Automatische Transformierung multilingualer Spracheingaben in Datenbankabfragen

Marcel Franzen[1]

Abstract: Um die Nutzung von Datenbanken zu vereinfachen, werden im Rahmen dieser Arbeit SQL-Abfragen auf Basis einer Fragestellung und dem Datenbankschema in Form der Spaltennamen erzeugt. Hierzu werden mehrere Neuronale Netze eingesetzt, die einzelne Teile der SQL-Abfrage vorhersagen. Darüber hinaus werden multilingualen Worteinbettungen zur Repräsentation der Frage und Tabellenspalten verwendet. Durch die Nutzung der Worteinbettungen können auch Synonyme auf die Spaltennamen abgebildet werden. Dadurch ist die im Trainingsprozess verwendete Sprache irrelevant. Die Ergebnisse zeigen, dass das entstandene Modell sowohl tabellenunabhängig als auch sprachunabhängig funktioniert. Demnach erfordert die Nutzung einer Datenbank nur noch wenig Wissen über das Datenbankschema und insbesondere über die Sprache der Spaltennamen.

Keywords: Maschinelles Lernen; Neuronale Netze; Natural Language Processing; SQL-Abfragen; Multilingualität

1 Einleitung

In den vergangenen Jahrzehnten haben Menschen eine große Menge von Daten erzeugt, die gespeichert werden müssen, zumal die Menge der Daten in den kommenden Jahren weiter steigen wird [ID21]. Im Zuge dessen wird die Entwicklung moderner Datenbanksysteme als eine der wichtigsten Aspekte der Digitalisierung angesehen. In allen Bereichen von Kommunikation über Banken, Transportwesen, Versicherungen bis zu Universitäten, sind Datenbanken unverzichtbar [Ga17]. Auch wenn es den Nutzern häufig nicht bewusst ist, sind Datenbanken in beinahe jeder alltäglichen Aktivität involviert. Dementsprechend speichern Datenbanken eine große Menge heutiger Informationen und bilden die Basis für moderne Technologien.

Gleichzeitig führt die steigende Menge von Daten zu immer komplexer werdenden Datenbanken, deren Abfrage von einem Nutzer Kenntnisse über Datenbanksprachen wie SQL erfordert [ZXS17]. Dieser Umstand führt zu einer Barriere zwischen Nutzern ohne technische Kenntnisse und den Vorteilen, die eine Datenbank aufweist [PSC21]. Aus diesem Grund beschäftigt sich diese Arbeit mit der Erzeugung von SQL-Abfragen anhand von Fragen in natürlicher Sprache, die ein Nutzer zu einem unbekannten Datenbankschema stellen kann. Hierfür kommen Techniken des Maschinellen Lernens und der natürlichen Sprachverarbeitung zum Einsatz. Somit kann jeder Informationen von einer Datenbank abfragen, ohne die SQL-Abfrage selber formulieren zu müssen. Diese Arbeit greift dazu die

[1] Universität Bremen, Bibliotheksstraße 1, 28359 Bremen, Germany mfranzen@uni-bremen.de

bereits in Vielzahl vorhandenen Ansätze auf und entwickelt sie in den folgenden Aspekten weiter:

1. Bei der Speicherung von Nutzerdaten sind aufgrund der DSGVO Aspekte der Privatsphäre und Sicherheit zu berücksichtigen. Daher verfolgt diese Arbeit das Ziel der vollständigen Aufrechterhaltung der Privatsphäre. Folglich basiert der in dieser Arbeit vorgestellte Ansatz ausschließlich auf der Frage eines Nutzers in natürlicher Sprache und dem Datenbankschema in Form der Spaltennamen. Der eigentliche Inhalt einer Datenbank wird dabei nicht verwendet, um die SQL-Abfrage zu erzeugen.
2. Da die Datensätze zur Lösung dieser Aufgabe in Englisch vorhanden sind [ZXS17, Yu18], präsentiert diese Arbeit einen auf multilingualen Sprachmodellen basierenden Ansatz, um sowohl Englisch als auch Deutsch verwenden zu können. Im Zuge dessen wird untersucht, inwiefern eine sprachunabhängige Repräsentation der Eingabedaten zu sprachunabhängigen Ergebnissen führt.

Folglich können die Ergebnisse dieser Arbeit dazu verwendet werden, um jedem den Zugang zu Datenbanken und deren Nutzung zu ermöglichen.

2 Verwandte Arbeiten

In [ZXS17] wird der WikiSQL-Datensatz vorgestellt, welcher die Grundlage zur Lösung des Problems mit lediglich einer Tabelle darstellt. Dieser besteht aus insgesamt 80654 englischen Beispielen, welche eine hand-annotierte Sammlung von Fragen über den Inhalt von Datenbanken mit entsprechenden SQL-Abfragen bilden. Dabei sind verschiedenste Fragearten enthalten und die Länge der Fragen sowie der finalen SQL-Abfragen und die Anzahl der Spalten pro Tabelle variieren. Die SQL-Abfragen im Datensatz folgen stets dem selben Schema.

Die vorhandenen Ansätze zur Erzeugung von SQL-Abfragen anhand natürlicher Sprache können grundsätzlich in zwei verschiedene Kategorien unterteilt werden [Hu21]:

Erstens gibt es die erzeugungs-basierte Methode, die sich durch einen Sequenz-zu-Sequenz-Prozess auszeichnen. Bei diesem Ansatz wird die Fragestellung als Sequenz von Wörtern auf eine Sequenz abgebildet, welche die fertige SQL-Abfrage darstellt [SVL14]. Beispiele für die Nutzung der erzeugungs-basierten Methode sind [ZXS17, Su18, KDG17].

Zweitens gibt es eine skizzenbasierte Methode, welche die Erzeugung der SQL-Abfrage in Submodule unterteilt, die einzelne Aspekte der SQL-Abfragen wie zum Beispiel die zu selektierende Spalte erzeugen. Diese erreicht grundsätzlich bessere Ergebnisse auf dem WikiSQL-Datensatz [Hu21]. Beispiele hierfür sind [GG20, XLS17, He19]. Im Vergleich zu [ZXS17] nutzen die Modelle in [GG20] nicht ausschließlich nur die Frage und das

Tabellenschema in Form der Spaltennamen. Um das Modell zur Erzeugung der SQL-Abfragen mit weiteren Informationen bezüglich der Beziehungen zwischen Frage und Tabellenspalten anzureichern, stellt [GG20] zwei neue Algorithmen vor. Zum einen wird der Tabelleninhalt mit der Frage verglichen. So entsteht ein Vektor mit der Länge der Frage, in dem alle Wörter markiert sind, die als Inhalt einer Zelle der Tabelle gefunden werden konnten. Zum anderen wird ein weiterer Vektor mit der Länge des Tabellenkopfes erzeugt, in dem alle Spalten markiert sind, die in der Frage erwähnt werden. Weiter werden in diesem Vektor noch die Spalten markiert, deren Zellen im vorherigen Algorithmus in der Frage wiedergefunden werden konnten. Durch diese zusätzlichen Informationen werden bessere Ergebnisse als in bisherigen Arbeiten erzielt. Als Repräsentation der Frage und der Spaltennamen wird ein BERT-Modell [De19] verwendet.

Der Ansatz aus [PSC21] präsentiert ebenfalls einen skizzenbasierten Ansatz, welcher zusätzliche Informationen wie [GG20] einbindet, dabei jedoch den Inhalt der Tabelle vernachlässigt, um die Privatsphäre der Daten zu wahren. Anstatt die Repräsentation der Eingabewerte durch ein BERT-Modell umzusetzen, nutzt dieses Paper die Weiterentwicklung RoBERTa [Li19].

In dieser Arbeit wird ein skizzenbasierter Ansatz vorgestellt, bei dem ebenfalls zusätzliche Informationen wie in [GG20] verwendet werden, wobei die Algorithmen dazu auf [PSC21] basieren, da die Privatsphäre der Daten auch in dieser Arbeit berücksichtigt wird. Diese Algorithmen sind jedoch nicht in der Lage, Synonyme aus der Frage mit der Tabelle zu verknüpfen und schlagen darüber hinaus fehl, wenn die Frage in einer anderen Sprache (z.B. Deutsch) als die Spaltennamen der Tabelle (z.B. Englisch) vorliegt. Dementsprechend werden die bereits existierenden Ansätze in dieser Arbeit aufgegriffen, aber hinsichtlich einer multi-lingualen Anwendung ergänzt.

3 Methodik

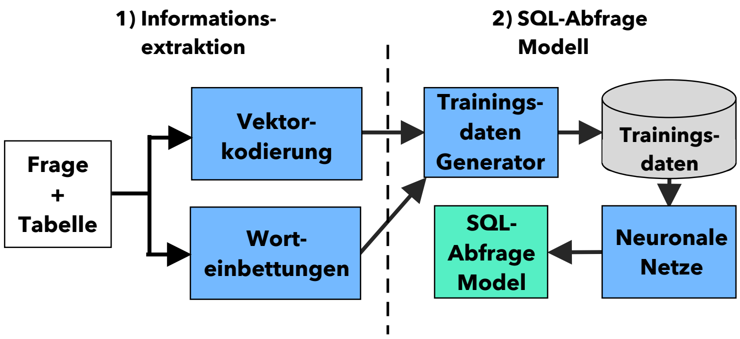

Abb. 1: Eine Übersicht über die Herangehensweise zur Erzeugung der SQL-Abfragen (Grafik inspiriert durch [MGD21]).

Die Herangehensweise in dieser Arbeit unterteilt sich in zwei Schritte: 1) Informationsextraktion und 2) ML-Model Generierung, welche in der Abbildung 1 dargestellt sind.

1. Informationsextraktion: Zunächst müssen die Einträge aus dem Datensatz zur weiteren Verarbeitung aufbereitet werden. Hierzu werden die Frage und das Tabellenschema in Form der Spaltennamen verwendet, um nützliche Informationen zu extrahieren.
2. SQL-Abfrage Modell: Anhand der Daten aus der Informationsextraktion wird ein finaler Datensatz gebildet, welcher von mehreren Neuronalen Netzen im Training genutzt wird. Aus diesen einzelnen Neuronalen Netzen wird dann das Gesamtmodell gebildet, welches SQL-Abfragen erzeugen kann.

3.1 Informationsextraktion

Zur Verwendung des Datensatzes müssen informationsreiche Merkmale extrahiert werden, in denen die Frage und die Spaltennamen enthalten sind, aber auch die Beziehungen untereinander erkenntlich sind. Hierzu wird der WikiSQL-Datensatz zunächst in eine numerische Repräsentation durch Worteinbettungen transformiert, welche als Eingabe für die Neuronalen Netze verwendet wird. Darüber hinaus werden weitere Vektoren aus dem Datensatz erzeugt, in denen zusätzliche Merkmale hinsichtlich der Beziehung zwischen Frage und Spalten enthalten sind. Nachfolgend werden die genannten Schritte näher erklärt:

3.1.1 Kodierung zusätzlicher Featurevektoren

Da der Fokus in dieser Arbeit unter anderem auf die Privatsphäre der Daten gerichtet ist, steht der Inhalt der Tabellen zur zusätzlichen Kodierung nicht zur Verfügung. Dadurch werden die Modelle generalsierter und unabhängig von Tabelleninhalten nutzbar. Im Folgenden werden zwei neue Algorithmen vorgestellt, die auf den Ideen aus [GG20] und [PSC21] aufbauen, aber hinsichtlich der Art des Vergleiches weiterentwickelt wurden.

Mit dem ersten Algorithmus wird ein Vektor erzeugt, in dem die Wörter der Frage mit dem Tabellenkopf verglichen werden. Hierzu wird die Frage vorher in einzelne Tokens unterteilt, wodurch sich jeder Index im Vektor auf einen Token der Frage bezieht. Wenn sich ein Token der Frage auf den Namen einer Spalte bezieht, werden die enstprechenden Indexe im Vektor markiert, die das gefundene Wort in der Frage repräsentieren. Die Unterteilung der Frage in Tokens wird durchgeführt, da die numerische Repräsentation in Kapitel 3.1.2 ebenfalls auf den einzelnen Tokens basiert. Für den Vektor wird eine One-Hot-Kodierung angewendet, bei der ein Index lediglich dann markiert wird, wenn der entsprechende Token in dem Namen einer Spalte des Tabellenkopfes wiedergefunden wird. Wie in [GG20] wird auch hier der erzeugte Vektor als QV (Question mark vector) bezeichnet.

Der oben angesprochene Unterschied bezüglich der Vergleichsmethode liegt darin, dass im ursprünglichen Paper [GG20] lediglich Wort für Wort verglichen wird. In diesem Algorithmus werden hingegen Worteinbettungen verwendet, die einerseits einen 1:1 Vergleich ermöglichen. Andererseits können so aber durch Synonyme auch ähnliche Wörter markiert werden. Hierzu wird die Konsinusähnlichkeit zwischen den Worteinbettungen ermittelt. Sollte diese bei über 50% Ähnlichkeit liegen, wird der entsprechende Index im Vektor markiert. Dadurch kann sich die Frage von dem gegebenen Tabellenschema stärker unterscheiden, als es in bisherigen Arbeiten [GG20, PSC21] möglich ist. Da der Nutzer oftmals kein Wissen über das zugrunde liegende Tabellenschema und den Namen der Spalten hat, wird diese Hürde durch die Worteinbettungen gelöst.
Des Weiteren kann die Nutzung muti-lingualer Worteinbettungen dazu genutzt werden, dass sich die Sprache der Frage von der, der Spaltennamen unterscheiden kann und die Markierung trotzdem möglich ist. Eine Vorraussetzung dafür ist das Vorhandensein angegelichener Worteinbettungen, in denen gleiche Wörter in unterschiedlichen Sprachen nah beieinander liegen, wie sie durch [fa17] bereitgestellt werden. Der zweite Algorithmus erzeugt einen Vektor, in dem alle Spalten markiert sind, die sich auf die Frage beziehen. Dieser wird fortan *HV* genannt (Table header mark vector).

In der Tabelle 1 werden exemplarisch zwei Beispiele gezeigt, anhand derer die Funktionsweise der oben ausgeführten Algorithmen verdeutlicht wird. Im ersten Beispiel bezieht sich das Wort „notes“ in der Frage auf die gleichnamige Spalte der Tabelle. Aus diesem Grund sind sowohl der entsprechende Index im QV als auch HV-Vektor mit einer *1* markiert.

Frage:	Tell me what the notes are for south australia?
Tabellenkopf:	[„territory“, „background colour“, „slogan“, „series“, „notes“]
QV:	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
HV:	[0, 0, 0, 0, 1]
Frage:	In welchem Büro sitzt Christopher Metz?
Tabellenkopf:	[„office“, „employee“, „mail“, „telephone“]
QV:	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
HV:	[1, 0, 0, 0, 0]

Tab. 1: Zwei Beispiele die zeigen, wie die QV -und HV Vektoren anhand einer Frage und einem Tabellenkopf erzeugt werden.

3.1.2 Transformierung in eine latente Repräsentation durch Worteinbettungen

Zur Verarbeitung von natürlicher Sprache ist eine latente Repräsentation als Vektoren notwendig, die durch Worteinbettungen erreicht werden kann. Wie bereits in 3.1.1 erwähnt, werden in dieser Arbeit diesbezüglich die multilingualen Worteinbettungen des Facebook MUSE Projektes verwendet, die auf fastText [Bo16] basieren und für ein Wort, einen Vektor mit 300 Werten zur Verfügung stellen.

Zunächst wird der Text durch das Sprachverarbeitungsframework spaCy [o.oJ] in die einzelnen Tokens zerlegt, wobei diese in Kleinbuchstaben umgewandelt werden und neben dem originalen Token noch die Grundform (Lemma) betrachtet wird. Für jeden dieser Tokens wird dann geprüft, ob es in dem Worteinbettungsmodell den passenden Vektor gibt. Falls nicht, wird geprüft, ob es einen Vektor für die Grundform des Tokens gibt. Jeder passende Vektor wird anschließend in eine Liste von Vektoren eingefügt, woraus die latente Repräsentation des gesamten Textes mit den Dimensionen $(n, 300)$ gebildet wird, wobei n der Anzahl der Token entspricht. Sollte ein Token nicht in den Worteinbettungen enthalten sein, wird der Token gegen einen speziellen Token *UNK* ersetzt, welcher aus 300 Nullen besteht.
Durch diese Methode werden die Frage und die Spaltennamen des Tabellenkopfes für jeden Eintrag im Datensatz transformiert. Die Spaltennamen werden zu diesem Zweck zu einem Text konkateniert und jeweils mit dem Token *</s>* getrennt. Da diese multilingualen Worteinbettungen nur für eine endliche Menge von Wörtern existieren, sollten die Spaltennamen in der Regel aus Wörtern bestehen, für die es Worteinbettungen gibt. Andernfalls würde die Repräsentation der Spaltennamen überwiegend aus Nullen bestehen und somit den Informationsgehalt stark reduzieren. Dies gilt ebenfalls für die Fragen, wobei diese oftmals länger sind als die Spaltennamen und somit mehr Wörter enthalten, wodurch die unbekannten Wörter einen geringeren Einfluss auf die gesamte Repräsentation haben.

3.2 SQL-Abfrage Modell

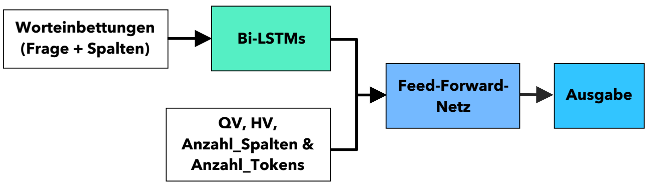

Abb. 2: Die Referenzarchitektur zur Vorhersage einzelner Aspekte einer SQL-Abfrage.

Die Erzeugung der SQL-Abfragen in der zugrundeliegenden Bachelorarbeit, auf der dieses Paper basiert, wird nur für einen konkreten Anwendungsfall benötigt. So wird die Komplexität des Problems stark reduziert und die Modelle müssen nur für kleine Tabellen funktionieren.

Als Grundlage zur Modellierung des Problems in dieser Arbeit dient [GG20], worin die Beschaffenheit des WikiSQL-Datensatzes genutzt wird, um das Problem der Abfrage-Erzeugung in kleinere Teil-Probleme zu zerlegen, da die SQL-Abfragen stets dem selben Schema folgen. Dadurch wird die Komplexität des Problems verringert, weshalb der Entwurf der Neuronalen Netze leichter und effizienter ist. Mehrere Modelle werden so trainiert, dass sie die einzelnen Attribute der Trainingsdaten vorhersagen können, um letztlich die einzelnen Lücken in der Abfrage zu füllen. Die Abbildung 2 zeigt die Referenzarchitektur dieser Arbeit, auf der die einzelnen Modelle mit geringfügigen Änderungen aufbauen.

Die bereits erwähnten Worteinbettungen werden genutzt, um für jeden Token der Frage sowie der Spalten einen latenten Vektor zu erzeugen. Darüber hinaus werden die zusätzlichen QV -und HV Vektoren aus Kapitel 3.1.1 als weitere Eingabewerte verwendet sowie die Anzahl der Fragetokens und die Anzahl der Spalten im Tabellenkopf.

Die Worteinbettungen werden zu einem Vektor konkateniert und durch zwei bidirektionale LSTMs *(Bi-LSTMs)* verarbeitet, damit der gesamte Kontext der Sequenz erfasst werden kann. Durch die gemeinsame Verarbeitung von Frage und Tabelle soll das Modell ein tieferes Verständnis für den Zusammenhang zwischen beiden Komponenten erlangen. Eine alternative Architektur, in der die Frage und die Tabellenspalten zunächst durch getrennte Bi-LSTMs verarbeitet werden, wurde ebenfalls in Betracht gezogen, aber aufgrund nicht signifikant besserer Ergebnisse wieder verworfen.

Zusammen mit den anderen Eingabedaten wird die Ausgabe der Bi-LSTMs anschließend von einem einfachen Feed-Forward-Netz verarbeitet. Die gewünschte Ausgabe unterscheidet sich hierbei jeweils hinsichtlich des jeweiligen Aspektes der SQL-Abfragen, den das Modell erzeugen soll:

SELECT-Spalte (sel): In dieser Arbeit werden nur Beispiele aus dem Datensatz verwendet, die maximal fünf Spalten in der Tabelle enthalten Aus diesem Grund werden zur Vorhersage der zu selektierende Spalte fünf Ausgabeneuronen benötigt. Folglich wird das Problem der zu selektierende Spalte als einfache Klassifikation betrachtet.

Aggregatfunktion (agg): Analog zur Vorhersage der SELECT-Spalte, wird zur Vorhersage der Aggregatfunktion ebenfalls eine Klassifikation verwendet, die sechs mögliche Klassen enthält (MAX, MIN, AVG, SUM, COUNT & NONE).

Anzahl WHERE-Klauseln: Die SQL-Abfragen in dieser Arbeit enthalten stets entweder eine oder maximal zwei WHERE-Klauseln, weshalb das Problem binär betrachtet werden kann und lediglich ein Ausgabeneuron notwendig ist.

WHERE-Spalten: Wenn zwei WHERE-Klauseln benötigt werden, ist die Reihenfolge dieser irrelevant. Aus diesem Grund wird für jede WHERE-Klausel ein eigenes Trainingsbeispiel erzeugt, welches im One-Hot-Vektor der gewünschten Ausgabe die entsprechende Spalte markiert. Die Ausgabe des Modells erzeugt folglich keine Wahrscheinlichkeitsverteilung per Softmax-Funktion, sondern wendet die Sigmoid-Funktion für jedes Neuron einzeln an. Falls nur eine WHERE-Klausel gesucht wird, kann der größte Wert in der Ausgabe betrachtet werden, deren Index die Spalte der WHERE-Klausel angibt. Wenn zwei WHERE-Klauseln gesucht werden, werden die zwei größten Werte mit deren Indexen betrachtet. Im Gegensatz zur oben vorgestellten Architektur ist die Anzahl der WHERE-Klauseln ein zusätzlicher Eingabewert des Modells.

WHERE-Werte: Die Vergleichswerte in den WHERE-Klauseln sind Ausschnitte aus den ursprünglichen Fragen. Aus diesem Grund werden zur Vorhersage der Vergleichswerte in den WHERE-Klauseln zwei getrennte Modelle verwendet. Ersteres bestimmt den Anfangstoken aus der Frage und das zweite Modell den Endtoken aus der Frage. Die Vorhersage der Vergleichswerte ist neben den üblichen Eingabewerten noch zusätzlich von der Spalte in der WHERE-Klausel abhängig, wodurch die Modelle sowohl für ein als auch für zwei WHERE-Klauseln genutzt werden können. Der Operator in der WHERE-Klausel wird in dieser Arbeit nicht betrachtet, da nur Beispiele aus dem Datensatz entnommen wurden, die den Gleichheits-Operator enthalten.

Jedes dieser Modelle wird verwendet, um einen bestimmten Aspekt der SQL-Abfrage zu erzeugen. Folglich wird die finale SQL-Abfrage anhand der Ausgaben aller einzelnen Modelle zusammengesetzt, wie es in der Abbildung 3 dargestellt ist.

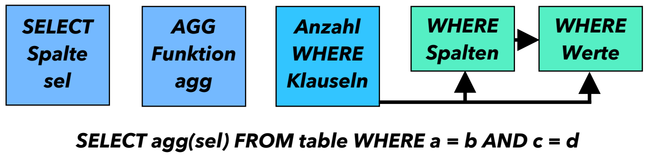

Abb. 3: Die einzelnen Modelle zur Vorhersage verschiedener Aspekte einer SQL-Abfrage und deren Zusammensetzung.

4 Ergebnisse

Die Modelle sind aufgrund der Datenprivatsphäre generalisiert und können grundsätzlich für jede Tabelle und jede Frage genutzt werden, solange sie nicht größer als die im Training verwendeten Datenbanken sind. Nichtsdestotortz führt die Privatsphäre der Daten und die daraus folgenden Nichtberücksichtigung der Tabelleninhalte auch dazu, dass wichtige Informationen in den HV -und QV Vektoren nicht enthalten sind, von denen die Modelle profitieren könnten. Dennoch wurden, wie in der Tabelle 2 zu sehen, vielversprechende Ergebnisse erreicht:

In der Tabelle 2 sind die Ergebnisse der einzelnen Modelle auf einem Testdatensatz von WikiSQL zusammengefasst. Mit einer 70.48% Genauigkeit für die Vorhersage der zu selektierenden Spalte sowie 84.37% für die Aggregatfunktion funktionieren diese Aspekte der SQL-Abfrage besonders gut. Auch die Ergebnisse für die Anzahl der WHERE-Klauseln sind mit 88.86% gut, obwohl insgesamt mehr Beispiele für nur eine WHERE-Klausel im Testdatensatz enthalten sind und die Genauigkeit durch diese Unausgewogenheit im Datensatz beeinflusst wird. Mit einer Genauigkeit von 70% funktioniert die Vorhersage der Vergleichswerte ebenfalls gut. Bei der Verwendung realer Daten werden die Ergebnisse für die Vergleichswerte der WHERE-Klauseln schlechter ausfallen, da hierfür die Spalte der

WHERE-Klausel als Eingabe verwendet wird. Da das Modell hierfür jedoch lediglich eine Genauigkeit von 46.83% aufweist, liefert es überwiegend falsche Ergebnisse, welche sich negativ auf die darauffolgenden Modelle auswirken.

Aufgabe	**Genauigkeit auf dem Testdatensatz**
SELECT-Spalte	70.48%
Aggregatfunktion	84.37%
Anzahl der WHERE-Klauseln	88.86%
SELECT-Spalte der WHERE-Klauseln	46.83%
Anfangstoken des Vergleichswerts in den WHERE-Klauseln	70.76%
Endtoken des Vergleichswerts in den WHERE-Klauseln	70.01%

Tab. 2: Die Ergebnisse der Modelle auf dem Testdatensatz.

Die Experimente haben gezeigt, dass die zusätzlichen Vektoren bezüglich der Verknüpfung zwischen der Frage und den Tabellenspalten eine starke Gewichtung in der Vorhersage der Spalten haben und folglich eine sinnvolle Erweiterung der Eingabedaten sind. Darüber hinaus wurde gezeigt, dass die Nutzung von Worteinbettungen für die zusätzlichen Vektoren die Fähigkeit, die Frage mit dem Tabellenschema zu verknüpfen, verstärken, da so auch Synonyme erkannt werden können und der Nutzer nicht zwingend Kenntnisse über die Tabelle benötigt. Außerdem ermöglicht dies, dass der Nutzer die Frage in mehr Variationen stellen kann und dennoch gleiche Ergebnisse erhält.

1:

Frage:	What is Terrence Ross nationality?				
Tabelle:	player	nationality	position	years in toronto	team
Zielausgabe:	SELECT nationality FROM table WHERE player='terrence ross'				
Ausgabe:	SELECT nationality FROM table WHERE player='terrence ross'				

2:

Frage:	What is the average length of Lucifer?				
Tabelle:	series	episode	title	directed by	written by
Zielausgabe:	SELECT AVG(length) FROM table WHERE series = 'lucifer'				
Ausgabe:	SELECT AVG(length) FROM table WHERE series = 'of lucifer'				

Tab. 3: Ergebnisse anhand deutscher- und englischer Fragen / Spaltennanmen (1).

Bei den Aggregatfunktionen ist aufgefallen, dass manchmal falsche Aggregatfunktionen vorhergesagt werden, wenn die Eingabesprache eine andere ist, als die im Training verwendete. So konnte das Neuronale Netz den Zusammenhang zwischen bestimmten Worteinbettungen und den richtigen Aggregatfunktionen erlernen wie zum Beispiel *how many* und *COUNT*. Da die Worteinbettungen zwar stark angeglichen sind, aber dennoch Abweichungen haben, hatten sie zwischen Deutsch und Englisch teilweise eine zu geringe Ähnlichkeit, damit das Modell auch hier den Zusammenhang zu den Aggregatfunktionen herstellen konnte. Die Tabellen 3 und 4 zeigen einige Ergebnisse anhand verschiedener Kombinationen aus deutschen und englischen Fragen bzw. Spaltennamen. Folglich ist die Herangehensweise mit multilingualen Sprachmodellen einerseits zielführend, andererseits werden noch stärker angeglichene Sprachmodelle die Ergebnisse weiter verbessern.

3:

Frage:	Was ist Terrence Ross Nationalität?				
Tabelle:	spieler	nationalität	position	jahre in toronto	verein
Zielausgabe:	SELECT nationalität FROM table WHERE spieler = ’terrence ross’				
Ausgabe:	SELECT nationalität FROM table WHERE spieler = ’terrence ross’				

4:

Frage:	Was ist die durchschnittliche Länge von Lucifer?				
Tabelle:	series	episode	title	directed by	length
Zielausgabe:	SELECT AVG(length) FROM table WHERE series = ’lucifer’				
Ausgabe:	SELECT AVG(length) FROM table WHERE length = ’länge von lucifer’				

Tab. 4: Ergebnisse anhand deutscher- und englischer Fragen / Spaltennanmen (2).

5 Fazit und Ausblick

Diese Arbeit beschäftigt sich mit der Erzeugung von SQL-Abfragen anhand einer gegebenen Frage und einer Tabelle. Hierzu wurde der WikiSQL Datensatz verwendet und hinsichtlich eines vereinfachten Trainings optimiert, sodass die Komplexität des Problems reduziert werden konnte. Der Anspruch der Datenprivatsphäre konnte erfüllt werden und folglich werden die Ergebnisse unabhängig vom Inhalt der Tabelle erzielt, wodurch das Modell insgesamt deutlich generalisierter ist. Darüber hinaus wurde in dieser Arbeit untersucht, inwiefern multilinguale Sprachmodelle dazu genutzt werden können, um die Ergebnisse Neuronaler Netze bei Problemen der natürlichen Sprachverarbeitung unabhängig von Sprachen zu erzielen. Diesbezüglich haben die Experimente gezeigt, dass dies möglich ist und dabei oftmals identische Ergebnisse erzielt werden können. Insbesondere konnten verschiedene Sprachkombinationen getestet werden, welche sehr ähnliche Ergebnisse lieferten. Da die Modelle für die Spalten einen starken Fokus auf die zusätzlichen Vektoren legen, funktioniert die Vorhersage der Spalten multilingual besonders gut. Dies ist darin begründet, dass mit Hilfe der angeglichenen Worteinbettungen oftmals die selben Eingabevektoren über verschiedene Sprachen hinweg erzeugt werden können. Noch stärker angeglichene Worteinbettungen werden potenziell zu noch besseren Ergebnissen führen. Die selben Modelle können für unterschiedliche Eingabesprachen verwendet werden, obwohl die Trainingsdaten lediglich in Englisch vorlagen.

Zusammengefasst konnten alle Ziele der Arbeit erreicht werden und es ist ein Modell zur Erzeugung von SQL-Abfragen entstanden, welches einerseits unabhängig vom Inhalt einer Tabelle und andererseits unabhängig von der Eingabesprache funktioniert. Die vorgestellen Algorithmen und die Nutzung der angeglichenen Worteinbettungen als Repräsentation der Eingabedaten kann potenziell auf andere Sprachen ausgeweitet werden, ohne einen signifikanten Verlust der Zuverlässigkeit zu haben. Die Stärken und Schwächen des Ansatzes konnten herausgearbeitet werden und insbesondere wurde versucht, die noch vorhandenen Schwierigkeiten zu erklären.

In einer hierauf aufbauenden Arbeit kann versucht werden, die Ergebnisse weiter zu verbessern. Folgende Ansätze könnten dabei verfolgt werden:

- Bessere Sprachmodelle: Die Verwendung besserer Sprachmodelle kann dazu genutzt werden, um eine allgemeinere Repräsentation der Frage und Tabellenspalte zu erreichen. Die Nutzung der Worteinbettungen ist unabhängig vom Kontext der Frage und entsprechend sehr abhängig vom genauen Wortlaut. Andere Arbeiten zu diesem Thema schlagen die Nutzung von BERT vor, welches nicht immer dieselben Vektoren für Wörter zuweist, sondern dies abhängig vom Gesamtkontext macht. Insbesondere könnte dadurch das Problem gelöst werden, dass die Worteinbettungen in verschiedenen Sprachen teilweise nicht ähnlich genug waren, um multilingual dieselben Ergebnisse zu erzielen.
- Mehr Informationen über den Zusammenhang von Frage und Tabelle: Die Vernachlässigung der Tabelleninhalte hatte positive und negative Auswirkungen. Die Algorithmen könnten hier dahingehend verbessert werden, dass die Wortart der Tokens berücksichtigt wird, sodass beispielsweise eine Ortsangabe mit einer dafür vorgesehenen Spalte verknüpft wird.

6 Danksagung

Dieses Paper basiert auf einer Bachelorarbeit, welche an der Universität Bremen in der Arbeitsgruppe Rechnerarchitektur von Prof. Dr. Rolf Drechsler entstand. Diese Arbeit wurde durch das Data Science Center der Universität Bremen (DSC@UB) unterstützt. Ein besonderer Dank gilt Christopher Metz, welcher als Betreuer der Bachelorarbeit fungierte.

Literaturverzeichnis

[Bo16] Bojanowski, Piotr; Grave, Edouard; Joulin, Armand; Mikolov, Tomás: Enriching Word Vectors with Subword Information. CoRR, abs/1607.04606, 2016.

[De19] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina: , BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019.

[fa17] facebookresearch/MUSE, Juli 2017. Abgerufen: 14.07.2021.

[Ga17] Gabriel, CRISTIAN Mihai: The Importance Of Databases In Economy - Some General Coordinates. Revista Economica, 69(4):92–98, November 2017.

[GG20] Guo, Tong; Gao, Huilin: , Content Enhanced BERT-based Text-to-SQL Generation, 2020.

[He19] He, Pengcheng; Mao, Yi; Chakrabarti, Kaushik; Chen, Weizhu: X-SQL: reinforce schema representation with context. CoRR, abs/1908.08113, 2019.

[Hu21] Hui, Binyuan; Shi, Xiang; Geng, Ruiying; Li, Binhua; Li, Yongbin; Sun, Jian; Zhu, Xiaodan: , Improving Text-to-SQL with Schema Dependency Learning, 2021.

[ID21] IDC, Statista: , Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025 (in zettabytes). https://www.statista.com/statistics/871513/worldwide-data-created/, 2021. Abgerufen: 10.07.2022.

[KDG17] Krishnamurthy, Jayant; Dasigi, Pradeep; Gardner, Matt: Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Copenhagen, Denmark, S. 1516–1526, September 2017.

[Li19] Liu, Yinhan; Ott, Myle; Goyal, Naman; Du, Jingfei; Joshi, Mandar; Chen, Danqi; Levy, Omer; Lewis, Mike; Zettlemoyer, Luke; Stoyanov, Veselin: RoBERTa: A Robustly Optimized BERT Pretraining Approach. CoRR, abs/1907.11692, 2019.

[MGD21] Metz, Christopher A.; Goli, Mehran; Drechsler, Rolf: Early Power Estimation of CUDA-Based CNNs on GPGPUs: Work-in-Progress. CODES/ISSS ’21, Association for Computing Machinery, New York, NY, USA, S. 29–30, 2021.

[o.oJ] o.A.: , spaCy: Industrial-Strength Natural Language Processing, o.J. Abgerufen: 07.10.2021.

[PSC21] Pal, Debaditya; Sharma, Harsh; Chaudhari, Kaustubh: , Data Agnostic RoBERTa-based Natural Language to SQL Query Generation, 2021.

[Su18] Sun, Yibo; Tang, Duyu; Duan, Nan; Ji, Jianshu; Cao, Guihong; Feng, Xiaocheng; Qin, Bing; Liu, Ting; Zhou, Ming: Semantic Parsing with Syntax- and Table-Aware SQL Generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Melbourne, Australia, S. 361–372, Juli 2018.

[SVL14] Sutskever, Ilya; Vinyals, Oriol; Le, Quoc V.: , Sequence to Sequence Learning with Neural Networks, 2014.

[XLS17] Xu, Xiaojun; Liu, Chang; Song, Dawn: SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. CoRR, abs/1711.04436, 2017.

[Yu18] Yu, Tao; Zhang, Rui; Yang, Kai; Yasunaga, Michihiro; Wang, Dongxu; Li, Zifan; Ma, James; Li, Irene; Yao, Qingning; Roman, Shanelle; Zhang, Zilin; Radev, Dragomir R.: Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. CoRR, abs/1809.08887, 2018.

[ZXS17] Zhong, Victor; Xiong, Caiming; Socher, Richard: Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. CoRR, abs/1709.00103, 2017.