

Wissensbasierte Identifikation von Wertebereichen einer aktiven Ontologie

Bachelorarbeit
von

Yauhen Makhotsin

An der Fakultät für Informatik
Institut für Programmstrukturen
und Datenorganisation (IPD)

Erstgutachter:	Prof. Dr. Walter F. Tichy
Zweitgutachter:	Prof. Dr. Ralf Reussner
Betreuender Mitarbeiter:	Dipl.-Inform. Martin Blersch

Bearbeitungszeit: 10.04.2017 – 08.09.2017

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Die Regeln zur Sicherung guter wissenschaftlicher Praxis im Karlsruher Institut für Technologie (KIT) habe ich befolgt.

Karlsruhe, 08.09.2017

.....
(Yauhen Makhotsin)

Kurzfassung

Die Steuerung von Programmen mithilfe der natürlichen Sprache gehört heutzutage zur Funktionalität vieler Geräte. Durch die Sprachsteuerung ist eine erleichterte und intuitive Bedienung der Endgeräte möglich. Die erleichterte Bedienung hat eine positive Benutzererfahrung zur Folge. Die Grundlage hierfür bildet das korrekte Auslesen eines gewünschten Befehls sowie der Eingabedaten aus einem Ausdruck der natürlichen Sprache. Eine der Möglichkeiten zur Abbildung eines textuellen Befehls stellen aktive Ontologien dar, ein Sprachverarbeitungsmechanismus, der im Programm Siri von Apple umgesetzt ist.

Das Projekt EASIER befasst sich mit der semi-automatischen Kategorisierung, Zusammenfassung und anschließender Abbildung von Webformularen der Dienstanbieter auf aktive Ontologien. Dafür werden Konzepte und Relationen erkannt und eine aktive Ontologie erstellt. Da die Wertebereiche von Konzeptnoten dabei oft nicht in diesen Formularen enthalten sind, wird eine externe Datenquelle benötigt, um diese zu ermitteln.

Im Rahmen dieser Bachelorarbeit werden die Möglichkeiten zur Befragung der unterschiedlichen Datenquellen zur Ermittlung der Wertebereiche untersucht. Das Ergebnis dieser Arbeit stellt eine Software dar, die als Eingabe Werte der HTML-Attribute von Webformularelementen bekommt und durch Anfragen an externe Datenquellen mögliche Wertebereiche eines Konzeptknotens ermittelt.

Hierfür werden die übermittelten Attributwerte der Webformularelemente im ersten Schritt als mögliche Eingabewerte oder Oberbegriffe der Eingabewerte klassifiziert. Folgend werden externe Datenquellen befragt, um weitere Eingabewerte und Oberbegriffe zu ermitteln. Als externe Datenquellen dienen Wikipedia, ResearchCyc und WordNet.

Inhaltsverzeichnis

1. Einleitung	1
1.1. Zielsetzung	2
1.2. Beispiel der Ausgabe	2
1.3. Struktur der Arbeit	3
2. Grundlagen	5
2.1. Ontologie	5
2.2. Aktive Ontologien	5
2.2.1. Wissensdarstellung einer aktiven Ontologie	6
2.2.2. Bearbeitung der Eingabe in einer aktiven Ontologie	6
2.2.3. Fakten	6
2.2.4. Unifikation	7
2.2.5. Faktenspeicher und Auswertungszyklus	8
2.2.6. Kommunikationskanal	9
2.3. Active Semantic Network	9
2.3.1. Aufbau	9
2.3.1.1. Sensorknoten	9
2.3.1.2. Kombinationsknoten und Selektionsknoten	10
2.3.2. Beispiel der Ausführung	10
2.4. Webformulare	11
2.5. Externe Datenquellen	12
2.5.1. Wikipedia	12
2.5.2. WordNet	13
2.5.3. Cyc	13
2.6. Zusammenfassung	14
3. Verwandte Arbeiten	15
3.1. Ontologierstellung aus unstrukturierten Daten	15
3.1.1. Konstruierung einer biomedizinischen Datenbank aus dem Textkorpus von PubMed	15
3.1.1.1. Hintergrund	16
3.1.1.2. Ansatz	16
3.1.1.3. Fazit	17
3.2. Ontologierstellung aus semistrukturierten Daten	17
3.2.1. Ontologisches Lernen durch Web-Crawling	17
3.2.1.1. Hintergrund	17
3.2.1.2. Ansatz	18
3.2.1.3. Fazit	18
3.2.2. Konstruktion des Konzeptwissens aus HTML Seiten	19
3.2.2.1. Ansatz	19
3.2.2.2. Fazit	20

3.3.	Ontologierstellung aus strukturierten Daten	20
3.3.1.	Konstruktion einer domänenspezifischen Datenbank der Software- technik aus Wikipedia und StackOverflow	21
3.3.1.1.	Ansatz	21
3.3.1.2.	Evaluation	22
3.3.1.3.	Fazit	22
3.3.2.	Integration des Inhalts von Wikipedia in Cyc	23
4.	Analyse	27
4.1.	Analyse der Eingabedaten	27
4.1.1.	Semantik in HTML-Formularen	27
4.1.2.	Klassifizierung der Begriffe	29
4.1.3.	Lexikalische Analyse	29
4.1.4.	Eingabe	29
4.2.	Ermittlung weiterer Begriffe	30
4.2.1.	Vorverarbeitung der Daten	30
4.2.2.	Befragung der Datenquellen	32
4.2.2.1.	Abfrage von WordNet	32
4.2.2.2.	Abfrage von Wikipedia	35
4.2.2.3.	Abfrage von Cyc	37
4.2.3.	Konsolidierung der Ergebnisse	38
5.	Entwurf und Implementierung	41
5.1.	Entwurf	41
5.1.1.	Vorbereitung der Daten	42
5.1.2.	Befragung der Datenquellen und Zusammenführung der Ergebnisse .	43
5.1.3.	Weitere Komponenten	43
5.1.4.	Zusammenfassung	43
5.2.	Implementierung	44
5.2.1.	Vorbereitung der Daten	44
5.2.2.	Befragung der Datenquellen	44
5.2.2.1.	WordNet	45
5.2.2.2.	Wikipedia	45
5.2.2.3.	Cyc	45
5.2.3.	Weitere Komponenten	46
6.	Evaluation	47
6.1.	Konfigurationsparameter	47
6.2.	Testfälle	47
6.2.1.	Flugticketanbieter	48
6.2.2.	Bahnhaltestellen	48
6.2.3.	Hotelbuchung	49
6.3.	Fazit	49
7.	Zusammenfassung und Ausblick	51
7.1.	Zusammenfassung	51
7.2.	Ausblick: weitere Datenquellen	52
7.3.	Ausblick: Konfigurationsparameter	52
	Literaturverzeichnis	53
	Anhang	57
A.	Testfall 1. Flugticketanbieter.	57

B.	Testfall 2, Haltestellen	62
C.	Testfall 3, Hotels	67
D.	Semantische Ähnlichkeit nach Wu-Palmer	68
E.	Kategorien der Artikel in Wikipedia	69

Abbildungsverzeichnis

1.1.	Der Einbau der zu erstellenden Software in das Projekt EASIER	2
2.1.	Beispiel einer Ontologie (Darstellung als Graph)	6
2.2.	Grafische Darstellung einer aktiven Ontologie [Guz08]	7
2.3.	Beispiel der Verarbeitung durch ein semantisches Netzwerk	11
2.4.	Ein Beispiel eines Artikels auf Wikipedia	13
2.5.	Kategorienliste unter dem Artikel	13
2.6.	Beispiel der Hierarchie von Substantiven in WordNet [Erc06]	14
3.1.	Die Struktur des erstellten Systems [BK05]	19
4.1.	Die Struktur der Eingabedatei	30
4.2.	Suche nach den Kindern eines Oberbegriffs. Das Rechteck schließt die ausgegebenen Domänenwerte um	33
4.3.	Suche nach dem kleinsten gemeinsamen Oberbegriff. Das Rechteck schließt die ausgegebenen Domänenwerte um	33
4.4.	Suche nach dem kleinsten gemeinsamen Oberbegriff. Der Oberbegriff „Object“ ist zwar richtig, aber zu abstrakt, um behilflich zu sein	34
4.5.	Suche nach den Artikeln in Wikipedia und Kategorien dieser Artikeln bei der Eingabe eines Wortes. In Klammern ist die jeweilige Wahrscheinlichkeit.	36
5.1.	Allgemeiner Ablauf des Programms	41
5.2.	Klassendiagramm der Knotenerstellung	42
5.3.	Klassendiagramm der Klassifizierung von Begriffen	42
5.4.	Klassendiagramm der Befragung der Datenquellen	43
5.5.	Aufbau des Programms	44
5.6.	Beispiel der Eingabedatei	45
5.7.	Beispiel der Ausgabedatei	46

Tabellenverzeichnis

1.1. Beispiele von Wertebereichen der Webformularelemente	2
1.2. Beispiel der Ausgabe	3
2.1. Arten von Fakten in Active [Guz08]	7
2.2. Unifikation einfacher Fakten in Active	7
2.3. Unifikation komplexer Fakten in Active [Guz08]	8
2.4. Unifikation von Listenfakten in Active [Guz08]	8
2.5. Unifikation von Listenfakten in Active [Guz08]	8
2.6. Attribute eines Textfeldes	12
3.1. Morpho-syntaktische Muster	18
4.1. Wahrscheinlichkeiten des Auftretens der Oberbegriffe und Domänenwerte .	29
4.2. Verwendete Muster	29
4.3. Verwendete Präfixe	31
6.1. Trefferquoten bei Testfall 1, Flugticketanbieter	48
6.2. Trefferquoten bei Testfall 2, Bahnhaltestellen	49
6.3. Trefferquoten bei Testfall 2, Bahnhaltestellen	49
A.1. Testfall 1, Flugticketanbieter: Treffer 1 - 25 (Fehlkonzepte rot markiert) . .	59
A.2. Testfall 1, Flugticketanbieter: Treffer 26 - 50 (Fehlkonzepte rot markiert) .	60
A.3. Testfall 1, Flugticketanbieter: Treffer 51 - 75 (Fehlkonzepte rot markiert) .	61
A.4. Testfall 1, Flugticketanbieter: Treffer 76 - 100 (Fehlkonzepte rot markiert)	62
B.5. Testfall 2, Bahnhaltestellen: Treffer 1 - 25 (Fehlkonzepte rot markiert) . . .	64
B.6. Testfall 2, Bahnhaltestellen: Treffer 26 - 50 (Fehlkonzepte rot markiert) . .	65
B.7. Testfall 2, Bahnhaltestellen: Treffer 51 - 75 (Fehlkonzepte rot markiert) . .	66
B.8. Testfall 2, Bahnhaltestellen: Treffer 76 - 100 (Fehlkonzepte rot markiert) .	67
D.9. Semantische Ähnlichkeit nach Wu-Palmer	68
E.10. Kategorien der Artikel in Wikipedia	69

1. Einleitung

Grafische Benutzerschnittstellen erlauben selbst unerfahrenen Benutzern eine unkomplizierte Bedienung ihres PCs oder Smartphones. In vielen Situationen wäre es leichter, bequemer und gegebenenfalls schneller, ein Gerät mithilfe der natürlichen Sprache zu steuern. Zusätzlich erlaubt die Sprachsteuerung neue und barrierefreie Benutzungsmöglichkeiten von Geräten aller Art. Um eine Pizza zu bestellen, muss der Benutzer erst ein Restaurant aussuchen und die Bestellung über ein Webformular durchführen (Benutzeraufwand) oder mittels eines Telefonats (Mitarbeiteraufwand). Viel bequemer und mit deutlich geringerem Aufwand wäre es, wenn der Benutzer einer auf seinem Handy oder Rechner installierten Software in natürlicher Sprache seinen Wunsch ausdrücken könnte und der Rest von dieser Software übernommen werden würde. Verschiedene Sprachassistenten wie Alexa, Cortana oder Google Now stehen den Benutzern dafür zur Verfügung.

Um aus einem Satz die notwendige Information zu extrahieren und die entsprechenden Dienste aufzurufen, werden in der Software Siri von Apple aktive Ontologien verwendet. Eine aktive Ontologie verfügt über einen internen Sprachverarbeitungsmechanismus, mit dem aus einem Satz Instanzen von Konzepten erkannt und auf Knoten der aktiven Ontologie (AO-Knoten) abgebildet werden können. Voraussetzung hierfür ist, dass die Konzepte, Relationen zwischen den Konzepten und die möglichen Instanzen zuerst von den Anwendungsentwicklern erstellt wurden.

Da die Erstellung von aktiven Ontologien in der Regel manuell erfolgt, ist die Verwendung der ontologiebasierten Software nur auf die von Entwicklern vorgesehenen Anwendungsfällen begrenzt. Das Projekt „Easier“ [Ble16] setzt sich mit der semi-automatischen Generierung von aktiven Ontologien auseinander. Dafür werden zuerst Webformulare verschiedener Dienstleister in Kategorien zusammengefasst. Aus jeder Kategorie wird eine aktive Ontologie erstellt, indem ähnliche Webformularelemente auf AO-Knoten abgebildet werden. Die zusammengefassten Wertebereiche dieser Webformularelemente stellen den Wertebereich vom AO-Knoten dar.

Wenn der Wertebereich eines Elements nicht im HTML-Code enthalten ist, muss dieser vom Benutzer eingegeben werden. In dieser Arbeit wird die Möglichkeit zur Automatisierung dieses Vorgehens untersucht und die gefundene Lösung wird in einem Softwareprogramm implementiert.

1.1. Zielsetzung

Der Prozess der Erzeugung von aktiven Ontologien im Projekt EASIER läuft in mehreren Schritten ab.

Zuerst werden eingegebene Webformulare nach dem Anwendungsfall klassifiziert. Danach werden die Formulare jeder Kategorie in ein übergeordnetes Formular zusammengeführt, welches als eine XML-Datei realisiert ist. Dabei werden Elemente, die eine gleiche Eingabe erwarten (und später dem gleichen Sensor-knoten der AO entsprechen werden) jeweils als ein Element dargestellt. Die Vereinigung der aus dem HTML-Code bekannten Wertebereiche (z.B. „Option“-Felder einer Dropdown-Liste) stellen den Wertebereich des Endelements im übergeordneten Formular dar.

Die resultierende XML-Datei wird an den AO-Generator übergeben, um eine aktive Ontologie zu erstellen. Aktuell sind die Wertebereiche von Eingabe-elementen, deren Werte im HTML-Code nicht definiert sind, durch manuelle Eingabe zu bestimmen.

Wie in der Tabelle 1 zu sehen ist, enthält der HTML-Code eines Webformulars nicht immer den Wertebereich der Eingabe. Diese Information ist nicht im HTML-Code enthalten, und eine zusätzliche Datenquelle ist daher erforderlich, um eine aktive Ontologie zu generieren.

Nr	Elemententyp	Wertebereich
1	Select	Vordefinierte Optionen
2	Checkbox	Wahr / Falsch
3	datetime-local	Daten
4	text	Unbekannt

Tabelle 1.1.: Beispiele von Wertebereichen der Webformularelemente

Das Ziel dieser Bachelorarbeit ist die Entwicklung einer Software, die basierend auf den Attributwerten eines Eingabe-elementes den entsprechenden Wertebereich bestimmen kann, um somit den Grad an Automatisierung im Projekt EASIER zu erhöhen. Die Werte dafür können von externen Datenquellen abgefragt werden. Der Einbau dieser Bachelorarbeit in das Projekt EASIER ist in der Abbildung 1.1 veranschaulicht.

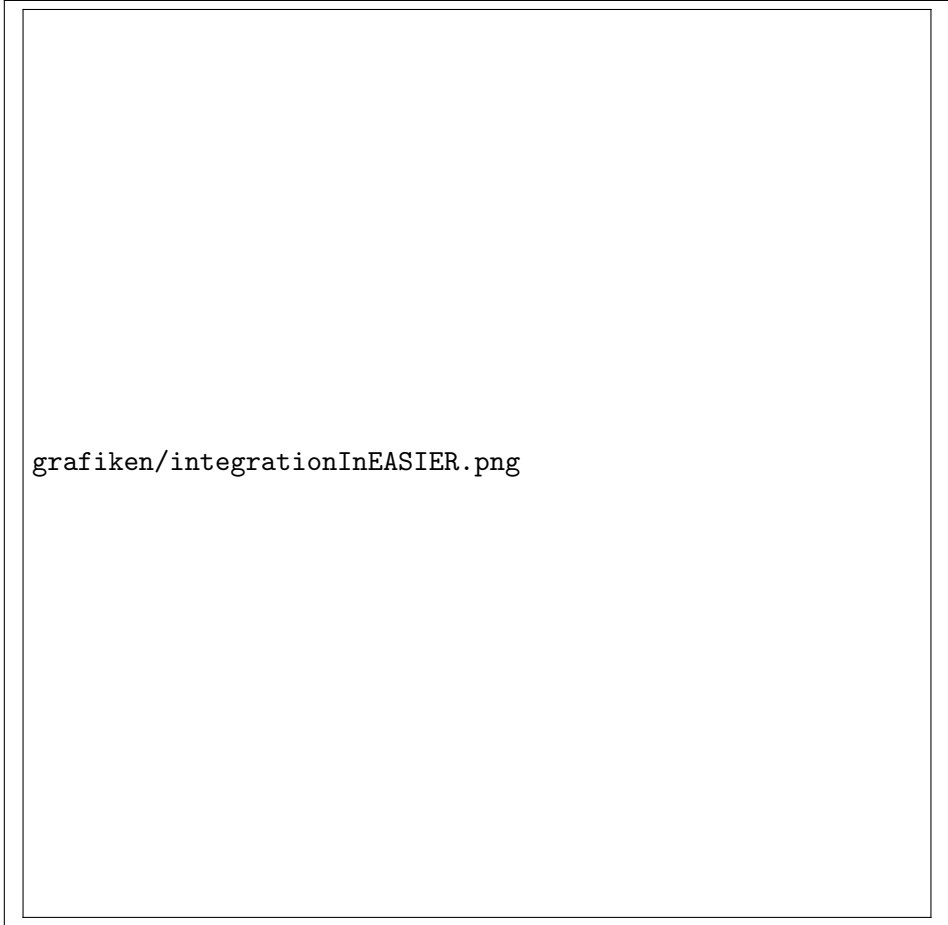
1.2. Beispiel der Ausgabe

Zur Buchung von Flugtickets über ein Webformular muss der Benutzer den Zielort eingeben. Als Eingabe werden eine Stadt, der Name oder die Kodierung eines Flughafens erwartet. Die Eingabedatei stellt somit eine Menge der Attributwerte von den Textfeldern, in die diese Informationen einzugeben sind. Eine Beschreibung dieses Testfalls befindet sich im Kapitel 6. In der Tabelle 1.2 sind die ersten zehn Ausgabewerte dargestellt.

Wie in der Tabelle 1.2 zu sehen ist, ist in diesem Fall eine Trefferquote von 90% unter den ersten 10 Ergebnissen erreicht. Dabei wurde ein Fehlkonzepit ausgegeben („Old Country“). Die Begriffe „Motor City“ und „Windy City“ bezeichnen im englischsprachigen Raum Städte Detroit und Chicago und zählen somit zu den richtigen Domänenwerten.

1.3. Struktur der Arbeit

Die vorliegende Arbeit ist in sieben Kapitel unterteilt. Zu Beginn der Arbeit wird dem Leser im Kapitel „Grundlagen“ ein Überblick der Begriffe und Methoden verschafft, die zum Verständnis dieser Arbeit erforderlich sind. Im Kapitel „Verwandte Arbeiten“ werden bereits bestehende wissenschaftliche Arbeiten besprochen, in denen ähnliche Ziele wie



grafiken/integrationInEASIER.png

Abbildung 1.1.: Der Einbau der zu erstellenden Software in das Projekt EASIER

das Ziel dieser Arbeit gesetzt wurden. Im Kapitel „Analyse und Entwurf“ werden ein Lösungsansatz entwickelt und ein Entwurf der zu erstellenden Software erstellt. Das Kapitel „Implementierung“ befasst sich mit den Implementierungsdetails des erstellten Programms. Die Bewertung der Ergebnisse erfolgt im Kapitel „Evaluation“. Abschließend wird im Kapitel „Zusammenfassung und Ausblick“ das Ergebnis der Arbeit zusammengefasst und mögliche Ausblicke dargestellt.

Nummer	Gefundener Wert
1	New York City
2	Kansas City
3	Atlantic City
4	Jersey City
5	Vatican City
6	Motor City
7	Windy City
8	Old Country
9	Lexington
10	Ho Chi Minh City

Tabelle 1.2.: Beispiel der Ausgabe

2. Grundlagen

Dieses Kapitel der Arbeit gibt einen Überblick über die theoretischen Grundlagen und Ansätze, die zum Verständnis dieser Arbeit erforderlich sind. Zuerst werden der Begriff „Ontologie“ und seine Verwendung in der Informatik erläutert, um dem Leser das Verständnis vom Konzept der aktiven Ontologien zu erleichtern. Folgend werden aktive Ontologien selbst, sowie das Active Semantics Network erklärt, die den Kern des Projektes EASIER darstellen. Anschließend werden verschiedene Wissensdatenbanken beschrieben, die als externe Datenquellen für die Erstellung der zu implementierenden Software verwendet wurden.

2.1. Ontologie

Der Ursprung des Wortes „Ontologie“ liegt im Bereich der Philosophie und bedeutet eine systematisierte Darstellung der Existenz [G⁺93]. In der Informatik findet dieser Begriff oft seine Anwendung in den Bereichen der künstlichen Intelligenz [Jon15], Computerlinguistik [OVQH13] und Data Mining [DWL15].

Eine Ontologie ist in der Informatik eine Menge von Konzepten und Relationen zwischen diesen Konzepten, die sich innerhalb einer vordefinierten Domäne befinden. Eine formale Darstellung einer Ontologie wäre $O = (C, R, A, \text{Top})$, dabei sind C eine nicht-leere Menge von Konzepten, R eine Menge von Relationen zwischen zwei oder mehreren Konzepten aus C , A eine Menge von Axiomen innerhalb der Domäne und Top das oberste Konzept der Ontologie [HEBR11].

Dabei sind Ontologien von den grundlegenden Datenstrukturen und Implementierungsstrategien unabhängig und stellen das Wissen über Objekte und ihre Attribute sowie Beziehungen zu anderen Objekten dar. Eine Ontologie wird in der Regel als ein Graph dargestellt, bei dem jeder Knoten einem Konzept entspricht und Relationen durch Kanten dargestellt werden (Abbildung 2.1).

2.2. Aktive Ontologien

Die Entwicklung einer ontologiebasierten Software ist aufwendig und setzt umfangreiche theoretische Kenntnisse vom Entwickler voraus [GBCC07]. Eine der möglichen Lösungen dafür ist die Verwendung von aktiven Ontologien, dessen Konzept in der Dissertation von Didier Guzzoni „Active: A Unified Platform for Building Intelligent Applications“ dargestellt ist [Guz08].

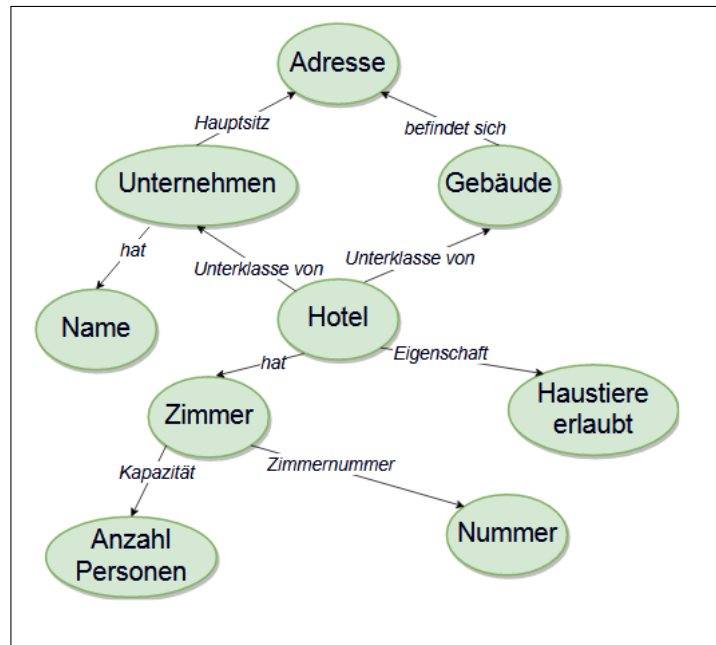


Abbildung 2.1.: Beispiel einer Ontologie (Darstellung als Graph)

Eine Ontologie ist lediglich eine Datenstruktur zur Darstellung der Konzepte und Relationen. Eine aktive Ontologie besitzt zusätzlich eine Menge der Verarbeitungsmechanismen zur Erkennung von Konzepten in einer Eingabe und Abbildung von diesen Konzepten auf Knoten der Ontologie. Somit stellen aktive Ontologien eine Erweiterung der Ontologien dar [Guz08]. In dieser Bachelorarbeit wird unter dem Begriff „aktive Ontologie“ eine Ontologie mit den Verarbeitungsmechanismen des Frameworks Active verstanden.

2.2.1. Wissensdarstellung einer aktiven Ontologie

Eine aktive Ontologie ist eine Erweiterung von Ontologien und erbt somit die Merkmale von Ontologien. Das Domänenwissen wird gleichermaßen durch Konzepte, Relationen und eine Menge von Axiomen (auch Fakten genannt) dargestellt. Relationen sind gerichtete Verbindungen zwischen einem Quell- und einem Zielkonzept. Es gibt verschiedene Arten von Attributen, z.B. „is a“-Relationen zwischen Konzepten und Oberbegriffen dieser Konzepte. Zudem können Relationen ein oder mehrere Attribute enthalten. Beispielsweise enthält eine „member of“-Relation das Attribut „is single“, das angibt, ob ein Konzept ein Mitglied von einem oder mehreren Konzepten ist [Guz08].

2.2.2. Bearbeitung der Eingabe in einer aktiven Ontologie

Konzepte einer aktiven Ontologie sind durch einen eindeutigen Namen definiert und beinhalten zudem eine Menge von Regeln zur Verarbeitung von Daten. Eine Regel besteht dabei aus einer Vorbedingung und einer Aktion, die durchgeführt wird, falls die Vorbedingung erfüllt ist. Eine graphische Darstellung ist in der Abbildung 2.2 zu finden.

2.2.3. Fakten

Eine der grundlegenden Datenstrukturen sind Fakten. Sie werden unter allem verwendet, um die Kommunikation zwischen Konzepten zu gewähren und den aktuellen Verarbeitungszustand zu speichern. Fakten haben einen Zyklus, in dem sie erstellt, verwendet und gelöscht werden. In Active gibt es 4 Arten von Fakten, die in der Tabelle 2.1 beschrieben sind.

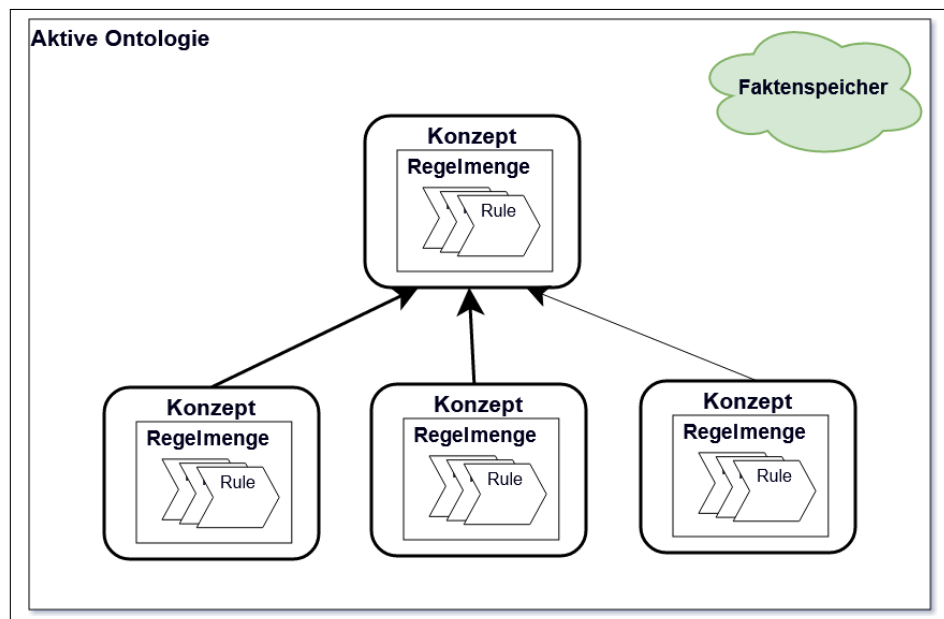


Abbildung 2.2.: Grafische Darstellung einer aktiven Ontologie [Guz08]

Faktenart	Beispiel	Bemerkung
Einfache Fakten	„Albert Einstein“, 1, a	Konstante Werte
Variablen	\$person, \$occupation	Stellen beliebige Fakten dar
Komplexe Fakten	Occupation(„Albert Einstein“, „genius“)	Prädikate mit einem oder mehreren Attributen
Listenfakten	[„simple fact“, [„this“, „is“, „a“ , „list“]]	Listen von Fakten

Tabelle 2.1.: Arten von Fakten in Active [Guz08]

2.2.4. Unifikation

Unter Unifikation wird der Vergleich von zwei Fakten verstanden. Da eine Vorbedingung die gleiche Form wie ein Fakt hat, können Fakten mit Vorbedingungen unifiziert werden. Das Ergebnis der Unifikation zwischen einem Fakt und einer Vorbedingung ist ein boolescher Wert, der angibt, ob dieser Fakt die Bedingung erfüllt oder nicht.

Zwei einfache Fakten sind nur dann unifizierbar, wenn sie identisch sind. Ein Beispiel hierfür ist in der Tabelle Tabelle 3.1 dargestellt.

Fakt 1	Fakt 2	unifizierbar
a	a	wahr
a	b	falsch
'Berlin'	'Berlin'	wahr
'Frankfurt'	'Frankfurt am Main'	falsch

Tabelle 2.2.: Unifikation einfacher Fakten in Active

Bei komplexen Fakten müssen die Prädikaten identisch sein, und alle Attribute des ersten Fakttes müssen mit den Attributen des zweiten Fakttes in der gleichen Reihenfolge unifizieren. Tabelle 2.3

Zur Unifikation der Listenfakten sind 3 Modi in Active vorgesehen. Im Modus *strict* (S)

Fakt 1	Fakt 2	unifizierbar
person(„Albert“, „Einstein“)	person(„Albert“, „Einstein“)	wahr
person(„Albert“, „Einstein“)	person(„Einstein“, „Albert“)	falsch
person(„Albert“)	person(„Einstein“, „Albert“)	falsch
person(„Albert“, „Einstein“)	genius(„Albert“, „Einstein“)	falsch
birth(name(„Einstein“), city(„Ulm“))	birth(name(„Einstein“), city(„Bühl“))	falsch

Tabelle 2.3.: Unifikation komplexer Fakten in Active [Guz08]

liefert der Unifikationsoperator den Wert „wahr“ nur in dem Fall, wenn 2 Listen komplett identisch sind inklusive der Reihenfolge ihrer Elemente. Im Modus *strict commutative* (SC) sind zwei Listen nur dann unifizierbar, wenn jedes Element einer Liste mit genau einem Elementen der zweiten Liste unifizierbar ist. Dabei müssen beide Listen die gleiche Länge haben, und die Reihenfolge der Elemente ist irrelevant. Im Modus *partial commutative* (PC) müssen alle Elemente der ersten Liste mit mindestens einem Element der zweiten Liste unifizierbar sein. Tabelle 2.4

Fakt 1	Fakt 2	S	SC	PC
[a, b, c, d, e(f)]	[a, b, c, d, e(f)]	wahr	wahr	wahr
[a, b, c, d, e(f)]	[c, b, a, e(f), d]	falsch	wahr	wahr
[a, b, c, d, e(f)]	[c, b, a, e(f), d, f, g]	falsch	falsch	wahr
[a, b, c, d, e(f)]	[s, t]	falsch	falsch	falsch

Tabelle 2.4.: Unifikation von Listenfakten in Active [Guz08]

Bei der Unifikation eines Fakttes mit einer Variable wird die Variable als dieser Fakt initialisiert und weiter als der Initialisierungswert behandelt. Bei der Unifikation mit einer anonymen Variable („\$“) findet keine Initialisierung statt. Tabelle 2.5

Fakt 1	Fakt 2	unifizierbar	Initialisierung
a	\$var	wahr	\$var = a
a	\$	wahr	Keine Initialisierung, da die Variable Anonym ist
a(b, c)	a(\$var, c)	wahr	\$var = b
a(b, c, d(b))	a(\$var, c, d(b))	wahr	\$var = b
a(b, c, d(b))	a(\$var, c, d(\$var))	wahr	\$var = b
a(b, c, d(c))	a(\$var, c, d(\$var))	falsch	Keine Initialisierung, da die Fakten nicht unifizierbar sind

Tabelle 2.5.: Unifikation von Listenfakten in Active [Guz08]

2.2.5. Faktenspeicher und Auswertungszyklus

Fakten werden im Faktenspeicher gespeichert und verwaltet. Der Faktenspeicher verfügt über eine eigene Funktionalität, um neue Fakten anzulegen und bereits existierende Fakten zu löschen. In regelmäßigen Zeitabständen prüft das Active Framework den Faktenspeicher auf Änderungen. Falls eine Änderung erkannt wurde, wird der Auswertungszyklus (engl. „Rule evaluation cycle“) gestartet.

Dabei werden die Vorbedingungen der Regeln von Konzeptknoten mit dem Inhalt vom Faktenspeicher unifiziert. Falls eine Vorbedingung erfüllt wurde, wird eine entsprechende Aktion durchgeführt.

2.2.6. Kommunikationskanal

Im Active Framework wird Nachrichtenaustausch zwischen Konzepten durch das Anlegen eines speziellen „pipe“ Faktes realisiert. Der „Pipe“ ist ein komplexer Fakt und enthält eine Quelle, einen Empfänger und die zu übertragende Nachricht als Attribute. Eine formale Beschreibung eines Kommunikationskanals ist `pipe($source_name, $destination_name, $data, $attribute_list)`.

2.3. Active Semantic Network

Einer der möglichen Anwendungsbereiche der aktiven Ontologien ist Sprachverarbeitung. Active Semantic Network [Guz08, p. 72] ist ein Spezialfall von aktiven Ontologien, der geeignet ist, natürliche Sprache zu interpretieren und deren Inhalt zu verarbeiten. Dafür wird den Konzepten eine spezielle Logik zugewiesen, mit deren Hilfe der Text in einer natürlichen Sprache auf eine Ontologie abgebildet wird.

2.3.1. Aufbau

Eine Ontologie stellt bei diesem Verfahren eine Baumstruktur mit Konzepten unterschiedlicher Arten dar, die einen speziellen Zweck in der Sprachverarbeitung erfüllen und auf eine Texteingabe reagieren. Die Ausführung eines aktiven semantischen Netzwerks „von unten nach oben“ ab, bis der Wurzelknoten erreicht wird. Die Arten von Konzeptknoten in Active Semantic Network sind in folgenden Abschnitten erläutert. Nach der Verarbeitung der Eingabe wird ein Befehl generiert.

2.3.1.1. Sensorknoten

Die Aufgabe der Sensorknoten ist, alle möglichen Konzepte in einem Satz zu erkennen. Der eingegebene Text wird dafür in Token (Wörter) aufgeteilt, wobei jedes Token als ein Fakt im Faktenspeicher hinterlegt wird. Falls bei dem nächsten Auswertungszyklus einer der neuen Fakten die Vorbedingungen der Sensorknoten erfüllt, wird ein neuer Pipe angelegt, der dieses Token zusammen mit seiner semantischen Bewertung an den Elternknoten übergibt.

Semantische Bewertung ist eine ganze Zahl von 0 bis 100, die den Grad an Sicherheit über die Bedeutung des Wortes darstellt. Falls ein Wort im Wertebereich des Sensorknotens enthalten ist, wird ihm in der Regel die Bewertung 100 zugewiesen.

Es werden nicht nur Token an sich betrachtet, sondern auch solche Informationen wie die Position des Tokens im Satz sowie vorherige und darauffolgende Wörter. Um dies zu ermöglichen, gibt es 4 Arten von Sensorknoten:

Wortschatzlisten (eng. „Vocabulary list“) sind eine Art der Sensorknoten, die eine Menge der möglichen Werte enthalten und die eingegebenen Token mit ihren Wertebereichen vergleichen. Die Einträge im Wertebereich bestehen dabei nicht nur aus einem Wort, sondern enthalten die Menge der möglichen Synonyme. Außerdem wird bei dem Vergleich nicht nur nach einer kompletten Übereinstimmung gesucht, sondern zusätzlich nach Einträgen mit der geringsten Levenshtein-Distanz zu dem gesuchten Token (die Levenshtein-Distanz oder Editierdistanz gibt an, wie viele Permutationen, #insert- und #delete-Operationen benötigt werden, um von einem gegebenen Text den Zieltext zu erhalten).

Prefix- und Postfixknoten werden verwendet, um Konzepte zu erkennen, die nicht im Wertebereich von Wortschatzlisten enthalten sind, anhand von davorstehenden und danachfolgenden Token. Beispielsweise deutet im Satz „Ich will einen Flug nach München“ die Präposition „nach“ daran an, dass „München“ der gewünschte Zielort ist.

Reguläre Ausdrücke vergleichen die Token mit den gespeicherten Mustern und können somit oft verwendete Eingabearten wie E-Mail-Adressen, Postleitzahlen oder Telefonnummern erkennen.

Spezialisierte Sensorknoten enthalten eine vordefinierte Logik, mit der weitere Arten von Token bewertet und transformiert werden können. Ein Beispiel hierzu wäre die Eingabe „Ich will morgen Abend um 10:45 nach New York fliegen“. Ein spezialisierter Sensorknoten kann z.B. das Wort „morgen“ in das Datum „10. August 2017“ transformieren.

2.3.1.2. Kombinationsknoten und Selektionsknoten

Elternknoten von Selektionsknoten sind nicht-terminale Knoten und können entweder Selektionsknoten, oder Kombinationsknoten sein.

Kombinationsknoten bekommen Token von ihren Sensorknoten und vereinigen diese in eine einheitliche Datenstruktur. Zudem bekommen Kombinationsknoten semantische Bewertungen von ihren Kindern, die sie gewichtet als eine semantische Bewertung berechnen und an weitere Elternknoten übergeben.

Selektionsknoten wählen einen ihrer Kinderknoten mit der höchsten semantischen Bewertung aus. Das gespeicherte Konzept und seine semantische Bewertung wird dann an seine Elternknoten übergeben.

2.3.2. Beispiel der Ausführung

In der Abbildung 2.3 ist ein Beispiel der Eingabeverarbeitung in einem Active Semantics Network dargestellt. Angenommen, die ursprüngliche Eingabe ist „I would like to reserve a table in the restaurant ‚licorne‘ for 10 pm“ (dt. „Ich möchte einen Tisch im Restaurant ‚licorne‘ um 10 Uhr abends reservieren“).

Zuerst wird der Text auf Token aufgeteilt. Jedes Token (Wort) wird als ein Fakt im Faktenspeicher der aktiven Ontologie gespeichert. Da eine Änderung im Faktenspeicher stattgefunden hat, wird der Auswertungszyklus gestartet.

Jeder Fakt wird mit den Vorbedingungen der Regeln von Sensorknoten unifiziert. Die Unifikation wird im *partial commutative*-Modus ausgeführt, d.h. wenn eine der Vorbedingungen im Knoten „Aktion“ die Liste [„book“, „reserve“, „reservation“] ist, ist diese Vorbedingung erfüllt, und die entsprechende Aktion kann durchgeführt werden. Die Aktion wäre in dem Fall die Berechnung der semantischen Bewertung und das Anlegen eines Kommunikationskanals zwischen dem Knoten „Aktion“ und seinem Elternknoten „Befehl“. Die Knoten, die an der Kommunikation bei der Verarbeitung der Eingabe in diesem Beispiel teilnehmen, sind in der Grafik grün markiert.

Es ist wahrscheinlich, dass bei der Eingabe eines Befehls an eine Software nach dem Wort „restaurant“ der Name des Restaurants folgen wird. Ein Präfixknoten kann in dem Fall das Wort „restaurant“ erkennen und das Wort „licorne“ als Konzept „Name“ betrachten. Die semantische Bewertung muss in diesem Fall nicht zwangsläufig 100 sein.

Mithilfe des Mustervergleichs kann auch die gewünschte Uhrzeit des Restaurantbesuchs erkannt werden. In diesem Fall wird sie erkannt und in das Format „22:00“ transformiert.

Nach der Erfüllung der Vorbedingungen in Sensorknoten, werden die erkannten Konzepte an die Elternknoten übergeben. In diesem Fall werden der Name des Restaurants und die Zeit des Besuchs an den Kombinationsknoten „Restaurant“ übergeben. Der Kombinationsknoten berechnet eine gemeinsame semantische Bewertung der erkannten Kinderkonzepten und kombiniert die erkannten Token in eine gemeinsame Datenstruktur, die er an den Selektionsknoten „Objekt“ übergibt.

Der Selektionsknoten wählt das Konzept mit der höchsten Bewertung aus. Da keine der Konzepte in „Hotel“ erkannt wurden, ist die Bewertung des Knotens „Hotel“ 0 und somit wird das Konzept „Restaurant“ gewählt. Die extrahierte Information wird im Knoten „Befehl“ gespeichert und wird danach verwendet, um eine Funktion eines externen Dienstes aufzurufen. Beispielsweise kann der Agent nach dem gewünschten Datum des Besuchs fragen, um danach die eingegebenen Daten an den Dienst des Reservierungsservices zu senden.

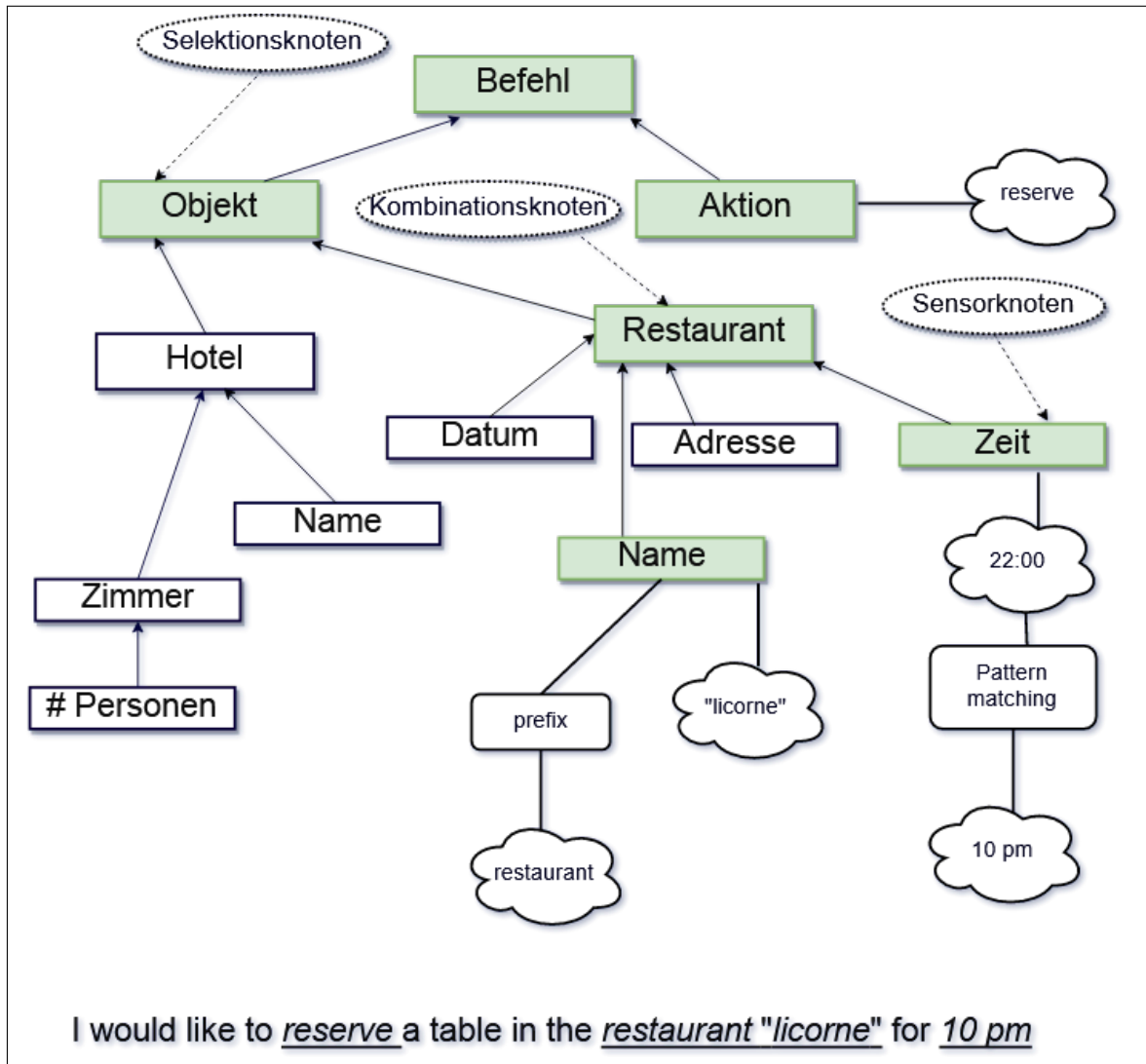


Abbildung 2.3.: Beispiel der Verarbeitung durch ein semantisches Netzwerk

2.4. Webformulare

Eine Webformular ist ein Bereich der HTML-Seite, in den Benutzer Informationen eintragen können, die später an den Server übersendet werden. [Col17]. Die Webformularelemente sind in der Regel Kontrollboxen, Optionsfelder, Dropdown-Listen und Textfelder. In die Letzteren muss der Benutzer selbständig einen Text eingeben. Da im Gegenteil zu anderen Elementen bei diesen Feldern der Wertebereich unbekannt ist, wird weiter unter dem Begriff „Webformularelement“ ein Textfeld verstanden. Im HTML-Code der Webformularelemente können folgende Attribute enthalten sein (Tabelle 2.6)[Col17].

Die Attribute **id** und **name** werden vom JavaScript und bei einem POST-Befehl als Identifikation eines Elementes verwendet, sind deswegen häufig benutzt und enthalten zudem

Attribut	Bedeutung
autofocus	gibt an, ob das Element beim Laden der Seite ausgewählt werden muss
disabled	gibt an, ob die Interaktion mit dem Benutzer erlaubt ist
id	Identifikator des Elements
inputmode	Beschreibt, welche Tastatur anzuzeigen ist (bei Touch-Screen Geräten)
maxlength	die maximale erlaubte Anzahl der Zeichen der Eingabe
minlength	die minimale erlaubte Anzahl der Zeichen der Eingabe
name	Name des Elements
pattern	ein Muster, das die mögliche Eingabe erfüllen muss
placeholder	Hinweise für den Benutzer, welche Eingabe erwartet wird
readonly	gibt an, ob der Wert verändert werden darf
required	gibt an, ob eine Eingabe benötigt wird, um eine Anfrage zu erstellen
size	die Länge des Textfeldes beim Anzeigen im Webbrowser (in Zeichen)
spellcheck	gibt an, ob die Rechtschreibung überprüft werden muss
type	Typ des Elements (Textfeld, Liste, Checkbox usw)
value	Ein Beispiel der Eingabe

Tabelle 2.6.: Attribute eines Textfeldes

oft nützliche Informationen für die zu erstellende Software. Dies wird genauer im Kapitel 4 erläutert. **Value** enthält den eingegebenen Wert und kann beim Laden der Webformular mit einem Domänenwert vorbelegt sein. Um dem Benutzer einen Hinweis zu geben, welche Informationen einzugeben ist, kann das Attribut **placeholder** oder das **Label** mit einem Text belegt sein, der ebenfalls von der Software verwendet werden kann.

2.5. Externe Datenquellen

Eine Wissensdatenbank (eng. „Knowledge base“) ist eine Darstellung des Wissens über Objekte einer Domäne und unterschiedliche Beziehungen (Relationen) zwischen diesen Objekten [SS89]. Sie sind ein erforderlicher Teil von Expertensystemen, deren Verhalten auf den gespeicherten Daten und Regeln basiert. Das Wissen kann dabei sowohl als eine relationale Datenbank dargestellt werden, als auch als eine Menge von Ontologien. In diesem Unterkapitel werden die Wissensdatenbanken dargestellt, die als eine externe Datenquelle von der entwickelten Software befragt werden.

2.5.1. Wikipedia

Wikipedia [wike] ist eine online Plattform, die als eine elektronische Enzyklopädie dient. Ursprünglich als eine textbasierte Ressource gedacht, stellt das Wissen von Wikipedia heutzutage eine organisierte Struktur dar. Diese Struktur enthält eine Menge von Relationen aller Arten zwischen Entitäten, zum Beispiel Familienbäume oder Taxonomie der Tierarten. Der Inhalt von Wikipedia wird von Benutzern in der ganzen Welt geschrieben und enthält über 30 Millionen Artikel in 287 Sprachen. [VK14]

In der Abbildung 2.4 ist ein Beispiel eines Artikels von Wikipedia zu finden. Ein Artikel auf Wikipedia besteht aus einem strukturierten Text, der Verweise zu anderen Artikeln enthält. Um dem Leser die Suche und Navigation in Wikipedia zu erleichtern, sind ähnliche Objekte (Artikel) in Wikipedia in Kategorien zusammengefasst. Dabei kann ein Objekt einen oder mehreren Kategorien angehören und jede Kategorie kann eine Unterkategorie einer anderen Kategorie sein [wika]. Der Benutzer findet in der Regel die Liste der möglichen Kategorien wie in der Abbildung 2.5 dargestellt.

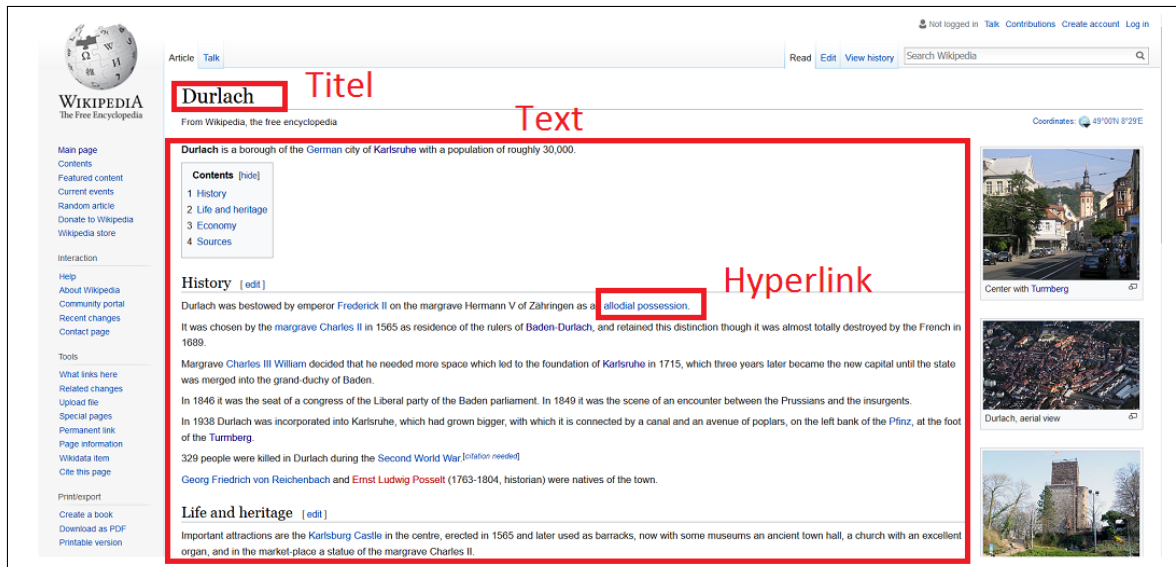


Abbildung 2.4.: Ein Beispiel eines Artikels auf Wikipedia

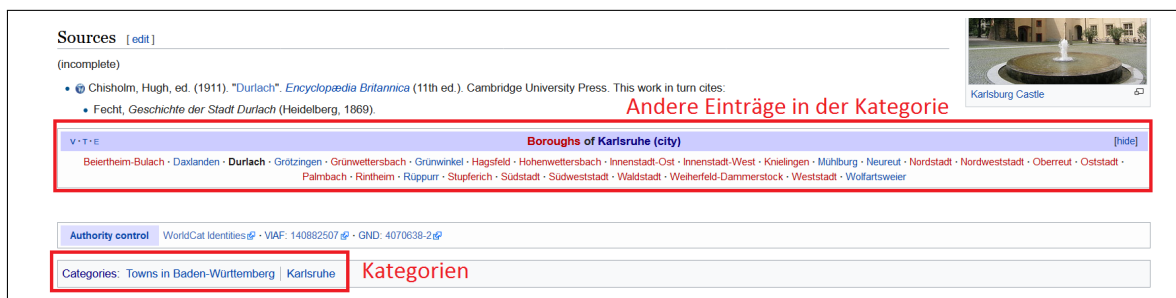


Abbildung 2.5.: Kategorienliste unter dem Artikel

2.5.2. WordNet

WordNet ist ein semantisches Netz der Princeton University, dessen Design der Vorstellung der Organisation des menschlichen Gedächtnisses ähnlich ist [MBF⁺90]. Es ist eine lexikalische Datenbank der englischen Sprache, die eine Menge von Wörtern und Definitionen enthält. WordNet ist eine der am meisten benutzten lexikalischen Ressourcen im NLP-Bereich (Natural Language Processing) [MFC14]. Zudem sind Substantive in einer Taxonomie organisiert [MBF⁺90]. In der Abbildung 2.6 ist ein Beispiel der Hierarchie von WordNet. Es sind nicht nur Relationen zu den Oberbegriffen, sondern auch in Wordnet enthaltene Relationen zu anderen Einträgen dargestellt (in der Grafik als Notiz).

Wortformen, Synonyme und Wörter mit ähnlicher Bedeutung sind in WordNet in über 117.000 Synsets gruppiert. Synsets sind miteinander mit lexikalischen (Wort-Wort) oder semantischen (Synset-Synset) Relationen verbunden. [MFC14]

2.5.3. Cyc

Cyc ist eine seit 1984 von Cycorp, Inc. entwickelte Datenbank, deren Zweck es ist, auf einer für eine Maschine lesbaren Weise die Information über den Alltagsverstand (Commonsense) zu speichern. [FEMR15]. Der menschliche Commonsense ist dabei das Hintergrundwissen, das bei einer beliebigen Aufgabestellung an einen Menschen vorausgesetzt wird [TMKW07].

Die Wissensbasis (engl. „Knowledge Base“) von Cyc besteht aus Annahmen (engl. „assertions“), die formal als eine Menge von Fakten unterschiedlicher Komplexität dargestellt sind.

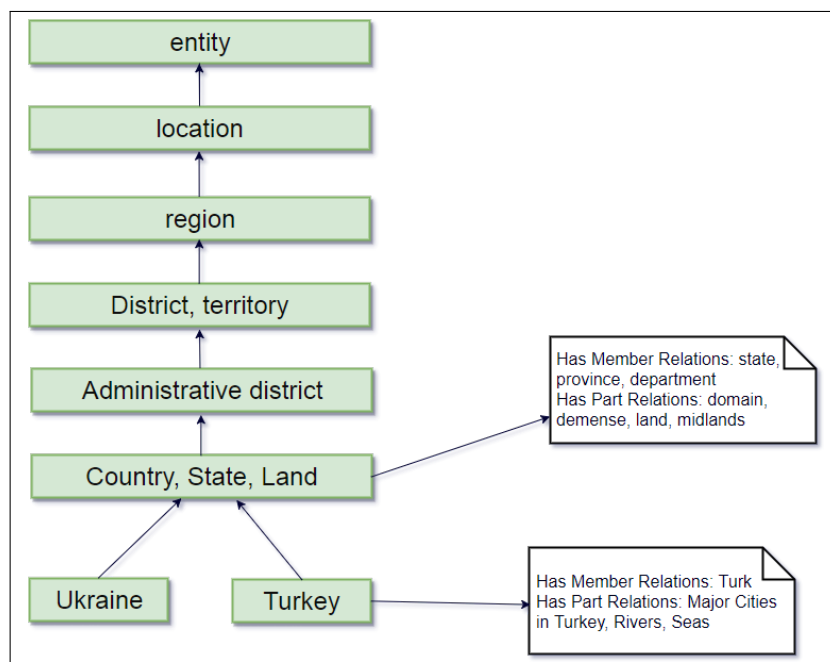


Abbildung 2.6.: Beispiel der Hierarchie von Substantiven in WordNet [Erc06]

Diese Annahmen bilden eine Hierarchie getrennter Kontexte, die in der Cyc-Terminologie Mikrotheorien (engl. „microtheories“) genannt werden [TMKW07].

Das System Cyc ist mit der Sprache CycL zu verwenden, welche die Prädikatenlogik erster Stufe für Cyc bietet [MCWD06]. Die atomaren Terme (z.B. `#$Tower`, `#$Person`, `#$Bill-Clinton`) sind als Individuals bezeichnet und gehören oft einer oder mehreren Gruppen (eng. „Collections“) an. Individuals und Gruppen sind in der Sprache CycL mit dem Präfix „#\$“ gekennzeichnet, und für sie können eine oder mehrere textuelle Beschreibungen enthalten sein [LP07]. Wortsequenzen „person“, „human“ und „individual“ entsprechen beispielsweise dem Individual `#$Person`. Es kann abgefragt werden, welche Individuals einer Gruppe angehören und gewisse Bedingungen erfüllen, und somit eine sinnvolle Benutzereingabe darstellen können.

Für diese Arbeit wurde die Version „ResearchCyc“ [res] verwendet, die über den kompletten Inhalt von Cyc verfügt und für Wissenschaftsorganisationen für die Forschung in der Informatik veröffentlicht wurde.

2.6. Zusammenfassung

In diesem Kapitel wurden die Konzepte und Begriffe erklärt, deren Verständnis für diese Bachelorarbeit wichtig ist. Zuerst wurden Ontologien als ein Verfahren der Datenorganisation in der Informatik beschrieben. Die in EASIER verwendeten Bausteine von Active Framework (aktive Ontologie und das aktive semantische Netzwerk) wurden in den darauffolgenden Abschnitten beschrieben. Da die zu erstellende Software als Eingabe eine Menge der Attribute von Webformularelementen bekommt, wurden Webformularelemente beschrieben. Anschließend wurden die externen Datenquellen genannt, mit deren Hilfe die möglichen Wertebereiche ermittelt werden. Die genannten Datenquellen wurden bereits in zahlreichen Forschungsprojekten verwendet, die im folgenden Kapitel beschrieben werden.

3. Verwandte Arbeiten

Im Rahmen dieser Bachelorarbeit soll eine Software entwickelt werden, die aus HTML-Attributen eines Formularelementes den Wertebereich der Eingabe ermitteln kann, um die Automatisierung der Ontologie-Erstellung zu verbessern. Es werden Daten aus einer externen Datenquelle verwendet, um eine bestehende Ontologie zu vervollständigen. In diesem Kapitel werden unterschiedliche Arbeiten erläutert, deren Techniken zur Lösung der vorliegenden Bachelorarbeit verwendet werden können. Vor allem sind die Ansätze aus dem Bereich der automatischen Ontologieerstellung von Interesse

Manuelle Ontologieerstellung benötigt die Zeit und den Aufwand von Ontologieexperten. Diese müssen über das Wissen der Anwendungsdomäne verfügen, um eine möglichst korrekte und vollständige Ontologie für die gegebene Anwendung zu erzeugen. Eine Automatisierung der Ontologieerstellung sollte Kosten senken und gleichzeitig die Wissensdomäne der Ontologie an die Anwendung anpassen [HEBR11]. Automatische und semi-automatische Ontologieerstellung wird oft in der englischsprachigen Literatur als „Ontology Learning“ bezeichnet. Dabei steht der Entwickler vor der Aufgabe, aus der Menge von Datensätzen eine domänenspezifische Ontologie zu erzeugen.

Externe Datenquellen, die zur Ontologieerstellung verwendet werden, können in drei Gruppen unterteilt werden: strukturierte (z.B. Datenbanken), semistrukturierte (z.B. HTML-Dokumente) und unstrukturierte (z.B. Texte) Datenquellen [HEBR11].

3.1. Ontologieerstellung aus unstrukturierten Daten

Bei dem Lernen aus unstrukturierten Daten müssen relevante Konzepte und Relationen aus einem Textkorpus erkannt werden. Das Lernen von diesem Typ von Daten ist aufwendig und hängt von dem grundlegenden Mechanismus der Sprachverarbeitung ab, wie zum Beispiel eine semantische Analyse des Textes, Part-of-Speech - Identifikation oder regelbasierte Verfahren. Die meisten dieser Techniken werden in Verbindung mit der Verarbeitung der natürlichen Sprache verwendet [HEBR11].

3.1.1. Konstruierung einer biomedizinischen Datenbank aus dem Textkorpus von PubMed

Die Arbeit „A Case Study on Sepsis Using PubMed and Deep Learning for Ontology Learning“ [ACMFD⁺17] befasst sich mit der Erstellung einer Ontologie der Biomedizin auf

Basis der Textkorpora von PubMed. Dabei wurden distributive semantische Modelle „Latent Semantic Analysis“ und „Latent Dirichlet Allocation“, sowie neuronale Sprachmodelle „Continuos Bag-of-Words“ und „Skip-gram“ von Mikolov verwendet.

PubMed [pub] ist eine Datenbank der wissenschaftlichen Arbeiten im Bereich Biomedizin der nationalen medizinischen Bibliothek der Vereinigten Staaten. Mithilfe von PubMed kann jeder Benutzer nach Studien suchen, die Kurzfassung dieser Studie lesen und den Link zu der Studie abrufen.

3.1.1.1. Hintergrund

Distributive semantische Modellierung (engl. „Distributional Semantic Modeling“ oder „distributional semantics“) ist ein Fachgebiet, das sich mit der Bewertung und Kategorisierung semantischer Ähnlichkeiten oder Relationen zwischen linguistischen Konzepten auf Basis ihrer Verteilung befasst [Wu16].

Der erste verwendete distributive Ansatz ist die **latente semantische Analyse** (LSA) (engl. „Latent Semantic Analysis“). Dafür wird die Textmenge in Einheiten aufgeteilt (z.B. Sätze) und eine Liste der interessierenden Wörter erstellt. Der Ansatz besteht in der Singulärwertzerlegung einer von der Textkorpora abgebildeten Matrix. In dieser Matrix entspricht jeder Eintrag an der Stelle (i, j) jeweils der Anzahl des Auftretens des Wortes i im Dokument j . Bei der Eingabe der gewünschten Anzahl von Konzepten kann der Algorithmus die Korrelation zwischen Wörtern berechnen, um zum Beispiel stark korrelierende Wörter in ein Konzept zu erfassen [LLRS97].

Latent dirichlet allocation [BNJ03] (LDA) ist der zweite verwendete distributive Ansatz. Die grundlegende Idee dieses Ansatzes ist, dass ein Textdokument eine zufallsverteilte Mischung aus verschiedenen Themen ist, in dem jedes Thema durch eine Verteilung von Wörtern dargestellt wird. Nach der Anwendung dieses Algorithmus auf ein Textkorpora ergibt sich eine gewichtete Aufteilung der Dokumente im Textkorpora auf Themen, sowie eine Zuteilung jedes Wortes in ein Thema.

Neben distributiven Modellen werden auch neuronale Modelle **Continuous Bag of Words** (CBOW) und **Skip-Gram** verwendet. Bei CBOW wird mit der Hilfe von neuronalen Netzwerken aus den vorherigen und darauffolgenden Wörtern das aktuelle Wort ermittelt. Bei Skip-Gram werden im Gegensatz dazu mögliche davorstehende und darauffolgende Wörter unter Eingabe des aktuellen Worts ermittelt.

Mithilfe der dargestellten Modelle können bei einer Suchanfrage zusammenhängende Konzepte ausgegeben werden.

3.1.1.2. Ansatz

Im ersten Schritt wurde aus dem Methathesaurus System UMLS (Unified Medical Language System) manuell eine kleine Ontologie mit potenziellen Konzepten erstellt. Anschließend wurde Vorverarbeitung der Texte mit allen vier Methoden in zwei Experimenten durchgeführt:

1. Verarbeitung unter Einhaltung der Großschreibung und Zahlen im Text
2. Verarbeitung ohne Einhaltung der Großschreibung und Zahlen im Text

Der nächste Schritt ist die Bewertung der Relevanz von erkannten Konzepten in Paaren (Suchterm, potenzieller Term) von Domänenexperten anhand von der Likert-Skala [BJB12], einem Verfahren, bei dem Befragte zu jeder Antwort eine Aussage treffen müssen, die ihre Einstellung am besten darstellt. Die Relevanz wurde mit 4 Stufen bewertet:

- 0 - komplett irrelevant
- 1 - eher irrelevant
- 3 - eher relevant
- 4 - sehr relevant

Aus den Ergebnissen der Experimente 1 und 2 wurden die am besten bewerteten Konzepte für die neue Ontologie ausgewählt. Es wurde nicht erklärt, wie Relationen extrahiert werden. Distributive Methoden haben eine höhere Genauigkeit bei der Konzepterkennung erwiesen, während neuronale Methoden eine bessere Laufzeit anbieten konnten.

3.1.1.3. Fazit

In der beschriebenen Arbeit wurden vier verschiedene Techniken zur Ermittlung semantisch verwandter Konzepte verwendet. Die Aufgabe dieser Bachelorarbeit ist Bestimmung der Wertebereiche von Elementen unter Eingabe einer aus mehreren Webformularen zusammengefassten Datei. Die vier verwendeten Ansätze können zur Suche nach semantisch verwandten Elementen benutzt werden, um somit einen gemeinsamen Wertebereich zu finden.

3.2. Ontologieerstellung aus semistrukturierten Daten

Beim ontologischen Lernen aus semistrukturierten Daten werden Techniken aus Data Mining und Web Content Mining verwendet [HEBR11]. Bei dieser Art der Ontologieerstellung wird sowohl die Semantik der Eingabedaten, sowie die Struktur dieser Daten verwendet.

3.2.1. Ontologisches Lernen durch Web-Crawling

Zur Erstellung einer domänenspezifischen Ontologie werden meistens Textkorpora in dieser Domäne benötigt, die nicht immer zur Verfügung stehen. Da es sehr wenige Arbeiten gibt, in denen eine automatische Textkorpussuche als ein Teil der Ontologieerstellung angesehen wird, wird in der Arbeit „Unsupervised Domain Ontology Learning from Text“ [Gee17] solch ein System dargestellt.

3.2.1.1. Hintergrund

In dieser Arbeit wird der HITS-Algorithmus („Hypertext-induced topic selection“) [NL07] zur Bewertung der Relevanz von Webseiten verwendet. Der HITS-Algorithmus unterteilt Seiten auf zwei Kategorien:

1. **Knotenpunkte** (engl. „Hubs“), die Verweise zu anderen Seiten enthalten
2. **Instanzen** (engl. „Authorities“), die einen wertvollen Inhalt haben

Die Bewertung der jeweiligen Seite ist von der Kategorie dieser Seite abhängig. Ein Knotenpunkt bekommt eine hohe Bewertung, wenn dieser auf viele hoch bewertete Instanzen verweist. Eine Instanz wird hoch bewertet, wenn diese aus vielen Knotenpunkten mit hohen Bewertungen erreichbar ist.

Eine der verwendeten Ansätze ist **Hearst Pattern Extraction** ist eine Technik, mit der Relationen zwischen Hyperonymen (Oberbegriffen) und Hyponymen (Unterbegriffen) in einem Text gefunden werden können. Hearst Patterns sind Muster, die Wortbindungen zwischen Hyperonymen und Hyponymen darstellen. Bei der Suche nach dem Muster „(Plural NP) such as (NP), (NP) [...] and (NP)“ kann ein Treffen wie „Cities, such as Berlin, Kaiserlauten and London“ ausgegeben werden. Diese Wortbindung wird verwendet, um den Begriff „City“ als Oberbegriff von den Begriffen „Berlin“, „Kaiserlauten“ und „London“ zu erkennen [ACS10].

3.2.1.2. Ansatz

Insgesamt besteht die Verarbeitung in der entwickelten Rahmenstruktur aus fünf Phasen:

1. Suche nach Textkorpora
2. Extraktion von Termen
3. Extraktion von taxonomischen Relationen
4. Extraktion von nicht taxonomischen Relationen
5. Ontologierstellung

Die Suche nach den relevanten Texten erfolgt mittels Web-Crawling. Die Relevanz jeder Web-Seite wird bewertet, und wenn sie erheblich sinkt, wird das Crawling terminiert. Alle Substantive der verarbeiteten Web-Seiten sind als potenzielle Terme betrachtet. Sie werden aus den Textkorpora mit dem Stanford Parts-of-Speech Tagger extrahiert. Anhand der extrahierten Terme wird die Relevanz zwischen den Testtermen und den Termen der Webseite berechnet und somit die Relevanz der Seite bestimmt.

Mittels Stanford Dependency Parser werden semantische Relationen zwischen Wörtern in den Sätzen der Domänendokumente ermittelt. Zusätzlich wurden Adjektivmodifikatoren (z.B. „biological research“) bestimmt. Mit dem HITS-Algorithmus werden aus den gefundenen Termen die relevantesten Terme ausgewählt.

In der nächsten Phase wird Taxonomie zwischen den Termen ermittelt. Dafür wird zuerst die Hearst Pattern Extraction angewendet. Zusätzliche taxonomische Relationen werden durch die folgende morpho-syntaktische Muster angewendet:

Bedingung	Folgerung	Beispiel
t0 Suffix von t1, beide Domänen- terme	t0 Hypernym von t1	„polysacharide“ Hypernym von „homopolysacharide“
t0 Kopfterm von t1, einer Do- mänenterm	t0 Hypernym von t1	„corn“ Hypernym von „sweet corn“

Tabelle 3.1.: Morpho-syntaktische Muster

Nicht-taxonomische Relationen werden durch Triplet Extraction und Association Rule Mining bestimmt.

Ein Satz besteht in der Regel aus einem Subjekt, Prädikat und Objekt. Ein Triplet ist eine Relation zwischen einem Subjekt und einem Objekt, die durch das Prädikat dargestellt wird.

Bei Association Rule Mining werden durch die Analyse des Auftretens der Terme in den Texten Abhängigkeiten von Termen voneinander bestimmt. Die Abhängigkeiten zwischen Termen, bei denen eine hohe Sicherheit an Richtigkeit dieser Abhängigkeit besteht, werden als Relationen erkannt.

Aus den gefundenen Termen und Relationen wird abschließend eine Ontologie erstellt.

3.2.1.3. Fazit

In dieser Arbeit wird ein Ansatz zur Erstellung domänenspezifischer Ontologie vorgestellt, der sich auf fast alle Domänen übertragen lässt. Obwohl diese Bachelorarbeit sich nur mit dem Lernen aus strukturierten Daten befasst, stellt die Technik zur Erstellung der taxonomischen Relationen eine weitere Möglichkeit zur Bestimmung der Wertebereiche eines Konzepts dar.

3.2.2. Konstruktion des Konzeptwissens aus HTML Seiten

In der Arbeit „A framework for retrieving conceptual knowledge from Web pages“ [BK05] wird ein Ansatz dargestellt, der aus den strukturellen und linguistischen Merkmalen eines HTML Dokumentes eine Ontologie erzeugen kann.

3.2.2.1. Ansatz

Im Rahmen der Arbeit [BK05] wurde eine Software entwickelt, die aus drei Komponenten besteht, die sequentiell aufgerufen werden (Abbildung 3.1). Die Aufgabe der ersten Komponente ist, eine Webseite in eine relationale Tabelle zu überführen. Diese Tabelle wird von der zweiten Komponente erweitert, indem die Struktur von der Seite und Semantik des Textes analysiert wird. Aus der resultierenden Tabelle erstellt die dritte Komponente eine Ontologie. Ein Benutzer verwaltet den Prozess und bewertet die Ausgaben von jeder Komponente.

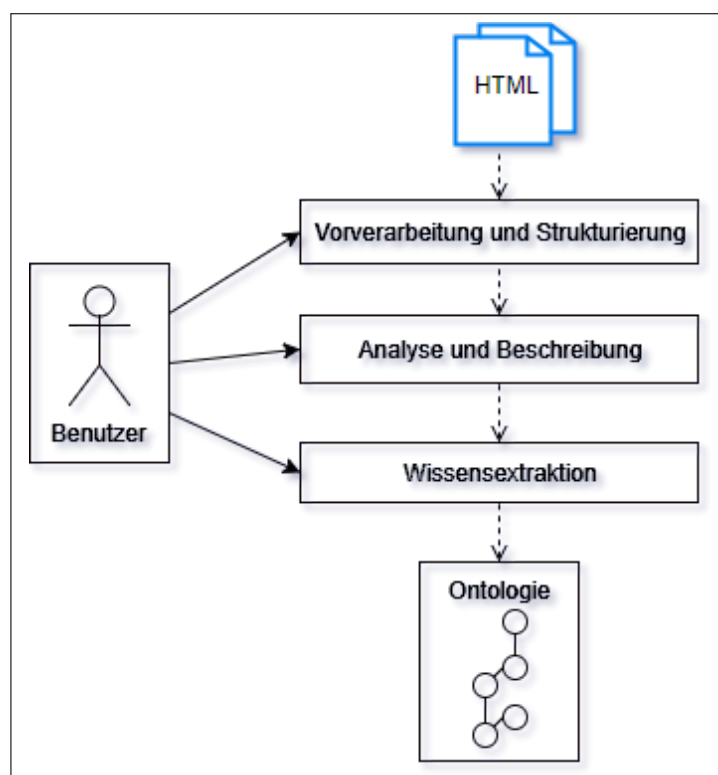


Abbildung 3.1.: Die Struktur des erstellten Systems [BK05]

Vorverarbeitung und Strukturierung

HTML-Seiten enthalten neben der textuellen Information Markierungen zur Vereinfachung der Darstellung dieser Information für den Benutzer. Eine Blockmarkierung wie `<h1>` (große Überschrift) wird verwendet, um die Relevanz der Information zu betonen. Die Markierung `<h2>` (Überschrift kleiner als `<h1>`) bedeutet eine leicht weniger relevante Überschrift. ``, `` und `` bedeuten ungeordnete verschachtelte Aufzählung. Mit der Markierung `` (Fettschrift) werden in der Regel die relevantesten Wörter im Text deutlich gemacht. Mithilfe dieser Markierung erfolgt die Bewertung der Relevanz des Textblocks auf der Seite von der erstellten Software.

Das Textkorpus zur Verarbeitung besteht aus einer Menge der HTML-Seiten, die eine spezifische Domäne darstellen. Die erste Komponente bereitet die eingegebenen Seiten vor, indem nur als relevant angesehene Textblöcke extrahiert werden. Im nächsten Schritt

werden relevante Terme erkannt, zum Beispiel die Bezeichnungen von einem Hyperlink (<TITLE_URL>) oder einer Checkbox (<CHOICE>) und die Menge aller Schlüsselwörter (<KEYWORDS>). Die erkannten Terme werden in einer relationalen Tabelle gespeichert.

Analyse der Dokumente

Im nächsten Schritt werden drei Arten von Analysen durchgeführt: Analyse der Struktur, lexikalische und linguistische Analysen der Dokumente.

Bei der **Analyse der Struktur** werden die Anzahl der Markierungen und Anteile der verbundenen Terme berechnet. Nach diesem Schritt werden dem Benutzer Gewichtungen zum weiteren Clustering der eingegebenen Texte angeboten.

Bei der **lexikalischen Analyse** wird Diversität und Homogenität des Textkorpus bewertet. Den erkannten Termen werden Bewertungen zugewiesen, wobei die Terme, die die Dokumente unterscheiden, höher bewertet werden. Durch das Hinzufügen neuer und das Löschen vorhandener Dokumente werden diese Ergebnisse angepasst, bis eine homogene Abdeckung der Domäne erreicht wird.

Ein weiterer Teil der Komponente ist die **linguistische Analyse**. Die erstellte relationale Tabelle wird um syntaktische Kategorien und Wortstämme der Terme ergänzt. Dies erlaubt, eine Menge syntaktischer Muster zu erzeugen, mit denen Relationen zwischen Termen erstellt werden können.

Erstellung des Konzeptwissens

Die Schlüsselwörter, deren Relevanz am höchsten bewertet ist, werden zu den Wurzeln des Baums. Danach wird ein Clustering der verbleibenden Terme durchgeführt, um weitere Hierarchiestufen zu erkennen. Zum Clustering der Terme in Konzepte werden zwei Arten von Berechnung von Distanzen zwischen Termen angewandt: Auftreten in gemeinsamen Kontext und Auftreten in gemeinsamer Struktur. Wenn zwei Terme sich innerhalb der gleichen Markierung befinden, werden diese Terme als Terme in der gleichen Struktur betrachtet. Unter Termen im gleichen Kontext werden Terme innerhalb verschiedener Markierungen verstanden, die durch eine Verknüpfung (ein gleicher Term in den Sätzen) verbunden sind. Aus den Clustern der Terme werden neue Ebenen des Baums erstellt und nach der berechneten Distanz zu den Elternknoten mit diesen verbunden.

3.2.2.2. Fazit

In dieser Arbeit wurden Konzepte durch Analyse des Inhalts und Struktur der HTML-Seiten erkannt. Der Prozess ist sehr aufwendig, betrachtet mehrere Dimensionen der Eingabe und verlangt eine permanente Steuerung durch einen Benutzer. Obwohl die Ergebnisse der Arbeit von den Autoren nicht vorgestellt wurden, eignet sich der Ansatz nicht für das Ziel der vorliegenden Bachelorarbeit, und andere Arten der externen Datenquellen sind zu überlegen.

3.3. Ontologierstellung aus strukturierten Daten

Alle in der vorliegenden Bachelorarbeit verwendeten externen Datenquellen stellen strukturierte Datenquellen dar. In diesem Abschnitt werden wissenschaftliche Arbeiten vorgestellt, die sich ebenfalls mit dem ontologischen Lernen aus den Strukturen von Wikipedia und Cyc befassen.

3.3.1. **Konstruktion einer domänenspezifischen Datenbank der Softwaretechnik aus Wikipedia und StackOverflow**

Wikipedia dient als Basis von vielen wissenschaftlichen Arbeiten im Bereich der Ontologieerstellung (z.B. ([SKW08], [SFJ08], [HBS06])). Dabei können nicht nur textuelle Inhalte verwendet werden, sondern auch die Struktur von Wikipedia und Metadaten über vorliegende Einträge.

In der Arbeit „Building a Domain Knowledge Base from Wikipedia: a Semi-supervised Approach“ [CDZS16] ist ein Verfahren beschrieben, mit dem eine Ontologie der Softwaretechnik semi-automatisch erzeugt wird. Die meisten Arbeiten im Bereich automatischer Wissensextraktion befassen sich mit der Erstellung einer Wissensquelle mit allgemeinem Wissen. Solche Wissensquellen sind nicht oder schwer verwendbar, wenn nur die Information in einer spezifischen Domäne gesucht wird.

Das beschriebene Verfahren stellt eine Kombination des Web-basierten und Enzyklopädie-basierten Verfahrens dar. Bei einer Web-basierten Wissensextraktion wird das Wissen aus Web-Inhalten extrahiert, wobei nur ein Teil der relevanten Webseiten für domänenspezifische Wissensdatenbanken relevant ist. Enzyklopädie-basierte Verfahren erlauben in der Regel eine höhere Genauigkeit bei der Wissenskonstruktion, dagegen eine geringere Ausbeute als Web-basierte Verfahren. Die am meisten in der Forschung verwendete Wissensquelle bei den Enzyklopädie-basierten Verfahren ist Wikipedia. [CDZS16]

3.3.1.1. **Ansatz**

Um die Vorteile beider Ansätze zu nutzen, werden Wikipedia als Hauptwissenquelle und StackOverflow [staa] als unterstützende Wissenquelle verwendet. StackOverflow ist ein Webservice, auf dem Benutzer Fragen zu softwaretechnischen Themen stellen und Antworten von Experten bekommen können. Bei der Erstellung der Fragen geben Benutzer Hashtags an, um die Fragen mit Schlüsselwörtern zu annotieren.

Zur Lösung wird die Aufgabe auf zwei Teilaufgaben aufgeteilt: die Entdeckung der Domänenkonzepte und Entdeckung der semantischen Relationen zwischen den Konzepten.

Entdeckung der Domänenkonzepte

Um das erste Problem zu lösen, werden domänenrelevante Hashtags von StackOverflow mit den Einträgen in Wikipedia verglichen.

Dafür werden im ersten Schritt domänenrelevante Tags ausgewählt. Die Hashtags von StackOverflow werden von den Benutzern erstellt und sind deswegen nicht immer domänenrelevant (z.B. „music“) oder fehlerhaft (z.B. „jave“ statt „java“). Um solche Tags auszufiltern, werden Hashtags der am meisten bewerteten Fragen ausgesucht und nach der Auftrittshäufigkeit sortiert. Davon werden die ersten 30% ausgewählt.

Im zweiten Schritt müssen ausgewählte Hashtags auf Einträge in Wikipedia abgebildet werden. Zuerst werden alle Trennsymbole durch Leerzeichen ersetzt und Versionsnummern (z.B. „Visual Studio 2012“) gelöscht. Danach wird das Hauptwort im Hashtag bestimmt. Falls das Hashtag aus einem Wort besteht, ist dieses Wort das Hauptwort. Falls das Hashtag aus zwei Worten mit einer Präposition „of“, „in“ oder „for“ dazwischen besteht, ist das Wort vor der Präposition das Hauptwort. Ansonsten wird das letzte Wort als Hauptwort identifiziert. Mit dem Programm StanfordNLP wird der Wortstamm des Ergebnisses ermittelt. Falls es Einträge in Wikipedia gibt, die mit einem oder mehreren ermittelten Wortstämmen übereinstimmen, werden diese Einträge als softwaretechnische Konzepte in die Datenbank aufgenommen. Die Entdeckung weiterer Konzepte erfolgt mithilfe der Struktur von Wikipedia, und zwar über Kategorien und Verweise in den Artikeln von Wikipedia.

Entdeckung der semantischen Relationen

Die Lösung des zweiten Problems besteht in der semantischen Verarbeitung der Seiten von Wikipedia, um die „relate“- , „subclass of“- und „equal“-Relationen zu bestimmen. Falls ein erkanntes Konzept einer Kategorie gehört, die auch als softwaretechnisches Konzept erkannt wurde, wird zwischen diesen Konzepten eine gerichtete „subclass of“-Relation erstellt. Wenn es in Wikipedia zwischen den Konzepten eine Weiterleitung auf einen Synonym gibt, wird eine „equal“-Relation erstellt. Zusätzlich wird eine normalisierte Google-Distanz [CV07] verwendet, um „relate“-Relationen zwischen den Konzepten zu finden.

Die normalisierte Google-Distanz (NGD) zwischen den Konzepten A und B ist:

$$NGD_{A,B} = \frac{\max(\log(f_A), \log(f_B)) - \log(f_{A,B})}{\log(N) - \min(\log(f_A), \log(f_B))} \quad (3.1)$$

Dabei ist N die gesamte Anzahl der Relationen, f_A und f_B sind die Anzahl der Konzepte, die jeweils mit A und B durch eine Relation verbunden sind, und $f_{A,B}$ ist die Anzahl der Konzepte, die sowohl mit A als auch mit B verbunden sind.

Abschließend wird die resultierende Datenbank auf Redundanz überprüft und überflüssige Relationen werden entfernt.

3.3.1.2. Evaluation

Nach der Anwendung des Verfahrens wurde die resultierende Informationsbank manuell von Studenten überprüft. Dafür wurden jeweils 500 Konzepte und Relationen jeder Art zufällig ausgewählt. Die Studenten haben Relationen ohne die Angabe ihrer Art bekommen und mussten diese selbst den Arten zuordnen. Danach wurde die gleiche Anzahl der korrekten Relationen aus Wikipedia gewählt, um die Trefferquote zu berechnen.

Der Ansatz erreicht eine Genauigkeit von 76.7% und eine Trefferquote von 92.4% für die Konzepte. Bei den Relationen ist die Genauigkeit ungefähr genauso hoch, hingegen ist die Trefferquote niedriger als erwartet. Die Autoren erklären die niedrige Trefferquote damit, dass der Fokus der Arbeit hauptsächlich auf der Konzepterkennung lag. Außerdem wurden die Relationen aus der Struktur von Wikipedia abgeleitet, und Wikipedia enthält nicht immer Verweise auf Kategorien zu jedem Konzept.

Da es keine anderen vergleichbaren Datenbanken der Softwaretechnik gab, wurde das Ergebnis mit den allgemeinen Taxonomien wie WikiTaxonomy und Yago verglichen. Die resultierende Datenbank enthält eine wesentlich höhere Anzahl von Konzepten als vergleichbare Datenquellen.

3.3.1.3. Fazit

In dieser Arbeit wird ein Ansatz dargestellt, mit dem eine Wissensdatenbank der Softwaretechnik erstellt werden kann. Der Ansatz besteht aus den folgenden Schritten:

1. Erkennung von Konzepten:
 - 1.1. Auswahl der am meisten verwendeten Hashtags der am meisten verwendeten Fragen auf StackOverflow
 - 1.2. Abbildung der ausgewählten Tags auf Stammformen
 - 1.3. Aufnahme der mit dem Ergebnis übereinstimmenden Einträge in Wikipedia als Konzepte
 - 1.4. Aufnahme weiterer Einträge in Wikipedia über Kategorien und Verweise

2. Erkennung von Relationen:

- 2.1. Erkennung der „subclass of“-Relationen über Kategorien in Wikipedia
- 2.2. Erkennung der „equals“-Relationen über Weiterleitungen in Wikipedia
- 2.3. Erkennung der „relate“-Relationen über Verweise in Wikipedia

Die vorgestellte Arbeit stellt ein Interesse für die vorliegende Bachelorarbeit dar, da die Struktur von Wikipedia ebenfalls als Basis für die Ontologieerstellung gewählt wurde. Die „subclass of“-Relationen in der resultierenden Datenbank sind in vielen Fällen den Einträgen in Wortschatzlisten der Sensorknoten äquivalent (z.B. „java“ - „subclass of“ - „programming language“ kann als Eintrag „java“ im Vokabular des Sensorknotens „programming language“ interpretiert werden). Außerdem kann die vorgestellte Methode der Wortstambildung behilflich sein, um den Konzeptnamen entsprechende Einträge in Wikipedia zu ermitteln.

3.3.2. Integration des Inhalts von Wikipedia in Cyc

Im Bereich der Ontologieerzeugung werden stets neue Ansätze entwickelt, um Wiederverwendung des Wissens zu ermöglichen [CKGB17]. Unter Wiederverwendung des Wissens wird meistens Erzeugung einer gemeinsamer Ontologie aus mehreren vorhandenen oder Ergänzung einer Ontologie durch den Inhalt der Datenquellen verstanden.

Die Arbeit „Integrating Cyc and Wikipedia: Folksonomy Meets Rigorously Defined Common-Sense“ [ML08] wird eine relationenreiche Ontologie von Cyc durch Konzepte aus Wikipedia ergänzt. Das Ziel dieser Arbeit ist, die Ontologie von Cyc zu erweitern, um gleichzeitig die Ausbeute von Wikipedia und die organisierte Struktur von Cyc zu benutzen.

Ansatz

Cyc verfügt über eine große Anzahl an Relationen, von denen viele nicht den allgemeinen Verstand darstellen und zuerst ausgefiltert werden müssen. Dafür werden alle Kategorien gelöscht, die das Wissen über den internen Aufbau von Cyc darstellen, Mikrotheorien, Sprachverarbeitungswissen, Prädikate und Instanzen von `#$Relation`.

Vor der Durchführung der Abbildung werden Cyc-Terme vorbereitet, um die entsprechenden Artikelnamen in Wikipedia zu finden. Die zusammengesetzten Namen werden auf Wörter aufgeteilt und den Standards von Wikipedia angepasst. Dabei wird die Großschreibung nicht beachtet, da diese in Cyc oft zur Trennung der Wörter (z.B. `#$OpticNerve`) verwendet wird und keine semantische Bedeutung darstellt.

Beispiel

Aus der Cyc-Konstante `#$Virgo-Constellation` ergibt sich der Begriff **Virgo (constellation)**.

Die Konstante `#$OpticNerve` entspricht den Artikeln **Optic nerve** (Medizin) und **Optic Nerve** (Buch).

Im nächsten Schritt werden Konzepte von Cyc auf Artikel von Wikipedia abgebildet. Dabei wird eine Abbildung auf Mehrdeutigkeitsseiten (engl. „Disambiguation pages“) von Wikipedia nicht erlaubt, da solche Seiten kein Konzept beschreiben und nur als Verweise zu mehreren Artikel verwendet werden. Dabei sind zwei Fälle möglich: eine Konstante kann entweder auf einen oder auf mehrere Artikel abgebildet werden.

Eindeutige Abbildung

Wenn bei der Suche keine Ergebnisse gefunden wurden, wird überprüft, ob eins der Synonyme der Konstante genau einen Titel oder einen Verweis in Wikipedia darstellt. Wenn eine Mehrdeutigkeitsseite ausgegeben wird, wird das Ergebnis unberücksichtigt.

Mehrdeutige Abbildung

In dem Fall, dass kein Ergebnis gefunden wird, wird der gesuchte Begriff vereinfacht, indem die Information in Klammern (falls vorhanden) entfernt wird. Außerdem werden auch Mehrdeutigkeitsseiten betrachtet, indem die Verweise aus diesen Seiten in die Liste der Suchergebnisse aufgenommen werden. Die Ergebnisliste besteht entsprechend aus folgenden Einträgen:

1. Artikel, deren Titel mit dem Suchbegriff übereinstimmen
2. Artikel, die über Verweise (engl. „Redirect“) erreicht wurden
3. Die ersten Artikel der ersten getroffenen Mehrdeutigkeitsseiten

Zusätzlich werden Verweisanker (engl. „link anchor“) abgeleitet. Ein Verweisanker ist der Text in einem Artikel, der als Hyperlink funktioniert und auf einen anderen Artikel verweist.

Die Gemeinsamkeit zwischen dem Suchterm \mathbf{a} und jedem Artikel \mathbf{T} wird nach der folgenden Formel berechnet:

$$\text{Commonness}_{\mathbf{a},\mathbf{T}} = P(\mathbf{T}|\mathbf{a}) \quad (3.2)$$

Dabei bedeutet die Gemeinsamkeit zwischen \mathbf{a} und \mathbf{T} die Anzahl von Artikeln in Wikipedia, in denen \mathbf{a} ein Verweisanker zu \mathbf{T} ist, geteilt durch die gesamte Menge der Artikel, in denen \mathbf{a} ein Verweisanker ist.

Beispiel

Das Wort „Jaguar“ fungiert in 927 Artikeln als Verweisanker. Dabei verweist es 426 mal auf den Artikel „Jaguar cars“. Die Gemeinsamkeit der Abbildung („Jaguar“, „Jaguar cars“ ist deswegen 0.5).

Anschließend werden zu jedem Cyc Term, zu dem keine eindeutige Abbildung gefunden wurde, jeweils fünf Artikel aus Wikipedia mit den größten Gemeinsamkeiten zu diesem Term ausgewählt und in die Liste der möglichen Begriffe aufgenommen.

Um aus den gefundenen Artikeln den passenden zu finden, muss jetzt die semantische Ähnlichkeit zwischen jedem Artikel und dem Kontext von Cyc Term berechnet werden. Der Kontext einer Cyc-Konstante wird als eine Menge der am meisten verbundenen Cyc-Konstante definiert. Dazu zählen direkte Ober- und Unterbegriffe, Geschwister-Begriffe (Unterbegriffe von den direkten Oberbegriffen) und Begriffe, die durch ausgewählte Prädikate wie `#$conceptuallyRelated` oder `#$countryOfCity` mit der jeweiligen Konstante verbunden sind. Wenn eine Konstante einen Bindestrich enthält (z.B. `#$Tool-MusicGroup`) wird das Wort nach dem Bindestrich ebenfalls in die Liste der Kontextterme aufgenommen.

Im nächsten Schritt werden alle gefundenen Kontextterme auf Artikel in Wikipedia abgebildet. Dabei werden nur eindeutige Abbildungen betrachtet. Zu jedem möglichen Artikel

\mathbf{x} in Wikipedia zu der Cyc-Konstante und jedem Wikipedia-Artikel \mathbf{y} zu den Kontextkonstanten werden zuerst die Mengen \mathbf{X} und \mathbf{Y} der Verweise auf \mathbf{x} und \mathbf{y} ermittelt. Mit diesen Mengen wird die semantische Ähnlichkeit zwischen \mathbf{x} und \mathbf{y} berechnet:

$$SIM_{x,y} = 1 - \frac{\max(\log|X|, \log|Y|) - \log|X \cap Y|}{N - \min(\log|X|, \log|Y|)} \quad (3.3)$$

Dabei ist N die gesamte Anzahl der Artikel in Wikipedia. Für jeden Artikel in der Liste der möglichen Abbildungen wird die durchschnittliche Ähnlichkeit berechnet und mit der Gemeinsamkeit multipliziert. Ein Artikel mit der höchsten solchen Bewertung stellt die gesuchte Abbildung dar.

Abschließend wird die Menge der gefundenen Abbildungen gefiltert und Abbildungen mit einer niedrigen Bewertung gelöscht.

Fazit

Zur Evaluation der Ergebnisse wurden Testkonstanten einmal von Testpersonen und einmal vom entwickelten Algorithmus auf Wikipedia Artikel abgebildet. Die Abbildungen stimmten zu 39,8% überein. Obwohl der Algorithmus in manchen Fällen eine Abbildung auf abstraktere Artikel gefunden hat (z.B. **#\$Crop** auf **Agriculture** statt auf **Crop (Agriculture)**) hat der Algorithmus bessere Ergebnisse als zwei der Testpersonen geliefert.

Diese Arbeit hat eine Möglichkeit der Kombination von Vorteilen des allgemeinen Verstands von Cyc und der Ausbeute von Wikipedia gezeigt. Die Ermittlung von Kontexten der Cyc-Konstanten kann auch bei der Ermittlung von Domänenwerten eines AO-Knoten verwendet werden.

4. Analyse

Im Projekt EASIER werden unterschiedliche Webformulare in Kategorien aufgeteilt und jede Kategorie wird auf eine aktive Ontologie abgebildet. Die Attribute der Eingabeelemente, die in jedem Formular enthalten sind, können verwendet werden, um die Menge der möglichen Eingaben zu bestimmen. In diesem Kapitel wird das Problem zuerst im Kapitel 4 untersucht und mögliche Lösungsansätze besprochen.

Die zu erstellende Software muss aus den verfügbaren Informationen über ein Formularelement den möglichen Wertebereich ausgeben können. Fast in allen Fällen geht es dabei entweder um Substantive oder Zusammensetzungen mit Substantiven. Aus diesem Grund konzentriert sich die dargestellte Lösung in erster Linie auf Substantive.

Als Eingabe stehen die HTML-Attribute zur Verfügung. Die Werte dieser Attribute beinhalten oft Beispieleingaben oder Oberbegriffe der möglichen Eingabewerte.

4.1. Analyse der Eingabedaten

In diesem Abschnitt wird besprochen, welche Daten die Eingabe für die Software darstellen können. Dafür wird der Aufbau und die Semantik eines Webformulars betrachtet. Anschließend wird die Struktur einer Eingabedatei dargestellt, die die ermittelten Informationen enthält.

4.1.1. Semantik in HTML-Formularen

Um eine Domäne der Substantive in natürlicher Sprache zu beschreiben, können entweder die Werte dieser Domäne oder ein gemeinsamer Bezeichner (Oberbegriff) genannt werden. Domänenwerte und Oberbegriffe befinden sich oft in den Attributen der jeweiligen Formularelemente.

Wie im folgenden Beispiel zu sehen ist, ist aus dem Code des Elementes ersichtlich, dass eine Stadt, der Name oder die Kodierung des Flughafens einzugeben sind. Der Wert des Parameters „name“ besteht aus den Wörtern „origin“ und „airport“. Das Letztere ist ein Oberbegriff für die gewünschte Eingabe. Im Placeholder sind zwei Oberbegriffe zu finden: „city“ und „airport“. Die Liste aller Städte und Flughäfen stellt somit eine mögliche Eingabedomäne dar. Außerdem wird das Textfeld mit der Beispieleingabe „MHG“ vorbelegt. Daraus kann erschlossen werden, dass eine Codierung des Flughafens ebenfalls eingegeben werden kann.

Beispiel

Im Webformular von der Fluggesellschaft „American Airlines“ [ame] kann der Benutzer den gewünschten Flughafen in ein Textfeld eingeben. Der Code vom Textfeld auf der Startseite lautet:

```
<input type="text" name="originAirport" value="MHG" id="
  reservationFlightSearchForm.originAirport" class="
  aaAutoComplete" placeholder="City_or_airport" >
```

Obwohl den meisten Attributen aussagekräftige Werte zugewiesen wurden (z.B. „originAirport“), können Attributen trotzdem irrelevante (z.B. „name1“) oder zu abstrakte (z.B. „object“) Werte enthalten. Außerdem ist es wichtig zu unterscheiden, ob es sich bei dem Wert (wenn dieser plausibel ist) um eine Beispieleingabe oder den Oberbegriff für Eingaben handelt. Das ist von der Art des Parameters abhängig. Dies wird auf drei empfundenen Beispielen illustriert:

Im folgenden Beispiel ist „Poodle“ eine Beispieleingabe, die dem Parameter „value“ zugewiesen wird. Als Eingabe werden mögliche Hunderassen erwartet.

Beispiel

Ein Tierarzt spezialisiert sich auf Behandlung von Hunden und bietet seinen Kunden eine Terminvorgabe über ein Online-Formular. Der Benutzer muss dabei die Rasse seines Hundes eingeben:

```
<input type="text" name="dogBreed" value="Poodle" id="
  doctorAppointment.dogBreed" placeholder="Enter_the_breed_of_
  your_dog" >
```

Hier ist „Poodle“ ein Oberbegriff der erwarteten Eingaben und der Name des Textfelds.

Beispiel

Auf der Startseite von dem Verein „Poodle Club of Europe“ befindet sich ein Formular für die Registrierung neuer Mitglieder. Der Benutzer kann bei der Registrierung die Unterart von seinem Pudel angeben, um passende Pflegeprodukte angeboten zu bekommen.

```
<input type="text" name="poodle" value="Toy_Poodle" id="
  registrationForm.poodleType" >
```

Hier ist es nicht eindeutig, ob „Poodle“ ein Oberbegriff oder ein Beispiel des einzugebenen Wertes ist.

Beispiel

In diesem Beispiel ist der Kontext nicht bekannt.

```
<input type="text" name="inputField" id="anonymousForm.inputField"
  placeholder="Poodle" >
```

Somit können die erkannten Wörter Beispielangaben, Oberbegriffe oder überhaupt nicht-relevante Konzepte sein. Die Wahrscheinlichkeit, dass der Wert des Attributes eine benötigte Information enthält, hängt vor allem vom Attributen ab. Im folgenden Abschnitt wird dies genauer erläutert.

Tabelle 4.1.: Wahrscheinlichkeiten des Auftretens der Oberbegriffe und Domänenwerte

Parameter	P(enthält Oberbegriff)	P(enthält Domänenwert)	P(nicht relevant)
label	0.4	0	0,6
value	0.25	0.75	0
placeholder	0.46	0.08	0,46
name	0.35	0	0,65
id	0.43	0	0,57

Tabelle 4.2.: Verwendete Muster

Muster	Beispiel
Substantiv	station
Substantiv + Substantiv	lava lamp
Adjektiv + Substantiv	bald eagle
Eigennamen	The Rolling Stones

4.1.2. Klassifizierung der Begriffe

Obwohl nicht alle Formulare nützliche Informationen liefern, ist es in den meisten Fällen möglich, Oberbegriffe und Beispielwerte zu erkennen oder abzuleiten. Um zu bewerten, mit welcher Wahrscheinlichkeit ein Attribut einen Domänenwert oder einen Oberbegriff von einem Domänenwert enthält, wurden Formulare von 30 unterschiedlicher Dienstleister ausgewählt, um eine Statistik zu berechnen. Für die Attribute `value`, `placeholder`, `name` und `id` und für den jeweiligen Label wurde gezählt, wie oft diese einen relevanten Begriff enthalte. In der Tabelle 4.1 ist die Auswertung dieser Statistik vorgestellt:

4.1.3. Lexikalische Analyse

Da die Lösung sich auf die Ermittlung der Substantive konzentriert, wird auf den erkannten Begriffen eine lexikalische Analyse durchgeführt, um nicht relevante Begriffe wie Präpositionen oder Adjektive auszuschließen. Dabei bleiben Wortbindungen mit Substantiven und Eigennamen in der Domäne. Wie in [Gee17] werden Wortbindungen mit Substantiven als Unterarten von Substantiven betrachtet (z.B. „Vatican city“ ist ein Unterbegriff für „city“). Eigennamen wie beispielsweise der Name der Rockband „The Who“ bestehen oft aus unterschiedlichen Wortarten, können aber Unterbegriffe von Substantiven sein und somit lässt sich die Hierarchie von Substantiven für solche Anwendungsfälle ebenfalls nutzen. In der Tabelle 4.2 sind die bei der Filterung verwendeten Muster erklärt.

4.1.4. Eingabe

Die Attribute `placeholder`, `name`, `id`, `value` und dem Element zugewiesener `Label` enthalten oft relevante Begriffe und müssen somit der Software übergeben werden. Im EASIER werden unterschiedliche Webformulare abhängig vom Anwendungsfall auf jeweils gleiche aktive Ontologien abgebildet. Dabei entsprechen semantisch gleiche Webformularelemente einem jeweils gleichen AO-Knoten. Aus diesem Grund wird nicht ein Wert zu jedem Attribut / Label, sondern eine Menge der Werte zu jedem Attribut / Label eingegeben.

Die Struktur der Eingabedatei kann somit als ein Baum, wie in der Abbildung 4.1, abgebildet werden.



Abbildung 4.1.: Die Struktur der Eingabedatei

4.2. Ermittlung weiterer Begriffe

Um weitere Begriffe zu ermitteln, werden externe Datenquellen abgefragt. Dabei wird nach neuen Oberbegriffen gesucht, um anschließend die Unterbegriffe auszugeben. Somit erweitern sich die bestehenden Listen der Domänenwerte und Oberbegriffe auf weitere Begriffe aus den externen Datenquellen. Bei der Wahl der externen Datenquelle müssen die folgenden Anforderungen berücksichtigt werden:

1. Oberbegriffe der gegebenen Begriffe müssen ermittelbar sein
2. Unterbegriffe der gegebenen Begriffe müssen ermittelbar sein

WordNet, ResearchCyc und Wikipedia erfüllen diese Anforderungen. Zudem sind diese Datenquellen für unterschiedliche Zwecke gedacht: WordNet ist eine Taxonomie der englischen Sprache, Wikipedia ist eine Enzyklopädie und Cyc ist eine Wissensdatenbank. Der gefundene Ansatz lässt sich somit auf ähnliche externe Datenquellen übertragen.

4.2.1. Vorverarbeitung der Daten

Zur Erstellung einer Abfrage an eine externe Datenquelle werden Listen der Oberbegriffe und Domänenwerte verwendet. Diese Listen beinhalten nicht eindeutige Einträge, sondern Einträge, die mit einer bestimmten Wahrscheinlichkeit richtig sind. Jedem Begriff, der im Laufe des Programms eingegeben, verwendet oder ausgegeben wird, wird eine Bewertung zugewiesen. Weiter wird diese als $P(\text{Wert})$ bezeichnet. Folglich muss eine Reihe der Rechenregeln zur Berechnung dieser Bewertung erstellt werden.

Die ursprüngliche Bewertung der Begriffe entspricht den Einträgen in in der Tabelle 4.1 für jeweiliges Attribut.

Tabelle 4.3.: Verwendete Präfixe

Präfix	Folgender Begriff
„Enter the“	Oberbegriff
„Enter a“	Oberbegriff
„Please provide a“	Oberbegriff
„Please provide the“	Unterbegriff
„e.g.“	Domänenwert
„Example:“	Domänenwert

Präfixe

Mithilfe der bestimmten Präfixen kann ermittelt werden, dass danach ein Oberbegriff oder ein Domänenwert folgt.

Beispiel

Ein Webformular ist mit dem Placeholder „Enter the city“ belegt. Der richtige Oberbegriff der möglichen Eingaben ist in diesem Fall „city“.

Wenn mithilfe eines Präfixes ein Begriff eindeutig bestimmt werden kann, in welche Liste der darauffolgende Begriff eingeteilt wird, müssen dieses Präfix entfernt und seine Bewertung auf 1.0 gesetzt werden. In der Tabelle 4.3 werden solche Präfixe dargestellt. Die Groß- und Kleinschreibung wird nicht beachtet.

Zusammengesetzte Begriffe

Außer einzelnen Wörtern können zusammengesetzte Begriffe oder Wortbindungen auftreten. Die Zusammensetzung der Wörter kann entweder durch ein Trennzeichen (Leerzeichen, Satzzeichen oder andere Zeichen) kenntlich gemacht werden, oder durch Groß- und Kleinschreibung. Dabei kann entweder eine Untermenge der Wörter oder die ganze Zusammensetzung den gesuchten Begriff darstellen. Solche Zeichenketten werden aufgeteilt und jeder Teil wird aufgenommen.

$$P(\text{Teil}) = P(\text{Zeichenkette}) \quad (4.1)$$

Beispiel

Im Placeholder ist der Satz „Enter the desired City“ eingetragen. Das Präfix „Enter the“ wird gelöscht. Aus dem Rest des Satzes werden folgende Begriffe in die Liste der Oberbegriffe aufgenommen:

1. „desired City“ mit $P(\text{„desired City“}) = 1.0$
2. „desired“ mit $P(\text{„desired“}) = 1.0$
3. „City“ mit $P(\text{„City“}) = 1.0$

Stoppwörter

Artikel, Präpositionen und andere nicht suchrelevante Wörter müssen ausgefiltert werden. Ebenfalls werden die Namen der HTML-Attribute oft vorkommen (z.B. „name“ entsteht bei der Teilung von „originName“). Aus diesem Grund wird eine Datei mit einer Liste der

Stoppwörter erstellt. Falls solch ein Wort sich in dieser Liste befindet, wird dieses Wort in keine Liste der Begriffe aufgenommen.

Dabei ist zu beachten, dass in mehreren Fällen (insbesondere bei Eigennamen) ein Stoppwort ein Teil des Begriffes sein kann. „Frankfurt on the Main“ ist beispielsweise eine richtige Bezeichnung der Stadt Frankfurt, und bei der Suche nach „Frankfurt Main“ eine externe Datenquelle keine Ergebnisse liefern kann. Um dies zu vermeiden, werden Stoppwörter nicht aus den ursprünglichen Begriffen gelöscht, sondern nur die Begriffe, die in der Liste der Stoppwörter enthalten sind.

Beispiel

Der Begriff „an Example Label“ wird wie folgt aufgeteilt

1. „an Example Label“ mit $P(\text{„an Example Label“}) = x$
2. „an Example“ mit $P(\text{„an Example“}) = x$
3. „Example Label“ mit $P(\text{„Example Label“}) = x$

Die folgenden Begriffe werden nicht akzeptiert, da diese Stoppwörter sind:

1. „an“
2. „Label“

4.2.2. Befragung der Datenquellen

Nach der Vorverarbeitung der Eingabedaten entstehen Listen von Oberbegriffen und Domänenwerten, wobei jedem Begriff eine Bewertung seiner Richtigkeit zugewiesen ist. Durch eine Suche nach den Oberbegriffen der bekannten Domänenwerten und anschließende Ausgabe der Unterbegriffe dieser Oberbegriffe ergibt sich der gesuchte Wertebereich eines AO-Knotens.

Um die Richtigkeit der neuen Begriffe zu bewerten, gelten die folgenden Formeln:

Ermittlung eines gemeinsamen Oberbegriffs

Durch Befragung der externen Datenquelle können gemeinsame Oberbegriffe von zwei bekannten Domänenwerte ermittelt werden. Die ermittelten Oberbegriffe sind dann korrekt, wenn diese mithilfe der korrekten Unterbegriffe ermittelt wurden. Folglich wurde für die Bewertung des Oberbegriffes die UND-Formel der Wahrscheinlichkeitstheorie verwendet:

$$P(\text{Oberbegriff}) = P(\text{UND}(\text{wert1}, \text{wert2})) = P(\text{wert1}) * P(\text{wert2}) \quad (4.2)$$

Ermittlung der Unterbegriffe eines Oberbegriffes

Bei der Befragung wird angenommen, dass die externen Datenquellen einen korrekten Inhalt haben. Wenn nach den Unterbegriffen eines Oberbegriffes gesucht wird, sind die Unterbegriffe dann richtig, wenn der Oberbegriff richtig war:

$$P(\text{Unterbegriff}) = P(\text{Oberbegriff}) \quad (4.3)$$

Mehrfache Ermittlung eines Wertes

Wenn bei der Abfrage der Datenbanken neue Domänenwerte oder Oberbegriffe ermittelt werden, werden diese gespeichert. Falls es diesen Wert in der Liste bereits gibt, ist dieser Wert dann der Oberbegriff oder Domänenwert, wenn mindestens einer der Wege seiner Ermittlung richtig ist und korrekte Ergebnisse liefert. Die Berechnung neuer Wahrscheinlichkeit erfolgt somit nach der ODER-Formel:

$$P_{neu}(wert) = P_1(wert) + P_2(wert) - P_1(wert1) * P_2(wert2) \quad (4.4)$$

Der konkrete Algorithmus der Abfrage hängt von der Struktur der jeweiligen Datenbank ab.

4.2.2.1. Abfrage von WordNet

In WordNet werden Begriffe der englischen Sprache in eine Taxonomie organisiert. Mit Hilfe von WordNet können Synonyme, Unter- und Oberbegriffe von Wörtern leicht ermittelt werden. Daher erfüllt WordNet allen gestellten Anforderungen und kann als eine Datenquellen verwendet werden. Für andere Sprachen existieren ähnliche Taxonomien wie GermaNet für Deutsch, die ähnlich aufgebaut sind und auf gleicher Weise wie WordNet abgefragt werden können.

Synonyme werden in WordNet in Synsets gruppiert. Folglich muss bei der Eingabe eines Begriffes zuerst der richtige Synset gefunden werden. Falls ein Oberbegriff bekannt ist, können alle direkten Kinder seines Synsets in den Wertebereich aufgenommen werden (Abbildung 4.2). Diese Kinder stellen den gesuchten Wertereich dar.

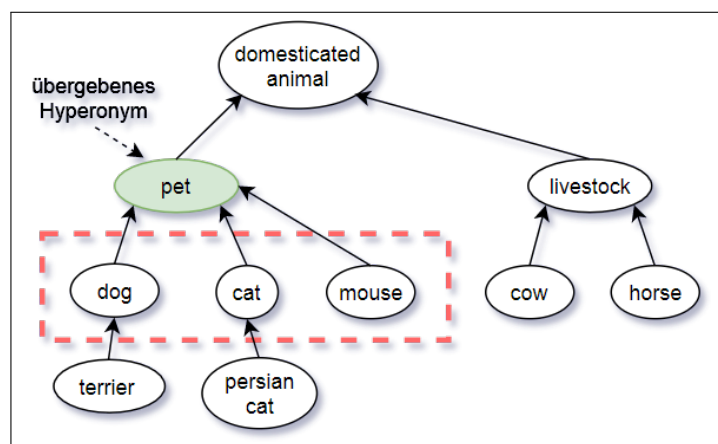


Abbildung 4.2.: Suche nach den Kindern eines Oberbegriffs. Das Rechteck schließt die ausgegebenen Domänenwerte um

Um die Genauigkeit des Ansatzes zu verbessern und weitere Konzepte zu finden, können auch vorhandene Beispielwerte verwendet werden, um den kleinsten gemeinsamen Oberbegriff zu finden (Abbildung 4.3).

Wenn nach einem gemeinsamen Oberbegriff zweier komplett unterschiedlicher Begriffe gesucht wird, wird dieser sich auf einer zu hohen Abstraktionsebene befinden (Abbildung 4.4).

Um dies zu vermeiden, müssen nur Konzepte mit starker semantischer Ähnlichkeit betrachtet werden. Semantische Ähnlichkeit ist dabei ein Wert, der angibt, wie ein Konzept einem anderen entspricht oder ähnlich ist. Das Konzept „Boot“ ist dem Konzept „Auto“ ähnlicher, als dem Konzept „Baum“, da sie in der Hierarchie von WordNet einen gemeinsamen Elternknoten „Fahrzeug“ besitzen [PPM04].

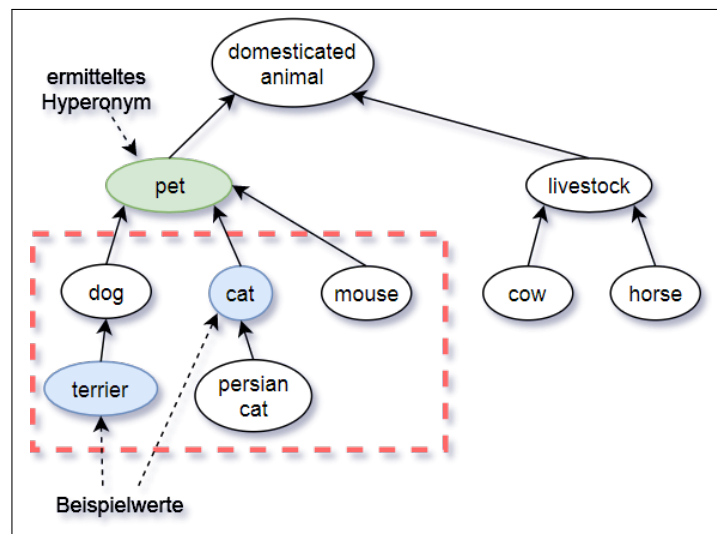


Abbildung 4.3.: Suche nach dem kleinsten gemeinsamen Oberbegriff. Das Rechteck schließt die ausgegebenen Domänenwerte um

Um nur semantisch ähnliche Konzepte zu betrachten, bietet sich der Wu-Palmer Algorithmus an. Der Wert der semantischen Ähnlichkeit wird durch die folgende Formel berechnet (depth(a) ist dabei der Abstand vom Knoten a zum Wurzelknoten) [WP94]:

$$\text{score} = 2 * (\text{depth}(\text{kleinster gemeinsamen Oberbegriff})) / (\text{depth}(\text{Konzept 1}) + \text{depth}(\text{Konzept 2})) \quad (4.5)$$

Der Algorithmus zur Abfrage von Wordnet kann somit aus den folgenden Schritten bestehen:

1. Für jedes unterschiedliche Paar der Werte aus der Liste der bekannten Domänenwerte :
 - 1.1. Alle Synsets dieser Werte ermitteln
 - 1.2. Wenn es einen Synset zu einem Wert und einen Synset zum zweiten Wert gibt, so dass die semantische Ähnlichkeit kleiner als der Schwellwert ist, den Synset des Oberbegriffes ermitteln und den maximalen Abstand **n** zwischen diesen Synset und dem gefundenen Oberbegriff.
 - 1.3. Die Zeichenketten dieses neuen Synsets in die Liste der Oberbegriffe aufnehmen
2. Für jeden Wert in der Liste der Oberbegriffe:
 - 2.1. Natürlichsprachige Bezeichnungen seiner Kinder in die Liste der Domänenwerte aufnehmen.
 - 2.2. Falls dieser Wert im Schritt 1. ermittelt wurde, werden seine Kinder bis zur Tiefe **n** ausgegeben.
3. Die Liste der Domänenwerte nach der Wahrscheinlichkeit absteigend sortieren.

WordNet ist für ähnliche Aufgaben wie das Ziel dieser Arbeit gedacht und kann Ergebnisse mit hoher Genauigkeit liefern. Der Nachteil von WordNet ist, dass nur oft verwendeten Konzepte enthalten sind. Deswegen bietet WordNet nicht die beste Ausbeute im Vergleich zu anderen Datenquellen.

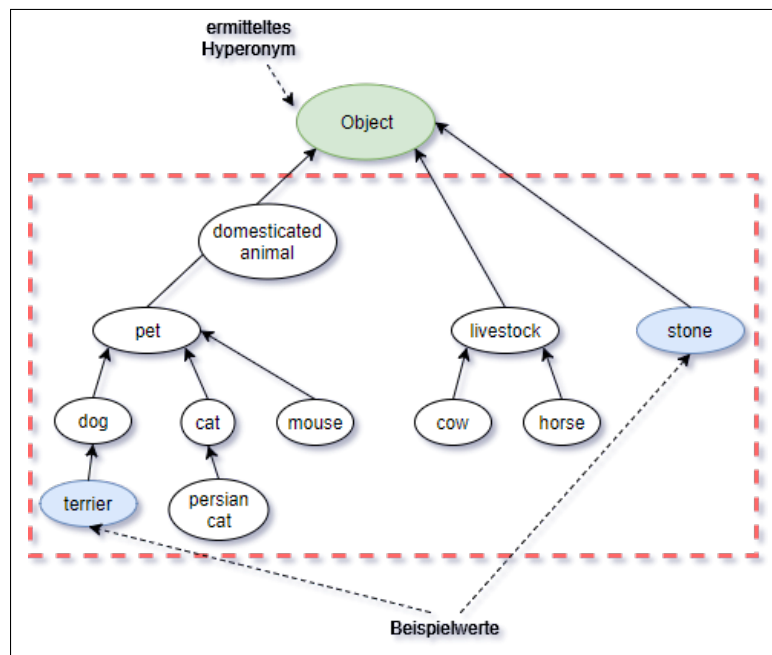


Abbildung 4.4.: Suche nach dem kleinsten gemeinsamen Oberbegriff. Der Oberbegriff „Object“ ist zwar richtig, aber zu abstrakt, um behilflich zu sein

4.2.2.2. Abfrage von Wikipedia

Wikipedia ist eine beliebte Plattform zur Suche nach unterschiedlichen Informationen unter den Nutzern aller Welt. Da der Inhalt und die Struktur von Wikipedia leicht extrahiert werden kann, kann diese ebenfalls als eine Datenquelle für die Software verwendet werden.

Der erste Schritt bei der Befragung von Wikipedia ist die Abbildung der bekannten Begriffe auf Artikeln von Wikipedia. Bei der Benutzung der Suchfunktion von Wikipedia werden mehrere Artikel ausgegeben, die das Suchwort im Namen oder in der Beschreibung enthalten. Um aus diesen Artikeln nur die auszuwählen, die dem Suchkonzept entsprechen, wird die Ähnlichkeit zwischen dem Suchwort und dem Titel des Artikels berechnet. Der Wert dieser Berechnung hängt von der Leveshtein Distanz und der Länge der Konzepte ab:

$$Similarity_{x,y} = 1 - \frac{levenshteinDistance(x,y)}{\max(length(x), length(y))} \quad (4.6)$$

Jedem der gefundenen Artikel wird ähnlich den Domänenwerten eine Bewertung zugewiesen. Diese Bewertung ist proportional dem Grad an Sicherheit, dass der Artikel eine Entität beschreibt, dessen Bezeichnung einen Domänenwert darstellt. Der Wert ist der Bewertung von der Bewertung des Suchworts und der Ähnlichkeit zwischen dem Suchwort und dem Titel des Artikels proportional. Nach der folgenden Formel wird diese Bewertung berechnet:

$$P_{Artikel} = P_{Suchwort} * Similarity_{Suchwort, Titel_{Artikel}} \quad (4.7)$$

Wenn ein Artikel bei der Suche nach verschiedenen Suchbegriffen gefunden wurde, erfolgt die Zusammenfassung in eine Liste nach den gleichen Regeln, wie bei Domänenwerten und Oberbegriffen.

Zusätzlich zum Namen enthalten viele Artikel in Wikipedia einen Oberbegriff in Klammern (z.B. „Chihuahua (dog)“ oder „Chihuahua (city)“). Wenn der in Klammern stehende Begriff mit einem der bekannten Oberbegriffe übereinstimmt, wird die Bewertung erhöht. Der Wert, um den die Bewertung erhöht wird, ist proportional der Bewertung des Oberbegriffs:

$$P_{\text{Artikel}(\text{Oberbegriff})} = P_{\text{Artikel}} + P_{\text{Oberbegriff}} - P_{\text{Artikel}} * P_{\text{Oberbegriff}} \quad (4.8)$$

Um die Anzahl der Abfragen zu reduzieren und eine bessere Genauigkeit des Ansatzes zu erreichen, werden Artikel mit zu kleiner Wahrscheinlichkeit nicht betrachtet.

Im Gegenteil zu WordNet stellt der Inhalt von Wikipedia keine strikte Taxonomie dar. Jeder Artikel gehört in der Regel mehreren Kategorien. Dadurch, dass Wikipedia nicht zur Verarbeitung der natürlichen Sprache gedacht ist, entsprechen diese Kategorien auch nicht immer dem Oberbegriff des im Artikel beschriebenen Objektes im allgemeinen Verstand. Der Artikel „Berlin“ gehört gleichzeitig 8 Kategorien, darunter „Capitals in Europe“, „City-states“, „Members of the Hanseatic League“. Außerdem entspricht ein Wort meisten mehreren Artikeln in Wikipedia. Das führt dazu, dass der Pfad zu der kleinsten gemeinsamen Kategorie sehr schwer zu finden ist und ein anderer Weg gefunden werden muss.

Die Abbildung von Oberbegriffen auf Kategorien ist in Wikipedia auch nicht immer möglich, da die Kategorien meistens nicht den Oberbegriffen der Artikel entsprechen, sondern vereinigen eine Menge der Artikel mit ähnlichen Merkmalen wie z.B. „People born in 1994“. Eine offensichtliche Lösung bei der Eingabe „Dog“ wäre die Ausgabe der Liste „List of dog breeds“. Allerdings werden in diesem Fall 16 verschiedene Listen ausgegeben. Aus diesem Grund wäre es sonnvoll, gleich nach den Kategorien der Beispielwerte zu suchen. Dabei müssen alle Kategorien bis zu einer gewählten Tiefe durchgesucht werden, um gemeinsame Kategorien für die Beispielwerte zu ermitteln. Diese Tiefe darf nicht zu klein sein, da die gesuchte Kategorie kann dann nicht gefunden werden. Diese Tiefe darf auch nicht zu groß sein, um die Ausgabe zu abstrakter Kategorien zu vermeiden.

Anstatt nach einem gemeinsamen Oberbegriff zu suchen, kann für jeden Beispielwert nach einer Liste seiner direkten Kategorien oder Oberkategorien der Kategorien gesucht werden. Ein Beispiel solcher Suche ist in der Gleichung 4.2.2.2 zu finden. Die Bewertung des Wortes ist gleich \mathbf{p} . Wenn bei der Suche mehrere Ergebnisse ausgegeben werden, wird davon ausgegangen, dass nur eins dieser Ergebnisse der gesuchte Artikel oder die gesuchte Kategorie ist. Den Suchergebnissen wird dann die Bewertung \mathbf{p} zugewiesen.

Anschließend werden alle gefundene Kategorien in eine Liste zusammengeführt. Ähnlich wie bei WordNet, steigt die Bewertung ihrer Richtigkeit mit der Anzahl von Auftreten jeweiliger Kategorie. Im nächsten Schritt wird nach den Unterkategorien / Artikeln der gefundenen Kategorien gesucht. Die Bewertung der jeweiligen Ausgabewertes soll dann der Bewertung der ursprünglichen Kategorie entsprechen.

Die mögliche Lösung der gestellten Aufgabe kann unter Verwendung von Wikipedia wie folgt aussehen:

1. Für jeden Wert aus der Liste der bekannten Domänenwerte :
 - 1.1. Suche nach den Artikeln in Wikipedia, dessen Titel diesen Wert enthalten
 - 1.2. Suche nach allen Kategorien bis zur festgelegter Tiefe
 - 1.3. Für die am meisten wahrscheinlichen Kategorien die Liste der Unterkategorien und Artikel bis zur gewünschten Tiefe ausgeben.
 - 1.4. Wenn im Titel des gefundenen Artikels ein Oberbegriff in den Klammern steht, die Bewertung dieses Artikels erhöhen.

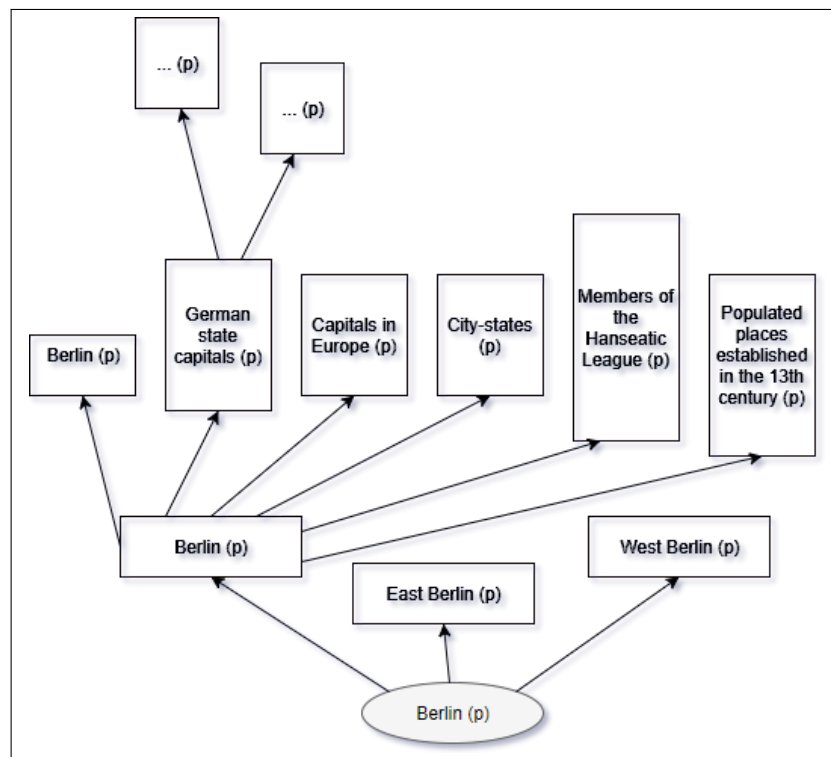


Abbildung 4.5.: Suche nach den Artikeln in Wikipedia und Kategorien dieser Artikeln bei der Eingabe eines Wortes. In Klammern ist die jeweilige Wahrscheinlichkeit.

Dadurch, dass Wikipedia von der Wikipedia-Community aus der ganzen Welt befüllt ist und über mehr Inhalt als WordNet verfügt, kann diese Methode eine bessere Ausbeute bieten als Benutzung von WordNet. Da mehrere Pfade durchgesucht werden müssen, wird eine große Anzahl der Fehlkonzepte ausgegeben und somit senkt die Genauigkeit des Verfahrens im Vergleich zur Abfrage von WordNet.

4.2.2.3. Abfrage von Cyc

Cyc ist eine Datenbank mit dem Inhalt, der dem allgemeinen Verstand eines Menschen darstellen soll. Cyc eignet sich für viele Aufgaben, die von einem Menschen gelöst werden können. Der eingebaute Schlussfolgerungsmechanismus ermöglicht es, genaue und logische Ergebnisse zu bekommen. Deswegen kann Cyc als eine weitere Datenquelle für die Software verwendet werden.

Die Inferenzmaschine (eng. „Inference Engine“) von Cyc kann aus einer Menge von Behauptungen neue Behauptungen ableiten. Außerdem ist es möglich, durch Verwendung von Variablen eine Menge von Individuals (Informationseinheiten in Cyc) zu finden, für die eine gegebene Annahme erfüllt ist.

Beispiel

Die Behauptung (*#\$isa #CityOfBerlinGermany #City*) bedeutet, dass die Konstante *#CityOfBerlinGermany* der Gruppe *#City* gehört und ist als den Satz „Berlin ist eine Stadt“ zu verstehen.

In der Anfrage (*#\$isa ?X #City*) steht an der Stelle des ersten Argumentes vom Prädikat *#\$isa* eine Variable. Cyc wird jetzt alle Konstanten ausgeben, für die diese Annahme gilt, z.B. *#CityOfBerlinGermany*, *#CityOfFrankfurtGermany*, *#CityOf*

MoscowRussia usw.

Als Eingabe bekommt die Software Konzepte natürlicher Sprache. Aus diesem Grund müssen diese erst auf Cyc Konstanten abgebildet werden. Dafür gibt es im Cyc das Prädikat „termPhrases“, das eine Cyc-Konstante auf eine oder mehrere Zeichenketten abbildet.

Da Cyc ebenfalls wie Wikipedia keine strikte Taxonomie ist, ist die Suche nach dem kleinsten gemeinsamen Oberbegriff erschwert. Es besteht zwar die Möglichkeit, unnötige Behauptungen durch Angabe der Mikrotheorie (Wissensdomäne) abzugrenzen. Aber da die Lösung für verschiedene Wissensdomänen gelten muss, müssen alle Annahmen betrachtet werden.

Hier wäre es möglich, wie bei der Abfrage von Wikipedia vorzugehen, alle Gruppen von den Beispielwerten zu sammeln und deren Inhalt auszugeben. Allerdings dadurch, dass Cyc zur Entwicklung von künstlicher Intelligenz gemacht ist, bietet es bessere Mechanismen zur Suche in seinem Inhalt.

Beispiel

Nach der Sortierung der Konzepte ergibt sich die folgende Liste der Beispielwerte: Berlin, Frankfurt, London, from, enter, Moscow....

Angenommen, ein Mensch hat die Aufgabe, aus den Wörtern „London“, „To“ und „Bird“ das passende Wort zu wählen. Am wahrscheinlichsten ist, dass der Menschen das Wort „London“ wählt. Das ist sinnvoll, weil:

1. Dieses Wort sich in der gleichen Domäne befindet, wie eins oder mehrere Wörter aus der Liste.
2. Dieses Wort sich in der Domäne der meisten Wörter in der Liste befinden.

Um die Domäne zu finden, wird nach allen Cyc-Konstanten gesucht, die den gleichen Gruppen wie jede zwei bekannte Cyc-Konstanten gehören. Ähnlich wie bei WordNet besteht die Gefahr, dass die gemeinsame Gruppen zu abstrakt sind und falsche Konstanten eine Auswirkung auf das Ergebnis haben können. Um dies zu verhindern, muss eine Alternative zur Berechnung der semantischen Ähnlichkeit zweier Cyc-Konstanten gefunden werden. Die Konstante **#\$ClarifyingCollectionType** (Erklärungsgruppe) bezeichnet alle Gruppen in Cyc, die zur Aufklärung der Wörter in natürlicher Sprache gedacht sind. Ein Beispiel hierfür wäre eine Erklärungsgruppe **#\$City** zu der Konstante **#\$CityOfBerlinGermany**.

Somit kann der Algorithmus zur Abfrage von Cyc aus den folgenden Schritten bestehen:

1. Für jeden bekannten möglichen Oberbegriff
 - 1.1. Eine Gruppe der Individuals ermitteln, die den gleichen Namen wie der Oberbegriff trägt
 - 1.2. Die in dieser Gruppe enthaltenen Individuals in die Liste der Domänenwerte aufnehmen
2. Für jedes unterschiedliche Paar der Werte aus der Liste der Domänenwerte :
 - 2.1. Wenn die Anzahl gemeinsamer Erklärungsgruppe größer als der Schwellwert ist:
 - i. Die Cyc-Konstanten zu diesen Werten ermitteln.
 - ii. Alle Konstanten der gleichen Domäne ermitteln und in eine Liste einfügen

- iii. Die Bezeichnungen dieser Konstanten in natürlicher Sprache in die Liste der Domänenwerte aufnehmen

Dank dem Schlussfolgerungsmechanismus können bei einer richtig erstellten Abfrage viele unnötige Ergebnisse ausgefiltert werden. Das ermöglicht eine bessere Genauigkeit als bei Wikipedia. Cyc enthält zudem mehr Information als WordNet und kann somit eine bessere Deckung als WordNet gewähren.

4.2.3. Konsolidierung der Ergebnisse

Nach jeder Abfrage der externen Datenquellen erweitern sich die Listen von Oberbegriffen und Domänenwerten. Aus diesem Grund wäre es möglich, die Datenquellen iterativ abzufragen. Bei den Abfragen werden dabei die Ausgaben aus der vorherigen Datenquelle als Eingabe dienen. Allerdings kann es dazu führen, dass die Laufzeit des Programms steigt und mehrere redundante Schritte durchgeführt werden. Eine andere Möglichkeit wäre die Abfrage jeder Datenbank mit den ursprünglichen Daten. Der Benutzer muss zudem eine Möglichkeit haben, selbst zu entscheiden, ob alle Datenquellen abgefragt werden müssen.

Am Ende des Durchlaufs des Programms entsteht eine Liste der möglichen Domänenwerte mit Bewertungen der Richtigkeit dieser Domänenwerte. Die Begriffe, bei denen diese Bewertungen größer als ein Schwellwert sind, werden als der gesuchte Wertebereich ausgegeben.

Wie in [Gee17] zu sehen ist, können bestimmte Morpho-syntaktische Muster auf eine Hyponymie Relation zwischen Konzepten deuten (z.B. „sweet corn“ ist eine Unterart von „corn“). Bei der Konsolidierung der Ergebnisse wird die Bewertungen von den gefundenen Begriffen erhöht, falls diese auf einen bekannten Oberbegriff enden. Die folgende Formel wird verwendet, um die neue Bewertung zu berechnen:

$$P_{neu}(Begriff) = P_{alt}(Begriff) + P_{alt}(Endung) - P_{alt}(Begriff) * P_{alt}(Endung) \quad (4.9)$$

5. Entwurf und Implementierung

In diesem Kapitel wird der Aufbau und die Funktionsweise der zu erstellenden Software erklärt.

5.1. Entwurf

Das Ziel dieses Abschnittes ist, dem Leser einen Überblick über den Aufbau der zu erstellenden Software zu geben. Hierfür wird sowohl der allgemeine Aufbau, als auch der Aufbau einzelner Komponenten erklärt.

Die gestellte Aufgabe kann in folgenden Schritten zusammengefasst werden:

1. Auslesen der Begriffe aus der eingegebenen Datei
2. Klassifizierung der Attributwerte (Oberbegriff / Domänenwert)
3. Suche nach den domänenrelevanten Konzepten durch Befragung von Datenquellen.
4. Zusammenführung der ermittelten Domänenwerte

In der Abbildung 5.1 ist der Prozess etwas genauer illustriert. Der Aufbau der Software richtet sich an die abgebildeten Schritte. Die einzelnen Komponenten sind in den folgenden Abschnitten erklärt.

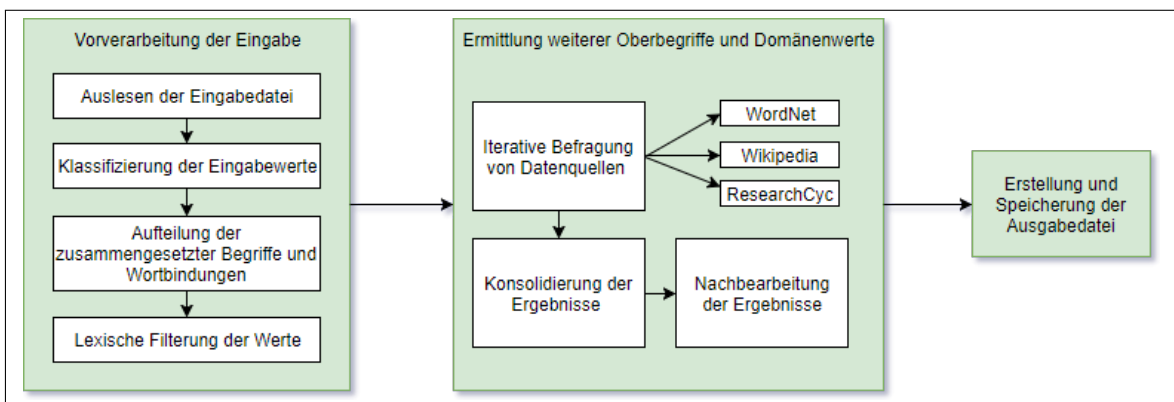


Abbildung 5.1.: Allgemeiner Ablauf des Programms

5.1.1. Vorbereitung der Daten

Der erste Schritt bei dem Programmablauf ist das Auslesen der Begriffe aus der eingegebenen Datei. Zu jedem Parameter aus der Datei muss eine Liste der Werte erstellt werden. Das Objekt, das diese Listen zusammenfasst, heißt **Node**. Zur Verarbeitung der Eingabedatei in ein Node wird die Klasse **NodeFactory** erstellt. **NodeFactory** bekommt die Adresse der Datei, liest alle enthaltene Werte aus und erstellt ein **Node** mit Listen der Werte für jeden Parameter (Abbildung 5.2).

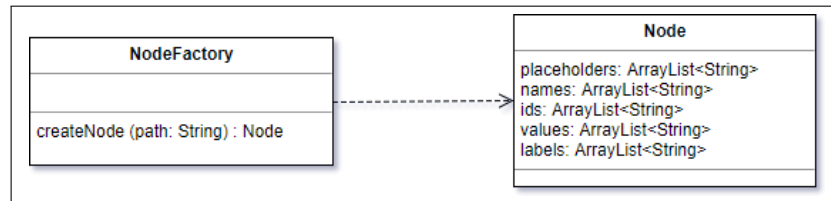


Abbildung 5.2.: Klassendiagramm der Knotenerstellung

Die erkannten Webformularelemente müssen im nächsten Schritt verarbeitet und in die Liste der Oberbegriffe und Domänenwerte mit ihren Wahrscheinlichkeiten aufgeteilt werden. Dafür wird eine Datenstruktur **Term** eingeführt, die ein Konzept und seine Wahrscheinlichkeit enthält. Zudem wird eine Datenstruktur **ConceptList** erstellt, in der beim Hinzufügen neuer Begriffe die im Kapitel „Analyse“ beschriebenen Berechnungsregeln berücksichtigt werden.

Die Aufteilung auf Oberbegriffe und Domänenwerte erfolgt mittels der Klasse **NodeProcessor**, die aus einem **Node** ein Objekt vom Typ **ProcessedNode** erstellt. Ein **ProcessedNode** enthält Listen von Oberbegriffen und Domänenwerten (Abbildung 5.3).

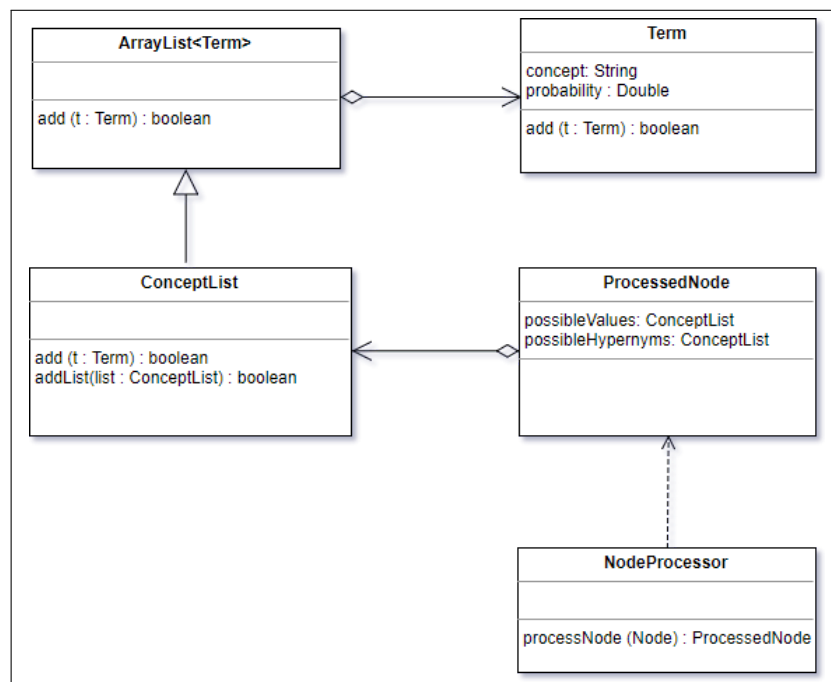


Abbildung 5.3.: Klassendiagramm der Klassifizierung von Begriffen

Die Komponente **ConceptDivider** enthält Methoden, die Teilwörter aus den erkannten Begriffen extrahieren und speichern. Anschließend erfolgt mittels der Klasse **LexicalFilter** die lexikalische Analyse und Filterung der Eingabe. Diese Klasse verfügt über Methoden,

die aus einer Liste alle Einträge entfernen, die keine Eigennamen sind und nicht den vorgegebenen syntaktischen Mustern entsprechen.

5.1.2. Befragung der Datenquellen und Zusammenführung der Ergebnisse

Wenn die Eingabe vorbereitet ist, kann die eigentliche Abfrage der Datenquellen durchgeführt werden. Dafür ist die Klasse **DomainFinder** vorgesehen, die ein **ProcessedNode** erweitert, d.h. weitere Oberbegriffe und Domänenwerte findet. Dies erfolgt indem Objekte vom Typ **DatabaseAccess** sequentiell aufgerufen werden. Es gibt drei Unterarten von **DatabaseAccess**: **CycAccess**, **WikiAccess** und **WordNetAccess**, die jeweils die Funktionalität zur Befragung der jeweiligen Datenbank einkapselt.

Die zweite Aufgabe von **DomainFinder** ist die Zusammenführung der Ergebnisse von Befragung unterschiedlicher Datenquellen.

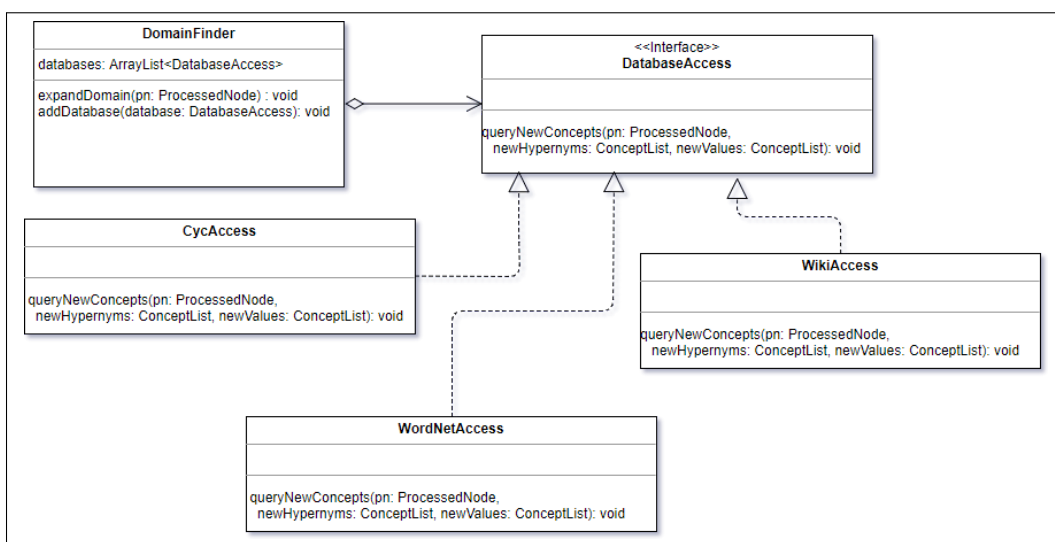


Abbildung 5.4.: Klassendiagramm der Befragung der Datenquellen

5.1.3. Weitere Komponenten

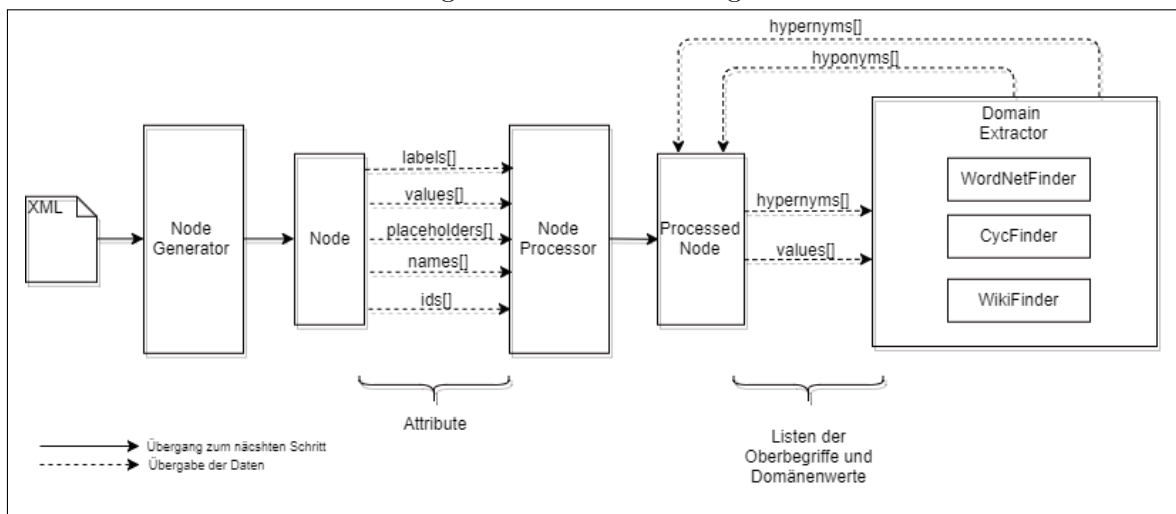
Um die Einstellungen ohne neue Kompilierung ändern zu können, ist die Klasse **Configuration** vorgesehen. Die Einstellungen, wie z.B. Time-out bei der Abfrage von Cyc oder die gewünschte semantische Ähnlichkeit bei dem Wu-Palmer Algorithmus, werden in einem JSON-File gespeichert und aus diesem File ausgelesen.

Nachdem alle Datenquellen abgefragt und die Ergebnisse zusammengeführt werden, speichert die Komponente **OutputWriter** den resultierenden Wertebereich in eine Datei zur weiteren Verwendung in EASIER.

5.1.4. Zusammenfassung

Die eingegebene Datei wird von der Komponente **NodeFactory** verarbeitet. Das Ergebnis dieser Verarbeitung, ein **Node**, dient als Eingabe für den **NodeProcessor**, der aus dem Inhalt von **Node** jeweils eine Liste von Oberbegriffen und Domänenwerten erstellt, wobei jedem Konzept eine Bewertung seiner Richtigkeit zugewiesen wird. Die Ausgabe von **NodeProcessor** ist ein Objekt vom Typ **ProcessedNode**, das diese Listen enthält. Die Komponente **DomainFinder** enthält Instanzen von **DatabaseAccess** (**CycAccess**, **WikiAccess** und **WordNetAccess**), mit denen die Listen im **ProcessedNode** durch

Abbildung 5.5.: Aufbau des Programms



Befragung der entsprechenden Datenbanken erweitert werden können. Der Aufbau ist in der Abbildung 5.5 illustriert.

Die Ausgabe der Software ist nach der Bewertung eine absteigend sortierte Liste der Domänenwerte von **ProcessedNode** nach der Anwendung von der Methode **expandDomain** der Komponente **DomainFinder**, die die Befragung der Datenquellen durchführt und die Listen der Oberbegriffe und Domänenwerte um neu entdeckte Konzepte aus diesen Datenquellen erweitert.

5.2. Implementierung

Die resultierende Software wurde in der Programmiersprache Java erstellt. In diesem Abschnitt wird die Implementierung der Software erklärt und die verwendeten Bibliotheken vorgestellt.

5.2.1. Vorbereitung der Daten

Als Eingabe bekommt die Software eine Datei im XML-Format. In der Abbildung 5.6 ist ein Beispiel solch einer Datei vorgestellt. Zu jedem der verwendeten HTML-Attributen sind Werte aus den zusammengefassten Elementen unterschiedlicher Formulare aufgelistet.

Zum Auslesen der Daten wurde die Bibliothek JDOM2 [jdo] verwendet, die aus einer XML-Datei eine Baumstruktur erstellt. Über diese Baumstruktur wird anschließend iteriert, um Listen von Zeichenketten zu den Attributen zu erstellen.

Bei der lexikalischen Filterung werden alle Einträge gelöscht, die nicht den vorgegebenen syntaktischen Mustern entsprechen und keine Eigennamen sind. Die Prüfung erfolgt mittels API Stanford CoreNLP [stab] und dem von Stanford angebotenen Part-Of-Speech Tagger [stad]. Zur Erkennung der Eigennamen (eng. „named entity recognition“) wurde der Klassifikator Stanford NER [stac] verwendet.

5.2.2. Befragung der Datenquellen

Jede der verwendeten Datenquellen verfügt über eine eigene Struktur. Aus diesem Grund musste der Algorithmus zur Abfrage an jede dieser Quellen angepasst werden. Zudem wurden unterschiedliche API's verwendet.

```
<node>
  <nodeName>testNode</nodeName>
  <placeholders>
    <placeholder>Departure airport</placeholder>
    <placeholder>Los Angeles</placeholder>
    <placeholder>example: Berlin</placeholder>
  </placeholders>
  <names>
    <name>originName1</name>
  </names>
  <ids>
    <id>From</id>
  </ids>
  <values>
    <value>Kopenhagen, Denmark</value>
    <value>Frankfurt am Main</value>
  </values>
  <labels>
    <label>Kopenhagen, Denmark</label>
    <label>Frankfurt am Main</label>
    <label>Frankfurt Airport</label>
  </labels>
</node>
```

Abbildung 5.6.: Beispiel der Eingabedatei

5.2.2.1. WordNet

Bei der Abfrage von WordNet werden zuerst neue mögliche Oberbegriffe der bekannten möglichen Domänenwerten ermittelt. Dazu werden zu jedem Paar der bekannten Domänenwerte x und y jeweils der kleinste gemeinsame Oberbegriff ermittelt, wenn die semantische Ähnlichkeit zwischen x und y nach dem WuPalmer Algorithmus kleiner als der Schwellwert ist.

Zum Zugriff auf den Inhalt von WordNet wurde die WS4J („WordNet Semantics for Java“) API [shi] von Hideki Shima verwendet. In dieser API werden unterschiedliche Algorithmen inklusive Wu-Palmer angeboten, daher entfällt die Notwendigkeit, diesen nochmal zu implementieren. Der Algorithmus zur Suche nach dem kleinsten gemeinsamen Oberbegriff ist ebenfalls in WS4J vorhanden. Der Zugriff zu WordNet erfolgt standardmäßig über eine SQL-Tabelle.

5.2.2.2. Wikipedia

Um eine Domäne mithilfe Wikipedia zu ermitteln, werden Kategorien zu jedem der bekannten Domänenwerte abgefragt. Die Plattform Wikipedia verfügt über eine serverseitige API MediaWiki, die nach HTTP-Anfragen eine Antwort an den Client senden kann. Als Format der Antwort wurde ein JSON-String gewählt.

Zur Erstellung der Client-Server Kommunikation wurde der HTTP Client von Apache verwendet. Nach der Antwort des Servers wird die erhaltene Zeichenkette mit der Bibliothek SimpleJSON ausgelesen. Die Berechnung der Levenshtein Distanz zwischen dem Suchwort und einem gefundenen Artikel erfolgt mittels API StringUtils von Apache Commons [str]. Um redundante Abfragen zu vermeiden, werden Antworten zu den Abfragen in einer Hashtabelle gespeichert und später aus dieser Hashtabelle ausgelesen.

5.2.2.3. Cyc

Um die Domäne der bekannten Werte mit Cyc zu erweitern, wird zuerst nach den Cyc-Konstanten gesucht, deren Beschreibung in natürlicher Sprache einer dieser Konstanten zugewiesenen Zeichenketten entspricht.

CycCorp bietet den Entwicklern die Java API CycApi an. Über CycApi können Abfragen in der Sprache CycL an den entfernten Server mit dem installierten ResearchCyc gestellt werden.

5.2.3. Weitere Komponenten

Die Konfigurationsdatei ist ein Textfile im JSON-Format. Zum Lesen der Konfigurationsdatei wird ebenfalls die Bibliothek SimpleJSON verwendet. Die Speicherung der Ergebnisse in einer XML-Datei erfolgt mittels einer Funktion die in JDOM2 enthalten ist. In der Abbildung Abbildung 5.7 ist ein Beispiel der Ausgabe illustriert.

```
<node>
  <value>
    <concept>victoria</concept>
    <probability>0.9152806949861108</probability>
  </value>
  <value>
    <concept>corozal</concept>
    <probability>0.822021484375</probability>
  </value>
  <value>
    <concept>London</concept>
    <probability>0.8128330131618828</probability>
  </value>
  <value>
    <concept>Grozny</concept>
    <probability>0.8118924755395808</probability>
  </value>
  <value>
    <concept>Houston</concept>
    <probability>0.8118924755395808</probability>
  </value>
  <value>
    <concept>montreal</concept>
    <probability>0.8109472115975687</probability>
  </value>
  <value>
    <concept>...</concept>
    <probability>...</probability>
  </value>
</node>
```

Abbildung 5.7.: Beispiel der Ausgabedatei

6. Evaluation

In diesem Kapitel wird die konzipierte Software evaluiert. Die Testdaten wurden auf Basis von existierenden Webformularen der Dienstleistungsanbieter erstellt. Da die Ausgabe mehreren tausend Werte beinhaltet, wurden nur hundert am höchsten bewerteten Domänenwerte betrachtet. Jeder Testfall wurde vier Mal durchgeführt. Hierbei wurde jeweils jede externe Datenquelle und abschließend die konsolidierten Ausgaben der drei externen Datenquellen verwendet.

6.1. Konfigurationsparameter

In diesem Abschnitt der Arbeit werden die für die Evaluation relevanten und verwendeten Konfigurationsparameter erläutert.

Um den Schwellwert nach Wu-Palmer zu ermitteln, den zwei Synsets in WordNet bei der Suche nach dem gemeinsamen Oberbegriff haben müssen, wurde eine Menge der Domäne ausgewählt. Zu jeder Domäne wurde die Ähnlichkeit zwischen zwei Werten aus dieser Domäne berechnet und das Minimum **minWP = 0,75** dieser Werte. Anschließend wurde die Ähnlichkeit zwischen einem Begriff in der Domäne und einem Begriff außerhalb der Domäne berechnet und das Maximum **maxWP = 0,67** der ermittelten Werte. Als Schwellwert wurde der Wert **0,7** ermittelt, der zwischen **minWP** und **maxWP** liegt.

Bei der Befragung von Wikipedia werden maximal **500** Ergebnisse bei jeder Abfrage ausgegeben. Dies ist eine Begrenzung von der Schnittstelle MediaWiki. Außerdem wurde eine Menge der Artikel betrachtet, um die Wahrscheinlichkeit zu bestimmen, mit der die Kategorie eines Artikels dem Oberbegriff zu diesem Artikel entspricht. **47%** der ausgewählten Kategorien waren Oberbegriffe zu dem Artikel. Aus diesem Grund wurde bei der Ermittlung jeder Kategorie die Bewertung von dieser Kategorie um das Faktor **0,47** multipliziert.

Die verwendeten Trainingsdaten befinden sich im Anhang dieser Bachelorarbeit.

6.2. Testfälle

Da die betrachteten Domänen eine große Anzahl der Werte einschließen, ist die Auswertung der Ausbeute in diesem Fall nicht möglich. Außerdem wird in der Regel eine erhebliche Anzahl von Konzepten ausgegeben (in der Regel mehrere Tausend), weswegen zur Evaluation jeweils 10, 50 und 100 Konzepte mit der höchsten Bewertung ausgewählt werden.

Tabelle 6.1.: Trefferquoten bei Testfall 1, Flugticketanbieter

Verwendete Datenquelle	Trefferquote (Top 10)	Trefferquote (Top 50)	Trefferquote (Top 100)
WordNet	0.8	0.76	0.73
Wikipedia	0.8	0.76	0.49
ResearchCyc	0.8	0.96	0.98
Alle Datenquellen	0.9	0.84	0.92

Innerhalb der Evaluation wurden zwei Testfälle erstellt. Für den ersten Testfall wurde eine Menge der Anbieter von Flugtickets betrachtet. Die Webformulare der Dienstanbieter wurden im Internet ausgesucht und die Werte der Attribute sowie Labels des Eingabefeldes, in das der Zielort einzugeben ist, in eine Eingabedatei gespeichert. Für den zweiten Testfall

Somit wurde die Möglichkeit der Verwendung von der erstellten Software sowohl in EASIER, als auch in weiteren Projekten für ähnliche Aufgaben untersucht. Zudem wurden für jeden Anwendungsfall Ergebnisse sowohl bei der Befragung einer einzelnen Datenquelle, als auch Ergebnisse bei der Befragung aller Datenquellen evaluiert.

6.2.1. Flugticketanbieter

Bei der Eingabe des Zielortes werden in Webformularen der Flugticketanbieter in der Regel Namen der Städte erwartet. In vielen untersuchten Formularen wurden ebenfalls auch Flughafennamen und -Kodierungen erlaubt. Aus diesem Grund werden alle Städte, Länder, Namen und Kodierungen der Flughäfen als richtige Domänenwerte akzeptiert. Zudem werden Spitznamen und andere Bezeichnungen der Eingabewerte als richtige Eingaben erkannt (z.B. „motor city“ als „Detroit“). Städte, die nicht mehr existieren (wie z.B. „Guonei City“), wurden als Fehlkonzepte betrachtet.

Nach der Klassifizierung der Attribute wurden folgende Begriffe richtig als Oberbegriffe der Eingabe erkannt: „City“, „Airport“ und „Town“. Als Beispieleingaben standen deutsche Städte und Flughäfen zur Verfügung, da diese vermutlich an den Standort des Benutzers angepasst wurden: „Karlsruhe/Baden-Baden“, „Frankfurt am Main (FRA)“ und „Heidelberg, Germany“.

Unter Ergebnissen sind Städte, Länder und Namen der Flughäfen. Keine Kodierung eines Flughafens war vorhanden. Die Verwendung von WordNet, ResearchCyc und Wikipedia hat die gleiche Genauigkeit unter den ersten zehn ausgegebenen Konzepten erwiesen. Bei der Verwendung von ResearchCyc wurde die höchste Genauigkeit erreicht. Somit hat sich ResearchCyc als die am meisten geeignete Datenquelle für diese Aufgabe gezeigt.

6.2.2. Bahnhaltstellen

Der nächste Testfall basiert auf den Daten, die in EASIER zur Erstellung einer Ontologie bereits verwendet wurden. Der gesuchte Wertebereich ist eine Menge der Haltestellen, zu denen der Benutzer eine Fahrkarte kaufen will. In der Testmenge waren Formulare vorhanden, in denen der Benutzer diesen Wert aus einer Auswahlliste wählen muss und somit mehrere Domänenwerte schon bekannt waren. Diese wurden gezielt nicht berücksichtigt, um die Möglichkeit der Verwendung der Software ohne Eingabe der Domänenwerte zu untersuchen. Dabei stehen lediglich die Oberbegriffe „place“, „station“ und „address“ zur Verfügung.

Die Suche mittels WordNet hat keine richtigen Domänenwerte geliefert. Dies kann damit begründet werden, dass WordNet eine Taxonomie der englischen Sprache ist und nur

Tabelle 6.2.: Trefferquoten bei Testfall 2, Bahnhofstestellen

Verwendete Datenquelle	Trefferquote (Top 10)	Trefferquote (Top 50)	Trefferquote (Top 100)
WordNet	0	0	0
Wikipedia	1.0	1.0	1.0
ResearchCyc	0	0	0
Alle Datenquellen	0	0	0

Tabelle 6.3.: Trefferquoten bei Testfall 3, Hotels

Verwendete Datenquelle	Trefferquote (Top 10)	Trefferquote (Top 50)	Trefferquote (Top 100)
WordNet	0,5	0,52	0,41
Wikipedia	0,6	0,3	0,26
ResearchCyc	1,0	1,0	0,84
Alle Datenquellen	0,6	0,64	0,62

oft verwendete Eigennamen wie Städte oder Marken der Autohersteller enthält. Die Suche mittels ResearchCyc hat ebenfalls keine Ergebnisse geliefert. Zudem hat ResearchCyc mehrere HTTP-Adressen ausgegeben. Folglich ist eine lexikalische Analyse der Ausgabe zu überlegen. Da bei der Suche in WordNet und ResearchCyc durch die Software in der Regel Konzepte mit wesentlich höheren Bewertungen ausgegeben werden, wurden bei der Konsolidierung der Ergebnisse aus allen drei Datenquellen unter ersten 100 am höchsten bewerteten Konzepte nur falsche Konzepte vorhanden.

Bei der Suche in Wikipedia waren alle ersten hundert Konzepte richtig. Somit beträgt die Genauigkeit des Ansatzes für die ersten Konzepte 100%. Dies lässt sich durch die Nachbearbeitung der Ergebnisse erklären: die gefundenen Haltestellen endeten auf das Wort „station“ (z.B. „London Victoria station“). Da das Wort „station“ unter den Oberbegriffen vorhanden war, wurden solchen Konzepte eine erheblich höhere Bewertung zugewiesen.

6.2.3. Hotelbuchung

Über Webformulare der entsprechenden Dienstleister können Benutzer ein Zimmer in einem Hotel finden und buchen. Für diesen Testfall wurde manuell eine Datei aus den Attributwerten der Formularelemente erstellt, in die das Ziel der Reise einzugeben ist. Es werden Städte, Adressen, Länder, Namen von Hotels und Örtlichkeiten als sinnvolle Eingabewerte erwartet.

Wie in der Tabelle Tabelle 6.3 zu sehen ist, haben sich alle drei Datenquellen für diese Aufgabe als geeignet erwiesen. Wie bei dem ersten Testfall, wurde die höchste Genauigkeit bei der Verwendung von ResearchCyc erreicht. Obwohl die Genauigkeit des Ansatzes bei der Verwendung von ResearchCyc und WordNet in diesem Fall höher ist, wurden nur von Wikipedia Namen von Hotels ausgegeben. Ähnlich wie im zweiten Testfall ist der Vorteil von Wikipedia die Ausbeute der Enzyklopädie.

6.3. Fazit

Die Genauigkeit der Ausgabe hängt an der ersten Stelle von der Anzahl der bekannten Domänenwerte und Oberbegriffe der Domänenwerte in der Eingabedatei ab. Bei der Suche nach oft verwendeten Wörtern wie Stadtnamen oder Substantiven haben sich alle drei

betrachteten externen Datenquellen als geeignet erwiesen. Bei der Suche nach sehr spezifischen Informationen wie Namen der Haltestellen ist die Wahrscheinlichkeit, dass diese in ResearchCyc oder in WordNet enthalten sind, relativ gering. Für solche Fälle bietet sich Wikipedia, da diese über umfangreicheren Inhalt verfügt und auch wenig bekannte Konzepte enthalten kann. Für die Verwendung in EASIER ist die Kombination der Verfahren zu empfehlen, um die Vorteile aller verwendeten Datenquellen zu kombinieren und die Ausbeute des ausgegebenen Wertebereichs zu verbessern.

7. Zusammenfassung und Ausblick

In EASIER werden aktive Ontologien aus den Webformularen von Diensteanbietern für spätere Verwendung in Software Agenten semi-automatisch erstellt. Dabei setzt die Erstellung von aktiven Ontologien für Software Agenten vom Ersteller die Kenntnisse der Anwendungsdomäne sowie allgemeinen Verstand voraus. Das erschwert die Automatisierung der Ontologierstellung und macht die Verwendung von externen Datenquellen zum unvermeidbaren Teil solcher Systeme. Aus diesem Grund wurde in der vorliegenden Bachelorarbeit eine Methode entwickelt, mit der aus Attributwerten eine Reihe von Abfragen an externe Datenquellen erstellt werden können, um einen Wertebereich eines AO-Knotens zu ermitteln. Als Datenquellen wurde die Taxonomie der englischen Sprache WordNet, die Ontologie des allgemeinen Verstands ResearchCyc und die Internet-Enzyklopädie Wikipedia verwendet.

7.1. Zusammenfassung

Zur Erstellung der Abfrage werden zuerst die Daten aus einer Eingabedatei ausgelesen und vorverarbeitet. Bei der Vorverarbeitung werden die zusammengesetzten Begriffe auf Teilbegriffe aufgeteilt und der Menge der Eingabewerte zusammen mit unterschiedlichen Kombinationen hinzugefügt. Die erstellte Menge der Begriffe wird anschließend in zwei Listen aufgeteilt: eine Liste der möglichen Domänenwerte und eine Liste der möglichen Oberbegriffe der Domänenwerte. Die Aufteilung in eine oder zwei Kategorien hängt dabei von der Art des Attributes ab. Jedem Begriff wird außerdem eine Bewertung zugewiesen, die der Wahrscheinlichkeit seiner Richtigkeit proportional ist. Diese Bewertung wird im Laufe der Ausführung erhöht, wenn der Grad an Sicherheit in Richtigkeit des Begriffes steigt. Im nächsten Schritt werden die zusammengesetzten Begriffe ermittelt, auf Teilbegriffe aufgeteilt und zusammen mit den möglichen Kombinationen den Listen hinzugefügt. Abschließend werden mittels lexikalischer Analyse aus den Listen Einträge, die keine Wortbindungen mit Substantiven, Substantive und Eigennamen sind, gelöscht.

Aus den ermittelten Begriffen werden Abfragen an die Datenquellen erstellt. Bei der Befragung von WordNet werden zuerst Paare der bekannten Domänenwerte mit starker semantischer Ähnlichkeit ausgewählt und den kleinsten gemeinsamen Oberbegriff sowie der größte Abstand zwischen den Begriffen aus dem Paar und dem ermittelten Oberbegriff berechnet. Danach werden direkte Unterbegriffe der bekannten Oberbegriffe sowie Unterbegriffe der ermittelten Oberbegriffe bis zur berechneten Tiefe ausgegeben.

Die Befragung von Wikipedia besteht aus Ermittlung der Artikel zu den bekannten Domänenwerten und Kategorien zu den gefundenen Artikeln. Zu den gefundenen Kategorien werden Artikel ausgegeben, deren Namen die gesuchte Domäne darstellt.

Um die Domäne mittels ResearchCyc zu finden, werden zuerst Gruppen (Collections) ermittelt, die den bekannten Oberbegriffen entsprechen. Anschließend werden zu jedem Paar der Begriffe aus der Liste der Domänenwerte Begriffe ausgegeben, die den gleich Gruppen gehören.

Die ermittelten Domänenwerte werden in eine Liste zusammengeführt und anschließend nachbearbeitet. Bei der Nachbearbeitung werden Begriffe, die auf bekannte Oberbegriffe enden, höher bewertet, da diese Unterbegriffe der bekannten Oberbegriffe sein können.

Bei der Evaluation hat der Ansatz für den Testfall mit bekannten Domänenwerten eine Trefferquote von 80% unter den ersten 50 am höchsten bewertete Ergebnisse erreicht. Bei der Durchführung des zweiten Testfalls ohne bekannte Domänenwerte und mit sehr abstrakten Begriffen hat die erstellte Software nur unter Verwendung von Wikipedia geschafft, die richtige Domäne zu ermitteln.

7.2. Ausblick: weitere Datenquellen

In dieser Bachelorarbeit wurden drei unterschiedliche Datenquellen betrachtet. Die entwickelte Lösung lässt sich ebenfalls auf weitere Datenquellen übertragen. Die Ontologie „Wikidata“ [wikb] verfügt über einen ähnlichen Aufbau wie Cyc und kann mittels der Sprache SPARQL abgefragt werden. Zur Ermittlung der Begriffe in deutscher Sprache kann die Taxonomie „Germanet“ [ger] verwendet werden. Zudem stellen Ontologien wie Yago [yag] oder Google Graph [goo] weitere potenziellen Datenquellen für die konzipierte Software da.

7.3. Ausblick: Konfigurationsparameter

Bei der Befragung der Datenquellen werden unterschiedliche Konfigurationsparameter wie der Schwellwert bei der semantischen Ähnlichkeit nach Wu-Palmer verwendet. Die Werte dieser Parameter wurden aus unterschiedlichen Stichproben ermittelt. Bei der Erhöhung der Anzahl von Stichproben, ist es möglich, genauere Parameter zu wählen und somit die Genauigkeit des Ansatzes zu verbessern.

Literaturverzeichnis

- [ACMFD⁺17] ARGUELLO CASTELEIRO, Mercedes ; MASEDA FERNANDEZ, Diego ; DEMETRIOU, George ; READ, Warren ; FERNANDEZ-PRIETO, MJ ; DES DIZ, Julio ; NENADIC, Goran ; KEANE, John ; STEVENS, Robert u. a.: A case study on sepsis using PubMed and Deep Learning for ontology learning. In: *Informatics for Health: Connected Citizen-Led Wellness and Population Health* 235 (2017), S. 516–520 (zitiert auf Seite 15).
- [ACS10] AASTRAND, Gunnar ; CELEBI, Remzi ; SAUERMANN, Leo: Using linked open data to bootstrap corporate knowledge management in the organik project. In: *Proceedings of the 6th International Conference on Semantic Systems* ACM, 2010, S. 18 (zitiert auf Seite 17).
- [ame] *American Airlines - Airline tickets und cheap flights at AA.com*. https://www.aa.com/homePage.do?locale=en_US (zitiert auf Seite 28).
- [BJB12] BOONE JR, Harry N. ; BOONE, Deborah A.: Analyzing likert data. In: *Journal of extension* 50 (2012), Nr. 2, S. n2 (zitiert auf Seite 16).
- [BK05] BENNACER, Nacéra ; KAROU, Lobna: A framework for retrieving conceptual knowledge from Web pages. In: *SWAP Citeseer*, 2005, S. 14–16 (zitiert auf den Seiten xi und 19).
- [Ble16] BLERSCH, Landhäußer: Easier: An Approach to Automatically Generate Active Ontologies for Intelligent Assistants. In: *The 20th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2016)*, 2016 (zitiert auf Seite 1).
- [BNJ03] BLEI, David M. ; NG, Andrew Y. ; JORDAN, Michael I.: Latent dirichlet allocation. In: *Journal of machine Learning research* 3 (2003), Nr. Jan, S. 993–1022 (zitiert auf Seite 16).
- [CDZS16] CHEN, Kai ; DONG, Xiang ; ZHU, Jiangang ; SHEN, Beijun: Building a Domain Knowledge Base from Wikipedia: a Semi-supervised Approach. In: *SEKE*, 2016 (zitiert auf Seite 21).
- [CKGB17] CHATTERJEE, Niladri ; KAUSHIK, Neha ; GUPTA, Deepali ; BHATIA, Ramneek: Ontology Merging: A Practical Perspective. In: *International Conference on Information and Communication Technology for Intelligent Systems* Springer, 2017, S. 136–145 (zitiert auf Seite 23).
- [Col17] *Kapitel HTML Form Elements*. In: COLLINS, Mark J.: *HTML Form Elements*. Berkeley, CA : Apress, 2017. – ISBN 978–1–4842–2463–2, 131–159 (zitiert auf Seite 11).
- [CV07] CILIBRASI, Rudi L. ; VITANYI, Paul M.: The google similarity distance. In: *IEEE Transactions on knowledge and data engineering* 19 (2007), Nr. 3 (zitiert auf Seite 22).

- [DWL15] DOU, Dejing ; WANG, Hao ; LIU, Haishan: Semantic data mining: A survey of ontology-based approaches. In: *Semantic Computing (ICSC), 2015 IEEE International Conference on IEEE*, 2015, S. 244–251 (zitiert auf Seite 5).
- [Erc06] ERCAN, Gönenç: *Automated text summarization and keyphrase extraction*, bilkent university, Diss., 2006 (zitiert auf den Seiten xi und 14).
- [FEMR15] FÄRBER, Michael ; ELL, Basil ; MENNE, Carsten ; RETTINGER, Achim: A Comparative Survey of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. In: *Semantic Web Journal, July* (2015) (zitiert auf Seite 13).
- [G⁺93] GRUBER, Thomas R. u. a.: A translation approach to portable ontology specifications. In: *Knowledge acquisition* 5 (1993), Nr. 2, S. 199–220 (zitiert auf Seite 5).
- [GBCC07] GUZZONI, Didier ; BAUR, Charles ; CHEYER, Adam ; CENTER, Artificial I.: Active, A Tool for Building Intelligent User Interfaces. In: *framework* 584 (2007), Nr. 048, S. 131 (zitiert auf Seite 5).
- [Gee17] GEETHA, TV: Unsupervised Domain Ontology Learning from Text. In: *Mining Intelligence and Knowledge Exploration: 4th International Conference, MIKE 2016, Mexico City, Mexico, November 13-19, 2016, Revised Selected Papers* Bd. 10089 Springer, 2017, S. 132 (zitiert auf den Seiten 17, 29 und 39).
- [ger] *Germanet*. <http://www.sfs.uni-tuebingen.de/GermaNet/> (zitiert auf Seite 52).
- [goo] *GoogleGraph*. <https://www.google.com/intl/en-419/insidesearch/features/search/knowledge.html> (zitiert auf Seite 52).
- [Guz08] GUZZONI, Didier: Active: a unified platform for building intelligent applications. (2008) (zitiert auf den Seiten xi, xiii, 5, 6, 7, 8 und 9).
- [HBS06] HEPP, Martin ; BACHLECHNER, Daniel ; SIORPAES, Katharina: OntoWiki: community-driven ontology engineering and ontology usage based on Wikis. In: *Proceedings of the 2006 international symposium on Wikis* ACM, 2006, S. 143–144 (zitiert auf Seite 21).
- [HEBR11] HAZMAN, Maryam ; EL-BELTAGY, Samhaa R. ; RAFEA, Ahmed: A survey of ontology learning approaches. In: *database* 7 (2011), S. 6 (zitiert auf den Seiten 5, 15 und 17).
- [jdo] *jDom*. <http://www.jdom.org/> (zitiert auf Seite 44).
- [Jon15] JONES, M T.: *Artificial Intelligence: A Systems Approach: A Systems Approach*. Jones & Bartlett Learning, 2015 (zitiert auf Seite 5).
- [LLRS97] LANDAUER, Thomas K. ; LAHAM, Darrell ; REHDER, Bob ; SCHREINER, Missy E.: How well can passage meaning be derived without using word order? A comparison of Latent Semantic Analysis and humans. In: *Proceedings of the 19th annual meeting of the Cognitive Science Society*, 1997, S. 412–417 (zitiert auf Seite 16).
- [LP07] LEAKE, David ; POWELL, Jay: Mining large-scale knowledge sources for case adaptation knowledge. In: *International Conference on Case-Based Reasoning* Springer, 2007, S. 209–223 (zitiert auf Seite 14).
- [MBF⁺90] MILLER, George A. ; BECKWITH, Richard ; FELLBAUM, Christiane ; GROSS, Derek ; MILLER, Katherine J.: Introduction to WordNet: An on-line lexical

- database. In: *International journal of lexicography* 3 (1990), Nr. 4, S. 235–244 (zitiert auf Seite 13).
- [MCWD06] MATUSZEK, Cynthia ; CABRAL, John ; WITBROCK, Michael J. ; DEOLIVEIRA, John: An Introduction to the Syntax and Content of Cyc. In: *AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, 2006, S. 44–49 (zitiert auf Seite 14).
- [MFC14] MCCRAE, John ; FELLBAUM, Christiane ; CIMIANO, Philipp: Publishing and Linking WordNet using lemon and RDF. In: *Proceedings of the 3rd Workshop on Linked Data in Linguistics*, 2014 (zitiert auf Seite 13).
- [ML08] MEDELYAN, Olena ; LEGG, Catherine: Integrating Cyc and Wikipedia: Folksonomy meets rigorously defined common-sense. (2008) (zitiert auf Seite 23).
- [NL07] NAVIGLI, Roberto ; LAPATA, Mirella: Graph Connectivity Measures for Unsupervised Word Sense Disambiguation. In: *IJCAI*, 2007, S. 1683–1688 (zitiert auf Seite 17).
- [OVQH13] OLTRAMARI, Alessandro ; VOSSEN, Piek ; QIN, Lu ; HOVY, Eduard: *New trends of research in ontologies and lexical resources: Ideas, projects, systems*. Springer Science & Business Media, 2013 (zitiert auf Seite 5).
- [PPM04] PEDERSEN, Ted ; PATWARDHAN, Siddharth ; MICHELIZZI, Jason: WordNet::Similarity: measuring the relatedness of concepts. In: *Demonstration papers at HLT-NAACL 2004* Association for Computational Linguistics, 2004, S. 38–41 (zitiert auf Seite 33).
- [pub] *PubMed - NCBI*. <https://www.ncbi.nlm.nih.gov/pubmed/> (zitiert auf Seite 16).
- [res] *ResearchCyc*. <http://www.cyc.com/platform/researchcyc/> (zitiert auf Seite 14).
- [SFJ08] SYED, Zareen S. ; FININ, Tim ; JOSHI, Anupam: Wikipedia as an Ontology for Describing Documents. In: *ICWSM*, 2008 (zitiert auf Seite 21).
- [shi] *Shima H., WS4J WordNet Similarity for Java*. <https://github.com/Sciss/ws4j> (zitiert auf Seite 45).
- [SKW08] SUCHANEK, Fabian M. ; KASNECI, Gjergji ; WEIKUM, Gerhard: Yago: A large ontology from wikipedia and wordnet. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 6 (2008), Nr. 3, S. 203–217 (zitiert auf Seite 21).
- [SS89] SALTON, Gerard ; SMITH, Maria: On the application of syntactic methodologies in automatic text analysis. In: *ACM SIGIR Forum* Bd. 23 ACM, 1989, S. 137–150 (zitiert auf Seite 12).
- [staa] *Stack Overflow*. <https://stackoverflow.com/> (zitiert auf Seite 21).
- [stab] *Stanford CoreNLP Demo*. <http://nlp.stanford.edu:8080/corenlp/> (zitiert auf Seite 44).
- [stac] *Stanford Named Entity Recognition*. <https://nlp.stanford.edu/software/CRF-NER.shtml> (zitiert auf Seite 44).
- [stad] *Stanford Part of Speech Tagger*. <https://nlp.stanford.edu/software/tagger.shtml> (zitiert auf Seite 44).

- [str] *StringUtils* von Apache Commons. <https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/StringUtils.html> (zitiert auf Seite 45).
- [TMKW07] TAYLOR, Matthew E. ; MATUSZEK, Cynthia ; KLIMT, Bryan ; WITBROCK, Michael J.: Autonomous Classification of Knowledge into an Ontology. In: *FLAIRS Conference*, 2007, S. 140–145 (zitiert auf den Seiten 13 und 14).
- [VK14] VRANDEČIĆ, Denny ; KRÖTZSCH, Markus: Wikidata: a free collaborative knowledgebase. In: *Communications of the ACM* 57 (2014), Nr. 10, S. 78–85 (zitiert auf Seite 12).
- [wika] *Help:Category*. <https://en.wikipedia.org/wiki/Help:Category> (zitiert auf Seite 12).
- [wikb] *Wikidata*. <https://www.wikidata.org/> (zitiert auf Seite 52).
- [wikc] *Wikipedia*. <https://www.wikipedia.org> (zitiert auf Seite 12).
- [WP94] WU, Zhibiao ; PALMER, Martha: Verbs semantics and lexical selection. In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* Association for Computational Linguistics, 1994, S. 133–138 (zitiert auf Seite 34).
- [Wu16] WU, Zhaohui: *SEMANTIC MODELING FOR NATURAL LANGUAGE USING*, The Pennsylvania State University, Diss., 2016 (zitiert auf Seite 16).
- [yag] *Yago*. <https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/> (zitiert auf Seite 52).

Anhang

A. Testfall 1. Flugticketanbieter.

Die Webrmulare unter den folgenden Adressen wurden verwendet, um den Testfall zu erstellen:

1. <https://www.justdial.com/travel/flight-booking>
2. <https://www.akbartravels.com/airlines/domestic-flight>
3. <https://www.goindigo.in/>
4. <https://www.opodo.com/>
5. <https://www.united.com/ual/en/gb/>
6. <http://www.lufthansa.com/us/en/Homepage>
7. <https://www.ryanair.com/gb/en/>
8. <https://www.expedia.com/?&rfr=Header.POSRedirect.www.expedia.de>
9. <https://www.virginaustralia.com/au/en/bookings/flights/make-a-booking/>
10. <https://www.alaskaair.com/planbook>
11. <https://www.kayak.com/>
12. <https://www.edreams.com/?mktportal=google&title=mcampaign.cheapflights>
13. <https://www.onetravel.com/travel/specials/flights1.asp?>
14. <https://www.goibibo.com/flights/>
15. <https://www.skyscanner.net/>
16. <https://www.cheapair.com/flights/>
17. <https://www.jetairways.com/en/in/planyourtravel/book-online.aspx>
18. <http://www.air.irctc.co.in/IndianRailways/>
19. <https://www.ryanair.com/de/de/>

Eingabedatei für den Testfall:

```

<node>
  <nodeName>flightBookingServices</nodeName>
  <placeholders>
    <placeholder>From</placeholder>
    <placeholder>From</placeholder>
    <placeholder>from</placeholder>
    <placeholder>Country, city or airport</placeholder>
    <placeholder>Departure airport</placeholder>
    <placeholder>City or airport</placeholder>
    <placeholder>From</placeholder>
    <placeholder>From where?</placeholder>
    <placeholder>From: City or Airport</placeholder>
    <placeholder>A city, town, or airport</placeholder>
    <placeholder>from1</placeholder>
    <placeholder>From</placeholder>
    <placeholder>From</placeholder>
    <placeholder>Type Departure City</placeholder>
    <placeholder>Enter departure city</placeholder>
  </placeholders>
  <names>
    <name>originName1</name>
    <name>Origin</name>
    <name>origin</name>
    <name>ShoppingRequestModel.DepartureCity1</name>
    <name>departure</name>
    <name>departureAirportName</name>
    <name>gcw-origin</name>
    <name>originSurrogate</name>
    <name>departureReturnOne</name>
    <name>ctl00$MainBody$ctl00$autoOrigin$txtCountry</name>
    <name>AirlineData [0].From</name>
  </names>
  <ids>
    <id>flightmanagerFlightsFormOrigin</id>
    <id>Origin</id>
    <id>origin</id>
    <id>fromCity</id>
    <id>Trips_0__Origin</id>
    <id>airport-selector-from</id>
    <id>js-origin-input</id>
    <id>flights-originSurrogate</id>
    <id>departureId</id>
    <id>W3l9-origin</id>
    <id>gosuggest_inputSrc</id>
    <id>from1</id>
    <id>From</id>
    <id>departure</id>
    <id>airport</id>
  </ids>
  <values>
    <value>Karlsruhe/Baden-Baden</value>
    <value>Frankfurt am Main (FRA)</value>
    <value>Heidelberg, Germany</value>
    <value>Type Departure Location Here</value>
  </values>
  <labels>
    <label>From</label>
  </labels>
</node>

```


Tabelle A.1.: Testfall 1, Flugticketanbieter: Treffer 1 - 25 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
motor city	Karlsruhe	kandahar	new york city
windy city	Heidelberg	japand	kansas city
old country	Mannheim	japand, london	atlantic city
cebu city	Frankfurt	toledo	jersey city
vatican city	Germany	san antonio	vatican city
new york city	Freiburg im Breisgau	jackson	motor city
kansas city	Baden	the american city of jackson	windy city
quezon city	Karlsruhe/Baden-Baden Airport	american city of jackson	old country
queen city	Barbarossa city	saalem	lexington
atlantic city	city	the american city of saalem	ho chi minh city
ho chi minh city	Frankfurt Airport	american city of saalem	charleston
sioux city	Vatican City	annapolis	cebu city
naha city	Pforzheim	annapolis, md	quezon city
jersey city	Guonei City	annapolis, maryland	saalem
city	Ho Chi Minh City	concord	queen city
state of the vatican city	Zamboanga City	the american city of concord	sioux city
central city	Zanzibar City	american city of concord	naha city
inner city	Sejong City	newark	city
the city	Greifswald	the american city of newark	kandahar
hegemon	Rostock	american city of newark	san antonio
capacity	Friedrichshafen Airport	charleston	toledo
anchorage	Global city	the american city of charleston	portland
natal	Frankfurt Egelsbach Airport	american city of charleston	chattanooga
country of origin	Reichelsheim Airport	portland	arlington
commonwealth	Bad Wimpfen	the american city of portland	newark
country		land	

Tabelle A.2.: Testfall 1, Flugticketanbieter: Treffer 26 - 50 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
developing country	United city	american city of portland	anchorage
foreign country	Bitburg Airport	lexington	state of the vatican city
columbus	Capital city	the american city of lexington	inner city
airport	Charter city	american city of lexington	mannheim
alexandria	airport	the american city of arlington	columbus
portsmouth	Bremen Airport	arlington	central city
vancouver	Leipzig-Altendorf Airport	american city of arlington	the city
oxford	Route capacity	chattanooga	Frankfurt
lexington	Walking City	chattanooga, tn	norfolk
brunswick	Hannover Airport	chattanooga, tennessee	san francisco
abilene	Mannheim City Airport	salt lake city	los angeles
wilmington	Stuttgart Airport	the american city of salt lake city	san diego
aberdeen	Cebu City	american city of salt lake city	burlington
decatour	Quezon City	oklahoma city	vancouver
santa cruz	Al Noor City	the american city of oklahoma city	Germany
greenville	Broadacre City	american city of oklahoma city	wilmington
florence	Jazan Economic City	new york city	santa cruz
gloucester	King Abdullah Economic City	the american city of new york city	rochester
rochester	Nanhui New City	american city of new york city	hegemon
panama city	Prince Abdulaziz Bin Mousaed Economic City	jefferson city	new york
new york	Sudair Industrial City	the american city of jefferson city	frankfort
madras	Bindlacher Berg Airport	american city of jefferson city	persia
campeche	Braunschweig Airport	carson city	capacity
mysore	Herzogenaurach Airport	the american city of carson city	egypt
chihuahua	Allendorf Airport	american city of carson city	concord

Tabelle A.3.: Testfall 1, Flugticketanbieter: Treffer 51 - 75 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
motown	Frankfurt?-Hahn Airport	jersey city	munich
cape town	Kassel Airport	the american city of jersey city	riverside
canton	Sister city	american city of jersey city	san jose
khabarovsk	Secondary city	atlantic city	springfield
youngstown	Megacity	the american city of atlantic city	augusta
morgantown	Coast Guard City	american city of atlantic city	monaco
allentown	Primate city	kansas city	luxembourg
johnson city	Sustainable city	the american city of kansas city	federal republic of germany
traverse city	International city	american city of kansas city	jackson
mason city	Independent city	Reston, ??????????	rostock
dodge city	Lost city	Reston, ????	campeche
rapid city	Smart city	Reston, Virg?inia	chihuahua
silver city	History of the city	reston, commonwealth of virginia	saint louis
morgan city	Caravan city	reston, the state of virginia	st. louis
Frankfurt	Conscious city	reston, state of virginia	oakland
cebu	Gigacity	reston, virginie	bridgeport
forbidden city	Human Rights City	reston, va	baltimore
tripoli	Inner city	reston	hamilton
santiago	Maya city	reston, virginia	bahrain
riverside	Open city	Wise, ??????????	bakersfield
charleston	Winter City	Wise, ????	omaha
leicester	Autonomous city	Wise, Virg?inia	dayton
atonicity	Free imperial city	wise, commonwealth of virginia	new orleans
reading	Baden-Baden	wise, the state of virginia	cambridge
independence	Glass Museum of Hsinchu City	wise, state of virginia	pittsburgh

Tabelle A.4.: Testfall 1, Flugticketanbieter: Treffer 76 - 100 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
motown	Frankfurt?-Hahn Airport	jersey city	munich
cape town	Kassel Airport	the american city of jersey city	riverside
canton	Sister city	american city of jersey city	san jose
khabarovsk	Secondary city	atlantic city	springfield
youngstown	Megacity	the american city of atlantic city	augusta
morgantown	Coast Guard City	american city of atlantic city	monaco
allentown	Primate city	kansas city	luxembourg
johnson city	Sustainable city	the american city of kansas city	federal republic of germany
traverse city	International city	american city of kansas city	jackson
mason city	Independent city	Reston, ??????????	rostock
dodge city	Lost city	Reston, ????	campeche
rapid city	Smart city	Reston, Virg?inia	chihuahua
silver city	History of the city	reston, commonwealth of virginia	saint louis
morgan city	Caravan city	reston, the state of virginia	st. louis
Frankfurt	Conscious city	reston, state of virginia	oakland
cebu	Gigacity	reston, virginie	bridgeport
forbidden city	Human Rights City	reston, va	baltimore
tripoli	Inner city	reston	hamilton
santiago	Maya city	reston, virginia	bahrain
riverside	Open city	Wise, ??????????	bakersfield
charleston	Winter City	Wise, ????	omaha
leicester	Autonomous city	Wise, Virg?inia	dayton
atonicity	Free imperial city	wise, commonwealth of virginia	new orleans
reading	Baden-Baden	wise, the state of virginia	cambridge
independence	Glass Museum of Hsinchu City	wise, state of virginia	pittsburgh

B. Testfall 2, Haltestellen

Eingabedatei für den Testfall:

```
<node>
  <nodeName>flightBookingServices</nodeName>
  <placeholders>
    <placeholder>From</placeholder>
    <placeholder>From</placeholder>
    <placeholder>from</placeholder>
    <placeholder>Country, city or airport</placeholder>
    <placeholder>Departure airport</placeholder>
    <placeholder>City or airport</placeholder>
    <placeholder>From</placeholder>
    <placeholder>From where?</placeholder>
    <placeholder>From: City or Airport</placeholder>
    <placeholder>A city, town, or airport</placeholder>
```

```

        <placeholder>from1</placeholder>
        <placeholder>From</placeholder>
        <placeholder>From</placeholder>
        <placeholder>Type Departure City</placeholder>
        <placeholder>Enter departure city</placeholder>
</placeholders>
<names>
    <name>originName1</name>
    <name>Origin</name>
    <name>origin</name>
    <name>ShoppingRequestModel.DepartureCity1</name>
    <name>departure</name>
    <name>departureAirportName</name>
    <name>gcw-origin</name>
    <name>originSurrogate</name>
    <name>departureReturnOne</name>
    <name>ctl00$MainBody$ctl00$autoOrigin$txtCountry</name>
    <name>AirlineData [0].From</name>
</names>
<ids>
    <id>flightmanagerFlightsFormOrigin</id>
    <id>Origin</id>
    <id>origin</id>
    <id>fromCity</id>
    <id>Trips\_0\_--Origin</id>
    <id>airport-selector-from</id>
    <id>js-origin-input</id>
    <id>flights-originSurrogate</id>
    <id>departureId</id>
    <id>W3l9-origin</id>
    <id>gosuggest_inputSrc</id>
    <id>from1</id>
    <id>From</id>
    <id>departure</id>
    <id>airport</id>
</ids>
<values>
    <value>Karlsruhe/Baden-Baden</value>
    <value>Frankfurt am Main (FRA)</value>
    <value>Heidelberg , Germany</value>
    <value>Type Departure Location Here</value>
</values>
<labels>
    <label>From</label>
</labels>
</node>

```

Tabelle B.5.: Testfall 2, Bahnhaltestellen: Treffer 1 - 25 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
station	South Station	fox studios	station
point of departure	North Station	fox	point of departure
polling station	Blackfriars station	fox movies	polling station
place	Back Bay station	fallout shelter	place
devastation	North Union Station	an office named "prime minister's office"	devastation
Start	London Victoria station	an office space named "prime minister's office"	Start
social station	Cannon Street station	an office suite named "prime minister's office"	social station
manifestation	Embankment tube station	an firmensitz named "prime minister's office"	manifestation
Departure	Temple tube station	an buero named "prime minister's office"	Departure
outstation	3rd Street station	an amt named "prime minister's office"	outstation
incrustation	5th Street station	prime minister's office	incrustation
encrustation	7th Street station	an office named "district coordination offices"	encrustation
house	10th Street Station	an office space named "district coordination offices"	house
way station	14th Street station	an office suite named "district coordination offices"	way station
protestation	23rd Street Station	an firmensitz named "district coordination offices"	protestation
birthplace	24th Street Station	an buero named "district coordination offices"	birthplace
address	28th Street station	an amt named "district coordination offices"	address
head	Gloucester Road tube station	district coordination offices	head
deanery	Liverpool Street station	a flat named "malysian apartment of yazid sufaat"	deanery
emirate	Mansion House tube station	a etagenwohnung named "malysian apartment of yazid sufaat"	khanate
khanate	St. James's Park tube station	a einzelzimmer named "malysian apartment of yazid sufaat"	emirate
lead	Sloane Square tube station	a appartement named "malysian apartment of yazid sufaat"	lead
judicature	South Kensington tube station	a apartment named "malysian apartment of yazid sufaat"	judicature
line	Westminster tube station	malysian apartment of yazid sufaat	line
seigniorry	Marylebone station	mohamed atta's hamburg apartment	seigniorry

Tabelle B.6.: Testfall 2, Bahnhofstestellen: Treffer 26 - 50 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
seigneury	Framingham Railroad Station	the production area	seigneury
regency	New London Union Station	production area	regency
womanhood	Providence station	austin recreation center dance studio	womanhood
incumbency	Congress Street Fire Station	the austin recreation center dance studio	incumbency
episcopate	Bayswater tube station	austin recreation center gymnasium	mastership
mastership	High Street Kensington tube station	the austin recreation center gymnasium	episcopate
receivership	Moorgate station	feigenbaum cr	receivership
contestation	Notting Hill Gate tube station	the feigenbaum conference room	contestation
presidency	Tower Hill tube station	the fish bowl	presidency
vice-presidency	Baker Street tube station	the think tank	vice-presidency
chair	Euston Square tube station	think tank	chair
residency	Great Portland Street tube station	fish bowl	residency
sainthood	King's Cross St. Pancras tube station	feigenbaum conference room	sainthood
fatherhood	Fenchurch Street railway station	sleep pods	fatherhood
generalship	London Bridge station	hub	generalship
plum	Aldgate East tube station	bridge	plum
point	Earl's Court tube station	hydroponics	point
baronetage	City Thameslink railway station	systems	baronetage
caliphate	Holyoke station	vehicles	caliphate
trusteeship	Route 128 station	engineering	trusteeship
admiralty	Park Street Railroad Station	observation deck	admiralty
academicianship	White River Junction station	cargo	academicianship
accountantship	Alexandria Union Station	maintenance deck	accountantship
ambassadorship	Chicago Union Station	supply area 31a	ambassadorship
apostleship	Dallas Union Station	lair	apostleship

In den folgenden Tabellen sind mehrere HTTP-Adressen enthalten. Zur Verbesserung der Lesbarkeit sind sie durch das Wort „HTTP“ ersetzt, da die genaue nicht relevant ist.

Tabelle B.7.: Testfall 2, Bahnhaltestellen: Treffer 51 - 75 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
apostleship	Dallas Union Station	lair	apostleship
apprenticeship	Denver Union Station	reactor room	apprenticeship
associateship	Joliet Union Station	HTTP	associateship
attorneyship	Kansas City Union Station	HTTP	attorneyship
bailiffship	Miami Airport Station	HTTP	bailiffship
bishopry	New Rochelle station	HTTP	bishopry
cadetship	Portland Union Station	HTTP	cadetship
captainship	Washington Union Station	HTTP	captainship
captaincy	Hartford Union Station	HTTP	captaincy
cardinalship	Lenox Railroad Station	HTTP	cardinalship
chairmanship	Congress Street Fire Station	station	chairmanship
chancellorship	Aberdeen station	HTTP	chancellorship
chaplainship	Adams Station	HTTP	chaplainship
chaplaincy	Ridgefield Park station	HTTP	chaplaincy
chieftainship	District 13 Police Station	HTTP	chieftainship
chieftaincy	Harvard Avenue Fire Station	HTTP	chieftaincy
clerkship	Egleston Substation	HTTP	clerkship
commandership	Roslindale Substation	HTTP	commandership
commandery	Aldgate tube station	HTTP	commandery
comptrollership	Barbican tube station	HTTP	comptrollership
consulship	Farringdon station	HTTP	consulship
controllership	Mark Lane tube station	HTTP	controllership
councilorship	Charing Cross railway station	HTTP	councilorship
councillorship	Euston railway station	HTTP	councillorship
counselorship	London King's Cross railway station	HTTP	counselorship
counsellorship	London Paddington station	HTTP	counsellorship

Tabelle B.8.: Testfall 2, Bahnhaltestellen: Treffer 76 - 100 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
curacy	London Waterloo station	HTTP	curacy
curatorship	St Pancras railway station	HTTP	curatorship
custodianship	Acton Town tube station	HTTP	custodianship
deanship	Barons Court tube station	HTTP	deanship
directorship	Chiswick Park tube station	HTTP	directorship
discipleship	Ealing Broadway station	HTTP	discipleship
editorship	Ealing Common tube station	HTTP	editorship
eldership	Fulham Broadway tube station	HTTP	eldership
foremanship	Mile End tube station	HTTP	foremanship
generalcy	Parsons Green tube station	HTTP	generalcy
governorship	Putney Bridge tube station	HTTP	governorship
headship	South Acton railway station	HTTP	headship
hot seat	St. Mary's (Whitechapel Road) tube station	HTTP	hot seat
inspectorship	Stepney Green tube station	HTTP	inspectorship
instructorship	West Brompton station	HTTP	instructorship
internship	West Kensington tube station	HTTP	internship
judgeship	Whitechapel station	HTTP	judgeship
lectureship	Elephant & Castle railway station	HTTP	lectureship
legation	Angel tube station	HTTP	legation
legateship	Bond Street tube station	HTTP	legateship
legislatorship	Borough tube station	HTTP	legislatorship
librarianship	Chancery Lane tube station	HTTP	librarianship
lieutenancy	Charing Cross tube station	HTTP	lieutenancy
magistrature	Covent Garden tube station	HTTP	magistrature
magistracy	Elephant & Castle tube station	HTTP	magistracy

C. Testfall 3, Hotels

Die Webformulare unter den folgenden Adressen wurden verwendet, um den ersten Testfall zu erstellen:

1. https://www.booking.com/index.html?label=gen173nr-1DCAEoggJCAlhYSDNiBW5vcvVmaDuIAQGyAQe4AQjIAQzYAQPoAQGSAGF5qAID;sid=b97ee8c5569b1166a9baace4a11f3144;sb_price_type=total&
2. <https://www.makemytrip.com/hotels/>
3. <https://www.trivago.co.uk/?>
4. <https://www.goibibo.com/hotels/>

5. <https://www.yatra.com/hotels>

Eingabedatei für den Testfall:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<node>
  <nodeName>hotels</nodeName>
  <placeholders>
    <placeholder>More places than you could ever go (but
      you can try!</placeholder>
    <placeholder>Enter City name, Area name or Hotel name</
      placeholder>
    <placeholder>Select City, Location or Hotel Name (
      Worldwide)</placeholder>
  </placeholders>
  <names>
    <name>ss</name>
    <name>BE_hotel_destination_city</name>
  </names>
  <ids>
    <id>ss</id>
    <id>hp-widget__sDest</id>
    <id>gosuggest_inputL</id>
    <id>horus-shadowtext</id>
    <id>BE_hotel_destination_city</id>
  </ids>
  <values>
    <value>Goa, India</value>
    <value>e.g. Paris</value>
  </values>
  <labels>
    <label>Destination, property name or address:</label>
    <label>City, Hotel or Area</label>
  </labels>
</node>
```

Tabelle C.9.: Testfall 3, Hotels: Treffer 1 - 25 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
columbus	Paris	Paris	columbus
bosnia	India	toledo	bosnia
point	Goa	charleston	point
position	Paris (mythology)	kandahar	position
russia	Luxembourg City	japand	center
center	Daman and Diu	japand, london	luxemburg
america	Noida Film City	san antonio	koweit
djibouti	Djibouti City	jackson	djibouti
georgia	Kowloon Walled City	the american city of jack- son	georgia
place	Aeneas	american city of jackson	place
trusteeship	Anchises	salem	trusteeship
top	Clonius	the american city of salem	top
head	Helenus	american city of salem	head
hegemon	Ilioneus	annapolis	hegemon
usa	Polites (prince of Troy)	annapolis, md	usa
old country	Ucalegon	annapolis, maryland	old country
monaco	Brazil	concord	monaco
principality of mo- naco	User:Southern Person/- sandbox	the american city of con- cord	principality of mo- naco
egypt	Capital city	american city of concord	egypt
persia	Troy: Fall of a City	newark	bahrain
fallow	Coast Guard City	the american city of ne- wark	bahrein
Paris	Primate city	american city of newark	fallow
anchorage	Secondary city	the american city of charle- ston	Paris
natal	Sister city	american city of charleston	anchorage
end	Sustainable city	portland	natal

Tabelle C.10.: Testfall 3, Hotels: Treffer 26 - 50 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
luxembourg	Cassandra	the american city of portland	end
luxemburg	Pullman Paris Montparnasse Hotel	american city of portland	russia
koweit	Directorate of Advertising and Visual Publicity	lexington	america
kuwait	International Exposition of Electricity	the american city of lexington	luxembourg
singapore	Indian states ranking by households having electricity	american city of lexington	kuwait
san marino	EuropaCity	the american city of arlington	singapore
pinnacle	Aberdeen micropolitan area	arlington	san marino
state of the vatican city central city	Air India	american city of arlington	pinnacle
inner city seat the city play	Russia Antiphates Acamas Acamas (son of Antenor) Agenor, son of Antenor	chattanooga chattanooga, tn chattanooga, tennessee salt lake city the american city of salt lake city	state of the vatican city china persia central city inner city
gold coast	Antimachus in Greek mythology	american city of salt lake city	seat
peak ireland	Gorgythion 25 km	oklahoma city the american city of oklahoma city	the city India
iceland	27 km	american city of oklahoma city	ireland
balkans madagascar	South Africa Pakistan	new york city the american city of new york city	iceland madagascar
china	Arunachal Pradesh	american city of new york city	japan
bahrain bahrein	Singapore Paris Commune	jefferson city the american city of jefferson city	qatar play
japan	Nigeria	american city of jefferson city	gold coast
kingdom victoria	Actor (mythology) Ajax the Lesser	carson city the american city of carson city	katar lexington

Tabelle C.11.: Testfall 3, Hotels: Treffer 51 - 75 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
control	Automedon	american city of carson city	peak
prefecture	Capys	jersey city	balkans
split	Clytius	the american city of jersey city	israel
qatar	Dares Phrygius	american city of jersey city	charleston
katar	Euryalus	atlantic city	kingdom
emirate	Pandarus	the american city of atlantic city	victoria
khanate	Panthous	american city of atlantic city	control
incumbency	Priam	kansas city	prefecture
front	Ripheus	the american city of kansas city	new york
episcopate	Theano	american city of kansas city	split
seignior	Template:Trojan race	hamilton	emirate
seignury	Antenor (mythology)	distrito federal	khanate
development	Antenorides	cuba	incumbency
meridian	Antiphus	cub	front
end point	Archelochus	cuban	vancouver
israel	Aretus	cubaner	episcopate
fatherhood	Caletor	cuba (cub)	seignior
new york	Eurydamas	cuba (cu)	seignury
hub	Glaucus (soldier)	panama	development
admiralty	Iamenus	pan	meridian
madras	Kebriones	panamanian	santa cruz
lead	Lycaon (Troy)	more panamanian	end point
receivership	Melanippus	republic of panama	iran
campeche	Pedasmus	panama (pan)	salem
mysore	Philoctetes	panama (pa)	fatherhood

Tabelle C.12.: Testfall 2, Bahnhalttestellen: Treffer 76 - 100 (Fehlkonzepte rot markiert)

WordNet	Wikipedia	ResearchCyc	Alle Datenquellen
chihuahua	Polydamas (mythology)	egypt	wilmington
alexandria	Schedius	egy	rochester
portsmouth	Thestor (mythology)	egyptian	brazil
vancouver	Thoon (mythology)	more egyptian	portsmouth
center stage	Sri Lanka	the arab republic of egypt	united arab republic
centre stage	Dominica	egypt (egy)	arab republic of egypt
leadership	Mizoram	egypt (eg)	state of bahrain
jurisdiction	Annexation of Goa	the united arab republic	islamic republic of iran
extinction	Creusa (wife of Aeneas)	united arab republic	alexandria
presidency	Dardanus	arab republic of egypt	south africa
vice-presidency	Alcimus (mythology)	iran	pakistan
land	Anchialus	irn	olympia
motor city	Asius (mythology)	irani	hub
windy city	Deiphobus	more irani	nigeria
summit	Phaenops	persian	admiralty
city centre	Abas (mythology)	more persian	madras
city center	Cycnus (king of Kolonai)	iranian	sudan
storm center	Eurytion	more iranian	mongolia
storm centre	List of Trojan War characters	the islamic republic of iran	lead
financial center	Troilus	the islamic republic	receivership
medical center	Trojan War	iran (irn)	campeche
midfield	Aaby	iran (ir)	mysore
midstream	Aachen (disambiguation)	the islamic republic of iran (irn)	chihuahua
city of london	?er	the islamic republic of iran (ir)	tibet
India	Aalst	persia	xizang

D. Semantische Ähnlichkeit nach Wu-Palmer

Tabelle D.13.: Semantische Ähnlichkeit nach Wu-Palmer

Oberbegriff in natürlicher Sprache	Richtiger Wert v1	Richtiger Wert v2	Falscher Wert f	kleinster Oberbegriff in WordNet (v1 und v2)	Ähnlichkeit (v1, v2)	kleinster Oberbegriff in WordNet (v1 und f)	Ähnlichkeit (v1, f)
Dog	terrier	chihuahua	Dog	0,88	coffee	0,58	organism
Beverage	coffee	wine	Beverage	0,86	stone	0,67	substance
City	berlin	sydney	City	0,91	book	0,52	artifact
Occupation	physician	teacher	Professional	0,8	donkey	0,47	organism
Country	Germany	Australia	Country	0,85	school	0,4	object
Furniture	Table	Wardrobe	furniture	0,9	cat	0,67	instrumentality
Food	cake	Burger	dish	0,8	tree	0,53	whole
Animal	Monkey	Dog	placental	0,83	Armchair	0,43	person
Vehicle	Plane	Car	vehicle	0,75	Carpet	0,64	instrumentality

E. Kategorien der Artikel in Wikipedia

Tabelle E.14.: Kategorien der Artikel in Wikipedia

Oberbegriff	Artikelname	Kategorien	Anzahl der Kategorien	Anzahl der Relevante Kategorien
Dog breed	Terrier	Terriers, Dog types, Hunting dogs	3	3
Beverage	Coffee	Coffea drinks, Crops, Herbal and fungal stimulants, Non-alcoholic drinks, Hot drinks, Turkish words and phrases	6	2
City	Moscow	Golden Ring of Russia, 1147 establishments in Russia, Capitals in Asia, Capitals in Europe, Moscow, Moscow Governorate, Populated places established in the 12th century	7	1
Occupation	Plumber	Construction trades workers, Industrial occupations, Occupations, Plumbing	4	3
Country	Germany	Germany, Central European countries, Countries in Europe, Federal republics, G7 nations, G8 nations, G20 nations, German-speaking countries and territories, Liberal democracies, Member states of NATO, Member states of the Council of Europe, Member states of the European Union, Member states of the Union for the Mediterranean, Member states of the United Nations, States and territories established in 1871, States and territories established in 1949, States and territories established in 1990	19	13
Electronic device	Mobile Phone	Mobile phones, Telecommunications-related introductions in 1973, 2000s fads and trends, Embedded systems, Mobile telecommunications, Mobile telecommunication services, New media-Telephony, Videotelephony Radio technology	10	0
Food	Pizza	Pizza, Flatbread dishes, Greek inventions, Italian cuisine, Italian inventions, Italian-American cuisine, Mediterranean cuisine, Popular culture, World cuisine, Cheese dishes, Snack foods, Convenience foods	12	8
Animal	Monkey	Monkeys, Extant Eocene first appearances, Paraphyletic groups	3	0
Vehicle	Car	Automobiles, Wheeled vehicles	2	2