Karlsruhe Institute of Technology

# Komplexe Abbildungen von Formularelementen zur Generierung von aktiven Ontologien

Bachelorarbeit
von


Dennis Weigelt

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: | M.Sc. Tobias Hey |

Bearbeitungszeit: 23.08.17 – 22.01.18

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, 22.01.18**

. . . . . .

(**Dennis Weigelt**)

**Kurzfassung**

Unser heutiges Leben wird zunehmend von Assistenzsystemen erleichtert. Hierzu gehören auch die immer häufiger verwendeten intelligenten Sprachassistenten wie Apple's Siri oder Microsoft's Cortana. Statt lästigem Flüge vergleichen auf diversen Internetportalen können Sprachassistenten durch die einfache Nutzereingabe „Buche für morgen einen Flug von Stuttgart nach Frankfurt" dieselbe Arbeit tun. Um Informationen verarbeiten und an den passenden Webdienst weiterleiten zu können, muss das Assistenzsystem natürliche Sprache verstehen und formal repräsentieren können. Hierfür werden bei Siri „aktive Ontologien (AOs)" verwendet. Eine Ontologie bildet Beziehungen zwischen Konzepten ab. Eine aktive Ontologie ist eine Ontologie, die Regeln für diese Beziehungen implementiert. Das Erstellen solcher AOs ist derzeit mit großem manuellem Aufwand verbunden, da für jede Kategorie eine eigene Ontologie erstellt werden muss. Das langfristige Ziel der Entwicklung solcher Sprachassistenzsysteme ist die Automatisierung des Generierungsprozesses von AOs. Die am KIT entwickelte Rahmenarchitektur EASIER [BL16] beschäftigt sich mit der automatischen Generierung von aktiven Ontologien aus Webformularen. Eine Herausforderung bei der Erstellung von AOs aus Webformularen ist die Zuordnung unterschiedlich ausgeprägter Formularelemente mit gleicher Semantik, da semantisch gleiche aber unterschiedlich realisierte Konzepte zu einem AO-Knoten zusammengefasst werden sollen. Es ist daher nötig, semantisch ähnliche Formularelemente identifizieren zu können. Diese Arbeit beschäftigt sich mit der automatischen Identifikation solcher Ähnlichkeiten und der Konstruktion von Abbildungen zwischen Formularelementen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1. Einleitung

Im Zuge der technischen Modernisierung werden immer häufiger intelligente Assistenzsysteme zur Erleichterung alltäglicher Prozesse eingesetzt. Hierzu gehören vor allem Sprachassistenzsysteme wie Apple's Siri, Microsoft's Cortana oder Google's GoogleNow. Durch Eingaben wie „Stelle den Wecker in 15 Minuten!“ können bereits einfache Aufgaben selbstständig vom Sprachassistenten ausgeführt werden. Hierfür muss natürliche Sprache vom Assistenzsystem verstanden werden.

Siri arbeitet hierfür mit „aktiven Ontologien (AOs)“ [Guz08]. Eine AO bildet Beziehungen zwischen Konzepten ab und enthält Logik, um diese Konzepte in natürlicher Sprache zu erkennen. Hierfür ist für jede Ontologie ein Regelsatz implementiert, der das Verhalten der AO steuert. Für jede Funktionalität wie beispielsweise das Stellen des Weckers muss eine neue AO erstellt werden. Dieser Prozess ist sehr mühsam, da die Erstellung von AOs derzeit mit großem manuellem Aufwand verbunden ist. Abbildung 1.1 zeigt stark vereinfacht die Eingabeverarbeitung einer AO für Flugbuchung. Die Knoten *Abflug*, *Ankunft* und *Zeit* erkennen hierbei Informationen in der Eingabe und leiten diese zu ihrem Elternknoten weiter. Dieser kombiniert dann die einzelnen Informationen zu einer gemeinsamen Information.

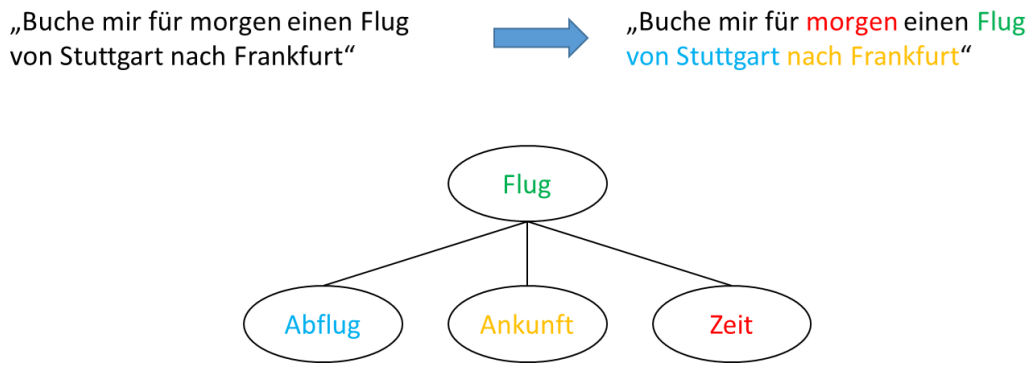



Abbildung 1.1.: Eingabeverarbeitung für die Kategorie „Flugbuchung“

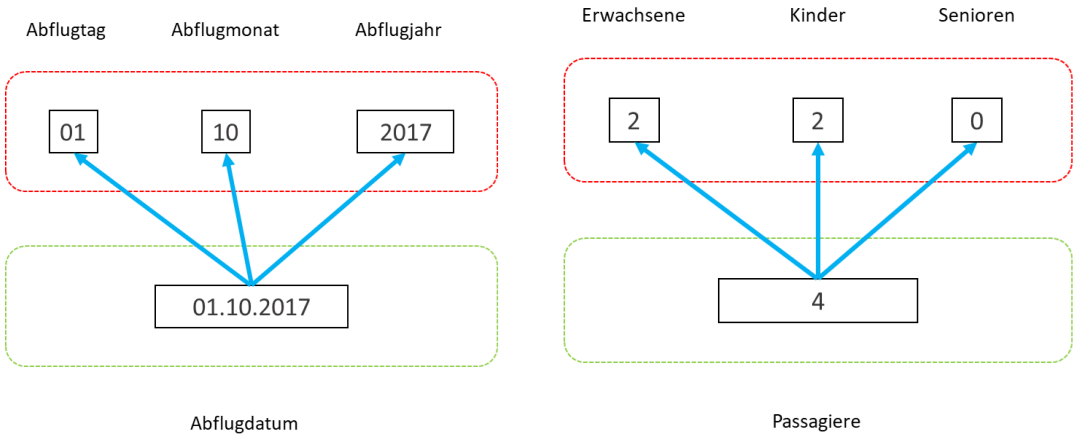
Das am Karlsruher Institut für Technologie (KIT) entwickelte Rahmenwerk EASIER bietet eine Möglichkeit diesen Generierungsprozess von AOs für formularbasierte Internetdienste zu vereinfachen, indem dieser weitestgehend automatisiert wird. Hierfür müssen semantisch ähnliche Formularelemente unterschiedlicher Formulare auf dasselbe Konzept in der AO abgebildet werden können. Für diese Arbeit sind zwei Arten von Abbildungen zwischen Formularelementen relevant: einfache (1:1) und komplexe (1:n) Abbildungen.

#### Einfache Abbildungen

Eine einfache Abbildung oder auch 1:1-Abbildung ist eine Zuordnung zwischen zwei unterschiedlich ausgeprägten Formularelementen mit gleicher Semantik. Abbildung 1.4a zeigt eine einfache Abbildung zwischen zwei Datumseingabefeldern.

#### Komplexe Abbildungen

Oftmals können einfache Abbildungen durch ihre Eindimensionalität jedoch nicht den gesamten semantischen Kontext abbilden. Um die Semantik vollständig erhalten zu können, werden komplexe Abbildungen verwendet. Durch komplexe Abbildungen bietet sich die Möglichkeit eine Zuordnung zwischen mehreren Formularelementen zu finden. So ist es möglich 1:n-Beziehungen abzubilden. Bei komplexen Abbildungen wird zwischen aggregierten Abbildungen und ist-ein-Abbildungen unterschieden [WYDM04].




(a) aggregierte Abbildung

(b) ist-ein-Abbildung

Abbildung 1.2.: Abbildungsarten komplexer Abbildungen [WYDM04]

Bei einer aggregierten Abbildung setzt sich das einzelne Element durch die Informationen der $n$ Abbildungspartner zusammen. Bei einer ist-ein-Abbildung ist jedes der $n$ Elemente eine Teilmenge des einzelnen Elementes. Abbildung 1.2 zeigt ein anschauliches Beispiel aus der Dienstkategorie Flugbuchung. Das „Abflugdatum“ setzt sich durch die einzelnen Elemente „Abflugtag“, „Abflugmonat“ und „Abflugjahr“ zusammen und ist daher eine aggregierte Abbildung. Die Elemente „Erwachsene“, „Kinder“ und „Senioren“ hingegen sind jeweils Teilmengen des Elementes „Passagiere“. Daher handelt es sich hier um eine ist-ein-Abbildung.

## 1.1. Zielsetzung

Das übergeordnete Ziel dieser Arbeit ist die Identifikation komplexer Abbildungen zwischen Formularelementen. Derzeit kann das Rahmenwerk EASIER lediglich einfache Abbildungen zwischen Formularelementen finden. In vielen Fällen sind einfache Abbildungen jedoch

nicht mächtig genug, um den gesamten semantischen Kontext korrekt abzubilden. Um die Genauigkeit der Abbildungen und damit die Effizienz der generierten AO in EASIER verbessern zu können, sollen im Rahmen dieser Arbeit komplexe Abbildungen identifiziert werden. Mithilfe dieser Abbildungen können später semantisch gleiche aber unterschiedlich realisierte Konzepte verschiedener Formulare auf einen AO-Knoten abgebildet werden. Im Vordergrund steht hierbei ein möglichst maximaler Automatisierungsgrad. Es soll ein Werkzeug entwickelt werden, welches potenzielle komplexe Abbildungen identifizieren und lösen kann. Eine Lösung einer solchen Abbildung ist eine Abbildungsvorschrift der gefundenen Elemente auf AO-Knoten in EASIER. Hierzu werden konkrete Muster zur Lösung von Anwendungsbeispielen entwickelt. Abbildung 1.3 zeigt die Zielsetzung der Bachelorarbeit. Das Werkzeug erhält eine Menge von Formularen einer Dienstkategorie als Eingabe. Nach dem Identifikationsprozess werden dann potenzielle komplexe Abbildungen ausgegeben. Diese werden dann manuell auf Muster untersucht, um Abbildungsvorschriften auf AO-Knoten zu konstruieren.

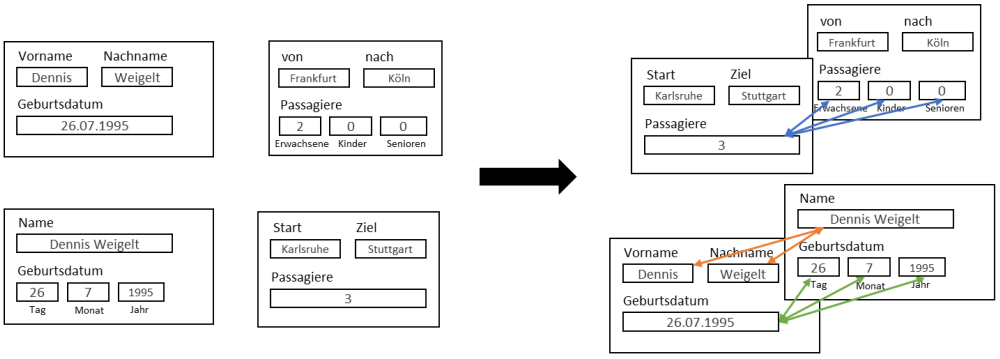
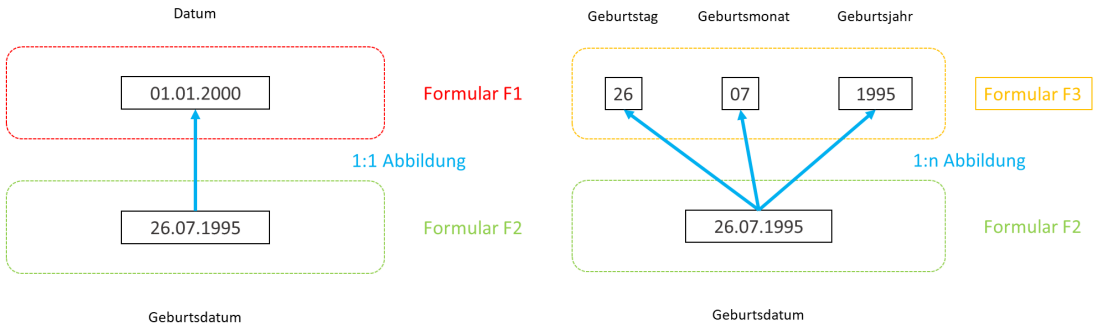



Abbildung 1.3.: Einordnung der Zielsetzung der Bachelorarbeit.

## 1.2. Beispiele

Abbildung 1.4 zeigt drei Formulare mit unterschiedlich realisierten Eingabemöglichkeiten für ein Datum. Die einfache Abbildung kann bereits von EASIER gefunden werden, jedoch ist es nicht möglich zwischen dem Feld „Geburtsdatum“ aus Formular F2 und den Feldern „Geburtstag“, „Geburtsmonat“ und „Geburtsjahr“ den komplexen Zusammenhang zu erkennen. Somit kann die AO nicht erkennen, dass es sich bei den Eingabefeldern aus Formular F3 um dieselbe Information wie in Formular F1 und Formular F2 handelt. Hierbei kommt es bei der Generierung der AO zu Ungenauigkeiten.




(a) 1:1-Abbildung (b) 1:3-Abbildung

Abbildung 1.4.: Vergleich einer einfachen und einer komplexen Abbildung

Beim Betrachten einiger Beispielformulare fiel auf, dass komplexe Abbildungen sehr häufig auftreten. Eine Auswahl von Abbildungsarten ist in Abbildung 1.5 dargestellt.

| Nr. | Formular A | | Formular B |
|---|---|---|---|
| 1 | PLZ: 76139 | Ort: Karlsruhe | PLZ und Ort: 76139 Karlsruhe |
| 2 | Stunde: 19 | Minute: 30 | Uhrzeit: 19:30 |
| 3 | Datum: 21.03.17 | Uhrzeit: 19:30 | Datum und Uhrzeit: 21.03.17 19:30 |
| 4 | Erwachsene: 2 | Kinder: 2 | Passagiere: 4 |

Abbildung 1.5.: Vergleich einer einfachen und einer komplexen Abbildung

## 1.3. Struktur der Arbeit

Die Arbeit ist wie folgt gegliedert. Zuerst werden die zum Verständnis der Arbeit benötigten Grundlagen eingeführt, sowie der aktuelle Stand der Forschung präsentiert. In Kapitel 5 wird die Problemstellung analysiert und es werden Möglichkeiten zur Identifikation komplexer Abbildungen vorgestellt und diskutiert. Kapitel 6 dokumentiert den Entwurf und die Implementierung des entwickelten Werkzeuges. Schließlich wird das entwickelte Identifikatonsverfahren in Kapitel 7 anhand bekannter Metriken evaluiert. Zum Schluss wird das Ergebnis dieser Arbeit zusammengefasst und ein kurzer Ausblick auf weiterführende Fragestellungen und Verbesserungsmöglichkeiten skizziert.

# 2. Grundlagen

In diesem Kapitel werden essenzielle Grundlagen eingeführt, die das Verständnis dieser Arbeit ermöglichen. Zuerst werden HTML-Formulare erläutert sowie deren Aufbau und Funktionsweise erklärt. Danach werden Aktive Ontologien (AOs) und Active Semantic Networks vorgestellt.

## 2.1. HTML-Formulare

In diesem Abschnitt werden HTML-Formulare eingeführt. Ein Formular ist eine Komponente einer Webseite, die Benutzereingaben abfragt und an einen Dienst weiterleitet. Ein Formular kann mehrere Formularelemente wie beispielsweise Checkboxes, Radio Buttons oder Textareas enthalten. Diese Elemente bilden innerhalb des `<form>` Tags das Formular. In den folgenden Abschnitten werden Komponenten von HTML4 [noab] und HTML5 [noac] vorgestellt.

### 2.1.1. Attribute

HTML-Elementen können durch Attribute weitere Eigenschaften übertragen werden. Attribute sind immer an ein zugehöriges Element gebunden und können nur innerhalb eines Elementes verwendet werden. Man unterscheidet Formularattribute, Listenattribute, Medienattribute, Metaattribute, Tabellenattribute, Verweisattribute und globale Attribute. Im Folgenden werden die für diese Arbeit relevanten Attribute vorgestellt. Für das Arbeiten mit Formularen werden lediglich Formularattribute und globale Attribute benötigt. Attribute werden als global bezeichnet, wenn sie in jedem HTML-Element verwendet werden können. Die wichtigsten globalen Attribute sind in Tabelle 2.1 dargestellt.

| Attribut | Funktion |
|---|---|
| `id` | Identifiziert ein eindeutiges Element in einem Dokument. |
| `class` | Ein Element kann einer oder mehrerer Klassen zugeordnet werden. |
| `hidden` | Dieses Element wird ausgeblendet. |
| `contenteditable` | Dieses Attribut legt fest, ob der Inhalt eines Elementes verändert werden darf. |
| `lang` | Legt die Sprache innerhalb des Elementes fest. |
| `spellcheck` | Bestimmt, ob die browserinterne Rechtschreibprüfung aktiviert sein soll. |
| `style` | Durch dieses Attribut können CSS-Eigenschaften für das Element festgelegt werden. |
| `accesskey` | Es kann ein Zeichen auf der Tastatur bestimmt werden, womit dieses Element angesprungen werden kann. |
| `contextmenu` | Legt fest, welches Kontextmenü mit einem Rechtsklick geöffnet werden soll. |

Tabelle 2.1.: Globale HTML-Attribute

Formularattribute sind Attribute, die ausschließlich für Formularelemente verwendet werden können. Hierbei ist es zusätzlich möglich, dass einzelne Attribute nur bei bestimmten Elementen zugelassen sind. Tabelle 2.2 zeigt alle Formularattribute, welche innerhalb der in Abschnitt 2.1.2 vorgestellten Formularelementen verwendet werden können.

### 2.1.2. Formularelemente

Beim Erstellen eines Formulars stehen viele verschiedene Elemente zur Verfügung. In den folgenden Abschnitten werden Formularelemente und ihre Attribute vorgestellt, die bei der Identifikation komplexer Abbildungen eine größere Rolle spielen könnten.

#### 2.1.2.1. Label

Das Element `<label>` fungiert als Bezeichner von Formularelementen in der Nutzeroberfläche. Es gibt zwei verschiedene Möglichkeiten Labels zu setzen. Um einem Element einen Bezeichner zuzuordnen, kann einerseits das Attribut `for` verwendet werden. Hierdurch werden Bezeichner und Formularelement über das Attribut `id` verknüpft. Eine weitere Option ist, das zu bezeichnende Element in das Tag `<label>` zu schachteln. Abbildung 2.1 zeigt die beiden Möglichkeiten zur Darstellung eines Bezeichners in HTML.

```
<label for="dep_city">Abflughafen</label>
<input type="text" id="dep_city"  name="departureCity"/>


<label> Geburtsdatum <input type="text" id="birthd" name="birthday"/>
</label>
```

Abbildung 2.1.: Möglichkeiten zur Verwendung von `label` in HTML.

#### 2.1.2.2. Input

Das Element `<input>` ist ein typisiertes Datenfeld, welches vom Nutzer zur Eingabe von Formulardaten verwendet wird. Zur Typisierung wird das Attribut `type` verwendet, wodurch der Wertebereich des Eingabeelementes eingeschränkt wird. Die wichtigsten Typen sind in Tabelle 2.3 aufgeführt.

| Attribut | Formularelemente | Funktion |
|---|---|---|
| `accept` | `input` | Legt fest, welche Dateitypen für einen Dateiupload akzeptiert werden. |
| `checked` | `input` | Legt eine Vorauswahl bei Auswahlelementen fest. |
| `cols` | `textarea` | Gibt die Breite des Textfeldes in Zeichen an. |
| `disabled` | `button, input, option, select, textarea` | Das Element wird ausgegraut. |
| `for` | `label` | Ordnet einem `input`-Feld ein Label zu. |
| `max` | `input` | Gibt die maximale Eingabe an. |
| `maxlength` | `input, textarea` | Legt die zulässige Maximalanzahl an Zeichen fest. |
| `min` | `input` | Gibt die minimale Eingabe an. |
| `multiple` | `input, select` | Erlaubt mehrfache Eingaben. |
| `name` | `button, input, select, textarea` | Legt einen Namen für das Element fest. |
| `pattern` | `input` | Die Eingabe wird anhand eines angegebenen regulären Ausdrucks geprüft. |
| `placeholder` | `input, textarea` | Der Platzhalter gibt eine Beispieleingabe an. |
| `readonly` | `input, textarea` | Schützt das Element vor Veränderungen. |
| `required` | `input, select, textarea` | Das Element wird zum Pflichtfeld und muss ausgefüllt werden. |
| `rows` | `textarea` | Gibt die Höhe des Textfeldes in Zeilen an. |
| `selected` | `option` | Es wird eine Option vorselektiert. |
| `size` | `input, select` | Bestimmt die Länge des Eingabefeldes (`input`) oder die Anzahl der Optionen (`select`). |
| `step` | `input` | Legt Intervallabstände zwischen Zahlen fest. |
| `type` | `button` | Legt den Typ eines Buttons fest. |
| | `input` | Legt den Typ eines Eingabefeldes fest. |
| `value` | `button, option, input` | Legt einen Wert für das Eingabefeld fest. |

Tabelle 2.2.: HTML-Formularattribute

| **Art der Eingabe** | **Typ** | **Funktion** |
|---|---|---|
| Texteingabe | `text` | Es wird eine einzeilige Texteingabe erwartet. |
| | `search` | Dieser Typ stellt ein Suchfeld dar und ist ebenfalls ein einzeiliges Texteingabefeld. |
| | `tel` | Es wird eine Telefonnummer erwartet. Hierbei wird keine bestimmte Syntax erzwungen. |
| | `url` | Es wird eine URL erwartet. Es wird eine korrekte URL-Syntax erzwungen. |
| | `email` | Es wird eine Email-Adresse erwartet. Es wird eine korrekte Syntax erzwungen. |
| | `password` | Das Formular erwartet eine Passworteingabe. |
| Zeiteingabe | `date` | Es wird eine Datumseingabe erwartet. |
| | `time` | Es wird eine Uhrzeiteingabe ermöglicht. |
| | `datetime` | Erwartet ein Datum mit Uhrzeit mit Einbeziehung der Zeitzone. |
| | `datetime-local` | Erwartet ein Datum mit Uhrzeit ohne Berücksichtigung der Zeitzone. |
| | `week` | Es werden eine Woche und ein Jahr als Eingabe erwartet. |
| | `month` | Es werden ein Monat und ein Jahr als Eingabe erwartet. |
| Zahleneingabe | `number` | Die Eingabe wird als Zahl interpretiert. |
| | `range` | Erwartet einen Zahlenbereich als Eingabe. |
| Auswahlmöglichkeiten | `radio` | Es kann eine Auswahl zwischen mehreren Auswahlbuttons mit derselben ID getroffen werden. |
| | `checkbox` | Es kann eine Mehrfachauswahl zwischen mehreren Auswahlbuttons mit derselben ID getroffen werden. |
| Benutzerinteraktionen | `button` | Stellt einen Knopf dar, der eine Nutzerinteraktion implementiert. |
| | `submit` | Das Formular wird abgesendet. |
| | `reset` | Das Formular wird auf den Ursprungszustand zurückgesetzt. |
| | `image` | Erzeugt einen grafischen `submit`-Knopf. |
| | `file` | Das Hochladen einer Datei wird ermöglicht. |
| | `color` | Es kann eine Farbauswahl getroffen werden. |

Tabelle 2.3.: Typen des `<input>` Elementes

#### 2.1.2.3. Textarea

Das Element `<textarea>` bezeichnet ein mehrzeiliges Textfeld, welches durch die Attribute `rows` und `cols` definiert wird. Es besteht außerdem die Möglichkeit, dem Nutzer die Bearbeitung dieses Feldes mit dem `readonly`-Attribut zu verweigern.

#### 2.1.2.4. Select

Auswahllisten können durch das Element `<select>` dargestellt werden. Der Nutzer trifft eine Auswahl aus einer vordefinierten Liste von mehreren Optionen, die jeweils mit `<option>` gekennzeichnet werden. Mehrfachauswahl kann mit dem Attribut `multiple` erlaubt werden. Das Attribut `size` gibt die Anzahl gleichzeitig angezeigter Optionen im Aufklappmenü an.

### 2.1.3. Formularübermittlung

Nach dem Abschicken eines Formulars, wird das Formular unter der in `method` angegebenen Methode an die in `action` spezifizierte Zieladresse gesendet. Diese Attribute werden im Tag `<form>` verwendet. Die Zieladresse wird als URL angegeben. Das Attribut `method` besitzt zwei mögliche Werte. Während der Wert *get* die *HTTP GET* Methode ausführt, wird beim Wert *post* die *HTTP POST* Methode ausgeführt.

## 2.2. Aktive Ontologien

Eine Ontologie ist im Kontext der Informatik eine Datenstruktur, in der Beziehungen zwischen Konzepten abgebildet werden. Hierdurch wird Wissen formal dargestellt. Bezeichnet man eine Ontologie nun als „aktiv“, wird die Datenstruktur um eine Ausführungsumgebung erweitert. Abbildung 2.2 zeigt den strukturellen Aufbau einer aktiven Ontologie (AO). Eine AO repräsentiert wie eine gewöhnliche Ontologie ein bestimmtes Wissen durch Konzepte und deren Beziehungen zueinander, jedoch implementiert bei einer AO jedes Konzept einen Satz an Regeln, die das Verhalten der AO steuern. Zusätzlich zu den Regelsätzen gibt es in einer aktiven Ontologie einen Faktenspeicher, der Fakten enthält, welche dann im Auswertungszyklus durch Regeln verarbeitet werden.

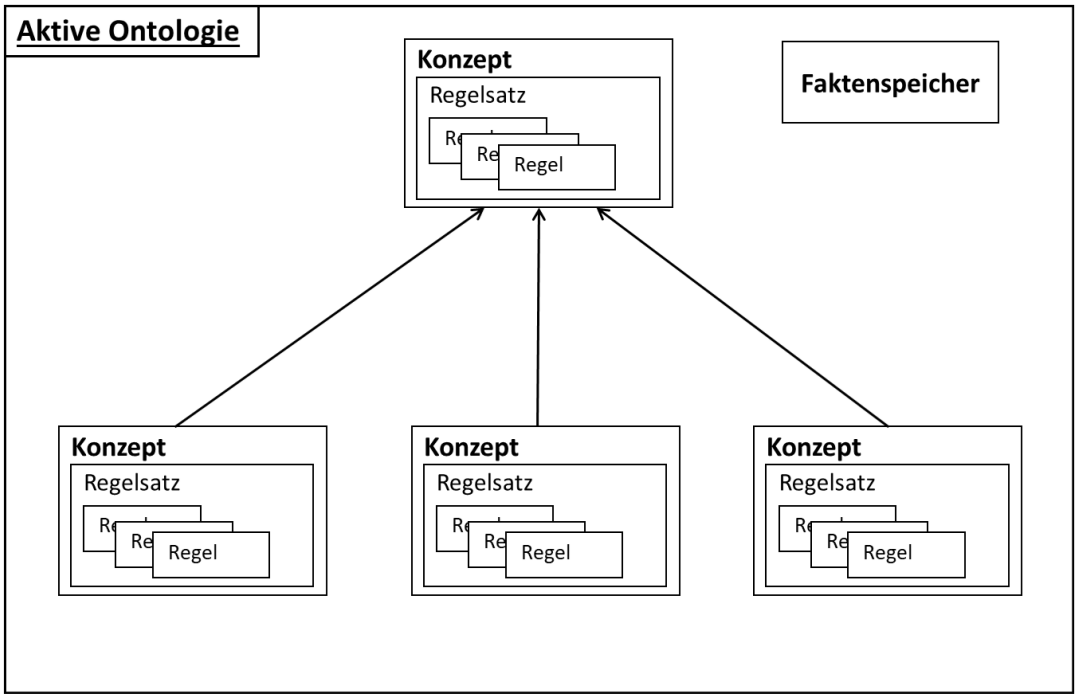



Abbildung 2.2.: Aufbau einer AO nach Guzzoni [Guz08]

### 2.2.1. Fakten und Faktenspeicher

Der Faktenspeicher dient als Speicherort für alle relevanten Fakten und repräsentiert hierdurch den Zustand der AO. Fakten können neu erstellt, verändert und wieder gelöscht werden. Dies wird als Lebenszyklus des Fakts bezeichnet. Nach Guzzoni [Guz08] gibt es vier verschieden Arten von Fakten.

#### Einfache Fakten

Einfache Fakten sind atomare Konstanten.
*Beispiel:* `12`, `test`

#### Komplexe Fakten

Komplexe Fakten sind Prädikate mit einem oder mehreren Argumenten. Argumente können wiederum Fakten sein.
*Beispiel:* `friend(max, tim)`, `coord(x(1),y(5),z(2))`

#### Listenfakten

Listenfakten sind unsortierte Listen von Fakten.
*Beispiel:* [`12`, `friend(max, tim)`], [`a`,`b`,`c`]

#### Variablen

Variablen können mit einem $-Zeichen definiert werden. Anonyme Variablen oder Platzhalter werden einfach als $ geschrieben.
*Beispiel:* `$friend`, `$A`, `$`

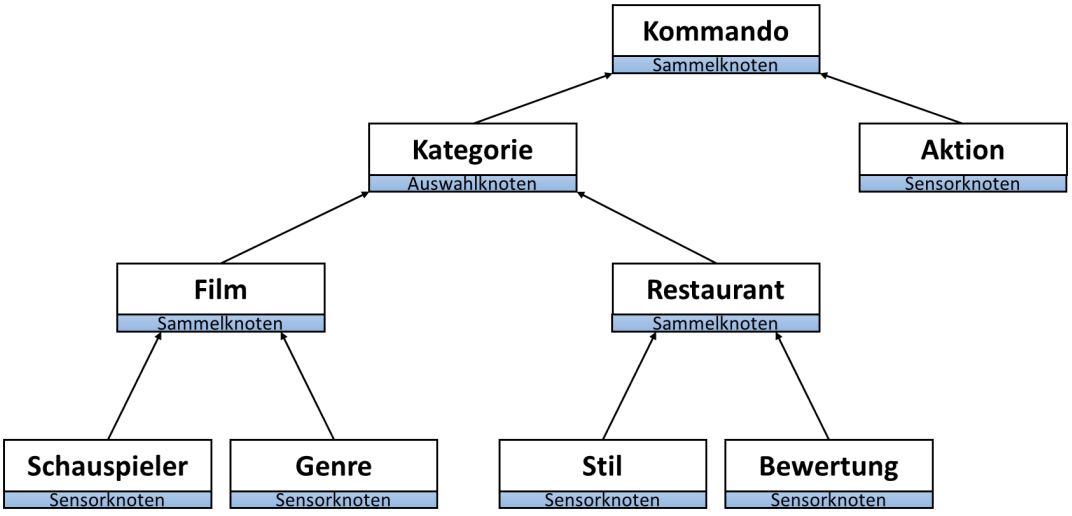



Abbildung 2.3.: Aufbau einer AO nach Guzzoni

### 2.2.2. Auswertung einer AO

Die Auswertung einer AO wird durch eine Veränderung im Faktenspeicher angestoßen. Sobald eine Veränderung im Faktenspeicher festgestellt wird, startet der Auswertungszyklus. Hierbei werden die Regeln der Konzepte mit den Daten im Faktenspeicher evaluiert. Eine Regel besteht hierbei aus einer Bedingung und einer Aktion. Bei der Evaluation wird zunächst die Bedingung überprüft. Trifft diese zu, wird die zugehörige Aktion ausgeführt. Der Faktenspeicher kann hierbei wieder verändert werden und weitere Auswertungszyklen anstoßen.

### 2.2.3. Active Semantic Networks

Active Semantic Networks [Guz08] werden zur Sprachverarbeitung verwendet. Der Erstellungsprozess solcher Netzwerke lässt sich in zwei wesentliche Schritte unterteilen. Zuerst wird mithilfe von realen Konzepten und Beziehungen eine Ontologie modelliert und in einer Baumstruktur repräsentiert. Dieser Baum stellt den Informationsfluss der AO dar. Die Knoten des Baumes repräsentieren hierbei die Konzepte. Die Kanten repräsentieren die Beziehungen der Ontologie. Im zweiten Schritt werden dann Ausführungselemente bestimmten Konzepten hinzugefügt, wodurch eine AO zur Sprachverarbeitung entsteht. Die Sprachverarbeitung wird bottom-up durchgeführt. Dies bedeutet, dass Informationen nur von Kindknoten zu ihren Elternknoten fließen. Die Blätter des Baumes, die Sensorknoten, erkennen hierbei durch die implementierte Logik bestimmte Informationen, die dann an ihre jeweiligen Elternknoten weitergeleitet werden. Es gibt außer dem Wurzelknoten noch zwei weitere Arten von internen Knoten, die Sammelknoten und die Auswahlknoten. Sammelknoten führen hierbei die erhaltenen Informationen der Kindknoten zu einer Information zusammen, während Auswahlknoten die beste ankommende Information auswählen. Beide Knotenarten leiten danach ihre Informationen wiederum an ihre Elternknoten weiter. Beim Wurzelknoten ankommend liegen dann alle Informationen vor und die erhaltene Information kann weiterverarbeitet werden. Abbildung 2.3 veranschaulicht ein beispielhaftes Netzwerk. In diesem Netzwerk werden Informationen über Filme, Restaurants und eine auszuführende Aktion durch die Sensorknoten gesammelt. Die Film- und Restaurantinformationen werden dann in den jeweiligen Sammelknoten zu einer Information vereint. Der Kategorieknoten wägt dann ab, um welche der beiden Kategorien es sich wohl handelt und leitet diese Information an den Kommandoknoten weiter. Dieser wiederum vereinigt als

Wurzelknoten die erhaltenen Informationen und führt die jeweilige Aktion für die Informationen der Kategorie aus. Die Güte der Informationen hängt vor allem von der Effizienz der Sensorknoten ab. Es werden verschiedene Typen von Sensorknoten unterschieden.

#### Vokabularknoten

Der Vergleich mit einem bekannten Wörterbuch ist die einfachste Art Informationen zu filtern. Hierbei besteht jeder Eintrag im Wörterbuch aus einem semantischen Wert und möglichen Synonymen. Zum Vergleich zwischen Wörterbucheinträgen und zu filternden Zeichenketten wird die Levenshtein-Distanz [MVM09] als Maß verwendet.

#### Präfix- und Postfixknoten

Diese Knotenart verwendet vorhergehende (Präfix) und nachstehende (Postfix) Signalwörter um Informationen zu klassifizieren. So könnte beispielsweise das Präfix *nach* im Kontext einer Flugbuchung die Information „Zielflughafen“ indizieren. Das Postfix *Euro* wäre demnach ein Indikator für einen „Geldbetrag“.

#### Knoten mit regulären Ausdrücken

Zur Erkennung von Informationen können ebenfalls reguläre Ausdrücke definiert werden. Hierdurch können bestimmte Muster effizient erkannt werden. Ein Beispiel hierfür wäre die Erkennung deutscher Postleitzahlen durch einen regulären Ausdruck, welcher fünfstellige Zahlen akzeptiert.

#### Spezialisierte Knoten

Manche Knoten hingegen kapseln lediglich eine vorher festgelegte Logik um spezielle Informationen zu filtern. Beispielsweise werden Sensorknoten zur Erkennung von Datum und Uhrzeit oft durch diese Art von Knoten implementiert. Hierdurch ist es möglich, Daten und Uhrzeiten wie *Montag um 15 Uhr* oder *morgen mittag* zu erkennen. Hierzu reagieren die AOs auf vorher implementierte Signalwörter.

# 3. Verwandte Arbeiten

Das übergeordnete Ziel dieser Arbeit ist das Erkennen und Lösen von komplexen Abbildungen zwischen HTML-Formularen, sodass komplexe Abbildungen konkreten Unterkonzepten von AOs zugeordnet werden können. Die meisten bestehenden Arbeiten beschäftigen sich lediglich mit der Identifikation einfacher Abbildungen. Um die Genauigkeit der Abbildungen zu verbessern, werden komplexe Abbildungen verwendet. Das bedeutet konkret, dass komplexe Zusammenhänge zwischen mehreren HTML-Formularelementen erkannt und zugeordnet werden müssen. Hierzu werden in diesem Kapitel Arbeiten vorgestellt, die sich mit den folgenden Themengebieten beschäftigen:

- Extraktion von Formularelementen
- Identifikation komplexer Abbildungen
- Zuordnungen von Ontologien

Da auf HTML-Formularen nicht direkt gearbeitet werden kann, müssen die Formularelemente zuerst extrahiert werden. Hierbei ist wichtig, dass die Semantik der einzelnen Formularelemente erhalten bleibt. Danach können komplexe Abbildungen identifiziert werden. Da es später auch sinnvoll sein kann, komplexe Abbildungen zwischen Ontologien zu finden, werden im Folgenden auch Ansätze zur Zuordnung und Verschmelzung von Ontologien vorgestellt.

## 3.1. Extraktion von Formularelementen

Damit Formularelemente ohne semantische Verluste weiterverarbeitet werden können, müssen Informationen aus den Eingabeformularen extrahiert werden. Im Folgenden werden zwei Ansätze zur Extraktion von Formulardaten vorgestellt.

### 3.1.1. Hierarchische Darstellung von Formularen

Für die hierarchische Darstellung von Formularen spielt vor allem die Platzierung der Formularelemente eine große Rolle [WYDM04]. Oft findet man semantisch ähnliche Elemente nahe beieinander gruppiert. Diese Tatsache wird genutzt, um semantische Gruppen von Formularelementen innerhalb eines Formulares zu identifizieren. Diese Gruppen können dann wieder Bestandteil einer größeren semantischen Gruppe sein, wodurch das Formular am Ende vollständig in einer hierarchischen Baumstruktur repräsentiert ist. Die Blätter des Baumes sind hierbei Formularelemente und die internen Knoten referenzieren entweder Gruppen oder Supergruppen. Die Überführung von Formularen in eine solche Baumstruktur wird meist manuell durchgeführt.

### 3.1.2. Automatische Extraktion

Oftmals ist es sehr aufwändig Formularelemente manuell zu extrahieren, da jedes einzelne Formular von Hand bearbeitet werden muss. Der Ansatz aus „Understanding Web Query Interfaces: Best-Effort Parsing with Hidden Syntax“ von Zhen Zhang et al. [ZHC04] beschäftigt sich mit der automatischen Extraktion von Formularelementen aus HTML-Formularen. Die Extraktion basiert auf der Beobachtung, dass Formulare immer aus ähnlichen Bausteinen bestehen. Ausgehend von dieser Beobachtung wird die Hypothese aufgestellt, dass es eine versteckte Syntax bei der Erstellung von Webformularen gibt. Durch diese Annahme ist es möglich, Formulare visuell mit einer kontextfreien Grammatik zu abstrahieren, wodurch das Verständnis dieser Sprache ein Parsingproblem wird. Diese visuelle Sprache wird durch die Produktionen der kontextfreien Grammatik repräsentiert, welche Präferenzregeln implementiert. Der resultierende Parsingbaum ist dann die Interpretation des Formulars. Dadurch, dass die erstellte Grammatik mehrdeutig und unvollständig sein kann, entstehen durch viele unvollständige Parsingbäume auch mehrere Interpretationen des Formulars. Daher muss der Parser unvollständige Bäume maximieren und zusammenführen können, um so eine möglichst gute Interpretation des Formulars zu erstellen. Außerdem darf der Parser aufgrund dieser Unvollständigkeit keine Eingaben als illegal ablehnen. Folgende Schritte werden vom Extraktor nacheinander ausgeführt: Tokenisierung, Parsing und die Zusammenführung der verschiedenen Parsingbäume. Der Ansatz erreicht mit einer einfachen Beispielgrammatik eine Präzision von über 85% auf zufälligen Quellen. Diese Effizienz auf unbekannten Quellen macht den Ansatz zu einer interessanten Möglichkeit der automatischen Extraktion von Formularelementen.

## 3.2. Identifikation komplexer Abbildungen

In den folgenden Arbeiten werden Ansätze zur Integration von Webformularen vorgestellt. Dies ist vor allem im Deep Web von großer Bedeutung, da im dort nicht mit Suchmaschinen gearbeitet werden kann. Durch die Integration von Webformularen wird ein einheitlicher Zugriff ermöglicht. Das größte Problem bei der Integration von Webformularen ist die Identifikation semantisch ähnlicher Formularbausteine. Hierbei reicht es oft nicht aus, einfache Abbildungen zwischen einzelnen Formularelementen zu betrachten, da diese oftmals nicht den gesamten semantischen Kontext abbilden können. Dieser Nachteil wird durch komplexe Abbildungen gelöst. Die erste Arbeit handelt von der Identifikation von semantischen Ähnlichkeiten und der Erkennung von Abbildungen mithilfe eines Clusteringverfahrens über diese Ähnlichkeiten. In der zweiten Arbeit wird ein Werkzeug präsentiert, welches aus verschiedenen Formularen Gemeinsamkeiten extrahiert und ein globales Formular erstellt. Die dritte Arbeit beschäftigt sich mit dem Mining von Korrelationen zur Identifikation komplexer Abbildungen aus Formularen.

### 3.2.1. Hierarchisches Clustering

Das Paper „An Interactive Clustering-based Approach to Integrating Source Query Interfaces on the Deep Web“ von Wensheng Wu et al. [WYDM04] stellt einen geschickten Ansatz zur Integration von Webformularen vor. Dieser Ansatz unterscheidet sich in ein paar Punkten signifikant von bestehenden Arbeiten. Diese leiden vor allem unter folgenden Problemen:

- Formulare werden durch zu flache Schemata modelliert.
- Es werden ausschließlich einfache Abbildungen betrachtet.
- Der Integrationsprozess findet Blackbox-artig statt.
- Es ist ein enormer Aufwand zum Einstellen der Parameter notwendig.

Diese Probleme werden im Ansatz vollständig aufgegriffen und gelöst. Zur Lösung dieser Probleme trägt vor allem die hierarchische Darstellung von Formularen, sowie die Einbindung des Benutzers in den Integrationsprozess bei. Zuerst werden die Formulare hierarchisch repräsentiert (vgl. Abschnitt 3.1.1), wodurch semantische Zusammenhänge leichter erkennbar sind als bei flachen Schemata. Danach werden semantische Ähnlichkeiten identifiziert, die anhand eines interaktiven, hierarchischen Verfahrens in Cluster eingeordnet werden.

#### Semantische Analyse

Um semantische Ähnlichkeiten feststellen zu können, muss eine umfassende Analyse durchgeführt werden. Jedes Formularelement wird dafür durch die Eigenschaften Name, Label und Wertebereich charakterisiert. Zwei Formularelemente sind genau dann ähnlich, wenn ihre Eigenschaften ähnlich sind. Um Ähnlichkeit zwischen Formularelementen zeigen zu können, werden zunächst die Eigenschaften Name und Label linguistisch auf Ähnlichkeiten geprüft und eine Analyse der Wertebereiche durchgeführt.

Bevor die linguistische Ähnlichkeit bestimmt werden kann, müssen die Namen und Label normalisiert werden. Zum Normalisieren werden zuerst Stoppwörter entfernt. Danach werden konkatenierte Namen und Label mit einem Domänenwörterbuch tokenisiert und anschließend transformiert. Somit wird zum Beispiel das Label „deptCity“ in „dept“ und „city“ zerteilt, falls „city“ ein bereits bekanntes Token im Wörterbuch ist. In der anschließenden Transformation würde dann „dept“ mithilfe des Wörterbuchs zu „departure“ erweitert werden. Die linguistische Ähnlichkeit setzt sich dann aus den Namens- und Labelähnlichkeiten zusammen.

Für die Wertebereichsanalyse werden zunächst die Datentypen der Wertebereiche verglichen. Anschließend wird die Ähnlichkeit der konkreten Werte im Wertebereich überprüft.

#### Finden komplexer Abbildungen

Die Identifikation von komplexen Abbildungen wird in drei Phasen durchgeführt. In der ersten Phase, der Vorverarbeitung, wird eine initiale Menge an komplexen Abbildungen bestimmt. In der anschließenden Clustering-Phase wird nach der Berechnung der semantischen Ähnlichkeiten ein Clustering-Algorithmus auf den einfachen Abbildungen ausgeführt. Dadurch werden Gruppen gebildet, welche alle semantisch ähnlichen Elemente enthalten. Im Anschluss werden die Ergebnisse der ersten beiden Phasen zu einer finalen Menge von komplexen Abbildungen kombiniert. Es ist dennoch immer möglich, dass vorhandene Abbildungen nicht erkannt werden oder gar falsche Abbildungen identifiziert werden. Aufgrunddessen setzt dieser Ansatz auf eine hohe Interaktion mit dem Benutzer. Der Nutzer kann bei den Parametern varriieren und wird gelegentlich nach der Korrektheit identifizierter Abbildungen gefragt. Die durchgeführten Experimente zeigen die Effizienz des Ansatzes. Das automatische Abbilden von Feldern erreicht eine Präzision zwischen 81% und 93,5% mit einer Ausbeute zwischen 83,5% und 96,7% und einem F1-Maß von durchschnittlich 90% auf Formularen aus den Kategorien „Flüge“, „Autos“, „Bücher“, „Jobs“ und „Immobilien“.

### 3.2.2. Mehrstufiges Clustering

Das Werkzeug WISE-Integrator von Hai He et al. [HMYW03] bietet die Möglichkeit automatisch verschiedene Formularoberflächen zu einem globalen Formular zu verschmelzen. In der Vergangenheit wurde dieser aufwändige Integrationsprozess meist per Hand oder semi-automatisch durchgeführt. Der WISE-Integrator zeichnet sich vor allem durch eine hohe Abbildungsgenauigkeit und hohe Qualität im globalen Interface ohne Nutzerinteraktion aus. Dieser Ansatz wird nur kurz beschrieben, da hiermit nur einfache Abbildungen

identifiziert werden können. Das Werkzeug besteht aus zwei großen Bestandteilen, dem Extraktor und dem Integrator. Der Extraktor führt eine Vorverarbeitung der eingegebenen Formulare durch. Die Formularelemente werden vom Extraktor durch ein zweistufiges Clusteringverfahren in semantische Gruppen zerteilt. Diese Gruppen werden dann vom Integrator weiterverarbeitet. Der Integrator erstellt dann für jede Gruppe ein globales Objekt für das resultierende Formular. Die Güte des Ansatzes wird über die beiden Maße Genauigkeit und Vollständigkeit der Abbildungen bemessen. Das Werkzeug erzielt in den Experimenten eine durchschnittliche Genauigkeit von 95,25% mit einer Vollständigkeit von 97,91% im Mittel.

### 3.2.3. Correlation Mining

In der Arbeit „Discovering Complex Matchings across Web Query Interfaces: A Correlation Mining Approach“ gehen die Autoren He et al. [HCH04] auf einen ganz anderen Ansatz ein. Sie präsentieren Mining von Korrelationen als einen neuartigen Ansatz zur Zusammenführung von Schemata. Der Vorteil dieses Ansatzes gegenüber anderen ist, dass sowohl positive als auch negative Korrelationen betrachtet und ausgewertet werden. Viele bestehende Arbeiten beschränken sich hierbei lediglich auf starke positive Korrelationen. Formularelemente, die oft gemeinsam auftreten, wie zum Beispiel *Vorname* und *Nachname* korrelieren positiv. Synonyme hingegen, die normalerweise nicht im selben Formular zu finden sind, korrelieren negativ. Der Ablauf ist in die drei Teilprozesse Datenvorbereitung, duales Mining von positiven und negativen Korrelationen und Auswahl der Abbildungen einteilbar. Der Ansatz ermöglicht eine effiziente Art komplexe m:n-Abbildungen innerhalb einer Domäne zu identifizieren. Hierfür werden im Gegensatz zu verwandten Arbeiten nicht nur zwei Schemata, sondern alle vorhandenen Schemata gleichzeitig auf potenzielle Abbildungen untersucht.

#### Datenvorbereitung

Da der Miningalgorithmus nicht direkt auf HTML-Formularen ausgeführt werden kann, müssen die Formulare zunächst vorbereitet und in eine für den Algorithmus verständliche Datenstruktur überführt werden. Hierzu müssen die Formularelemente mitsamt ihrer Wertebereiche extrahiert werden. Dies kann durch eines der in Abschnitt 3.1 vorgestellten Verfahren erreicht werden. Die Autoren verwenden hierfür den Ansatz aus [ZHC04]. Im Anschluss daran werden die extrahierten Formularelemente noch normalisiert. Nach der Extraktion wird ein Typerkennungsalgorithmus ausgeführt, um die jeweiligen Datentypen der Wertebereiche zu bestimmen. Der Algorithmus unterscheidet zwischen den Datentypen *any, string, integer, float, month, day, date, time und datetime.* Anschließend werden die Elemente auf Ähnlichkeit ihrer Namen und Wertebereiche getestet. Syntaktisch ähnliche Formularelemente werden hierbei verschmolzen, wodurch sich die Auswertung der Korrelationen verbessert.

#### Mining der Korrelationen

Am Anfang des Miningprozesses werden zuerst positive Korrelationen zwischen Formularelementen gesucht. Das bedeutet, dass Gruppen von Formularelementen gesucht werden, die oft zusammen in einem Formular zu finden sind. Die gebildeten Gruppen werden dann den Schemata hinzugefügt, in denen sie auftreten. Wenn nur Teile der Gruppe im Schema vorkommen, wird dennoch die gesamte Gruppe hinzugefügt. Hierbei bleiben die Teile ebenfalls erhalten, da es sich bei den Gruppen nur um „potenzielle“ Abbildungspartner handelt. Im Anschluss daran werden negative Korrelationen zwischen den Gruppen der Schemata gesucht. Anschaulich werden hiermit semantisch gleiche Gruppen identifiziert. Hinter diesen ähnlichen Gruppen werden Synonyme vermutet, da diese normalerweise nie zusammen

in einem Formular zu finden sind. Gängige Korrelationsmaße haben jedoch das Problem, dass selten auftretende Elemente stark negativ korrelieren und häufig auftretende Elemente stark positiv korrelieren. Hierfür wurde ein eigenes Korrelationsmaß entwickelt, welches robust gegen diese Probleme ist. Das Ergebnis dieses Schrittes sind dann Kandidaten für mögliche komplexe Abbildungen.

#### Auswahl der Abbildungen

Zuletzt müssen noch die korrekten Abbildungen unter den Kandidaten gefunden werden. Um potenzielle Konflikte zwischen Abbildungen lösen zu können, muss eine Rangordnung entwickelt werden. Zur Quantifizierung wird eine Funktion eingeführt, welche die Güte der Abbildungen unter dem H-Maß bestimmt. Mithilfe dieser Funktion lassen sich folgende Regeln für die Rangordnung zweier Abbildungen aufstellen:

1. Der Rang einer Abbildung mit höherer Güte ist ebenfalls höher als der Rang einer Abbildung mit geringerer Güte.
2. Falls die Güte gleich ist und eine Abbildung Obermenge einer anderen Abbildung ist, so erhält die Obermenge einen höheren Rang.
3. Ansonsten ist die Rangordnung dieser beiden Abbildungen beliebig.

Die Auswahl der korrekten Abbildungen wird durch einen Greedy-Algorithmus getroffen. Die Abbildung mit dem höchsten Rang wird gewählt und dazu inkonsistente Abbildungen aus der Liste der Kandidaten gelöscht. Dieser Vorgang wird wiederholt, bis alle Abbildungen überprüft wurden. Die Abbildungen, die dann noch in der Liste der Kandidaten enthalten sind, werden als finale Abbildungen ausgegeben.

## 3.3. Zuordnungen von Ontologien

Da auch Ontologien als Eingabe für das in dieser Arbeit entwickelte Werkzeug zugelassen werden sollen, müssen zwei zusätzliche Fälle betrachtet werden. Auf der einen Seite muss es möglich sein, komplexe Abbildungen zwischen Formularen und Ontologien identifizieren zu können. Andererseits müssen komplexe Abbildungen ebenso zwischen Ontologien erkannt werden. Gegebenenfalls werden mehrere lokale Ontologien zu einer globalen Ontologie verschmolzen. Für diese beiden Fälle werden im Folgenden Lösungsansätze präsentiert. Hierbei ist festzustellen, dass Ontologien durch die Beziehungen ihrer Unterkonzepte bereits in einer hierarchischen Struktur vorliegen. Hierdurch können Ontologien einfach als Spezialisierung der hierarchischen Darstellung von Formularen betrachtet werden. Daraus folgt, dass die bereits vorgestellten Ansätze aus Abschnitt 3.2 ebenfalls für Ontologien verwendet werden können.

### 3.3.1. Abbildungen zwischen Formularen und Ontologien

Im Paper „Learning to Discover Complex Mappings from Web Forms to Ontologies“ [AHS12] wird ein auf maschinellem Lernen basierender Ansatz zur Abbildung von Formularen auf Ontologien vorgestellt. Zuerst wird das Formular in einen Formularbaum überführt. Hiernach werden dann Ähnlichkeiten zwischen dem Formularbaum und der Ontologie bestimmt, wodurch später die Abbildung gebildet werden kann. Mithilfe von zahlreichen Trainingsbeispielen und des naiven Bayes-Ansatzes [AHS12] führt der Ansatz durch maschinelles Lernen zu sehr guten Ergebnissen. In über 80% der Testfälle über hunderte Abbildungskandidaten wird die erwartete Abbildung als erste Wahl identifiziert.

### 3.3.2. Abbildungen zwischen Ontologien

Zur Identifikation von Abbildungen zwischen Ontologien existieren bereits viele unterschiedliche Ansätze. Einige der in Abschnitt 3.2 präsentierten Ansätze können ebenfalls für Ontologien erweitert werden. Die Arbeit „Automatic Ontology Merging by Hierarchical Clustering and Inference Mechanisms“ [MFBB10] stellt einen skalierbaren clusterbasierten Algorithmus vor, wodurch mehrere lokale Ontologien zu einer globalen verschmolzen werden können. Zuerst werden Ontologien verglichen und mithilfe eines Clusterverfahrens in semantisch gleiche Gruppen aufgeteilt. Hiernach werden diese Gruppen auf globale Entitäten abgebildet und daraus eine globale Ontologie erstellt.

# 4. EASIER

Das Rahmenwerk EASIER [BL16], welches die Basis für diese Arbeit bildet, automatisiert den aufwändigen Prozess der Generierung von aktiven Ontologien. Ein mögliches Anwendungsgebiet für AOs ist die Entwicklung von intelligenten Sprachassistenzsystemen, wie beispielsweise Apple's Siri. Hier werden sie verwendet, um natürliche Sprache zu verstehen. Wenn nun ein Assistenzsystem um eine Funktion erweitert werden soll, müssen neue Ontologien erstellt werden. Mithilfe von aktiven Ontologien wird natürliche Sprache verstanden und verarbeitet, sodass die Informationen anschließend an verschiedene Dienstgeber weitergeleitet werden können. Aufgrund des hohen manuellen Aufwandes, welcher zur Erstellung von aktiven Ontologien benötigt wird, soll dieser Prozess möglichst automatisch ablaufen. „Die Frage ist nun nicht mehr wie man intelligente Assistenzsysteme baut, sondern wie man sie effizient baut" [BL16]. EASIER hakt an dieser Stelle ein und schafft als Ausführungsumgebung für aktive Ontologien eine Möglichkeit AOs automatisch aus Webformularen zu generieren.

### Ansatz

Hierfür sucht zuerst ein Crawler nach formularbasierten Dienstgebern und extrahiert deren Formulare. Anschließend werden die extrahierten Webformulare mithilfe eines Clusteringverfahrens in Dienstkategorien eingeteilt. Aus jeder Dienstkategorie wird dann schließlich eine aktive Ontologie erstellt. Nachdem die Ontologien generiert wurden, können sie gemeinsam mit verschiedenen Dienstgebern bei EASIER registriert werden. Um die Dienstgeber später mit den korrekten Informationen ansprechen zu können, müssen Abbildungen zwischen den Formularelementen der Dienstgeberformulare und der Ontologie gespeichert werden.

### EASIER Active Server

Der EASIER Active Server dient als Ausführungsumgebung für aktive Ontologien. Dieser ist eine Erweiterung des Active Servers von Guzzoni [Guz08]. Abbildung 4.1 zeigt den Aufbau des EASIER Active Servers. Aktive Ontologien verwenden Fakten um miteinander zu kommunizieren. Diese Fakten werden im Faktenspeicher abgelegt. In der Auswertungseinheit werden dann die Fakten evaluiert und gegebenenfalls verändert. Eine aktive Ontologie, die eine bestimmte Aufgabe erfüllen soll, sammelt die dafür benötigten Informationen und gibt diese zur Ausführung an den Broker weiter. Es gibt zwei spezielle Ontologien, den Dialog-Manager und den Broker. Der Broker übernimmt die Kommunikation mit den Dienstgebern und stellt Anfragen an eine oder mehrere Dienste. Hierfür nutzt

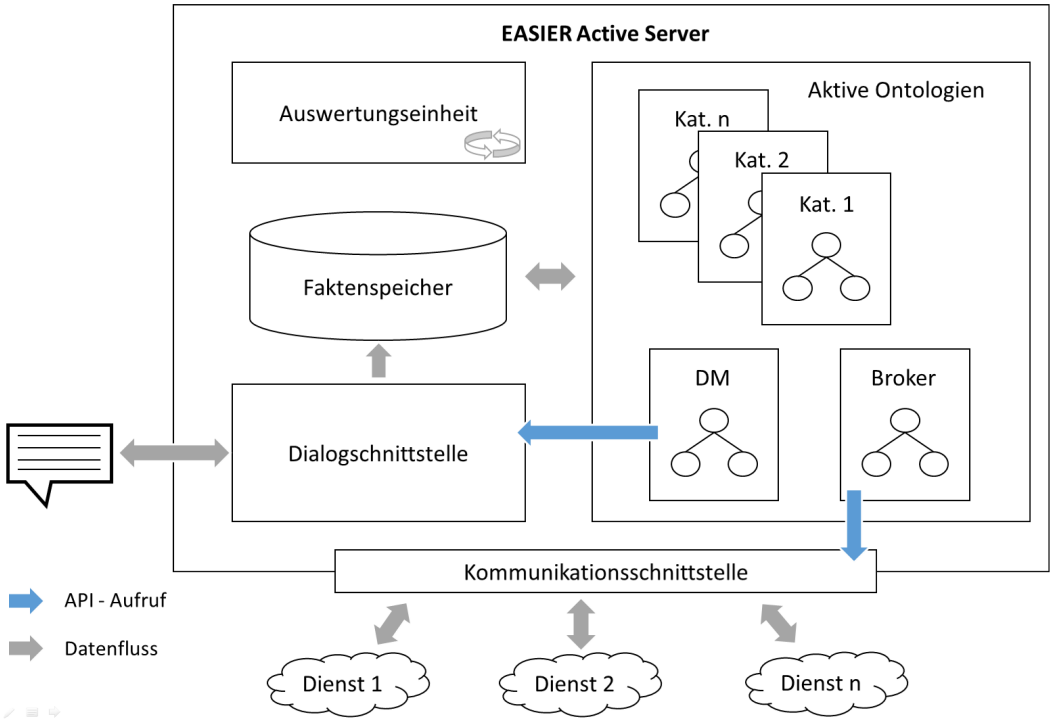



Abbildung 4.1.: Architektur des EASIER Active Servers [BL16].

er die Kommunikationsschnittstelle des EASIER Active Servers. Falls zu einem Zeitpunkt Nutzerinteraktion nötig ist, kommuniziert der Dialog Manager über die Dialogschnittstelle mit dem Benutzer und erfragt fehlende Informationen. Hierdurch müssen die Kategorieontologien keine Nutzerinteraktion durchführen. Falls eine weitere Information von einer dieser Ontologien benötigt wird, so wird ein Fakt im Faktenspeicher abgelegt. Dieser Fakt enthält die Information, dass eine verpflichtende oder optionale Information fehlt.

# 5. Analyse

Diese Arbeit beschäftigt sich mit komplexen Abbildungen von Formularelementen zur Generierung von aktiven Ontologien. Das Rahmenwerk EASIER soll so erweitert werden, dass durch die neu implementierte Identifikation komplexer Abbildungen semantische Zusammenhänge detaillierter abgebildet werden können, wodurch die Güte der generierten AOs verbessert werden soll. Es wird ein Werkzeug entwickelt, welches potenzielle komplexe Abbildungen zwischen Formularelementen findet und ausgibt. Auf Basis der durch das Werkzeug identifizierten Abbildungen werden außerdem für häufig auftretende Fälle konkrete Lösungen entworfen. Eine Lösung ist eine Abbildungsvorschrift von den gefundenen Elementen auf Konzepte einer AO in EASIER. Durch die Auflösung der komplexen Abbildungen können dann semantisch gleiche, aber unterschiedlich realisierte Informationen von diesen Konzepten erkannt werden. Dies soll eine bessere Informationserkennung der AOs ermöglichen. Das Werkzeug erhält als Eingabe eine Menge an Formularen, welche auf Abbildungen geprüft werden sollen. Abbildung 5.1 zeigt eine mögliche Beispieleingabe.

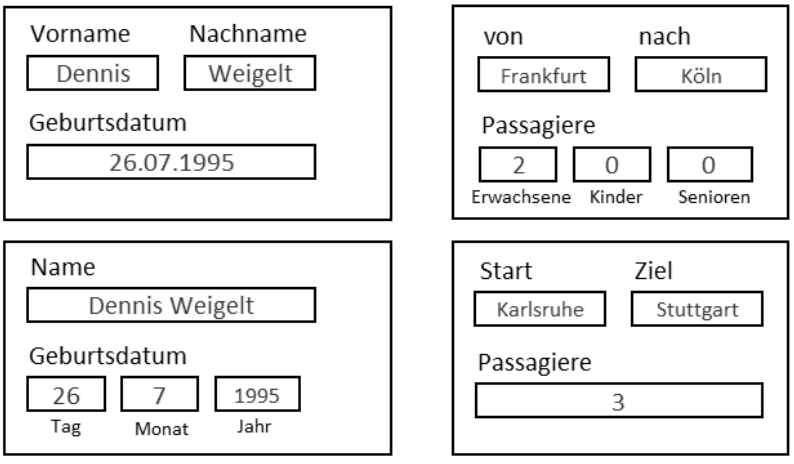



Abbildung 5.1.: Eine Menge an Beispielformularen, die als Eingabe für das zu entwickelnde Werkzeug dienen kann.

Da dies eine stark vereinfachte Darstellung der Formulare ist und diese in HTML weitaus mehr Informationen enthält, muss zunächst geprüft werden, welche Informationen in den Formularen zielführend für die Aufgabenstellung verwendet werden können. Da die Informationen aus HTML-Formularen so keine Aussage über komplexe Abbildungen ermöglichen, müssen die Formulare erst aufbereitet werden. Durch eine entsprechende Aufbereitung der Daten können dann wichtige von unwichtigen Informationen getrennt werden. Außerdem müssen die Daten, um später komplexe Abbildungen identifizieren zu können, untereinander vergleichbar sein. Da Formulare keine allgemeine Syntax zur Erstellung haben, müssen die Informationen zuerst vergleichbar gemacht werden. Hierdurch können die Daten einfacher auf semantische Ähnlichkeiten geprüft werden. Diese semantischen Ähnlichkeiten können dann verwendet werden, um komplexe Zusammenhänge zwischen Formularelementen zu finden. Hierfür muss zunächst festgestellt werden, inwiefern Formularelemente verglichen werden können, um zielführende Aussagen über ihre semantische Ähnlichkeit zueinander treffen zu können. Wenn die semantischen Ähnlichkeiten schließlich berechnet wurden, können diese anschließend dazu verwendet werden, komplexe Abbildungen zu finden. Wie solche Ähnlichkeiten gefunden werden können und hieraus komplexe Abbildungen gefunden werden können, wird in Abschnitt 5.2 genauer analysiert. Die identifizierten Abbildungen sollen anschließend so aufgelöst werden, dass ihre Elemente auf konkrete Konzepte von AOs abgebildet werden können. Wenn solche Lösungen konstruiert werden sollen, können erkannte Muster aus den zuvor identifizierten Abbildungen helfen. Sobald eine Abbildung ein Muster aufweist, können daraus Abbildungsvorschriften für bestehende Ontologien abgeleitet werden. In den folgenden Unterkapiteln werden Teilprobleme der Datenaufbereitung, Datenanalyse und Konstruktion von Lösungen vorgestellt, diskutiert und analysiert.

## 5.1. Datenaufbereitung

Um für die Aufgabenstellung wichtige Informationen betrachten zu können, müssen die Eingabeformulare zunächst aufbereitet werden. Die relevanten Informationen für die Identifikation komplexer Abbildungen müssen später umfassend auf semantische Ähnlichkeiten geprüft werden. Das heißt, dass die Eingabedaten so aufbereitet werden sollten, dass zunächst relevante Informationen von irrelevanten getrennt werden. Hierfür muss zuerst überlegt werden, welche Eingabedaten zur Identifikation komplexer Abbildungen beitragen könnten und wie diese gespeichert werden können. Anschließend können die Formulardaten mithilfe einer Normalisierung semantisch vergleichbar gemacht werden.

### 5.1.1. Extraktion aus HTML-Formularen

HTML-Formulare besitzen günstige Eigenschaften, die zum semantischen Verständnis beitragen können. Webformulare sind aufgrund der Charakteristiken der Sprache HTML hierarchisch aufgebaut. Diese hierarchische Struktur kann diverse semantische Bedeutungen enthalten. Dadurch, dass Formulare einfach benutzbar sein sollen, werden semantisch ähnliche Formularelemente oftmals im Formular nahe beieinander als semantische Gruppe dargestellt. Abbildung 5.2 zeigt die zwei semantischen Gruppen „Autor“ und „Erscheinungsdatum“. Man kann gut erkennen, dass semantische Gruppen im Formular auch strukturell gruppiert sind.

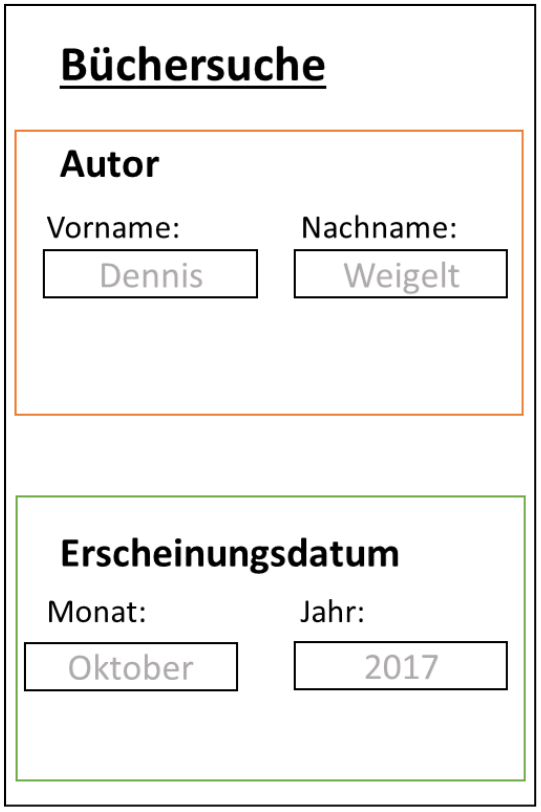



Abbildung 5.2.: Ein Webformular zur Suche nach Büchern in einem Onlinekatalog. Es können Bücher durch ihren Autor und ihr Erscheinungsdatum gesucht werden.

Zwar gestaltet jeder Webdienstanbieter seine Formulare individuell, jedoch wird zunehmend versucht, die Struktur nachvollziehbar und sinnvoll zu halten. Diese Tatsache kann genutzt werden, um später Ähnlichkeiten anhand der strukturellen Gruppen feststellen zu können und den Suchbereich für potenzielle semantische Gruppierungen in einem Formular auf Nachbarelemente zu beschränken. Während beispielsweise einzelne Formularelemente für sich kaum semantische Informationen beinhalten könnten, kann hier die Formulargruppe genutzt werden um diese Elemente in einen semantischen Vergleich mit einzubeziehen. Abbildung 5.3 zeigt die Eingabe des Autors aus Abbildung 5.2 und ein Textfeld zur Eingabe des Autors eines anderen Formulares.

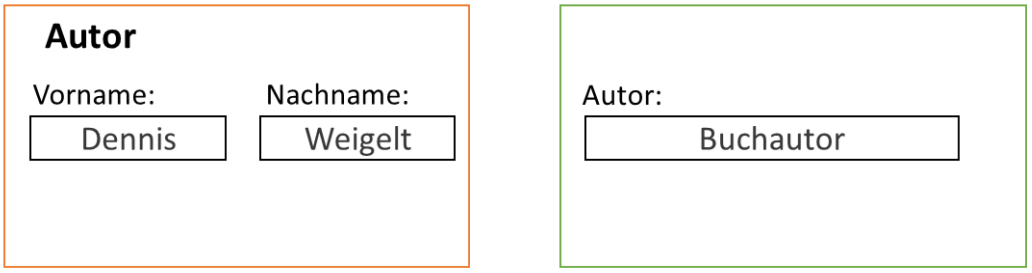



Abbildung 5.3.: Hier kann anhand der semantischen Gruppierung von „Vorname“ und „Nachname“ und dem Elternbezeichner „Autor“ dieser Gruppe eine Abbildung zum Eingabefeld „Autor“ im zweiten Formular identifiziert werden.

Die semantische Gruppe `Autor` besteht hierbei aus den Textfeldern `Vorname` und `Nachname`, wobei die beiden Textfelder im Formular nebeneinander angeordnet wurden. Während `Vorname` und `Nachname` für sich, bis auf den Wertebereich „Zeichenkette“, keinerlei semantische Ähnlichkeiten mit dem Textfeld `Autor` haben, so kann durch den Bezeichner `Autor` der semantischen Gruppe eine linguistische Ähnlichkeit zum Textfeld `Autor` gefunden werden. Zusammenfassend enthält die hierarchische Struktur, welche dem Formular durch die Eigenschaften von HTML verliehen werden, wichtige Informationen, die bei der Identifikation komplexer Abbildungen helfen können. Aus diesem Grund sollte die Struktur mit extrahiert werden und auch nach der Extraktion erhalten bleiben.

Jedoch kann nicht nur die Struktur von Webformularen zielführende Informationen liefern. Vor allem Wertebereiche und Attributwerte von Formularelemente bieten eine vielversprechende Anzahl an vergleichbaren Entitäten. Bei der Extraktion dieser ist es zunächst nötig, semantisch wichtige Informationen von unwichtigen zu unterscheiden. Hierfür muss zuerst festgestellt werden, welche Attribute zur Identifikation von komplexen Abbildungen beitragen können. Des Weiteren sind Wertebereiche nicht immer im Element angegeben. Daher sollte auch überlegt werden, ob und gegebenenfalls wie ein Wertebereich zu einem Formularelement aus seinen Attributen werden kann. Einige Formularelemente unterscheiden sich ausschließlich in ihren Attributwerten, nicht aber in ihrer Semantik. Im Folgenden werden die in Abschnitt 2.1.2 vorgestellten Formularelemente `label`, `input`, `textarea` und `select` auf relevante Informationen geprüft.

### Label

Das Element `<label>` dient als Bezeichnerelement für andere Formularelemente. Diese Bezeichner werden immer mit dem zu bezeichnenden Formularelement angezeigt. Da der Bezeichner als Verständnishilfe für den Benutzer dient, ist dieser meist aussagekräftig und für den Nutzer verständlich gehalten. Da das `<label>` durch zwei verschiedene Möglichkeiten im HTML-Formular verwendet werden kann, muss die Extraktion für beide Varianten eine Lösung bieten. Einerseits kann das Tag `<label>` das zugehörige Formularelement umschließen. Hierbei ist sofort klar, zu welchem Element der Bezeichner gehört. Andererseits kann das Label auch getrennt vom Formularelement stehen und durch die ID dem richtigen Formularelement zugeordnet werden. Bei diesem Fall muss beachtet werden, dass getrennt stehende Bezeichner-Formularelement-Paare bei der Extraktion zuerst zugeordnet und zusammengeführt werden sollten. Abbildung 5.4 zeigt die unterschiedlichen Verwendungsmöglichkeiten von `<label>`.

```
<label for="dep_city">Abflughafen</label>
<input type="text" id="dep_city"  name="departureCity"/>


<label> Geburtsdatum <input type="text" id="birthd" name="birthday"/>
</label>
```

Abbildung 5.4.: Zwei Möglichkeiten einem Formularelement einen Bezeichner zuzuweisen.

### Input

Durch das Element `<input>` können Benutzereingaben im Formular gefordert werden. Nach Durchsicht einiger Beispielformulare aus verschiedenen Dienstkategorien fällt auf, dass die Attribute `name`, `id`, `placeholder`, `value` und `type` von `<input>` semantische Informationen tragen, die bei der Identifikation komplexer Abbildungen eine Rolle spielen können. Da das `<label>` des Elementes `<input>` meist so gestaltet wird, dass hierdurch die Benutzereingabe vereinfacht wird, eignet es sich manchmal nicht für linguistische Vergleiche. Abbildung 5.5 zeigt ein solches Bezeichnerproblem.

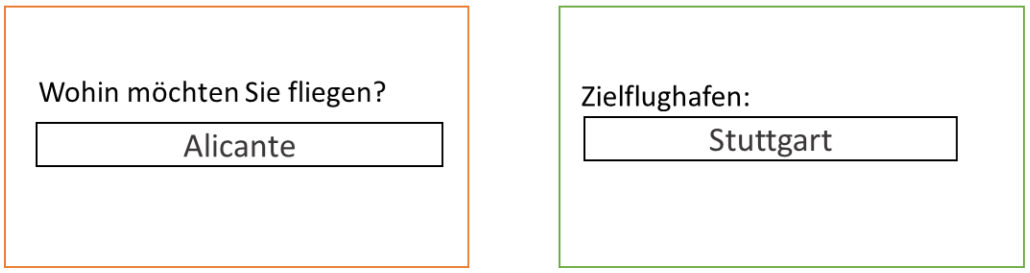



Abbildung 5.5.: Es kann keine linguistische Ähnlichkeit der Bezeichner gefunden werden.

In diesem Fall können die Attributwerte von `name` oder `id` zum Vergleich hinzugezogen werden. Wie genau solche linguistischen Vergleiche aussehen könnten, wird in Abschnitt 5.2.1 analysiert. Die Attribute `value` und `placeholder` haben eine ähnliche Funktion und spezifizieren einen vorher gesetzten Wert beziehungsweise eine Beispieleingabe. Diese Informationen sollten extrahiert werden, da diese Beispiele aus dem Wertebereich darstellen und sich dadurch möglicherweise der Wertebereich des Elementes eingrenzen lässt. Durch das Attribut `pattern` können Eingaben anhand eines regulären Ausdrucks eingeschränkt werden. Auch diese Information kann den Wertebereich eingrenzen. Das womöglich wichtigste Attribut für die Analyse von `<input>` ist `type`. Durch die Angabe von `type` im Formular kann die Art der Eingabe abgeleitet werden. Eine Auflistung möglicher Typen wurde bereits in Tabelle 2.3 aufgelistet. Hierbei lassen sich Gruppen von ähnlichen Typen erkennen, wodurch der Wertebereich eingegrenzt werden kann. Bei Auswahlmöglichkeiten wie den Typen `radio` und `checkbox` wird der Wertebereich durch die auswählbaren Optionen festgelegt. Zeiteingabetypen wie beispielsweise `date`, `time` oder `datetime` schränken den Wertebereich auf Datums- und Zeiteingaben ein. Außer diesen Typen gibt es noch Zahleneingabetypen und Texteingabetypen. Zahleneingabetypen schränken den Wertebereich auf eine Zahl oder einen Zahlenbereich ein. Der Typ `number` repräsentiert hierbei eine Zahl und der Typ `range` definiert einen erwarteten Zahlenbereich und stellt somit einen vollständigen Wertebereich dar. Bei Texteingabetypen können ebenfalls Wertebereiche abgeleitet werden. Durch Typen wie `tel`, `url` oder `email` können Wertebereiche

anhand von regulären Ausdrücken abgeleitet werden, da Telefonnummern, Webadressen und Emailadressen bestimmten Mustern folgen. Es gibt jedoch auch Typen wie `text` und `search`, aus denen kein Wertebereich abgeleitet werden kann.

#### Textarea

Das Element `<textarea>` enthält zusätzlich zum zugeordneten `<label>` und dem Attribut `name` meist keine sinnvollen semantischen Informationen, da semantisch wertvolle Attribute wie beispielsweise `pattern` oder `placeholder` bei einer mehrzeiligen Texteingabe zwar verwendet werden können, jedoch in der Praxis eher unüblich sind. Die Attribute `rows` und `cols` haben für die Identifikation komplexer Abbildungen keinen semantischen Wert.

#### Select

Das Element `<select>` hingegen beinhaltet einige semantisch wertvolle Informationen. Durch `<select>` wird eine Auswahlliste dargestellt, wodurch der Benutzer aus einer vorgegebenen Menge eine oder mehrere Optionen auswählen kann. Die Optionen mit dem Tag `<option>` können hierbei als Wertebereich des Elementes extrahiert werden. Dieser Wertebereich kann später zur Ermittlung des Datentyps oder zu einer Wertebereichsanalyse beitragen. Außerdem kann auch einem `<select>`-Element mit dem Attribut `value` ein Wert zugewiesen werden. Dieser Wert kann ebenfalls als Repräsentant des Wertebereiches angesehen werden und sollte auch extrahiert werden. Das Attribut `size` legt die Anzahl gleichzeitig angezeigter Elemente in der Liste fest, hat aber im Hinblick auf Abbildungen keinerlei semantische Bedeutung. Durch das Attribut `multiple` wird festgelegt, dass eine Mehrfachauswahl möglich ist. Diese Information kann extrahiert werden, da sie die möglichen Abbildungspartner einschränkt. Ein `<select>` kann beispielsweise auf Radio Buttons oder Checkboxes abgebildet werden. Durch `multiple` wird dies eingeschränkt, da Radio Buttons keine Mehrfachauswahl zulassen. Mit dem Attribut `selected` kann eine Option vorselektiert werden. Diese Information ändert jedoch weder den Wertebereich noch die möglichen Abbildungspartner und wird daher verworfen.

### 5.1.2. Normalisierung

Um semantische Ähnlichkeiten zwischen den unterschiedlichen Arten von Formularelementen finden zu können, müssen diese untereinander vergleichbar sein. Linguistische Vergleiche von Zeichenketten, wie Werte eines Wertebereiches oder Bezeichnern, sind besonders schwierig. Oftmals enthalten vor allem Bezeichnerelemente lange Zeichenketten, die nur schwer mit anderen Elementen verglichen werden können. In Abbildung 5.5 zeigt sich das Problem für den Bezeichner „Wohin möchten Sie fliegen?“. Dieser ist eine sehr lange Zeichenkette, die in dieser Form nur schwer mit anderen Attributwerten auf linguistische Ähnlichkeit geprüft werden kann. Wichtig ist hierbei, solche langen Zeichenketten so zu normalisieren, dass sie dennoch mit anderen Elementen verglichen werden können. Hierzu können Normalisierungen aus den Ansätzen [WYDM04] und [HMYW03] wiederverwendet werden. Zunächst können die Zeichenketten tokenisiert werden. Eine Tokenisierung bewirkt, dass zusammengesetzte Zeichenketten anhand von Großbuchstaben, Zahlen und Trennzeichen in einzelne Wörter (Tokens) zerteilt werden. Hierdurch könnten dann zwei Zeichenketten durch den Vergleich ihrer Tokens verglichen werden. Um diese Tokens jedoch miteinander vergleichen zu können, müssen diese ebenfalls untereinander vergleichbar sein. Hierbei könnte der Einfluss von Groß- und Kleinschreibung ein Problem für effiziente Vergleiche darstellen. Zwei Token, die sich lediglich in ihrer Schreibweise der Buchstaben unterscheiden, sind dennoch semantisch ähnlich. Auch sogenannte Stoppwörter wie beispielsweise „und“ oder „mit“ verfälschen das Ergebnis der Tokenvergleiche, da solche Stoppwörter oft verwendet werden, jedoch zur Semantik oftmals keinen großen Beitrag

leisten. So wären unter Umständen zwei völlig unterschiedliche Zeichenketten ähnlich, nur weil ähnliche Stoppwörter verwendet wurden. Andererseits können Stoppwörter durchaus Semantik tragen. Daher muss gut überlegt werden, ob es Sinn macht Stoppwörter zu entfernen. Falls Stoppwörter entfernt werden, sollte die Menge dieser zu entfernenden Wörter sehr sorgfältig ausgewählt sein, sodass dabei keine Semantik dabei verloren geht. Abbildung 5.6 zeigt ein Beispiel, in dem die Stoppwörter „von“ und „nach“ einen hohen semantischen Nutzen tragen und nicht entfernt werden sollten.

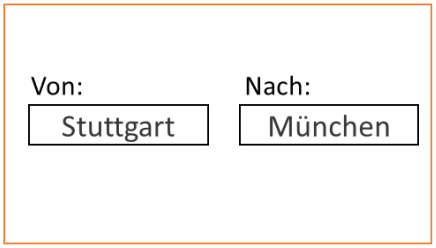



Abbildung 5.6.: Das Beispiel zeigt zwei Eingabefelder für Start- und Zielorte. Die Stoppwörter „von“ und „nach“ haben hier einen hohen semantischen Wert.

Um effiziente Tokenvergleiche durchführen zu können, dürfen also bestimmte Stoppwörter und die Schreibweise der Buchstaben keinen Einfluss auf die Vergleiche haben.

## 5.2. Datenanalyse

Sind nun die relevanten Daten extrahiert und vergleichbar, muss überlegt werden, wie semantische Ähnlichkeiten gefunden und quantifiziert werden können. Es muss also zunächst der Begriff „Ähnlichkeit“ im Kontext von Formularelementen definiert werden. Hierfür muss festgelegt werden, wann Formularelemente ähnlich zueinander sind und woran diese Ähnlichkeit gemessen werden kann. Hierfür muss zunächst herausgefunden werden, welche Eigenschaften von Formularelementen verglichen werden können und welche dieser Vergleiche zielführende Ergebnisse zur Identifikation komplexer Abbildungen liefern. Mithilfe dieser Ähnlichkeiten kann später ein Identifikationsverfahren zum Finden von komplexen Abbildungen entwickelt werden.

### 5.2.1. Semantische Analyse

Um Ähnlichkeiten zwischen Formularelementen zu finden und zu quantifizieren, werden die Daten zunächst semantisch analysiert. Hierfür wird geprüft, wie Formularelemente aufgebaut sind und anhand welcher Kriterien und Eigenschaften sie verglichen werden können.

```
<label for="bd">Geburtsdatum</label>
<input type="text" id="bd" name="birthday" placeholder="DD.MM.YYYY" value="01.01.1999"/>
```

Abbildung 5.7.: Ein Texteingabefeld für das Geburtsdatum. Hier werden ein Bezeichner und die Attribute `name`, `id`, `placeholder` und `value` verwendet.

Ein Formularelement hat normalerweise einen zugewiesenen Bezeichner und einige Attribute wie beispielsweise `name`, `id`, `placeholder` oder `value`, wie in Abbildung 5.7 gezeigt.

Bei zwei Formularelementen, die dieselben semantischen Informationen tragen, ist es daher sehr wahrscheinlich, dass sie sich auch in ihren Bezeichnern und Attributwerten ähneln. Um also Ähnlichkeit auf Basis von Bezeichnern und Attributwerten festzustellen, können diese untereinander auf einer linguistischen Ebene verglichen werden. Wie in Abschnitt 5.1.1 analysiert, können auch Wertebereiche von Formularelementen ganz bestimmt oder teilweise eingegrenzt werden. Diese Wertebereiche können hierbei entweder endliche Mengen (z.B. `<select>`), unendliche Mengen (z.B. `<input type=text>`) oder Zahlenbereiche (z.B. `<input type=range>`) sein. Manchmal ist es sogar möglich, Wertebereiche durch die gegebenen Informationen des Formularelementes zu typisieren. Eine Bestimmung der Typen kann allerdings beliebig komplex sein, da nicht immer alle benötigten Informationen dafür im vorliegenden Formular gegeben sind. Hierdurch ist es sinnvoll, zuerst den Typ des Wertebereiches zu bestimmen, da Wertebereiche verschiedenen Typs nicht ähnlich sind, denn semantisch ähnliche Formularelemente haben auch ähnliche Wertebereiche [WYDM04]. Außer diesen elementbezogenen Eigenschaften gibt es noch die Positionierung des Formularelementes im Formular. Anhand der Nachbarn und zugehörigen semantischen Gruppen eines Formularelementes können strukturelle Ähnlichkeiten zwischen Formularelementen identifiziert werden, denn oftmals wird dieselbe Information in eine ähnliche Struktur eingebettet. Dies folgt aus der Annahme, dass semantische Gruppen im Formular auch strukturell gruppiert sind [WYDM04]. Ähnliche Elemente weisen hierdurch in verschiedenen Formularen eine ähnliche Struktur der Nachbarelemente und Positionierung auf. Ähnlichkeit zwischen Formularelementen kann also anhand von drei Teilähnlichkeiten bestimmt werden - der linguistischen Ähnlichkeit, der strukturellen Ähnlichkeit und der Ähnlichkeit der Wertebereiche und Typen [WYDM04][HMYW03]. Diese drei Teilähnlichkeiten sollen dazu verwendet werden, Ähnlichkeiten zwischen Formularelementen zu quantifizieren.

### Linguistische Analyse

Bei der linguistischen Analyse werden die extrahierten Daten linguistisch verglichen. Hierfür sollten zur Verbesserung der Genauigkeit so viele Informationen wie möglich verwendet werden. Als mögliche Vergleichsoperanden können `label`, `name`, `placeholder`, `id` und `value` verwendet werden. Es ist sinnvoll, alle Operanden miteinander zu vergleichen, da Ähnlichkeiten auch unter verschiedenen Operanden gefunden werden können, denn oftmals enthalten unterschiedliche Operanden dieselben semantischen Informationen wie beispielsweise `value` und `placeholder`. Insbesondere der Vergleich von `label` und `name` kann hohe Ähnlichkeiten aufzeigen. Vor allem, wenn einem der Formularelemente kein `label` zugewiesen ist, hilft der Vergleich zwischen `name` und `label`. Abbildung 5.8 zeigt ein solches Beispiel, in welchem der Vergleich von unterschiedlichen Attributwerten und Bezeichnern sinnvoll ist.

```
<label for="bd">Birthday</label>
<input type="text" id="bd"  name="birthday" placeholder="birth date"/>

<input type="text" name="input_birthday" placeholder="MM/DD/YYYY">
```

Abbildung 5.8.: Ein Texteingabefeld für das Geburtsdatum. Hier werden ein Bezeichner und die Attribute `name`, `id` und `placeholder` verwendet.

Um die Elemente miteinander vergleichen zu können, müssen Zeichenketten verglichen werden. Hierfür gibt es verschiedene Möglichkeiten. Die Token könnten mithilfe eines Verfahrens zur Reduktion auf den Wortstamm in eine Stammform gebracht werden. Anschließend könnte man die Wortstämme miteinander vergleichen. Eine andere Möglichkeit könnte sein, die Tokenmengen anhand von Vektoren zu repräsentieren und diese zu

vergleichen [WYDM04]. Hierfür würde zunächst die Vereinigungsmenge über beide Tokenmengen gebildet werden. Anschließend können die Tokenmengen als Vektoren über dieser Vereinigungsmenge repräsentiert werden. Eine weitere Vorgehensweise könnte sein, die Ähnlichkeit zweier Zeichenketten durch die längste gemeinsame Teilzeichenkette zu definieren.

### Strukturelle Analyse

Da die Struktur der Formulare durch die Extraktion erhalten bleiben soll, bietet sich die Möglichkeit, diese Struktur ebenfalls zur Bestimmung von Ähnlichkeiten einzubeziehen [WYDM04]. Formularelemente haben eine Menge an vorhergehenden Nachbarn und eine Menge an nachfolgenden Nachbarn. Ähnliche Elemente sind oftmals in ähnliche Nachbarstrukturen eingebettet. Um nun strukturelle Aussagen über die Ähnlichkeit zweier Formularelemente treffen zu können, können sie anhand ihrer Nachbarn verglichen werden. Hierzu können die Nachbarn paarweise nach ihrer Entfernung linguistisch auf Ähnlichkeit geprüft werden. Dieses Verfahren kann bis zu einer beliebigen Entfernung vom Zielelement weitergeführt werden, jedoch ist zu beachten, dass mit steigender Entfernung der Nachbarn die Ähnlichkeiten immer unwichtiger werden, da diese dann mit wachsender Wahrscheinlichkeit nicht mehr zur gleichen semantischen Gruppe gehören. Für die Nachfolger kann dies analog berechnet werden. Abbildung 5.9 zeigt beispielhaft den Vergleich der ersten Nachbarn der strukturell zu analysierenden Elemente. Analysiert wird die strukturelle Ähnlichkeit der Formularelemente „Autor“ und „Schriftsteller“. Hierfür werden zunächst die nächsten Nachbarn analysiert. Die direkten Vorgänger „ISBN“ und „ISBN-Nummer“, sowie die direkten Nachfolger „Preis“ und „Preis in Euro“ werden auf linguistische Ähnlichkeiten geprüft. Im Anschluss daran werden dann die Vorgänger und die Nachfolger mit der Entfernung 2 miteinander verglichen. Die resultierenden linguistischen Ähnlichkeiten aus diesen Vergleichen fließen mit wachsender Entfernung mit geringerer Gewichtung in die Berechnung der strukturellen Ähnlichkeit ein.

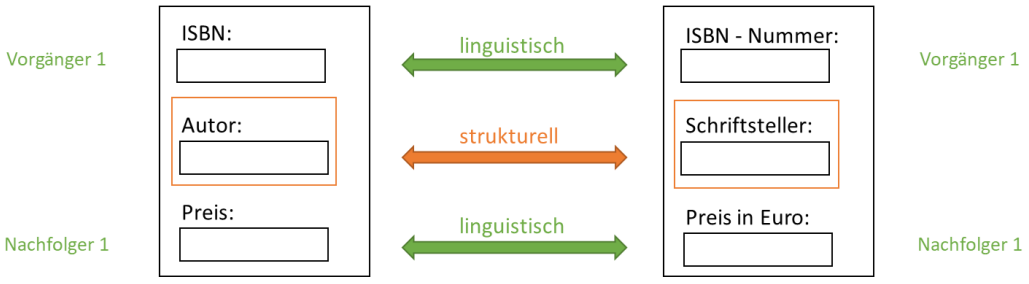



Abbildung 5.9.: Hier wird die Analyse der ersten Nachbarn für die strukturelle Analyse der beiden Felder „Autor“ und „Schriftsteller“ dargestellt.

### Analyse der Wertebereiche

Formularelemente besitzen jedoch nicht nur linguistische und strukturelle Merkmale, die zur Berechnung der Ähnlichkeit verwendet werden können. Ähnliche Formularelemente besitzen auch ähnliche Wertebereiche [WYDM04]. Auch der Wertebereich eines Formularelementes kann Ähnlichkeiten aufzeigen. Hierzu muss zuerst der Wertebereich der Formularelemente bekannt sein. Dies ist nicht immer möglich, da oftmals die hierzu nötigen Informationen im Formular fehlen. Jedoch können durch die Verwendung einiger Attribute

zumindest Eingrenzungen der Wertebereiche vorgenommen werden (vgl. Abschnitt 5.1.1) oder teilweise sogar ganze Wertebereiche identifiziert werden. Relevante Attribute zur Identifikation von Wertebereichen sind `min`, `max`, `pattern`, `placeholder`, `type` und `value`. Die Funktionen dieser Attribute sind in Tabelle 2.2 beschrieben. Durch `min` und `max` können numerische Wertebereiche eingegrenzt werden. Durch `value` wird ein Beispielwert aus dem Wertebereich gegeben. `Placeholder` liefert ebenfalls einen Hinweis darauf, welche Information hier erwartet wird. Dieser Hinweis könnte beispielsweise ein Bezeichner oder eine Beispieleingabe sein. Mit `pattern` wird die Eingabe auf einen regulären Ausdruck geprüft. Dieser kann Hinweise auf den Wertebereich enthalten. Das Attribut `type` von `<input>` schränkt den Wertebereich auf den angegebenen Typ ein. Eine Übersicht der möglichen Typen kann Tabelle 2.3 entnommen werden.

Es gibt endliche und unendliche Wertemengen und Zahlenbereiche. Endliche Wertebereiche wie bei `<select>` oder `<input type=radio>` können recht einfach miteinander verglichen werden. Da endliche Wertebereiche Listen von Zeichenketten sind, können diese linguistisch untersucht werden. Hierbei können die Zeichenketten paarweise auf linguistische Ähnlichkeiten geprüft werden. Mithilfe dieser Berechnungen kann dann die Ähnlichkeit der Listen bestimmt werden. Auch Zahlenbereiche sind endliche Wertebereiche. Hier kann allerdings die Eigenschaft genutzt werden, dass jeder dieser Zahlenbereiche eine Teilmenge der Ganzzahlen ist. Hierdurch kann die Ähnlichkeit zweier Zahlenbereiche durch die Größe des gemeinsamen Intervalls beider Wertebereiche quantifiziert werden. Auch der Typ der Wertebereiche kann bei der Ermittlung der Ähnlichkeit helfen. Bei Zahlen- und Zahlenbereichen ist der Typ sehr einfach zu bestimmen. Bei endlichen Wertebereichen wie Listen von Zeichenketten kann auch versucht werden, den Wertebereich zu typisieren. Anschließend können, falls typisierbar, die Typen der Wertebereiche miteinander verglichen werden. Falls die Typen kompatibel sind, so können die Wertebereiche ähnlich sein. Falls nicht, so kann sofort eine Ähnlichkeit der Wertebereiche ausgeschlossen werden. Ein nicht lösbares Problem ist die Berechnung der Wertebereichsähnlichkeit, sobald eine unendliche Wertebereichsmenge wie bei `<input type=text>` hinzukommt. Unendliche Wertebereichsmengen können nicht typisiert werden, da keinerlei Informationen über die möglichen Werte im Wertebereich vorliegen. Daher kann keine Wertebereichsähnlichkeit bestimmt werden, falls eines der zu vergleichenden Formularelemente einen unendlichen Wertebereich besitzt. Aufgrunddessen ist ein unendlicher Wertebereich zu keinem anderen Wertebereich ähnlich.

### 5.2.2. Identifikation komplexer Abbildungen

Nach der semantischen Analyse muss nun ein Identifikationsprozess entwickelt werden, indem diese Ähnlichkeiten verwendet werden können, um potenzielle komplexe Abbildungen identifizieren zu können. Hierzu gibt es in der Literatur einige Vorgehensweisen. Die meisten Forschungsarbeiten über Abbildungen verwenden einen Clustering-Ansatz [WYDM04] [HMYW03], um semantisch ähnliche Elemente in Cluster einzuteilen. Diese Cluster repräsentieren dann Abbildungen zwischen ihren Elementen. Dieser Ansatz ist vor allem bei der Identifikation einfacher Abbildungen sehr effektiv. Bei der Identifikation komplexer Abbildungen hingegen muss mit der Herausforderung gekämpft werden, dass das Finden komplexer Abbildungen deutlich schwieriger ist als die Identifikation einfacher Abbildungen wie in [HMYW03]. Es müssen viel mehr Faktoren in Betracht gezogen werden als bei einer einfachen Analyse. Ein komplexes Clusteringverfahren müsste nicht nur erkennen können, dass zwei Elemente semantisch ähnlich sind, sondern dass die Informationen mehrerer Elemente vereint semantisch ähnlich zu einem anderen Element sind. Da die Komplexität der Identifikation einfacher Abbildungen deutlich geringer ist als die der Identifikation komplexer Abbildungen, kann überprüft werden, ob einfache Abbildungen unterstützend beim Identifikationsprozess für komplexe Abbildungen mitwirken können.

Sowohl einfache als auch komplexe Abbildungen sind transitiv [WYDM04]. Das bedeutet, dass durch das transitive Zusammenführen bereits identifizierter Abbildungen neue Abbildungen generiert werden können. Es ist somit sinnvoll, alle Formulare gleichzeitig anstatt paarweise zu betrachten und zu vergleichen, da hierdurch die Transitivität von Abbildungen ausgenutzt werden kann. Im folgenden Beispiel wird erläutert, inwiefern die Transitivität von Abbildungen dabei helfen kann, neue Abbildungen zu generieren.

**Transitivität zur Generierung neuer Abbildungen**

Oftmals gibt es Abbildungen, die nicht anhand von direkten Vergleichen gefunden werden können, da die Ähnlichkeit der Attributwerte hierfür nicht ausreicht. Um diese Abbildungen doch identifizieren zu können, hilft die Transitivitätseigenschaft der Abbildungen.

**Transitivität bei einfachen Abbildungen**

Sei $a$ ein Formularelement aus Formular $A$, $b$ ein Formularelement aus Formular $B$ und $c$ ein Formularelement aus Formular $C$. Der Identifikationsalgorithmus findet folgende einfache Abbildungen:

- $a = b$
- $a = c$

Durch die Transitivität von Abbildungen kann dann noch eine weitere Abbildung, $b = c$, identifiziert werden, welche durch reine semantische Vergleiche zwischen $b$ und $c$ nicht gefunden wurde.

**Transitivität bei komplexen Abbildungen**

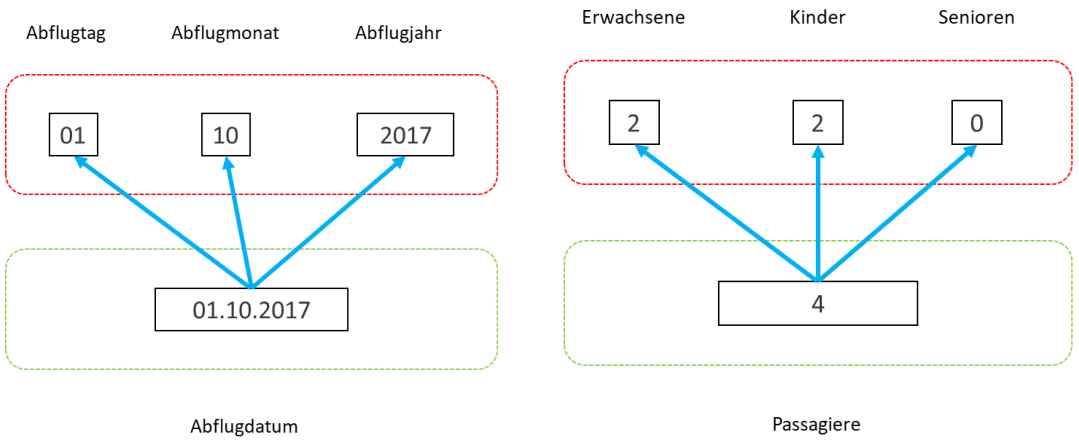
Bei komplexen Abbildungen kann ebenfalls die Transitivität zur Generierung neuer Abbildungen verwendet werden. Sei $a$ ein Formularelement aus Formular $A$, $b_1, b_2$ Formularelemente aus Formular $B$ und $c_1, c_2$ Formularelemente aus Formular $C$. Es werden folgende einfache und komplexe Abbildungen identifiziert:

- $a = \{b_1, b_2\}$
- $b_1 = c_1$
- $b_2 = c_2$

Durch die Transitivität kann dann die komplexe Abbildung $a = \{c_1, c_2\}$ identifiziert werden, obwohl die semantische Ähnlichkeit zwischen $a$ und $c_1, c_2$ selbst nicht ausgereicht hätte.

Anhand dieses Beispiels ist zu erkennen, dass es sinnvoll für die Identifikation komplexer Abbildungen sein kann, auch einfache Abbildungen zu betrachten. Durch diese einfachen Abbildungen können anschließend mithilfe der Transitivität neue Abbildungen generiert werden, die durch semantische Vergleiche der Elemente nicht gebildet werden konnten, da die komplexe Ähnlichkeit hierfür nicht genügt. Doch um die Transitivitätseigenschaft nutzen zu können, muss zunächst eine initiale Menge an komplexen Abbildungen anhand ihrer semantischen Ähnlichkeiten identifiziert werden können. Das bedeutet, dass $m$ Formularelemente in einem Formular gefunden werden müssen, die gemeinsam dieselbe semantische Information wie das abzubildende Element repräsentieren. Weiterhin muss überprüft werden, welche Arten von komplexen Abbildungen es überhaupt gibt und wie diese Abbildun-

gen gefunden werden können. In der Literatur werden zwei Arten von komplexen Abbildungen unterschieden, ist-ein-Abbildungen und aggregierte Abbildungen [WYDM04]. Die $m$ Elemente einer aggregierten Abbildung sind jeweils ein Teil des abzubildenden Elementes und setzen sich zum abzubildenden Element zusammen. Bei einer ist-ein-Abbildung ist jedes der $m$ Elemente eine Teilmenge des abzubildenden Elementes. Abbildung 5.10 veranschaulicht die beiden Abbildungsarten in einem Beispiel.




(a) aggregierte Abbildung

(b) ist-ein-Abbildung

Abbildung 5.10.: Abbildungsarten komplexer Abbildungen [WYDM04]

Fraglich ist nun, wie solche komplexe Abbildungskandidaten gefunden werden können. Der clusteringbasierte Ansatz zur Identifikation komplexer Abbildungen nach [WYDM04] liefert eine Möglichkeit durch semantische Ähnlichkeiten mit Einbeziehung der Transitivität. Im ersten Schritt wird eine initiale Menge an komplexen Abbildungen identifiziert. Anschließend werden einfache Abbildungen mithilfe eines Clusterverfahrens gefunden. Im letzten Schritt werden komplexe und einfache Abbildungen vereint und durch Transitivität neue Abbildungen generiert. Die Herausforderung besteht darin, eine möglichst große initiale Menge korrekter komplexer Abbildungen zu finden. Hierfür wird für jedes Formularelement $e$ überprüft, ob es eine Menge $f = \{f_1, f_2, ..., f_n\}$ an Formularelementen in einem anderen Formular gibt, sodass gilt:

1. Alle $f_i$ der semantischen Gruppe $f$ haben denselben Elternknoten $p$.
2. Der Bezeichner des Elternknotens $p$ und der Bezeichner des Formularelementes $e$ sind linguistisch ähnlich.
3. Die Wertebereiche der $f_i$ sind entweder ähnlich zum Wertebereich von $e$ oder sie sind Unterwertebereiche des Wertebereiches von $e$.

Hierbei ist anzumerken, dass Bedingung (1) die strukturelle Komponente betrachtet. Formularelemente können nur dann Teil einer komplexen Abbildung sein, wenn sie Geschwisterknoten im Formularbaum sind und somit im ursprünglichen Formular eine semantische und strukturelle Gruppe bilden. Dieses Verfahren kann sehr gut auf manuell extrahierte Formularbäume angewendet werden, da hier die Eltern-Kind-Beziehungen zwischen den Knoten sauber extrahiert werden können und somit Bedingung (2) geprüft werden kann. Bei einer automatischen Extraktion, wie sie in dieser Arbeit implementiert wird, sind diese Bedingungen jedoch nur teilweise oder sogar überhaupt nicht prüfbar. Die Datenvorbereitung in [WYDM04] basiert auf einer manuellen Extraktion der Formulare in hierarchische Formularbäume. Das Hauptaugenmerk des Rahmenwerkes EASIER liegt auf

einem möglichst maximalen Automatisierungsgrad. Bei der automatischen Extraktion von Formularelementen aus Formularen können semantische Zusammenhänge, die durch einen manuellen Extraktor aufgelöst werden können, zum Teil nicht extrahiert werden, da hierfür die gegebenen Informationen im Formular nicht genügen. Ein menschlicher Extraktor kann anhand von Zusammenhängen gewisse semantische Informationen herleiten, die einem automatischen Extraktor nicht zugänglich sind. Vor allem die automatische Zuordnung von Eltern-Kind-Beziehungen von Bezeichnern ist kaum möglich. Oftmals sind Bezeichner nicht durch das `<label>`-Element einem Eingabefeld zugeordnet, sondern stehen als Text in der Nähe des zu bezeichnenden Elementes im Formular. Hierbei kann nicht allgemeingültig bestimmt werden, ob ein Bezeichner das Eingabefeld rechts von ihm oder das Eingabefeld unter ihm bezeichnen soll. In diesem Fall kann vor allem bei Elternbezeichnern nicht identifiziert werden, welche der folgenden Elemente noch zu diesem Elternbezeichner gehören. Der Ansatz aus [WYDM04] kann für die Identifikation der Abbildungskandidaten also nicht verwendet werden, da dieser Wissen über Eltern-Kind-Beziehungen der Formularelemente benötigt. Aus diesem Grund muss im ersten Schritt zur Findung vorläufiger Abbildungen ein anderer Ansatz zur Identifikation dieser verwendet werden. Es muss also ein Verfahren entworfen werden, wodurch komplexe Abbildungen auch ohne Einbeziehung dieser nicht zugänglichen Informationen identifiziert werden können.

### Identifikation semantischer Gruppen

Wenn Eltern-Kind-Beziehungen nicht verwendet werden können, muss zunächst herausgefunden werden, wie semantische Gruppen innerhalb eines Formulares ohne diese Information bestimmt werden können. Semantische Gruppen können nicht anhand von semantischen Ähnlichkeiten gefunden werden, da die Formularelemente einer semantischen Gruppe meist unterschiedliche Informationen tragen, die dann vereinigt eine gemeinsame Information repräsentieren. Ein Beispiel hierfür wäre die Identifikation einer komplexen Datumsabbildung. Hierfür muss zunächst die semantische Gruppe $\{Tag, Monat, Jahr\}$ identifiziert werden. Diese drei Eingabefelder besitzen jedoch keinerlei semantischen Ähnlichkeiten untereinander. Nach Betrachtung einiger Beispielformulare fällt auf, dass für Formularelemente derselben semantischen Gruppe in diesen Formularen oftmals ein gemeinsames Namenspräfix oder -suffix verwendet wird. Diese Information kann dazu verwendet werden, Gruppen von Formularelementen im Formular zu erstellen, indem alle Formulare mit demselben Namenspräfix oder -suffix in eine Gruppe eingeteilt werden. Abbildung 5.11 zeigt ein Beispiel für die Gruppierung von fünf Formularelementen durch die Verwendung der Prä- und Suffixe.

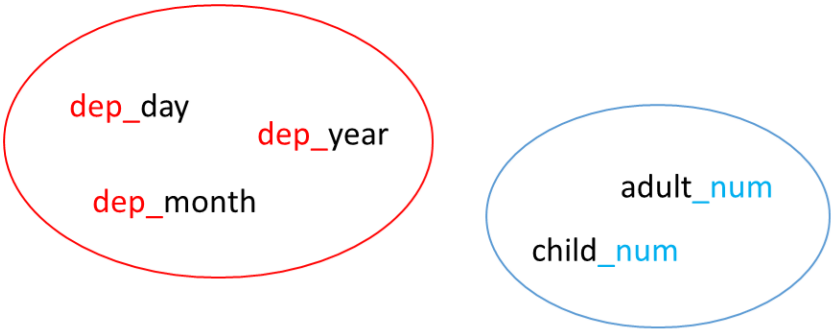



Abbildung 5.11.: Generierung von zwei semantischen Gruppen durch die Verwendung der Prä- und Suffixe „dep_“ und „_num“.

Hierbei müssen Präfixe wie „in“ oder „input“, die häufig als Präfix für Eingabefelder verwendet werden, zuerst entfernt werden. Es besitzen zwar viele Formulare eine solche Präfix-/Suffixstruktur, jedoch muss für alle anderen Formulare eine weitere Möglichkeit gefunden werden, semantische Gruppen zu identifizieren. In der Arbeit von [HCH04] wird ein Ansatz zum Finden von positiv korrelierten Elementen innerhalb eines Formulares mittels „Correlation Mining“ beschrieben. Hierbei werden zunächst für jedes Elementpaar aller Formulare Korrelationstabellen erstellt, die speichern, wie oft die beiden Elemente gemeinsam, jeweils einzeln und nie miteinander in einem Formular auftauchen. Mithilfe von Korrelationsmaßen können dann positiv korrelierte Gruppen gebildet werden. Diese Elementgruppen haben eine hohe Wahrscheinlichkeit eine semantische Gruppe zu sein, da sich ihre Elemente offenbar semantisch bedingen, wenn sie überwiegend gemeinsam in einem Formular verwendet werden. Die Identifikation solcher Gruppen mittels Mining von Korrelationen funktioniert aufgrund der Häufigkeitsanalyse nur dann, wenn für die Erstellung der Korrelationstabellen semantisch ähnliche Formularelemente „gezählt“ werden können. Dies könnte durch die Erstellung gemeinsamer Miningobjekte aus ähnlichen Formularelementen durchgeführt werden. Ähnliche Formularelemente werden hierbei zu einem Miningobjekt verschmolzen, welches dann immer anstelle der Formularelemente in diesem Formular verwendet wird. Hierdurch werden ähnliche Formularelemente zählbar. Die gefundenen positiven Korrelationen müssen dann allerdings nach dem Miningprozess wieder auf Formularebene aufgelöst werden, sodass die konkreten semantischen Gruppen innerhalb eines Formulares gespeichert werden können.

**Identifikation passender Abbildungspartner**

Nach der Identifikation der semantischen Gruppen, die potenziell Teil einer komplexen Abbildung sein können, müssen noch passende Abbildungspartner zur Erstellung von komplexen Abbildungen gefunden werden. Hierfür können die paarweise berechneten semantischen Ähnlichkeiten zwischen Formularelementen unterschiedlicher Formulare verwendet werden. Konkret kann dann für jedes Formularelement überprüft werden, ob eine semantische Gruppe eines anderen Formulares ähnlich zu diesem Formularelement ist. Hierfür muss die Ähnlichkeit zwischen einem Formularelement und einer semantischen Gruppe von Formularelementen definiert werden. Eine Möglichkeit zur Identifikation von Abbildungspartnern bietet wieder das Mining von Korrelationen. Hierbei werden in diesem Fall negative Korrelationen gesucht. Ein Formularelement und eine semantische Gruppe korrelieren stark negativ, wenn sie nie gemeinsam in einem Formular anzutreffen sind. Negative Korrelationen spiegeln Synonymbeziehungen wider. Außer negativen Korrelationen können ebenfalls linguistische Charakteristika und Wertebereiche der Formularelemente überprüft werden. Falls die Wertebereiche einer semantischen Gruppe jeweils ähnlich zum Wertebereich oder den Unterwertebereichen des abzubildenden Elementes sind, so deutet dies auf eine komplexe Abbildung hin [WYDM04]. Außerdem kommt es häufig vor, dass ebenfalls linguistische Ähnlichkeiten zwischen dem abzubildenden Element und den Formularelementen der semantischen Gruppe gefunden werden können. So könnte also eine linguistische Ähnlichkeit zwischen einem Formularelement mit dem Namen „*dep_date*“ zu der roten semantischen Gruppe in Abbildung 5.11 gefunden werden. Reine semantische Vergleiche können allerdings nur sehr eindeutige Abbildungen identifizieren, da nur bei solchen Abbildungen linguistische Ähnlichkeiten und Ähnlichkeiten zwischen den Wertebereichen gefunden werden können. Sobald also im vorigen Beispiel der Name „*date*“ anstatt „*dep_date*“ ist, kann keine linguistische Ähnlichkeit mehr gefunden werden. Wenn dieses Formularelement dann auch noch ein Texteingabefeld ohne weitere semantische Informationen wie `placeholder` oder `pattern` ist, so kann auch über die Wertebereiche keine Ähnlichkeit festgestellt werden. Fraglich ist also, wie solche Abbildungen dann identifiziert werden können, wenn die semantischen Ähnlichkeiten zwischen Formularelement und semantischer Gruppe nicht ausreichen. Eine Option hierfür ist die Verwendung von

spezifischen Mustern, die anhand der Eigenschaften des abzubildenden Elementes festlegen, wie potenzielle Abbildungspartner aussehen könnten. Eine mögliche Eigenschaft wäre die Zusammensetzung von Wertebereichen. Wenn beispielsweise herausgefunden werden kann, dass es sich bei dem derzeit betrachteten Element um ein Datumsfeld handelt, kann der Wertebereich der Zielfelder eingegrenzt werden. Dies kann durch Überprüfung von `label`, `name`, `value`, `id` oder `placeholder` auf Charakteristika eines Datums erkannt werden. Ein solches Muster könnte die Suche nach einer semantischen Gruppe einschränken, indem nach einer Gruppe gesucht wird, welche drei Formularelemente mit numerischem Wertebereich beinhaltet. Diese Werte stehen jeweils für Tag, Monat und Jahr. In Abschnitt 5.3 wird beschrieben, wie solche Muster erkannt werden können und wie daraus konkrete Regeln zur Identifikation komplexer Abbildungen erstellt werden können. Nachdem einige komplexe Abbildungen erkannt wurden, können mithilfe von einfachen Abbildungen und der Transitivität neue komplexe Abbildungen generiert werden. Einfache Abbildungen können im Rahmenwerk EASIER bereits mithilfe eines einfachen Clusterverfahrens ([HMYW03], [WYDM04]) identifiziert werden. Hierfür werden alle Formularelemente in semantisch ähnliche Cluster aufgeteilt. Ein Cluster enthält dann alle Elemente, die ähnlich zueinander sind, also Abbildungen repräsentieren.

## 5.3. Lösung komplexer Abbildungen

Im Rahmen dieser Arbeit sollen komplexe Abbildungen erkannt werden, die zur Generierung von AOs im Rahmenwerk EASIER verwendet werden können. Eine Lösung einer komplexen Abbildung ist eine Abbildungsvorschrift, welche die Abbildung definiert und durch die ihre Elemente auf AO-Knoten in EASIER abgebildet werden können. Eine solche Lösung kann mithilfe von identifizierten Mustern in Abbildungen erstellt werden. Häufig zeigen Abbildungen ähnliche Abbildungsmuster. Diese Muster können dann dazu verwendet werden, Abbildungsvorschriften zu konstruieren. Diese können im Anschluss daran verwendet werden, um Abbildungselemente automatisch auf AOs im Rahmenwerk EASIER abzubilden und diesen Abbildungstyp besser zu erkennen.

### 5.3.1. Muster

Oftmals können in Abbildungen einfache Muster erkannt werden. Diese Muster können später verwendet werden, um Abbildungen zu identifizieren und Abbildungspartner ineinander umzuwandeln.

Falls konkrete Muster einen Abbildungstypen definieren, ist es hilfreich, diese Muster auch bei der Identifikation solcher Abbildungen zu verwenden. Muster zeigen, wie eine komplexe Information aus den einfachen Elementen der Abbildungen erstellt werden kann und wie die komplexe Information in seine einzelnen einfachen Informationen zerlegt werden kann.

Nach der Betrachtung einiger Formulare fällt auf, dass oftmals ein Muster für eine 1:m-Abbildung durch das Zusammenfügen der $m$ Elemente durch Trennzeichen wie Leerzeichen oder Punkt erstellt werden kann.

**Beispiel: Mustererkennung und Abbildungsvorschrift**

Sei $Datum = \{Tag, Monat, Jahr\}$ eine häufige Abbildung. Nun soll eine Abbildungsvorschrift konstruiert werden, wodurch diese Abbildung zukünftig leichter gefunden und auf AO-Knoten abgebildet werden kann. Hierbei kann festgestellt werden, dass diese Abbildung ein konkretes Muster aufweist. Ein Datum besteht immer aus drei Informationen für Tag, Monat und Jahr. Tage und Jahre sind hierbei immer Zahlen, während der Monat entweder als Zahl oder als Zeichenkette angegeben werden kann. Diese drei Informationen sind jeweils durch Trennzeichen wie beispielsweise „/“ im Englischen oder „.“ im Deutschen getrennt. Durch diese Abbildungsvorschrift kann

dann zukünftig ein Datum bereits durch dieses Muster in einem Label oder einem Platzhalter erkannt werden und nach drei korrespondierenden Eingabefeldern mit entsprechendem Wertebereich gesucht werden. Außerdem wird hierdurch festgelegt, wie ein komplexes Element aus einfachen Elementen konstruiert werden kann.

Es muss darauf geachtet werden, dass die Datentypen miteinander kompatibel sind. Das bedeutet, dass die einfachen Datentypen des Musters im komplexen Datentyp verwendet werden können beziehungsweise eine Untermenge des Datentyps sind. So kann im Beispiel sowohl der Datentyp *int* der Elemente Tag, Monat und Jahr, als auch die Trennzeichen „/“ und „.“ im komplexen Datentyp *Zeichenkette* verwendet werden. Ebenso kann die Zeichenkette durch Zerteilen an den Trennzeichen in drei Zahlen vom Typ *int* umgewandelt werden. Aus den erkannten Mustern können anschließend Regeln herausgearbeitet werden, wodurch die Identifikation von Abbildungen vereinfacht wird. Im Hinblick auf die Generierung aktiver Ontologien können diese Regeln helfen, Informationen auf Konzepte abzubilden. Eine mögliche aktive Ontologie für das Datumsbeispiel könnte wie in Abbildung 5.12 aussehen.

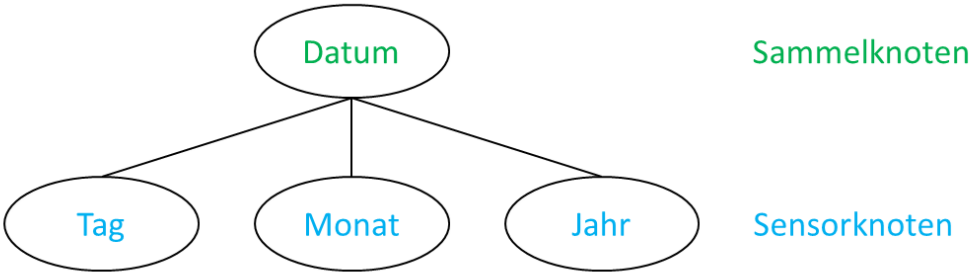



Abbildung 5.12.: Aktive Ontologie für das Datumsbeispiel. Die drei Sensorknoten Tag, Monat und Jahr leiten ihre Informationen an den Sammelknoten Datum weiter. Der Sammelknoten kann nun die Regel verwenden, um diese Daten zusammenzusetzen. Hierfür wird eine Zeichenkette erstellt. Es wird wie in der Regel spezifiziert Tag, Trennzeichen, Monat, Trennzeichen und Jahr nacheinander zur Zeichenkette hinzugefügt.

Die hergeleitete Regel kann nun dabei helfen ein als Zeichenkette eingegebenes Datum zu zerlegen und den Konzepten der aktiven Ontologie zuzuordnen oder einzeln eingegebene Werte für Tag, Monat und Jahr im Sammelknoten zu einer Information zu verschmelzen. Hierbei sind die Regeln vor allem wichtig für die Informationseingabe beim Aufruf verschiedener Dienstgeber.

**Beispiel: Automatische Eingabe Dienstgeber**

*Dienstgeber A* hat eine Datumseingabe als einfaches Texteingabefeld. *Dienstgeber B* hingegen erwartet die Eingabe des Tages und des Monates anhand von zwei Auswahllisten. Für die Eingabe des Jahres hat *Dienstgeber B* ein Texteingabefeld.

Beim Aufruf verschiedener Dienstgeber muss der Aufrufer nun in der Lage sein, die erkannten Informationen der Ontologie so umzuwandeln, dass diese als gültige Eingabe in den Zielformularen akzeptiert werden. Durch die in der Regel gespeicherten Kon-

vertierungsregeln können die in der AO gespeicherten Informationen in das passende Zielformat umgewandelt werden.

### 5.3.2. Regelentwurf

Nachdem Muster für Abbildungen erkannt wurden, muss überlegt werden, wie daraus Regeln konstruiert werden können. Regeln sollten so allgemein wie möglich gehalten werden, damit sie für jeden Anwendungsfall dieses Abbildungstyps verwendet werden können. Hierbei kann es helfen, zunächst aus Mustern Regeln zu entwerfen und diese dann so weit wie möglich zu verallgemeinern. Ein möglicher Regelentwurf könnte wie folgt aussehen. Das Muster *„DD.MM.YYYY“* wurde für Abbildungen eines deutschen Datums erkannt. Hieraus kann die Regel entworfen werden, dass bei einem Datum Tag, Monat und Jahr durch einen Punkt getrennt sind. Hierbei ist ein Tag vom Typ *int* mit dem Wertebereich {*1..31*}, Monat vom Typ *int* mit dem Wertebereich {*1..12*} und Jahr vom Typ *int* mit dem Wertebereich vierstelliger Zahlen {*1000..9999*}. Diese Wertebereichsinformationen können verwendet werden, um mithilfe der Regel diesen Abbildungstyp zu erkennen. Nachdem ein Datumsfeld anhand des Musters identifiziert wurde, kann gezielt nach einer semantischen Gruppe mit diesen Eigenschaften gesucht werden. Auf Basis dieser Regel können dann weitere Regeln zur besseren Abdeckung des Abbildungstyps definiert werden. Im Beispiel würden dann noch andere Trennzeichen außer dem Punkt betrachtet werden. In einigen Datumsformaten werden Schrägstriche, Bindestriche oder sogar Unterstriche verwendet. In anderen Sprachen können außerdem Tag und Monat in der Reihenfolge vertauscht sein. Hierfür könnte dann beispielsweise eine Regel für englische Daten erstellt werden, die auf dem Muster *„MM/DD/YYYY“* basiert. Die Daten selbst könnten natürlich auch in einer anderen Form gegeben sein. Das Jahr könnte als zweistellige Zahl repräsentiert werden. Der Monat könnte auch durch die Menge {*Januar, Februar, März, ..., Dezember*} in der jeweils unterstützten Sprache repräsentiert werden. Auch hiermit sollten die Regeln des Abbildungstyps umgehen können. Durch das weitere Hinzufügen solcher Regeln verbessert sich die Abdeckung dieses Abbildungstyps und es können mehr komplexe Abbildungen dieses Typs erkannt werden. Außerdem speichert jede dieser Regeln Konvertierungsregeln, die festlegen wie einfache und komplexe Informationen ineinander umgewandelt werden können.

# 6. Entwurf und Implementierung

Im vorangegangenen Kapitel wurden die Probleme und Limitierungen der Identifikation komplexer Abbildungen analysiert und Möglichkeiten zur Lösung dieser aufgezeigt. Dieses Kapitel beschäftigt sich mit der konkreten Lösung dieser Probleme und deren Umsetzung. Zunächst wird der Entwurf des zu erstellenden Werkzeuges dokumentiert. Am Ende jedes Entwurfskapitels wird die konkrete Implementierung vorgestellt. Der Aufbau des Werkzeuges wird anhand des Prozessablaufes in Abbildung 6.1 erläutert. Im Rahmen der Bachelorarbeit wird eine Menge an Webformularen aus derselben Dienstkategorie (z.B. Flugbuchung) auf komplexe Abbildungen von Formularelementen untersucht. Hierfür werden die Formulare zunächst aufbereitet und in eine eigene Datenstruktur überführt, sodass einfache und effiziente semantische Vergleiche möglich sind. Dann werden die Eingabedaten normalisiert, um die Daten vergleichen zu können. Anschließend können die Daten auf semantische Ähnlichkeiten geprüft werden. Hierzu werden die Formularelemente zunächst paarweise auf Ähnlichkeiten analysiert. Untersucht werden hierbei linguistische Charakteristika, strukturelle Eigenschaften und mögliche Wertebereiche der Formularelemente. Hiernach können anhand der Ähnlichkeiten in einem Identifikationsverfahren komplexe Abbildungen identifiziert werden. Hierzu werden zunächst anhand von vordefinierten Regeln potenzielle komplexe Abbildungskandidaten identifiziert. Anschließend wird das in EASIER implementierte Clusterverfahren zur Identifikation einfacher Abbildungen verwendet. Durch die identifizierten komplexen Abbildungen und die durch das Clustering gefundenen einfachen Abbildungen werden dann durch die Abbildungstransitivität neue Abbildungen generiert. Komplexe Abbildungen weisen oftmals Muster auf, die diese Abbildungen charakterisieren. Diese Muster werden nicht automatisiert erkannt, sondern manuell herausgearbeitet. Aus diesen Mustern werden dann Regeln abgeleitet, die dann verwendet werden können, um Abbildungen zwischen Formularelementen und AOs im Rahmenwerk EASIER zu entwerfen. Hierdurch kann die AO ihre gesammelten Informationen jederzeit in das passende Format des Zielformulares umwandeln. Für die Umsetzung können einige der in Kapitel 3 vorgestellten Ansätze zur Lösung von Teilproblemen wiederverwendet werden.

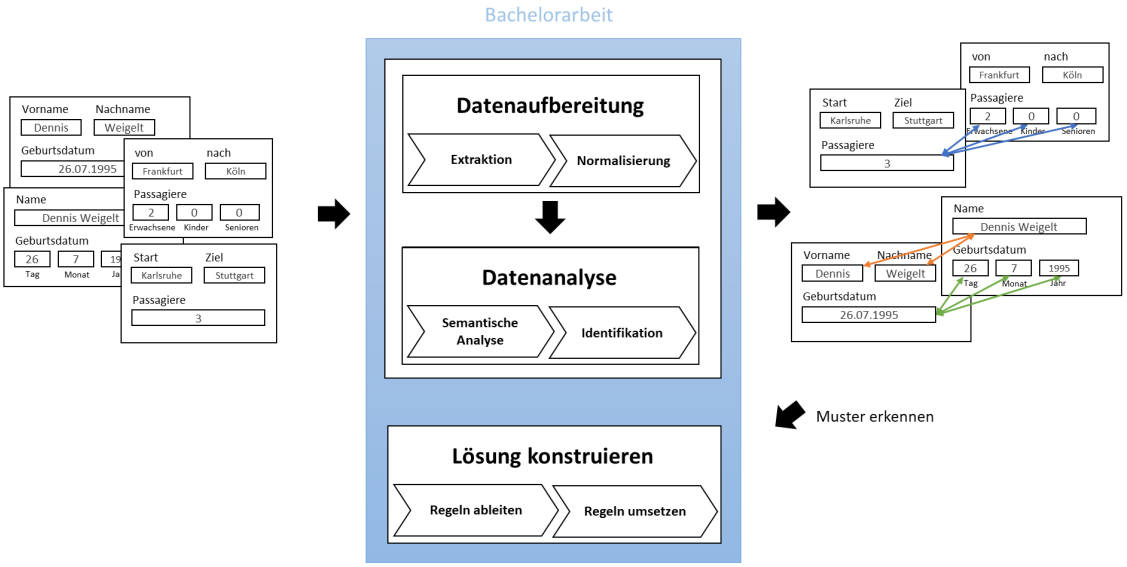



Abbildung 6.1.: Prozessablauf für das erarbeitete Werkzeug. Die HTML-Formulare werden zuerst extrahiert und aufbereitet. Danach werden die Daten semantisch analysiert und komplexe Abbildungen identifiziert. Anhand der identifizierten Abbildungen können dann Muster erkannt werden. Aus diesen Mustern werden dann Regeln entworfen, die diese Muster erkennen können.

## 6.1. Datenaufbereitung

In der Datenaufbereitung geht es darum, die in Kapitel 5 identifizierten wichtigen Informationen von unwichtigen zu trennen. Da die Struktur von Formularen ein wichtiges Vergleichskriterium bietet, müssen diese Informationen in einem strukturerhaltenden Extraktionsprozess so extrahiert und gespeichert werden, dass keine semantischen und strukturellen Informationen verloren gehen.

### 6.1.1. Extraktion aus HTML-Formularen

Zur Analyse der Formulardaten auf semantische Beziehungen zueinander werden diese zunächst extrahiert. Hierfür müssen einige Teilprobleme gelöst werden. Die hierarchische Struktur von HTML kann bei der Identifikation von Ähnlichkeiten auf struktureller Ebene verwendet werden. Der strukturelle Aufbau von Formularen ist zwar bei jedem Webanbieter anders, jedoch gestalten Webdienste ihre Oberflächen oft so, dass semantisch ähnliche Informationen auch strukturell gruppiert sind. Es muss also ein Extraktionsverfahren für HTML-Formulare und eine entsprechende Datenstruktur entworfen werden, sodass die enthaltenen strukturellen Merkmale in die Zieldatenstruktur abgebildet werden können. Hierdurch bleiben strukturelle Merkmale wie die Reihenfolge der Elemente und semantische Gruppen erhalten. Diese Merkmale können dann dazu verwendet werden, um Tupel von Formularelementen in einem Formular anhand der strukturellen Gruppierung zu identifizieren. Zur Abbildung von Hierarchien eignet vor allem eine Baumstruktur [AHS12][WYDM04]. In diesem Fall repräsentieren Blattknoten die Formularelemente und deren Elternknoten die semantische Gruppe zu der sie gehören. Durch diese Struktur können die in HTML hierarchisch repräsentierten Gruppen und deren Formularelemente ohne Verlust der strukturellen Semantik in die Datenstruktur überführt werden. Ein Baum als Datenstruktur eignet sich ebenfalls im Hinblick auf effiziente semantische Vergleiche, da

hierdurch die Ähnlichkeit zweier semantischer Gruppen lediglich durch einen Vergleich der Ähnlichkeiten der entsprechenden Teilbäume berechnet werden kann. Während des Extraktionsprozesses wird dann für jedes Formular ein Formularbaum erstellt, welcher die Formularelemente und deren Attribute speichert. Hierfür müssen den Formularelementen zunächst ihre Bezeichner zugeordnet werden. Es muss beachtet werden, dass Bezeichner sowohl das Formularelement umschließen können, als auch getrennt vom zu bezeichnenden Element im HTML-Formular stehen können. Geschachtelte Bezeichner können einfach zugeordnet werden, da sie ihr bezeichnendes Element umschließen. Falls der Bezeichner allerdings an einer anderen Stelle im Formular steht, muss dieser zunächst über das Attribut `id` zugeordnet werden. Die Zuordnung geschieht dadurch, dass der Inhalt des Bezeichners als zusätzliches Attribut `label` im Formularelement gespeichert wird. Außer der Zuordnung der Bezeichner müssen außerdem die Wertebereiche der Formularelemente gespeichert werden. Hierfür werden jeweils Wertebereiche aus Auswahllisten, Checkboxes, Radio Buttons und anderen Optionsfeldern erstellt und dem Baumknoten hinzugefügt. Somit speichert ein Elementknoten im Baum die Eigenschaften `label`, `name`, `id`, `placeholder`, `pattern` und `value`. Bei `<input>`-Elementen wird noch das Attribut `type` als Eigenschaft gespeichert. Außerdem werden die Wertebereiche und jeweils eine Liste der vorgehenden und nachfolgenden Geschwisterelemente dem Baumknoten hinzugefügt. Abbildung 6.2 zeigt ein Beispiel der Extraktion eines Formulares aus der Bücherdomäne. Zur Vereinfachung werden Formularelemente in der Baumstruktur anhand ihrer Labels repräsentiert. Die Elemente werden in Formularreihenfolge von links nach rechts in den Baum geschrieben, sodass das erste Element ganz links steht und das letzte Element ganz rechts. Hierdurch bleibt ebenfalls die Reihenfolge der Elemente intakt und kann verwendet werden.

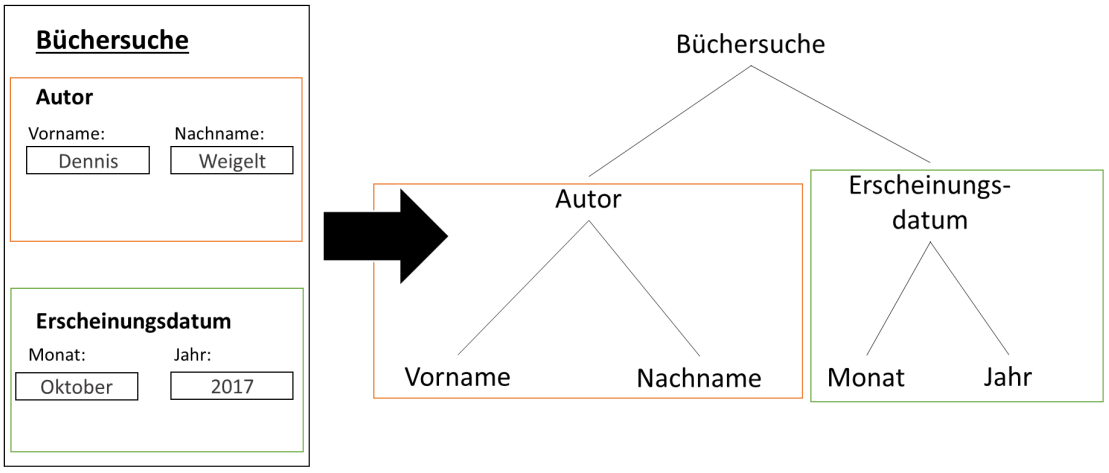



Abbildung 6.2.: Überführung eines HTML-Formulares in die Baumstruktur

Jeder Blattknoten in Abbildung 6.2 repräsentiert ein Formularelement. Alle anderen Knoten repräsentieren Gruppen, die wiederum wieder aus Gruppen bestehen können. So besteht die Supergruppe „Büchersuche“ aus den beiden Gruppen „Autor“ und „Erscheinungsdatum“. Blattknoten und ihre gespeicherten Informationen werden im Folgenden „Terme“ genannt.

### 6.1.2. Normalisierung

Da später semantische Ähnlichkeiten zwischen den Formularelementen anhand von Vergleichen festgestellt werden sollen, bietet sich zunächst eine Normalisierung der Baumknoten

an. Vor allem potenziell lange Zeichenketten wie Bezeichner oder Wertebereichsrepräsentanten sind aufgrund von Trennzeichen und Groß-/Kleinschreibung besonders schwierig zu vergleichen. Zur Normalisierung werden die Zeichenketten zunächst tokenisiert. Hierzu können Ansätze aus [WYDM04] und [HMYW03] verwendet werden. Die Zeichenketten werden an Großbuchstaben, Zahlen und nicht-alphanumerischen Zeichen getrennt und in so genannte „Token“ zerteilt. Diese Tokenmengen können dann anhand einer paarweisen Analyse ihrer Token verglichen werden. Um die Vergleichbarkeit der Token zu erhöhen, wird die Groß- und Kleinschreibung eliminiert, indem alle Großbuchstaben in Kleinbuchstaben umgewandelt werden. Hierdurch hätte die Schreibweise der Buchstaben keinen nachteiligen Einfluss mehr auf linguistische Tokenvergleiche. Da außerdem Stoppwörter das Ergebnis eines solchen Vergleiches verfälschen können, werden die gängigen englischen Stoppwörter aus den Tokenmengen entfernt. Das Ergebnis der Normalisierung einer Zeichenkette ist dann ein Tupel von Token, welche zusammengefügt die ursprüngliche Zeichenkette ohne Stoppwörter repräsentieren. Der Normalisierungsprozess ist in Abbildung 6.3 anhand einer Beispielzeichenkette verdeutlicht. Zuerst wird die Zeichenkette durch die definierten Regeln der Tokenisierung in Token aufgeteilt. Anschließend werden alle Großbuchstaben in Kleinbuchstaben umgewandelt und Stoppwörter aus der Tokenmenge entfernt. Da oftmals Präfixe wie „input“ oder „in“ für Eingabeformularelemente verwendet werden, wird das Token „eingabe“ aus der Tokenmenge entfernt.

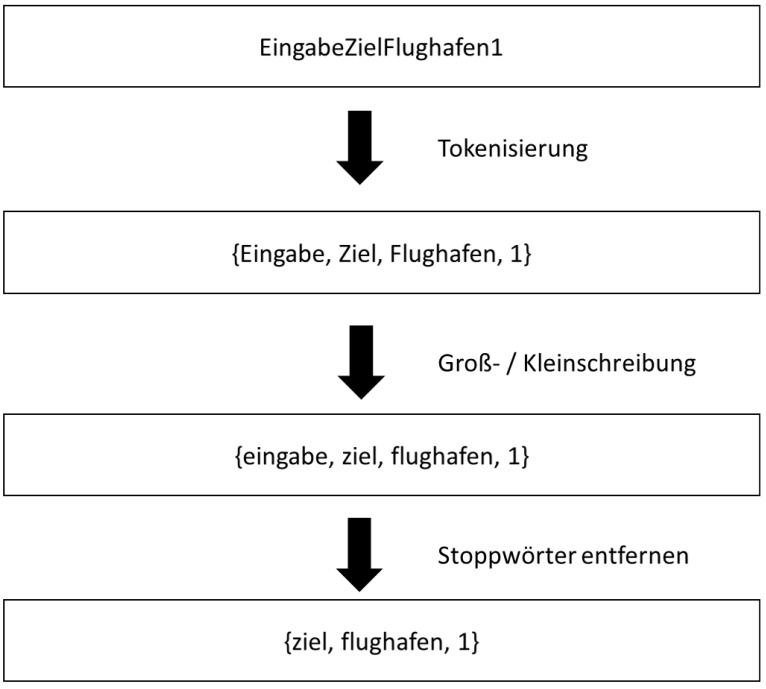



Abbildung 6.3.: Normalisierung der Zeichenkette „EingabeZielFlughafen1“

Normalisiert werden alle Zeichenketten eines Formularelementes. Schlussendlich erhält man tokenisierte Baumknoten, die gute linguistische Vergleiche ermöglichen.

### 6.1.3. Implementierung

Implementiert wird die Extraktion mithilfe der Bibliothek JSOUP[1] [noad]. JSOUP erstellt aus HTML-Formularen einen „Document Object Model (DOM) - Baum“. Der DOM-Baum

[1] JSOUP, online erhältlich unter https://jsoup.org/; abgerufen am 21. Januar 2018

stellt das Formular in seiner von HTML gegebenen hierarchischen Struktur dar. Mithilfe dieses DOM-Baumes können die Formularelemente dann als „Terme“ in die gewünschte Datenstruktur überführt werden. Anschließend werden die Terme normalisiert. Diese Funktionalitäten stellt bereits das Rahmenwerk EASIER zur Verfügung.

## 6.2. Datenanalyse

Im nächsten Schritt werden die zuvor extrahierten und normalisierten Baumknoten auf semantische Ähnlichkeiten geprüft. Hierzu werden zunächst alle Terme eines Formulares mit den Termen aller anderen Formulare paarweise analysiert. Beim Termvergleich werden Ähnlichkeiten zwischen den Eigenschaften verglichen. Diese semantischen Ähnlichkeiten tragen anschließend dazu bei, potenzielle Kandidaten für komplexe Abbildungen zu finden. In einem Auswahlverfahren können dann die korrekten Abbildungen aus der Kandidatenliste bestimmt werden. Die durchzuführende semantische Analyse besteht aus einer linguistischen Analyse, einer Strukturanalyse und einer Analyse der Wertebereiche. Hierzu werden für jedes Formularpaar die Ähnlichkeiten ihrer Terme gespeichert. Die Berechnung der folgenden drei semantischen Teilanalysen ist bereits im Rahmenwerk EASIER zur Identifikation einfacher Abbildungen implementiert und kann wiederverwendet werden.

### 6.2.1. Linguistische Analyse

Bei der linguistischen Analyse werden die Eigenschaften der Terme linguistisch miteinander verglichen. Als Eingabe für die linguistische Analyse dienen die normalisierten Terme aus dem Datenaufbereitungsschritt. Zur Berechnung der linguistischen Ähnlichkeit kann der Ansatz aus [WYDM04] erweitert werden. Wu et. al beschränken sich hierbei auf die Eigenschaften `name` und `label` der Formularelemente. In der Arbeit wird zunächst die Ähnlichkeit der Namen und die Ähnlichkeit der Bezeichner getrennt voneinander ermittelt. Im Anschluss daran wird dann noch die Ähnlichkeit zwischen Bezeichner und Name berechnet. Diese Berechnung wird in [WYDM04] mithilfe der Kosinusfunktion durchgeführt, die in Gleichung 6.1 dargestellt ist.

$$Cos(a, b) = \frac{a \cdot b}{||a|| \cdot ||b||} \tag{6.1}$$

Der $Cos(a, b)$ wird berechnet, indem das Skalarprodukt der Vektoren $a$ und $b$ durch das Produkt ihrer Normen geteilt wird. Für die Kosinusfunktion werden zwei Vektoren benötigt. Die Vektoren werden wie folgt aus den Tokenmengen $A$ und $B$ zweier Formularelemente konstruiert:

1. Konstruiere die Vereinigungsmenge $V = A \cup B$.

2. Stelle die Tokenmengen $A$ und $B$ als Vektor der Länge $|V|$ dar, sodass $v_i = 1$ gilt, wenn das $i$-te Token von $V$ in $A$ beziehungsweise $B$ vorkommt.

Nach der Konstruktion der Vektoren kann die linguistische Ähnlichkeit durch einsetzen in die Kosinusfunktion berechnet werden. Die Konstruktion der Vektoren ist anhand eines Beispiels in Abbildung 6.4 veranschaulicht.

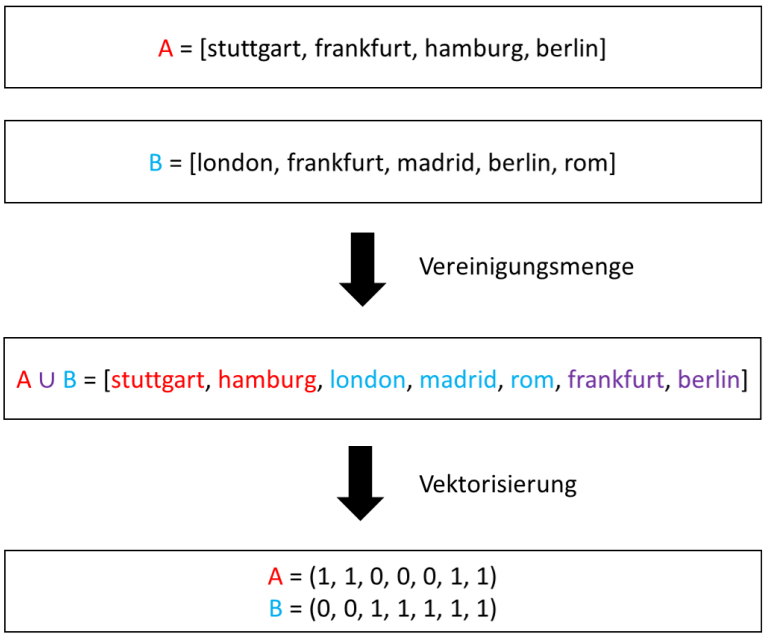



Abbildung 6.4.: Vektorisierung von zwei Tokenmengen zur Berechnung der Ähnlichkeit mithilfe der Kosinusfunktion nach [WYDM04].

Für den konkreten Anwendungsfall in EASIER macht es jedoch Sinn, nicht nur Bezeichner und Namen der Formularelemente zu betrachten, da Formulare mehr semantisch vergleichbare Entitäten als `name` und `label` bieten. Auch Attribute wie `placeholder`, `id` und `value` können einen hohen semantischen Wert haben und sollten ebenfalls verglichen werden. Hierfür kann die Formel zur Berechnung der Ähnlichkeit der Namen und Bezeichner so erweitert werden, dass alle Attribute miteinander verglichen werden können. Das folgende Beispiel zeigt den Vergleich der beiden Attribute `placeholder` und `value`

**Beispiel: Vergleich `placeholder` und `value`**

Sei $value(A)$ der Wert des Attributs `value` des Formularelementes $A$ und analog $placeholder(A)$ der Wert des Formularelementes `placeholder` aus Formular $A$. Um zwei Formularelemente $A$ und $B$ anhand der beiden Attribute `placeholder` und `value` zu vergleichen, werden zunächst die paarweisen Ähnlichkeiten berechnet. Hierfür werden $sim(placeholder(A), placeholder(B))$ und $sim(value(A), value(B))$ berechnet. Anschließend wird das Maximum der Ähnlichkeiten $sim(placeholder(A), value(B))$ und $sim(value(A), placeholder(B))$ berechnet. Die Ähnlichkeit $sim()$ wird mithilfe der Kosinusfunktion berechnet. Die beiden paarweisen Ähnlichkeiten und das Maximum der zuvor berechneten attributübergreifenden Ähnlichkeiten bilden dann mithilfe einer Gewichtung dieser Werte die linguistische Ähnlichkeit.

Hierbei werden dann alle möglichen Kombinationen zwischen den Attributen durch die Kosinusfunktion getestet und als Teilähnlichkeit gespeichert. Die maximale Teilähnlichkeit ergibt dann die linguistische Ähnlichkeit. Zusätzlich zum Vektorvergleich durch die Kosinusfunktion wird noch die Ähnlichkeit der Token anhand der längsten gemeinsamen Teilzeichenkette berechnet. Die Ähnlichkeit zweier Zeichenketten $S$ und $T$ kann durch die längste gemeinsame Zeichenkette $L$ mithilfe von Gleichung 6.2 berechnet werden.

$$sim_z(S,T) = \frac{|L|}{max(|S|,|T|)}. \tag{6.2}$$

Diese Analyse wird ebenfalls für alle Paare von Eigenschaften aus den Termen durchgeführt. Das Ergebnis der linguistischen Analyse wird dann durch eine Gewichtung der Vektoranalyse und der Teilzeichenkettenanalyse definiert.

### 6.2.2. Strukturelle Analyse

Bei der strukturellen Analyse kann die Ähnlichkeit anhand von strukturellen Merkmalen wie beispielsweise Positionierung oder Nähe der Elemente im Formular definiert werden. Anhand der extrahierten hierarchischen Baumstruktur kann festgestellt werden, ob sich zwei Formularelemente strukturell ähnlich sind, indem ihre Teilbäume verglichen werden. Da Elemente einer semantischen Gruppe im Formular oftmals benachbart sind, können ähnliche Nachbarn ein Indiz auf Ähnlichkeit der zu vergleichenden Formularelemente sein. Um zwei Formularelemente strukturell vergleichen zu können, werden also die Nachbarn der jeweiligen Elemente auf Ähnlichkeit zueinander überprüft. Hierzu können die Teilbäume der beiden zu vergleichenden Elemente untersucht werden. Um diese Teilbäume auf Ähnlichkeit zu prüfen, werden die Nachbarelemente der zu vergleichenden Elemente auf linguistische Ähnlichkeiten überprüft. Die strukturelle Ähnlichkeit der $i$-ten Nachbarelemente wird wie in Gleichung 6.3 und Gleichung 6.4 berechnet.

$$SA(prec_{T1}, prec_{T2}) = \frac{1}{4 * 2^i} \cdot lingSim(prec_{T1}(i), prec_{T2}(i)) \tag{6.3}$$

$$SA(succ_{T1}, succ_{T2}) = \frac{1}{4 * 2^i} \cdot lingSim(succ_{T1}(i), succ_{T2}(i)) \tag{6.4}$$

Hierbei repräsentiert $succ_T(i)$ den $i$-ten Nachfolger und $prec_T(i)$ den $i$-ten Vorgänger von Term $T$. $LingSim()$ bezeichnet die linguistische Ähnlichkeit. Durch den Faktor $\frac{1}{4*2^i}$ werden weiter entfernte Nachbarn weniger gewichtet. Die strukturelle Teilähnlichkeiten der Vorgänger und der Nachfolger ergeben gemittelt die strukturelle Ähnlichkeit.

### 6.2.3. Wertebereichsanalyse

Zur Berechnung der Wertebereichsähnlichkeiten können Ansätze aus [WYDM04] verwendet werden. Die Ähnlichkeit der Wertebereiche berechnet sich durch die Typähnlichkeit des Wertebereiches und die Ähnlichkeit der Werte im Wertebereich. Die Bestimmung der Typähnlichkeit wird in EASIER mithilfe von Formulargruppentypen durchgeführt. Oftmals sind Formularelementtypen in der semantischen Bedeutung ähnlich und können gruppiert werden. Diese Gruppentypen können wiederverwendet werden. In Tabelle 6.1 wird für jeden Gruppentyp angegeben, welche Formularelemente diesem in EASIER zugeordnet werden. Die Typähnlichkeit $TA$ lässt sich dann wie folgt berechnen. $TA$ ist 1, wenn die beiden Formularlemente denselben Typ haben, ansonsten 0. Etwas schwieriger hingegen ist der Vergleich der Werte im Wertebereich. Hierbei gibt es drei verschiedene Vergleichsarten, die zu berücksichtigen sind [WYDM04]. Erstens sollen zwei Listen $d$ und $d'$ miteinander vergleichbar sein. Dies repräsentiert einen Vergleich zwischen zwei endlichen Wertebereichen von Zeichenketten. Um die Zeichenketten der Listen miteinander zu vergleichen, kann wieder die Kosinusfunktion aus Gleichung 6.1 verwendet werden. Mithilfe der Kosinusfunktion werden dann zuerst die paarweisen Ähnlichkeiten der beiden Wertebereiche berechnet. Nachdem die paarweisen Kosinusberechnungen vorliegen, wird das Paar mit der höchsten Ähnlichkeit ausgewählt, aus den Wertebereichen entfernt und in eine neue Menge $C$ eingefügt. Dieser Schritt wird so lange wiederholt, bis die Ähnlichkeiten der Paare einen festgelegten Schwellwert nicht mehr erreichen. Die Ähnlichkeit der Listen wird dann mithilfe der Dice-Funktion [DR02][Dic45] berechnet:

$$WB = \frac{2 * |C|}{|d| + |d'|} \tag{6.5}$$

| Globaler Typ | Elementtyp |
|---|---|
| TEXT_NO_PATTERN | <input type="text■ |
| | <input type=ßearch■ |
| | <input type="password■ |
| | <textarea> |
| TEXT_PATTERN | Alle Elemente von TEXT_NO_Pattern mit Angabe des Attributes pattern |
| PATTERN | <input type="tel■ |
| | <input type=ürl■ |
| | <input type=ëmail■ |
| NUMERICAL | <input type="number■ |
| | <input type="range■ |
| CHOICE | <input type="radio" |
| | <input type="checkbox■ |
| | <select> |
| TIMEDATE | <input type="date■ |
| | <input type="datetime-local■ |
| | <input type="datetime■ |
| | <input type="week■ |
| | <input type="month■ |
| | <input type="time■ |

Tabelle 6.1.: Übersicht über die globalen Formulargruppentypen in EASIER.

Zweitens sollen zwei numerische Wertebereiche $h$ und $h'$ verglichen werden. Um zwei numerische Wertebereiche zu vergleichen, wird überprüft wie groß der Schnitt der beiden Intervalle ist [WYDM04]. Die Ähnlichkeit zweier numerischer Wertebereiche wird dann wie folgt berechnet:

$$WB = \frac{min\{max(h), max(h')\} - max\{min(h), min(h')\}}{max\{max(h), max(h')\} - min\{min(h), min(h')\}}, \tag{6.6}$$

wobei $min(x)$ und $max(x)$ jeweils das Minimum beziehungsweise Maximum des Wertebereiches $x$ ausgeben. Da Wertebereiche von <input>-Elementen oftmals nicht vollständig in Listenform oder numerischer Repräsentation konstruiert werden können, werden in diesem Fall die Typen zur Ermittlung der Ähnlichkeit der Werte im Wertebereich verwendet. Haben zwei Formularelemente denselben Wert des Attributes type von <input>, so gilt $WB = 1$, ansonsten $WB = 0$ [WYDM04]. Ein Sonderfall hierbei ist der <input>-Typ „text“, da dieser mit allen Eingabefeldern übereinstimmen kann [WYDM04]. Vergleiche, in denen Elemente mit dem Typ „text“ vorkommen, erhalten die Ähnlichkeit $WB = 0$. Die Ähnlichkeit von zwei Wertebereichen wird dann mithilfe einer Gewichtung der beiden Teilähnlichkeiten $TA$ und $WB$ berechnet.

### 6.2.4. Berechnung der Ähnlichkeit zweier Formularelemente

Anhand der zuvor erläuterten Ähnlichkeiten kann die semantische Ähnlichkeit zweier Formularelemente definiert werden. Sei $LA$ das Ergebnis der linguistischen Analyse, $SA$ das Ergebnis der strukturellen Analyse und $WA$ das Resultat der Wertebereichsanalyse, so berechnet sich die Gesamtähnlichkeit der Felder nach [WYDM04] durch die Gewichtung der Einzelanalysen:

$$SemA = \omega_{LA} * LA + \omega_{SA} * SA + \omega_{WA} * WA, \tag{6.7}$$

wobei $\omega_{LA}, \omega_{SA}$ und $\omega_{WA}$ Gewichtungsparameter der semantischen Ähnlichkeit sind.

## 6.3. Komplexer Identifikationsalgorithmus

In diesem Abschnitt wird beschrieben, wie komplexe Abbildungen mit Einbeziehung der zuvor berechneten semantischen Analyse gefunden werden können. Der verwendete Identifikationsalgorithmus für komplexe Abbildungen ist an das Identifikationsverfahren in [WYDM04] angelehnt. Der Prozessablauf des Identifikationsverfahrens ist in Abbildung 6.5 dargestellt.

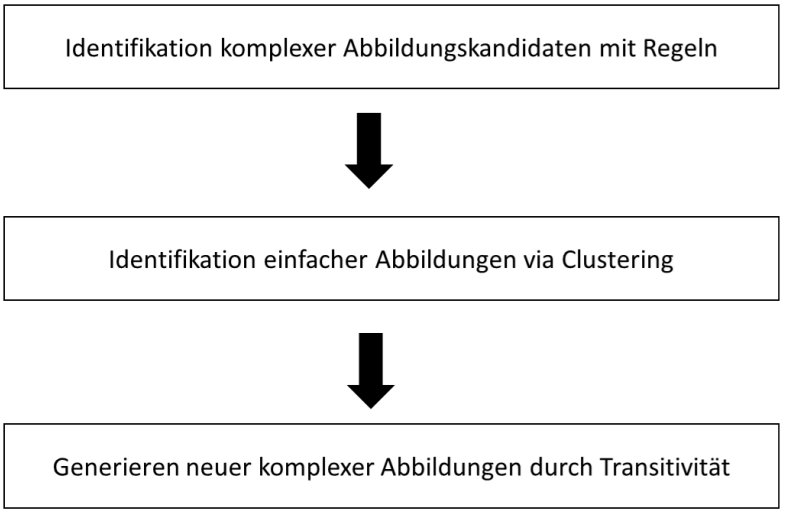



Abbildung 6.5.: Prozessablauf der Identifikation komplexer Abbildungen.

Im ersten Schritt wird eine initiale Menge von komplexen Abbildungen mithilfe eines vorher definierten Regelsatzes identifiziert. Dieser Regelsatz wird in Abschnitt 6.5 vorgestellt. Im zweiten Schritt werden dann zunächst alle Formularlemente entfernt, die in mindestens einer 1:m-Abbildung auf der 1-Seite stehen, da nur einfache Abbildungen zwischen Elementen der m-Seite für die Transitivität betrachtet werden sollen. Anschließend werden dann die übrigen Elemente mithilfe des Clusteringverfahrens aus [WYDM04] in semantisch ähnliche Cluster eingeteilt. Hierzu werden die zuvor berechneten semantischen Ähnlichkeiten verwenden. Durch das Clustering können einfache Abbildungen identifiziert werden. Die Identifikation einfacher Abbildungen wird bereits von EASIER bereitgestellt und kann verwendet werden. Diese einfachen Abbildungen können dann im letzten Schritt dazu verwendet werden, um mithilfe der zuvor identifizierten komplexen Abbildungen durch Transitivität neue komplexe Abbildungen zu generieren. Abbildung 6.6 zeigt den Identifikationsalgorithmus an einem Beispiel. Zuerst wird regelbasiert die Abbildung (rot) *Datum* auf *Tag*, *Monat* und *Jahr* identifiziert. Angenommen die regelbasierte Identifikation kann zwischen *Datum* aus Formular 1 und den Formularelementen aus Formular 3 keine hinreichende Ähnlichkeit feststellen. Anschließend wird das Formularelement *Datum* in Schritt 2 entfernt und der Clusteringalgorithmus für einfache Abbildungen angewendet. Dieser findet die drei Cluster (grün) {*Tag, Abflugtag*}, {*Monat, Abflugmonat*} und {*Jahr, Abflugjahr*}. Im dritten Schritt wird mithilfe der Transitivitätseigenschaft die Abbildung *Datum* auf *Abflugtag*, *Abflugmonat* und *Abflugjahr* hergeleitet werden. Somit findet der komplexe Identifikationsalgorithmus schließlich zwei komplexe Abbildungen (rot und blau) und gibt diese aus.

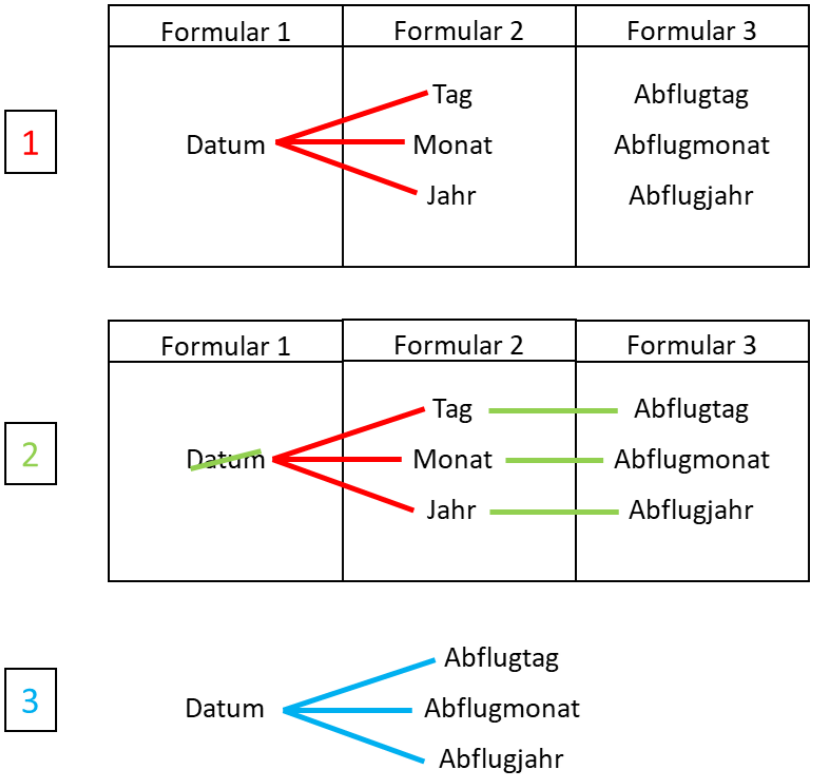



Abbildung 6.6.: Der Identifikationsalgorithmus wird auf den Formularen „Formular 1“, „Formular 2“ und „Formular 3“ ausgeführt.

Im Folgenden wird erläutert, wie die initiale Menge komplexer Abbildungen im ersten Schritt regelbasiert gefunden werden kann.

## 6.4. Identifikation semantischer Gruppen

Um komplexe Abbildungen identifizieren zu können, werden zunächst semantische Gruppen innerhalb eines Formulares gesucht. Zur Identifikation komplexer Abbildungen können dann Terme mit diesen Gruppen verglichen werden. Semantische Gruppen werden in dieser Arbeit auf zwei Arten identifiziert. Die erste Art nutzt die Präfix- und Suffixeigenschaft von semantischen Gruppen aus. Hierbei werden in jedem Formularbaum die Formularelemente zu einer semantischen Gruppe gruppiert, die entweder dasselbe Präfix oder Suffix haben. Allerdings ist es auch gelegentlich der Fall, dass semantische Gruppen nicht durch Prä- oder Suffixe gekennzeichnet sind. Hierfür muss dann ein anderer Ansatz zur Identifikation der Gruppen verwendet werden. Die Identifikation wird dann in diesem Fall mithilfe des Miningansatzes über Korrelationen aus [HCH04] realisiert. Dieser gruppiert Formularelemente, die häufig gemeinsam in einem Formular auftreten.

### 6.4.1. Präfix- und Suffixerkennung

Es wurde bereits herausgearbeitet, dass bei Formularelementen einer semantischen Gruppe oftmals dieselben Namenspräfixe oder -suffixe zur Benennung verwendet werden. Dies wird verwendet, um semantische Gruppen innerhalb eines Formulares zu identifizieren. Hierfür wird für jedes Element eines Formulares geprüft, ob es ein gemeinsames Präfix oder Suffix mit seinem Vorgänger- oder Nachfolgerelement hat. Hierbei werden die Standardpräfixe

„in“ und „input“ von Eingabeelementen entfernt. Ein Prä- und Suffix wird nur dann als gültig definiert, wenn es mindestens die Länge 3 hat. Dies stellt sicher, dass zufällige Prä- und Suffixähnlichkeiten wie beispielsweise das Präfix „de“ von „destination“ und „departure“ nicht betrachtet werden. Nachdem ein gültiges Prä- oder Suffix gefunden wurde, werden die Nachbarelemente hinzugefügt, die dieses Prä- oder Suffix mit dem betrachteten Element teilen. Anschließend werden die Nachbarn der neu hinzugefügten Elemente ebenfalls auf dieses Prä- oder Suffix überprüft. Dies wird so lange durchgeführt, bis keine direkten Nachbarelemente mehr gefunden werden können, die dieses Prä- oder Suffix enthalten. Die semantische Gruppe ist dann abgeschlossen und wird im Formular gespeichert. Da immer nur direkte Nachbarn der hinzugefügten Gruppenelemente überprüft werden, wird ausgenutzt, dass semantische Gruppen im Formular auch strukturell gruppiert sind. Dieser Identifikationsprozess wird für jedes Formular durchgeführt, sodass am Ende dieser Iteration für jedes Formular die enthaltenen semantischen Gruppen gespeichert werden können. Falls alle Formularelemente eines Formulares dieselben Prä- oder Suffixe haben, so werden keine semantischen Gruppen gebildet. Diese werden dann durch das Mining von Korrelationen identifiziert.

Die Identifikation semantischer Gruppen durch Prä- und Suffixerkennung wird mithilfe einfacher Stringvergleiche implementiert. Für jeden Term in einem Formular wird zunächst mithilfe der Nachbarelemente das längste gemeinsame Präfix, sowie das längste gemeinsame Suffix mithilfe von Charvergleichen ermittelt. Anschließend werden ausgehend vom betrachteten Element die Vorgänger und Nachfolger auf diese Prä- und Suffixe überprüft und gegebenenfalls zur Gruppe hinzugefügt. Durch die Suche semantischer Gruppen von jedem Element aus ist es möglich, dass dieselben semantischen Gruppen mehrmals identifiziert werden. Aus diesem Grund werden semantische Gruppen in einer Instanz von *java.util.HashSet* im Formular gespeichert. Durch die Eigenschaften von *java.util.HashSet* wird bei jeder Einfügeoperation überprüft, ob diese semantische Gruppe bereits im Formular enhalten ist. Dies stellt sicher, dass jede Gruppe nur einfach zum Formular hinzugefügt wird.

### 6.4.2. Korrelationsmining

Eine weitere Möglichkeit zur Identifikation semantischer Gruppen in einem Formular ist Korrelationsmining. Elemente die häufig miteinander in einem Formular auftreten sind stark positiv korreliert, während Elemente die kaum beziehungsweise nie miteinander in einem Formular stehen negativ korreliert sind. Um semantische Gruppen zu finden, werden positive Korrelationen zwischen Formularelementen gesucht. Da dieser Ansatz auf einer Häufigkeitsanalyse basiert und die Terme sehr unterschiedlich sind, werden diese zunächst zu gemeinsamen Miningobjekten verschmolzen, welche dann dem Miner als Eingabe übergeben werden können. Hierbei wird eine Abbildung zwischen den Originalelementen und dem Miningobjekt gespeichert, um die Korrelationen später wieder aufzulösen.

Das Mining wird mithilfe von Korrelationstabellen implementiert. Hierzu wird für jedes Elementpaar eine Korrelationstabelle erstellt. Diese Korrelationstabelle hält fest, wie oft dieses Elementpaar zusammen in einem Formular vorkommt ($f_{11}$), wie oft die beiden Elemente alleine vorkommen ($f_{01}$ und $f_{10}$) und wie oft sie überhaupt nicht vorkommen ($f_{00}$). Für ein + werden 0 und 1 eingesetzt und die Werte addiert. Beispielhaft gilt $f_{+1} = f_{11} + f_{01}$. Anhand zweier Korrelationsmaße können dann positive und negative Korrelationen zwischen Formularelementen gefunden werden. Bei der Konstruktion solcher Korrelationsmaße müssen einige Probleme gelöst werden. Ein Problem ist, dass selten auftretende Formularelemente tendenziell negativ zu vielen anderen Elementen korrelieren. Ein weiteres Problem ist, dass häufig auftretende Formularelemente durch ihre Häufigkeit stärker positiv korrelieren. In [HCH04] werden zwei Korrelationsmaße vorgestellt, die robust gegen

diese Probleme sind. Diese Maße sind in Gleichung 6.8 und Gleichung 6.9 aufgeführt. $F_1$ und $F_2$ sind Formularelemente, $m_n$ das negative Maß und $m_p$ das positive Maß.

$$m_n(F_1, F_2) = \frac{f_{01} * f_{10}}{f_{+1} * f_{1+}}. \tag{6.8}$$

$$m_p(F_1, F_2) = \begin{cases} 1 - m_n(F_1, F_2), & \frac{f_{11}}{f_{++}} < T_d \\ 0, & otherwise. \end{cases} \tag{6.9}$$

Hierbei ist $T_d$ ein Schwellwert für positive Korrelationen.

Zur Identifikation der semantischen Gruppen werden dann alle dieser Objekte jeweils in eine einzelne Gruppe eingefügt, sodass viele einelementige Gruppen entstehen. Hiernach werden positive Korrelationen gesucht. Während des Miningschrittes werden alle Paare von Formularelementen auf ihr Korrelationsmaß getestet. Falls ein Paar diesen Schwellwert überschreitet, so werden die beiden Elementgruppen zu einer zweielementigen Gruppe vereinigt. Anschließend können dann apriori aus den neugewonnenen zweielementigen Gruppen und den schon bestehenden einelementigen Gruppen dreielementige Gruppen gebildet werden. Dieser Schritt wird wiederholt, bis keine neuen Gruppen mehr gebildet werden können.

## 6.5. Muster zur Identifikation komplexer Abbildungen

Die Identifikation einer vorläufigen Menge an komplexen Abbildungen geschieht mithilfe einiger vordefinierter Regeln. Diese sind sehr spezifisch und werden dafür verwendet, bestimmte Abbildungstypen zu erkennen. Die resultierende Abbildungsmenge kann dann genutzt werden um mithilfe der Transitivität von Abbildungen neue komplexe Abbildungen zu finden. Diese Regeln werden mithilfe eines *RuleEnforcers* implementiert. Regeln können beim *RuleEnforcer* angemeldet werden. Der *RuleEnforcer* prüft dann für jeden Term, ob diese Regel darauf anwendbar ist. Wenn die Regel angewendet werden kann, werden anhand von der Regel definierten Kriterien passende semantische Gruppen in den Formularen gesucht. Hierfür legt jede Regel für den derzeit betrachteten Term eine Variable *confidence* an, die angibt, wie sicher sich die Regel ist, dass sie hier angewendet werden sollte. So würde beispielsweise ein `<input>`-Typ `timedate` die Sicherheit 1.0 von einer Datumserkennungsregel erhalten, da es sich hierbei mit Sicherheit um ein Datumsfeld handelt. Diese Sicherheit kann dann mit der Ähnlichkeit des betrachteten Terms mit dem identifizierten Termtupel verrechnet werden. Anschließend wird die Abbildung mit dem höchsten Wert in einem „greedy“-Verfahren ausgewählt.

### 6.5.1. Datumsabbildungen

Datumsabbildungen sind sehr häufige Abbildungen. In vielen Formularen werden Daten in vielen unterschiedlichen Darstellungsarten abgefragt. Somit ist es von großer Bedeutung Daten effizient erkennen zu können. Hierzu muss zunächst für einen Term überprüft werden, ob es sich hier um ein Datumsfeld handelt. Hierzu wird der `<input>`-Typ des Terms überprüft. Beim Typ `timedate` handelt es sich definitiv um ein Datumsfeld, während beim Typ `text` weitere Überprüfungen gemacht werden müssen. Hierbei wird der Term auf weitere Hinweise auf ein Datumsfeld untersucht. Einerseits wird in allen Eigenschaften nach dem Signalwort „date“ gesucht. Andererseits können die Eigenschaften auf das Muster „DD.MM.YYYY“ untersucht werden, indem geprüft wird, ob „DD“„MM“ und „YYYY“ darin vorkommen. Wenn ein Term als Datumsfeld identifiziert wurde, hat die Regel eine klare Definition der Abbildungspartner. Gesucht wird eine Kombination von Feldern, die ein Datum repräsentieren. Ein Tag hat immer den Wertebereich {1..31}, ein Monat

{1..12, January.. December} und ein Jahr ist immer eine 4-stellige Zahl. Mithilfe dieser Eigenschaften werden Termtupel in den semantischen Gruppen der Formulare gesucht, die diese Voraussetzungen erfüllen. Wurde ein solches Termtupel gefunden, so wurde eine neue komplexe Abbildung identifiziert.

### 6.5.2. Integer-Abbildungen

Integer-Abbildungen sind ebenfalls sehr häufig auftretende Abbildungen. In der Flugbuchungsdomäne ist dieser Abbildungstyp beispielsweise durch die Abbildung von *Passagier* auf *Erwachsener* und *Kind* gegeben. Hierbei muss zunächst geprüft werden, ob diese Regel auf den betrachteten Term anwendbar ist. Hierzu wird der Term auf Zahlen untersucht. Die `<input>`-Typen `number` oder `range` und das Formularelement `<select>` mit numerischem Wertebereich sind Indikatoren für diesen Abbildungstyp. Ob ein Element numerisch ist, kann mithilfe eines regulären Ausdruckes überprüft werden. Ist ein solcher Abbildungstyp identifiziert, so können anhand des Wertebereiches und der linguistischen Ähnlichkeit Termtupel in Formularen gesucht werden, die ähnlich zum betrachteten Term sind.

### 6.5.3. Range-Abbildungen

Range-Abbildungen definieren sich durch die Abbildung zweier Formularelemente, die ein Intervall kennzeichnen, auf ein einzelnes Formularelement, dessen Eingabe ein Repräsentant dieses Intervalls ist. Eine Beispiel für eine solche Abbildung wäre *Baujahr* auf *Baujahr von* und *Baujahr bis* bei einer Gebrauchtwagenbörse. Auch Formularelemente, die einen Preis repräsentieren, sind oft von diesem Abbildungstyp. Hierbei wird wie bei der „Integer-Abbildung“ zunächst überprüft, ob der Wertebereich numerisch ist. Wenn dies der Fall ist, werden die möglichen Termtupel der Formulare ebenfalls auf einen numerischen Wertebereich und auf Signalwörter wie „von“ oder „bis“ untersucht. Hierbei werden lediglich Termtupel gesucht, die aus zwei Termen bestehen, da Intervalle so angegeben werden.

## 6.6. Lösung komplexer Abbildungen

Die Lösung komplexer Abbildungen wird mithilfe konkreter Abbildungsvorschriften realisiert. Hierfür werden für jedes Erkennungsmuster aus Abschnitt 6.5 Konvertierungsregeln festgelegt. Eine Konvertierungsregel definiert, wie sich das komplexe Element der Abbildung in die einfachen Elemente zerlegen lässt und wie aus den einfachen Elementen das komplexe Element konstruiert werden kann.

**Beispiel: Konvertierungsregel Datum**

Die Datumserkennungsregel kann Daten anhand des Musters „DD.MM.YYYY“ erkennen. Mithilfe dieses Muster wird nun eine Konvertierungsregel erstellt. Diese Regel besagt, dass sich ein komplexes Datum durch das Zusammenfügen der einfachen Elemente „Tag“, „Monat“ und „Jahr“, jeweils mit einem Punkt getrennt, zusammensetzen lässt. Andersrum können die einfachen Elemente gebildet werden, indem das komplexe Datum an dem Trennzeichen getrennt wird. Die resultierenden Zahlen werden dann nacheinander „Tag“, „Monat“ und dann „Jahr“ zugeordnet.

## 6.7. Architektur

In diesem Abschnitt wird zusammenfassend ein Überblick über die Architektur des zu entwickelnden ComplexMatcher-Werkzeuges gegeben. Abbildung 6.7 stellt die wichtigsten Funktionseinheiten des Werkzeuges und die Abhängigkeiten zum bestehenden Rahmenwerk EASIER dar.

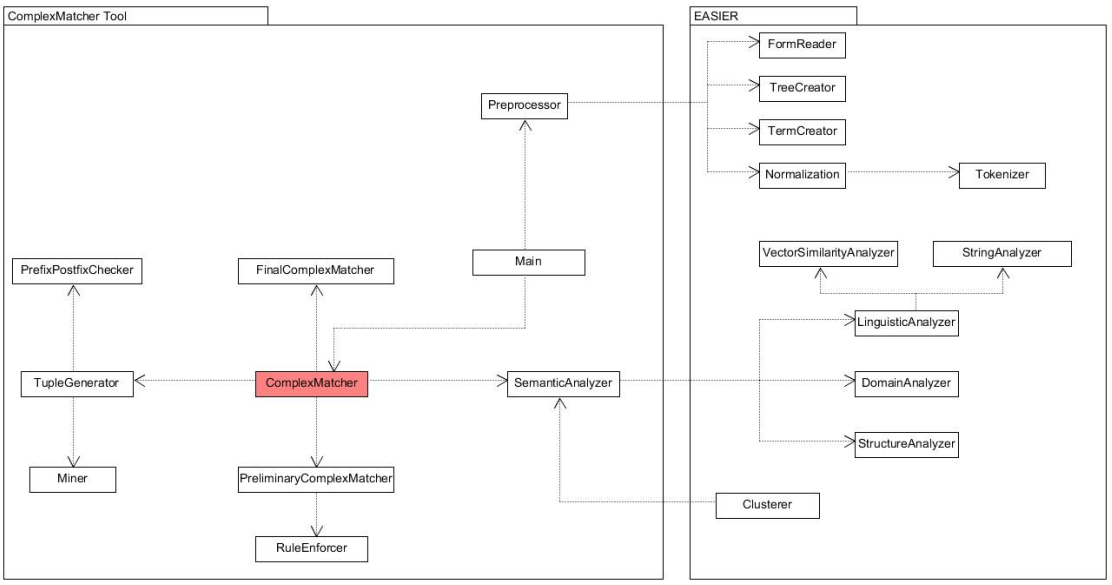



Abbildung 6.7.: Übersicht über die Funktionseinheiten des ComplexMatcher-Werkzeuges und dessen Abhängigkeiten zum Rahmenwerk EASIER. Größere Ansicht einsehbar in Abbildung B.1 im Anhang.

Die HTML-Formulare werden zunächst mit dem *Preprocessor* eingelesen. Der *FormReader* verwendet hierbei die beschriebene Bibliothek JSOUP um die HTML-Formulare einzulesen. Nachdem die Formular eingelesen wurden erstellt der *TreeCreator* die Baumstruktur, welche mit den erstellten Termen des *TermCreator*s gefüllt wird. Die generierten Terme werden dann mithilfe der *Normalization*-Klasse zunächst tokenisiert und dann normalisiert. Jeder Baumknoten speichert außerdem sowohl eine Liste der Vorgängernachbarn, als auch eine Liste der nachfolgenden Nachbarn, welche ebenfalls vom *TermCreator* erstellt werden. Bei der semantischen Analyse berechnet der *SemanticAnalyzer* mithilfe der in EASIER implementierten Klassen *LinguisticAnalyzer*, *DomainAnalyzer* und *StructureAnalyzer* jeweils die entsprechende paarweise Ähnlichkeit zwischen den Formularelementen und speichert diese. Hierbei werden die Ergebnisse der einzelnen Analysen getrennt voneinander gespeichert, sodass diese jederzeit zugänglich sind. Das Ergebnis dieser Analyse wird dann vom *Clusterer* zur Identifikation einfacher Abbildungen und vom *ComplexMatcher* zur Identifikation komplexer Abbildungen verwendet. Der *ComplexMatcher* steuert die Prozesse zur Identifikation komplexer Abbildungen. Der Prozessablauf ist wie folgt:

1. Berechne semantische Ähnlichkeiten mithilfe des *SemanticAnalyzers*.
2. Generiere semantische Gruppen mithilfe des *TupleGenerators* mit Korrelationsmining und Prä- und Suffixverarbeitung.
3. Berechne mit dem *PreliminaryComplexMatcher* eine Menge vorläufiger komplexer Abbildungen unter Anwendungen der Regeln im *RuleEnforcer*.
4. Berechne einfache Abbildungen mit dem *Clusterer*.
5. Generiere neue komplexe Abbildungen durch die Transitivität der bereits identifizierten komplexen und einfachen Abbildungen mit dem *FinalComplexMatcher*.

# 7. Evaluation

In diesem Kapitel wird das erstellte Werkzeug anhand von verschiedenen Testmengen ausgewertet. Hierfür wird das angewendete Verfahren zur Identifikation komplexer Abbildungen anhand mehrerer Metriken evaluiert. Diese werden zunächst gemeinsam mit den Testmengen vorgestellt und erläutert. Anschließend werden die Ergebnisse diskutiert.

## 7.1. Aufbau

In diesem Abschnitt wird der Testaufbau vorgestellt. Hierfür wird zunächst die Testmenge vorgestellt und anschließend die Auswertungszyklen beschrieben.

### 7.1.1. Datensatz

Evaluiert wird das Werkzeug am ICQ-Datensatz des UIUC Web Integration Repositorys [UIU03]. Getestet wird an den Kategorien „Flüge“, „Autos“ und „Bücher“. Jede Kategorie enthält 20 Formulare unterschiedlicher Dienstgeber aus dieser Kategorie. Diese Formulare wurden mithilfe von zwei Onlineverzeichnissen zusammengetragen. Für jede Dienstkategorie wurde im Rahmen dieser Arbeit manuell ein Goldstandard erstellt, welcher die einfachen und komplexen Abbildungen dieser Kategorie enthält. Anhand der Abbildungen im Goldstandard können die identifizierten Abbildungen des Werkzeuges mithilfe der Metriken in Abschnitt 7.2 evaluiert werden.

### 7.1.2. Mögliche Kombinationen von Regeln

Für die Evaluation des Identifikationsverfahrens werden alle Kombinationen der drei entworfenen Regeln getrennt voneinander betrachtet. Hierdurch können einzelne Regeln und deren Nutzen isoliert evaluiert werden. Außerdem wird überprüft, welchen Einfluss die Nutzung der Transitivität von Abbildungen auf die Ergebnismenge hat. Hierfür wird jede Regelkombination mit und ohne Transitivität ausgeführt. Tabelle 7.1 zeigt die zu evaluierenden Kombinationen. Durch das Zeichen $\times$ wird gekennzeichnet, dass diese Regel Teil der jeweiligen Kombination ist.

| Kombinationsnr. | Transitivität | Datumsregel | Integerregel | Rangeregel |
|---|---|---|---|---|
| 1 | | × | | |
| 2 | | | × | |
| 3 | | | | × |
| 4 | | × | × | |
| 5 | | × | | × |
| 6 | | | × | × |
| 7 | | × | × | × |
| 8 | × | × | | |
| 9 | × | | × | |
| 10 | × | | | × |
| 11 | × | × | × | |
| 12 | × | × | | × |
| 13 | × | | × | × |
| 14 | × | × | × | × |

Tabelle 7.1.: Übersicht über die zu evaluierenden Regelkombinationen.

Da die Güte der durch Transitivität gefundenen Abbildungen von der Güte des einfachen Clusteringverfahrens abhängt, wird dieses ebenfalls mithilfe der in Abschnitt 7.2 aufgeführten Metriken evaluiert.

## 7.2. Metriken

Zur Evaluation der Arbeit werden zunächst einige Metriken eingeführt. Die Metriken werden so definiert, dass sie auf den Ergebnisclustern des Identifikationsverfahrens für komplexe Abbildungen berechnet werden können. Hierzu werden die resultierenden Clustereinträge zunächst in die folgenden vier Mengen aufgeteilt:

**Richtig positiv (RP):**
Der Menge RP wird jeder Clustereintrag hinzugefügt, welcher vom Werkzeug identifiziert wurde und im zugehörigen Goldstandardcluster enthalten ist. Diese Menge repräsentiert die Treffer. Es wird also für jeden gefundenen Ergebniscluster des Werkzeuges überprüft, wie viele seiner Terme im Goldstandardcluster enthalten sind. Diese Terme sind die richtig positiven Terme.

**Falsch positiv (FP):**
Der Menge FP wird jeder Clustereintrag hinzugefügt, welcher vom Werkzeug identifiziert wurde, jedoch nicht im Goldstandardcluster enthalten ist. Diese Menge repräsentiert die falsch identifizierten Clustereinträge. Hierfür wird die Differenz zwischen der Anzahl der Terme im Ergebniscluster und den richtig positiven Termen gebildet.

**Richtig negativ (RN):**
Der Menge RN wird jeder Clustereintrag hinzugefügt, der weder im Goldstandard enthalten ist, noch vom Werkzeug identifiziert wurde. Diese Menge repräsentiert alle Terme, die korrekterweise nicht in diesem Ergebniscluster liegen.

**Falsch negativ (FN):**
Der Menge FN wird jeder Clustereintrag hinzugefügt, der im Goldstandard enthalten ist, jedoch nicht vom Werkzeug identifiziert wurde. Diese Menge repräsentiert die fälschlicherweise nicht identifizierten Terme des Werkzeuges. Hierfür wird die Differenz zwischen

der Anzahl der Terme im Goldstandardcluster und den richtig positiven Termen bestimmt.

Mithilfe dieser vier Klassifikationsmengen können anschließend die folgenden Metriken berechnet werden.

### 7.2.1. Präzision

Die Präzision gibt an, wie groß der Anteil richtig identifizierter Abbildungen über der Menge aller vom Werkzeug identifizierten Abbildungen ist.

$$precision = \frac{RP}{RP + FP} \tag{7.1}$$

### 7.2.2. Ausbeute

Die Ausbeute gibt den Anteil der richtig identifizierten Abbildungen über der Menge von allen korrekten Abbildungen an.

$$recall = \frac{RP}{RP + FN} \tag{7.2}$$

### 7.2.3. F1-Maß

Das F1-Maß ist das harmonische Mittel von Präzision und Ausbeute.

$$f1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{7.3}$$

### 7.2.4. Genauigkeit

Die Genauigkeit ist definiert durch den Anteil der richtig positiven Abbildungen und der richtig negativen Abbildungen über der Menge aller Abbildungen.

$$accuracy = \frac{RP + RN}{RP + FP + RN + FN} \tag{7.4}$$

### 7.2.5. Reinheit

Die Reinheit gibt an, wie groß der Anteil der richtig positiven Terme im Ergebniscluster ist [noaa].

$$purity = \frac{TP}{clustersize} \tag{7.5}$$

## 7.3. Auswertung der Ergebnisse

Zur Auswertung der Ergebnisse wurden für jede Kombination die festgelegten Metriken berechnet. Die vollständige Auswertung kann in Abschnitt A eingesehen werden. In den folgenden Abschnitten werden die wichtigsten Erkenntnisse aus diesen Auswertungen zusammengetragen und erläutert. Da die komplexen Abbildungen später in AO-Knoten integriert werden sollen, ist es wichtig, dass kaum falsche Abbildungen (FPs) gefunden werden. Aus diesem Grund ist die Präzision das wichtigste Maß bei der Evaluation komplexer Abbildungen. Dieses gibt nämlich den Anteil richtig identifizierter Abbildungen über der Menge aller gefundenen Abbildungen an.

### 7.3.1. Auswertung ohne Transitivität

Zunächst wurde der Identifikationsprozess für komplexe Abbildungen ohne Generierung neuer Abbildungen durch Transitivität evaluiert. Tabelle 7.2 zeigt die Ergebnisse der einzelnen Regeln und die beste Kombination der Regeln.

| Kategorie | Nr. | Präzision | Ausbeute | F1-Maß | Genauigkeit | Reinheit |
|---|---|---|---|---|---|---|
| Flüge | 1 | 52,94 | 15,00 | 23,38 | 74,46 | 52,94 |
| | 2 | 37,01 | 13,91 | 20,22 | 93,29 | 16,04 |
| | 3 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 39,37 | 14,79 | 21,50 | 93,33 | 17,06 |
| Autos | 1 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 22,22 | 8,33 | 12,12 | 85,64 | 4,65 |
| | 3 | 0 | 0 | 0 | 0 | 0 |
| Bücher | 1 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 30,00 | 12,62 | 17,77 | 94,37 | 13,93 |
| | 3 | 50,00 | 25,00 | 33,33 | 88,57 | 10,00 |
| | 6 | 30,30 | 12,78 | 18,00 | 94,71 | 13,79 |

Tabelle 7.2.: Auswertung der einzelnen Regeln und der besten Kombination.

Die Datumsregel erzielt mit 52,94% die höchste Präzision. Die Rangeregel findet keine Abbildungen, was jedoch zu erwarten war, da die Kategorie „Flüge“ keine komplexen Abbildungen enthält, welche von dieser Regel erkannt werden kann. Derselbe Fall tritt bei der Datumsregel in den Kategorien „Autos“ und „Bücher“ auf. Anhand der Präzision und Reinheit ist erkennbar, alle Regeln unter einer hohen Anzahl an FPs leiden. Die Präzision kann jedoch erhöht werden, indem der Schwellwert für die Ermittlung der Ähnlichkeit der Abbildungspartner erhöht wird. Hierfür muss dann allerdings Ausbeute und Genauigkeit eingebüßt werden, da nun zwar weniger FPs gefunden werden, jedoch durch den erhöhten Schwellwert unter Umständen auch weniger RPs identifiziert werden.

### 7.3.2. Auswertung mit Transitivität

Die Güte der durch die Transitivität identifizierten Abbildungen hängt stark von der Güte des Clusteringverfahrens zur Identifikation einfacher Abbildungen ab. Tabelle 7.3 zeigt die Metriken für das einfache Clusterverfahren. Das Clustering wurde mit einem Schwellwert von 0.3 für die Clusterähnlichkeit durchgeführt.

| Kategorie | Präzision | Ausbeute | F1-Maß | Genauigkeit |
|---|---|---|---|---|
| Flüge | 86,27 | 17,78 | 29,48 | 97,11 |
| Autos | 98,63 | 31,03 | 47,21 | 95,48 |
| Bücher | 100,00 | 37,59 | 54,64 | 97,48 |

Tabelle 7.3.: Auswertung der durch Clustering identifizierten einfachen Abbildungen.

Das einfache Clusterverfahren erzielt eine recht niedrige Ausbeute, aber dafür sehr hohe Präzision und Genauigkeit. Das bedeutet, dass zwar nicht alle korrekten Abbildungen gefunden werden können (Ausbeute), jedoch die Abbildungen, die gefunden werden überwiegend korrekt sind (Präzision und Genauigkeit). Dies ist eine gute Voraussetzung für die Transitivität.

Tabelle 7.4 stellt für jede der drei Kategorien die Ergebnisse mit Einbeziehung der Transitivität dar.

| Kategorie | Nr. | Präzision | Ausbeute | F1-Maß | Genauigkeit | Reinheit |
|---|---|---|---|---|---|---|
| Flüge | 8 | 40,91 | 15,00 | 21,95 | 75,48 | 40,91 |
| | 9 | 38,78 | 16,86 | 23,51 | 93,50 | 18,21 |
| | 10 | 0 | 0 | 0 | 0 | 0 |
| | 11 | 39,74 | 17,75 | 24,54 | 93,55 | 18,93 |
| Autos | 8 | 0 | 0 | 0 | 0 | 0 |
| | 9 | 12,12 | 8,33 | 9,88 | 87,83 | 2,96 |
| | 10 | 0 | 0 | 0 | 0 | 0 |
| Bücher | 8 | 0 | 0 | 0 | 0 | 0 |
| | 9 | 30,00 | 12,62 | 17,77 | 94,37 | 13,93 |
| | 10 | 50,00 | 25,00 | 33,33 | 88,57 | 10,00 |
| | 13 | 30,30 | 12,78 | 18,00 | 94,71 | 13,79 |

Tabelle 7.4.: Auswertung der einzelnen Regeln und der besten Kombination.

Trotz der guten Ergebnisse des einfachen Clusterverfahrens haben sich die Ergebnisse des komplexen Identifikationsverfahrens überwiegend verschlechtert statt verbessert. Dies liegt an der sehr hohen Anzahl an identifizierten FPs. Wenn schlechte Abbildungen gefunden werden, so liefert die Transitivität, trotz guter einfacher Abbildungen, ebenso schlechte Abbildungen.

## 7.4. Diskussion der Ergebnisse

Die Testmengen erzielten jeweils stark unterschiedliche Ergebnisse. Aus den Auswertungen geht hervor, dass nicht jede Regel in allen Kategorien sinnvoll anwendbar ist. Dies liegt unter anderem daran, dass der durch die Regel definierte Abbildungstyp in dieser Kategorie nicht präsent ist. Die niedrige Präzision in den Ergebnissen zeigt, dass die entworfenen Regeln sehr anfällig für FPs sind. Hierdurch verschlechtert sich die Präzision drastisch. Dennoch wurden einige sehr gute Abbildungen vom Identifikationsalgorithmus erkannt. Zukünftig sollte daher darauf geachtet werden, die vorhandenen Regeln zu präzisieren, sodass sie weniger FPs generieren. Außerdem sollten weitere Regeln hinzugefügt werden, um mehr Abbildungstypen vom Regelsatz abdecken zu können. An dem Ergebnis sieht man auch, dass es entsprechend schwierig ist, Regeln zu entwerfen, die nicht nur einen Anwendungsfall eines Abbildungstyps, sondern den gesamten Abbildungstyp abdecken können. Eine weitere Auffälligkeit ist, dass grundsätzlich die Abbildungen auf dem Datensatz sehr schwach sind. Dies zeigt sich beispielsweise durch den Ähnlichkeitsschwellwert von 0.3 beim einfachen Clusterverfahren. Bereits bei einem Schwellwert von 0.5 werden fast keine Abbildungen mehr gefunden. Dies kann daran liegen, dass bei einem automatischen Extraktionsverfahren für HTML-Formulare einem Formularelement nicht alle zugehörigen semantischen Informationen zugeordnet werden können, wodurch die Formularelemente schlechter semantisch vergleichbar sind.

# 8. Zusammenfassung und Ausblick

Das Erstellen von aktiven Ontologien ist ein sehr mühsamer manueller Prozess. Durch das Projekt EASIER ist es möglich, aktive Ontologien für formularbasierte Internetdienste automatisiert zu generieren. Hierfür müssen Formularelemente auf AO-Knoten abgebildet werden. Ein Teilproblem dieses Automatisierungsprozesses ist die Abbildung komplexer Beziehungen. Diese Arbeit stellt einen Identifikationsprozess für komplexe Abbildungen vor, wodurch nun auch 1:m-Beziehungen zwischen Formularelementen erkannt werden können.

## 8.1. Zusammenfassung

Zur Identifikation komplexer Abbildungen wurde zunächst ein Extraktionsverfahren für HTML-Formulare entwickelt, wodurch Formularelemente strukturerhaltend in eine Baumstruktur überführt werden können. Hierbei wurden für die Identifikation der Abbildungen irrelevanten Formularelemente herausgefiltert. Die Elemente in der Baumstruktur wurden anschließend so normalisiert, dass sie untereinander vergleichbar sind. Im nächsten Schritt wurde eine semantische Analyse der Formularelemente durchgeführt. Hierbei wurden alle Paare von Elementen unterschiedlicher Formulare auf linguistische und strukturelle Ähnlichkeiten sowie auf Ähnlichkeiten der entsprechenden Wertebereiche geprüft. Im Identifikationsschritt konnte dann anhand von entwickelten Regeln und dem Ergebnis der semantischen Analyse eine initiale Menge an Abbildungen identifiziert werden. Aus dieser Menge komplexer Abbildungen und den einfachen Abbildungen, die durch das bereits in EASIER implementierte Clusterverfahren gefunden wurden, konnten dann mithilfe der Transitivitätseigenschaft neue komplexe Abbildungen generiert werden. Anschließend wurde für Abbildungstypen, die häufig identifiziert wurden, eine manuelle Mustererkennung durchgeführt. Hierdurch konnten Abbildungsvorschriften konstruiert werden, die sowohl das Umwandeln der Abbildungspartner ineinander ermöglichen, als auch die Identifikation komplexer Abbildungen durch Verwendung der Muster in den Regeln verbessert.

Die Ergebnisse der Evaluation haben gezeigt, dass die Identifikation komplexer Abbildungen eine deutlich höhere Komplexität als das Finden einfacher Abbildungen hat. Außerdem hat sich gezeigt, dass die entworfenen Regeln weiter präzisiert werden müssen, um weniger FPs zu generieren.

## 8.2. Ausblick: Verbesserung Datenextraktion

Sehr maßgeblich für die Ergebnisse des Identifikationsverfahrens ist die Güte der automatischen Extraktion. Je mehr semantische Informationen aus den HTML-Formularen extrahiert werden können, umso besser kann das System abwägen, ob Formularelemente ähnlich sind. Zukünftige Verbesserungen am automatischen Extraktionsverfahren wie beispielsweise eine verbesserte Zuordnung der Bezeichner-Formularelementpaare können die Güte der Identifikation komplexer Abbildungen qualitativ deutlich steigern.

## 8.3. Ausblick: Regelsatzerweiterung und -verbesserung

Zur Identifikation der initialen Menge komplexer Abbildungen werden derzeit festgelegte Regeln verwendet. Diese Regeln definieren Muster, die zur Identifikation einer passenden semantischen Gruppe für das derzeit betrachtete Element verwendet werden. Durch die Erweiterung des bisher bestehenden Regelsatzes durch verallgemeinerte Regeln können weitere komplexe Abbildungstypen identifiziert werden. Eine mögliche Herausforderung der Regelsatzerweiterung könnte die automatische Generierung solcher Regeln sein. Des Weiteren sollte darauf geachtet werden, dass die vorhandenen Regeln so zu präzisieren, dass sie weniger FPs generieren.

## 8.4. Ausblick: Häufigkeitsanalyse

Zur Identifikation semantischer Gruppen wurde bereits eine Häufigkeitsanalyse verwendet. Diese könnte so erweitert werden, dass sie nicht nur positiv korrelierende Elemente identifizieren kann, sondern auch negative. Dies würde dazu führen, dass Synonymbeziehungen zwischen Elementen oder semantischen Gruppen, anhand der Annahme, dass Synonyme stark negativ korrelieren, gefunden werden können. Das Untersuchen negativer Korrelationen zwischen semantischen Gruppen könnte sogar die Identifikation komplexer n:m-Beziehungen ermöglichen.

## 8.5. Ausblick: Wissensontologien

Ein Problem des Identifikationsprozesses ist, wenn zwischen den Elementen einer korrekten Abbildung keine linguistischen Ähnlichkeiten identifiziert werden können und kein Wertebereich vorhanden ist wie beispielsweise beim Eingabetyp `<input type="text"`. Dieser Fall tritt beispielsweise häufiger bei der Abbildung von „Autor“ auf „Vornamen“ und „Nachname“ oder bei der Abbildung von „Geburtsdatum“ auf „Tag“, „Monat“ und „Jahr“ auf. Hierfür könnten Wissensontologien verwendet werden. Eine Wissensontologie könnte speichern, dass sich ein Datum aus einem Tag, einem Monat und einem Jahr zusammensetzt und dass ein Autor einen Vornamen und einen Nachnamen hat. Dies würde die Erkennung solcher bislang schwer zu identifizierenden Abbildungen verbessern.

# Literaturverzeichnis


[AHS12] An, Yuan ; Hu, Xiaohua ; Song, Il-Yeol: Learning to discover complex mappings from web forms to ontologies, ACM, Oktober 2012. – ISBN 978–1–4503–1156–4, 1253–1262 (zitiert auf den Seiten 17 und 40).

[BL16] Blersch, Martin ; Landhaeusser, Mathias: Easier: An Approach to Automatically Generate Active Ontologies for Intelligent Assistants. In: *The 20th World MultiConference on Systemics, Cybernetics and Informatics (WMSCI 2016) Orlando, FL, USA 05.07.2016*, 2016 (zitiert auf den Seiten v, ix, 19 und 20).

[Dic45] Dice, Lee R.: Measures of the Amount of Ecologic Association Between Species. In: *Ecology* 26 (1945), Juli, Nr. 3, 297–302. `http://dx.doi.org/10.2307/1932409`. – DOI 10.2307/1932409. – ISSN 1939–9170 (zitiert auf Seite 45).

[DR02] Do, Hong-Hai ; Rahm, Erhard: COMA: A System for Flexible Combination of Schema Matching Approaches. In: *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB Endowment, 2002 (VLDB '02), 610–621 (zitiert auf Seite 45).

[Guz08] Guzzoni, Didier: Active: a unified platform for building intelligent applications. (2008). `http://dx.doi.org/10.5075/epfl-thesis-3990, urn:nbn:ch:bel-epfl-thesis3990-1`. – DOI 10.5075/epfl–thesis–3990, urn:nbn:ch:bel–epfl–thesis3990–1 (zitiert auf den Seiten ix, 1, 10, 11 und 19).

[HCH04] He, Bin ; Chang, Kevin Chen-Chuan ; Han, Jiawei: Discovering Complex Matchings Across Web Query Interfaces: A Correlation Mining Approach. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA : ACM, 2004 (KDD '04). – ISBN 978–1–58113–888–7, 148–157 (zitiert auf den Seiten 16, 34, 48 und 49).

[HMYW03] He, Hai ; Meng, Weiyi ; Yu, Clement ; Wu, Zonghuan: Wise-integrator: An Automatic Integrator of Web Search Interfaces for E-commerce. In: *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*. Berlin, Germany : VLDB Endowment, 2003 (VLDB '03). – ISBN 978–0–12–722442–8, 357–368 (zitiert auf den Seiten 15, 26, 28, 30, 35 und 42).

[MFBB10] Maiz, Nora ; Fahad, Muhammad ; Boussaid, Omar ; Bentayeb, Fadila: Automatic ontology merging by hierarchical clustering and inference mechanisms. In: *Proceedings of I-KNOW*, 2010 (zitiert auf Seite 18).

[MVM09] Miller, Frederic P. ; Vandome, Agnes F. ; McBrewster, John: *Levenshtein Distance: Information Theory, Computer Science, String (Computer*

*Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press, 2009. – ISBN 6130216904, 9786130216900 (zitiert auf Seite 12).

[noaa] *Evaluation of clustering*. `https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html` (zitiert auf Seite 55).

[noab] *HTML 4.01 Specification*. `https://www.w3.org/TR/html4/` (zitiert auf Seite 5).

[noac] *HTML 5.2*. `https://www.w3.org/TR/html5/` (zitiert auf Seite 5).

[noad] *jsoup Java HTML Parser, with best of DOM, CSS, and jquery*. `https://jsoup.org/` (zitiert auf Seite 42).

[UIU03] *The UIUC Web Integration Repository*. Computer Science Department, University of Illinois at Urbana-Champaign. http://metaquerier.cs.uiuc.edu/repository, 2003 (zitiert auf Seite 53).

[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*. New York, NY, USA : ACM, 2004 (SIGMOD '04). – ISBN 978–1–58113–859–7, 95–106 (zitiert auf den Seiten ix, x, 2, 13, 14, 26, 28, 29, 30, 31, 32, 33, 34, 35, 40, 42, 43, 44, 45, 46 und 47).

[ZHC04] Zhang, Zhen ; He, Bin ; Chang, Kevin Chen-Chuan: Understanding Web Query Interfaces: Best-effort Parsing with Hidden Syntax. In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, 2004 (SIGMOD '04). – ISBN 978–1–58113–859–7, 107–118 (zitiert auf den Seiten 14 und 16).

# Anhang

## A. Evaluationsergebnisse

### A.1. Flüge

| Kombinationsnr. | Präzision | Ausbeute | F1-Maß | Genauigkeit | Reinheit |
|---|---|---|---|---|---|
| 1 | 52,94 | 15,00 | 23,38 | 74,46 | 52,94 |
| 2 | 37,01 | 13,91 | 20,22 | 93,29 | 16,04 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 39,37 | 14,79 | 21,50 | 93,33 | 17,06 |
| 5 | 52,94 | 15,00 | 23,38 | 74,46 | 52,94 |
| 6 | 37,01 | 13,91 | 20,22 | 93,29 | 16,04 |
| 7 | 39,37 | 14,79 | 21,51 | 93,33 | 17,06 |
| 8 | 40,91 | 15,00 | 21,95 | 75,48 | 40,91 |
| 9 | 38,78 | 16,86 | 23,51 | 93,50 | 18,21 |
| 10 | 0 | 0 | 0 | 0 | 0 |
| 11 | 39,74 | 17,75 | 24,54 | 93,55 | 18,93 |
| 12 | 40,91 | 15,00 | 21,95 | 75,48 | 40,91 |
| 13 | 38,78 | 16,86 | 23,51 | 93,50 | 18,21 |
| 14 | 39,73 | 17,75 | 24,54 | 93,55 | 18,93 |

Tabelle A.1.: Ergebnisse der Kategorie „Flüge“ in %.

## A.2. Autos

| Kombinationsnr. | Präzision | Ausbeute | F1-Maß | Genauigkeit | Reinheit |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 22,22 | 8,33 | 12,12 | 85,64 | 4,65 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 22,22 | 8,33 | 12,12 | 85,64 | 4,65 |
| 5 | 0 | 0 | 0 | 0 | 0 |
| 6 | 22,22 | 8,33 | 12,12 | 85,64 | 4,65 |
| 7 | 22,22 | 8,33 | 12,12 | 85,64 | 4,65 |
| 8 | 0 | 0 | 0 | 0 | 0 |
| 9 | 12,12 | 8,33 | 9,88 | 87,83 | 2,96 |
| 10 | 0 | 0 | 0 | 0 | 0 |
| 11 | 12,12 | 8,33 | 9,88 | 87,83 | 2,96 |
| 12 | 0 | 0 | 0 | 0 | 0 |
| 13 | 12,12 | 8,33 | 9,88 | 87,83 | 2,96 |
| 14 | 12,12 | 8,33 | 9,88 | 87,83 | 2,96 |

Tabelle A.2.: Ergebnisse der Kategorie „Autos“ in %.

## A.3. Bücher

| Kombinationsnr. | Präzision | Ausbeute | F1-Maß | Genauigkeit | Reinheit |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 30,00 | 12,62 | 17,77 | 94,37 | 13,93 |
| 3 | 50,00 | 25,00 | 33,33 | 88,57 | 10,00 |
| 4 | 30,00 | 12,62 | 17,77 | 94,37 | 13,93 |
| 5 | 50,00 | 25,00 | 33,33 | 88,57 | 10,00 |
| 6 | 30,30 | 12,78 | 18,00 | 94,71 | 13,79 |
| 7 | 30,30 | 12,78 | 18,00 | 94,71 | 13,79 |
| 8 | 0 | 0 | 0 | 0 | 0 |
| 9 | 30,00 | 12,62 | 17,77 | 94,37 | 13,93 |
| 10 | 50,00 | 25,00 | 33,33 | 92,45 | 3,57 |
| 11 | 30,00 | 12,62 | 17,77 | 94,37 | 13,93 |
| 12 | 50,00 | 25,00 | 33,33 | 92,45 | 3,57 |
| 13 | 30,30 | 12,78 | 18,00 | 94,71 | 13,79 |
| 14 | 30,30 | 12,78 | 18,00 | 94,71 | 13,79 |

Tabelle A.3.: Ergebnisse der Kategorie „Bücher“ in %.

# B. Architektur

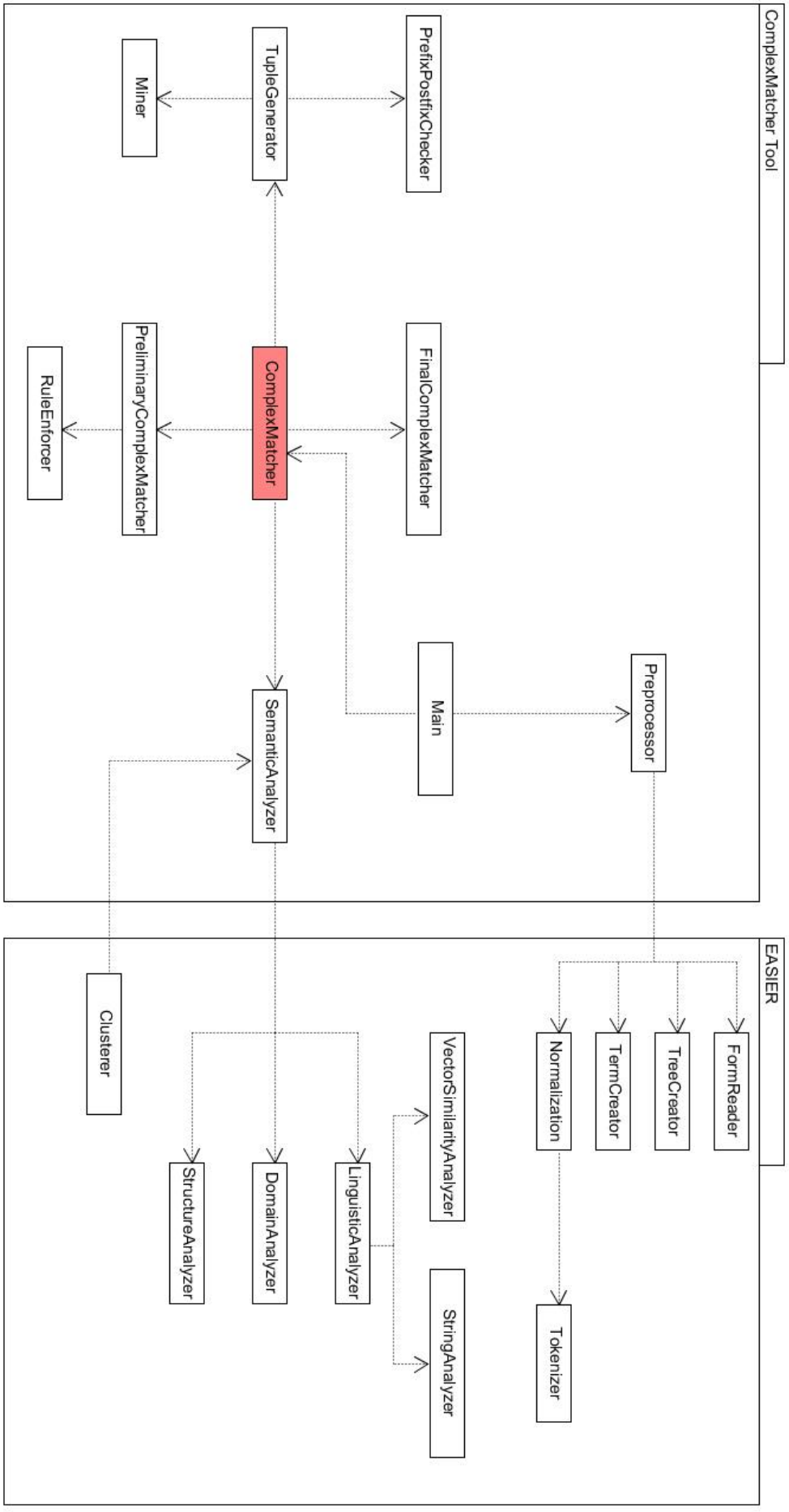



Abbildung B.1.: Architektur des erstellten Werkzeuges.