AutoSPARQL

Let Users Query Your Knowledge Base

Christian Olczak

Seminar aus maschinellem Lernen WS 11/12
Fachgebiet Knowledge Engineering
Dr. Heiko Paulheim / Frederik Janssen

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

- Warum AutoSPARQL?
- Query Trees
- QTL-Algorithmus
- Demo
- Evaluation
- Fazit

Warum AutoSPARQL?

TECHNISCHE UNIVERSITÄT DARMSTADT

Let Users Query Your Knowledge Base

Warum AutoSPARQL?

TECHNISCHE UNIVERSITÄT DARMSTADT

Ausgangssituation:

- Mehr als 100 SPARQL Endpoints mit Milliarden von Tripeln.

Ziel:

- Wissen soll einfach abfragbar und nutzbar sein.

Problem:

- Kenntnisse der Syntax und Struktur der Knowledge Base (KB) werden benötigt.

Warum AutoSPARQL?

TECHNISCHE UNIVERSITÄT DARMSTADT

Bisherige Verfahren:

Knowledge Base Specific Interfaces

+ schirmen den Nutzer von der Komplexität der KB ab
+ decken meist die wichtigsten Queries ab
- Queries müssen im voraus bekannt sein
- nicht flexibel im Bezug auf Änderungen und Erweiterungen

Facet-Based Browsing (z.B. Neofonie Browser[1])

+ Nutzer kann Restriktionen festlegen
+ nicht KB spezifisch
- Probleme bei komplexeren Queries

[1] http://dbpedia.neofonie.de/

Warum AutoSPARQL?

TECHNISCHE UNIVERSITÄT DARMSTADT

Visual SPARQL Query Builders (z.B. SPARQL Views[1])

+ vereinfachte Erstellung von SPARQL Queries
- Nutzer braucht Kenntnisse in SPARQL
- Nutzer braucht Kenntnisse des Schemas

Question Answering Systems (z.B. Ginseng[2])

+ Fragen in natürlicher Sprache
- Komplexe Fragen schwer zu beantworten

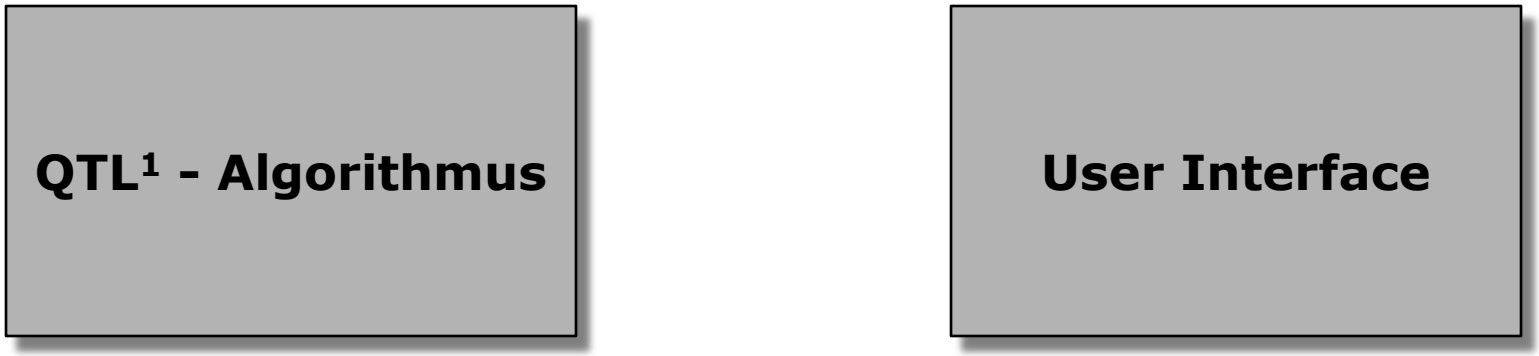
[1] http://drupal.org/project/sparql_views
[2] https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/talking-to-the-semantic-web/ginseng/index.html

Warum AutoSPARQL?

TECHNISCHE UNIVERSITÄT DARMSTADT

AutoSPARQL

Aktives, überwachtes maschinelles Lernen mit positiven und negativen Beispielen, um SPARQL Queries zu erzeugen.



[1] QTL = Query Tree Lerner

Warum AutoSPARQL?

TECHNISCHE UNIVERSITÄT DARMSTADT

Ablauf:

1. Nutzer stellt Frage in natürlicher Sprache oder gibt konkrete Ressource an.
   Bsp.: „all films starring Brad Pitt“ oder „Brad Pitt“

2. Nutzer bekommt Liste mit Vorschlägen von AutoSPARQL.

3. Nutzer wählt positive Beispiele aus.

4. AutoSPARQL stellt Nutzer Fragen, ob bestimmte Ressourcen zur Antwort gehören.

5. Abbruch durch den Nutzer, wenn die gewünschte Query gelernt ist.

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

- Warum AutoSPARQL?
- Query Trees
- QTL-Algorithmus
- Demo
- Evaluation
- Ausblick & Fazit

Query Trees

TECHNISCHE UNIVERSITÄT DARMSTADT

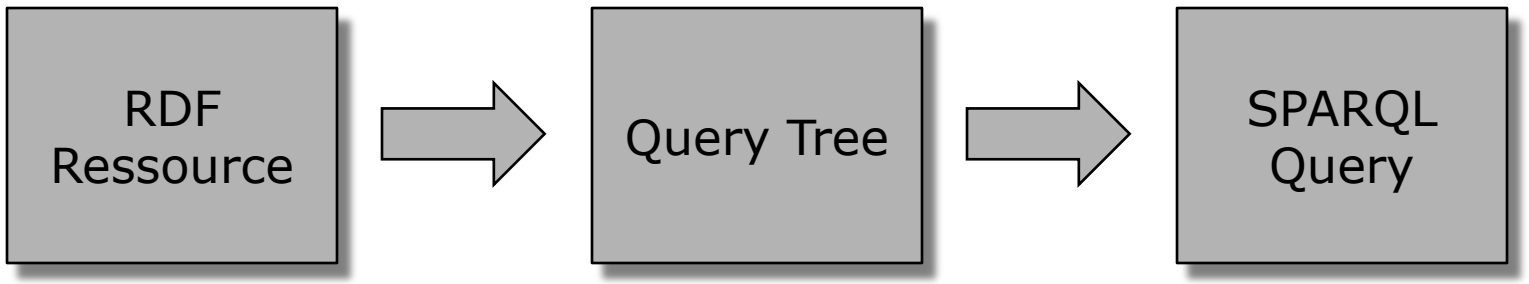
- Datenstruktur die innerhalb des Algorithmus genutzt wird
- gerichteter, gelabelter Baum mit Wurzel und keinen Zyklen
- Die Tiefe des Baums ist die Länge des längsten Pfades

T	Query Tree (T = (V,E,ℓ))
V	Endlichen Menge von Knoten
E	Endliche Menge von Kanten (E ⊂ V x R x V)
ℓ	Labelling Funktion (ℓ: V -> NL)
R	Menge von RDF Ressourcen
L	Menge von RDF Literalen
NL	Menge von Knoten Labels (NL = L ∪ R ∪ {?})

Query Trees

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jeder Query Tree entspricht einer SPARQL Query.
- Die Umkehrung gilt nicht.
- Query Trees bilden Brücke zwischen den RDF Ressourcen und einer SPARQL Query.



Query Trees

TECHNISCHE UNIVERSITÄT DARMSTADT

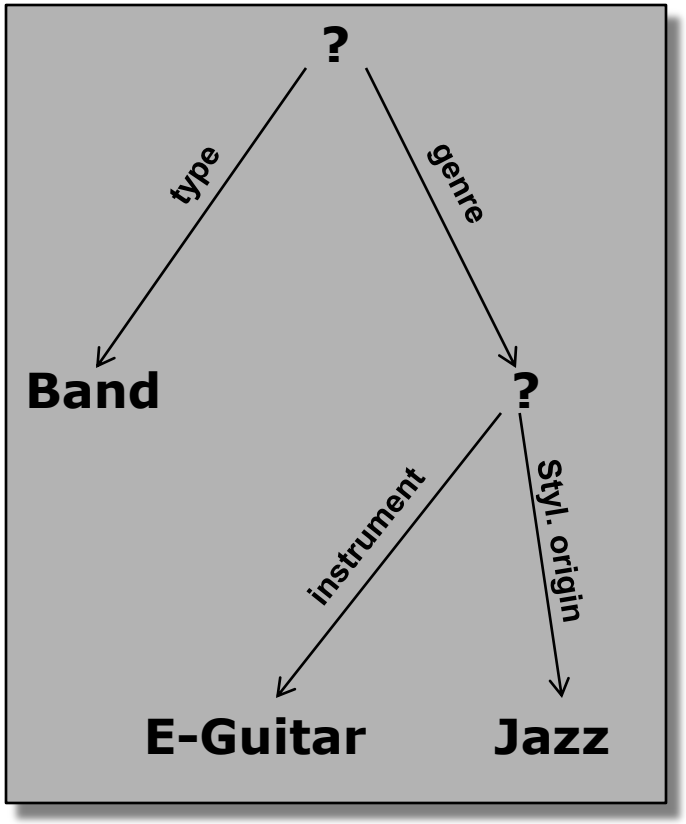
Mapping von RDF Ressourcen auf Query Trees

- Jede Ressource eines RDF Graphen lässt sich auf einen Query Tree abbilden.
- Der Baum richtet sich nach der Nachbarschaft der Ressource.
- Suche nur bis zu einer bestimmten Tiefe, aus Gründen der Effizienz.

Query Trees

TECHNISCHE UNIVERSITÄT DARMSTADT

Mapping eines Query Tree auf eine SPARQL Query:




```
SELECT ?x0 WHERE {
?x0 rdf:type dbo:Band.
?x0 dbo:genre ?x1.
?x1 dbo:instrument dbp:Electric-guitar.
?x1 dbo:stylisticOrigin dbp:Jazz.
}
```

Query Trees - Operationen

TECHNISCHE UNIVERSITÄT DARMSTADT

Query Tree Subsumption

Definition:
T_1 und T_2 seien Query Trees. T_1 wird von T_2 subsumiert, wenn die Ergebnisse die die SPARQL Query von T_1 liefert, eine Untermenge der Ergebnisse der SPARQL Query von T_2 bilden.

Query Trees - Operationen

TECHNISCHE UNIVERSITÄT DARMSTADT

Least General Generalisation (LGG)

- Aus zwei Query Trees (T_1,T_2) wird ein neuer Query Tree (T) erzeugt.
- T subsumiert danach sowohl T_1 als auch T_2

Query Trees - Operationen

TECHNISCHE UNIVERSITÄT DARMSTADT

Negative Based Reduction (NBR)

- Wird benutzt um einen Query Tree zu verallgemeinern.
- Ziel ist es Kanten aus einem Query Tree zu entfernen.
- Dies geschieht anhand der negativen Beispiele.

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

- Warum AutoSPARQL?
- Query Trees
- QTL-Algorithmus
- Demo
- Evaluation
- Fazit

QTL-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1 query = "SELECT ?x0 { ?x0 ?y ?z . ";
2 nodequeue = [root(T)];
3 while nodequeue not empty do
4 |   v = poll(nodequeue) // pick and remove first node ;
5 |   foreach edge (v, e, v') in E(T) do
6 |   |   query += mapedge((v, e, v')) ;
7 |   |   if ℓ(v') =? then add v' at the end of nodequeue
8 query += "}";
9 return query
```

Function `sparql(T)`

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

- Warum AutoSPARQL?
- Query Trees
- QTL-Algorithmus
- Demo
- Evaluation
- Fazit

Agenda

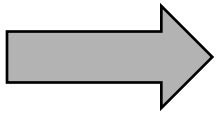
TECHNISCHE UNIVERSITÄT DARMSTADT

- Warum AutoSPARQL?
- Query Trees
- QTL-Algorithmus
- Demo
- Evaluation
- Fazit

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Getestet mit dem Benchmark Data Set des „1st Workshop on Question Answering over Linked Data“ (QALD)
- 50 Fragen an DBpedia und die entsprechenden Antworten
- Auswahl der Fragen die mindestens 3 Ressourcen zurück liefern
- Entfernen der Queries die SPARQL Konstrukte enthalten die von AutoSPARQL nicht unterstützt werden.



15 Fragen in natürlicher Sprache

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

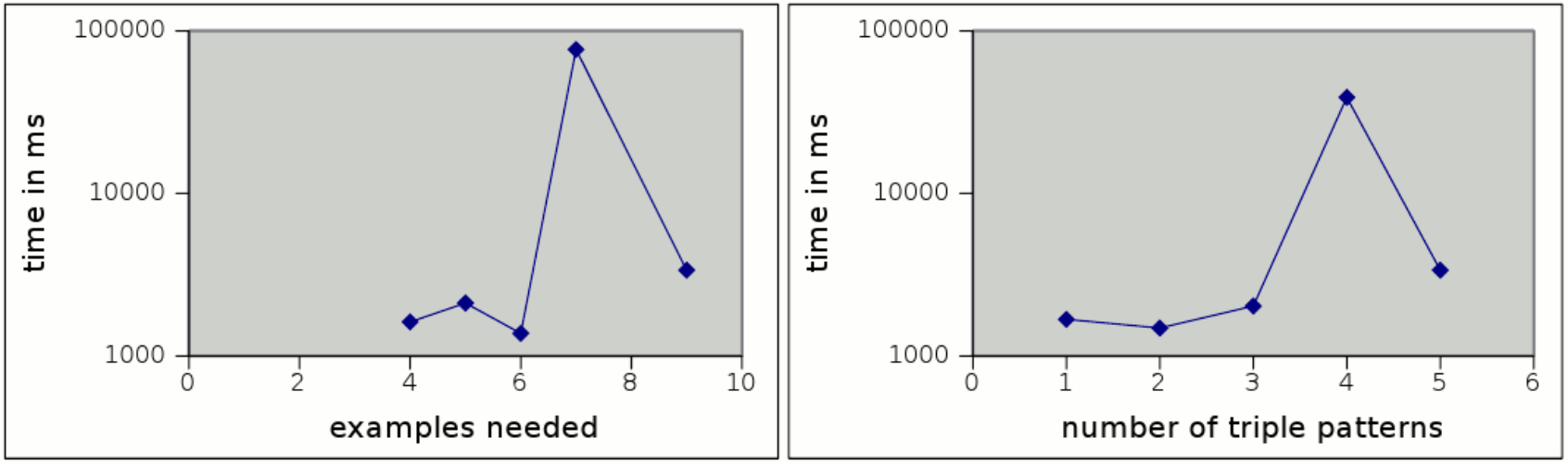
- Höchstens drei positive Beispiele wurden pro Frage aus den ersten 20 Suchergebnisse ausgewählt.
- Das aktive Lernen begann immer mit drei positiven und einem negativen Beispiel.

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fragen:
  - Wie viele Beispiele werden benötigt, um die Queries zu lernen?
  - Ist die Performanz von AutoSPARQL ausreichend?
- Ergebnis:
  - 4-9 Beispiele wurden benötigt (durchschnittlich 5)
  - bis zu 77 Sekunden wurden benötigt um eine Query zu lernen (durchschnittlich 7 Sekunden)

Evaluation



- Weitere Ergebnisse
  - Die benötigte Zeit pro neuem Beispiel bleibt ungefähr konstant.
  - Keine Korrelation zwischen der Länge der Query und der Lerndauer.

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

- Warum AutoSPARQL?
- Query Trees
- QTL-Algorithmus
- Demo
- Evaluation
- Fazit

Fazit

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Was behaupten die Autoren?**
  - AutoSPARQL ist effizient.
    - Subset von SPARQL
    - Query Trees
  - Es werden nur wenige Fragen benötigt um eine Query zu lernen.
  - AutoSPARQL kann nicht mit fehlerhaften Daten umgehen, weist aber den Nutzer auf mögliche Fehler hin.

Fazit

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Eigene Erfahrungen**

AutoSPARQL

- AutoSPARQL funktioniert gut bei einfachen Fragen
  Bsp.: „films directed by …" oder „books written by …"
- Bei komplexeren Fragen häufig Fehlermeldungen und Probleme

Paper

- Teilweise nicht ausführlich genug
  - Mapping von Ressourcen auf Query Trees
  - Negative Based Reduction
- Evaluation könnte umfangreicher sein (nur 15 Fragen)

Fragen

TECHNISCHE UNIVERSITÄT DARMSTADT

Vielen Dank für Ihre Aufmerksamkeit!

Anhang

TECHNISCHE UNIVERSITÄT DARMSTADT

Abbildungen und Algorithmen aus:

J. Lehmann und L. Bühmann. AutoSPARQL – Let Users Query Your Knowldge Base. In Proceedings of ESWC 2011. 2011

Anhang

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition 1 (Query Tree). *A* query tree *is a rooted, directed, labelled tree* $T = (V, E, \ell)$*, where* V *is a finite set of* nodes*,* $E \subset V \times R \times V$ *is a finite set of* edges*,* $NL = L \cup R \cup \{?\}$ *is a set of* node labels *and* $\ell : V \to NL$ *is the* labelling function*. The* root *of* T *is denoted as* $root(T)$*. If* $\ell(root(T)) =?$*, we call the query tree* complete*. The set of all query trees is denoted by* $\mathcal{T}$ *and* $\mathcal{T}_C$ *for complete query trees. We use the notions* $V(T) := V$*,* $E(T) := E$*,* $\ell(T) := \ell$ *to refer to nodes, edges and label function of a tree* T*. We say* $v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_n} v_{n+1}$ *is a* path *of length* n *from* v_1 *to* v_{n+1} *in* T *iff* $(v_i, e_i, v_{i+1}) \in E$ *for* $1 \leq i \leq n$*. The depth of a tree is length of its longest path.*

Anhang

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition 2 (Subtrees as Query Trees). *If* $T = (V, E, \ell)$ *is a query tree and* $v \in V$*, then we denote by* $T(v)$ *the tree* $T' = (V', E', \ell')$ *with* $root(T') = v$*,* $V' = \{v' \mid$ *there is a path from* v *to* $v'\}$*,* $E' = E \cap V' \times R \times V'$ *and* $\ell' = \ell_{|V'}$*.*

Definition 3 (Node Label Replacement). *Given a query tree* T*, a node* $v \in V(T)$ *and* $n \in NL$*, we define* $\ell[v \mapsto n]$ *as* $\ell[v \mapsto n](v) := n$ *and* $l[v \mapsto L](w) := \ell(w)$ *for all* $w \neq v$*. We define* $T[v \mapsto n] := (V(T), E(T), \ell(T)[v \mapsto n])$ *for* $v \in V(T)$*. We say that the* label of v is replaced by n*.*