Karlsruhe Institute of Technology

Abbildung von Webformularen auf aktive Ontologien

Masterarbeit
von

Wasim Said

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
Zweiter betr. Mitarbeiter:	Dipl.-Inform.Wirt. Mathias Landhäußer

Bearbeitungszeit: 17.08.2015 – 16.04.2016

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, den 15.04.2016

. . . . . .

(**Wasim Said**)

Kurzfassung

Immer häufiger werden intelligente Sprachassistenten von Smartphone-Nutzern verwendet. Dabei greift der Sprachassistent Siri von Apple auf die aktive Ontologie zurück. Hierbei muss für jeden Gegenstandbereich eine eigene aktive Ontologie definiert werden. Dies manuell durchzuführen, ist mit einem hohen Zeit- und Kostenaufwand verbunden. Deswegen arbeitet das Projekt EASIER des IPD Prof. Tichy an der semiautomatischen Erstellung aktiver Ontologien.

Diese Arbeit stellt einen Baustein von EASIER dar und definiert aktive Ontologien zur Nutzung von Webformularen. Diese verstehen sprachliche Anfragen von Nutzern über Flug-, Bahn- und Hotelbuchungen. Es wurde eine Umfrage durchgeführt um sprachliche Ausdrücke beim Buchungsvorgang zu sammeln. Danach wurde ein interaktiver Dialog-Manager entwickelt der mit dem Nutzer interagiert wenn unvollständige sprachliche Anfragen eingegeben werden. Zudem präsentiert er die Ergebnisse. Die Evaluation zeigt, dass die sprachlichen Anfragen mit hoher Präzision und einer Ausbeute von 73% erkannt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Immer mehr Nutzer verwenden intelligente Sprachassistenten wie z.B. „Siri“ von Apple, „Google Now“ von Google und „Cortana“ von Microsoft. Laut einer aktuellen Studie benutzen 47% der Menschen in Deutschland ihr Smartphone mittels Sprachbefehlen, anstatt über den Bildschirm Texte einzugeben und Anwendungen auszuführen [WP].

Eine in Siri verwendete Technik ist die „aktive Ontologie“, mit deren Hilfe sprachliche Anfragen verstanden werden und darauf reagiert wird. Die benötigten Informationen werden dabei durch die Sensorknoten der aktiven Ontologie erkannt.

Für jeden Gegenstandsbereich muss eine eigene aktive Ontologie definiert werden, um die sprachlichen Anfragen zu verstehen. Das erfordert einen großen manuellen Aufwand. Viel Zeit und Aufwand kann eingespart werden, wenn die Erstellung der aktiven Ontologien automatisiert wäre. Dies ist das Ziel des Projekts EASIER des IPD Prof. Tichy [BL], das sich mit der semiautomatischen Generierung von aktiven Ontologien beschäftigt.

In dieser Arbeit werden die aktiven Ontologien aus Webformularen abgeleitet. Es wurden Webformulare für Flug-, Bahn- und Hotelbuchungen ausgewählt, weil viele Studien aufzeigen, dass Nutzer bis zu zehn verschiedene Webseiten besuchen, um ihre Reise zu planen [MAY]. Eine andere Studie stellt fest, dass Nutzer ihre Präferenzen selten auf einer einzigen Webseite finden [Dav]. Dies zeigt unter anderem, dass Flug-, Bahn- und Hotelbuchungswebseiten einige der am meist genutzten Webseiten im Internet sind [Zee]. Interessanterweise hat jedoch die Art und Weise, wie die Nutzer deren Dienste verwenden, keine bemerkenswerten Fortschritte seit den 90er Jahren gemacht, insofern als die Nutzer heutzutage immer noch ihre Daten in ein Webformular per Tastatur eingeben und dementsprechende Suchergebnisse angezeigt werden.

Siri kann noch keine Flug-, Bahn- und Hotelbuchungen erledigen, da noch keine aktiven Ontologien für diese Kategorien erstellt werden. In dieser Arbeit werden Webformulare für die oben genannten Kategorien auf aktive Ontologien abgebildet. Somit könnten die Nutzer ihre Buchungen in Zukunft per Spracheingabe erledigen.

1.1. Zielsetzung

Das übergeordnete Ziel ist es, Formular-basierte Dienste semiautomatisch nutzen zu können, sodass der Nutzer seine Buchungswünsche dem Computer sprachlich übermittelt, woraufhin der Computer die Suchaufgabe in den von den Buchungswebseiten angebotenen

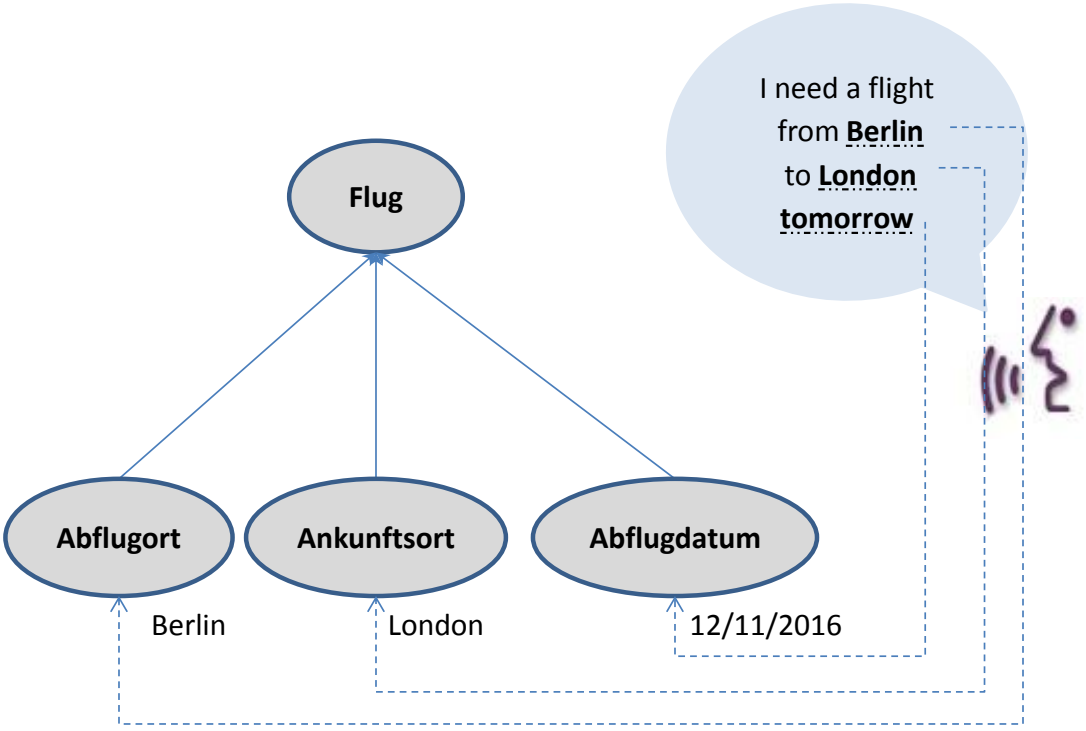

Abbildung 1.1.: Erkennen einer Anfrage mithilfe von aktiven Ontologien

Diensten ausführt. Das kann eine Einsparung von Zeit und Aufwand zur Folge haben sowie die Erschließung neuer Nutzer dieser Dienste.

Aus technischer Sicht ist es das Ziel dieser Arbeit, aktive Ontologien für Flug-, Bahn - und Hotelbuchungen zu entwerfen. Deren Konzepte, Knoten, Regeln, Aktionen und Beziehungen werden aus den Analysen der Webformulare von den jeweiligen Buchungswebseiten abgeleitet. Dazu muss eine geeignete Technik gewählt werden, welche die benötigten Informationen (Felder der Webformulare bzw. Blätter der aktiven Ontologie) – z.B. für Flugbuchungen: Startpunkt, Zielpunkt oder Datum – aus den sprachlichen Eingaben der Nutzer herausfiltert. Abbildung 1.1) zeigt die Erkennung einer sprachlichen Anfrage von aktiven Ontologie. Dabei werden die Informationen „Berlin“, „London“ und „tomorrow“ den Konzepten Abflugort, Ankunftsort und Abflugdatum zugewiesen.

Um das gesamte System interaktiver und einfacher verwenden zu können, wird ein Dialog-Manager entworfen. Er konvertiert die für das System benötigten Informationen zu interaktiven Fragen mit verschiedenen Antwortmöglichkeiten. So kann der Benutzer seine Wünsche und Anforderungen vollständig eingeben. Anschließend können die Dienste mit dieser Eingabe aufgerufen werden.

1.2. Ablauf des Verfahrens

Der Ablauf des hier vorgestellten Verfahrens besteht aus den folgenden Phasen (siehe Abbildung 1.2): Zunächst werden zehn Webseiten pro Kategorie (Flug, Bahn und Hotel) ausgewählt, die den gleichen Dienst in der jeweiligen Kategorie wie z.B. Flugbuchung in Abbildung 1.2 anbieten. Diese zehn Webseiten werden unter Verwendung verschiedener Kriterien ausgewählt. Ein Beispiel hierfür ist die Art und Weise, wie Hotels aus verschiedenen Klassen (Sternen), Kontinenten und Kulturen ausgewählt werden, um die größt-

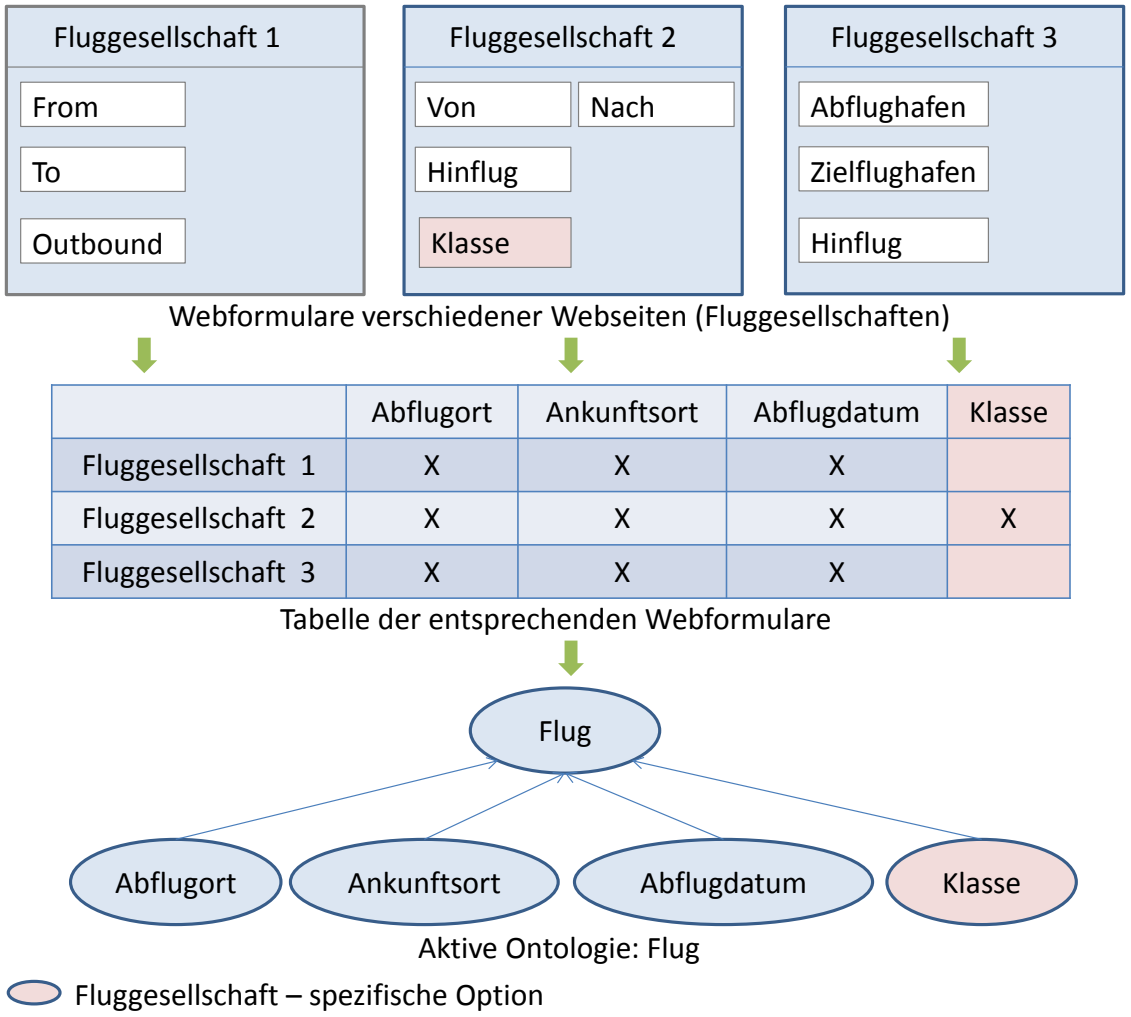

Abbildung 1.2.: Webformulare auf aktive Ontologien abbilden

mögliche Anzahl von Informationen über Formularelemente zu sammeln. Abbildung 1.2 zeigt den Arbeitsansatz.

Die Semantik der Formularelemente wird analysiert, um einen Namen für jedes Formularfeld bzw. für jedes Konzept der aktiven Ontologie festzulegen. Zu diesem Zweck wird eine Tabelle mit den Webseiten und den unterschiedlichen Optionen erstellt.

Die Felder der Webformulare unterscheiden sich je nach Sprache und Fluggesellschaft. „From“ und „To“ auf Englisch wird „Von“ und „Nach“ auf Deutsch oder „Abflughafen“ und „Zielflughafen“ usw. (siehe Abbildung 1.2). Diese Varianten werden in einem Konzept vereinigt und in einem Knoten in der aktiven Ontologie repräsentiert. Hier hilft auch die Tabelle, um die Konzepte der Kategorie (die Sensorknoten der generellen aktiven Ontologie) festzulegen.

In der zweiten Phase wird eine aktive Ontologie manuell so gebaut, dass jedes Feld des Webformulars unter Verwendung verschiedener Techniken, wie z.B. Wortlisten, Präfix, Postfix und reguläre Ausdrücke, zu einem Blatt der entsprechenden aktiven Ontologie abgebildet wird (siehe Abbildung 1.2). Diese aktive Ontologie enthält alle notwendigen und gemeinsamen Optionen aller Webseiten und gilt als generelle Ontologie für die jeweilige Kategorie.

Auf diese Weise entsteht eine vereinfachte aktive Ontologie für z.B. einen Flug aus drei obligatorischen Knoten: „Abflugort“, „Ankunftsort“ und „Abflugdatum“. Diese Knoten sind

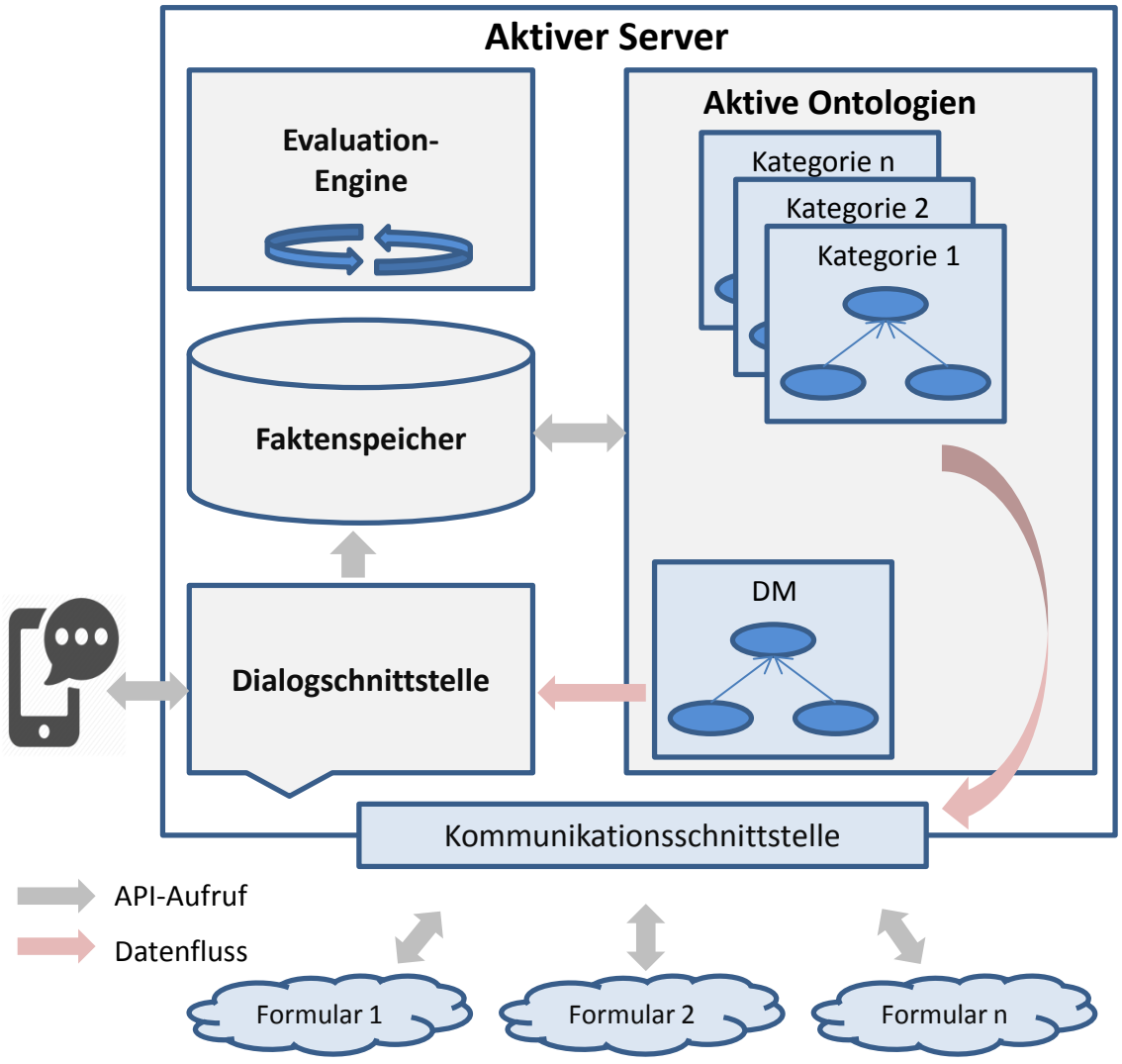

Abbildung 1.3.: Aktiver Server [BL]

notwendig für jeden Flug. Webseitenspezifische Optionen wie z.B. „Klasse“ werden als optionale Knoten zu dieser Ontologie hinzugefügt, da sie zusätzliche, nicht kritische Informationen für den Flug beinhalten (Abbildung 1.2). Die Eingabe wird in den Sensorknoten oder Blättern empfangen, wo die eintreffenden Informationen gefiltert und erkannt werden (siehe Abbildung 1.1).

1.3. Durchführung des Verfahrens

Das Projekt „EASIER“ bietet die Plattform „Aktiver Server“, welche eine Evaluation-Engine und einen Faktenspeicher beinhaltet. In dieser Arbeit wird die Entwicklung eines Dialog-Managers vorgestellt, welche das Projekt „EASIER“ ergänzt. (Abbildung 1.3 zeigt den gesamten Aktiven Server).

Ein Dialog zwischen Assistent und Nutzer wird mittels eines Dialog-Managers geführt. Dieser stellt Fragen an den Benutzer und bittet um noch fehlende Informationen. Gibt ein Nutzer z.B. bei seiner Flugbuchung etwa lediglich Reiseziel und -datum an, so wird er im Folgenden um die Eingabe des Startpunkts gebeten. Diese Informationen sind notwendig für die Suche nach dem gewünschten Flug.

Nach der Auswahl eines Flugs erfährt der Nutzer zusätzliche Optionen wie Mahlzeit und Getränke, Gepäck und andere Fluggesellschaft-spezifische Optionen im Dialog. Der Dialog-Manager erstellt je nach angebotenem Service der gewählten Fluggesellschaft individuelle Fragen, die dem Nutzer Antwortmöglichkeiten in interaktiver Art und Weise präsentieren. Fehlt bei der Eingabe beispielsweise das Reisedatum, so wird dem Nutzer ein Kalender angezeigt. Fehlt dagegen die Reiseklasse, erhält er eine Liste möglicher Optionen.

Wenn der Nutzer alle notwendige Informationen eingibt, werden sie mittels Web-Scraping in den Webformularen der Dienstleister abgeschickt. Die Suchergebnisse bzw. die Informationen, die der Nutzer braucht, werden ebenfalls mithilfe von Web-Scraping gesammelt und dem Nutzer angezeigt.

1.4. Evaluation

Das in dieser Arbeit entwickelte System erhält sprachliche Anfragen von den Nutzern und versucht, diese Anfragen sinngemäß zu verstehen und zu beantworten. Daher wird im Laufe der Arbeit eine Umfrage durchgeführt, in der Nutzer von Sprachassistenten, darunter Personen mit Englisch als Erst- oder Zweitsprache, gebeten werden, eine Flug-, Bahn- und Hotelbuchung sprachlich durchzuführen. Diese Umfrage hilft bei der Sammlung von Ausdrücken und sprachlichen Varianten, die die Nutzer verwenden, um Informationen über ihre Anfrage zu geben. Die eine Hälfte der Ergebnisse dieser Umfrage wird für die Entwicklung des Softwareprogramms genutzt. Die andere wird für die Evaluation benutzt. Jede Anfrage wird an das System gestellt. Dabei werden die Antworten bzw. die Rückfragen des Systems betrachtet und es wird bestimmt, ob alle Informationen aus der Anfrage erkannt wurden und ob die richtigen Rückfragen gestellt werden.

1.5. Struktur der Arbeit

Die Arbeit gliedert sich wie folgt auf: Zuerst werden die Grundlagen in Kapitel 2 vorgestellt. Kapitel 3 bietet zudem einen Überblick über den Forschungskontext. Kapitel 4 diskutiert die Problemstellung und stellt Lösungsansätze vor. Dies beinhaltet die Analyse der Formulare, die Identifizierung der Konzepte der jeweiligen aktiven Ontologien und den Ansatz des Dialog-Managers. Aufbauend auf der Analyse werden Lösungsstrategien in Kapitel 5 in einen Entwurf überführt und die Implementierung der Software realisiert. Die Evaluation des Systems wird in Kapitel 6 durchgeführt. Zum Schluss wird in Kapitel 7 die vorliegende Arbeit zusammengefasst und ein Ausblick auf sich daraus ableitende Entwicklungen gegeben.

2. Grundlagen

In diesem Kapitel werden Konzepte, Technologien und alle für den weiteren Verlauf der Arbeit relevanten Begriffe erläutert. Dazu zählen das Basiswissen und der spezielle Wirkungsmechanismus der aktiven Ontologie. Zunächst wird die Struktur der Auszeichnungssprache HTML und insbesondere der Aufbau und die Eigenschaften der Webformulare dargestellt. Weiterhin sind Grundlagen und Besonderheiten des Web-Scraping Gegenstand der Betrachtung. In diesem Zusammenhang wird auch das in dieser Arbeit benutzte Werkzeug „Selenium“ beschrieben.

2.1. Wissensrepräsentation und Ontologien

Die Wissensrepräsentation wurde in den letzten Jahren im Hinblick auf die Implementation größerer Systeme und der Entwicklung einer neuen Klasse von Programmiersprachen intensiv bearbeitet. Eine dieser Repräsentationen ist die Ontologie, die als die Beschreibung und Organisation von bestehenden Entitäten definiert wird. Sie fasst die Hierarchien der Konzepte, Definitionen, Beziehungen und Axiome zusammen, die zu einem Gegenstandsbereich gehören [FMTT13]. Die Konzepte beschreiben eine Menge von individuellen Objekten, die gemeinsame Eigenschaften haben. Mit dem Konzept „Reisende“ sind z.B. alle Passagiere, egal ob Erwachsene oder Kinder, ob ankommende oder abfahrende, enthalten. Die Beziehungen beschreiben jedoch, welche Relationen zwischen den Konzepten bestehen. Beispielsweise ist die Beziehung zwischen „Auto“ und „Fahrzeug“ „ist eine“ und die Beziehung zwischen „Person“ und „Geburtsdatum“ „hat ein“.

Die Abbildung 2.1 beschreibt, wie eine Ontologie „Buch_Ontologie“ die Buchdomäne beschreiben kann. Diese Ontologie enthält die Konzepte „Titel“, „Autor“, „Erscheinungsdatum“, „Verlag“, „Thema“ und „Kategorie“. Alle diese Konzepte sind „Mitglieder“ von dem Konzept „Buch“, deswegen heißt die Beziehung „ist Mitglied von“. Die Beziehung zwischen „Roman“ und „Buch“ heißt „ist ein“, denn ein „Roman“ „ist ein“ „Buch“ und hat alle Bucheigenschaften.

Die Ontologie ermöglicht, gemeinsame Informationsstrukturen zwischen Personen und Software zu teilen, um die Domäne explizit zu repräsentieren und deren Wissensbestand wieder verwenden zu können. Aus diesem Grund wurde die Ontologie in vielen Bereichen verwendet, wie z.B. in der Computerlinguistik, der künstlichen Intelligenz, in den Informationssystemen oder im World-Wide-Web. Dabei wird auf die Ontologie als Datenstruktur zurückgegriffen. Sie führt jedoch noch keine Verarbeitung oder Ausführung durch. Sie ist in diesem Sinne noch nicht „aktiv“.

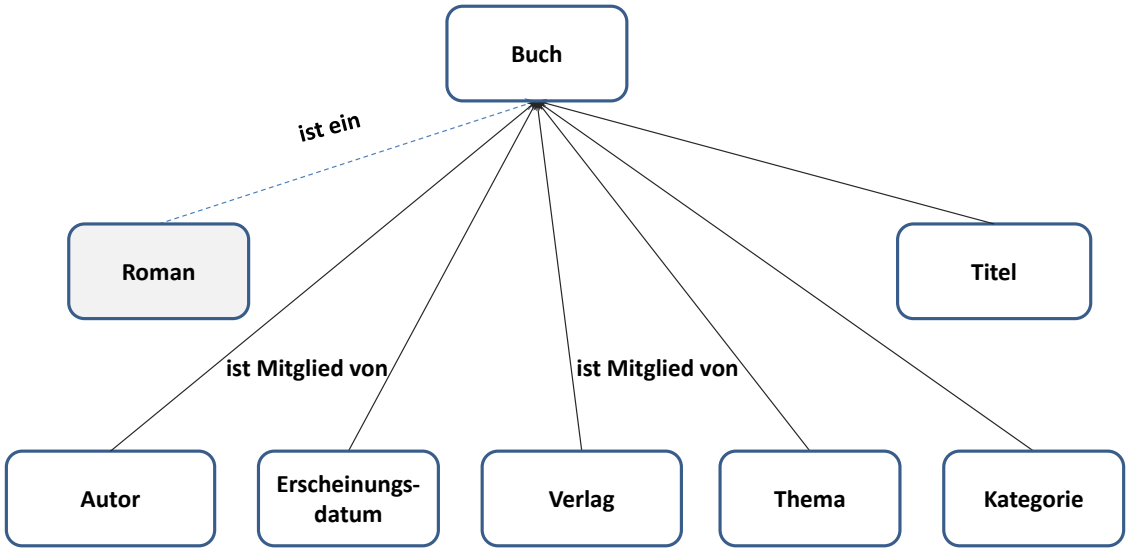

Abbildung 2.1.: Konzepte und Beziehungen der Ontologie „Buch“

2.2. Aktive Ontologien

Die aktive Ontologie [Guz08] basiert auf der konventionellen Ontologie, sie behält deren Grundeigenschaften wie die Verwendung von Konzepten und Beziehungen. Die Konzepte in der aktiven Ontologie haben jedoch noch weitere Verarbeitungselemente, um die Domänobjekte, -erreignisse, -aktionen und -prozesse zu repräsentieren. Die Verarbeitungselemente können als Verarbeitungsschicht in den Konzepten vorgestellt. Die Verarbeitung wird in der aktiven Ontologie durch Regelsätze, die zu Konzepte abgehängt werden, dargestellt. Ein Regelsatz lässt sich als eine Sammlung von Regeln bezeichnet, dabei besteht jede Regel aus einer Bedienung und einer oder mehreren Aktionen, die ausgeführt werden wenn die Bedienung erfüllt wird. Jede aktive Ontologie ist mit einem Datenspeicher zugeordnet, der verwendet wird, um die Fakten, die den Zustand und Variablen der aktuellen Verarbeitung beschreiben, anzuhalten. Dieser Datenspeicher wird als Faktenspeicher bezeichnet. Die Daten und Ereignisse, die im Faktenspeicher abgelegt sind, werden durch die Regeln bearbeitet. Wurde ein Fakt im Faktenspeicher geändert, wird ein neuer Evaluationszyklus gefeuert und werden die Bedienungen der Konzepte ausgewertet. Die Beziehungen zwischen Konzepte in der aktiven Ontologie repräsentieren nicht nur wie die Konzepte untereinander stehen, sonder stellen auch Kommunikationskanäle für die Verarbeitungselemente zur Verfügung. Auf diese Art und Weise werden die Domäne einer Anwendung zusammen mit den damit verbundenen Verarbeitungsregeln in einem einzigen Blick dargestellt. Daher ist die aktive Ontologie nicht nur eine Datenstruktur sondern eine Ausführungsumgebung und anders ausgedrückt, sie ist aktive. Abbildung 2.2 zeigt die Komponente der aktive Ontologien, die im folgenden fortlaufend erklärt werden.

2.2.1. Fakten

Fakten sind zentrale Datenstrukturen, die zusätzlich zur Darstellung des Domänenwissens verwendet werden, werden auch zur Aufrechterhaltung des aktuellen Ausführungszustands der Ontologie oder Kommunikation zwischen Konzepten eingesetzt. Sie werden im Faktenspeicher abgelegt und ausgelesen. Jeder Fakt hat einen Lebenszyklus, der vom Program-

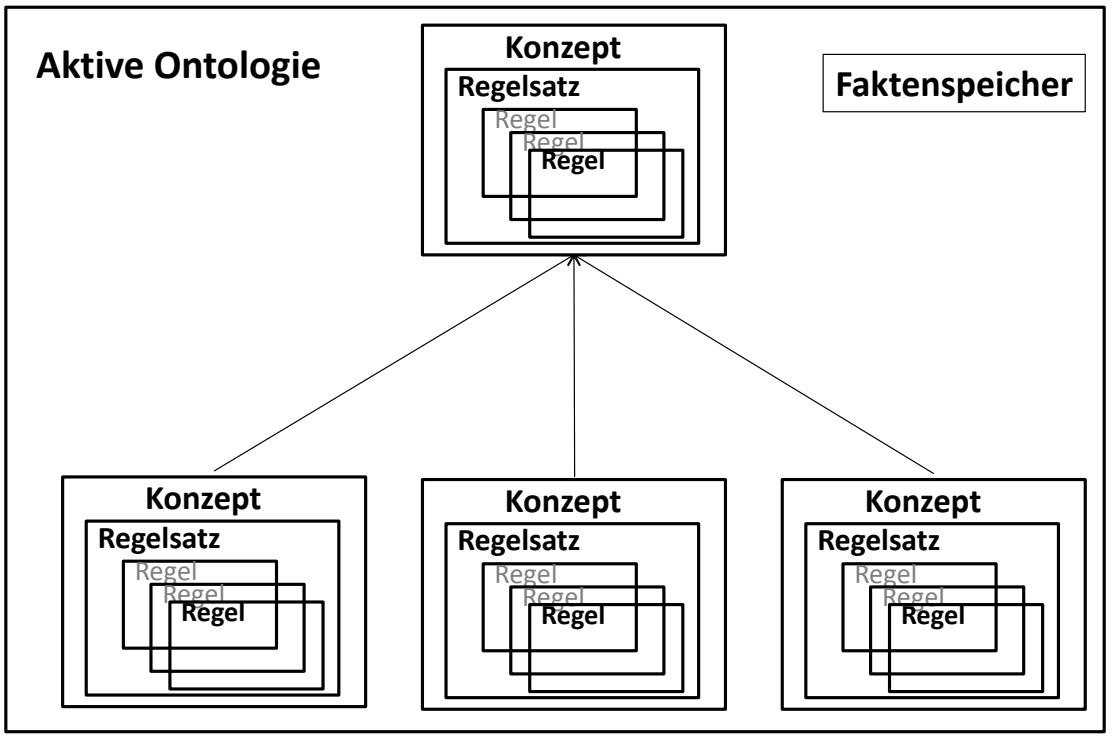

Abbildung 2.2.: Komponente der aktiven Ontologie

mierer erstellt wird. Die Fakten können ebenso vom Programmierer geltend gemacht und gelöscht werden.

Fakten repräsentieren Informationen in vier unterschiedlichen Arten:

1. Einfache Fakten: Darunter sind atomare Konstanten zu verstehen, wie z.B. 2, 'Karlsruhe'.
2. Komplexe Fakten: Sie werden als benannte Prädikate mit Argumenten bezeichnet, die einen anderen Fakt sein können. Im Fakt Flug(Departure('Karlsruhe'), Arrival('London'), Datum(Tag(2), Monat(5), Jahr(2015))) ist ein Flug dargestellt, der von Karlsruhe nach London am zweite Mai 2015 fliegt.
3. Listen: Sie sind sortierte Sammlungen von Fakten wie z.B. ['Karlsruhe', 'Berlin', 'London'].
4. Variablen: Dabei handelt es sich um Identifikatoren, die mit '$' beginnen und unterschiedliche Werte haben können . 'Wildcards' und anonyme Variablen, bestehen ausschließlich aus $ und stehen für einen oder mehreren beliebigen Werte, Variablen oder ebenfalls anderen Fakten. Die Variablen sind notwendig, um die unbekannte Informationen in einer Abfrage zu repräsentieren. Beispielsweise steht $Flughafen für einen beliebigen Flughafen. In diesem Sinn ist Flug('Karlsruhe', $Ziel , Datum(Tag($), Monat(5), Jahr(2015))) ein Flug, der von Karlsruhe nach irgendeinem Ziel an irgeneinem Tag im Mai 2015 fliegt. „$“ (alleine) steht jedoch für irgendeinen Wert oder irgendeinen Fakt.

2.2.2. Regeln

Jede Regel hat einen Namen, wie z.B printFlight und besteht aus einer Vorbedingung und einer oder mehrerer Aktionen. Die Vorbedingungen sind Boole'sche Ausdrücke, die als

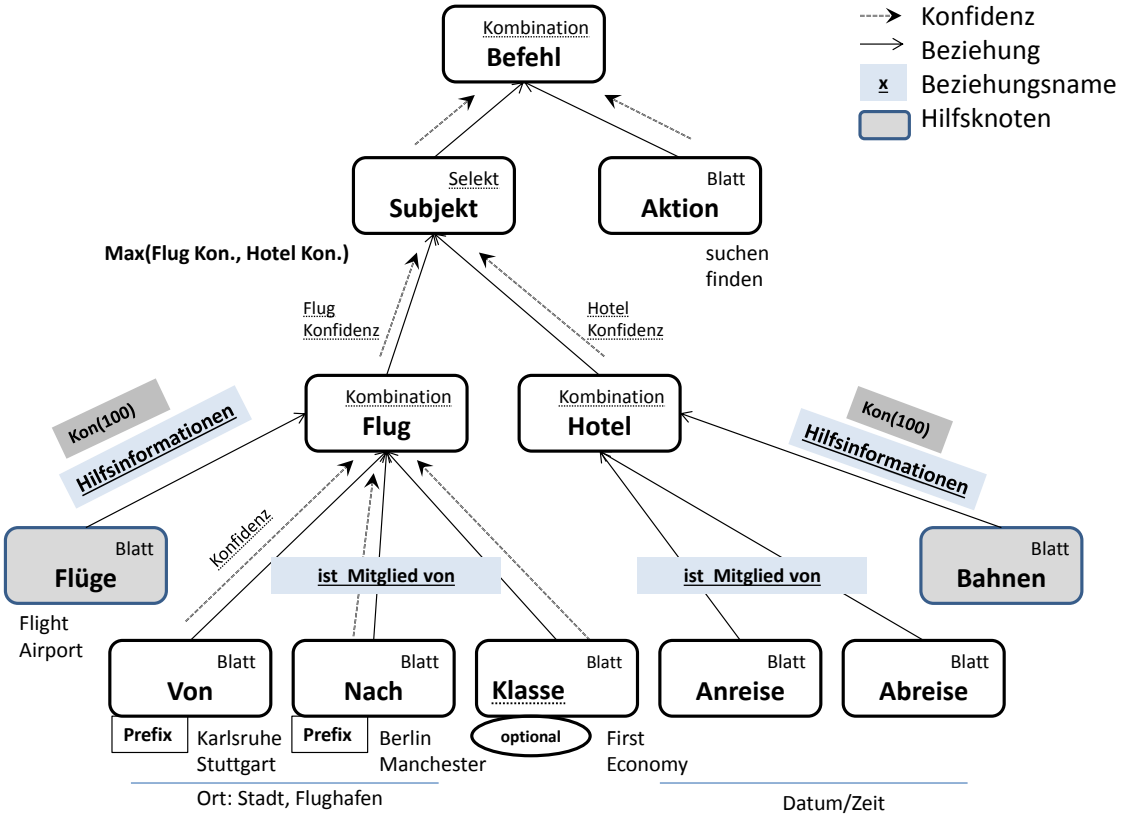

Abbildung 2.3.: Darstellung der Bestandteilen der aktiven Ontologie

wahr oder falsch ausgewertet werden. Sie können Variablen enthalten und mit AND, OR oder NOT verknüpft sein. Die Aktionen können beispielsweise neue Ereignisse erstellen und neue Daten bzw. Fakten in den Faktenspeicher einsetzen und die Variablen in der Vorbedingung nutzen.

2.2.3. Darstellungselemente der aktiven Ontologie

Die aktive Ontologie lässt sich durch TerminalKnoten (Blätter), NonterminalKnoten und Beziehungen darstellen. Abbildung 2.3 zeigt wie diese Bestandteile in einer aktiven Ontologie repräsentiert werden.

2.2.3.1. Sensorknoten (Terminal)

Die Sensorknoten oder Blätter arbeiten als Filter, die die sprachliche Anfrage des Nutzers (Wortsequenzen) empfangen, die eintreffende Informationen (Wörter) erkennen und ihre Bedeutung bewerten. Z.B. sind die Knoten „Von“ und „Nach“ in der Abbildung 2.3 Blätter. Sie erkennen die Wörter „Karlsruhe“, „Stuttgart“ „Berlin“ und „Manchester“. Sensorknoten verwenden folgende Techniken, um die Wörter zu erkennen.

- Wortlisten: Diese Technik basiert auf dem Vergleich von Eingabewörtern mit einer vorgegebenen Wortliste. Diese vorgegebene Liste kann die Wörter, ihre verschiedene Bedeutungen und Synonyme beinhalten, z.B reagiert das Blatt „klasse“ auf „first“ und „economy“, die Optionen der Flugklasse darstellen (siehe Abbildung 2.3). Um mehrere Möglichkeiten wie ungrammatische Ausdrücke und Falscherkennungen zu

unterstützen, benutzen die Sensoren den Vergleich über Metrik (z.B. Levenshtein-Distanz [1]).

- Heuristiken:

  1. Präfix und Postfix: Sie sind dazu da, um nachfolgende und vorangegangene Wörter zu erkennen, wie z.B die Präfixe „from“ und „to“ im Satz „find me a flight from Karlsruhe to Berlin“, um die Städte oder Flughäfen als Startpunkt bzw. Zielpunkt in einer Flugontologie zu filtern.

  2. Reguläre Ausdrücke: Diese Technik wird für Datumsangaben wie „12/05/2015“ E-Mails, Postleitzahlen usw. benutzt, das heißt, für die Wörter, die ein bestimmtes Muster übereinstimmen.

  3. Spezialheuristiken: Darunter sind hart-kodierte Logiken zu verstehen, die spezialisierte Bewertungen generieren, z.B werden „tomorrow“ und „next Saturday“ in folgender Form transformiert: date($year, $month, $day).

2.2.3.2. Konzeptknoten (Non Terminal)

Dazu zählen folgende Ausdrücke:

- Kombinationsknoten: Sie kombinieren die Informationen ihrer Kinder zu dem Knoten mit neuer Bewertung. Die Knoten „Flug“ und „Hotel“ in Abbildung 2.3 sind Kombinationsknoten. So besteht ein Flug aus den Kind-knoten „Von“, „Nach“ und „Klasse“.
- Selektionsknoten: Darunter fallen Elternknoten, die das Kind mit der höchsten Bewertung auswählen. Als Beispiel ist der Knoten „Subjekt“ ein Selektionsknoten (siehe Abbildung 2.3).

Jeder Konzeptknoten gibt seine Bewertung auch an seine Elternknoten weiter.

Zu Kombinationsknoten können besondere Sensorknoten, die als Hilfsknoten bezeichnet werden, verbunden werden, um Zusatzinformationen mit 100% Konfidenz zu liefern, ohne zum weitergegebenen Inhalt beizutragen: z.B „Flight“ und „Airport“ zeigen an, dass der Nutzer über „Flug“ spricht. Abbildung 2.3 zeigt die Hilfsknoten „Flüge“ und „Bahnen“.

Die Kindknoten in einer aktiven Ontologie können Obligatorisch oder optional sein:

1. Obligatorische Knoten: Sie beinhalten notwendige und unverzichtbare Informationen, um übergeordnete Konzepte zu beschreiben. Bei einem Flug bspw. sind der „Von“ und „Nach“ obligatorisch. Fehlt bei der Eingabe einer dieser Knoten, muss der Nutzer gefragt werden, um diese Informationen einzugeben.
2. Optionale Knoten: Sie überliefern zusätzliche und nicht kritische Informationen. So stellt die „Klasse“ mehr Informationen für den Flug bereit (siehe Abbildung 2.3).

2.2.3.3. Beziehungen

Die Beziehungen beschreiben, wie die Knoten miteinander verbunden sind. Als Beispiel ist der Knoten „Von“ ein Bestandteil des Knotens „Flug“. Deswegen wird „Von“ als „Mitglied von“ „Flug“ bezeichnet. Dahingegen wird die Beziehung zwischen „Flug“ und „Subjekt“ wie folgt definiert: „Flug“ „ist ein“ „Subjekt“ (siehe Abbildung 2.3).

[1] Die Levenshtein-Distanz zwischen zwei Strings ist die minimale Anzahl von Operationen (Zeichenpermutation, Addieren und Entfernung), die auf einen String angewendet werden müssen, um äquivalent zu den anderen zu sein.

2.2.3.4. Pipes

Pipes sind gerichtete Kommunikationskanäle zwischen den Konzepten. Sie etablieren Verbindungen zwischen den Konzepten einer Ontologie oder Relationen zu anderen aktiven Ontologien. Die Pipes tragen weitere Informationen wie z.B. die Konfidenz. Das Wort „first“ wird z.B. mit 75% Konfidenz als Wort des Knotens „Klasse“ erkannt.

2.3. Hypertext Markup Language

Die Hypertext Markup Language (HTML) ist eine Markierungssprache. HTML-Dokumente werden mithilfe von Markierungsbefehlen oder „Tags“ formatiert und durch einen Browser interpretiert. Mithilfe der HTML können Dokumente mit Überschriften, Texten, Tabellen, Fotos, Soundclips und anderen Applikationen online formatiert werden. Eine HTML-Datei besteht aus Tags und Angaben. Die Befehle werden durch einen Start- und einen End-Tag dargestellt. Z.B. definieren die Tags *<html> </html>* das gesamte HTML-Dokument. Kopfdaten der HTML-Datei werden in den *<head> </head>* notiert. Der Textkörper lässt sich durch *<body> </body>* markiert. Dazwischen wird der eigentliche Inhalt, der im Browser angezeigt werden soll, notiert.
HTML-Elemente können mehrere Attributen haben, die zusätzliche Angaben zu diesem Element beinhalten. Attribute befinden sich nur in dem Starttag als Schlüssel-Wert-Paare wie name= „value“. Tabelle 2.1 listet die wichtigsten Attribute von HTML-Elementen.

Attribut	**Wert**	**Beschreibung**
pattern	<regulärer Ausdruck>	die Eingaben werden mit regulären Ausdrücken geprüft
max	<number>, <date>	maximaler Wert des Eingabefeldes
min	<number>, <date>	minimaler Wert des Eingabefeldes
required	required	Element ist erforderlich, ansonsten kann das Formular nicht abgeschickt werden
placeholder	<text>	hilft dem Benutzer durch einen vordefinierten Text zur möglichen Eingabe.
multiple	multiple	mehrere Werte können im Element eingegeben werden
list	<datalist-id>	vordefinierte Optionswerte für input-Elemente
autocomplete	on, off, default	speichert die Eingabe für eine erneute Nutzung
maxlength	<number>	maximale Anzahl von Zeichen

Tabelle 2.1.: Häufig verwendete Attribute von HTML-Elementen

2.4. Webformulare

Formulare sind ein wichtiger Bestandteil aller Webangebote, da sie Interaktionen mit dem Nutzer erlauben, der Eingabefelder ausfüllen, einen Text eingeben oder Daten aus Listen auswählen kann. Der Anwender kann nach dem Ausfüllen des Formulars die Daten an den

Server übermitteln, wo diese Daten abgearbeitet werden, um dem Nutzer die passende Antwort zur Verfügung zu stellen. Webformulare sind eine der besten Möglichkeiten, um Eingaben von Nutzern zu erhalten. Aus diesem Grund sind sie ein unverzichtbares Mittel aller Webseiten, besonders für die Webseiten, die Produkte verkaufen oder Dienstleistungen ihren Kunden anbieten.

Mit *<form> </form>* kann der Entwickler ein Formular definieren. Alle Bestandteile, die zum Formular gehören, werden zwischen diesen beiden Tags angegeben. Das Element *<form>* kann Attribute beinhalten wie „action“, das eine „URL“ eines Skripts oder Programms auf dem Server hat, wo die Daten verarbeitet werden sollen. Das Attribut „method“ definiert die Art, in der Daten an den Server geschickt werden. Die möglichen Werte sind „get“ und „post“. Bei der Methode „get“ werden die Daten als Parameter an die Aufrufadresse angehängt und bei der „post“ sind die Formulareingaben ein Teil der HTTP-Anfrage und werden vom Webserver daraus ausgelesen und über den Standardeingabekanal zur Verfügung gestellt.

Formularfelder können unterschiedliche Formen annehmen und für verschiedene Funktionen benutzt werden, wie z.B als Eingabefelder für Daten. Tabelle 2.2 klassifiziert Formularelemente nach deren Funktionen.

Eingabefelder für Daten	**Absenden des Formulars**	**Darstellung des Formulars**
input	input	fieldset
select / option	button	label
textarea		legend

Tabelle 2.2.: Klassifikation der Formularelemente

Das folgende Beispiel weist die wichtigste Elemente des Webformulars auf. Der Browser wird diesen Code genauso wie in der Abbildung 2.4 anzeigen. Der Code des Labels wiederholt sich in alle Labels, deswegen werden nicht alle gezeigte Labels in diesem Beispiel kodiert.

Quelltextausschnitt 2.1: HTML Code für einige Formularelemete

```
1  <form action="http://www.cs.tut.fi/cgi-bin/run/~jkorpela/echo.cgi">
2
3  <label for="but">Button:
4  <button id="but" type="submit" name="foo" value="bar">Button 1</button>
       </label>
5
6  <label for="f0">Reset button:
7  <input id="f0" type="reset" name="reset" value="Reset">
8
9  <input id="f1" name="text" size="20" value="Default␣text.">
10 <textarea id="f2" name="textarea">Default text.</textarea>
11 <input id="f3" type="radio" name="radio" value="1">
12 <input id="f4" type="radio" name="radio" value="2" checked="">
13 <input id="f5" type="checkbox" name="checkbox">
14 <input id="f6" type="checkbox" name="checkbox2" checked="">
15
16 <label for="f10">A <code>select</code> element with
17 <code>size="1"</code>
18 (dropdown box):
19 <select id="f10" name="select1" size="1">
20 <option>one
21 </option><option selected="">two (default)
22 </option><option>three
```

```
23 </option></select></label>
24
25 <input id="f99" type="submit" name="submit" value="Just␣a␣test">
26 </form>
```

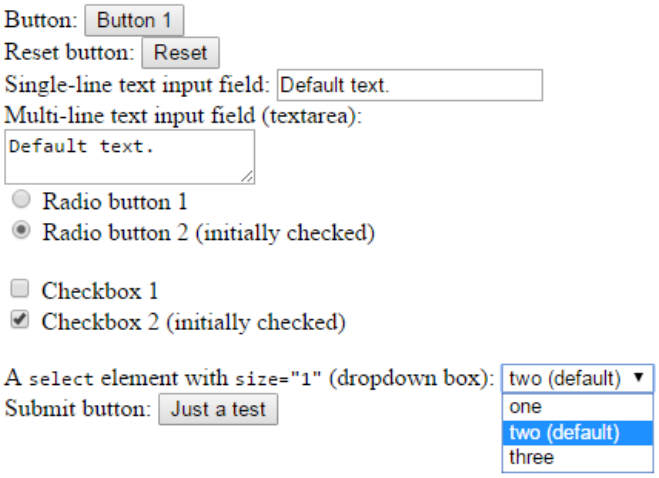

Abbildung 2.4.: Elemente des Webformulare im Browser

2.4.1. Input

Input-Elemente wie Text, Schaltflächen, Checkboxen und Radiobuttons stellen dem Formular nützliche Funktionen zur Verfügung. Ihre Funktion wird durch das Attribut *type* festgelegt. Die mögliche Werte der *Type* Attribute werden in der Tabelle 2.3 gezeigt. Das Attribut *name* identifiziert das Eingabefeld und *value* beinhaltet den angezeigten Wert. Als Beispiel dafür ist das Element mit dem Namen „text“ in der Zeile neun von Quelltextausschnitt 2.1.

2.4.2. Select - Option

Das Element <*select*> stellt eine Drop-Down-Liste zur Verfügung und lässt die Wahl einer oder mehrerer Optionen zu. Die Optionen werden über das Element <*option*> angegeben. Zu diesem Zweck können auch *input type=„checkbox“*, *type=„radio“* oder (<*datalist*>, <*optgroup*> - Mit Gruppierungsauswahl und nicht auswählbaren Überschriften - in HTML5) eingesetzt werden. Die Zeilen 19 bis 23 in Quelltextausschnitt 2.1 definieren ein <*select*> mit drei Optionen.

2.4.3. Textarea

Mit dem *Textarea* wird ein mehrzeiliges Eingabefeld für Benutzereingaben bereit gestellt.

2.4.4. Button

Buttons lassen sich durch das Element <*button*> in einem Formular darstellen. Der Inhalt der Schaltfläche kann aus einem Bild oder Text bestehen. Es werden die Typen reset, button und submit in der Tabelle 2.3 beschrieben.

2.4.5. Label

Das Element <*label*> gibt Eingabefeldern, Buttons und Checkboxen eines Formulars eine Bezeichnung bzw. Titel. Die Verknüpfung wird darüber durchgeführt, dass das for-Attribut des Labels genau den gleichen Wert wie das ID-Attribut im Eingabefeld hat.

Kategorie	Typ	Beschreibung
Texteingabe	text	Einzeiliges Eingabefeld
	search	Suchfeld
	email	Einzeiliges Eingabefeld, überprüft die Gültigkeit der Email-Adresse
	URL	Einzeiliges Eingabefeld, überprüft die Gültigkeit der URL
	password	Einzeiliges Eingabefeld, Textzeichen werden als Sternchen dargestellt
Zahleneingabe	number	Einzeiliges Eingabefeld, besteht aus Ziffern
	tel	Einzeiliges Eingabefeld, gibt kein festes Schema vor
Zeit-Angaben	date	Eingabefeld, Auswahl eines Datums
	time	Eingabefeld, Auswahl der Uhrzeit
	datetime	Eingabefeld, Datum und Uhrzeit mit UTC-Zeitzone
	datetime-local	Eingabefeld, Datum und Uhrzeit
	month	Eingabefeld, Auswahl eines Monats
	week	Eingabefeld, Auswahl einer Kalenderwoche
Buttons	button	Schaltfläche, löst clientseitige Aktionen aus
	submit	Schaltfläche, sendet das Formular ab, default-Wert
	reset	Schaltfläche, setzt das Formular zurück
	image	Schaltfläche, als Grafik
Auswahl	checkbox	Auswahlbox, kann selektiert oder nicht selektiert sein
	radio	Auswahlbox, innerhalb einer Gruppe nur ein aktivierter Radiobutton
Weitere Elemente	range	Eine bestimmte Spanne numerischen Werte
	color	Eingabefeld für Farbauswahl
	file	Eingabefeld und Schaltfläche, lädt eine Datei hoch
	hidden	Unsichtbare Feld

Tabelle 2.3.: Beschreibung der Eingabeelemente der Webformulare

2.4.6. Fieldset - Legend

Mit dem Element <*fieldset*> wird ein Rahmen um einen Bereich (Gruppe von zusammengehörigen Feldern eines Formulars) gelegt. Das Element <*legend*> gibt eine Überschrift oder Kurzbeschreibung für einen Block. Die beiden Elemente beinhalten keine Informationen und werden als Abstandhalter benutzt.

Andere Formularelemente wie beispielsweise <*keygen*>, <*output*>, <*progress*>, <*meter*> werden in HTML5 in dieser Rheinfolge als Schlüsselgenerator, Ergebnisanzeiger, Fortschrittsanzeiger wie z.B beim Ladevorgängen und Anzeiger für Werte eines Minimal- und Maximalwertes verwendet.

2.5. Web-Scraping

Web-Scraping, bekannt auch als Web-Harvesting oder Screen-Scraping, ist die Technik, die automatisch große Mengen an Daten und den Inhalt von Webseiten durch die Verarbeitung des HTML Codes sammelt. Dieser Inhalt kann normalerweise nur mit dem Webbrowser angezeigt werden. In einigen Fallen ist es nicht möglich, eine Kopie der Daten zu speichern. Die einzige Möglichkeit ist es, die Daten zu kopieren und in einer lokalen Datei auf dem Rechner einzufügen, was viel Zeit und Aufwand kostet.

Web-Scraping automatisiert diesen Prozess binnen kürzester Zeit. Dabei wird eine Person simuliert, die im Internet surft. Der Web Scraper ruft eine Webseite genauso wie ein Browser auf. Der Webserver schickt dann die Webseite zurück, die verarbeitet werden kann, um spezifische Informationen zu extrahieren und strukturierte Daten in einer lokalen Datei oder Datenbank zu speichern wie z.B. (Textdatei, SQL-Datei oder XML-Datei)(siehe Abbildung 2.5).

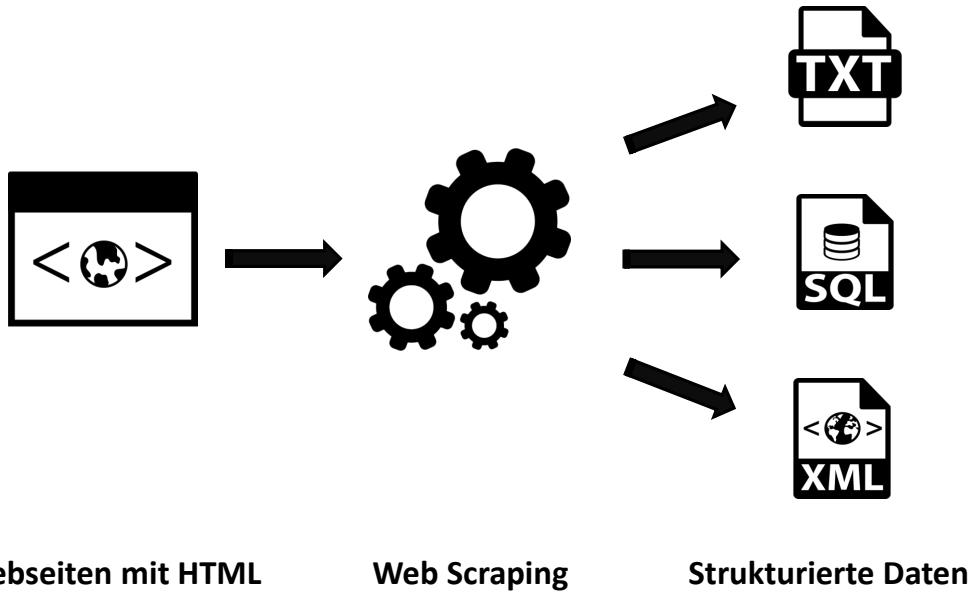

Abbildung 2.5.: Mögliche Ausgabe des Web-Scrapings

Die Web Scraper nehmen den Inhalt der Webseiten, anders als es beim Web Indexing der Fall ist, bei dem Suchmaschinen den Inhalt nach den Protokollen in Bots[2] indizieren.

[2]Internet Bot, auch bekannt als Web Robot, ist eine Softwareanwendung, die benutzt wird, um Webres-

Web-Scraping kann mit aller Art von Webseiten verwendet werden. Dies geschieht bspw. bei Wettervorhersagen, bei der Immobilienwirtschaft, beim E-Commerce und besonders, was diese Arbeit betrifft, im Reisebusiness, bei dem Preise, Reisezeiten und andere Daten aus verschiedenen Fluggesellschaften, dem Bahnverkehr oder Hotels gesammelt und angezeigt werden.

Heutzutage gibt es viele einfache und kostenlose Werkzeuge, die die aufwendige Arbeit automatisiert: beispielsweise Kimono, ScreenScraping und Metascraper. Das Selenium-Werkzeug, das in dieser Arbeit verwendet wird, ist in der Lage, jede beliebige webbasierte Aufgabe zu automatisieren und besonders die Webformulare automatisch auszufüllen. Dabei unterscheidet es sich kaum vom menschlichen Verhalten beim Surfen im Browser.

Ein wichtiges Merkmal von Selenium ist die WebDriver API, die eine bessere Unterstützung für dynamische Webseiten bietet, auf denen sich Elemente der Seite ändern können, ohne die Seite vollständig neu zu laden. Der Selenium-WebDriver ruft den Browser direkt auf, von dem abhängt, welche Eigenschaften der WebDriver unterstützt. Der einfachste Weg, Selenium in einem Java Projekt zu verwenden, ist durch Maven. Trägt man die Abhängigkeiten von Selenium in der pom.xml ein, werden diese von Maven verwaltet (siehe den Anhang für weitere Informationen über Selenium-WebDriver API Befehle Tabelle E.10 und Tabelle E.11).

2.6. Sprachdialog-Systeme

Sprachdialog-Systeme bieten eine Schnittstelle zwischen dem Benutzer und einer Computer basierten Anwendung, die Sprachinteraktion in einer natürlichen Weise erlaubt. Sie imitieren die menschliche Leistungsfähigkeit im Bezug auf Wahrnehmen, Verstehen, Schlussfolgern, Erzeugen und Emittieren der Sprache. Jedes Sprachdialog-System enthält die folgenden Phasen (siehe Abbildung 2.6):

- Automatische Spracherkennung (ASR): ASR konvertiert Sprache zu Text.
- Sprachverstehen (NLU): NLU führt die semantische Interpretation der geschriebenen Texte durch.
- Dialog-Manager (DM) und Datenbank (DB): In dieser Phase werden die geeigneten Antworten für die Benutzereingabe durch DM und DB über einen organisierten Plan erstellt, weil die Datenbank die Welt- und Dämoneninformationen enthält.
- Natürliche Sprache-Generation (NLG): Hier wird eine geeignete Formulierung für die Ausgabesemantik generiert.
- Text-in-Sprache (TTS): Durch TTS wird die Antwort in Form von Sprache ausgegeben.

Der Dialog kann Zielorientiert sein, dabei wird eine bestimmte Aufgabe ausgeführt, wie z.B. Sprachdialog-Systeme mit verschieden Zielen im Auto (Navigation, Parken usw.), oder nicht Zielorientiert. Hier führt das System ein intelligentes Gespräch mit dem Nutzer, wie z.B das Chat System ALICE [Wal09]. Eine andere Art des Dialogs ist der soziale Dialog. Er konzentriert sich auf das Verhalten oder die Meinungen der am Dialog Beteiligten, in einer natürlichen und interaktiven Art und Weise. Ein Beispiel hierfür ist das System SimSensei [DAB+14], ein Psychotherapeut, der depressive Menschen ermutigt, über ihre Probleme zu sprechen.

sourcen zu gewinnen und zu sammeln. Dies passiert mittels der Protokolle in der Datei robots.txt auf der Serverseite [SC01], [GKD07].

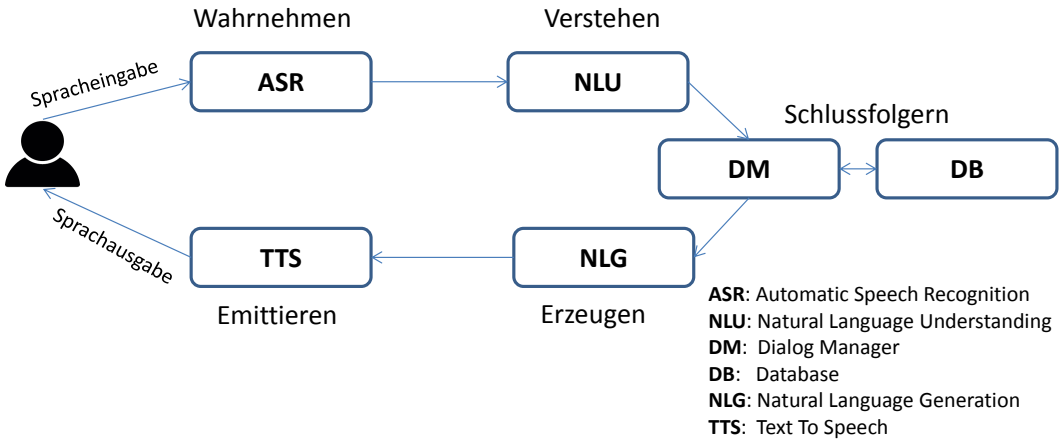

Abbildung 2.6.: Phasen des Sprachdialog-Systems [McT02]

2.6.1. Arten des Dialog-Managers

Der Dialog-Manager Systeme können in drei Hauptkategorien eingeteilt werden, abhängig von den Verfahren, die verwendet werden, um den Dialog mit dem Nutzer zu kontrollieren:

- Systeme mit endlichen Automaten
- Frame-basierte Systeme
- Agenten-basierte Systeme

2.6.1.1. Systeme mit endlichen Automaten

In Systeme, die auf endlichen Automat basieren, auf englisch „Finite-state based systems“, führt der Nutzer einen Dialog, der aus einer Sequenz von vordefinierten Dialogzuständen mit Übergängen besteht. Das System erzeugt Eingabeaufforderungen in jedem Dialogzustand, akzeptiert oder weist spezielle Ausdrücke als Antwort auf die Eingabe zurück, und erzeugt Aktionen anhand der erkannten Antwort wie z.B. in Abbildung 2.7. In diesem Beispiel wird ein Dialog zwischen einem Buchungssystem und eine Benutzer dargestellt. Das Buchungssystem ist am Anfang im ersten Zustand und zwar (Ziel der Reise). Es fragt den Nutzer nach seinem Ziel und wenn es die Antwort des Benutzers erkennt, mach es eine Übergang zum nächsten Zustand (Tag der Reise). Falls das System die Benutzerantwort nicht versteht, wiederholt es die Frage und macht keine Übergang bis es die Antwort erkennt.

System: What is your destination?
User: London
System: Was that London?
User: Yes
System: What day do you want to travel?
User: Friday
System: Was that Sunday
User: No
System: What day do you want to travel?

Abbildung 2.7.: Dialog in einem System mit endlichen Automaten [DBD98]

2.6.1.2. Frame-basierte Systeme

In einem Frame-basierten System beantwortet der Benutzer die Fragen so, dass diese Antworten Platzhalter in einer Vorlage ausfüllen. Ein Beispiel hierfür ist die Bereitstellung von Fahrplaninformationen durch das Busfahrplaninformationssystem. In diesem System ist der Dialogfluss nicht vorherbestimmt, sondern ist abhängig von der Benutzereingabe und den Informationen, die das System bereitstellt. Die Abbildung 2.8 zeigt ein Beispiel für eine Dialog in einem Frame-basierten System.

System: What is your destination?
User: London on Friday around 10 in the morning
System: I have the following connection

Abbildung 2.8.: Dialog in einem Frame-basierten System [AOSS95]

Frame-basierte Systeme sind sehr flexibel, da der Benutzer mehr Informationen eingeben kann als vom System gefordert. Allerdings sind sie ungeeignet wenn eine komplexe Interpretation des Dialogs erforderlich ist. Das ist bei der Systeme mit einer komplexen Informationsdarstellung, die Empfehlungen oder Vergleiche liefern müssen, der Fall ist. Ein Beispiel hierfür ist das System MATCH (Multimodal Access To City Help) in der Arbeit von (Johnston et al., 2002) [JBV+02]. Das System liefert nach der Benutzeranfrage Auskünfte über Restaurants in New York City.

Die Fragen, die das System stellen kann, stehen mit ihren Voraussetzungen in einer Liste. Einige Fragen für ein Reisesystem könnten wie in Abbildung 2.9 lauten:

condition: unknown(origin) AND unknown(destination)
question: Which route do you want to travel?

condition: unknown(origin)
question: Where do you want to travel from?

condition: unknown(destination)
question: Where do you want to travel to?

Abbildung 2.9.: Fragen für ein Reisesystem in einem Frame-basierten System [AOSS95]

2.6.1.3. Agenten-basierte Systeme

Agenten-basierte Systeme erlauben komplexe Kommunikationen zwischen dem System, den Benutzern und der zugrundeliegende Anwendung. Agenten-basierte Systeme berücksichtigen die vorhergehenden Kontexte für die Entwicklung des Resultats so, dass sie dynamisch eine Sequenz miteinander verbundener Schritte bilden. Sie beinhalten außerdem Mechanismen für Fehlererkennung und -korrektur und versuchen die Nutzereingabe vorherzusagen und zu interpretieren. Deswegen können diese Systeme als mixed-initiative Systeme beschrieben werden. Das heißt, der Nutzer kann immer die Kontrolle über den Dialog übernehmen, neue Themen vorschlagen und Beiträge einbringen, die nicht durch Systemaufforderungen beschränkt sind.

2.6.2. Statistische Dialogsysteme für gesprochene Sprache

In den statistischen Methoden werden die Entscheidungen unter Unsicherheit getroffen. Dies ist in Dialogsystemen genau dann der Fall, wenn es in der Umgebung Lärm gibt, Fehler

beim Spracherkenner stattfinden oder wenn der Nutzer seine Intention nicht komplett bekannt gibt. Die Nutzerintention kann in diesen Fällen nur unvollständig erkannt werden. Daher ergibt sich hier die Notwendigkeit, die Dialogabläufe automatisch zu lernen.

Die Vorteile der statistischen Methode liegen darin, dass die Aktionen zwischen den Dialogzuständen durch einen Optimierungsprozess ausgewählt werden. Diese Technik ist domänenunabhängig; das heißt, sie kann bei vielen Domänen problemlos angewandt werden.

Verschiedene statistische Methoden können für die Erstellung eines Dialog-Managers implementiert werden, beispielsweise Bestärkendes Lernen auf englisch Reinforcement Lernen(RL), Markow-Entscheidungsproblem auf englisch Markov Decision Processes (MDPs) und partiell beobachtbare Markow-Entscheidungsproblem auf englisch Partially Observable Markov Decision Processes (POMDP).

2.6.2.1. Bestärkendes Lernen

Reinforcement Lernen ist die Versuchs-und-Irrtumsmethode zum Lernen. Dabei werden die Lösungsmöglichkeiten solange wiederholt probiert, bis das gewünschte Ziel erreicht wird oder das System die Versuche beendet. Im Dialog-Manager berechnet das System die kumulierte Belohnung jeder Lösung (Der Profit, der aus jeder Lösung gewonnen werden kann) und lernt, wie die Aktionen zwischen den Dialogzustände ausgewählt werden müssen, um die kumulierte Belohnung zu maximieren.

2.6.2.2. Markow-Entscheidungsprozess

Markov decision processes (MDPs) sind ein Formalismus zur Modellierung der Entscheidungsfindung unter Anwendung der Markov-Bedingung, die besagt, dass keine Abhängigkeit von der Vergangenheit besteht.

In der Domäne des Dialog-Managers wird der Dialog als „MDP“ modelliert, dann wird Reinforcement Lernen verwendet, um die optimalen Dialogabläufe zu finden.

2.6.2.3. Partiell beobachtbare Markow-Entscheidungsproblem

Partially Observable Markov Decision Processes (POMDPs) setzen das Reinforcement Lernen mit dem Vertrauenszustand zusammen. Darum hat jeder Zustand *st* einen Vertrauenszustand *b(st)*, die die Wahrscheinlichkeit des Zustands beschreibt. POMDP kann mit dem Nutzer-Simulations-Ansatz implementiert werden. Die Nutzer-Simulation erlaubt dem System mit einem Dialogkorpus und durch Beispiele zu lernen. Das System hat infolgedessen die Antworten realer Nutzer.

2.7. Zusammenfassung

In diesem Kapitel wurden die für die vorliegende Arbeit relevanten Grundlagen und Technologien eingeführt. Dabei wurde das Basiswissen der aktiven Ontologien, Webformulare, Sprachdialog-Systeme und Web-Scraping im Einzelnen erklärt, was essentiell und notwendig zum Verständnis der weiteren Arbeit ist.

3. Verwandte Arbeiten

Eine gründliche Recherche ergab keine vergleichbare Arbeit zur Abbildung von Webformularen auf aktive Ontologien gefunden werden. Allerdings gibt es viele Untersuchungen, die sich mit der Abbildung von verschiedenen Ressourcen einschließlich Webformularen auf konventionelle Ontologien beschäftigen. Daher werden in diesem Kapitel Arbeiten vorgestellt und verglichen, die sich mit den folgenden Bereichen beschäftigen:

- Aktive Ontologien
- Die Abbildung von verschiedenen Datenstrukturen und besonders von Webformularen auf konventionelle Ontologien
- Herangehensweisen zum Entwerfen eines Dialog-Managers.

Es werden die Stärken und Schwächen der Ansätze analysiert, um daraus Erkenntnisse für die eigene Konzeption zu gewinnen.

3.1. Aktive Ontologien

Aktive Ontologien wurden von Guzzoni in seiner Dissertation [Guz08] entwickelt und implementiert. Eine Vorstellung von aktiven Ontologien und von der „active-Plattform“ wurde in den Arbeiten [GCB06], [GBC07] und [GBC06] auch von Guzzoni präsentiert. Dann wurde die aktive Ontologie als Patent [CG14] von Didier Guzzoni und Adam Cheyer für Apple Inc. registriert und vermutlich im intelligenten Assistenten von Apple „Siri“ eingesetzt. Guzzoni hat die aktive Ontologie in vielen Domänen verwendet, wie z.B. in der Sprachverarbeitung, Service-Brokering und Workflow-Programmierung. Aber er hat sich mit den Webformularen nicht beschäftigt, daher wird in dieser Arbeit untersucht, wie Webformulare auf aktive Ontologien abgebildet werden können. Die Grundlagen der aktiven Ontologie von Guzzoni werden in Abschnitt 2.2 präsentiert.

3.2. Automatische Erstellung von konventionellen Ontologien

Ontologieerstellung ist eine schwierige, zeitaufwändige und fehleranfällige Aufgabe. Deswegen ist eine automatische Erstellung einer maschinenlesbareren Repräsentation des menschlichen Wissens (Ontologien) wünschenswert. Eine vollautomatische Erzeugung von Ontologien aus verschiedenen Resourcen ohne menschliche Überprüpfung ist in der Forschung aktuell eine große Herausforderung. In dem folgenden Abschnitt werden Arbeiten über die Abbildung von XML-Dateien, Datenbanken und Tabellen auf Ontologien vorgestellt.

3.2.1. Abbildung von XML-Dateien und Datenbanken

In den Veröffentlichungen *„Constructing complex semantic mappings between XML data and ontologies“* [ABM05a], *„Discovering and maintaining semantic mappings between XML schemas and ontologies“* [ABM08] (beide von Y. An et al.) und *„Translating XML web data into ontologies“* [AM05] von Y. An und J. Mylopoulos wurden verschiedene Werkzeuge zur Abbildung von XML-Dateien auf Ontologien entwickelt. Die Eingaben in diesen Werkzeugen sind eine XML-Datei, eine Ontologie und vereinfachte Übereinstimmungen zwischen der Ontologie und der XML-Datei. Dann werden semantische Abbildungen von XML-Dateien auf Ontologien durch heuristische Algorithmen als Ausgabe erzeugt.

Mit Abbildungen von Datenbanken auf Ontologien beschäftigen sich auch die Arbeiten *„Maintaining semantic mappings between database schemas and ontologies“* von Yuan An [AT08] und *„Building semantic mappings from databases to ontologies“* von Y. An et al. [AMB06]. Es werden Korrespondenzen erzeugt zwischen dem Datenbankschema (z.B. Spaltennamen in den Tabellen) und den Ontologieelementen (wie z.B. Attribute und Konzepte). Diese Abbildungen werden anschließend durch verschiedene Algorithmen verfeinert und evaluiert. Die Abbildung, die die größte Übereinstimmung hat, wird dann ausgewählt.

Tabellen der Datenbanken werden auch auf Ontologien abgebildet, wie in den Arbeiten von Y. An et al. mit dem Titel *„Inferring complex semantic mappings between relational tables and ontologies from simple correspondences“* [ABM05b] und *„Refining Semantic Mappings from Relational Tables to Ontologies“* [ABM04] ausgeführt. Hier werden die Primär- und Fremdschlüssel der Tabellen sowie Kardinalitätbeschränkungen und die „ist-ein“-Beziehungen der Ontologien verwendet, um die Abbildungen zu erstellen.

3.2.2. Analyse der Webformulare

Viele Arbeiten haben sich mit der Modellierung und dem Verstehen der Webformulare beschäftigt, um semantische Informationen daraus zu extrahieren. Die Webformulare wurden dabei aus verschiedenen Perspektiven analysiert.

In der Arbeit von B. He et al. mit dem Titel *„Statistical schema matching across web query interfaces“* [HC03] wird eine statistische Übereinstimmung erforscht, die auf der Generierung neuer Hypothesen basiert, um gleiche Formularelemente mit verschiedenen Namen zu vereinigen wie z.B „author“, „writer“ und „name“ oder „category“ und „subject“. Hier wird das Webformular als ein Satz von Attributen betrachtet.

Die Arbeiten *„An interactive clustering-based approach to integrating source query interfaces on the deep web“* [WYDM04] von W. Wu et al. und *„A hierarchical approach to model web query interfaces for web source integration“* [DKYL09] von E. Dragut et al. analysieren das Formular als eine hierarchische Struktur. Einige generelle Gestaltungsregeln der HTML-Seiten werden hier ausgenutzt, um Bäume, welche die Formulare darstellen, zu extrahieren. Mithilfe dieser Ansätze werden z.B. „Departure Date“ und „when do you want to go“ als nur ein Konzept betrachtet.

Die optische Gestaltung der Webformulare wird ebenso analysiert und genutzt wie z.B. in den Arbeiten von H. He et al. mit dem Titel *„Towards Deeper Understanding of the Search Interfaces of the Deep Web“* [HML+07] und *„Automatic integration of Web search interfaces with WISE-Integrator“* [HMYW04]. Dabei werden die Labels und andere Steuerungselemente des Webformulars untersucht, um das Webformular in eine bestimmte Kategorie zu überführen.

3.2.3. Automatische Erstellung von konventionellen Ontologien aus den Webformularen

Webseiten bieten dem Nutzer eine Umgebung, um domänenspezifische Begriffe zu erlernen. Die Extraktion der Ontologie mit den meist verwendeten Webseitkonzepten kann

wichtige Informationen und Eigenschaften über die Domäne darstellen sowie Daten zum Nutzerverständnis der Domäne sammeln.

Die Arbeiten, bei denen Ontologien aus Webformularen entwickelt wurden, erstellen die Ontologie von Grunde auf durch die Verwendung der DOM-Struktur der Webseiten mithilfe von Methoden der künstlichen Intelligenz. Beispielsweise extrahiert „DeepMiner“ von W. Wu et al. [WDYM05] die Konzepte der Ontologie durch die Analyse der Position der Formularelemente in dem DOM-Baum. Dann werden die neuen Konzepte durch die Verwendung eines Clusteringalgorithmus herausgefunden. OntoMiner von H. Davalcu et al. [DVNR03] findet die Labels und ordnet die entsprechenden Kategorien zu durch die Verwendung eines Taxonomy-mining-Algorithmus, der eine „ist ein“-Beziehung zwischen Labels und Kategorien findet. Die Attribute werden mit deren Werten extrahiert.

„OntoBuilder“ von H. Roitman und A. Gal [RG06] ist ein Pionierprojekt in der Ontologieextraktiondomäne, das die Webformulare verschiedener Webseiten analysiert und alle Formularelement und deren Labels durch das Parsen der HTML-Seite identifiziert. Anschließend bildet dieses Werkzeug anfängliche Versionen einer globalen Ontologie und lokaler Ontologien (Kandidaten). Diese Ontologien werden iterativ abgebildet, um eine verfeinerte globale Ontologie zu erstellen. Daraufhin werden die Konzepte bzw. Eingabefelder des Webformulars durch ihre Datentypen, die Einschränkungen auf Datenzuweisung und ihre Anordnung innerhalb des Webformulars abgebildet. Ontobuilder erkennt die zwei Beziehungen zwischen Konzepten, und zwar die Komposition und die Präzedenz, die in dem Übereinstimmungsalgorithmus verwendet werden. Die Labels und Feldernamen werden durch Stringübereinstimmung verglichen, während die Werte wie Drop-down-Listen und Radio Buttons eine automatische Maßnahme für die Ähnlichkeit zwischen Domäne verwendet. Die Verwendung der Präzedenz kann das Problem von schlechtem Entwurf oder Entwurffehler wie z.B. wenn „dep-time-1“ und „dep-time-2“ als Namen von „Departure Time“ und „Return Time“ verwendet werden. Hier wird durch die Präzedenzbeziehung „dep-time-2“ als „Return Time“ erfolgreich erkannt.

In der Arbeit *„Semantic deep web: automatic attribute extraction from the deep web data sources“* von Y. An et al. [AGWC07a] wird die Ontologie automatisch durch die Extraktion der Webseitenattribute erstellt. Diese Attribute werden in 2 Sätzen angeordnet.

- Programmer Viewpoint Attributes (PVAs): Diese Attribute werden aus den Attribut-Value-Paaren der HTML-Tags extrahiert, wie z.B. die Attribute (name) und (id).
- User Viewpoint Attributes (UVAs): Diese Attribute werden aus der Analyse der Texte in Webformularen bzw. Texteingabe-Bereichen gesammelt wie z.B. der Text zwischen den Tags <*OPTION*> und </*OPTION*> einer Drop-down-Liste.

Der finale Satz der Attribute wird durch die Verwendung von PVAs und UVAs und deren Synonymen in WordNet festgestellt. Hier wird eine Ausnahme für die Präpositionen berücksichtigt. Ansonsten werden z.B. „from“ und „to“ automatisch verworfen, da sie keine Nomenbedeutung in WordNet haben. Genauso ist es der Fall mit den Abkürzungen wie z.B. „dep“ und „yr“, die häufig für „departure“ und „year“ benutzt werden. Außerdem werden viele Varianten von Wörtern in Singular oder Plural wie „babies“, „infants“, „kind“, „kinder“, „child“, „children“ beibehalten.

Die Arbeit *„Automatic Generation of Ontology from the Deep Web“* von Yoo J. et al. [AGWC07b] erstellt eine domänenspezifische Ontologie durch die Verflechtung von Konzepten und Beziehungen von WordNet wie z.B. Hypernym mit Informationen, die aus domänenspezifischen Webseiten (Fluggesellschaften, Hotels usw.) extrahiert werden. In diesem Ansatz werden „Leaving from“ und „City Name“ Synonyme betrachtet, da sie analoge Felder beschreiben.

FAETON von R. Berlanga et al. [BJNS10] ist ein halbautomatisches Werkzeug für die Erstellung von Ontologien aus Datenerfassungsformularen. Dabei werden die optischen und geometrischen Eigenschaften der Formularlabels und -felder analysiert, um ein logisches Modell, Gestaltungsmodell und Linkmodell zu extrahieren. Das logische Modell beinhaltet die Struktur der Elemente im Formular, während das Gestaltungsmodell die optischen Eigenschaften (Farbe, Schriftart, Ausrichten) und das Linkmodell den Workflow des Formulars umfassen. Die Ontologie wird dann durch die Verwendung von Heuristiken erstellt.

Andere Ansätze verwenden maschinelles Lernen für die Abbildung von Webformularen auf die Ontologie wie z.B. in der Arbeit *„Learning to Discover Complex Mappings from Web Forms to Ontologies“* von Y. An et al. [AHS12], welche sich mit dem Thema des automatischen Findens von komplexen Abbildungen von Webformularen auf Ontologien beschäftigt. Das Webformular wird durch das Parsen von dessen Quellcode zu einem DOM-Baum konvertiert. Im nächsten Schritt wird eine einfache Abbildung zwischen den Elementen des Webformularbaums und Elementen der Ontologie erstellt. Ein Bayes-Klassifikator wird nun durchgeführt, um den wahrscheinlichsten Zusammenhang in der Ontologie angesichts des Formularbaums zu finden, und zwar den Subgraph im Ontologiegraph mit der größten Wahrscheinlichkeit. Im Vergleich zu anderen Ansätzen, welche dem Nutzer die Wahl des Kandidaten überlassen, wählt dieser Ansatz den besten Kandidaten selbst aus.

3.2.4. Diskussion

In den vorher vorgestellten Arbeiten beinhalten die Eingaben eine Ontologie und eine einfache Abbildung, die als Basis verwendet werden. Diese Abbildung wird mithilfe verschiedener Algorithmen weiterentwickelt, um mehrere Abbildungskandidaten mit jeweils einem Ranking zu erzeugen. Im Anschluss daran werden viele Kandidaten vom Werkzeug vorgeschlagen.

Alle diese Werkzeuge beschäftigen sich mit einer relativ kleinen Anzahl von Webformularen. Wenn diese Werkzeuge in einer Maschine als „Blackbox“ eingesetzt werden, dann kann dies aus zwei Gründen problematisch sein. Zum einen, weil es für den Benutzer sehr aufwändig ist, zwischen allen Kandidaten auszuwählen. Zum andern ist es nicht möglich, vorherzusagen, welche Zeit das Werkzeug benötigt, um die Ontologie zu erstellen.

3.3. Dialog-Manager

Sprachdialog-Systeme und besonders Dialog-Manager sind seit mehr als 40 Jahren ein Forschungsgegenstand. Durch die fortschreitende Entwicklung in der Sprachtechnologie und -verarbeitung sowie der Dialogmodellierung wurden sie für Wissenschaft und kommerzielle Zwecke immer interessanter.

In der Arbeit von L. Dybkjer et al. mit dem Titel *„A methodology for diagnostic evaluation of spoken human machine dialogue“* [DBD98] wurde ein System mit einem endlichen Automat zur Flugreservierung entwickelt. Der Automat besteht aus acht Hauptzuständen und zwar: Die Anzahl der Personen, Abflugort, Ankunftsort, Hin- und Rückflug, Datum des Flugs, Uhrzeit des Flugs, Datum des Rückflugs und Uhrzeit des Rückflugs. Das System stellt die Fragen in dieser Reihenfolge und die Antwort des Benutzers besteht aus maximal 10 Wörtern.

Ein ähnliches System zur Busfahrplaninformation wurde in der Arbeit von C. Bennett et al. [BFS+02] präsentiert. Das System fragt zuerst nach dem Standort, dann nach der Richtung und die letzte Frage ist die nach der Route.

Flugreservierung und Busfahrplaninformation in den beiden Systemen sind einfache und gut strukturierte Aufgaben, die in eindeutig definierte Teilaufgaben zergliedert werden

können und bei denen die Eingabe erforderlicher Daten nacheinander erfolgt. Systeme, die auf endlichen Automaten basieren, sind geeignet für solche Aufgaben, daher wurden die beiden Systeme als erfolgreiche Systeme betrachtet und viele kommerzielle Sprachdialogsysteme wurden sodann nach dieser Art des Dialogs entworfen.

In den beiden Systemen muss der Benutzer jedoch eine strikte Reihenfolge einhalten. Idealerweise soll die Antwort des Benutzers kurz, in einzelnen Wörtern oder einfachen Ausdrücken sein. Diese Systeme haben eine klare Semantik. Der Nachteil dieser Systeme besteht darin, dass sie unflexibel sind. Es ist schwierig, das System so zu entwickeln, dass der Nutzer vorangegangene Eingaben korrigieren oder zu anderen Teilaufgaben wechseln kann, so die Arbeit von M. McTear [McT02].

In der Arbeit *„The Philips automatic train timetable information system“* von H. Aust et al. [AOSS95] wurde das Philips-Zugfahrplaninformationsystem mit einem Frame-basierten Dialogsystem entwickelt. Dabei erhält der Benutzer telefonisch Informationen über Zugverbindungen zwischen ungefähr 1200 deutschen Städten. In diesem System kann der Benutzer mehr als die geforderten Informationen eingeben. Das System kann diese Informationen akzeptieren und überprüfen, ob zusätzliche Informationen notwendig sind, bevor es anschließend in der Datenbank nach einer Verbindung sucht. Allerdings können die Nutzer ihre Antworten nicht näher ausführen oder im Nachhinein korrigieren, so kann der Benutzer zum Beispiel die Ziele während des Abfrageprozesses nicht mehr ändern.

In der Arbeit von D. Mori und S. Sadek mit dem Titel *„Spoken Dialogues with Computers“* [Mor97] wurde ein kooperativer Dialog mit dem Benutzer durchgeführt, der als Agenten-basiertes System klassifiziert werden kann. Die Abbildung 3.1 ist ein Dialogbeispiel aus dieser Arbeit, in dem die Antwort des Systems negativ ist. Das System versucht jedoch statt eines einfachen „Nein“ eine kooperativere Antwort zu geben.

User: I am looking for a job in the Calais area. Are there any servers?
System: No, there are not any employment servers for Calais. However, there is an employment server for Pas-de Calais and an employment sever for Lille. Are you interested in one of these?

Abbildung 3.1.: Dialog in einem Agenten-basierten System. Das System gibt dem Benutzer Vorschläge, wenn es keine Ergebnisse für seine Anfrage findet [Mor97]

Eine große Auswahl Agenten-basierter Systeme wird bereits verwendet, jeweils mit unterschiedlichen Ansätzen zur Implementierung des Systems. Es gibt jedoch keine umfassende Studie über die Kosten und den Nutzen Agenten-basierter Systeme, gemäß der Arbeit von M. McTear [McT02].

In der Arbeit von C. Mitchell et al. mit dem Titel *„Toward a Markov Decision Process Dialogue Policy for Computer Science Tutoring“* wurde ein Dialog entwickelt, der mit dem Markow-Entscheidungsprozess modelliert wurde. Ein Beispiel dafür wird in der Abbildung 3.2 dargestellt. Das Ziel in diesem Dialog ist es, die Küche zu erreichen. Dabei werden drei Zustände definiert. „Wissen nichts“, „Ziel ist“ und „Ziel erreicht“. Diese Zustände beschreiben den Fortschritt im Dialogablauf. „Wissen nichts“ repräsentiert, dass das Ziel des Dialogs noch nicht determiniert ist, deswegen muss eine Aktion durchgeführt bzw. nach Anweisungen gefragt werden. Ist das Ziel „Küche“ determiniert, wird der Zustand zu „Ziel ist Küche“ geändert und die Aktion „Geh in die Küche“ ausgeführt. Wenn das Ziel erreicht wird, wird der Zustand zu „Ziel erreicht“ geändert.

Die Abbildung 3.3 aus der Arbeit von J. Williams und S. Young mit dem Titel *„Partially Observable Markov Decision Processes for Spoken Dialog Systems“* [WY07] zeigt einen

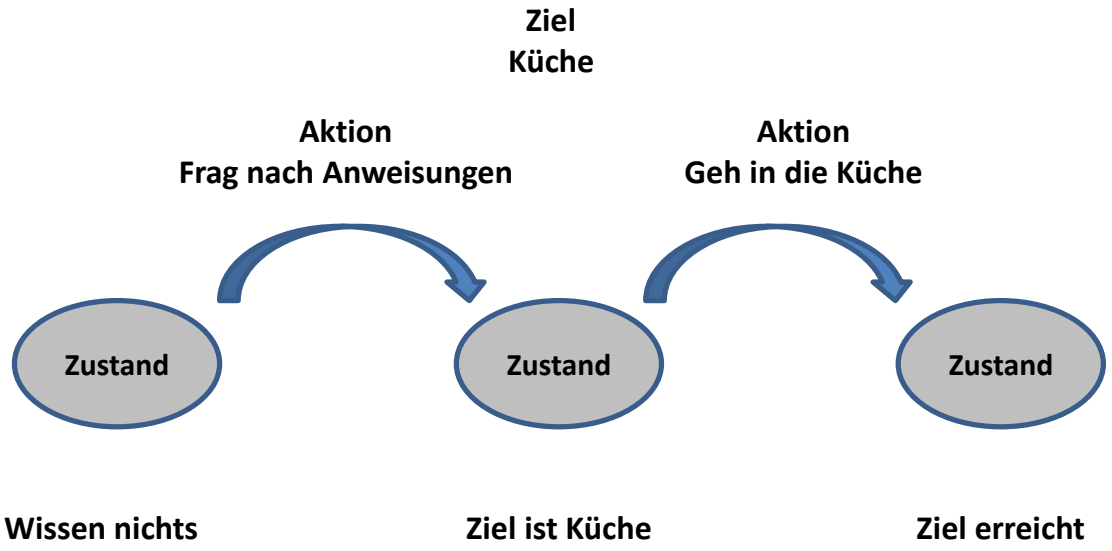

Abbildung 3.2.: Dialog durch Markow-Entscheidungsprozess [MBL13]

Dialog, der nach dem partiell beobachtbaren Markow-Entscheidungsproblem und mit Vertrauenszuständen modelliert wurde. Die Wahrscheinlichkeit ist am Anfang zwischen den Zuständen gleich verteilt, das heißt, dass alle Zustände „small“, „medium“ und „large“ die gleiche Wahrscheinlichkeit haben. Nach der Antwort des Nutzers verändert sich diese entsprechend.

3.3.1. Diskussion

Statistische Methoden sind besonders gut, wenn viele Trainingsbeispiele zur Verfügung stehen. In dieser Arbeit ist das Domänenwissen schon definiert und der Dialog-Manager wird erst aufgerufen, wenn der Nutzer vergessen hat, notwendige Informationen einzugeben oder das System dem Benutzer weitere Informationen anbietet. Weiterhin erwartet das System bestimmte Antworten für bestimmte Fragen, da es Platzhalter hat, die ausgefüllt werden müssen. Aus diesem Grund ist die Nutzung statistischer Dialogsysteme in diesem Fall nicht geeignet, da sie viel Aufwand erfordern.

3.4. Zusammenfassung

In diesem Kapitel wurde ein Überblick an Forschungsergebnissen in den Bereichen der aktiven Ontologien und der automatischen Erstellung von konventionellen Ontologien mit Fokus auf der Extraktion der Ontologien aus den Webformularen und dem Dialog-Manager präsentiert. Die Arbeiten wurden im Einzelnen vorgestellt und bewertet. Im folgenden Kapitel wird die Aufgabenstellung analysiert. Dabei werden die entsprechende Lösungen detailliert beschrieben.

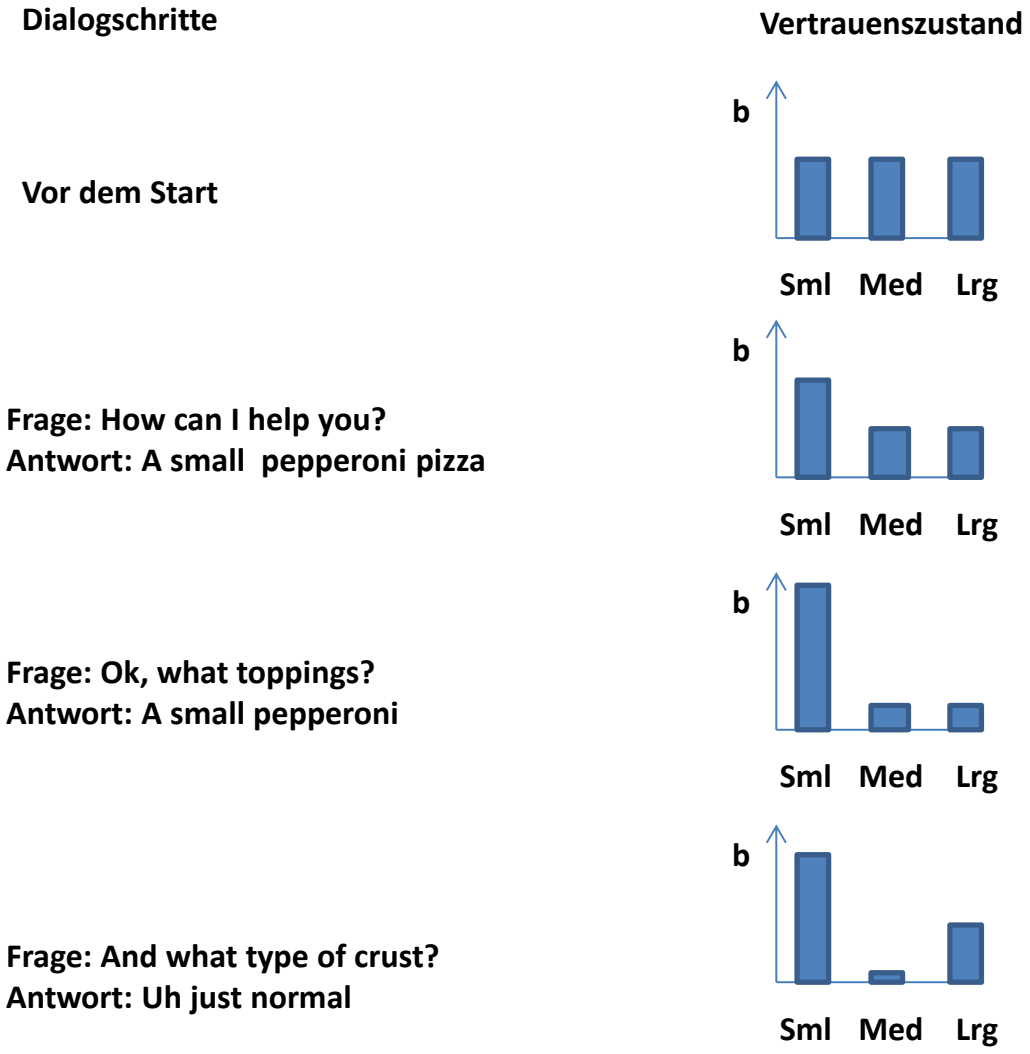

Abbildung 3.3.: Dialog durch POMDP mit mit den Belief states [WY07]

4. Analyse

Das Ziel dieser Arbeit ist es, die Webformulare einer bestimmten Domäne in einer einzigen umfassenden Methode aufzurufen. Dies kann erreicht werden, durch die Abbildung von Feldern der Webformulare auf Knoten einer aktiven Ontologie, zur Ermöglichung der intelligenten Assistenten zum Umgang mit der Flug-, Bahn- und Hotelbuchung. In diesem Kapitel werden die einzelnen Teilaufgaben im Detail analysiert. Dabei liegt der Fokus zum einen auf der Analyse der Webformulare von Flug-, Bahn- und Hotelbuchungen mit dem Ziel der Erstellung einzelner Formulare, welche die jeweilige Kategorie komplett darstellen. Zum anderen auf der Definition von aktiven Ontologien. Damit ist gemeint, Konzepte, Beziehungen und geeignete Techniken auszuwählen, um die benötigten Informationen aus den sprachlichen Eingaben der Nutzer herauszufiltern. Ein anderer Schwerpunkt dieser Arbeit liegt in dem Entwerfen eines Dialog-Managers. Damit kann das System in interaktiver Art und Weise mit dem Nutzer kommunizieren. Mithilfe des Dialog-Managers werden Informationen vom Nutzer gesammelt, die für die Suche in den Webformularen der Buchungswebseiten benötigt werden. Dabei entsteht eine andere Aufgabe, nämlich das Abschicken der Informationen aus der Nutzeranfrage in den Webformularen der Buchungswebseiten und die Sammlung der Suchergebnisse. Der Lösungsansatz zu dieser Aufgabenstellung wird mittels Web-Scraping durchgeführt. Im Folgenden werden diese Aufgaben detailliert untersucht.

Ausgehend von dem oben genannten Ziel soll mit der hier vorliegenden Arbeit eine Technik entwickelt werden, die die verschiedenen Webformulare der Flug-, Bahn- und Hotelweb-

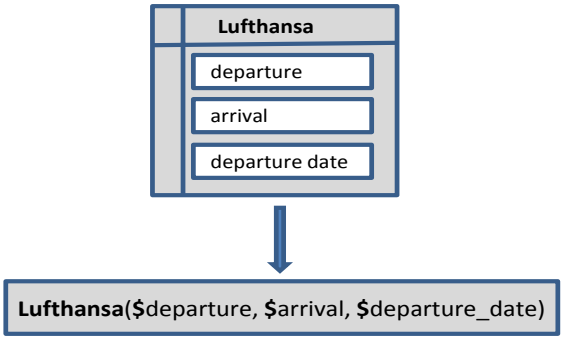

Abbildung 4.1.: Webformular als Methodenaufruf darstellen

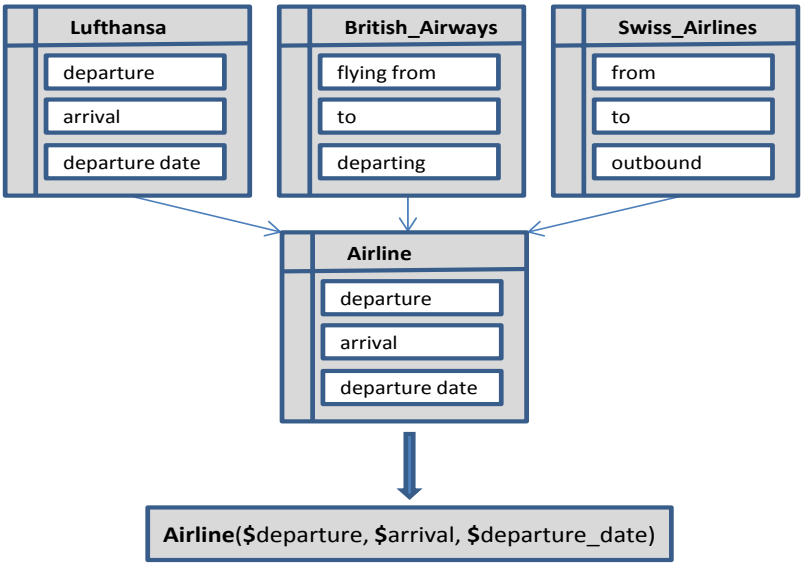

Abbildung 4.2.: Verschiedene ähnliche Webformulare in Domänenformular vereinigen und dieses als Methodenaufruf darstellen

seiten automatisch ausfüllen kann. Anders ausgedrückt, es muss eine Methode definiert werden, die ein Webformular ausfüllt und abschickt. Die Felder des Formulars müssen als Argumente der Methode repräsentiert werden.

Die Abbildung 4.1 zeigt, wie die Felder des Formulars „Lufthansa“ als Methodenaufruf aussehen könnten. Die hier definierte Methode hat einen Namen, der den Name der Fluggesellschaft repräsentiert, und Parameter, die Formularfelder darstellen. Die Namen der Parameter beginnen mit „$“.

Das ist der Fall, wenn nur ein Formular zur Verfügung steht. In der Praxis kommt es jedoch häufig vor, dass viele ähnliche Formulare in einer bestimmten Domäne aufgerufen werden sollen, wie z.B. Flugdomäne, um deren Ergebnisse zu sammeln und zu vergleichen. Die Webformulare einer Domäne stellen die Konzepte dieser Domäne dar, aber mit unterschiedliche Konzeptnamen. Ein Beispiel hierfür sind die Felder „departure city“ und „flying from“, die zwei verschiedenen Fluggesellschaftsformularen gehören und das gleiche Konzept „departure“ repräsentieren, jedoch mit zwei verschiedenen Namen. Nun besteht die Notwendigkeit, dass die Konzepte mit unterschiedlichen Namen in nur einem Konzept vereinigt werden. So können alle Formulare in nur einem Methodenaufruf ausgefüllt werden. Dadurch werden beispielsweise die Konzepte „departure city“, „flying from“ und „from“ von den Formularen „Lufthansa“, „British Airways“ und „Swiss Airlines“ bzw. in dem Konzept „departure“ in dem Domänenformular „Airline“ sowie als Argument „departure“ im Methodenaufruf repräsentiert werden. Abbildung 4.2 verdeutlicht dies.

Der gewünschte Methodenaufruf steht genau im Wurzelknoten der aktiven Ontologie, und genau im Befehlsknoten „commandNode“ in der Ontologie. Der Knoten „command“ ist für die Ausführung bestimmter Befehle verantwortlich und beinhaltet alle notwendige Informationen für diese Ausführung. Die Informationen kommen aus den Knoten der aktiven Ontologie. Beispielsweise besteht eine aktive Ontologie für die Methode aus Abbildung 4.1 aus den Knoten „departure“, „arrival“ und „departure date“. Diese Knoten erkennen die Informationen, die in der Nutzeranfrage stehen, und geben diese Informationen zu den Elternknoten weiter und so bis zum „command“, wo die Informationen aus allen Knoten zur Verfügung stehen, um sie in einem Methodenaufruf verwenden zu können. Die Abbildung 4.3 zeigt die Knoten der vereinfachten aktiven Ontologie „Flug“, und wie die Knoten

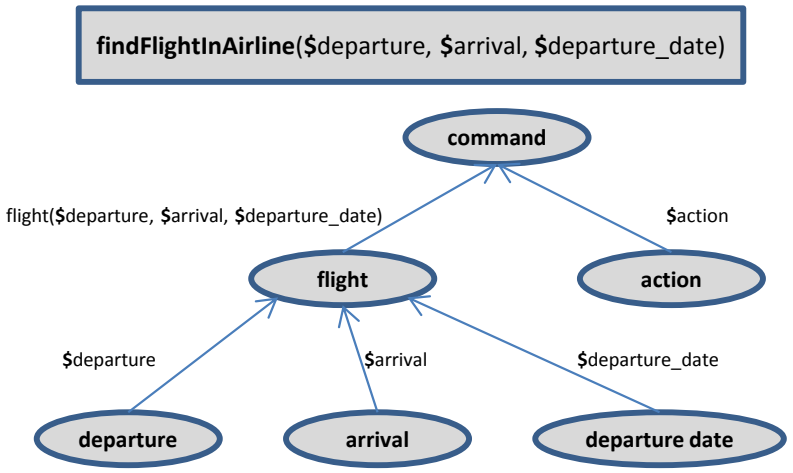

Abbildung 4.3.: Aktive Ontologie für die Suche nach Flügen: von Informationen in Knoten zu Methodenaufruf

Informationen an ihre Eltern bis zum „command“ weitergeben. Der gesamte Prozess wird in Abbildung 4.4 verdeutlicht. Dabei werden Webformulare von „Lufthansa“, „British Airways“ und „Swiss Airlines“ durch das Formular „Airline“ repräsentiert. Die Felder des Formulars „Airline“ werden auf Blätter der Flugontologie abgebildet. Die Informationen von allen Blättern werden im Knoten „command“ gesammelt und als Argumente für das Ausfüllen der oben genannten Webformulare verwendet.

4.1. Von Webformularen zu aktiven Ontologien

Viele Informationen über Flug-, Bahn- und Hotelbuchungen werden gesammelt, um diese Domänen zu modellieren. Dazu werden Buchungswebseiten untersucht und verschiedene Webformulare analysiert, um Gemeinsamkeiten und Unterschiede zwischen den ausgewählten Webformularen zu erkennen.

4.1.1. Webseite auswählen

Zehn Webseiten werden für die Kategorien Flug-, Bahn- und Hotelbuchungen jeweils ausgewählt. Diese Auswahl wird unter Verwendung verschiedener Kriterien durchgeführt. Ein Beispiel dafür ist wie die Fluggesellschaften aus verschiedenen Ländern, Typen und Bewertungen ausgewählt werden, um die größtmögliche Anzahl von Konzeptnamen und Domänenbegriffe zu sammeln. Dies wird im Einzelnen in den nächsten Abschnitten beschrieben.

4.1.1.1. Flugkategorie

Verschiedene Arten von Fluggesellschaften stehen zur Verfügung und werden in dieser Arbeit untersucht. So gibt es staatlich betriebene Fluggesellschaften oder „Flagcarrier“, was bedeutet, dass sie unter der Flagge eines bestimmten Landes unterwegs sind. Ein Beispiel hierfür ist „Air New Zealand“. Es gibt ebenfalls verschiedene Arten privater Fluggesellschaften wie die Charterfluggesellschaften, bei denen die Flüge von Reiseveranstaltern speziell für eine Reihe von Flügen gemietet und mit den Hotelunterkünften kombiniert werden können wie z.B. „Condor“. Andere Arten der privaten Fluggesellschaften sind die Linienfluggesellschaften, deren Flüge regelmäßig stattfinden wie bei „Lufthansa“, und Billig-Airlines, die zu günstigen Konditionen Flugreisen anbieten wie „Ryanair“. Die Fluggesellschaften werden auch aus verschiedenen Kontinenten ausgewählt, um kulturelle

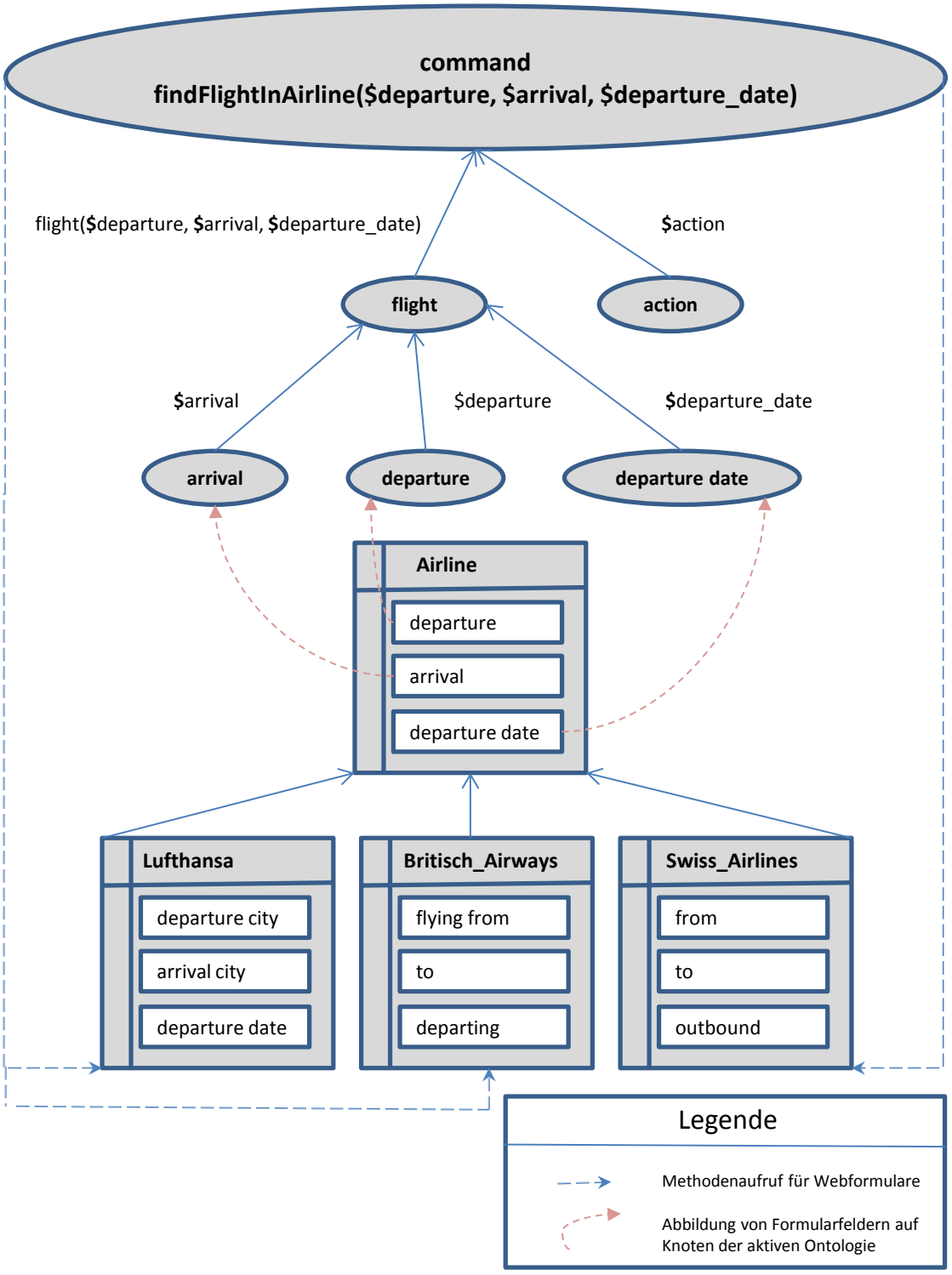

Abbildung 4.4.: Abbildung von Webformularen auf aktive Ontologie, Ausfüllen von Webformularen durch Methodenaufruf in aktiver Ontologie

Kriterien einzubeziehen. In dieser Art und Weise werden die Webseiten der Fluggesellschaften aus der ganzen Welt untersucht. Beispiele dafür sind „South African Airlines“, „Air Canada“, „British Airways“, „Air New Zealand“, „Cathay Pacific Airways“ aus China und „Emirates“ aus Nahost in Asien. Die Qualitätsbewertung [1] wurde auch bei der Auswahl der Fluggesellschaften in Betracht gezogen. Z.B. die Klassifizierung der „Cathay Pacific Airways“ ist fünf Sterne, vier Sterne für „Lufthansa“, drei für „Condor“ und zwei für „Ryanair“. Die ausgewählten Fluggesellschaften fliegen weltweit wie „Lufthansa“ oder nur kurze Strecken wie „Ryanair“. Die finalen zehn Fluggesellschaften werden in der Tabelle 4.1 mit dem Kontinent, dem Land, dem Typ und der Bewertung gelistet. Die meisten ausgewählten Fluggesellschaften sind staatliche und haben 4 Sterne, da die größte Anzahl von Reisenden mit diesen Fluggesellschaften fliegen.

Fluggesellschaft	**Kontinent**	**Land**	**Typ**	**Bewertung**
Lufthansa	Europa	Deutschland	privat	4
Air New Zealand	Neuseeland	Neuseeland	staatlich	4
Cathay Pacific Airways	Asien	China	staatlich	5
South African Airlines	Afrika	Südafrika	staatlich	4
British Airways	Europa	Großbritannien	staatlich	4
Ryanair	Europa	Irland	privat	2
Air Canada	Nordamerika	Kanada	staatlich	4
Condor	Europa	Deutschland	privat	3
Emirates	Asien	UAE	staatlich	4
Swiss Airlines	Europa	Schweiz	staatlich	4

Tabelle 4.1.: Die zehn ausgewählten Fluggesellschaften

4.1.1.2. Bahnkategorie

Die Webseiten betreffend den Bahnverkehr repräsentieren eine rein deutsche Auswahl, um die Webformulare verschiedener Dienstleister lokal oder in nur einem Land zu analysieren. Die Webseiten umfassen ganz Deutschland mit der Deutschen Bahn, eine große Stadt mit dem Verkehrsverbund Berlin-Brandenburg (VBB) oder eine Stadt mit ihrer Umgebung mit dem Karlsruher Verkehrsverbund (KVV). Die zehn Anbieter mit den dazu gehörigen Bundesländern und Regionen werden in Tabelle 4.2 gelistet.

Anbieter	**Bundesland**	**Region**
Deutsche Bahn	Deutschland	Deutschland
Karlsruher Verkehrsverbund	Baden-Württemberg	Karlsruhe
Münchner Verkehrs- und Tarifverbund	Bayern	München
Verkehrsverbund Berlin-Brandenburg	Berlin	Berlin
Verkehrsver. Bremen-Niedersachsen	Bremen-Niedersa.	Bremen-Niedersa.
Hamburger Verkehrsverbund	Hamburg	Hamburg
Nordhessischer Verkehrsverbund	Hessen	Nord Hessen
Aachener Verkehrsverbund	Nordrhein-Westfalen	Aachen
Mitteldeutscher Verkehrsverbund	Thüringen	Thüringen
Saarländischer Verkehrsverbund	Saarland	Saarland

Tabelle 4.2.: Die zehn ausgewählten Bahn-Anbieter in Deutschland

[1] Die Bewertung kommt aus der Webseite `http://www.airlinequality.com/` Zugriff: 05/09/2015

4.1.1.3. Hotelkategorie

Die Hotels werden genauso wie die Fluggesellschaften ausgewählt. Das bedeutet, Webformulare von Hotels aus verschiedenen Ländern, Kulturen und Klassen (Sternen) werden untersucht. Die finalen zehn Hotels werden mit den genannten Kriterien in Tabelle 4.3 dargestellt.

Hotel	Land	Stadt	Klasse
Ibis	Deutschland	Karlsruhe	3
Swissotel	Deutschland	Dresden	5
Copacabana Rio	Brasilien	Rio de Janeiro	3
Andaz Wall Street	USA	New York	4
Hotel Windsor	Schweiz	Genf	3
The Toren	Niederlande	Amsterdam	5
Burj Al Arab	Vereinigte Arabische Emirate	Dubai	5
ITC Hotel	Indien	Neu-Delhi	5
Seven Stars Galleria	Italien	Mailand	7
Dar Zemora	Marokko	Marrakesch	4

Tabelle 4.3.: Die zehn ausgewählten Hotels

4.1.2. Konzepte determinieren

Nach der Auswahl von zehn Webseiten für jede Kategorie werden die Webformulare dieser Webseiten so analysiert, dass jedes Feld des ersten Webformulars mit den neun übrigen Formularen der Kategorie verglichen wird. In diesem Schritt werden die Namen jedes Konzepts in den Webformularen in Tabellen gesammelt und anschließend wird ein finaler Name für das Konzept ausgewählt.

An dieser Stelle könnte eine Ontologie für jedes einzelne Webformular erstellt werden. Danach kann daraus eine Ontologie aus allen Formularen definiert werden. Wir brauchen jedoch diese Informationen über jedes Formular, um die Konzepte auszuwählen und ihnen einen geeigneten Namen zu geben. Die Tabellen geben einen klaren Überblick über die Konzepte aller Webseiten und erledigen damit diese Aufgabe in viel weniger Zeit als die Erstellung einer Ontologie für jedes Formular.

4.1.2.1. Flugkonzepte

In Tabelle 4.4 werden die verschiedenen Namen der wichtigsten Flugkonzepte gelistet, die aus den zehn Fluggesellschaften gesammelt werden. Der ausgewählte finale Name ist die Überschrift. Die anderen Konzepte der Flugontologie werden im Anhang Tabelle D.7 aufgezeigt.

Fluggesellschaft	Departure	Arrival	Departure Date	Return Date
Lufthansa	From	To	Departing	Returning
Air New Zealand	From	To	Depart date	Return date
Cathay Pacific	Leaving from	Going to	From	Returning
South African	Dep. city	Destination city	Leaving	Returning
British Airways	From	To		
Ryanair	From	To	Fly out	Fly back
Air Canada	From	To	Departure date	Return date
Condor	From	To		
Emirates	Dep. Airport	Arr. Airport	Dep. date	Ret. date
Swiss Airlines	From	To		

Tabelle 4.4.: Die wichtigsten Konzeptnamen in den Flugformularen

Für das Konzept „Flight class“ wird in den Webseiten entweder der Name ”class” oder ”flight class” verwendet. Aber es ist nicht in allen Webseiten verfügbar. Wenn das Konzept „Flight class“ im Formular steht, hat es die Optionen „Economy“, „Premium economy“, „Business“ und „First“. Wenn es keine Reiseklasse gibt, zeigen die Webseiten die Klassenoptionen mit den Suchergebnissen. Es gibt hier verschiedene Klassen, die sich je nach Fluggesellschaft unterscheiden. Beispielsweise „FlyDeal“ oder „Light“, „FlyClassic“, „FlyFlex“ und „Business plus“, wie z.B. bei „Ryanair“.

Der Flug kann Hin- oder Rückflug sein. Hier werden Optionsfelder mit den beiden Möglichkeiten benutzt, oder der Flug wird Standard als Rückflug betrachtet mit einem Markierungsfeld oder Knopf für Hinflug.

4.1.2.2. Bahnkonzepte

In der Bahnkategorie werden die Webformulare der zehn Webseiten genauso wie bei der Flugkategorie analysiert. Das heißt, der am meisten wiederholte Name wird für das Konzept ausgewählt. Die Tabelle 4.5 zeigt die wichtigsten Konzeptnamen der Bahnkategorie. Die anderen werden im Anhang Tabelle D.8 und Tabelle D.9 dargestellt.

Anbieter	Start	Destination	Date	Time	Departure/Arrival
DB	From	To			Dep./Arr.
KVV	From	To	On	At	Dep./Arr.
MVV	From	To	Date	Time	Dep./Arr.
VBB	From	To			Dep./Arr.
VBN	From	To	Date	Time	Dep./Arr.
HVV	From	To	Date	Time	Dep./Arr.
NVV	Start	Destination	Date	Time	Dep./Arr.
AVV	Start	Destination		Time	Dep./Arr.
MDV	Startort	Zielort	Datum	Uhrzeit	Abfahrt/Ankunft
saarVV	Abfahrtsort	Zielort			

Tabelle 4.5.: Die wichtigsten Konzeptnamen der Bahnformulare

Die Formulare, die zu Webseiten der Nahverkehrsverbünde gehören, beinhalten die Konzepte in Tabelle 4.5. Die Webseiten bieten zusätzliche Suchoptionen, die durch den Link „Erweitere Suche“ erreichbar sind. Die Optionen lauten: Fahrt über bestimmte Haltestelle, Verkehrsmittel, Mobilitätseinschränkung, Wege zur Haltestelle, Fußwegzeit und Umsteigen. Die Deutsche Bahn umfasst Verbindungsangaben über ganz Deutschland hinweg

einschließlich den lokalen Verkehr. Daher bietet das Formular der Deutschen Bahn nicht nur die Nahverkehr-Optionen, sondern auch die Optionen, die zum Fernverkehr gehören. Beispiele dafür sind die Anzahl der Reisenden, die Nutzung der BahnCards und Sitzplatzreservierung.

4.1.2.3. Hotelkonzepte

Die Formulare der zehn Hotels werden auch gleichermaßen analysiert und die geeignete Namen werden für jedes Konzept ausgewählt. Die ausgewählten Namen werden in Tabelle 4.6 dargestellt.

Hotel	**Arrival date**	**Departure date**	**Rooms**
Ibis	Arrival	Departure	Room(s)
Swissotel	Arrival date	Departure date	Room(s)
Copacabana Rio	Arrival	Departure	Number of Rooms
Andaz Wall Street	Arrive	Depart	
Hotel Windsor	Arrival	Departure	Rooms
The Toren	Arr.	Dep.	
Burj Al Arab	Check in date	Check out date	Rooms
ITC Hotel	Arrival	Departure	
Seven Stars Galleria	Arrival	Departure	
Dar Zemora	Check in	Check out	

Tabelle 4.6.: Die wichtigsten Konzeptnamen der Hotelformulare

Die meisten Webformulare der Hotelwebseiten bieten die Konzepte in der Tabelle 4.6 an oder nur „Arrival date“ und „Departure date“. Andere Hotels besitzen auch „Promotion code“. Alle andere Optionen wie z.B. „Rates“ oder „Zimmer Informationen“ spielen gar keine Rolle in den Formularen und werden erst nach der Suche in den Suchergebnissen angezeigt. Anschließend kann der Nutzer aus den verschiedenen Hotels spezifische Optionen auswählen. Diese Optionen werden trotzdem betrachtet, um so viele Informationen über diese Domäne zu sammeln.

4.1.3. Die Obligatorik der Webformularkonzepte

Einige Konzepte bzw. Felder der Webformulare repräsentieren notwendige Informationen, die in allen Suchanfragen und allen Formularen stehen müssen. Deswegen sind sie obligatorisch. Andere Konzepte stellen zusätzliche nicht kritische Informationen dar. Die Suche lässt sich ohne diese Felder erledigen. Diese Konzepte sind daher optional. In diesem Schritt wird jedes Konzept dahingehend überprüft, ob es obligatorisch oder optional ist. Es wird versucht, die Suche ohne jeweiliges Konzept auszuführen, anschließend wird über die Obligatorik des Konzepts im Webformular entschieden.

Darauf aufbauend werden die Konzepte „Departure“, „Arrival“ und „Departure date“ immer obligatorisch. Das Konzept „Return date“ ist obligatorisch unter der Bedingung, dass der Flug ein „Return-Flug“ ist. Da das Feld „Return Date“ erscheint nicht, wenn der Flug „One-way“ ist.

Einige Konzepte haben einen Standardwert. Wenn der Nutzer keine Informationen in diesen Feldern eingibt, wird dieser genommen. Beispiele sind die Anzahl der Reisenden in einem Flug und besonders das Konzept „Adults“. Gibt der Nutzer keine Informationen in diesem Feld an, wird die Suche mit dem Wert „1“ für dieses Feld ausgeführt. Das Konzept wird in diesem Fall nicht obligatorisch betrachtet. Derselbe Fall gilt für das Konzept „Flight class“. Wird keine Klasse ausgewählt, wird die Suche mit der Klasse „Economy“

ausgeführt. Hier ist jedoch das Konzept „Flight class“ nicht in allen Webformularen verfügbar, wie z.B. bei „Ryanair“ und „Swiss Airlines“, daher wird „Flight class“ als optional betrachtet. Alle anderen Konzepte in der Flugdomäne sind ebenfalls optional.

Bei der Bahndomäne sind die Konzepte „Start“, „Destination“, „Date“, „Time“ obligatorische Konzepte. Das Konzept „Departure/Arrival“ gibt Auskunft darüber, ob die angegebenen Werte für Datum und Uhrzeit für die Abfahrt oder die Ankunft gelten. Es wird als optional betrachtet. Wird es nicht vom Nutzer eingegeben, wird der Standardwert „Departure“ verwendet. Andere Konzepte sind immer optional.

In den Hotelformularen sind nur die Konzepte „Arrival date“ und „Departure date“ obligatorisch und alle anderen optional. Die Formulare mit den Feldern „Rooms“ und „Number of adults“ haben immer den Wert „1“ für diese Felder und werden als optional betrachtet.

Abbildung 4.5, Abbildung 4.6 und Abbildung 4.7 zeigen grafisch die Konzepte in den Flug-, Bahn- und Hotelontologien. Diese Ontologien stellen die wichtigsten Konzepten jeder Kategorie dar und werden als Ontologie der Kategorie bezeichnet.

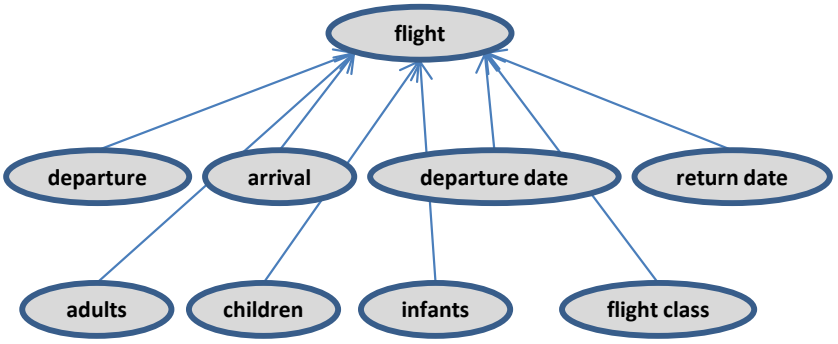

Abbildung 4.5.: Flug-Ontologie

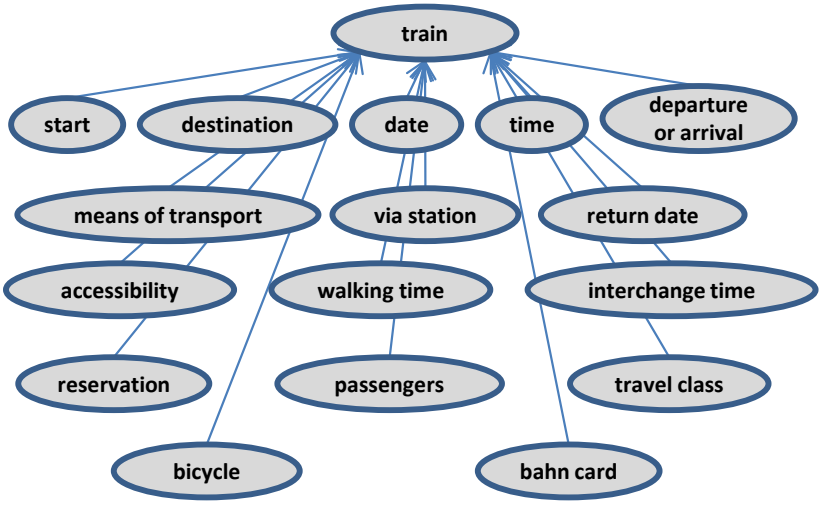

Abbildung 4.6.: Bahn-Ontologie

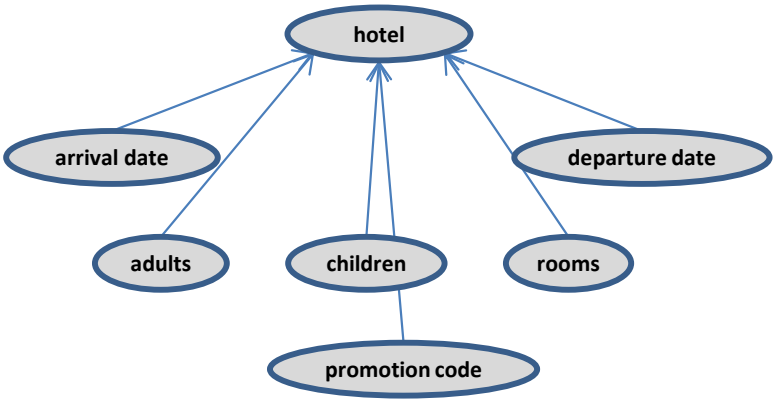

Abbildung 4.7.: Hotel-Ontologie

4.1.4. Nutzeranfrage analysieren

Die aktive Ontologie empfängt die sprachlichen Eingaben der Nutzer in den Sensorknoten, wo die eintreffenden Informationen der Nutzeranfragen gefiltert und erkannt werden. Die Sensorknoten verwenden verschiedene Techniken, um die Informationen herauszufiltern. Diese Techniken werden ermittelt je nach Existenz der Informationen in den Nutzeranfragen. Ein Beispiel hierfür ist der Satz „Find me a flight from Karlsruhe to Berlin“. Hier steht der Startpunkt „Karlsruhe“ nach dem Wort „from“, deswegen kann das Wort „from“ als Präfix verwendet werden, um den Startpunkt eines Flugs zu erkennen. Von daher ist es nötig zu wissen, wie die Nutzer sprachlich nach einer Flug-, Bahn- oder Hotelbuchung fragen können, vor allem, da die schon zur Verfügung gestellten intelligenten Sprachassistenten noch keine Buchung erledigen können. Um diese Frage zu beantworten, wurde in dieser Arbeit eine Umfrage durchgeführt, in der Nutzer von Sprachassistenten gebeten wurden, eine Flug-, Bahn- und Hotelbuchung sprachlich durchzuführen. Die Antworten werden gesammelt und hier analysiert, um geeignete Techniken zu ermitteln.

4.1.4.1. Umfrage entwerfen

Nun stellt sich die Frage danach, wie Fragen entworfen werden. Insbesondere, da die Buchungsanfragen der Nutzer sich je nach Reiseerfahrung unterscheiden können. Viele Nutzer reisen gerne und kennen sich mit den Reiseoptionen sehr gut aus. Sie können deswegen ihre Präferenzen sehr gut ausdrücken. Auf der anderen Seite sind einige Nutzer noch nicht gereist, und werden die Fragen in der Umfrage zum ersten Mal beantworten. Daher brauchen sie vielleicht einen Überblick über die verschiedenen vorhandenen Optionen. Also wird eine Vorstellung gemacht, damit die Nutzer, die noch nicht gereist sind, den intelligenten Assistenten zum ersten Mal verwenden können. Aus diesem Grund wird die Umfrage so entwickelt, dass die Nutzer drei Fragen bekommen. Bei der ersten Frage werden die Nutzer gebeten, das Flugticket, Bahnticket und Hotel sprachlich zu buchen, ohne Hinweise zu geben. In der zweiten Frage werden den Nutzern Bilder von im Internet verfügbaren Buchungsformularen einiger sehr bekannter Fluggesellschaften, Bahnanbieter und Hotels angezeigt, um einen Überblick über die Optionen in den Webformularen zu haben. In der letzten Frage werden den Nutzern einige Optionen gegeben. Diese Optionen können die Reisenden in Fluggesellschaftswebseiten, Bahnen und Hotels bestellen, wie z.B. Sitzplatz und zusätzliches Gepäck bei einem Flug.

Wegen der großen Rolle, die der Sprachfaktor bei der Erkennung der Informationen einer Flug- oder Bahnbuchung spielt, und aufgrund der unterschiedlichen Sprachniveaus der Nutzer – die Nutzer sollen ihre Anfrage auf Englisch formulieren – werden in der Umfrage Nutzer mit Englisch als Erst- oder Zweitsprache sowie Anglisten gefragt, um die größtmöglichen Sprachvarianten zu sammeln. Probanden aus verschiedenen Fachrichtungen werden ebenfalls gebeten, an der Umfrage teilzunehmen. Das Alter der Probanden liegt zwischen 15 und 58 Jahren.

Die finalen Fragen zur Flugbuchung werden in Abbildung 4.8 gezeigt. Die ganze Umfrage über Bahn- und Hotelbuchung sowie die Antworten der Nutzer werden im Anhang Tabelle A.1, Tabelle B.2 und Tabelle C.3 aufgelistet.

Imagine that you can vocally command your own mobile phone to make a flight ticket reservation:

1. Exclusively in English, state the sentence which you want to say to your mobile phone to book the ticket for you.
2. These two samples belong to two of the international flight booking websites. Look at them carefully and restate your sentence. They might remind you of more options.
3. Restate your request specifying additional options that you would like to have on the plane such as a seat by the window, special food, extra weight, etc.

Abbildung 4.8.: Die Umfrage - Fragen zur Flugbuchung

Die Ergebnisse der Umfrage werden in dieser Arbeit so verwendet, dass die eine Hälfte der Ergebnisse dieser Umfrage für die Entwicklung des Softwareprogramms genutzt wird, und zwar für die Auswahl der geeigneten Technik, die die eintreffenden Informationen in den sprachlichen Anfragen der Nutzer filtern und erkennen können. Die andere wird für die Evaluation des Systems benutzt. Es wird überprüft, ob die übrigen Nutzeranfragen mit dem schon entwickelten System erkannt und verstanden werden. Das vollständige Ergebnis der Umfrage wird in Tabelle A.1, Tabelle B.2 und Tabelle C.3 im Anhang dargestellt.

4.1.5. Erkennungstechnik für Ontologiekonzepte in Nutzeranfragen

Die Ergebnisse der Umfrage werden so analysiert, dass für jedes Konzept in den Flug-, Bahn- und Hotelontologien analysiert wird, wie die Nutzer in ihren Anfragen dieses Konzept ausgedrückt haben oder wie sie Informationen über das Konzept eingegeben haben.

Im Folgenden wird detailliert beschrieben, welche Techniken verwendet werden, um Konzepte der Flugontologie zu erkennen. Für die Bahn- und Hotelontologie werden ähnliche Konzepte gelistet.

Die Konzepte der Flugontologie werden im Einzelnen wie folgt untersucht.

4.1.5.1. Abflugort

Wie kann das System ein Wort als Abflugort erkennen? Diese Frage wird hier beantwortet. Das Wort sollte zunächst ein Ort sein und zwar eine Stadt, ein Land oder ein Flughafen. Alle Orte müssen in einer Liste gespeichert sein. Dann wird überprüft, ob das Wort in der Liste vorhanden ist. Ist dies der Fall, kann es als gültiger Ort bezeichnet werden. Bei einem Flug gibt es zwei wichtige Orte: Abflugort und Ankunftsort. Daher muss der Ort identifiziert werden. Das passiert mittels der Semantik in der Benutzeranfrage. Ein Ausdruck muss daher die Bedeutung von Abflug oder Start andeuten. Die Nutzer haben in allen ihren Antworten das Wort „from“, um auf den Abflug zu verweisen, wie z.B. „book

me a flight from Karlsruhe“ oder „I want to travel from Karlsruhe“. In beiden Beispielen wird „Karlsruhe“ nach „from“ und als Abflugort genannt. „From“ wird auch mit anderen Ausdrücken benutzt, wie „leave from“ und „fly from“, deswegen wird „from“ als „Präfix“ verwendet, um das „Departure“-Konzept oder die Bedeutung von Abflug zu erkennen. Trotz der Verwendung von „from“ in allen Nutzerantworten der Umfrage, werden weitere Sprachanfragen vorgestellt. Beispielsweise sind „The departure [place] is Karlsruhe“ oder „Karlsruhe is the departure [place]“ mögliche Anfragen. Daher werden „departure is“ und „is the departure“ als Präfixe und Postfixe hinzugefügt.

Der Ort nach „from“ besteht aus einem Wort wie „Karlsruhe“, zwei Wörter wie „Los Angeles“ oder drei Wörter wie „New York City“, deswegen werden drei Wörter nach dem Präfix so überprüft, dass das erste Wort, dann die ersten zwei Wörter und danach die drei Wörter zusammen getestet werden, ob sie in der Ort-Liste stehen bzw. einen gültigen Ort darstellen.

Die Verwendung der Präfixe reicht jedoch nicht. Das Wort nach z.B. „from“ muss ein Ort, und zwar Stadt, Flughafen oder Land sein. Hier wird zur Webformulare zurückgekehrt, um zu untersuchen, wie der Nutzer einen beliebigen Abflugort eingibt. Alle Formulare verwenden vordefinierte und bestimmte Listen von Städten oder Flughäfen, die die Fluggesellschaft bedient. Gibt der Nutzer „Manchester“ z.B. im Felder „from“ des Formulars von „British Airways“ ein, wird der vordefinierte Option „Manchester, Manchester (MAN), United Kingdom“ genommen. Alle Städte, die von einer Fluggesellschaft nicht bedient werden, können vom Nutzer nicht eingegeben werden. Ein Beispiel hierfür ist „Karlsruhe“, der Nutzer kann sie im Webformular von „British Airways“ nicht eingeben. Deswegen wird eine Liste von Städten, Flughäfen, Staaten und Ländern definiert, um die Kandidaten dahingehend zu überprüfen, ob sie zu einer dieser Listen gehören.

4.1.5.2. Ankunftsort

Um ein Wort bzw. Ausdruck als Ankunftsort zu erkennen, muss das Wort ein Ort sein und muss ebenfalls die Bedeutung von Ankunft erkannt werden, genauso wie beim Abflugort. Die Nutzer haben die Wörter „to“ und „towards“ verwendet, um auf Ankunft zu verweisen. Beispielsweise sind die Sätze „Please book me a ticket to Milan“ und „I would like to book a ticket on the flight heading towards London“ Im gleichen Maße wie bei „Departure“ werden zusätzliche Optionen für die Semantik unterstützt, die nicht in der Umfrage stehen. Wie z.B. „The Arrival [place] is Berlin“ oder „Berlin is the Arrival [place]“.

4.1.5.3. Abflug- und Rückflugdatum

Das Abflugdatum und das Rückflugdatum müssen ebenfalls zwei Voraussetzungen erfüllen: ein Datum beinhalten und semantisch die Inhalte „Abflug“ oder „Rückreise“ andeuten. Die Nutzer haben z.B. das Datum als 17/08/2016 eingegeben oder Ausdrücke wie „next Monday“, „tomorrow“ oder „Friday 24th“ verwendet. Diese Ausdrücke müssen so umformuliert werden, dass sie zu Daten wie 17/08/2016 konvertiert werden.

Als nächstes wird das Problem der Erkennung von Bedeutungen wie „Abflug“ oder „Rückreise“ betrachtet. In der Umfrage steht eine große Variante von Beispielen wie „I want to depart on 27/07/2015 and return on 15/10/2015“, „I want a return ticket from 11/11/2015 to 20/11/2015“ und „I want to fly tomorrow and return next Monday“. Es wird erkannt, dass Präfixe wie „depart on“, „return on“, „from“ und „to“ verwendet werden, um auf Abflug oder Rückreise zu verweisen. In dem dritten Beispiel gibt es jedoch keine Präfixe für das Abflugdatum. Aus diesem Grund wird eine spezielle Verarbeitung so entwickelt, dass für jedes Konzeptdatum Listen von Präfixen und Postfixen vordefiniert sind. Die Präfix-liste von Abflugdatum beinhaltet z.B. die Wörter „departure on“ und „fly on“, und die Liste für

Rückreisedatum enthält die Wörter „return on“ und „back on“. Nun wird für jedes erkannte Datum nach dem verwendeten Präfix oder Postfix in diesen Listen gesucht. Das Datum nach dem Präfix oder vor dem Postfix wird zum entsprechendem Konzept zugeordnet. Ein bestimmter Fall ist es, wenn gar kein Wort in den Listen in der Nutzeranfrage steht, wie z.B. „I want a ticket from Karlsruhe to Berlin, the dates are 13/03/2016 and 14/04/2016“. Hier gibt es keine Wörter, die Hinweise auf Abflug oder Rückreise haben. Deswegen wird das früheste Datum dem Abflug und das späteste der Rückreise zugeordnet.

Falls nur ein Datum in der Anfrage steht, wird es als Abflug betrachtet. Wie z.B. in dem Satz „I want to fly on 14/04/2016“. Auf diese Art und Weise wird jedes Datum richtig dem geeigneten Konzept zugewiesen. Die Erweiterung dieser Implementierung kann einfach für eine beliebige Anzahl von Daten benutzt werden. Jedes weitere Datum muss ein Wort oder eine Liste von Wörtern haben, die auf dieses Datum verweisen.

4.1.5.4. Anzahl der Erwachsenen, Kinder und Babys

Hier werden die Schlüsselwörter: adults, children und infants mit ihren Synonymen entweder als Präfixe oder Postfixe verwendet. Die dazu gehörigen Zahlen werden dann als die Anzahl der Reisenden erkannt. Wie im Beispiel „ for two adults“: hier ist „adults“ ein Postfix. In dem Satz „the number of children is three“ ist „children is“ ein Präfix.

4.1.5.5. Flugklasse

In den Nutzeranfragen wird eine endliche Liste von Optionen für die Wahl der Klasse verwendet und zwar „economy, first, business, premium economy“. Deswegen wird diese Liste erstellt, um diese Klassenoptionen in der Nutzeranfragen zu erkennen. Dem Konzept „Flugklasse“ wird das Wort, das im Satz kommt und in dieser Liste steht, zugewiesen.

Die erkannte Informationen werden als Fakten im Faktenspeicher abgelegt. Der Fakt beinhaltet den Namen des Konzepts und den zugehörigen Wert, wie z.B. „Departure(Karlsruhe)“. So können diese Informationen als Fakten wieder verwendet oder bearbeitet werden. Das wird detailliert im Kapitel 5 beschrieben.

Die Konzepte der Flug-, Bahn- und Hotelontologie werden mit der verwendeten Erkennungstechnik in Tabelle 4.7, Tabelle 4.8 und Tabelle 4.9 gelistet.

Konzept	**Verwendete Technik**	**Beispiel**
departure, arrival	Präfixe, Postfixe, Wörterliste	from Karlsruhe
dep. date, ret. date	Präfixe, Postfixe, reguläre Ausdrücke	11/12/2015
flight class	Präfixe, Postfixe, Wörterliste	economy class
adults, children, infants	Präfixe, Postfixe, Wörterliste	with 2 children

Tabelle 4.7.: Verwendete Techniken für Erkennung der Konzepte der Flugkategorie

4.1.5.6. Identifizierung der aktiven Ontologie

Der Nutzer soll in der Lage sein, seine Anfrage direkt einzugeben, dann muss das System die Anfrage verstehen, ob es um Flug, Bahn oder Hotel geht. Ein Beispiel hierfür ist die Anfrage „book me a flight to Berlin“. In diesem Beispiel geht die Anfrage um einen Flug. Dies kann unmittelbar entschieden werden, wegen der Verwendung des Wortes „Flight“ für Flug. Solche Wörter werden ausgenutzt, um die entsprechende aktive Ontologie auszuwählen, wie z.B. (flight, fly) für Flug, (train) für Bahn, (hotel) für Hotel.

Das System sollte ebenfalls in der Lage sein, die passende Ontologie selbst herauszufinden. Der Satz „book me a ticket from Stuttgart to Berlin“ beinhaltet kein Wort, das Hinweise

Konzept	Verwendete Technik	Beispiel
start, destination	Präfixe, Postfixe, Wörterliste	to Berlin
date, return date	Präfixe, Postfixe, reguläre Ausdrücke	01/11/2015
time	Präfixe, Postfixe, reguläre Ausdrücke	12:00
departure/arrival	Wörterliste	arrive
means of transport	Wörterliste	ICE
accessibility	Wörterliste	no stairs
via station	Präfixe	via Durlach
walking time	Präfixe	walking time is
interchange time	Präfixe	interchange time is
travel class	Wörterliste	first class
passengers	Präfixe, Postfixe, Wörterliste	2 passengers
reservation	Wörterliste	reserve a seat
bicycle	Wörterliste	bicycle
bahn card	Präfixe, Wörterliste	bahn card

Tabelle 4.8.: Verwendete Techniken für Erkennung der Konzepte der Bahnkategorie

Konzept	Verwendete Technik	Beispiel
departure date	Präfixe, Postfixe, reguläre Ausdrücke	05/01/2016
arrival date	Präfixe, Postfixe, reguläre Ausdrücke	02/01/2016
adults	Präfixe, Postfixe	for 2 adults
children	Präfixe, Postfixe	with 2 children
rooms	Präfixe, Postfixe	2 rooms
promotion code	Präfixe, Wörterliste	the promotion code is

Tabelle 4.9.: Verwendete Techniken für Erkennung der Konzepte der Hotelkategorie

über Flug oder Bahn gibt, und es ist nicht klar, ob der Benutzer ein Flug- oder Bahnticket buchen möchte. Das System berechnet in diesem Fall den Anteil von fehlenden obligatorischen Konzepten unter der Gesamtzahl von obligatorischen Konzepten. Im letzten Beispiel und im Fall, dass der Benutzer eine Flugbuchung meinte: Die Anzahl der gesamten obligatorischen Konzepten ist drei und zwar: „Departure“, „Arrival“, „Departure date“. In der Anfrage wird „Stuttgart“ als „Departure“ und „Berlin“ als „Arrival“ erkannt. Es fehlt noch das „Departure date“. In diesem Fall beträgt die Anzahl der fehlenden obligatorischen Konzepte eins und der Anteil ist $1/3 = 0.33$ oder (33%).

Für das Bahnticket sind die obligatorischen Konzepte vier: „Start“, „Destination“, „Journey date“ und „Journey time“. „Start“, „Destination“ wurden durch „Stuttgart“ und „Berlin“ erkannt. Der Anteil in diesem Fall ist $2/4 = 0.5$ oder (50%). Bei Hotel fehlen die beiden obligatorischen Konzepte „Arrival date“ und „Departure date“ und der Anteil ist 100 %.

Es wird ein maximaler Anteil von fehlenden obligatorischen Konzepten festgelegt. Dieser Anteil ist 49%. Alle Ontologien, bei denen mehr als 49% an obligatorischen Konzepten fehlen, werden nicht ausgewählt. In dem Beispiel wird entschieden, dass der Nutzer eine Flugbuchung möchte, weil der Anteil der fehlenden notwendigen Informationen kleiner als 49% ist. Gibt es mehrere Ontologien, bei denen der Anteil kleiner als 49% ist, wird diejenige mit dem kleinsten Anteil ausgewählt.

Es kann jedoch vorkommen, dass der Benutzer so wenige Informationen eingibt, dass es für die Entscheidung nicht ausreicht, wie z.B. in der Anfrage „I want to go to Berlin“. Die Anteile sind 66%, 75% und 100% für Flug, Bahn und Hotel. Alle Anteil sind großer als 49%. In diesem Fall muss der Benutzer gefragt werden, um mehr Informationen einzugeben.

4.2. Dialog-Manager

Die Suche in den Webformularen erfordert die Eingabe der obligatorischen Felder, wie z.B. „departure“, „arrival“ und „departure date“ in einem Flugformular. Hat der Nutzer keine Eingabe in diesen Feldern gemacht, wird eine Meldung im Formular angezeigt, dass diese Felder ausgefüllt werden müssen.

Bei den intelligenten Assistenten läuft es genau so, dass die obligatorischen Knoten der aktiven Ontologie erfüllt werden müssen. Hat der Nutzer obligatorische Informationen vergessen einzugeben, muss er nach der Eingabe dieser Informationen gefragt werden. Nun besteht die Möglichkeit, einen dynamischen Dialog-Manager zu entwerfen, um den Dialog mit dem Nutzer in einer interaktiven Art und Weise zu führen. Der Nutzer hat seine Informationen sprachlich eingegeben, deswegen wird ihm zum einen die Option gegeben, die fehlenden Informationen wieder sprachlich einzugeben. Zum anderen wird dem Nutzer zusätzlich ein passendes Formularfeld angezeigt, um die fehlende Eingabe kurz einzugeben, ohne einen kompletten Text zu sagen. Zum Beispiel fehlt „departure date“ in einer Anfrage für Flugbuchung, kann der Nutzer seine Antwort entweder als Text, wie „I want to depart on 12/02/2016“ oder das Datum „12/02/2016“ aus einem Kalender direkt auswählen.

Ein Dialog mit dem Nutzer wird auch durchgeführt, um zusätzliche Optionen zu bieten, die jeder Dienstleister bietet. Dieser Dialog gibt dem Nutzer die Möglichkeit, seine Präferenzen sprachlich als Text oder durch die Auswahl von geeigneten Formularelementen einzugeben. Z.B. bekommt der Nutzer die folgende Frage, wenn er einen Flug von „Swiss Airlines“ ausgewählt hat: „The preferred flight has three different classes: Light, FlyClassic and Business plus. Please select your preferred one!“

4.2.1. Fragen entwerfen

Die nächste Frage ist, wie die richtige Frage nach dem fehlenden Konzept erstellt werden soll. Die Konzepte stellen bei der Flugontologie z.B. Orte, Daten, Anzahl von Reisende

oder Liste von bestimmten Möglichkeiten dar. Die Fragen können – je nach dem fehlenden Konzept – entweder Ja-/Nein-Fragen oder W-Fragen sein. Ein Beispiel hierfür ist die Frage nach dem „departure date“: „Departure date is missing!“. Hier muss der Nutzer diese Frage beantworten. „This Flight has different travel classes! If you have a preferred choice, you can select it from the list or enter a text!“ ist eine Frage nach der optionalen Information „flight class“, Der Nutzer muss diese Frage nicht beantworten, deswegen hat die Frage die Bedeutung von Möglichkeit im Satz „If you have a preferred choice“.

Der Nutzer will seine Buchung so schnell wie möglich erledigen und hat möglicherweise kein Interesse daran, optionale Fragen zu beantworten, weshalb er zunächst gefragt wird ob er dies überhaupt möchte. Bei seiner Bestätigung bekommt er die entsprechenden Fragen. Falls er keine weiteren Fragen möchte, wird direkt die Suche mit den schon gegebenen Informationen durchgeführt.

Die Fragen können entweder manuell im System gespeichert oder automatisch vom System generiert werden. Bei dem manuellen Ansatz wird eine Frage für jedes definierte Konzept gespeichert. Dabei werden auch Informationen über dem Typ der Antwort gespeichert, wie z.B. ob die geeignete Antwort von einem Kalender, einem Textfeld oder einer Liste gewonnen wird. Dieser Ansatz hat den Vorteil, dass die Fragen ausdrucksvoll sind. Ein Beispiel hierfür ist die Frage „Where do you want to fly from“ für den Abflugort.

Wenn die Fragen automatisch vom System generiert werden, wird der Name des fehlenden Konzepts in ein vordefiniertes Muster eingesetzt, wie z.B. der Name „Departure“ in der Frage „Departure is missing! Please enter it!“. Diese Formulierung klingt weniger gut als „Where do you want to fly from?“. Damit das System wohlklingend formulierte Fragen generiert, müssen geeignete Fragewörter und Verben für jedes Konzept ebenfalls generiert werden. Dies erfordert, dass viele Trainingsbeispiele zur Verfügung gestellt werden müssen, damit das System wohlklingend formulierte Fragen stellen kann.

Eine Kombination aus den beiden Ansätze wird für die Erstellung der Fragen definiert. Die Frage für „departure“ wird z.B. so formuliert: „Departure: Where do you want to fly from?“ Der Name „departure“ wird vom System automatisch generiert und die Frage „Where do you want to fly from?“ wird im System manuell gespeichert.

Der Nutzer bekommt die Fragen in einer bestimmten Reihenfolge so, dass jede Frage eine Priorität hat, die aussagt, wo diese Frage in Bezug auf andere Fragen steht. Diese Priorität wird nach der Reihenfolge der Felder in den Webformularen determiniert. Z.B. ist das Feld „departure“ das erste Feld in allen Fluggesellschaftsformularen, deswegen wird das Konzept die Priorität eins haben. „arrival“ wird die Priorität zwei gegeben, weil es genau nach „departure“ steht. Die Ausdrücken „departure date“ und „arrival date“ werden die Prioritäten drei bzw. vier gegeben. Genau so werden die Fragen nach optionalen Konzepten gehandhabt. Die Priorität wird manuell zu jeder definierten Frage hinzugefügt.

4.2.2. Verarbeitung der Nutzerantwort

Wie es bereits erwähnt wurde, kann der Nutzer die Fragen entweder als Text beantworten oder die Antwort in einem Formularelement eingeben. Der Text wird direkt an die aktive Ontologie weitergeleitet, so dass die aktive Ontologie die eintreffenden Informationen herausfiltert. Die Antwort in einem Formularfeld muss zu einem Satz konvertiert werden. Dieser Satz beinhaltet notwendige Wörter, die die Ontologie braucht, um die Antwort richtig zu erkennen. Ein Beispiel hierfür ist, wenn der Nutzer das Datum „12/02/2016“ als „return date“ von dem gezeigten Kalender auswählt. Die aktive Ontologie erfordert bestimmte Wörter, um dieses Datum als „return date“ zu verstehen. Daher werden ein der für die Erkennung des Konzepts verwendeten Präfixe wie z.B. „return“ zum Datum hinzugefügt. Der Satz kann z.B. „return on 12/02/2016“ sein. Dieser Satz wird in die aktive Ontologie abgeschickt und das Datum wird nun als „return date“ erkannt.

4.2.3. Die geeignete Dialog-Strategie

In Abschnitt 2.6.1 wurden drei Strategien für die Entwicklung eines Dialog-Managers beschrieben. Die erste Strategie wird durch bestimmte Zustände in einer bestimmten Reihenfolge repräsentiert. In dieser Arbeit ist der Nutzer jedoch in der Lage, seine Antworten in beliebiger Reihenfolge einzugeben. Der Dialog-Manager wurde daher nicht entsprechend dieser Strategie entworfen.

Nun wird die Agent-basierte Strategie diskutiert. In dieser Arbeit werden dem Nutzer keine Vorschläge angezeigt, wenn das System keine Suchergebnisse für die Nutzeranfrage finden kann. Ein Beispiel für eine Anfrage ist „find me a flight from Durlach to Berlin“. Das System liefert hier keine Ergebnisse. Im Fall des Agent-basierten Systems würde die Antwort z.B. lauten: „ There are no results from Durlach, but you can fly from Baden-Baden, it is the nearest Airport to your request.“. Diese Eigenschaft könnte in der Zukunft durch den Aufruf externer Dienste durchgeführt werden.

Der hier in dieser Arbeit entwickelte Dialog-Manager ist entsprechend der Strategie (Frame-basierte Strategie) entworfen, da das System Platzhalter hat. Diese Platzhalter sind die fehlenden Konzepte in der aktiven Ontologie. Der Nutzer kann die Antworten geben, ohne Rücksicht auf die Reihenfolge der Fragen zu nehmen. Das System ist deswegen dynamisch und flexibel.

4.2.4. Arbeitsprozess des Dialog-Managers

Alle von der aktiven Ontologie erkannten Informationen werden im Faktenspeicher abgelegt. Wenn obligatorische Informationen fehlen, wird ein Fakt mit dem Namen „missing_mandatory“ in den Faktenspeicher geschrieben. Die Argumente dieses Fakts sind der Name des fehlenden Konzepts und der Name der betreffenden aktiven Ontologie. Für fehlende optionale Konzepte wird jeweils ein „missing_optional“-Fakt geschrieben. Der Dialog-Manager muss diese Fakten haben, um Fragen nach fehlenden Informationen zu erstellen. Der beste Entwurf für den Dialog-Manager wird daher die aktive Ontologie sein. Denn wenn ein Fakt im Faktenspeicher abgelegt wird, wird eine entsprechende Regel in der aktiven Ontologie abgefeuert. Auf dieser Art und Weise kann der Dialog die Fakten im Faktenspeicher bekommen und daraus Fragen entwerfen. Die aktive Ontologie des Dialog-Managers besteht aus zwei Knoten. Der Knoten „Mandatory“ ist verantwortlich die „missing_mandatory“-Fakten zu bearbeiten und die entsprechende Fragen zu erstellen. Der zweite Knoten ist der „Optional“, der die Fakten „missing_optional“ bearbeitet.

Den Dialog-Manager als aktive Ontologie zu entwerfen, ist eine neuartige Ansatz. Dieser Ansatz ist so flexibel, dass der Nutzer die Informationen auf einmal oder nacheinander in egal welcher Reihenfolge eingeben kann. Die Informationen sind gut repräsentiert in Form von Fakten im Faktenspeicher.

4.3. Formularaufruf

Nach der Sammlung aller für die Buchung benötigten Informationen werden diese Informationen an die Webformulare der Buchungswebseiten geschickt, um die Ergebnisse zu sammeln und dem Nutzer anzuzeigen. Zu diesem Zweck wird in dieser Arbeit das Werkzeug Selenium verwendet, das automatisch die Felder des Webformulars ausfüllt, das Formular abschickt und die Suchergebnisse mit den entsprechenden Optionen sammelt. Für jede Nutzeranfrage wird ein Aufruf von zehn Formularen durchgeführt. Das benötigt viel Zeit. Diese Aufgabe wird daher in dieser Arbeit parallelisiert durch ein Thread-Pool von zehn Fäden, um die Zeit der Suchaufgabe so kurz wie möglich zu machen. Wie diese Aufgabe im Detail implementiert wird, ist im Kapitel 5 dargestellt.

4.4. Zusammenfassung

In der Analyse lag der Fokus zu Beginn auf der Ableitung der Ontologie von Webformularen. Dabei wurden die Webformulare analysiert, um die Konzepte mit der gleichen Semantik und verschiedenen Darstellungen in einer Ontologie zu vereinigen. Diese Ontologie repräsentiert alle mögliche Konzepte einer Kategorie von Flug-, Bahn- und Hotelbuchungen. Die aktive Ontologie wurde von der übergeordneten Ontologie jeder Kategorie erstellt. Die in der aktiven Ontologie verwendeten Technik, die Informationen von Nutzeranfragen herauszufiltern, wird mithilfe einer Umfrage entwickelt. Es wurde ebenfalls analysiert, wie ein interaktiver Dialog mit dem Nutzer durchgeführt werden kann, indem der Nutzer nach allen fehlenden Informationen oder vom Anbieter zusätzliche Dienste gefragt wird. Die Ergebnisse werden von Webformularen gesammelt, und zwar durch die Verwendung der Web-Scraping Technik, welche die Formulare automatisch ausfüllt und abschickt und deren Ergebnisse liest.

5. Entwurf und Implementierung

In diesem Kapitel werden die technischen Einzelheiten der im Kapitel 4 vorgestellten Planung genauer erklärt. Dabei werden Lösungsstrategien entworfen und umgesetzt, die für die Erstellung der aktiven Ontologie verwendet werden sowie für die Erkennung der Informationen in den sprachlichen Anfragen, den Entwurf des Dialog-Managers und das Sammeln der Ergebnisse von den Webformularen. An ausgewählten Beispielen werden zusätzlich Implementierungsbeispiele vorgestellt und näher beschrieben.

5.1. Aktive Ontologie

Der Entwurf und die Implementierung für die in dieser Arbeit definierten aktiven Ontologien werden in diesem Abschnitt vorgestellt. Dabei wird der Entwurf der verschiedenen Arten von Knoten präsentiert und wie diese Knoten wiederverwendet werden können, um die Konzepte einer bestimmten aktiven Ontologie zu implementieren.

5.1.1. Entwurf der aktiven Ontologie

Der Entwurf einer aktiven Ontologie besteht in dieser Arbeit aus den folgenden Phasen:

1. Es werden zehn Webformulare der Webseiten für jede Domäne oder Kategorie wie Flug- oder Hotelbuchung ausgewählt.
2. Diese Webseiten werden manuell analysiert, um die Konzepte jeder Kategorie zu identifizieren.
3. Für jedes Konzept wird ein Name ausgewählt. Dieser Name ist derjenige, der die Bedeutung des Konzepts trägt oder der sich häufig in den zehn Formularen wiederholt.
4. Jedes Konzept wird dann überprüft, ob es obligatorisch oder optional zu der Ontologie ist.
5. Es wird für jedes Konzept ein Wertebereich definiert. Dieser Bereich kann manuell identifiziert werden wie z.B. bei dem Abflugort oder Ankunftsort. Die beiden Konzepte müssen eine Liste von Orten erkennen. Diese Liste unterscheidet sich je nach Fluggesellschaft bzw. danach, welche Flughäfen die jeweilige Fluggesellschaft anbietet. In dieser Arbeit wird eine Liste von 40 Orten definiert, die von den meisten Fluggesellschaften bedient werden. Ein anderes Beispiel wäre die Anzahl der Reisenden in jeder der Kategorien „Erwachsene“, „Kinder“ oder „Babys“, bei denen der Mittelwert aller Fluggesellschaften ermittelt wird.

6. Nach der Ermittlung der Konzepte ist es wichtig zu wissen, wie der Nutzer Informationen über diese Konzepte sprachlich eingibt oder anders ausgedrückt, welche Begriffe bzw. Semantik der Nutzer verwendet, um Informationen über ein bestimmtes Konzept zu übermitteln. In dieser Arbeit wurde z.B. eine Umfrage entwickelt, bei der gezeigt wird, wie die Nutzer eine Flug-, Bahn- oder Hotelbuchung sprachlich erledigen. Die Werte aller Konzepte der Flug-, Bahn- und Hotelontologie werden in Tabelle D.4, Tabelle D.5 und Tabelle D.6 im Anhang dargestellt.

7. Jedes Konzept muss die entsprechenden Informationen in der Anfrage erkennen. Dafür benötigt es verschiedene Arten von Knoten, die verschiedene Arten von Informationen identifizieren, wie z.B. Ort, Datum, Uhrzeit oder Zahlen. Deswegen werden für jedes Konzept alle Knoten und Knotentypen sowie die Beziehungen zwischen diesen Knoten definiert.

Diese Schritte werden in Kapitel 4 dargestellt. Es wurde analysiert, welche Knoten oder Klassen benötigt werden, um die Informationen zu erkennen. Ein Beispiel hierfür sind die notwendigen Techniken, um Informationen über das Konzept „Start“ zu erkennen (siehe Tabelle 4.8). Es werden Knoten benötigt, die die Orte und Semantik (Präfixe und Postfixe) aus der sprachlichen Eingabe extrahieren können.

Im Folgenden wird vorgestellt, wie die Informationen in der sprachlichen Anfrage in der aktiven Ontologie erkannt werden. Der Arbeitsprozess in der aktiven Ontologie wird mithilfe des folgenden Beispiels beschrieben. Der Benutzer hat die Anfrage „find me a flight from Karlsruhe to Berlin depart on 13/03/2016 and return on 14/04/2016“ [1] eingegeben. Die aktive Ontologie für den Flug beinhaltet die Konzepte „Departure“, „Arrival“ und „Departure date“. Dazu müssen die Informationen in der Anfrage so zugeordnet werden, dass „Karlsruhe“ das Konzept „Departure“ repräsentiert, „Berlin“ das „Arrival“ und „13/03/2016“ das „Departure date“. Die Blätter bzw. Sensorknoten in der aktiven Ontologie empfangen die Anfrage und müssen die Informationen, die sie enthalten, erkennen.

Wie in Abschnitt 4.1.5 erwähnt wird, muss das Konzept „Departure“ zwei Voraussetzungen erfüllen. Die vom Benutzer gesprochenen Wörter müssen einen Ort (Stadt oder Flughafen) wie „Karlsruhe“ und einen Begriff beinhalten, der die Semantik von „Departure“ andeutet, wie z.B. „from“. Deswegen müssen hier zwei Arten von Knoten entworfen werden. Die erste ist verantwortlich für die Erkennung der Orte und die zweite erkennt die Semantik. Der Knoten der Orte erkennt so „Karlsruhe“ und „Berlin“, während der Knoten der Semantik je nach Konzept die entsprechenden Wörter begreift. Dies ist zum Beispiel der Fall, wenn „from“ für „Departure“ steht und „to“ für „Arrival“. Diese Knoten geben ihren Elternknoten die erkannten Informationen weiter. Die Elternknoten müssen nun diese Informationen überprüfen. Als Beispiel hierfür erhält der Knoten, der „Departure“ identifizieren muss, „Karlsruhe“ und „Berlin“ als Orte und „from“ als die Semantik von „Departure“. Es wird überprüft, welche Zusammensetzung gültig ist: „from Karlsruhe“ oder „from Berlin“. In diesem Fall wird für „Karlsruhe“ entschieden, weil die beiden Wörter „from“ und „Karlsruhe“ bei der sprachlichen Eingabe direkt nacheinander kommen. Deswegen hat das Konzept „Departure“ den Wert „Karlsruhe“ und wird so als Departure(Karlsruhe) repräsentiert.

Auf diese Art und Weise werden alle Informationen den entsprechenden Konzepten zugewiesen. Die Konzepte sind Mitglieder der Knoten „Flight“. Diese Konzept(Wert)-Paare werden im Knoten „Flight“ zusammengetragen und zum Knoten „Command“ der aktiven Ontologie weitergeleitet, um die entsprechende Aktion, wie z.B. „Flugsuche“, durchzuführen.

Die aktive Ontologie besteht daher aus drei Phasen. Die erste ist die Worterkennungsphase, bei der bestimmte Wörter oder Ausdrücke erkannt werden. Die zweite ist die Kon-

[1] Dieser Satz und alle Beispielsätze kommen aus der Umfrage und wurden nicht bearbeitet.

zeptphase, wo jedes Konzept Wörter erhält, wobei diejenigen Wörter, welche die meisten Konzeptvoraussetzungen erfüllen, ausgewählt werden. Die Knoten in dieser Worterkennungsphase geben die erkannten Informationen an die Elternknoten in der Konzeptphase weiter. Die Kommandophase ist die dritte und letzte Phase, in der alle Konzepte mit deren Werten stehen, um das Kommando (Aktion) auszuführen. Diese Phasen sind in Abbildung 5.1 dargestellt, in der ersichtlich wird, dass die Konzepte „Departure“, „Arrival“, „Departure date“ und „Return date“ mit den Wörtern „Karlsruhe“, „Berlin“, „13/03/2016“ und „14/04/2016“ verknüpft sind. Die Informationen kommen aus der Anfrage „find me a flight from Karlsruhe to Berlin depart on 13/03/2016 and return on 14/04/2016“.

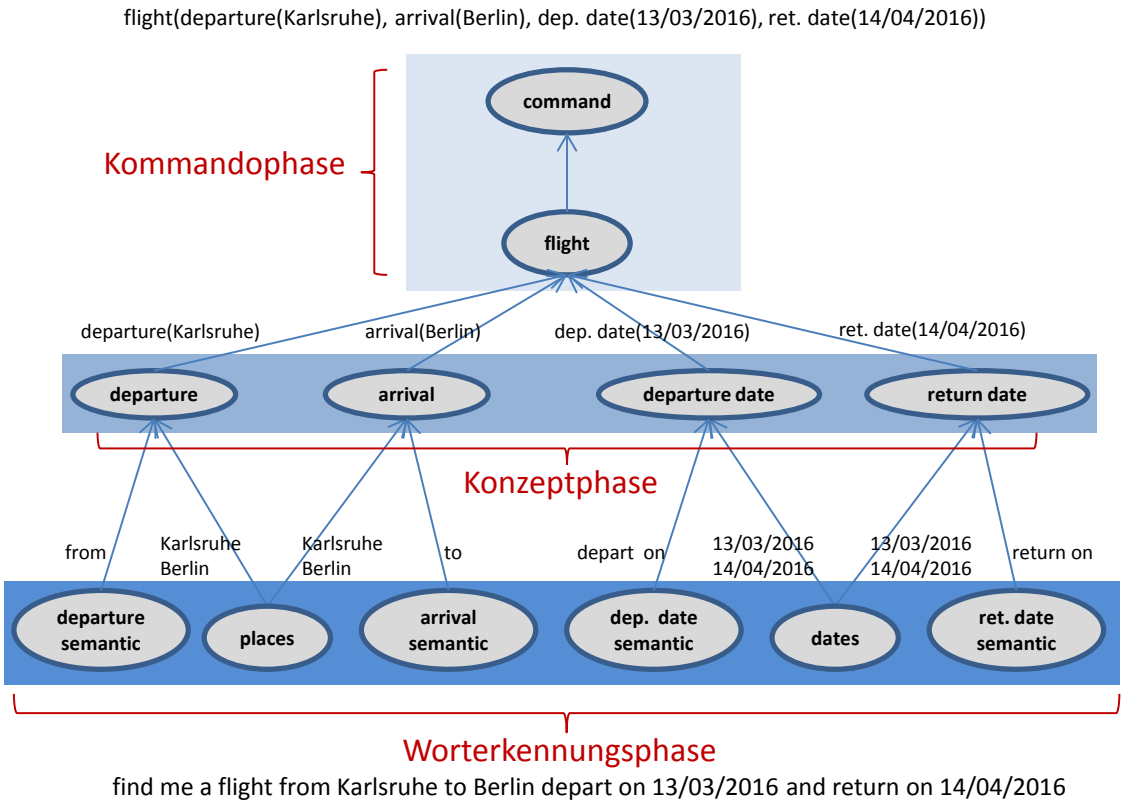

Abbildung 5.1.: Phasen der aktiven Ontologie

Dieser Entwurf mit den drei Phasen trennt die Worterkennung von der Konzeptzuweisung. Dadurch werden Knoten wie „places“ oder „dates“, die zu mehreren Konzepten in der aktiven Ontologie gehören, nur einmal definiert und mehrmals verwendet. Das wird erreicht, indem eine Beziehung „ist Mitglied von“ zwischen den Datum- oder Ortserkenner-Knoten und den Konzept-Knoten hergestellt wird. Ein Beispiel hierfür ist der Knoten „Dates“ in der aktiven Ontologie „Flug“. Er erkennt alle in der Benutzeranfrage vorkommenden Daten und schickt sie zu den Elternknoten „Departure date“ und „Return date“. Dort werden mithilfe der jeweils eigenen Semantik Wörter dem richtigen Datum des entsprechenden Konzepts zugeordnet. Das gilt auch für den Knoten „Places“ mit den Konzepten „Departure“ und „Arrival“.

5.1.1.1. Bausteine der aktiven Ontologie

Die Konzepte der aktiven Ontologie werden so implementiert, dass jedes Konzept, je nach deren Aufgabe, durch die Klasse „Concept“ oder einer deren Unterklassen repräsentiert

wird. Abbildung 5.2 zeigt ein UML-Klassendiagramm der am meisten verwendeten Klassen. Es ist erwähnenswert, dass die Klassen „Concept“, „TerminalNode“, „NonTerminalNode“, „WordListTerminalLeafNode“, „HelperNode“, „GatherNode“ und „SelectNode“ in EASIER [BL] definiert wurden. Die andere Klassen wurden in dieser Arbeit erstellt.

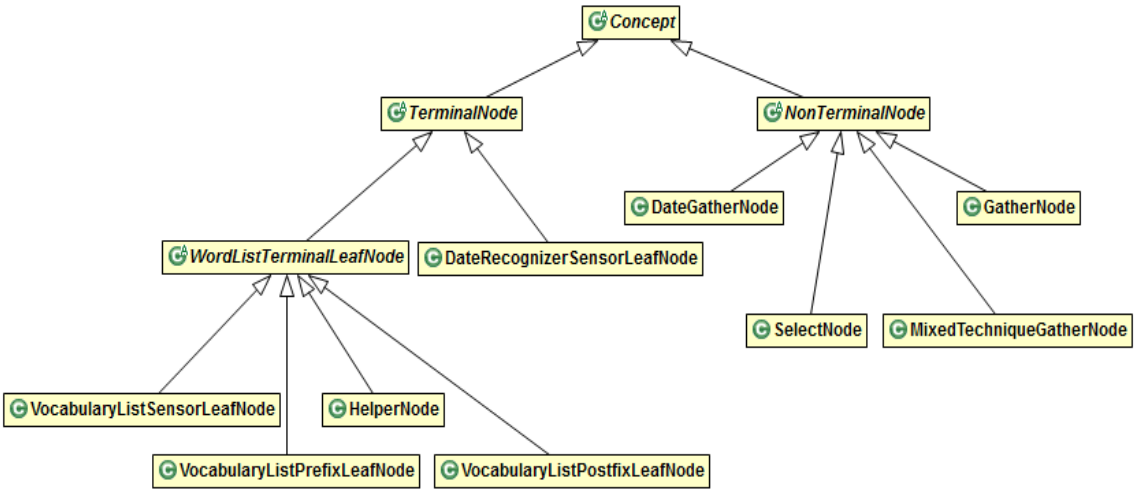

Abbildung 5.2.: Klassendiagramm für die wichtigsten Konzepte der aktiven Ontologie

In dieser Abbildung wird die Klasse „Concept“ in die Klassen „TerminalNode“ und „NonTerminalNode“ unterteilt. Die Klasse „TerminalNode“ beinhaltet die Unterklassen „WordListTerminalLeafNode“ und „DateRecognizerSensorLeafNode“. „WordListTerminalLeafNode“ repräsentiert alle Blätterknoten, die mithilfe einer Liste von Wörtern die Informationen in Nutzeranfragen erkennen.

Ein Beispiel dafür ist die Klasse „VocabularyListSensorLeafNode“, die für die Erkennung bestimmter Wörter wie z.B. Orte verwendet wird. „VocabularyListPrefixLeafNode“ und „VocabualaryListPostfixLeafNode“ erkennen Präfixe und Postfixe wie z.B. das Präfix „from“ und das Postfix „is departure date“. „HelperNode“ wird für die Erkennung der Wörter verwendet, die die aktive Ontologie unterscheidet wie „flight“ or „airport“ für die Flugontologie. „DateRecognizerSensorLeafNode“ erkennt die Wörter, die das Datum darstellt, wie „13/03/2016“.

Die Klasse „NonTerminalNode“ hat die Unterklassen „DateGatherNode“, „MixedTechniqueGatherNode“, „GatherNode“ und „SelectNode“. „MixedTechniqueGatherNode“ weist einem Wort das passende Konzept zu. „DateGatherNode“ ist verantwortlich für die Zuordnung eines Datums, das dem Konzept entspricht. In „DateGatherNode“ werden mehrere Voraussetzungen für die Datenerkennung definiert. Das ist der Fall, wenn zwei Daten ohne Semantik (Präfix oder Postfix) verfügbar sind. Dabei wird das frühere Datum zu „Departure date“ und das spätere zu „Return date“ zugewiesen, da das „Departure date“ immer vor dem „Return date“ erfolgt. Beispielsweise sind die Daten „13/03/2016“ und „14/04/2016“ im Satz „The dates are 13/03/2016 and 14/04/2016“, wo es keine Hinweise über „departure“ und „arrival“ gibt (siehe Abschnitt 4.1.5.3).

„GatherNode“ ist lediglich für das Sammeln der Informationen aus allen Kind-Knoten zuständig, wohingegen „MixedTechniqueGatherNode“ und „DateGatherNode“ Informationen von ihren Kind-Knoten sammeln und verarbeiten.

„SelectNode“ wählt die Informationen vom Kind-Knoten, dem die höchste Wahrscheinlichkeit innewohnt.

5.1.2. Implementierung der aktiven Ontologie

Die in dem letzten Abschnitt vorgestellten Klassen werden in jeder zu definierenden aktiven Ontologie wiederverwendet. In diesem Abschnitt wird vorgestellt, wie diese Klassen in der aktiven Ontologien „Flug“, „Bahn“ und „Hotel“ implementiert werden und welche Aufgaben die Methoden in diesen Klasse durchführen. Zunächst wird beschrieben, wie die sprachliche Anfrage des Nutzers gespeichert wird.

5.1.2.1. Sprachliche Anfrage als Fakten speichern

Wenn der Benutzer seine Anfrage eingibt, wird zunächst der Text der Anfrage als Fakten dargestellt und im Faktenspeicher gespeichert. Jedes Wort in diesem Text wird zum komplexen Fakt „WordWithIndex“ konvertiert, der die folgenden Parameter beinhaltet:

Die Anfrage „find me a flight from Karlsruhe to Berlin on 13/03/2016“ wird für die Erklärung folgender Fakten verwendet.

- *Word(value)*: Ein komplexer Fakt mit dem Namen „word“ und dem Parameter „value“, der das Wort in der Anfrage speichert z.B. word(find), word(Karlsruhe).
- *Index(value)*: Ein komplexer Fakt mit dem Namen „index“ und dem Parameter „value“, der den Index des Wortes im Satz speichert z.B index(0) für „find“, index(1) für „me“, index(5) für „Karlsruhe“. Der Index wird verwendet, um die Distanz zwischen zwei Ausdrücken in der Nutzeranfrage zu berechnen. Ein Beispiel hierfür ist die Distanz zwischen einem Präfix wie „from“ und einem Ort wie „Karlsruhe“. Um „Karlsruhe“ als Abflugort zu erkennen, müssen die beide Ausdrücke „from“ und „Karlsruhe“ direkt nacheinander stehen. Das heißt, die Distanz muss 1 betragen.
- *WordType(value)*: Ein komplexer Fakt mit dem Namen „wordType“ und dem Parameter „value“, der den Typ des Wortes speichert, z.B. wordType(Date) für den Ausdruck „13/03/2015“ und wordType(simpleWord) für das Wort „Karlsruhe“. Durch die Verwendung von „wordType“ können alle Wörter in der Benutzeranfrage, die zu einem bestimmten Typ gehören, aus dem Faktenspeicher schneller aufgerufen werden. Bei Datum z.B. ist es dann nicht mehr nötig, die Wörter zu überprüfen, ob sie mit dem Muster von Datum dd/MM/yyyy übereinstimmen. Eine Anfrage an den Faktenspeicher mit dem Argument „WordType(Date)“ gibt alle Daten zurück.

Alle diese Fakten werden für jedes Wort in diesem Beispiel in einem komplexen Fakt „WordWithIndex“ dargestellt: Ein Beispiel dafür ist die Repräsentation von „Karlsruhe“:

WordWithIndex(word(Karlsruhe), index(5), wordType(simpleWord))

Die Bearbeitung von sprachlichen Anfragen wird in der Methode *insertText(String sentence)* in der Klasse „FactStore“ realisiert. „sentence“ ist der Text der Nutzeranfrage. Die Methode *insertText(String sentence)* wurde in EASIER [BL] implementiert, wobei die Wörter der Benutzeranfrage ohne weitere Informationen gespeichert wurden. In dieser Arbeit wird jedes Wort als Komplexer Fakt mit den Argumente „word“, „index“ und „wordType“ verknüpft und abgespeichert.

5.1.2.2. Aktive Ontologie: Flug

Die Implementierung der aktiven Ontologie „Flug“ wird für jedes Konzept im Detail beschrieben. Alle Klassen der Knoten der aktiven Ontologie „Flug“ werden in der Tabelle 5.1 gelistet und im nächsten Abschnitt beschrieben. Die aktiven Ontologien „Bahn“ und „Hotel“ werden durch Abbildungen präsentiert. Falls eine neue Technik in der Bahn- oder Hotelontologie vorkommt, wird sie ebenfalls im Einzelnen dargestellt.

Klasse	Knoten
VocabularyListPrefixLeafNode	departurePrefix - arrivalPrefix - adultPrefix - childrenPrefix - babiesPrefix - flightClassPrefix - departureDatePrefix - returnDatePrefix
VocabularyListPostfixLeafNode	departurePostfix - arrivalPostfix - adultPostfix - childrenPostfix - babiesPostfix - flightClassPostfix - departureDatePostfix - returnDatePostfix
VocabularyListSensorLeafNode	places - numbers - flightClassOptions - action
DateRecognizerSensorLeafNode	dateRecognizer
MixedTechniqueGatherNode	departure - arrival - adultNumber - childrenNumber - babiesNumber - flightClass
DateGatherNode	departureDate - returnDate
AOGatherNode	flight
HelperSensorLeafNode	flightHelper
GatherNode	command

Tabelle 5.1.: Klassen der aktiven Ontologie (Flug)

- *Departure/Arrival*:

  Ein Sensorknoten mit dem Namen „places“ wird als ein Objekt der Klasse „VocabularyListSensorLeafNode“ definiert. Zu diesem Knoten wird eine Liste von Orten (Städte, Flughäfen, Staaten, Länder) erstellt. Diese Liste beinhaltet Städte von der besten 40 Flughäfen im Jahr 2015 [2] . Außerdem werden andere Städte manuell hinzugefügt, die von alle in dieser Arbeit zehn analysierten Webseiten bedient werden. Die Orte werden hier erkannt und dem Elternknoten weitergegeben.

  Falls der Ort aus nur einem Wort besteht, wie z.B. „Karlsruhe“, wird der Fakt dieses Wortes den Eltern gegeben. Aber wenn der Ort aus mehreren Wörtern besteht, wie z.B. „New York“, hat „New“ einen Index, z.B. (x), und „York“ dann den Index (x+1). In diesem Fall werden die beiden Wörter in nur einem Wort „New York“ mit dem Index (x) zusammengefasst. Dieser Schritt ist notwendig, um das Konzept den kompletten Wert zu haben und nicht nur ein Teil davon wie Z.B. nur „New“ oder nur „York“. Sie vereinfacht ebenfalls die Berechnung der Distanz zwischen dem Ort und einem Präfix oder Postfix. Die Distanz wird mit den gesamten Wert „New York“ einmal berechnet und nicht einmal mit „New“ und einmal mit „York“.

  Um die „Departure“ Semantik zu erkennen, wird ein Sensorknoten „departurePrefix“ von der Klasse „VocabularyListPrefixLeafNode“ erstellt. Alle möglichen Präfixe für „Departure“ werden zu diesem Konzept hinzugefügt, um sie aus der sprachliche Eingabe zu extrahieren, wie z.B. (from, departure is). Ein anderer Sensorknoten „departurePostfix“ wird von „VocabualaryListPostfixLeafNode“ mit den Wörtern (is the departure, is the leaving place) erstellt. Wenn das Präfix oder Postfix aus mehr als einem Wort besteht, geschieht das gleiche wie bei Orten.

  Das Konzept „Departure“ wird von der Klasse „MixedTechniqueGatherNode“ definiert. Deren Aufgabe ist die geeignete Ersetzung für „Departure“ zu determinieren. Als Beispiel wird noch mal die Anfrage „find me a flight from Karlsruhe to Berlin on 13/03/2016“ verwendet.
  Der Knoten „Departure“ empfängt (word(Karlsruhe), index(5)) und (word(Berlin), index(7)) von „places“ und (word(from), index(4)) von „departurePrefix“ und keine Information von „departurePostfix“.

[2] Die Liste entstammt der Webseite `http://www.worldairportawards.com/Awards/world_airport_rating.html` Zugriff: 11/01/2016

Die Aufgabe von „Departure“ ist nun zu determinieren, welches Wort – entweder „Karlsruhe“ oder „Berlin“ – dem Abflugort entspricht. Das geschieht durch die Berechnung der Distanz zwischen den beiden Orten und den vorhandenen Präfixen oder Postfixen wie z.B. in Abbildung 5.3. In diesem Beispiel gibt es keine Postfixe, deswegen werden nur die Präfixe getestet.

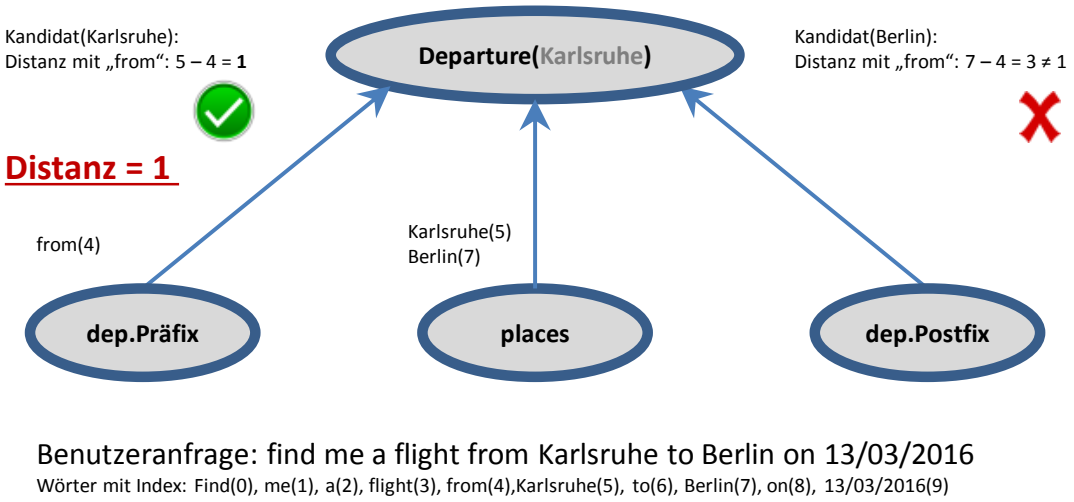

Abbildung 5.3.: Arbeitsprozess der Erkennung von „Departure“

Der Test wird folgendermaßen durchgeführt:
Es wird überprüft, ob die Distanz zwischen dem Ort und dem Präfix 1 beträgt. Das gilt für „Karlsruhe“ und „from“ (5 - 4 = 1) und nicht für „Berlin“ und „from“ (7 - 4 = 3). Beim Präfix ist es irrelevant, ob der Ort aus einem Wort oder mehreren Wörtern besteht, da der Ort immer den Index des ersten Wortes enthält. Wenn das Präfix aus mehreren Wörter besteht, wird der Index des Präfixes der Index des letzten Worts des Präfixes sein. Als Beispiel hierfür ist der Satz „The departure place is Karlsruhe“. Das Präfix „The departure place is“ hat den Index 3 und der Ort „Karlsruhe“ hat den Index 4. Die Distanz bleibt 1 (4-3) zwischen dem Ort und dem Präfix.

Für das Postfix wird das gleiche durchgeführt, außer wenn der Ort aus mehreren Wörtern besteht. Dies ist bei dem Satz „New York is the departure place“ der Fall. Hier trägt der Ort den Index(0), während das Postfix „is the departure“ den Index(2) enthält (Index des ersten Wortes). Es wird hier überprüft, ob die Distanz zwischen Ort und Postfix die Anzahl der Wörter, die den Ort repräsentieren, übereinstimmt. Die Anzahl der Wörter „New York“ ist zwei, und die Distanz zwischen „New York“ mit Index (0) und „is the departure“ mit Index (2) ist auch zwei [3].

Das gleiche gilt für „Arrival“, jedoch mit anderer Semantik, wie z.B. „to“ als Präfix und „is the arrival“ als Postfix.

Tabelle 5.2 zeigt die verwendeten Präfixe und Postfixe für „Departure“ und „Arrival“. Alle Präfixe und Postfixe für alle Konzepte der Flug-, Bahn- und Hotelontologie werden in Tabelle D.4, Tabelle D.5 und Tabelle D.6 im Anhang dargestellt.

[3]Die Anzahl der Wörter wird durch die Methode Split(String regex) berechnet. Split(String regex) wird verwendet, um eine String-Variable an allen Stellen, an denen regex vorkommt, in Unterstrings aufzuteilen. Sie gibt einen Array mit Teilen des gegebenen Strings zurück. Split() wird hier durch die Leertaste durchgeführt und die Anzahl der Wörter ist die Länge des Arrays.

Konzept	Semantik (Ausdrücke)
Departure_Prefix	from, departure is
Departure_Postfix	is the departure, is departure
Arrival_Prefix	to, towards, arrival is
Arrival_Postfix	is the arrival, is

Tabelle 5.2.: Semantik aller Konzepte der aktiven Ontologie „Flight“

- *Departure date und Return date*

  Zunächst wird beschrieben, wie das Datum in der sprachliche Eingabe des Nutzers erfasst wird. Es wird überprüft, welche Wörter in der Anfrage z.B. mit dem Ausdruck dd/MM/yyyy übereinstimmen. Das wird durch die Verwendung von Klassen „Pattern“ und „Matcher“ in „ java.util.regex“ erreicht. Die Methode *compile(String regex)* kompiliert den regulären Ausdruck zu einem Muster. Die Eingaben, die mit diesem Muster übereinstimmen, werden durch die Methode *matcher(CharSequence)* [4] erkannt.

  Quelltextausschnitt 5.1: Die Verwendung von Pattern und Matcher in java.util

```
Pattern p = Pattern.compile(regex);
Matcher m = p.matcher(sentence);
```

  Jedes erkannte Datum wird mit den Fakten „Word“, „Index“ und „WordType“ gespeichert. „WordType“ erlangt dabei den Wert „date“.

  Das Datum kann aber auch durch einige Ausdrücke, wie z.B. „tomorrow“, „next Friday“ oder „24th September“, eingegeben werden. Für solche Ausdrücke wird die Bibliothek „SUTime“ von „The Stanford NLP Group“ [5] verwendet. „SUTime“ erkennt und normalisiert Zeitangaben so, dass „next Friday at 2pm“ zu „2015-12-04T14:00 “ konvertiert wird, wenn „SUTime“ diese Zeiteingabe z.B. am 01/12/2015 erhält. Das Datum und die Zeit können verarbeitet werden, um sie in der gewünschten Formulierung darzustellen wie z.B. 04/12/2015 oder Fr, 04.12.2015.

  Die Daten werden im Knoten „dateRecognizer“ von der Klasse „DateRecognizerSensorLeafNode“ erkannt. Der „dateRecognizer“ gibt sodann alle Daten an seinen Elternknoten, wie z.B. „Departure date“ und „Return date“, weiter. „Departure date“ benötigt Informationen über die Semantik von „Departure“. Die Erkennung von „Departure date“ wird in Abbildung 5.4 dargestellt. Der Knoten „Departure date“ erhält durch den Knoten „departureDatePrefix“ von der Klasse „VocabularyListPrefixLeafNode“, Präfixe wie „depart on“ oder durch „departureDatePostfix“ von „VocabularyListPostfixLeafNode“ (in diesem Beispiel wird kein Postfix verwendet).

[4] Java Plattform. Zugriff: 15/12/2015 URL: `https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html`

[5] Stanford Temporal Tagger. Zugriff: 15/12/2015 URL: `http://nlp.stanford.edu/software/sutime.html`

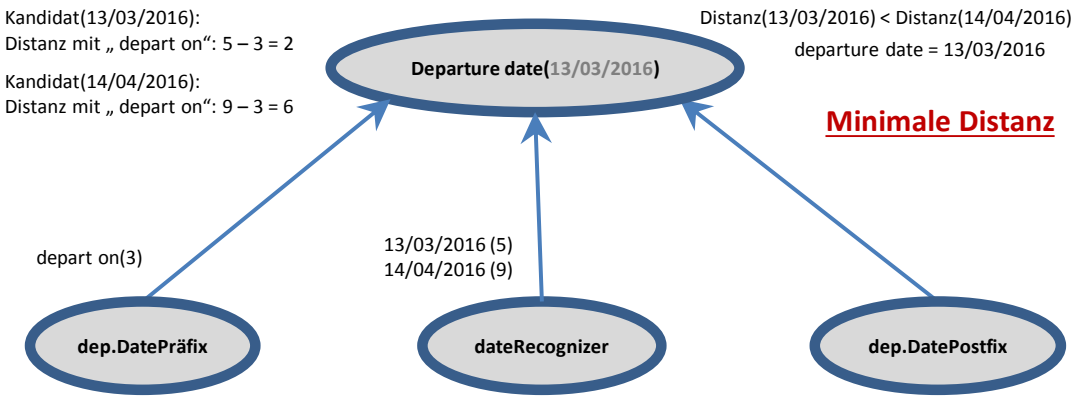

Abbildung 5.4.: Arbeitsprozess der Erkennung von „Departure date“

Das „departureDate“ wird von der Klasse „DateGatherNode“ erstellt. „departure-Date“ wählt dann das passende Datum durch die Berechnung der Distanz zwischen der „departure-Semantik“ und jedem erkannten Datum aus, genau so wie es beim Konzept „Departure“ der Fall ist.

Der Nutzer kann aber seine Daten eingeben, ohne eine weitere Semantik für „Departure“ oder „Return“ zu übertragen, wie z.B. in der Anfrage „find me a return flight from Karlsruhe to Berlin. The dates are 13/03/2015 and 14/04/2015“. Hier gibt es keinen expliziten Hinweis, welches Datum für „Departure date“ und welches für „Return date“ steht. In diesem Fall wird überprüft, welches Datum zeitlich vor dem anderen liegt. Das frühere 13/03/2015 wird dem „Departure date“ zugeordnet und das spätere 14/04/2015 dem „Return date“. Diese Strategie gilt nur, wenn lediglich zwei Daten in der Benutzeranfrage stehen: Bei drei Daten ohne Semantik kann es bspw. nicht eindeutig entschieden werden, welches Datum welchem Konzept angehört. In diesem Fall werden die Datum-Konzepte gar keinem Datum zugewiesen.

- *Adults, children und babies*

  Diese Konzepte werden auf dieselbe Weise so implementiert, dass zwei Knoten die Semantik des jeweiligen Konzepts erkennen wie „the number of children is“ als Präfix oder „children“ im Satz „with two children“ als Postfix. An dieser Stelle wird der Knoten „numbers“ von der Klasse „VocabularyListSensorLeafNode“ definiert. Dieser erkennt alle Zahlen in der sprachlichen Eingabe. Die Zahlen und die Semantik werden dann im Knoten „adultNumber“, „childrenNumber“ oder „babiesNumber“ von der Klasse „MixedTechniqueGatherNode“ getestet. Der Test wird ebenfalls über die Berechnung der Distanz durchgeführt.

- *Flight class*

  In der Flugklasse gibt es bestimmte Optionen, wie „economy“, „premium economy“, „business“ und „first“. Daher wird ein Knoten „flightClassOptions“ definiert, der diese bestimmte Optionen erkennt. Die Klasse dieses Knotens ist „VocabularyListSensor-LeafNode“. Zwei Knoten werden ebenfalls definiert, um die Semantik der Klasse zu begreifen wie z.B. „class“ im Satz „with first class“ als Postfix und „the flight class is“ als Präfix. Die Definition der beiden Knoten der Semantik erhöhen die Konfidenz für

das Konzept „Flight class“, weil z.B. „first“ in anderen Kontexten verwendet werden kann. Das ist der Fall bei dem Ausdruck „first of all“, der in der Benutzeranfrage stehen kann. Hier bezieht sich „first“ nicht auf die Flugklasse.

Der Knoten „flightClass“ von „MixedTechniqueGatherNode“ erhält die Optionen und Semantik und berechnet die Distanz wie bisher.

Alle diese Knoten sind in Abbildung 5.5 dargestellt. In dieser Abbildung stehen alle Knoten in der Worterkennungsphase, nämlich in der ersten, zweiten und dritten Reihe von unten wie „Flight class options“, „Flight class prefix“ und „Flight class postfix“. Die Konzepte in der Konzeptphase werden in der vierten Reihe aufgelistet.

5.1.2.3. Aktive Ontologie: Bahn

Die aktive Ontologie „Bahn“ wird durch die Verwendung der gleichen Klassen implementiert. Im Folgenden werden die Konzepte der aktiven Ontologie „Bahn“ aufgezählt. Die neuen Techniken werden jedoch detailliert diskutiert.

- *Start/Destination/Via station*: sie werden durch „places“, „präfixe“ und „postfixe“ implementiert, genau so wie bei „Departure“ oder „Arrival“ im Flug.
- *Date/Return date*: sie werden wie beim „Departure date“ und „Return date“ im Flug realisiert. Das heißt mit „dateRecognizer“ und geeigneten „präfixe“ und „postfixe“.
- *Travel class*: sie wird durch eine ähnliche Implementierung wie bei „Flight class“ umgesetzt, mit „Class Options“ und geeigneter Semantik.
- *Passenger/Walking time/Interchange time*: sie wiederholen das „Adult“ in der Ontologie Flug.
- *Time*: Das Konzept „Time“ wird durch reguläre Ausdrücke erkannt, wie bei „Departure date“ der Flug. Dadurch werden alle Uhrzeiten in der sprachlichen Eingabe erkannt und das Konzept nimmt die am besten geeignete Uhrzeit je nach verwendeter Semantik. Diese Uhrzeit ist diejenige, die am nächsten zu der Konzept-Semantik in der Benutzeranfrage liegt. Jedoch wird hier die Distanz nicht mit 1 verglichen, weil die Semantik nicht direkt vor der Uhrzeit stehen kann. Das folgende Beispiel erklärt diesen Fall: Der Nutzer sagt in den meisten Fällen das Datum und die Uhrzeit mit nur einem Präfix wie in der Anfrage: „I want to leave on 13/03/2016 at 13:00 and return on 14/04/2014 at 14:30“. In diesem Beispiel erhält das Konzept „Time“ die beiden Uhrzeiten „13:00“ und „14:30“ mit dem Wort „leave“. Die Uhrzeit, die die minimale Distanz zu „leave on“ hat, wird als „Time“ der Fahrt ausgewählt (hier: 13:00). Für das „Return Time“ wird die Distanz zwischen den Uhrzeiten und „return“ berechnet, und darauf aufbauend wird für 14:30 entschieden.

  Falls keine Semantik für die Hin- oder Rückfahrt in der Nutzeranfrage steht, wird der Nutzer nach den beiden Uhrzeiten gefragt, weil die frühere Uhrzeit z.B. nicht zum früheren Datum gehören muss.
- *(Departure/Arrival)/ Accessibility/ Means of transport/ Reservation/ Bicycle/ Bahn card*: Diese Konzepte können durch ihre mögliche Optionen direkt erkannt werden. Dies ist der Fall, wenn der Nutzer „I want an ICE Ticket from Karlsruhe to Berlin, and I want to take my bike with me“ sagt. „ICE“ ist dabei eine Option des Konzepts „Means of transport“, daher wird das Konzept „Means of transport“ dem Wert „ICE“ zugewiesen. Das gleiche gilt auch für „Bicycle“, „bike“ wird dem Konzept „Bicycle“ zugeordnet. Diese Konzepte werden durch die Klasse „VocabularyListSensorConceptLeafNode“ definiert, bei der die Optionen bzw. Werte erkannt und direkt mit dem Konzept verknüpft werden. Diese Art von Knoten gehört deswegen zu den beiden Worterkennungs- und Konzeptphasen.

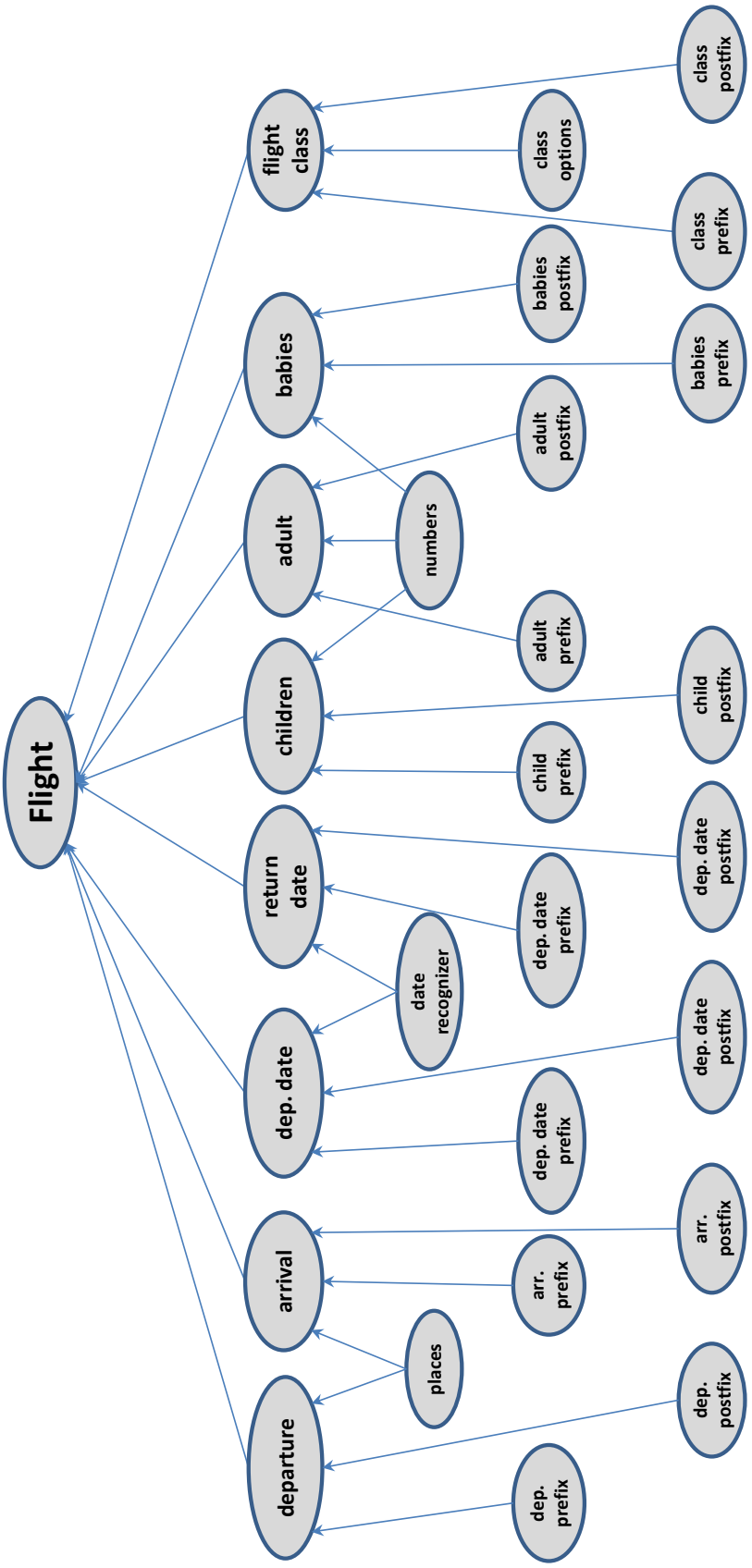

Abbildung 5.5.: Implementierung der Konzepte für die aktive Ontologie: Flug

Die Realisierung der Konzepte für die aktive Ontologie „Bahn“ wird in Abbildung 5.6 gezeigt. Dabei sind die Knoten, die dieselbe Implementierung besitzen, so dargestellt, dass nur ein Knoten im Einzelnen graphisch repräsentiert wird und der andere neben diesem Knoten mit heller Farbe steht.

5.1.2.4. Aktive Ontologie: Hotel

Die Konzepte der aktiven Ontologie „Hotel“ werden ebenfalls durch die Klassen, die bei „Flug“ und „Bahn“ verwendet werden, implementiert. Das zeigt unter anderem, wie diese Klassen bzw. dieser Entwurf einfach wiederverwendet und erweitert werden kann.

Die Konzepte werden wie folgt realisiert:

- *Arrival date/ Departure date*: wiederholt die Implementierung von „Departure date“ und „Return Date“ in der Ontologie Flug.
- *Adult/ Children/ Rooms*: werden wie „Adult“ und „Children“ in der Flug realisiert.
- *Places*: ist ein Knoten von der Klasse „VocabularyListSensorConceptLeafNode“. Er erkennt einen Ort und nimmt ihn direkt als einen Wert für das Konzept „places“. Die Liste der Orte für das Konzept „places“ ist die Liste der Städte, die bei der Flugontologie verwendet wurden. Aber es können mehrere Städte zu dieser Liste hinzugefügt werden. So ist es nicht notwendig, dass die Stadt des Hotels einen Flughafen hat. Der Knoten gehört auf dieselbe Weise der Wort- und Konzeptphase.
- *Promotion Code*: Der Knoten in diesem Fall wird nur durch die Semantik erkannt. Ein Beispiel hierfür ist es, wenn die Benutzeranfrage so aussieht: „I want to use my promotion code XYZ“ oder „XYZ is the code“ Hier ist es wichtig, die Wörter (is the code, promotion code) herauszufiltern. „promotion code“ ist ein Prätfix im ersten Beispiel, daher wird das nächste Wort „XYZ“ dem Konzept „Promotion code“ zugewiesen. Im zweiten Beispiel ist „is the code“ ein Postfix und das vorherige Wort ist der Wert des Konzept „Promotion Code“.

Abbildung 5.7 stellt die Implementierung für die Konzepte von der aktiven Ontologie „Hotel“ dar. Der Knoten „places“ mit dem besonderen Hintergrundmuster gehört zu der Wort- und Konzeptphase.

5.2. Dialog-Manager

Im Folgenden werden der Entwurf sowie die Implementierung des Dialog-Managers erläutert. Als Erstes wird das Entwurfsmuster des Dialog-Managers beschrieben, anschließend wird erklärt, wie dieser Entwurf implementiert wurde.

5.2.1. Entwurf des Dialog-Managers

Der Dialog-Manager führt einen Dialog mit dem Benutzer durch, sobald notwendige Informationen für die Ausführung der Nutzeranfrage fehlen oder das System oder der Dienstleister dem Nutzer weitere Optionen anbieten. Dabei ist es wichtig zu wissen, wann eine Frage an den Nutzer gestellt wird, wie nach den fehlenden Informationen gefragt wird und wie die Antworten des Benutzers empfangen werden.

Die aktive Ontologie überprüft durch die Verarbeitung der Benutzeranfrage jedes Konzept danach, ob es obligatorisch ist. Dies geschieht, weil die Informationen über die Obligatorik jedes Konzepts bei der Erstellung der aktiven Ontologie definiert wurden. Ist das Konzept obligatorisch und wurde ihm kein Wert zugewiesen, wurde ein Fakt im Faktenspeicher geschrieben. Dieser enthält die folgenden Informationen:

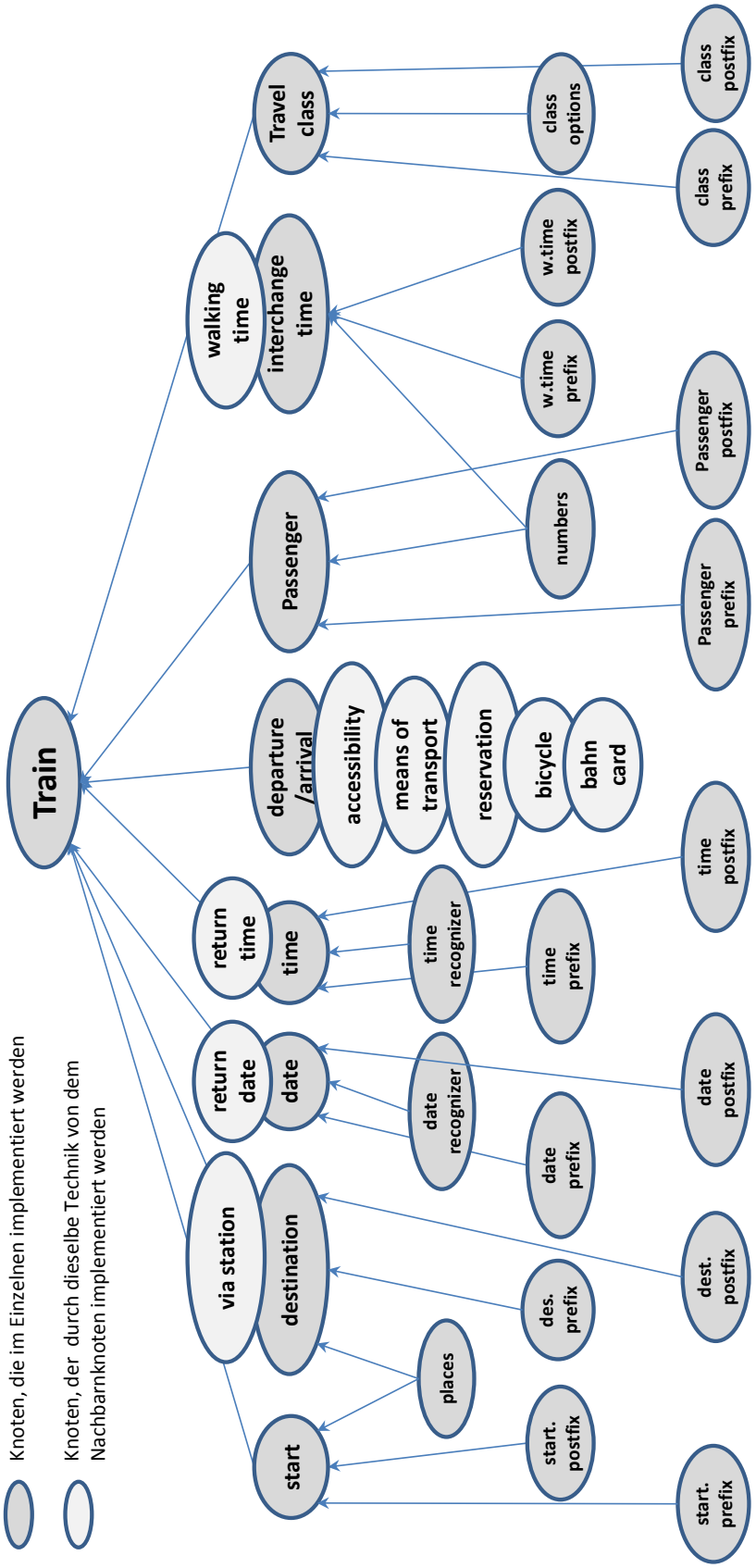

Abbildung 5.6.: Implementierung der Konzepte für die aktive Ontologie: Bahn

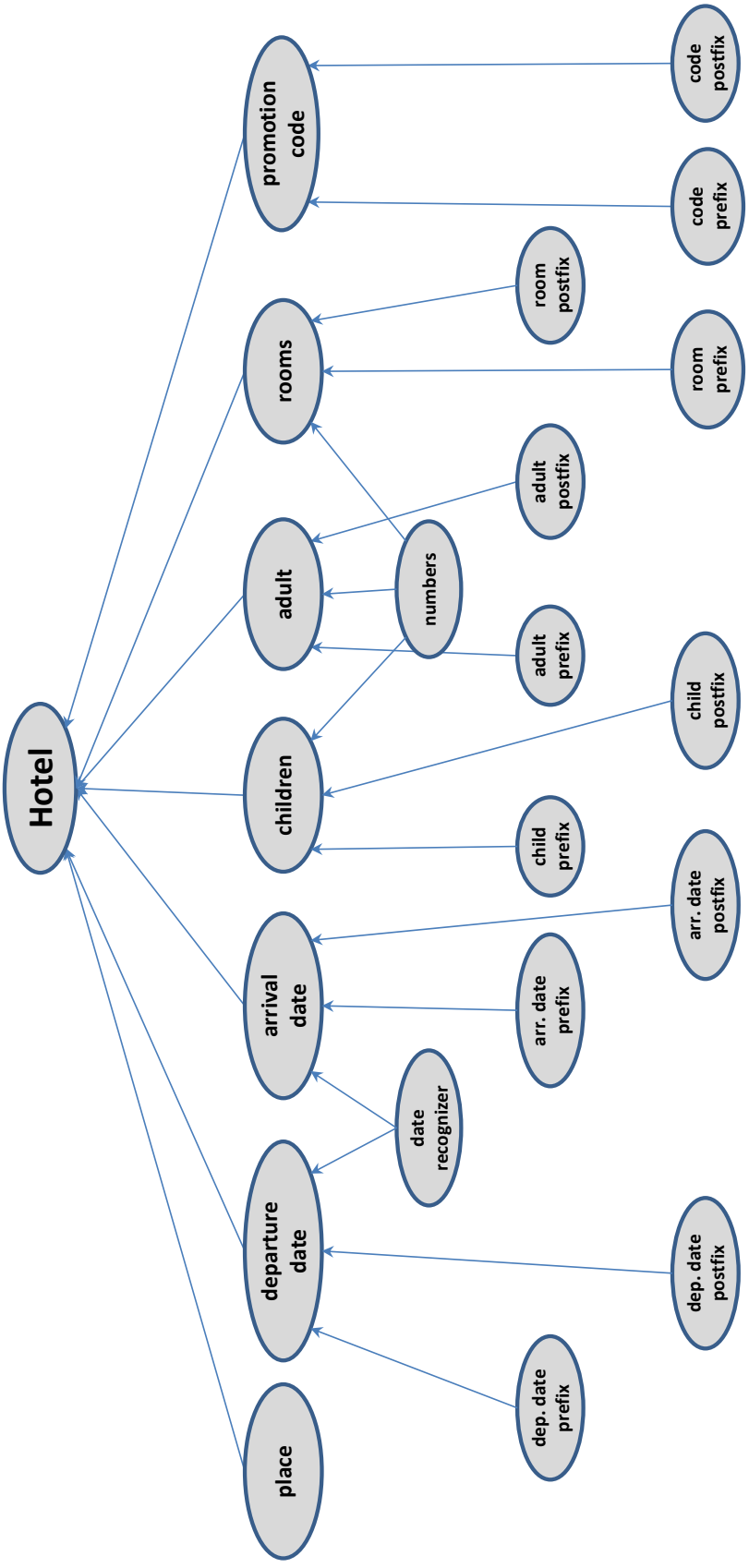

Abbildung 5.7.: Implementierung der Konzepte für die aktive Ontologie: Hotel

- Die aktive Ontologie: Zu welcher aktiven Ontologie gehört das Konzept, das vom Benutzer nicht erfüllt wurde?
- Der Konzeptname: Wie wird das Konzept bezeichnet?
- Die Priorität: Wie wichtig ist das Konzept im Vergleich zu den anderen obligatorischen Konzepten? Die Priorität entscheidet die Reihenfolge der Fragen an den Benutzer. Das ist der Fall, wenn z.B. die Konzepte „Arrival“ und „Arrival date“ fehlen. „Arrival“ hat höhere Priorität als „Arrival date“, weil es in den Formularen der Fluggesellschaften immer vor „Arrial date“ steht. Daher wird dem Benutzer die Frage nach dem „Arrival“ vor dem „Arrival date“ gestellt (siehe Abschnitt 4.2.1).
- Der Antworttyp: Wird die Antwort des Nutzers in ein Eingabefeld eingegeben oder in einem Kalender markiert oder aus einer Liste ausgewählt?
- Antwortsemantik: Es wurde in Abschnitt 5.1.2.2 gezeigt, dass jedes Konzept mithilfe seiner Semantik erkannt wurde. Deswegen wird ein Ausdruck von der Semantik des fehlenden Konzepts hinzugefügt, um bei der Zuweisung der Nutzerantwort des entsprechenden Konzepts zu helfen: z.B. für das Konzept „Departure“ wird der Ausdruck „from“ verwendet. Die Semantik für alle Konzepte wird aus den möglichen Präfixen des Konzepts ausgewählt, um allen Antworten die gleiche Verarbeitung zukommen zu lassen.

Alle diese Informationen werden im Fakt „missing_mandatory“ geschrieben. Wenn das Konzept „Departure“ bspw. in der aktiven Ontologie Flug fehlt, sieht der Fakt wie folgt aus:

missing_mandatory(ActiveOntology(Flight), Name(Departure), Priority(1),
AnswerType(TextField), PrefixAnswer(from)).

Diese Fakten werden von den bereits definierten aktiven Ontologien „Flug“, „Bahn“ oder „Hotel“ gespeichert. Wurde ein „missing_mandatory“-Fakt im Faktenspeicher geschrieben, feuert ein neuer Evaluationszyklus (siehe Abschnitt 2.2), bei dem diese Fakten von den Knoten der aktiven Ontologie des Dialog-Managers empfangen und verarbeitet werden. Die aktive Ontologie des Dialog-Managers besteht aus zwei Knoten. Der eine ist für die Verarbeitung der Fakten „missing_mandatory“ zuständig, während der andere für „missing_optional“-Fakten verantwortlich ist. In diese beiden Knoten werden die Informationen, die in den Fakten „missing_mandatory“ und „missing_optional“ geschrieben wurden, extrahiert.

Die Abbildung 5.8 zeigt den Entwurf der aktiven Ontologie des Dialog-Managers bzw. die Knoten „Mandatory“ und „Optional“, die die Informationen extrahieren und dem Elternknoten „DialogCommand“ weitergeben. „DialogCommand“ führt eine Aktion aus, die diese Informationen zu interaktiven Fragen konvertiert und mit den entsprechenden Antwortoptionen, wie z.B. den Kalender für das Datum, dem Benutzer anzeigt.

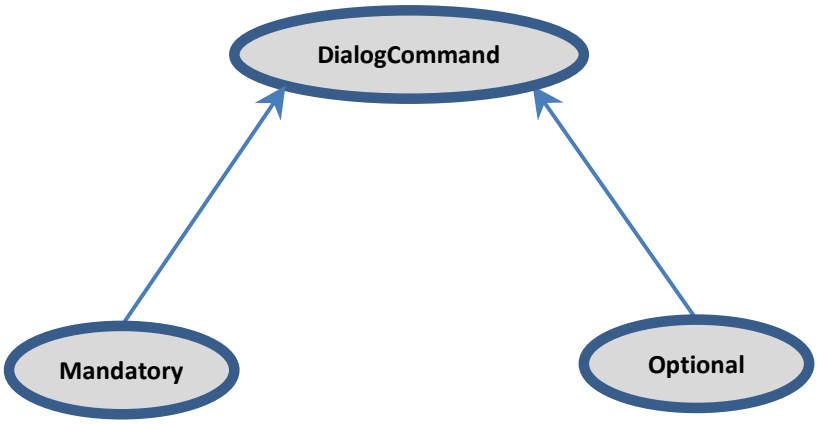

Abbildung 5.8.: Aktive Ontologie des Dialog-Managers

Dem Nutzer werden dementsprechend eine Benutzeroberfläche mit allen Fragen und Antwortfeldern präsentiert. Der Nutzer kann nun die Fragen beantworten, indem er die fehlenden Informationen eingibt.

Die Nutzerantworten müssen dabei zuerst verarbeitet werden, um sie mit der entsprechenden Semantik zu verbinden. Das ist bei der Nutzerantwort „Berlin“ für die Frage nach „Departure“ der Fall. „Berlin“ muss mit der Semantik „from“ verknüpft werden (from Berlin), um die Flugontologie „Berlin“ mit höherer Konfidenz zu erkennen. Die Semantik ist besonders wichtig, wenn zwei oder mehrere Konzepte, die dem gleichen Typ entsprechen, fehlen. Ein Beispiel hierfür wäre, wenn „Departure“ und „Arrival“ fehlten und der Benutzer „Berlin“ und „Karlsruhe“ antworten würde. Ohne diese Antworten mit der Semantik „from Berlin to Karlsruhe“ zu verknüpfen, würde die aktive Ontologie die beiden Antworten nur als Orte erkennen. Es würden die Konzepte „Departure“ und „Arrival“ diesen Orten nicht zugewiesen, weil hierfür keine Semantik vorhanden wäre. Die Antworten mit der Semantik werden wieder in der Flugontologie eingesetzt und normal bearbeitet.

Der gleiche Prozess wird für die fehlenden Informationen durchgeführt, die zu optionalen Konzepten gehören. Diese werden als „missing_optional“-Fakten gespeichert.

Dies erfordert den Entwurf von drei Komponenten: Die erste beinhaltet die Informationen über die fehlenden Konzepte. Die zweite konvertiert diese Informationen zu interaktiven Fragen an den Benutzer und verarbeitet seine Antworten. Die letzte zeigt dem Nutzer die Fragen an und wie er darauf antworten kann. Dieser Entwurf stimmt mit dem Muster „Modell-Präsentation-Steuerung“, auf englisch „Model View Controller“ (MVC), überein. Dabei stellt das „Model“ die aktive Ontologie des Dialog-Managers dar, welche die für die Erstellung der Frage benötigten Informationen aus den Fakten extrahiert und sie an den „Controller“ schickt. Der „Controller“ konvertiert sie zu Fragen und organisiert, wie die Fragen vom Benutzer beantwortet werden können. Das „View“ zeigt dem Benutzer die Fragen mit den Formularfeldern an, die für jede Frage geeignet sind. Dieser Entwurf wird in Abbildung 5.9 dargestellt.

Das „Model“ speichert im ersten Schritt die Informationen über die Konzepte „Departure“ und „Departure date“. Diese Informationen sind der Name, die Priorität, der Antworttyp und das Antwortpräfix. Im zweiten Schritt konvertiert der „Controller“ diese Informationen zu Fragen wie „Where do you want to fly from?“ für „Departure“ und sortiert diese Fragen

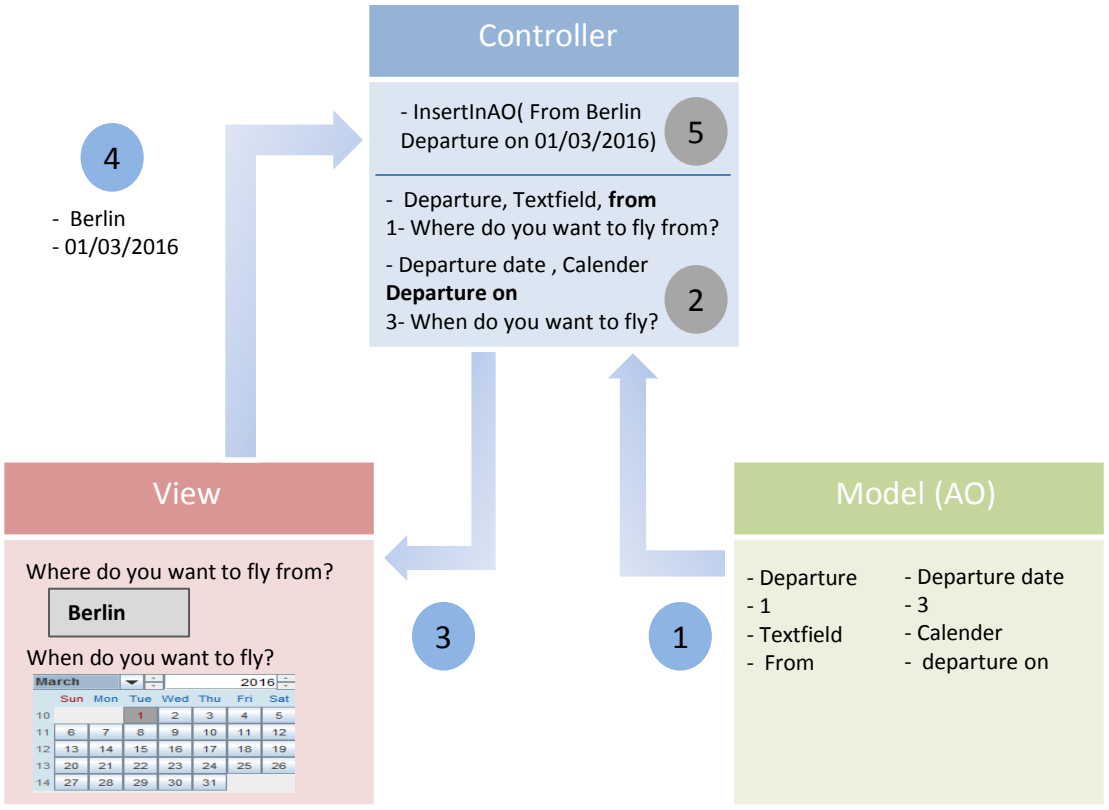

Abbildung 5.9.: Entwurf des Dialog-Managers als Model View Controller: Schritte des Arbeitsprozesses. Die Anfrage „Book me a flight to Karlsruhe“

nach deren Priorität. Das „View“ im dritten Schritt zeigt dem Nutzer die Fragen mit den geeigneten Feldern wie dem Kalender für „Departure date“ an. Der Benutzer gibt seine Antworten „Berlin“ und „13/03/2016“ im vierten Schritt ein. Diese Eingaben werden mit den entsprechenden Präfixen durch den „Controller“ im letzten Schritt zusammengesetzt als „From Berlin Departure on 13/03/2016“ und in der aktiven Ontologie eingesetzt.

Im Code stellen die Klassen „ActiveServerController“, „ActiveServerView“ und „InfosForViewDM“ den „Controller“, das „View“ und das „Model“ dar. Abbildung 5.10 zeigt das entsprechende Klassendiagramm.

Abbildung 5.10.: Klassendiagramm: Dialog-Manager

5.2.2. Implementierung des Dialog-Managers

Nun werden die Methoden, die bei dem Entwurf des Dialog-Managers verwendet werden, beschrieben. Bei dem Entwurf wurde das Muster MVC verwendet und die Aufgaben jeder Komponente des Musters zusammengefasst. Bei der Implementierung werden diese Aufgaben aus technischer Sicht dargestellt.

5.2.2.1. Erkennung der fehlenden Informationen

Jeder Fakt „missing_mandatory“ oder „missing_optional“, der von einer aktiven Ontologie geschrieben wurde, verursacht die Feuerung eines neuen Evaluationszyklus, wobei die Aktionen „CheckMandatoryActionDM“ und „CheckOptionalActionDM“ in der aktiven Ontologie des Dialog-Managers gefeuert werden. Diese Aktionen sind mit den Knoten „Mandatory“ und „Optional“ verbunden (siehe Abbildung 5.8). Jede Aktion beinhaltet die Methode *fire()*, die die Fakten „missing_mandatory“ oder „missing_optional“ als Eingabe erhält. Sodann werden diese Informationen an den Knoten „DialogCommand“ durch die

Methode *sendToAllConnectedNodes()* geschickt. „DialogCommand“ bekommt diese Informationen und speichert die Informationen jedes Faktes in einem „MissingInfos“-Objekt. Jedes Objekt repräsentiert ein fehlendes Konzept. Diese Objekte werden in einer „Map“ zusammengesetzt. Nun wird der „Controller“ aufgerufen wie in Abbildung 5.9, der die Map der „MissingInfos“-Objekte als Parameter erhält und der mit der Erstellung der Fragen beginnt.

5.2.2.2. Die Erstellung der Fragen

Die Fragen für alle Konzepte werden in einer „Props-Datei“ gespeichert, die in der Klasse „QuestionProperties“ erstellt wurde. Jedes „Property“ ist durch ein (Schlüssel, Wert)-Paar dargestellt. „Key“ ist hier der Konzeptname und „Value“ ist die Frage, die erstellt werden muss, wenn das entsprechende Konzept fehlt. Ein Beispiel hierfür ist die Frage „Where do you want to fly from?“ für das Konzept „Departure“. Der „Controller“ erhält in der Methode *prepareInfoForView()* den Konzeptnamen als Parameter. In dieser Methode wird die entsprechende Frage aus der Datei „Question.Properties“ aufgerufen. Der Konzeptname und die Frage werden so zusammengesetzt, dass der Nutzer das fehlende Konzept mit der Frage erhält. Als Beispiel hierfür wird dem Nutzer „Departure: Where do you want to fly from?“ angezeigt, wenn das „Departure“ fehlt. Dieser Satz wird dem Attribut „Label“ des Objekts „InfosForViewDM“ zugewiesen. Der Konzeptname wird angezeigt, falls keine Frage für das Konzept verfügbar ist, wie z.B. wenn die aktive Ontologie automatisch aus einem Formular generiert wird. Hierbei kann es jedoch der Fall sein, dass keine Frage für ein Konzept generiert werden kann. Der Konzeptname gibt dabei dem Nutzer Hinweise über die fehlenden Informationen.

Im Objekt „InfosForViewDM“ werden für jedes fehlende Konzept die folgenden Daten gespeichert:

- Label: Dabei handelt es sich um den Text, der dem Nutzer angezeigt wird. Dieser Text beinhaltet den Konzeptnamen und die entsprechende Frage.
- „answerType“: Dieses Attribut repräsentiert das geeignete Formularfeld für das fehlende Konzept, wie z.B. das Eingabefeld oder den Kalender.
- „prefixAnswer“: Hier wird das Präfix gespeichert, das für die Erkennung der Nutzerantwort in der aktiven Ontologie verwendet wird (Abschnitt 4.2.2).

Die Attribute „answerType“ und „prefixAnswer“ beziehen ihre Werte aus dem Parameter „Map“. Im Falle von „Departure“ haben die Attribute „answerType“ und „prefixAnswer“ die Werte „Textfield“ und „from“.

Anschließend wird eine List der Objekte „InfosForViewDM“ erstellt und das „View“ aufgerufen.

5.2.2.3. Das Anzeigen der Frage für den Nutzer

Das „View“ erhält die Liste der Informationen, die dem Nutzer dargestellt werden sollen. Das View erstellt zunächst ein „Frame“ mit dem Titel „Required information“ für obligatorische Informationen, worauf alle Formularelemente sichtbar sind. Das erste Feld dieses Formulars ist ein „Label“ mit dem Text „Hello Sir! Would you please enter the following required Information?“.

Daraufhin wird die Liste der Objekte „InfosForViewDM“ wie folgt durchiteriert:

- Für jedes „Label“ wird ein „JLabel“ mit dem Text in diesem Attribut erstellt.
- Für jedes „answerType“ wird das entsprechende Feld so erzeugt, dass ein „JTextField“ für „Textfield“ entsteht, ein „JComboBox“ für ein „List“ und ein „JDateChooser“ für ein „Calender“.

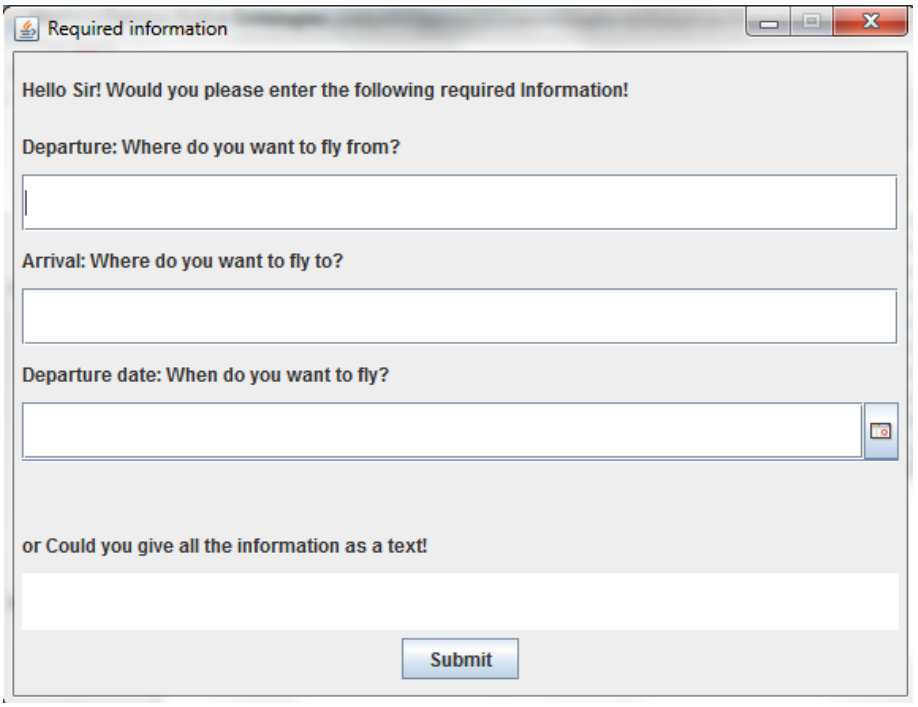

Abbildung 5.11.: Benutzeroberfläche des Dialog-Managers

Nach der Erstellung der Felder dieser Informationen werden ein „JLabel“ und ein „JTextArea“ zusätzlich erzeugt. Sie werden verwendet, wenn der Nutzer seine Antworten sprachlich (als Text) eingeben möchte. Das Label zeigt den Text „or could you give all the information as a text!“ an. Der Text wird dann vom Nutzer im „JTextArea“ eingegeben.

Ein „Submit“-button wird ebenfalls erstellt, um diese Informationen abzusenden.

Die Abbildung 5.11 stellt die Benutzeroberfläche des Dialog-Managers dar, woraufhin die Fragen sowie die Antwortmöglichkeiten der Konzepte „Departure“, „Arrival“ und „Departure date“ angezeigt werden. Der Nutzer hat an dieser Stelle keine Informationen über diese Konzepte während seiner erste Anfrage über die Flugbuchung eingegeben. In dieser Abbildung werden die „Labels“ dargestellt, die das fehlende Konzept sowie die Frage enthalten. Dabei werden die Eingabefelder für „Departure“ und „Arrival“ und ein Kalender für „Departure date“ ebenfalls angezeigt. Die Möglichkeit, dass der Nutzer die Antworten als Text eingeben kann, wird durch „TextArea“ mit dem entsprechenden Label dargestellt.

5.2.2.4. Die Verarbeitung der Antworten

Der Nutzer gibt seine Antworten entweder in den Formularfeldern oder als Text ein. Die Antworten aus den Feldern werden mit den Werten, die im Attribut „prefixAnswer“ des Objekts „InfosForViewDM“ gespeichert sind, verknüpft. Das ist der Fall, wenn der Nutzer „Berlin“ im Feld „Departure“ eingibt, sodass „Berlin“ mit dem Präfix „from“ zusammengesetzt wird. Der Satz „from Berlin“ wird dann in der aktiven Ontologie „Flug“ wieder eingesetzt. Die Verknüpfung mit dem Präfix ist notwendig, weil „Berlin“ alleine bspw. in der aktiven Ontologie als Ort erkannt wird. „Berlin“ wird dann an die beiden Konzepte „Departure“ und „Arrival“ gesendet, sodass nicht entschieden werden kann, zu welchem Konzept „Berlin“ gehört. Aus diesem Grund wird die Semantik „from“ verwendet, um „Berlin“ dem „Departure“ zuzuweisen.

Wenn der Nutzer seine Antworten als Satz eingibt, wird die Antwort direkt in die aktive Ontologie eingesetzt. Das ist ungefähr so, wie wenn man auf die Rückfrage des intelligenten Assistenten mit einem Satz antwortet.

Der Entwurf und die Implementierung ist für die optionale Informationen genauso, wie für die obligatorischen. Es werden nur kleine Änderungen durchgeführt, um dem Nutzer Hinweise zu geben, dass er auf diese Fragen nicht antworten muss. Der Nutzer erhält zunächst die Frage „There are more optional informations, that can help you finding the best matching result. Would you like to enter some of them?“ sowie den Bestätigungs- und Abbruchsbutton. Bei einer Bestätigung wird der Dialog durchgeführt, während bei einem Abbruch keine Frage angezeigt wird. Der Titel des Fensters zeigt ebenfalls den Text „Optional information for more matching results! “ an.

Der Dialog über weitere optionale Informationen wird erst durchgeführt, wenn alle obligatorischen Informationen erfüllt sind.

5.3. Formularaufruf

Die Suche in den Webformularen ist die Aktion, die vom „Commandknoten“ der aktiven Ontologie ausgeführt wird. Diese Ausführung erfordert, dass die Formulare aus der in der aktiven Ontologie erkannten Informationen ausgefüllt werden müssen, um nach der Nutzerpräferenz zu suchen und dem Nutzer die Suchergebnisse zu zeigen. Der Entwurf und die Implementierung dieser Aufgabe wird im Folgenden genauer vorgestellt.

5.3.1. Entwurf des Formularaufrufs

Nach der Erkennung der obligatorischen Informationen in der aktiven Ontologie kann die Suche in den Webformularen beginnen. Dies benötigt jedoch eine Vorverarbeitung der erkannten Informationen. Die Informationen stehen in dem „Commandknoten“ der aktiven Ontologie, wie z.B. „Karlsruhe“, „Manchester“ und „13/03/2016“ als „Departure“, „Arrival“ und „Departure date“. „Karlsruhe“, „Manchester“ und „13/03/2016“ sind die Informationen, die alle Webformulare der Fluggesellschaften ausfüllen. Nun wird versucht, diese Werte in den Formularen einzugeben. Ein Beispiel hierfür ist das Formular von Lufthansa. Wenn der Nutzer diese Informationen abschickt, wird die Suche nicht durchgeführt, sondern er erhält die Meldung „Destination multiple matches“. „Manchester“ kann zwei verschiedene Ziele darstellen „Manchester, MAN, United Kingdom“ und „Manchester, MHT, USA“. Hierbei muss der Nutzer gefragt werden, welches „Manchester“ gemeint ist. Solche Möglichkeiten werden in einer „Props-Datei“ gespeichert. Wenn „Manchester“ aus dieser Datei aufgerufen wird und mehrere Optionen zurückgegeben werden, werden dem Nutzer diese Möglichkeiten angezeigt, damit er seine gewünschte Stadt auswählen kann. In dieser Arbeit werden nur bestimmte Orte als Testfälle hinzugefügt. „Manchester“ ist einer der Orte, der in dieser Arbeit als die Stadt in Großbritannien erkannt wird. Deswegen wird der Nutzer nicht mehr gefragt, zu welchem „Manchester“ er reisen möchte. Die Option „Großbritannien“ muss automatisch ausgewählt werden. Das erfordert, dass „Manchester“ als „Manchester, MAN, United Kingdom“ dargestellt wird. Wenn der Wert „Manchester, MAN, United Kingdom“ im Feld Ankunftsort eingegeben wird, wird die Suche problemlos durchgeführt und werden die Ergebnisse dargestellt.

Im Formular des „British Airways“ muss der Nutzer das Land „Großbritannien“ auswählen, dann kann er „Manchester“ aus einer Liste von allen Flughäfen in Großbritannien, die von „British Airways“ bedient werden, auswählen. Aus diesem Grund müssen diese Informationen für jedes einzelne Formular vorverarbeitet werden. Die Darstellung der Städte wie z.B. „Karlsruhe/Baden-Baden“ für Karlsruhe beim Lufthansa-Formular wird ein einer „Props-Datei“ für jede Fluggesellschaft gespeichert und für jede erkannte Stadt und aus den entsprechenden „Props-Datei“ der Fluggesellschaft aufgerufen.

In dieser Arbeit werden die Ergebnisse als ein Dialog mit dem Benutzer so dargestellt, dass der Nutzer einen Satz wie z.B. „You have the following Options! Would you like to choose

your preferred one?“ mit den gefundenen Ergebnisse erhält. Hierbei ist der Nutzer ebenfalls in der Lage, weitere Informationen zu sehen, die jede Fluggesellschaft anbieten kann, um den Flug, der mit seinen Präferenzen übereinstimmt, auszuwählen. Diese Informationen sind Sitzplatzreservierung, Anzahl der Gepäckstücke, Mahlzeit, Getränke usw. Dann kann der Benutzer einen Flug auswählen und buchen.

Die Ergebnisse aus allen Webseiten werden dem Benutzer zusammen angezeigt. Deswegen werden diese Informationen in einer vereinigten Datenstruktur für die jeweilige Domäne gespeichert.

Dieser Prozess kann in drei Schritte zusammengefasst werden. Im ersten werden die Informationen aus der „CommandKnoten“ der aktiven Ontologie in einer Datenstruktur gespeichert. Die Suchergebnisse aus allen Webseiten einer Domäne werden ebenfalls in einer anderen Datenstruktur gespeichert. Diese Informationen werden im zweiten Schritt konvertiert, damit sie mit der Formatierung der Felder jedes Webformulars übereinstimmen. Die Formularfelder werden sodann ausgefüllt und abgeschickt und die Suchergebnisse werden aus den Webseiten der Dienstleister herausgesucht. Im dritten Schritt werden die Ergebnisse dem Nutzer als Dialog präsentiert, um die gewünschten Ergebnisse zu buchen.

Diese Schritte werden auf das Muster Model-View-Controller (MVC) abgebildet. Das „Model“ schreibt die Informationen in einer Datenstruktur um, die in der aktiven Ontologie erkannt werden sowie die Ergebnisse aus den Webseiten einer Domäne in einer anderen Datenstruktur. Der „Controller“ konvertiert diese Informationen zu der geeigneten Formatierung, füllt das Formular aus und schickt es ab. Außerdem ist der „Controller“ für das Extrahieren der Suchergebnisse aus den Webseiten zuständig. Das „View“ zeigt dem Benutzer die Ergebnisse an, und ermöglicht es ihm, seine Lieblingsergebnisse auszuwählen.

Der Entwurf des Formularaufrufs als Model-View-Controller wird in Abbildung 5.12 schrittweise dargestellt. Das „Model“ im ersten Schritt speichert die erkannten Informationen und gibt sie dem „Controller“, der diese Informationen im zweiten Schritt bearbeitet, um mit der Formatierung jeder Webseite übereinzustimmen. Das „View“ zeigt dem Nutzer im letzten Schritt die Ergebnisse an.

5.3.2. Implementierung des Formularaufrufs

In diesem Abschnitt wird die Implementierung für den oben erläuterten Entwurf dargestellt. Dazu gehört die Klasse und die Methode, die verwendet werden, um die Formulare auszufüllen und die Ergebnisse zu sammeln.

5.3.2.1. Speicherung der Daten aus der aktiven Ontologie

Die aus der sprachlichen Eingabe des Benutzers erkannten Informationen werden in „CommandKnoten“ der aktiven Ontologie zusammengeführt. Diese Informationen sind die Ressourcen aller Webformulare einer bestimmten Domäne und müssen daher gespeichert werden. Eine Klasse für jede Domäne wird für diese Aufgabe definiert, wie z.B. „FlightFormInfos“ für Flugformulare und „TrainFormInfos“ für Bahn und „HotelFormInfos“ für Hotel. Diese Klassen entsprechen den Konzepten der aktiven Ontologie bzw. den Feldern der Webformulare. Als Beispiel hierfür beinhaltet die Klasse „FlightFormInfos“ die Attribute „departure“, „departureDate“, „departureTime“, „arrival“, „returnDate“, „flightClass“, „adultNumber“, „childrenNumber“ und „babiesNumber“. Diese Attribute werden durch die Methode *getTheValue()* erfüllt. Diese Methode ruft je nach aktiver Ontologie „Flug“, „Bahn“ oder „Hotel“ die entsprechende Methode auf. Beim Flug z.B. gibt die Methode *getTheFlightConceptValue()* den Wert jedes Konzepts der aktiven Ontologie Flug zu dem entsprechenden Attribut des Objekts von „FlightFormInfos“. Auf diese Art und Weise werden die Informationen in diesem Objekt gespeichert. Dies stellt das „Model“ im Muster „MVC“ dar.

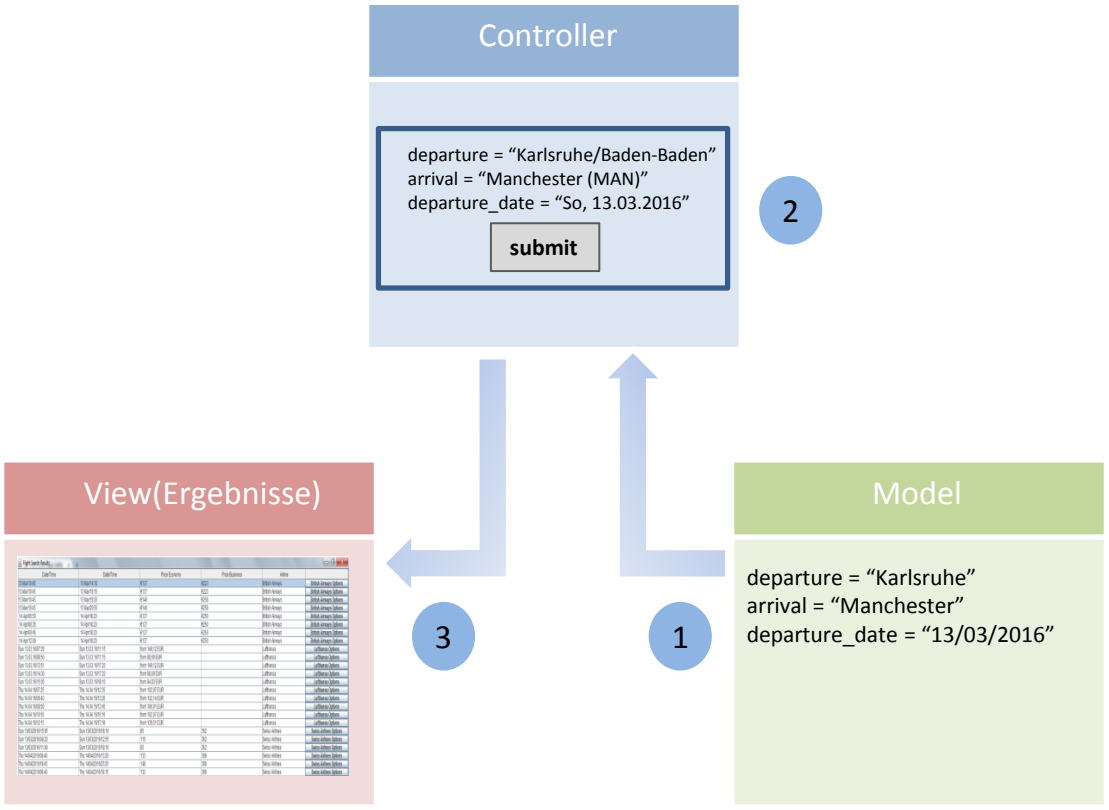

Abbildung 5.12.: Entwurf des Formularaufrufs als Model View Controller: Schritte des Arbeitsprozesses

5.3.2.2. Verarbeitung der Informationen

Wie bereits erwähnt wurde, müssen die Informationen für jedes Webformular verarbeitet werden. Deswegen wird für jedes Formular eine einzelne Klasse definiert, wie z.B. die Klasse „Lufthansa“. Diese Klasse beinhaltet die Methoden *prepareFormInfo()*, *connectFillsubmit()* und *collectResults()*.

Die Methode *prepareFormInfo()* bereitet die Informationen vor, um die Formatierung jedes Formulars anzupassen. Als Beispiel dafür konvertiert diese Methode in der Klasse „Lufthansa“ das „Departure date“ und das „Arrival date“ von bspw. „13/03/2016“ zu „So, 13.03.2016“. Die Orte im jeweiligen Webformular werden in einer „Props-Datei“ gespeichert. Beispielsweise ist „Stuttgart“ in der „BritishAirways.properties“ ein „Key“ für „Stuttgart, Stuttgart (STR), Germany“; wenn der Abflugort oder Ankunftsort „Stuttgart“ ist, ruft die Methode *prepareFormInfo()* den Wert „Stuttgart, Stuttgart (STR), Germany“ aus der „BritishAirways.properties“ durch das „Key“ „Stuttgart“ auf. Für Städte wie „Manchester“, die mehrere Optionen darstellen, wird für das „Key“ eine Liste aller Optionen wie (MAN, MHT) gespeichert. Wenn diese Liste mehr als ein Element hat, dann wird der Benutzer gefragt, um eine einzige Option auszuwählen.

Die Methode *connectFillsubmit()* verwendet das Werkzeug „Selenium“, um die Seite des Webformulars aufzurufen und die Felder des Formulars durch ihre „Id“, „Name“, „Xpath“ oder den „cssSelector“ zu finden. Der „Name“, „Id“, „Xpath“ sowie „cssSelector“ aller Felder des Webformulars werden ebenfalls in der „Props-Datei“ zuvor gespeichert und an dieser Stelle aufgerufen. Jedes Feld wird dann mit den entsprechen Wert ausgefüllt und dann das Formular abgeschickt.

In Quelltextausschnitt 5.2 werden Beispiele für das Aufrufen einer Seite, das Finden und Ausfüllen der Formularfelder und das Abschicken des Formulars dargestellt. In der ersten Zeile ist der Seitenaufruf des Lufthansa-Formulars zu erkennen. Der Name des Feldes „Departure“ wird in der zweiten Zeile gefunden und in der dritten mit dem Wert für „Stuttgart“, der in der „Props-Datei“ gespeichert wurde, ausgefüllt. Das gesamte Formular wird in der letzten Zeile abgeschickt [6].

Quelltextausschnitt 5.2: Formular durch Selenium ausfüllen

```
1 driver.navigate().to(properties.getProperty("Lufthansa_URL"));
2 WebElement depPlace = driver.findElement(By.name(properties.
      getProperty("DepPlace")));
3 depPlace.sendKeys(properties.getProperty("Stuttgart"));
4 driver.findElement(By.id(properties.getProperty("FormLH")).submit();
```

Nach dem Abschicken des Formulars sammelt die Methode *collectResults()* die Suchergebnisse. Um die Ergebnisse zu speichern, wird eine Klasse für jede Domäne definiert, wie z.B. „FlighResult“ für die Ergebnisse des Flugs. Die Attribute dieser Klasse sind die Informationen, die der Benutzer benötigt, um die Buchung abzuschließen: z.B. hat die Klasse „FlightResult“ die Attribute „departureTime“, „departureDate“, „arraivalTime“, „arrivalDate“, „priceEconomy“, „priceBusiness“ und „airline“.

Die Klassen „FlightFormInfos“, „FlightResult“, „SwissAirlines“, „BritishAirways“ und „Lufthansa“ werden in einem Klassendiagramm in Abbildung 5.13 dargestellt. Die Klassen „FlightResultApp“ und „FlightWebScrapingThread“ werden in Abschnitt 5.3.2.4 beschrieben.

Dabei sammelt „Selenium“ die Informationen der Ergebnisse. Zunächst werden die „WebElemente“ durch den „Name“, „Xpath“ usw. gefunden, und der Wert dabei ausgelesen.

[6] „By“ ist eine Klasse von Selenium Tabelle E.11

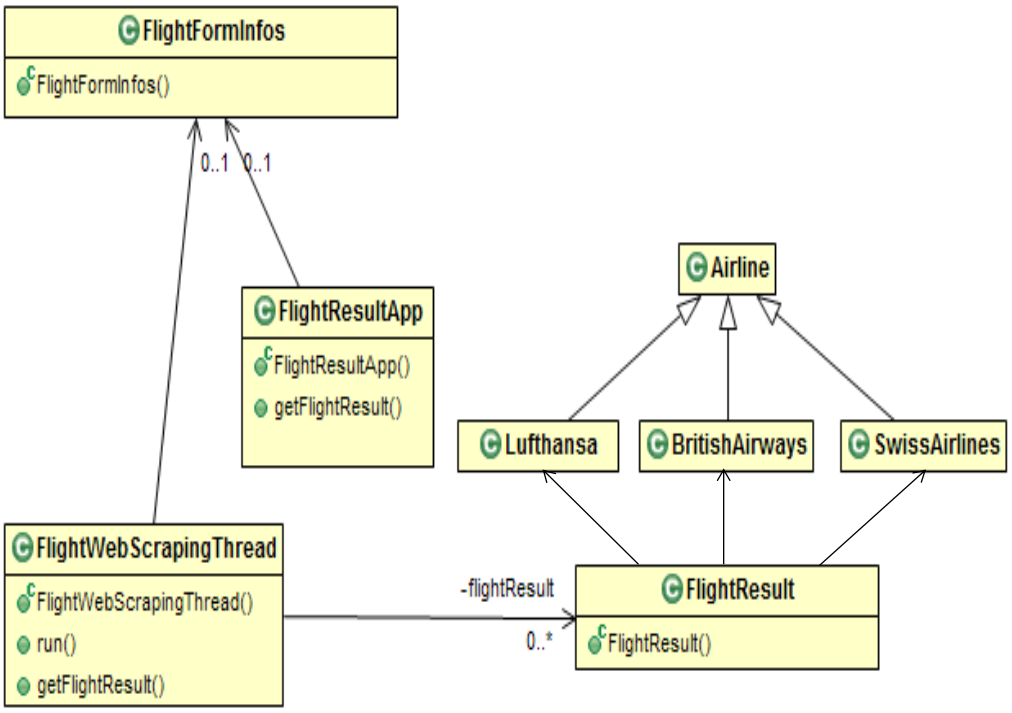

Abbildung 5.13.: Klassendiagramm: Die Suche nach Flügen in den Fluggesellschaften

wie in Quelltextausschnitt 5.3. Die erste Zeile wird das „WebElement“ „Price“ durch das „XPath“ gefunden. In der zweiten wird der Wert zu dem Attribut „priceEconomy“ des Objekts von „Flightresult“ zugewiesen.

Quelltextausschnitt 5.3: Formular durch Selenium ausfüllen

```
1 List<WebElement> ecoPrices = driver.findElements(By.xpath(properties.
      getProperty("EcoPrices")));
2 hinResults.setPriceEconomy(ecoPrices.getText());
```

Diese Methode gibt eine Liste der Objekte der Klasse „FlightResult“ zurück, die alle in der Webseite gefundene Flüge, die die Präferenzen des Benutzers treffen, darstellt.

Alle diese Aufgaben gehören zu dem „Controller“, außer die Klasse „FlightResult“, die dem „Model“ angehört.

5.3.2.3. Darstellung der Ergebnisse

Die Liste der Suchergebnisse wird durch die Methode *createFlightResultView()* dem Benutzer angezeigt. Zunächst wird ein „Frame“ erstellt, der den Titel „Flight Search Results“ für Flugergebnisse trägt. Dabei wird eine Tabelle erstellt, deren Spalten die Attribute des Objekts „FlightResult“ widerspiegelt. Die Tabelle wird mit den Werten der Liste von Objekten der „Flightresult“ aufgefüllt.

Die Abbildung 5.14 zeigt die Benutzeroberfläche an, die die Ergebnisse von „Lufthansa“, „British Airways“ und „Swiss Airlines“ beinhaltet. Diese Ergebnisse sind die Antwort der Benutzeranfrage „find me a flight from Stuttgart to Manchester on 13/03/2016 and return on 14/04/2016“.

Ein zusätzlicher Button wird für jedes Resultat erstellt. Durch dessen Anklicken wird ein anderes „Frame“ dargestellt, in dem die Optionen wie Sitzplatzreservierung oder Anzahl

Flight Search Results

Date/Time	Price Economy	Airline	
Sun 13.03.16/07:26	from 149,98 EUR	Lufthansa	Lufthansa Options
Sun 13.03.16/08:50	from 88,95 EUR	Lufthansa	Lufthansa Options
Sun 13.03.16/13:51	from 149,98 EUR	Lufthansa	Lufthansa Options
Sun 13.03.16/14:30	from 88,95 EUR	Lufthansa	Lufthansa Options
Sun 13.03.16/15:05	from 84,88 EUR	Lufthansa	Lufthansa Options
Thu 14.04.16/07:05	from 101,39 EUR	Lufthansa	Lufthansa Options
Thu 14.04.16/08:40	from 130,25 EUR	Lufthansa	Lufthansa Options
Thu 14.04.16/08:50	from 105,33 EUR	Lufthansa	Lufthansa Options
Thu 14.04.16/10:50	from 131,39 EUR	Lufthansa	Lufthansa Options
Thu 14.04.16/12:30	from 105,33 EUR4 seat(s) left for this price	Lufthansa	Lufthansa Options
13 Mar/10:45	€159	British Airways	British Airways Options
13 Mar/10:45	€159	British Airways	British Airways Options
13 Mar/10:45	€177	British Airways	British Airways Options
13 Mar/10:45	€177	British Airways	British Airways Options
14 Apr/06:50	€159	British Airways	British Airways Options
14 Apr/08:35	€159	British Airways	British Airways Options
14 Apr/09:45	€159	British Airways	British Airways Options
14 Apr/12:00	€159	British Airways	British Airways Options
Sun 13/03/2016/15:05	85	Swiss Airlines	Swiss Airlines Options
Sun 13/03/2016/08:20	115	Swiss Airlines	Swiss Airlines Options
Sun 13/03/2016/11:00	85	Swiss Airlines	Swiss Airlines Options
Thu 14/04/2016/08:40	131	Swiss Airlines	Swiss Airlines Options
Thu 14/04/2016/18:45	161	Swiss Airlines	Swiss Airlines Options
Thu 14/04/2016/08:40	131	Swiss Airlines	Swiss Airlines Options

Abbildung 5.14.: Ergebnisse der Bentzeranfrage (find me a flight from Stuttgart to Manchester on 13/03/2016 and return on 14/04/2016)

der Gepäckstücke angezeigt werden. Anbei steht ein Text, der den Benutzer auffordert, eine der Optionen auszuwählen. Dieser Text lautet „This flight has the following Options for you, would you like to choose your preferred option?“ Diese Optionen werden als „RadioButtons“ dargestellt, so kann der Benutzer einen bestimmten Flug auswählen. Die Abbildung 5.15 stellt zusätzliche Optionen des ersten Fluges der Lufthansa dar. Dabei werden drei Flugklassen angezeigt: „Light Economy“, „Economy classic“ und „Economy Flex“. Jede Klasse hat ihre eigenen Angebote und Preise. Diese Aufgabe gehört zum „View“ im Muster „MVC“.

5.3.2.4. Parallelisierung der Suchaufgabe

Viele Formulare bzw. Fluggesellschaften werden gesucht, um die Ergebnisse der verschiedenen Anbieter dem Nutzer anzuzeigen. Werden diese Formulare sequentiell ausgefüllt, dauert dieser Prozess so lange an, bis alle Ergebnisse gesammelt wurden. Das bedeutet, dass der Benutzer so lange warten muss, bis er die Ergebnisse erhält. Aus diesem Grund muss die Aufgabe parallelisiert werden. Dies wird in dieser Arbeit durch die Verwendung von „ThreadPool“ implementiert.

Das „ThreadPool“ wird in der Klasse „FlightResultApp“ definiert. Es führt einen Faden, der in der Klasse „FlightWebScrapingThread“ erstellt wird, für jedes Formular aus. Daher beträgt die Anzahl der Faden immer die gleiche Anzahl der Webformulare. Nach der Beendigung aller Faden werden die Ergebnisse aller Webformulare in einer vereinigten Liste in der Klasse „FlightResultApp“ zusammengetragen und dem „View“ gesendet. Die Klassen „FlightResultApp“ und „FlightWebScrapingThread“ werden in Abbildung 5.13 dargestellt.

5.4. Zusammenfassung

In diesem Kapitel wurde der Entwurf jeder zu definierenden aktiven Ontologie dargestellt. Dadurch wurde die Verarbeitung der sprachlichen Eingabe des Nutzers in drei Phasen gegliedert. Die Knoten, Aktionen sowie die Java-Klassen, die für die Erstellung der aktiven Ontologie notwendig sind, wurden ebenfalls erläutert.

Abbildung 5.15.: Lufthansa-spezifische Optionen

Die Entwürfe von Dialog-Manager und Formularaufruf sind nach dem Model-View-Controller-Architekturmuster entworfen. Dabei hat jede Komponente dieses Musters unterschiedliche Aufgaben. Die Implementierung des Dialog-Managers wurde schrittweise beschrieben, ausgehend von der Erstellung der Fragen bis zum Einsetzen der Antworten in die aktive Ontologie. Der Formularaufruf wurde gleichermaßen implementiert. Dabei wurden die Vorverarbeitung der Informationen, das Ausfüllen der Formulare und die Sammlung und das Anzeigen der Suchergebnisse beschrieben.

6. Evaluation

Bei der Evaluation des Systems liegt der Fokus auf der Erkennung der gewünschten Präferenzen des Benutzers, die er in seiner sprachlichen Anfrage eingegeben hat. Dabei werden Testsätze untersucht, die Buchungsinformationen beinhalten. Die Ergebnisse werden gesammelt und mit dem eigentlichen Vorhaben des Nutzers verglichen.

6.1. Datensatz

Das in dieser Arbeit entwickelte System soll von dem Nutzer so verwendet werden, dass er seine Flug-, Bahn- und Hotelbuchungen sprachlich erledigen kann. Das System wird als gut bewertet, wenn es unterschiedliche Varianten der Nutzeranfragen korrekt versteht und die in der Anfrage beinhalteten Informationen richtig erkennt und den entsprechenden Konzepten zuweist.

Zu diesem Zweck wurde eine Umfrage durchgeführt, in der Nutzer freiwillig und probeweise ihre Buchungen sprachlich bzw. als Texte verschriftlicht erledigen. Als Hilfe haben die einzelnen Probanden Bilder von Buchungsformularen verschiedener Dienstleister erhalten, sodass sie sich eine Vorstellung darüber machen konnten, welche Optionen ihre Anfragen beinhalten können. Um die Umfrage möglichst effizient zu gestalten, wurden den Nutzern dabei erklärt, dass sie die Anfrage für ein Siri-artiges System formulieren und welche Ziele die Umfrage hat, nämlich die Sammlung von möglichst vielen unterschiedlichen Varianten sprachlicher Anfragen. Die Auswahl der Teilnehmer wurde unter verschiedenen Kriterien durchgeführt.

Diese Kriterien werden im Folgenden zusammenfassend aufgelistet.

- Sprachniveau: Wie die Nutzer ihre Anfrage oder ihren Buchungswunsch ausdrücken, hängt von ihrem Englischniveau ab. Daher wurden Teilnehmer mit Englisch als Mutter- und Zweitsprache gebeten, ihre sprachliche Anfrage aufzuschreiben. Beispiele für den Unterschied im Sprachniveau sind Sätze wie *„I am headed London and I would like to fly out of Frankfurt“* von einem Engländer und *„I want to fly from Frankfurt to London“* von einen Benutzer, der Englisch lediglich als Zweitsprache spricht.
- Onlinebuchungserfahrung: Die Nutzer, die viel reisen und ihre Tickets online buchen, geben ihre Präferenzen anders an als diejenigen, die noch nicht gereist oder ihre Ticket noch nie online gebucht haben. Der Satz *„I would like to use my 25% Bahncard*

to book a first class train ticket to Berlin and I want to take my Bicycle with me“ ist z.B von einem Nutzer, der häufig mit der Bahn reist.

- Alter: Junge Nutzer, die ihr Smartphones viel verwenden, tendieren zu der Verwendung kurzer Sätze, die grammatisch nicht unbedingt korrekt sind, aber die Absichten direkt ausdrücken, wie z.B. die Anfrage *„Book me a flight from Stuttgart to Manchester first class 2 adults on 12/01/2016 and back on 20/01/2016“* von einem 22-jährigen Benutzer. Die Nutzer, die sich selten mit den Smartphones beschäftigen, geben am häufigsten lange und fehlerfreie Sätze an. Das ist der Fall in der Anfrage *„I would like to enquire about flights to Berlin from Frankfurt Airport please. I am off to a concert at September 28th until October 4th. Could you tell me about the flight availability and prices.“* von einem 52-jährigen Nutzer.
- Ausbildung: Angestellte, Hochschulabsolventen, Studenten, Schüler und Nutzer, die nicht über die allgemeine Hochschulreife verfügen, haben an der Umfrage teilgenommen. Für dieses Kriterium interessant sind im Besonderen Informatiker und diejenigen, die sich mit der Sprachverarbeitung beschäftigen. Bei den Informatikern, die bei der Umfrage teilgenommen haben, handelt es sich um Teilnehmer eines Praktikums zum Thema „Dialogmodellierung“ am KIT. Ein Beispiel hierfür ist die Anfrage *„I want to do my Christmas shopping in New York, so I want to fly there from Frankfurt and I do not want the flight to be first class“* ist ein Beispiel dafür. Hier wurde der Ankunftsort und Abflugdatum implizit gegeben. Der Satz mit „NOT“ ist ebenfalls ein Sonderfall.

An der Umfrage haben 40 Probanden teilgenommen, bei der sie Anfragen zur Flug-, Bahn- und Hotelbuchung gemacht haben. Die Teilnehmer konnten die drei Fragen der jeweiligen Kategorie beantworten oder ihre Anfrage sofort auf einmal übermitteln. Dies hängt von der Reiseerfahrung des Nutzers ab.

Die Eigenschaften der Probanden und die Häufigkeit davon werden in Tabelle 6.1 dargestellt.

Kriterium	**Kategorie**	**Anzahl**
Sprachniveau	Englisch als Muttersprache	20
	Englisch als Zweitsprache	20
Onlinebuchungserfahrung	Mit Erfahrung	24
	Ohne Erfahrung	16
Alter	15 - 35 Jahre	24
	36 - 45 Jahre	10
	46 - 55 Jahre	4
	mehr als 55 Jahre	2
Ausbildung	Angestellte	5
	Hochschulabsolventen	7
	Informatiker	7
	Studenten	15
	Schüler	3
	Kein Abitur	3

Tabelle 6.1.: Datensatz: Die Eigenschaften der 40 Probanden und die Häufigkeit davon

In der Umfrage gibt es drei Fragen: Die erste hat keine Hinweise, die zweite zeigt Bilder von Buchungsformularen als Hinweis und die dritte zeigt dem Teilnehmer einige Optionen wie z.B. Sitzplatz bei der Flugbuchung. In Tabelle 6.2 wird die Teilnahme der Nutzer in der jeweiligen Kategorie Flug, Bahn und Hotel detailliert beschrieben. Die Spalte „3 Anfragen“

stellt die Anzahl der Nutzer dar, die drei Anfragen gemacht haben. Das Gleiche gilt für die Spalten „2 Anfragen“ und „1 Anfrage“ für zwei bzw. eine Anfrage.

Kategorie	3 Anfragen	2 Anfragen	1 Anfrage	Insgesamt
Flug	9	4	27	62
Bahn	2	2	36	46
Hotel	1	1	38	43

Tabelle 6.2.: Nutzerteilnahme an der Umfrage

Ob die Nutzer drei, zwei oder nur eine Anfrage eingegeben haben, hängt von ihrer Erfahrung bei der Onlinebuchung ab. Diejenigen, die viel Erfahrung haben, konnten ihre Optionen auf einmal ohne Hinweise formulieren. Daher haben sie nur ein Anfrage eingegeben. Die Spalte „Insgesamt“ gibt die Anzahl der Anfragen der jeweiligen Kategorie an, die von den Teilnehmern gesammelt wurden und für das Testen des Systems verwendet wurden.

6.2. Metriken für die Evaluation des Systems

Die Anfragen der Nutzer werden im System untersucht. Die Antworten bzw. die Rückfragen des Systems werden betrachtet und es wird bestimmt, ob alle Informationen aus der Anfrage erkannt wurden und ob die richtigen Rückfragen gestellt werden. Jedes Ergebnis wird einer der vier folgenden Kategorien zugeordnet, die mit Hilfe der Abbildung 6.1, die als Beispiel für die Erkennung des Konzepts „Departure date“ verwendet wird, erklärt werden.

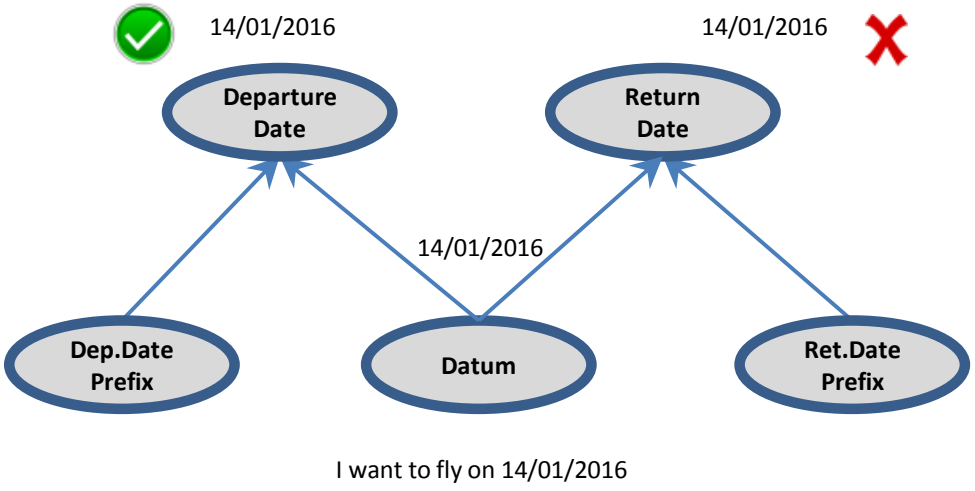

Abbildung 6.1.: Zuordnung der Ergebnisse für die Erkennung der Anfragen

- *Richtig positives Ergebnis RP:* Die Anfrage beinhaltet Informationen über ein bestimmtes Konzept und das System erkennt diese Informationen richtig und weist sie richtig dem entsprechenden Konzept zu. Ein Beispiel hierfür ist die Anfrage „I want to fly on 14/01/2016“. Wenn das Datum „14/01/2016“ erkannt wurde und dem Konzept „Departure date“ zugewiesen, dann wird dieses Ergebnis als richtiges positives Ergebnis zugeordnet.

- *Richtig negatives Ergebnis RN:* Die Anfrage beinhaltet keine Informationen über ein bestimmtes Konzept und das System weist diesem Konzept keinen Wert zu. Wenn der Benutzer z.B. die Anfrage „Book me a ticket from Karlsruhe to Berlin“ eingibt. Diese Anfrage beinhaltet keine Informationen über den Abflugort. Falls das System keinen Wert dem Konzept „Departure date“ zugewiesen hat, wird das Ergebnis von „Departure date“ als richtiges negatives Ergebnis betrachtet.
- *Falsch positives Ergebnis FP:* Die Anfrage beinhaltet keine Informationen über ein bestimmtes Konzept. Das System erkennt andere Informationen oder Wörter und weist sie diesem Konzept zu. Das ist bei der letzte Anfrage „Book me a ticket from Karlsruhe to Berlin“ der Fall ist, wenn das System ein Datum dem Konzept „Departure date“ zuweist, obwohl kein Datum in der Anfrage steht.
- *Falsch negatives Ergebnis FN:* Die Anfrage beinhaltet Informationen über ein bestimmtes Konzept. Das System erkennt diese Informationen nicht oder das System erkennt die Informationen richtig, aber weist diese Informationen einem anderen Konzept oder keinem Konzept zu. Das ist der Fall wenn der Benutzer die Anfrage „I want to fly on 14/01/2016“ eingibt und das System das Datum „14/01/2016“ nicht erkennt. Oder das System erkennt das Datum „14/01/2016“, aber weist es einem anderen Konzept zu, wie z.B. das Konzept „Return date“.

Aus den oben genannten Kategorien lassen sich zudem die Präzision und Ausbeute ableiten.

6.2.1. Präzision:

Die Präzision bewertet, wie exakt die zu einem Konzept erkannten und zugewiesenen Informationen tatsächlich zu diesem Konzept gehören. Sie lässt sich als Anteil der richtig positiv zugewiesenen Informationen an der Gesamtheit der als positiv zugewiesenen Informationen berechnen.

$$P = \frac{RP}{RP + FP} \qquad (0 \leq P \leq 1) \tag{6.1}$$

6.2.2. Ausbeute:

Die Ausbeute besagt, wie viele Informationen richtig einem Konzept zugewiesen wurden im Vergleich zu der Gesamtheit aller Informationen, die richtig zugewiesen wurden, und denjenigen Informationen, die dem Konzept entsprechen würden, aber ihm nicht zugewiesen wurden.

$$A = \frac{RP}{RP + FN} \qquad (0 \leq A \leq 1) \tag{6.2}$$

6.3. Evaluation des Systems

Die Evaluation der Flug-, Bahn- und Hotelontologie wird im Folgenden beschrieben. Dabei werden jeweils die Evaluationen in Hinblick auf die Präzision und Ausbeute bewertet. Eine Diskussion über die Testergebnisse wird geführt, um zu erklären, was diese Ergebnisse bedeuten und wie die Leistung des System verbessert werden kann.

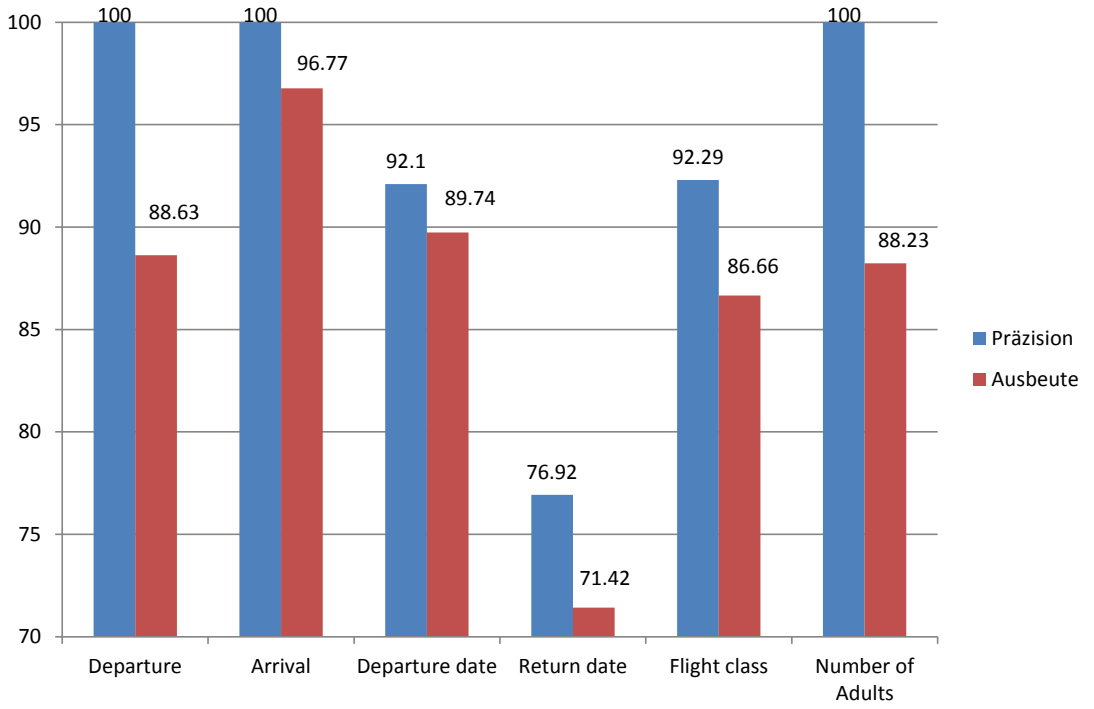

Abbildung 6.2.: Präzision und Ausbeute der Konzepte der aktiven Ontologie „Flug“

6.3.1. Evaluation der Flugontologie

In Abbildung 6.2 werden Präzision und Ausbeute aus den Untersuchungen der 62 Anfragen der Flugbuchung gestellt. Die als falsch positive und falsch negative erkannten Anfragen werden im Einzelnen diskutiert. Dabei erfahren die Testergebnisse der obligatorischen Konzepte und derjenigen Konzepte, die häufig von den Probanden verwendet wurden, besondere Betrachtung. Die Konzepte, die nicht getestet wurden, sind optionale Konzepte. Die Probanden haben sehr wenige Informationen über diese Konzepte gegeben und außerdem sind die Ausdrücke der Probanden im System definiert, da es wenige sprachliche Varianten von diesen gibt. Des Weiteren sind diese Konzepte wie getestete Konzepte implementiert und diese Konzepte haben fast keine Auswirkung auf die Ergebnisse, die der Nutzer erhält, da sie keine wichtigen Informationen tragen.

6.3.1.1. Departure

Das Konzept „Departure“ wurde mit 100% Präzision und 88,63% Ausbeute erkannt. Einige Anfragen beinhalten Abflugort-Informationen, die jedoch vom System nicht zum Konzept „Departure“ zugewiesen worden sind. Dies lag daran, dass die Nutzer einige Ausdrücke verwendet haben, die im System nicht definiert wurden, um die Ausdrücke, die dem Konzept „Departure“ zugeordnet werden sollen, zu bestimmen. Diese Ausdrücke sind „fly out of“ wie z.B. in der Anfrage „I am flying out of Frankfurt“ und „I want to fly out of Karlsruhe“. Diese Anfragen können anschließend durch das Hinzufügen des Präfixes „out of“ zur „Departure-Präfixe“ im System erkannt werden.

6.3.1.2. Arrival

Die Präzision und die Ausbeute des Konzepts „Arrival“ sind 100% und 96,77%. Lediglich zwei Anfragen wurden vom System nicht richtig erkannt aufgrund der Verwendung des

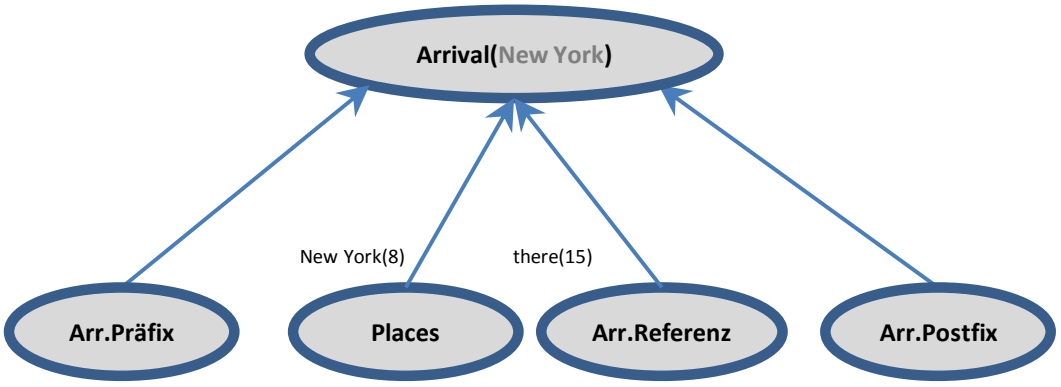

Abbildung 6.3.: Referenz-Auflösung in der Erkennung des Konzepts „Arrival“

Präfixes „my Destination is“, die im System definiert werden kann, und aufgrund des impliziten Verweises beim „Arrival“ im Satz „I want to do my Christmas shopping in New York, so I want to fly there“. In dieser Anfrage wurde „New York“ als Ort richtig erkannt, aber nicht dem Konzept „Arrival“ zugewiesen. Der Benutzer hat das Lokaladverb „there“ verwendet, um auf „New York“ als „Arrival“ zu verweisen. Um solche Informationen richtig den entsprechenden Konzepten zuzuweisen, ist es notwendig, neue Sensorknoten zu definieren, die derartige lokale Adverbien erkennen und ihre Referenz auflösen. Diese Referenzen könnten mit einem Präfix oder Postfix verbunden werden. „there“ im Ausdruck „fly there“ referenziert ein Ziel, während es im Ausdruck „from there“ auf einen Startpunkt verweist. Dies könnte durch einen zusätzlichen Sensorknoten implementiert werden, der dem Knoten „Ort“, „Präfix“ und „Postfix“ in Abbildung 5.3 hinzugefügt wird [1]. Alle Informationen würden anschließend im Gatherknoten „Arrival“ gesammelt und der Ort richtig dem Konzept „Arrival“ zugewiesen. Diese Implementierung könnte bei allen Knoten verallgemeinert werden. Die Abbildung 6.3 zeigt den neuen Knoten „Referenz“ und wie „New York“ in der Anfrage „I want to do my Christmas shopping in New York, so I want to fly there“ dem Konzept „Arrival“ zugewiesen werden könnte, da die Distanz zwischen „New York“ und „there“ die kleinste zwischen einem Ort und „there“ ist.

6.3.1.3. Departure date und Return date

Das Abflugdatum wurde im System mit einer Präzision von 92,10% und einer Ausbeute von 89,74% erkannt, während „Return date“ mit 76,92% Präzision und 71,42% Ausbeute erkannt wurde. Das Datum in drei Anfragen wurde falsch zugewiesen aufgrund der Verwendung der Uhrzeit, das ist z.B. in der Anfrage „I want to fly to Paris on 05/05/2016 at 07:00“ der Fall. In dieser Anfrage wird die Uhrzeit „7:00“ von der Bibliothek „SUTime“ von „The Stanford NLP Group“ erkannt und in das heutige Datum konvertiert. Dieses Datum steht zeitlich vor dem Datum „05/05/2016“. Deswegen wird dieses Datum als „Departure date“ erkannt und das Datum „05/05/2016“ als „Return date“. Ohne die Verwendung von „SUTime“ wurde das Datum richtig erkannt und zugewiesen. Dieses Problem könnte behoben werden, wenn die Anfrage zwei mal verarbeitet würde: Zuerst mit „SUTime“ und

[1] Diese Abbildung stellt das Konzept „Departure“ dar; die gleiche Abbildung kann für „Arrival“ verwendet werden, bei der lediglich die verwendeten Präfixe oder Postfixe verändert sind.

dann ohne „SUTime“. Wenn die Ergebnisse nicht übereinstimmen, dann würde das System die Verarbeitung ohne „SUTime“ verwenden.

Ein anderer Fall tritt in der Anfrage „I would like to book a flight to London between 15/11/2015 und 17/11/2015“ auf. Die Daten „15/11/2015“ und „17/11/2015“ wurden vom System als „departure date“ und „return date“ erkannt. Jedoch hat der Benutzer nicht beabsichtigt, dass das Datum „15/11/2015“ automatisch als „departure date“ erkannt wird sowie das Datum „17/11/2015“ als „return date“. Er wollte lediglich nach Flügen zwischen den beiden Datumsangaben suchen. Dieser Fall „between and“ muss speziell verarbeitet werden, sodass der Nutzer durch den Dialog-Manager gefragt werden muss, ob er einen Rückflug oder nur einen Hinflug wünscht. Wenn er nur einen Hinflug wünscht, wird nach möglichen Hinflügen an allen diesen Tagen gesucht.

Ein weiterer Fall tritt ein, wenn der Benutzer kein gültiges Datum eingibt, wie z.B. bei „I would like to fly on Tuesday 24th of August“. Der Tag von „24th of August“ im Jahr 2016 ein Mittwoch. Die Bibliothek „SUTime“ hat „On Tuesday“ zu dem nächste Dienstag in der aktuellen Woche und das Datum „24th of August“ zum „24/08/2016“ konvertiert. „On Tuesday“ wurde sodann als „Departure date“ und „24/08/2016“ als „Return date“ erkannt. Deswegen ist es notwendig zu wissen, dass das System keine Validierung für das falsche Datum durchführt. In diesem Fall ist es dem System nicht möglich, den Benutzer auf den Fehler hinzuweisen, weil es ihn nicht meldet.

6.3.1.4. Flight class

Die Anfragen werden bezüglich dem Konzept „Flight class“ mit 96,26% Präzision und 86,66% Ausbeute erkannt. Dies liegt in der Verwendung von im System undefinierten Ausdrücken, wie z.B. beim Satz „I want the flight class to be the best“, begründet. Wurde dem System dieser neue Ausdruck hinzugefügt, wird es „the best“ einfach als „first“ erkennen.

Ein besonderer Fall tritt in der Anfrage „I want to fly on Saturday first class“ ein. Die Bibliothek „SUTime“ hat „Saturday first“ als Datum erkannt und in das Datum mit dem Muster dd/MM/yyyy konvertiert. Daher wurde die Klasse „first“ nicht mehr erkannt. Das könnte durch die Verarbeitung der sprachlichen Anfrage zwei mal behoben werden, genau so wie bei der vorgestellten Lösung bei „Departure and return date“.

Ein Teilnehmer an der Umfrage hat die Anfrage „I do not want to fly first class“ verwendet. Die Negation wurde in dieser Arbeit nicht unterstützt; das System hat die Klasse „first“ bestimmt. Dieser Fall könnte mit der Erstellung eines neuen Knotens „Negation“ vermieden werden. Dieser Knoten greift jeden negativen Ausdruck wie „Not“ oder „No“ auf. Der GatherKnoten jedes Konzepts sucht in den vorherigen und nächsten Indizes der Wörter vor und nach dem Wort des Konzepts, in diesem Fall „first“. Wenn der negative Ausdruck in diesem Bereich stehen würde, würde der Wert dem Konzept nicht zugewiesen. Falls viele Werte für das Konzept stehen, könnte der Benutzer danach gefragt werden, einen Wert davon auszuwählen.

6.3.1.5. Number of Adults

Die Präzision und die Ausbeute für die Erkennung der Anzahl der Erwachsenen waren 100% und 88,23%. Lediglich die Ausdrücke wie „We are couple“ „book me a flight for me and my wife“ wurde vom System nicht verstanden. Diese Fälle könnten durch spezielle Verarbeitung erkannt werden. Ein neuer Knoten würde z.B. für die Auflösung solcher Ausdrücke erstellt.

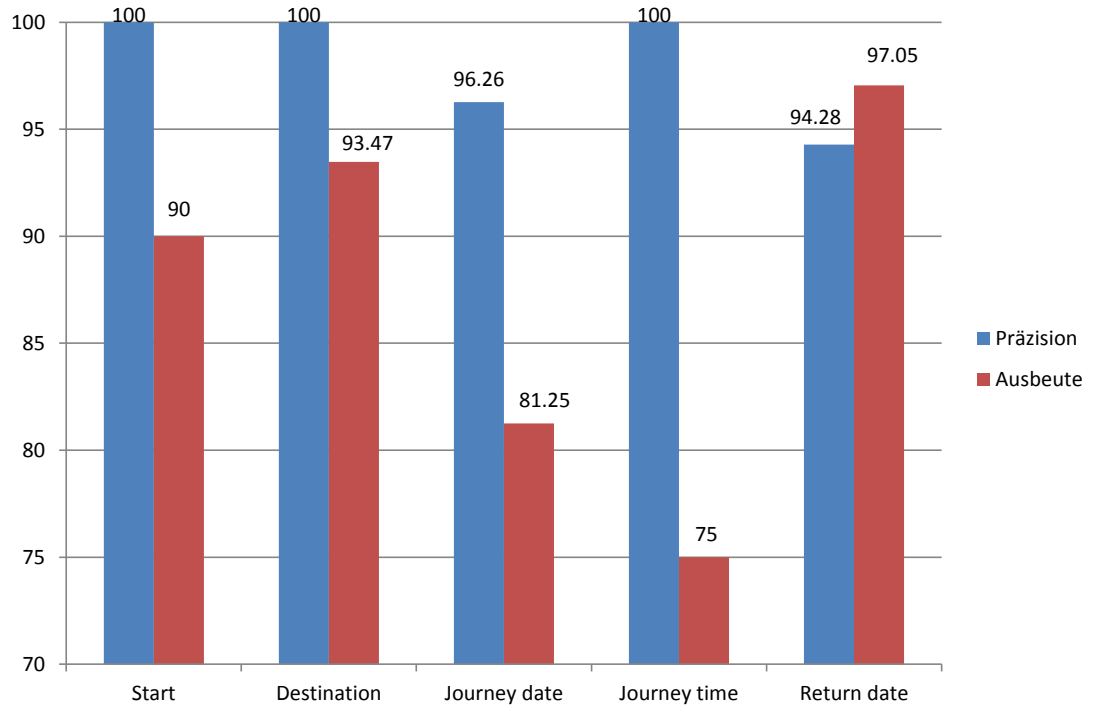

Abbildung 6.4.: Präzision und Ausbeute der Konzepte der aktiven Ontologie „Bahn“

6.3.2. Evaluation der Bahnontologie

Die Präzision sowie die Ausbeute der obligatorischen Konzepte der aktiven Ontologie „Bahn“ werden in Abbildung 6.4 dargestellt, dazu gehören ebenso die optionalen Konzepte, über die die Nutzer ausreichend viele Informationen gegeben haben. Die evaluierten Konzepte in der Bahnontologie sind:

6.3.2.1. Start

Die Präzision und die Ausbeute des Konzepts „Start“ sind 100% und 90%. Beim „Start“ werden nur zwei Anfragen als falsch negativ ausgewertet. Die erste Anfrage betrifft „Tell me the train schedule for tomorrow morning between London and Manchester“. „London“ und „Manchester“ wurden als Orte erkannt, aber nicht mit den Konzepten „Start“ und „Destination“ verbunden, da die Ausdrücke „between ... and ...“ für das System nicht definiert wurden. Wenn sie definiert wären, würden die Anfragen richtig verstanden.

Die zweite sprachliche Eingabe wird als „I would like to book a train ticket to Stuttgart on 12/11/2015 and I want to return to Karlsruhe on 19/11/2015“ ausgedrückt. Damit das System „Karlsruhe“ als Startpunkt erfasst, braucht es die Bestimmung von „return to“ als Präfix, das für die Erkennung des Startpunkts verwendet werden kann.

6.3.2.2. Destination

Das Konzept „Destination“ wird ebenso mit guter Präzision und Ausbeute ausgewertet, und zwar mit 100% und 93,47%. Die Informationen über „Destination“ wurden in drei Anfragen nicht identifiziert. Zwei Benutzer haben das Präfix „headed“ verwendet, wie im Satz „I am headed Londen next Saturday“. Dies lässt sich einfach durch die Definition von „headed“ als Endpunkt-Präfix korrigieren. Der dritte Fall war die Anfrage „between ... and ...“, die im letzten Abschnitt „Start“ bereits beschrieben wurde.

6.3.2.3. Journey date und Return date

Die Präzision und die Ausbeute für „Journey date“ trifft zu 96,26% und 81,25% zu und für „Return date“ zu 94,28% und 97,05% zu. Viele Fälle sind bei den Konzepten „Departure date“ und „Return date“ bereits in der Flugontologie beschrieben. Bei der Bahnontologie gibt es noch weitere Gründe für die falsche Erkennung des Datums, wie z.B. bei der Aussage „I want to leave in 20 minutes“. Das System kann das Datum nicht als das heutige Datum ohne einen expliziten Hinweise verstehen. Bei der Anfrage „I would like to book a ticket to Berlin next Friday in the morning“ erkennt die Bibliothek „SUTime“ den Ausdruck „next Friday“ richtig, jedoch „in the morning“ nicht. Sie konvertiert diesen Ausdruck in das heutige Datum. Das ist ebenfalls genau so der Fall beim Ausdruck „at around 02:00“ im Satz „17/11/2015 at around 02:00“, wo „at around 02:00“ zum heutigen Datum umgewandelt wird.

6.3.2.4. Journey time

Die Uhrzeit der Reise wird mit 100% Präzision und 75% Ausbeute bewertet. Dies liegt in denselben Ursachen wie beim Datum in dem letzten Abschnitt begründet, nämlich in den Anfragen „I want to leave in 20 minutes“ und „next Friday in the morning“.

6.3.3. Evaluation der Hotelontologie

Die Abbildung 6.5 stellt die Präzision und die Ausbeute für die Konzepte der Hotelontologie dar. Diese Evaluation wird für jedes Konzept im Folgenden beschrieben.

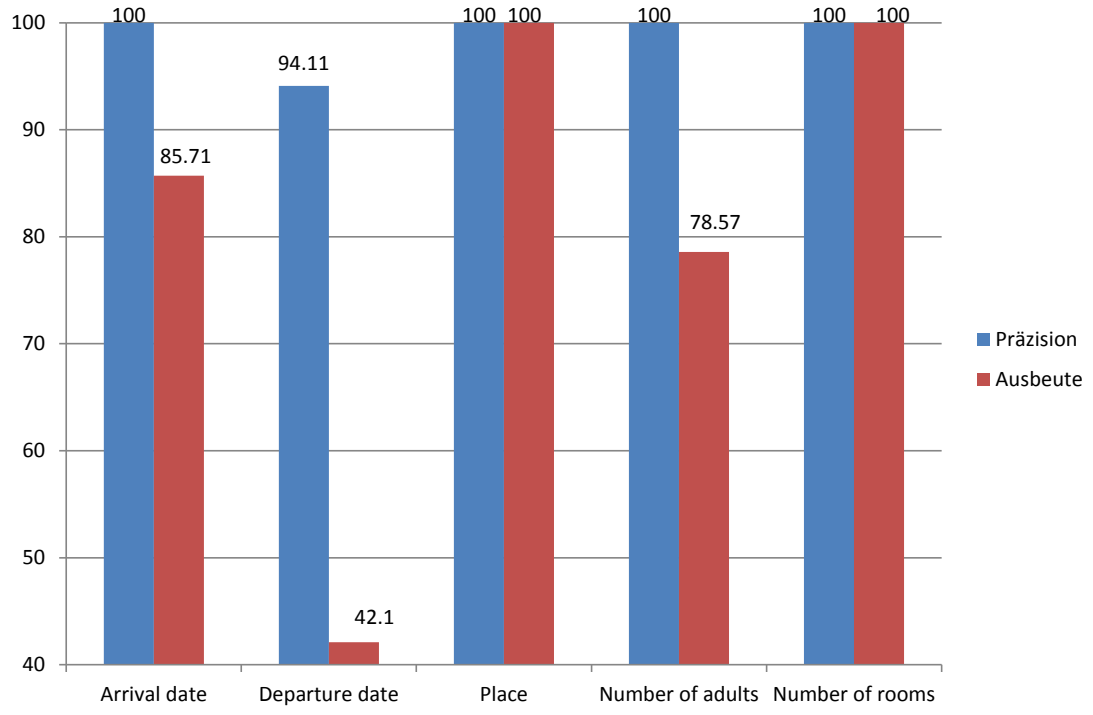

Abbildung 6.5.: Präzision und Ausbeute der Konzepte der aktiven Ontologie „Hotel“

6.3.3.1. Arrival date und Departure date

Die Präzision für das „Arrival date“ beträgt 100% und die Ausbeute 85,71%. Für das „Departure date“ liegen eine Präzision von 94,11% und eine Ausbeute von 42,10% vor.

Das liegt unter anderem an der Verwendung einiger undefinierter Ausdrücke, wie z.B. die Ausdrücke „for the month of November“, „the third weekend in September“, die von „SUTime“ nicht richtig erkannt wurden. Die Dauer der Unterkunft wurde im System mit Tagen berechnet. Einige Benutzer haben die Dauer jedoch mit Wochen angegeben, wie z.B. „for three weeks“. Dieser Fall könnte einfach erkannt werden, sodass das „Departure date“ als das „Arrival date“ plus die Dauer von 7*3 = 21 Tagen berechnet würde.

Der Hauptgrund der niedrigen Ausbeute liegt an der Verwendung von „nights“, um die Dauer einzugeben. Ein Beispiel hierfür ist die Anfrage „I need to book a room in Paris Our stay will be beginning on May 9th for 2 nights“, bei der „nights“ von „SUTime“ als das Datum von heute konvertiert wird. Dieser Fall ist häufig (in beinahe 70% der falschen Fälle) vorgekommen. Das kann dadurch behoben werden, dass die Anfrage mit und ohne „SUTime“ verarbeitet würde. Wenn die Ergebnisse nicht übereinstimmen, würde „SUTime“ nicht verwendet.

6.3.3.2. Hotel place

Das Konzept „Place“ wurde mit 100% Präzision und 100% Ausbeute erkannt. Es gab keine Anfrage, die nicht oder falsch erkannt wurde.

6.3.3.3. Number of adults

Die Präzision des Konzepts „Number of Adults“ beträgt 100%, während die Ausbeute bei 78,57% liegt. Drei Anfragen wurden nicht erkannt. Der Grund hierfür ist die Verwendung von im System undefinierten Ausdrücke, wie bspw. „for me and my wife“, „we are couple“ und „we are two students“. Für die ersten zwei Ausdrücke muss eine spezielle Verarbeitung definiert werden, um die Ausdrücke mit „and“ aufzuzählen und „couple“ als zwei Personen aufzulösen. „students“ hingegegen kann einfacherweise als Postfix im System definiert werden.

6.3.3.4. Number of Rooms

Das Konzept „Number of Rooms“ wurde erfolgreich erkannt, und zwar mit 100% Präzision und 100% Ausbeute. Keine Anfrage wurde falsch oder nicht erkannt.

6.4. Evaluation der Erkennung von Nutzeranfragen

Bei den Nutzeranfragen über Flugbuchungen wurden 21 Anfragen vollständig von den Nutzern eingegeben und vom System vollständig erkannt. Elf Nutzeranfragen wurden vollständig eingegeben, aber vom System nicht vollständig erkannt. Aufgrund dessen hat das System die Nutzer nach den nicht erkannten Informationen gefragt.

24 Anfragen wurden von den Probanden nicht vollständig eingegeben, sodass das System den Nutzer nach den fehlenden Informationen vollständig gefragt hat. Sechs Nutzeranfragen wurden nicht vollständig eingegeben; jedoch hat das System einige Informationen falsch erkannt und deswegen nicht mehr nach den fehlenden Informationen gefragt.

Diese Ergebnisse werden für Flug-, Bahn- und Hotelontologie in Tabelle 6.3 gelistet. Die Ergebnisse für Bahn- und Hotelontologien können gleichermaßen in der Tabelle abgelesen werden.

Eingabe	vollständig		unvollständig	
Erkennung	vollständig	unvollständig		
Rückfragen			vollständig	unvollständig
Flug	21	11	24	6
Bahn	15	7	14	10
Hotel	14	5	17	7

Tabelle 6.3.: Evaluation des gesamten Systems

6.5. Zusammenfassung

Bei der Evaluation des Systems, nämlich der Informationserkennung in den sprachlichen Anfragen der Nutzer, wurde ersichtlich, dass die Information über die Flug-, Bahn- und Hotelbuchungen mit einer Präzision von mehr als 92% erkannt wurde, während die Ausbeute bei mehr als 75% lag, außer bei der Erkennung von „Departure date“ in der aktiven Ontologie „Hotel“. Das Hauptproblem bei der Erkennung von „Departure date“ liegt an der Verwendung des Worts „nights“, um die Dauer der Unterkunft einzugeben, wie z.B. im Satz „for 2 nights“. Die Bibliothek „SUTime“, die verwendet wurde, um die zeitlichen Ausdrücke wie „tomorow“, „next friday“, „in three days“ herauszufiltern und in das Muster dd/MM/yyyy umzuwandeln, konvertiert das „night“ zum heutigen Datum. Dieses Problem könnte durch die Überprüfung der zeitlichen Ausdrücke behoben werden.

Bei den anderen Ausfällen hat das System die Informationen nicht erkannt. Dies liegt an der Verwendung einer im System undefinierten Ausdrücke. Diese Ausdrücke wurden einfach im System definiert. Anschließend wurden die betreffenden Informationen richtig erkannt. Das Definieren von weiteren Ausdrücke und andere Ausblicke werden im nachfolgenden Kapitel beschrieben.

7. Zusammenfassung und Ausblick

Wie können Webformulare auf aktive Ontologien abgebildet werden? Das ist die Fragestellung, die in dieser Arbeit untersucht wurde. Die Betrachtung erfolgte dabei im Rahmen des EASIER-Projektes des IPD Prof. Tichy [BL]. Die aktiven Ontologien in dieser Arbeit wurden hierbei für Flug-, Bahn- und Hotelbuchungen erstellt, weil diese Buchungsvorgänge von intelligenten Assistenten noch nicht vollständig unterstützt werden.

7.1. Zusammenfassung

Es wurden jeweils zehn Webformulare unterschiedlicher Fluggesellschaften, Verkehrsdienstleister und Hotels ausgewählt, um die größtmögliche Informationen über diese Domäne zu sammeln.

Die Webformulare wurden analysiert, um die Konzepte der Ontologie zu ermitteln. Dabei wurden die Beziehungen zwischen den Konzepten sowie die Werte ermittelt, die jedes Konzept erhalten kann. Dies wurde durch das Speichern aller Konzeptnamen, -werte und -eigenschaften in einer Tabelle durchgeführt. Ein bezeichnender Name für jedes Konzept wurde ausgewählt und mit dessen Werten und Eigenschaften für die Erstellung der aktiven Ontologie verwendet. Ebenso wurde durch die Analyse entschieden, ob jedes Konzept obligatorisch in der entsprechenden Domäne bzw. Ontologie enthalten ist oder nicht. Als Resultat der Analyse werden die Felder aller Webformulare und deren Optionen in der entsprechenden aktiven Ontologie repräsentiert.

Die aktiven Ontologien werden in den intelligenten Assistenten eingesetzt, um die sprachlichen Eingaben der Nutzer zu verstehen. Deswegen ist es notwendig zu wissen, wie die Nutzer sprachlich nach ihren Wünschen fragen können. Zu diesem Zweck wurde eine Umfrage im Laufe dieser Arbeit durchgeführt. Probanden mit unterschiedlichem Englischniveau, Reiseerfahrung und Alter haben sprachliche Anfragen für Flug-, Bahn- und Hotelbuchung eingegeben. Die Anfragen wurden analysiert und es wurde festgestellt, wie und welche Ausdrücke und Varianten die Nutzer verwenden, wenn sie Informationen im jeweiligen Konzept der aktiven Ontologie übermitteln. Diese gesammelten Informationen werden für die Entwicklung der Software sowie für die Evaluation verwendet. Mithilfe der Umfrage wurden viele Ausdrücke und sprachliche Varianten ermittelt, die von den Nutzern verwendet werden können, um Informationen über die Konzepte der Ontologie einzugeben. Es wurde versucht, die Knotentypen von EASIER zu verwenden. Dort, wo diese nicht ausgereicht haben, wurden neue erstellt. Es wurden Sensorknoten sowie Kombinationsknoten verwendet. Die Sensorknoten erkennen Orte, Nummern, reguläre Ausdrücke z.B. bei Datum und

Uhrzeit, bestimmte Informationen wie Präfixe und Postfixe. Die Kombinationsknoten weisen die von den Sensoren erkannten Informationen den entsprechenden Konzepten zu. Die Ergebnisse dieser Implementierung waren, dass die sprachlichen Anfragen der Probanden mit einer Präzision von mehr als 92% für alle Konzepte und einer Ausbeute von mehr als 75% mit Ausnahme des Konzepts „Departure date“ von Hotelontologie erkannt wurden. Insgesamt wurden mehr als 85% der sprachlichen Anfrage von Probanden vollständig vom System erkannt oder nachgefragt, wenn ein Konzept vom Nutzer nicht eingegeben wurde.

Zudem wurde in dieser Arbeit ein Dialog-Manager entwickelt, um einen interaktiven Dialog mit den Nutzern zu führen, falls sie wichtige Informationen nicht eingegeben haben oder falls das System mehrere Optionen anbietet. Der Nutzer kann die Fragen entweder als Freitext beantworten oder seine Antworten in geeignete Formularfelder wie z.B. in einen Kalender oder eine Liste eingeben. Der Nutzer erhält ebenfalls die in den Webseiten gefundenen Ergebnisse als Dialog.

Das System zeigt dem Nutzer die gefundenen Ergebnisse (Flüge, Bahnen oder Hotels) als Optionen an, damit er seine gewünschte Buchung auswählen kann. Der Dialog-Manager hat den Nutzer immer erfolgreich nach den vom System nicht erkannten Informationen gefragt.

Die Ergebnisse werden durch Web-Scraping gesammelt. Die Informationen aus der aktiven Ontologie wurden für jedes Formular verarbeitet, und deren Felder ausgefüllt. Das Formular wurde abgeschickt und die Ergebnisse werden gesammelt. Die Informationen wurden parallel an die Webseiten gesendet und die Ergebnisse parallel gesammelt.

7.2. Ausblick: Aktive Ontologien

In dieser Arbeit wurden die aktiven Ontologien so erstellt, dass die Webformulare manuell analysiert werden; darauf aufbauend wurden die Konzepte der aktiven Ontologien gefertigt.

Da die manuelle Erzeugung der aktiven Ontologie aufwändig ist, soll sie automatisiert werden. Die automatische Erstellung der aktiven Ontologie würde automatisch die Konzepte bzw. die Knoten einer aktiven Ontologie aus den entsprechenden Feldern eines Formulars erzeugen. Dies erfordert, dass die Art des Feldes wie z.B. eine Texteingabe oder ein Kalender in einem Webformular identifiziert und der Name des Felds dem richtigen Konzept zugeordnet werden muss. Die möglichen Namen, Wertebereiche und andere Eigenschaften, die in der Domäne der Flug-, Bahn- und Hotelbuchungen beinhaltet sein können, wurden in dieser Arbeit gesammelt. Die im EASIER bestehenden Knotentypen wurden ebenfalls erweitert, um jedes Feld eines Formulars zu repräsentieren. Dies ist z.B. für einen Knoten mit Datum, Nummer und einer Liste von Optionen der Fall. Dies kann bei der Automatisierung der Ontologie-Erstellung sehr hilfreich sein. Aus diesem Grund kann diese Arbeit als Basis für die weitere automatische oder zumindest semiautomatische Erstellung von aktiven Ontologien betrachtet werden.

Weiterhin müssen ebenfalls viele Informationen gesammelt werden, um herauszufinden, wie die Nutzer ihre Anfrage sprachlich formulieren. Dies erfordert, dass das System von möglichst vielen Nutzern getestet wird. Dadurch kann eine hohe Bandbreite an Varianten und Formulierungen im System definiert werden.

Die sprachlichen Varianten, die das vorliegende System nicht unterstützt, könnten in der Zukunft entwickelt werden. Das beinhaltet die Negation wie z.B. im Satz „I do not want the flight class to be first“. Ein anderer Fall ist die Eingabe des Datums als Zeitbereich wie z.B. bei der Verwendung des Ausdrucks „between 12/01/2016 and 14/01/2016“, was bedeutet, dass der Nutzer einen Flug buchen möchte, der an einem Tag in diesem Zeitbereich ist. Die Verwendung von Referenzen könnte auch zukünftig unterstützt werden. Das is der Fall

im Satz „I want to do my christmas shopping in New York, so I want to fly there“. Das System müsste „New York“ als Ankunftsort erkennen.

7.3. Ausblick: Dialog-Manager

Zukünftig sollen dem Nutzer nicht nur Ergebnisse angezeigt werden, sondern ihm auch Vorschläge präsentiert werden. Wenn das System keine Flüge findet, muss es andere Flüge anbieten, die von einer Stadt in der Nähe abfliegen. Dies könnte durch den Aufruf externer Dienste durchgeführt werden. Ebenso sollten die Fragen im Dialog-Manager zukünftig automatisch generiert werden. Für jedes Konzept werden sodann das geeignete Verb, Fragewörter und andere Aussagen, die in der Frage stehen können, automatisch generiert. Ein Beispiel dafür ist das Konzept „Departure“ bei Flugontologie. Dabei müssen das Verb „fly“, das Fragewort „Where“ und das Wort „from“ automatisch abgeleitet werden. Somit könnte der Benutzer wohl formulierte und automatisch generierte Fragen erhalten wie z.B. „Where do you want to fly from?“.

Literaturverzeichnis

[ABM04] An, Yuan ; Borgida, Alexander ; Mylopoulos, John: Refining Semantic Mappings from Relational Tables to Ontologies. In: Bussler, Christoph (Hrsg.) ; Tannen, Val (Hrsg.) ; Fundulaki, Irini (Hrsg.): *Semantic Web and Databases*. Springer Berlin Heidelberg, August 2004 (Lecture Notes in Computer Science 3372). – ISBN 978–3–540–24576–6 978–3–540–31839–2, S. 84–90 (zitiert auf Seite 22).

[ABM05a] An, Yuan ; Borgida, Alex ; Mylopoulos, John: Constructing complex semantic mappings between XML data and ontologies. In: *The Semantic Web–ISWC 2005*. Springer, 2005, S. 6–20 (zitiert auf Seite 22).

[ABM05b] An, Yuan ; Borgida, Alex ; Mylopoulos, John: Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In: *On the move to meaningful internet systems 2005: CoopIS, DOA, and ODBASE*. Springer, 2005, S. 1152–1169 (zitiert auf Seite 22).

[ABM08] An, Yuan ; Borgida, Alex ; Mylopoulos, John: Discovering and maintaining semantic mappings between XML schemas and ontologies. In: *Journal of computing Science and Engineering* 2 (2008), Nr. 1, S. 44–73 (zitiert auf Seite 22).

[AGWC07a] An, Yoo J. ; Geller, James ; Wu, Yi-Ta ; Chun, Soon: Semantic deep web: automatic attribute extraction from the deep web data sources. In: *Proceedings of the 2007 ACM symposium on Applied computing*, ACM, 2007, S. 1667–1672 (zitiert auf Seite 23).

[AGWC07b] An, Yoo J. ; Geller, James ; Wu, Yi-Ta ; Chun, Soon A.: Automatic Generation of Ontology from the Deep Web, IEEE, September 2007. – ISBN 978–0–7695–2932–5, S. 470–474 (zitiert auf Seite 23).

[AHS12] An, Yuan ; Hu, Xiaohua ; Song, Il-Yeol: Learning to Discover Complex Mappings from Web Forms to Ontologies. In: *Proceedings of the 21st ACM international conference on Information and knowledge management*, ACM, 2012 (CIKM '12). – ISBN 978–1–4503–1156–4, S. 1253–1262 (zitiert auf Seite 24).

[AM05] An, Yuan ; Mylopoulos, John: Translating XML web data into ontologies. In: *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, Springer, 2005, S. 967–976 (zitiert auf Seite 22).

[AMB06] An, Yuan ; Mylopoulos, John ; Borgida, Alexander: Building semantic mappings from databases to ontologies. In: *Proceedings of the National Conference on Artificial Intelligence* Bd. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, S. 1557 (zitiert auf Seite 22).

[AOSS95] Aust, Harald ; Oerder, Martin ; Seide, Frank ; Steinbiss, Volker: The Philips automatic train timetable information system. In: *Speech Communication* 17 (1995), November, Nr. 3–4, S. 249–262. http://dx.doi.org/10.1016/0167-6393(95)00028-M. – DOI 10.1016/0167–6393(95)00028–M. – ISSN 0167–6393 (zitiert auf den Seiten 19 und 25).

[AT08] An, Yuan ; Topaloglou, Thodoros: *Maintaining semantic mappings between database schemas and ontologies.* Springer, 2008 (zitiert auf Seite 22).

[BFS+02] Bennett, Christina ; Font Llitjos, Ariadna ; Shriver, Stefanie ; Rudnicky, Alexander ; Black, Alan W.: Building VoiceXML-based applications / DTIC Document. 2002. – Forschungsbericht (zitiert auf Seite 24).

[BJNS10] Berlanga, Rafael ; Jimenez-Ruiz, Ernesto ; Nebot, Victoria ; Sanz, Ismael: Faeton: Form analysis and extraction tool for ontology construction. In: *International Journal of Computer Applications in Technology* 39 (2010), Nr. 4, S. 224–233 (zitiert auf Seite 24).

[BL] Blersch, Martin ; Landhäusser, Mathias: *EASIER: An Approach to Automatically Generate Active Ontologies for Intelligent Assistants* (zitiert auf den Seiten 1, 4, 50, 51 und 87).

[CG14] Cheyer, Adam ; Guzzoni, Didier: *Method and Apparatus for Building an Intelligent Automated Assistant.* April 2014. – U.S. Classification 706/11; International Classification G06N5/02; Cooperative Classification G06N5/02, G09B23/28, G09B21/00 (zitiert auf Seite 21).

[DAB+14] DeVault, David ; Artstein, Ron ; Benn, Grace ; Dey, Teresa ; Fast, Ed ; Gainer, Alesia ; Georgila, Kallirroi ; Gratch, Jon ; Hartholt, Arno ; Lhommet, Margaux ; others: SimSensei Kiosk: A virtual human interviewer for healthcare decision support. In: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2014, S. 1061–1068 (zitiert auf Seite 17).

[Dav] David, Patty: *AARP Online Travel Study. 2013* (zitiert auf Seite 1).

[DBD98] DYBKJÆR, LAILA ; BERNSEN, NIELS O. ; DYBKJÆR, HANS: A methodology for diagnostic evaluation of spoken human — machine dialogue. In: *International Journal of Human-Computer Studies* 48 (1998), Mai, Nr. 5, S. 605–625. http://dx.doi.org/10.1006/ijhc.1997.0183. – DOI 10.1006/ijhc.1997.0183. – ISSN 1071–5819 (zitiert auf den Seiten 18 und 24).

[DKYL09] Dragut, Eduard C. ; Kabisch, Thomas ; Yu, Clement ; Leser, Ulf: A hierarchical approach to model web query interfaces for web source integration. In: *Proceedings of the VLDB Endowment* 2 (2009), Nr. 1, S. 325–336 (zitiert auf Seite 22).

[DVNR03] Davalcu, H. ; Vadrevu, Srinivas ; Nagarajan, Saravanakumar ; Ramakrishnan, I.V.: OntoMiner: bootstrapping and populating ontologies from domain-specific Web sites. In: *IEEE Intelligent Systems* 18 (2003), September, Nr. 5, S. 24–33. http://dx.doi.org/10.1109/MIS.2003.1234766. – DOI 10.1109/MIS.2003.1234766. – ISSN 1541–1672 (zitiert auf Seite 23).

[FMTT13] Falquet, Gilles ; Mtral, Claudine ; Teller, Jacques ; Tweed, Christopher: *Ontologies in Urban Development Projects.* Springer Publishing Company, Incorporated, 2013. – ISBN 1–4471–2697–1 978–1–4471–2697–3 (zitiert auf Seite 7).

[GBC06] Guzzoni, Didier ; Baur, Charles ; Cheyer, Adam: Active: A Unified Platform for Building Intelligent Web Interaction Assistants. In: *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology - Workshops, Hong Kong, China, 18-22 December 2006*, IEEE Computer Society, Dezember 2006. – ISBN 0–7695–2749–3, S. 417–420 (zitiert auf Seite 21).

[GBC07] Guzzoni, Didier ; Baur, Charles ; Cheyer, Adam: Modeling Human-Agent Interaction with Active Ontologies. In: *nteraction Challenges for Intelligent Assistants, Papers from the 2007 AAAI Spring Symposium, Technical Report SS-07-04, Stanford, California, USA, March 26-28, 2007*. Stanford, California, USA : AAAI, März 2007, S. 52–59 (zitiert auf Seite 21).

[GCB06] Guzzoni, Didier ; Cheyer, Adam ; Baur, Charles: Active, a platform for building intelligent software. In: Kovalerchuk, Boris (Hrsg.): *Proceedings of the Second IASTED International Conference on Computational Intelligence, San Francisco, California, USA, November 20-22, 2006*, IASTED/ACTA Press, 2006. – ISBN 0–88986–603–1, S. 126–131 (zitiert auf Seite 21).

[GKD07] Ghoula, N. ; Khelif, K. ; Dieng-Kuntz, R.: Determining Bias to Search Engines from Robots.txt. In: *IEEE/WIC/ACM International Conference on Web Intelligence*, 2007, S. 149–155 (zitiert auf Seite 17).

[Guz08] Guzzoni, Didier: *Active: a unified platform for building intelligent applications*, École Polytechnique Fédérale De Lausanne, PhD Thesis, Januar 2008 (zitiert auf den Seiten 8 und 21).

[HC03] He, Bin ; Chang, Kevin Chen-Chuan: Statistical schema matching across web query interfaces. In: *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, ACM, 2003, S. 217–228 (zitiert auf Seite 22).

[HML+07] He, Hai ; Meng, Weiyi ; Lu, Yiyao ; Yu, Clement ; Wu, Zonghuan: Towards Deeper Understanding of the Search Interfaces of the Deep Web. In: *World Wide Web* 10 (2007), Mai, Nr. 2, S. 133–155. `http://dx.doi.org/10.1007/s11280-006-0010-9`. – DOI 10.1007/s11280–006–0010–9. – ISSN 1386–145X, 1573–1413 (zitiert auf Seite 22).

[HMYW04] He, Hai ; Meng, Weiyi ; Yu, Clement ; Wu, Zonghuan: Automatic integration of Web search interfaces with WISE-Integrator. In: *The VLDB Journal* 13 (2004), September, Nr. 3, S. 256–273. `http://dx.doi.org/10.1007/s00778-004-0126-4`. – DOI 10.1007/s00778–004–0126–4. – ISSN 1066–8888, 0949–877X (zitiert auf Seite 22).

[JBV+02] Johnston, Michael ; Bangalore, Srinivas ; Vasireddy, Gunaranjan ; Stent, Amanda ; Ehlen, Patrick ; Walker, Marilyn ; Whittaker, Steve ; Maloor, Preetam: MATCH: An architecture for multimodal dialogue systems. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 2002, S. 376–383 (zitiert auf Seite 19).

[MAY] MAY, Kevin: *Survey: how travellers use technology to search, book and play when away* (zitiert auf Seite 1).

[MBL13] Mitchell, Christopher M. ; Boyer, Kristy E. ; Lester, James C.: When to Intervene: Toward a Markov Decision Process Dialogue Policy for Computer

Science Tutoring. In: *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, 2013, S. 40 (zitiert auf Seite 26).

[McT02] McTear, Michael F.: Spoken Dialogue Technology: Enabling the Conversational User Interface. In: *ACM Comput. Surv.* 34 (2002), März, Nr. 1, S. 90–169. http://dx.doi.org/10.1145/505282.505285. – DOI 10.1145/505282.505285. – ISSN 0360–0300 (zitiert auf den Seiten 18 und 25).

[Mor97] Mori, Renato D.: *Spoken Dialogues with Computers*. Orlando, FL, USA : Academic Press, Inc., 1997. – ISBN 0–12–209055–1 (zitiert auf Seite 25).

[RG06] Roitman, Haggai ; Gal, Avigdor: Ontobuilder: Fully automatic extraction and consolidation of ontologies from web sources using sequence semantics. In: *Current Trends in Database Technology–EDBT 2006*. Springer, 2006, S. 573–576 (zitiert auf Seite 23).

[SC01] Szymanski, B.K. ; Chung, Ming-Shu: A method for indexing Web pages using Web bots. In: *2001 International Conferences on Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing* Bd. 3, 2001, S. 1–6 vol.3 (zitiert auf Seite 17).

[Wal09] Wallace, Richard S.: The Anatomy of A.L.I.C.E. In: Epstein, Robert (Hrsg.) ; Roberts, Gary (Hrsg.) ; Beber, Grace (Hrsg.): *Parsing the Turing Test*. Springer Netherlands, 2009. – ISBN 978–1–4020–9624–2 978–1–4020–6710–5, S. 181–210 (zitiert auf Seite 17).

[WDYM05] Wu, Wensheng ; Doan, AnHai ; Yu, Clement ; Meng, Weiyi: Bootstrapping Domain Ontology for Semantic Web Services from Source Web Sites. In: Bussler, Christoph (Hrsg.) ; Shan, Ming-Chien (Hrsg.): *Technologies for E-Services*. Springer Berlin Heidelberg, September 2005 (Lecture Notes in Computer Science 3811). – ISBN 978–3–540–31067–9 978–3–540–32889–6, S. 11–22 (zitiert auf Seite 23).

[WP] Weicksel, Johannes ; Pentsi, Angelika: *Smartphones: Tippst Du noch oder sprichst Du schon?* (zitiert auf Seite 1).

[WY07] Williams, Jason D. ; Young, Steve: Partially Observable Markov Decision Processes for Spoken Dialog Systems. In: *Comput. Speech Lang.* 21 (2007), April, Nr. 2, S. 393–422. http://dx.doi.org/10.1016/j.csl.2006.06.008. – DOI 10.1016/j.csl.2006.06.008. – ISSN 0885–2308 (zitiert auf den Seiten 25 und 27).

[WYDM04] Wu, Wensheng ; Yu, Clement ; Doan, AnHai ; Meng, Weiyi: An interactive clustering-based approach to integrating source query interfaces on the deep web. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, ACM, 2004, S. 95–106 (zitiert auf Seite 22).

[Zee] Zeendo, Mark: *Mobile Flight Booking System* (zitiert auf Seite 1).

Anhang

A. Umfrage-Flug

Tabelle A.1.: Nutzeranfrage über Flugbuchung

Flugbuchung-Anfrage
1- Book a flight from Karlsruhe to Berlin on the 8th of February at around 2pm or later There are two passengers My wife is vegetarian
2- I want to book a flight from Frankfurt to New York between 2/2 and 4/2
3- I want to fly from Frankfurt to New York I want to leave on 2/2 and come back at 9/2 I want to take economy class 2 adults no children price should be as low as possible
4- I want to fly from Frankfurt to New York I want to leave on 2/2 and come back at 9/2 I want to take economy class 2 adults no children price should be as low as possible I want to take my violin with me I want to have an internet access Payment should be by credit card4
5- I would fly to London next Wednesday
6- I would fly to London next Wednesday I need an economy class
7- I would fly to London next Wednesday I need an economy class I require vegetarian food wireless internet and if possible a window seat
8- I want to book a flight to Los Angeles from Karlsruhe
9- I want to book a flight to Los Angeles from Karlsruhe on one-way at the 27th of March in economy class
10- I want to book a flight to Los Angeles from Karlsruhe on one-way at the 27th of March in economy class with the lowest price Additionally I want to have wireless internet
11- I would like to reserve a plane ticket for a flight to Berlin I would be flying out of karlsruhe on 10th February
12- I would like to reserve a one-way first class plane ticket for an adult on the flight to Berlin I would be flying out of Karlsruhe on 10th February
13- I would like to reserve a one-way first class plane ticket for an adult on the flight to Berlin I would be flying out karlsruhe on 10th February my seat shoul be next to the window equipped with a socket for laptop charge and internet facilities The flight should offer vegetarian meals and allow traveling with pets
14- Show me the price of flights from Berlin to Frankfurt
15- I 'd like to fly to London next month leaving on March 3rd what are the available flights
16- I need a business class flight from Frankfurt to New York next weekend
17- I would like to book a one-way flight from Paris to London with a window seat and vegetarian food
18- Book me a flight to Los Angeles from Washington for me and my 2 Kids I would like to travel in economy class with a seat by the window

19- Please book me a ticket to Paris
20- Hello I'd like to book a ticket for a flight to the USA from Damascus Airport in Syria I am off to a holiday at the end of the month Friday 24th until Tuesday 28th Could you tell me about the flight availability and price please
21- Hello! I'd like to make a reservation for one first class ticket from Syria to Germany on Saturday
22- Hello! I'd like to make a reservation for one first class ticket from Lattakia Syria to Berlin Germany on Saturday
23- Hello! I'd like to make a reservation for one first class from Lattakia Syria to Berlin Germany on Saturday the forth of July and I want it to be one way flight Please make the reservation include an internet service and a charger for my portable
24- I would like to travel to London please I would like a snack some salad sounds good I would like my ticket to be one way I am going to go with my uncle
25- I want to reserve a two-way ticket from Syria to Brazil
26- I'm from Syria I want to reserve a two-way ticket to Brazil I want the flight class economy and the ticket of low price for adult I would like to have a charger for my laptop by my site
27- I would like to book a one-way ticket on the flight heading towards London next monday at 09
28- I would like to book a one-way ticket 1 adult first class on the flight heading from Damascus to London next monday at 09
29- I would like to book a one-way ticket 1 adult first class next to a window if possible on the flight heading from Damascus to London next monday at 09
30- I'd like to have a reservation for two persons in business class from Damascus to Paris between 07
31- I'd like to have a reservation for two adults (one-way ticket) in business class from Damascus to Paris on 1 April 2019 between 07
32- Hello I want to travel from Syria to Egypt Book two tickets please
33- Hello I want to travel from Syria to Egypt on Sunday next week Book two tickets i want the lowest price please
34- Hello I want to travel from Syria to Egypt on Sunday next week Book two tickets for me and my mother who is an old woman) want two seats next to the window and the lowest price please
35- I want to book a plane card from Syria to Iraq
36- Please i want to book one ticket to paris I prefer a seat beside the window with a mobile and labtop charger I want the food to be light and has a good tast No problem if there are any animals in the plane
37- I want to book a plane card from Syria to Iraq at the first class Normal level flight Halal food on board A window seat
38- I want to book a plane card from Syria to Iraq at the first class Moreover I want to sit beside the window Concerning the food I do not want any type of it However I may need some juice or coffee
39- I'd like to have a reservation for two adults (one-way ticket) in business class from Damascus to Paris on 1 April 2019 between 07
40- I am asking for a Round-trip I want to travel from Syria/ Damascus Airport to KSA/ Jeddah/King Abdul Aziz Airport I want to depart on 27/7/2015 and return on I5/I0/2015 Type and number of passengers 1 Adult I am looking for a ticket in the Economy class with the lowest price available

41- Book two round trip tickets from Lattakia to Maldives with all necessary transportations We want to depart on Wednesday the 7th and return on Thursday the 30th No children
42- I want to fly to China reserve a ticket for me
43- l'm from Syria and I want to reserve a two-way ticket to China I want the flight class to be tourism and the ticket of Iow price for adult
44- I want a round trip ticket from Syria to China It's a departing flight for one adult The flight class is tourism and the ticket type is low price I would like to have 2 vegetarian meals and to have access to the InterneI prefer a seat beside the window if possible?naIIy I want to enquire about the paymenthow should I pay?
45- I want to book a plane ticket to London I 'd like to leave between May 10th and 12th
46- I want to book a flight from Duplin to Karlsruhe on Wednesday evening
47- I would like to make a reservation for a flight to London on the 1st of July please
48- I am looking for a flight ticket to Las Vegas on Saturday first class please
49- I am planning to go to Miami for ten days and I need to book a flight I am thinking of leaving on June 15 and returning on June 24
50- I 'd like to book two round trip tickets to Paris I want to fly from Chicago on 15/11/2015 and let the return day open
51- I'd like to book the earliest flight for the day after tomorrow to Detroit first class please!
52- I 'd like to reserve a flight to Tokyo for the first of October I will fly economy class
53- I'd like to enquire about flights to Berlin from Frankfurt Airport please I'm off to a concert at the end of the month - Thursday 22nd until Tuesday 27th Could you tell me about the flight availability and prices?
54- I 'd like to book a flight from Munich to Köln on Tuesday 10 November I'd like to arrive in Köln by 15
55- I'd like to buy a ticket in first class ticket to Hamburg I want to leave in about 2 hours
56- I want to book a flight first class to Hamburg I want to leave today and return next Saturday
57- Do you have any flights to Madrid next Tuesday afternoon? Economy class
58- I need a flight to Rome next week I have to reach Rome by the 24th and return on 31st in the evening
59- I am traveling to Dubai and I would like to book a flight on November 14th I would like to fly out of Frankfurt airport
60- I want to buy a plane ticket I want to fly from Manchester and my destination is Amsterdam
61- I want to do my Christmas shopping in New York So I want to fly there from Frankfurt and I do not want it to be first class
62- I want to book a flight,first class to Hamburg I want to leave today and return next Saturday

B. Umfrage-Bahn

Tabelle B.2.: Nutzeranfrage über Bahnbuchung

Bahnbuchung-Anfrage
1- Book me a train ticket from Karlsruhe to Berlin on 5th February at around 02:00 pm There are 2 passengers We want to travel back one week later
2- I want to reserve a train ticket from Karlsruhe to Berlin I want to leave at 02/02/2016 and come back at 09/02/2016 I want to have first class for 2 adults fastest possible connection
3- I would like to take the train to London next Wednesday
4- I would like to take the train to London next Wednesday I require wireless Internet and if possible a window seat
5- I would like to take the train to London next Wednesday I require wireless Internet and if possible a window seat I need a firsrt class ticket
6- I want to purchase a train ticket to Berlin from Karlsruhe on the 27th of March
7- I would like to reserve a one-way first class train ticket for an adult from Karlsruhe Hbf to Munich Hbf on 10th February The train services should include vegetarian meals and allow traveling with pets
8- I'd like a ticket from Karlsruhe to Stuttgart for the 18th of November second class please
9- I need a return ticket from Mannheim to Karlsruhe I'm going on Saturday and coming back next Thursday
10- Book me a ticket to London I'm going on Tuesday and coming back next Monday I'm planning to go at 05:30 come back on either the 06:15 or the 06:45
11- I am headed London this next week I want a train ticket on at 14:00
12- I'd like a ticket to Karlsruhe in 20 minutes
13- I'd like a ticket to Karlsruhe I want to arrive there before 21:00
14- I would like to buy a one-way ticket to Heidelberg I am leaving next Monday at 10:00 and coming back on Tuesday at 8:00
15- A single ticket to Karlsruhe main station please
16- A sin gle ticket to Karlsruhe main station at 12:30 please
17- A single ticket to Karlsruhe main station at 12:30 second class please
18- Hello I would like to reserve two tickets to Berlin please for the 5:00 PM train
19- Is there a train from Karlsruhe to Frankfurt via Heidelberg leaving tomorrow at 4:00 PM
20- Find me a train to Freiburg after 2:00
21- Find me a train to Freiburg after 2:00 second class
22- Tell me when the next train from Frankfurt to Munich are leaving
23- What time does the next train to Karlsruhe leave
24- What is the soonest train from Karlsruhe to Paris leave?
25- Tell me the train schedule for tomorrow morning between London and Manchester
26- I would like to buy a a round ticket from Berlin to Karlsruhe I would like to leave on Monday and return on Friday
27- I want to book 2 tickets to Stuttgart please for the 5:00 PM train I want to travel second class
28- I'd like a second class return ticket to Stuttgart please I don't want to change trains I want to leave on 7:30 and return on 18:30
29- I'd like a ticket to Karlsruhe today please The train should arrive before 12:00 And I want to reserve a seat

30- I would like 2 first class tickets to Paris on 27 this month at 9:00
31- get me a ticket from Heidelberg to Berlin for the 18th of November please I want it to be Single ticket and second class I want to leave around 7:00 PM
32- Would you book a ticket to Köln for me please I want to leave on 14/11/2015 at 3:00 pm I want to return to Karlsruhe on 16/11/2015 at 4:00 The date and time are at the departure
33- I'd would like to buy a ticket to Aachen please tomorrow after 14:00 am I want to take the ICE and the ticket should be first class
34- Can I have a return ticket from Karlsruhe to Munich please The dates are 12/11/2015 and 16/11/2015 I want to go and back in the morning
35- I want to know the availability to Berlin I want to travel 18th or 19th this month The travel class is not important but it is necessary to be an ICE I want to return after 25th
36- could you please tell me if I can get a ticket to Frankfurt from Karlsruhe next Friday in the morning I want to travel via Mannheim
37- I want to get a ticket on the train leaving from Heidelberg to Karlsuhe via Bruchsal today at 17:00 pm
38- Could I get two tickets to Freiburg please for the 5:00 PM train I want to take the regional train
39- I need a train ticket tomorrow at 12:00 I am headed Heidelberg and I want to return the day after tomorrow at 17:00
40- Can I make a reservation for two passengers on the train heading towards Berlin today at 8:00 pm First class please
41- I would like to purchase a return ICE ticket from Stuttgart to Berlin I have a Bahn card with 50% discount The dates are 12/01/2016 at 12:00 and 17/01/2016 at 16:00 The dates are by the arrival
42- I need a return ticket from Karlsruhe to Hamburg I'm going on Saturday and coming back next Thursday I'm planning to leave at 15:00 and come back at 20:30
43- Can I get one ticket for the 9:00 PM train to Mannheim please I want to use my 25% bahn card
44- I would like to book a ticket to Paris I want it to be booked on Friday at 9:00
45- I would like to Know how much does it cost a train ticket from Karlsruhe to Berlin on Saturday at 13:00 Can I have a discount price
46- I want to go from Köln to Karlsruhe tomorrow morning I want to take only ICE and first class

C. Umfrage-Hotel

Tabelle C.3.: Nutzeranfrage über Hotelbuchung

Hotelbuchung-Anfrage
1- Book me a room in an hotel in Karlsruhe on 5th February for 2 days There are 2 persons
2- I want to book a hotel in Berlin for 2 adults I want to stay from 2/2 until 9/2
3- I want to book a hotel in Berlin for 2 adults I want to stay from 2/2 until 9/2 he hotel should have at least 4 stars and a pool internet and breakfast included I want a room with beach view
4- I would like to book a hotel in Downtown Manhattan next Wednesday
5- I need a suite with double bed in Downtown Manhattan on January 4
6- I need a suite with double bed and wireless internet in Downtown Manhattan ASAP
7- I want to book a room in a hotel in Berlin on the 27th of February I want my pet sleeping with me
8- I would like to reserve a luxury hotel room for an adult in Munich for five days I would be arriving in Munich on 14 February The room services should include internet facilities vegetarian meals
9- I would like to book a hotel for two weeks in August in Berlin starting on August 1st with two children
10- I need a hotel in London for two nights this weekend that has vegetarian food and wifi
11- I'd like to book a hotel room in Hong Kong for five nights
12- I want to reserve a room at a hotel in Stuttgart I need a room with a single bed that is large enough for one person to sleep in from 12/11/2015 until 14/11/2015
13- I'd like to book a room for 2 people for 2 nights starting on 03/12/2015
14- Do you have any rooms available for June 10th I'd like to make a reservation for 2 people with 2 kids
15- I'd like to book a double room for two nights from Monday 2 August to Wednesday 4 August and for two people
16- I'd like to book a hotel in Paris for 3 nights from 24th to 26th of this month
17- I'd like to book a room for two nights starting on January 12th
18- I'd like to make a reservation for a single room for the night of 15/11/2015 please
19- I would like to book a double room for the month of November 2015 I need an air-conditioned room with bath and shower which faces the sea
20- I am looking for a suitable hotel near the main station in Duplin for three days from 11th December
21- I want to book a room at the Sea Hotel in Hamburg for this weekend
22- Hello I would like to make a booking for a single room please from August the 14th to August the 25th
23- Can I reserve a single hotel room please? I will staying for 1 week from 05/12/2015
24- I would like to reserve 2 rooms in Manchester I will be needing the rooms until the 1st of January
25- I'd like to make a reservation for the third weekend in September There will be two of us and we'll be staying for two nights I would love to have an ocean view
26- I´d like to book a room in Las Vegas for November the 23rd I will be staying three nights

27- I'd like to book a room for tomorrow I have a conference in Hong Kong for 2 days The breakfast must be included
28- I was wondering if I can get a good room in Paris for the 12th of this month?
29- I would like a room for the 19th of July I am going to stay in Stuttgart for 3 days
30- I want to book a room in Köln tomorrow for 2 nights I want a double room with a shower
31- Would you mind reserving a nonsmoking room for me and my wife for 13/11/2015 We will be spending three nights
32- I'd like to make a reservation for October 12th That's four nights
33- can I reserve a room in Berlin on 21/01/2016 We are 2 students and we are going to need the room until January 23rd
34- I would like to book a hotel reservation in Rome The reservation is for May 14th and for 2 days
35- I need two rooms in Washington today for a total of 2 adults and 2 children for about 4 days
36- I'd like to book a single room for two nights from Next Monday please
37- I need to book a room in 4 seasons Hotel in Paris We are 3 people Our stay will be beginning on May 9th for 2 nights
38- I'd like to book a single room for this Friday to Sunday UBHNJ is my promotion code I think I become 25% discount for this code
39- Can you suggest a good hotel in Munich I want to book a room for 2 days I arrive on 08/01/2016
40- We would like to book a double room in Heidelberg from January 1st till January 5th without any services
41- Book me a room for tomorrow We are a couple and we want to have a breakfast at 7:30 We need the room for 5 days
42- I 'd like to book 2 rooms for the weekend of Januray 16th
43- I would like to make a hotel reservation in Karlsruhe I will be arriving on February 19th I need the room for 3 nights

D. Ontologie-Konzepte

Konzept	**Semantik (Ausdrücke)**
Places	Frankfurt, Stuttgart, Berlin, Karlsruhe, New York, Manchester, Amsterdam, London, Los Angeles, Paris, Washington,Durlach, Karlsruhe hbf, Berlin hauptbahnhof,Brandenburger tor
Departure_Prefix	from, departure is
Departure_Postfix	is the departure, is departure
Arrival_Prefix	to, towards, arrival is
Arrival_Postfix	is the arrival, is arrival
Adult_Prefix	number of adults is, adults number is
Adult_Postfix	adult, adults, person, persons, people, passenger, passengers, is the number of adults, is the adults number
Children_Prefix	number of children is, children number is, number of kinder is, kinder number is
Children_Postfix	child, children, kind, kinder, is the number of children, is the children number, is the number of kinder, is the kinder number
Babies_Prefix	number of babies is, babies number is
Babies_Postfix	baby, babies, is the number of babies, is the babies number
Flight_Class_Options	economy, premium economy, business, first
Flight_Class_Prefix	flight class is
Flight_Class_Postfix	class, is the flight class
Departure_Date_Prefix	depart on, departure date is, departure is on
Departure_Date_Postfix	is the departure date
Return_Date_Prefix	return on, back on, return date is, return is on
Return_Date_Postfix	is the return date, is the return
Helper_Flight	flight, flights, fly, airport, airways, airline, plane

Tabelle D.4.: Semantik aller Konzepte der aktiven Ontologie „Flight“

Tabelle D.5.: Semantik aller Konzepte der aktiven Ontologie „Train“

Konzept	**Semantik (Ausdrücke)**
Places	Frankfurt, Stuttgart, Berlin, Karlsruhe, New York, Manchester, Amsterdam, London, Los Angeles, Atlanta, Beijing, Dubai, Chicago, Tokyo, Denver, Madrid, Hong Kong, Delhi, Miami, Damascus, Aleppo, Latakia, Prag, Las Vegas, Beirut, Rome, Milan, Paris, Washington,Durlach, Karlsruhe hbf, Berlin hauptbahnhof, Brandenburger tor
Start_Prefix	from, start is, start station is
Start_Postfix	is the start, is start

Destination_Prefix	to, towards,destination is, destination station is
Destination_Postfix	is the destination, is destination
Means_Of_Transport	ice, only local transport, ec, ic, ir, regional, bus, suburban, boat, underground tram, dial a ride, taxi, all excluding ice
Accessibility	no stairs, no escalators, no elevators, low floor vehicles, vehicles with lift platform
Bicycle	bicycle, bike
BahnCard	bahn card
Reservation	reserve a seat, reservation a seat, seat reservation, seat, reservation
Passenger_Prefix	number of passengers is, passenger number is
Passenger_Postfix	passenger, passengers, is the number of passengers, is the passengers number
Travel_Class_Options	first, second
Travel_Class_Prefix	travel class is, class is
Travel_Class_Postfix	class, is the travel class
Journey_Date_Prefix	go on, travel on, ravel date is, date of journey is,date of trip is, leave on, depart on, departure date is,departure is on
Journey_Date_Postfix	is the travel date, is the trip date,is the journey date
Journey_Return_Date_Prefix	return on, back on, return date is, return, is, on
Journey_Return_Date_Postfix	is the return date, is the return
Journey_Time_Prefix	go at, travel at, travel time is, time of journey is, time of trip is, go on, travel on, leave on, travel time is, time of journey is, time of trip is, depart on, departure date is, departure is on
Journey_Time_Postfix	is the travel time, is the trip time, is the journey time
Journey_Return_Time_Prefix	return at, back at, return time is, return is at, return on, back on, return date is, return is on
Journey_Return_Time_Postfix	is the return time, is the back time
Departure_OR_Arrival	at the departure,at the start, at the arrival, at the destination, by the departure, by the arrival, by leaving, by departure, by departuring, by arriving, by arrival, departing before, departing by, leaving before, leaving by, arriving before, arriving by
Walking_Time_Prefix	walking time is, walking time
Walking_Time_Postfix	is the walking time, minutes is the walking time, min is the walking time
Interchange_Time_Prefix	interchange time is, interchange time
Interchange_Time_Postfix	is the interchange time, minutes is the interchange time, min is the interchangetime
Via_Station_Prefix	stopover at, via station, via

Via_Station_Postfix	is the stopover station
Helper_Train	train, trains, ice, bus

Konzept	**Semantik (Ausdrücke)**
Hotel_Place	Frankfurt, Stuttgart, Berlin, Karlsruhe, New York, Manchester, Amsterdam, London, Los Angeles, Paris, Washington,Durlach, Karlsruhe hbf, Berlin hauptbahnhof,Brandenburger tor
Arrival_Date_Prefix	arrive on, from, arrival date is, arrive is on
Arrival_Date_Postfix	is the arrival date, is the arrival
Departure_Date_Prefix	depart on, departure date is, departure is on
Departure_Date_Postfix	is the departure date, is the departure
Adult_Prefix	number of adults is, adults number is
Adult_Postfix	adult, adults, person, persons,people, passenger, passengers, is the number of adults, is the adults number
Children_Prefix	number of children is, children number is, number of kinder is, kinder number is
Children_Postfix	child, children, kind, kinder, is the number of children, is the children number, is the number of kinder, is the kinder number
Room_Prefix	number of rooms is
Room_Postfix	room, rooms, is the number of rooms
Hotel_Duration	day, days, night, nights
Hotel_Promotion_Code	promotion code is, is the promotion code
Helper_Hotel	hotel, room, rooms, suite, suites

Tabelle D.6.: Semantik aller Konzepte der aktiven Ontologie „Hotel“

Adults	**Children**	**Infants**	**Flight class**
Adults	Children	Infants	Class
Teens (neue Kategorie)			Flight class
Young adults			
Youth			

Tabelle D.7.: Konzeptenamen in der Fluggesellschaftformularen

Means of transport	**Via station**	**Interchange time**	**Walking time**
Means of transport	Via station	interchange time	Walking time
With	Via	Dwell-time	On foot
Route preferences	Travelling via		
Vehicle selection	Stopover		

Tabelle D.8.: Konzeptenamen der Bahnkategorie - Teil 1

Reservation	**Passengers**	**Travel class**	**BahnCard**	**Bicycle**	**Accessibility**
Reservation	Passengers	Travel class	BahnCard	Bicycle	Accessibility

Tabelle D.9.: Konzeptenamen der Bahnkategorie - Teil 2

E. Selenium: Web-Driver

Befehle	**Code**
Driver definieren	WebDriver driver = new FirefoxDriver();

Seiten abrufen	driver.get(”http://www.britishairways.com/”);
	
	driver.navigate().to(”http://www.britishairways.com”)
	driver.navigate().forward();
	driver.navigate().back();

Navigieren	driver.switchTo().window(”windowName”); driver.switchTo().frame(”frameName”);
Pop-up Dialoge	Alert alert = driver.switchTo().alert();

Cookies	Cookie cookie = new Cookie(”key”, ”value”);
	driver.manage().addCookie(cookie);
	Set<Cookie> allCookies = driver.manage().getCookies();
	driver.manage().deleteCookieNamed(”CookieName”);
	driver.manage().deleteCookie(loadedCookie);
	driver.manage().deleteAllCookies();

Formulare Einsenden	WebElement depCtr = driver.findElement(By.id(”depCountry”)); depCtr.sendKeys(”Germany”); WebElement retDate = driver.findElement(By.id(”retDate”)); retDate.submit(); submitButton.click ();

Tabelle E.10.: Selenium Befehle - Beispiele

BY	HTML Java Code
ID	<select id=”depCountry”> WebElement depCtr = driver.findElement(By.id(”depCountry”));
Class Name	17 Nov driver.findElements(By.className(”daymonth ”));
Tag Name	<iframe src=”abc”></iframe> driver.findElement(By.tagName(”iframe”));
Name	<input type=”radio” name=”journeyType” value=”RTFLT” /> driver.findElement(By.name(”journeyType”));
Link Text	 lufthansa driver.findElement(By.linkText(”lufthansa ”));
Partial Link Text	lufthansa flights driver.findElement(By.partialLinkText(”lufthansa”));
CSS	10:45 driver.findElements(By.cssSelector(”div[class=’outerBar’]”));
XPATH	<td class=” arrivaltrue”>11:40 driver.findElements(By.xpath(”//td[@class=’ arrivaltrue’]”));

Tabelle E.11.: WebElement finden durch HTML-Tags