Webanwendung zur Extraktion von Teildatensätzen aus DBpedia

Christian Ernst, Dominik Strohscheer, Hans Angermann
Till Nestler, Marvin Hofer, Robert Bielinski, Jonas Rebmann

Inhaltsverzeichnis

Modellierungsbeschreibung

Allgemeines

DBPedia enthält strukturierte Daten von Wikipedia. Dieses Dokument beschreibt das Softwareprodukt, das Daten von der DBPedia anfragt, sie auf Fehler überprüft und möglichst konsistent ausgibt, und gibt einen Überblick über den Entwurf und die Funktionalitäten des Softwareprodukts.

Produktübersicht

Das Softwareprodukt stellt eine Weboberfläche zur Verfügung, auf welcher der Benutzer die Möglichkeit hat, eine Datenauswahl vorzunehmen. Die gewählten Daten werden dann von der serverseitig installierten Software beim DBPedia-Endpunkt abgerufen und in einer lokalen Datenbank gespeichert.

Im nächsten Schritt ermöglicht das Web UI dem Benutzer, die Konsistenzfilterung zu konfigurieren. Dabei können zum Beispiel Attribute aus den vorher abgefragten Daten ausgewählt werden.

Wenn die Webanwendung diese Konfiguration erhält, dann startet sie dementsprechend die Filterung der Strukturierten Datensätze auf dem lokalen Datenbestand.

Am Ende können Statistiken, Logs und die verarbeiteten Datensätze vom Benutzer angezeigt beziehungsweise heruntergeladen werden.

Grundsätzliche Struktur und Entwurfsprinzipien

Für die Implementierung des Softwarepakets wird die Programmiersprache Ruby zusammen mit dem Framework Ruby on Rails benutzt.

Das Rails Framework basiert auf der Model-View-Controller-Architektur.

Das Model "ActiveRecord" basiert auf einer Datenbank und Methoden zur Manipulation der Daten. Der View "ActionView" besteht aus verschiedenen Templates, in deren Ausgabe Daten aus dem Model eingebunden werden. Der Datenfluss zwischen Model und View wird vom Controller, dem "ActionController" koordiniert. Der Controller steuert somit die Interaktionen zwischen der Anwendung und dem Benutzer.

Ruby bietet viele Bibliotheken, sogenannte Gems, welche Klassen und Methoden zur Nutzung in unserem Softwarepaket zur Verfügung stellen.

Mittels des Gem "sparql-client" können vorallem Sparql-Anfragen an verschiedene Endpunkte gesendet werden. Da DBPedia das Hauptaugenmerk des Projekts darstellt, wird dessen Endpunkt derzeit für die verschiedenen Tests genutzt.

Das Sparql-Client Gem liefert Ergebnisse, welche durch das Gem "RDF.rb" in Ruby definiert sind. Dieses Gem wird im späteren Verlauf des Projekts auch noch zur Verarbeitung von RDF-Daten genutzt.

Da jede Ruby on Rails im Normalfall eine Datenbank als Backend besitzt, machen wir uns diesen Vorteil zu Nutze. Die zu verarbeitenden Datensätze können mittels ActiveRecord in einer relationalen Datenbank gespeichert und durch die Ruby-Klassen verändert werden.

Struktur und Entwursprinzipien einzelner Pakete

Releasebündel 1

Das erste Paket beinhaltete die Möglichkeit Beispielanfragen an die DBPedia zu stellen und einige Antwortsätze zu verarbeiten.

Dafür wurde eine Klasse QueryManager implementiert (Klassendiagramm im Abschnitt "Datenmodell"). Diese erlaubt es eine SPARQL-Anfrage erstellen, diese an einen SPARQL-Endpunkt schicken und sich den Antwortsatz zurückgeben zu lassen. Da maximal 10.000 Antwortsätze zurückgegeben werden, wurde die Möglichkeit zur Eingabe von Offset und Limit gegeben, um damit die Grenze von 10.000 Antwortsätzen zu umgehen.

Es wurde außerdem eine Methode implementiert, welche es ermöglicht die von einer Query erhaltenen Ergebnisse zu zählen. Dies ermöglicht außerdem die unterschiedliche Anzahl an Datensätzen vor und nach der späteren Konsistenzprüfung zu überwachen.

Die Klasse QueryManager wird für eine (Demo-)Weboberfläche benutzt, die dem DBPedia Endpunkt ähnelt. Die Antwortsätze werden noch nicht auf Fehler oder Inkonsistenzen überprüft.

Die Beispielanfrage, die auf der Weboberfläche angegeben ist, findet Filme und ihre Laufzeiten.

Releasebündel 2

Das zweite Paket erweitert das Softwareprodukt um eine Klasse zur Verarbeitung von Datentypen und beinhaltet außerdem eine überarbeitete Version der Weboberfläche um erste Konfigurationsmöglichkeiten für die Filter anzuzeigen.

Mithilfe der neuen Klasse Stats ist es nun möglich die verschiedenen Datentypen und deren Anzahl anzuzeigen, welche unter den Ergebnissen einer Sparql-Query zu finden sind. Dadurch kann später eine Übersicht an vorhandenen Datentypen erstellt werden, welche dem Benutzer der Webanwendung zur Verfügung gestellt wird um entsprechende Datensätze auszuwählen. Außerdem können mithilfe der Klasse erste Statistiken zur Anzahl der auftretenden Datentypen von Properties generiert werden.

Die Weboberfläche enthält nun ein weiteres Textfeld für Logs, welche im späteren Verlauf hinzugefügt werden. Außerdem existieren neue Schaltflächen damit im Laufe des Projekts vom Benutzer Datensätze und Datentypen ausgewählt werden können und die erhaltenen Daten auch als Download zur Verfügung stehen.

Weitere Releasebündel

Im Laufe des Projekts werden weitere Klassen zur Erweiterung der Funktionalität implementiert.

Zur Speicherung und zum Auslesen von gespeicherten RDF-Datensätzen wird eine Klasse RdfRepository erstellt, welche Datensätze aus dem QueryManager in die relationale Datenbank des Backends einlesen kann und diese auch unkompliziert wieder für die Stats-Klasse und Andere zur Verfügung stellt.

Es wird mindestens eine Filter-Klasse zur Bearbeitung der gespeicherten Datensätze benötigt, damit die Daten welche nicht dem gewählten Datentyp entsprechen aussortiert bzw. umgewandelt werden.

Die Stats-Klasse muss selbst noch erweitert werden um die erhaltenen Informationen über Datentypen zu verarbeiten und diese an die Filter-Klassen und den Controller weiterzugeben.

Zur Erstellung von Statistiken werden noch Helper-Methoden entwickelt, die es dem Controller ermöglichen entsprechende Daten an die View weiterzureichen.

Die Weboberfläche wird stets entsprechend der Erweiterung der Funktionalität erweitert und angepasst.

Benutzeroberfläche

Die Benutzeroberfläche der Anwendung sollte es ermöglichen

1. Die von der DBPedia abzufragenden Daten zu konfigurieren
2. Die Test- und Korrekturfunktionalitäten auszuwählen und ggf. zu konfigurieren
3. Inkosistenzen und Fehler an zu zeigen
4. Relevante Statistiken an zu zeigen
5. Die verarbeiteten Datensätze herunter zu laden

Der Entwurf welcher im zweiten Releasebündel realisiert *1.* durch ein einfaches Eingabefeld für SPARQL-Queries. Direkt rechts davon Informiert der Aktions Log über durchgeführte Aktionen wie die Behebung von Inkonsistenzen.

Der Druckknopf *Absenden* weist den Server an, mithilfe der angegebenen SPARQL-Afrage die Daten vom DBPedia Endpunkt anzufordern.

Anschließend könnte nach Auswahl eines Properties und einer Methode zur bereinigung von inkonsistenzen hinsichtlich der Datentypen das Programm durch tätigen der Schaltfläche *Korrigieren* gestartet werden.

Eine Methode um alle Datentypen eines bestimmten Properties an zu gleichen wäre zum Beispiel den am häufigsten auftretenden Datentyp als Zieldatentyp zu wählen. Einträge anderen Datentyps könnten mithilfe von Umrechnungsregeln in diesen Datentypen umgewandelt werden. Widersprüchliche Werte oder Werte die nicht in den gewünschten Datentyp konvertiert werden konnten sollten dann verworfen werden.

Abbildung 1: UI-Entwurf

Datenmodell

Im ersten Releasebündel wurde die Klasse QueryManager eingeführt welche grundfunktionen für die durchführung von SPARQL anfragen an den DBPedia endpunkt zur Verfügung stellt.

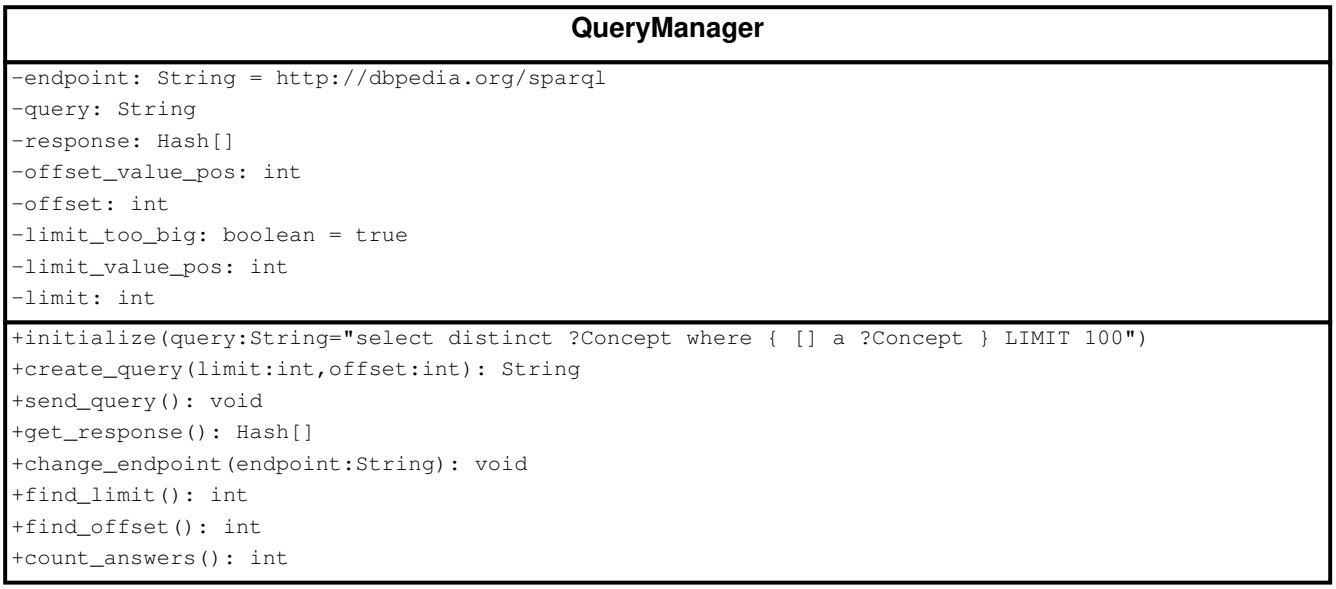

Abbildung 2: Klassendiagramm für das erste Releasebündel

Im zweiten releasebundle wurde neben der Erweiterung des bestehenden Codebase eine Klasse zur Analyse der Abgefragten Datensätze eingeführt

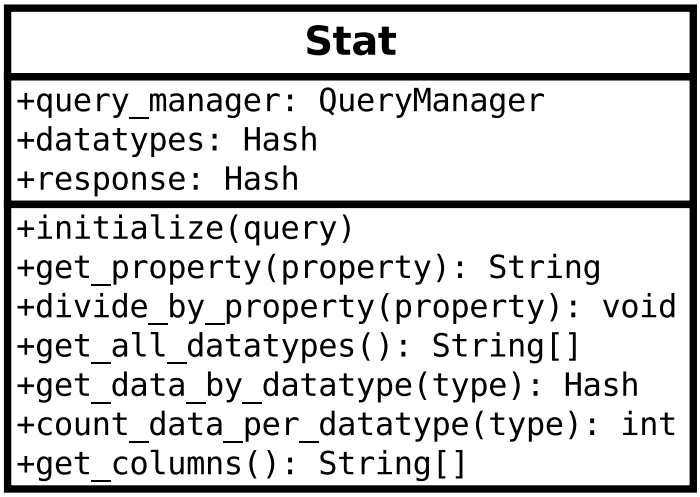

Abbildung 3: Klassendiagramm für das zweite Releasebündel

Glossar

Gem

Module und Pakete in Ruby, die über eine Softwarepaketverwaltung namens Rubygems[1] verbreitet werden können.

Hash-Klasse

Die Hash-Klasse in Ruby ist mit der Hashtable-Klasse in Java gleichzusetzen. Ein Hash ist eine Sammlung von Primärschlüsseln und deren Attributen ohne bestimmte Reihenfolge.

Limit

Das Limit ist die maximale Anzahl an Datensätzen, welche vom SPARQL-Endpoint zurückgegeben werden.

Offset

Das Offset ist die Anzahl der zu überspringenden Datensätze, welche vom Endpoint zurückgegeben werden, bevor die gewünschten Daten abgerufen werden.

Sparql-Client-Gem

Das Sparql-Client-Gem[2] ist eine Ruby Implementation eines Sparql-Clienten auf Basis des Ruby-RDF-Gems. Es ermöglicht die Ausführung von Sparql-Abfragen über HTTP.

RDF-Gem

Das RDF.rb-Gem[3] ist eine Ruby-Bibliothek, deren Klassen die Verarbeitung von RDF-Daten in Ruby erleichtern.

Fußnoten

1 https://rubygems.org/

2 https://github.com/ruby-rdf/sparql-client

3 https://github.com/ruby-rdf/rdf