Softwaretechnik-Praktikum
2016
Datum: 17. Mai 2016

Gruppe: wrd16
Betreuer: René Speck
Tutor: Marius Brunnert

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

Entwurfsbeschreibung

Inhaltsverzeichnis

1 Allgemeines

Das Framework „Axolotl“ bietet die Möglichkeit, einen im Plain-Text- oder NIF-Format gegebenen Text mithilfe der Tools Fox für Named Entity Recognition und BOA oder Patty für Relation Extraction zu analysieren. Dabei werden aus dem gegebenen Korpus Entitäten und Relationen zwischen diesen Entitäten extrahiert und in einem oder mehreren der Serialisierungsformate `N3`, `JSON-LD`, `N-Triples`, `TriG`, `TURTLE`, `RDF/JSON`, `RDF/XML` in eine Datei sowie in allen genannten Formaten außer `N-Triples` und `RDF/JSON` auf die Konsole ausgegeben. Die genannten Tools können durch Implementierung von Interfaces einfach ausgetauscht werden.

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 oder Patty für Relation Extraction eingesetzt. Der Eingabetext wird (in den bereits implementierten Modulen) in Form einer Textdatei oder einer NIF-Datei im Format `TURTLE` in das Programm eingegeben, in Sätze zerlegt und anschließend in den einzelnen Sätzen von den oben genannten Tools verarbeitet. Die erhaltenen Daten werden in den oben genannten Formaten in Dateien oder auf die Konsole ausgegeben. Es ist mit den bereits implementierten Tools die Verarbeitung von Texten in den Sprachen Deutsch, Englisch und Französisch möglich.

Der Zugriff auf die einzelnen Module ist über Interfaces abstrahiert, sodass die konkret verwendeten Tools durch die Implementierung dieser Interfaces austauschbar sind. Dadurch können der Funktionsumfang von AXOLOTL sowie zum Beispiel die unterstützten Sprachen deutlich erweitert werden. Im finalen Release ist es möglich, beliebige Klassen, die die betreffenden Interfaces implementieren, zu laden und für die Verarbeitung zu nutzen.

Am Ende des Analysevorgangs wird zusätzlich Provenance zu den Daten im Format der PROV-Ontologie abgelegt. Aus diesen Daten kann entnommen werden, mit welchen Tools die Ausgabedaten konkret erzeugt wurden, woraus Rückschlüsse auf die Qualität der Daten gezogen werden können.

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. Hinzufügen von Provenance
5. Ausgabe von RDF-Daten (bzw. annotiertem Text)

Das Hauptprogramm greift auf jede dieser Kernfunktionalitäten abstrahiert über ein Interface zu, sodass das konkret verwendete Modul durch Implementierung des zugehörigen Interfaces einfach austauschbar ist. Die Module werden sequentiell in der oben genannten Reihenfolge angefragt. In der aktuellen Version wird der Klassenpfad nach sämtlichen Klassen durchsucht, die die benötigten Interfaces implementieren. Diese können über Optionen auf der Kommandozeile geladen werden. Steht keine Implementierung für ein bestimmtes Interface zur Verfügung, kann AXOLOTL nicht ausgeführt werden. Steht nur genau eine Implementierung zur Verfügung, so wird diese automatisch gewählt und muss nicht explizit angegeben werden.

3.1 Parallelisierung

Das Framework bietet eine Möglichkeit, `for-` und `forEach`-Schleifen parallel auszuführen. Dadurch ist es möglich, die Verarbeitung verschiedener Sätze parallel ablaufen zulassen, was zu einem deutlichen Geschwindigkeitsgewinn führen kann, sofern die verwendeten NER- bzw. RE-Tools eine Parallelisierung nicht verhindern.

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 finalen Release werden einzelne Sätze mithilfe eines frei wählbaren `InputProvider` eingelesen.

Ein `RdfModel` wird initialisiert. Anschließend werden Entitäten durch einen beliebigen `NerProvider` herausgesucht und im `RdfModel` gespeichert. Darauffolgend werden die durch einen ebenfalls beliebigen `ReProvider` gefundenen Relationen ebenfalls im `RdfModel` gesichert.

Die `AxolotlMain` beinhaltet im finalen Relase die Ausgabe des `RdfModel`s mittels eines oder mehrerer beliebiger `OutputProvider`.

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. Im Interface `InputProvider` steht die Hilfsmethode `sentenceSplitting` zur Verfügung, die den vorher ausgelesenen Inhalt des Korpus tokenisiert und in Sätze aufteilt. Der Rückgabetyp ist somit eine `ArrayList` von `String`s.

Im finalen Release ist ein `PlainTextInputProvider` und `NifInputProvider` implementiert, der die Textdaten aus einer `Plaintext-` bzw. NIF-`ttl`-Datei einliest und in Sätze aufteilt.

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. Die aktuelle Implementierung ist thread-sicher und nutzt, falls der einprogrammierte oder vom Nutzer angegebene Fox-Service nicht erreichbar ist, einen Fallback auf die Fox-Demo der Universität Leipzig.

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.

Für das finale Release sind die Klassen `BoaReProvider` und `PattyReProvider` implementiert, um Relationen in Sätzen zu finden. Dabei benutzt der `BoaReProvider` einen Index, auf dem entweder nach Pattern oder nach zwei benachbarten Entitäten gesucht werden kann. Für die Typenübergabe besteht der `FoxBoaTypeMapProvider`. Der `PattyReProvider` benutzt mehrere einfache Dateien, in denen nach Pattern gesucht werden kann. Für die Typenübergabe wurden die Klassen `DbpediaMapper` und `DbpediaHelper` implementiert, die effizient prüfen können, ob Entitätstypen auf verschiedenen Abstraktionsebenen zusammenpassen.

4.5 Modul Provenance

Das Provenance Modul besteht aus zwei Methoden: die Methode `add` fügt Provenance-Informationen zum `ProvenanceHandler` hinzu, welcher die Daten dann mit `writeToGraph` in das `RdfModel` schreibt. Die Provenance Daten werden dabei mit der PROV-Ontologie verknüpft.

4.6 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 schreibt. Im finalen Release ist eine Ausgabe in Dateien, auf die Konsole oder in einen String in mehreren Serialisierungsformaten implementiert (siehe oben).

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, der Getter liefert ein geordnetes Array zurück. Es kann die Eigenschaft der Endposition im Text mittels `getEndIndex` abgefragt werden, diese Anfrage liefert ebenfalls ein geordnetes Array.

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`)
`endIndex`	`http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#endIndex` (`nif:endIndex`)
`sameAs`	`http://www.w3.org/2002/07/owl#sameAs` (`owl:sameAs`)

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 speichert für jedes Tool, also für jedes Interface Provenance Informationen. Dabei wird der Name des Tools, die Version und (falls benötigt) ein Kommentar gespeichert.

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`-Methode hinzugefügt.

Das `RdfModel` kann in einem separaten Thread ausgeführt werden, wodurch die Möglichkeit entsteht, aus verschiedenen parallel laufenden Satzverarbeitungs-Threads synchron Entitäten und Relationen in das `RdfModel` einzufügen. Läuft der Model-Worker-Thread, so können die Methoden `addEntities`- und `addRelations`-Methoden nicht aus einem anderen Thread und die `getModel`- und `output`-Methoden überhaupt nicht aufgerufen werden, um eine Kompromittierung der Daten zu verhindern. Stattdessen steht dann zum Hinzufügen von Entitäten und Relationen die Methode `addLater` zur Verfügung, die die gewonnenen Daten zwischenspeichert, bis der Thread sie verarbeiten kann.

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.

PROV Ontologie Die Prov Ontologie beschreibt das Prov Datenmodell in OWL, um Metadaten über die verwendeten Tools, Daten und Vorgänge in einem RDF Graph speichern zu können.

NLP Interchange Format (NIF) Das NLP Interchange Format (NIF) ist ein Format, in dem natursprachliche Texte abgelegt werden können. Es bietet Attribute zur Beschreibung von Sätzen, Wörtern, Phrasen,... und ihre Beziehungen untereinander (z.B. ein Wort gehört zu einem Satz). Das NIF-Format wird in der Regel in RDF/TURTLE umgesetzt.

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..