Softwaretechnik-Praktikum 2016
Datum: 11. April 2016

Gruppe: wrd16
Betreuer: René Speck
Tutor: Marius Brunnert

Team:
Tim Niehoff
Felix Lange
Philip Fritzsche
Matti Wilhelmi
Simon Kaleschke
Johannes Leupold

Modellierungsbeschreibung

Inhaltsverzeichnis

1 Allgemeines

Das Framework „Axolotl“ bietet die Möglichkeit, einen im Plain-Text-Format gegeben Text mithilfe der Tools Fox für Named Entity Recognition und BOA für Relation Extraction zu analysieren. Dabei werden aus dem gegebenen Korpus Entitäten und Relationen zwischen diesen Entitäten extrahiert und die gewonnenen Daten formatiert auf der Konsole ausgegeben.

2 Produktübersicht

Die Funktionalität des Frameworks besteht in der Extraktion von Wissen in Form von Entitäten und Relationen aus einem gegebenen Text. Dafür werden die Tools Fox für Named Entity Recognition sowie BOA für Relation Extraction eingesetzt. Der Eingabetext wird in Form einer Textdatei in das Programm eingegeben und anschließend als Ganzes von den oben genannten Tools verarbeitet. Die erhaltenen Daten werden auf der Konsole aufgeteilt nach Entitäten und Relationen ausgegeben.

Der Zugriff auf die implementierten Tools ist über Interface-Klassen abstrahiert, sodass die konkret verwendeten Tools durch die Implementierung dieser Interfaces austauschbar sind.

3 Grundsätzliche Struktur- und Entwurfsprinzipien

Das System wurde modular entworfen, indem für jede Kernfunktionalität des Frameworks ein Modul geschaffen wurde. Diese sind

1. Eingabe von Textdaten
2. Named Entity Recognition
3. Relation Extraction
4. Ausgabe von RDF-Daten (bzw. annotiertem Text)

Das Hauptprogramm greift auf jede dieser Kernfunktionalitäten abstrahiert über ein Interface zu, sodass das konkret implementierte Modul über Änderungen im Hauptprogramm einfach austauschbar ist. Die Module werden sequentiell in der oben genannten Reihenfolge angefragt.

4 Struktur- und Entwurfsprinzipien einzelner Pakete

4.1 Modul Main (Hauptprogramm)

Das Main-Modul enthält die Klasse `AxolotlMain`, die den sequentiellen Programmablauf festlegt und gleichzeitig die ausführbare Hauptklasse der Software darstellt. Im aktuellen Release wird mithilfe des `PlainTextInputProvider`s lediglich eine Zeichenkette eingelesen.

Ein `RdfModel` wird initialisiert. Anschließend werden Entitäten durch den `FoxNerProvider` herausgesucht und im `RdfModel` gespeichert. Darauffolgend werden die durch den `BoaReProvider` gefundenen Relationen ebenfalls im `RdfModel` gesichert. Die `AxolotlMain` beinhaltet im aktuellen Relase die Ausgabe des `RdfModel`s über die Konsole.

Geplante Veränderungen Zukünftig soll der `PlainTextInputProvider` der `AxolotlMain` eine Liste von Sätzen übergeben, die von der `AxolotlMain` Satz für Satz wie oben beschrieben bearbeitet werden soll. Außerdem sollen in einer der nächsten Releasebündel die Provenancen der NER- und RE-Provider in dem RdfModel gespeichert werden. Zukünftig soll die `AxolotlMain` das RdfModel an `OutputProvider` übergeben, der für die Ausgabe der gefundenen Entitäten und Relationen sorgt.

4.2 Modul Input

Das Input-Modul sieht die Funktion loadFile() vor, um den Inhalt einer Datei in Form von Sätzen in das Programm zu laden. Der Rückgabetyp dieser Funktion ist eine `ArrayList` von `String`s.

Im aktuellen Release ist ein `PlainTextInputProvider` implementiert, der die Textdaten aus einer Plain-Text-Datei einliest und nicht in Sätze zerlegt, sondern den ganzen Text als eine Zeichenkette speichert.

Geplante Veränderungen Zukünftig wird der `PlainTextInputProvider` in der Lage sein, die gegebene Plain-Text-Datei in Sätze zu zerlegen und diese als ArrayList von Sätzen als String zu speichern.

4.3 Modul NER (Named Entity Recongition)

Das Interface zum Zugriff auf das NER-Modul bietet eine einzelne Methode an, die zum Finden von Entitäten verwendet wird. Diese Methode ist als thread-sicher auszulegen und liefert eine `ArrayList` von `Entity` zurück.

Der Zugriff auf das Tool Fox findet über eine speziell entwickelte API statt, die Anfragen an einen laufenden Fox-REST-Service stellt.

4.4 Modul RE (Relation Extraction)

Das Interface des Moduls für Relation Extraction besteht ebenso nur aus einer Methode, die bei Eingabe eines Satzes als `String` und einer `ArrayList` aus `Entities` eine `ArrayList` aus `Relations` zurückgibt. Da die Verarbeitung der Sätze unabhängig voneinander funktioniert, kann auch hier Thread-Sicherheit garantiert werden. Damit die Übergabe von Entitätstypen gelingt, wird ein Interface zum Abbilden von Typen zwischen Modul NE und Modul RE bereitgestellt.

Aktuell ist die Klasse `BoaReProvider` implementiert, die Pattern des BOA-Index nutzt, um Relationen in Sätzen zu finden. Zwischen den aktuell implementierten NER (Fox) und RE (Boa)-Tools wurde die Klasse `FoxBoaTypeMapProvider` implementiert, die das Abbilden von Entitätstypen übernimmt, um die Kompatibilität zwischen Fox und Boa zu garantieren.

Geplante Veränderungen Zukünftig soll die Klasse parallelisierbar gemacht und weitergehend optimiert werden, um eventuelle Schreibfehler in der Eingabe tolerieren zu können.

4.5 Modul Output

Das Output-Modul hat nur eine Funktion namens `writeFile`, die eine Ausgabe von in einem Apache-Jena-Graph gegebenen RDF-Daten auf ein beliebiges Ziel zu schreiben. Aktuell ist eine Ausgabe von `RDF/TURTLE` auf der Konsole implementiert.

In der aktuellen Version findet die Datenausgabe nicht über das Output-Modul statt, sondern im Main-Modul (zu Testzwecken.)

Geplante Veränderungen Zukünftig soll der `ConsoleOutputProvider` die Ausgabe übernehmen. Außerdem soll eine Klasse zur Ausgabe von `RDF/XML` in eine Datei geschaffen werden.

5 Datenmodell

5.1 Der Datentyp `Entity`

Der `Entity`-Datentyp stellt eine Entität mit ihrem konkreten Bezug zum Eingabetext dar. Dazu wird ein `label` der Entität, die sprachliche Repräsentation dieses Objektes (z.B. „Leipzig“), ein `type` (Entitätsklasse) des Objektes (z.B. „LOCATION“), die Position der Entität im Text (`index`) sowie die URI einer Resource, die die selbe Entität beschreibt (`sameAs`), gespeichert. Die Position der Entität im Text ist als eine geordnete Menge (`TreeSet`) ausgelegt, sodass mehrere Vorkommen derselben Entität in demselben Text unterstützt werden.

Die Eigenschaften des `Entity`-Typs (außer `provenance`) sind direkt auf bestimmte RDF-Prädikate abgebildet:

`label`	`http://www.w3.org/2000/01/rdf-schema#label` (`rdfs:label`)
`type`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#type` (`rdf:type`)
`index`	`http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#beginIndex` (`nif:beginIndex`)
`sameAs`	`http://www.w3.org/2002/07/owl#sameAs` (`owl:sameAs`)

Geplante Veränderungen Es ist vorgesehen, die Provenance des erzeugenden Tools in der Entität abzulegen (`provenance`).

5.2 Der Datentyp `Relation`

Der `Relation`-Datentyp besteht aus den drei Entitäten:

- Subjekt
- Prädikat
- Objekt

Die Prädikat-Entität beschreibt dabei einen konkreten Zusammenhang zwischen der Subjekt- und der Objekt-Entität. Somit entsteht nach dem RDF-Datenmodell ein Tripel (Subjekt, Prädikat, Objekt).
In der aktuellen Version wird das Pattern, aus dem die Relation gewonnen wurde, als Label des Prädikates und der Relationstyp als Type gesetzt.
Beispiel:
Aus dem Satz: „Albert Einstein wurde geboren in Ulm“, aus dem bereits das Subjekt „Albert Einstein“ und das Objekt „Ulm“ gewonnen wurden, beschreibt der Teil „wurde geboren in“ den Relationstypen „Geburtsort“. So entsteht das Tripel (Albert Einstein, Geburtsort, Ulm).

5.3 Der Datentyp `Provenance`

Der `Provenance`-Typ ist im aktuellen Release ohne Funktion.

Geplante Veränderungen Die Provenance soll nach den Vorgaben der PROV-Ontologie abgelegt werden. Dafür muss ein Datenmodell entwickelt werden.

5.4 Das Datenmodell `RdfModel`

Das `RdfModel` hinterlegt die gefundenen Entitäten und Relationen als RDF-Tripel. Hierfür wird das Apache Jena-Model verwendet, welches Entitäten und Relationen als Resources und ihre oben beschriebenen Eigenschaften als Properties speichert. Entitäten werden mit der addEntities(), Relationen mithilfe der addRelations() Methoden hinzugefügt.

Geplante Veränderungen Zukünftig sollen auch als Eigenschaft von Entitäten die Provenancen hinterlegt werden, um zu erkennen, durch welche Tools diese gefunden worden.

6 Glossar

Entität Eine Entität ist ein eindeutig identifizierbares, konkretes oder abstrakes Objekt der realen Welt oder unserer Anschauung.

Interface (Java) Ein Interface stellt eine Schnittstelle dar. Eine Schnittstelle besteht nur aus Methodendeklarationen, die niemals ausformuliert werden (abstrakte Methoden). Klassen können Interfaces implementieren, indem sie die abstrakten Methoden implementieren. Interfaces dienen der Abstraktion, damit der Zugriff auf bestimmte Funktionalitäten von der konkret implementierten Klasse abstrahiert werden kann.

Named Entity Recognition (NER) Named Entity Recognition ist ein Teilgebiet der Wissensextraktion. Hierbei werden meist unannotierte Textblöcke genommen und in annotierte Textblöcke umgewandelt. Dabei können im Text Personen, Unternehmen, Zeitangaben etc. aufgefunden werden. Die Ausgabe kann dann weiterverwendet werden.

Fox ist ein Tool für NER, welches als REST-Service angefragt werden kann. (siehe `http://fox-demo.aksw.org`)

Plain-Text Plain-Text beschreibt unformatierten Text, in diesem Dokument bezieht sich der Begriff immer auf in UTF-8 codierte Textdateien.

Relation Eine Relation (Beziehung) besteht zwischen zwei Entitäten und beschreibt eine statische Beziehung dieser Entitäten zueinander oder eine Interaktion miteinander.

Relation Extraction (RE) RE ist wie NER ein Teilgebiet der Wissensextraktion, bei dem aus einem annotierten Text und einer Wissensdatenbank aus bekannten Relationen neue Beziehungen zwischen Entitäten abgeleitet werden.

BOA ist ein Tool, was dieses Vorgehen implementiert, indem es DBPedia als Wissensdatenbank nutzt.

REST-Service Representational State Transfer (abgekürzt REST, seltener auch ReST) bezeichnet ein Programmierparadigma für verteilte Systeme, insbesondere für Webservices. REST-Schnittstellen werden mittels HTTP angefragt und bieten eine zustandslose, einheitliche Schnittstelle an..