Z i t a t i o n
Andreas Kuczera, Das Deutsche Textarchiv in der Graphenwelt, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 4. April 2017,
https://mittelalter.hypotheses.org/10025

Das Deutsche Textarchiv in der Graphenwelt

von Andreas Kuczera

Einleitung

Das Deutsche Textarchiv (DTA: http://www.deutschestextarchiv.de/) 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.

Was ist eine Graphdatenbank? Vom relationalen Datenmodell zum Graphen

In den digitalen Geschichtswissenschaften werden Forschungsdaten in den verschiedensten Formaten wie Word-Tabellen, Excel, XML oder relationalen Datenbanken gespeichert. In relationalen Datenbanken wie z.B. MYSQL werden die Informationen in Tabellenform abgespeichert und die verschiedenen Tabellen verknüpft. In der Regel wird das zu Grunde liegende Datenschema normalisiert, um Abfragen und allgemeine Performance der Datenbank zu optimieren. Das Resultat ist aber oft, dass die gespeicherten Daten und Datenstrukturen für Menschen nur schwer les- bzw. strukturierbar sind. Zum anderen kommt hinzu, dass sich im relationalen Datenbankmodell in Bezug auf bestimmte Abfragetypen (wie z.B. traversale Abfragen, Friend-of-a-Friend-Abfragen) Performanceprobleme ergeben, die sich momentan nicht lösen lassen[1]. Seit dem Aufkommen der sozialen Netzwerke und des Internets fallen sehr große Mengen an vernetzten Daten an, die von Forschern und Unternehmen in vielfältiger Weise genutzt werden. Von Firmen werden sie bspw. zur Analyse des Kundenverhaltens oder zur Weiterentwicklung der eigenen Produkte, zur Angebotsoptimierung oder für Werbezwecke verwendet. Auch in den Digitalen Geschichtswissenschaften nimmt die Menge der verfügbaren Forschungsdaten stetig zu, während die Entwicklung der zu ihrer Handhabung notwendigen Techniken und Methoden eher schleppend verläuft[2].

1Vgl. Andreas Kuczera, Digital Editions beyond XML – Graph-based Digital Editions, in: HistoInformatics 2016 - The 3rd HistoInformatics Workshop. Proceedings of the 3rd HistoInformatics Workshop on Computational History (HistoInformatics 2016), hg. von Marten Düring, Adam Jatowt, Johannes Preiser-Kappeller, co-located with Digital Humanities 2016 conference (DH 2016), Krakow, Poland, July 11, 2016, S. 37-46, online unter: http://ceur-ws.org/Vol-1632/paper_5.pdf (CEUR Workshop Proceedings Vol. 1632, http://ceur-ws.org/Vol-1632/) (letzter Abruf: 27.2.2017), künftig zitiert als *Kuczera, Digital Editions beyond XML*.

2Vgl. Andreas Kuczera, Big Data History, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 10.10.2014. URL: mittelalter.hypotheses.org/3962; Andreas Kuczera, Digitale Perspektiven mediävistischer Quellenrecherche, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 18.04.2014. URL:

Zitation
Andreas Kuczera, Das Deutsche Textarchiv in der Graphenwelt, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 4. April 2017,
https://mittelalter.hypotheses.org/10025

Was ist ein Graph?

Für die Speicherung und Analyse dieser umfangreichen, vernetzten Daten werden seit einigen Jahren verstärkt Graphdatenbanken eingesetzt. Dort werden die Daten nicht nach dem relationalen Datenbankmodell in Tabellen abgelegt, sondern in sogenannten Graphen. Diese bestehen aus Knoten und Kanten, die jeweils wieder über Eigenschaften verfügen können.

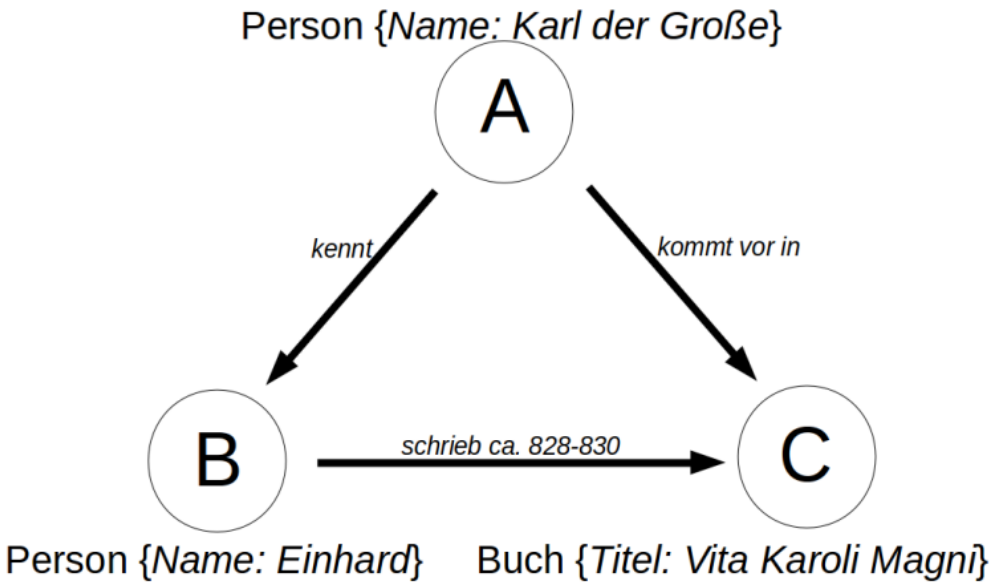

Abbildung 1: Beispielgraph (Quelle: Kuczera).

In Abbildung 1 wird beispielhaft ein solcher Graph gezeigt. Er hat einen Knoten A vom Typ Person, einen Knoten B vom Typ Person und einen Knoten C vom Typ Buch. Alle drei Knoten können noch weitere Eigenschaften haben und sind mit Pfeilen, den sogenannten Kanten, verknüpft. Insgesamt können wir dem Graphen folgende Informationen entnehmen:

1. Die Person A mit dem Namen „Karl der Große“ kennt die Person B mit dem Namen „Einhard“.

2. Die Person B „Einhard“ schrieb etwa in den Jahren 828-830 das Buch C mit dem Titel „Vita Karoli Magni“.

3. Die Person A mit dem Namen „Karl der Große“ kommt in Buch C mit dem Titel „Vita Karoli Magni“ vor, welches von der Person B etwa in den Jahren 828-830 geschrieben wurde.

---

https://mittelalter.hypotheses.org/3492.

Zitation
Andreas Kuczera, Das Deutsche Textarchiv in der Graphenwelt, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 4. April 2017,
https://mittelalter.hypotheses.org/10025

In Graphdatenbanken werden diese Informationen genau so abgespeichert, so dass sich Analysen und Anfragen sehr eng am Datenmodell orientieren und formulieren lassen. Im Vergleich zu oft eher unflexiblen normalisierten Datenbankstrukturen von relationalen Datenbank ist das Graphmodell flexibel erweiterbar. Die in der Datenbank vorhandenen Daten können bei neo4j einerseits mit dem Graphbrowser visuell erkundet oder über die Abfragesprache Cypher analysiert werden. Cypher lehnt sich an SQL an. Befehle werden in der sog. ASCII-Art eingegeben, d.h. es wird versucht die Graphstruktur in der Abfrage grafisch mit Zeichen nachzubilden[3]. Ein cypher-Befehl, der als Ergebnis den in Abb.1 abgebildeten Graphen liefern würde, sähe etwa wie folgt aus:

```
MATCH (A:Person)-[:kommt vor in]->(B:Buch)<-[:schrieb ca. 828-830]-
(C:Person) WHERE A.Name = 'Karl der Große' RETURN *;
```

Hier werden die einzelnen Befehle nun erläutert:

	Erklärung	Cypher-Befehl
1	Finde ...	`match`
2	... eine Person A ...	`(A:Person)`
3	... die mit einer "kommt vor in"-Kante verbunden ist mit einem ...	`-[:kommt vor in]->`
4	... Knoten vom Typ Buch ...	`(B:Buch)`
5	... die mit einer "schrieb ca. 828-830"-Kante verbunden ist mit einem ...	`<-[:schrieb ca. 828-830]-`
6	... Knoten vom Typ Person ...	`(C:Person)`
7	... wobei die Eigenschaft "Name" des Knotens A den Wert "Karl der Große" haben soll ...	`WHERE A.Name = 'Karl der Große'`
8	... und liefere mit alles als Ergebnis zurück.	`RETURN *;`

3 Zu ASCII-Art vgl. https://de.wikipedia.org/wiki/ASCII-Art: „ASCII-Art (englisch für *ASCII-Kunst*) ist eine Kunstrichtung, die mit Buchstaben, Ziffern und Sonderzeichen einer nichtproportionalen Schrift kleine Piktogramme oder ganze Bilder darzustellen versucht. Auf Computern eignet sich der ASCII-Zeichensatz hierfür besonders, da er weltweit auf nahezu allen Systemen verfügbar ist“ (abgerufen am 12.02.2017).

Zitation
Andreas Kuczera, Das Deutsche Textarchiv in der Graphenwelt, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 4. April 2017,
https://mittelalter.hypotheses.org/10025

Abgeschlossen wird eine Befehlszeile immer mit einem Semikolon.

Von XML zum Graphen

Neben tabellenartig vorliegenden Daten lassen sich auch digitale Texte, die in XML vorliegen, in Graphdatenbanken importieren. In *Kuczera, Digital Editions beyond XML* wird ein erster Versuch erläutert, bei dem ein TEI-XML-codierter Brief in die Graphdatenbank neo4j importiert wird. Dabei wird der Text in eine Folge von Wort-Knoten umgewandelt, die mit NEXT-Kanten verbunden sind und damit den Textfluss wiedergeben. Das weitere XML-Markup wird dann entsprechend angelagert.

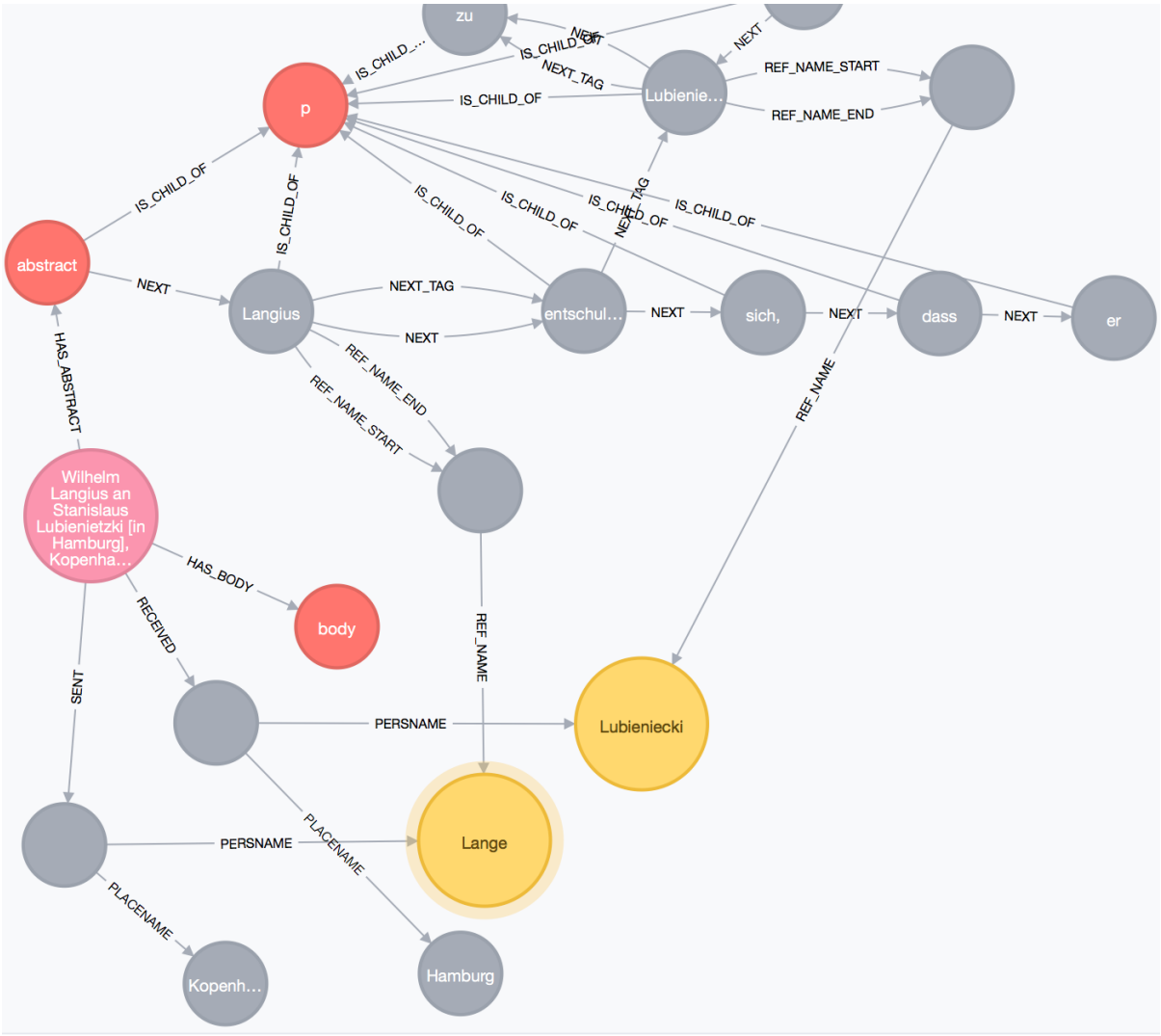

Abbildung 2: TEI-XML-Struktur in der Graphansicht (Quelle: Kuczera).

Zitation
Andreas Kuczera, Das Deutsche Textarchiv in der Graphenwelt, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 4. April 2017,
https://mittelalter.hypotheses.org/10025

DTA-Bf Importer

Das DTA bietet zu den bereitgestellten Texten verschiedene Formate zum Download an. 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 (http://deutschestextarchiv.de/book/show/goethe_faust01_1808) in der TCF-Fassung (http://deutschestextarchiv.de/book/download_fulltcf/16181) in die Graphdatenbank importiert. Für die weiteren Schritte ist eine installierte neo4j-Datenbank (https://neo4j.com/download/community-edition/) inkl. der apoc-Bibliotheken (https://github.com/neo4j-contrib/neo4j-apoc-procedures) notwendig. Die Apoc-Library muss in den plugin-Ordner der neo4j-Datenbank kopiert werden und stellt die XML-Import-Funktion für den Import der DTA-Bf-Dateien bereit. Mit folgendem Befehl kann überprüft werden, ob die Apoc-Bibliotheken installiert sind:

```
CALL dbms.functions()
```

Wenn eine Liste mit Funktionen ausgegeben wird, war die Installation erfolgreich und es kann mit dem Import begonnen werden. Falls nicht, sollte die Datenbank neu gestartet werden.
Um mit dem Importprozess zu beginnen müssen nun zunächst vorbereitete Befehle in die Eingabezeile eingegeben werden. Dabei muss jeder Befehl einzeln aufgerufen werden, da die Browser-Shell nur jeweils einen Befehl ausführen kann:

```
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. Anschließend folgt der erste Import-Befehl:

```
call
apoc.load.xmlSimple("http://deutschestextarchiv.de/book/download_fulltcf/16
181") yield value as doc

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. Im nächsten Schritt werden die NEXT_TOKEN-Kanten zwischen den eingespielten Wörtern erstellt.

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

```
call
apoc.load.xmlSimple("http://deutschestextarchiv.de/book/download_fulltcf/16
181") yield value as doc

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);
```

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/16
181") yield value as doc

unwind doc._TextCorpus._lemmas._lemma as lemma

match (t:Token{id:lemma.tokenIDs})

merge (l:Lemma{text:lemma._text})

create (t)-[:LEMATISIERT]->(l);
```

Zitation
Andreas Kuczera, Das Deutsche Textarchiv in der Graphenwelt, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 4. April 2017,
https://mittelalter.hypotheses.org/10025

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

```
call
apoc.load.xmlSimple("http://deutschestextarchiv.de/book/download_fulltcf/16181") yield value as doc

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.

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 <id> "45414", die Eigenschaft Lemma hat den Wert "Tragödie" während die Eigenschaft text das Originalwort "Tragödie" enthält.

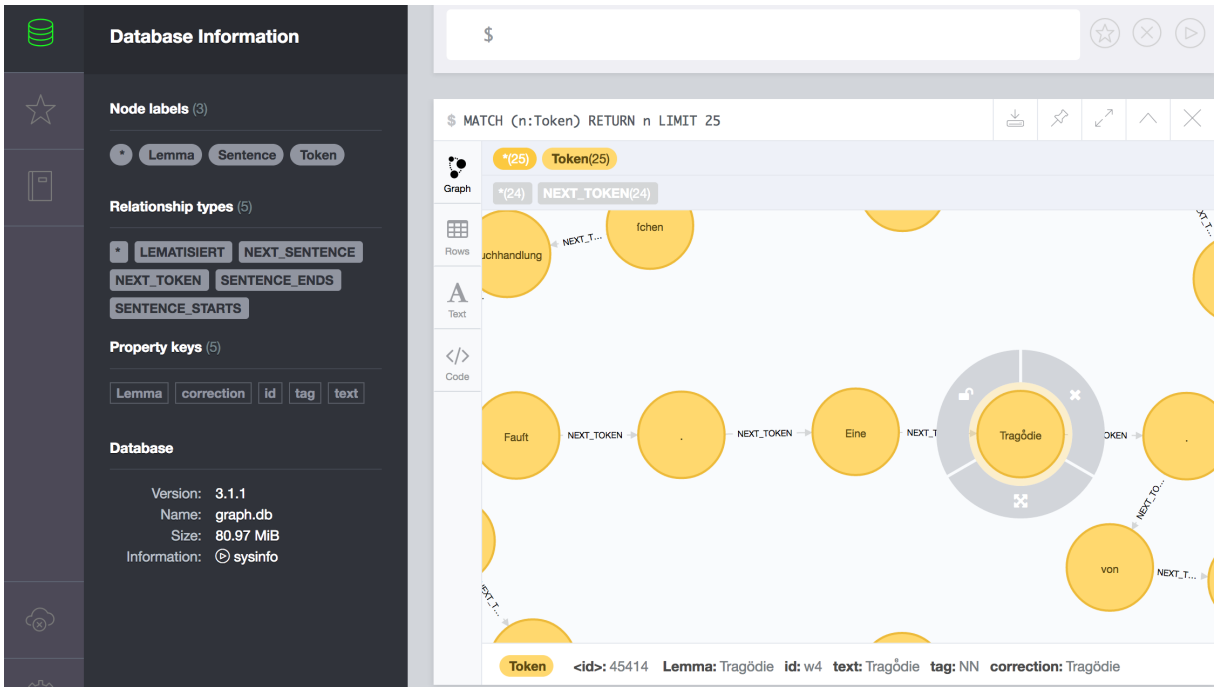

Abbildung 3: Blick in den Neo4j-Graphbrowser mit den ersten Wörtern von Goethes Faust (Quelle: Kuczera).

Zitation
Andreas Kuczera, Das Deutsche Textarchiv in der Graphenwelt, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 4. April 2017,
https://mittelalter.hypotheses.org/10025

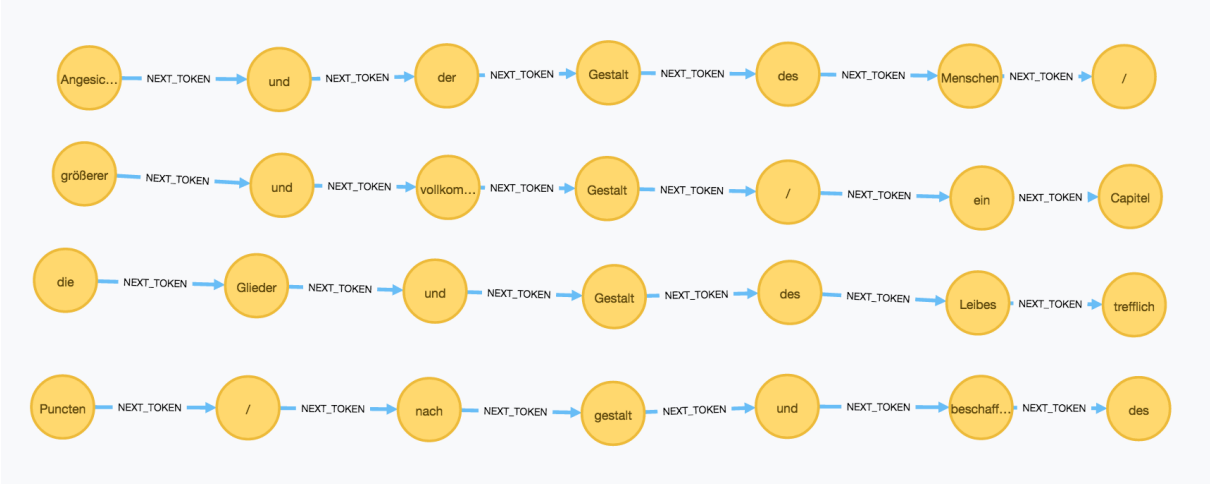

Abbildung 4: Beispiel mit 4 Textketten rund um das Lemma "Gestalt" (Quelle: Kuczera).

Bei Cypher-Abfragen können alle Eigenschaften von Knoten und Kanten miteinbezogen werden. Abbildung 4 zeigt das Ergebnis der Abfrage nach allen Vorkommen von Wörtern mit dem Lemma *Gestalt* im Faust mit den jeweils vorhergehenden und anschließenden drei Wörtern.

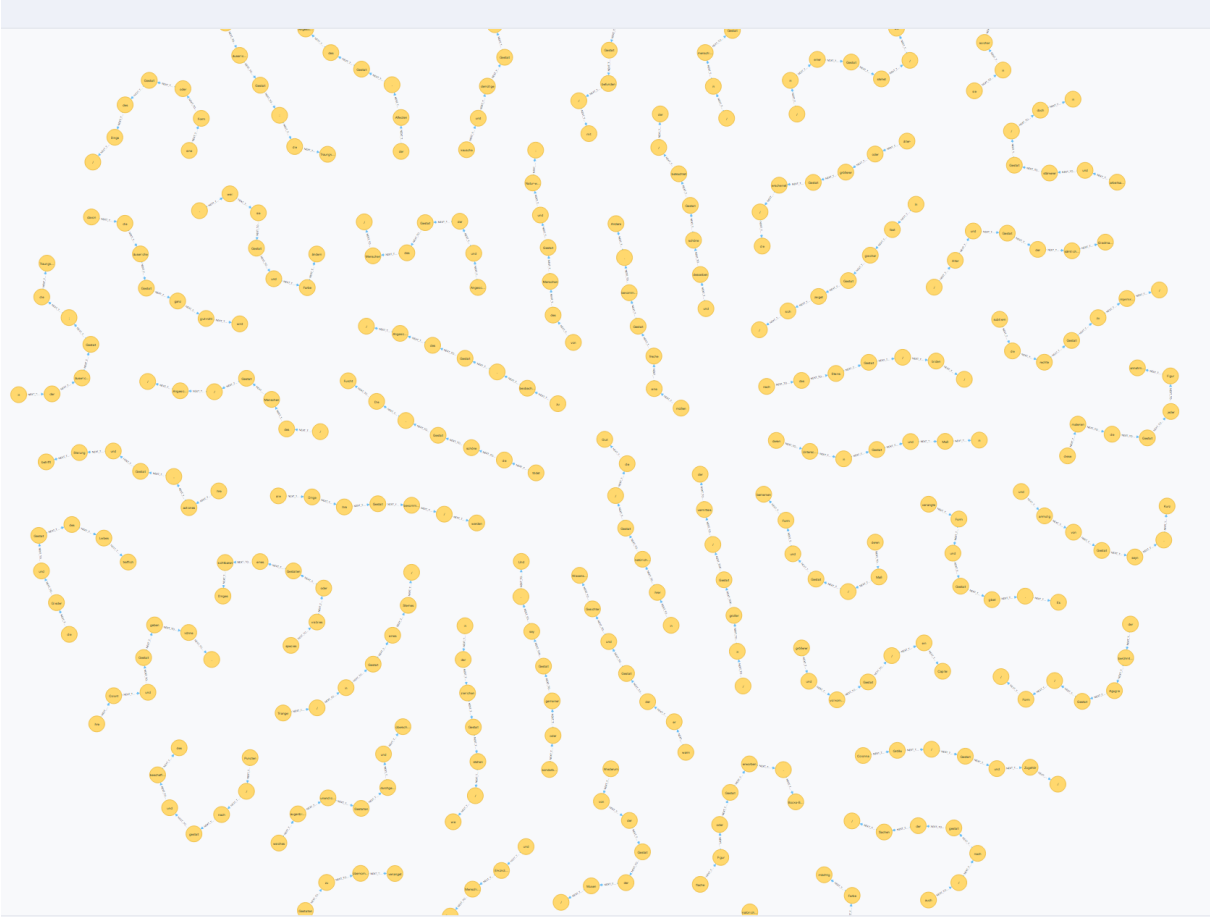

Abbildung 5: Darstellung aller 56 Textstellen mit dem Lemma "Gestalt" (Quelle: Kuczera).

Z i t a t i o n
Andreas Kuczera, Das Deutsche Textarchiv in der Graphenwelt, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte, 4. April 2017,
https://mittelalter.hypotheses.org/10025

Die Abfrage hierzu lautet:

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

Zusammenfassung

Im vorliegenden Beitrag 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.