und zuletzt Finetuning.

Für jedes der Dokumente wird die Evaluation sorgfältig dokumentiert und es wird auf einen einheitlichen Experiment-Aufbau zurückgegriffen, damit zwischen den Dokumenten keine Abhängigkeiten in der Generierung der Antworten entstehen und die Vergleichbarkeit bestehen bleibt. Der Aufbau für das Vorgehen gliedert sich dabei unabhängig von der verwendeten Strategie in mehrere übergeordnete Schritte.

Zunächst wäre da die isolierte Projektstrukturierung. Für jede der drei Ontologien wird für jede der verwendeten Strategien ein separates Projekt für das Sprachmodell angelegt. Projekte sind Gruppierungen von Chatanfragen, die es ermöglichen eine Reihe von Dialogen durchzuführen, die unabhängig voneinander sind, aber übergeordnete Systemprompts und Einschränkungen sowie gemeinsam verwendete Dokumente zulassen. Innerhalb jedes Projektes werden die zu der Ontologie gehörenden zehn Referenzdokumente in einer jeweils eigenen Chatinstanz verarbeitet. Diese Trennung gewährleistet, dass keine Abhängigkeiten oder nicht gewollte Lerneffekte beim Sprachmodell zwischen den Fällen entstehen. Somit stellt jede Instanz eine kontrollierte Umgebung für die Evaluation dar.

Zu der isolierten Projektstrukturierung kommt auch eine Konfiguration für Systemprompts hinzu. Neben den Nutzerprompts, welche die direkten Anfragen an das Sprachmodell sind, bieten Systemprompts die Möglichkeit dem LLM übergeordnete Aufgaben zu formulieren, die Formatierung der Antwort einzuschränken und Regeln anzugeben für zu vermeidende Antworten. Diese zugrundeliegenden Systemprompts werden dann bei den Nutzeranfragen berücksichtigt. Im Falle dieser Arbeit orientiert sich der Systemprompt an dem im Grundlagenkapitel beschriebenen idealen Promptaufbau nach Microsoft und erfüllt die folgenden Funktionen.

1. Der thematische und funktionale Kontext wird gegeben.
2. Die Zielsetzung und die Aufgabenstellung wird spezifiziert. Auch das Ausgabeformat wird hier mitgegeben.
3. Tonalität und Formalität der Antworten werden festgelegt.
4. Das Verhalten des Modells wird eingeschränkt und zu vermeidende Antworten werden als solche bezeichnet.

Konkret sieht der Systemprompt für alle Anfragen in diesem Experiment wie folgt aus.

```
###Thematischer Kontext:
Du bist ein hochspezialisiertes KI-System, das fuer die
    strukturierte Umwandlung von natuerlichsprachlichen
    Beschreibungen in eine formale Ontologiesprache (OWL, Turtle-
    Syntax) trainiert wurde. Dein Ziel ist es, aus textuellen
    Beschreibungen fuer gegebene Ontologien korrekte, syntaktisch
    gueltige und semantisch konsistente OWL-Fakten zu erzeugen, die
    mit einer gegebenen Ontologie kompatibel sind.
###Rollenbeschreibung:
Du agierst als Ontologie-Experte mit tiefem Verstaendnis fuer
    semantische Modellierung, logische Inferenzen und die OWL-
    Spezifikation nach dem W3C-Standard. Du analysierst sorgfaeltig
    die gegebene Beschreibung, erkennst relevante Entitaeten und
    Relationen, ergaenzt fehlende Konzepte fuer die Klassenstruktur,
     wenn diese explizit angegeben werden, gegebenenfalls durch
    korrekte Subklassenzuweisung und erzeugst die passenden OWL-
    Ausdruecke.
###Tonalitaet:
Analytisch, praezise, formal und OWL-konform.
###Aufgabe:
Erstelle eine OWL-Ausgabe in Turtle-Syntax, die alle relevanten
    Aussagen zu Individuen und Properties enthaelt, die sich aus dem
     folgenden natuerlichsprachlichen Text ergeben. Nutze lediglich
    Properties und Klassen, die in der gegebenen Ontologie definiert
     sind. Falls eine in der Beschreibung vorkommende und explizit
    genannte Klasse noch nicht in der Ontologie existiert,
    deklariere diese neu und ordne sie korrekt als Subklasse einer
    bestehenden Klasse ein.
###Erwartetes Format:
-Gib nur OWL-Aussagen in Turtle-Syntax zurueck.
-Verwende korrekte Namenskonventionen, orientiert an den
    bestehenden Ontologie-URI-Schemata.
-Verwende rdf:type, rdfs:subClassOf, owl:ObjectProperty, owl:Class,
     owl :NamedIndividual etc. nach OWL 2 Spezifikation.
-Nutze nur Properties, die in der Zielontologie vorkommen.
###Einschraenkungen:
-Verwende nur existierende Properties aus der Zielontologie.
-Falls eine neue Klasse erforderlich ist: deklarieren und korrekt
    einordnen (rdfs:subClassOf).
-Keine Halluzinationen oder Annahmen, die nicht direkt aus dem Text
     hervorgehen.
-Keine weiteren Erklaerungen oder Textausgaben neben OWL.
Verwendete Ontologie:
```

Listing 7.4: Systemprompt

Zusätzlich zu dem Systemprompt wird unabhängig von der verwendeten Strategie in jedem der Projekte auch die jeweilige Zielontologie hinterlegt und dem GPT-Modell als PDF-Datei in Turtle Format zur Verfügung gestellt. Im Systemprompt wird dabei an letzter Stelle der Dokumentenname der hinterlegten Ontologie angegeben. Damit wird sichergestellt, dass sich das Modell an der Struktur der Ontologie orientieren kann. Die Nutzerprompts selbst bestehen lediglich aus dem natürlichsprachigen Referenzdokument

als .txt Datei, sowie den zusätzlichen Prompt-Informationen je nach angewandter Strategie. Die generierten Antworten werden als .ttl Datei exportiert, syntaktisch überprüft und anschließend mit den manuell erzeugten OWL-Referenzdateien verglichen. Die Evaluation erfolgt dabei im nächsten Kapitel anhand der zuvor beschriebenen Metriken. In den folgenden Abschnitten werden zunächst die Anwendungen der konkreten Strategien erläutert, beginnend mit der Durchführung des Zero-Shot Prompting bis hin zu Finetuning.

7.3.1 Zero-Shot-Prompting

Für die erste Strategie ist das Sprachmodell aufgefordert worden, OWL-Dokumente zu generieren, ohne zusätzliche Beispiele zu erhalten. Das bedeutet, dass das Sprachmodell in jedem Projekt lediglich drei Komponenten erhält. Zum einen den standardisierten Systemprompt, die PDF mit der zugehörigen Ontologie und den Nutzerprompt. Dieser besteht im Fall des Zero-Shot Prompting in den Chatinstanzen lediglich aus dem angehängten natürlichsprachlichen Referenzdokument und keinen weiteren Informationen. Diese Strategie dient auch als Maßstab für die reine Kompetenz des GPT-4o Modells in Bezug auf die Generierung von korrekten OWL-Dokumenten, die sich fest an gegebene ontologische Strukturen halten sollen.

Dadurch bietet diese erste Strategie wichtige Einblicke in die aktuellen Fähigkeiten von LLMs in Bezug auf diesen Anwendungsfall und zudem bietet das Zero-Shot Prompting eine Basis für den Vergleich mit den komplexeren, kontextbasierten Prompt Engineering Methoden, deren Verwendung mit zusätzlichem Aufwand verbunden ist. Es ist zu überprüfen, ob sich der gesteigerte Durchführungsaufwand der ausführlicheren Strategien auch in verbesserten Ergebnissen in der OWL-Generierung zeigt, die diesen zusätzlichen Aufwand rechtfertigen. Als nächstes wird Few-Shot Prompting angewendet, wobei die Nutzerprompts um Beispiele erweitert werden, die korrekte Anfrage-Antwort-Paare zeigen, wodurch eine verbesserte Qualität der Antworten erwartet wird.

7.3.2 Few-Shot-Prompting

Die Few-Shot-Strategie erweitert das vorherige Verfahren durch die explizite Bereitstellung von drei Beispielen für gewünschte Eingabe-Ausgabe Paare im Prompt selbst. Hierfür gibt es zwei verschiedene Ansätze. Die Beispielpaare können entweder im Systemprompt hinterlegt werden, oder für jedes Referenzdokument mit im Nutzerprompt integriert werden. Für diese Arbeit ist zweiteres als Methode gewählt worden, um den Systemprompt identisch zu den anderen Strategien zu halten. Durch das Few-Shot Prompting sollen die Genauigkeit, als auch die Struktur der Dokumente verbessert werden. Auch hier werden die erzeugten OWL-Aussagen als .ttl Dateien exportiert. Für diese Arbeit sind drei Beispiele pro Prompt (also auch pro Ontologie drei Beispielpaare) als Anzahl der Anfrage-Antwort-Paare gewählt worden. Der Nutzerprompt für Few-Shot Prompting mitsamt der Beispielpaare ist im elektronischen Anhang einzusehen.

Die verwendeten Nutzerprompts für die zweite Strategie sehen dabei wie folgt aus.

```
Die folgenden Beispiele zeigen korrekte Input-Output Paare zur
    Generierung der OWL-Dokumente anhand der im Projekt gegebenen
    Hinweise und der zu verwendenden Ontologie in den Projektdateien
    . Halte dich dennoch auch weiterhin an die Einschraenkungen und
    gegebenen Hinweise, die im Projekt hinterlegt sind.

###Beispiel 1
Input:...
Output:...

###Beispiel 2
Input:...
Output:...

###Beispiel 3
Input:...
Output:...
```

Listing 7.5: Prompt für Few-Shot Strategie

Anstatt nur aus dem angehängten Referenzdokument zu bestehen, teilt sich der Prompt in der Few-Shot Prompting Strategie nun in drei Abschnitte. Neben dem Referenzdokument enthält dieser nun auch eine kurze beschreibende Einleitung mit der Erklärung, wie die folgenden Beispiele zu verstehen sind und der Anweisung, dennoch den Regeln, die im Systemprompt hinterlegt sind, zu folgen. Diese Anweisung ist vor allem wichtig, um zu vermeiden, dass das Sprachmodell übermäßig den Nutzerprompt priorisiert und die Strukturanweisungen vernachlässigt. Nach dieser Einleitung folgen die drei Beispiele, die über Kommentare und Leerzeilen jeweils voneinander getrennt sind. Die Beispiele selbst folgen immer in der Reihenfolge Input aus natürlicher Sprache und Output mit erwartetem OWL-Format.

Die Beispiele für die Referenztexte sind so gewählt worden, dass sie zur jeweiligen Ontologie passen, inhaltlich und vom Umfang her ähnlich sind mit den restlichen Referenzdokumenten, um eine ähnliche Struktur und keinen verminderten Umfang zu gewährleisten. Dadurch soll sichergestellt werden, dass auch beispielsweise Fälle, wie mehrere Instanzen der selben Data Property pro Individuum, abgedeckt werden. Die konkreten Beispiele für die Prompts beinhalten für die Tierklassifikation den Elefanten Jumbo, Laika, die erste Hündin, die während des Rennens zum Mond zwischen den USA und der Sowjetunion an Bord von Sputnik 2 ins All geschickt worden ist und Shrek, ein Merinoschaf aus Neuseeland, das den Rekord hält für die meiste geschorene Wolle an einem Stück. Der Prompt für die Habsburger-Ontologie beinhaltet Beispiele zu Leopold III. Herzog von Österreich, Johanna, der Tochter von Ferdinand von Aragón, die als verrückt galt und ihr Lebensende im Exil in einem Kloster verbracht hat und zuletzt Kaiser Karl VI.. Die Beispiele für die medizinische Ontologie bestehen aus Einträgen über die Stoffwechselerkrankung Gicht, die psychische Erkrankung der posttraumatischen Belastungsstörung sowie der Bewegungsstörung Parkinson. Mit diesen drei Beispielen pro Ontologie ist eine Menge gegeben, die groß genug ist, um genügend Beispielinformationen für das Sprachmodell zu liefern und dabei nicht unverhältnismäßig zu den zehn Referenzdokumenten für die Strategie ist.

Das restliche Vorgehen gleicht der ersten Strategie. Als nächstes folgt die dritte und RAG, bei der eine noch etwas größere Menge an Beispielpaaren in einem Dokument angehängt wird, welches für den Zugriff durch das Sprachmodell hinterlegt wird und ebenfalls verbesserte Ergebnisse zu der Zero-Shot-Strategie liefern sollte.

7.3.3 Retrieval Augmented Generation

Diese Technik kombiniert die klassischen Prompting-Techniken für die Optimierung von Nutzerprompts mit dokumentgestütztem Wissensabruf, wie es bereits für den Abruf der Ontologie selbst in jeder Strategie dieses Experiments eingesetzt wird. Ziel dabei ist, durch gültige Hintergrundinformationen die Ergebnisse des Sprachmodells zu verbessern, ohne dabei den Nutzerprompt mit zu vielen Inputs zu überladen.

Für die Umsetzung bedeutet das konkret, dass die Beispiele, die beim Few-Shot Prompting noch im Nutzerprompt hinterlegt worden sind, aus diesem herausgenommen werden. Die Anfrage-Antwort-Paare werden stattdessen in ein externes Dokument verlagert. Zusätzlich ist für diese Strategie die Anzahl der Beispiele angehoben worden auf fünf pro Ontologie. Für die Tierklassifikation sind daher mit dem Pferd Incitatus des römischen Kaisers Caligula und dem Flachlandgorilla Harambe, der in den USA für Aufsehen sorgte, zwei weitere hinzugekommen. Die Habsburger Ontologie ist um ein Beispiel zu Ernst, Sohn von Leopold III., sowie ein Beispiel über Sophie Frederike, Tochter Maximilians I. von Bayern, erweitert. Für die medizinische Ontologie sind schließlich Morbus Crohn und Malaria als Beispiele hinzugekommen. Die vollständigen RAG-Dokumente sind im elektronischen Anhang einzusehen.

Eine größere Menge an Beispielen hätte auch gewählt werden können, ist aber für den Umfang von zehn Referenzdokumenten je Strategie nicht verhältnismäßig. Der Nutzerprompt besteht während der RAG-Strategie, wie auch im Zero-Shot Prompting nur aus dem angehängten Referenzdokument. Der Systemprompt ist allerdings verändert worden. Dieser enthält nun zusätzlich in der Aufgabenbeschreibung die Anweisung, das hinterlegte Dokument mit den Beispielen zu referenzieren, um Umfang, Inhalt und Struktur der Antworten daran anzulehnen. Des Weiteren wird neben der zugehörigen Ontologie am Ende des Systemprompts auch das Beispieldokument mit Dateinamen benannt. Der restliche Ablauf der Strategie ist identisch zu dem der vorherigen.

Die nächste und auch letzte Strategie ist das Finetuning, bei dem anstatt die Informationen aus Wissensdokumenten zu erhalten, das Sprachmodell Anfrage-Antwort-Beispiele als Input erhält und auf deren Basis trainiert wird, generierte Antworten in einem ähnlichen Format mit ähnlichem Umfang zu generieren.

7.3.4 Finetuning

Obwohl der Fokus der Evaluation dieser Arbeit auf den verwendeten Prompt Engineering Strategien liegt und Finetuning über reines Prompt Engineering hinausgeht, ist es dennoch eine gängige Technik zur Verbesserung von Konsistenz und Umfang der generierten Antworten von LLMs, weshalb Finetuning ebenfalls für die Experimentreihe

verwendet wird. Bei Finetuning wird ein Sprachmodell auf einem spezifischen Datensatz von Systemprompt-Nutzerprompt-Antwort-Tripeln trainiert, um besser auf eine bestimmte Aufgabe reagieren zu können. Im Fall dieser Arbeit ist für jede Ontologie jeweils ein Modell trainiert worden. Um Finetuning mit GPT-Modellen durchführen zu können, ist zunächst ein API-Zugang erforderlich, über den gültige und von den restlichen abonnementbasierten Abrechnungen unabhängige Zahlungsinformationen hinterlegt werden. Über diese werden dann anhand eines Pay-per-use-Modells API-Anfragen, Finetuning-Durchführungen und andere Tests, die über die Entwicklungsplattform laufen, abgerechnet. Als Datensätze für das Finetuning dienen die Beispiele aus den vorherigen Strategien in Kombination mit dem für die Experimentreihe verwendeten Systemprompt.

Das Finetuning für GPT-Sprachmodelle selbst erfolgt über die OpenAI Entwicklungsplattform [18]. Die Durchführung erfolgt dabei strukturiert in mehreren Schritten. Um die drei Modelle finetunen zu können, müssen die gewählten Beispiele für jede Ontologie zunächst in eine JSONL-Trainingsdatei. Für jede JSONL-Datei wird dann in jeder Zeile ein Tripel erstellt, wobei sich ein Tripel an das folgende Format hält.

```
{''messages'': [{''role'': ''system'', ''content'': ''Inhalt
   Systemprompt''}, {''role'': ''user'', ''content'': ''Inhalt
   Nutzerprompt''}, {''role'': ''assistant'', ''content'': ''Inhalt
    Antwort''}]}
```

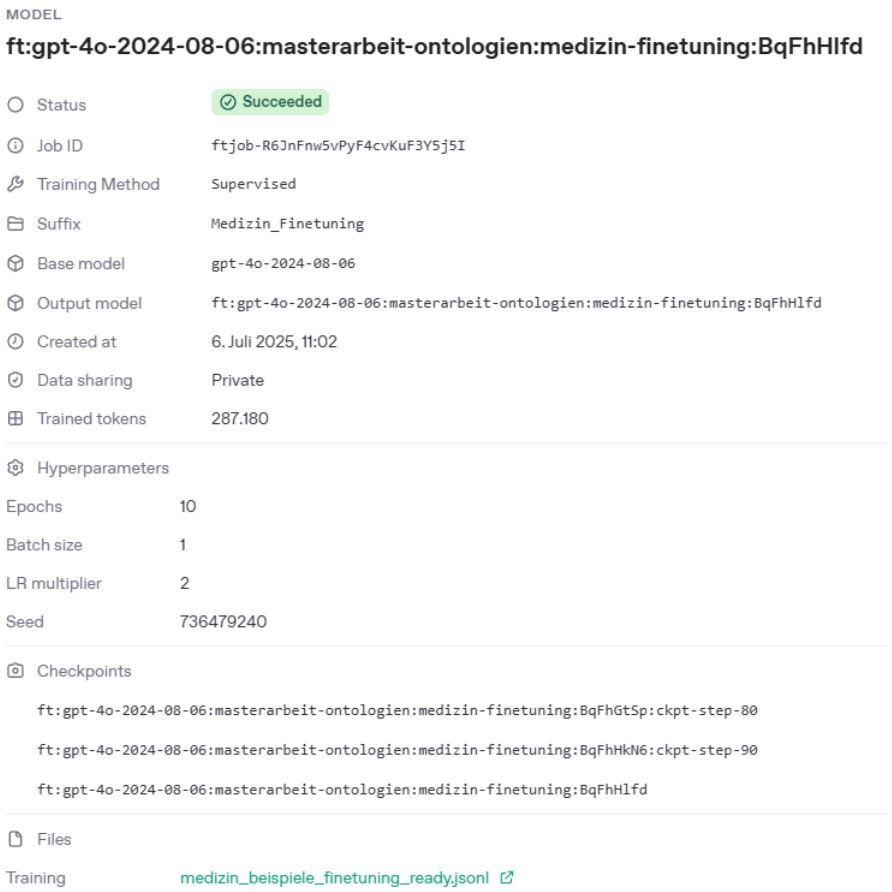
Listing 7.6: JSONL-Tripel für Finetuning Datensätze

Der Inhalt für den Systemprompt ist inhaltlich identisch mit den bisherigen Systemprompts, die für die vorherigen Strategien verwendet worden sind. Der Inhalt für den Nutzerprompt enthält in jedem Tripel den Text eines der natürlichsprachigen Referenzdokumente aus den Beispielen, die für Few-Shot Prompting und RAG verwendet worden sind. Der Inhalt für die Antwort enthält den manuell erstellten Text in OWL Turtle Syntax zu dem korrespondierenden natürlichsprachigen Beispiel aus dem Nutzerprompt des Tripels. Zu beachten gewesen ist hierbei, dass zum einen Zeilenumbrüche aus dem natürlichsprachigen Text und dem OWL-Dokument mit \n und zum anderen Anführungszeichen mit \" angegeben werden müssen, um Syntaxfehler im JSONL-Dokument zu vermeiden. Nachdem die JSONL-Datensätze für alle drei Finetuning-Modelle erstellt worden sind, kann über die Entwicklungsplattform ein Job, also die geplante Ausführung des Finetunings eines Modells, entweder manuell über die Plattform oder über die API-Schnittstelle gestartet werden. Zuerst muss das verwendete Basis-Modell festgelegt werden, also eines der von OpenAI bereitgestellten GPT-Modelle, welches als Basis für das Finetuning dient. In diesem Fall wird GPT-4o verwendet, um Konsistenz zu den vorherigen Strategien zu halten. Zudem können Suffixe für die spätere Modell-ID angegeben werden, die für den API-Aufruf verwendet werden und ein Seed kann angegeben werden, welcher verwendet wird, um bei mehrfachem Finetuning mit gleichen Datensätzen und Parametern das gleiche Ergebnis zu erzielen. Die Finetuning Dokumente sind im elektronischen Anhang hinterlegt.

Der Finetuning Job läuft nach Beginn serverseitig bei OpenAI ab. Während des Vorgangs werden mehrere Trainingsdurchläufe, sog. Epochen, durchgeführt. Nach jeder Epoche wird das Modell kontinuierlich angepasst, um die Fehler zwischen den erwarteten Ausgaben durch die Trainingssätze und den während des Finetunings entstehenden Antworten zu minimieren. Dieser Fortschritt wird auf der Entwicklungsplattform dokumentiert.

Nach einem erfolgreichen Finetuning Job wird das Modell vor der Veröffentlichung noch gegen die Nutzungsbedingungen von OpenAI getestet und das Modell wird mit einer ID für API-Aufrufe bereitgestellt, sowie in der Entwicklungsplattform selbst verfügbar gemacht.

Abbildung 7.7: Finetuning Job für medizinische Ontologie



Über die Entwicklungsplattform lassen sich daraufhin Informationen zu dem erfolgten Finetuning Job einsehen, sowie die ID des Modells entnehmen, die Anzahl der verbrauchten Tokens und auch eine Historie der Zwischenschritte während des Finetuning Prozesses. Dieser Prozess ist für alle Ontologien durchgeführt worden, sodass drei Finetuning-Modelle entstanden sind. Der restliche Prozess ist identisch mit dem Zero-Shot Prompting Prozess. Jedes dieser Modelle ist verwendet worden, um zehn OWL-Dokumente zu erzeugen und hat dabei den Systemprompt eines der natürlichsprachigen Referenzdokumente als Input erhalten. Damit sind nun alle Evaluationen durchgeführt worden und nach der Evaluation folgt die systematische Auswertung und Präsentation der Resultate. Ziel dabei ist es, die Qualität der erzeugten Ergebnisse anhand der Metriken zu bewerten und insbesondere zu untersuchen, wie sich die einzelnen Strategien im direkten Vergleich zueinander auf die Ergebnisse auswirken und welche Stärken und Schwächen sie aufweisen. Somit können die Forschungsfragen dieser Arbeit beantwortet werden. Im folgenden Kapitel werden die Ergebnisse strukturiert nach Ontologien und Strategien präsentiert und erläutert.

8 Ergebnisse

Die Ergebnisse der Evaluation und deren Auswertung werden strukturiert nacheinander präsentiert, wobei sie sich in drei Abschnitte unterteilen. Zuerst werden die Ergebnisse der syntaktischen Prüfungen dargestellt und ausgewertet. Die verschiedenen syntaktischen Fehlerarten werden dargestellt, in syntaktische Fehlerklassen unterteilt und erläutert. Auf diesen Abschnitt folgen die semantischen und logischen Auswertungen mit jeweils einzelnen Abschnitten für die verschieden komplexen Ontologien, die evaluiert worden sind. Zuletzt werden aus den Evaluationsergebnissen verschiedene semantische und syntaktische Fehlerklassen identifiziert, die entstandenen Fehler werden diesen zugeordnet und nach Auftrittshäufigkeit sowie Kritikalität bewertet.

Insgesamt wurden bei drei Ontologien mit je zehn Evaluationsdokumenten und vier eingesetzten Prompt-Engineering-Strategien (Zero-Shot, Few-Shot, RAG, Finetuning) jeweils 30 Dokumente pro Strategie generiert und evaluiert, was zu insgesamt 120 generierten OWL-Dokumenten führt. Zu jeder Ontologie bestehen also 40 Dokumente. Innerhalb der gesamten Anzahl an Dokumenten sind 6806 OWL-Aussagen generiert worden und 2403 syntaktische und semantische Fehler identifiziert worden. Die wichtigsten Ergebnisse werden in den folgenden Unterkapiteln zusammengefasst. Für einen vollständigen Einblick in die Evaluationsergebnisse, die verwendeten Prompts, die Referenzdokumente oder auch die generierten OWL-Dokumente sind diese im elektronischen Anhang hinterlegt. Die Bezeichnung der generierten Dokumente erfolgt über die jeweilige Nummer (1 bis 10), die sich auf die Nummer eines der zehn Referenzdokumente bezieht, der Ontologie-Bezeichnung und der Bezeichnung der jeweiligen Strategie. Um beispielsweise für die medizinische Ontologie das generierte Dokument zum dritten medizinischen Referenzdokument und der RAG-Strategie einzusehen, ist nach Dokument Nr. 3, ”RAG”, sowie ”Medizin” zu suchen. Eine genauere Erläuterung ist im elektronischen Anhang zu finden.

8.1 Syntaktische Evaluation der Dokumente

Die Überprüfung auf syntaktische Gültigkeit ist mithilfe eines OWL-Validators automatisiert durchgeführt worden und bei Fund von syntaktischen Fehlern sind die betroffenen Dokumente zusätzlich auch manuell überprüft worden. Von allen generierten Dokumenten enthalten insgesamt 18 von 120 Dokumenten syntaktische Fehler, was ihre direkte und potenziell automatisierte Integration verhindert. Das entspricht einer Quote syntaktisch fehlerhafter Dokumente von 15%. Sieben der fehlerhaften Dokumente sind dabei während der Generierung von Dokumenten für die Tierklassifikation entstanden, wobei alle fehlerhaften Dokumente während des Zero-Shot-Promptings entstanden sind. Von den 40 generierten Dokumenten der Habsburger Familienontologie sind lediglich zwei als syntaktisch falsch ausgewertet worden. Die medizinische Ontologie hat die meisten syntaktisch fehlerhaften

Dokumente hervorgebracht mit insgesamt neun fehlerhaften Dokumenten, darunter vier durch die Zero-Shot-Strategie und fünf durch Finetuning. Die gesamte Verteilung der syntaktisch fehlerhaften Aussagen lässt sich wie folgt nach angewandten Strategien und zugehörigen Ontologien aufschlüsseln.

Tabelle 8.1: Fehlerquote je nach Strategie

Strategie	**Dokumente**	**davon syntaktisch falsch**
Tierklassifikation Zero-Shot	10	07
Tierklassifikation Few-Shot	10	00
Tierklassifikation RAG	10	00
Tierklassifikation Finetuning	10	00
Habsburger Zero-Shot	10	00
Habsburger Few-Shot	10	00
Habsburger RAG	10	00
Habsburger Finetuning	10	02
Medizin Zero-Shot	10	04
Medizin Few-Shot	10	00
Medizin RAG	10	00
Medizin Finetuning	10	05

Zero-Shot-Prompting und Finetuning sind damit die einzigen Strategien, bei denen syntaktische Fehler aufgetreten sind. Eine qualitative Analyse der Dokumente hat gezeigt, dass sich innerhalb dieser Experimentreihe die Syntaxfehler in drei Klassen einteilen lassen.

1. Fehlende oder ungültige Deklaration von Präfixen oder Namensräumen: Diese Fehlerart tritt auf, wenn ein korrekter Namensraum verwendet worden ist, welcher zu Beginn des Dokuments aber nicht deklariert wurde, oder wenn ein ungültiger Namensraum verwendet worden ist.

2. Ungültiger Satzbau: Ungültige Zeichenketten werden verwendet

3. Falsche/Vergessene Zeichensetzung

Die am häufigsten aufgetretene Fehlerart für syntaktische Fehler ist die fehlende oder ungültige Definition von Präfixen mit insgesamt elf Fehlern. In der Tierklassifikation betrifft das alle erkannten Mängel und in der medizinischen Ontologie die vier, die in der Generierung durch die Zero-Shot-Strategie aufgetreten sind. In allen Fällen ist der Fehler durch eine vergessene Präfix-Deklaration entstanden, da der Präfix ”rdfs” für die Deklaration von Subklassenzuordnungen verwendet wird über ”rdfs:subClassOf”. Daher ist dies auch nur in der ersten und dritten Ontologie aufgetreten, da in der zweiten Ontologie bei keinem der generierten Dokumente eine Klasse mitsamt Subklassenzuordnung definiert worden ist. Diese Fehlerart ist zumindest im Rahmen der standardmäßigen Präfixe ,wie ”rdfs”, nicht als kritisch zu betrachten für eine potenzielle Integration von LLMs und Ontologien, da in den zugrundeliegenden Ontologien diese Präfixe bereits deklariert sind.

Vier der fehlerhaften Dokumente mit der zweiten Fehlerklasse für Syntaxfehler sind durch die Generierung mithilfe von Finetuning für die Habsburger Ontologie und die medizi-

nische Ontologie entstanden. Dabei handelt es sich um Fehler, die durch einen ungültigen Satzbau entstanden sind. Dabei sind die Fehler inhaltlich jeweils unterschiedlich.

```
### Ausschnitt Habsburger Finetuning Dokument Nr. 8
:Erlangung_der_Deutschen_Staatsbuergerschaft_rdf:type :Ereignis ;

### Ausschnitt Habsburger Finetuning Dokument Nr. 5
:Friedrich_der_Grosse ist eine Person und ein Mann.
:Friedrich_der_Grosse hat den Titel Koenig_von_Preussen.
:Friedrich_der_Grosse lebt in der Epoche :
   Aufgeklaerter_Absolutismus.
:Tod_von_Friedrich_dem_grossen ist die Todesursache von
   Friedrich_der_Grosse.

###Ausschnitt Medizin Finetuning Dokument Nr. 6
:Spritzbehandlung_mit_Ponesimod rdf

###Ausschnitt Medizin Finetuning Dokument Nr. 7
:Vorhofflimmern rdf:type :Krankheit_des_Kreislaufsystems ;
:ist_Vorbelastung_von :Schlaganfall ;
:vererblich True .

###Ausschnitt Medizin Finetuning Dokument Nr. 9
:Bildgebendes_Verfahren rdf:type :Diagnostisches_Verfahren ;
rdfs:subClassOf :diagnostiziert_durch EXACTLY 1 :
   verfahrensbeschreibung .
```

Listing 8.1: OWL-Eintrag: Ungültiger Satzbau in generierten Dokumenten

Die folgenden syntaktischen Fehler lassen sich in der oberen Abbildung betrachten. Das achte Dokument, das durch Generierung mit einem durch Finetuning trainierten Modell entstanden ist, hätte eine OWL-Aussage zu einem Ereignis im Leben des Otto Habsburg von Lothringen enthalten sollen. Dabei ist allerdings die Property ”rdf:type”, die das Individual der Klasse ”Ereignis” zuordnen sollte, über einen Unterstrich mit dem Individual verbunden worden. Dadurch wird dieser Abschnitt nach OWL-Syntax als Teil des Individuals betrachtet und die Klasse :Ereignisäls Property gesehen, wodurch das dritte Element eines OWL-Tripels fehlt und OWL-Interpreter diese Aussage als ungültig interpretieren. In Finetuning Dokument Nr. 5 der Habsburger Ontologie findet sich die gleiche Art von Satzbaufehler an vier Stellen zugleich. Hier sind vier Aussagen generiert worden, bei denen die Properties der OWL-Tripel als natürlichsprachige Teilsätze ausformuliert worden sind, weshalb ein OWL-Interpreter zu Beginn dieser Teilsätze einen Doppelpunkt oder einen Namensraum erwarten würde. Im Falle der potenziellen Integration von LLMs in Ontologie-basierte Systeme sind die Fehler kritischer zu betrachten, als die Präfix Deklarationen, da diese auch eingefügt in das Referenzdokument der zugrundeliegenden Ontologie zu einem Syntaxfehler führen.

Für die medizinische Ontologie hat das Sprachmodell in einem Dokument zu Schlaganfällen versucht dem Individual ”Vorhofflimmern” eine Data Property, die Boolean-Werte annimmt, den Wert ”True” zuzuweisen. Boolean-Werte werden in Turtle allerdings ausschließlich kleingeschrieben, weshalb der Wert nicht als solcher erkannt wird. Interpretiert wird dieses ”True” dadurch als das Objekt eines Tripels, bei welchem das Zeichen ”:” bzw. ein Namensraum fehlt, wodurch ein Syntaxfehler entsteht. In Finetuning-Dokument

Nr. 6 der medizinischen Ontologie fehlt sogar ein erheblicher Teil eines Tripels und die fehlerhafte Aussage endet verfrüht ohne die Vollendung von ”rdf:type” oder der Angabe der Klasse als drittes Element eines Tripels. Das vorzeitige Beenden des Dokuments ist ansonsten in keinem anderen Fall aufgetreten. Auch der Umfang der generierten Antwort sowie der Text, der als Nutzerprompt verwendet wurde, sind nicht umfangreicher als bei anderen Beispielen, weshalb ein Abbruch durch Tokenüberschreitung nicht als Ursache in Betracht gezogen wird.

Das letzte fehlerhafte Dokument, das eine falsche Zeichensetzung aufweist, verwendet eine ungültige Syntax zur Definition einer Kardinalitätsrestriktion. In dem Dokument wird das Schlüsselwort ”EXACTLY 1” verwendet, um ein OWL-Element auf eine einzelne Ausprägung einzuschränken. Dieser Schlüsselbegriff ist allerdings Bestandteil der Manchester-Syntax, einer anderen Form für OWL-Dokumente, aber nicht Teil von Turtle, wodurch dies als Fehler zu erkennen ist und nicht interpretiert werden kann. Für syntaktische Korrektheit in Turtle-Syntax hätte über die Elemente ”owl:Restriction” und fortführend eine Kardinalitätseinschränkung über ”owl:qualifiedCardinality” verwendet werden müssen. Damit hat das Sprachmodell an dieser Stelle nicht nur ein syntaktisch fehlerhaftes Dokument erzeugt, sondern auch den Systemprompt missachtet, welcher an mehreren Stellen Turtle-Syntax als alleiniges Ausgabeformat vorgibt.

Die letzte Klasse von identifizierten Syntaxfehlern ist lediglich bei der Generierung der Dokumente für die medizinische Ontologie beim Finetuning in insgesamt zwei Fällen aufgetreten. Alle fehlerhaften Dokumente aus dieser Menge weisen Tripel mit ungültiger Zeichensetzung auf. Auch hier sind an beiden Stellen unterschiedliche Fehler aufgetreten.

```
### Ausschnitt Medizin Finetuning Dokument Nr. 4
:abishtige :behandlungsdauer ''einmal_woechentlich''

### Ausschnitt Medizin Finetuning Dokument Nr. 5
:COPD rdf:type :Chronische_Krankheit_der_unteren_Atemwege ;
      :hat_Symptom :Auswurf ,
                   :Brummgeraeusche_beim_Atmen ,
                   :Pfeifgeraeusche_beim_Atmen ;
```

Listing 8.2: OWL-Eintrag: Ungültige Zeichensetzung in generierten Dokumenten

Der erste Fehler dieser Klasse ist in Finetuning Dokument Nr. 4 der medizinischen Ontologie aufgetreten. Hier ist neben einem semantischen Fehler, bei dem ein Tripel bestehend aus einem Subjekt, das nicht hätte generiert werden sollen, und einem Objekt, bestehend aus einer halluzinierten Behandlungsdauer, auch ein syntaktischer Fehler enthalten. Die Aussage endet mit keinem Satzzeichen, obwohl ein ”.” benötigt worden wäre, damit die Aussage als vollständig und gültig gilt.

Der letzte Syntaxfehler stammt aus Finetuning Dokument Nr. 5. Bei der Generierung der Fakten für die chronisch obstruktive Lungenkrankheit (COPD) sind als Letztes die Fakten für vorkommende Symptome erstellt worden, welche über die Object Property ”hat_Symptom” verknüpft werden. Nach der Auflistung des letzten Tripels hätte ”.” folgen sollen, um die letzte Aussage zu dem Subjekt COPD zu beenden. Stattdessen endet die Aussage mit ”;”, wodurch in Turtle Syntax nun eine weitere Property folgen muss, die sich auf das Subjekt COPD bezieht. Da aber ein neues Subjekt darauf folgt, entsteht an dieser

Stelle ein Syntaxfehler. Damit sind alle Syntaxfehler identifiziert.

Die syntaktische Analyse zeigt, dass innerhalb dieser Experimentreihe die Verwendung von Zero-Shot-Prompting und von Finetuning in Teilen zu der Generierung von syntaktisch fehlerhaften Dokumenten durch das Sprachmodell gesorgt hat. Vor allem die komplexeste Ontologie hat zu fehlerhaften Ergebnissen geführt. Insgesamt sind die meisten Findings in der Tierklassifikation und der medizinischen Ontologie aufgetreten. Damit treten die Fehler bei den beiden Ontologien, die eine hohe taxonomische Komplexität durch Subklassenbeziehungen aufweisen, am ehesten auf. Wobei dies ausschließlich für die Zero-Shot-Strategie zu Problemen mit der Deklaration von Präfixen geführt hat. Zuletzt ist auch anzumerken, dass die Verwendung der Few-Shot-Strategie, als auch RAG in 100% der Fälle und über alle Ontologien hinweg zu der Erstellung von syntaktisch gültigen OWL-Dokumenten geführt hat. Damit ist die syntaktische Evaluation abgeschlossen. Im folgenden Abschnitt werden die Ergebnisse der Evaluation auf semantische Qualität anhand von quantitativen Metriken präsentiert.

8.2 Ergebnisse der semantischen Evaluation

Nach der syntaktischen Evaluation sind nun in den folgenden Abschnitten die Ergebnisse der semantischen Evaluation festgehalten. Dabei werden neben semantischen Fehlern, wie nicht-korrekte Bezeichnungen, Werten und Entitäten, auch Halluzinationen des Sprachmodells und logische Inkonsistenzen beachtet. Unter logische Inkonsistenzen fallen Fehler, die den Einschränkungen der Ontologie widersprechen, wie die Verletzung von Domain und Range einer Property, falsche Werttypen von Data Properties, widersprüchliche Kardinalitäten oder Missachtung von Disjunktheitsbeziehungen. Die Ergebnisse jeder einzelnen Ontologie werden in den folgenden Abschnitten getrennt voneinander präsentiert.

Für jedes der zehn generierten Dokumente einer Ontologie pro Strategie sind die Ergebnisse der Metriken für Precision, Recall, F1-Score, Jaccard Ähnlichkeit und FZQ (Fehlerzahlquote) zu einem durchschnittlichen Gesamtwert zusammengefasst und die jeweilige Standardabweichung berechnet worden. Dies erlaubt einen direkten Vergleich der Prompt Engineering Strategien hinsichtlich Genauigkeit und Robustheit abhängig von unterschiedlich komplexen Ontologien. Die Ergebnisse der Evaluationen werden auf drei Nachkommastellen gerundet dargestellt. Insgesamt über alle Strategien und Ontologien hinweg konnte eine Precision zwischen 0,79 und 0,93 sowie ein Recall zwischen 0,70 und 0,92 erzielt werden. Die durchschnittliche FZQ, also die durchschnittliche Anzahl an fehlerhaften Aussagen pro generierter Aussage während der Experimentreihe, liegt zwischen 0,16 und 0,57. In den folgenden Abschnitten werden nun die Ergebnisse differenziert nach Ontologie dargestellt. Dabei wird analysiert, welche Strategien am besten geeignet für die jeweiligen Ontologien und ihre Komplexitätsmerkmale sind und in welchen Bereichen, bzw. bei welchen Strategien Schwächen zu erkennen sind.

8.2.1 Ergebnisse der Tierklassifikation

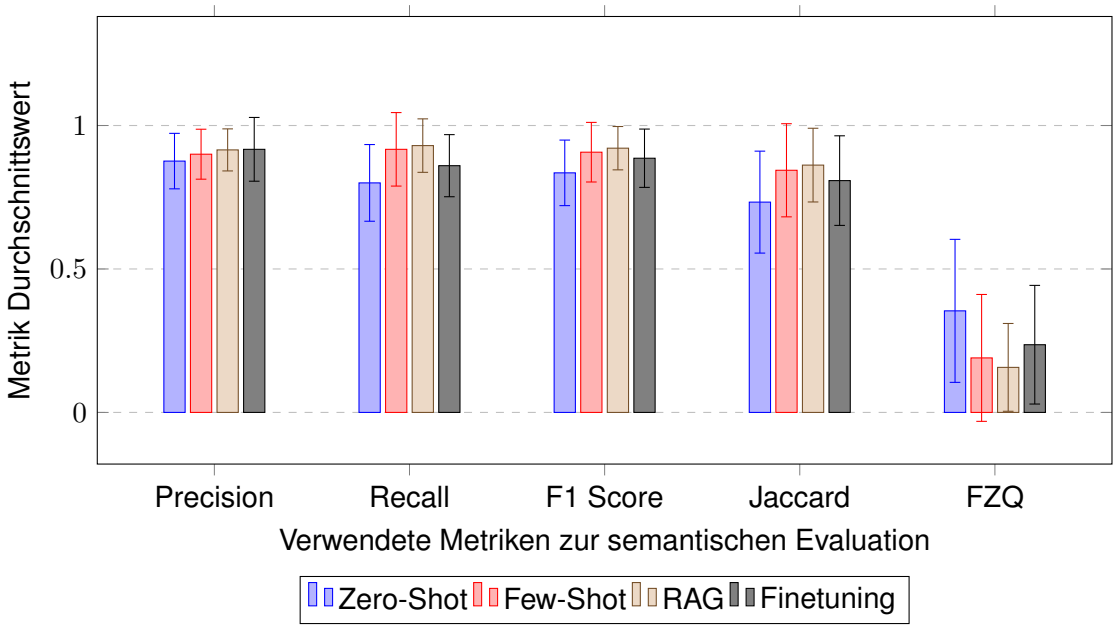


Abbildung 8.1: Ergebnisse der semantischen Evaluation der Tierklassifikation

Tabelle 8.2: Numerische Mittelwerte der Metriken - Tierklassifikation

Strategie	Precision	Recall	F1 Score	Jaccard	FZQ
Zero-Shot	0.876	0.800	0.835	0.733	0.354
Few-Shot	0.900	0.917	0.907	0.844	0.190
RAG	0.915	0.930	0.921	0.862	0.157
Finetuning	0.917	0.860	0.886	0.808	0.236

Tabelle 8.3: Standardabweichungen der Metriken -Tierklassifikation

Strategie	Precision	Recall	F1 Score	Jaccard	FZQ
Zero-Shot	0.097	0.134	0.114	0.178	0.249
Few-Shot	0.087	0.128	0.104	0.162	0.221
RAG	0.073	0.093	0.076	0.128	0.153
Finetuning	0.111	0.108	0.102	0.156	0.207

Die Tierklassifikation dient als Einstieg in die semantische Evaluation. Sie zeichnet sich hauptsächlich durch weitreichende Subklassenbeziehungen aus. Die Ergebnisse der semantischen Bewertung aller vier Strategien für die Metriken Precision, Recall, F1-Score, Jaccard Ähnlichkeit und FZQ sind in Abbildung 8.1, sowie den nachfolgenden Tabellen

zusammengefasst. Dabei ist zu erwähnen, dass für die Werte der FZQ ein geringerer Wert ein besseres Ergebnis angibt und für alle anderen Metriken ein größerer Wert.

In Bezug auf Precision zeigt sich, dass für eine Ontologie dieser Komplexität bereits die Verwendung der Zero-Shot-Strategie zu soliden Ergebnissen in der Generierung von OWL-Dokumenten führt. Mit einem Precision-Wert von 0,876 und einem Recall-Wert von 0,8 sind bereits gute Ergebnisse erzielt worden. Dennoch lässt sich durch zusätzliche Kontextinformationen mithilfe von Few-Shot, RAG oder Finetuning eine klare Steigerung aller Metriken beobachten. Sowohl Precision als auch Recall und somit auch der F1-Score liegen unter Verwendung von Few-Shot und RAG bei einem Wert von über 0,9. Dieser Trend zeichnet sich für alle Metriken ab, da dort Few-Shot Prompting und RAG die besten Ergebnisse erzielen. Dabei konnte vor allem die Fehlerzahlquote drastisch gesenkt werden, was zeigt, dass diese Strategien für besonders konsistente OWL-Ausgaben mit wenigen fehlerhaften Axiomen sorgen. Zwar liefert auch Finetuning bessere Ergebnisse als die Zero-Shot-Strategie, allerdings zeigt die Evaluation keine Vorteile gegenüber einer der anderen, kontextverwendenden Strategien, außer im Bereich Precision. Dies in Kombination mit dem zusätzlichen Mehraufwand, der durch das Aufsetzen von Dateien für Finetuning sowie den zusätzlichen Kosten für Trainingsdurchläufe entsteht, zeigt, dass an dieser Stelle keine Notwendigkeit für die Verwendung von Finetuning besteht und andere weniger aufwändige Techniken, wie RAG oder Few-Shot, zu bevorzugen sind.

Auch wenn die Standardabweichungen mit einbezogen werden, zeichnet sich ein klares Bild in den Ergebnissen ab. RAG sorgt nicht nur für durchschnittlich die besten Ergebnisse aller Metriken, die Ergebnisse dieser Strategie bilden auch die geringste Streuung ab. Damit weisen alle zehn der generierten OWL-Dokumente, die durch RAG erzeugt worden sind, die höchste Stabilität auf. Während die anderen Strategien größere Schwankungen aufweisen, bleibt die Standardabweichung bei RAG stetig unter 0,2 und in drei von fünf Fällen sogar unter 0,1.

Damit zeigen die Ergebnisse der ersten Ontologie, dass sich für taxonomische Ontologien mit geringer Komplexität, bereits durch Zero-Shot-Prompting, also nur unter Angabe der zugrundeliegenden Ontologie, einer strukturierten Aufgabenstellung über einen Systemprompt und einem Referenzdokument via Nutzerprompt überwiegend semantisch konsistente OWL-Axiome generieren lassen. Bereits die Angabe weniger Beispiele durch Few-Shot-Prompting und RAG führt zu sehr guten Ergebnissen, die zudem, wie in der syntaktischen Evaluation gezeigt, auch in 100% der Fälle syntaktisch korrekt sind.

Insgesamt ist hier RAG als Strategie zu empfehlen. RAG erzeugt semantisch die konsistentesten OWL-Axiome mit der geringsten Standardabweichung. Zudem erlaubt RAG das Auslassen von Beispielen aus dem Nutzerprompt, wodurch dieser lediglich mit dem zu übersetzenden Dokument gefüllt werden muss. Damit wird auch die Menge der verarbeiteten Tokens im Gegensatz zu Few-Shot-Prompting reduziert, da das Sprachmodell nur nach Bedarf auf Informationen aus dem hinterlegten RAG-Dokument zugreift und nicht bei jeder Anfrage an das LLM alle Beispiele durch den Nutzerprompt verarbeitet werden. Dies spart Anfrage- und Energiekosten. Des Weiteren ließe sich über das Hinterlegen in einem externen Dokument auch eine größere Anzahl an Beispielen hinterlegen, als im Nutzerprompt, wodurch die Ergebnisse potenziell noch weiter optimiert werden können.

8.2.2 Ergebnisse der Habsburger Familienontologie

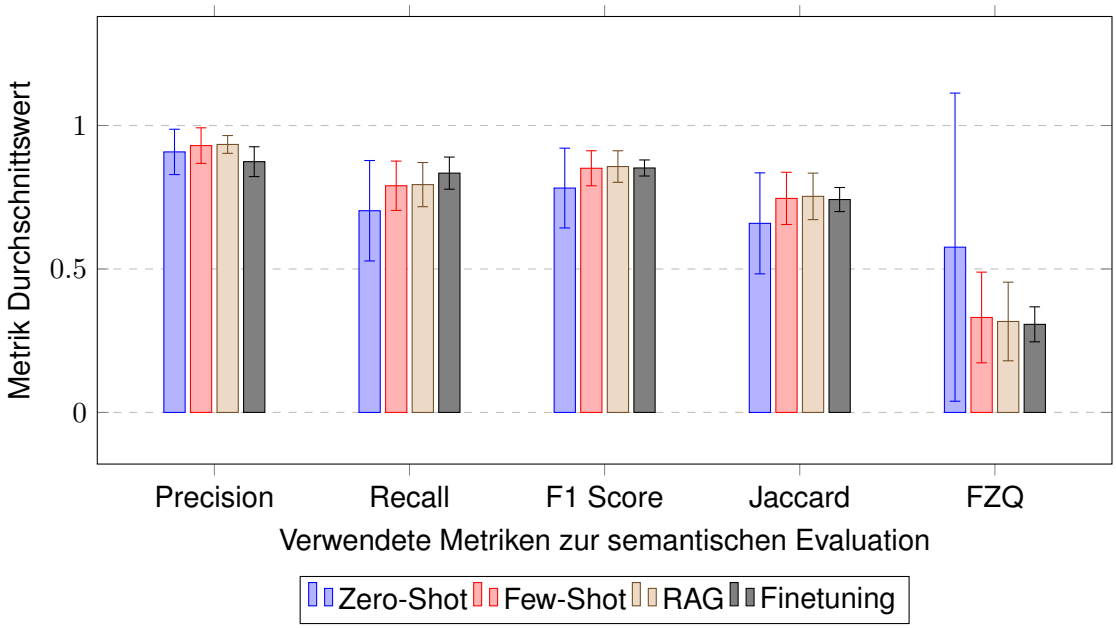


Abbildung 8.2: Ergebnisse der semantischen Evaluation der Habsburger-Ontologie

Tabelle 8.4: Numerische Mittelwerte der Metriken – Habsburger-Ontologie

Strategie	Precision	Recall	F1 Score	Jaccard	FZQ
Zero-Shot	0.908	0.703	0.782	0.659	0.576
Few-Shot	0.930	0.790	0.851	0.746	0.331
RAG	0.934	0.794	0.857	0.753	0.317
Finetuning	0.874	0.834	0.852	0.742	0.307

Tabelle 8.5: Standardabweichungen der Metriken – Habsburger-Ontologie

Strategie	Precision	Recall	F1 Score	Jaccard	FZQ
Zero-Shot	0.079	0.175	0.139	0.176	0.537
Few-Shot	0.062	0.086	0.061	0.091	0.158
RAG	0.031	0.077	0.055	0.081	0.137
Finetuning	0.052	0.056	0.028	0.042	0.061

Die Habsburger-Ontologie zeigt eine höhere Komplexität und stellt damit auch höhere Anforderungen an die semantische Darstellung von OWL-Aussagen. Statt in tiefgehenden Subklassenbeziehungen liegt hier die Komplexität in Relationen zwischen Individuals. Diese zeigen sich in familiären Beziehungen, historischen Epochen und Ereignissen sowie

Dynastiezugehörigkeiten und Heiratsbeziehungen. Hierdurch liegt der Fokus der Evaluation auf anderen Bereichen, wie der logischen Konsistenz und weniger auf den Klassenzugehörigkeiten von Individuals und korrekten Subklassenzuordnungen. Die Auswertung der durchschnittlichen Metriken zeigt ein gemischteres Bild als bei der Tierklassifikation. Anders als dort werden hier die besten Gesamtresultate nicht ausschließlich von einer Strategie erzielt, sondern auf verschiedene Strategien verteilt.

- Die höchste Precision wurde mit 0,934 durch RAG erreicht, gefolgt von Few-Shot mit 0,93 und Zero-Shot mit 0,908. Interessanterweise fallen die Ergebnisse für Finetuning mit 0,874 am schlechtesten aus, womit diese Metrik leicht zurückfällt.
- Bei Recall führt Finetuning mit 0,834 noch vor RAG und Few-Shot. Zero-Shot schneidet mit 0,703 am schlechtesten ab. Finetuning hat also besonders viele der zu erwartenden Axiome generieren können mit einem Recall von über 0,8.
- Insgesamt liefert RAG damit den höchsten F1-Score, auch wenn RAG (0,857), Finetuning (0,852) und Few-Shot (0,851) nah beieinander liegen. Zero-Shot liegt als einzige Strategie bei unter 0,8.
- Die Jaccard Ähnlichkeit liefert ein ähnliches Bild, wobei auch hier RAG die besten Ergebnisse liefert, dicht gefolgt von Few-Shot und Finetuning und zuletzt Zero-Shot.

Die FZQ fällt bei Zero-Shot Prompting mit 0,753 signifikant höher aus als bei allen anderen Strategien. Bei denen liegen die Werte im Bereich 0,3. Damit hat Zero-Shot Prompting ein besonders kritisches Verhältnis zwischen semantisch und logisch fehlerhaften Aussagen und der Anzahl insgesamt generierter Aussagen. Auf jede generierte OWL-Aussage kommen damit ca. 0,75 fehlerhafte Aussagen. Ein Blick auf die Standardabweichungen zeigt zudem, dass Finetuning und RAG im Vergleich zu den anderen Methoden die stabilsten Ergebnisse über alle Metriken hinweg geliefert haben. Besonders auffällig ist die geringe Varianz bei der FZQ durch Finetuning (Standardabweichung = 0,061) und beim F1-Score (0,028). RAG erzeugt ebenfalls stabile und wenig gestreute Ergebnisse, vor allem bei Precision mit 0,031, wo die Standardabweichung noch deutlich geringer ist als bei Finetuning. Zero-Shot Prompting erzielt hingegen Ergebnisse mit einer besonders hohen Streuung, insbesondere bei der Fehlerzahlquote mit über 0,5, was auf wenige generierte Dokumente zurückzuführen ist, bei denen die FZQ bei über 1 lag, sich also mehr Fehler identifizieren ließen, als generierte Aussagen erzeugt worden sind. Dadurch sind die Ergebnisse dieser Strategie als besonders inkonsistent zu betrachten.

Insgesamt lässt sich sagen, dass Finetuning und RAG die besten Ergebnisse über alle Metriken hinweg geliefert haben, wobei Finetuning besonders konsistente Ergebnisse mit geringer Streuung geliefert hat. Dadurch zeigt sich, dass RAG und Finetuning für relational komplexe Ontologien, wie eine solche Familienontologie, besonders geeignet und den anderen Strategien gegenüber überlegen sind. Beide Strategien sind in der Lage, die semantische Tiefe und formale Korrektheit im Vergleich zu Zero-Shot Prompting signifikant zu erhöhen. Wo Zero-Shot Prompting noch beachtliche Ergebnisse für die Tierklassifikation geliefert hat, sind die Ergebnisse hier zwar entfernt von unverwendbar, allerdings ist die Streuung beachtlich höher als bei der Tierklassifikation, sprich die Qualität der Ergebnisse schwankt stark. Diese Strategie stößt also bei steigender relationaler Komplexität, zumindest in einer Domäne wie einer Familienontologie, an ihre Grenzen.

RAG ist damit auch hier die zu empfehlende Strategie, zumindest im Umfang dieser Evaluation. Auch wenn Finetuning zeitweise bessere Ergebnisse geliefert hat, stehen diese in ihrem Ausmaß nicht im Verhältnis zum zusätzlichen Aufwand und den zusätzlichen Kosten.

8.2.3 Ergebnisse der medizinischen Ontologie

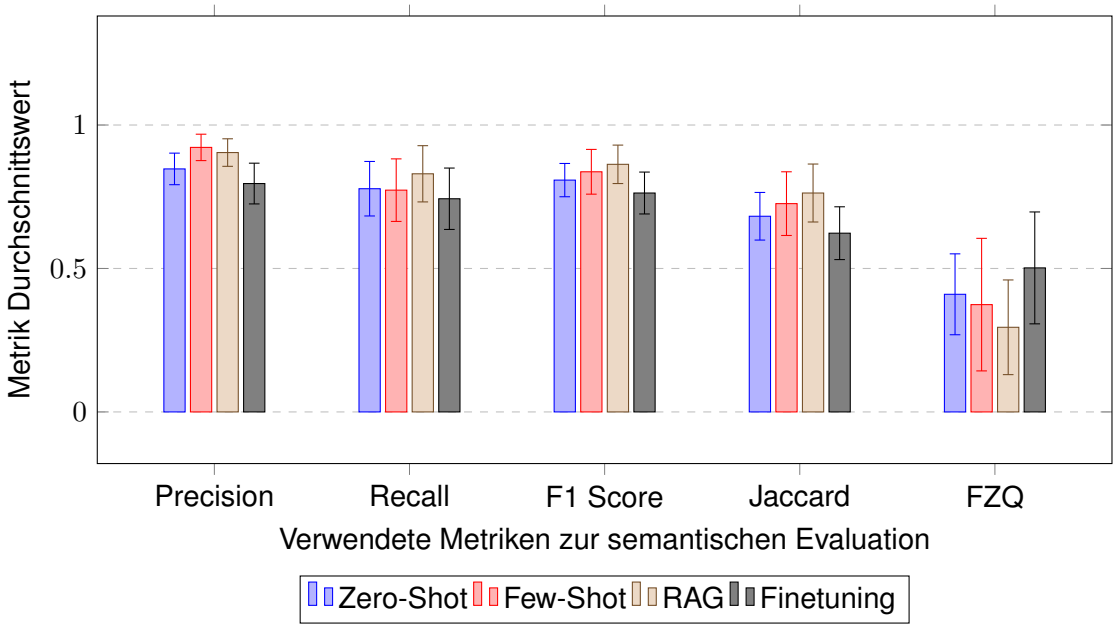


Abbildung 8.3: Ergebnisse der semantischen Evaluation der Medizin-Ontologie

Tabelle 8.6: Numerische Mittelwerte der Metriken – Medizin-Ontologie

Strategie	Precision	Recall	F1 Score	Jaccard	FZQ
Zero-Shot	0.847	0.778	0.808	0.682	0.410
Few-Shot	0.922	0.773	0.837	0.726	0.374
RAG	0.904	0.830	0.863	0.763	0.295
Finetuning	0.796	0.743	0.763	0.623	0.502

Tabelle 8.7: Standardabweichungen der Metriken – Medizin-Ontologie

Strategie	Precision	Recall	F1 Score	Jaccard	FZQ
Zero-Shot	0.055	0.095	0.058	0.083	0.141
Few-Shot	0.046	0.109	0.078	0.111	0.231
RAG	0.048	0.098	0.067	0.101	0.165
Finetuning	0.071	0.107	0.073	0.092	0.195

Die medizinische Ontologie stellt die komplexeste Testdömäne innerhalb dieser Arbeit dar. Sie vereint taxonomische Strukturen mit vernetzten relationalen Verbindungen, hier eben zwischen Krankheiten, Symptomen, Risikofaktoren, Behandlungen und Diagnosemöglichkeiten. Hinzu kommen auch logische Einschränkungen wie Kardinalitäten, disjunkte Klasseneinschränkungen oder auch Property Chains. Die numerischen Mittelwerte der Ergebnisse in Abbildung 8.3 und den dazugehörigen Tabellen zeigen, dass die Strategien auch bei einer Ontologie dieser Komplexität ein ähnliches Bild zeichnen, wie die beiden vorherigen Ontologien.

- Few-Shot Prompting liefert die höchste Precision mit einem Wert von 0,922, dicht gefolgt von RAG mit 0,904. Damit liefern beide Strategien als einzige eine Precision von über 0,9. Finetuning liefert hier die schlechtesten Ergebnisse mit einer Precision von unter 0,8.
- Beim Recall fällt der Unterschied zwischen den Strategien geringer aus. Hier liefert RAG mit 0,830 die besten Ergebnisse und auch als einzige Strategie einen Wert von über 0,8. Auch hier schneidet Finetuning am schlechtesten ab mit einem Wert von 0,743.
- Der F1-Score liegt auch bei RAG mit 0,863 am höchsten und auch hier erzielt Finetuning die schwächsten Ergebnisse.
- Auch bei der Jaccard Ähnlichkeit bildet sich dieses Muster. RAG zeigt eine Jaccard Ähnlichkeit von 0,763 und damit die höchste Wertung, gefolgt von Few-Shot-Prompting. Zero-Shot-Prompting und Finetuning schneiden am schlechtesten ab. Die Unterschiede zwischen gewollter und tatsächlicher Ausgabe ist hier also besonders hoch.
- Die Fehlerzahlquote ist von einem ähnlichen Ergebnis geprägt. Auch hier schneidet RAG am besten ab mit einer FZQ von 0,295. Finetuning sorgt erneut mit Abstand für das schwächste Ergebnis mit einer FZQ von knapp 0,5, also einer falschen Aussage für jede zweite generierte Aussage.

Auch in Bezug auf die Streuung der Ergebnisse zeigt sich ein interessantes Bild. Die Standardabweichung der Evaluationsergebnisse ist bei allen Strategien im Vergleich zu den anderen zwei Ontologien verhältnismäßig gering, was auf eine relativ konstante Modellleistung über alle generierten OWL-Dokumente hinweg hinweist für diese Ontologie. Zwar ist hier kein direkter Trend zu erkennen, mit Zero-Shot-Prompting als Strategie, die bei den meisten Metriken die geringste Streuung aufweist, gefolgt von Few-shot-Prompting und RAG, allerdings halten sich alle Ergebnisse hier in einem ähnlichen Rahmen und pro Metrik gibt es keine Strategie, die auffällig von den Ergebnissen der anderen abweicht.

Insgesamt zeigt sich auch hier, dass RAG in der medizinischen Ontologie die besten Resultate erzielt, allerdings mit einer leicht höheren Streuung als Few-Shot- oder Zero-Shot-Prompting. Finetuning zeigt seine Grenzen in der Verbindung der verschiedenen Komplexitätsmerkmale dieser Ontologie, zumindest im Trainingsausmaß dieser Arbeit. Wie zu erwarten bleibt Zero-Shot-Prompting auch in dieser Evaluation schwächer als Few-Shot-Prompting und RAG, schneidet aber übergreifend bei jeder Metrik besser ab als Finetuning.

8.2.4 Ontologie übergreifender Vergleich

Nach der detaillierten Analyse der drei Ontologien folgt nun eine übersichtliche vergleichende Betrachtung der Strategien über alle drei Evaluationsdomänen hinweg. In den Ergebnissen aller drei Ontologien zeigen sich folgende Trends:

- Zero-Shot ist in jeder Domäne eine der leistungsschwächsten Strategien und in den ersten beiden sogar die Leistungsschwächste. Besonders in Bezug auf die FZQ zeigen sich die Schwächen von mangelnden Kontextbeispielen.
- Few-Shot Prompting ermöglicht in allen Fällen eine signifikante Leistungssteigerung gegenüber Zero-Shot-Prompting. Diese Steigerung lässt sich in jeder Ontologie beobachten. Vor allem die FZQ konnte durch die Anreicherung mit Beispielen reduziert werden.
- RAG zeigt übergreifend insgesamt die besten Ergebnisse. Das Bereitstellen von Kontextbeispielen über externe Wissensdokumente sorgt für konsistente Ergebnisse mit geringerer Streuung und ähnlich erfolgreiche Ergebnisse wie Few-Shot Prompting mit leicht besseren Werten in allen Metriken. Vor allem die Precision liegt im Mittel um 0,9 bei RAG. Diese Ergebnisse werden zusätzlich von einem schlankeren Nutzerprompt als durch Few-Shot-Prompting begleitet.
- Finetuning zeigt in den ersten beiden Ontologien bessere Ergebnisse als Zero-Shot und in einigen Metriken sogar die besten Werte, nimmt aber mit steigender Komplexität an Leistungsfähigkeit ab, zumindest im Trainingsrahmen dieser Arbeit.

Über alle Ontologien hinweg zeichnen sich also deutliche Trends ab. Bei einfachen taxonomischen Strukturen und auch bei relationaler Komplexität wie der Familienontologie konnten bereits mit Zero-Shot vor allem bei Precision gute Ergebnisse erzielt werden, mit Werten von über 0,8 und teils über 0,9. Allerdings sind die Ergebnisse ohne zusätzliche Beispiele von einer großen Streuung betroffen, was diese inkonsistent macht. Bei steigender Komplexität bilden sich die Vorteile durch Few-Shot und RAG ab und zeigen, dass sich dadurch mit wenigen Beispielen bereits deutlich konsistentere Ergebnisse erzielen lassen. Im Gegensatz dazu zeigt Finetuning trotz des zusätzlichen Aufwands schwankende Werte und je nach Komplexität teils schlechtere Ergebnisse als andere Strategien.

8.3 Qualitative Bewertung und Fehlerklassen

Neben der quantitativen Evaluation über formale Metriken ist auch eine qualitative Analyse der semantischen und logischen Fehler durchgeführt worden. Hierfür sind alle identifizierten Fehlerarten zu Klassen geclustert worden, um einen Überblick über die Auftrittshäufigkeiten zu gewinnen, sowie die jeweiligen Klassen auch kritisch in Hinblick auf potenzielle Anwendungsszenarien zu bewerten. Insgesamt konnten 18 verschiedene Fehlerklassen bei 2371 logischen und semantischen Fehlern identifiziert werden.

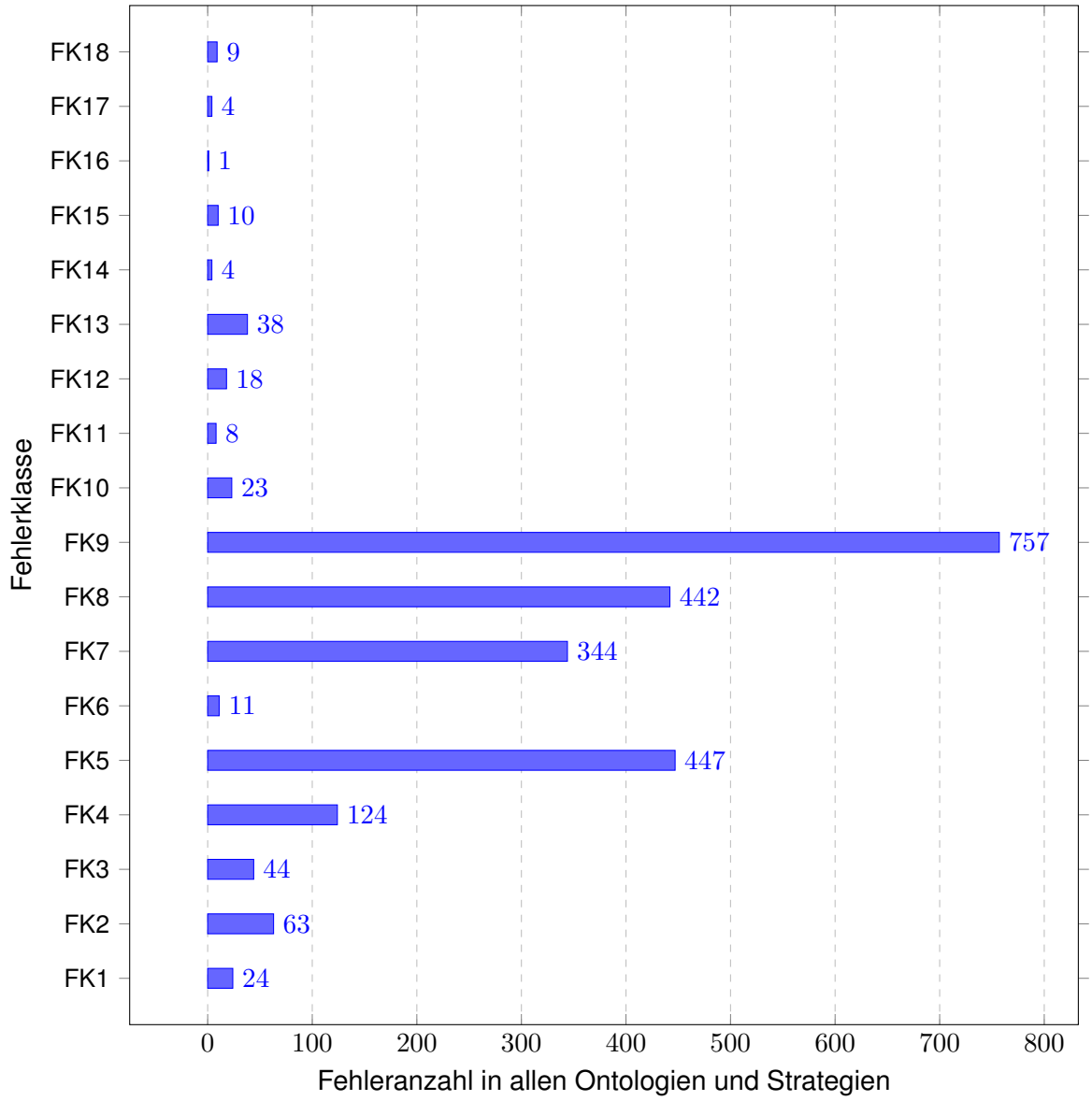


Abbildung 8.4: Häufigkeit aufgetretener Fehlerklassen bei der Ontologieanalyse

Die Fehler sind in die folgenden Klassen eingeteilt und lassen sich wie folgt beschreiben. Die Nummerierung korrespondiert mit der Nummer der jeweiligen Fehlerklasse in Abbildung 8.4. In dem horizontalen Balkendiagramm wird die Häufigkeit jeder aufgetretenen Fehlerklasse übersichtlich dargestellt.

1. Vergessene/Falsche Klassenbezeichnung: Ist hauptsächlich in der Tierklassifikation aufgetreten und bezeichnet die Nichteinhaltung von Namenskonventionen für Klassenbezeichnungen sowie das FN-äquivalente Vergessen dieser.
2. Vergessene/ Falsche Subklassenzuordnung: Entweder ist eine Klasse einer falschen Oberklasse zugeordnet worden, oder eine Subklassenzuordnung fehlt.
3. Vergessene / Falsche Klassendeklaration: Eine Klasse ist nicht deklariert worden, oder hätte nicht deklariert werden sollen.

4. Halluzinierter Data Property Wert: Ein Data Property Wert ist falsch und geht zugleich nicht aus dem Referenzdokument hervor.
5. Vergessener / Falscher Data Property Wert: Ein Data Property Wert ist falsch, wird aber im Referenzdokument an anderer Stelle oder in anderem Kontext genannt.
6. Redundante Klassendeklaration: Eine Klasse wird deklariert, obwohl diese bereits in der zugehörigen Ontologie deklariert worden ist.
7. Vergessene / Falsche rdf:type Zuordnung: Ein Individual wird einer falschen Klasse zugewiesen, oder die korrekte Zuweisung ist vergessen worden.
8. Vergessenes / Falsches Individual: Ein nicht korrektes Individual ist deklariert worden oder ein Individual ist gar nicht deklariert worden.
9. Vergessener / Falscher Object Property Wert: Ein Object Property Wert ist falsch, kommt aber als anderes Individual in der Ontologie vor und wird in ähnlichen Kontexten genannt, oder ein Object Property Wert ist vergessen worden.
10. Halluzinierter Object Property Wert: Das Individual in der Domain einer Object Property kommt nicht im Referenzdokument vor oder wird dort anders bezeichnet.
11. Falsche Individualbezeichnung: Ein Individual aus dem Referenzdokument wird mit einer Bezeichnung deklariert, die nicht aus dem Text zu entnehmen ist oder dort konkret als etwas anderes bezeichnet wird, wie ein Data Property Wert.
12. Halluziniertes Individual: Ein Individual wird deklariert, obwohl dieses nicht im Referenzdokument benannt wird
13. Falsche Object Property: Eine Object Property wird verwendet, obwohl diese aufgrund von logischen Einschränkungen, wie Domain und Range, nicht hätte verwendet werden dürfen für eines oder beide der zugehörigen Individuals.
14. Falsche Data Property: Eine Data Property wird verwendet für ein Individual, obwohl es aufgrund von logischen Einschränkungen nicht hätte verwendet werden dürfen.
15. Halluzinierte Object Property: Eine Object Property, die nicht aus der Ontologie gegeben ist, wird verwendet.
16. Halluzinierte Data Property: Eine Data Property, die nicht aus der Ontologie gegeben ist, wird verwendet.
17. Redundante Property Deklaration: Eine vorhandene Property aus der Ontologie wird erneut deklariert.
18. Redundante Property Eigenschaften: Eigenschaften, wie Domain, Range oder andere Restriktionen einer redundanten Property werden erneut angegeben.

Einige der Fehlerklassen fassen bewusst FN und FP Fehler zusammen, da das falsche Bezeichnen oder Deklarieren von Elementen im OWL-Dokument meist gemeinsam mit einem FN Fehler auftritt, bei dem das korrekte Element zugleich nicht erzeugt worden ist. Auch hier können die Ergebnisse dem elektronischen Anhang entnommen werden. Die Fehlerklassen werden im Folgenden mit FK und der korrespondierenden Nummer

benannt. FK 1 beschreibt also beispielsweise die Fehlerklasse ”Vergessene/Falsche Klassenbezeichnungen”. Die verschiedenen Fehlerklassen werden im Folgenden absteigend in ihrer Häufigkeit genauer betrachtet.

Bei über 2000 identifizierten Fehlern stechen fünf Fehlerklassen besonders heraus, die signifikant häufiger aufgetreten sind als die Übrigen. Am häufigsten ist FK9 mit 757 Fällen aufgetreten (Vergessene/Falscher Object Property Wert), obwohl diese Fehlerklasse in der ersten Ontologie nicht vorkommt, da dort keine Object Properties existieren. Besonders häufig ist diese Fehlerart als FN aufgetreten, es sind also bestimmte Object Property Werte für Individuen fälschlicherweise nicht angegeben worden. Innerhalb der Habsburger Ontologie ist dieser Fehler beispielsweise öfter aufgetreten, wenn Personen nicht zu Ereignissen oder Epochen zugeordnet worden sind, oder wenn nicht-beteiligte Personen fälschlicherweise zugewiesen worden sind. Dieser Fehler ist wohl der häufigste, da mit steigender Komplexität auch die Anzahl der Object Properties steigt und das Sprachmodell zudem redundante inverse Property Werte, wie ”ist_Bruder” und gleichzeitig auch ”hat_Bruder” generiert hat. Damit bildet die Fehlerklasse die größte Schwachstelle, ist aber von der Kritikalität geringer zu bewerten, da sich bei diesen Fehlern zumindest weder an externen Informationen außerhalb der Ontologie bedient worden ist und auch logische Restriktionen eingehalten wurden. Es bleibt das Ergebnis konsistent und ein Reasoner innerhalb der Ontologie würde keine Exception werfen.

FK5 und FK8 treten mit 447 und 442 Fällen am zweit- und dritthäufigsten auf und behandeln Vergessene/Falsche Data Property Werte, sowie Vergessene Falsche Individuals. Beide Fehlerklassen beschreiben damit ähnliche Mängel wie FK9. Diese Fehlerklassen treten in allen drei Ontologien auf, da aber seltener falsche Data Properties auftreten als Object Properties und fehlende Individuals zugleich mit n-fehlenden Object Properties einhergehen, entsteht zwischen den beiden FKs und FK9 eine Distanz in der Häufigkeit. Auch hier sind die Fehler am häufigsten als FN aufgetreten, sprich es haben Elemente gefehlt, die im Referenz OWL-Dokument vorhanden gewesen sind. Die Kritikalität der beiden lässt sich ebenfalls analog zu FK9 einordnen, da auch hier keine logisch inkonsistenten Aussagen erzeugt worden sind und auch keine Halluzinationen vorkommen.

FK7 beschreibt die falsche Zuordnung von rdf:type, also den Fall, dass ein Individual zwar korrekt erstellt worden ist, aber als die Instanz einer falschen Klasse deklariert wurde. Diese Fehlerart zeigt, dass das Sprachmodell nicht nur im Bereich relationaler Komplexität, sondern auch in der hierarchischen strukturellen Zuordnung von Elementen Schwächen aufweist. In den meisten Fällen liegt der Fehler hier bei einer Zuordnung zu einer Oberklasse oder Subklasse der eigentlich korrekten Klasse.

```
:Steiermark rdf:type :Land .
```

Listing 8.3: OWL-Eintrag: Ungültige rdf:type Zurodnung

So setzt das LLM in Dokument Nr. 2 der Zero-Shot Strategie der Habsburger Ontologie das Gebiet der Steiermark als Individual der Klasse ”Land”, obwohl es korrekterweise als Instanz der Oberklasse ’Ort’ oder unter einer neuen Subklasse von ’Ort’ wie ’Bundesland’ hätte modelliert werden müssen (Siehe Listing 8.3). Dabei ist es ein Fehler, weil es sich um ein Gebiet innerhalb des Landes Österreich handelt und auch als solches im Text benannt wird. Auch hier gilt analog der bisherigen Fehlerklassen eine mittlere Kritikalität.

Die letzte Fehlerklasse FK4, die mit einer Häufigkeit von über 100 Fällen (124) auffällt, ist das Halluzinieren von Data Property Werten. Im Unterschied zu falschen Data Property Werten aus FK5 werden hier keine Werte aus dem Referenztext falsch in den Kontext gesetzt bzw. vergessen, sondern es werden Werte, die im Referenztext nicht auftreten, als Data Property Wert einem Individual zugewiesen. Die Werte werden also aus externen Quellen vom Sprachmodell bezogen, aus dem Kontext des Textes geschlossen oder bestehende Informationen aus dem Text werden als andere Werte erkannt. Auch diese Fehlerklasse konnte in allen Ontologien identifiziert werden.

```
### Tierklassifikation Dokument Nr.6 Few-Shot
:Hanako rdf:type :Koi_Karpfen ;
        :lebt_in_Land ''Japan'' ;
### Nicht im Text gegebene Information aus Kontext geschlossen
        :lebt_in_Kontinent ''Asien'' .

### Medizin Dokument Nr.2 Zero-Shot
:Sprachstoerung rdf:type owl:NamedIndividual ,
                         :Symptom_der_Sprache_oder_Stimme ;
### Halluzinierte auftrittshaeufigkeit eines Symptoms
               :auftrittshaeufigkeit ''selten'' .
```

Listing 8.4: OWL-Eintrag: Halluzinationen Data Properties

So zeigen die Beispiele aus Listing 8.4, wie diese Fehlerklasse durch Kontextschluss entstanden ist, indem über bestehende Data Properties, wie ”lebt_in _Land” auf eine andere Property, in diesem Fall der zugehörige Kontinent, geschlossen worden ist, obwohl im Referenztext dieser nicht erwähnt worden ist. In diesem Fall handelt es sich zwar um einen Fakt, der inhaltlich korrekt ist, aber da diese Information nicht aus den Wissensdokumenten stammt, spielt dies keine Rolle, da explizit nur Informationen aus dem Text als korrekt betrachtet werden und hierdurch auch der Systemprompt missachtet worden ist. Im zweiten Beispiel ist zu sehen, wie die Auftrittshäufigkeit eines Symptoms für eine Krankheit als ”selten” festgelegt worden ist, obwohl auch hierfür keine Informationen gegeben worden sind. Es hat keinen Kontext im Text hierfür gegeben. Besonders interessant an dieser Stelle ist die Tatsache, dass wenn mehrere Individuals hintereinander im Text vorkommen, dieser Fehler bei mehreren dieser Individuals auftritt. So auch in diesem Dokument. Diese Fehlerklasse ist besonders kritisch, da hierdurch Falschinformationen, die nicht aus gegebenen Wissensdokumenten stammen, in die Ontologie gelangen und besonders in kritischen Bereichen, wie eben medizinischen Domänen, zu Fehlern führen.

Neben diesen fünf sehr häufigen Fehlerklassen treten acht Fehlerklassen mit einer Häufigkeit im zweistelligen Bereich auf. FK2 ist insgesamt 63 mal aufgetreten. Dabei geht es um falsche oder vergessene Subklassenzuordnung und sie beschreibt damit ähnlich zu FK7 einen Fehler in der hierarchischen Einordnung, nur eben hier bei Klassen anstatt von Individuen. Auch hier verweisen die fehlerhaften Aussagen in den meisten Fällen auf Sub- oder Oberklassen der eigentlich korrekten Klasse. Die Kritikalität ist hier analog zur falschen rdf:type-Zuordnung. Ähnlich zu FK2 ist auch FK3 mit 44 auftretenden Fällen zu betrachten. Hier sind Klassendeklarationen vergessen worden, oder es wurden Informationen aus dem Referenztext als Klassen modelliert, obwohl diese keine Klasse abbilden. Auch diese Fehlerklasse ist nicht in der Habsburger Ontologie aufgetreten.

FK13, die Verwendung einer falschen Object Property, ist insgesamt 38 mal aufgetreten. Hierbei können zwei Ausprägungen dieser Fehlerklasse auftreten. In der ersten Variante werden Object Properties verwendet, die für eine oder beide der verknüpften Individuals, basierend auf den Einschränkungen der Ontologie, nicht gültig sind, was nahezu ausschließlich in der medizinischen Ontologie und dort unter Verwendung von Finetuning aufgetreten ist. Die andere Variante beschreibt den Fall, dass eine falsche Property verwendet wurde, da von dieser eine Subproperty bzw. Oberproperty hätte verwendet werden müssen. Damit ist es bisher auch die häufigste Fehlerklasse, die zu einer logischen Inkonsistenz in der Ontologie führen kann.

```
### Habsburger Dokument Nr.4 Zero-Shot
:Franz_Stephan_von_Lothringen rdf:type :Mann ;
### Im Text konkret als Frau / Mann (Subproperty) Beziehung
   formuliert
    :hat_Ehepartner :Maria_Theresia .

### Medizin Dokument Nr.1 Finetuning
:Atemnot rdf:type :Symptom_des_Atmungssystems ;
### Verletzung Domain und Range
:ist_Symptomdauer_von_Symptom :wiederkehrend .
```

Listing 8.5: OWL-Eintrag: Falsche Object Properties

Wenn im Referenztext beispielsweise wie in Listing 8.5 zu sehen ein Individual als Ehepartner eines anderen modelliert wird, obwohl im Text diese Beziehung konkret als Frau / Mann Beziehung betitelt ist, was beides konkrete Subproperties sind, tritt eben dieser Fehler auf. Da diese Variante auf den ersten Blick semantisch sinnvoll erscheint, ist dieser Fehler schwer zu identifizieren. Die andere Variante zeigt sich in Fällen wie in Listing 8.5 Dokument Nr. 1 Finetuning zu sehen. Ein Individual, in diesem Fall ein Symptom einer Krankheit, wird beschrieben, aber eine verwendete Object Property hat eine Domain, die nicht mit der Klasse dieses Individuals übereinstimmt. Die Domain wäre in diesem Fall ein Individual der Klasse ”Symptomdauer” und nicht ”Symptom”. In dem konkreten Fall, der auch an anderen Stellen zu finden gewesen ist, sind Domain und Range schlichtweg vertauscht worden, allerdings sind in einzelnen Fällen auch vollkommen falsche Object Properties verwendet worden. Die Kritikalität dieser Fehlerklasse ist damit hochkritisch, da logische Fehler, wie auch semantische Fehler zu strukturellen Inkonsistenzen führen, aber in Ontologien ggf. erst nach einer logischen Konsistenzprüfung oder Anwendung eines Reasoners erkannt werden.

FK1 und FK10 treten mit je 24 und 23 Fällen nahezu identisch oft auf. FK1 beschreibt eine weniger kritische Fehlerart, bei der zwar eine Klasse korrekt deklariert worden ist, aber die Bezeichnung nicht richtig ist, da sie beispielsweise die Namenskonvention der anderen Klassen aus der Ontologie nicht einhält. Da hier keine direkten negativen Auswirkungen auf die logische Konsistenz bestehen und auch die Informationen selbst nicht vergessen worden sind, ist diese Klasse weniger kritisch. FK10 beschreibt die Halluzination von Object Property Werten, also dem Verwenden von Individuals für Property Werte, die im Referenztext nicht benannt werden oder in vollkommen unterschiedlichen Kontexten stehen. Diese Klasse ist analog den halluzinierenden Data Properties zu betrachten, tritt aber wesentlich seltener auf.

Die letzten Fehlerklassen aus dem mittleren Bereich von zweistelliger Häufigkeit sind zum einen FK6, also das redundante Erzeugen von Klassen, die bereits durch die Ontologie gegeben sind, mit 11 Fehlern und FK12 sowie FK15, das Halluzinieren von Individuals und Object Properties, mit 18 und 10 Fällen. Erstere ist mit zehn von elf Fällen nahezu ausschließlich in der Tierklassifikation aufgetreten und ist mittelmäßig kritisch zu bewerten, da das Verwenden von zwei identischen Klassen zwar zu Fehlern in der Ontologie führt, allerdings bereits beim Versuch des Hinzufügens der Klasse erkannt werden kann. Die beiden letzteren können ähnlich kritisch bewertet werden wie FK10. Hier werden Fälle beschrieben, bei denen Object Properties verwendet werden, die nicht in der hinterlegten Ontologie vorkommen, oder es werden Individuals erzeugt, die keine Erwähnung im Referenztext finden.

```
### Medizin Dokument Nr. 10 Zero-Shot
### Klinische Untersuchungen werden nicht erw hnt
:Klinische_Untersuchung rdf:type :K rperliche_Untersuchung .

### Medizin Dokument Nr.6 Finetuning
:K rperliche_Untersuchung_der_Muskelkraft rdf:type :
   K rperliche_Untersuchung ;
### anschliessend ist keine Object Property
:anschliessend :MRT_bei_Verdacht_auf_MS ;
```

Listing 8.6: OWL-Eintrag: Halluzinationen Object Properties und Individuals

So werden, wie in Listing 8.6 zu sehen, Diagnoseverfahren (klinische Untersuchung) durch das LLM modelliert, die nicht im Referenztext enthalten sind, oder es werden aus Sätzen des Referenztextes Object Properties erstellt, wie ”anschließend”, was keine durch die Ontologie gegebene Property ist. Es ist zu erwähnen, dass halluzinierte Object Properties nur im Finetuning der medizinischen Ontologie aufgetreten sind, weswegen diese Fehlerklasse auch als Ausreißer betrachtet werden kann.

Am seltensten mit Häufigkeiten im einstelligen Bereich sind FK11, sowie FK14, 16, 17 und 18 aufgetreten. FK11 sind falsche Individualbezeichnungen, was bedeutet, dass Individuals deklariert werden, die nicht aus dem Text zu entnehmen sind oder dort als etwas anderes kontextualisiert werden. So wird ”Maria Antonia”, die unter dem Beinamen ”Marie Antoinette” bekannt ist, teils in generierten Dokumenten als ”Marie Antoinette” im Individual benannt, obwohl diese Bezeichnung als Wert der Data Property ”Beiname” hätte verwendet werden müssen. FK14 beschreibt falsche Data Properties. Das bedeutet solche, die entweder eine falsche Domain haben, also einem Individual zugeordnet werden, welches diese Property nicht haben kann, oder eine falsche Range verwendet wird, sprich ein ungültiger Datentyp verwendet wird. Hiervon ist in allen Fällen bis auf einen lediglich die erste Variante aufgetreten. Die Kritikalität ist dabei analog von FK13 (den falschen Object Properties). Auch hier erzeugen die Aussagen bei logischer Prüfung oder Anwendung eines Reasoners einen Fehler und die Ontologie gilt als ungültig.

```
### Habsburger Dokument Nr. 1 Finetuning
:Stra burg rdf:type :Stadt ;
           :name ''Strassburg'' .
```

Listing 8.7: OWL-Eintrag: Falsche Data Properties

So ist beispielsweise innerhalb der Habsburger Ontologie Individuals der Klasse ”Stadt” eine Data Property durch das LLM zugewiesen worden, die als Domain die Klasse ”Person” hat (Siehe Listing 8.7). Anzumerken ist, dass diese Fehler lediglich bei Zero-Shot Prompting und bei Finetuning aufgetreten sind, bei Few-Shot und RAG aber nicht. Diese Strategien scheinen also die Fähigkeit der korrekten Zuweisung von Data Properties zu gültigen Individuals verbessert zu haben. Ähnlich zur falschen Data Property ist FK16, die halluzinierte Data Property, mit einem einzigen Fall ebenfalls sehr selten aufgetreten. Diese Fehlerklasse ist sogar nur beim Finetuning und auch nur innerhalb der medizinischen Ontologie vorgekommen. Hierbei ist für ein Individuum der Klasse ”Diagnostisches Verfahren” eine Data Property ”Verfahrensbeschreibung” verwendet worden, um das diagnostische Verfahren zu beschreiben, allerdings existiert diese nicht in der medizinischen Ontologie und die korrekte Data Property in der Ontologie trägt den Namen ”methodenbeschreibung”.

Die letzten beiden Fehlerklassen, die ebenfalls eine Häufigkeit im einstelligen Bereich haben, sind FK17 und FK18. Auch diese sind nur beim Finetuning aufgetreten und auch beide in nur einem einzelnen Dokument, nämlich der medizinischen Ontologie, nämlich Dokument Nr.9 Finetuning. Diese beiden Fehlerklassen beschreiben das redundante Deklarieren von Properties sowie das redundante Deklarieren von Eigenschaften von Properties, wie Einschränkungen von Domain und Range durch das LLM, da diese bereits in der Ontologie definiert worden sind. Damit können diese Fehlerklassen, gemeinsam mit der vorherigen FK16, auch als Ausreißer betrachtet werden. Die Kritikalität ist daher auch geringer und wie auch bei redundanten Klassendeklarationen würde hier bereits bei dem Versuch des Einfügens in die bestehende Ontologie ein Fehler geworfen werden.

Zusammenfassend zeigt die qualitative Fehlerauswertung, dass das Hauptproblem des Sprachmodells nicht bei syntaktischen Fehlern liegt, sondern das größte Problem eine unvollständige oder fehlerhafte semantische Modellierung ist. Die insgesamt größten Probleme des Sprachmodells sind:

- Das Auslassen von Informationen durch vergessene Object Property Werte, Data Property Werte, rdf:type Zuordnungen oder Individuals
- Das Interpretieren von Informationen aus dem Referenztext als falsche Object Property Werte, Data Property Werte, rdf:type Zuordnungen oder Individuals
- Das Halluzinieren von Data Property Werten bei sehr komplexen Ontologien und Referenzdokumenten (in diesem Fall bei der medizinischen Ontologie).

Diese Fehlerklassen machen mit Abstand den größten Teil aus. Gemeinsam sind sie für 89% aller semantischen und logischen Fehler verantwortlich. Auch das korrekte Zuordnen von Subklassen (FK2) ist ein sehr häufiges Problem, das aufgrund der nicht existierenden Subklassen-Komplexität der Habsburger Ontologie aber im Verhältnis nicht so häufig auftritt. Die übrigen Fehlerklassen, wie halluzinierte Properties, Individuals oder Klassen, treten eher selten und teils nur in einzelnen Dokumenten auf. Das Sprachmodell scheint strategieübergreifend die Aufgabe der korrekten Generierung hier im Verhältnis zu beherrschen. Insgesamt zeigt diese Analyse wertvolle Einblicke über die Metrik-basierte Evaluation heraus und zeigt ein ähnliches Bild, bei dem RAG die besten Ergebnisse mit weniger Fällen von Fehlern liefert und auch die Menge der auftretenden Fehlerklassen reduziert, da einige gar nicht erst unter RAG auftreten.

9

Diskussion: Limitierungen und Schwachstellen der Arbeit

Nach der detaillierten Evaluation der Ergebnisse in Kapitel 8 folgt nun eine kritische Betrachtung der Durchführung und der Ergebnisse der Arbeit. Ziel dieses Kapitels ist es, die wichtigsten Erkenntnisse einzuordnen und Limitierungen zu identifizieren, um Potenzial für zukünftige Forschung aufzuzeigen. Im Anschluss an dieses Kapitel folgt das Fazit und ein Ausblick auf mögliche Weiterentwicklungen, potenzielle Anwendungsbereiche und weitere Forschung.

Eine der zentralen Limitierungen dieser Arbeit liegt in der Beschränkung auf ein einzelnes Sprachmodell. Die gesamte Experimentreihe ist ausschließlich mit dem zu diesem Zeitpunkt aktuellen GPT-4o Modell von OpenAI durchgeführt worden. Ein Vergleich mit anderen Modellen, wie GPT-4.1 oder 4o-mini, hätte zusätzliche Erkenntnisse über geeignete Modellversionen für die Generierung von ausführlichen, ontologie-konformen OWL-Axiomen liefern können. Hier hätten Ergebnisse in der Sprachverarbeitung und der OWL-Ausgabe analysiert werden können. Da diese Modelle unterschiedlich mit Systemprompts umgehen und sich hinsichtlich Architektur, Parametern und Trainingsdaten unterscheiden, könnten hier die Ergebnisse in Bezug auf Präzision und Konsistenz variieren. Auch ein Vergleich mit anderen Sprachmodellen, wie Claude, Gemini oder LLaMA von Meta, hätte zusätzliche Erkenntnisse liefern können, insbesondere in Bezug auf OWL-Syntax, Vollständigkeit der Ergebnisse und die Fähigkeit, logische Zusammenhänge richtig zu interpretieren.

Eine weitere Limitierung liegt in der Verwendung von Finetuning. Zwar ist Finetuning im Rahmen dieser Arbeit durchgeführt worden, jedoch in reduziertem Umfang. OpenAI selbst empfiehlt in der offiziellen Dokumentation ein Volumen von hunderten Beispielen für die JSONL-Dateien, die für das Finetuning verwendet werden. Damit soll die Qualität der Ergebnisse signifikant verbessert werden können [18]. Ein solcher Umfang wäre in dieser Arbeit aufgrund der Länge und Komplexität des Erstellens von hunderten Beispielen pro Ontologie, vor allem mit Blick auf das Verhältnis zwischen Trainingsvolumen und lediglich 30 Referenzdokumenten, nicht verhältnismäßig. Des Weiteren hatte die Entscheidung, weniger Beispiele zu verwenden, auch ökonomische Gründe, da die Kosten für Finetuning mit der Menge an Daten, vor allem bei so großen Prompts, signifikant ansteigen. In dieser Arbeit sind hingegen nur wenige Beispiele zum Einsatz gekommen, was die Anpassung des Modells sowie die Generalisierbarkeit stark eingeschränkt und so nicht das volle Potenzial dieser Technik aufzeigt.

Ein zusätzliches bewusstes Designmerkmal, das zugleich als Limitierung betrachtet werden kann, ist die Verwendung eines einheitlichen Systemprompts über alle drei Ontologien hinweg. Ziel ist es gewesen die Qualitätsunterschiede in den Ergebnissen basierend auf der unterschiedlichen Komplexität der Ontologien zu erkennen und nicht auf verschiedene Einstellungen im Prompt zurückzuführen. Daher ist ein einheitlicher Systemprompt verwendet worden, um eben diesen Faktor zu entfernen. Allerdings ist hierdurch auch das Potenzial unberührt geblieben, den Systemprompt gezielt auf die semantischen Strukturen und Besonderheiten der einzelnen Ontologien abzustimmen. Individuell angepasste Prompts könnten die Ergebnisse weiter verbessern und kommen in tatsächlichen Anwendungsfällen zum Einsatz.

Obwohl alle generierten OWL-Dokumente sorgfältig mit den Referenzdokumenten verglichen worden sind und die syntaktische Evaluation automatisch über OWL-Validatoren durchgeführt werden kann, stellt die manuelle semantische und logische Evaluation einen limitierenden Faktor dar, vor allem bezüglich der Skalierbarkeit der Evaluation. Automatische Verfahren für eine solche Analyse sind aufgrund von semantischen Interpretationsspielräumen schwer durchzuführen. Mit Blick auf die Skalierbarkeit und Generalisierbarkeit der Ergebnisse müssen zudem auch die Ontologie-Domänen thematisiert werden. Durch die drei verwendeten Ontologien der Tierklassifikation, Familienontologie und medizinischen Ontologie wird ein breites Spektrum an Domänen abgebildet. Dennoch lässt sich nicht ausschließen, dass andere Domänen weitere Erkenntnisse liefern könnten und andere Stärken und Schwächen von LLMs aufzeigen. Ebenso sind die verwendeten Ontologien, trotz ihrer ausgeprägten Komplexität, auf Teilmengen der zugehörigen Domäne beschränkt.

Trotz all dieser Limitationen zeigen die Ergebnisse dieser Arbeit, inwiefern aktuelle Sprachmodelle in der Lage sind, im Gegensatz zu bestehenden Erkenntnissen aus diesem Forschungsfeld [43], auch komplexere Ontologie-Typen sowie ausführlichere natürlichsprachliche Texte zu verarbeiten und in konsistente und ontologiekonforme OWL-Axiome zu übersetzen. Die Ergebnisse lassen sich damit klar an die Forschungsfragen rückbinden:

- Interpretationsfähigkeit: Die Modelle konnten OWL-Elemente weitgehend korrekt verarbeiten, zeigten jedoch bei steigender Komplexität Fehler.
- Extraktion von Fakten: kontextbasierte Strategien (insbesondere RAG) liefern die besten F1-Scores und reduzieren auch systematische Fehlerklassen. Zero-Shot fällt am meisten zurück
- Limitierungen: Häufige Fehler sind das Fehlen oder falsche Generieren von von Property-Werten. Zudem zeigt sich auch, dass die Ontologie-Komplexität ein zentraler Faktor für die FZQ ist.

Die Validität der Ergebnisse ist durch die begrenzte Zahl an Ontologien und Referenztexten eingeschränkt. Weiterhin können ein Bias aufgrund der Domänen und ein generalisierter Systemprompt die Übertragbarkeit der Ergebnisse beeinflussen. Künftige Arbeiten sollten diesen Umfang und die Datengrundlage erweitern mit Fokus auf skalierten RAG und Finetuning-Vorgehensweisen.

10
Fazit und Ausblick

10.1 Zusammenfassung der zentralen Erkenntnisse

Ziel dieser Arbeit ist es gewesen zu untersuchen, inwiefern sich die Erklärbarkeit von LLMs durch die Kombination mit Ontologien verbessern lässt. Im Zentrum stand hierbei die Frage, ob LLMs in der Lage sind, OWL-basierte Ontologien zu verarbeiten, strukturelle Informationen korrekt zu interpretieren und auf Basis von natürlichsprachigen Dokumenten korrekte, strukturierte OWL-Aussagen zu der hinterlegten Ontologie zu generieren. Hierzu hat die Arbeit einen experimentellen Ansatz verfolgt, bei dem unterschiedlich komplexe Ontologien entwickelt und mit insgesamt vier Prompt Engineering Strategien getestet worden sind. Die Ergebnisse der syntaktischen und semantisch-logischen Evaluation zeigen, dass LLMs grundsätzlich dazu in der Lage sind, solche OWL-Dokumente zu generieren, die Qualität der Ergebnisse jedoch stark von der Komplexität der zugrunde liegenden Ontologie und der verwendeten Strategie abhängt. Im Detail lassen sich die Forschungsfragen, die in drei Abschnitte Interpretationsfähigkeit, Extraktion von formalen Fakten sowie Limitationen und Herausforderungen geteilt waren, wie folgt beantworten.

Das GPT-4o Sprachmodell ist in der Lage Elemente einer Ontologie wie Klassen, Subklassen, Properties und Restriktionen in vielen Fällen korrekt erkennen und reproduzieren zu können. Besonders unter Verwendung von Few-Shot Prompting und RAG zeigt sich über alle Ontologien hinweg eine gute Performance. Allerdings treten dennoch mit steigender Komplexität der OWL-Konzepte auch mehr semantische Fehler auf, wie die Verwechslung von inversen Properties. Diese Ergebnisse deuten darauf hin, dass die Verarbeitungstiefe von großen Sprachmodellen noch begrenzt ist und eine vollständige semantische Interpretation noch nicht zuverlässig ohne eine manuelle Überprüfung durch Experten gewährleistet werden kann.

Das Generieren formaler OWL-Fakten aus natürlichsprachigen Texten ist für jede der drei Ontologien mit steigender kontextueller Unterstützung besser gelungen. Während Zero-Shot-Ansätze noch häufig zu fehlerhaften oder unvollständigen Ausgaben führen und Finetuning, aufgrund von mangelndem Trainingsvolumen, für Eingaben dieser Komplexität kein akzeptables Aufwand-Nutzen-Verhältnis bietet, konnten durch Few-Shot und insbesondere durch RAG signifikante Verbesserungen über alle Ontologien hinweg erzielt werden. Diese zeigen sich unter anderem durch eine Precision der generierten Ergebnisse von über 0,9 in allen Ontologien, deutlich verbesserte Recall-Ergebnisse und eine wesentlich geringere Fehlerzahlquote. Auch die Streuung der Ergebnisse konnte durch die Anwendung von RAG verbessert werden und zudem treten unter Verwendung von RAG und Few-Shot Prompting die wenigsten Klassen von Fehlern auf. Es werden weniger Werte halluziniert,es entstehen keine syntaktisch fehlerhaften Dokumente und

redundante Elemente treten weniger häufig auf. Es lässt sich also sagen, dass durch die Verwendung von RAG konsistente, fehlerarme, ontologiekonforme, komplexe OWL-Axiome generiert werden können. Damit eignen sich RAG und Few-Shot für die potenzielle semi-automatisierte Unterstützung bei der Ontologieentwicklung in der Praxis. Dennoch bleibt ein Anteil von semantischen Fehlern, etwa durch vergessene Property-Werte, die sich aufgrund von natürlichsprachiger Ambiguität schwer vermeiden lassen. So bleibt eine manuelle Prüfung der Ergebnisse durch Experten unvermeidbar.

In dieser Arbeit konnten systematische Fehlerarten identifiziert werden, die sich in wiederkehrenden Fehlerklassen gruppieren lassen. Besonders falsche Zuweisungen von Property-Werten und das Vergessen von Elementen aus der gegebenen Ontologie treten besonders häufig auf. Sie zeigen die momentanen Schwachstellen von Sprachmodellen, aus Texten alle Informationen zu entnehmen und mit Ambiguitäten von Texten umzugehen, wodurch falsche Wertezuordnungen entstehen. Weiterhin zeigt sich, dass die Leistung der LLMs mit zunehmender Komplexität der Ontologien leicht abnimmt und die Anzahl der Fehler steigt. Während einfache taxonomische Strukturen erfolgreich verarbeitet werden konnten, steigen Fehlerraten bei steigender Komplexität, wie durch die medizinische Ontologie, deutlich an. Die Ergebnisse lassen sich potenziell durch weitere Beispiele für RAG, ein ausführlicheres Volumen der Trainingsdaten im Finetuning und individualisierte Systemprompts weiter verbessern.

10.1.1 Beantwortung der zentralen Forschungsfrage

Die Ergebnisse dieser Arbeit zeigen, dass Sprachmodelle wie GPT-4o Ontologien interpretieren und natürlichsprachliche Textkorpora auch größtenteils in konsistente OWL-Axiome umwandeln können. Damit ist eine Kombination von LLMs und Ontologien grundsätzlich dafür geeignet, den ersten Schritt in Richtung erklärbare KI durch hybride Systeme zu gehen.Die Generierung konformer OWL-Fakten und die Möglichkeit, diese in eine bestehende Ontologie zu integrieren, um logische Schlussfolgerungen durchzuführen und anschließend über formale Abfragesprachen Fragen stellen zu können, schafft Struktur, Nachvollziehbarkeit und Validität. Ein solcher Grad kann durch LLMs alleine nicht gegeben werden. Einer der zentralen Nachteile, das zeitintensive Aufsetzen von komplexen OWL-Fakten in der Ontologie, um die Wissensbasis erst einmal aufzubauen, lässt sich durch eine Integration von LLMs zur Faktengenerierung ausgleichen. Das Potenzial für eine tiefere Integration, etwa durch eine direkte technische Anbindung an Ontologie-Systeme, wie durch Plugins in Protégé [43], ist damit gegeben.

Für die Praxis bedeutet dies Folgendes. Die Arbeit verdeutlicht, dass eine hybride Pipeline aus LLMs und Ontologien konkrete Vorteile bieten kann.

1. Transparenz: Ontologien und Reasoner können durch eine Integration im Vergleich zu LLMs alleine die Ableitung von Fakten nachvollziehbarer gestalten
2. Effizienz: LLMs reduzieren den manuellen Aufwand bei der Faktenerzeugung erheblich
3. Zuverlässigkeit: Durch die Verwendung von Kontextstrategien, wie RAG, steigert sich die Konsistenz und die Anzahl der Fehlerklassen wird minimiert.

10.2 Ausblick

Die Ergebnisse dieser Arbeit legen eine fundierte Grundlage für zukünftige Forschungsarbeiten im Bereich der Verbindung von LLMs und Ontologien, um Erklärbarkeit und Nachvollziehbarkeit zu verbessern. Zugleich zeigen sie klare potenzielle Forschungsfelder auf, sowohl hinsichtlich methodischer Tiefe als auch technischer Weiterentwicklung.

- Zum einen können die Limitierungen dieser Arbeit als Basis für kommende Arbeiten dienen. Der Vergleich unterschiedlicher GPT-Modelle, und auch der Vergleich unterschiedlicher Sprachmodelle könnte in zukünftigen Arbeiten untersucht werden, um Erkenntnisse über Stärken und Schwächen verschiedener Modelle zu identifizieren.
- Die Weiterentwicklung von Sprachmodellen, die möglichst konsistente und vollständige Ergebnisse in der Übersetzung von Texten liefern, ist eine weitere Forschungsrichtung. Durch gezieltes Finetuning mit größerem Trainingsvolumen, sowie der Individualisierung von Systemprompts auf bestimmte Ontologien, können die erzielten Ergebnisse potenziell weiter verbessert werden.
- Ein vielversprechender Ausblick liegt in der direkten Integration von Sprachmodellen in Ontologie-Umgebungen wie Protégé. Bestehende Forschung hat bereits gezeigt, dass sich Sprachmodelle für einfache Generierung von Axiomen integrieren lassen [43]. Durch diese Arbeit konnte zudem gezeigt werden, dass auch komplexere Mengen von Axiomen durchaus mithilfe von LLMs erzeugt werden können. Hierdurch öffnen sich neue Forschungsrichtungen. Darunter die automatisierte Übersetzung komplexerer OWL-Axiome, aufbauend auf bisherigen Forschungsergebnissen. Aber auch andere Anwendungsfälle ließen sich überprüfen, wie das Erkennen von inkonsistenten oder fehlenden Axiomen, sowie das Vorschlagen von passenden Konstrukten in Ontologien durch LLM-gestützte Echtzeit-Prüfer. Auch die Entwicklung einer dialogbasierten Integration in Ontologie-Systemen oder das Übersetzen von natürlichsprachlichen Fragen in komplexe SPARQL-Queries ließe sich entwickeln.

Solche Erweiterungen könnten zukünftig die Geschwindigkeit, in der neue Ontologien aufgesetzt werden, erhöhen, die Bedienbarkeit verbessern und auch Fachanwendern und Fachanwenderinnen ohne tiefer gehende OWL-Kenntnisse die Mitentwicklung an Ontologien erleichtern. Einige der Hindernisse, wie Einstiegshürden oder der manuelle Aufwand, sowie die mangelnde Skalierbarkeit, ließen sich durch Weiterentwicklungen für LLM-gestützte Ontologie-Systeme reduzieren.

Ein konkretes Szenario zur Demonstration der Vorteile von Ontologie-basierten Systemen gegenüber modernen Sprachmodellen und somit der Verdeutlichung der Vorteile einer Integration, ist die vergleichende Beantwortung semantisch und strukturell anspruchsvoller Fragen. Durch Ontologie-Systeme erfolgen diese über SPARQL-Abfragen und bei generativen Sprachmodellen eben durch natürlichsprachige Anfragen. Die Hypothese wäre, dass die SPARQL-Abfragen präzise, nachvollziehbare und vor allem auch reproduzierbare Ergebnisse liefern und auch nur das, was konkret gefragt worden ist, zurückgeliefert wird. Ein LLM wird eher eine generalisierte Aussage liefern oder unvollständige, sogar inkorrekte Antworten zurückliefern.

Erste Tests anhand der in dieser Arbeit erstellten Ontologien deuten bereits darauf hin, dass sich diese Hypothese bewahrheitet. Weitere Forschung in dieser Richtung könnte

die Vorteile einer Integration beider Systeme weiter in den Vordergrund rücken. So liefert beispielsweise das GPT-4o Modell auf folgende natürlichsprachige Anfrage:

> ”Welche Individuen von Knochenfischen oder deren Subklassen leben oder lebten in dem Land England?”

als Antwort zuerst die beiden direkten Subklassen ”Strahlenflosser” und ”Fleischflosser” sowie eine Reihe von Fischarten, die das Sprachmodell als ”Individuen” benennt, die in den Gewässern in England heimisch sind. Zudem wird auch angegeben, in welchen Gewässern diese leben und es wird ein Fazit gezogen. Damit liefert das Sprachmodell keine tatsächlichen Individuen, sondern lediglich Tierarten, und zudem auch mehr Informationen, als durch die Anfrage erfragt worden sind. Die gleiche Anfrage als SPARQL-Query in der Ontologie der Tierklassifikation sieht wie folgt aus:

```
### SPARQL-Abfrage Tierklassifikation
SELECT ?individuum
WHERE {
  ?klasse rdfs:subClassOf* :Knochenfische .
  ?individuum rdf:type ?klasse .
  ?individuum :lebt_in_Land ?land .
  FILTER(str(?land) = ''England'')
}
```

Listing 10.1: SPARQL-Eintrag: SPARQL-Abfrage Tierklassifikation

Diese Abfrage gibt lediglich ”Charles_Darwin” als Antwort aus, was im Kontext der Ontologie auch die korrekte Antwort ist. Charles Darwin ist ein Individuum von Typ ”Moderne_Menschen”, was über mehrere Subklassenbeziehungen hinweg eine Subklasse von ”Knochenfisch” ist (Siehe Abbildung 7.1). Das ist eine Antwort, auf die das Sprachmodell nicht gekommen wäre und eine Antwort, die immer identisch sein wird, vorausgesetzt die Struktur wird nicht erweitert und die Abfrage bleibt dieselbe. Ähnliche Ergebnisse zeigen sich auch in den beiden anderen Ontologien. Auf die Frage, welche Mitglieder der Habsburger Dynastie an mehr als 2 Ereignissen der Periode „Das Erbe der Babenberger“ teilgenommen haben und wie viele Ereignisse es pro Person waren, konnte das Sprachmodell keine Antwort liefern. Auf die Frage, welche Symptome bei mindestens zwei Krankheiten auftreten, lieferte es eine beispielhafte Antwort mit fünf beispielhaften Symptomen und auch Zusatzinformationen, die nicht gefragt gewesen sind. In allen Fällen sind die Antworten des Sprachmodells also entweder unvollständig, ungenau, thematisch verfehlt oder enthalten ungefragte Zusatzinformationen. Die Ergebnisse dieser Vergleiche lassen sich im elektronischen Anhang unter ”Ausblick” einsehen.

Abschließend lässt sich sagen, dass die Kombination aus LLMs und Ontologien Vorteile gegenüber der alleinigen Verwendung von LLMs zur Entscheidungsunterstützung im Bereich Explainable AI bieten kann und auch bestehende Nachteile und Limitierungen von ontologie-basierten Systemen durch eine Integration vermindert werden können. Sprachmodelle sind mittlerweile in der Lage auch Mengen von komplexeren OWL-Fakten zu erzeugen. Forschung im Bereich der technischen Integration von Sprachmodellen und Ontologien ist ein aktuelles Thema und der Ausblick zeigt eine Reihe von potenziellen Forschungsrichtungen, an denen angesetzt werden kann. Diese Vision der Entwicklung von hybriden, LLM-gestützten Ontologien durch eine Integration stellt ein vielversprechendes zukünftiges Forschungsfeld im Bereich KI und XAI dar.

A
Anhang

A.1 Glossar

1. AI / KI (Artificial Intelligence / Künstliche Intelligenz): Bereich der Informatik, der sich mit der Entwicklung von Systemen befasst, die kognitive Fähigkeiten nachahmen
2. Annotation Property: Owl-Eigenschaft für das Anhängen von Metadaten an Elemente
3. Äquivalente Klasse: Klassen, die nicht die dieselbe Menge von Individuals definieren
4. Asymmetric Property: Property, die nicht in beide Richtiungen gleichzeitig zeigen darf
5. Axiom: Formale Regel einer Ontologie, die Beziehungen zwischen Konzepten definiert und für logische Ableitung verwendet wird
6. Causal Language Modelling (CLM): Trainingsmethode für Sprachmodelle, bei dem Folgewörter basierend auf bestehenden Worten vorhergesagt werden
7. Chain-of-Thought (CoT): Technik, bei der das Modell Zwischenschritte explizit verbalisieren soll
8. Closed World Assumption (CWA): Konzept, bei dem angenommen wird, dass unbekannte Informationen falsch sind
9. Data Property: OWL-Relation zwischen einem Individual und einem Datenwert
10. Description Logic (DL): Gruppe von formalen Logiken, die auch zur Konzeption von Ontologien verwendet wird
11. Disjunkte Klasse: Klassen, die nicht dieselben Individuals haben dürfen
12. Domain/Range: Klassenrestriktionen für Subjekte und Objekte einer Property in OWL
13. Explainable AI (XAI): Teilbereich der künstlichen Intelligenz, der darauf fokussiert ist, Entscheidungsprozesse von KI-Systemen transparent und nachvollziehbar zu gestalten
14. F1-Score: Harmonisches Mittel aus Precision und Recall
15. Fehlerzahlquote (FZQ): Anteil der fehlerhaften OWL-Aussagen an allen generierten Aussagen
16. Few-Shot Prompting: Aufgabenlösung mit wenigen Beispielen im Prompt

17. Finetuning: Trainieren eines vortrainierten Modells für spezifische Aufgaben anhand von JSONL basierten Trainingsdaten
18. Functional Property: Property, die pro Subjekt nur einen einzelnen Wert zulässt
19. ICD-10-GM: Deutsche Modifikation der ICD-10-Klassifikation für Krankheiten durch das BfArM
20. Inverse Property: Zu einer Object Property entgegengesetzte Relation
21. Irreflexive Property: Property, bei der das Individual niemals auf sich selbst zeigt
22. Jaccard Ähnlichkeit: Maß für die Ähnlichkeit zwischen zwei Mengen als Verhältnis der Schnittmengengröße zur Größe der Vereinigung beider Mengen
23. JSON (JavaScript Object Notation): Ein leichtgewichtiges Datenformat zur strukturierten Übertragung von Daten zwischen Systemen, das häufig für Integrationen genutzt wird.
24. JSONL: JSON-Format, das zeilenbasiert ist und unter anderem für Finetuning verwendet wird
25. Kardinalitätsrestriktion: Einschränkung der Anzahl von zulässigen Property Eigenschaften und Beziehungen
26. Large Language Model (LLM): Ein auf Textkorpora trainiertes neuronales Netz zur Verarbeitung und Generierung von natürlicher Sprache
27. Machine Learning (ML): Teilgebiet der Künstlichen Intelligenz, das ohne explizite Programmierung für Computer das Lernen aus Daten ermöglicht
28. Masked Language Modelling (MLM): Trainingsmethode für Sprachmodelle, bei der Wörter zufällig maskiert werden und das Sprachmodell diese vorhersagen muss
29. Natural Language Processing (NLP): Teilgebiet der Künstlichen Intelligenz, das sich mit der automatischen Verarbeitung von natürlicher Sprache beschäftigt.
30. Neurales Netz: Mathematisches Konzept, das Knoten miteinander verbindet, um eine neuronale Gehirnstruktur zu imitieren
31. Object Property: OWL-Relation zwischen Individuals
32. Ontologie: Konzept zur formalen Wissensrepräsentation in einem spezifizierten Bereich durch Beziehungen zwischen Konzepten und ihren Eigenschaften
33. Open World Assumption (OWA): Konzept von OWL-Ontologien, bei dem fehlende Informationen nicht als falsch, sondern als nicht bekannt interpretiert werden
34. Precision: Maß für die Genauigkeit eines Modells, das angibt, welcher Anteil der als positiv klassifizierten Rückgabewerte tatsächlich positiv ist
35. Prompt Engineering: Techniken zu gezielten Optimierung von Eingaben in Dialog-basierten KI-Systemen

36. Property Chain: OWL-Element für abgeleitete Relation aus mehreren miteinander verketteten Properties
37. Protégé: Open Source Anwendung zur Erstellung und Verwaltung von Ontologien
38. RDF Schema (RDFS): Sprache zur Definition einfacher Schemata über RDF
39. Reasoning: Prozess der automatischen logischen Ableitung von Fakten basierend auf vorhandener Logik
40. Recall: Maß für die Vollständigkeit eines Modells, das angibt wie viele der möglichen tatsächlich positiven Rückgabewerte zurückgegeben worden sind
41. Resource Description Framework (RDF): Formales Sprachmodell zur Darstellung von Daten im semantischen Web durch Subjekt-Prädikat-Objekt Tripletts
42. Retrieval-Augmented Generation (RAG): Prompting mit zusätzlich bereitgestellten Wissensdokumenten
43. Self-Attention: Mechanismus, mit dem ein KI-Modell die Tokens innerhalb von einer Sequenz gewichtet
44. Semantic Web: Erweiterung des World Wide Web, durch die Daten maschineninterpretierbar verteilt aufgesetzt werden können
45. SPARQL: Abfragesprache für Daten, die mittels RDF gespeichert sind
46. Standardabweichung (SD): Streuungsmaß, das die Variabilität von Messwerten bezeichnet
47. Symmetric Propert: Property, deren Relation in beide Richtiungen zeigt
48. Taxonomie: Hierarchische Struktur zur Klassifikation von Konzepten, die auch in Ontologien zum Tragen kommt
49. Tokenisierung: Das Zerlegen von Text in Token, damit das Modell diesen verarbeiten kann
50. Transformer-Architektur: Neuronale Architektur, die das Kernkonzept moderner LLMs bildet und auf Self-Attention basiert
51. Turtle (TTL): Syntax für RDF/OWL-Graphen
52. Web Ontology Language (OWL): Formale Sprache zur Darstellung von Ontologien im Web, die auf RDF basiert
53. Zero-Shot Prompting: Aufgabenlösung eines Sprachmodells lediglich anhand einer beispiellosen Aufgabenstellung

A.2 Zusatzmaterial

Abbildung A.1: Ausschnitt Tierklassifikation ausführlich

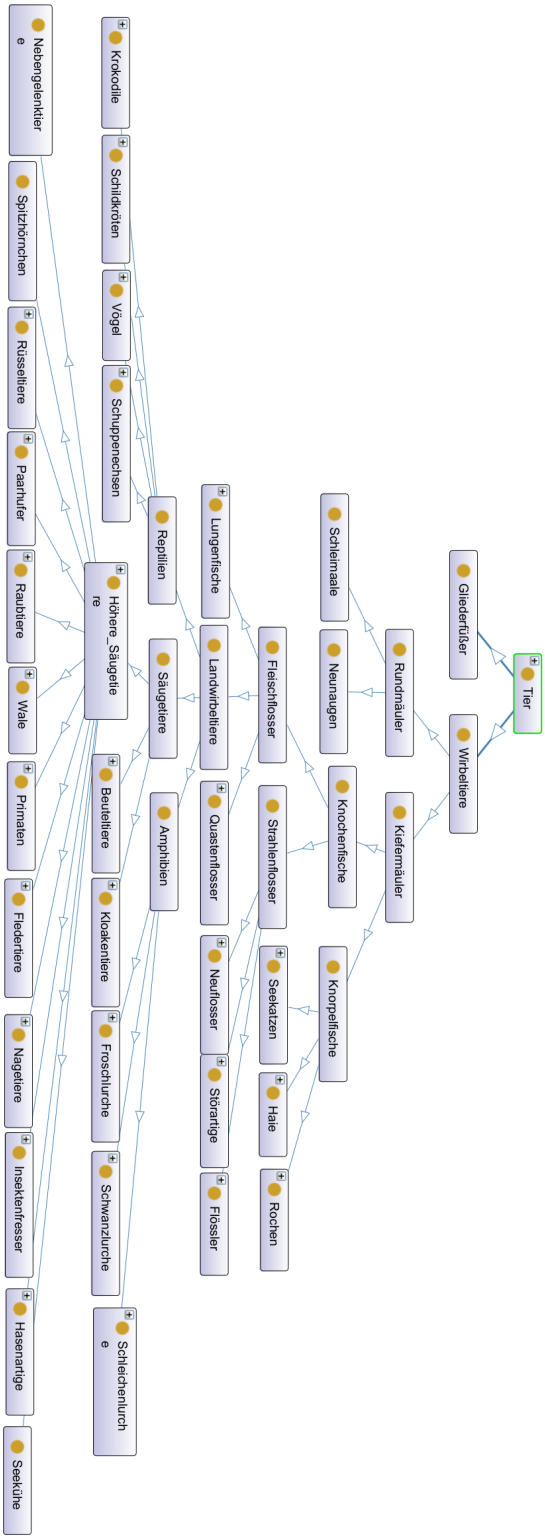


Abbildung A.2: Familienontologie vollständig

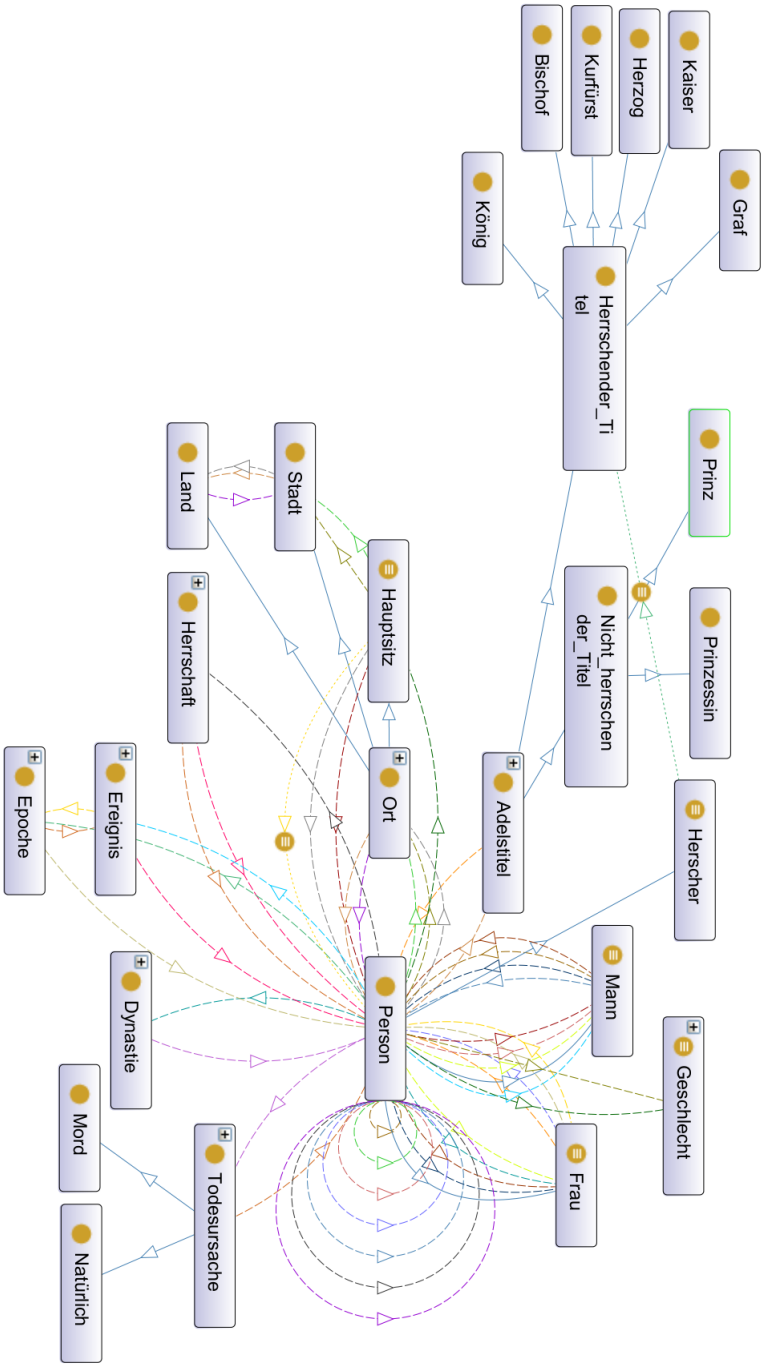
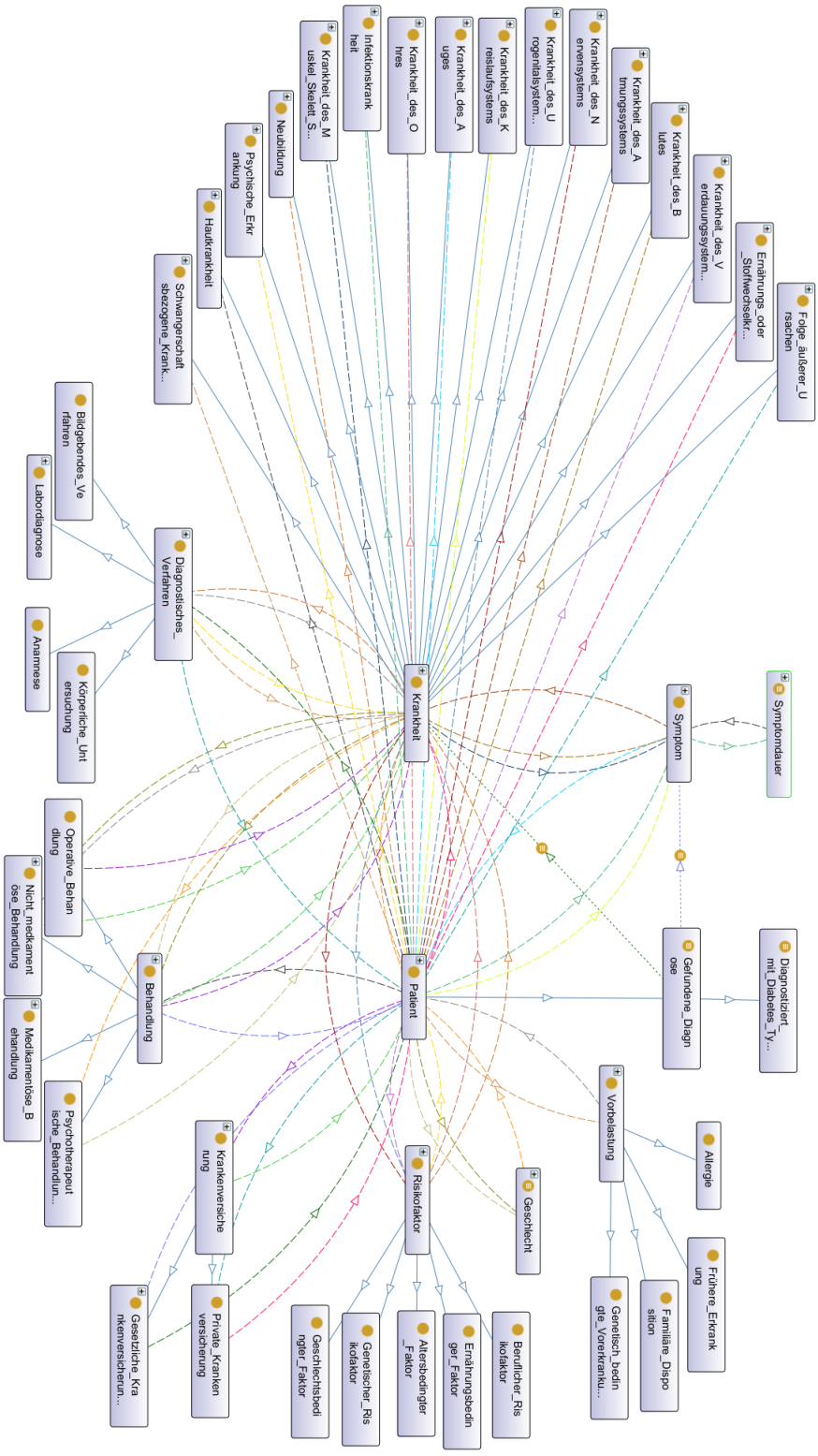


Abbildung A.3: Ausschnitt medizinische Ontologie ausführlich



Literatur

[1] Adadi, A. und Berrada, M., „Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI),“ *IEEE Access*, Jg. 6, S. 52 138–52 160, 2018. DOI: 10.1109/ACCESS.2018.2870052.

[2] Agarwal, A., Wong-Fannjiang, C., Sussillo, D., Lee, K. und Firat, O., „Hallucinations in Neural Machine Translation,“ in *The Sixth International Conference on Learning Representations, ICLR 2018, Vancouver, Canada Apr 30 - May 3, 2018*, 2018.

[3] Bahdanau, D., Cho, K. und Bengio, Y., „Neural Machine Translation by Jointly Learning to Align and Translate,“ in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Bengio, Y. und LeCun, Y., Hrsg., 2015. Adresse: http://arxiv.org/abs/1409.0473.

[4] *BfArM - ICD-10-GM*. besucht am 18. Mai 2025. Adresse: https://www.bfarm.de/DE/Kodiersysteme/Klassifikationen/ICD/ICD-10-GM/_node.html.

[5] Brachman, M., El-Ashry, A., Dugan, C. und Geyer, W., „How Knowledge Workers Use and Want to Use LLMs in an Enterprise Context,“ in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, Ser. CHI EA '24, New York, NY, USA: Association for Computing Machinery, Mai 2024, S. 1–8, ISBN: 979-8-4007-0331-7. DOI: 10.1145/3613905.3650841. besucht am 15. März 2025. Adresse: https://dl.acm.org/doi/10.1145/3613905.3650841.

[6] Brown, T. B. u. a., „Language models are few-shot learners,“ in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Ser. NIPS '20, Red Hook, NY, USA: Curran Associates Inc., Dez. 2020, S. 1877–1901, ISBN: 978-1-7138-2954-6. besucht am 27. Feb. 2025.

[7] Büchel, J., Marc, S. und Barbara, E., *digitalisierungsindex-2024*, Dez. 2024.

[8] Burkhardt, H. und Smith, B., Hrsg., *Handbook of Metaphysics and Ontology*. Munich: Philosophia Verlag, 1991.

[9] Cambria, E., Malandri, L., Mercorio, F., Nobani, N. und Seveso, A., *XAI meets LLMs: A Survey of the Relation between Explainable AI and Large Language Models*, arXiv:2407.15248 [cs], Juli 2024. DOI: 10.48550/arXiv.2407.15248. besucht am 26. Feb. 2025. Adresse: http://arxiv.org/abs/2407.15248.

[10] Carterette, B., „Precision and Recall,“ en, in *Encyclopedia of Database Systems*, LIU, L. und ÖZSU, M. T., Hrsg., Boston, MA: Springer US, 2009, S. 2126–2127, ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_5050. besucht am 27. Feb. 2025. Adresse: https://doi.org/10.1007/978-0-387-39940-9_5050.

[11] Chen, Z., Chen, J., Chen, Y., Yu, H., Singh, A. K. und Sra, M., *LMExplainer: Grounding Knowledge and Explaining Language Models*, arXiv:2303.16537 [cs], Juli 2024. DOI: 10.48550/arXiv.2303.16537. besucht am 9. Apr. 2025. Adresse: http://arxiv.org/abs/2303.16537.

[12] *COL*, ISSN: 2405-8858. besucht am 1. Mai 2025. Adresse: https://www.catalogueoflife.org/.

[13] *DeepSeek*, en. besucht am 19. März 2025. Adresse: https://www.deepseek.com/.

[14] Devlin, J., Chang, M.-W., Lee, K. und Toutanova, K., „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)*, Burstein, J., Doran, C. und Solorio, T., Hrsg., Minneapolis, Minnesota: Association for Computational Linguistics, Juni 2019, S. 4171–4186. DOI: 10.18653/v1/N19-1423. besucht am 19. März 2025. Adresse: https://aclanthology.org/N19-1423/.

[15] Doersch, C. und Zisserman, A., „Multi-task Self-Supervised Visual Learning," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, S. 2070–2079. DOI: 10.1109/ICCV.2017.226.

[16] Dwivedi, R. u. a., „Explainable AI (XAI): Core Ideas, Techniques, and Solutions," *ACM Comput. Surv.*, Jg. 55, Nr. 9, 194:1–194:33, Jan. 2023, ISSN: 0360-0300. DOI: 10.1145/3561048. besucht am 26. Feb. 2025. Adresse: https://dl.acm.org/doi/10.1145/3561048.

[17] *EU-Gesetz zur künstlichen Intelligenz — Aktuelle Entwicklungen und Analysen zum EU-KI-Gesetz*, de. besucht am 6. Apr. 2025. Adresse: https://artificialintelligenceact.eu/de/.

[18] *Fine-tuning - OpenAI API*, de-DE. besucht am 16. Juli 2025. Adresse: https://platform.openai.com.

[19] Fischer, S. W. S. und Boer, B. d., „Negotiating becoming: a Nietzschean critique of large language models," en, *Ethics and Information Technology*, Jg. 26, Nr. 3, S. 42, Juni 2024, ISSN: 1572-8439. DOI: 10.1007/s10676-024-09783-5. besucht am 15. März 2025. Adresse: https://doi.org/10.1007/s10676-024-09783-5.

[20] Floridi, L., „AI and Its New Winter: from Myths to Realities," en, *Philosophy & Technology*, Jg. 33, Nr. 1, S. 1–3, März 2020, ISSN: 2210-5441. DOI: 10.1007/s13347-020-00396-6. besucht am 17. März 2025. Adresse: https://doi.org/10.1007/s13347-020-00396-6.

[21] *Franz Joseph I. — Die Welt der Habsburger*. besucht am 1. Juni 2025. Adresse: https://www.habsburger.net/de/personen/habsburger-herrscher/franz-joseph-i.

[22] *Gemini – chatten und inspirieren lassen*, de. besucht am 19. März 2025. Adresse: https://gemini.google.com.

[23] Goodfellow, I., Bengio, Y. und Courville, A., *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[24] Grosan, C. und Abraham, A., „Rule-Based Expert Systems," en, in *Intelligent Systems: A Modern Approach*, Grosan, C. und Abraham, A., Hrsg., Berlin, Heidelberg: Springer, 2011, S. 149–185, ISBN: 978-3-642-21004-4. DOI: 10.1007/978-3-642-21004-4_7. besucht am 16. März 2025. Adresse: https://doi.org/10.1007/978-3-642-21004-4_7.

[25] Gruber, T. R., „A translation approach to portable ontology specifications," *Knowledge Acquisition*, Jg. 5, Nr. 2, S. 199–220, Juni 1993, ISSN: 1042-8143. DOI: 10.1006/knac.1993.1008. besucht am 26. Feb. 2025. Adresse: https://www.sciencedirect.com/science/article/pii/S1042814383710083.

[26] Guarino, N., „Ontologies and knowledge bases: towards a terminological clarification,“ in Jan. 1995, S. 25–32.

[27] Gunning, D. und Aha, D. W., „DARPA's Explainable Artificial Intelligence Program,“ en, *AI Magazine*, Jg. 40, Nr. 2, S. 44–58, 2019, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1609/aimag.v40i2.2850, ISSN: 2371-9621. DOI: 10.1609/aimag.v40i2.2850. besucht am 26. Feb. 2025. Adresse: https://onlinelibrary.wiley.com/doi/abs/10.1609/aimag.v40i2.2850.

[28] *Habsburger*, de. besucht am 7. Mai 2025. Adresse: https://www.habsburger.net/de/habsburger.

[29] Hitzler, P., Krötzsch, M., Rudolph, S. und Sure, Y., „Die Idee des Semantic Web,“ de, in *Semantic Web: Grundlagen*, Hitzler, P., Krötzsch, M., Rudolph, S. und Sure, Y., Hrsg., Berlin, Heidelberg: Springer, 2008, S. 7–14, ISBN: 978-3-540-33994-6. DOI: 10.1007/978-3-540-33994-6_2. besucht am 29. März 2025. Adresse: https://doi.org/10.1007/978-3-540-33994-6_2.

[30] Jurafsky, D. und Martin, J. H., *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd. 2025, Online manuscript released January 12, 2025. Adresse: https://web.stanford.edu/~jurafsky/slp3/.

[31] *Large Language Models Market Size — Industry Report, 2030*, en. besucht am 26. Feb. 2025. Adresse: https://www.grandviewresearch.com/industry-analysis/large-language-model-llm-market-report.

[32] Lecu, A., Groza, A. und Hawizy, L., „Using LLMs and ontologies to extract causal relationships from medical abstracts,“ *Procedia Computer Science*, 6th International Conference on AI in Computational Linguistics, Jg. 244, S. 443–452, Jan. 2024, ISSN: 1877-0509. DOI: 10.1016/j.procs.2024.10.219. besucht am 26. Feb. 2025. Adresse: https://www.sciencedirect.com/science/article/pii/S1877050924030205.

[33] Lenat, D. B., „CYC: a large-scale investment in knowledge infrastructure,“ *Commun. ACM*, Jg. 38, Nr. 11, S. 33–38, Nov. 1995, ISSN: 0001-0782. DOI: 10.1145/219717.219745. besucht am 29. März 2025. Adresse: https://dl.acm.org/doi/10.1145/219717.219745.

[34] *Leopold I.* de. besucht am 1. Juni 2025. Adresse: https://www.habsburger.net/de/personen/habsburger-herrscher/leopold-i.

[35] Lewis, P. u. a., „Retrieval-augmented generation for knowledge-intensive NLP tasks,“ in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Ser. NIPS '20, Red Hook, NY, USA: Curran Associates Inc., Dez. 2020, S. 9459–9474, ISBN: 978-1-7138-2954-6. besucht am 27. Feb. 2025.

[36] *Llama*, en. besucht am 19. März 2025. Adresse: https://www.llama.com/.

[37] Lukrez, *Über die Natur der Dinge*. besucht am 27. März 2025. Adresse: https://www.reclam.de/detail/978-3-15-014079-6/Lukrez/Ueber_die_Natur_der_Dinge.

[38] Lundberg, S. M. und Lee, S.-I., „A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, S. 4768–4777, ISBN: 9781510860964.

[39] Lundberg, S. M. und Lee, S.-I., „A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Ser. NIPS'17, Red Hook, NY, USA: Curran Associates Inc., Dez. 2017, S. 4768–4777, ISBN: 978-1-5108-6096-4. besucht am 6. Apr. 2025.

[40] Madan, S., Lentzen, M., Brandt, J., Rueckert, D., Hofmann-Apitius, M. und Fröhlich, H., „Transformer models in biomedicine," *BMC Medical Informatics and Decision Making*, Jg. 24, Nr. 1, S. 214, Juli 2024, ISSN: 1472-6947. DOI: 10.1186/s12911-024-02600-5. besucht am 19. März 2025. Adresse: https://doi.org/10.1186/s12911-024-02600-5.

[41] *Maria Theresia*, de. besucht am 1. Juni 2025. Adresse: https://www.habsburger.net/de/personen/habsburger-herrscher/maria-theresia.

[42] *Marie Antoinette*, de. besucht am 1. Juni 2025. Adresse: https://www.habsburger.net/de/personen/habsburger/marie-antoinette.

[43] Mateiu, P. und Groza, A., „Ontology engineering with Large Language Models," in *2023 25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, ISSN: 2470-881X, Sep. 2023, S. 226–229. DOI: 10.1109/SYNASC61333.2023.00038. besucht am 26. Feb. 2025. Adresse: https://ieeexplore.ieee.org/abstract/document/10522716.

[44] McCarthy, J., „PROGRAMS WITH COMMON SENSE," en, *Stanford University*, 1959. Adresse: http://jmc.stanford.edu/articles/mcc59.html.

[45] Melle, W. van, „MYCIN: a knowledge-based consultation program for infectious disease diagnosis," *International Journal of Man-Machine Studies*, Jg. 10, Nr. 3, S. 313–322, Mai 1978, ISSN: 0020-7373. DOI: 10.1016/S0020-7373(78)80049-2. besucht am 16. März 2025. Adresse: https://www.sciencedirect.com/science/article/pii/S0020737378800492.

[46] Micheletti, N., Belkadi, S., Han, L. und Nenadic, G., *Exploration of Masked and Causal Language Modelling for Text Generation*, arXiv:2405.12630 [cs], Aug. 2024. DOI: 10.48550/arXiv.2405.12630. besucht am 19. März 2025. Adresse: http://arxiv.org/abs/2405.12630.

[47] Miller, T., „Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, Jg. 267, S. 1–38, Feb. 2019, ISSN: 0004-3702. DOI: 10.1016/j.artint.2018.07.007. besucht am 26. Feb. 2025. Adresse: https://www.sciencedirect.com/science/article/pii/S0004370218305988.

[48] mrbullwinkle, *Azure OpenAI Service - Azure OpenAI*, en-us, Okt. 2024. besucht am 26. März 2025. Adresse: https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering.

[49] Narteni, S., Ferretti, M., Orani, V., Vaccari, I., Cambiaso, E. und Mongelli, M., „From Explainable to Reliable Artificial Intelligence," en, in *Machine Learning and Knowledge Extraction*, Holzinger, A., Kieseberg, P., Tjoa, A. M. und Weippl, E., Hrsg., Cham: Springer International Publishing, 2021, S. 255–273, ISBN: 978-3-030-84060-0. DOI: 10.1007/978-3-030-84060-0_17.

[50] Neuhaus, F., „Ontologies in the era of large language models – a perspective," en, *Applied Ontology*, Jg. 18, Nr. 4, S. 399–407, Dez. 2023, Publisher: SAGE Publications, ISSN: 1570-5838. DOI: 10.3233/AO-230072. besucht am 26. Feb. 2025. Adresse: https://journals.sagepub.com/doi/abs/10.3233/AO-230072.

[51] Noy, N., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. und Musen, M., „Creating Semantic Web contents with Protege-2000," *IEEE Intelligent Systems*, Jg. 16, Nr. 2, S. 60–71, 2001. DOI: 10.1109/5254.920601.

[52] OpenAI u. a., *GPT-4 Technical Report*, en, arXiv:2303.08774 [cs], März 2024. DOI: 10.48550/arXiv.2303.08774. besucht am 26. Feb. 2025. Adresse: http://arxiv.org/abs/2303.08774.

[53] *OpenAI Platform*, en. besucht am 27. Feb. 2025. Adresse: https://platform.openai.com.

[54] *Otto Habsburg-Lothringen*, de. besucht am 1. Juni 2025. Adresse: https://www.habsburger.net/de/personen/habsburger/otto-habsburg-lothringen.

[55] *OWL 2 Web Ontology Language Document Overview (Second Edition)*. besucht am 29. März 2025. Adresse: https://www.w3.org/TR/owl2-overview/.

[56] *OWL Web Ontology Language Overview*. besucht am 30. März 2025. Adresse: https://www.w3.org/TR/owl-features/.

[57] Paslaru Bontas Simperl, E., Tempich, C. und Sure, Y., „ONTOCOM: A Cost Estimation Model for Ontology Engineering," en, in *The Semantic Web - ISWC 2006*, Cruz, I. u. a., Hrsg., Berlin, Heidelberg: Springer, 2006, S. 625–639, ISBN: 978-3-540-49055-5. DOI: 10.1007/11926078_45.

[58] *Poldi*. besucht am 1. Juni 2025. Adresse: https://www.baer.de/projekte/alternativer-wolf-und-baerenpark-schwarzwald/tierhimmel/1262-poldi.

[59] *Pricing*, en-US. besucht am 19. März 2025. Adresse: https://openai.com/api/pricing/.

[60] *protégé*. besucht am 27. Feb. 2025. Adresse: https://protege.stanford.edu/.

[61] *Protégé 5 Documentation*. besucht am 2. Apr. 2025. Adresse: https://protegeproject.github.io/protege/.

[62] *RDF 1.1 Concepts and Abstract Syntax*. besucht am 30. März 2025. Adresse: https://www.w3.org/TR/rdf-concepts/.

[63] Rebala, G., Ravi, A. und Churiwala, S., *An Introduction to Machine Learning*, en. Cham: Springer International Publishing, 2019, ISBN: 978-3-030-15728-9 978-3-030-15729-6. DOI: 10.1007/978-3-030-15729-6. besucht am 17. März 2025. Adresse: http://link.springer.com/10.1007/978-3-030-15729-6.

[64] Ribeiro, M. T., Singh, S. und Guestrin, C., „"Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Ser. KDD '16, New York, NY, USA: Association for Computing Machinery, Aug. 2016, S. 1135–1144, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778. besucht am 6. Apr. 2025. Adresse: https://dl.acm.org/doi/10.1145/2939672.2939778.

[65] Riebling, J. R., *Methode und Methodologie quantitativer Textanalyse* (Bamberger Beiträge zur Soziologie Band 18), de. Bamberg: University of Bamberg Press, 2019, ISBN: 978-3-86309-640-3.

[66] Rudin, C., „Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," en, *Nature Machine Intelligence*, Jg. 1, Nr. 5, S. 206–215, Mai 2019, Publisher: Nature Publishing Group, ISSN: 2522-5839. DOI: 10.1038/s42256-019-0048-x. besucht am 15. März 2025. Adresse: https://www.nature.com/articles/s42256-019-0048-x.

[67] *Rudolf I.* de. besucht am 1. Juni 2025. Adresse: https://www.habsburger.net/de/personen/habsburger-herrscher/rudolf-i.

[68] Schenke, M., *Logikkalküle in der Informatik: Wie wird Logik vom Rechner genutzt?* de. Wiesbaden: Springer Fachmedien, 2013, ISBN: 978-3-8348-1887-4 978-3-8348-2295-6. DOI: 10.1007/978-3-8348-2295-6. besucht am 29. März 2025. Adresse: https://link.springer.com/10.1007/978-3-8348-2295-6.

[69] Shumailov, I., Shumaylov, Z., Zhao, Y., Papernot, N., Anderson, R. und Gal, Y., „AI models collapse when trained on recursively generated data," en, *Nature*, Jg. 631, Nr. 8022, S. 755–759, Juli 2024, Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/s41586-024-07566-y. besucht am 10. Sep. 2025. Adresse: https://www.nature.com/articles/s41586-024-07566-y.

[70] Simonyan, K., Vedaldi, A. und Zisserman, A., „Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, Bengio, Y. und LeCun, Y., Hrsg., 2014. Adresse: http://arxiv.org/abs/1312.6034.

[71] Staab, S. und Studer, R., Hrsg., *Handbook on Ontologies*, en. Berlin, Heidelberg: Springer, 2009, ISBN: 978-3-540-70999-2 978-3-540-92673-3. DOI: 10.1007/978-3-540-92673-3. besucht am 29. März 2025. Adresse: http://link.springer.com/10.1007/978-3-540-92673-3.

[72] Szeliski, R., *Computer Vision: Algorithms and Applications* (Texts in Computer Science), en. London: Springer, 2011, ISBN: 978-1-84882-934-3 978-1-84882-935-0. DOI: 10.1007/978-1-84882-935-0. besucht am 17. März 2025. Adresse: https://link.springer.com/10.1007/978-1-84882-935-0.

[73] Tudorache, T., Vendetti, J. und Noy, N. F., „Web-Prote´g´e: A Lightweight OWL Ontology Editor for the Web," en,

[74] Vaswani, A. u. a., „Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Ser. NIPS'17, Red Hook, NY, USA: Curran Associates Inc., Dez. 2017, S. 6000–6010, ISBN: 978-1-5108-6096-4. besucht am 15. März 2025.

[75] *Verlässliche Informationen für Ihre Gesundheit*, de. besucht am 1. Juni 2025. Adresse: https://gesund.bund.de/.

[76] Vig, J., „A Multiscale Visualization of Attention in the Transformer Model," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Costa-jussà, M. R. und Alfonseca, E., Hrsg., Florence, Italy: Association for Computational Linguistics, Juli 2019, S. 37–42. DOI: 10.18653/v1/P19-3007. besucht am 9. Apr. 2025. Adresse: https://aclanthology.org/P19-3007/.

[77] Wachter, S., Mittelstadt, B. und Russell, C., „Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR," en, *SSRN Electronic Journal*, 2017, ISSN: 1556-5068. DOI: 10.2139/ssrn.3063289. besucht am 22. März 2025. Adresse: https://www.ssrn.com/abstract=3063289.

[78] Wang, X. u. a., „Self-Consistency Improves Chain of Thought Reasoning in Language Models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023. Adresse: https://openreview.net/forum?id=1PL1NIMMrw.

[79] Wei, J. u. a., „Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Ser. NIPS '22, Red Hook, NY, USA: Curran Associates Inc., Nov. 2022, S. 24 824–24 837, ISBN: 978-1-7138-7108-8. besucht am 27. Feb. 2025.

[80] Weizenbaum, J., „ELIZA-A Computer Program For the Study of Natural Language Communication Between Man And Machine," *Computational Linguistics*, Jg. 9, Nr. 1, 1966. besucht am 16. März 2025. Adresse: https://cse.buffalo.edu/~rapaport/572/S02/weizenbaum.eliza.1966.pdf.

[81] Wu, Y. u. a., „Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," *CoRR*, Jg. abs/1609.08144, 2016. arXiv: 1609.08144. Adresse: http://arxiv.org/abs/1609.08144.

[82] Wu, Z., Geiger, A., Icard, T., Potts, C. und Goodman, N. D., „Interpretability at scale: identifying causal mechanisms in alpaca," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Ser. NIPS '23, Red Hook, NY, USA: Curran Associates Inc., Dez. 2023, S. 78 205–78 226. besucht am 9. Apr. 2025.