Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Masterarbeit

Ontologiemetriken zur Datenqualitätsverbesserung

Leipzig, Dezember 2013

vorgelegt von
Cherix, Didier
Studiengang Informatik

Betreuender Hochschullehrer:

Prof. Dr. Klaus-Peter Fähnrich
Fakultät für Mathematik und Informatik
Betriebliche Informationssysteme

Zusammenfassung

Die Datenqualität ist ein weitreichendes Thema. Bei vielen Anwendungen und Verfahren spielt sie eine große Rolle. *Semantic Web* ist da keine Ausnahme. Die Vollständigkeit, Fehlerfreiheit und Genauigkeit der Daten ist maßgebend für die Qualität des Ergebnisses. Im *Semantic Web* sind Ontologien die wichtigsten Datenquellen. Deswegen ist es wesentlich, diese auf ihre Datenqualität untersuchen zu können. In dieser Arbeit stellen wir ein Verfahren vor, um die Datenqualität einer Ontologie zu überprüfen und potentielle Fehler zu erkennen. Als erstes zeigen wir, wie aus einer Startmenge an fehlerhaften Daten (Goldstandard) andere Fehlerquellen gefunden werden können. Mit Hilfe von Clustern erweitern wir einen Goldstandard, um neue Fehler zu finden. Mit Hilfe dieser Verfahren konnten fehlerhafte Daten in DBpedia wiedergefunden werden.
Da ein solcher Goldstandard nicht immer existiert, zeigen wir Methoden, um Fehlerquellen ohne ihn zu finden. Die verschiedenen Verfahren liefern eine Menge an potentiell fehlerhaften Daten. Diese Daten sollen per Hand evaluiert werden und daraus die nötigen Regeln oder Tests abgeleitet werden. Mit diesen Verfahren konnte ein hoher Recall an fehlerhafte Daten erzielt werden. Außerdem zeigen wir Fälle, die von anderen Verfahren unter anderem Databugger [41], nicht erkannt werden.

Danksagung

Ich bedanke mich an dieser Stelle bei Herrn Ricardo Usbeck für seine Unterstützung zu fachlichen Inhalten. Weiterhin gilt mein Dank Herrn Dr. rer. nat. Andreas Both für die Möglichkeit mir die technischen Mitteln innerhalb der Forschungsabteilung der Unister GmbH zur Verfügung zu stellen. Schließlich bedanke ich mich bei Frau Dr. med. Stephanie Walther für die orthographische und grammatikalische Korrekturen.

Inhaltsverzeichnis

1 Einleitung

Die wachsende Menge an Web-Daten erzeugt neue Probleme für den Nutzer. Die erste Schwierigkeit ist bereits die Wahl der richtigen Quelle: Welche der zahlreichen Quellen ist angepasst für den anvisierten Zweck? Wie sehen die Daten in dieser Quelle aus? Wie können Fakten abgefragt werden? Woher stammen die Informationen? Sind diese richtig, vollständig und vertrauenswürdig? Das Finden der Antworten auf all diese Fragen wird durch die Menge an Daten erschwert.

Das klassische Web von Dokumenten (Web of Document) ermöglicht nicht Informationen direkt abzufragen. Um an Informationen zu gelangen, ist es notwendig die einzelne Dokumente durchzugehen. Mit Hilfe von Suchmaschinen ist es möglich, die Anzahl an Dokumenten, die gelesen werden sollen, zu reduzieren. Sie kann jedoch nicht Fakten beantworten oder gewährleisten, dass die empfohlenen Dokumente die gesuchte Information enthält. Aus diesem Grund wurde der *Semantic Web*[1] oder *Web of Data* definiert.

Die Strukturierung der Daten in einem *Web of Data* soll dabei helfen. Aus den in diesem *Web of Data* enthaltenen Informationen können Ableitungen, Folgerungen und andere induktive sowie deduktive Verfahren entstehen. Die richtigen Schlussfolgerungen können allerdings nur gezogen werden, wenn die zugrundeliegenden Daten korrekt sind. Das Prinzip des *Web of Data* sieht vor, dass neue Daten zu bereits existierenden verlinkt werden können. Die Gesamtqualität hängt somit auch von der Qualität der einzelnen Datenquellen ab.

Die verschiedenen Quellen des *Web of Data* sollen untereinander verknüpft sein, um ein Netz (Web) von Daten zu erzeugen. Diese Sammlung an zusammenhängenden Datenquellen wird *Linked Data* [10] genannnt. Jeder der Datenquellen verfügt über eine Wissendatenbank (Ontologie), die Konzepte und die zugehörigen konkreten Fakten enthält.

Aus diesen Gründen ist es wichtig, die Datenqualität einer Ontologie messen zu können. Die meisten Ontologien werden per Hand überprüft. Nur durch unerwartete Ergebnisse

[1] `http://www.w3.org/standards/semanticweb/`

mancher Anwendungen werden per Zufall Fehler aufgedeckt. Es ist jedoch unmöglich, eine Ontologie komplett per Hand zu untersuchen. Die aus Wikipedia[2] extrahierte Ontologie DBpedia[11] enthält 4 Millionen Fakten[3] in englischer Sprache. Selbst in Fallen kleinerer Ontologien müssten alle Daten durchgearbeitet werden. Anschließend müsste entschieden werden, ob diese korrekt sind. Dies wäre nur mit einem allwissenden Experten möglich, der die Korrektheit jeder Aussage eindeutig evaluieren könnte.

Viele Informationen sind nicht statisch. Fakten ändern sich oder es kommen neue hinzu. In solchen Fällen ist es notwendig, eine Ontologie zu aktualisieren. Außerdem müssen fehlerhafte Daten bei einer Aktualisierung so früh wie möglich entdeckt werden, um Konsistenzverluste zu vermeiden.

Auf der Suche nach fehlerhaften Daten können die Technologien des *Information Retrievals* und des maschinellen Lernens (*Machine Learning*) verwendet werden. Unter Erstere, versteht man die Technologien, um Informationen zu suchen. *Machine Learning* beschreibt die Verfahren, um Daten zu gruppieren und automatische Analysen durchzuführen. Statistische Verfahren sollen helfen, die Quellen an potentiellen Fehlern zu finden.

Ein Verfahren, das aus den anfänglich per Hand gefundenen Fehlern potentiell neue entdeckt, soll im Rahmen dieser Arbeit entwickelt werden. Diese potentiellen Fehler sollen dann gezielt von Datenexperten überprüft werden.

[2] `http://wikipedia.org`
[3] `http://blog.dbpedia.org/category/dataset-releases/`

2 Grundlagen

In diesem Kapitel werden die für diese Arbeit nötigen Grundlagen vorgestellt. In den nachfolgenden Abschnitten widmen wir uns als erstes dem Kernthema: Semantic Web.

2.1 Semantisches Web (Semantic Web)

Das semantische Web besteht, wie der Name bereits vermuten lässt, aus zwei Teilen: Semantik und Web. Während der Begriff Web jedem bekannt sein sollte, ist es komplizierter sich etwas unter dem Begriff Semantik vorzustellen. Jedoch um ein einheitliches Verständnis der Begriffe gewährleisten zu können, beschäftigen wir uns erst mit dem Web.

2.1.1 Web

Das World Wide Web kurz Web, entstand in den 90er Jahren im Forschungszentrum des CERN (Centre Europeen de Recherche Nucleaire) in Genf. Tim Berners-Lee [4] hatte die Idee, Textdateien mit einer Auszeichnungssprache HTMl (Hyper Text Markup Language) zu versehen. Das Ziel war, seine Entwicklungen dokumentieren zu können und somit seinen Forscherkollegen verständlich zu machen. Schnell wurde klar, dass diese Idee ihre Vielfältigkeit am besten über das Internet nutzbar machen würde. In dieser Zeit entstand auch der Name „World Wide Web“.

Das Web ist somit eine Vernetzung von elektronisch abrufbaren HTML-Dokumenten sogenannten Webseiten.

2.1.2 Semantik

Semantik (aus dem Grieschichen $\sigma\epsilon\mu\alpha\iota\nu\epsilon\nu$ bezeichnen)

[4] `http://www.w3.org/People/Berners-Lee/`

1. Teilgebiet der Linguistik, das sich mit den Bedeutungen sprachlicher Zeichen und Zeichenfolgen befasst
2. Bedeutung, Inhalt (eines Wortes, Satzes oder Textes)

[1]

Semantik steht allgemein für „Bedeutung von Wörtern, Phrasen oder Symbolen." In der Linguistik steht Semantik für die Wissenschaft der Bedeutung in verschiedenen Sprachformen. In der Informatik, und insbesondere im Bereich Semantic Web, versteht man unter Semantik die Bedeutung von Wörtern bzw. Zeichen(-ketten) und ihre Beziehungen untereinander. [34]

Das semantische Web ist ein Konzept, um die sich im World Wide Web befindlichen Informationen in von Computern verständliche Daten zu transformieren und miteinander zu vernetzen. Um diese Vision von Tim Berners-Lee zu realisieren, sind mehrere Technologien nötig. Diese werden in den folgenden Abschnitten vorgestellt.

2.1.3 Ontologie

Die maschinelle Verarbeitung von Daten setzt voraus, dass diese von allen teilnehmenden Prozessen in gleicher Weise verstanden werden. „ Die Ontologie (vom griech. ov: sein und $\lambda o\gamma o\varsigma$ Lehre) ist eine Studie über das, was es gibt, eine Bestandsaufnahme dessen, was existiert. Eine ontologische Zusage ist eine Zusage zu einer Existenzhypothese." [61] In der Informatik existieren mehrere Definitionen: „Eine Ontologie ist eine explizite Spezifikation einer Konzeptualisierung." [29] Eine Ontologie ist ein von mehreren Akteuren geteiltes Verständnis zu einigen Themen [57]. Sie benötigt ein abstraktes Modell. Dieses beschreibt die relevanten Begriffe und die Beziehungen dieser zueinander. In einer Ontologie erkennt man folgende Elemente:

Ressourcen ist der Oberbegriff für alle Elemente einer Ontologie.

Klassen , auch Konzepte genannt, beschreiben die verschiedenen Kategorien. Diese Kategorien sind hierarchisch organisiert.

Instanz ist ein konkretes Objekt, das einer bestimmten Klasse zugeordnet wird.

Relationen (Properties) beschreiben die Abhängigkeiten von Klassen und Instanzen zueinander. Die Relation `ist-ein` definiert die Zugehörigkeit einer Instanz zu ihrer(n) Klasse(n).

Um die Elemente einer Ontologie logisch miteinander verknüpfen zu können, wird die Beschreibungslogik verwendet.

2.1.4 Beschreibungslogik

Die Beschreibungslogiken sind eine Untermenge der Prädikatenlogik erster Stufe. Die meisten Beschreibungslogiken sind entscheidbar. Formal unterteilt man sie in eine *terminological Box (TBox)* und eine *assertional box (ABox)*. Die TBox enthält die Klassen und deren Relationen untereinander. In der ABox befinden sich die Instanzen und ihre Relationen. Eine Ontolgie besteht genau aus diesen beiden Boxen [6].

Um eine Ontolgie definieren zu können, ist eine formale Sprache notwendig. Aus diesem Grund stellen wir im nächsten Kapitel die vom W3C empfohlene Grundsprache RDF vor.

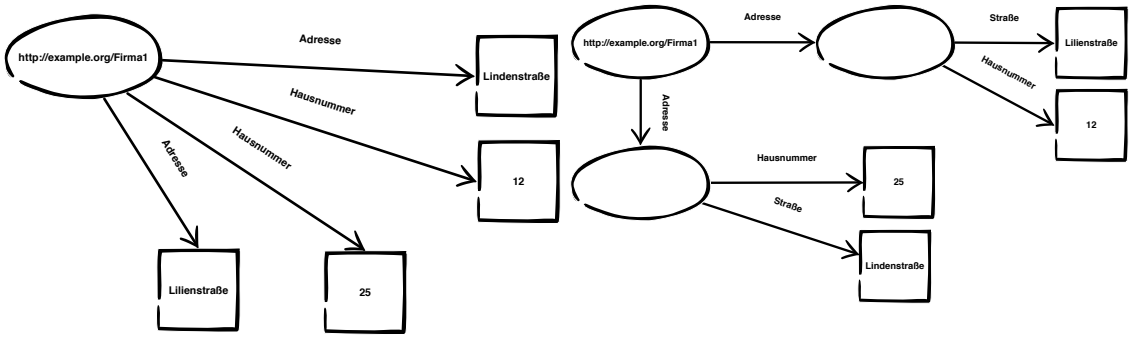
2.1.5 Resource Description Framework (RDF) [8]

Das *Resource Description Framework* ist ein standardisiertes Modell zum Austausch von Informationen auf dem Web. Mittels RDF-Dokumenten werden Graphen beschrieben. Diese wiederum enthalten Knoten, die mittels gerichteten Kanten verbunden werden können. Als Hauptbestandteil dieser Graphen werden Uniform Resource Identifier (URI) verwendet. URI sind eine verallgemeinerte Form der Uniform Resource Locator (URL). Eine URI identifiziert genau eine Resource im Web. Somit ist es möglich beispielweise zwischen den beiden französischen Schriftstellern Alexandre Dumas Vater und Sohn zu unterscheiden.

Im RDF werden Datenwerte „Literale“ genannt. Da sie keine URI besitzen, unterscheiden sie sich von Instanzen, Relationen und Klassen. Literale sind allerdings nicht die einzigen Ressourcen, die keine URI besitzen, sie stellen aber konkrete Werte dar, wie Zahlen, Daten, ... Einem Literal ist ein bestimmter Datentyp zugeordnet. Dieser Datentyp ermöglicht, zu unterscheiden, ob dieses Literal ein Integer, Double, Datum, Zeichenkette etc. darstellt.

Ein RDF-Graph wird mittels Tripeln beschrieben. Jedes Tripel besitzt ein Subjekt, ein Prädikat und ein Objekt.

Exemplarisch besteht eine Adresse aus einem Straßennamen und einer Hausnummer. Mit den bisher vorgestellten Elementen müsste die Adresse einer Person mit zwei Relationen beschrieben werden. A wohnt in der Lindenstraße. A wohnt im Haus 34. Wenn



(a) Ohne Blank Nodes **(b)** Mit Blank Nodes

Abbildung 2.1: Beispiel Modellierung einer Firma mit mehreren Adressen

es sich aber, statt um eine Person um eine Firma handelt, und diese mehrere Adressen hat, wäre die Zusammengehörigkeit der Straßen und Hausnummern nicht mehr nachvollziehbar. Abbildung 2.1a zeigt die Modellierung einer Firma mit mehreren Adressen. Um dieses Problem zu umgehen, existieren leere Knoten (Blank Nodes). Abbildung 2.1b zeigt wie das Problem mit Hilfe von Blank Nodes gelöst werden kann.

2.1.6 RDF-Schema (RDFS) [13]

RDF-Schema (RDFS) ist wie RDF ein von W3C empfohlener Standard. RDFS definiert das nötige Vokabular, um, aus mit RDF beschriebenen Elementen, eine Ontologie zu erstellen. Folgendes Vokabular wird von RDFS definiert:

Class: definiert Klassen von Objekten.

Resource: ist eine besondere Klasse. Jede Entität in RDF ist automatisch eine Instanz von Resource.

Property: ist eine Unterklasse von Resource und wird benutzt, um Eigenschaften zu definieren.

Literal: ist eine Klasse für Werte, wie z.B.: Zeichenketten, Integer, etc.

subClassOf ist eine Eigenschaft, um die Unterklassenbeziehungen zu beschreiben. Die Klasse `Frau` ist eine Unterklasse der Klasse `Mensch`, da alle Frauen Menschen sind.

subPropertyOf ist eine Eigenschaft, um die Untereigenschaftsbeziehungen zu beschreiben. Die Eigenschaft `Sohn von` kann als Untereigenschaft von `Kinder von` mo-

delliert werden, da alle Söhne einer Person automatisch auch Kinder dieser Person sind.

domain: wird benutzt, um die Klasse des Subjekts einer Eigenschaft zu determinieren. Zum Beispiel muss eine Eigenschaft `Sohn von` die Klasse `Mann` als *domain* haben.

range: wird benutzt, um die Klasse des Objekts einer Eigenschaft zu determinieren. Zum Beispiel muss eine Eigenschaft `Sohn von` die Klasse `Mensch` als *range* haben.

RDF und RDFS ermöglichen aber nicht die Darstellung komplexer logischer Zusammenhänge. Für solche Konstrukte ist die OWL-Spezifikation nötig.

2.1.7 Web Ontology Language (OWL) [45]

Die Web Ontology Language, kurz OWL, ist eine vom W3C definierte Spezifikationssprache. Sie wird genutzt, um Konzepte einer Ontologie und deren Relationen (Rollen in der Beschreibungslogik) beschreiben zu können. OWL basiert dabei auf dem RDF. Allerdings umfasst sie weitaus mehr als RDF und RDF-Schema [BG04]. OWL existiert in drei verschiedenen Versionen : OWL Lite, OWL DL und OWL Full. OWL Lite ist eine Untermenge der Sprache OWL DL. OWL Lite wurde entwickelt, um die Unterstützung von OWL einfacher zu gestalten und eine volle ReasoningUnterstützung zu erlauben. Aus diesem Grund sind die Konstrukte in OWL Lite entscheidbar. OWL Full und OWL DL unterstützen dieselbe syntaktische Menge. OWL Full erlaubt die Vermischung von OWL mit RDFS und fordert keine strikte Trennung von Klassen, Eigenschaften, Individuen und Datenwerten. OWL DL schränkt die Vermischung mit RDFS ein und benötigt die Trennung von Klassen, Eigenschaften, Individuen und Datenwerten. Von vielen Softwares wird auch OWL DL unterstützt, wenn Reasoning Anwendung findet. Um die mittels RDF oder OWL strukturierten Daten abfragen zu können, ist eine Anfragesprache notwendig. Eine dabei mögliche Sprache ist SPARQL.

2.1.8 SPARQL

SPARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage (SPARQL) [51] ist der Standard vom W3C, um RDF-Wissensdatenbanken abzufragen. SPARQL benutzt Tripeln als Grundstruktur. Die Tripeln bestehen aus Subjekt, Prädikat und Objekt. In der Aussage `Rex ist ein Hund` ist `Rex` das Subjekt, `ist` das Prädikat und `Hund` das Objekt. Eine Anfrage mittels

SPARQL kann mehrere solcher Tripeln enthalten. Jedes Element eines Tripel kann durch eine Variable ersetzt werden.

2.2 Machine Learning

Das Machine Learning beschäftigt sich mit Systemen, die aus Daten lernen können. Eine typische Anwendung ist die Erkennung von SPAM bei E-Mail-Anwendungen [23]. Angefangen mit Rohdaten (z.B.: alle sich im Postfach befindenden E-Mails) werden Eigenschaften dieser berechnet, z.B.: der Absender der E-Mail. Aus bereits als SPAM bekannte E-Mails wird ein Modell trainiert, welches benutzt wird, um anschließend die neuen E-Mail zu sortieren.

Terminologisch werden die Rohdaten (auch Instanzen genannt, aus Verständlichkeit wird aber in dieser Arbeit der Terminus Instanz nur für ein konkretes Objekt einer Ontologie verwendet) durch den Datenraum $\mathfrak{X}$ dargestellt. Um ein Modell zu lernen, wird eine Menge an Beispielen Tr (Training Set) gebraucht. Die zwei Hauptwege, um ein Modell zu konstruieren sind die Klassifizierung und das Clustering.

2.2.1 Klassifizierung

Als Hauptbestandteil der Klassifizierung wird ein Klassifizierer benutzt. Ein Klassifizierer ist eine Abbildung $\hat{c} : \mathfrak{X} \rightarrow \mathfrak{C}$, wobei $\mathfrak{C} = \{C_1, C_2, ..., C_k\}$ eine gewöhnlich kleine Menge an Etiketten [23] ist. $\hat{c}$ ist eine geschätzte Funktion, die die unbekannte aber wahre Funktion c approximiert [23]. Die Erstellung des Klassifizierers soll die Funktion $\hat{c}$ so auswählen, dass diese c sich so gut wie möglich annähert; nicht nur für den Training Set Tr sondern für alle Daten in $\mathfrak{X}$.

2.2.2 Clustering

Ein Cluster ist eine Gruppierung von Daten die ähnliche Merkmale besitzen. Im Gegensatz zur Klassifizierung sind die verschiedenen Gruppen nicht im Voraus definiert, sondern werden von den Merkmalen bestimmt. Allgemein wird zwischen überwachtem (*supervised*) und unüberwachtem (*unsupervised*) Lernen unterschieden. Klassifizieren gehört zum überwachten Lernen, da die benutzten Daten etikettiert sein müssen [23], Clustering im Gegensatz zum unüberwachten. Darüber hinaus wird zwischen preditkivem und de-

skriptivem Lernen unterschieden. Prediktives Clustering trainiert ein Modell, um Daten, die nicht zu dem Training Set gehören, zu clustern. Bei deskriptivem Clustering kann das erzeugte Modell, nur die für die Erzeugeung des Modells benutzten Daten clustern [23]. Tabelle 2.1 zeigt die Zuordnung der verschiedenen Verfahren.

	Prediktiv	Deskriptiv
Überwacht	Klassifizierung, Regression	Untergruppen Entdecken
Unüberwacht	prediktives Clustering	deskriptives Clustering

Tabelle 2.1: Aufteilung der Verfahren im maschinellen Lernen nach [23]

In dieser Arbeit wurden zwei verschiedene Cluster-Verfahren verwendet. Diese werden in den nächsten Abschnitten beschrieben.

2.2.2.1 KMeans++

Der KMeans++-Algorithmus ist eine Erweiterung des KMeans-Algorithmus[4]. Letzterer ist einer der populärsten Clustering-Algorithmen[47]. Es besteht aus folgenden Teilaufgaben:

1. Wähle k-Punkte als Zentroiden
2. Für eine Entität: Wähle den nächsten Zentroid und füge die Entität dem entsprechenden Cluster hinzu.
3. Berechne den Zentroid des geänderten Clusters.

Die Anfangszentroiden werden zufällig ausgewählt. Ein KMeans-Algorithmus muss nicht die beste Lösung finden. Diese ist starkabhängig von den gewählten Startpunkten [47]. Eine Möglichkeit, dies zu umgehen ist, den Algorithmus mehrmals hintereinander laufen zu lassen und das beste Ergebnis auszuwählen. Der KMeans++-Algorithmus wählt die Anfangszentroiden nicht zufällig. Die Auswahl der Startzentroiden erfolgt gemäß:

1. Wähle einen Zentroid c_1 zufällig.
2. Wähle einen neuen Zentroid c_i aus $x \in X$ mit Wahrscheinlichkeit

$$\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$$

3. Wiederhole den Punkt 2 bis k-Zentroiden gefunden sind.

Die größte Schwierigkeit ,die sich aus diesem Algorithmus ergibt, ist, die Anzahl an Clustern festzulegen. Bei verschiedener Werten von k können sich komplett andere Lösungen ergeben [47]. Eine Möglichkeit wäre, die Anzahl an Clustern zu erhöhen und diese zu vergleichen.

Außerdem darf der Datensatz nicht zu viel Rauschen enthalten. Als Rauschen sind die Ausreißer gemeint. Die gefundenen Cluster sind immer convex, d.h.: von jedem Punkt innerhabl des Clusters ist es möglich ein gerader Segment zu jedem anderen Punkt zu ziehen, ohne die Grenze des Clusters zu berühren.

Andere Algorithmen können andere Formen finden und mit Rauschen umgehen, unter anderem DBSCAN.

2.2.2.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) ist ein Algorithmus, der die Dichte nutzt, um Daten zu gruppieren [20]. In einem zweidimensionalen Raum können die Cluster visuell voneinander bzw. von den Ausreißern getrennt werden. Abbildung 2.2 stellt Punkte in einem zweidimensionalen Raum dar. Visuell kann ein Viereck erkannt werden. Zusätzlich sind zwei Punkte entfernt zu den anderen dargestellt und können dem Viereck nicht zugeordnet werden. Diese beiden Punkte nennt man Ausreißer (Rauschen).

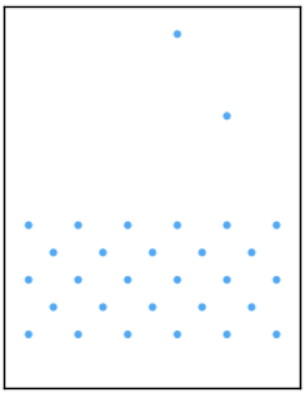


Abbildung 2.2: 2d-Raum

DBSCAN verfügt über zwei Parameter, ϵ und $MinPts$. ϵ definiert den maximalen Abstand, den zwei Punkte haben können, um benachbart zu sein. $MinPts$ beschreibt die Mindestanzahl an Punkten pro Cluster. Erreicht ein Cluster nicht die Mindestanzahl, werden dessen Punkte als Ausreißer betrachtet. DBSCAN kann mit einer beliebigen Abstandsfunktion angewendet werden. Die Reihenfolge der Daten hat keinen Einfluss auf die Ergebnisse.

2.3 Genetische Algorithmen

Die Wahl der richtigen Parametern oder der richtigen Eigenschaften der Daten beeinflusst die Qualität eines Clusterverfahren. Wir stellen hier die Grundlagen vor, um solche Parametern zu optimieren. Seit längerer Zeit streiten sich Biologen, ob die Evolution ein Optimierungsverfahren ist [55]. Das heißt, ob die natürliche Selektion allgemein eine Verbesserung der einzelnen Organismen bringt. Auf die Informatik angewandt, wandelt sich diese Frage um:

> Lässt sich durch die Simulation der Evolution ein Mechanismus finden, um Lösungen zu optimieren? [7]

Genetische Algorithmen basieren auf vereinfachten Modellen der Evolution [7]. Diese Prinzipien werden in verschiedenen Bereichen der Informatik angewandt. Es existieren viele genetische (evolutionäre) Algorithmen. Unter den am häfigsten verwendeten befindet sich der *Genetic Algorithm* (GA) [40]. (GA) benutzt eine Zeichenkette einer festen Länge, um Individuen zu repräsentieren. Diese Zeichenkette entspricht einem Chromosom in der Biologie.

Chromosomen sind biologische Strukturen, die aus DNA und andere Proteinen bestehen. Chromosomen enthalten Gene. Diese sind wiederum verantwortlich für die Kodierung bestimmter Proteine. Bei den Untereinheiten der Gene spricht man von Allelen. Jedes Allel entspricht einer möglichen Ausprägung eines Gens. Der Aufbau der kompletten Chromosomen ist komplizierter, für die genetischen Algorithmen reicht jedoch ein vereinfachtes Modell. Das Chromosom wird durch ein Datenfeld repräsentiert. Jeder Teil dieses Feldes ist ein Allel und kann verschiedene Werte annehmen, z.B. boolesche Werte. Mehrere Allele können als Gen zusammengefasst werden.

Die Chromosomen werden entweder zufällig generiert oder ausgehend von einer Anfangspopulation untereinander gekreuzt. Es existieren verschiedene Kreuzungsmechanismen, bevor diese jedoch angewendet werden können, müssen die richtigen Chromosomen ausgewählt werden. In der Biologie geht man davon aus, dass die zu ihrer Umwelt am besten angepassten Individuen sich eher reproduzieren. Das Maß der Anpassung an die Umwelt wird Fitness genannt. In der Informatik ist die Umwelt das zu lösende Problem. Die Fitness ist also die Anpassung der Lösung an das Problem. Die Fitness wird als Funktion, die die Korrektheit der durch ein Chromosom definierten Lösung charakterisiert, definiert.

Werden mit Hilfe der Fitnessfunktion die besten Chromosomen der Population ausgewählt, können diese untereinander gekreuzt werden. Die Produkte der Kreuzungen sind neue Chromosomen, deren Fitness auch evaluiert wird. Ziel ist dabei die beste Lösung zu finden. Aus Komplexitätsgründen wird auch manchmal das erste Chromosom, der eine vordefinierte Fitness erreicht, als die beste Lösung betrachtet. Eine weitere Möglichkeit die Anzahl der Operationen zu begrenzen, ist die Zahl an Generationen zu minimieren. Eine Generation entspricht den Individuen, die aus Kreuzungen von Individuen der vorherigen Generation entstanden sind.

2.4 Information Retrieval

Information Retrieval (IR) beschäftigt sich mit den Technologien, um Informationen wiederzufinden [22]. Ein wichtiger Punkt des IR ist die Evaluierung der verschiedenen Verfahren. Dazu exisitieren verschiedene Methoden. Eine vor allem in der Medizin weitverbreitete Methode ist die Benutzung eines Goldstandards.

2.4.1 Goldstandard

> *Goldstandard:* Irgendeine standardisierte Beurteilung, Methode, Verfahren, Intervention oder Maß, deren Gültigkeit und Zuverlässigkeit bekannt sind, in der Regel als die bestmögliche betrachtet, and dem andere Verfahren gemessen werden. [53]

In der Informatik kann der Goldstandard ein allwissender Experte sein bzw. eine vordefinierte Menge an Regeln. Um ein Verfahren am Goldstandard messen zu können, existieren mehrere Methoden darunter Precision und Recall.

2.4.2 Precision und Recall

Um die Genauigkeit eines Verfahrens zu messen, müssen Messwerte definiert werden. Tabelle 2.2 zeigt die vier Möglichkeiten, die ein Objekt haben kann (nach Klassifizierung unter Verwendung eines Goldstandards). Mittels dieser vier Zustände lassen sich zwei Messwerte definieren: Precision und Recall.

		Fakt	
		Positiv	Negativ
Hypothese	Positiv	Richtig positiv (tp)	Falsch positiv (fp)
	Negativ	Falsch negativ (fn)	Richtig negativ (rn)

Tabelle 2.2: Konfusionsmatrix

Die Precision ist die Quote der Ergebnisse, die relevant sind. Precision ist die Anzahl der relevanten geteilt durch die Anzahl der gefundenen Instanzen.

$$\text{Precision} = \frac{|\text{relevante Dokumente}| \bigcap |\text{gefundene Dokumente}|}{|\text{gefundene Dokumente}|} = \frac{|tp|}{|tp + fp|} \text{ [22]}$$

Die Anzahl richtiger Antworten im Verhältnis zur Gesamtzahl ist aber nicht ausreichend, um die Qualität eines Algorithmus zu charakterisieren. Um die Aussage über einen Algorithmus zu verfeinern, kann unter anderem der Recall benutzt werden. Der Recall ist die Quote der gefundenen relevanten Ergebnisse.

$$\text{Recall} = \frac{|\text{relevante Dokumente}| \bigcap |\text{gefundene Dokumente}|}{|\text{relevante Dokumente}|} = \frac{|tp|}{|tp + fn|} \text{ [22]}$$

Da das Endergebnis des in dieser Arbeit vorgestellten Verfahren zwar auf die Erkennung von fehlerhaften Daten abzielt, aber die Teilschritte allgemeiner betrachtet werden müssen, eignet sich als dritter Messwert die Korrektheit. Ein Algorithmus wird als korrekt bezeichnet, wenn er für jede Eingabe die richtige Lösung ausgibt [17].

2.4.3 F-Score

In manchen Fällen ist es von Nachteil zwei Messwerte (Precision und Recall) verwenden zu müssen, um die Ergebnisse evaluieren zu können. In diesem Fall wird der F-Score (auch F-Measure genannt) als Kombination von Precision und Recall definiert.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \text{ [22]}$$

Der F-Score kann durch β gewichtet werden. Wird β gleich zwei gesetzt, spricht man vom F_2-Score.

2.4.4 ROC-Kurve

Eine Receiver Operating Characteristics (ROC)-Kurve ist eine graphische Darstellung, um die Performanz von Klassifizierer zu visualisieren [21]. ROC-Kurven werden in der medizinischen Forschung viel angewendet und basieren dementsprechend, statt auf Precision und Recall auf die zwei in der Medizin verwendete Messwerten Spezifizität und Sensitivität. Dabei ist die Sensitivität gleich dem Recall und die Spezifizität wird folgendermaßen definiert:

$$\text{Spezifizität} = \frac{|tn|}{|fp + tn|} \text{[21]}$$

Bei einer ROC-Kurve wird als Abzisse **eins minus die Spezifizität** und als Ordinate die Sensitivität verwendet. Die Diagonale entspricht einer Klassifizierung per Zufall.

Um aus einem Klassifizierungsverfahren eine ROC-Kurve zeichnen zu können, ist ein Algorithmus notwendig. Ein dafür verwendbarer einfacher Algorithmus wird im nächsten Abschnitt vorgestellt.

2.4.4.1 Generierung

„Viele Klassifizierer, wie beispielweise der naive Baysian-Klassifizierer, nutzen ein Maß, um die Zugehörigkeit eines Objektes zu einer Klasse zu determinieren“ [21]. Ein solcher Klassifizierer kann mit Hilfe einer Schwelle in einen diskreten umgewandelt werden. Um eine ROC-Kurve zu erstellen, wird diese Schwelle von $-\infty$ zu ∞ gesetzt.

2.4.4.2 Fläche unter der Kurve (AUC)

Um zwei Verfahren vergleichen zu können, soll die ROC-Kurve auf eine einzige Metrik reduziert werden. Hierzu wird die Fläche unter der Kurve berechnet. Die Fläche bewegt sich zwischen 0 und 1,0. Bei einer Fläche von 0,5 entspricht die Kurve der Diagonalen. Die AUC eines Verfahrens ist äquivalent zur Wahrscheinlichkeit, die einem Verfahren ein per Zufall ausgewähltes positives Objekt als tatsächlich positiv zuordnet [21].

3 Verwandte Arbeiten

Vrandecic u. Sure [59] stellen Methoden vor, um Ontologiemetriken zu entwickeln. Der Artikel befasst sich vor allem mit der Definition einer Ontologienormalisation. Die Autoren betrachten die Normalisation der Ontologie als die einzige Möglichkeit, stabile Metriken zu entwickeln. Stabil sind laut den Autoren die Metriken, welche die *Open World Assumption*[19] respektieren. Das Hauptziel ist, zwei Ontologien desselben Themengebietes vergleichen zu können. Die vorgestellten Metriken messen allerdings nur die Struktur der Ontologie.

Böhm et al. [14] untersuchen die Möglichkeit, *Linked Open Data* zu profilen. Unter profilen verstehen die Autoren das Erzeugen von Metadaten, um die Daten verständlicher zu machen und das Entdecken von Anomalien zu vereinfachen. Um mit den Datenmengen umgehen zu können, werden die Daten der Ontologie zuerst geclustert. Die somit erstellten Partitionen der Ontologie enthalten ausschließlich miteinander semantisch korrelierte Daten. Anschließend werden Metriken vorgestellt, die auf dem entwickelten Schema basieren.

In [25] stellen Fürber u. Hepp ein regel-basiertes Framework zur Messung der Datenqualität einer Ontologie vor. Als erstes definieren sie neun Datenqualitätsregeln. Diese beschreiben die Hauptursachen mangelnder Qualität in einer Ontologie. Die Anzahl an Instanzen, die eine Regel nicht respektiert, wird durch die Gesamtanzahl an relevanten Instanzen geteilt. Dieses Verhältnis kann anschließend gewichtet werden, um die Wichtigkeit einzelner Regeln an die Anwendung der Daten anzupassen.

[62] stellt eine Evaluierung der Datenqualität von DBpedia[11] vor. Diese Veröffentlichung gebraucht Nutzerangaben zur Evaluierung einzelner Tripel. Das Paper stellt eine Taxonomie der Qualitätsdimensionen vor. Dabei werden vier Hauptdimensionen beschrieben:

- Fehlerfreiheit: Diese Dimension enthält die falsch extrahierten Tripel, Datentyp-Probleme und impliziten Relationen zwischen Attributen.

- Relevanz: Die Relevanz zeigt an, ob die extrahierten Informationen bedeutsam sind.
- Repräsentationskonsistenz: Sie misst, ob die Repräsentationen der numerischen Werte konsistent zueinander sind.
- Interlinking: Interlinking enthält Links zu externen Ressourcen und anderen Datensammlungen.

Außerdem werden zwei verschiedene Evaluierungsmethoden vorgestellt: zuerst wird eine manuelle Evaluierung vorgestellt. Diese Methode wird unterstützt vom *TripleCheckMate* Tool[5]. Die zweite Methode ist semi-automatisch und nutzt den DL-Learner[6], um ein erweitertes Schema zu generieren. Im nächsten Schritt werden die generierten Axiome per Hand evaluiert. Mit diesen Methoden konnte eine Fehlerquote von 11.93% in DBpedia gemessen werden.

Wang u. Strong [60] untersuchen den Einfluß der Datenqualität auf die Nutzer. Dafür wurden zwei Experimente durchgeführt. Das Ziel des ersten Experiments war eine vollständige Liste der möglichen Datenqualitätseigenschaften zu generieren. Das zweite untersuchte die Wichtigkeit der Eigenschaften für den Nutzer. Aus beiden Experimenten entstand eine Klassifikation der für den Nutzer wichtigen Qualitätsdimensionen. Die Studie ermöglicht somit, die Datenqualität in vier verschiedene Kategorien zu unterteilen und zeigt, welche Parameter betrachtet werden sollen, um die Qualität der Daten in den Augen des Endnutzers zu erweitern. Die kleine Anzahl an Testpersonen relativiert jedoch die Gültigkeit der Unterteilung.

Zaveri et al. [63] stellen einen systematischen Literaturrückblick über Methoden dar, um die Datenqualität zu messen. Die Autoren haben 21 Artikel ausgewählt und haben hieraus 26 verschiedene Qualitätsdimensionen ermittelt. Diese wurden in Kategorien, definiert in [60], klassifiziert. Aus diesem Paper entsteht eine Übersicht, welche Fehlertypen existieren, ob diese händisch, semi-automatisch oder automatisch behebbar sind, und ob ein Tool zur Behebung der Fehler existiert.

Kontokostas et al. [41] stellen eine Methode vor, um die Datenqualität von SPARQL[7]-Endpunkten zu überprüfen. Dafür haben die Autoren 14 Muster erzeugt. Diese definieren verschiedene Fehlertypen. Listing 3.1 zeigt ein Beispiel von solch einem Muster, welches den Vergleich zwischen den Literalen `v1` und `v2` definiert.

[5] `http://github.com/AKSW/TripleCheckMate`
[6] `http://dl-learner.org/Projects/DLLearner`
[7] `http://www.w3.org/TR/rdf-sparql-query/`

```
1 SELECT ?s WHERE { ?s %%P1%% ?v1 . ?s %%P2%% ?v2 .
2 FILTER ( ?v1 %%OP%% ?v2 ) }
```

Listing 3.1: Beispiel eines Muster

Für jedes Muster werden anschließend konkrete Fälle konstruiert. Ein für das in Listing 3.1 dargestellte Beispiel konkreter Fall ist das Geburtsdatum einer Person, welches vor seinem Todesdatum liegen soll. Jeder dieser Fälle ist ein Test, der mittels SPARQL auf dem Endpunkt durchgeführt wird.

4 Problemdefinition

Die Datenqualität einer Ontologie ist ein sehr weites Feld. Wie in [60] beschrieben sind viele Aspekte aus Sicht des Nutzers an die Qualität gebunden. Alle Aspekte können von einem einzelnen Verfahren nicht abgedeckt werden. In diesem Kapitel soll anhand praktischer Beispiele definiert werden, welche Teilbereiche der Datenqualität betrachtet werden.

Tabelle 4.1 zeigt die Korrespondenzen, der in [63] referenzierten Publikationen zu den von [60] definierten Datenqualitätskriterien. Die Tabelle wurde mit Hilfe der Tabellen 8 und 9 von [63] erstellt. [9] deckt zwar alle Dimensionen der Datenqualität ab, das Verfahren ist aber händisch. Um mit Hilfe des in [9] vorgestellten Verfahrens die Datenqualität zu erhöhen, müssen die Daten per Hand mit einem speziellen Vokabular annotiert und im Anschluss Regeln zum Filtern von Ressourcen definiert werden. Um diese Aufgaben stemmen zu können, muss eine genaue Kenntnis der Daten vorhanden sein. Bei größeren Datenmengen ist es nicht mehr möglich, alle Aspekte zu betrachten. Ein semi-automatisches System reduziert den manuellen Aufwand und ist dementsprechend bei großen Datenmengen anwendbar.

Unter den semi-automatischen Verfahren beschäftigen sich mehrere [14, 24, 35] mit der Fehlerfreiheit. Um die Problemdefinition zu verfeinern, betrachten wir, welches Ziel diese drei Publikationen verfolgen.

[14] Die zu untersuchende Ontologie wird analysiert und anhand des Schemas geclustert, um zusammenhängende Instanzen zu finden. Die Werte der einzelnen Properties werden nicht betrachtet, somit ist die Dimension der Fehlerfreiheit nicht ganz abgedeckt.

[24] Laut Zaveri et al. [63] werden in dieser Arbeit publizierte Daten beurteilt.

[35] Es werden RDF-Fehler identifiziert. Die Datenwerte werden hierbei nicht betrachtet.

Die Fehlerfreiheit kann nicht, ohne die Werte zu betrachten, vollständig geprüft werden. Sie ist schwer von der Vollständigkeit und Konsistenz trennbar. In dieser Arbeit betrach-

		Verfahren			
Dimension		Autom.	Semi-autom.	Manuell	keine Angabe
Intrisische DQ	Glaubwürdigkeit		[24]	[9]	
	Objektivität			[9]	
	Reputation		[33]	[9, 46]	[12, 26–28, 39, 54]
	Fehlerfreiheit		[14, 24, 35]	[9, 46]	[25, 44, 48]
Kontextuelle DQ	Vollständigkeit	[30]	[35]	[9, 46]	[25]
	Mehrwert	[30]	[24]	[9]	[44]
	Relevanz			[9]	
	Zeitlosigkeit		[24]	[9]	[25]
	Datenmenge		[24]	[9]	[16]
Daten-modellierung	Interpretabilität			[9]	[36]
	Verständlichkeit		[24]	[9]	
	Konsistenz		[24]	[9]	[36]
	Prägnanz			[9]	[36]
Erreich-barkeit	Erreichbarkeit		[24]	[9]	[36]
	Antwortzeit			[9]	
	Sicherheit			[9]	

Tabelle 4.1: Tabelle der Publikationen für jede Dimension nach [60]. Der Inhalt ist aus [63, Tabellen 8-9].
: betrachtete Dimensionen.

ten wir die folgenden drei Aspekte *Fehlerfreiheit*, *Vollständigkeit* und *Konsistenz*. Das Ziel ist vor allem Inkonsistenzen, Ungenauigkeiten oder Unvollständigkeit auf A-Box-Ebene zu finden. Es werden dabei auch die Werte verschiedener Properties betrachtet, um die drei besprochenen Dimensionen zu behandeln.

Das Auffinden von Inkonsistenzen und Ungenauigkeiten innerhalb einer Ontologie erfordert auf grund der Menge an Tripeln viel Arbeit. In vielen Fällen wird anfänglich die Qualität der Daten per Hand von Experten geprüft.

In den nächsten Abschnitten werden Beispiele für Qualitätsproblemen vorgestellt.

4.1 Goldstandardansatz

Ein weit verbreites Verfahren im *Information Retrieval* ist die Nutzung eines Goldstandards. Mit Hilfe des Goldstandards ist es möglich, zu messen, wie genau ein bestimmtes Prozedere arbeitet. Im folgenden Abschnitt wird der Fall eines vorhandenen Goldstandards zur Identifizierung von Problemen vorgestellt.

Problem A (Goldstandarderweiterung)

Aus einer Ontologie sind einige Instanzen als fehlerhaft bekannt. Es ist jedoch nicht unbedingt klar, aus welchem Grund diese Instanzen falsch sind. Da diese Instanzen manuell markiert wurden, ist außerdem unbekannt, ob alle fehlerhafte Instanzen gefunden worden sind.

Lässt sich mittels der bereits als fehlerhaft bekannten Ressourcen eine Menge an potentiell falschen Instanzen ermitteln? Diese Menge soll so klein wie möglich gehalten werden und doch alle fehlerhaften Instanzen wiederfinden.

4.2 Explorative Probleme

Ein weiteres Problem ist, dass in vielen Fälle existiert kein Goldstandard. Selbst bei einer bekannten Menge an fehlerhaften Instanzen kann nicht davon ausgegangen werden, dass diese Menge alle möglichen Fehlertypen enthält. Eine einfache Möglichkeit ist, sich alle Instanzen anzuschauen und die fehlerhaften zu korrigieren. Bei einer großen Ontologie (in erster Linie bei einer großen A-Box) ist die manuelle Korrektur mühsam und nicht zuverlässig. Vor allem wenn die Ontologie verschiedene Themen abdeckt, ist es illusorisch, dass alle Aspekte der Daten beurteilt werden können. Aus diesen Gründen ist ein exploratives Verfahren notwendig. Im folgenden Abschnitt werden die Problemfälle, die von solch einem Verfahren abgedeckt werden sollten, vorgestellt.

4.2.1 Problem B (Vollständigkeit der Properties)

Die Vollständigkeit einer Instanz in einer Ontologie ist nur überprüfbar, wenn eine allwissende Beschreibung der Instanz vorliegt. Es ist aber auch möglich, einen Rumpf zu definieren, der beschreibt, welche Eigenschaften eine Instanz einer bestimmten Klas-

se besitzen muss, um als vollständig betrachtet zu werden. In vielen Fällen liegt aber weder das eine noch das andere vor. Listing 4.1 zeigt eine Beispielontolgie. Diese enthält die Klasse `:Hotel`. Außerdem besitzt sie eine Klasse `:Portal`. Die Eigenschaft `:referenziertVon` verbindet ein Hotel mit einem Portal. Um zu wissen, ob eine Instanz der Klasse `:Hotel` vollständig ist, muss überprüft werden, ob alle Portale, die dieses Hotel referenzieren in der Ontologie korrekt eingetragen sind. Abbildung 4.1 stellt das Beispiel graphisch dar. Die Beispielontologie zeigt allerdings insgesamt einen einfachen Fall, denn nur `hotel_3` wird nicht von allen Portalen referenziert. Es sollte aber überprüft werden, ob diese Instanz vollständig ist. In einer Ontologie mit hunderten Hotels wäre das weitaus schwieriger. Das sich daraus ergebende Problem ist, welche Instanzen im Vergleich zu anderen Instanzen derselben Klasse, andere Properties bzw. eine andere Anzahl im Vorkommen einer bestimmten Property haben.

```
1  Prefix: : <http://www.example.org/ontology#>
2  Class: Hotel
3  Class: Portal
4  ObjectProperty: referenziertVon
5      Domain: Hotel
6      Range: Portal
7  Individual: portal_a
8      Types: Portal
9  Individual: portal_b
10     Types: Portal
11 Individual: portal_c
12     Types: Portal
13 Individual: hotel_1
14     Types: Hotel
15     Facts:
16         referenziertVon  portal_c,
17         referenziertVon  portal_b,
18         referenziertVon  portal_a
19 Individual: hotel_2
20     Types: Hotel
21     Facts:
22         referenziertVon  portal_a,
23         referenziertVon  portal_b,
24         referenziertVon  portal_c
25 Individual: hotel_3
26     Types: Hotel
27     Facts:
28         referenziertVon  portal_a
```

Listing 4.1: Beispielontologie in Manchester Syntax

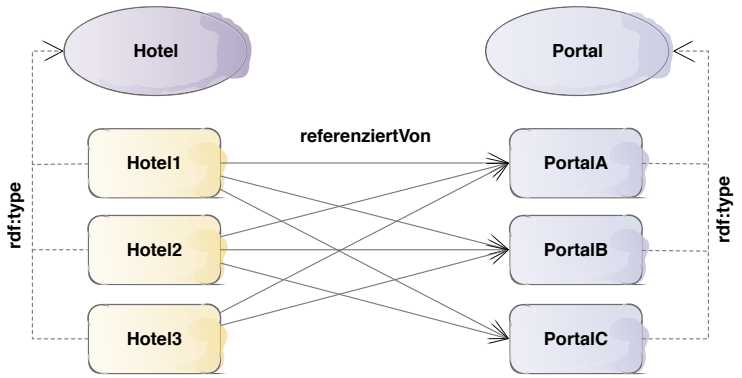


Abbildung 4.1: Graphische Darstellung von Listing 4.1

4.2.2 Problem C (Semantische Korrektheit der Properties)

Listing 4.2 zeigt eine Beispielontologie. Alle in dieser auftretenden Hotels besitzen 4 mal die Property `locatedIn`. Näher betrachtet unterscheiden sich alle drei Hotels voneinander. Das erste Hotel ist auf einem Kontinent, in einem Land, einem Bundesland und einer Stadt lokalisiert. Das zweite ist auf einem Kontinent, in einem Land, einem Bundesland und zwei Städten aufzufinden. Das dritte steht auf einem Kontinent, in zwei Ländern, und einer Stadt. Das dritte Hotel ist logischerweise mit seinen zwei Ländern fehlerhaft. Beim zweiten Hotel hängt es von der Modellierung ab, werden nämlich Stadtteile als Stadt (`City`) modelliert, ist er korrekt. Das Ziel sind Fälle, wie `Hotel3` zu erkennen und zu entscheiden, ob `Hotel2` fehlerhaft ist oder nicht.

```
1  Prefix: ont: <http://www.example.org/ontology#>
2  Prefix: res: <http://www.example.org/resource/>
3  #Definitionen
4  ObjectProperty:  ont:locatedIn
5      Domain: ont:Hotel
6  Range: ont:PlaceClass:  ont:Hotel
7  Class:  ont:City SubClassOf: ont:Place
8  Class:  ont:Continent SubClassOf: ont:Place
9  Class:  ont:Country SubClassOf: ont:Place
10 Class:  ont:Place
11 Class:  ont:AdminDivision SubClassOf: ont:Place
12 #Hotels
13 Individual: res:Hotel1
14   Types: ont:Hotel
15   Facts: ont:locatedIn res:Deutschland,
16          ont:locatedIn res:Leipzig,
17          ont:locatedIn res:Europe,
```

```
18          ont:locatedIn res:Sachsen
19 Individual: res:Hotel2
20   Types: ont:Hotel
21   Facts: ont:locatedIn res:Friedrichshain,
22          ont:locatedIn res:Europe,
23          ont:locatedIn res:Deutschland,
24          ont:locatedIn res:Berlin
25 Individual: res:Hotel3
26   Types: ont:Hotel
27   Facts: ont:locatedIn  res:Europe,
28          ont:locatedIn  res:Schweiz,
29          ont:locatedIn  res:Freiburg,
30          ont:locatedIn  res:Deutschland
31 #Länder, Städte, Regionen
32 Individual: res:Friedrichshain
33   Types: ont:City
34 Individual: res:Schweiz
35   Types: ont:Country
36 Individual: res:Freiburg
37   Types: ont:City
38 Individual: res:Deutschland
39   Types: ont:Country
40 Individual: res:Leipzig
41   Types: ont:City
42 Individual: res:Berlin
43   Types: ont:City,
44          ont:AdminDivision
45 Individual: res:Europe
46   Types:  ont:Continent
47 Individual: res:Sachsen
48   Types:  ont:AdminDivision
```

Listing 4.2: Beispielontologie in Manchester Syntax für 4.2.2

4.2.3 Problem D (Konsistenz der Werte)

Listing 4.3 und Abbildung 4.3 zeigen eine weitere Beispielontologie. Das `Hotel3` hat eine Sterneanzahl von sechs, jeder, der sich ein wenig mit dem Sternesystem der Hotels auskennt, wird diese Instanz sofort als fehlerhaft erkennen. Es existieren jedoch Länder, in denen die Hotels mit sechs bis sieben Sterne klassifiziert sein können. Die Anzahl an Zimmern ist eindeutiger. Ein Hotel wie `Hotel3` mit 20000 Zimmern ist unwahrscheinlich. Über die aktuell größte Anzahl an Zimmer verfügt das Venetian Ressort mit 7200 Zimmern [8]. Auf einem anderen Gebiet und einen höheren Anzahl an Instanzen könnte

[8] `http://de.wikipedia.org/wiki/Liste_der_gr%C3%B6%C3%9Ften_Hotels`

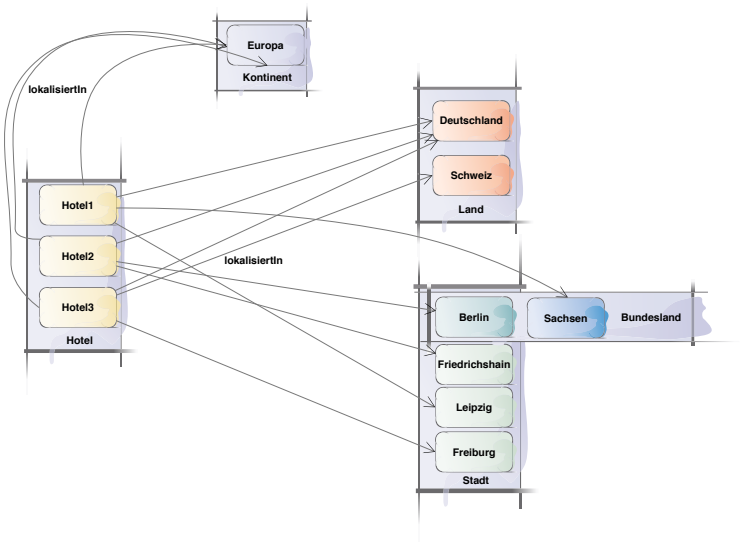


Abbildung 4.2: Graphische Darstellung von Listing 4.2

ein solcher Fehler unentdeckt bleiben. Das Ziel ist somit, Werte von Datatypeproperties, die sich signifikant von den anderen differenzieren, zu finden.

```
1  Prefix: ont: <http://www.example.org/ontology#>
2  Prefix: res: <http://www.example.org/resource/>
3  DataProperty: ont:sternenAnzahl
4      Domain: ont:Hotel
5      Range: xsd:int
6  Class: ont:Hotel
7  Individual: res:Hotel1
8      Types: ont:Hotel
9          Facts: ont:sternenAnzahl "3"^^xsd:int
10                ont:zimmerAnzahl "100"^^xsd:int
11 Individual: res:Hotel2
12         Types: ont:Hotel
13         Facts: ont:sternenAnzahl "4"^^xsd:int
14                ont:zimmerAnzahl "200"^^xsd:int
15 Individual: res:Hotel3
16     Types: ont:Hotel
17     Facts: ont:sternenAnzahl "6"^^xsd:int
18                ont:zimmerAnzahl "2000"^^xsd:int
```

Listing 4.3: Beispielontologie in Manchester Syntax für 4.2.3

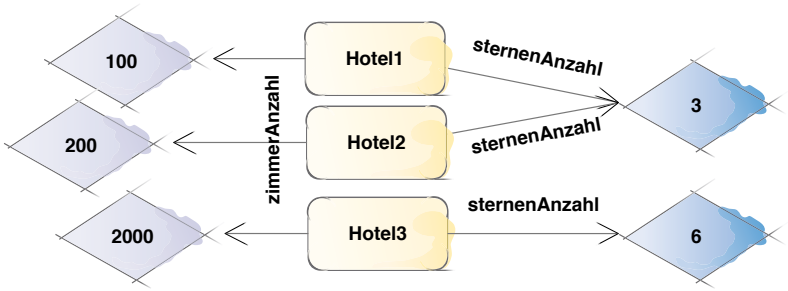
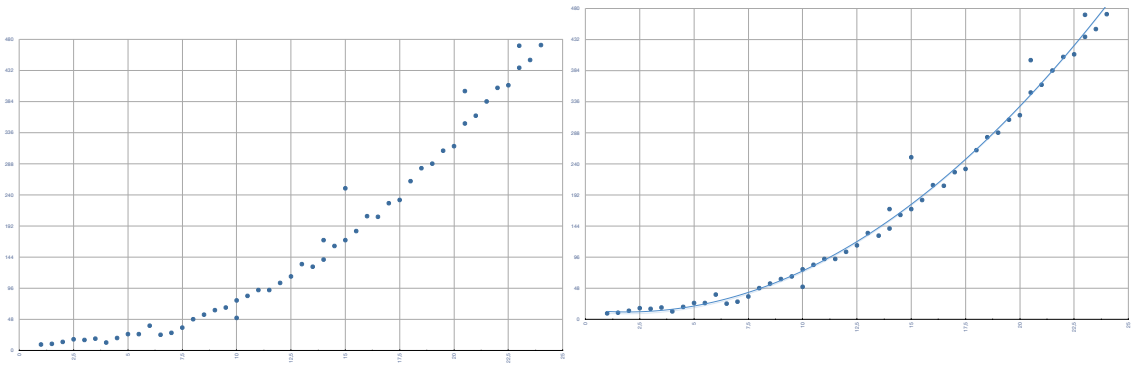


Abbildung 4.3: Graphische Darstellung von Listing 4.3



(a) Korrelation **(b)** Regressionskurve

Abbildung 4.4: Beispiel korrelierte Daten und mögliche Regressionsfunktion

4.2.4 Problem E (Konsistenz der Werte (Korrelation))

Bei bestimmten Klassen können die Werte der Instanzen korrelieren. Beispielsweise bei der Klasse Mensch sollte die Größe mit dem Körpergewicht in Relation zueinander stehen. Bei einer komplexen Ontologie ist nicht eindeutig, welche Werte eventuell mit welchen korrelieren. Das Problem ist somit zu determinieren, welche Werte miteinander zusammenhängen und dann die Ausreißer zu finden.

4.2.5 Problem F (Konsistenz der Werte (Korrelation zw. Instanzen))

Die Konsistenz einer Instanz ist nicht, ohne die mit dieser verbundenen Instanzen zu betrachten, evaluierbar. Listing 4.4 zeigt mehrere Konsistenzprobleme. Das erste ist `BundesLandA`. Ein Bundesland kann nicht mehr Einwohner haben als das Land, dem es angehört. Einen etwas komplizierteren Fall zeigen `Stadt1` und `Stadt2`. Beide verfügen

zwar über weniger Einwohner als das Bundesland, in dem sie liegen, aber addiert man beide Einwohnerzahlen, so ist die Summe höher als die Einwohnerzahl des Bundeslands. Problematisch ist somit die richtigen Klassen und Properties zu finden, um eine solche Überprüfung durchführen zu können. Darüber hinaus muss bestimmt werden, wie die Relation zueinander ist. In diesem Beispiel soll die Summe der Einwohner aller Städte eines Landes kleiner gleich zu der Einwohnerzahl desselben Landes sein.

```
1  Class: ont:Country
2  Class: ont:City
3  Class: ont:AdminDivision
4  DataProperty: ont:sternenZahl
5      Domain: ont:Country or ont:City or ont:AdminDivision
6      Range: xsd:int
7  ObjectProperty: ont:locatedIn
8      Domain: ont:City or ont:AdminDivision
9      Range: ont:Country or ont:AdminDivision
10 Individual: res:LandX
11     Types: ont:Country
12     Facts: ont:einwohnerZahl "500"^^xsd:int
13 Individual: res:BundesLandA
14     Types: ont:AdminDivision
15     Facts: ont:einwohnerZahl "600"^^xsd:int,
16     ont:locatedIn res:LandX
17 Individual: res:Stadt1
18     Types: ont:City
19     Facts: ont:einwohnerZahl "300"^^xsd:int,
20     ont:locatedIn res:BundesLandA
21 Individual: res:Stadt2
22     Types: ont:City
23     Facts: ont:einwohnerZahl "400"^^xsd:int,
24     ont:locatedIn res:BundesLandA
```

Listing 4.4: Beispielontologie in Manchester Syntax für 4.2.5

5 Implementierung

5.1 Architektur des Systems

Der allgemeine Aufbau des hier vorgestellten Systems basiert auf dem Pipeline-Entwurfsmuster [58]. Der Quellcode ist unter `https://bitbucket.org/d_cherix/ontology-metrics/overview` verfügbar. Die Pipeline besteht aus 3 Hauptteilen. Abbildung 5.1 zeigt die Reihenfolge der Stufen. Der erste Teil ist die Importstufe. Dieser Prozessabschnitt sorgt dafür, dass die über einen SPARQL-Endpoint erreichbaren Daten importiert werden können. Der Import stellt die Untermenge an Tripeln, die für die nächsten Schritten benötigt werden, zur Verfügung. Auf diese Stufe folgen nun die Funktionalitätsstufen. Jede Stufe innerhalb dieses Abschnittes berechnet auf Basis der importierten Daten verschiedene Kenngrößen. Die Stufen sollen kombinierbar sein, um verschiedene Kenngrößen in einem Durchgang integrieren zu können. Mit Hilfe der berechneten Größen werden im nächsten Abschnitt die Daten geclustert. Diese Clusteringstufe ist, wie auch bei den vorherigen, an jede Problemstellung anpassbar. Schließlich werden die Ergebnisse des Clusterings zurückgegeben und anhand der vom System als fehlerhaft erkannten Instanzen evaluiert. Dafür werden für diese Instanzen Qualitätsbewerter angezeigt. Qualitätsbewerter sind Menschen mit ausreichenden Datenkenntnissen, um zu entscheiden, ob ein Fakt richtig oder falsch ist. Die Qualitätsbewerter entscheiden, ob die Instanzen wirklich fehlerhaft sind. Anschließend wird die Pipeline gegebenenfalls mit angepassten Einstellungen neu gestartet.

5.2 Allgemeine Implementierungen

Als erstes müssen die Daten der Ontologie importiert werden.

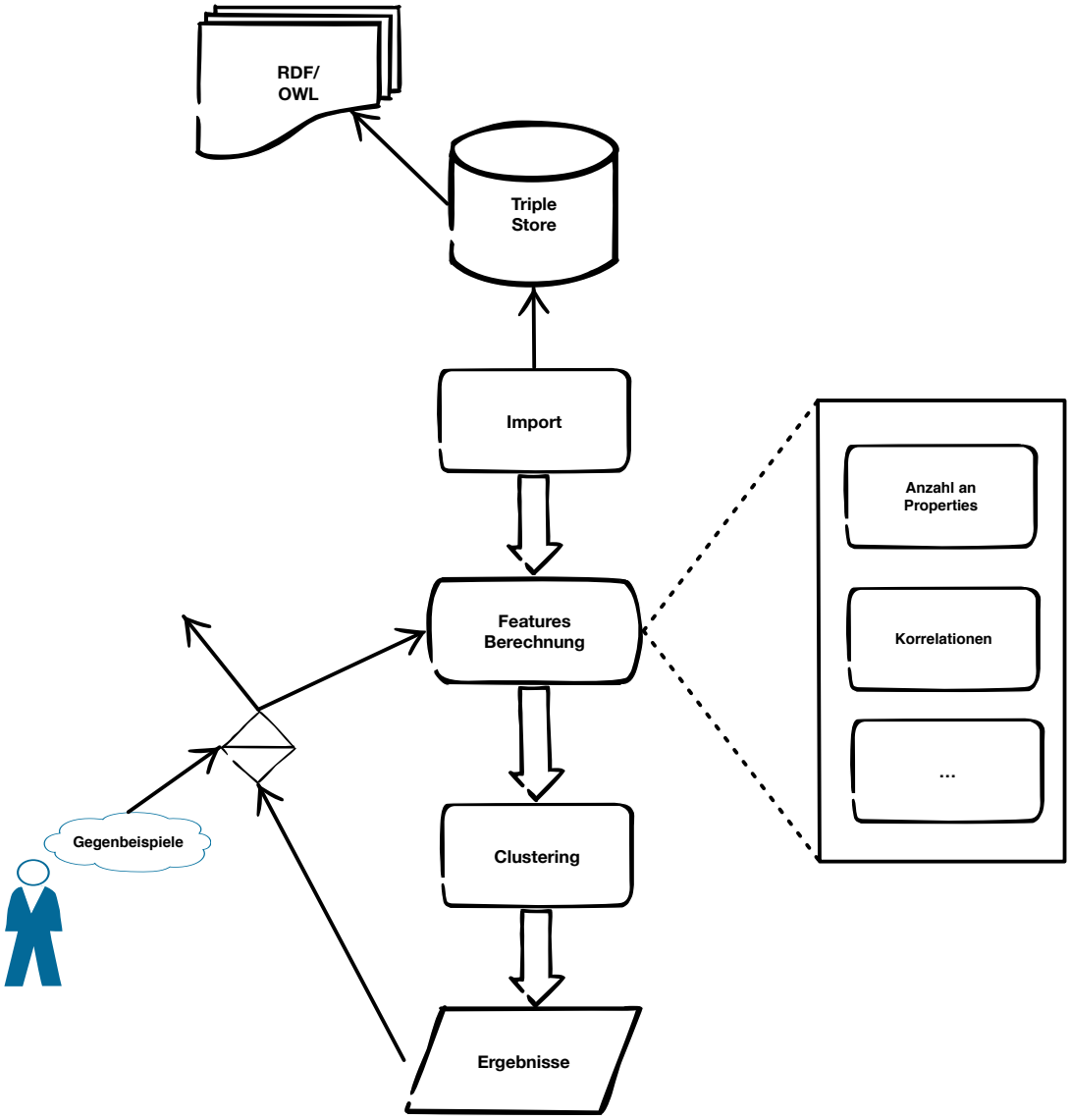


Abbildung 5.1: Pipeline Grundaufbau

5.2.1 Import

Um aus jedem beliebigen Triple Store die Ontolgiedaten gewinnen zu können, ist die Benutzung eines definierten Standards notwendig. SPARQL[52] ist ein W3C[9]-Standard, um Ontologiedaten aus Triple Stores abzufragen. Dabei soll jede Instanz durch ihre Eigenschaften beschrieben werden. SPARQL bietet mit `DESCRIBE`-Anfragen einen mögli-

[9] `http://www.w3.org`

chen Weg, Beschreibungen von Instanzen zu erhalten. Jedoch wird `DESCRIBE` von jedem Triple Store anders umgesetzt. Um die Nachteile einer Store-abhängigen Anfrage zu vermeiden, wird der *Concise Bounded Description* (CBD)[56]-Ansatz verfolgt.

Gegeben seien ein Startknoten in einem RDF-Graph, und eine Tiefe, (Quellgraph) die Concise Bounded Description enthält:

1. alle Aussagen, deren Subjekt der Startknoten ist
2. rekursiv, die Auflösung aller Blanknode, die in den bereits integrierten Aussagen auftreffen
3. rekursiv für alle Aussagen in der Beschreibung die CBD-Beschreibung, deren Objekte bis die vordefinierte Tiefe erreicht ist

Der CBD-Algorithmus liefert nur Tripel, deren Subjekt dem Startknoten, mit Ausnahme der Blanknodes, entspricht. Die eingehenden Kanten können dennoch einen Einfluß auf die betrachtete Instanz haben. Zum Beispiel: Um einen Flughafen zu beschreiben, ist nicht nur interessant, wo er liegt, wie viele Passagiere am Tag abfertigt werden usw., sondern eventuell auch, welche Fluggesellschaften diesen Flughafen nutzen. Ist die Ontologie so modelliert, dass es nur ein Attribut bei Fluggesellschaften `bedient` enthält und keine Eigenschaft `wird angeflogen von`, somit ist *CDB* nicht ausreichend. Zu den oben beschriebenen Regeln, ist folgendes hinzufügen: Die Beschreibung enthält alle Aussagen, deren Objekt der Startknoten ist. Außerdem wird die rekursive Regel (Punkt 3) auf die Subjekte erweitert. Diese erweiterte Version wird *Symmetric Concise Bounded Description* (SCDB)[56] genannt.

5.2.2 Metriken

Unter dem Begriff Metriken werden in diesem Fall die einzelnen Pipelinestufen, die im zweiten Teil des Systems angewandt werden, verstanden. Die Instanzen einer Ontologie sind in Triplen enthalten. Die Menge der Triple, die eine bestimmte Instanz als Subjekt oder Objekt enthalten, kann sehr groß sein. Außerdem können die anderen Anteile eines Tripel verschiedene Formen besitzen. Um Instanzen genau vergleichen zu können, müssen diese vergleichbare Attribute besitzen. Metriken definieren, wie aus den Tripeln, die eine bestimmte Instanz als Subjekt oder Objekt besitzen, Kennzahlen berechnet werden können.

5.3 Anpassungen zu den jeweiligen Problemfällen

Alle hier vorgestellten Implementierungen sind in Java[10]. Für das Pipeline-Entwurfsmuster und die Konfigurationen wird das Framework Spring[11] genutzt. Für jeden der im Kapitel 4 definierten Fälle wurden spezifische Implementierungen durchgeführt.

5.3.1 Implementierung für A (Goldstandarderweiterung)

Die verschiedenen Pipelinestufen, die in Abschnitt 5.1 beschrieben sind, müssen an das Problem A (cf. 4.1) angepasst werden. Nur die Importstufe muss nicht geändert werden.

5.3.1.1 Metriken

Im nächsten Abschnitt werden allgemeine Metriken für dieses Problem vorgestellt.

Grundmetriken OWL unterscheidet zwischen zwei Typen von Attributen [49]: einerseits die Objektattribute (Object Property) und andererseits die Datenattribute (Data Property). Als erstes widmen wir uns den Datenattributen.

Datenattribute Datenattribute enthalten keine Objekte als dritten Wert des Tripels, sondern Literale. Diese Literale können verschiedene Datentypen annehmen [15].

Instanzen können mehrere Attribute besitzen, die über dasselbe Prädikat verbunden sind. Ein typisches Beispiel ist das Prädikat `rdfs:label`[12], das üblicherweise einen Wert für jede in der Ontologie dargestellte Sprache besitzt. In derartigen Fällen ist die Anzahl der Attribute interessanter als deren Länge, um eventuell fehlende Sprachen zu entdecken. Tabelle 5.1 zeigt die Behandlung von Datenattributen.

Objektattribute Objektattribute sind Attribute, die kein Literal enthalten, sondern eine Instanz. Diese Instanz kann je nach Prädikat einer beliebigen Klasse angehören. Wie bei den Datenattributen können manche Prädikate mehrere Werte annehmen. Beispielweise

[10] `http://www.java.com`
[11] `http://spring.io/`
[12] rdfs: `http://www.w3.org/2000/01/rdf-schema#`

hat das Prädikat `rdf:type`[13] sehr oft mehr als nur einen Wert. Die Anzahl an Klassen, zu denen eine Instanz angehört, ermöglicht zwar einen Vergleich, jedoch wird die Semantik nicht beachtet. Um die semantische Dimension zu bewahren, wird der Jaccard-Koeffizient [38] verwendet. (5.1) definiert den Jaccard-Koeffizient für zwei Mengen: A und B.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.1}$$

Sonderfall

In vielen Ontologien werden Koordinaten verwendet. Diese können in verschiedenen Formen dargestellt werden. Eine Möglichkeit ist die Benutzung der Attribute: `geo:lat`[14]und `geo:long`[14]. Diese beiden Attribute zählen zum numerischen Typ, sollen jedoch nicht als solches behandelt werden. Aus beiden Werten bzw. anderen Darstellungsformen wird ein *Geopoint Objekt* erzeugt. Die Metrik für Geopoint Objekte ist der geographische Abstand geteilt durch die Hälfte der Äquatorlänge. Die Hälfte der Äquatorlänge ist die größtmögliche Entfernung zweier Punkte auf der Erde.

Abstandsfunktion

Aus den verschiedenen Metriken soll der Abstand zwischen zwei Instanzen ermittelt werden. Dieser Abstand wird für das Cluster-Verfahren benötigt. Für jeden Metriktyp definiert Tabelle 5.1 Abstandsfunktionen. Definition 1 zeigt, wie aus den einzelnen die globale Abstandsfunktion definiert wird. Diese basiert auf der euklidischen Norm [3].

	Typ	Kennzahl	Abstandsfunktion
Literale	Numerisch	Wert	$\left\lvert \frac{a}{max} - \frac{b}{max} \right\rvert$
	Text	Textlänge	
	Menge von Texten	Anzahl an Elementen	
Objekte	Menge von Objekten	Menge	$1 - J(A,B) = 1 - \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$
	Koordinaten	Latitude und Longitude	$\frac{geodistance(A,B)}{\text{Halbe Äquatorlänge}}$

Tabelle 5.1: Typen von Attributen und deren Abstandsfunktion

[13] rdf: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
[14] geo: `http://www.w3.org/2003/01/geo/wgs84_pos#`

Definition 1. *Gegeben seien die Instanzen I, J mit den Metriken $(i_1, ..., i_n)$ bzw. $(j_1, ..., j_n)$:*

$$dist(I, J) = \sqrt{\sum_{k=1}^{n} \left(dist'\left(i_k, j_k\right)^2 \right)}$$

wobei $dist'(i_k, j_k)$ die zum dem Typ von i_k und j_k korrespondierende Abstandsfunktion ist.

Ausgabeformat Um die Metriken mittels verschiedener Clusterverfahren auswerten zu können, müssen diese in einem vordefinierten Format ausgegeben werden können. Als Grundlage wurde das Attribute-Relation File Format (ARFF)[15] von WEKA [32] verwendet. Eine ARFF-Datei teilt sich in zwei Bereiche. Der erste enthält die Definitionen. Zuerst wird festgelegt, welchen Name den Datensatz erhalten soll. Darauf folgt eine Liste der Metrikennamen und deren Typ. Im zweiten Bereich, der Datenbereich, wird auf jeder Zeile ein Objekt beschrieben. Wie im CSV-Format werden die Zeilen mittels Komma in Spalten getrennt. Die erste Spalte enstpricht dem Wert der ersten Metrik in der zuvor definierten Liste. Diesem Format wird grundsätzlich gefolgt, aber da WEKA nicht alle Datentypen unterstützt, die wir in den Metriken gesehen haben, muss er erweitert werden. Für die Metriken vom Typ Menge oder Geopunkt wird die JSON[16] Repräsentation des Objekts als Zeichenkette in die Spalte eingetragen. Ein Geopunkt Objekt sieht zum Beispiel folgendermaßen aus: {"latitude":41.8219,"longitude":−103.652}

5.3.1.2 Clusterverfahren

Die fehlerhaften Instanzen sollen von den korrekten getrennt werden. Als erster Schritt ist dabei notwendig, fehlerhafte Daten per Hand zu finden. Die so ermittelten Daten sollen als Goldstandard dienen.

Genetischer Algorithmus Für jede Instanz werden viele Metriken berechnet. Es ist nicht vorhersehbar, welche dieser Metriken die fehlerhaften von den korrekten Instanzen trennen können. Bei n Metriken gibt es 2^n Möglichkeiten diese zu kombinieren. Das bedeutet es muss entschieden werden, ob eine Metrik zum Ergebnis beiträgt oder nicht. Bei dieser Problematik hilft die genetische Programmierung [42]. Als Chromosom wird ein Vektor λ definiert dessen Länge gleich der Anzahl der Metriken ist. Mit Hilfe von λ wird Definition 1 wie folgt erweitert:

[15] http://weka.wikispaces.com/ARFF+%28stable+version%29
[16] http://www.json.org/

Definition 2.

$$dist_w(I, J) = \sqrt{\sum_{1=k}^{n} \left(\lambda_k \cdot dist'\left(i_k, j_k\right)^2\right)}$$

Genetische Algorithmen besitzen eine Fitness-Funktion, um die Güte eines Chromosoms zu berechnen. Als Chromosom wird hier der Vektor λ verwendet. Um die Güte eines Information Retrieval Verfahrens zu messen, ist es üblich den F-Score zu verwenden [37]. Gleichung 5.2 zeigt die Berechnung des F-Scores.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{5.2}$$

Im darauf folgenden Schritt werden die gewichteten Attribute geclustert. Für diesen Schritt wurde der KMeans++-Algorithmus[5] verwendet.

5.3.2 Implementierung für B (Vollständigkeit der Properties) und C (Semantische Korrektheit der Properties)

Die Importstufe muss nicht erneut geändert werden. Soll aber die Vollständigkeit der Properties überprüft werden, müssen neue Metriken entwickelt werden.

5.3.2.1 Metriken

Die Vollständigkeit der Properties kann mit der Existenz derselben geprüft werden. Die Metriken für alle existierenden Properties einer Klasse entsprechen 1 für existent und 0 für nicht existent. Es ist aber sinnvoll, gleichzeitig zu überprüfen, ob eine Property mehrfach für eine Instanz vorhanden ist. So kann überprüft werden ob eine Property eventuell als funktional deklariert werden sollte. Eine als funktional deklarierte Property kann maximal einen Wert pro Instanz besitzen. Die Metriken sind also für jede Property zu jeder Instanz der Anzahl des Vorkommens. Im Fall des Problem C wird jede Property in den auftreffenden Typen der verlinkten Objekte unterteilt. Statt nur die Property `locatedIn` zu betrachten werden deren Unterteilungen in `locatedIn:Country`, `locatedIn:City` usw. herangezogen.

5.3.2.2 Clusterverfahren

Da jede Property, die bei den Instanzen der untersuchten Klassen auftritt, eine eigene Dimension darstellt, kann die Anzahl an Dimensionen sehr groß werden. Zuerst wird für jede Metrik geschaut, ob alle Instanzen denselben Wert besitzen. Falls dem so ist, kann die Metrik ignoriert werden. Um die Anzahl der Dimensionen zu reduzieren, wird jede Metrik einzeln geclustert. Da nur noch eine Dimension pro Clusterverfahren existiert, ist kein komplizierter Clusteralgorithmus notwendig. Es werden alle Instanzen, die denselben Wert aufweisen, in ein Cluster geschoben, und dieses erhält der Name der Property und deren Anzahl an Vorkommen. Somit ist die Überprüfung der Ergebnisse einfacher.

Dieses einfache Clusterverfahren hat jedoch den Nachteil, dass Zusammenhänge zwischen den Werten nicht betrachtet werden. Dementsprechend werden die Metriken als ARFF Datei ausgegeben und können anschließend zusammen geclustert werden. Da alle Metriken Standardtypen entsprechen, können WEKA oder ELKI[2] zum clustern verwendet werden. Mit WEKA können sogar die Metriken mit niedriger Varianz automatisch gefiltert werden.

5.3.3 Implementierung für D (Konsistenz der Werte)

Die Implementierung dieser Fall ist bezüglich der Metriken gleich dem Goldstandardansatz **??**.

5.3.4 Implementierung für E (Korrelation)

Um Problem E (4.2.4) lösen zu können, sind vor allem Anpassungen bei den Feature-Stufen nötig. Die Importstufe bleibt identisch.

5.3.4.1 Metriken

Die Korrelation benötigt nur eine Metrik. Die Korrelation zwischen zwei Properties wird über deren Werte berechnet. Um aber aus diesen die Instanzen herauszufinden, die ein anderes Verhältnis besitzen, reicht der Korrelationskoeffizient nicht aus.

Zunächst muss überprüft werden, ob eine Korrelation vorliegt. Als Beispiel nutzen wir eine Ontologie mit der Klasse `Mensch`. Die Klasse `Mensch` besitzt die Properties `Größe` und `Gewicht`. Untersucht wird die Korrelation des Gewichtes in Bezug zu der Größe des

Menschen. Für jede Instanz der Klasse `Mensch` werden die Werte für Größe und Gewicht betrachtet. Um zu testen, ob eine Korrelation vorliegt, wird ein Korrelationskoeffizient genutzt. Es muss allerdings in Betracht gezogen werden, dass mehrere Korrelationskoeffizienten existieren und die Werte nicht als parametrisch angesehen werden können. Wir wählen einen gegen Ausreißer robusten Koeffizienten aus, wobei robust gegen Ausreißer bedeutet, dass das Ergebniss nicht von den Ausreißern beeinflusst wird. Dies ist notwendig, weil wir erwarten, dass die Daten Ausreißer enthalten. Deswegen wird der Spearman'sche Korrelationskoeffizient benutzt [50].

Liegt eine Korrelation vor, muss eine Metrik charakterisieren, ob eine Instanz dieser Korrelation folgt oder nicht. Aus den Daten kann anschließend eine Regressionsfunktion ermittelt werden. Die Metrik besteht aus dem Abstand zwischen dem Wert, der die Funktion für bspw. ein Körpergewicht bei einer bestimmten Größen ermittelt, und dem in der Ontologie vorhandenen Wert für das Körpergewicht. Um die Regressionsfunktion ermitteln zu können, muss aber das richtige Regressionsmodell ausgewählt werden. Für die Regressionsanalyse wird empfohlen ein Plot der Daten zu zeichnen und visuell zu entscheiden [18]. Um dieses zu umgehen, werden polynomiale Regressionen verschiedenen Grades berechnet. Mittels des Determinationskoeffizienten R^2 wird entschieden, welche Funktion besser die zu Grunde liegenden Daten erfasst.

5.3.4.2 Clusterverfahren

Bei der Korrelationsuntersuchung ensteht eine einzige Metrik. Diese mit Hilfe von klassischen Clusterverfahren zu untersuchen, ergibt keine Ergebnisse. Deswegen wurden zwei Verfahren entwickelt.

Quantil-Methode Das erste Verfahren nutzt die statistische Verteilung der Instanzen. Es hat nur ein einzigen Parameter p. Die Instanzen, die im oberen p-Quantil liegen, werden als fehlerhaft markiert.

Sprung-Methode Die Wahl des Parameters p beeinflusst die Ergebnisse. Um dies zu umgehen wurde ein anderes Verfahren entwickelt. Dieses ordnet die Instanzen aufsteigend nach deren Korrelationsmetrik. Anschließend wird die Differenz zwischen zwei aufeinanderfolgenden Werten berechnet. Dabei wird die größte Differenz ermittelt. Die Instanzen, die sich in der geordneten Liste vor der größten Differenz (Sprung) befinden,

werden als richtig betrachtet. Die Instanzen nach dem Sprung werden als fehlerhaft markiert.

5.3.5 Implementierung für F (Externe Korrelation)

Erneut bleiben Importstufe und Clustering unangetastet. Die Tiefe der SCBD-Anfrage muss jedoch größer gleich zwei sein.

Metriken

Zunächst müssen die Properties ausgewählt werden. Nicht jede Datatypeproperty einer Instanz soll mit alle Instanzen, welche mit dieser direkt in Relation stehen, untersucht werden. Für eine Instanz A werden alle Instanzen $B_1, B_2, ..., B_n$ gesucht, die in einem Tripel $\{A, P, B\}$ stehen. Anschließend werden für alle $B_1, ..., B_n$ die Namen (URI) der Datatypeproperty durchgegangen und nur die beibehalten, die dieselbe URI wie eine der Datatypeproperty von A besitzen. Die Metrik betrachtet dann die Werte der beiden Instanzen für die gleiche Datatypeproperty. Zum Beispiel wird bei einer Stadt die Einwohnerzahl mit der des Landes verglichen, in dem sie liegt. Die Metrik kann drei verschiedene Werte annehmen: -1 falls der Wert von A kleiner ist, 0 falls beide Werte gleich sind oder 1 falls der Wert von A größer ist.

Wie in Unterabschnitt 4.2.5 gezeigt, können die Einwohnerzahlen aller Städte im Einzelnen kleiner als die Einwohnerzahl des zugehörigen Landes, jedoch in der Summe größer. Um solche Fehler zu finden, wird das vorherige Prozedere leicht angepasst. Für alle Instanzen $A_i \in \{A_1, ..., A_k\}$, die in einem Tripel $\{A_i, P, B\}$ stehen, wird die Summe der Werte für eine bestimmte Property betrachtet.

6 Auswertung

In diesem Kapitel werden die durchgeführten Experimente vorgestellt und deren Ergebnisse. Die Ergebnisse dieser werden ebenfalls vorgestellt und diskutiert. Um Experimente durchführen zu können sind Daten notwendig. Da die hier vorgestellten Fälle sehr verschieden sind, werden verschiedene Datensätze benötigt. Die für jedes Problem benutzte Datensätze werden in den Jeweiligen Kapiteln vorgestellt. Alle Fälle wurden auch auf eine Ontologie angewändt. Diese Ontologie (folgend Reiseontologie bezeichnet) beschäftigt sich hauptsächlich mit dem Reisebereich.

6.1 Evaluierung von A (Goldstandardansatz)

Um die Lösung für das in 4.1 definierte Problem evaluieren zu können, sind Daten nötig. Der nächste Abschnitt beschreibt den hierfür verwendeten Datensatz.

6.1.1 Datensatz

Die DBpedia ist das Ergebnis der Extraktion von strukturierten Informationen aus Wikipedia [11]. Sie beschreibt mehr als 2.6 Millionen Ressourcen in 30 Sprachen [11]. Bei allen auf DBpedia durchgeführten Experimenten wurde die Version 3.8 verwendet. Um die Datenmenge zu reduzieren, betrachten wir zunächst nur die englischsprachige DBpedia. Daraus extrahieren wir alle deutschen Universitäten. Listing 6.1 zeigt die SPARQL-Anfrage, um die gewollten Hochschulen als Ergebnis zu erhalten.

```
1 SELECT DISTINCT ?instance
2     WHERE {
3         {   ?instance A dbpedia-owl:University.
4             ?instance dbpedia-owl:country dbpedia:Germany.
5             ?instance foaf:homepage ?h.
6         } UNION {
7             ?instance A dbpedia-owl:University.
8             ?instance dbpprop:country dbpedia:Germany.
9             ?instance foaf:homepage ?h.
```

```
10          } UNION {
11              ?instance A dbpedia-owl:University.
12              ?instance dbpprop:country "Germany"@en.
13              ?instance foaf:homepage ?h.
14          }}
```

Listing 6.1: SPARQL-Anfrage, um alle deutschen Universitäten aus der englischen DBpedia zu extrahieren

6.1.2 Experimente

Für jede Universität im Datensatz wurden alle numerischen Eigenschaften extrahiert. Außerdem wurde der Typ jeder Instanz (`rdf:type`) als Mengenargument eingesetzt. Aus den Textattributen wird die jeweilige Textlänge extrahiert. Da die Universitäten sehr viele Attribute besitzen, ist die Anzahl der Dimensionen für das Clusterverfahren zu hoch. Infolgedessen wurden nur die Attribute betrachtet, die bei mindestens 50% der Instanzen existent sind. Der Goldstandard entstand durch eine manuelle Evaluierung der Instanzen.

6.1.2.1 Ermittlung des idealen λ

Der genetische Algorithmus dient der Gewichtung der einzelnen Dimensionen im Clusterverfahren. In diesem Fall wurde sie auf 1 oder 0 begrenzt, sodass eine Dimension an- oder ausgeschaltet werden kann. Ziel ist, wie bereits beschrieben, herauszufinden welche Dimensionen (bzw. Metriken) das beste Ergebnis ($F_2 = 0,98$) erzielen. Um den λ-Vektor zu bestimmen, wurde eine 10-fold Cross Validation durchgeführt. Letztere besteht aus 10 Teilexperimenten, bei denen zufällig 10% der Instanzen entfernt werden. Diese 10% sind aber so ausgewählt, dass das Verhältnis der fehlerhaften zu den richtigen Instanzen dasselbe ist, wie im gesamten Datensatz. Anschließend wird der restliche Datensatz geclustert. Das Clusterverfahren erfolgt mittels des genetischen Algorithmus: Für jede Dimension wird zufällig gewählt, ob diese vom Clusteralgorithmus betrachtet wird. Es entsteht ein möglicher λ-Vektor. Der Clusteralgorithmus berechnet alle Cluster und das mit den meisten als fehlerhaft markierten Instanzen wird bestimmt. Mit Hilfe des Goldstandard werden Precision und Recall bestimmt. Die richtig positiven Instanzen sind die, die im Goldstandard als fehlerhaft markiert sind. Aus Precision und Recall wird der F_2-Wert berechnet. Erreicht der F_2-Wert eine vordefinierte Schwelle, wird der genetische Algorithmus unterbrochen. Als letzter Schritt wird für jede Instanz aus den

zuvor entfernten 10% bestimmt, welchem Cluster sie angehört. Eine Instanz wird zum Cluster hinzugefügt, zu denen Zentroid sie am nähesten liegt. Nach dieser Bestimmung wird für die 10% ein eigener F_2-Wert berechnet.

Aus dem gesamten Experiment entstehen 10 λ-Vektoren und für jeden dieser Vektoren zwei F_2-Werte.

6.1.2.2 Validierung des λ

Aus den 10 im vorherigen Experiment ermittelten λ wird der mit den besten F_2-Werten ausgewählt. Aus Abbildung 6.1 kann abgelesen werden, dass λ_6 die besten Ergebnisse erzielt hat. Dieser λ soll für den gesamten Datensatz validiert werden. Dafür werden erneut 10% der Daten zufällig ausgewählt und die restlichen 90% geclustert. Das Clusterverfahren erfolgt jedoch ohne genetischen Algorithmus. In der Abstandsfunktion 1 fließt der λ_6-Vektor ein. Nachdem dieser Schritt erfolgt ist, wird der Abstand zwischen jeder Instanz der vorher ausgewählten 10% zum Cluster, der die meisten vom Goldstandard als fehlerhaft markierten Instanzen enthält, gemessen. Diese Abstände werden mit dem Vermerk, ob die zugehörige Instanz im Goldstandard liegt oder nicht ausgegeben. Das Experiment wird 100 mal wiederholt.

6.1.3 Ergebnisse

Abbildung 6.1 zeigt die in 6.1.2.1 ermittelten F_2-Werte für jeden Vektor. Abbildung 6.2 zeigt eine ROC-Kurve der Validierungsergebnisse des λ. Für jede der 100 Durchgänge wurde eine Kurve gezeichnet. Abbildung 6.2 zeigt die gemittelte Kurve und die Vertrauensintervalle.

6.1.4 Interpretation

Das erste Experiment 6.1.2.1 zeigt, dass manche λ in Bezug auf 90% der Daten sehr gute Ergebnisse erzielen können, jedoch nicht für den Rest der Daten. Der KMeans++-Algorithmus hat allerdings ebenfalls einen Einfluss. Je nachdem wie die Zentroiden während des ersten Schrittes des Algorithmus gewählt werden, kann es passieren, dass die restlichen Daten schlecht abschneiden, obwohl mit demselben λ bessere Ergebnisse erzielt werden könnten.

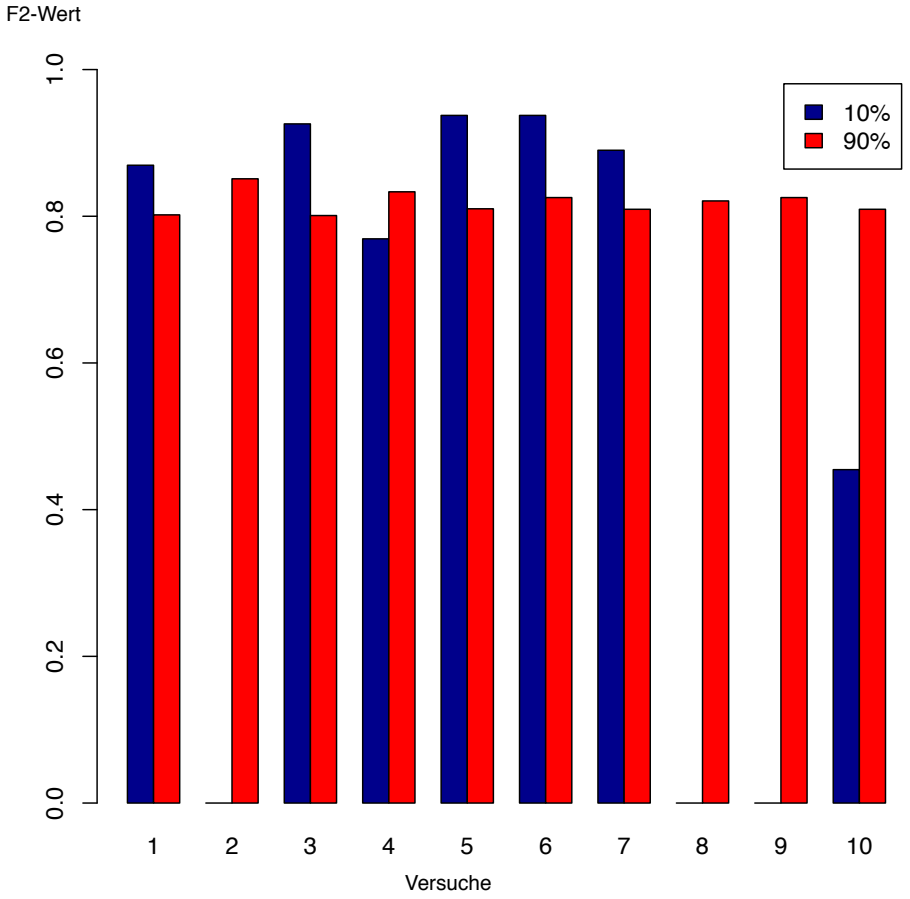


Abbildung 6.1: F_2-Werte

Die ROC-Kurve (Abbildung 6.2) verdeutlicht, dass die Ergebnisse nicht zufällig sind. Sie zeigt aber auch, dass der Einfluss des Clusteralgortihmus relativiert werden kann. Es ist möglich mit dem genetischen Algorithmus einen λ zu finden, das unabhängig von dem Anteil der gewählten Daten, die Testdaten richtig clustert.

Tabelle 6.1 bildet ab, welche Metriken verwendet worden sind, um beim Clustern die erhaltenen Ergebnisse zu erzielen. Die ausgewählten Properties sind alle, außer `rdfs:comment`, aus dem Namespace `dbpprop`. Diese Properties werden beim Import der Daten aus Wikipedia bei Seiten, die keine bzw. eine unzureichende Infobox besitzen, verwendet. Bei der manuellen Markierung der falschen Instanzen wurden Instanzen als falsch gekennzeichnet, die keine Property `dbpedia-owl:country` besitzen. Diese `dbpedia-owl:country` wird zwar beim Clustern nicht betrachtet, jedoch die Property `dbpprop:country`. Tabelle 6.1 zeigt, dass es eine numerische und eine Mengenme-

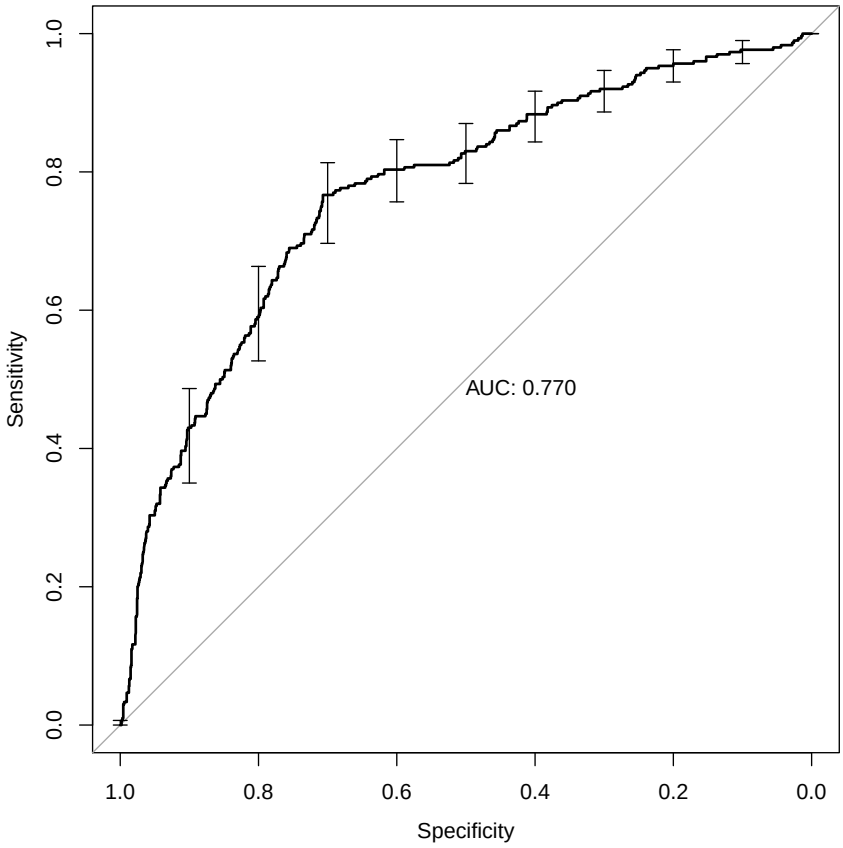


Abbildung 6.2: ROC-Kurve

trik für die Property `dbpprop:country` gibt. Die numerische Metrik existiert, weil einige Instanzen eine Zeichenkette statt der Instanz `dbpedia:Germany` besitzen. Der λ-Vektor enthält diese Metrik und somit können die Universitäten, die nicht die Instanz `dbpedia:Germany` als Land anzeigen, erkannt werden. Tabelle 6.2 stellt die typischen Fehler für jede Property dar.

Die beschriebenen Experimente zeigen, dass es auf diese Weise möglich ist, den Goldstandard zu erweitern. Jedoch werden keine Fehlertypen entdeckt, die nicht in den im Goldstandard auftretenden Instanzen vorkommen. Außerdem reicht es nicht aus, bei Properties wie `rdfs:label` die Anzahl an Einträgen zu zählen. Eine Unterteilung pro Sprache ist notwendig. Der λ zeigt, dass man mittels des Verfahrens für A präzisieren kann, welche Properties eventuell falsch sind.

Property	Type	Ausgewählt
dbpprop:established	numeric	+
foaf:name	numeric	-
dbpprop:staff	numeric	+
geolocation	geopoint	-
rdfs:label	numeric	+
dbpedia-owl:abstract	numeric	-
rdfs:comment	numeric	+
dbpedia-owl:country	set	-
dbpprop:country	set	-
dbpprop:city	set	-
dbpprop:city	numeric	-
dbpprop:country	numeric	+
dbpedia-owl:numberOfStudents	numeric	-
dbpprop:headLabel	numeric	-
dbpprop:faculties	numeric	-
dbpprop:internationalStudents	numeric	+
dbpprop:vicepresidentAccademicAffairs	numeric	-

Tabelle 6.1: Werte des λ für das 2. Experiment 6.1.2.2

6.2 Evaluierung von B (Vollständigkeit der Properties)

Um Problem B (4.2.1) validieren zu können, sind andere Datensätze nötig.

6.2.1 Datensätze

Als Datensatz wurde der LUBM Benchmark [31] Datensatz verwendet. Der LUBM ermöglicht es Daten für eine Ontologie, die eine Universität modelliert, zu generieren. Es wurde ein Datensatz mit genau einer Universität generiert. Außerdem wurde eine Ontologie mit dem Hauptthema Reise verwendet. Diese Ontologie (später Reiseontologie genannt) besitzt Klasse wie `Hotel`, `City`, `Country`, usw.

6.2.2 Experimente

Es wurden zwei verschiedene Experimente durchgeführt. Das erste ähnelt einem Unit-Test und soll nur die korrekte Funktionsweise des Algorithmus zeigen.

Property	Typische Fehler
dbpprop:established dbpprop:internationalStudents dbpprop:staff	Werten wie 10 oder 16, textuelle Werten statt numerische.
rdfs:label rdfs:comment	Nur der Anzahl an Einträge ändert sich, keine deutlichen Fehlern.
dbprop:country	"Germany"@en statt `dbpedia:Germany`

Tabelle 6.2: Typische Fehler bei den ausgwählten Properties

6.2.2.1 Korrekheit

Der LUBM generiert zufällige Daten, die fehlerfrei sind. Um das Verfahren testen zu können, wurde eine fehlerhafte Instanz per Hand hinzugefügt. Listing 6.2 zeigt diese Instanz. Sie unterscheidet sich nur in der doppelten Telefonnummer.

```
1  @prefix ns0: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#> .
2  @prefix ns1: <http://www.Department2.University0.edu/AssociateProfessor7/> .
3  @prefix ns2: <http://www.example.org/> .
4  @prefix ns3: <http://www.Department2.University0.edu/AssistantProfessor3/> .
5  @prefix ns4: <http://www.Department2.University0.edu/Lecturer1/> .
6  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
7  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
8  @prefix ns7: <http://www.Department2.University0.edu/> .
9
10 ns1:Publication5 ns0:publicationAuthor ns2:GraduateStudentB .
11 ns3:Publication1 ns0:publicationAuthor ns2:GraduateStudentB .
12 ns3:Publication8 ns0:publicationAuthor ns2:GraduateStudentB .
13 ns4:Publication1 ns0:publicationAuthor ns2:GraduateStudentB .
14 ns2:GraduateStudentB rdf:type ns0:GraduateStudent .
15 ns2:GraduateStudentB ns0:name "GraduateStudentB"^^xsd:string ;
16  ns0:telephone "xxx-xxx-xxxx"^^xsd:string ;
17  ns0:telephone "yyy-yyy-yyyy"^^xsd:string ;
18  ns0:emailAddress "GraduateStudentB@Department2.University0.edu"^^xsd:string .
19 ns2:GraduateStudentB ns0:takesCourse ns7:GraduateCourse19 ,
20                 ns7:Course23 ;
21  ns0:memberOf <http://www.Department2.University0.edu> ;
22  ns0:undergraduateDegreeFrom <http://www.University587.edu> ;
23  ns0:advisor ns7:AssociateProfessor10 .
```

Listing 6.2: Hinzugefügte Instanz für das Problem B

Anschließend wurden die vorberechneten Metriken mit WEKA [32] geclustert. Als Algorithmus wurde DBSCAN mit einem Abstand von 0.9 und einer Mindestanzahl an Elementen pro Cluster von 6 verwendet.

6.2.2.2 Reiseontologie

Die Firmenontologie mit dem Schwerpunkt Reise soll auf fehlerhafte Daten überprüft werden. Für diese Teilaufgabe wurde das Konzept `country` gewählt. Die Metriken wurden zunächst nur statistisch ausgewertet. Es wurden beispielweise Berechnungen von Aussagen durchgeführt wie 10% der Instanzen besitzen zwei mal die Property `Adresse`. Anschließend wurden die Metriken mit DBSCAN geclustert. Als Parameter wurden ein ϵ von 0.9 und ein $minPts$ von 100 gewählt. Die Auswahl der Parametern erfolgte empirisch, so dass die Anzahl an Clustern und Ausreißern gering bleibt.

6.2.3 Ergebnisse

Das erste Experiment diente der Überprüfung der Korrektheit des Algorithmus. Abbildung 6.3 zeigt die möglichen Werte für alle Metriken. Wie das Plot zeigt, sind nur die Metriken `telephone`, `teacherOf` und `headOf` genutzt werden. Der DBSCAN-Algorithmus erstellt vier Cluster. Diese unterscheiden sich in den Metriken `teacherOf` und `headof`. Als Ausreißer wurden zwei Instanzen markiert. Davon entspricht der fehlerhaften Instanz mit zwei Telefonnummern. Die andere ist die einzige, die ein `teacherOf`-Wert von vier und ein `headof`-Wert von eins hat. Dies ergibt eine Precision von 0.5 und einen Recall von 1.

Die Evaluierung der Reiseontologie erfolgt auf zwei verschiedene Weisen. Als erstes wurden die Metriken berechnet, anschließend einfache Statistiken berechnet und diese per Hand angeschaut. Als Fehler konnte bei mehreren Metriken festgestellt werden, dass eine Property, die genau einen Wert haben sollte, mehrere oder keinen hatten. Zum Beispiel haben 4.78% der Länder zwei Longituden. Dieser Fehlertyp konnte an sieben verschiedenen Properties festgestellt werden. Da es relativ mühsam wäre, die Fehler in der Weise zu suchen, und den im Kapitel 4 definierten Anforderungen nicht entspräche, wurden die Metriken mit DBSCAN geclustert. Als Parameter wurden erneut ein $epsilon$ von 0.9 und ein $minPts$ von 6 gewählt. Der Cluster-Algorithmus lieferte drei verschiedene Cluster. Außerdem wurden 79 Instanzen als Ausreißern erkannt. Tabelle 6.3 zeigt die Werten der verschiedenen Metriken für jedes Cluster und die Ausreißer. Von den ausreißern sind 41 tatsächlich fehlerhaft. Das ergibt eine Precision von 0,51. Da die ine den Cluster enthaltene Instanzen alle richtig sind, ist der Recall eins.

Metrik	Cluster 1	Cluster 2	Cluster 3	Noise
rdfs:comment	1	1	1	0-2
skos:prefLabel	3	3	2	1-3
rdfs:label	4	3	2	1-9
geo:long	1	1	1	0-2
geo:lat	1	1	1	0-2
:population	1	1	1	0-2
purl:description	1	1	1	0-2
dbo:areaTotal	1	1	1	0-2
:iso31661Alpha2Code	1	1	1	0-3
geo:lat_long	1	1	1	0-2
skos:hiddenLabel	0	0	0	0-9
skos:altLabel	1	0	0	0-6
unister-owl:iso31662Code	0	0	0	0-2
:cityId	0	0	0	0-3
:stateId	0	0	0	0-1

Tabelle 6.3: Metriken und deren Werte in den verschiedenen Clustern (Experiment 6.2.2.2)

6.2.4 Interpretation

Das erste Experiment (6.2.2.1) zeigt, dass das Verfahren erwartungsgemäß funktioniert. Der Recall von 1 beweist, dass alle fehlerhaften Instanzen erkannt werden. Zusätzlich verdeulticht die niedrigere Precision, dass potentielle Fehler auf diese Weise erkannt werden können. Die Ergebnisse sind abhängig von den Parametern des DBSCAN. Wird zum Beispiel der $minPts$ von zwei auf sechs erhöht, werden nur drei Cluster gebildet. Der verschwundene Cluster enthält nur vier Instanzen, die alle als Ausreißer markiert werden, was die Precision auf 0,14 sinken lässt. Bei diesem Experiment handelt es sich um Testdaten und wir wissen, dass wir genau eine Instanz suchen. Dementsprechend soll DBSCAN zwei ähnliche Instanzen bereits als Cluster erkennen und $minPts$ soll auf zwei gesetzt sein. Bei reellen Daten ist es wichtig zu wissen, ob es viele kleine Gruppen an ähnlichen Instanzen gibt.

Der Wert von ϵ ist schwierig einzuschätzen. Da es sich aber bei den Metriken nur um natürliche Zahlen handelt, ist es möglich, die maximale Anzahl an Unterschieden zwischen zwei Instanzen mittels ϵ einzugrenzen. Ein $epsilon < 1$ ermöglicht nur Instanzen mit gleichen Werten zusammen zu clustern. $1 < \epsilon < \sqrt{2}$ lässt maximal bei einer Metrik eine Differenz von eins zu.

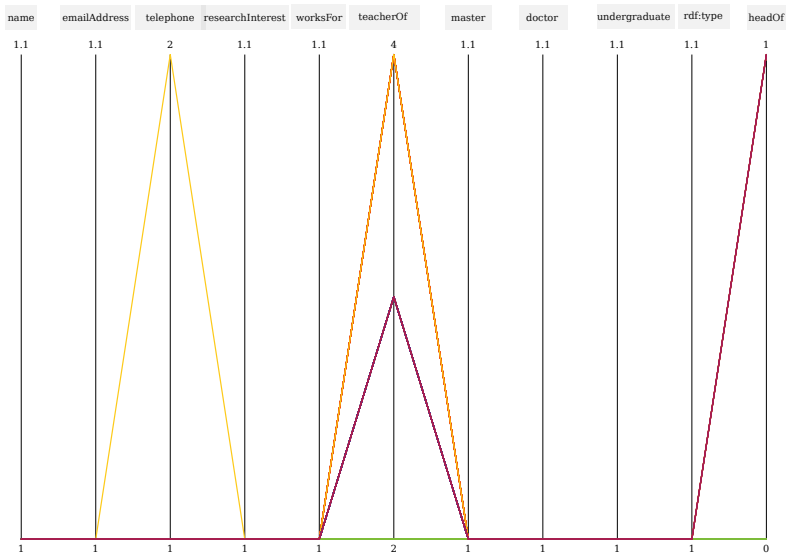


Abbildung 6.3: Parallel Plot der LUBM Daten verwendet für das Unit-Test Experiment (6.2.2.1)

Das zweite Experiment (6.2.2.2) zeigt, dass auch mit reellen Daten das Verfahren funktioniert. Als Ausreißer werden die Instanzen markiert, bei denen entweder eine Property fehlt, oder doppelt vorhanden ist. Die richtigen Instanzen erzeugen die Kriterien, um festzustellen, welche Properties vorhanden sein sollen und in welcher Anzahl. Interessant ist, dass die Instanzen, die zwei Koordinatenpaare besitzen, eine höhere Anzahl an `rdfs:label` besitzen. Dies lässt vermuten, dass zwei verschiedene Objekte bei Erstellung der Ontologie zusammengeführt wurden. Außerdem sind die falsch positive Instanzen, Instanzen die Properties besitzen, die die Mehrheit der Instanzen nicht haben.

Die gefundenen Fehler wären auch mit *Databugger* entdeckt worden, vorausgesetzt die richtigen Test werden geschrieben. Der Vorteil, des in dieser Arbeit vorgestellten Verfahrens ist, dass Vorschläge für solche Tests gemacht werden. Nach der Evaluierung der potentiell fehlerhaften Instanzen, können für die Properties, die zu richtigen Fehlern geführt haben, Tests geschrieben werden.

6.3 Evaluierung von C (Semantische Korrektheit der Properties)

Bei der Evaluierung dieses Falls werden erneut zwei verschiedene Datensätze verwendet. Die LUBM-Ontologie und die Reiseontologie (6.2.1) fanden erneut Verwendung.

6.3.1 Experimente

Als erstes soll die Korrekheit des Verfahrens überprüft werden. Anschließend soll es mit reellen Daten arbeiten und darauf folgend evaluiert werden. Die hierfür verwendeten Experimente werden in den nächsten Abschnitten vorgestellt.

6.3.1.1 Korrektheit

Wie bei Experiment 6.2.2.1 müssen fehlerhafte Daten im LUBM Datensatz hinzugefügt werden. Es wurden drei Instanzen des Typs `GraduateStudent` erzeugt. Diese unterscheiden sich in der Property `takesCourse` von den anderen. Die neu erzeugten Absolventen besuchen Kurse, die anderen verfolgen Graduiertekurse. Listing 6.3 zeigt eine dieser drei Instanzen.

```
1  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
3  @prefix ns0: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#> .
4  @prefix ns1: <http://www.Department5.University0.edu/AssociateProfessor7/> .
5  @prefix ns2: <http://www.example.org/> .
6  @prefix ns3: <http://www.Department5.University0.edu/AssistantProfessor5/> .
7  @prefix ns6: <http://www.Department5.University0.edu/> .
8  @prefix ns7: <http://www.Department5.University0.edu/Lecturer1/> .
9
10 ns1:Publication3 ns0:publicationAuthor ns2:GraduateStudentC .
11 ns3:Publication1 ns0:publicationAuthor ns2:GraduateStudentC .
12 ns3:Publication7 ns0:publicationAuthor ns2:GraduateStudentC .
13 ns2:GraduateStudentC rdf:type ns0:GraduateStudent .
14 ns2:GraduateStudentC ns0:name "GraduateStudentC"^^xsd:string ;
15  ns0:telephone "xxx-xxx-xxxx"^^xsd:string ;
16  ns0:emailAddress "GraduateStudentC@Department5.University0.edu"^^xsd:string .
17 ns2:GraduateStudentC ns0:takesCourse ns6:Course11 ;
18  ns0:memberOf <http://www.Department5.University0.edu> ;
19  ns0:undergraduateDegreeFrom <http://www.University657.edu> ;
20  ns0:advisor ns6:FullProfessor3 .
21 ns7:Publication2 ns0:publicationAuthor ns2:GraduateStudentC .
```

Listing 6.3: Beispiel einer hinzugefügten fehlerhaften Instanz

Nach Berechnung der Metriken wurden sie mit DBSCAN geclustert. Als Parameter wurden $\epsilon = 1$ und $minPts = 2$ verwendet. Die Auswahl der Parameter erfolgte nach denselben Kriterien wie im Abschnitt 6.5.3 beschrieben.

6.3.1.2 Reiseontologie

Das Verfahren soll erneut mit Hilfe der Reiseontologie auf reelle Daten geprüft werden. Dafür wurde die Klasse `City` ausgewählt. Nach Import und Berechnung der Metriken, wurden diese in zwei verschiedenen Varianten geclustert. Als erstes Clusterverfahren wurde erneut das statistische Aufteilen verwendet, als zweites DBSCAN mit den Parametern $\epsilon = 0.9$ und $minPts = 6$.

6.3.2 Ergebnisse

Das erste Experiment soll die Korrektheit des Verfahrens für diesen Fall überprüfen. Das Ergebnis des Clusterns sind 26 Cluster und vier als Ausreißer erkannte Instanzen. Unter den vier Ausreißern sind die drei fehlerhaften Instanzen. Die vierte Instanz ist der einzige Doktorand mit zwei Doktorvätern. Diese Instanz ist nicht fehlerhaft, wird aber als Ausreißer erkannt, weil sie die einzige mit derartigen Werten ist. Dies wirft aber auch die Frage auf, ob die anderen Instanzen vollständig sind. Gibt es tatsächlich nur einen Doktorand mit zwei Doktorväter oder fehlt bei einigen Doktoranden der zweite Betreuer? Der Recall beträgt 1 und die Precision 0.75.

Beim zweiten Experiment konnte mittels des statistischen Aufteilens erkannt werden, dass nur eine einzige Instanz die Property `locatedIn` in einem Objekt des Typs `unextendedPlace` besitzt. Die Klasse `unextendedPlace` repräsentiert Orte, die keine ausgedehnten Flächen besitzen. Zum Beispiel ist der Hauptbahnhof einer Stadt eine Instanz dieser Klasse, da es unwahrscheinlich ist das eine Stadt in einem Bahnhof liegt somit ist die tatsächliche Instanz fehlerhaft.

Mit DBSCAN wurden ähnliche Ergebnisse erzielt. Die Cluster enthalten nur Instanzen mit gleichen Metriken. Es entstanden 69 Clusters. Als Ausreißer wurden 67 Instanzen markiert, davon sind 51 tatsächlich fehlerhaft. Unter denen war auch die vorherige Instanz die in einem Objekt des Typs `unextendedPlace` lokalisiert ist. Tabelle 6.4 listet die verschiedenen Fehlern, die an den Ausreißern gefunden wurden. Die Tabelle beschreibt in welchem Objekt Typ die Instanzen lokalisiert sind und wie oft pro Instanz die Property mit der Klasse vorkommt. Die Summe der Betroffenen Instanzen ist höher, als die

rdf:type	Vorkommen	Betroffene Instanzen
Continent	2	6
Country	>1	24
City	1-2	20
Island	>1	12
Airport	1	1
enextendedPlace	1	1

Tabelle 6.4: Fehlertypen bei der Property `locatedIn`

Anzahl der Ausreißern. Dies ist zu erklären, dadurch dass manchen Instanzen von zwei Fehler Typen betroffen sind.

6.3.3 Interpretation

Wie bei der Vollständigkeit der Properties (6.2.4), zeigt das erste Experiment, dass das Verfahren funktioniert. Die Auswahl der Parameter ist von denselben Effekten beeinflusst. Das zweite Experiment verdeutlicht, dass das Verfahren angewendet auf reelle Daten auch funktioniert. Das Clustern mittels DBSCAN hat den Vorteil, konkrete Instanzen und potentielle Fehlern vorschlagen zu können. Diese potentiellen Fehler sind zwar mit der statistischen Aufteilung auch erkennbar, es erfordert aber eine gute Kenntniss der Daten. Die Ausreißer von DBSCAN sind konkrete Instanzen und die Richtigkeit kann auch von Nicht-Experten überprüft werden. Mit einer mehrfach Validierung ist es sogar möglich, die Glaubwürdigkeit der Evaluierungen zu testen. Außerdem ermöglicht diese Methode Fehlern an anderen Instanzen zu erkennen. Die Instanz, die in einem Objekt der Klasse `City` sich befindet, ist eigentlich in einem Objekt der gleichzeitig eine Insel und eine Stadt ist lokalisiert. Letzteres ist wahrscheinlich falsch und dieser Fall zeigt, dass die Qualität einer Instanz durch Ihre Nachbarn beeinflusst wird.

6.4 Evaluierung von D (Konsistenz der Werte)

Die Korrektheit des Verfahrens kann ohne Experiment gezeigt werden. Die Metriken sind in diesem Fall keine aggregierten oder berechneten Werte, sondern die direkten Werte der einzelnen numerischen Datatypeproperties. Ein Experiment mit reellen Werten wurde jedoch durchgeführt. Dafür wurden alle Instanzen der Klasse `hotel` betrachtet. Nach Exportierung der Metriken wurden diese erneut mit DBSCAN geclustert. Die hohe Anzahl

an Instanzen (336000) erzeugte eine längere Laufzeit des Algorithmus. Der Clusterframework *ELKI* konnte jedoch diese Aufgabe stemmen.

6.4.1 Ergebnisse

Das Ergebnis des Clustern ergab 300 Cluster und 19000 Ausreißer. Um bessere Werte zu erzielen, wurden die Metriken normalisiert, sodass jede Metrik zwischen 0-1 verläuft. Damit ist die Schätzung des ϵ einfacher. Für $\epsilon = 0.9$ und $\epsilon = 0.5$ existierte nur ein Cluster und keine Ausreißer. $\epsilon = 0.3$ ergab ein Cluster und fünf Ausreißer. Diese fünf Ausreißer haben gemeinsam, dass die Property `lowrate` sehr hohe Werte erweist. Die Property beschreibt den niedrigsten Preis des Hotels. Da die Währung nicht einheitlich ist, ist es nicht möglich, eine Aussage über deren Richtigkeit zu treffen.

6.4.2 Interpretation

Die Clusteranzahl der Rohmetriken (nicht normalisiert) ist kontraproduktiv. Die Versuche auf normalisierten Daten mit verschiedenen ϵ Werten, ergab keine Besserung. Es wird damit deutlich, dass die Werte allein, ohne einen Auswahlmechanismus wie beim Goldstandard 4.1, keine aussagekräftigen Ergebnisse erzielen. An diesem Punkt ist mehr Wissen im System nötig.

Die Werte der Literale einzeln zu betrachten und auf ein bestimmtes Range zu überprüfen ist beispielweise mit Databugger möglich. Diese Tests ergeben jedoch keine Aussagen über Ausreißer. So wäre zum Beispiel ein 40 Quadratmeter große Wohnung in Leipzig mit einer Kaltmiete von 1000€ ein Ausreißer. Es existieren zwar Wohnungen von 40qm und einige die eine Kaltmiete von 1000€ haben, die Kombination ist jedoch äußerst unwahrscheinlich.

6.5 Evaluierung von E (Korrelation)

Um die Korrektheit des Korrelationsfalls überprüfen zu können, ist ein Datensatz der eine Korrelation enthält, nötig. Obwohl in der Medizin und Biologie viele Korrelationsstudien über verschiedene Parameter existieren, sind die Datensätze nicht öffentlich. Deswegen mussten Datensätze generiert werden. Dafür wurden zufällig drei polynomielle Funktionen erstellt: eine des 1., des 2. und des 3 Grades. Für jede dieser Funktionen

f_1, f_2, f_3 wurden fehlerfreie Instanzen generiert. Diese wiederum haben zwei Properties: x und y. Für einen zufälligen Wert von X wurde $y = f(x) + z$ berechnet, wobei $-5 < z < 5$. Die fehlerhaften Instanzen wurden mit $z \in \{-100..-5\} \cup \{5..100\}$ gleichzeitig berechnet.

6.5.1 Experimente

Die generierten Daten wurden verwendet, um drei Experimente durchzuführen. Diese drei Experimente korrespondieren zu den Graden der Datensätze. Anschließend erfolgte das Clustern mit Hilfe der Sprungmethode (5.3.4.2).

6.5.2 Ergebnisse

Um die Ergebnisse zu evaluieren, wurden diese gegen einen Zufallsklassifier getestet. Der Zuffallsklassifier ordnet zufällig, die Instanzen zu den Klassen richtig oder falsch zu. Anschließend wurden die Ergebnisse des Experiments und des Zufallsklassifier statistisch überprüft. Dafür wurde der T-Test verwendet. Als Nullhypothese galt, dass die Ergebnisse beider Methoden gleich sind. Für alle drei Experimente ergab sich ein P-Wert von $1,406 \cdot 10^{-08}$. Da der P-Wert sehr niedrig ist, kann die Nullhypothese abgelehnt werden. Das Verfahren ist also signifikant besser als ein Zufallsklassifier.

6.5.3 Interpretation

Die Wahl der Clustermethode beeinflusst die Ergebnisse. Während die Quantil-Methode die Kenntnis der Fehlerquote voraussetzt, ist die Sprungmethode Extrema empfindlich. Schätzt man die Fehlerquote auf 5%, sind es jedoch 10%, sodass viele Fehler unentdeckt bleiben. Andererseits zwingt eine zu hohe Schätzung zu vielen unnötigen Überprüfungen.

Ist ein extremer Ausreißer bei der Sprungmethode enthalten, wird nur dieser als fehlerhaft erkannt. Ein möglicher Ausweg ist es nicht nur ein Sprung zu bestimmen sondern mehrere. Anschließend sollen erst die Instanzen, die sich nach dem höchsten Sprung befinden evaluiert werden. Sind diese fehlerhaft, wird weiter mit dem zweithöchsten Sprung gearbeitet. Eine weitere Möglichkeit wäre, beide Cluster-Methoden zu kombinieren.

6.6 Evaluierung von F

Um die Korrektheit der externen Korrelation überprüfen zu können, musste erneut ein Datensatz generiert werden. Mit der externen Korrelation ist gemeint, dass ein Wert einer Instanz irgendwie in Zusammenhang steht mit dem Wert einer Nachbarinstanz. Zum Beispiel, soll der Einwohnernanzahl einer Stadt kleiner sein als der des Landes wo sie liegt. Der Datensatz beinhaltet zwei Klassen `A` und `B`. Außerdem eine Objectproperty `o` und eine Datatypeproperty `d`. Listing 6.4 zeigt das Schema des Datensatzes.

```
1 Class: A
2 Class: B
3 ObjectProperty: o
4     Domain: A
5     Range: B
6 DataProperty: d
7     Domain: A,B
8     Range: xsd:decimal
```

Listing 6.4: Schema des generierten Datensatzes für die externe Korrelation in Manchester Syntax

Als erstes wurden die fehlerfreien Instanzen generiert. Die der Klasse `A` haben einen Wert für `d`, der kleiner ist als der Wert, der über `o` verbundenen Instanz von `B`. Außerdem soll die Summe der Werte für `d` aller Instanzen von `A`, die mit einer bestimmten Instanz von `B` b verbunden sind, gleich dem Wert für `d` von b sein. Es wurden zwei verschiedene Typen von fehlerhaften Instanzen generiert. Der erste Typ entspricht Instanzen von `A`, deren Wert für `d` größer ist als der Wert der korrespondierenden Instanz von `B`. Der andere Typ deckt den Fall der Instanzen von `A` $a_1, ..., a_n$, die mit einer bestimmten Instanz von `B` verbunden sind. Die Summe der Werte von `d` der Instanzen $a_1, ..., a_n$ ist größer als der Wert von b.

6.6.1 Experimente

Für den generierten Datensatz wurden die Metriken für die externe und für die aggregierte externe Korrelation verwendet. Anschließend wurden die Instanzen über beiden Metriken geclustert. Als Clusteralgorithmus wurde erneut DBSCAN mit den Parametern $\epsilon = 0.9$ und $minPts = 6$ verwendet.

Die Klassen `City`, `Country` und `Continent` der Reiseontologie besitzen alle die Property `population`. Die Instanzen der Klasse `City` werden überprüft. Dafür wurden die

externe Korrelation und die aggregierte Korrelation getrennt voneinander geclustert. Als Cluster-Algorithmus wurde erneut DBSCAN verwendet mit $\epsilon = 0.9$ und $minPts = 6$.

6.6.2 Ergebnisse

Das erste Experiment ergab drei Cluster. Das erste enthält alle fehlerfreien Instanzen. das zweite alle Instanzen, deren Wert von `d` einzeln zu groß ist. Das letzte Cluster umfasst alle Instanzen, deren Werte als Summe zu groß sind.

Bei der externen Korrelation für die Reiseontolgie ergaben sich 8 Cluster. Keine Instanz wurde als Ausreißer erkannt. Da die Daten sehr viele Fehler enthalten konnten alle Instanzen einem Cluster zugeordnet werden. Tabelle 6.5 zeigt die Beschaffung der Cluster. Cluster null und eins entsprechen die als richtig erkannte Instanzen. Cluster zwei bis sie-

	Beschreibung	Anzahl
0	Richtige Instanzen, die in keiner Stadt liegen.	33699
1	Richtige Instanzen, die in einer Stadt liegen.	4946
2	Instanzen ohne Einwohneranzahl, die in keinem Land liegen.	308
3	Instanzen ohne Einwohneranzahl.	4422
4	Instanzen, die weder in einer Stadt, ein Land oder auf ein Kontinent liegen.	718
5	Instanzen, die auf keinem Kontinent liegen.	1029
6	Instanzen, die in einem Land liegen, der keine Einwohneranzahl hat.	8
7	Instanzen ohne Einwohneranzahl, die weder in einer Stadt, ein Land oder auf ein Kontinent liegen.	322

Tabelle 6.5: Cluster Beschreibung für die Reiseontologie mit der externen Korrelation

ben enthalten die als fehlerhaft markierten Städte. Da die Städte im Cluster sechs nicht selber fehlerhaft sind, sondern das Land in denen sie liegen, handelt es sich um falsch Positive. Dies ergibt eine Precision von 0,998 und ein Recall von 1.

Für die aggregierte Korrelation wurde nur die Städte und Länder betrachtet. Da Stadtteile auch Städte sind, ergeben sich zwei Metriken. Das Clustering ergab 7 Cluster. Tabelle 6.6 zeigt die Beschaffung der Cluster. Cluster eins und vier enthalten die Korrekten Instanzen. Bei den anderen Cluster sind viele Instanzen enthalten, die zwar richtig sind, aber deren Schwesterinstanzen falsch sind. Als Schwesterinstanz ist eine Stadt die im selben Land wie eine andere gemeint. Außerdem kann der Fehler vom Land kommen. Dement-

	Country	City	Anzahl
0	<	>	281
1	<	=	3096
2	>	<	3477
3	>	=	36263
4	<	<	329
5	=	=	1047
6	>	>	950

Tabelle 6.6: Cluster Beschreibung für die Reiseontologie mit der aggregierten externen Korrelation. <: bedeutet, dass für eine Stadt die in diesem Cluster ist, die Summe der Einwohneranzahl der Städte, die im selben Land wie diese liegen kleiner ist als die Einwohneranzahl des Landes. > und = analog

sprechend ist die Precision sehr niedrig. Da aber alle möglichen Fehlern auftreten ist der Recall 1.0.

6.6.3 Interpretation

Das Clusteralgorithmus ergab drei Cluster und keine Ausreißer. Dies hängt mit dem Parameter $minPts$ zusammen. Ist $minPts$ größer als die Anzahl an Fehlern eines bestimmten Types, dann sind die fehlerhaften Instanzen als Ausreißer erkannt. Da die Menge an fehlerhafte Instanzen im reellen Fall nicht bekannt ist, ist zu erwarten, dass die fehlerhaften Instanzen zusammen geclustert werden. Dies ermöglicht auch die Differenzierung der Fehlertypen, die sehr nützlich für die Behebung der Fehler ist.

Auf die Reiseontolgie zeigte sich, dass sehr viel verschiedene Fehlern auftreten können. Die einfache externe Korrelation ermöglicht schnell und präzis zu beurteilen ob eine Stadt fehlerhaft ist. Mit der aggregierten Korrelation ist es schwieriger die Ergebnisse zu deuten, da die Instanzen sich gegenseitig beeinflussen. Eine korrekte Stadt wird als potentiell fehlerhaft gesehen, wenn ihre Schwesterstädte oder das Land in dem sie liegt fehlerhaft ist.

7 Zusammenfassung und Ausblick

Die Datenqualität einer Ontologie ist wie bereits beschrieben sehr wichtig für die Qualität der Ergebnisse, der auf dieser Ontologie beruhenden Verfahren. Im Kapitel 4 haben wir gesehen, dass die Datenqualität viele Aspekte hat. Von allen der aufgeführten Gesichtspunkten haben wir uns auf die drei folgenden Fehlerfreiheit, Vollständigkeit und Konsistenz konzentriert. Wie Tabelle 4.1 zeigt, existieren schon viele Verfahren für die Erkennung fehlerhafter Daten. Das Ziel dieser Arbeit war allerdings ein semiautomatisches Verfahren zu entwickeln, der die Anzahl der zu prüfenden Instanzen reduziert.

Wenn ein Goldstandard vorlag, konnte gezeigt werden, dass es möglich ist, andere Instanzen, die ähnliche Fehler besitzen, zu finden. Wir konnten mit dem hiervorgestellten Verfahren eine AUC-Fläche von 0.77 erzielen. Dies zeigt, dass wir den Goldstandard erfolgreich erweitern konnten.

In den meisten Fällen liegt aber einen solcher Goldstandard nicht vor oder er enthält alle Fehler eines bestimmten Typs und kann somit nicht mehr erweitert werden. Um diese Problematik zu Lösen wurden explorative Vorgehensweise entwickelt. Einige dieser Verfahren sind mit der Anwendung des Databugger auch möglich. Die Ergebnisse dieser Arbeit ermöglichen es jedoch, Tests für den Databugger zu erstellen. Das Auffinden von Objekten, die in zwei verschiedenen Länder liegen ist mit dem Databugger nicht möglich. Im Gegensatz dazu ist es mit den in dieser Arbeit vorgestellten Vorgänge möglich solche Fälle zu lösen. Zusätzlich wurden (wenn nicht nur eine Instanz betrachtet wurde) besonders interessante Fehlern gefunden, sondern auch deren verlinkte Instanzen, so zum Beispiel die Einwohneranzahl einer Stadt im Vergleich zu der des Landes in der sie steht.

Tabelle 7.1 zeigt, wie viele potentielle Fehler je nach Verfahren entdeckt wurden. Die Zahlen zeigen, dass die Korrelation und die externe Korrelation mehr potentielle Fehler erzeugen. Dies ist der Fall, weil die anderen Verfahren (ausgenommen A) einfachere Fehler entdecken. Solche Datenfehler sind beim Datenimport leichter zu vermeiden, da sie einfacher zu überprüfen sind. Zum Beispiel kann die Anzahl an Werten bei einer Proper-

Verfahren	Precision	Recall	Datensatz
B	0,51	1	Reiseontologie
C	0,76	1	
D	niedrig	niedrig	
E	0,7	0,8	
F	0,9	1	
F aggr.	niedrig	1	

Tabelle 7.1: Precision und Recall je Verfahren

ty als Auschluss-Kriterium beim Import verwendet werden. Diese Verfahren sind sehr nützlich, um zu klären, bei welchen Properties Fehler auftreten könnten.

Es existieren Möglichkeiten, um bestimmte Fehler zu vermeiden. Als erste Regel sollte bei jeder Datatypeproperty überprüft werden, ob sie funktional ist. Mit funktional ist gemeint, dass eine Property maximal ein Wert haben kann. Bei numerischen Werten sollte im Falle von mehreren möglichen Werten überlegt werden, ob die Modellierung optimal ist. Im Rahmen dieser Arbeit konnte kein Fall gefunden werden, wo mehrere numerische Werte für eine Property sinnvoll wären. In den meisten Fällen wird der zweite Wert durch einen anderen konditioniert. Beispielweise hat eine börsenquotierte Firma nur einen Börsenwert, andere Börsenwerte sollten mit einem Zeitstempel versehen sein, da diese nicht mehr gültig sind.

Bei Properties, die viele Klassen als Range haben können, ist zu überlegen, ob Unterproperties mit dem jeweiligen Range sinnvoll wären. Zum Beispiel kann die Property `lokalisiertIn` eine Stadt, ein Land oder ein Kontinent als Range besitzen kann. Aus diesem Grunde wäre es sinnvoll, die Properties als `stadt range:Stadt`, `land range:Land`, usw. zu definieren. Somit ist es möglich, die Unterproperties in den nötigen Fällen als funktional zu deklarieren.

Die erzielten Ergebnisse sind jedoch nur ein Anfang. In vielen Fällen sind die Metriken erweiterbar. Der Goldstandardansatz könnte ergänzt werden in dem die Tiefe des CDB-Algorithmus erhöht wird. Dies bedeutet, dass nicht nur die Literale der Instanzen maßgebend sein müssen, sondern auch die verlinkten Instanzen einen Einfluß auf die Richtigkeit der betrachteten Instanz haben können. Wie Tabelle 7.1 zeigt, lohnt es sich, die Werte von verlinkten Instanzen zu überprüfen. Außerdem ermöglicht eine Metrik, die angibt, ob eine Instanz mit einer fehlerhaften Instanz verlinkt ist, eine andere Möglichkeit, mehr Wissen im Verfahren einfließen zu lassen. Aus dem einfach Beispiel der An-

wohnerzahlen einer Stadt und des Landes, können kompliziertere Metriken entwickelt werden. Beispielweise, ob die Koordinate einer Stadt in dem Land, in der sie zu finden ist, stehen. Dafür sind jedoch neue Wissensquellen nötig, die zum Beispiel die Grenzen des Landes in einem Polygon darstellen.

Diese Arbeit zeigt, dass die Entdeckung von potentiellen Fehlern in einer Ontologie mit Metriken möglich ist. Wie Tabelle 7.1 zeigt, sind einfache Metriken schon sehr Aussagekräftig. Die Betrachtung der Nachbarinstanzen ermöglicht jedoch mehr potentielle Fehlern zu finden. Je besser die potentiellen Fehlern abgedeckt werden je besser können spezifische Tests erzeugt werden.

Literaturverzeichnis

[1] *Duden online*. http://www.duden.de/woerterbuch 4

[2] ACHTERT, Elke ; HETTAB, Ahmed ; KRIEGEL, Hans-Peter ; SCHUBERT, Erich ; ZIMEK, Arthur: Spatial Outlier Detection: Data, Algorithms, Visualizations. In: *SSTD*, 2011, S. 512–516 34

[3] AMANN, Herbert ; ESCHER, Joachim: *Analysis I (Grundstudium Mathematik) (German Edition)*. Birkhäuser, 2006. – ISBN 3764377550 31

[4] ARTHUR, David ; VASSILVITSKII, Sergei: k-means++: The advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* Society for Industrial and Applied Mathematics, 2007, S. 1027–1035 9

[5] ARTHUR, David ; VASSILVITSKII, Sergei: k-means++: the advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA : Society for Industrial and Applied Mathematics, 2007 (SODA '07), S. 1027–1035. – http://dl.acm.org/citation.cfm?id=1283383.1283494 33

[6] BAADER, Franz ; CALVANESE, Diego ; MCGUINNESS, Deborah L. ; NARDI, Daniele ; PATEL-SCHNEIDER, Peter F.: *The description logic handbook: theory, implementation, and applications*. Cambridge university press, 2010. – ISBN 0521150116 5

[7] BANZHAF, Wolfgang ; NORDIN, Peter ; KELLLER, Robert E. ; FRANCONE, Frank D.: *Genetic programming - An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Dpunkt-verlag, 1998. – ISBN 3920993586 11

[8] BECKETT, Dave ; MCBRIDE, Brian: RDF/XML syntax specification (revised). In: *W3C recommendation* 10 (2004). http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ IV, 5

[9] BIZER, Christian ; CYGANIAK, Richard: Quality-driven information filtering using the WIQA policy framework. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 7 (2009), Nr. 1, S. 1–10 18, 19

[10] BIZER, Christian ; HEATH, Tom ; BERNERS-LEE, Tim: Linked data-the story so far. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 5 (2009), Nr. 3, S. 1–22 1

[11] BIZER, Christian ; LEHMANN, Jens ; KOBILAROV, Georgi ; AUER, Sören ; BECKER, Christian ; CYGANIAK, Richard ; HELLMANN, Sebastian: {DBpedia} - A crystallization point for the Web of Data. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 7 (2009), Nr. 3, 154 - 165. http://www.sciencedirect.com/science/article/pii/S1570826809000225 2, 15, 37

[12] BONATTI, Piero A. ; HOGAN, Aidan ; POLLERES, Axel ; SAURO, Luigi: Robust and scalable Linked Data reasoning incorporating provenance and trust annotations. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 9 (2011), Nr. 2, 165 - 201. http://www.sciencedirect.com/science/article/pii/S1570826811000394 19

[13] BRICKLEY, Dan ; GUHA, Ramanathan V.: RDF vocabulary description language 1.0: RDF schema. In: *W3C recommendation* (2004). http://www.w3.org/TR/rdf-schema/ IV, 6

[14] BÖHM, C ; NAUMANN, F ; ABEDJAN, Z ; FENZ, D ; GRUTZE, T ; HEFENBROCK, D ; POHL, M ; SONNABEND, D: Profiling linked open data with ProLOD. In: *Data Engineering Workshops ICDEW 2010 IEEE 26th International Conference on* (2010), 175–178. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5452762. ISBN 9781424465217 15, 18, 19

[15] CARROLL, Jeremy J. ; PAN, Jeff Z.: XML Schema Datatypes in RDF and OWL / W3C. 2006. – W3C Note. – http://www.w3.org/TR/2006/NOTE-swbp-xsch-datatypes-20060314/ 30

[16] CHEN, Ping ; GARCIA, Walter: Hypothesis generation and data quality assessment through association mining. In: *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on* IEEE, 2010, S. 659–666 19

[17] CORMEN, T.H.: *Algorithmen-Eine Einführung*. Oldenbourg Wissenschaftsverlag, 2010 13

[18] DRAPER, Norman R. ; SMITH, Harry ; POWNELL, Elizabeth: *Applied regression analysis*. Bd. 3. Wiley New York, 1966 35

[19] DRUMMOND, Nick ; SHEARER, Rob: The open world assumption. In: *eSI Workshop: The Closed World of Databases meets the Open World of the Semantic Web*, 2006 15

[20] ESTER, Martin ; KRIEGEL, Hans-Peter ; SANDER, Jörg ; XU, Xiaowei: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD* Bd. 96, 1996, 226–231 10

[21] FAWCETT, Tom: ROC graphs: Notes and practical considerations for researchers. In: *Machine Learning* 31 (2004), S. 1–38 14

[22] FERBER, Reginald: *Information Retrieval*. Dpunkt.Verlag GmbH, 2003 http://information-retrieval.de/irb/ir.html. – ISBN 3898642135 12, 13

[23] FLACH, Peter: *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012. – ISBN 1107096391 8, 9

[24] FLEMMING, A.: *Quality characteristics of linked data publishing datasources.*, Humboldt-Universität zu Berlin, Diplomarbeit, 2010 18, 19

[25] FÜRBER, Christian ; HEPP, Martin: Swiqa – a semantic web information quality assessment framework. In: *ECIS*, 2011 15, 19

[26] GAMBLE, Matthew ; GOBLE, Carole: Quality, trust, and utility of scientific data on the web: Towards a joint model. In: *Proceedings of the ACM WebSci'11* (2011) 19

[27] GIL, Yolanda ; ARTZ, Donovan: Towards content trust of web resources. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 5 (2007), Nr. 4, S. 227–239

[28] GOLBECK, Jennifer ; MANNES, Aaron: Using Trust and Provenance for Content Filtering on the Semantic Web. In: *MTW*, 2006 19

[29] GRUBER, Thomas R. et al.: A translation approach to portable ontology specifications. In: *Knowledge acquisition* 5 (1993), Nr. 2, S. 199–220 4

[30] GUÉRET, Christophe ; GROTH, Paul T. ; STADLER, Claus ; LEHMANN, Jens: Assessing Linked Data Mappings Using Network Measures. In: *Proceedings of the 9th Extended Semantic Web Conference* Bd. 7295, Springer, 2012 (Lecture Notes in Computer Science), 87–102 19

[31] GUO, Yuanbo ; PAN, Zhengxiang ; HEFLIN, Jeff: LUBM: A benchmark for {OWL} knowledge base systems. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3 (2005), Nr. 2–3, 158 - 182. `http://www.sciencedirect.com/science/article/pii/S1570826805000132` 42

[32] HALL, Mark ; FRANK, Eibe ; HOLMES, Geoffrey ; PFAHRINGER, Bernhard ; REUTEMANN, Peter ; WITTEN, Ian H.: The WEKA data mining software: an update. In: *SIGKDD Explor. Newsl.* 11 (2009), November, Nr. 1, 10–18. `http://doi.acm.org/10.1145/1656274.1656278`. – ISSN 1931–0145 32, 43

[33] HARTIG, Olaf: Trustworthiness of data on the web. In: *Proceedings of the STI Berlin & CSW PhD Workshop* Citeseer, 2008 19

[34] HITZLER, Pascal ; KRÖTZSCH, Markus ; RUDOLPH, Sebastian ; SURE, York: *Semantic Web: Grundlagen (eXamen.press) (German Edition)*. Springer, 2007. – ISBN 3540339930 4

[35] HOGAN, Aidan ; HARTH, Andreas ; PASSANT, Alexandre ; DECKER, Stefan ; POLLERES, Axel: Weaving the pedantic web. (2010) 18, 19

[36] HOGAN, Aidan ; UMBRICH, Jürgen ; HARTH, Andreas ; CYGANIAK, Richard ; POLLERES, Axel ; DECKER, Stefan: An empirical survey of Linked Data conformance. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 14 (2012), Nr. 0, 14 - 44. http://www.sciencedirect.com/science/article/pii/S1570826812000352 19

[37] HRIPCSAK, George ; ROTHSCHILD, Adam S.: Agreement, the f-measure, and reliability in information retrieval. In: *Journal of the American Medical Informatics Association* 12 (2005), Nr. 3, S. 296–298 33

[38] JACCARD, Paul: *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901 31

[39] JACOBI, Ian ; KAGAL, Lalana ; KHANDELWAL, Ankesh: Rule-based trust assessment on the semantic web. In: *Rule-Based Reasoning, Programming, and Applications*. Springer, 2011, S. 227–241 19

[40] JOHN, Holland: *Holland, Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, 1992 11

[41] KONTOKOSTAS, Dimitris ; AUER, Sören ; HELLMANN, Sebastian ; LEHMANN, Jens ; WESTPHAL, Patrick ; CORNELISSEN, Roland ; ZAVERI, Amrapali: Test-driven Data Quality Evaluation for SPARQL Endpoints. In: *Submitted to 12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, 2013. – http://svn.aksw.org/papers/2013/ISWC_Databugger/public.pdf II, 16

[42] KOZA, JohnR. ; BENNETT, III ForrestH. ; STIFFELMAN, Oscar: Genetic Programming as a Darwinian Invention Machine. In: POLI, Riccardo (Hrsg.) ; NORDIN, Peter (Hrsg.) ; LANGDON, WilliamB. (Hrsg.) ; FOGARTY, TerenceC. (Hrsg.): *Genetic Programming* Bd. 1598. Springer Berlin Heidelberg, 1999, S. 93–108 32

[43] KRELL, Christoph: *MapReduce auf Graphen*. Akademikerverlag, 2013. – ISBN 978–3–639–49092–3

[44] LEI, Yuangui ; NIKOLOV, Andriy ; UREN, Victoria ; MOTTA, Enrico: Detecting Quality Problems in Semantic Metadata without the Presence of a Gold Standard. In: *Workshop on „Evaluation of Ontologies for the Web" (EON) at the WWW*, 2007, S. 51–60 19

[45] MCGUINNESS, Deborah L. ; VAN HARMELEN, Frank et al.: OWL web ontology language overview. In: *W3C recommendation* 10 (2004), Nr. 2004-03, 10. http://www.w3.org/TR/owl-features/ IV, 7

[46] MENDES, Pablo N. ; MÜHLEISEN, Hannes ; BIZER, Christian: Sieve: linked data quality assessment and fusion. In: *LWD*, 2012 19

[47] MIRKIN, Boris: *Clustering: A Data Recovery Approach, Second Edition (Chapman & Hall/CRC Computer Science & Data Analysis)*. Chapman and Hall/CRC, 2012. – ISBN 9781439838426 9, 10

[48] MOSTAFAVI, Mir-Abolfazl ; EDWARDS, Geoffrey ; JEANSOULIN, Robert et al.: An ontology-based method for quality assessment of spatial data bases. In: *International Symposium on Spatial Data Quality* Bd. 4, 2004, S. 49–66 19

[49] MOTIK, Boris ; PARSIA, Bijan ; PATEL-SCHNEIDER, Peter F.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax / W3C. 2009. – W3C Recommendation. – `http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/` 30

[50] MYERS, Jerome L. ; WELL, Arnold D.: *Research Design & Statistical Analysis*. Routledge, 2002. – ISBN 0805840370 35

[51] PRUD'HOMMEAUX, Eric ; SEABORNE, Andy et al.: SPARQL query language for RDF. In: *W3C recommendation* 15 (2008). `http://www.w3.org/TR/rdf-sparql-query/` 7

[52] QUILITZ, Bastian ; LESER, Ulf: Querying Distributed RDF Data Sources with SPARQL. Version: 2008. `http://dx.doi.org/10.1007/978-3-540-68234-9_39`. In: BECHHOFER, Sean (Hrsg.) ; HAUSWIRTH, Manfred (Hrsg.) ; HOFFMANN, Jörg (Hrsg.) ; KOUBARAKIS, Manolis (Hrsg.): *The Semantic Web: Research and Applications* Bd. 5021. Springer Berlin Heidelberg, 2008, 524-538 28

[53] SEGEN, J.C.: *The Dictionary of Modern Medicine (Dictionary Series)*. CRC Press, 1992. – ISBN 1850703213 12

[54] SHEKARPOUR, Saeedeh ; KATEBI, SD: Modeling and evaluation of trust with an extension in semantic web. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 8 (2010), Nr. 1, S. 26–36 19

[55] SMITH, Eric A. ; DUPRÉ, J: *The Latest on the Best: Essays on Evolution and Optimality*. MIT Press, 1987 11

[56] STICKLER, Patrick: Cbd-concise bounded description. In: *W3C Member Submission* 3 (2005) 29

[57] USCHOLD, Mike ; GRUNINGER, Michael et al.: Ontologies: Principles, methods and applications. In: *Knowledge engineering review* 11 (1996), Nr. 2, S. 93–136 4

[58] VERMEULEN, Allan ; BEGED-DOV, Gabe ; THOMPSON, Patrick: The pipeline design pattern. In: *Proceedings of OOPSLA'95 Workshop on Design Patterns for Concurrent, Parallel, and Distributed Object-Oriented Systems* Citeseer, 1995 27

[59] VRANDECIC, Denny ; SURE, York: LNCS 4519 - How to Design Better Ontology Metrics. (2007), S. 311–325 15

[60] WANG, Richard Y. ; STRONG, Diane M.: Beyond Accuracy. What Data Quality Means to Data Consumers. In: *Journal of Management Information Systems* 12 (1996), Nr. 4, S. 5–33. – URL http://w3.cyu.edu.tw/ccwei/PAPER/ERP/dataquality(JMIS).pdf 16, 18, 19

[61] WITMER, Gene: Ontology. In: *Dictionary of Philosophy of Mind* https://sites.google.com/site/minddict/ontology 4

[62] ZAVERI, Amrapali ; KONTOKOSTAS, Dimitris ; SHERIF, Mohamed A. ; BÜHMANN, Lorenz ; MORSEY, Mohamed ; AUER, Sören ; LEHMANN, Jens: User-driven Quality Evaluation of DBpedia. In: *To appear in Proceedings of 9th International Conference on Semantic Systems, I-SEMANTICS '13, Graz, Austria, September 4-6, 2013*, ACM, 2013. – http://svn.aksw.org/papers/2013/ISemantics_DBpediaDQ/public.pdf 15

[63] ZAVERI, Amrapali ; RULA, Anisa ; MAURINO, Andrea ; PIETROBON, Ricardo ; LEHMANN, Jens ; AUER, Sören ; HITZLER, Pascal: Quality Assessment Methodologies for Linked Open Data. In: *Submitted to SWJ* (2012). – http://www.semantic-web-journal.net/content/quality-assessment-methodologies-linked-open-data 16, 18, 19

Erklärung

„Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann“.

Ort	Datum	Unterschrift