Das DTA im Graphen

Contents

1 Inhalt **1**

2 Das DTA im Graphen **2**

3 Die Downloadformate des DTA **2**

4 Vorbereitungen **2**

5 Import **3**
5.1 Tokenimport 3
5.2 Satzstrukturen 3
5.3 Lemmaimport 4
5.4 Neo4j-Browser 4
5.5 Beispielabfrage 5

6 Zusammenfassung **5**

1 Inhalt

{:.no_toc}

- Will be replaced with the ToC, excluding the “Contents” header {:toc}

2 Das DTA im Graphen

Das Deutsche Textarchiv (DTA) stellt einen Disziplinen übergreifenden Grundbestand deutscher Werke aus dem Zeitraum von ca. 1600 bis 1900 im Volltext und als digitale Faksimiles frei zur Verfügung und bereitet ihn so auf, dass er über das Internet in vielfältiger Weise nutzbar ist. Das DTA-Korpus soll in größtmöglicher Breite widerspiegeln, was an bedeutenden Werken in deutscher Sprache veröffentlicht wurde. Die ausgewählten Texte stehen repräsentativ für die Entwicklung der deutschen Sprache seit der Frühen Neuzeit. Alle DTA-Texte werden unter einer offenen Lizenz veröffentlicht (CC BY-NC). Das DTA fördert die Wiederverwendung seiner Texte in allen Bereichen der Digitalen Geisteswissenschaften.

3 Die Downloadformate des DTA

Das DTA bietet zu den bereitgestellten Texten verschiedene Formate zum Download an.

- TEI-P5
- TCF
- Plain-Text

Für den Import in eine Graphdatenbanken bietet sich das TCF-Format an, da es den Text tokenisiert, serialisiert, lemmatisiert und normalisiert bietet. In diesem Format lässt er sich mit cypher-Befehlen in die Graphdatenbank importieren. Im Beispiel wird Goethes Faust in der TCF-Fassung in die Graphdatenbank importiert.

Beispiel aus der XML-Datei

4 Vorbereitungen

Als Vorbereitung müssen einige Constraints eingerichtet werden.

```
create constraint on (t:Token) assert t.id is unique;
create constraint on (s:Sentence) assert s.id is unique;
```

```
create constraint on (l:Lemma) assert l.text is unique;
```

Mit den Befehlen wird sichergestellt, dass die im nächsten Schritt importierten Knoten eindeutige IDs haben.

5 Import

5.1 Tokenimport

Nun folgt der Import-Befehl mit der apoc-procedure *apoc.load.xmlSimple*.

```
call apoc.load.xmlSimple('http://deutschestextarchiv.de/book/download_fulltcf/1618
unwind doc._TextCorpus._tokens._token as token
create (t:Token{id:token.ID, text:token._text})
with collect(t) as tokens
unwind apoc.coll.pairs(tokens)[0..-1] as value
with value[0] as a, value[1] as b
create (a)-[:NEXT_TOKEN]->(b);
```

In der ersten Zeile wird der apoc-Befehl *apoc.load.xmlSimple* aufgerufen, der als Argument die URL der TCF-Version von Goethes Faust im Deutschen Textarchiv erhält. Die weiteren cypher-Befehle parsen die XML-Datei und spielen die Token (also die einzelnen Wörter) als Wortknoten in die Graphdatenbank ein. Schließlich werden die NEXT_TOKEN-Kanten zwischen den eingespielten Wörtern erstellt.

5.2 Satzstrukturen

Der nächste Befehl lädt wieder die gleiche XML-Datei und importiert die Satzstrukturen.

```
call apoc.load.xmlSimple("http://deutschestextarchiv.de/book/download_fulltcf/1618
unwind doc._TextCorpus._sentences._sentence as sentence
match (t1:Token{id:head(split(sentence.tokenIDs, " "))})
match (t2:Token{id:last(split(sentence.tokenIDs, " "))})
create (s:Sentence{id:sentence.ID})
create (s)-[:SENTENCE_STARTS]->(t1)
```

```
create (s)-[:SENTENCE_ENDS]->(t2)
with collect(s) as sentences
unwind apoc.coll.pairs(sentences)[0..-1] as value
with value[0] as a, value[1] as b
create (a)-[:NEXT_SENTENCE]->(b);
```

5.3 Lemmaimport

Im folgenden Befehl werden die Lemmata importiert und jedes Token mit dem zugehörigen Lemma verknüpft.

```
call apoc.load.xmlSimple('http://deutschestextarchiv.de/book/download_fulltcf/1618
unwind doc._TextCorpus._lemmas._lemma as lemma
match (t:Token{id:lemma.tokenIDs})
merge (l:Lemma{text:lemma._text})
create (t)-[:LEMMATISIERT]->(l);
```

Der letzte Befehl ergänzt bei jedem Token-Knoten noch die Lemma-Information als Proptery.

```
call apoc.load.xmlSimple('http://deutschestextarchiv.de/book/download_fulltcf/1618
unwind doc._TextCorpus._lemmas._lemma as lemma
match (t:Token{id:lemma.tokenIDs}) set t.Lemma = lemma._text;
```

Damit ist nun die Fassung von Goethes Faust aus dem Deutschen Textarchiv in die Graphdatenbank importiert worden und kann weiter untersucht werden (hier klicken, um den Code mit den cypher-Querys für den gesamten Artikel herunterzuladen).

5.4 Neo4j-Browser

Der neo4j-Graphbrowser bietet einen ersten Überblick zu den in der Datenbank vorhandenen Knoten- und Kantentypen. In Abbildung 3 ist der Wortknoten Tragödie ausgewählt, so dass in der Fußleiste der Graphvisualisierung die verschiedenen Eigenschaften des Knotens angezeigt werden. Der Knoten hat beispielsweise die “45414”, die Eigenschaft Lemma hat den Wert “Tragödie” während die Eigenschaft text das Originalwort enthält.

5.5 Beispielabfrage

Bei Cypher-Abfragen können alle Eigenschaften von Knoten und Kanten miteinbezogen werden.

```
match w=()-[:NEXT_TOKEN*5]->(a:Token{Lemma:'Gestalt'})-[:NEXT_TOKEN*5]->() return
```

Die folgende Abbildung zeigt das Ergebnis der Abfrage nach allen Vorkommen von Wörtern mit dem Lemma Gestalt im Faust mit den jeweils vorhergehenden und anschließenden drei Wörtern.

Abildung mit Gestalt.

6 Zusammenfassung

Im vorliegenden Kapitel wurden die Schritte für den Import der DTA-TCF-Fassung von Goethes Faust in die Graphdatenbank neo4j vorgestellt. Die qualitativ hochwertigen Text-Quellen des Deutschen Textarchivs bieten in Verbindung mit Graphdatenbanken sehr interessante neue Möglichkeiten zur Auswertung der Texte. Durch Austausch des Links zur TCF-Fassung können auch andere Texte des DTA eingespielt werden.