Fernuniversität Hagen

Modul 63458 - Fachpraktikum

Natural Language Processing, Information Extraction und Retrieval

WiSe 2022/23

Praktikumsbericht

Automatische Erstellung einer Wissensrepräsentation aus einem medizinischen Text

eingereicht von

Anne Koch, Clara Jansen, Dietrich Tönnies

Betreuer: Prof. Dr. Hemmje
Dr. Christian Nawroth
Stephanie Heidepriem, M.Sc.

Hagen, 9. März 2023

INHALTSVERZEICHNIS

DCMI	Dublin Core Metadata Initiative
MENHIR	Mental health monitoring through interactive conversations
MeSH	Medical Subject Headings
NER	Named Entity Recognition
NLM	United States National Library of Medicine
NLP	Natural Language Processing
RDF	Resource Description Framework
SPARQL	SPARQL Protocol And RDF Query Language
STOP	STop Obesity Platform
UMLS	Unified Medical Language System
W3C	World Wide Web Consortium

1 EINLEITUNG

Diese Praktikumsarbeit behandelt die automatische Erstellung einer Wissensrepräsentation aus einem medizinischen Text. Eine prototypische Softwarelösung wird entwickelt, die für einen vorgegebenen Beispieltext einen Wissensgraphen ermittelt.

Gemäß einem Artikel des *Economist* von 2017 [**the_economist_worlds_2017**], der in Verbindung mit den aktuellen technischen Entwicklungen, Künstlicher Intelligenz und Data Science viel zitiert wird, ist nicht mehr Öl die wertvollste Ressource, sondern Daten. Ebenso wie Öl müssen die Daten jedoch erst aufbereitet werden, um tatsächlich von Nutzen zu sein. Auch Informationen, die in Form von für den Menschen lesbaren Texten zur Verfügung stehen, sind nicht direkt für Computer auswertbar. Daher wurden mit den Methoden des *Natural Language Processing (NLP)* Verfahren entwickelt, um automatisch Informationen aus Texten extrahieren zu können. Die allgemeinsprachlichen NLP-Verfahren bedürfen jedoch noch einer Spezialisierung, um auch für Fachtexte wie zum Beispiel aus dem Bereich der Medizin nutzbringend angewendet werden zu können.

Aus diesen Rahmenbedingungen ergibt sich die Motivation und die Problembeschreibung für diese Praktikumsaufgabe, die in den beiden nachfolgenden Abschnitten beschrieben werden.

1.1 MOTIVATION

Die spezielle Herausforderung der Praktikumsaufgabe liegt darin, automatisiert Zusammenhänge zwischen Entitäten medizinischer Texte zu finden und diese Zusammenhänge zu repräsentieren. Dem zugrunde liegt das Ziel des Dissertationsprojektes von Stephanie Heidepriem, zu erforschen, wie ein textbasierter Chatbot zur Unterstützung psychisch kranker Menschen entwickelt werden kann.

Das Dissertationsprojekt ist unter anderem an das Projekt *Mental health monitoring through interactive conversations (MENHIR)* angelehnt [**noauthor_menhir_2022**]. Dieses Projekt dient der Erforschung von Konversationstechnologien, die psychisch kranke Menschen unterstützen sollen. In diesem Zusammenhang wird auch ein MENHIR-Chatbot entwickelt, der personalisierte Unterstützung und hilfreiche Bewältigungsstrategien bieten soll.

Das Forschungsprojekt *STop Obesity Platform (STOP)* beschäftigt sich mit der Extraktion von Wissen unter anderem aus Chatbots, das anschließend aufbereitet und mit weiteren Informationen kombiniert werden soll. Dieses Wissen soll medizinischem Fachpersonal zur Verfügung gestellt werden und außerdem Menschen mit Adipositas dabei helfen, eine gesündere Ernährung einzuhalten [**noauthor_stop_2022**].

Es ist aber auch denkbar, dass die Problemstellung der Praktikumsaufgabe auch darüber hinaus für weitere Anwendungen interessant ist und mögliche Ergebnisse in verschiedenen Gebieten eingesetzt werden können, in denen Informationen in (medizinischen) Texten zueinander in Zusammenhang gebracht werden sollen.

1.2 PROBLEMBESCHREIBUNG

Es soll eine Konsolenapplikationen entwickelt werden, die englischsprachige medizinische Texte analysiert und das darin enthaltene Wissen strukturiert in einem RDF-Graphen hinterlegt. Die Konsolenapplikation soll in Python programmiert werden, wobei für die Textanalyse Klassen und Methoden der Open-Source-Bibliothek *spaCy* o.ä. genutzt werden sollen.

Die medizinischen Texte enthalten z.B. Aussagen zu Krankheiten (z.B. Depression) und zählen deren Symptome (z.B. Motivationsverlust) auf. Aufgabe der Konsolenapplikation ist es, Schlüsselbegriffe im Text zu finden und miteinander in Bezug zu setzen, indem z.B. Symptome den ihnen zugrunde liegenden Krankheiten zugeordnet werden. Die von *spaCy* oder anderen NLP-Bibliotheken zur Verfügung gestellte Funktionalität besteht aus einer Pipeline von Analyse-Komponenten, die nacheinander auf den zu analysierenden Text angewendet werden. Diese Komponenten dienen dazu, Texte in einzelne Sätze zu zerlegen und die Sätze grammatikalisch zu analysieren.

PROBLEM 1: VORVERARBEITUNG EINES TEXTES Medizinische und wissenschaftliche Texte bestehen häufig aus Aufzählungen und Zwischenüberschriften. Der Text sollte so aufbereitet werden, dass anschließende Analyse-Algorithmen möglichst erfolgreich arbeiten können. Es soll erprobt werden, wie dies am besten bewerkstelligt werden kann, z.B. durch Umwandlung von Aufzählungen in mehrere Aussagesätze. Auch sollte automatisch erkannt werden, welche Sätze Aussagen (Wissen) formulieren etwa durch Unterscheidung von Indikativ und Konjunktiv oder durch Ignorieren von Fragesätzen.

PROBLEM 2: TRAINING DER NER - KOMPONENTE Neben der grammatikalischen Analyse besteht eine wesentliche Aufgabe von *spaCy* oder anderen NLP-Bibliotheken darin, Schlüsselbegriffe zu finden und nach Möglichkeit zu kategorisieren. Diese Art der Analyse wird als *Named Entity Recognition (NER)* bezeichnet. Bei *spaCy* übernimmt diese Aufgabe der regelbasierte *Entity-Ruler* oder der auf statistischen Modellen basierende *Entity-Recognizer*. Eine weitere Komponente ist der *Entity-Linker*, mit dem Begriffe eindeutig den in einer Wissensbasis gespeicherten Entitäten zugeordnet werden können. Auch der *Entity-Linker* basiert auf statistischen Modellen und wird mit Beispielsätzen trainiert.

Die Standard-NER-Funktionalität von *spaCy* erkennt medizinische Begriffe nur unzureichend. Es ist daher notwendig, eine Datenbank mit

medizinischen Begriffen und Kategorien zusammenzustellen, die für das Training der NER-Komponente verwendet werden kann.

Die *United States National Library of Medicine (NLM)* stellt mit dem *Unified Medical Language System (UMLS)* ein mächtiges Werkzeug für die Textanalyse zur Verfügung. Teil von UMLS ist der sogenannte Metathesaurus, der aus einer Vielzahl von Thesauri unterschiedlicher Organisationen zusammengestellt wird. Zu diesen Thesauri gehört u.a. der von der NLM entwickelte Thesaurus *Medical Subject Headings (MeSH)*. MetaMap und das weniger umfangreiche MetaMapLite sind eigene *Entity-Recognition*-Werkzeuge der *National Library of Medicine*. Die zugrundeliegende Datenbasis dieser Werkzeuge sollen im Rahmen dieses Praktikums dafür verwendet werden, die *Named Entity Recognition*-Komponenten von *spaCy* zu trainieren. Aus den von der NLM zur Verfügung gestellten Begriffslisten soll eine geeignete Auswahl erfolgen, die für das Training der NER-Komponente verwendet werden kann.

PROBLEM 3: ZUORDNUNG DER ENTITÄTEN Basierend auf den gefundenen Entitäten soll das Python-Programm in der Lage sein, Begriffe wie Krankheiten und Symptome richtig zuzuordnen. Hierzu muss die Struktur von Aussagesätze, Fragen, Aufzählungen und Überschriften analysiert werden und so aufbereitet werden, dass eine automatische Analyse durch NER und *Entity-Linker* möglichst erfolgreich ist. Die Ausgabe der Konsolenapplikation erfolgt im RDF/XML-Format.

1.3 FORSCHUNGSFRAGEN

Dieser Abschnitt beschreibt die Forschungsfragen, die sich aus dem in Abschnitt 1.2 angeführten Problemen ableiten.

FF 1 Wie kann ein medizinischer Text vorverarbeitet werden, so dass relevante Aussagen (Überschriften, Aufzählungen, Leerzeilen und Dialoge) automatisch erkannt werden?

FF 2 Wie kann ein medizinisches Fachvokabular über depressive Erkrankungen automatisiert in eine maschinenlesbare Form überführt werden?

FF 3 Wie kann eine Wissensrepräsentation über einen gegebenen Text maschinenlesbar erstellt werden?

1.4 FORSCHUNGSMETHODE

In dieser Arbeit kommt die in der Forschung zu Informationssystemen etablierte Methode von Nunamaker, Chen und Purdin [**nunamaker_systems_1990-1**] zum Einsatz, die einen strukturierten Rahmen zur Durchführung von Forschungsarbeiten bereitstellt.

Die Forschungsmethode umfasst die folgenden in Abbildung 1.1 dargestellten Phasen:

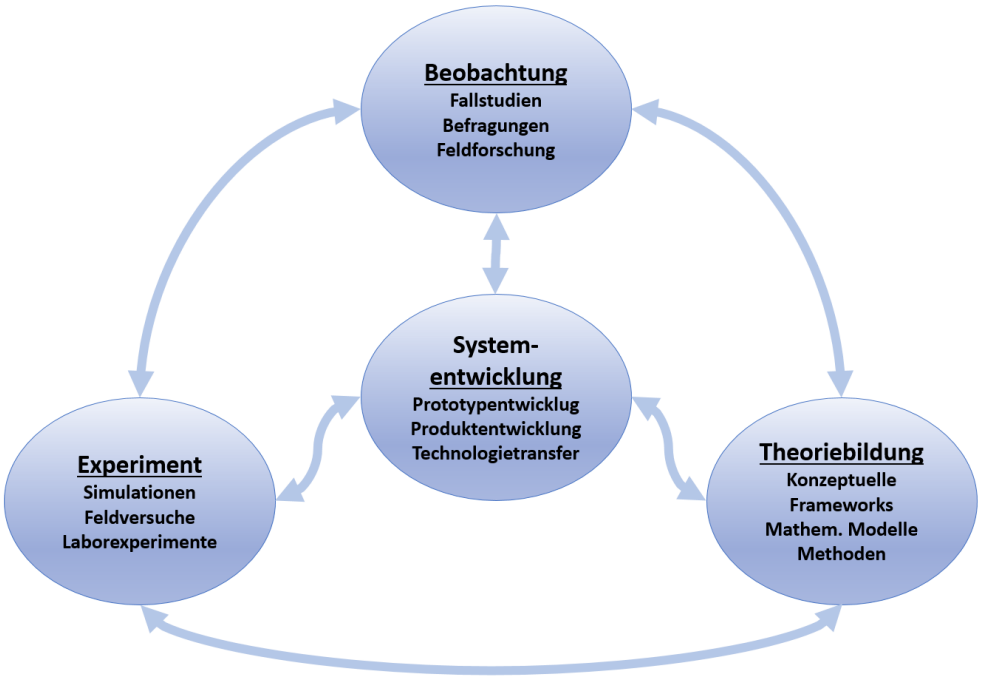

Abbildung 1.1: Forschungsmethode nach [**nunamaker_systems_1990-1**]

BEOBACHTUNG: Insbesondere wenn der Untersuchungsgegenstand relativ unbekannt ist, werden Fallstudien, Feldversuche oder Umfragen durchgeführt, um ein „Gefühl" für den Aufwand zu erhalten. Auf dieser Grundlage können dann konkrete Hypothesen erstellt werden, die durch Experimente geprüft werden.

In dieser Arbeit findet aufgrund der Art des Untersuchungsgegenstandes in der Beobachtungsphase hauptsächlich die Literaturrecherche und Ermittlung des Standes von Wissenschaft und Technik statt.

THEORIEBILDUNG: In dieser Phase werden neue Ideen, Konzepte Methoden oder Modelle entwickelt. Diese Theorien beschreiben das Systemverhalten allgemein, haben jedoch wenig praktische Bedeutung für die Zieldomäne. Sie können aber genutzt werden, um Forschungshypothesen aufzustellen, die Planung von Experimenten zu unterstützen und systematische Beobachtungen durchzuführen.

SYSTEMENTWICKLUNG: Diese Phase besteht aus fünf Teilen:

- Konzeptentwurf
- Erstellung der Systemarchitektur
- Erstellung von Prototypen
- Produktentwicklung
- Technologietransfer

EXPERIMENT: In dieser Phase werden die gefundenen Theorien und entwickelten Systeme evaluiert. Die Ergebnisse der Experimente können

genutzt werden, um die Theorien weiterzuentwickeln und die Systeme zu verbessern.

Obwohl die Phasen in der Methode keine vorgegebene Reihenfolge haben, sondern sich gegenseitig beeinflussen, wird in der vorliegenden Arbeit die oben angeführte Abfolge verwendet.

1.5 FORSCHUNGSZIELE

Dieser Abschnitt beschreibt die Forschungsziele, die sich aus den in Abschnitt 1.3 angeführten Problemen ableiten.

FZ 1 Wie kann ein medizinischer Text vorverarbeitet werden, so dass relevante Aussagen (Überschriften, Aufzählungen, Leerzeilen und Dialoge) automatisch erkannt werden?

- FZ 1.1 Aufbau und Struktur medizinischer Texte (Observation)
- FZ 1.2 Theoriebildung zur Vorverarbeitung medizinischer Texte
- FZ 1.3 Implementierung zur Vorverarbeitung medizinischer Texte
- FZ 1.4 Evaluation zur Vorverarbeitung medizinischer Texte

FZ 2 Wie kann ein medizinisches Fachvokabular über depressive Erkrankungen automatisiert in eine maschinenlesbare Form überführt werden?

- FZ 2.1 Überblick über depressive Erkrankungen (Observation)
- FZ 2.2 Automatisierung durch NLP (Observation)
- FZ 2.3 Theoriebildung zur Überführung medizinischen Fachvokabulars in maschinenlesbare Form zur weiteren Verarbeitung durch NLP
- FZ 2.4 Implementierung zur Überführung medizinischen Fachvokabulars in maschinenlesbare Form zur weiteren Verarbeitung durch NLP
- FZ 2.5 Evaluation zur Überführung medizinischen Fachvokabulars in maschinenlesbare Form zur weiteren Verarbeitung durch NLP

FZ 3 Wie kann eine Wissensrepräsentation über einen gegebenen Text maschinenlesbar erstellt werden?

- FZ 3.1 Wissensrepräsentation mittels RDF (Observation)
- FZ 3.2 Theoriebildung zur Wissensrepräsentation
- FZ 3.3 Implementierung zur Wissensrepräsentation
- FZ 3.4 Evaluation zur Wissensrepräsentation

1.6 LÖSUNGSANSATZ

Die Wissensrepräsentation medizinischer Texte soll Entitäten unterschiedlicher Kategorie miteinander in Beziehung setzen. Naheliegend ist es z.B., automatisch in einem Text Symptome und Krankheiten zu identifizieren, diese miteinander in Beziehung zu setzen und eine Wissensrepräsentation der Art

loss of appetite - is a symptom of - depression

zu erzeugen. Dazu muss spaCy befähigt werden, nicht nur Fachbegriffe zu erkennen, sondern diese auch richtig zu kategorisieren, als z.B. Krankheit oder Symptom. Folgender Lösungsansatz ist angedacht:

MetaMapLite verwendet eine auf UMLS basierende Datenbasis mit medizinischen Fachbegriffen. Der Umfang der Datenbasis von MetaMapLite ist etwas reduziert und umfasst z.B. nur englische Fachbegriffe. Auf der MetaMapLite-Website kann das zip-Archiv

public_mm_data_lite_usabase_2022aa.zip

heruntergeladen werden. In diesem Archiv befindet sich die Datei *postings* in dem Unterordner

\public_mm_lite\data\ivf \2022AA\USAbase\indices\cuisourceinfo.

Die Datei *postings* enthält ca. 11 Millionen englischsprachige Einträge und ist mit einer Größe von etwa 739 MB etwas handlicher als die Datenbestände von UMLS. Um zu prüfen, ob diese Datenbasis geeignet ist, ist ein Auszug von Einträgen, die für den zu analysierenden Beispieltext über Depressionen relevant sind, in der folgenden Tabelle aufgelistet.

CUI bezeichnet den Concept Unique Identifier, mit dem Synonyme gefunden werden können. Der CUI besteht nur aus dem mit dem Buchstaben 'C' beginnenden Teil und ist hier ergänzt durch einen sogenannten Term-Type. Der Term-Type 'PT' kennzeichnet z.B. bevorzugte Einträge. SUI bezeichnet den String Unique Identifier, mit dem gleichlautende (und damit redundante) Bezeichnungen gefunden werden können.

Die Tabelle listet nur Einträge auf, in denen der Begriff, z.B. 'depression', einer in Klammern hinter dem Begriff stehenden Kategorie zugeordnet ist, hier z.B. 'disease'. Über den CUI können in der Datei *postings* Synonyme gefunden werden, wobei nur der mit 'C' beginnende Teil relevant ist. Diese Synonyme enthalten häufig keine Kategorien in Klammern. Die Kategorien können aber über den CUI den Synonymen zugeordnet werden. Die sehr zahlreichen Synonyme bzw. alternativen Ausdrücke sind in der Tabelle nicht aufgeführt.

CUI	SUI	Item	Source
FNC0232933	S3225525	Abnormal menstrual cycle (finding)	SNOMEDCT_US
PTC0424569	S3221759	Circumstances interfere with sleep (disorder)	SNOMEDCT_US
YC0009806	S3235964	Constipation (finding)	SNOMEDCT_US
LAC3845528	S14560529	Depressed mood (e.g., feeling sad, tearful)	LNC
SYC0344315	S3252744	Depressed mood (finding)	SNOMEDCT_US
GTC0011570	S1431189	depression (disease)	AOD
FNC2939186	S3264511	Disturbance in mood (finding)	SNOMEDCT_US
FNC1288289	S3312713	Fearful mood (finding)	SNOMEDCT_US
FNC0150041	S3313279	Feeling hopeless (finding)	SNOMEDCT_US
FNC0022107	S3313282	Feeling irritable (finding)	SNOMEDCT_US
FNC0424000	S3313310	Feeling suicidal (finding)	SNOMEDCT_US
ETC0917801	S3373158	Insomnia (disorder)	SNOMEDCT_US
PTC0015672	S3386372	Lack of energy (finding)	SNOMEDCT_US
FNC2981158	S3386389	Lack of libido (finding)	SNOMEDCT_US
FNC1971624	S3397609	Loss of appetite (finding)	SNOMEDCT_US
FNC0178417	S3397620	Loss of capacity for enjoyment (finding)	SNOMEDCT_US
FNC0456814	S3397668	Loss of motivation (finding)	SNOMEDCT_US
FNC0679136	S3398077	Low self-esteem (finding)	SNOMEDCT_US
PTC5444612	S20749480	mood (physical finding)	MTH
FNC2945580	S3485453	Poor self-esteem (finding)	SNOMEDCT_US
FNC0235160	S3513783	Restless sleep (finding)	SNOMEDCT_US
PTC0233481	S3620195	Worried (finding)	SNOMEDCT_US

In der Tabelle finden sich drei Kategorien, ‘disease’, ‘disorder’ und ‘finding’ (Die *postings*-Datei enthält noch weitere wie z.B. ‘situation’ oder ‘procedure’). Hier bietet sich an, die Begriffe und Kategorien dieser Datenbasis für das Trainieren des Entity-Recognizers oder Entity-Rulers von spaCy zu verwenden. Die Zuordnung ‘finding’ bedeutet eigentlich Befund, kann hier aber auch als Symptom verstanden werden. ‘disorder’, also Störung, kennzeichnet in den meisten Fällen Krankheiten. Der Begriff ‘insomnia’ findet sich nicht im zu analysierenden Text, dort ist stattdessen von ‘disturbed sleep’ die Rede ist. Hier muss versucht werden, über Synonyme eine richtige Kategorisierung zu erreichen. Die meisten Einträge stammen von der Datenbank SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms), die seit 2003 Teil des UMLS Metathesaurus ist.

Der Inhalt der Datei *postings* muss aufbereitet (und ggf. auch gekürzt) werden, so dass die Daten für das Training des Entity-Recognizers verwendet

werden können. Es muss untersucht werden, wie basierend auf der Wörterliste geeignete Trainingsdaten erzeugt werden können, etwa durch automatisch erzeugte Beispielsätze. Alternativ kann der Entity-Ruler zum Einsatz kommen, dem die Wörterliste einfach übergeben werden kann und dessen Funktion vorhersagbarer ist als die des Entity-Recognizers.

Der zu analysierende Text muss durch ein Python-Programm aufbereitet werden, mit dem Ziel, dass die NLP-Komponenten der Prozesspipeline des NLP-Frameworks, möglichst erfolgreich arbeiten. Ein Vorgehen könnte beispielsweise sein, Aufzählungen in mehrere vollständige Sätze zu zerlegen, so dass möglichst sinnvolle vollständige Sätze entstehen, die eine Krankheit und ein Symptom enthalten, so dass die automatische Textanalyse nicht überfordert wird, Krankheit und Symptome zusammenzubringen. Hierbei werden die von *spaCy* zur Verfügung gestellten Pipeline-Komponenten wie der Lemmatizer und Parser genutzt.

Es ist zu prüfen, inwieweit der *Dependency Parser* von *spaCy* genutzt werden kann, um Abhängigkeiten und Subjekt-Prädikat-Objekt-Beziehungen in Sätzen zu erkennen, die für die Erstellung des RDF-Graphen genutzt werden können.

Idealerweise entsteht jedoch bereits durch die NER ein Text mit Aussagesätzen, die jeweils eine Krankheit und ein oder mehrere Symptome enthalten. Für die Wissensrepräsentation soll der Entity-Linker eingesetzt werden. Normalerweise wird der Entity-Linker verwendet, um (mehrdeutige) Entitäten, die vom Entity-Recognizer oder Entity-Ruler gefunden werden, einer eindeutigen Entität zuzuordnen (z.B. einem Wikipedia-Eintrag). Der Entity-Linker wird trainiert, so dass die Zuordnung kontextbasiert erfolgt.

Für die Erstellung der Wissensrepräsentation kann der Entity-Linker auf unorthodoxe Weise verwendet und mit dieser Komponente die Wissensrepräsentation aufgebaut werden. Wesentlich ist hierbei die vorhandene Datenstruktur der Wissensbasis des Entity-Linkers. In dieser werden durch das Python-Programm als zusammengehörend erkannte Krankheiten und Symptome gespeichert. Dabei können Symptome mehreren Krankheiten zugeordnet werden. Die nach Analyse eines Textes in der Wissensbasis gespeicherten Daten werden dann als XML-Datei oder RDF-Modell ausgegeben. Alternativ zum Entity-Linker kann eine eigene Datenstruktur zum Zwischenspeichern der Wissensrepräsentation genutzt werden.

Ein Vorteil des Entity-Linkers ist, dass er mit Sätzen aus den zu analysierenden Texten trainiert werden kann. Eine kontinuierlich wachsende Wissensbasis (durch zahlreiche automatische Textanalysen) vorausgesetzt, erlaubt dem Entity-Linker, im Laufe der Zeit in beliebigen Texten kontextbasiert einem Symptom die richtige Krankheit zuzuordnen. Analysiert man einen Text dann mit dem Entity-Recognizer und danach mit dem Entity-Linker, dann entsteht automatisch die Wissensrepräsentation, wobei etwa die Entität ‘lack of energy’ vom Entity-Recognizer gefunden wird und als ‘Symptom’ kategorisiert wird und anschließend vom Entity-Linker der Krankheit ‘depression’ zugeordnet wird, sofern sich dies aus dem Kontext ergibt.

Arbeitspakete:

- Manuelle Erstellung einer Wissensrepräsentation basierend auf dem zu analysierenden Text.
- Identifizierung eines medizinischen Vokabulars, das zum Training der NER-Komponente verwendet werden kann.
- Untersuchung, auf welche Weise die Trainingsdaten für möglichst gute Ergebnisse der NER-Komponente aufbereitet werden müssen (z.B. durch automatisch erzeugte Beispielsätze). Alternativ Verwendung des Entity-Rulers.
- Entwicklung einer Programm-Komponente, die die zu analysierenden Texte vorverarbeitet, so dass die NLP-Pipeline möglichst effizient arbeitet.
- Entwicklung einer Programm-Komponente, die basierend auf den vom NER gefundenen Entitäten zusammengehörende Entitäten (etwa Symptom und Krankheit) identifiziert und eine Wissensbasis aufbaut, ggf. unter Ausnutzung der Wissensbasis des Entity-Linkers.
- Entwicklung einer Programm-Komponente, die die Wissensbasis als RDF/XML-Datei ausgibt.

2 STAND DER WISSENSCHAFT

2.1 FZ 1.1. AUFBAU UND STRUKTUR MEDIZINISCHER TEXTE

Medizinische Texte existieren in verschiedenen Formen mit unterschiedlichen Zielsetzungen. Beispielsweise gibt es Lehrbuch- und Enzyklopädietexte, die überblicksweise über medizinische Sachverhalte informieren, wissenschaftliche Studien, in denen neue wissenschaftliche Erkenntnisse vorgestellt werden und außerdem Arztberichte, in denen über individuelle Patienten und deren Krankheitsverlauf informiert wird. Die für diese Arbeit genutzten Analysetexte stellen enzyklopädische Texte dar, in denen insbesondere Symptome und Merkmale psychologischer Erkrankungen beschrieben werden. Charakterisiert sind diese Texte dadurch, dass sie durch Zwischenüberschriften gegliedert sind und neben Fließtext auch Aufzählungen enthalten.

2.2 FZ 2.1. ÜBERBLICK ÜBER DEPRESSIVE ERKRANKUNGEN

Als Quellen für diesen Abschnitt dienten [**mpg_depression**], [**who_depression**] und [**psychrembel_depression**].

2.2.1 *Allgemein*

Depression ist eine psychische Krankheit, die geschätzt 3,8% der Weltbevölkerung und 5% der Erwachsenen betrifft. Sie kann dazu führen, dass betroffene Menschen Schwierigkeiten haben im Arbeits- und Familienleben zurecht zu kommen.

2.2.2 *Symptome*

Übliche Symptome eine Depression sind eine traurige Grundstimmung, Konzentrationsprobleme, Hoffnungslosigkeit, Müdigkeit und Reizbarkeit. Auch Schlafstörungen, eine Libidostörung, verminderter oder gesteigerter Appetit und ein vermindertes Selbstwertgefühl oder Selbstbewusstsein sind Symptome. Im schlimmsten Fall treten Selbsttötungsgedanken auf.

2.2.3 *Formen der Depression*

Die Schwere einer typischen Depression wird durch leichte, mittelgradige und schwere Episoden unterschieden. Dabei sind die Übergänge fließend. Besondere Formen der Depression sind die chronische Depression, bei der trotz therapeutischer Maßnahmen nur wenig Besserung eintritt. Des Weite-

ren gibt es die manisch-depressive Depression, bei der sich depressive und manische Phasen abwechseln. Außerdem gibt es auch kurze, akute depressive Verstimmungen, die nur zwischen einem Tag und zwei Wochen dauern.

2.2.4 *Ursachen*

Depressionen entstehen durch ein komplexes Zusammenspiel von sozialen, psychologischen und biologischen Faktoren. Dabei wird angenommen, dass für eine typische Depression die Genetik 50% der Ursachen ausmacht. Konkret können Kindheitserfahrungen, Verluste und Arbeitslosigkeit eine Depression begünstigen.

2.2.5 *Behandlung*

Je nach Schweregrad der Krankheit werden zur Behandlung von Depressionen psychologische Behandlungen angewandt, und/oder den betroffenen Personen Antidepressiva verschrieben.

2.3 FZ 2.2. AUTOMATISIERUNG DURCH NLP

2.3.1 *Natural Language Processing*

Natural Language Processing (NLP) ist ein Teil der Künstlichen Intelligenz. Er befasst sich mit der Aufgabe Maschinen den Umgang mit der menschlichen Sprache beizubringen, also das Verstehen der Sprache und dem richtigen Antworten darauf. Mithilfe von NLP-Methoden werden zum Beispiel Sätze analysiert, die Bedeutung von Texten erfasst, Chatbots erstellt und sogar ganze Geschichten geschrieben. Die grundlegendsten Schritte, die beim Erarbeiten eines NLP-Modells (oder allgemeiner eines Machine-Learning-Modells) ausgeführt werden, sind folgende:

1. Aufbereitung von Daten
2. Wählen eines geeigneten Algorithmus
3. Trainieren des Modells mit Trainingsdaten
4. Testen des Modells mit Testdaten
5. Modell anwenden / Vorhersagen treffen

2.3.2 *Named Entity Recognition*

Named Entity Recognition (NER) bezeichnet einen Teilbereich des Natural Language Processing, bei dem es darum geht wichtige Entitäten, wie zum Beispiel Personen, Orte oder Institutionen, in einem Text zu erkennen. Methoden zur Bewältigung dieser Aufgabe werden seit circa 30 Jahren entwickelt.

Darunter fallen grammatikbasierte und statistische Methoden, sowie auch Methoden des Machine Learning.

2.3.3 *Entity Linking*

Als Entity Linking wird im Bereich des NLP die Aufgabe beschrieben, die den Entitäten (z.B. Personen, Orte) in einem Text das korrekte Äquivalent in einer Wissensbasis zuordnen soll. In [**shen_entity_2021**] wird Entity Linking folgendermaßen definiert (übersetzt):

Definition 2.3.1 (Entity Linking) *Gegeben sei ein Dokument D, welches die erkannten Entitäten $M = \{m_1, m_2, \ldots, m_{|M|}\}$ enthält, sowie eine Ziel-Wissensbasis KB, welches die Entitäten $E = \{e_1, e_2, \ldots, e_{|M|}\}$ enthält. Das Ziel ist es jede Entität m_i in M seinem korrekten Äquivalent e_i in E zuzuordnen.*

2.3.4 *spaCy*

spaCy ist eine Python-Bibliothek, die die erforderlichen Daten und Algorithmen enthält, die zum verarbeiten von Texten mit natürlicher Sprache benötigt werden. SpaCy enthält vortrainierte Modelle für über 70 verschiedene Sprachen, unter anderem Spanisch, Englisch, Griechisch und Deutsch. Zudem ermöglicht spaCy das Einbinden von Modellen, die mithilfe anderer Python-Bibliotheken (zum Beispiel PyTorch oder TensorFlow) trainiert wurden. Anders als einige andere NLP-Bibliotheken konzentriert sich spaCy darauf, Software für den Produktionseinsatz zur Verfügung zu stellen. Erstmals wurde spaCy 2015 von Matthew Honnibal veröffentlicht. ([**vasiliev2020natural**], [**github_spacy**], [**spacy**])

2.3.5 *pySBD*

[**sadvilkar_pysbd_2020**] stellen die Python-Bibliothek *pySBD* vor, die Satzgrenzen in Texten ermittelt. Dieser Schritt der Datenaufbereitung ist wichtig, da viele weiterführende NLP-Schritte auf Satzebene durchgeführt werden und daher die korrekte Bestimmung der Satzgrenzen Auswirkungen auf den Erfolg der weiteren Verarbeitung hat. Die Autoren der Studie vergleichen die Genauigkeit ihrer Bibliothek mit anderen Bibliotheken anhand der sogenannten *Golden Rules*, die zahlreiche Beispiele für Sonderfälle von Satzgrenzen in mehreren Sprachen enthalten. Laut ihren Angaben übertrifft die Genauigkeit der Satzgrenzenerkennung von pySBD die Genauigkeit der Satzgrenzenerkennung von spaCy und anderen Bibliotheken.

2.3.6 *Weitere Python-Bibliotheken für Machine Learning (ML) und NER*

Eine der wichtigsten Python-Bibliotheken für NLP ist NLTK (Natural Language Toolkit) ([**bird2006nltk**]). Weitere Python-Bibliotheken, die für NLP-

Aufgaben genutzt werden können, sind unter Anderem Gensim ([**vrehuuvrek2011gensim**]), Pattern ([**de2012pattern**], [**github_pattern**]), scikit-learn und PyTorch.

2.4 FZ 3.1. WISSENSREPRÄSENTATION MITTELS RDF

Das *Resource Description Framework (RDF)* [**w3c_all_2022**] ist ein Framework zur Darstellung von Informationen im Semantischen Web, das von der *RDF Working Group* des *World Wide Web Consortium (W3C)* erstellt wurde. Das RDF-Modell besteht aus einem Datenmodell, mit dem Aussagen über Ressourcen in Form eines Graphen dargestellt werden. Die Informationen werden als Tripel von Subjekt, Prädikat und Objekt gespeichert und ermöglichen auf diese Weise eine maschinenlesbare Bereitstellung semantischer Informationen.

Die Subjekte und Objekte sind dabei die Knoten des Graphen und die Prädikate die Kanten zwischen diesen Knoten.

Abbildung 2.1 zeigt einen einfachen RDF-Graphen, in dem das Subjekt „depression“ mit dem Prädikat „hasSymptom“ mit zwei Symptomen verbunden und über das Prädikat „isA“ mit dem Objekt „mentalDisorder“ verbunden ist.

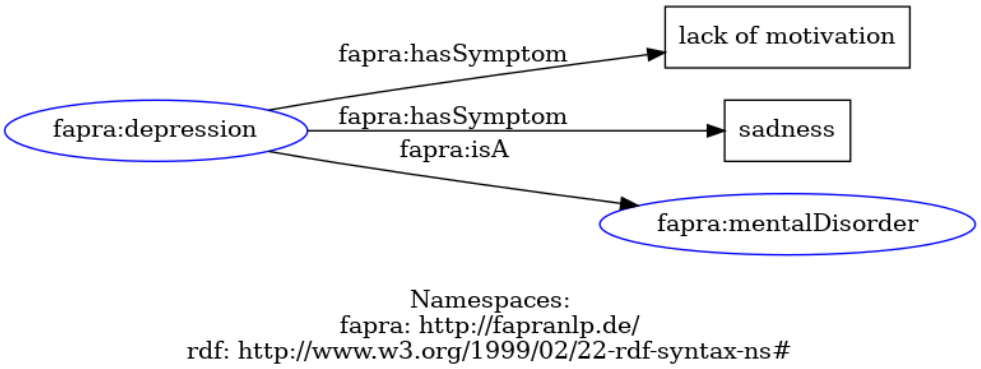

Abbildung 2.1: einfacher RDF-Graph (visualisiert mit `https://www.ldf.fi/service/rdf-grapher`)

2.4.1 *Resource Description Framework Schema*

Zusätzlich zu RDF stellt das W3C auch eine Empfehlung für ein standardisiertes Datenmodellierungsvokabular für RDF-Daten bereit: das RDF-Schema. Dies stellt eine Erweiterung des grundlegenden RDF-Vokabulars dar und bietet eine Reihe von vordefinierten Klassen (*rdfs:Class*), Ressourcen (*rdfs:Class*) und Eigenschaften (*rdf:Property*), die zur Definition von Klassen und Eigenschaften eines RDF-Modells verwendet werden.

Klassen können in einer Hierarchie organisiert werden, wobei Unterklassen Merkmale von ihren Oberklassen erben. Eigenschaften können als zu einer bestimmten Domäne (*rdfs:domain*) und einem bestimmten Bereich (*rdfs:range*) gehörend definiert werden, womit sich präzise Beziehungen zwischen Ressourcen erstellen lassen.

2.4.2 *Dublin Core Metadata Initiative*

Die Dublin Core Metadata Initiative (DCMI)**[DCMI_dcmi_2022]** ist ein Konsortium von Organisationen, die Standards für die Beschreibung von Online-Ressourcen entwickeln und pflegen. Die Metadatenelemente des *Dublin Core* umfassen 15 standardisierte Kernfelder, die zur Beschreibung der Merkmale von Online-Ressourcen dienen.

Die Dublin-Core-Metadatenelemente sind leicht zu verwenden, aber dennoch flexibel genug, um zahlreiche Ressourcentypen und -kontexte beschreiben zu können. Die Elemente umfassen grundlegende beschreibende Informationen wie Titel, Ersteller und Thema sowie speziellere Elemente zur Identifizierung der Art der Ressource, ihrer Sprache und ihrer Beziehung zu anderen Ressourcen. Außerdem umfasst das DCMI auch mehrere Codierschemata für Wortschätze bestimmter Anwendungsgebiete, unter anderem für MeSH.

2.4.3 *Serialisierung von RDF-Graphen*

Für die Serialisierung von RDF-Graphen stehen mehrere Formate zur Verfügung. So wird der MeSH-Datensatz im *N-Triples*-Format bereitgestellt und in diesem Praktikum erfolgt die Ausgabe der Wissensrepräsentation im RDF/XML-Format.

Ein Beispiel für die Serialisierung des in Abbildung 2.1 dargestellten Graphen im RDF/XML-Format ist in Listing 2.1 angeführt.

Listing 2.1: RDF/XML

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:fapra="http://fapranlp.de/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="http://fapranlp.de/depression">
    <fapra:hasSymptom>lack of motivation</fapra:hasSymptom>
    <fapra:hasSymptom>sadness</fapra:hasSymptom>
    <fapra:isA rdf:resource="http://fapranlp.de/mentalDisorder"/>
  </rdf:Description>
</rdf:RDF>
```

Für Python steht mit RDFLib **[rdflib_team_rdflib_2022]** eine Bibliothek zur Arbeit mit RDF-Graphen bereit.

2.4.4 *SPARQL*

SPARQL Protocol And RDF Query Language (SPARQL) ist eine Abfragesprache für das semantische Web, mit der Daten, die im RDF-Format vorliegen, abgerufen und bearbeitet werden können. Sie ermöglicht die Abfrage mehre-

rer Datenquellen und -typen und enthält integrierte Funktionen und Operatoren zur Datentransformation. SPARQL kann auch zum Aktualisieren und Erstellen von RDF-Daten verwendet werden und enthält Befehle für Datenänderungen und Integritätsbedingungen.

3

KONZEPTUELLE MODELLIERUNG UND ENTWURF

In diesem Kapitel wird der Anwendungskontext und konkrete Anwendungsfälle nach UCSD sowie Modelle der technischen Umsetzung einer Automatisierungsunterstützung (Informationsmodell, Komponenten-/Dienstemodelle, Architekturmodell) entsprechend RUP modelliert.

3.1 USER CENTERED SYSTEM DESIGN

Der Anwendungskontext und die Anwendungsfälle werden mithilfe des Ansatzes des User Centered System Design nach [**norman1986user**] ermittelt. Auf der Grundlage der Anwendungsfälle wird das Informations- und Datenmodell entwickelt und daraus das Komponenten- und Architekturmodell.

Gemäß Aufgabenstellung umfasst die Entwicklung keine graphische Benutzeroberfläche.

3.1.1 *Anwendungskontext und Anwendungsfall*

Der Anwendungskontext ist eine Unterstützung bei der Behandlung und Betreuung psychisch erkrankter Personen durch Chatbots. Um den Chatbots den erforderlichen Kontext und das entsprechende Hintergrundwissen zu vermitteln, wird eine Wissensrepräsentation über die psychischen Erkrankungen benötigt. Die zu entwickelnde Konsolenapplikation *MedExtractor* ist dazu gedacht, Forschenden und später auch Chatbots zu ermöglichen, aus einem Input in Form einer reinen Textdatei die Wissensrepräsentation zu erstellen, siehe Abbildung 3.1. Dazu soll die Applikation mit dem Namen des Textes als Parameter aufgerufen werden. Das Ergebnis der Applikationsausführung ist eine RDF-Datei.

Als zusätzliche Funktionalität ist denkbar, die Wissensbasis in Form der trainierten spaCy-EntityLinker-Repräsentation auszugeben.

Abbildung 3.1: Anwendungsfall der Konsolenapplikation

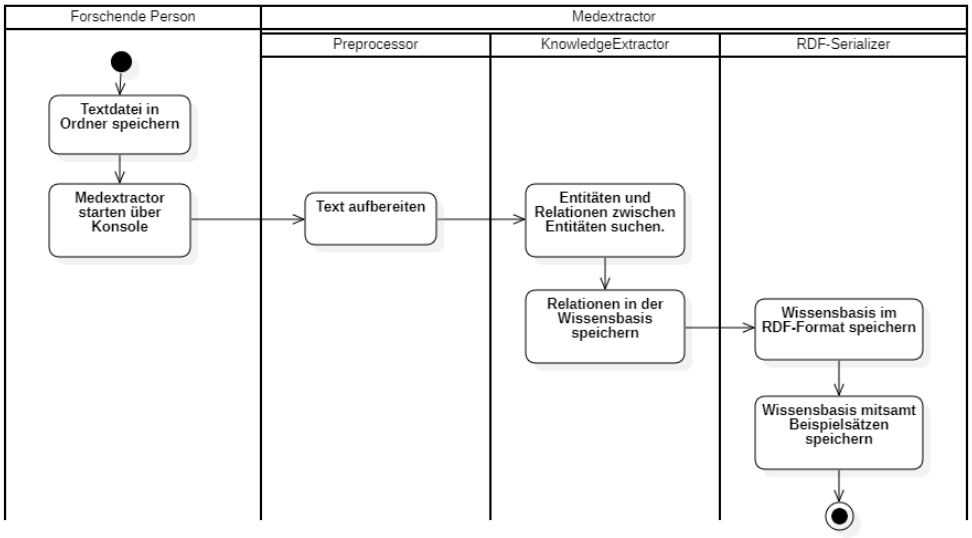

Abbildung 3.2: Aktivitätsdiagramm der Konsolenapplikation

3.1.2 *Aktivitätsmodell*

Für die Modellierung des Anwendungsfalls sind die in Abbildung 3.2 dargestellten Aktivitäten erforderlich.

3.1.3 *Informationsmodell*

Für den Aufbau der Wissensbasis sind erstens die gefundenen Entitäten und zweitens die Beziehungen zwischen diesen von Bedeutung. Diese werden in Form von semantischen Beziehungen in der Wissensbasis gespeichert, wie dies in Abbildung 3.3 des Informationsmodells (ER-Diagramm) der Applikation dargestellt ist.

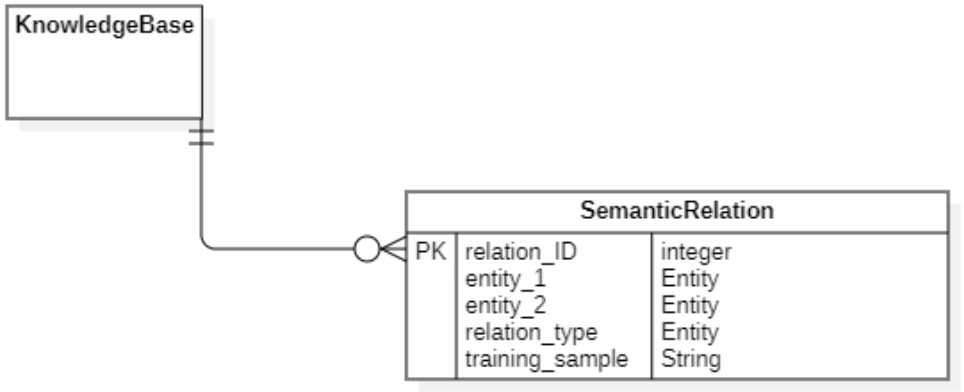

Abbildung 3.3: Informationsmodell der Konsolenapplikation

3.1.4 *Architektur- und Komponentenmodell*

Aus dem Anwendungsfall und dem Aktivitätsmodell ergibt sich die grobe Gesamtarchitektur der Applikation wie in Abbildung 3.4 dargestellt.

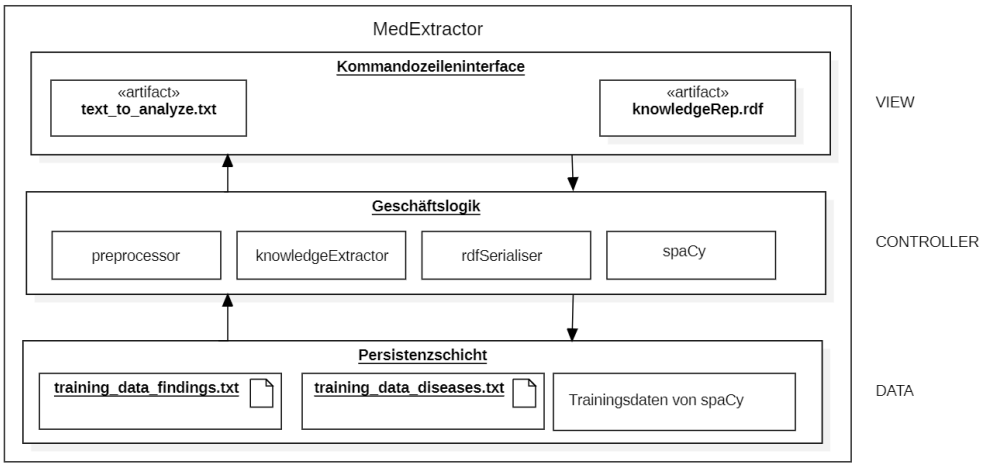

Abbildung 3.4: Architekturmodell der Konsolenapplikation

Abbildung 3.5 stellt das Komponentenmodell der Applikation dar.

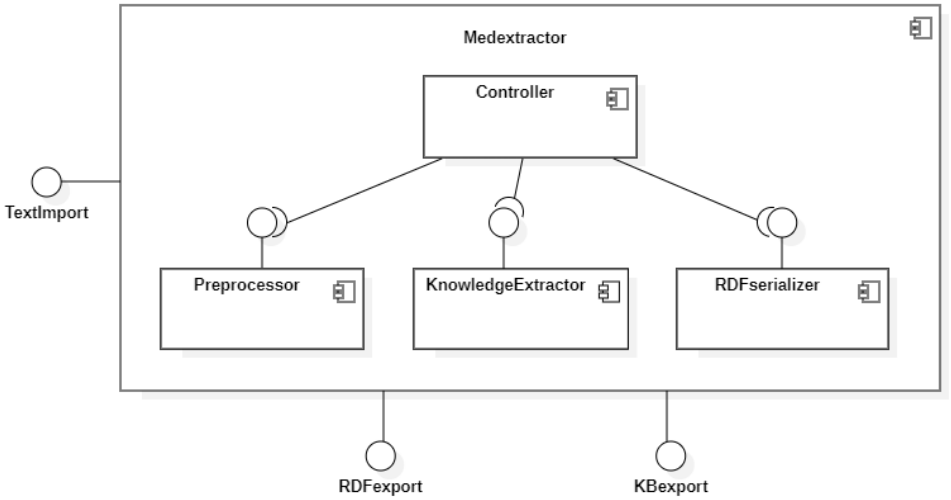

Abbildung 3.5: Komponentenmodell der Konsolenapplikation

3.2 FZ 1.2 THEORIEBILDUNG ZUR VORVERARBEITUNG MEDIZINISCHER TEXTE

Um die Vorverarbeitung medizinischer Texte durchführen zu können, wird dem Präprozessor (Abbildung 3.6) der Originaltext übergeben. Die Umwandlung und Ausgabe des Textes erfolgt über die Methode get_preprocessed_text(). Der Text wird in vollständige Sätze umgewandelt, die für sich alleine stehen

können. D.h. Aufzählungen werden in Sätze umgewandelt und Pronomen durch konkrete Substantive ersetzt.

Für die Erkennung der Satzeinheiten kann die Bibliothek *pySBD - python Sentence Boundary Disambiguation* verwendet werden, die regelbasiert auf Grundlage von 100 „Goldenen Regeln“ arbeitet.

Falls möglich, soll auch eine Indexstruktur (Kapitel-/Abschnittsüberschriften) ermittelt werden.

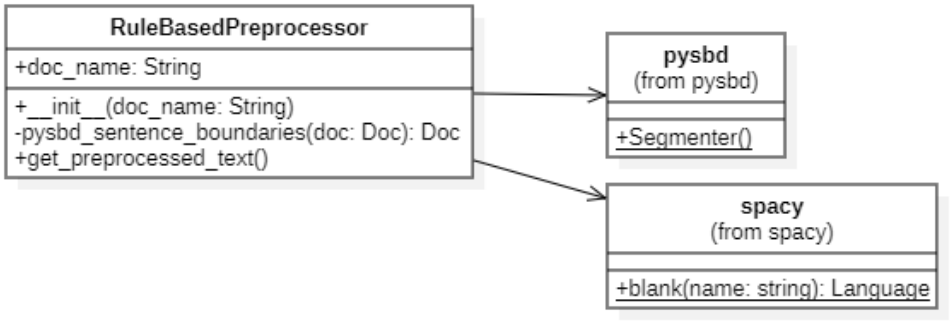

Abbildung 3.6: Klassendiagramm des Präprozessors der Konsolenapplikation

3.3 FZ 2.3 THEORIEBILDUNG ZUR ÜBERFÜHRUNG MEDIZINISCHEN FACHVOKABULARS IN MASCHINENLESBARE FORM ZUR WEITEREN VERARBEITUNG DURCH NLP

Beim Starten des MedExtractors wird eine KnowledgeExtractor-Instanz erzeugt (Abbildung 3.7). Zur Instanziierung wird der Dateiname der persistent gespeicherten Wissensbasis übergeben. Das KnowledgeExtractor-Objekt verwaltet das KnowledgeBase-Objekt. Der EntityRuler wird der spaCy-Pipeline hinzugefügt.

Dieser Instanz kann durch den Aufruf KnowledgeExtractor(text:String) ein vom Preprocessor vorbereiteter Abschnitt oder Satz zur Analyse übergeben werden. Daher muss KnowledgeExtractor die Methode __call__(text:String) implementieren. Zusätzlich kann über die Methode set_context(Entity[]) dem KnowledgeExtractor mitgeteilt werden, um welches Thema es in dem Abschnitt oder Satz geht. Das Thema ergibt sich aus Entitäten, die in Kapitel- oder Abschnittsüberschriften gefunden werden.

is_related() ist eine private Methode, die vom KnowledgeExtractor-Objekt verwendet wird, um zu prüfen, ob zwei beliebige vom EntityRuler gefundene Entitäten in Beziehung zueinander stehen. Die SemanticRelation-Objekte werden auf Basis eines wiederholten Aufrufs von is_related() mit unterschiedlichen Kombinationen gefundener Entitäten erzeugt.

3.4 FZ 3.2 THEORIEBILDUNG ZUR WISSENSREPRÄSENTATION

Der RDFSerialiser (Abbildung 3.8) serialisiert die Wissensrepräsentation in eine RDF/XML-Datei. Dabei arbeitet der RDFSerialiser mit der Python-Biblio-

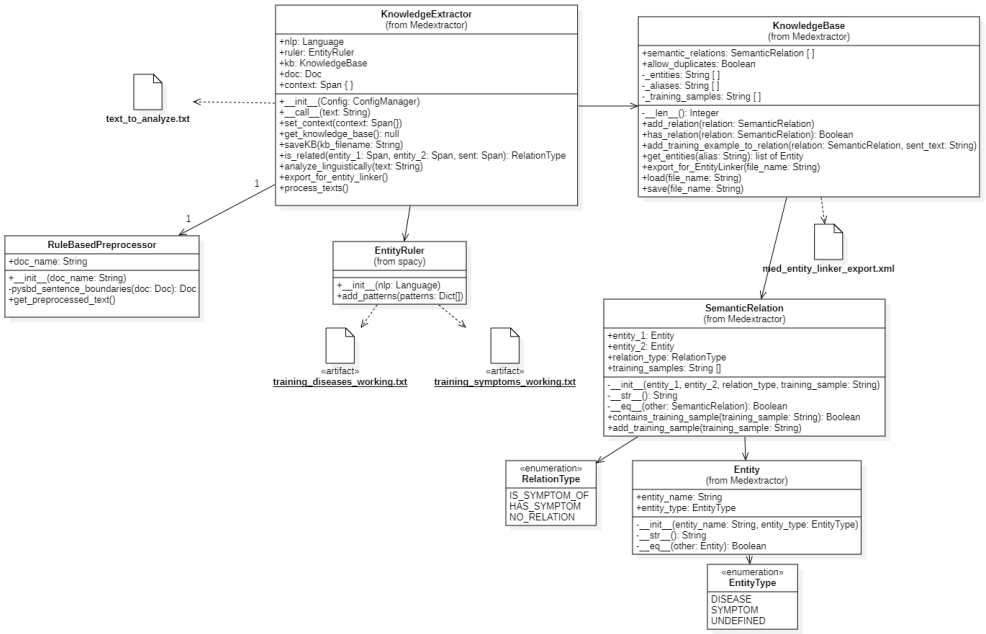

Abbildung 3.7: Klassendiagramm des KnowledgeExtractors der Konsolenapplikation

thek rdflib. Zunächst soll ein Graph ohne Inhalt erstellt werden mit der Funktion create_graph, dies geschieht mit der Klasse GraphManager. Anschließend sollen die Inhalte der übergebenen KnowledgeBase in den Graphen übertragen werden (knowledgebase_to_graph), dies geschieht mit den für GraphManager implementierten Methoden. Der so entstandene Graph wird dann mit der Methode serialise_graph in das in serialisation_format spezifizierte Format übertragen und anschließend die so entstandene Wissensrepräsentation gespeichert.

3.5 ZUSAMMENFASSUNG

Abbildung 3.9 stellt das Hauptklassendiagramm der Applikation dar.

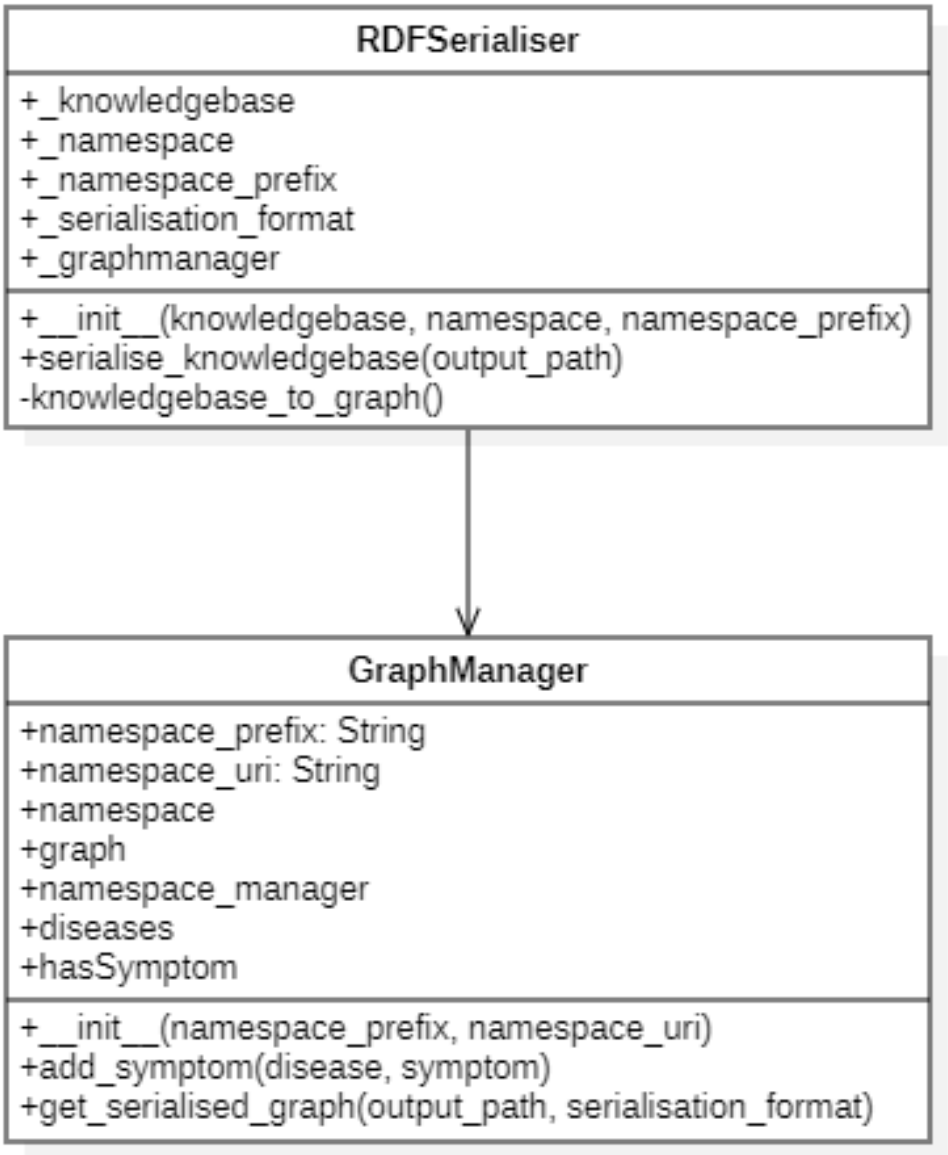

Abbildung 3.8: RDFSerialiser der Konsolenapplikation

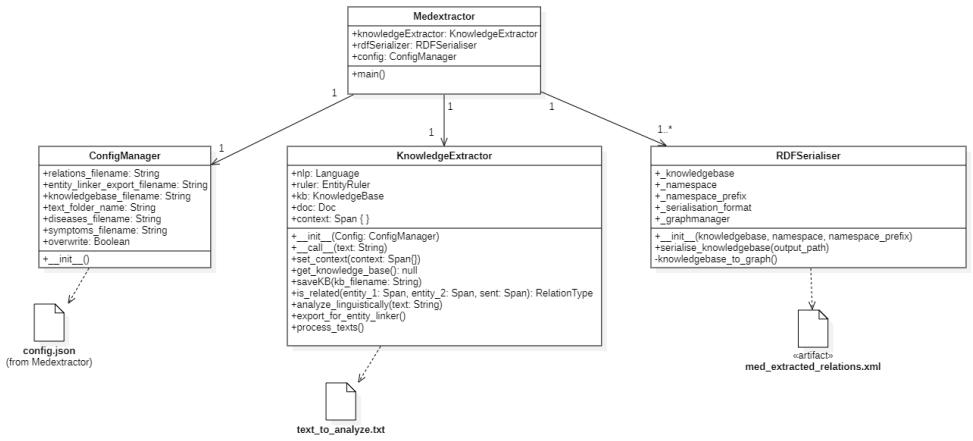

Abbildung 3.9: Hauptklassendiagramm der Konsolenapplikation

4 PROOF-OF-CONCEPT IMPLEMENTIERUNG

4.1 FZ 1.3 IMPLEMENTIERUNG ZUR VORVERARBEITUNG MEDIZINISCHER TEXTE

Da die zu verarbeitenden medizinischen Texte nicht nur Fließtext enthalten, sondern auch Überschriften und Aufzählungen, ist als erster Schritt eine sinnvolle Unterteilung in Satzeinheiten erforderlich. Dazu kommt der *Sentence Boundary Detector* PySBD ([**sadvilkar_pysbd_2020**]) zum Einsatz. Diese unterteilt den Text zunächst in einzelne Sätze, siehe **??** für weitere Informationen zu PySBD. Anschließend werden die Satzeinheiten noch so bearbeitet und zusammengefügt, dass möglichst sinnvolle Sätze für die weitere Verarbeitung entstehen. Ist eine erkannte Satzeinheit zum Beispiel ein leerer String, so wird diese nicht weiter verarbeitet. Nachgestellte Leerzeichen werden gelöscht. Außerdem werden Aufzählungen möglichst in einzelne Sätze verwandelt. So wird beispielsweise aus der Auflistung

The psychological symptoms of depression include:
- continuous low mood or sadness
- feeling hopeless and helpless
- having low self-esteem

eine Reihe von Sätzen:

The psychological symptoms of depression include continuous low mood or sadness.
The psychological symptoms of depression include feeling hopeless and helpless.
The psychological symptoms of depression include having low self-esteem.

Um dies zu erreichen wird bei jedem zu verarbeitenden Satz z.B. versucht herauszufinden, ob dieser Satz eine Aufzählung einleitet (*enumeration* im Skript), und während einer Aufzählung wird das Aufzählungszeichen zum Abgleich gespeichert (*bullet_point* im Skript).
Insgesamt wurde sich bei der Implementierung sehr an den vorhandenen Textbeispielen orientiert, sodass bei einer Verallgemeinerung auf Texte unterschiedlichster Art voraussichtlich noch weitere Anpassungen gemacht werden müssten, siehe auch Abschnitt 5.1.1.

4.2 FZ 2.4 IMPLEMENTIERUNG ZUR ÜBERFÜHRUNG MEDIZINISCHEN FACHVOKABULARS IN MASCHINENLESBARE FORM ZUR WEITEREN VERARBEITUNG DURCH NLP

Wie in Abschnitt 1.6 beschrieben, wurde das Vokabular der Datei *postings* des MetaMapLite-Projektes verwendet. Aus dieser Datei wurden zwei Dateien *training_diseases_original.txt* und *training_symptoms_original.txt* extrahiert, die jeweils nur die Einträge enthalten, die in der *postings*-Datei als *disease* bzw. *disorder* oder *finding* markiert sind. Diese Dateien bleiben als Referenz unverändert. Das Programm verwendet Kopien namens *training_diseases_working.txt* und *training_symptoms_working.txt*, aus denen bei Bedarf Begriffe entfernt werden können. Dies ist sinnvoll, da die Vokabular-Dateien sehr umfangreich sind, dadurch aber auch Einträge enthalten, die zum Auffinden von Krankheiten und Symptomen nicht hilfreich sind.

Jeder Eintrag besteht aus dem CUI, der Krankheit bzw. dem Symptom und der Kategorisierung als *DISEASE* oder *FINDING*. Als Beispiel seien hier die Einträge für die Begriffe *depression* und *loss of interest* angegeben:

C0011570	depression	DISEASE
C0424091	loss of interest	FINDING

Für das Auffinden der Named Entities wird die *spaCy*-Komponente *Entity Ruler* verwendet. Diese durchsucht einen Text konventionell nach den in den beiden Vokabular-Dateien eingetragenen Begriffen und gibt im Fall von Mehrdeutigkeiten die jeweils längsten Begriffe aus. Der *Entity Ruler* wird dem *Entity Recognizer* vorgezogen, da letzterer auf statistischer Modellierung beruht und das spezifische medizinische Vokabular durch eine Vielzahl von Beispielsätzen trainiert werden muss. Da das Vokabular sehr umfangreich ist und Trainingssätze für das Vokabular (noch) nicht zur Verfügung stehen, kann der *Entity Recognizer* nicht zuverlässig verwendet werden.

Die Verwendung des *Entity Rulers* erfolgt im Modul *knowledge_extractor.py*. Der *Entity Ruler* wird bei Instanziierung des *KnowledgeExtractor*-Objektes trainiert, indem im Konstruktor zunächst die Vokabular-Dateien geladen, ihre Einträge aufbereitet und diese schließlich dem *Entity Ruler* durch die *spaCy*-Funktion *add_patterns()* übergeben werden. Ebenfalls im Konstruktor der *KnowledgeExtractor*-Klasse wird ein *KnowledgeBase*-Objekt erzeugt. Dieses dient dazu, gefundene Wissensinhalte zu speichern.

Die Suche nach Wissensinhalten ist in diesem Prototypen sehr einfach gehalten. Es wird in jedem vorprozessiertem Satz nach Entitäten (Krankheiten, Symptome) gesucht und eine Beziehung zwischen Krankheit und Symptom dann angenommen, wenn beide Begriffe in einem Satz vorkommen. Wird z.B. dem *KnowledgeExtractor* der Satz *„During a period of depression, your symptoms may include loss of interest in everyday activities«* übergeben, so findet der *Entity Ruler* die Krankheit *depression* und das Symptom *loss of interest*. Der *KnowledgeExtractor* sieht dann im Verlust von Interesse ein Symptom für eine Depression.

Symptome sind in der Vokabular-Datei nicht mit ihren Negationen enthalten. So gibt es z.B. in dem Symptom-Vokabular den Eintrag *motivation*, aber nicht den Eintrag *no motivation*. Aus dem Satz *„The psychological symptoms of depression include having no motivation or interest in things.«* würde der KnowledgeExtractor daher den Zusammenhang extrahieren, dass *motivation* ein Symptom von *depression* ist. Um solche Fehler zu vermeiden, prüft der *KnowledgeExtractor* für jedes gefundene Symptom, ob dessen Negierung, hier also *no motivation*, in dem zu analysierenden Text enthalten ist. Ist dies der Fall, so speichert er die Negierung der gefundenen Entität in der Wissensbasis.

Allerdings funktioniert dieses einfache Konzept für obigen Beispielsatz nicht bei dem Symptom *interest*, da vor diesem das Wort *no* fehlt und aus dem Kontext erschlossen werden muss. Im *spaCy Universe* (https://spacy.io/universe) gibt es das Projekt *negspaCy*, das negierte Begriffe mit einem entsprechenden Tag versieht. Leider funktioniert dieses Modul nur für einfache Sätze und liefert bei den von uns untersuchten Beispieltexten zu häufig keine guten Ergebnisse. Daher wurde auf die Verwendung von *negspaCy* verzichtet.

4.3 FZ 3.3 IMPLEMENTIERUNG ZUR WISSENSREPRÄSENTATION

4.4 IMPLEMENTIERUNG WEITERER FUNKTIONALITÄT

Zusätzlich zu den ausgearbeiteten Forschungszielen haben sich während der Implementierung die folgenden Funktionalitäten herauskristallisiert, die einen Mehrwert für das Programm darstellen.

4.4.1 *Erstellung von Trainingsdaten für den Entity-Linker*

Analysiert der *KnowledgeExtractor* zahlreiche Texte, so ergibt sich die Möglichkeit, nicht nur eine Wissensrepräsentation zu erstellen, sondern auch Beispielsätze zu sammeln. Der *KnowledgeExtractor* speichert daher auch Beispielsätze in der Wissensbasis. Diese Beispielsätze können für das Training statistischer Modelle wie z.B. des *Entity Linkers*, einer weiteren optionalen Komponente der *spaCy*-Pipeline, eingesetzt werden.

Der *Entity Linker* arbeitet zusammen mit dem *Entity Ruler* bzw. *Entity Recognizer* und verknüpft von diesen Komponenten gefundene Entitäten mit Entitäten aus seiner eigenen Wissensbasis. Die Idee des *Entity Linkers* ist es, mehrdeutigen Entitäten (z.B. *Apple*) kontextbezogen konkrete Entitäten zuzuordnen, also etwa die Frucht, die Firma *Apple* oder die Stadt New York (*Big Apple*). Die vom *Entity Recognizer* oder *Entity Ruler* gefundenen Begriffe werden in diesem Zusammenhang als *Aliase* bezeichnet.

Der *Entity Linker* kann somit auch dafür verwendet werden, Symptomen mögliche Krankheiten zuzuordnen, denn Symptome können zu mehreren Krankheiten gehören. Es wurde daher eine Funktion implementiert, den Inhalt der Wissensbasis des *MedExtractors* in eine xml-Datei zu exportieren, in der die Daten so aufbereitet sind, dass sie einfach vom *Entity Linker* impor-

tiert werden können. Wie dieser Import funktioniert, wird in dem Jupyter-Notebook *entity_linker_demo.ipynb* vorgeführt.

Abbildung 4.1 zeigt einen Ausschnitt aus diesem Jupyter Notebook. Der *spaCy*-Pipeline wird ein Text übergeben, in dem ein Patient seine Beschwerden während des Besuchs einer Shopping Mall beschreibt. In den nächsten beiden Zeilen wird jeweils über die vom *Entity Ruler* gefundenen Entitäten in der Liste *doc.ents* iteriert. Der *Entity Linker* besitzt als Attribut das *spacy.KnowledgeBase*-Objekt *kb*[1], in dem die vom Medextractor gefundenen Entitäten und Aliase mitsamt der Verknüpfungen (Links) zwischen Entitäten und Aliasen gespeichert sind (nicht jedoch die Beispielsätze). Im ersten Fall wird die Funktion *get_alias_condidates()* des *KnowledgeBase*-Objektes verwendet, um sich für jedes im Text gefundene Symptom die dazu verknüpften Krankheiten anzeigen zu lassen.

```
In [5]: doc = nlp('Yesterday I entered the shopping mall when out of a sudden I had a panic attack.'
                  'I was hyperventilating for a moment and felt hot and sweaty. '
                  'And I started to feel chest pain.')

In [6]: for ent in doc.ents:
            candidates = entity_linker.kb.get_alias_candidates(ent.text)
            print(ent,ent.label_,[cand.entity_ for cand in candidates])

panic attack SYMPTOM ['agoraphobia', 'dysphagia', 'panic disorder', 'shock', 'social anxiety disorder']
hyperventilating SYMPTOM ['agoraphobia']
hot and sweaty SYMPTOM ['agoraphobia']
chest pain SYMPTOM ['agoraphobia']

In [7]: predicts = entity_linker.predict(doc)
        for i,ent in enumerate(doc.ents):
            print(ent,ent.label_,predicts[i])

panic attack SYMPTOM panic disorder
hyperventilating SYMPTOM agoraphobia
hot and sweaty SYMPTOM agoraphobia
chest pain SYMPTOM agoraphobia
```

Abbildung 4.1: Auszug aus dem Jupyter notebook *entity_linker_demo.ipynb*

In den meisten Fällen wird als Krankheit *Agoraphobia (Platzangst)* angegeben, speziell für das Symptom *panic attack* aber auch z.B. *panic disorder* oder *shock*. Eine einfache Auswertung könnte darin bestehen, die am häufigsten genannt Krankheit, hier also *Agoraphobia* als die wahrscheinlichste Erkrankung zu betrachten, die zu der Beschreibung des Patienten passt.

Alternativ dazu verwendet das zweite Beispiel die Funktion *predict()* des *Entity Linkers*. Dieser Funktion wird das gesamte *doc*-Objekt übergeben und für alle vom *Entity Ruler* gefundenen Entitäten von *predict()* eine Vorhersage der Erkrankung ausgegeben. Hier wird das durch die Beispielsätze trainierte statistische Modell des *Entity Linkers* verwendet. Im Idealfall ist der *Entity Linker* in der Lage kontextbezogen die wahrscheinlichste Erkrankung zu identifizieren. Die Zuordnung des Symptoms *panic attack* zur Erkrankung *panic disorder* ist hier sinnvoll, da eine *Agoraphobia* nicht immer mit Panikattacken verbunden ist. Eine weitere Untersuchung des *Entity Linkers* ist noch nicht erfolgt, da davon auszugehen ist, dass die Menge an Trainingssätzen noch nicht ausreicht, um eine zuverlässige Funktion des *Entity Linkers* zu demonstrieren.

1 die dazugehörige *spaCy*-Klasse entspricht nicht der *KnowledgeBase*-Klasse im Modul *base.py* des *Medextractors*

4.4.2 *Linguistische Analyse der Entitäten*

Die erstellte prototypische Implementierung erbringt bereits mit der reinen Vokabelanalyse zu Krankheiten und Symptomen gute Ergebnisse. Für weitergehende Untersuchungen scheint es von Vorteil, zu den Entitäten auch die entsprechenden Wortarten und ihre Funktion im jeweiligen Satz zu ermitteln, um darauf aufbauend genauere Beziehungen zwischen den Entitäten ermitteln zu können. Eine grundlegende Implementierung dieser Funktionalität steht mit der Methode `analyze_linguistically()` zur Verfügung.

5 EVALUIERUNG

Verglichen mit mathematischen Berechnungen sind computerlinguistische Aufgabenstellungen weniger deterministisch und es ist weniger offensichtlich, was die *richtige* Antwort ist. Aus diesem Grunde sollen die Zwischen- und Endergebnisse des Programms mittels stichprobenartiger Betrachtungen und Untersuchungen hinsichtlich Auffälligkeiten ausgewertet werden.

Für die Evaluierung der erzeugten RDF-Dateien wurden zudem von einem Teil der verarbeiteten Texte manuell RDF-Dateien zum Abgleich erzeugt. Für diese RDF-Dateipaare wurde Precision und Recall ermittelt.

5.1 FZ 1.4 EVALUIERUNG ZUR VORVERARBEITUNG MEDIZINISCHER TEXTE

Zur Evaluierung des Preprocessors wurden sowohl allgemeine Untersuchungen des Preprocessors angestellt als auch eine Analyse des Sentence Boundary Detectors *PySBD* durchgeführt.

5.1.1 *Allgemeine Evaluation des Preprocessors*

Ziel des Preprocessors ist es, die gegebenen Texte so umzuwandeln, dass der anschließend angewandte SpaCy-Sentencizer den Text in möglichst sinnvolle Sätze aufteilt, die dann weiterverarbeitet werden. Ein Hauptaugenmerk während der Implementierung lag auf der sinnvollen Aufteilung von Aufzählungen. Wie schon im Kapitel 4 geschrieben gelingt das.

Ohne den Preprocessor würde man der weiteren Verarbeitung z. B. den „Satz“aus dem *TextToAnalyze.txt* übergeben:

> „Psychological symptoms\n\nThe psychological symptoms of depression include:\n- continuous low mood or sadness\n- feeling hopeless and helpless\n- having low self-esteem\n- feeling tearful\n- feeling guilt-ridden\n- feeling irritable and intolerant of others\n- having no motivation or interest in things\n- finding it difficult to make decisions\n- not getting any enjoyment out of life\n- feeling anxious or worried\n- having suicidal thoughts or thoughts of harming yourself\n\nPhysical symptoms\n\nThe physical symptoms of depression include:\n- moving or speaking more slowly than usual\n- changes in appetite or weight (usually decreased, but sometimes increased)\n- constipation\n- unexplained aches and pains\n- lack of energy\n- low sex drive (loss of libido)\n- changes to your menstrual cycle\n- disturbed sleep – for example, finding it difficult to fall asleep at night or waking

> up very early in the morning\n\nSocial symptoms\n\nThe social symptoms of depression include:\n- avoiding contact with friends and taking part in fewer social activities\n- neglecting your hobbies and interests\n- having difficulties in your home, work or family life\n\nSeverities of depression\n\nDepression can often come on gradually, so it can be difficult to notice something is wrong."

Mit Anwendung des Preprocessors übergibt man stattdessen die folgenden Sätze:

1. „The psychological symptoms of depression include continuous low mood or sadness.\n"
2. „The psychological symptoms of depression include feeling hopeless and helpless.\n"
3. „The psychological symptoms of depression include having low self esteem.\n"
4. „The psychological symptoms of depression include feeling tearful.\n"
5. „The psychological symptoms of depression include feeling guilt ridden.\n"
6. „The psychological symptoms of depression include feeling irritable and intolerant of others.\n"
7. „The psychological symptoms of depression include having no motivation or interest in things.\n"
8. „The psychological symptoms of depression include finding it difficult to make decisions.\n"
9. „The psychological symptoms of depression include not getting any enjoyment out of life.\n"
10. „The psychological symptoms of depression include feeling anxious or worried.\n"
11. „The psychological symptoms of depression include having suicidal thoughts or thoughts of harming yourself.\n"
12. „Physical symptoms.\n"
13. „The physical symptoms of depression include moving or speaking more slowly than usual.\n"
14. „The physical symptoms of depression include changes in appetite or weight (usually decreased, but sometimes increased).\n"
15. „The physical symptoms of depression include constipation.\n"

16. „The physical symptoms of depression include unexplained aches and pains.\n“
17. „The physical symptoms of depression include lack of energy.\n“
18. „The physical symptoms of depression include low sex drive (loss of libido).\n“
19. „The physical symptoms of depression include changes to your menstrual cycle.\n“
20. „The physical symptoms of depression include disturbed sleep for example, finding it difficult to fall asleep at night or waking up very early in the morning.\n“
21. „Social symptoms.\n“
22. „The social symptoms of depression include avoiding contact with friends and taking part in fewer social activities.\n“
23. „The social symptoms of depression include neglecting your hobbies and interests.\n“
24. „The social symptoms of depression include having difficulties in your home, work or family life.\n“
25. „Severities of depression.\n“
26. „Depression can often come on gradually, so it can be difficult to notice something is wrong.\n“

Der Preprocessor wurde unter Berücksichtigung der gewählten Beispieltexte aufgebaut. Diese Texte haben alle eine ähnliche Struktur (z. B. viele Texte mit Aufzählungen, wenige Literaturangaben, keine Dialog-Texte). Es kann sein, dass Texte mit stark abweichender Struktur von dem Preprocessor nicht optimal verarbeitet werden.

Während der Evaluation des Preprocessors wurde getestet, wie sich die Ergebnisse des MedExtractors unterscheiden, wenn man die Texte mit dem Preprocessor vorverarbeitet bzw. dies nicht tut. Dabei ist aufgefallen, dass ohne die Vorverarbeitung des Preprocessors (d. h. wenn die Texte nur mit dem SpaCy-Sentencizer aufgeteilt werden), anschließend mehr Einträge in der resultierenden KnowledgeBase gespeichert sind. Dieser Test wurde auf zwei verschiedenen Textmengen durchgeführt. Die Anzahl der jeweiligen KnowledgeBase-Einträge sind in folgender Tabelle dokumentiert:

	mit Preprocessing	ohne Preprocessing
Textmenge 1	1566	2089
Textmenge 2	432	903

Bei einer Weiterentwicklung des Preprocessors sollte näher untersucht werden wodurch diese Unterschiede entstehen und überlegt werden, wie man damit umgeht.

5.1.2 *PySBD*

PySBD ([**sadvilkar_pysbd_2020**]) ist ein regelbasierter *Sentence Boundary Detector/Disambiguater* (Satzgrenzen-Finder). Die Entwicklung von PySBD basiert auf einem Golden Rule Set ([**golden_rules**]). Diese Golden Rules sind Spezialfälle in Sätzen, nach deren richtiger Bearbeitung ein Sentence Boundary Detector beurteilt werden kann. Eine der Golden Rules ist zum Beispiel

> **„U.S. as non sentence boundary“**
> I have lived in the U.S. for 20 years.
> => [„I have lived in the U.S. for 20 years.“]

Im Vergleich mit anderen Sentence Boundary Detectors erfüllt PySBD diese Golden Rules sehr gut, siehe [**sadvilkar_pysbd_2020**].

5.1.2.1 *Vergleich der Einteilung von Texten in Sätze mit und ohne PySBD*

Es soll untersucht werden worin die Vor- und Nachteile beim Einteilen von Texten in Sätze mit der Bibliothek *PySBD* liegen. Alternativ gibt es die Möglichkeit den Sentencizer von SpaCy zu nutzen. Für den Vergleich wird die Verarbeitung der 51 Texte in *medextractor/resources/to_analyze* verglichen. Dies geschieht in der python-Datei *test/pysbd_evaluation_sentences.py*.

Es fällt schnell auf, dass der Hauptunterschied in der Einteilung von Texten in einzelne Sätze mit und ohne PySBD darin besteht, dass ein Zeilenumbruch unterschiedlich interpretiert wird. Aus dem folgenden Textabschnitt in der Datei *agorophobia.txt*

> „\nSymptoms - Agoraphobia\n\nThe severity of agoraphobia can vary significantly between individuals.“

erhält man mit dem SpaCy Sentencizer zum Beispiel den Satz

> „\nSymptoms - Agoraphobia\n\nThe severity of agoraphobia can vary significantly between individuals.“,

während die Aufteilung in Sätze mit PySBD folgendermaßen aussieht:

1. „Symptoms - Agoraphobia\n\n“
2. „The severity of agoraphobia can vary significantly between individuals.\n\n“

Insgesamt wird von den 51 Texten nur einer vom PySBD- und SpaCy-Sentencizer auf die gleiche Art in Sätze eingeteilt.
Wendet man nun die folgende Art der Nachverarbeitung der Texte an:

1. Sätze, die ein „\n“enthalten, werden an dieser Stelle in zwei Sätze aufgesplittet,

2. Leerzeichen am Anfang und am Ende von Sätzen werden entfernt,
3. leere Sätze werden entfernt,

werden schon 41 der 51 Texte komplett gleich verarbeitet. Ausnahmen bilden dann (fast) nur noch Sätze, die Zeichen wie z.B. „]“, „:“und „-“enthalten (diese speziellen Zeichen werden vom PySBD- und SpaCy-Sentencizer unterschiedlich behandelt), oder Satzanfänge ohne vorhergehendes Leerzeichen.

5.1.2.2 *Fazit zu PySBD*

Der Preprocessing-Schritt des MedExtractors enthält nicht nur das Erkennen von Satzgrenzen, sondern noch weitere Schritte um die Sätze so umzuformen, dass sie in der weiteren Verarbeitung mit dem MedExtractor gut genutzt werden können (siehe Kapitel [TODO]). Für die Zwecke des MedExtractors und für die zur Evaluierung genutzten Texte war PySBD sehr hilfreich. Während der Evaluierung ist aber auch aufgefallen, dass mit Berücksichtigung der Eigenheiten des SpaCy Sentencizers wahrscheinlich ähnlich gute Ergebnisse im Preprocessing-Schritt hätten erzielt werden können. Somit kann PySBD für die Zwecke des MedExtractors zwar empfohlen werden, mit anderen Sentence Boundary Detectorn könnten aber voraussichtlich ähnlich gute Ergebnisse erzielt werden.
In der weiteren Entwicklung des Preprocessors, bzw. des MedExtractors, könnte man versuchen mit einer besseren Einstellung der Parameter von PySBD bessere Ergebnisse zu erzielen, oder mit einem anderen *Sentence Boundary Detector* (mehrere Alternativen werden in [**sadvilkar_pysbd_2020**] genannt) zu arbeiten.

5.2 FZ 2.5 EVALUIERUNG ZUR ÜBERFÜHRUNG MEDIZINISCHEN FACHVOKABULARS IN MASCHINENLESBARE FORM ZUR WEITEREN VERARBEITUNG DURCH NLP

Für die *Named Entity Recognition* ist der *EntityRuler* das einfachste Mittel, das von *spaCy* angeboten wird. Der *Entity Ruler* sucht konventionell die trainierten Begriffe im Text, wobei bei Mehrdeutigkeit / Überlappungen der jeweils längste Ausdruck (gemessen an der Anzahl der Worte) oder, wenn dies nicht eindeutig ist, der als erstes auftretende Ausdruck angezeigt wird. Der *Entity Ruler* benötigt keine Trainingssätze und ist daher für das Medextractor-Projekt am Anfang besser geeignet als der auf statistischen Modellen basierende *Entity Recognizer*.

Mit den Exporten für den *Entity Linker* steht nun aber eine größere Anzahl von Trainingsätzen zur Verfügung und es stellt sich die Frage, ob diese ein Training des *Entity Recognizers* erlauben, so dass dieser zuverlässig Entitäten erkennt.

In dem Jupyter Notebook *entity_recognizer_demo.ipynb* werden *Entity Ruler* und *Entity Recognizer* gegenübergestellt. Die Trainingsdaten stammen in

beiden Fällen aus der Datei *all_entity_linker_export.xml*. Diese wiederum wurden vom Medextractor aus den drei Quellen extrahiert, die auch in Abschnitt 5.3.2 analysiert werden (vgl. Tabelle 5.4). Als Test-Text wurde der englischsprachige Artikel über *Agoraphobie* auf Wikipedia verwendet, der nicht Teil der drei Quellen ist, aus denen die Trainingsdaten stammen.

Der *Entity Ruler* findet 105 und der *Entity Recognizer* 98 Entitäten. Überwiegend decken sich die Ergebnisse von *Entity Ruler* und *Entity Recognizer*. Jedoch findet der *Entity Recognizer* 12 Entitäten, die nicht trainiert wurden und falsch sind. Dazu gehören z.B. die Begriffe *scholar*, *social sciences*, *prone* oder *tense*, die vom neuronalen Netz fälschlich als Entitäten identifiziert werden.

Interessanterweise findet der *Entity Recognizer* den Begriff *suicide ideation* (Suizidgedanken), obwohl dieser Begriff nicht im trainierten Vokabular enthalten ist. Das in dieser Arbeit aus der *MetaMapLite*-Datenbank abgeleitete Trainingsvokabular kennt nur die Pluralform *suicidal ideations* und verwendet das Wort *suicidal* anstelle von *suicide*. Daher findet der Medextractor den Begriff nicht in dem Wikipedia-Text über Agoraphobie und auch nicht in den anderen Texten, die von ihm analysiert wurden, so dass er nicht in der Wissensrepräsentation und auch nicht in der Datei *all_entity_linker_export.xml* enthalten ist. Auch den Begriff *suicide* findet der Medextractor nicht, da dieses Wort nicht im Symptom-Vokabular enthalten ist.

In den vom Medextractor gefundenen Trainingssätzen kommt der Begriff *suicidal ideation* im Singular dagegen mehrfach vor, da der Medextractor in diesen Sätzen andere Symptome gefunden hat. Man mag in dem Umstand, dass der *Entity Recognizer* den Begriff *suicide ideation* findet, ein 'intelligentes' Verhalten des zugrundeliegenden neuronalen Netzes sehen. Jedoch zeigt der Vergleich insgesamt, dass der *Entity Ruler* zuverlässiger arbeitet als der *Entity Recognizer*. Um die Vorteile eines statistischen Modells nutzen zu können (z.B. kontextsensitive Worterkennung), bedarf es offenbar noch mehr Trainingsdaten.

5.3 FZ 3.4 EVALUIERUNG ZUR WISSENSREPRÄSENTATION

In Abschnitt 5.3.1 erfolgt eine statistische Auswertung der Vergleichsergebnisse von manuell und mit dem MedExtractor erstellten RDF-Dateien. Danach werden diese und weitere beim stichprobenartigen Sichten der MedExtractor-Ergebnisse gefundene Erkenntnisse in Abschnitt 5.3.2 diskutiert.

5.3.1 *Vergleich der manuell und vom MedExtractor erstellten RDF-Dateien*

Zur Evaluierung des Systems wurden von zehn der verwendeten Texte manuelle RDF-Dateien erstellt, um diese mit dem vom MedExtractor erstellten RDF-Dateien zu vergleichen.

Da der MedExtractor Krankheiten Symptome zuordnet, wurden ebenso bei der manuellen Dateierstellung aus den Texten Symptome ausgewählt, die in die RDF-Dateien aufgenommen wurden. Diese manuelle Erstellung er-

fordert jedoch Augenmaß, da es kein absolutes Kriterium dafür gibt, was als Symptom einer Krankheit anzusehen ist. Insbesondere enthalten die Arbeitstexte häufig Konjunktionen wie „feeling irritable and intolerant of others“, die an sich als zwei separate Symptome „feeling irritable“ sowie „feeling intolerant of others“ zu betrachten sind. Da der MedExtractor regelbasiert arbeitet und keine Funktionalität enthält, die den Bezug zwischen „feeling“ und dem hinter der Konjunktion folgenden „intolerant of others“ herstellen könnte, wurden Gesamtwortgruppen in die manuellen RDF-Dateien aufgenommen mit dem Risiko, dass nur ein Teil der Wortgruppen erkannt wird.

Zur Auswertung wurden die erstellten Dateien miteinander verglichen und Precision (Genauigkeitsquote) und Recall (Vollständigkeitsquote) ermittelt:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Dabei wurden diese Werte wie folgt berechnet:

- True Positive: als korrekt gefundene Symptome wurden Symptome gewertet, die der MedExtractor erkannt hat und die in der vom MedExtractor erkannten Form einer Teilzeichenkette eines manuell aufgenommenen Symptoms entsprechen.
- False Positive: als falsch gefundene Symptome wurden Symptome gewertet, die der MedExtractor erkannt hat und die in der vom MedExtractor erkannten Form keiner Teilzeichenkette eines manuell aufgenommenen Symptoms entsprechen.
- False Negative: als nicht gefundene Symptome wurden Symptome gewertet, die manuell aufgenommenen wurden, vom MedExtractor jedoch nicht (auch nicht in Form einer Teilzeichenkette) erkannt wurden.

Als Beispiel wird hier die Auswertung des Depressionstextes angeführt. In Tabelle 5.1 ist gegenübergestellt, welchen Entitäten in der manuell erstellten Wissensrepräsentation Entsprechungen in der MedExtractor-Wissensrepräsentation gegenüberstehen. Die Gesamtanzahl Symptome in der manuell erstellten Datei (entspricht True Positives + False Positives) beträgt 27. Von diesen 27 manuell ermittelten Symptomen wurden vom Medextractor 14 Symptome in Teilen erkannt (teilweise erkannte Symptome werden in dieser Evaluierung als positiv erkannt gezählt). Damit ergibt sich eine Precision von xxxx.

Umgekehrt stellt Tabelle 5.2 dar, welche Entitäten in der manuell erstellten Wissensrepräsentation die vom MedExtractor gefundenen Entitäten entsprechen. Der MedExtractor hat insgesamt 20 Symptome gefunden (entspricht True Positives + False Positives). 18 dieser Symptome entsprechen Symptomen, die in der manuell erstellten Datei enthalten sind (True Positives) und

Symptom_manual	Symptom_found
not getting any enjoyment out of life	
low sex drive (loss of libido)	low sex drive;loss of libido
lack of energy	lack of energy
having no motivation or interest in things	no motivation;interest
disturbed sleep	
changes to your menstrual cycle	
avoiding contact with friends	
suicidal thoughts	suicidal thoughts
having difficulties in your home, work or famil...	
taking part in fewer social activities	
moving or speaking more slowly than usual	
finding it difficult to make decisions	
feeling hopeless and helpless	feeling hopeless
thoughts of harming yourself	
finding it difficult to fall asleep at night	
feeling guilt-ridden	feeling guilt
continuous low mood	low mood
constipation	constipation
feeling tearful	tearful
neglecting your hobbies and interests	interests;interest;neglecting
sadness	
low self-esteem	
feeling irritable and intolerant of others	feeling irritable
feeling anxious or worried	worried
changes in appetite or weight	changes in appetite
unexplained aches and pains	aches;pains
waking up very early in the morning	

Tabelle 5.1: manuell erstellte Wissensrepräsentation verglichen mit vom MedExtractor gefundenen Entitäten

Symptom_Medextractor	Original_Symptom
lack of energy	lack of energy
interests	neglecting your hobbies and interests
worried	feeling anxious or worried
low mood	continuous low mood
feeling hopeless	feeling hopeless and helpless
no motivation	having no motivation or interest in things
suicidal thoughts	suicidal thoughts
interest	having no motivation or interest in things;negl...
changes in appetite	changes in appetite or weight
aches	unexplained aches and pains
low sex drive	low sex drive (loss of libido)
feeling guilt	feeling guilt-ridden
constipation	constipation
pains	unexplained aches and pains
loss of libido	low sex drive (loss of libido)
neglecting	neglecting your hobbies and interests
psychological symptoms	
low self esteem	
tearful	feeling tearful
feeling irritable	feeling irritable and intolerant of others

Tabelle 5.2: Vom MedExtractor gefundene Entitäten verglichen mit den Entsprechungen in der manuell erstellten Wissensrepräsentation

Text	No. entries manual	No. entries Medextractor	Precision	Recall
depression	27	20	0.900	0.875
anxietydisorders	42	16	0.500	0.529
bulimia	25	5	0.800	0.800
claustrophobia	37	0	0.000	0.000
dementia	21	15	0.267	0.214
eatingdisorders	41	5	0.600	0.500
panicdisorder	24	5	0.800	0.800
psychosis	22	5	1.000	1.000
socialanxiety	16	3	0.000	0.000
stress	18	0	0.000	0.000

Tabelle 5.3: Precision und Recall der Beispieltexte

2 der vom MedExtractor gefundenen Symptome sind nicht in der manuell erstellten Datei enthalten (False Positives). Damit ergibt sich ein Recall von 90

Von den beiden False Positives ist „psychological symptoms“ kein Symptom an sich und „low self esteem“ ist in der Form als „low self-esteem“ im ursprünglichen Text und daher auch in der manuellen Datei enthalten, aber bei der Vorverarbeitung des MedExtractors wurden die Bindestriche entfernt, und deshalb konnte diese Übereinstimmung nicht gefunden werden.

Tabelle 5.3 führt die Auswertungsergebnisse aller Beispieltexte an.
Klaustrophobie, Stress

5.3.2 *Auswertung von falsch erkannten Beziehungen*

Im Folgenden sollen die Fälle untersucht werden, in denen der Medextractor falsche Symptom-Krankheit-Zuordnungen findet. Die vom Medextractor analysierten Texte stammen dabei aus drei Quellen:

1. Webseite des *National Health Service UK (NHS)* - 49 Texte zu mehreren mentalen Erkrankungen oder Störungen (Agoraphobie, Alkoholmissbrauch, Angststörungen, Essstörungen, Demenz, Depression, Panikstörungen, u.s.w.) [**nhs_webpage**]
2. Ein Artikel zum Thema *Mental Disorder* auf Wikipedia [**wikimentaldisorder**]
3. Clinical Handbook of Psychological Disorders [**clinicalhandbook**]

Zur Auswertung dienen die xml-Dateien für den Entity-Linker, da diese auch die Sätze enthalten, in denen Krankheiten bzw. Symptome gefunden

Quelle	Dateigröße	Entitäten	Aliase	Sätze
NHS	272 KB	93	221	262
Wikipedia	78 KB	55	67	62
Handbook	3,5 MB	204	485	917

Tabelle 5.4: Von Medextractor gefundene Entitäten, Aliase und Beispielsätze

wurden. Die Anzahl gefundener Krankheiten, Symptome und Trainingssätze lässt sich einfach durch Zählen der Tags <\entity>, <\alias> und <\sample> ermitteln. Das Ergebnis zeigt Tabelle 5.4. Entitäten bedeuten hierbei Krankheiten und Aliase bedeuten Symptome.

Das in MetaMapLite enthaltene Vokabular erweist sich dabei als umfangreich genug, um zahlreiche richtige Symptom-Krankheit-Relationen zu finden. Man erkennt insbesondere an der Analyse des Handbuchs, dass die Zahl gefundener Begriffe nicht mit dem Umfang des Dokumentes skaliert. Dies liegt (neben der Begrenztheit des Trainingsvokabulars) daran, dass mentale Krankheiten bereits durch die Analyse der kürzeren Texte bereits weitgehend erfasst werden.

Neben den meist richtig erkannten Zusammenhänge finden sich in den xml-Dateien auch einige Zusammenhänge, die vom Medextractor falsch erkannt werden. Häufig liegt dies daran, dass im Trainingsvokabular ungeeignete Begriffe enthalten sind oder dass Grammatik und Semantik eines Satzes unzureichend analysiert werden. Es werden nun in den folgenden Abschnitten typische Fälle untersucht und Wege aufgezeigt, wie insbesondere durch Verwendung der *spaCy*-Pipeline eine Verbesserung des Medextractors erzielt werden könnte.

Da der bisherige Schwerpunkt der Arbeit auf der Verwendung der *Named Entity Recognition* lag und der Standard-NER von *spaCy* in diesem Zusammenhang wenig hilfreich ist, wurden die Stärken von *spaCy* bisher nicht ausgenutzt. Als Ziel sollte verfolgt werden, das Vokabular nicht manuell zu bearbeiten, sondern Algorithmen zu finden, mit denen *spaCy* möglichst automatisch ungeeignete Begriffe erkennt oder auch Begriffe findet, die nicht identisch dem Entity Ruler trainiert wurden.

5.3.2.1 *Verben im Symptomvokabular*

Bei Durchsicht der gefundenen Symptom-Krankheit-Relationen fällt zunächst auf, dass das Trainingsvokabular aus MetaMapLite Verben enthält, die zwar bei der Beschreibung von Krankheiten oder Symptomen häufig verwendet werden, für sich allein aber keine Rückschlüsse auf eine Krankheit oder ein Symptom erlauben.

Dazu gehört z.B. das Verb *aggravate*, das auch in den Formen *aggraveted by* und *aggravating* im Vokabular enthalten ist. So wird z.B. in folgendem Satz der Begriff *aggravate* als Symptom einer *anxiety disorder* (Angststörung)

Text	Lemma	POS	TAG	DEP
Certain	certain	ADJ	JJ	amod
substances	substance	NOUN	NNS	nsubj
such	such	ADJ	JJ	amod
as	as	ADP	IN	prep
caffeine	caffeine	NOUN	NN	pobj
may	may	AUX	MD	aux
aggravate	aggravate	VERB	VB	ROOT
the	the	DET	DT	det
symptoms	symptom	NOUN	NNS	dobj
of	of	ADP	IN	prep
anxiety	anxiety	NOUN	NN	compound
disorders	disorder	NOUN	NNS	pobj
or	or	CCONJ	CC	cc
interact	interact	VERB	VB	conj
with	with	ADP	IN	prep
prescribed	prescribed	ADJ	JJ	amod
medication	medication	NOUN	NN	pobj

Tabelle 5.5: Ergebnis der *spaCy*-Pipeline

gefunden:

„Certain substances such as caffeine ... may aggravate the symptoms of anxiety disorders or interact with prescribed medication."

Da das Verb *aggravate* allein kein Symptom ist, stellt sich die Frage, wie der Medextractor entscheiden könnte, den hier nun gefundenen Zusammenhang zu verwerfen. Eine Lösung könnte mit Hilfe von *spaCy* erfolgen: der Text lässt sich nämlich auch mit der normale *spaCy*-Pipeline analysieren. Das Ergebnis der grammatikalische Analyse ist in Tabelle 5.5 wiedergegeben.

In der Spalte *Text* befinden sich die einzelnen Wörter des zu analysierenden Satzes. Als *Lemma* wird das Grundwort verstanden, also z.B. der Singular (*substance*) eines im Plural stehenden Begriffs (*substances*). Das Kürzel *POS* steht für *Part of Speech Tag*. Dies entspricht der Wortart der einzelnen Begriffe. Die grobe Einordnung der Wortarten steht in der Spalte *POS*, während in der Spalte *Tag* eine genauere Bestimmung der Wortart eingetragen ist. Dazu gehört u.a. bei Substantiven die Unterscheidung zwischen Singular und Plural sowie bei Verben die Zeitform.

An dieser Stelle kann bereits festgehalten werden, dass die Wortart des Verbs *aggravate* von *spaCy* richtig erkannt wird. Der Tag *VB* drückt aus, dass das Verb in der Infinitiv-Form vorliegt. In der Spalte *DEP* werden Abhängigkeiten der Wörter untereinander charakterisiert. So wird *aggravate* als zentra-

les Verb (*root*) des Satzes identifiziert. Mit *nsubj* wird das Subjekt des Satzes markiert, hier also der Begriff *substances*.

Eine Verbesserung des Medextractors könnte nun dadurch erfolgen, dass die Sätze, in denen Symptome und Krankheiten gefunden wurden, zusätzlich durch die *spaCy*-Pipeline analysiert werden und einfache Verben, die sich im Trainingsvokabular befinden, ignoriert werden.

5.3.2.2 *Generische Überbegriffe*

Im Trainingsvokabular finden sich generische Überbegriffe wie z.B. *ailments* (Beschwerden, Krankheiten), die ebenfalls weder als konkrete Krankheit oder Symptom taugen. Im folgenden Satz wird *ailments* als Symptom einer *anxiety disorder* gefunden:

„Depression often occurs in children ... experiencing loss, or having ... anxiety disorders and other chronic physical ailments."

Wie könnte hier nun der Medextractor entscheiden, ob *ailments* ein konkreter oder doch eher ein generischer Begriff ist? Der Dependency-Parser von *spaCy* erkennt, dass in diesem Satz der Begriff *ailments* in einer Konjunktion zum Begriff *disorder* steht und dass das Wort *other* zu *ailments* gehört. Der Medextractor könnte z.B. an dem Wort *other* erkennen, dass der Begriff *ailments* wahrscheinlich eher generischer Natur ist.

5.3.2.3 *Mehrdeutige Begriffe*

Einige Begriffe im Trainingsvokabular sind mehrdeutig und werden daher mitunter irrtümlich als Krankheit oder Symptom gedeutet. Dies ist z.B. für den Begriff *cut* in folgendem Satz der Fall:

„A dependent drinker usually experiences physical and psychological withdrawal symptoms if they suddenly cut down or stop drinking."

Withdrawal symptoms (Entzugserscheinungen) werden daher falsch als Symptom von Schnitt(wunden) erkannt. In diesem Fall kann erneut der Ansatz helfen, keine Verben als Krankheiten zu akzeptieren. Auch wenn *cut* sowohl Nomen als auch Verb sein kann, ist *spaCy* in der Lage, zu erkennen, dass *cut* in diesem Satz ein Verb ist.

Ähnlich verhält es sich mit dem Begriff *cold*. Der Medextractor findet in folgendem Satz fälschlicherweise den Zusammenhang *cold - physical signs*:

„You may also notice physical signs and symptoms such as feeling cold, dizzy or very tired."

Dabei deutet der Medextractor *cold* (Erkältung) als Krankheit und *physical signs* als Symptom. Durch Analyse mit *spaCy* lässt sich bestimmen, dass *cold* in diesem Satz ein Adjektiv ist, während *spaCy* etwa in dem Satz *„I have a*

bad cold.“ das Wort *cold* als Nomen identifiziert. Ähnlich wie Verben, sollten einzelne Adjektive nicht als Krankheit gewertet werden.

5.3.2.4 *Adjektive*

Bei Symptomen können Adjektive nicht ohne weiteres ignoriert werden. Allerdings kommen sie, sofern sie ein Symptom beschreiben, häufig mit einem Verb wie *feeling* vor, also etwa *feeling cold*, *feeling dizzy* oder *feeling very tired*. Tatsächlich enthält das Trainingsvokabular zahlreiche Einträge wie z.B. *feeling angry*, *feeling helpless* oder *feeling bad*. Allerdings fehlen die zuvor genannten und in dem Beispielsatz vorhandenen Ausdrücke.

Es ist daher sinnvoll, alle Adjektive, die mit Verben wie z.B. *feeling* eingeleitet werden, als Symptom zu deuten. Abbildung 5.1 zeigt graphisch das Ergebnis der Analyse durch die *spaCy*-Pipeline und zeigt, dass der Dependency Parser einerseits *feeling* und *cold* als zusammengehörig identifiziert (acomp = adjectival complement) und auch erkennt, dass die weiteren Adjektive *dizzy* und *tired* ebenfalls über (grammatikalische) Konjunktionen mit dem Verb *feeling* verbunden sind.

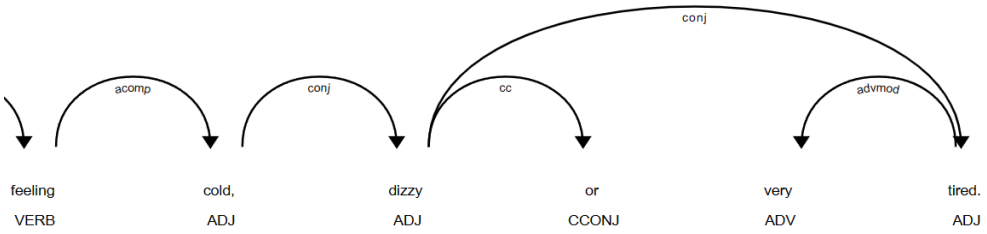

Abbildung 5.1: Dependency Parser

Es eröffnet sich damit eine einfache Methode, auch Symptome, die nicht im Trainingsvokabular enthalten sind, von *spaCy* als Symptome erkennen zu lassen.

5.3.2.5 *Negationen*

In folgendem Satz wird fälschlich vom Medextractor erkannt, dass Interesse ein Symptom einer Depression ist:

„The psychological symptoms of depression include having no motivation or interest in things.“

Dies liegt daran, dass der Medextractor nicht aus dem Zusammenhang erkennt, dass *interest* in diesem Fall negiert ist, auch wenn das *no* lediglich vor *motivation* steht.

Abbildung 5.2 zeigt, dass auch hier der Dependency Parser von *spaCy* gute Dienste leisten kann. Er erkennt, dass *motivation* und *interest* über eine Konjunktion miteinander verbunden sind und dass *no* damit sowohl die Begriffe motivation als auch *interest* näher bestimmt.

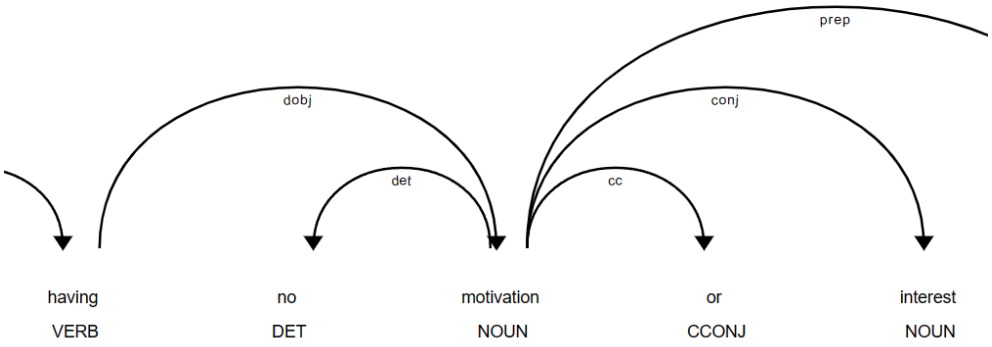

Abbildung 5.2: Dependency Parser bei Negation

5.3.2.6 *Korrelation zwischen Krankheit und Symptom*

Kann von einer Krankheit relativ sicher auf Symptome geschlossen werden, so ist der umgekehrte Schluss von Symptom auf Krankheit häufig jedoch sehr unsicher. Ein Beispiel ist der Begriff *anxiety* (Angst / Besorgnis). Bei der Analyse des Handbuchs findet der Medextractor diesen Begriff im Zusammenhang mit über 70 Krankheiten / Störungen. Dieser Umstand ermöglicht es einer Software, automatisch zu erkennen, dass ein Symptom oder Befund keinen sicheren Rückschluss auf eine Krankheit erlaubt. Auch kann durch Auswertung der Häufigkeit ein Modell entwickelt werden, um die Aussagekraft eines Befund zu bewerten.

Problematischer ist es aber z.B. mit dem Zusammenhang *agoraphobia - chest pain*, der in dem Satz:

„The physical symptoms of agoraphobia can be similar to those of a panic attack and may include chest pain.“

gefunden wird. Da nur Texte über psychische Erkrankungen untersucht wurden, erscheinen Brustschmerzen nun als eindeutig der Agoraphobie (Angst vor Menschenansammlungen) zugeordnet zu sein. Würde man auch andere medizinische Texte untersuchen, so würde deutlicher werden, dass Brustschmerzen allein keinen deutlichen Hinweis auf eine Agoraphobie geben. Es erscheint daher sinnvoll, den Umfang analysierter Texte auf allgemeine medizinische Texte zu erweitern, auch wenn ein Projekt sich nur auf psychologische Themen beschränkt.

In vielen Fällen kann ein voreiliger Rückschluss von Symptom auf Krankheit auch zu Stigmatisierungen führen. So ist zwar ein dementer Mensch oft alt, alte Menschen dürfen aber nicht pauschal für dement gehalten werden. Da der Begriff *aging* im Symptom-Vokabular enthalten ist, findet der Medextractor in dem Satz:

„But dementia is not a normal part of aging.“

den Zusammenhang *dementia - aging*. Ähnlich verhält es sich mit dem Satz:

„...alcohol misuse can lead to social problems ... such as unemployment ...“,

in dem der Zusammenhang *alcohol misuse - unemployment* gefunden wird. Auch wenn Alter und (Langzeit-)Arbeitslosigkeit keine unbegründeten Indikatoren für psychische oder mentale Probleme sind, muss darauf geachtet werden, dass eine Software nicht stigmatisierend agiert. Hier kann wahrscheinlich nicht auf ein automatisches Erkennen gesetzt werden, sondern die Software-Entwickler müssen sicherstellen, dass ihre Software mit Begriffen wie Arbeitslosigkeit und Alter angemessen umgeht. Im Fall des Medextractors könnten solche Begriffe z.B. manuell aus dem Vokabular entfernt oder durch Einführung einen Attributes als problematisch markiert werden.

5.3.2.7 *Bedeutung von Aussagen*

Die einfache Auswertung der gefundenen Entitäten ermöglicht es dem Medextractor nicht, die Bedeutung von Sätzen richtig zu erfassen. In dem Artikel auf Wikipedia findet sich z.B. die Aussage:

„These challenges came ... from gay rights activists who criticised the APA's listing of homosexuality as a mental disorder.“

Hier wird, da auch in dem MetaMapLite Vokabular *homosexuality* als Befund - aber nicht als Krankheit/Störung - enthalten ist, ein Zusammenhang zwischen Homosexualität und einer geistigen Störung gefunden. Dies entspricht aber natürlich nicht der Aussage des Satzes. Um die Bedeutung von Aussagen zu erfassen, ist ein höherer Aufwand zu treiben als in den bisherigen Beispielen. Ein Ansatz könnte sein, in den bisher gefundenen Beispielsätzen nach Satzmustern zu suchen, mit denen typischerweise Symptome beschrieben werden. Betrachten wir hierzu den Satz

„Other typical symptoms of severe depression are changes in appetite.“

Hier wird *changes in appetite* als Symptom und *severe depression* als Krankheit gefunden. Dieser Satz beschriebe auch das Symptom einer Krankheit, wenn die Begriffe *change of appetite* und *severe depression* durch andere Begriffe aus dem Trainingsvokabular ersetzt würden. Allerdings sind auch viele Abwandlungen dieses Satzes denkbar und es sollte die Zielsetzung verfolgt werden, nach Möglichkeit nicht alle diese Abwandlungen trainieren zu müssen.

Mit Hilfe der *spaCy*-Pipeline können Sätze auf ihren Kerninhalt reduziert werden. Dazu wird ein Satz zunächst durch die Pipeline analysiert (s. Tabelle 5.6). In obigem Beispiel wird als zentrales Verb (root) an Position 6 das Wort *are* gefunden. *spaCy* bietet Methoden, mit denen Eltern und Kinder von einzelnen Wörtern gefunden werden können. Die Kinder des Wortes *are* sind z.B. *symptoms* und *changes* (Abbildung 5.3).

Text	Lemma	POS	TAG	DEP
Other	other	ADJ	JJ	amod
typical	typical	ADJ	JJ	amod
symptoms	symptom	NOUN	NNS	nsubj
of	of	ADP	IN	prep
severe	severe	ADJ	JJ	amod
depression	depression	NOUN	NN	pobj
are	be	AUX	VBP	ROOT
changes	change	NOUN	NNS	attr
in	in	ADP	IN	prep
appetite	appetite	NOUN	NN	pobj
.	.	PUNCT	.	punct

Tabelle 5.6: Ergebnis der *spaCy*-Pipeline

```
doc = nlp('Other typical symptoms of severe depression are changes in appetite.')
print(doc[6].text, list(doc[6].children))
print(doc[2].text, list(doc[2].rights))
print(doc[3].text, list(doc[3].children))
```

```
are [symptoms, changes, .]
symptoms [of]
of [depression]
```

Abbildung 5.3: Dependency Parser Ausgabe

Man kann nun einen typischen Satz folgendermaßen charakterisieren:

1. Die Wurzel (root) des Satzes muss entweder das Verb *is* oder *are* sein.
2. Eines der Kinder der Wurzel muss als Lemma den Begriff *symptom* besitzen.
3. Das andere Kind[1] muss ein Begriff aus dem Symptom-Vokabular sein (ggf. zusammengesetzt).
4. Rechts neben dem Begriff mit Lemma *Symptoms* steht das Wort *of*.
5. Das Kind des Begriffs *of* muss ein Begriff aus dem Krankheiten-Vokabular sein (ggf. zusammengesetzt)

Auf diese Weise können auch andere Satzkonstrukte festgelegt werden, mit denen üblicherweise die Symptome von Krankheiten beschrieben werden. Vorteil der bisher sehr einfachen Logik des Medextractors ist, dass alle Beziehungen von Krankheiten und Symptomen in einem Text gefunden werden, sofern die Begriffe im Vokabular enthalten sind. Die vom Medextractor gefundene Sammlung von Beispielsätzen ist daher eine gute Grundlage, um typische Sätze finden zu können.

1 Es kann sich auch um mehrere Kinder (also Symptome) handeln.

5.4 ZUSAMMENFASSUNG

6 ZUSAMMENFASSUNG UND AUSBLICK