Masterarbeit
zur Erlangung des akademischen Grades

– **Master of Science** –

im Studiengang Informatik

Generierung eines Englisch-SPARQL-Korpus zur neuronalen maschinellen Übersetzung für Question Answering mit RDF-Wissensdatenbanken

Eingereicht von:	Ann-Kathrin Hartmann
Im:	Februar 2018
Betreuer:	Prof. Dr. Thomas Riechert
Zweitgutachter:	MA. Sc. Edgard Marx
Fachbereich:	Fakultät Informatik, Mathematik und Naturwissenschaften
	Hochschule für Technik, Wirtschaft und Kultur Leipzig

Abkürzungsverzeichnis

IRI International Resource Identifier

KB Knowledge Base

LC-QuAD Large-Scale Complex Question Answering Dataset

LOD Linked Open Data

LSTM Long-Short Term Memory

ML Machine Learning

MT Machine Translation

NMT Neural Machine Translation

NSpM Neural SPARQL Machine

OOV Out-of-Vocabulary

QA Question Answering

QALD Question Answering over Linked Open Data

RDF Resource Description Framework

RNN Rekurrente Neuronale Netze

SPARQL Simple Protocol and RDF Query Language

Inhaltsverzeichnis

1 Einleitung

Die Sprachsteuerung, die noch vor Jahren als Zukunftsvision und Science-Fiction galt, ist heute etablierte Technik. Dem Smartphone Termine ansagen oder dem Smart-Home-System einen Musikwunsch äußern ist nichts Ungewöhnliches mehr. Es mag der Gewohnheit geschuldet sein, dass es noch normal erscheint, bei einer Standardinternetsuche die Anfrage in Schlüsselwörtern auszudrücken, um dann eine Liste mit Texten zu erhalten, die dazu womöglich relevante Informationen beinhalten, anstatt eine natürlichsprachliche Frage zu formulieren und darauf eine konkrete Antwort zu erhalten.

Die dafür notwendigen im RDF-Format veröffentlichten Daten und damit für Computer verständliche Information nehmen zu. Der Zugriff darauf erfordert allerdings Semantic-Web-Fachwissen, insbesondere Kenntnisse in der formalen Abfragesprache SPARQL und zu den verwendeten Vokabularen und Ontologien der RDF-Wissensdatenbanken. Question Answering mit RDF-Wissensdatenbanken hat zum Ziel, diese Barriere abzubauen, indem Benutzer eine in ihrer natürlichen Sprache formulierte Frage stellen können. Diese wird automatisch in eine für den Computer verständliche und somit im Idealfall auch beantwortbaren Frage übersetzt.

Maschinelles Lernen insbesondere mit neuronalen Netzen hat in den letzten Jahren zu großen Verbesserungen im Bereich der maschinellen Übersetzung geführt und ist deshalb auch in den Fokus von Question-Answering-Entwicklern gerückt (vgl. Abschnitt 3.2). Doch der Erfolg maschinellen Lernens beruht auf ausreichend vorhandener Trainingsdaten. Dabei spielen Parallelkorpora, repräsentative Sammlungen von Texten, die in zwei Sprachen vorliegen, eine wesentliche Rolle. Korpora für natürliche Sprachen lassen sich mit Texten aus verschiedensten Quellen, z.B.

Onlinenachrichtenportalen, Büchern, Blogs, Filmuntertiteln oder sogar Protokollen des Europäischen Parlaments[1] aufbauen. Betrachtet man wie Soru et al. [2017] SPARQL als eine Fremdsprache, in die sich z.B. Englisch mit Methoden der neuronalen maschinellen Übersetzung übersetzen lässt, so sucht man derzeit vergeblich nach adäquaten Englisch-SPARQL-Korpora (vlg. Abschnitt 3.3).

Ziele dieser Masterarbeit sind:

- die Generierung eines Englisch-SPARQL-Korpus und
- der Beweis, dass der in Abschnitt 3.1 beschriebene neuronale maschinelle Übersetzungsprozesses von Englisch zu SPARQL unter Verwendung des erstellten Korpus für domänenunspezifisches Question Answering eingesetzt werden kann.

Die Arbeit ist nach dem methodischen Vorgehen wie folgt gegliedert: Kapitel 2 thematisiert die zum Verständnis notwendigen theoretischen Grundlagen zum maschinellen Lernen, Semantic Web und Question Answering. Im Kapitel 3 wird der neuronale maschinelle Übersetzungsansatz von Soru et al. [2017], auf den diese Arbeit aufbaut, vorgestellt und ein Überblick zu weiteren ähnlichen aktuellen Forschungsarbeiten und -datensätzen aufgeführt. Das Kapitel 4 beinhaltet die Beschreibung des Korpuserstellungsprozesses und der Korpuseigenschaften. Die Evaluierung der neuronalen maschinellen Übersetzungsmethode nach Soru et al. [2017] unter Verwendung des Korpus wird in Kapitel 5 behandelt. Im Kapitel 6 werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick zu weiterführenden Arbeiten gegeben.

[1] Europarl: `http://www.statmt.org/europarl/`

2 Grundlagen

Das Kapitel beleuchtet die zwei Fachgebiete, die in dieser Arbeit aufeinandertreffen. Das ist zum einen die maschinelle Übersetzung mithilfe von rekurrenten neuronalen Netzen. Dafür wird auch ein Abriss zum maschinellen Lernen und der Entwicklungsgeschichte der maschinellen Übersetzung gegeben. Zum anderen werden Question Answering mit Linked Open Data und die erforderlichen Semantic Web Grundlagen erörtert.

2.1 Maschinelles Lernen

Der derzeitige Technologiestand ermöglicht die Speicherung sehr großer Datenmengen. Diese sind allerdings nur von Nutzen, wenn daraus verwertbare Informationen gewonnen werden können. In den Fällen, wo maschinelle Lernverfahren eingesetzt werden, ist unklar wie diese Information genau gewonnen werden können. Es besteht aber die Annahme, dass dies möglich ist und der Prozess nicht rein zufällig sein kann. Der Mensch zieht mithilfe seines in der Vergangenheit aufgebauten – gelernten – Erfahrungsschatzes logische Schlüsse aus seinen Sinneswahrnehmungen oder zu bewertenden Informationen. Analog zum menschlichen Lernprozess ist es vorstellbar, dass eine Maschine lernen und einen „Erfahrungsschatz“ aufbauen kann, mit dem sie Muster oder Regelmäßigkeiten zu erkennen vermag. Ein mathematisches Modell ist die Nachbildung dieses Erfahrungsschatzes. Alpaydın und Linke [2008a] definieren Machine Learning (ML) als Optimierung eines „bestimmten Leistungskriteriums anhand von Beispieldaten oder Erfahrungswerten“. Sie vergleichen die Justierung der Modellparameter mit einem Lernprozess. Das

übrige Kapitel gibt zunächst eine allgemeine Einführung zu Arten, Anwendungsbereichen und Eigenschaften von ML-Verfahren, um dann auf neuronale Netze und rekurrente neuronale Netze eingehen zu können.

2.1.1 Einführung

Raschka [2017a] unterscheidet beim ML zwischen überwachtem, unüberwachtem und be- oder verstärkendem Lernen. Be- oder Verstärkendes Lernen verfolgt das Erlernen von Taktiken. Das können siegbringende Schachzüge aber auch geschickt vollzogene Roboteraktionen sein. Überwachtes Lernen zeichnet aus, dass Trainingsdaten mit bekannten, erwünschten Ergebniswerten vorliegen. Beim unübewachten Lernen ist das nicht der Fall. Hier sollen verborgene Strukturen erkannt werden, indem Clusteranalysen benutzt werden. Anwendungsbeispiele sind Dimensionreduktion, Bildkompression oder DNS-Sequenzanalyse. Anwendungen wie Assoziationsbildung, Klassifikation, Mustererkennung oder Regressionsprobleme fallen in den Bereich des überwachten Lernens. Als Regressionsproblem gehört auch die in dieser Arbeit vorgestellte Methode zum maschinellen Übersetzen dazu. Folgende Arbeitsschritte werden beim ML angewandt [Raschka, 2017c]:

1. Vorverarbeitung: Daten in Form bringen, eventuell Dimensionsreduktion, Datenbestand in Trainings- und Testdaten aufteilen
2. Trainieren und Auswählen eines Vorhersagemodells, Bewertungskriterien festlegen
3. Bewertung von Modellen und Vorhersage anhand unbekannter Dateninstanzen

Hauptproblem des ML sind laut Raschka [2017e] Über- und Unteranpassung. Bei der Überanpassung funktioniert das Modell sehr gut für die Trainingsdaten, kommt aber nicht mit unbekannten Daten zurecht. Zu viele Parameter machen das Modell zu komplex, sodass von einer zu großen Varianz gesprochen wird. Übertraining ist ebenfalls eine mögliche Ursache für Überanpassung. Das Training muss frühzeitiger abgebrochen werden. Bei der Unteranpassung tendiert das Modell zu stark für

eine bestimmte Entscheidung. Das Modell war nicht komplex genug, um Muster in den Trainingsdaten zu erkennen und besitzt damit ein zu großes Bias. Die Generalisierungsfähigkeit lässt sich mithilfe einer Kreuzvaliderung erkennen, bei der Modellresultate für Trainings- und Testdaten verglichen werden [Alpaydın und Linke, 2008c]. Der Generalisierungsfehler nimmt unter Zunahme von Trainingsdaten ab. Je reicher der Erfahrungsschatz eines Menschen ist, desto wahrscheinlicher ist, dass er auch in Situationen, die er so nie zuvor erlebt hat, gute Entscheidungen trifft. Damit das Modell gute Ergebnisse auch für Nicht-Trainingsdaten liefert, benötigt es ebenfalls viele Trainingsdaten.

2.1.2 Neuronale Netze

Neuronale Netze sind inspiriert von biologischen Gehirnen, welche anspruchsvolle Aufgaben bewältigen und das mit offensichtlich weniger Speicherkapazität und Rechengeschwindigkeit als moderne Computer. Außerdem sind sie robust gegenüber beschädigten oder unvollkommenen Signalen und erlauben so Unschärfe und Mehrdeutigkeit. Die Grundeinheit des biologischen Gehirns ist das Neuron [Rashid, 2017a]. Elektrische Signale werden von den Dendriten über Axone an Terminale übertragen. Es erfolgt eine Übersetzung von einem Eingabesignale in ein Ausgabesignal. Dabei werden die Ausgaben so lange unterdrückt, bis die Eingabe hinreichend groß ist. Eine mathematische Nachbildung hierfür stellt eine Aktivierungsfunktion dar. Die Verarbeitung mehrerer Eingabesignale wird durch eine geeignete Kombination wie z.B. die Summe der Eingaben modelliert. Ein Neuron allein bewirkt nicht viel. Stattdessen nehmen Neurone Signale von anderen Neuronen auf und geben Signale an weitere Neuronen weiter. Ein neuronales Netz besteht somit aus Schichten von Knoten, die mit den in der nächsten Schicht liegenden Knoten verbunden sind.

Demonstriert an einem Beispiel: Angenommen ein Neuron hat zwei Vorgängerneuronen und erhält von diesen Signalwerte von $1,0$ und $0,5$. Die Sigmoidfunktion

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

wird aufgrund ihrer guten Berechenbarkeitseigenschaften häufig als Schwellwertfunktion genutzt. Dann würde das Neuron eine Ausgabe von

$$\sigma(1,0+0,5)=0,8175$$

erzeugen. Was könnte dieses Modell aber lernen, wenn alle Werte fix sind? Dafür benutzt das Modell gewichtete Verbindungen, wobei die Gewichte erlernt werden. Sei die erste Eingangsverbindung mit $0,9$ und die zweite mit $0,3$ gewichtet. Das Neuron erzeugt somit eine Ausgabe von

$$\sigma(1,0*0,9+0,5*0,3)=0,7408.$$

Rashid [2017b] erklärt, der Fehler an den Ausgabeknoten – die Differenz vom Ist- und Sollwert – wird verringert, indem eine Verfeinerung der Gewichte vorgenommen wird. Wäre das Neuron im Beispiel ein Ausgabeneuron und läge dessen Sollwert bei $0,5$, so bringt die Anpassung der Gewichte auf $0,5$ und $0,2$ eine bessere Annäherung mit

$$\sigma(1,0*0,5+0,5*0,2)=0,6457.$$

Der Gesamtfehler ist eine Funktion der internen Verknüpfungsgewichte, d.h. um den Fehler zurückzuführen, werden alle internen Gewichte justiert, nicht nur die letzten. Diese Aufgabenstellung ist bei mehrschichtigen Netzen zu komplex, um direkt die richtigen Gewichte anzupassen, weshalb das Gradientenverfahren für die Fehlerrückführung, engl. Backpropagation, genutzt wird. Die Lernrate moderiert die Aktualisierungen der Fehlerrückführung, damit das Modell nicht nur dem letzten Trainingsbeispiel entspricht. Ist die Lernrate zu groß, ist ein Überschwingen, engl. Overshooting, wahrscheinlicher, ist sie zu klein, wird die Geschwindigkeit des Gradientenabstiegs stark begrenzt und Feststecken in lokalen aber nicht globalen Minima begünstigt. Dem Finden falscher Minima wird mit Epochen entgegengewirkt. Eine Epoche ist ein kompletter Durchlauf durch alle Trainingsdaten. Die Durchführung mehrerer Durchläufe erlaubt die Verwendung kleinerer Lernraten, erhöht aber auch die Gefahr der Überanpassung. Die Anzahl der Knoten in den verborgenen Schichten und die Anzahl der Schichten beeinflusst die Lernkapazität. Zu wenige könnten diese zu sehr einschränken. Zu viele führen zu vielen Möglichkeiten

diese anzupassen und somit wird es schwieriger das Netz zu trainieren. Es existieren keine exakten Vorschriften, wie diese (Hyper-)Parameter zu setzen sind, um ein gutes Modell aufzubauen. Geeignete Werte müssen experimentell oder durch Erfahrungswerte bestimmt werden. Dabei spielen auch die Wahl der Aktivierungs- und Kombinationsfunktionen, die Anfangsgewichte und die Normierung der Trainingsdaten wichtige Rollen, wie Rashid [2017c] betont. Die Konvergenz des Lernprozesses ist nicht in jedem Fall garantiert. Das Durchmischen der Trainingsdaten nach Epochen oder eine adaptive Lernrate sind fördernde Mittel [Raschka, 2017d]. Notfalls muss sie durch die Vorgabe einer maximalen Epochenanzahl gewährleistet werden.

2.1.3 Rekurrente Neuronale Netze

Raschka [2017b] spricht von tiefen neuronalen Netzen, sobald ein Netz mehr als eine verdeckte Schicht besitzt. Neben mehrschichtigen Netzen, die nur vorwärtsgerichtete Verbindungen aufweisen, sogenannten Feedforward-Netzen, existieren auch andere Netzarchitekturen. Hauptgruppen sind Konvolutionale Neuronale Netze, die besonders lokalen Strukturen Rechnung tragen, und Rekurrente Neuronale Netze (RNN) für zeitliche Zusammenhänge. Zeitliche Zusammenhänge sind von Interesse, wenn die Eingaben nicht mit einem Mal alle zusammen eingespeist werden, sondern als zeitlich variable Sequenz [Alpaydın und Linke, 2008b]. Sequenzerkennung, Sequenzreproduktion und temporale Assoziation sind Beispiele für Lernen mit Zeitreihen, die besonders im Bereich der Sprachverarbeitung von Bedeutung sind. Soll z.B. eine Zahlenreihe in ein Zahlwort übersetzt werden, wobei die Ziffern dem Modell nur nach und nach zur Verfügung stehen, so würde ein einfaches neuronales Netz die Reihe *100* nur in die Sequenz *eins null null* und nicht *einhundert* transformieren, da dem Modell nie die Information einer vorangegangenen Eingabe zur Verfügung steht.

RNN enthalten zusätzlich zu den Vorwärtsverbindungen auch Verbindungen zu sich selbst, Knoten aus vorangegangenen Schichten oder der eigenen Schicht (direkte/indirekte/seitliche Rückkopplung). Neuronen, die Empfänger solcher Rückkopplungen sind, werden als Kontexteinheiten bezeichnet. Jedes Neuron der Schicht i

speist eine Kopie seiner Ausgabe in jeweils eine Kontexteinheit ein. Die Kopie wird durch eine Aktivierungsfunktion transformiert und im nächsten Schritt an alle Neuronen der Schicht i zurückgegeben, wobei diese Verbindungen erneut zu Lernzwecken gewichtet sind. Zur Demonstration soll wieder das Beispiel aufgegriffen werden, dass im Kapitel 2.1.2 verwendet wurde. Dort lag das Eingangssignal

$$\begin{pmatrix} 1,0 \\ 0,5 \end{pmatrix}$$

an dem Neuron. Nun sei die Sequenz

$$\begin{pmatrix} 1,0 & 1,0 \\ 0,5 & 0,5 \end{pmatrix}$$

das Eingangssignal. Sei die Aktivierungsfunktion der Kontexteinheit ebenfalls die Sigmoidfunktion und das Verbindungsgewicht von Kontexteinheit zum Neuron $1,0$. Beim ersten Verarbeitungsschritt erhält das Neuron von der Kontexteinheit noch keinen Input und gibt demnach die Ausgabe

$$\sigma(1,0 * 0,9 + 0,5 * 0,3 + 0 * 1,0) = 0,7408$$

aus und leitet diese auch an die Kontexteinheit. Der berechnete Wert

$$\sigma(0,7408) = 0,6772$$

wird im nächsten Schritt als weitere Eingabe dem Neuron eingespeist. Dieses erzeugt nun die Ausgabe

$$\sigma(1,0 * 0,9 + 0,5 * 0,3 + 0,6772 * 1,0) = 0,2689.$$

Das Neuron gibt damit zwei verschiedene Werte aus, obwohl ursprünglich zweimal das gleiche Signal eingespeist wurde. Kontexteinheiten erinnern sich an Vorangegangenes und können somit laut Rey und Wender [2011] als „dynamisches Gedächtnis des neuronalen Netzes“ betrachtet werden. Die zeitlich versetzt notwendige Fehlerrückführung macht RNN schwieriger zu trainieren. Laut Raschka

[2017b] bewältigen Long-Short Term Memory (LSTM)-Einheiten dieses Problem und machen RNN damit praktikabel.

2.2 Maschinelle Übersetzung

Luong [2016] stellt die Entwicklungsgeschichte der maschinellen Übersetzung, engl. Machine Translation (MT), dar und beschreibt, dass die Anfänge in den 50er/60er Jahren schnell deutlich machten, dass MT eine große Herausforderung ist und einfache Wort-zu-Wort-Übersetzung basierend auf bilingualem Wörterbüchern keine Lösungsoption darstellt. Das Kapitel beschäftigt sich mit der statistischen maschinellen Übersetzung und der neuronalen maschinellen Übersetzung, engl. Neural Machine Translation (NMT), welche als Spezialfall des Sequenz-zu-Sequenz-Lernens betrachtet wird. Auch Wortvektoren als wichtiges Produkt und Rohstoff von MT finden Erwähnung.

2.2.1 Statistische maschinelle Übersetzung

In den 90ern begann die moderne statistische maschinelle Übersetzung. Diese nutzt Parallelkorpora, also Textsammlungen, die sowohl in der Ausgangs- als auch der Zielsprache vorliegen. Der Systemaufbau ist für alle Sprachen gleich und somit sprachunabhängig. Das Übersetzungsmodell lernt Wortzuordnungen: Je häufiger zwei Wörter aus Ziel und Quellsprache in verschiedenen Sätzen miteinander auftreten, desto wahrscheinlicher ist, dass sie aufeinander ausgerichtet sind und eine äquivalente Bedeutung besitzen. Das Verfahren bestimmt anhand eines Sprachmodells wie lang eine Übersetzung und wie die Zuordnung sein könnte. Dann wird für jede Satzposition die beste Übersetzung gesucht. Die Kandidaten dafür stammen ebenfalls vom Sprachmodell. Dieses hat zuvor aus einem monolingualen Korpus gelernt, welche Wörter häufig miteinander auftreten und bewertet somit zusammenhängende/natürlich-klingende Übersetzungen höher als schlechte. Es stellte sich heraus, dass wortbasierte statistische Modelle nicht genügen, da Wörter für eine richtige Übersetzung Kontext benötigen. Um 2000 fand der

Übergang zu phrasenbasierten Modellen statt. Hierbei werden zusammenhängende Satzteile und deren Kombination vom Sprachmodell bewertet und übersetzt. Cho et al. [2014] stellten das RNN Encoder-Decoder Modell zum Lernen von Phrasen vor und legten damit den Grundstein für den Sequenz-zu-Sequenz-Ansatz und bewiesen, dass neuronale Netze und maschinelles Lernen vielversprechende Methoden zur Übersetzung darstellen. Das Modell besteht aus zwei RNN, wobei das erste Symbolsequenzen der Ausgangssprache als Vektoren kodiert und das zweite diese Vektoren in Symbolsequenzen der Zielsprache dekodiert. Dies wurde zunächst verwendet, um die statistische maschinelle Übersetzung zu verbessern.

2.2.2 Neuronale maschinelle Übersetzung

Die statistische maschinelle Übersetzung erzielte große Erfolge in letzten Jahren, allerdings – so bemängelt Luong [2016] – bestehen die Systeme aus vielen Komponenten und der Aufbau der mehrstufigen Prozesse ist aufwändig und erfordert enorme menschliche Expertise. Diese Komplexität erschwert weitere Fortschritte und die Übersetzungsqualität stagniert. Das auf Phrasen basierende Modell leidet auch unter Lokalität. Abhängigkeiten zwischen Wörtern, die weiter auseinander stehen, engl. Long-Distance Dependencies, werden stark ignoriert. In diese Lücke greift die NMT, wobei ein großes neuronales Netz so designt ist, die gesamte maschinelle Übersetzungspipeline abzubilden. Anstatt jede einzelne Komponente separat zu trainieren, was Domänenwissen erfordert, wird nur das eine Netz trainiert. Das End-zu-End-Lernen zeigt besonders Potenzial in der Bewältigung der Long-Distance Dependencies und dem Generalisierungsproblem.

2.2.3 Sequenz-zu-Sequenz-Lernen

Wie in Kapitel 2.1.3 beschrieben, lassen sich RNN für das Lernen mit Zeitreihen – der Übersetzung von Sequenzen in Sequenzen – benutzen. Dafür muss die Zuordnung zwischen Eingabe und Ausgabe allerdings bekannt sein und das schränkt die Anwendung auf Sequenzen mit festen Längen ein. Die Anwendbarkeit auf

Eingabe-Ausgabe-Sequenzen von variabler Länge und mit verschiedenen Zuordnungsrelationen lässt sich durch einen Zwischenschritt herstellen. Dafür wird die Eingabesequenz mithilfe eines RNN in einen Vektor fester Größe/Dimensionalität übersetzt und mithilfe eines anderen RNN wird dieser Vektor anschließend in die Ausgabesequenz übersetzt, was dem Encoder-Decoder Modell von Cho et al. [2014] entspricht. Das Encoder-RNN liest in jedem Zeitschritt ein Eingabesymbol x_t und aktualisiert seinen internen Zustand $h_t = f(h_{t-1}, x_t)$. Nachdem das letzte Sequenzsymbol gelesen wurde, entspricht der interne Zustand des RNN c einer Zusammenfassung der gesamten Eingabesequenz $c = q(h_1, \ldots, h_T)$ mit q als Encoder-Zusammenfassungsfunktion. Das Decoder-RNN wird trainiert, eine Ausgabesequenz zu erzeugen durch Vorhersage des nächsten Symbols y_t auf Grundlage des internen Zustands zum Zeitpunkt t. Dieser interne Zustand h_t berücksichtigt den Vorgängerzustand h_{t-1}, die Vorgängerausgabe y_{t-1} und die Zusammenfassung der gesamten Eingabesequenz c:

$$h_t = f(h_{t-1}, y_{t-1}, c)$$

Eine sogenannte versteckte Einheit ist für die Anpassung des internen Zustands zuständig. LSTMs Stärke, mit Long-Distance Dependencies umgehen zu können, ist der Grund, weshalb sie hier Verwendung finden. Wie bereits angedeutet, wird das Übersetzungsproblem in ein Wahrscheinlichkeitsproblem transformiert [Sutskever et al., 2014]. Das RNN schätzt die Wahrscheinlichkeit einer Ausgabesequenz $y_1, ..., y_{T'}$ unter einer gegebenen Eingabesequenz $x_1, \ldots, x_T$, also die bedingte Wahrscheinlichkeit $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$, wobei die Sequenzen von verschiedener Länge sein können. Das LSTM bezieht zunächst die Vektorrepräsentation v fester Größe der Eingabesequenz aus dem letzten internen Zustand und berechnet dann die bedingte Wahrscheinlichkeit für die Ausgabesequenz mit der Formel

$$p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, \ldots, y_{t-1})$$

Ein Beispiel nach Neubig [2017]:

$$p(Er, trinkt, Gin) = p(Er|v) \cdot p(trinkt|v, Er) \cdot p(Gin|v, Er, trinkt)$$

Die Wahrscheinlichkeit einer Ausgabesequenz wird somit auf die Vorhersage des nächsten Wortes unter Berücksichtigung der gegebenen Wörter transformiert. Im Training werden die bedingten Wahrscheinlichkeiten für eine korrekte Übersetzung T zur Eingabe S, mit S und T aus der Trainingsdatenmenge M, maximiert:

$$1/|M| \sum_{(T,S)\in M} p(T|S)$$

Ist das Training beendet, wird eine Übersetzung produziert, indem die wahrscheinlichste Übersetzung gefunden wird.

$$\hat{T} = arg \max_{T} p(T|S)$$

Diese wird bei Sutskever et al. [2014] mit einem sogenannten Beam Seach Decoder gefunden, der eine gewisse Anzahl B von Teilhypothesen verwaltet. Eine Teilhypothese ist Prefix einer Übersetzung. In jedem Zeitschritt wird jede Teilhypothese mit jedem möglichen Wort aus dem Vokabular erweitert. Im Anschluss daran werden alle Teilhypothesen, die nicht unter denen mit den B besten Wahrscheinlichkeiten liegen, verworfen, um die Zahl der Teilhypothesen handhabbar zu halten. Am Ende wird die Hypothese mit der höchsten Wahrscheinlichkeit als beste Übersetzung angenommen. Wird nur eine Teilhypothese verwaltete, handelt es sich um die Greedy-Variante. BLEU ist eine automatische Methode zur Evaluation von maschinellen Übersetzungen, welches beansprucht nah am menschlichen Urteil zu liegen. Die Beurteilung liegt zwischen 0 und 1, wobei 1 einer sehr hohen Ähnlichkeit entspricht [Papineni et al., 2002]. Ein weiteres Bewertungsmaß, das von Sequenz-zu-Sequenz-Modellen herangezogen wird, ist die Perplexity, dt. Ratlosigkeit/Verwirrung, welches ein Maß dafür ist, wie unsicher das Modell bezüglich seiner erzeugten Ausgabe ist – je größer die Zahl desto größer die Unsicherheit [Neubig, 2017].

2.2.4 Sprachmodell

Im Sprachmodell werden Wörter als symbolische Einheit betrachtet und in eine Zahlenrepräsentation transformiert. Der Encoder verarbeitet nicht direkt Wörter sondern deren Zahlenrepräsentation, Wortvektoren. Und analog gibt der Decoder auch nicht Wörter sondern Wortvektoren aus. Eine einfache Zahlenrepräsentation wäre, wenn alle Wörter durchnummeriert werden und die Position des ersten Auftretens genommen wird. Die Zuordnung ist abhängig von der Position im Eingabetext und damit zufällig. Am Beispiel: Angenommen das Wort „Topspin“ erhält die Zuordnung zur Zahl 1390 und das Wort „Flip“ zu 5296, dann enthält diese Zuordnung keinerlei Information über die Ähnlichkeit dieser beiden Begriffe. Da dies beides Angriffsschläge im Tischtennis sind, ist die Wahrscheinlichkeit groß, dass sie aber in einem ähnlichen Kontext auftreten. Aus diesem Grund existieren Verfahren, die eine kontextabhängige Zuordnung – engl. Word Embeddings – erlernen. Dabei werden ähnliche Wörter in Vektoren transformiert, die im Vektorraum nah beieinander gruppiert sind, wobei ihre Ähnlichkeit mathematisch bzw. statistisch auf Basis eines Korpus, einer Textsammlung, entdeckt wird [Mikolov et al., 2013]. Vektorkomponenten werden gelernt, sodass sie den zugehörigen Kontext gut widerspiegeln.

Das Sequenz-zu-Sequenz-Modell muss diese Embeddings nachschlagen, um die zugehörigen Wörter zu erhalten. Dafür werden die Vokabulare der Ausgangs- und Zielsprache genutzt, die aus den Trainingsdatensatz generiert werden. Desto größer die Vokabulare sind, umso aufwendiger werden die Wahrscheinlichkeitsberechnungen, weshalb im Allgemeinen die Größe $|V|$ der Vokabulare auf 30.000 - 80.000 beschränkt wird und somit nur die $|V|$ häufigsten Wörter enthalten sind [Luong, 2016]. Alle anderen werden in ein Sondersymbol wie *unknown* konvertiert. Damit ist das Sequenz-zu-Sequenz-Modell stark vokabularabhängig. Selbst wenn keine Einschränkung der Vokabulargröße angewandt wird, können im Testszenario Wörter auftauchen, die nicht ein einziges Mal im Trainingsdatensatz enthalten waren und demnach auch nicht im Vokabular. Es existieren verschiedene Verfahren zum Umgang mit diesen unbekannten Wörtern, engl. Out-of-Vocabulary (OOV) Words [Neubig, 2017]:

Verwendung von `<unk>` Vorhersagen, in welchem Kontext ein unbekanntes Wort auftritt, um mit Wahrscheinlichkeit des unbekannten Wortes weiterzurechnen.

Kopiermechanismus Statt der Ausgabe von `<unk>` das Wort aus dem Originalsatz ausgeben, das nicht übersetzt werden konnte. Dies ist mithilfe eines Attention-Mechanismus umsetzbar. Vorteil ist, dass der Benutzer sieht, was nicht übersetzt werden konnte.

Ersetzungsmechanismus Ähnlich zum Kopiermechanismus, außer das versucht wird, das unbekannte Originalwort mithilfe eines zusätzlichen Wörterbuchs zu übersetzen.

Verwendung vortrainierter Wortvektoren Anstatt die Word Embeddings selber aus dem Eingabetext zu erlernen, lassen sich Wortvektoren von Word2Vec[1] oder GloVe[2] benutzen, die auf Basis sehr großer Korpora und damit größerer Vokabulare trainiert wurden. Die Wahrscheinlichkeit für unbekannte Worte kann so verringert werden.

Analog zu den Wortvektoren prägte der Forscher Geoffrey Hinton den Begriff „Thought Vectors“ für die Vektoren, die vom Sequenz-zu-Sequenz-RNN gebildet werden. Bei der NMT wird die Bedeutung von Sätzen erfasst und Satzrepräsentationen gelernt, sodass Vektoren ähnlicher Sätze näher beieinander liegen als die von verschiedenen [Sutskever et al., 2014].

2.3 Question Answering mit Linked Open Data

Die Linked Data Cloud wächst stetig. Für Laien sind diese Informationen dennoch schwer zugänglich, da sie kein Simple Protocol and RDF Query Language (SPARQL) beherrschen. Question Answering über Linked Open Data soll hier Abhilfe schaffen. Es setzt sich zur Aufgabe, typischen Webnutzern Zugriff auf Wissensdatenbanken und damit Profit aus dem Semantic Web zu verschaffen. Fragen sollen

[1] Word2Vec: `https://code.google.com/archive/p/word2vec`
[2] GloVe: `https://nlp.stanford.edu/projects/glove`

automatisch mithilfe der in Wissensdatenbanken enthaltenen Fakten beantwortet werden. Das Kapitel reißt die wichtigsten Grundlagen: Semantic Web, Linked Open Data und Question Answering (QA) an.

2.3.1 Semantic Web und Linked Open Data

Das Semantic Web hat die Absicht, im Internet gesammelte Informationen maschinenlesbar zu gestalten, um Wissen gezielt abfragbar zu machen und manuelle Aufgaben automatisieren zu können. Es kann als Erweiterung des World Wide Webs angesehen werden und basiert auf Standards und Technologien, die größtenteils im Resource Description Framework (RDF) enthalten sind. Linked Data ist ein Teilkonzept, das die Veröffentlichung und Verlinkung strukturierter Daten verfolgt, was als Linked Open Data (LOD) Cloud bezeichnet wird.

Das Framework RDF enthält eine gleichnamige formale Sprache, die zur Beschreibung von Wissen dient. Nach Yu [2011] beruht die Grundidee darauf, Wissen in kleine Teile zu zerlegen, sodass jede Information dargestellt werden kann. Diese kleinste Informationseinheit ist eine Aussage über eine Entität bzw. Ressource, was etwas physisch Reales oder ein Konzept sein kann. Eine Aussage enthält ein Subjekt, was eine Entität ist; ein Objekt, was ebenfalls eine Entität oder ein Literal sein kann; und ein Prädikat, was eine Eigenschaft vom Subjekt oder die Beziehung zwischen Subjekt und Objekt ist. Diese Dreierkonstellation wird als Tripel bezeichnet. Wissen kann nun als Liste von Aussagen oder auch in einem gerichtetem Graph mit Subjekten und Objekten als Knoten und Prädikaten als Kanten ausgedrückt werden. Subjekte und Objekte referenzieren durch International Resource Identifiers (IRIs) Ressourcen. Dies allein würde allerdings nur eine syntaktisch andere Art der Beschreibung von Faktenwissen darstellen. Für die Semantik bedarf es Ontologien, welche Wissensbasen sind, die das Wissen einer Domäne modellieren. Mithilfe von Ontologien kann implizites Wissen durch maschinelles Schlussfolgern extrahiert werden.

Eine sehr große Sammlung von RDF-Tripeln wird als Wissendatenbank, engl. Knowledge Base (KB) oder Knowledge Graph (KG) bezeichnet. DBPedia, Freeba-

se und Wikidata sind Beispiele. Es existieren auch proprietäre wie z.B. Knowledge Graph von Google oder Satori Knowledge Base von Microsoft Bing. Informationen müssen nicht wie bei gewöhnlichen Datensammlungen durch eine Textsuche herausgezogen werden, sondern können mit einer direkten Abfrage aus dem mithilfe von RDF beschriebenem Wissen gewonnen werden. Zum Formulieren von Abfragen an RDF-Datenquellen, engl. Triplestores, dient die Sprache SPARQL. Die drei Hauptbestandteile einer Abfrage sind die Ergebnisbeschreibung, die WHERE-Klausel und Ergebnismodifikatoren. Die Ergebnisbeschreibung legt die Abfrageart und damit den Resultattyp fest. Die gebräuchlichste ist die SELECT-Abfrage, welche eine Tabelle zurückgibt. Sie enthält die Werte, die mit den Variablen der WHERE-Klausel gebunden wurden und zu einem Treffer führten. ASK liefert eine Wahrheitswert, der angibt, ob der RDF-Graph das in der WHERE-Klausel beschriebene Graph-Pattern enthält. DESCRIBE gibt Tripel zu einer Ressource zurück und CONSTRUCT dient zum Erstellen und Verändern von RDF-Graphen. Die WHERE-Klausel besteht aus einem Graph-Pattern, das die gesuchten Informationen beschreibt. Marx et al. [2013] definieren Graph-Pattern als ein einzelnes Tripel-Pattern, die Verknüpfung von Tripel-Pattern durch AND, OPTIONAL, UNION oder ein Graph-Pattern in Verbindung mit einer Filterbedingung. Ein Tripel-Pattern unterscheidet sich von einem Tripel darin, dass an jeder Position des Tripels auch eine Variable stehen kann und mindestens eine Variable enthalten sein muss. Die Ergebnismenge wird mithilfe von Modifikatoren weiter eingeschränkt, sortiert oder aggregiert.

Das Listing 2.1 zeigt eine SPARQL-Abfrage vom Typ SELECT, dessen Ergebnistabelle die Werte enthält, die für die Variable `?label` in das Graph-Pattern eingesetzt werden können und eine Übereinstimmung im RDF-Graph ergeben. Durch den Modifikator DISTINCT werden Duplikate aus der Ergebnisliste gestrichen. Der Modifikator ORDER BY ASC bewirkt eine aufsteigende Sortierung und die Modifikatoren OFFSET 0 LIMIT 1 schränken das Ergebnis auf den Wert an der ersten Stelle ein. Das Graph-Pattern ist eine AND-Verknüpfung von vier Tripel-Pattern.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT DISTINCT ?label
WHERE {
    ?album rdf:type dbo:Album .
    ?album dbo:artist res:Elvis_Presley .
    ?album dbo:releaseDate ?date .
    ?album dbo:recordLabel ?label .
}
ORDER BY ASC(?date)
OFFSET 0 LIMIT 1
```

Listing 2.1: Beispiel einer SPARQL-Abfrage

2.3.2 Question Answering

Laut LODStats[3] enthält DBPedia zum April 2016 1,4 Milliarden Tripel. Nach der Seite LOD cloud diagram[4] enthält DBPedia 9,5 Milliarden zum Juni 2017 und Freebase 337 Millionen zum Juli 2016. Benutzer müssen die Abfragesprache beherrschen und die Struktur und Beziehungen in der Wissensdatenbank, also zugrundeliegende Ontologien und Vokabulare kennen. QA-Systeme versuchen für ein breiteres Publikum einen einfacheren Zugriff auf Wissen zu ermöglichen, wobei ein Kompromiss zwischen der leichten Bedienbarkeit und der Ausdrucksmächtigkeit gefunden werden muss. Die Komplexität, wie z.B. der Zugriff auf mehrere verteilte Wissensdatenbanken, soll verborgen werden. Sprachschnittstellen sind von besonderem Interesse, da sie auf intuitive Weise eine Möglichkeit bieten, die eigenen Informationsbedürfnisse auszudrücken. Hierbei wird sich der Mittel der maschinellen Sprachübersetzung bedient. Viele QA-Systeme übersetzen Fragen in Tripel, die mit RDF-Wissensdatenbanken abgeglichen werden. So spiegelt die in Listing 2.1 dargestellte SPARQL-Abfrage die Frage *For which label did Elvis record his first album?* wider.

[3]LODStats: `http://stats.lod2.eu`
[4]LOD cloud diagram: `http://lod-cloud.net`

QA ist wie Semantic Search ein Teilgebiet des Information Retrieval Fachgebiets, welches „die Wissenschaft, die Technik und Praxis des Suchens und Findens von Informationen“ [Stock, 2007] ist. Anders als bei der Semantic Search sollen nicht die nach einer bestimmten Klassifikation k-besten Ergebnisse zurückgegeben, sondern eine natürlichsprachliche Frage präzise beantwortet werden. Man unterscheidet zwischen QA innerhalb einer geschlossenen Domäne, wie z.B. der Medizin, und universellem, domänenoffenem QA. Die Herausforderungen für QA-Systeme sind nach Lukovnikov et al. [2017]:

Lexikalische Lücken Die Wissensdatenbank benutzt einen anderen als der in der Frage verwendete Ausdruck z.B. ein Synonym für die gleiche Beziehung,

Ambiguität Der gleiche Ausdruck kann für semantisch Verschiedenes verwendet werden (z.B. Homonyme oder Wörter, die für mehrere Entitäten stehen wie Mitglied für Vereins, Team- oder Bandmitglied),

keine klaren Wissensgrenzen QA-Systeme können schwer entscheiden, ob eine Frage mit der gegebenen Wissensdatenbank überhaupt beantwortbar ist.

Standard-Beurteilungsmetriken für QA-Systeme sind nach Makhoul et al. [1999] Genauigkeit, engl. Precision, Trefferquote, engl. Recall, sowie das F1-Maß. Genauigkeit und Trefferquote sind Metriken für Mengen. Die Genauigkeit gibt an, welcher Anteil der zurückgegebenen Ergebnismenge relevant ist und die Trefferquote, welcher Anteil der relevanten Lösungen zurückgegeben wird. Der F1-Wert ist ein kombiniertes Maß dieser beiden Werte und kann zwischen 0 und 1 liegen, wobei 1 der beste Wert ist.

3 Aktuelle Forschung

Dieses Kapitel widmet sich ausführlich dem Forschungsansatz von Soru et al. [2017] zum Question Answering mit Linked Open Data unter Benutzung von rekurrenten neuronalen Netzen, denn der ist die Basis, auf welche diese Arbeit aufbaut. Im zweiten Teil werden andere Forschungsarbeiten zum Question Answering mit Linked Open Data vorgestellt. Und anschließend werden QA-Datensätze, ihre Charakteristika und ihre Relevanz für das Forschungsfeld betrachtet.

3.1 Die NSpM: SPARQL als Fremdsprache

Der Kerngedanke von Soru et al. [2017] ist, SPARQL als Fremdsprache anzusehen und direkt vom natürlichsprachlichen Ausdruck in die konkrete SPARQL-Abfrage zu übersetzen, was somit einen End-zu-End-Ansatz darstellt. Sie führen den Begriff Neural SPARQL Machine (NSpM) ein. Die NSpM besteht aus drei Komponenten, dargestellt in Abbildung 3.1: Generator, Learner und Interpreter. Auf die in der Abbildung grau hinterlegten Felder wird in Abschnitt 4.1 und 4.2 eingegangen. Der Generator erzeugt die Trainingsdatenmenge mithilfe von Template-Fragen. Der Learner trainiert das Sequenz-zu-Sequenz Modell, das zum Ziel hat, die bedingte Wahrscheinlichkeit für die korrekte Ausgabe unter der zugehörigen Eingabe zu maximieren. Der Interpreter erzeugt die finale Version aus den vom Modell generierten SPARQL-Abfragen, wenn sich das System nicht mehr im Trainingsmodus befindet. In den folgenden Kapiteln werden diese Komponenten genauer betrachtet.

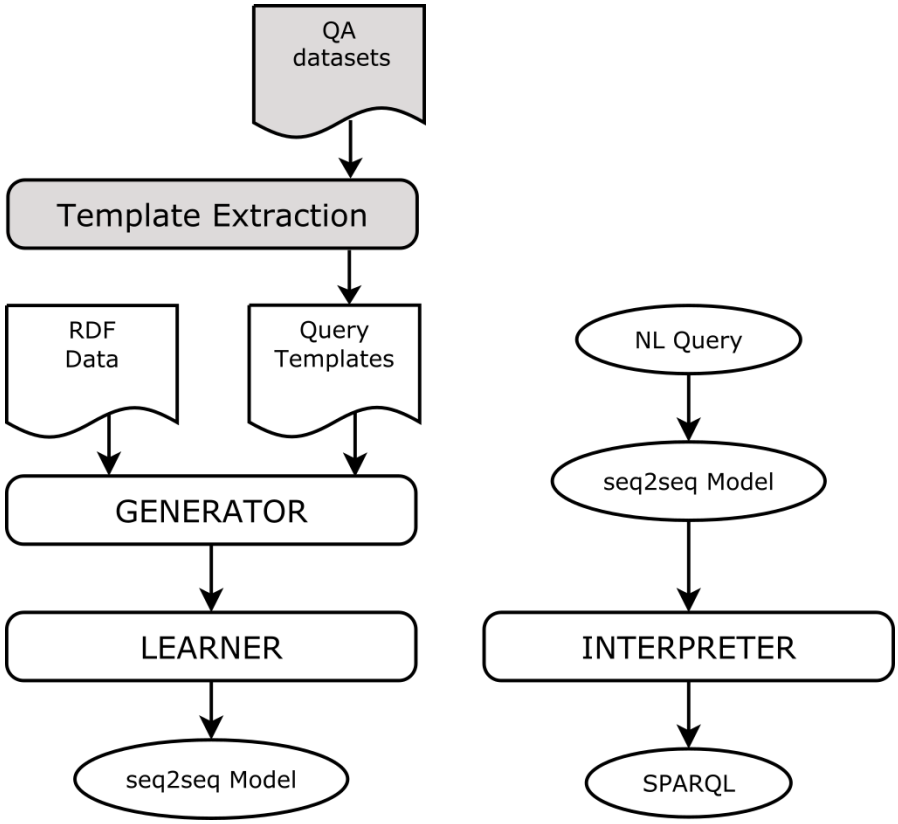

Abbildung 3.1: Architektur der NSpM nach Soru et al. [2017], ergänzt um Vorarbeitsschritt (grau hinterlegt)

3.1.1 Learner

SPARQL als Fremdsprache kann wörtlich genommen werden. Der Learner benutzt derzeit das von TensorFlow[1] veröffentlichte Sequenz-zu-Sequenz-Modell Luong et al. [2017] mit LSTM zur NMT. Allerdings nicht von Vietnamesich in Englisch oder Englisch in Deutsch übersetzt wie in Luong et al. [2017], sondern von Englisch in SPARQL, wie im Listing 3.1 an den Parametern `src` und `tgt` zu erkennen. Die Parameter `train_prefix`, `dev_prefix`, `test_prefix` und `vocab` spezifizieren die jeweiligen Datensätze bzw. das Vokabular. Trainings- und Testdaten dienen, wie in Kapitel 2.1.1 erwähnt, zum Trainieren der Gewichte und zur Bewertung des trainierten Modells. Mithilfe des Development-Datensatzes entscheidet das Modell zwischen alternativen Hyperparametereinstellungen (vgl. Abschnitt 2.1.2) [Neubig, 2017]. Die

[1] TensorFlow: `https://www.tensorflow.org`

```
python -m nmt.nmt --src=en --tgt=sparql --vocab_prefix=../$1/vocab
    --dev_prefix=../$1/dev --test_prefix=../$1/test --train_prefix=../$1/train
    --out_dir=../$1_model --num_train_steps=$2 --steps_per_stats=100
    --num_layers=2 --num_units=128 --dropout=0.2 --metrics=bleu
```

Listing 3.1: Befehl zum Trainieren der NSpM

Architektur des Sequenz-zu-Sequenz-Modells und die Trainingslänge kann durch die Anzahl der Schichten (`num_layers`), der versteckten Einheiten (`num_units`) und der Epochen (`num_train_steps`) bestimmt werden. Mit `dropout` lässt sich ein Mechanismus zur Vermeidung von Überanpassung einstellen. `steps_per_stats` beeinflusst die Ausgabe der Statistik. Diese liefert die Perplexity und mit dem Parameter `metrics` lässt sich ein weiteres Maß wie BLEU während des Trainings berechnen (vgl. Abschnitt 2.2.3 zur Erläuterung).

3.1.2 Generator

Der Generator erzeugt das Englisch-/SPARQL-Korpus aus den Templates, indem er die Platzhalterpositionen mit passenden Entitäten füllt. Die Templates und deren Instanziierungen bestimmen somit die QA-Einsatzfähigkeit der NSpM, da deren Übersetzungen trainingsabhängig sind. Für das Template-Paar (F, A) kann der Generator mit der SPARQL-Abfrage A' passende Instanzen finden wie z.B. I_1, I_2 und I_3.

F *What is <A> about?*

A
```
SELECT ?x WHERE { <A> dct:subject ?x }
```

A'
```
SELECT DISTINCT(?a), (str(?laba)as ?la) WHERE {
    ?a dct:subject [] . ?a rdfs:label ?laba . FILTER(lang(?laba)= 'en')
}
```

I_1 *What is Statue of Liberty about?*

```
SELECT ?x WHERE { dbr:Statue_of_Liberty dct:subject ?x }
```

I_2 *What is Glenfinnan Monument about?*

```
SELECT ?x WHERE { dbr:Glenfinnan_Monument dct:subject ?x }
```

I_3 *What is Monument to the Battle of the Nations about?*

```
SELECT ?x WHERE { dbr:Monument_to_the_Battle_of_the_Nations dct:subject ?x }
```

Der Platzhalter `<A>` wird in der englischen Frage durch die englische Bezeichnung (Variable `?la` in A') und in der SPARQL-Abfrage durch die IRI (Variable `?a` in in A') ersetzt. Problematisch ist die Anzahl der gültigen Matches für diese Abfragen. Für das Beispiel sind es ca. 24 Millionen. Einschränken lässt sich das durch Zusatzbedingungen, z.B. dass die Entität von einem bestimmten Typ ist, wie z.B. der `dbo:Monument`-Klasse. Dafür wird die WHERE-Klausel um das Tripel-Pattern `?a a dbo:Monument` ergänzt. Bei der Verwendung erreicht man für das Beispiel eine Reduktion auf 3350 Ergebnisse. Soru et al. [2017] verfolgen mit der Verwendung dieser Klasse und dazu passender Template-Fragen den Aufbau eines kohärenten Datensatz bezüglich dieser Domäne und der Wissensdatenbank DBPedia. Dadurch erreichen sie, wenn jedes der 38 Template-Paare 300 mal instanziiert wird, dass eine Entität im Durchschnitt 13 - 14 mal im Datensatz vorkommt, was bedeutet, dass sooft das Sequenz-zu-Sequenz-Modell die Zuordnung zwischen der englischen Bezeichnung und der IRI lernen kann. Wenn jedes Template 600 mal instanziiert wird, kommt jede Entität durchschnittlich 23 - 24 mal vor. Ihre Experimente mit diesen zwei Datensätzen zeigen, dass der Datensatz, mit mehr Beispielen je Template eine bessere Übersetzungsqualität besitzt.

3.1.3 Interpreter

Da SPARQL als Fremdsprache als End-zu-End-Ansatz beschrieben ist, mag es merkwürdig erscheinen, dass nach der Übersetzung durch das Sequenz-zu-Sequenz-Modell erst eine weitere Komponente die finale Übersetzung ausgibt. Dies ist der Tatsache geschuldet, dass SPARQL eine formale Sprache ist und der Adressat ein SPARQL-Endpunkt, der die Nachricht interpretiert und anders als ein Mensch

Syntaxfehler nicht akzeptiert. Interpunktionszeichen, Klammern und andere Sonderzeichen spielen für die Syntax und Semantik der formalen Sprache eine wesentliche Rolle und werden deshalb durch Wörter kodiert, damit sie im Vokabular enthalten sind. Das findet bereits Berücksichtigung bei der Erzeugung der Trainingsdaten durch den Generator. Die erste Beispielinstanziierung I_1 aus dem vorigen Abschnitt bekommt das Sequenz-zu-Sequenz-Modell also eigentlich so zu Gesicht:

```
SELECT var_a WHERE brack_open dbr_Statue_of_Liberty dct_subject var_a brack_close
```

Bisher hat der Interpreter keine weitere Aufgabe, als diese Kodierung rückgängig zu machen. Soru et al. [2017] zeigen in ihrer Arbeit aber auch auf, dass der Interpreter anspruchsvollere Aufgaben übernehmen kann, die sich z.B. aus der Eingabe von komplexen Fragen, die die NSpM zuvor nicht gelernt hat, ergeben und dann in der Überarbeitung des generierten SPARQL bestehen.

3.2 Andere Arbeiten zum KB-basierten QA

Zhang et al. [2016] unterscheiden beim KB-basierten QA zwischen zwei Hauptforschungsrichtungen. Die eine basiert auf semantischem Parsing und die andere ist Information Retrieval-basiert. Bei der Parsing-basierten Methode wird ein semantischer Parser konstruiert, der die natürlichsprachliche Frage zunächst in eine Zwischenform konvertiert, die dann in einen strukturierten Ausdruck konvertiert wird. Ein Beispiel dazu von Berant et al. [2013]:

Äußerung: *Which college did Obama go to?*
Logische Zwischenform: `(and (Type University)(Education BarackObama))`
Antwort: *Occidental College, Columbia University*

Weitere Vertreter dieser Richtung sind Dubey et al. [2016] und Yih et al. [2015], wobei die Methode der letztgenannten schon als Mischform mit der zweiten Forschungsrichtung angesehen werden kann, da sie eine Äußerung zunächst mit einer formalen, logischen Repräsentation umschreiben, dann aber stufenweise einen Abfrage-Graphen unter früher Einbeziehung der Wissensdatenbank generieren.

Denn das Suchen von Antwortkandidaten mithilfe der Wissensdatenbank anhand der in der natürlichsprachlichen Frage gegebenen Informationen entspricht dem Information-Retrieval-basierten Ansatz. Aus den Antwortkandidaten wird sich für denjenigen entschieden, der nach einer bestimmten Klassifizierung am besten abschneidet. Bei SINA [Shekarpour et al., 2015] wird eine Eingabe als Menge von Schlüsselwörtern modelliert, wobei Benutzer sowohl Schlüsselwörter als auch natürlichsprachliche Fragen benutzen können. Ein Hidden Markov Model, ein stochastischer Automat, entscheidet zwischen möglichen Abfrage-Kandidaten, wobei diese mit dem Jaccard-Ähnlichkeitsmaß für Mengen bewertet werden. Der richtige Abfragetyp wird durch vordefinierte Regeln ermittelt und Ressourcen mit DBPedia Spotlight verlinkt. TBSL [Unger et al., 2012] ist ein template-basierter Ansatz. Natürlichsprachliche Fragen werden zu SPARQL-Templates konvertiert, deren Platzhalter unter Benutzung von statistischer Entitätsidentifikation und Prädikatserkennung instanziiert werden. Die Detektion des richtigen Templates wird mit Positiv- und Negativbeispielen gelernt. Er unterstützt komplexe Abfragen, also Abfragen, die Filter und Modifikatoren enthalten.
Auch neuronaler Netze wird sich in der zweiten Forschungsrichtung bedient. Hierbei werden die Fragen und Antworten als Vektoren kodiert und die Ähnlichkeit des Fragevektors mit den möglichen Antwortvektoren im Vektorraum entscheidet darüber, welcher Kandidat als Antwort zurückgegeben wird. Die Vektoren müssen zuvor erlernt werden. Bei Lukovnikov et al. [2017] lernt das neuronale Netz Subjekt-Prädikat-Paare zu klassifizieren, die es ermöglichen relevante Fakten zu einer gegeben Frage zu ermitteln. Dies ist wie bei Soru et al. [2017] ein End-zu-End-Ansatz. Sie benutzen ebenfalls RNN, beschränken sich derzeit aber noch auf simple Fragen, für deren Beantwortung nur ein einziger Fakt benötigt wird. Zum Training benutzen sie 21.687 Fragen des Datensatzes SimpleQuestions, der auf der Wissensdatenbank Freebase basiert. Um besser mit seltenen und OOV-Wörtern umgehen zu können, werden Entitäten und Prädikate auch zeichenbasiert kodiert. Dagegen ist die wortbasierte Kodierung in der Lage wortbasierte Semantik zu erschließen mithilfe der Word Embeddings. Um diese Fähigkeit nicht zu verlieren, wird ein hybrides Modell benutzt. Auch Luong [2016] veröffentlichte ein hybrides Modell und gibt das schnellere Training im Vergleich zum rein zeichenbasierten Modell als weiteren Vorteil an.

Zhang et al. [2016] nutzt ein neuronales Netz mit Attention-Mechanismus.
Als weitere Vertreter dieser Richtung sind Bordes et al. [2015], Golub und He [2016], Yin et al. [2016] und Dai et al. [2016] zu nennen. Ein wesentlicher Unterschied von Soru et al. [2017] ist, dass sie das RNN nicht benutzen, um Antwortkandidaten klassifizieren zu können, sondern als reiner Übersetzer von einer natürlichsprachlichen Frage zu der SPARQL-Abfrage.

3.3 Datensätze für KB-basiertes QA

Tabelle 3.1 stellt Datensätze gegenüber, die von KB-basierten QA-Methoden zu Trainings- und Testzwecken benutzt werden. Wesentlich unterscheiden sie sich in ihrer Größe (Anzahl der Fragen), der Fragenkomplexität und der Antwortform.

Datensatz	Größe	Entitäten	Prädikate	Wissensdb.	Antwortform	Fragenkomplexität
SimpleQuestions	108.442	k.A.	k.A.	Freebase	Freebase-Tripel	einfach
Free917	917	733	852	Freebase	λ-Calculus	komplex
WebQuestions	5810	k.A.	k.A.	Freebase	Text	komplex
QALD-7-Train	215	240	131	DBpedia	SPARQL	hoch komplex
LC-QuAD	5000	5042	615	DBpedia	SPARQL	komplex

Tabelle 3.1: Vergleich von Datensätzen

Lukovnikov et al. [2017] verwenden den Datensatz SimpleQuestions. Dieser besteht aus 108.442 englischen von Menschen verfassten Fragen und ist damit einer der größten QA-Datensätze. Zu jeder Frage gehört ein Freebase-Tripel, das Fakten in der Form `(Subjekt, Beziehung, Objekt)` ausdrückt. Der Datensatz wird ebenfalls von Bordes et al. [2015], Golub und He [2016], Yin et al. [2016] und Dai et al. [2016] benutzt. All den Fragen des Datensatzes ist gemeinsam, dass sie einfacher Natur sind, was beim KB-basiertem QA heißt, dass zur Beantwortung der Frage nur ein einziger Fakt, also ein Tripel, notwendig ist. Die Ansätze, die ihre System mit diesem Datensatz trainieren, beschränken sich demnach auch auf die Beantwortung einfacher Fragen. Ein größerer Datensatz, der auch Fragen enthält, für deren Beantwortung mehrere Fakten herangezogen werden müssen oder Aggregatfunktionen, ist Free917. Er enthält 917 Fragen und zu jeder Frage deren Repräsentation im λ-Calculus-Stil. Genutzt wird er von Cai und Yates [2013] für

ihre Parsing-basierte Methode. Ein vom Fragenstil ähnlicher Datensatz mit 5810 Fragen und zugehörigen Antworten ist WebQuestions, der von Yih et al. [2015] benutzt wird. Dubey et al. [2016] benutzen einen Datensatz der QALD-Serie. Die Kampagne Question Answering over Linked Open Data (QALD)[2] organisiert jährlich Wettbewerbe zu KB-basiertem QA und stellt dafür Trainings- und Testdatensätze zur Verfügung. Serie 7 umfasst 215 Trainings- und 50 Testfragen. Sie sind mehrsprachig gegeben, Schlüsselwörter, Antworten und SPARQL-Abfragen sind annotiert. Die Fragen sind zum Teil sehr komplex. Für die Beantwortung kann es erforderlich sein Aggregat-, Filterfunktion und/oder Modifikatoren einzusetzen. Usbeck et al. [2017] demonstrieren das mit folgenden Beispielen:

Zählfragen: *How many children does Eddie Murphy have?*
Superlative: *Which museum in New York has the most visitors?*
Vergleiche: *Is Lake Baikal bigger than the Great Bear Lake?*
Zeitliche Aggregation: *How many companies were founded in the same year as Google?*

Trivedi et al. [2017] bemängeln an den vorgestellten Datensätzen Umfang und/oder Komplexität. Sie nahmen das zum Anlass mit Large-Scale Complex Question Answering Dataset (LC-QuAD), einen Datensatz zu schaffen, der Wert auf eine große Anzahl, Vielfalt und Komplexität der Fragen legt. Der Datensatz enthält 5000 Fragen, die alle von einem Menschen gestellt sein könnten. 18 % der Fragen sind einfache Fragen. Bei den restlichen sind mindestens zwei Tripel zur Beantwortung notwendig und/oder die COUNT-Aggregatfunktion. Sie begründen die Relevanz eines solchen Datensatzes für beide Forschungsrichtungen des KB-basierten QA. Zum einen gibt er Entwicklern der Information-Retrieval-basierten Forschungsrichtung Versuchsobjekte zur Beantwortung komplexer Fragen in die Hand. Zum anderen können auch die Parsing-basierten Methoden profitieren, die auf händisch erstellter Regeln beruhen. Die größenmäßig limitierten Datensätze behindern das Automatisieren dieser manuellen Schritte. Mit der Verfügbarkeit vieler komplexer Fragen kann untersucht werden, ob die Präzision der Parsing-basierten Methoden verbessert werden kann.

[2]QALD: `https://qald.sebastianwalter.org`

4 Korpus-Generierung

Im Abschnitt 3.1.2 wird beschrieben, wie Soru et al. [2017] einen Trainingsdatensatz für ihr Sequenz-zu-Sequenz-Modell generieren. Dieses Verfahren soll zur Erzeugung eines noch umfangreicheren Englisch-SPARQL genutzt werden. Damit kann zum einen der Forschungsfrage nachgegangen werden, ob die NSpM auch für größere, domänenunspezifische Datensätze gute Ergebnisse liefert und damit skaliert. Und zum anderen können auch andere ML-basierte QA-Verfahren davon profitieren, wie in Abschnitt 3.3 beschrieben.

Soru et al. [2017] instanziieren generische Templates, um einen adäquaten Trainingsdatensatz mit Englisch-SPARQL-Übersetzungen zu erzeugen. Dafür haben sie 38 QA-Template-Paare manuell entworfen. Dieser aufwändige händische Prozess wird in dieser Arbeit gescheut. Der Clou ist, den Prozess zunächst umzukehren, also aus vorhandenen konkreten Frage-SPARQL-Paaren generische Templates zu extrahieren. Dafür bieten sich die in Abschnitt 3.3 vorgestellten Datensätze der QALD-Serie und LC-QuAD an, die strukturell vielfältige und komplexe Fragen enthalten.

Im ersten Abschnitt dieses Kapitels wird beschrieben, wie die Template-Paare extrahiert werden. Die Generatorkomponente benötigt, um passende Instanziierungskandidaten für die Templateplatzhalter zu ermitteln, eine weitere SPARQL-Abfrage, die im Rest der Arbeit als Generator-Abfrage bezeichnet wird. Der zweite Abschnitt des Kapitels beschäftigt sich mit der Konstruktion der Generator-Abfrage. Der letzte Abschnitt stellt den Datensatz bzw. das Englisch-SPARQL-Korpus vor, das sich mit den extrahierten Templates, den Generator-Abfragen und der weiterentwickelten Generator-Komponente generieren lässt.

Abbildung 3.1 ordnet den Prozess der Template-Extraktion und der Konstruktion der Generator-Abfrage, die grau hinterlegten Felder, kontextuell in den NSpM-Prozess ein. Die Abbildung 4.1 dient der Veranschaulichung der Template-Extraktion und -Instanziierung und zeigt, wofür die Originalentität eines Frage-SPARQL-Paars benutzt wird.

4.1 Template-Extraktion

Die 215 Englisch-SPARQL-Paare vom QALD-7-Train Datensatz wurden händisch in Templates überführt, die 5000 des LC-QuAD-Datensatzes maschinell. Hierbei war von Vorteil, dass der Datensatz bereits teilweise maschinell erstellt wurde und Graph-Pattern-Templates dem SPARQL zugrunde liegen, 38 einzigartige an der Zahl. Listing 4.1[1] zeigt einen Beispieleintrag des Datensatzes. Die `verbalized_question` ist ein Artefakt des automatischen Erstellungsprozesses und `corrected_question` die von einem Korrektor überarbeitete Fassung. Das zugehörige Graph-Pattern-Template findet sich in Listing 4.2. Es kann nicht direkt als NSpM-Template übernommen werden, da es noch zu generisch ist und z.B. auch Platzhalter für Prädikate enthält. Dafür wurden die Keys `triples` und `placeholders` ergänzt, die nicht aus dem Originaldatensatz stammen.
`placeholders` gibt an, welche Platzhalter für das NSpM-Template übernommen werden. Damit lässt sich ausmachen, welche konkrete IRI im SPARQL durch einen Platzhalter ersetzt werden muss. `triples` enthält die Struktur des Graph-Patterns.

[1] Ausgeschriebene IRIs werden zur Übersichtlichkeit in der ganzen Arbeit durch die Prefix-Schreibweise ersetzt.
`PREFIX dbo: <http://dbpedia.org/ontology/> PREFIX dbp: <http://dbpedia.org/property/> PREFIX dbr: <http://dbpedia.org/resource/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX yago: <http://dbpedia.org/class/yago/>`

```
{
    "_id": "ede3db931b55452880393aec59f27131",
    "sparql_template_id": 301,
    "sparql_query": "SELECT DISTINCT ?uri WHERE {?uri dbp:notableInstruments
        dbr:Gibson_Les_Paul . ?uri rdf:type dbo:MusicalArtist}",
    "verbalized_question": "Who are the <musical artists> whose <notable
        instruments> is <Gibson Les Paul>?",
    "corrected_question": "What are some famous artists who rocked a Les Paul?"
}
```

Listing 4.1: Beispieleintrag des LC-QuAD-Datensatzes

```
{
    "template": "SELECT DISTINCT ?uri WHERE {?uri <%(e_to_e_out)s> <%(e_out)s>
        . ?uri rdf:type class }",
    "id": 301,
    "n_entities": 1,
    "type": "vanilla",
    "triples": [
        {
            "subject": "?uri",
            "predicate": "<%(e_to_e_out)s>",
            "object": "<%(e_out)s>"
        },
        {
            "subject": "?uri",
            "predicate": "rdf:type",
            "object": "class"
        }
    ],
    "placeholders": ["<%(e_out)s>"]
}
```

Listing 4.2: LC-QuAD-Graph-Pattern-Template

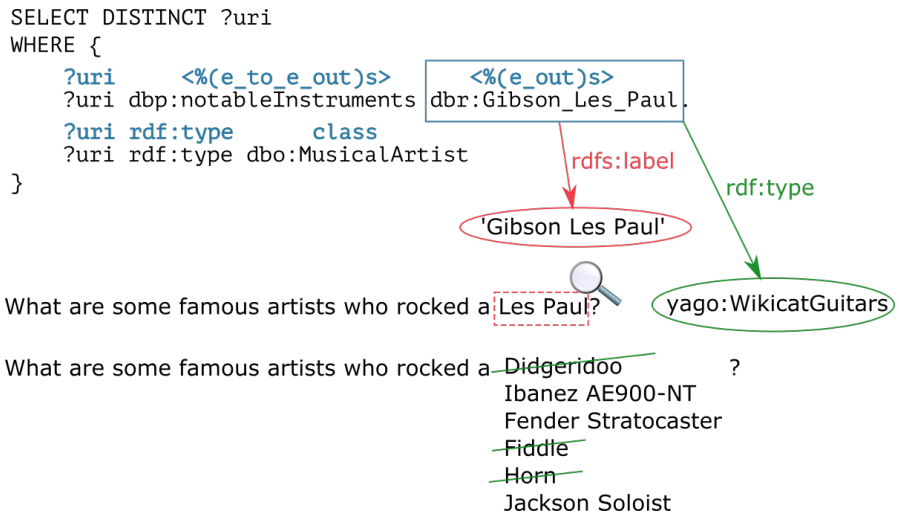

ABBILDUNG 4.1: Template-Extraktion und -Instanziierung,
blau: LC-QuAD-Graph-Pattern-Template, rot: Label, das in der Frage durch einen Platzhalter ersetzt werden soll, grün: Typ wird als Instanziierungsfilter benutzt

In der Abbildung 4.1 ist das LC-QuAD-Graph-Pattern-Template in blauer Schrift über die zugehörigen Tripel-Pattern-Komponenten geschrieben. Bekannt ist, dass `<%(e_out)s>` die Position eines Platzhalters ist, weshalb `dbr:Gibson_Les_Paul` als zu ersetzende Ressource ausgemacht werden kann. Die Bezeichnung dieser Ressource muss sich demnach in der natürlichsprachlichen Frage wiederfinden und ersetzen lassen, was in der Abbildung rot markiert ist. Listing 4.3 zeigt einen Ausschnitt der Lösung. Zunächst wird es mit einer RegEx-Suche versucht (Zeile 5-8). Ist diese nicht erfolgreich, weil wie im Beispiel die Bezeichnung durch einen gebräuchlicheren Namen ersetzt wurde, dann wird eine Fuzzy-Suche angewandt (Zeile 10), die die Levenshtein-Distanz nutzt. Scheitert auch diese, wird mit der `verbalized_question` weiterverfahren, da diese stets die vollständigen Label enthält. Das zum LC-QuAD-Eintrag 4.1 extrahierte NSpM-Template lautet somit:

What are some famous artists who rocked <A>?

```
SELECT DISTINCT ?uri WHERE {
    ?uri dbp:notableInstruments <A> .
    ?uri rdf:type dbo:MusicalArtist
}
```

```
...
names = getattr(entity, "names")
escapedNames = map(re.escape, names)
placeholder = getattr(entity, "placeholder")
matchInQuestion = re.search(r"(" + "|".join(escapedNames) + r")s?", question,
    flags=re.IGNORECASE)

if (matchInQuestion):
    question = string.replace(question, matchInQuestion.group(0), placeholder)
else:
    fuzzySearchMatch = fuzzySearch(names, question)
    if (fuzzySearchMatch):
        question = string.replace(question, fuzzySearchMatch.group(0),
            placeholder)
    else:
        ...
...
```

LISTING 4.3: Funktionsauszug zum Ersetzen von Namen mit Platzhaltern in der natürlichsprachlichen Frage

Eine nachträglich manuelle Revision der Templates ist notwendig. Durch die Ersetzung der Entitäten mit Platzhaltern entstehen Template-Duplikate, die entfernt werden, womit die Differenz der Gesamtanzahl in den Tabellen 3.1 und 5.1 zu begründen ist. Die Fuzzy-Suche führt in 83 Fällen zu einer fehlerhaften Platzhalterdetektion wie z.B.

Where <A> of 16 take place?

In 32 Fällen tritt das Problem von semantisch abweichende Frage-SPARQL-Paaren auf, was an folgenden Beispielen ersichtlich ist:

verbalized	*What is the <draft team> of the <Dave Bing> and <Ron Reed>?*
corrected	*In which team did Dave Bing and Ron Reed started their basketball career?*
Template	*In which team did <A> and <B> started their basketball career?*
SPARQL	`SELECT DISTINCT ?uri WHERE { dbr:Dave_Bing dbp:draftTeam ?uri. dbr:Ron_Reed dbr:draftTeam ?uri . }`
Template	`SELECT DISTINCT ?uri WHERE { <A> dbp:draftTeam ?uri. <B> dbr:draftTeam ?uri . }`

Der Korrektor hat die Information ergänzt, dass es sich bei Dave Bing und Ron Reed um Basketballspieler handelt. Dies spiegelt sich aber nicht im SPARQL wider. Damit geht bei der Übersetzung Information verloren. In diesem Fall sollte *basketball carrer* mit *career* ersetzt werden. Nicht immer hat der Korrektor Schuld, manchmal tritt der Fehler auch erst mit Umwandlung einer Frage in ein Template auf, wie hier:

verbalized	*What is the <location city> of the <american football Team> which is the <debut team> of <Lee Robinson (American football)> ?*
corrected	*What city has the football team in which Lee Robinson debuted?*
Template	*What city has the football team in which <A> debuted?*
SPARQL	`SELECT DISTINCT ?uri WHERE { dbr:Lee_Robinson_(American_football) dbo:debutTeam ?x . ?x dbo:locationCity ?uri . }`
Template	`SELECT DISTINCT ?uri WHERE { <A> dbo:debutTeam ?x . ?x dbo:locationCity ?uri . }`

In diesem Fall wurde *football* nicht bewusst ergänzt, sondern war bereits Teil der Originalfrage. Da das Wort aber in der IRI enthalten ist, welche durch einen Platzhalter ersetzt wird, spiegelt sich die Spezifik des generierten Frage-Templates erneut nicht im SPARQL-Template wider.

4.2 Generator-Abfrage-Konstruktion

Soru et al. [2017] erstellen die Abfragen zur Instanziierung der 38 Template-Paare manuell. Dieser Aufwand wird für die 5165 neuen Template-Paare durch eine Transformation des SPARQL-Templates umgangen. Der Platzhalter im Template wird mit einer Variablen ersetzt und die Abfrage wird um eine Label-Bedingung ergänzt. Weiterhin werden Abfrage-Modifikatoren, die nach der WHERE-Klausel kommen, abgeschnitten. So kann das Template T mit den Ergebnissen der Abfrage T' instanziiert werden:

T	T'	T''
`SELECT DISTINCT ?uri`	`SELECT DISTINCT ?a,`	`SELECT DISTINCT ?a,`
	`(str(?laba)as ?la)`	`(str(?laba)as ?la)`
`WHERE {`	`WHERE {`	`WHERE {`
`?uri rdf:type <A> .`	`?uri rdf:type ?a .`	`?uri rdf:type ?a .`
`?uri dbo:height ?num`	`?uri dbo:height ?num .`	`?uri dbo:height ?num .`
`}`	`?a rdfs:label ?laba .`	`?a rdfs:label ?laba .`
`ORDER BY DESC(?num)`	`FILTER(lang(?laba)= 'en')`	`FILTER(lang(?laba)= 'en').`
`OFFSET 0 LIMIT 1`		`?a rdfs:subClassOf dbo:Athlete`
	`}`	`}`

Nicht alle der zurückgegebenen Entitäten passen zur englischen Frage. So wäre mit dem obigen Beispiel auch die Instanziierung *Who is the tallest automobile?* möglich. Eine andere Variante ist, Entitäten zu benutzen, die die gleiche Klasse wie die Entität besitzen, die in der Originalfrage benutzt wurde, aus der das Template generiert wird. Problematisch ist, dass die meisten Entitäten aber zu mehreren Klassen gehören und sich automatisch nicht leicht entscheiden lässt, welche für welche Frage die passendste wäre. Deshalb wird eine Kombination dieser beiden Ansätze genutzt. Nach passenden Instanzen wird mit der oben beschrieben transformierten SPARQL-Abfrage gesucht, wobei Typ-Zusatzbedingungen hinzugefügt werden. Dafür wird eine Menge von Klassen definiert, deren Auftreten im Datensatz nach Analyse der Themen wahrscheinlich ist. Aufgelistet werden diese in Tabelle A.1 und A.2. Die Klassen der Entität, die ursprünglich an der Platzhalterposition stand, werden mit dieser Menge abgeglichen und der Platzhalter wird bei Überschneidung mit den entsprechenden Typinformation versehen. So wäre für die Template-Frage F aufgrund des Originals J_0 bekannt, dass der Platzhalter `<A>` eine Subklasse von `dbo:Athlete` ist. Der Generator findet passende Instanziierungen mit der Abfrage T''.

F *Who is the tallest <A>?*
J_0 *Who is the tallest basketball player?*

In Abbildung 4.1 wird dieser Vorgang grün hinterlegt. Dort sind passende Entitäten nicht Subklassen von `dbo:Athlete` sondern Ressourcen vom Typ `yago:WikicatGuitars`. Ein Problem des Ansatzes tritt bei Fragen mit mehreren Platzhaltern ein, wie

Does <A> have more episodes than <B>?

Wenn die Platzhaltervariablen in der Generator-Abfrage wie hier

```
SELECT DISTINCT ?a, ?b WHERE { ?a dbo:numberOfEpisodes ?x . ?b dbo:numberOfEpisodes ?y }
```
[2]

unabhängig voneinander sind[3], dann ist das Resultat dieser Abfrage ein Kreuzprodukt aus der Menge der für `?a` und der Menge für `?b` passenden Variablenbelegungen. Das bedeutet, dass der erste Wert der ersten Menge zunächst mit allen Werten der zweiten Menge verpaart wird, dann der zweite Wert der ersten Menge mit allen Werten der zweiten Menge und so fort. Dadurch häufen sich einige Werte, wenn man nur die k-ersten Antworttupel betrachtet, die benötigt werden, um k-mal ein Template zu instanziieren, was dazu führt, dass z.B. 200 der 300 Instanziierungen die gleiche Entität enthalten. Ein Lösungsversuch unter Verwendung einer SPARQL-Unterabfrage und der Aggregatfunktion SAMPLE scheiterte am Timeout des angefragten Triplestores.

Weiterhin generiert das beschriebene Instanziierungsverfahren für Fragen, deren zugehörige SPARQL-Abfrage keine Filterfunktion oder Modifikatoren enthält, im Falle von ASK-Abfragen nur Fragen, die eine positive Antwort zurückgeben und im Falle von SELECT-Abfragen nur Fragen, die eine nicht-leere Ergebnistabelle zurückgeben.

4.3 Das Englisch-SPARQL-Korpus

Mit den 5165 extrahierten und duplikatfreien QALD- und LC-QuAD-Templates, den zugehörigen Generator-Abfragen und der Weiterentwicklung der von Soru et al. [2017] entworfenen Generator-Komponente wurde ein Datensatz geschaffen, der 894.499 englische Fragen und ihre äquivalenten SPARQL-Abfragen umfasst. Der Generator kann nicht in jedem Fall 300 Instanzen für die Templates finden, was erklärt, warum die Anzahl der generierten Fragen nicht exakt 300 mal der

[2] Generator-Abfrage wurde zur Übersichtlichkeit gekürzt.

[3] Es existiert kein Pfad von der einen zu der anderen Variablen, wenn man das Graph-Pattern in einem Graphen darstellt.

B_1	*Who is the tallest automobile?*
B_2	*For which label did Aretha Franklin record his second single?*
B_3	*Was the wife of president Lincoln called Mary?*
	`ASK WHERE { res:Abraham_Lincoln dbo:spouse ?spouse .`
	`?spouse rdfs:label ?name . FILTER(regex(?name,'Mary'))}`
B_4	*Give me all books with more than 300 pages!*
	`SELECT DISTINCT ?uri WHERE { ?uri rdf:type dbo:Book .`
	`?uri dbo:numberOfPages ?x . FILTER (?x > 300)}`

Tabelle 4.1: Beispiele für Korpusdefizite

Anzahl der Templates entspricht. Für die 833 nicht-instanziierten Templates gibt es mehrere Ursachen. So gibt es Entitäten, die für einen Platzhalter eingesetzt werden könnten, aber keine englische Bezeichnung besitzen und damit ausscheiden. Des Weiteren gibt es Templates, deren Original-SPARQL-Abfragen kein oder ein negatives Ergebnis zurückgeben, weshalb nicht ausgeschlossen werden kann, dass das Graph-Pattern unerfüllbar ist. Als dritte Ursache ist ein Triplestore-Timeout anzunehmen, eventuell durch kurzzeitige Überlastung, da bei Wiederholung der Generator-Abfrage, Ergebnisse zurückgegeben wurden.

Die Tabelle 4.1 listet Beispiele für Defizite des Korpus auf. Werden weniger als 100 passende Instanziierungen gefunden, versucht der Generator die Zahl durch die Aufweichung der Platzhaltertypbedingungen zu vergrößern, was aber auch zu unstimmigen Fragen wie Beispiel B_1 führt. Eine weitere Schwäche des Verfahrens ist, dass Personal- und Possessivpronomen nicht berücksichtigt werden, wodurch ebenfalls fehlerhafte Fragen entstehen, siehe Beispiel B_2. Derzeit unterstützt das System nur die Ersetzung von Entitäten mit Platzhaltern und nicht die von Literalen, die häufig in Filterbedingungen verwendet werden, wie z.B. *Mary* in B_3 und *300* in B_4.

5 Evaluierung der NSpM-Übersetzung mit dem Korpus

Soru et al. [2017] zeigen die Tauglichkeit ihrer NSpM an einer geschlossenen Domäne. Die Forschungsfrage, welche in diesem Kapitel analysiert wird, ist, ob dieser Ansatz auch für domänenunspezifisches QA unter Benutzung größerer Trainingsdatensätze, wie dem im vorigen Kapitel erarbeitetem Englisch-SPARQL-Korpus, gute Ergebnisse liefert. Oder ob die geschlossene Domäne bzw. die starke Kohärenz der Fragen eine Voraussetzung ist.

Der erste Abschnitt widmet sich der Versuchsvorbereitung und Durchführung. Die Auswertung unter Begutachtung allgemein definierter Kriterien erfolgt im zweiten Abschnitt. Der letzte Abschnitt vermittelt durch die Betrachtung von Einzelfällen ein Gefühl für das Trainings- und Übersetzungsverhalten der NSpM.

5.1 Vorbereitung und Durchführung

Tabelle 5.1 zeigt die Verteilung der NSpM-Templates nach Themengebieten. Rechnet man die Templates beider Datensätze zusammen, stellen Kunst (24 %), Geographie (16 %), Sport (12 %) und Gesellschaft (10 %) die größten dar. Das Thema Kunst beinhaltet Fragen zu Musik (30 %), Film (42 %), Literatur (17 %) und

Thema	QALD-7		LC-QuAD		Gesamt	
	#	%	#	%	#	%
Kunst	49	24,5 %	1199	24,1 %	1248	24,2 %
Geographie	47	23,5 %	766	15,4 %	813	15,7 %
Sport	12	6,0 %	627	12,6 %	639	12,4 %
Gesellschaft	15	7,5 %	506	10,2 %	521	10,1 %
Wirtschaft	13	6,5 %	463	9,3 %	476	9,2 %
Biographie	37	18,5 %	207	4,2 %	244	4,7 %
Wissenschaft	9	4,5 %	150	3,0 %	159	3,1 %
IT	7	3,5 %	104	2,1 %	111	2,1 %
Zeit	6	3,0 %	0	0,0 %	6	0,1 %
Andere	5	2,5 %	944	19,0 %	948	18,4 %
Duplikate	15	8,3%	34	0,7 %	52	1 %
Gesamt	200		4966		5166	

Tabelle 5.1: NSpM Templates nach Themengebieten

anderem. Gesellschaftsbezogene Fragen sind Fragen zu Politik, Staat und Religion. Unter Biographie sind Fragen gebündelt, die Verwandtschaftsbeziehungen, Geburt, Tod, Ausbildung und Beruf einer Person thematisieren.

Für die Experimente werden Datensätze zusammengestellt, die auf Templates q mit $topic(q)$ als Thema aus folgenden Mengen bestehen:

1. QALD-7-Train
2. QALD-7-Train $\cup$ $\{q | q \in$ LC-QuAD $\wedge\ topic(q) \in \{$Biographie$\}\}$
3. QALD-7-Train $\cup$ $\{q | q \in$ LC-QuAD $\wedge topic(q) \in \{$Biographie, Kunst, Sport$\}\}$
4. QALD-7-Train $\cup$ LC-QuAD

Es wird die Strategie verfolgt, die Menge der Templates und damit die Trainingsmenge der NSpM stückweise zu vergrößern. Damit kann der These nachgegangen werden, ob die Hinzunahme von Trainingsdaten die Übersetzungsqualität steigert. QALD-7-Train stellt die Templates für das erste Experiment. Im zweiten

Experiment werden die Biographie-Templates des LC-QuAD-Datensatzes hinzugefügt. Die Wahl fällt auf diese Sparte, um die Vorkommenshäufigkeit der Entitäten im Trainingsdatensatz zu vergrößern, was wie in Kapitel 3.1.2 beschrieben, von Bedeutung ist. Es erhöht die Kohärenz des Datensatzes und trägt damit zu repräsentativen Word Embeddings bei (vgl. Abschnitt 2.2.4). Viele Fragen anderer Themenbereiche, insbesondere von Kunst und Sport, sind personenbezogen. Templates biographischer Fragen lassen sich mit den dort verwendeten Entitäten instanziieren. Hier zeigt sich ein weiterer Vorteil typisierter Platzhalter. Neben der Ausfilterung unpassenden Instanzen, lässt sich auch eine Verfeinerung vornehmen. Die Generatorkomponente wurde angepasst, Platzhalter, die mit Instanzen der bestimmten Klasse zu besetzen sind, mit Instanzen aus z.B. spezifischeren Unterklassen zu besetzen. Listing 5.1 zeigt einen Programmauszug mit der entsprechenden Funktionalität. In Zeile 15 wird geprüft, ob die Klasse eines Platzhalters einer zu ersetzenden Klasse entspricht. Diese werden durch das Dictionary `SPECIAL_CLASSES` definiert (Zeilen 1-11). Im Fall einer Ersetzung, wird die Generator-Abfrage um die entsprechenden Bedingungen ergänzt (Zeile 17). Für das Experiment sollen Platzhalter, die mit Instanzen der Klasse `dbo:Person` zu besetzen sind, mit Instanzen der spezifischeren Unterklassen besetzt werden: `yago:Wikicat21st-centuryActors`, `yago:WikicatEnglishMusicians`, `dbo:LacrossePlayer`, `yago:Honoree110183757`, `yago:Wikicat20th-centuryNovelists`.
Weiterhin wird `dbo:LacrossePlayer` anstatt `dbo:Athlete` verwendet, um ebenfalls die Zahl der möglichen Treffer kleiner zu halten. Der Typ eines Platzhalters wurde wie in Abschnitt 4.2 beschrieben unter Verwendung der Originalfragen und -instanzen ermittelt.

Tabelle 5.2 stellt Kennzahlen der Datensätze gegenüber. Die Sequenz-zu-Sequenz-Modelle werden jeweils mit 12.000 Iterationen trainiert. Datensatz i dient als Grundlage für Modell i. Dem Modell werden 80 % der Fragen zum Training zur Verfügung gestellt und jeweils 10 % für Test und Development (vgl. Abschnitt 3.1.1). Die Experimente laufen auf einem Linux-Rechner ohne GPU mit 64 Prozessoren und 256 GB RAM.

```
SPECIAL_CLASSES = {
    "dbo:Person": [
        "<http://dbpedia.org/class/yago/Wikicat21st-centuryActors>",
        "<http://dbpedia.org/class/yago/WikicatEnglishMusicians>",
        "<http://dbpedia.org/class/yago/Wikicat20th-centuryNovelists>",
        "<http://dbpedia.org/class/yago/Honoree110183757>",
        "dbo:LacrossePlayer"
    ],
    "dbo:Athlete": ["dbo:LacrossePlayer"]
}
...
CLASSES_REPLACEMENT = " where {{ ?{variable} a ?t . VALUES (?t) {{ {classes}
    }} . "
...

if normalized_target_class in SPECIAL_CLASSES and do_special_class_replacement:
    classes = " ".join(map(lambda c : "({})".format(c),
        SPECIAL_CLASSES[normalized_target_class]))
    generator_query = add_requirement(generator_query,
        CLASSES_REPLACEMENT.format(variable=variable, classes=classes))
...
```

Listing 5.1: Auszug aus Generator-Pythoncode zur Spezifikation von Platzhaltervariablen

Datensatz	1	2	3	4
Anzahl Templates	200	407	2.233	5.166
davon nicht instanziierte Templates	7	10	362	833
Anzahl generierte Fragen	51.292	99.274	390.444	894.499
Vokabulargröße Englisch	28.869	40.599	84.963	134.788
Vokabulargröße SPARQL	33.730	51.399	144.720	249.395

Tabelle 5.2: Übersicht der erzeugten Datensätze

Modell	1	2	3	4
Laufzeit (hh:mm)	3:03	4:11	10:33	21:29
Bester Test BLEU	0,634	0,620	0,619	0,601
Beste Test Perplexity	2,13	2,19	2,39	2,78

Tabelle 5.3: Kennzahlen zum Trainingsprozess der Modelle

5.2 Allgemeine Auswertung

In Tabelle 5.3 sind die vom Sequenz-zu-Sequenz-Modell ermittelten Trainingslaufzeiten sowie BLEU- und Perplexity-Bestwerte bezüglich der Testdaten aufgeführt (vgl. Abschnitt 2.2.3 zur Erläuterung der Bewertungsmaße). Die Vokabulargröße der dritten und vierten Trainingsmenge bewegt sich überhalb der empfohlenen Werte von 30.000 bis 80.000, was sich deutlich in den Trainingslaufzeiten bemerkbar macht. Diese ist in Stunden und Minuten angegeben. Der BLEU-Wert nimmt vom kleinsten zum größten Datensatz nur geringfügig ab und die Perplexity steigt ebenfalls nur geringfügig an, was zeigt, dass die Übersetzungsqualität nicht stark unter den größeren Trainingsdatenmengen bzw. Vokabularen leidet. Die BLEU-Werte sind um 0,12 bis 0,15 schlechter im Vergleich zu dem von Soru et al. [2017] erzielten Bestwert von 0,753. Da diese Werte nur einen sehr groben Eindruck bezüglich der Güte des Modells bieten, werden im Folgenden die speziellen Eigenschaften betrachtet:

- Erkennen der korrekten Prädikate/Subjekte/Objekte
- Erzeugung von validem SPARQL
- Erkennen des korrekten Abfragetyps (SELECT, ASK, DESCRIBE).

Hierfür kommt ein selbstgeschriebenes Skript zum Einsatz. Die Auswertungen erklären, warum der F1-Wert derzeit nicht als Güteangabe verwendet wird. Die Tabellen A.3 - A.8 offenbaren die eigentlichen Schwächen und Stärken des Modells. Im Testdatensatz (Tabelle A.4) sind Fragen enthalten, die aus dem generierten Trainingsdatensatz der Modelle stammen, aber nicht für das Training verwendet

wurden. Da jedes Template im besten Fall 300 mal instanziiert wurde, kam das Modell im Training aber mit anderen Fragen des gleichen Templates in Kontakt. Damit lässt sich auch die Ähnlichkeit zu den Ergebnissen in Tabelle A.3 erklären, die unter Verwendung von einer Teilmenge des Trainingsdatensatzes entstanden sind. Die Testdatensätze zum QALD-7- und QALD-8-Wettbewerb enthalten Fragen, die den Modellen im Training nie begegnet sind. Sie sind disjunkt zum Trainingsdatensatz, beinhalten aber Fragen aus den gleichen Themengebieten. Die darauf beruhenden Ergebnisse (Tabelle A.7/A.8) sollten mit Vorsicht genossen werden, da sie mit 41 bzw. 43 Fragen auf nur sehr wenigen Übersetzungen basieren. Die Anzahl der analysierten Fragen steht entweder als Zahl hinter dem Rautezeichen in der zweiten Zeile der Tabelle oder im Tabellentitel. Sie entspricht bei den Analysen mit Trainings- bzw. Testdatensatz (Tabelle A.3/A.4) jeweils ein Prozent des Modells zugrundeliegenden Gesamtdatensatzes. Jede Zahl, die in der Spalte unter dem Raute-Zeichen steht, ist die absolute Anzahl an Übersetzungen, für die die jeweilige Eigenschaft zutrifft. Die relativen Prozentangaben stehen rechts daneben.

Abfragetyperkennung Das Erkennen des richtigen Abfragetyps funktioniert in den meisten Szenarios sehr gut. Von den Trainings-/Testfragen werden in fünf von acht Fällen alle richtig erkannt und in den übrigen 99,9 %. Auch bei den QALD-7- und QALD-8-Testfragen wird zu 95-100 % der richtige Typ erkannt. Dies ist auch keine sonderlich komplexe Aufgabe, da nur drei Möglichkeiten zur Auswahl stehen, wenn man die Vorkommenshäufigkeit betrachtet, sogar nur wirklich zwei. Von den QALD-Templates sind 174 vom Typ SELECT, 25 vom Typ ASK und nur eine einzige vom Typ DESCRIBE. Bei den LC-QuAD-Templates sind 4602 SELECT und 365 ASK. Die QALD-7- und QALD-8-Testfragen sind alles SELECT-Abfragen. Bei der Analyse wurde auch geprüft, ob der Aggregator COUNT erkannt wird, der bei den LC-QuAD-Templates 638 mal auftritt.

SPARQL-Syntax Ebenfalls positiv zu bewerten ist, dass die Übersetzungen zu ca. 70-90 % valides SPARQL sind. Die Syntaxüberprüfung wird mithilfe der Python-Bibliothek `rdflib` vorgenommen. Fehlende Punkte zwischen Tripeln führen in eini-

gen Fällen zu Syntaxfehlern. Die Zeichenkette `<unk>`, die die NSpM erzeugt, wenn sie ein Wort nicht übersetzen kann, wird vom Parser als gültige IRI behandelt. Da in den meisten Fällen die Subjekte oder Objekte nicht übersetzt werden können, ist die Wahl dieser Zeichenkette passend. Teilweise führt sie aber auch zu falscher Syntax, z.B.

```
SELECT DISTINCT ?uri where{<unk>(album)dbo:recordedIn ?uri }
```

Sehr häufig wird bemängelt, dass nach `DISTINCT` eine Variable zu erwarten ist, wie hier:

```
SELECT DISTINCT COUNT(?uri)WHERE{?x dbp:lyrics <unk> . ?x dbo:basedOn ?uri }
```

Leider sind diese Fehler bereits im original LC-QuAD-Datensatz enthalten. Dort steht jedem `COUNT` (638 Vorkommen) ein `DISTINCT` voran. Der Fehler wurde somit erlernt. Damit stellen die Werte eine Untergrenze dar, für die Übersetzung in valides SPARQL.

Prädikat- und Entitätserkennung Bei der Identifikation der richtigen Subjekte, Prädikate und Objekte sind die Unterschiede zwischen den Szenarien deutlich. Die Modelle 1, 2 und 3 identifizieren in ca. 97 % der Übersetzungen von Training und Test alle Prädikate korrekt. Das Modell 4 schafft dies noch zu 83 %. Wenig verwundert, dass Modell 4 am besten bei der Übersetzung der original LC-QuAD-Fragen abschneidet, denn es wurde als einziges mit Fragen aller LC-QuAD-Templates trainiert. Berücksichtigt man, dass der Generator 826 von den 5000 LC-QuAD-Templates nicht instanziieren konnte, so verbessern sich der Wert für die Erkennung aller Prädikate von 31 % auf 37 % und der Wert für die Erkennung einzelner Prädikate von 26 % auf 32 %. Die Modelle mit den größeren Trainingsfundus können sich bei den QALD-8-Testfragen profilieren. Bei den QALD-7-Testfragen ist allerdings das Gegenteil der Fall. Am Szenario mit den original QALD-Fragen ist ersichtlich, dass das erste Modell, welches auch nur mit den QALD-Templates trainiert wurde, die besten Resultate bei der Erkennung der Prädikate und Enitäten liefert. Das erweiterte Training mit Fragen anderer Templates bringt keinen Gewinn.

Die Erklärung für die wesentliche Verschlechterung bei den original LC-QuAD-Fragen im Vergleich zu den Trainingsfragen kann ein Blick auf die Entitätserkennung geben. Das Erkennen aller in einer Frage enthaltenen Entitäten funktioniert bereits mit den Trainingsfragen kaum. Lediglich bei einem Fünftel bis einem Drittel der Übersetzungen können die Entitäten teilweise übersetzt werden. Bei den original LC-QuAD-Fragen übersetzt das Modell 3 ein Achtel der Entitäten teilweise richtig und Modell 4 etwa ein Sechstel. Dabei wurde festgestellt, dass die meisten der erkannten Entitäten solche sind, die entweder bereits im Template verankert sind oder in einer zwei- oder dreistelligen Größenordnung instanziiert wurden. Eine genaue Untergrenze für das Mindestvorkommen der Entitäten im Trainingsdatensatz und der Garantie, dass diese anschließend dem Modell bekannt sind, kann nicht angeben werden. Dafür sind die Werte zu gestreut und es existieren sehr viele Ausnahmen von Entitäten, die häufig im Trainingsdatensatz Verwendung finden, aber dennoch nicht zuverlässig übersetzt werden.

Bereits Sutskever et al. [2014] beobachteten, dass Sätze mit vielen seltenen Wörtern schlechtere Übersetzungsergebnisse erzielen als Sätze mit gebräuchlichen Wörtern. Angenommen man hat das Template

Trinkt <A> Gin?

und instanziiert *<A>* mit Namen, die auch für die Instanziierung anderer Templates wie den Biographie-Templates benutzt werden. Dann wird diesen Namen durch das Word Embedding womöglich ein ähnlicher Wortvektor zugewiesen. Das Modell lernt im Training, diese Wortvektoren mit der korrekten Übersetzung zu assoziieren. Wenn dieser Wortvektor nun aber nicht identifiziert werden kann und damit fehlt, fehlt dem Modell auch ein wesentliches Indiz für die korrekte Übersetzung. Das Erkennen der richtigen Prädikate und Entitäten übt eine große Wechselwirkung aufeinander aus.

Englisch-Template	SPARQL-Template
Does <A> play <B>?	`ASK WHERE <A> dbo:programmeFormat <B>`
Does <A> play <B>?	`ASK WHERE <A> dbp:instrument <B>`
Does <A> play the <B>?	`ASK WHERE <A> dbp:instrument <B>`
Does <A> play as <B>?	`ASK WHERE <A> dbp:position <B>`
Does <A> play as a <B>?	`ASK WHERE <A> dbo:position <B>`
Does <A> play for <B>?	`ASK WHERE <A> dbp:currentTeam <B>`
Does <A> play for the <B>?	`ASK WHERE <A> dbp:currentclub <B>`
Does <A> play for the <B>'s national team?	`ASK WHERE <A> dbp:nationalteam <B>`

Tabelle 5.4: Ähnliche Templates

5.3 Einzelfallbetrachtung

Anhand einiger ausgewählter Beispiele sollen weitere Spezifika der NSpM vorgestellt werden. Alle Tests wurden mit Modell 3 und wenn nicht anders erwähnt mit Trainingsfragen vorgenommen.

Sensitivität und Robustheit Einige Templates ähneln sich sehr, wie z.B. die in Tabelle 5.4 aufgeführten. Von den insgesamt 2919 der aus diesen Templates generierten Trainingsfragen übersetzt Modell 3 in 2853 Fällen (97 %) die Prädikate korrekt. Achtet man nur auf die Frage

F_1: *Does <A> play <B>?*

so unterscheidet das Modell in 616 von 640 Fällen korrekt zwischen `dbp:instrument` und `dbo:programmeFormat`. Tabelle A.9 zeigt die Übersetzung von Modell 3 zu einigen selbst konstruierten Fragen zum Template

F_1': *Does <A> play [the] <B>?*

gibson les paul und *fender stratocaster* zählen mit 93- bzw. 89-maliger Verwendung zu den am stärksten trainierten Entitäten und werden gut erkannt. Interessant ist, dass *fender telecaster*, 11-malige Verwendung, nicht erkannt wird, aber immerhin zu *fender stratocaster* übersetzt wird. Die erfundenen Instrumente *fender les paul* und *gibson stratocaster* zeigen, dass das Modell fehlertolerant (robust) ist und zu

Übersetzungen zu semantisch/syntaktisch Ähnlichem in der Lage ist. Die ebenfalls erfundenen Instrumente *qpeqewtqe les paul* und *qpeqewtqe stratocaster* können vom Modell dagegen nicht übersetzt werden und führen zum Auftreten von `<unk>`. Daran zeigt sich der negative Einfluss von OOV-Wörtern.
Weiterhin fällt an der Neigung zur Übersetzung in `dbr:Xian_Lim`, `dbr:Guitar` und `dbr:Country_music` auf, dass das Modell unter Unteranpassung leidet, da es zu stark zu diesen Entitäten tendiert.
Das Beispiel beweist, dass es möglich ist, ähnliche oder gar gleiche Fragen-Templates in unterschiedliche Abfragen zu übersetzen und zeigt so die Sensitivität des Modells. Auch wenn die Musikgenre *soft rock*, *chill-out music*, *deep house* und *gospel music* nicht erkannt wurden, tragen sie dazu bei, das richtige Prädikat zu finden. Hier macht sich ein Unterschied zwischen mehr und weniger trainierten Entitäten bemerkbar. Die zuvor genannten Musikgenre kommen 10, 11, 15 bzw. 20 mal im Trainingsdatensatz vor. *blues* und *reggea* treten nur zwei- bzw. einmal auf und sind dem Modell somit nicht genug Indiz für das richtige Prädikat.

Ambiguität Im Folgenden werden zwei Beispiel herangezogen, bei denen das Modell an der Ambiguität der Fragen scheitert. Es liegt die Frage F_2 vor, zu der zwei sehr ähnliche Übersetzungen $\ddot{U}_{2,1}$ und $\ddot{U}_{2,2}$ trainiert werden.

F_2: *Who created <A>?*
$\ddot{U}_{2,1}$: `select distinct ?uri where { <A> dbo:author ?uri . }`
$\ddot{U}_{2,2}$: `select distinct ?uri where { <A> dbo:creator ?uri . }`

Bei den 494 zugehörigen Trainingsfragen übersetzt die NSpM nur in 289 Fällen (59 %) zum richtigen Prädikat. Ein ähnlich schlechtes Ergebnis liefern die Übersetzungen zu der Frage F_3. In der ersten Übersetzungsvariante $\ddot{U}_{3,1}$ wird eine geografischen Höhenlage erfragt und in der zweiten $\ddot{U}_{3,2}$ die Größe eines Gebäudes.

F_3: *How high is the <A>?*
$\ddot{U}_{3,1}$: `select distinct ?num where { <A> dbo:elevation ?num. }`
$\ddot{U}_{3,2}$: `select distinct ?num where { <A> dbo:height ?num . }`

Nur 312 Übersetzungen der 477 Trainingsfragen (65 %) enthalten das richtige Prädikat. Als Ursache für die schlechte Übersetzungsleistung kommt das seltene Auftreten der Entitäten im Trainingskorpus und die damit wenig repräsentativen Word Embeddings in Frage. 424 der 494 bzw. 476 der 477 Entitäten sind unter zehnmal im Korpus vorhanden. Für das erste auf der Frage F_2 basierende Beispiel kann noch eine weitere Ursache hinzukommen. Dort wird nach dem Schöpfer eines (künstlerischen) Werkes gefragt. Das kann ein Buch, ein Comic, ein Gemälde oder eine Fernsehserie sein. Möglich ist, dass deren Word Embeddings im Vektorraum nah beieinander liegen und der NSpM somit zu wenig Differenz für die Unterscheidung der Prädikate liefern. Zur Überprüfung, ob die NSpM tatsächlich nicht in der Lage ist, mit dieser Art der Ambiguität umzugehen, müssten Word Embeddings benutzt werden, die mit einem repräsentativerem Trainingskorpus trainiert wurden.

Lange und komplexe Abfragen Die Frage F_4 hat eine lange und komplexe zugehörige SPARQL-Abfrage.

F_4	*give me the new <A> series*	
$\ddot{U}_4$	Soll-Template	Ist
	`describe ?a where {`	`describe ?a where {`
	`<A> rdfs:label ?l .`	`<unk> rdfs:label ?l .`
	`?a a dbo:TelevisionShow .`	`?a a dbo:TelevisionShow .`
	`?a foaf:name ?l .`	`?a foaf:name ?l .`
	`?a dbo:completionDate ?ad`	`?a dbo:completionDate ?ad`
	`} order by desc(?ad)limit 1`	`} order by desc(?ad)limit 1`

Setzt man für *<A> star trek: deep space nine*, *game of thrones*, *grey's anatomy* und *how i met your mother* ein, was bis auf die erste Instanziierung nicht trainiert wurde, so erzeugt Modell 3 in jedem Fall die Übersetzung $\ddot{U}_4$-Ist. Abgesehen vom ersten Subjekt ist diese Übersetzung identisch mit dem Soll-Template. Mit den gleichen Entitäten aber dem Template-Paar (F_5, $\ddot{U}_5$)

F_5 *does the new <A> series have more episodes than the old one?*

$\ddot{U}_5$	Soll-Template	Ist

	`ask where {`
	`  <A> rdfs:label ?l .`
	`  ?a a dbo:TelevisionShow .`
	`  ?a foaf:name ?l .`
	`  ?a dbo:completionDate ?ad .`
	`  ?a dbo:numberOfEpisodes ?an .`
	`  ?b rdf:type dbo:TelevisionShow .`
	`  ?b foaf:name ?l .`
	`  ?b dbo:completionDate ?bd .`
	`  ?b dbo:numberOfEpisodes ?bn .`
	`  filter(?ad > ?bd && ?an > ?bn)`
	`}`
		`ask where{`
		`  <unk> rdfs:label ?l .`
		`  ?a a dbo:TelevisionShow .`
		`  ?a foaf:name ?l .`
		`  ?a dbo:completionDate ?ad`
		`} order by desc(?ad)limit 1`

werden Übersetzungen der Art $\ddot{U}_5$-Ist erzeugt. Dies ist ein interessante Übersetzung, da sie zwar nicht komplett das Soll-Graph-Pattern erzeugt, aber eine Komposition aus zwei Templates ist. So entstammt der erste Teil der Abfrage, die Ergebnisbeschreibung, von $\ddot{U}_5$-Soll und der Rest, die WHERE-Klausel und die Modifikatoren vom obigen Beispiel $\ddot{U}_4$, wobei die ersten vier Tripel der beiden Graph-Pattern identisch sind. Beispiele für weitere gute Übersetzungen von komplexen bzw. langen Abfragen sind:

F_6: *was the nine years' war earlier than the 1905 russian revolution?*
Ist: `ask where{<unk> dbo:date ?x <unk> dbo:date ?y . filter(?x < ?y)}`
Soll: `ask where { <A> dbo:date ?x .<B> dbo:date ?y . filter (?x < ?y)}`

F_7 *which architectural structure in cologne has the most visitors?*

$\ddot{U}_7$	Soll-Template	Ist

	`select distinct ?uri where{`
	`  ?uri rdf:type <A>.`
	`  ?uri dbo:location <B> .`
	`  ?uri dbo:numberOfVisitors ?num .`
	`}`
	`order by desc(?num)`
	`offset 0 limit 1`
		`select distinct ?uri where{`
		`  ?uri rdf:type <unk>`
		`  ?uri dbo:location <unk> .`
		`  ?uri dbo:numberOfVisitors ?num .`
		`}`
		`order by desc (?num)`
		`offset 0 limit 1`

F_8 *which of mel gibson films had the highest budget?*

$\ddot{U}_8$	Soll-Template	Ist

	`select distinct ?uri where{`
	`  ?uri dbo:director <A> .`
	`  ?uri dbo:budget ?b .`
	`}`
	`order by ?b`
	`offset 0 limit 1`
		`select distinct ?uri where{`
		`  ?uri dbo:director dbr:David_Bowie .`
		`  ?uri dbo:budget ?b .`
		`}`
		`order by asc (?b)`
		`offset 0 limit 1`

6 Zusammenfassung

Das erste Ziel dieser Arbeit war die Generierung eines Datensatzes, der in seiner Mächtigkeit die Voraussetzung schafft, den von Soru et al. [2017] vorgeschlagenen und innerhalb einer geschlossenen Domäne getesteten ML-basierten QA-Ansatz auf eine offene Domäne zu übertragen. Dieses Ziel wurde erreicht. Der Datensatz kann auch anderen ML-basierten QA-Verfahren zu Trainings- und Testzwecken dienen. Das manuelle Kreieren von Templates wurde durch einen automatischen Prozess ersetzt, der den Aufwand dieses System zu verwenden auch für andere Nutzer senkt. Die Generatorkomponente wurde so erweitert, dass durch die Typisierung von Platzhaltern eine nach eigenen Wünschen steuerbare Instanziierung möglich ist. Wichtigste Kriteria für den Datensatz sind Größe und die Natürlichkeit der Fragen, die darin besteht, dass die Formulierung jeder Frage von einem Menschen stammen könnte. Diese sind teils unvereinbar. So zeigte sich, dass der Versuch, mehr Fragen aus einem Template zu instanziieren, zur Verletzung der Natürlichkeit führt. Derzeit ist nicht ausgeschlossen, dass der Prozess unpassende Instanziierungen erzeugt. Die Erweiterung der Menge von Klassen, die als gültige Platzhaltertypen anzunehmen sind, könnte das Problem vermindern. Für die Vielfältigkeit der Templates wäre es außerdem förderlich, wenn auch Literale durch Platzhalter in den Templates ersetzt werden könnten, was aktuell noch nicht umgesetzt ist. Die größte Version des so erzeugten Datensatzes umfasst 894.499 englische Fragen und ihre äquivalenten SPARQL-Abfragen, wobei die Version die zuvor erwähnten Schwächen besitzt (vgl. Abschnitt 4.3).

Das zweite Ziel, was im Rahmen dieser Arbeit nicht erreicht wurde, war der Beweis, dass mithilfe des Korpus die NSpM in der Lage ist, Fragen verschiedenster Art zu unterschiedlichen Domänen zu beantworten. Dieser Beweis konnte nicht

erbracht werden. Das Hauptproblem der mit dem generierten Korpus trainierten NSpM ist, dass sie die in der Frage enthaltenen Entitäten nicht zu übersetzen vermag, was der Übersetzung der Frage im Ganzen nicht zuträglich ist (vgl. Abschnitt 5.2). Dennoch zeigt sich das Potenzial des Ansatzes. Die mit Testdaten erreichte Übersetzungsgenauigkeit bezüglich des BLEU-Maßes liegt bei durchschnittlich 0,62. Ebenfalls mit Testdaten (vgl. Tabelle A.4) konnte gezeigt werden, dass bei im Durchschnitt 100 % der erzeugten Abfragen der Abfragetyp und bei 93 % alle Prädikate richtig erkannt wurden. 86 % dieser Übersetzungen sind valides SPARQL. Durch die Wahl geeigneter Trainingsfragen, lässt sich sowohl Robustheit als auch Sensitivität bezüglich Abwandlungen in den Fragen erreichen. Annahme war, dass durch häufige Verwendung einer Entität im Trainingsdatensatz die NSpM die Chance hat, die Zuordnung zwischen Label und IRI zu erlernen. Auch wenn das nicht eingetreten ist, ist das Streben nach umfangreicher Instanziierung hilfreich, wenn man dem Problem der Ambiguität begegnen will, was besonders im Abschnitt 5.3 belegt wurde.

Weiterführende Arbeiten könnten analysieren, ob die in Abschnitt 2.2.4 vorgestellten Verfahren wie der Ersetzungsmechanismus, die von Lukovnikov et al. [2017] propagierte wort- und zeichenbasierte Kodierung von Entitäten (vgl. Abschnitt 3.2) oder die Verbesserung von Qualität und Quantität des Englisch-SPARQL-Korpus (vgl. Abschnitt 4.3) zur Lösung des Übersetzungsproblems der Entitäten beiträgt. Eine andere Forschungsrichtung ist die Erprobung des NSpM-Ansatzes an einem konkreten, domänenspezifischen Einsatzszenario wie z.B. dem Projekt „Professorale Karrieremuster der Frühen Neuzeit“[1].

[1] PCP-On-Web: `https://pcp-on-web.htwk-leipzig.de/project/`

Listings

Tabellenverzeichnis

Abbildungsverzeichnis

A Anhang

Sport	Kunst
dbo:SportsClub	dbo:Artwork
dbo:SportsTeam	yago:Painting103876519
dbo:SportsEvent	dbo:Cartoon
dbo:SportsLeague	dbo:LineOfFashion
dbo:SportsSeason	dbo:ArchitecturalStructure
dbo:Sport	dbo:ComicsCreator
dbo:SportFacility	dbo:FashionDesigner
dbo:HorseTrainer	dbo:Painter
dbo:RaceHorse	dbo:Photographer
dbo:Coach	dbo:Sculptor
yago:Trainer110722575	dbo:Architect
dbo:Athlete	dbo:Award
yago:WikicatIceHockeyPositions	Film
yago:WikicatAmericanFootballPositions	dbo:Film
yago:WikicatBaseballPositions	dbo:TelevisionEpisode
yago:Position108621598	dbo:TelevisionSeason
Musik	dbo:TelevisionShow
dbo:MusicalArtist	dbo:Actor
dbo:Instrumentalist	dbo:ScreenWriter
dbo:MusicalWork	dbo:Broadcaster
dbo:Instrument	yago:Actor109765278
dbo:MusicGenre	yago:Entertainer109616922
dbo:RecordLabel	yago:FilmDirector110088200
dbo:Band	yago:FilmMaker110088390
yago:WikicatGuitars	yago:Cameraman109889539
yago:WikicatGuitarAmplifierManufacturers	yago:PrizeWinner109627807
	Literatur
	dbo:Writer
	dbo:Philosopher
	dbo:WrittenWork
	dbo:Publisher
	yago:Editor110044879

Tabelle A.1: Klassentabelle sortiert nach Themengebieten

Geographie	
dbo:Archipelago	dbo:HistoricPlace
dbo:Beach	dbo:Mine
dbo:Cape	dbo:Monument
dbo:Cave	dbo:Park
dbo:Crater	dbo:ProtectedArea
dbo:Desert	dbo:SiteOfSpecialScientificInterest
dbo:Forest	dbo:WineRegion
dbo:Glacier	dbo:WorldHeritageSite
dbo:HotSpring	dbo:Agglomeration
dbo:Mountain	dbo:Community
dbo:MountainPass	dbo:Continent
dbo:MountainRange	dbo:Country
dbo:Valley	dbo:GatedCommunity
dbo:Volcano	dbo:Intercommunality
dbo:Bay	dbo:Island
dbo:Lake	dbo:Locality
dbo:Ocean	dbo:Region
dbo:Sea	dbo:State
dbo:Stream	dbo:Street
dbo:ArchitecturalStructure	dbo:Territory
dbo:CelestialBody	dbo:City
dbo:Cemetery	dbo:CityDistrict
dbo:ConcentrationCamp	dbo:HistoricalSettlement
dbo:CountrySeat	dbo:Town
dbo:Garden	dbo:Village

Tabelle A.2: Fortsetzung: Klassentabelle sortiert nach Themengebieten

Modell / Eigenschaft	1		2		3		4	
	# 512	%	# 992	%	# 3904	%	# 8945	%
alle Prädikate	496	96,9 %	971	97,8 %	3806	97,5 %	7473	83,5 %
einzelne Prädikate	2	0,4 %	4	0,4 %	32	0,8 %	625	7,0 %
alle Entitäten	13	2,5 %	14	1,4 %	10	0,3 %	10	0,1 %
einzelne Entitäten	153	29,9 %	203	20,4 %	1267	32,5 %	2319	25,9 %
Abfragetyp	512	100,0 %	992	99,9 %	3904	100,0 %	8936	99,9 %
SPARQL valide	461	90,0 %	907	91,3 %	3274	83,9 %	7564	84,6 %

Tabelle A.3: Analyse unter Verwendung von Fragen aus dem Trainingsdatensatz

Modell / Eigenschaft	1		2		3		4	
	# 513	%	# 993	%	# 3904	%	# 8945	%
alle Prädikate	495	96,5 %	962	96,9 %	3793	97,2 %	7421	83,0 %
einzelne Prädikate	6	1,2 %	11	1,1 %	34	0,9 %	602	6,7 %
alle Entitäten	9	1,8 %	3	0,3 %	9	0,2 %	12	0,1 %
einzelne Entitäten	151	29,4 %	214	21,6 %	1261	32,3 %	2248	25,1 %
Abfragetyp	513	100,0 %	993	100,0 %	3904	100,0 %	8936	99,9 %
SPARQL valide	439	85,6 %	902	90,8 %	3291	84,3 %	7514	84,0 %

Tabelle A.4: Analyse unter Verwendung von Fragen aus dem Testdatensatz

Modell / Eigenschaft	1		2		3		4	
	#	%	#	%	#	%	#	%
alle Prädikate	9	0,2 %	193	3,9 %	861	17,2 %	1549	31,0 %
einzelne Prädikate	511	10,2 %	197	3,9 %	759	15,2 %	1320	26,4 %
alle Entitäten	0	0,0 %	0	0,0 %	1	0,0 %	1	0,0 %
einzelne Entitäten	12	0,2 %	14	0,3 %	623	12,5 %	881	17,6 %
Abfragetyp	2722	54,4 %	4833	96,7 %	4986	99,7 %	4991	99,8 %
SPARQL valide	3339	66,8 %	3846	76,9 %	3568	71,4 %	3543	70,9 %

Tabelle A.5: Analyse unter Verwendung der 5000 LC-QuAD-Originalfragen

Modell / Eigenschaft	1		2		3		4	
	#	%	#	%	#	%	#	%
alle Prädikate	34	15,8 %	9	4,2 %	14	6,5 %	26	12,1 %
einzelne Prädikate	31	14,4 %	14	6,5 %	16	7,4 %	22	10,2 %
alle Entitäten	0	0,0 %	0	0,0 %	0	0,0 %	0	0,0 %
einzelne Entitäten	5	2,3 %	1	0,5 %	4	1,9 %	4	1,9 %
Abfragetyp	157	73,0 %	202	94,0 %	214	99,5 %	214	99,5 %
SPARQL valide	156	72,6 %	194	90,2 %	150	69,8 %	143	66,5 %

Tabelle A.6: Analyse unter Verwendung der 215 QALD-Originalfragen

Modell / Eigenschaft	1		2		3		4	
	#	%	#	%	#	%	#	%
alle Prädikate	6	14,0 %	8	18,6 %	7	16,3 %	8	18,6 %
einzelne Prädikate	7	16,3 %	5	11,6 %	3	7,0 %	1	2,3 %
alle Entitäten	0	0,0 %	0	0,0 %	0	0,0 %	1	2,3 %
einzelne Entitäten	0	0,0 %	0	0,0 %	0	0,0 %	0	0,0 %
Abfragetyp	41	95,3 %	41	95,3 %	42	97,7 %	41	95,3 %
SPARQL valide	34	79,1 %	35	81,4 %	34	79,1 %	32	74,4 %

Tabelle A.7: Analyse unter Verwendung der 43 QALD-7-Testfragen

Modell / Eigenschaft	1		2		3		4	
	#	%	#	%	#	%	#	%
alle Prädikate	0	0,0 %	2	4,9 %	3	7,3 %	7	17,1 %
einzelne Prädikate	0	0,0 %	0	0,0 %	1	2,4 %	0	0,0 %
alle Entitäten	0	0,0 %	0	0,0 %	0	0,0 %	0	0,0 %
einzelne Entitäten	0	0,0 %	0	0,0 %	0	0,0 %	0	0,0 %
Abfragetyp	20	48,8 %	40	97,6 %	41	100,0 %	41	100,0 %
SPARQL valide	29	70,7 %	32	78,0 %	31	75,6 %	24	58,5 %

Tabelle A.8: Analyse unter Verwendung der 41 QALD-8-Testfragen

Englisch	Übersetzung von Modell 3
Does jimmy page play the gibson les paul?	ASK WHERE dbo:instrument(musician)dbo:instrument dbr:Gibson_Les_Paul
Does jimmy page play the fender stratocaster?	ASK WHERE dbr:Xian_Lim dbo:instrument dbr:Fender_Stratocaster
Does jimmy page play the fender telecaster?	ASK WHERE dbr:Xian_Lim dbo:instrument dbr:Fender_Stratocaster
Does jimmy page play the gibson stratocaster?	ASK WHERE dbr:Xian_Lim dbo:instrument dbr:Fender_Stratocaster
Does jimmy page play the fender les paul?	ASK WHERE dbo:instrument(musician)dbo:instrument dbr:Gibson_Les_Paul
Does jimmy page play the qpeqewtqe sadjfasdjf?	ASK WHERE <unk> dbp:instrument <unk>
Does jimmy page play the qpeqewtqe stratocaster?	ASK WHERE <unk> dbp:instrument <unk>
Does jimmy page play the qpeqewtqe les paul?	ASK WHERE <unk> dbo:instrument <unk>
Does jimmy page play guitar?	ASK WHERE dbr:Xian_Lim dbp:instrument dbr:Guitar
Does jimmy page play ukulele?	ASK WHERE dbr:Xian_Lim dbp:instrument dbr:Guitar
Does jimmy page play saxophone?	ASK WHERE dbr:Xian_Lim dbp:instrument dbr:Guitar
Does jimmy page play piano?	ASK WHERE dbr:Xian_Lim dbp:instrument dbr:Guitar
Does jimmy page play table tennis?	ASK WHERE dbr:Xian_Lim dbp:instrument dbr:Guitar
Does rsa play country music?	ASK WHERE dbr:Xian_Lim dbo:programmeFormat dbr:Country_music
Does rsa play contemporary christian music?	ASK WHERE dbr:Xian_Lim dbo:programmeFormat dbr:Contemporary_Christian_music
Does rsa play soft rock?	ASK WHERE dbr:Xian_Lim dbo:programmeFormat dbr:Country_music
Does rsa play chill-out music?	ASK WHERE dbr:Xian_Lim dbo:programmeFormat dbr:Country_music
Does rsa play deep house?	ASK WHERE dbr:Xian_Lim dbo:programmeFormat dbr:Country_music
Does rsa play gospel music?	ASK WHERE dbr:Xian_Lim dbo:programmeFormat dbr:Country_music
Does rsa play latin pop?	ASK WHERE dbr:Xian_Lim dbo:programmeFormat dbr:Country_music
Does rsa play reggae?	ASK WHERE dbr:Xian_Lim dbp:instrument dbr:Guitar
Does rsa play blues?	ASK WHERE dbr:Xian_Lim dbp:instrument dbr:Guitar
Does rsa play rhythm and blues?	ASK WHERE <unk> dbp:instrument dbr:Guitar

Tabelle A.9: Testübersetzungen zum Template `Does <A> play [the] <B>?`

Literaturverzeichnis

Alpaydın, E. und Linke, S. (2008a). 1.1 Was ist maschinelles Lernen. In *Maschinelles Lernen*. Oldenbourg, München.

Alpaydın, E. und Linke, S. (2008b). 11.12 Lernen mit Zeitreihen. In *Maschinelles Lernen*. Oldenbourg, München.

Alpaydın, E. und Linke, S. (2008c). 2.7 Modellauswahl und Generalisierung. In *Maschinelles Lernen*. Oldenbourg, München.

Berant, J., Chou, A., Frostig, R., und Liang, P. (2013). Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*.

Bordes, A., Usunier, N., Chopra, S., und Weston, J. (2015). Large-scale Simple Question Answering with Memory Networks. *CoRR*, abs/1506.02075.

Cai, Q. und Yates, A. (2013). Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, Sofia, Bulgaria. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., und Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078.

Dai, Z., Li, L., und Xu, W. (2016). CFO: Conditional Focused Neural Question Answering with Large-scale Knowledge Bases. *CoRR*, abs/1606.01994.

Dubey, M., Dasgupta, S., Sharma, A., Höffner, K., und Lehmann, J. (2016). AskNow: A Framework for Natural Language Query Formalization in SPARQL. In *Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains - Volume 9678*, pages 300–316, New York, NY, USA. Springer-Verlag New York, Inc.

Golub, D. und He, X. (2016). Character-Level Question Answering with Attention. *CoRR*, abs/1604.00727.

Lukovnikov, D., Fischer, A., Lehmann, J., und Auer, S. (2017). Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1211–1220, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Luong, M.-T. (2016). *Neural Machine Translation*. PhD thesis, Stanford University.

Luong, M.-T., Brevdo, E., und Zhao, R. (2017). Neural Machine Translation (seq2seq) Tutorial. *https://github.com/tensorflow/nmt*. Zuletzt abgerufen am 29.01.2018.

Makhoul, J., Kubala, F., Schwartz, R., und Weischedel, R. (1999). Performance Measures For Information Extraction. In *In Proceedings of DARPA Broadcast News Workshop*, pages 249–252.

Marx, E., Soru, T., Shekarpour, S., Auer, S., Ngomo, A.-c. N., und Breitman, K. (2013). Towards an efficient RDF Dataset Slicing. *International Journal of Semantic Computing*, 07(04):455–477.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., und Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., und Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Neubig, G. (2017). Neural Machine Translation and Sequence-to-sequence Models: A Tutorial. *CoRR*, abs/1703.01619.

Papineni, K., Roukos, S., Ward, T., und Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Raschka, S. (2017a). 1.2 Die drei Arten des Machine Learnings. In *Machine Learning mit Python: das Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning*. mitp, Frechen, 1. Auflage edition.

Raschka, S. (2017b). 12.7 Weitere neuronale Netzarchitekturen. In *Machine Learning mit Python: das Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning*. mitp, Frechen, 1. Auflage edition.

Raschka, S. (2017c). 1.4 Entwicklung eines Systems für das Machine Learning. In *Machine Learning mit Python: das Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning*. mitp, Frechen, 1. Auflage edition.

Raschka, S. (2017d). 2.3 Adaptive lineare Neuronen und die Konvergenz des Lernens. In *Machine Learning mit Python: das Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning*. mitp, Frechen, 1. Auflage edition.

Raschka, S. (2017e). 3.3.4 Überanpassung durch Regularisierung verhindern. In *Machine Learning mit Python: das Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning*. mitp, Frechen, 1. Auflage edition.

Rashid, T. (2017a). 1 Wie neuronale Netze arbeiten. In *Neuronale Netze selbst programmieren: ein verständlicher Einstieg mit Python*, pages 30–43. O'Reilly, Heidelberg, 1. Auflage edition.

Rashid, T. (2017b). 1 Wie neuronale Netze arbeiten. In *Neuronale Netze selbst programmieren: ein verständlicher Einstieg mit Python*, pages 59–94. O'Reilly, Heidelberg, 1. Auflage edition.

Rashid, T. (2017c). 2 Do it yourself mit Python. In *Neuronale Netze selbst programmieren: ein verständlicher Einstieg mit Python*, pages 153–159. O'Reilly, Heidelberg, 1. Auflage edition.

Rey, G. D. und Wender, K. F. (2011). 3.4 Rekurrente Netze. In *Neuronale Netze: eine Einführung in die Grundlagen, Anwendungen und Datenauswertung*, Aus dem Programm Verlag Hans Huber Psychologie Lehrbuch. Huber, Bern, 2., vollst. überarb. und erw. Auflage edition.

Shekarpour, S., Marx, E., Ngonga Ngomo, A.-C., und Auer, S. (2015). SINA: Semantic interpretation of user queries for question answering on interlinked data. *Semantic Search*, 30:39–51.

Soru, T., Marx, E., Moussallem, D., Publio, G., Valdestilhas, A., Esteves, D., und Neto, C. B. (2017). SPARQL as a Foreign Language. *CoRR*, abs/1708.07624.

Stock, W. G. (2007). *Information retrieval: Informationen suchen und finden ; [Lehrbuch]*. Number 1 in Einführung in die Informationswissenschaft. Oldenbourg, München.

Sutskever, I., Vinyals, O., und Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *CoRR*, abs/1409.3215.

Trivedi, P., Maheshwari, G., Dubey, M., und Lehmann, J. (2017). LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. *The Semantic Web - ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, pages 210–218.

Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., und Cimiano, P. (2012). Template-based Question Answering over RDF Data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 639–648, New York, NY, USA. ACM.

Usbeck, R., Ngomo, A.-C. N., Haarmann, B., Krithara, A., Röder, M., und Napolitano, G. (2017). 7th Open Challenge on Question Answering over Linked Data (QALD-7). In Dragoni, M., Solanki, M., und Blomqvist, E., editors, *Semantic Web Challenges*, pages 59–69, Cham. Springer International Publishing.

Yih, S. W.-t., Chang, M.-W., He, X., und Gao, J. (2015). Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. ACL - Association for Computational Linguistics.

Yin, W., Yu, M., Xiang, B., Zhou, B., und Schütze, H. (2016). Simple Question Answering by Attentive Convolutional Neural Network. *CoRR*, abs/1606.03391.

Yu, L. (2011). 2.2 The Abstract Model of RDF. In *A developer's guide to the semantic web*. Springer, Berlin.

Zhang, Y., Liu, K., He, S., Ji, G., Liu, Z.-y., Wu, H., und Zhao, J. (2016). Question Answering over Knowledge Base with Neural Attention Combining Global Knowledge Information. *CoRR*, abs/1606.00979.

Eidesstattliche Erklärung

Ich versichere, dass die Masterarbeit mit dem Titel nicht anderweitig als Prüfungsleistung verwendet wurde und diese Masterarbeit noch nicht veröffentlicht worden ist. Die hier vorgelegte Masterarbeit habe ich selbstständig und ohne fremde Hilfe abgefasst. Ich habe keine anderen Quellen und Hilfsmittel als die angegebenen benutzt. Diesen Werken wörtlich oder sinngemäß entnommene Stellen habe ich als solche gekennzeichnet.

Leipzig, den Unterschrift