h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

– Fachbereich Informatik –

Wissensgenerierung für deutschprachige Chatbots

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

vorgelegt von

Tilo Werner Michel

Matrikelnummer: 745788

Referent	:	Prof. Dr. Ute Trapp
Korreferent	:	Prof. Dr. Bettina Harriehausen-Mühlbauer

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 1. Juli 2022

Tilo Werner Michel

ABSTRACT

Modern chatbot systems use methods from the area of artificial intelligence and natural language processing to process user requests. The development of such systems involves a lot of work, as developers have to map use cases into possible conversations with users.

In order to reduce this workload and speed up the development of chatbots, this thesis explored different approaches to ease the workload of chatbot developers. It was also considered to what extent the approaches could be used for the German language.

It was first investigated what possibilities exist to convert websites into question-answer pairs to train chatbots.

Subsequently neural methods of text generation were tested to automatically create training examples for the training of chatbots. Concretely questions were generated from paragraphs and paraphrases from already created training utterances. Since most of the approaches have already been implemented in English, it was also investigated to what extent these methods can be transferred to the German language.

The text generation experiments were compared using automatic metrics. In addition, the approach with the best results from the automatic metrics was evaluated manually with texts from selected websites.

The results of the automatic metrics showed that trained models for text generation in German perform worse compared to models in English. One reason for this is that the available datasets in German for training are smaller than English-language datasets.

In a manual evaluation of the question generation, it was found that the tested approach was mostly able to generate grammatically correct questions that can also be answered by the input text. However, the usability in a chatbot training scenario is rarely given, as the questions are inappropriate for training chatbots.

Two approaches were tested for paraphrasing, with one approach using machine translation models achieving better results than an approach using a model trained with German language data.

The results confirm that there is a lack of training data for training text generation models in German. Therefore, using machine translation models for text generation in German is currently the most sensible approach.

ZUSAMMENFASSUNG

Moderne Chatbotsysteme nutzen Methoden der künstlichen Intelligenz und der natürlichen Sprachverarbeitung, um Nutzeranfragen zu bearbeiten. Die Entwicklung dieser Chatbotsysteme ist mit einem hohen Aufwand an Arbeit verbunden, da Entwickler Anwendungsfälle in mögliche Konversationen mit Nutzern abbilden müssen.

Um diesen Arbeitsaufwand zu reduzieren und die Entwicklung von Chatbots zu beschleunigen, wurden in dieser Arbeit verschiedene Ansätze untersucht, um die Arbeitslast von Chatbotentwicklern zu erleichtern. Außerdem wurde betrachtet, inwiefern die Ansätze für die deutsche Sprache nutzbar sind.

Es wurde zuerst untersucht, welche Möglichkeiten es gibt, um Webseiten in Frage-Antwort-Paare für das Training von Chatbots umzuwandeln.

Anschließend wurden neuronale Methoden der Textgenerierung getestet, um Trainingsbeispiele für das Training von Chatbots automatisiert zu erstellen. Speziell wurden Fragen aus einem Absatz und Paraphrasen aus bereits erstellten Trainingsbeispielen generiert. Da die meisten Ansätze bereits in englischer Sprache umgesetzt wurden, wurde außerdem untersucht, inwiefern diese Methoden auf die deutsche Sprache übertragbar sind.

Die Experimente zur Textgenerierung wurden mithilfe automatischer Metriken verglichen. Zusätzlich wurde der Ansatz mit den besten Ergebnissen der automatischen Metriken manuell evaluiert, anhand von Texten ausgewählter Webseiten.

Die Ergebnisse der automatischen Metriken zeigen, dass trainierte Modelle zur Textgenerierung in deutscher Sprache im Vergleich zu Modellen in englischer Sprache schlechter abschneiden. Ein Grund dafür ist beispielsweise, dass die vorhandenen deutschsprachigen Trainingsdatensätze kleiner sind als englischsprachige Datensätze.

Bei einer manuellen Evaluation der Fragengenerierung wurde festgestellt, dass der beste Ansatz größtenteils grammatikalisch korrekte Fragen generieren konnte, die auch durch den Eingabetext beantwortet werden können. Allerdings ist die Verwendbarkeit in einem Trainingsszenario für Chatbots selten gegeben, da die Fragen unpassend zum Training von Chatbots sind.

Für die Paraphrasierung wurden zwei Ansätze getestet, wobei ein Ansatz mit Modellen zur maschinellen Übersetzung bessere Ergebnisse bei einer manuellen Evaluierung erzielen konnte, als ein Ansatz mit einem Modell, welches mit deutschsprachigen Daten trainiert wurde.

Die Ergebnisse bestätigen, dass zum Training von Modellen für die Textgenerierung in deutscher Sprache ein Mangel an Trainingsdaten herrscht. Deshalb ist das Verwenden von Modellen zur maschinellen Übersetzung für die Verwendung der Textgenerierung in deutscher Sprache aktuell am sinnvollsten.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

API Application Programming Interface

KI Künstliche Intelligenz

ML Machine Learning

DL Deep Learning

NLP Natural Language Processing

NLU Natural Language Understanding

IE Information Extraction

NER Named Entity Recognition

UI User-Interface

FAQ Frequently Asked Questions

BOW Bag-of-words

KNN Künstliche Neuronale Netze

BERT Bidirectional Encoder Representations from Transformers

GPT Generative Pre-Training

HTML Hypertext Markup Language

CSS Cascading Style Sheets

JS JavaScript

HTTP Hypertext Transfer Protocol

CSV Comma Separated Values

JSON JavaScript Object Notation

DOM Document Object Model

XML eXtensible Markup Language

Teil I

THESIS

1 EINLEITUNG

Chatbots sind Softwaresysteme, die über natürliche Sprache mit einem Nutzer interagieren können. Das Verwenden natürlicher Sprache hat den Vorteil, dass ein Nutzer beispielsweise in seiner Muttersprache Fragen an den Chatbot stellen kann, um Informationen zu erhalten.

In verschiedenen Bereichen wie dem Onlinehandel sind Chatbots mittlerweile ein wichtiger Bestandteil, beispielsweise für die Kundenbetreuung. Im Idealfall ist die Kommunikation zwischen Chatbot und Mensch identisch zur Kommunikation zwischen zwei Menschen.

Im Gegensatz zu Chatbots als Mensch-Maschine-Schnittstellen sind vor allem grafische Benutzeroberflächen verbreitet, bei diesen muss ein Nutzer zu gesuchten Informationen navigieren. Chatbots können im Idealfall hingegen eine direkte Antwort auf eine Frage oder Aussage liefern.

Ein Chatbot kann, je nach Implementierung, gesprochene oder geschriebene Sprache verarbeiten.

Viele Informationen über verschiedene Themen wie Krankheiten usw. liegen im World Wide Web (WWW) vor. Diese Daten können zum Erstellen von Chatbots für verschiedene Domänen wertvoll sein, da durch das Verwenden dieser Informationen der Aufbau einer Wissensbasis für einen Chatbot beschleunigt werden kann. In dieser Arbeit wird zum einen betrachtet, wie Informationen aus dem WWW für einen Frage-Antwort-Chatbot strukturiert werden können, sodass diese schließlich in einer Konversation mit einem Chatbot abgefragt werden können. Zum anderen werden zwei verschiedene Vorgehen aufgezeigt, wie Trainingsdaten für Chatbots automatisch erweitert werden können.

1.1 MOTIVATION

Große Technologieunternehmen wie Microsoft, IBM oder Google bieten sogenannte No-Code-Plattformen an, um Chatbots zu entwickeln [Agr+20; Ibm; Gooc]. No-Code bedeutet dabei, dass diese Plattformen Abstraktionen für Nutzer über eine Benutzeroberfläche anbieten, die das Erstellen von Chatbots vereinfachen.

Allerdings benötigt der Prozess der Chatbotentwicklung zusätzlich Expertise von Domänenexperten, die passende Dialoge entwerfen und Informationen validieren. Informationen der verschiedenen Domänen können unter anderem im Internet vorliegen, falls diese dokumentiert wurden. Die Informationen sind dabei Teil einer Webseite (semi-strukturiert), die Text (unstrukturiert) enthält.

Die Masterarbeit ist Teil eines Kooperationsprojektes der Hochschule Darmstadt mit der MakeIT Consulting GmbH & Co. KG mit dem Namen Smart-

Multi-Purpose-Emergency-Bot-Tools (SMEBT), das seit 2020 durchgeführt wird. Im Rahmen des SMEBT-Projekts werden Werkzeuge rund um Chatbot-Plattformen entwickelt, um die Entwicklung von Chatbots zu beschleunigen. Die Möglichkeiten zur Extraktion von Webseiteninformationen (unstrukturierte bzw. semi-strukturierte Daten zu strukturierten Daten umwandeln) und die automatische Generierung von Trainingsdaten (Fragen und Paraphrasierungen) sollen mit dieser Arbeit betrachtet werden. Die extrahierten Informationen von Webseiten und die generierten Trainingsdaten müssen im Anschluss von Domänenexperten verifiziert werden.

1.2 ZIEL DER ARBEIT

Ziel dieser Masterarbeit ist es, Möglichkeiten zur Trainingsdatenerweiterung für Chatbots zu finden und zu bewerten im Hinblick auf die Verwendbarkeit im Kontext der Chatbotentwicklung. Dazu werden zwei Ansätze zur Textgenerierung in deutscher Sprache vorgestellt, die die Erstellung von Chatbots beschleunigen sollen. Zum einen wird die Fragengenerierung aus Webtexten betrachtet. Zum anderen wird das Paraphrasieren aus bereits erstellten oder generierten Trainingsbeispielen getestet. Es wird versucht, beide Ansätze mithilfe von neuronalen Methoden zu lösen und die Qualität mit automatischer sowie manueller Evaluation zu bewerten.

1.3 GLIEDERUNG

Kapitel 2 beinhaltet nötige Grundlagen für die natürliche Sprachverarbeitung (engl. Natural Language Processing (NLP)), Sprachmodelle, Machine Learning (ML) und Deep Learning in Verbindung mit NLP und schließlich die Transformer-Architektur, die bei vortrainierten neuronalen Sprachmodellen genutzt wird.

Außerdem wird auf die Definition von Fragen und Paraphrasierungen eingegangen, um ein besseres Verständnis dafür zu geben, was generiert werden soll. Zusätzlich wird der aktuelle Stand der Chatbotentwicklung beschrieben, um zu verstehen, welche Trainingsdaten notwendig sind. Schließlich wird die Informationsextraktion aus dem Web beschrieben und mit welchen Mitteln diese umsetzbar ist.

Im darauf folgenden Kapitel 3 werden verwandte Arbeiten vorgestellt. Zuerst werden Arbeiten zum Strukturieren von Webseitendaten vorgestellt (Kapitel 3.1). Darauf folgen speziellere Arbeiten, die Webseitendaten in ein Format für Frage-Antwort-Chatbots strukturieren. Schließlich werden Arbeiten zu den Aufgaben Fragengenerierung und Paraphrasierung vorgestellt (Kapitel 3.2 und 3.3).

In Kapitel 4 werden die verwendeten Methoden vorgestellt. Dabei wird zwischen automatisierter Evaluierungen (Kapitel 2.3.3) und der Evaluierung über einen Fragebogen unterschieden (Kapitel 4.4). Kapitel 5 beschreibt die getätigten Experimente. Das 6 Kapitel beschreibt die Ergebnisse der Experimente. Darauf folgt Kapitel 7 mit einer Diskussion der Ergebnisse und ein

Vergleich zu den verwandten Arbeiten. Im letzten Kapitel 8 folgt ein Fazit und Ausblick.

2 GRUNDLAGEN

In diesem Kapitel werden die nötigen Grundlagen beschrieben, um das Vorgehen dieser Arbeit nachvollziehen zu können. In Kapitel 2.1 werden Grundlagen der natürlichen Sprachverarbeitung beschrieben.

Anschließend werden in Kapitel 2.2 Sprachmodelle erklärt, da diese nötig sind, um Texte zu generieren.

Aufbauend auf Sprachmodellen wird Textgenerierung beschrieben, besonders im Hinblick auf die Generierung von Fragen und Paraphrasen für Chatbots, welche im darauf folgenden Kapitel 2.3 beschrieben werden.

Ergänzend zur Textgenerierung werden Metriken in Kapitel 2.3.3 beschrieben, die zur automatischen Evaluation der Textgenerierung genutzt werden können.

Darauf folgend wird die Funktionsweise von Chatbots erläutert, da diese im Zentrum der Forschungfrage stehen, wenn Trainingsdaten für diese generiert werden sollen.

Schließlich wird die Informationsextraktion aus dem Web beschrieben.

2.1 NATÜRLICHE SPRACHVERARBEITUNG

Das Gebiet der natürlichen Sprachverarbeitung (engl. NLP) hat zum Ziel natürliche Sprachen, wie Deutsch oder Englisch, durch Computer verarbeitbar zu machen [RN21, S. 874]. Menschen kommunizieren, um Wissen zu teilen und natürliche Sprachen sind eine Möglichkeit, dieses Wissen zu repräsentieren [RN21, S. 874]. Ein Sender teilt durch eine natürliche Sprache in gesprochener Sprache oder Text sein Wissen mit einem Empfänger [RN21, S. 874]. Der Empfänger nimmt Sprache oder Text wahr und schlussfolgert daraus die beabsichtigte Bedeutung [RN21, S. 874]. Russell und Norvig nennen drei Hauptgründe, um NLP mit Computern durchzuführen [RN21, S. 874]:

KOMMUNIKATION Es ist praktischer für Menschen durch natürliche Sprache mit einem Computer zu interagieren. Formale Sprachen sind komplizierter, da der Mensch diese erst lernen muss, natürliche Sprachen erlernt der Mensch in der Regel in den ersten Lebensjahren.

LERNEN Viel Wissen der Menschheit ist in natürlicher Sprache beschrieben. Wenn ein System viel Wissen haben soll, muss es natürliche Sprache verstehen können.

WISSENSCHAFTLICHES VERSTÄNDNIS Das wissenschaftliche Verständnis von Sprachen und deren Verwendung nutzt die Wissenschaftsdisziplinen Künstliche Intelligenz (KI) in Verbindung mit Linguistik, kognitiver Psychologie und Neurowissenschaft, um das wissenschaftliche Verständnis über Sprachen zu erweitern.

Natürliche Sprachen, wie beispielsweise Deutsch oder Englisch, unterscheiden sich von formalen Sprachen wie Programmiersprachen. Programmiersprachen können nach festgelegten Regeln (Grammatik, Syntax und Semantik) interpretiert werden [RN21, S. 874]. Bei natürlichen Sprachen ist das nicht immer möglich, da diese mehrdeutig sind [RN21, S. 874].

2.1.1 *Vorverarbeitung von Textdaten*

Ein Text kann aus verschiedenen Dateien extrahiert werden. Beispiele für solche Dateiformate sind PDF und HTML. Im Kontext dieser Arbeit wird vor allem das HTML-Format betrachtet, da dieses im Web verwendet wird. Im Folgenden werden Vorverarbeitungsschritte für Text erläutert, die für nachfolgende Aufgaben im NLP notwendig sein können. Die Definitionen entstammen [Vaj+20, Pre-Processing]:

TOKENISIERUNG (engl. Tokenization) beschreibt das Segmentieren eines Textes. Beispiele für diese Segmente sind Buchstaben, Wörter, Sätze und Absätze.

KLEINSCHREIBUNG / GROSSSCHREIBUNG Um Wörter besser vergleichen zu können, werden Wörter in Kleinbuchstaben oder Großbuchstaben konvertiert. Das ermöglicht einen Vergleich von Wörtern unabhängig von Groß- und Kleinschreibung.

STOPPWÖRTER Stoppwörter sind Wörter, die keinen Inhalt eines Textes beinhalten und häufig vorkommen. Beispiele sind „ein, eine, der, die, das“ usw. Stoppwörter sind abhängig vom Kontext, in dem gearbeitet wird. Es gibt vorgefertigte Listen mit Stoppwörtern, die genutzt werden können, um Stoppwörter zu entfernen.

ENTFERNEN SPEZIELLER ZEICHEN Sonderzeichen, Punktierung und Nummern können entfernt werden, wenn diese keinen Mehrwert für eine spätere Analyse beinhalten.

WORTSTAMMBILDUNG (engl. Stemming) Umwandlung eines Wortes in seine Stammform. Das ist nützlich für Vergleiche der Wörter, um z. B. Suchen in Dokumenten mit der Nutzereingabe vergleichbar zu machen. Wenn ein Nutzer das Wort „Autos“ in Dokumenten sucht, kann er Dokumente mit dem Wort „Auto“ nur mithilfe von Wortstammbildung finden.

LEMMATISIERUNG (engl. Lemmatization) Ein Wort wird zurück zu seiner Ursprungsform umgewandelt, dem sogenannten *Lemma*. Das *Lemma* unterscheidet sich insofern, dass hier anstatt von festgelegten Algorithmen, wie beim Stemming, linguistisches Wissen genutzt wird, um die Stammform zu bestimmen. Das Lemma „besser“ ist beispielsweise das Wort „gut“.

WORTART TAGGING (engl. Part-Of-Speech (POS) Tagging) bedeutet, die Wortarten von Wörtern zu bestimmen. Wortarten sind abhängig vom Satz,

in dem das Wort vorkommt. Ein Beispiel dazu ist in Kapitel 2.1.2 zu finden.

RECHTSCHREIBKORREKTUR (engl. spelling correction) bedeutet Rechtschreibfehler, Tippfehler oder ähnliche Fehler zu beheben, da diese Fehler ggf. das korrekte Verarbeiten eines Textes verhindern können.

2.1.2 *Named Entity Recognition*

Bei der Named Entity Recognition (NER) werden aus einem Text Entitäten extrahiert, wie beispielweise eine Person, eine Organisation, ein Datum usw. [Vaj+20, Kapitel 5]. Ein einfacher Ansatz ist, eine Liste von Entitäten und deren jeweilige Kategorie zu nutzen [Vaj+20, Kapitel 5]. Ein erweiterter Ansatz ist das Nutzen von POS-Tags in Verbindung mit NER [Vaj+20, Kapitel 5].

Bei dem Beispiel „Wo wurde Albert Einstein geboren?“ ist Albert Einstein eine Entität, die zu der Kategorie Person zugeordnet werden kann [Vaj+20, Kapitel 5]. Mit einem State-of-the-Art POS-Tagger, wie spaCy [Hon+20], kann der Satz wie folgt annotiert werden, es werden Universal POS-Tags verwendet [Uni; Spa]:

```
Wo/ADV wurde/AUX Albert/PROPN Einstein/PROPN geboren/VERB ?/PUNCT
```

Die Wörter Albert und Einstein wurden als PROPN getaggt d. h., dass es sich um einen Eigennamen (engl. proper noun) handelt. Mit eigens erstellten Regeln könnten Entitäten auch anhand ihrer POS-Tags erkannt werden, ohne dass eine Liste von Wörtern gepflegt werden muss [Vaj+20, Kapitel 5].

Schließlich können auch ML Methoden verwendet werden, um Entitäten zu identifizieren [Vaj+20, Kapitel 5]. Dabei ist die Suche nach Entitäten im Text ein Klassifizierungsproblem, wobei Wörter nicht isoliert betrachtet werden können, sondern als Sequenz [Vaj+20, Kapitel 5]. Vajjala u. a. nennen dieses Problem auch Sequenzklassifizierung, wobei POS-Tagging ebenso unter diese Kategorie fällt [Vaj+20, Kapitel 5].

2.2 SPRACHMODELLE

Um natürliche Sprache zu verarbeiten, können sogenannte Sprachmodelle (engl. language models) verwendet werden [RN21, S. 875]. Ein Sprachmodell ist eine Wahrscheinlichkeitsverteilung, welche die Wahrscheinlichkeit einer beliebigen Wortsequenz beschreibt [RN21, S. 875]. Ein grammatikalisch korrekter Satz hat dabei eine hohe Wahrscheinlichkeit, während die Wahrscheinlichkeit eines grammatikalisch falschen Satzes eine niedrige Wahrscheinlichkeit hat [RN21, S. 875].

Sprachmodelle können verschiedene Aufgaben bewältigen [RN21, S. 875]:

VORHERSAGE EINES NÄCHSTEN WORTES Mit einem Sprachmodell kann das wahrscheinlichste nächste Wort eines Textes bestimmt werden und dadurch ein Text vervollständigt werden.

ÜBERSETZUNG Mit einem Sprachmodell kann eine Übersetzung von einer Sprache zu einer anderen Sprache aufgrund von Wahrscheinlichkeiten des Sprachmodells berechnet werden.

FRAGE BEANTWORTEN Mit Frage-Antwort-Paaren als Trainingsdaten kann das Sprachmodell die wahrscheinlichste Antwort zu einer Frage aus einem Text bestimmen.

Da natürliche Sprachen komplex sind, sind Sprachmodelle im besten Fall eine Annäherung an diese [RN21, S. 875]. Das bedeutet, es gibt kein eindeutiges Sprachmodell für eine natürliche Sprache wie Deutsch, in der Form von einer formalen Sprache wie der Programmiersprache Python [RN21, S. 875].

2.2.0.1 *Bag-of-words und N-Gramm Modelle*

Das Bag-of-words (BoW)-Modell ist eine Möglichkeit, um Sprache zu repräsentieren. Ein **Dokument** bzw. ein Text wird als eine Sammlung von Wörtern dargestellt [Vaj+20, Kapitel 3]. Eine Sammlung von Wörtern steht dabei jeweils für eine Klasse, die durch die Wörter in der Sammlung repräsentiert wird [Vaj+20, Kapitel 3]. Das bedeutet, wenn zwei Dokumente die gleichen Wörter verwenden, gehören sie demnach zum gleichen BoW und damit zu einer gleichen Klasse [Vaj+20, Kapitel 3].

Folgendes Beispiel erläutert das Vorgehen eines BoW. Gegeben sind folgende Dokumente in Tabelle 2.1 (übersetzt aus [Vaj+20, Kapitel 3]):

Id	Text
D_1	Hund beißt Mann
D_2	Mann beißt Hund
D_3	Hund isst Futter
D_4	Mann isst Essen

Tabelle 2.1: BoW Beispieldokumente.

Das Vokabular des BoW besteht aus den einzigartigen Wörtern in allen Dokumenten. Im Beispiel sind sechs Wörter im Vokabular des BoW: Hund, beißt, Mann, isst, Futter, Essen. Im BoW werden die Sätze als Vektoren dargestellt und deren Anzahl im Dokument gezählt.

Id	Hund	beißt	Mann	isst	Futter	Essen
D_1	1	1	1	0	0	0
D_2	1	1	1	0	0	0
D_3	1	0	0	1	1	0
D_4	0	0	1	1	0	1

Tabelle 2.2: BoW Vektoren für Dokumente aus Tabelle 2.1.

Im Beispiel wird ein Nachteil ersichtlich: Die Dokumente D_1 und D_2 haben die gleiche Repräsentation, obwohl sie unterschiedliche Bedeutung haben, da die Reihenfolge der Wörter in der Repräsentation nicht berücksichtigt wird.

Weitere Probleme des BoW Modells sind nach Vajjala u. a. [Vaj+20, Kapitel 3]:

- Die Reihenfolge der Wörter im Dokument und der Kontext ignoriert.
- Die Größe der Vektoren ist abhängig vom Vokabular.
- Wörter, die nicht im Vokabular sind, können nicht berücksichtigt werden.

Ein erweiterter Ansatz des BoW Modells ist das **N-Gramm** Wort Modell [Vaj+20, Kapitel 3]. Dabei werden nicht einzelne Wörter in das Vokabular hinzugefügt, sondern aufeinander folgende Wörter. Das N steht dabei für die Anzahl an Wörtern im Text, die zusammen berücksichtigt werden sollen. Tabelle 2.3 zeigt, wie das Vokabular mit den Beispieldokumenten aus Tabelle 2.1 bei N gleich zwei (Bigramm) aussehen würde.

Id	Hund beißt	beißt Mann	Mann beißt	beißt Hund	Hund isst	isst Futter	Mann isst	isst Essen
D_1	1	1	0	0	0	0	0	0
D_2	0	0	1	1	0	0	0	0
D_3	0	0	0	0	1	1	0	0
D_4	0	0	0	0	0	0	1	1

Tabelle 2.3: Bigramm Vektoren für Dokumente aus Tabelle 2.1.

Durch die Verwendung eines Bigramm Vokabulars sind die Vektoren D_1 und D_2 nicht mehr identisch wie in Tabelle 2.2, das kann spätere Analysen verbessern.

Aufbauend auf den Konzepten dieser einfachen Sprachmodelle nutzen moderne Sprachmodelle vor allem neuronale Ansätze, um Sprache zu erlernen und zu verarbeiten.

2.2.1 *Machine Learning und Deep Learning*

Wie in vorherigen Kapiteln beschrieben wurde, kann natürliche Sprache mithilfe von algorithmischen Ansätzen verarbeitet werden. Viele State-of-the-Art-Ansätze für verschiedene NLP-Aufgaben zur Zeit dieser Arbeit (2022) verwenden Methoden aus dem ML. Beim ML wird mit einem Lernalgorithmus ein Modell erstellt, das aus Beispieldaten lernt [RN21, S. 669]. Ein Grund der Verwendung von ML im NLP ist, dass für einige NLP-Probleme

keine algorithmischen Lösungen bekannt sind oder algorithmische Lösungen schlechtere Performanz für NLP-Aufgaben haben im Vergleich zu ML Lösungen [RN21, S. 928].

Bei ML gibt es verschiedene Formen zu lernen [RN21, S. 671] :

ÜBERWACHTES LERNEN (engl. supervised learning) Ein Modell wird mit Eingabe-Ausgabe-Paaren erstellt. Dabei bestimmt der Lernalgorithmus eine passende Funktion, welche aus einer Eingabe die passende Ausgabe berechnet. Der Ausgabewert wird auch *Label* genannt.

UNÜBERWACHTES LERNEN (engl. unsupervised learning) Der Lernalgorithmus erhält die Beispieldaten ohne Label und versucht Muster (engl. patterns) in den Daten zu finden.

VERSTÄRKENDES LERNEN (engl. reinforcement learning) Ein Modell wird durch eine Serie von Belohnungen und Bestrafungen trainiert. Der Lernalgorithmus zielt darauf ab, mehr Belohnungen zu erhalten und verändert die Problemlösungsstrategie demnach. Ein Beispiel dafür wäre ein Modell, dass zum Ziel hat Schachspiele zu gewinnen.

Ziel bei der Erstellung von ML-Modellen ist es, Modelle zu trainieren, die Eingaben richtig vorhersagen können, die sie beim Training nicht verarbeitet haben.

Bei Chatbots ist eine häufig genutzte Natural Language Understanding (NLU) Aufgabe das Herausfinden der Intention des Nutzers (Intentionsklassifizierung). Diese Aufgabe wird mithilfe eines Chatbotentwicklers trainiert, indem dieser Intentionen erstellt und für diese Trainingsbeispiele bereitstellt (mehr dazu in Kapitel 2.4).

Angenommen, die Beispieldaten, mit denen ein Modell erstellt wird, werden in Trainings- und Testdaten unterteilt. Dann werden Trainingsdaten verwendet, um das Modell zu trainieren, während die Performanz des Modells mit den Testdaten evaluiert wird [RN21, S. 672]. Wenn die Testdaten durch das Modell größtenteils korrekt vorhergesagt werden, dann scheint es zu generalisieren (engl. generalization) [RN21, S. 672].

Trainingsdaten haben eine bestimmte **Verzerrung** (engl. bias) und Schwankungen bzw. Varianz in den Werten (engl. variance) [RN21, S. 672 - 673]. Ein Modell kann dabei **wenig angepasst** (engl. underfitting) sein, wenn die Verzerrung, die das Modell aus den Trainingsdaten erlernt, zu sehr vereinfacht [RN21, S. 672 - 673]. Eine **hohe Anpassung** tritt auf, wenn die Trainingsdaten eine hohe Varianz besitzen [RN21, S. 672 - 673]. Bei Über- und Unteranpassung ist die Folge eine schlechte Performanz auf den Testdaten bzw. Daten, die das Modell vorher noch nicht verarbeitet hat. Dieses Problem wird auch **Verzerrung-Varianz-Dilemma** (engl. bias-variance-tradeoff) genannt [RN21, S. 672 - 673].

Ein Modell trainiert, indem es eine Verlustfunktion (engl. loss function) maximiert oder minimiert, abhängig von der verwendeten Funktion und dem gewünschten Ziel [RN21, S. 687-688]. Bei Sprachmodellen wird die Kreuzentropie-Verlustfunktion verwendet (siehe Kapitel 2.2.5).

Um die Vorhersageperformanz von Modellen zu evaluieren, können verschiedene Evaluationsmetriken verwendet werden, die vom Ziel der Aufgabe des Modells abhängig sind. Bei binären Klassifizierungsproblemen werden häufig die Metriken Treffergenauigkeit (engl. accuracy), Genauigkeit (engl. precision), Trefferquote (engl. recall) und F1-Score angegeben. Für diese Metriken werden die vier möglichen Fälle einer binären Klassifizierung zusammengezählt:

1. Wirklich positive Klassifizierungen (engl. true positive, kurz TP)
2. Wirklich negative Klassifizierungen (engl. true negative, kurz TN)
3. Falsch positive Klassifizierungen (engl. false positve, kurz FP)
4. Falsch negative Klassifizierungen (engl. false negative, kurz FN)

Aus der Anzahl dieser vier möglichen Fälle werden die Metriken wie folgt berechnet [Gooa; Goob; Sci]:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

$$\text{F1} = 2 * \frac{precision * recall}{precision + recall} \tag{2.4}$$

Beim Deep Learning (DL) werden sogenannte Künstliche Neuronale Netze (KNN) genutzt. KNNs sind Netze aus künstlichen Neuronen, deren Vorbild echte Nervenzellen bzw. Neuronen sind, wobei die Ähnlichkeit allerdings oberflächlich ist [RN21, S. 801]. DL zeichnet aus, dass ein KNN aus vielen Schichten (engl. Layers) bestehen und deshalb viele Schritte von Eingabe nach Ausgabe erfolgen [RN21, S. 801]. Der Vorteil am Vorgehen des DL ist, dass durch die vielen Berechnungsschritte in einem KNN, die Eingabevariablen in komplexen Weisen interagieren können und so die Komplexität von Daten der echten Welt dargestellt werden kann [RN21, S. 801].

Ein künstliches Neuron besteht nach dem McCulloch und Pitts Design [MP43] aus der Summe aus gewichteten Eingabeverbindungen und der Anwendung einer nicht-linearen Funktion, um eine Ausgabe zu berechnen [RN21, S. 802 - 803]. Die nicht-lineare Funktion eines Neurons wird Aktivierungsfunktion (engl. activation function) genannt [RN21, S. 803].

Abbildung 2.1 zeigt den Aufbau eines künstlichen Neurons. Die Eingaben (x_n) werden mit gewichteten Verbindungen (w_{nj}) zum Neuron berechnet [RN21, S. 802 - 803]. Die Summe der gewichteten Verbindungen und

Eingaben wird anschließend von einer Aktivierungsfunktion (φ) verarbeitet [RN21, S. 803].

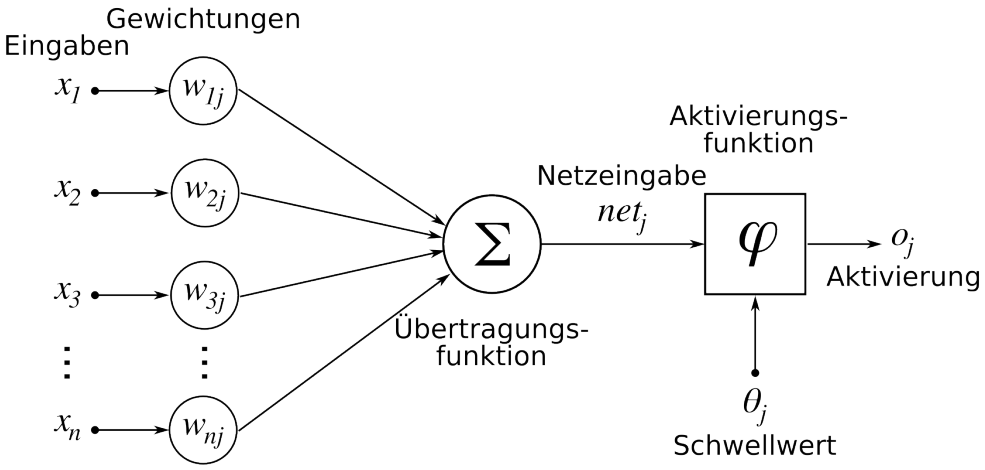

Abbildung 2.1: Aufbau eines künstlichen Neurons. Quelle: [Chr05].

Ein Beispiel für ein einfaches neuronales Netzwerk ist das sogenannte Feed-forward-Netzwerk (zu dt. etwa Vorwärtskopplungsnetzwerk) [RN21, S. 802]. Das Netzwerk ist ein gerichteter azyklischer Graph, besteht aus Eingabe- und Ausgabeknoten und die Verbindungen sind von Eingabe- nach Ausgabeknoten gerichtet [RN21, S. 802]. Jeder Knoten im Netzwerk berechnet eine Funktion aus seinen Eingaben und gibt das Ergebnis an die Ausgabeknoten weiter [RN21, S. 802]. In Abbildung 2.2 ist ein Beispiel für ein Feedforward-Netzwerk. Das Netzwerk besteht aus drei Layern: einem Eingabe-Layer, einem versteckten Layer (engl. hidden layer) und einem Ausgabe-Layer.

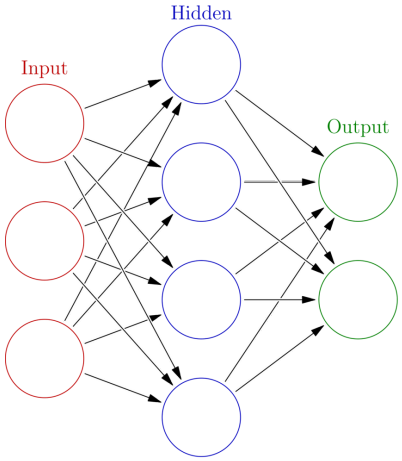

Abbildung 2.2: Aufbau eines Feedforward-Netzwerks nach [Add19].

Ein weiteres Beispiel für ein neuronales Netzwerk ist ein rekurrentes Netzwerk (engl. recurrent neural network kurz RNN) [RN21, S. 802]. Bei RNNs wird der Ausgabewert der Neuronen wieder als Eingabe genutzt, um so sequentielle Eigenschaften der Eingabe zwischenzuspeichern [RN21, S. 802].

Abbildung 2.3 zeigt links (a) ein rekurrentes Netzwerk. Eine Eingabe (x) wird verarbeitet und ein versteckter Status (z) und eine Ausgabe (y) wird berechnet. Die Gewichte (W) werden beim Training im Netzwerk angepasst. Auf der rechten Seite (b) ist das Netzwerk aufgeklappt für drei Schritte. Die Eingaben werden sequentiell verarbeitet und der versteckte Status (z) berücksichtigt beispielsweise bei x_2 die Berechnung von x_1 im Gewicht $W_{z,z}$.

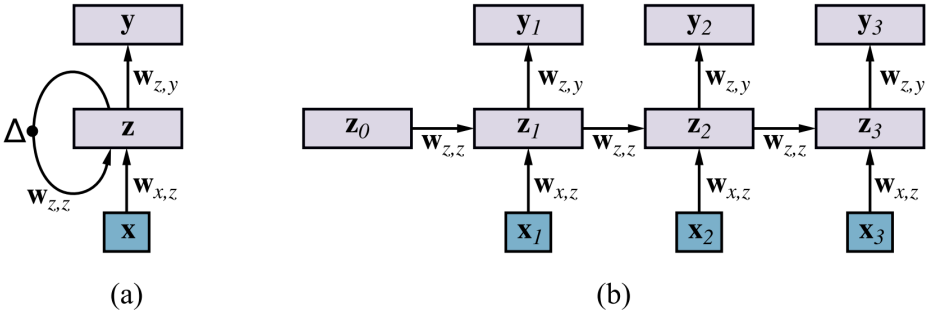

Abbildung 2.3: Aufbau eines rekurrenten Netzwerks nach [RN21, S. 824].

Damit ein neurales Netzwerk lernt, d. h. die Gewichte der Verbindungen zwischen den Neuronen angepasst werden, wird das Gradientenverfahren (engl. gradient descent) genutzt [RN21, S. 805]. Dabei wird der Gradient einer vorher ausgewählten Verlustfunktion (engl. loss function) berechnet und versucht, den Verlust zu verringern, indem die Gewichte entlang der Gradientenrichtung angepasst werden [RN21, S. 805]. Die Anpassung der Gewichte wird ausgehend vom Ausgabelayer über die Hidden Layer zurück zum Anfang der Gewichte getätigt [RN21, S. 805]. Dieser Vorgang wird auch als Backpropagation bezeichnet, weil die Anpassung durch den Fehler wieder zurück durch das Netzwerk getragen wird [RN21, S. 806].

Das Problem des verschwindenden Gradienten (engl. vanishing gradient) kann auftreten, wenn Netzwerke mit vielen Layern den Fehler zurück durch das Netzwerk geben, aber der Fehler zu gering ist (gleich oder nahe null), sodass Gewichte nicht mehr angepasst werden [RN21, S. 806].

Um unabhängig von der Erstellung vieler Trainingsdaten zu sein, werden weitere Trainingsformen verwendet wie unüberwachtes Lernen (siehe 2.2.1), teilüberwachtes Lernen (engl. semisupervised) und Transferlernen [RN21, S. 827]. Bei teilüberwachten Lernverfahren wird eine Mischung aus unüberwachtem und überwachtem Lernen angewandt. Dabei können die Modelle von einigen Daten mit Labels lernen und ihre Performanz weiter verbessern in Bezug auf Daten ohne Label [RN21, S. 827]. Das Transferlernen bedeutet, dass ein Modell mit der Erfahrung vom Training für eine bestimmte Aufgabe A für eine andere Aufgabe B trainiert werden kann und so vom Training für Aufgabe A profitieren kann [RN21, S. 832]. Für neuronale Netze bedeutet das, dass ein Modell mit seinen Gewichten kopiert wird und die Gewichte für eine andere Aufgabe angepasst werden [RN21, S. 832]. Ein Vorteil dieses Vorgehens ist, dass Zeit gespart werden kann für das Training eines Modells, weil das Modell nicht von Grund auf mit vielen Daten trainiert werden muss [RN21, S. 832]. Eine weitere Form des Lernens ist das

Lernen für mehrere Aufgaben gleichzeitig (engl. multitask learning) [RN21, S. 833]. Die Annahme hinter diesem Ansatz ist, dass ein Modell, das mehrere Aufgaben in einem ähnlichen Kontext bewältigen kann, besser die echte Welt repräsentiert [RN21, S. 833].

2.2.2 *Deep Learning für NLP*

DL kann in Verbindung mit NLP genutzt werden. Dabei haben sich in der Vergangenheit neuronale Architekturen mit rekurrenten Netzwerken für Verarbeitung von Text bewährt, da diese Bedeutung und Kontext erfassen können und Eingaben sequentiell verarbeiten [RN21, S. 907].

Ein weiterer Punkt, der für die Verarbeitung von Text von Bedeutung ist, dass KNNs nicht direkt Wörter oder Wortketten verarbeiten können. Deshalb werden Wörter als Vektoren dargestellt [RN21, S. 907]. In Kapitel 2.2 wurden in einem Beispiel mögliche Vektordarstellungen vorgestellt, wie das BoW- oder das N-Gramm-Modell. Durch DL ist es möglich Wörter als niedrig dimensionale Vektoren zu repräsentieren, die Bedeutung von Wörter repräsentieren, sogenannte Worteinbettungen (engl. word embeddings) [RN21, S. 907 - 908]. Dieses Vorgehen hat den Vorteil, dass ähnliche Wörter ähnliche Vektoren haben können [RN21, S. 908].

Das Nutzen von Worteinbettungen ist für verschiedene NLP-Aufgaben bewährt [RN21, S. 908]. Diese Vektoren können in verschiedenen Sprachen vortrainiert genutzt werden. Beispiele für trainierte Worteinbettungen sind word2vec [Mik+13], GloVe (Global Vectors) [PSM14] und FastText [Boj+16] [RN21, S. 908].

Ein Vorteil der Verwendung von Wortrepräsentationen als Vektoren ist, dass Vektoren verglichen werden können, beispielsweise mit der Kosinusähnlichkeit [Vaj+20, Kapitel 3]. Die Kosinusähnlichkeit misst dabei den Winkel zwischen zwei Vektoren [Vaj+20, Kapitel 3]. Der Wertebereich der Kosinusähnlichkeit ist -1 bis 1, bzw. die Winkel 0° bis 180° wobei ein Wert nahe 1 (Kosinus von 0°) Ähnlichkeit impliziert, während niedrigere Werte bedeuten, dass die Vektoren nicht ähnlich sind [Vaj+20, Kapitel 3].

Eine NLP-Aufgabe, die mit neuronalen Ansätzen bewältigt werden kann, ist das Übersetzen zwischen Sprachen (engl. neural machine translation kurz NMT) [RN21, S. 915]. Übersetzung kann beispielsweise mit zwei rekurrenten neuronalen Netzen (RNN) umgesetzt werden [RN21, S. 916]. Die zwei Netze werden dabei auch Encoder und Decoder genannt [RN21, S. 916]. Dabei wird mit dem Encoder der zu übersetzende Satz verarbeitet und als Vektorrepräsentation mit fester Länge an ein zweites RNN, den Decoder übergeben, der dann aus dem Vektor des Encoders eine Übersetzung berechnet [RN21, S. 916]. Die Netzwerkarchitektur, bei der dieses Vorgehen verwendet wird, wird auch Sequenz-zu-Sequenz (engl. sequence-to-sequence) genannt [RN21, S. 916].

Ein weiterer Baustein, um Sequenz-zu-Sequenz-Modelle zu verbessern, ist der sogenannte Attention-Mechanismus nach der Arbeit von Bahdanau, Cho und Bengio [BCB14] [RN21, S. 917]. Der Attention-Mechanismus ist dabei ei-

ne Gewichtung der Eingabewörter in Vektorform, der angibt, welche Wörter wichtig zur Generierung eines nächsten Wortes sind [RN21, S. 917-918].

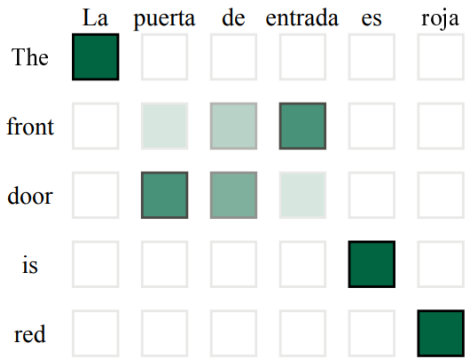

Abbildung 2.4: Beispiel des Attention-Mechanismus zur Übersetzung von Englisch zu Spanisch nach [RN21, S. 917].

Abbildung 2.4 zeigt ein Beispiel für den Attention-Mechanismus bei der Übersetzung von Englisch zu Spanisch des Satzes „Die Eingangstür ist rot“ (Beispiel von [RN21, S. 917]). Dabei stellen dunkelgrüne Felder eine hohe Gewichtung zwischen Wörtern beider Sprachen dar. Beispielweise wird im ersten Schritt der Übersetzung eine hohe Gewichtung auf das Wort *The* gelegt und das Modell generiert daraus das wahrscheinlichste Wort im Spanischen *La*. Im zweiten Schritt wird das Wort *door* am höchsten gewichtet, da sich die Wortreihenfolge im Spanischen unterscheidet und das Wort *puerta* wird generiert. Das neuronale Netzwerk lernt im Training, welche Wörter wichtig sind zur Generierung des nächsten Wortes mithilfe von Trainingsbeispielen [RN21, S. 917].

2.2.3 *Transformer-Architektur*

Die Arbeit von Vaswani u. a. führte schließlich zu einem Durchbruch für neuronale Sprachmodelle, indem es die Transformer-Architektur einführte [Vas+17]. In der Transformer-Architektur wird der Attention-Mechanismus genutzt, um Kontext zu modellieren [Vas+17], [RN21, S. 919]. Ein weiterer Vorteil der Transformer-Architektur ist, dass im Gegensatz zu RNN viele Berechnungen parallelisierbar sind und dadurch die Trainingszeit von Modellen reduziert werden kann [Vas+17]. Abbildung 2.5 zeigt einen Überblick über die Transformer-Architektur nach Vaswani u. a. [Vas+17]. Dabei hat der Transformer eine Encoder-Decoder-Struktur, die in Abbildung 2.5 zu sehen ist.

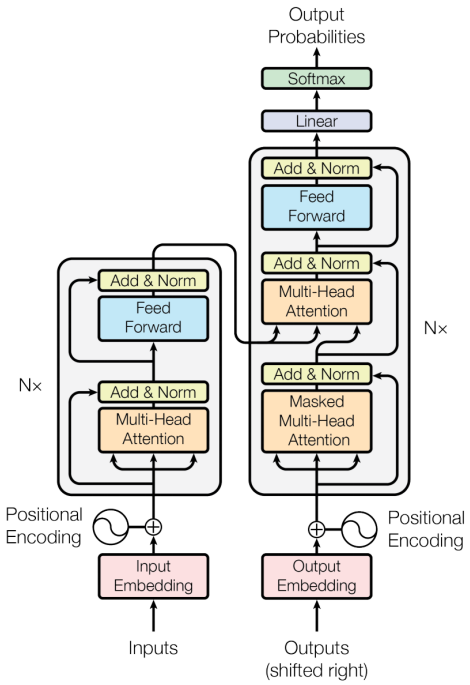

Abbildung 2.5: Aufbau der Transformer-Architektur mit Transformer-Encoder (links) und Decoder (rechts) nach Vaswani u. a. [Vas+17].

Der Transformer-Encoder besteht aus zwei Unterlayern, einem Multi-Head-Attention- und einem Feed-Forward-Layer, die jeweils von einem Normalisierungslayer gefolgt werden [Vas+17, S. 3]. Vor der Normalisierung wird das Ergebnis der Unterlayer mit der vorherigen Eingabe addiert, das ist möglich durch eine Residualverbindung (engl. residual connection), die neben den Unterlayern verläuft und die Eingabe des Unterlayers weiterleitet [Vas+17, S. 3]. In der Arbeit von Vaswani u. a. werden sechs Transformer-Blöcke genutzt, in Abbildung 2.5 wird dies durch Nx (N mal) notiert [Vas+17, S. 3].

Der Transformer-Decoder hat im Vergleich zum Encoder zusätzlich ein Multi-Head-Attention-Layer, das die Ausgabe des Encoders nutzt (Abbildung 2.5 Mitte rechts). Außerdem wird im ersten Multi-Head-Attention-Layer eine Maske hinzugefügt. Die Maske kann genutzt werden, um zu verhindern, dass Attention-Werte für nachfolgende Wörter beim Dekodieren berechnet werden [Phi], [Vas+17, S. 3].

Die Eingaben (Tokens) werden in Worteinbettungen umgewandelt [Phi]. Die Vektoren sind erlernt und es gibt eine erlernte Vektorrepräsentation pro Token im Vokabular eines Modells [Phi].

Um die Reihenfolge der Eingaben zu speichern, wird die Position durch eine Positionseinbettung (engl. positional embedding) kodiert und mit den Worteinbettungen addiert [Vas+17, S. 5 - 6]. Positionseinbettungen werden beim Encoder und Decoder für die Eingaben genutzt [Vas+17, S. 6].

In der Transformer-Architektur wird auch die sogenannte Self-Attention verwendet [RN21, S. 919]. Der Unterschied zum Attention-Mechanismus mit

Sequenz-zu-Sequenz RNNs aus Bahdanau, Cho und Bengio ist, dass in jedem Transformerblock der Attention-Mechanismus für die Eingabesequenz selbst ausgeführt wird, anstatt nur zwischen Encoder und Decoder [BCB14], [RN21, S. 919]. Das bewirkt, dass so für die einzelnen Eingaben selbst eine Gewichtung der relevanten Wörter stattfindet [RN21, S. 919].

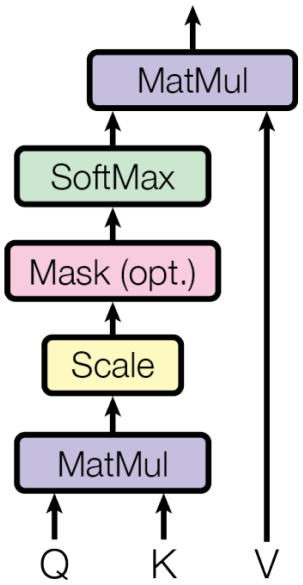

Abbildung 2.6: Darstellung des Transformer Attention-Mechanismus namens skaliertes Punktprodukt nach Vaswani u. a. [Vas+17].

Der Attention-Mechanismus in der Arbeit von Vaswani u. a. nennt sich skaliertes Punktprodukt (engl. scaled dot-product) und wird in Abbildung 2.6 dargestellt [Vas+17, S. 4]. Die Buchstaben stehen dabei für folgende Vektoren: Q: Abfrage, K: Schlüssel und V: Wert, die aus den Eingabevektoren erzeugt werden [Vas+17, S. 4]. Die einzelnen Vektoren von Q, K und V werden zur Berechnung als drei Matrizen zusammengefügt, pro Art von Vektor eine Matrix [Vas+17, S. 4]. Folgende Schritte werden ausgeführt, um die Attention zu berechnen (Abbildung 2.6, vgl. [Vas+17, S. 4]):

MATMUL 1 Matrix Multiplikation von Q und K

SCALE Skalierung des Ergebnisses mit der Wurzel der Dimension d_k des Vektors K $\frac{QK}{\sqrt{d_k}}$

MASK (OPTIONAL) Anwendung einer Maske (im Transformer-Decoder)

SOFTMAX Normalisiert die Werte der Vektoren in den Wertebereich $(0, 1)$, die Summe der einzelnen Reihen der Matrix ist danach eins. $softmax(\frac{QK}{\sqrt{d_k}})$

MATMUL 2 Matrix Multiplikation mit dem Ergebnis von Q und K mit V $softmax(\frac{QK}{\sqrt{d_k}})V$

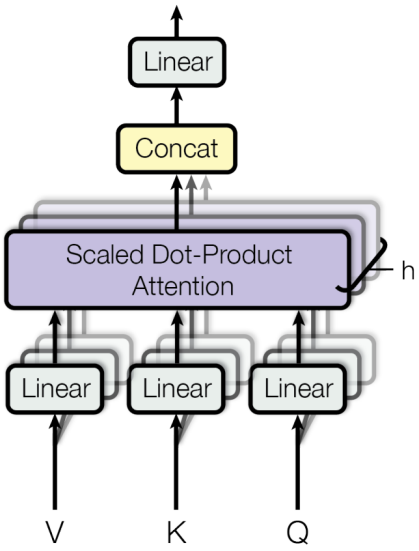

Abbildung 2.7: Multi-Head-Attention nach Vaswani u. a. [Vas+17].

In der Transformer-Architektur wird Multi-Head-Attention verwendet [Vas+17, S. 4]. Bei der Multi-Head-Attention werden die Vektoren Q, K und V linear projiziert mit verschiedenen erlernten linearen Projektionen [Vas+17, S. 4]. Jede Projektion entspricht einem Head [Vas+17, S. 4]. Bei jeder projizierten Version von Q, K und V wird die Attention-Funktion parallel ausgeführt, schließlich werden die Ergebnisse zusammengeführt [Vas+17, S. 4]. Das Vorgehen wird in Abbildung 2.7 dargestellt.

2.2.4 *Vortrainierte neuronale Sprachmodelle*

Aus der Transformer-Architektur gingen schließlich Modelle hervor, die zum Transferlernen genutzt werden können. Der Vorgang des Nachtrainierens wird im Englischen als *Finetuning* bezeichnet.

Eines der vortrainierten Sprachmodelle ist Bidirectional Encoder Representations from Transformers (BERT), welches von Devlin u. a. (Google) entwickelt wurde auf Grundlage der Transformer-Architektur [Dev+18]. BERT besteht im Gegensatz zur klassischen Transformer-Architektur nur aus den Transformer-Encodern [Dev+18].

Ein weiteres Beispiel für ein Sprachmodell das Teile der Transformer-Architektur nutzt ist die Reihe der Generative Pre-Training (GPT) Modelle. Bei GPT wird ein angepasster Decoder des Transformers genutzt [RN18, S. 5].

Das Training dieser Sprachmodelle besteht daraus, dass sie unüberwacht aus einem großen Textkorpus lernen.

Bei BERT werden zwei Aufgaben für das Vortraining des Modells genutzt, zum einen Masked Language Modelling (MLM) (zu Deutsch etwa maskierte Sprachmodellierung) und Next Sentence Predicition (NSP) (zu Deutsch etwa Vorhersage des nächsten Satzes) [Dev+18, Kap. 3.1]. Beim MLM werden Wörter per Zufall durch mehrere Maskierungs-Tokens ersetzt und das Modell muss diese Tokens vorhersagen; dadurch können Gewichte im Modell

angepasst werden [Dev+18, Kap. 3.1]. Die Aufgabe der NSP zielt darauf ab, die Beziehung zwischen zwei Sätzen zu erlernen [Dev+18, Kap. 3.1]. Dabei werden zwei Sätze aus dem Trainingskorpus ausgewählt und das Modell muss bestimmen, ob ein Satz B auf einen Satz A folgt [Dev+18, Kap. 3.1].

Das GPT Modell nutzt als unüberwachtes Vortraining die Vorhersage von einem nächsten Wort abhängig von einem gegebenen Text. Diese Aufgabe wird von den Autoren Radford und Narasimhan als Standard-Sprachmodellierungsziel bezeichnet [RN18].

Das Text-to-Text Transfer Transformer (T5) Modell ist ein weiteres Sprachmodell der Firma Google [Raf+19]. Das T5-Modell besteht aus dem Transformer Encoder-Decoder und ist ähnlich originalen Architektur von Vaswani u. a. [Raf+19]. Die Vortrainingsmethode für T5 ist ähnlich zu dem in BERT d. h. es wird Masked Language Modeling bzw. das Vorhersagen von maskierten Tokens trainiert [Raf+19]. Die veröffentlichten T5-Modelle sind außerdem für mehrere Aufgaben gleichzeitig trainiert, wie beispielsweise die Übersetzung von Englisch zu Deutsch [Raf+19]. Diese Aufgaben können dann mit einem Prefix wie „translate English to German: This is good.“ an das Modell übergeben werden und dieses kann die richtige Aufgabe erkennen und als Rückgabewert „Das ist gut.“ ausgeben [Raf+19]. Ein Vorteil von T5 ist, dass alle Aufgaben als Text-zu-Text-Problem gelöst werden können [Raf+19], im Gegensatz dazu steht Beispielsweise das BERT Modell, das neue Ausgabelayer benötigt für verschiedene Aufgaben [Raf+19; Gro+21]. Das Text-zu-Text-Framework des T5-Modells macht dieses sofort nutzbar für ein Finetuning, ohne weitere Layer hinzuzufügen oder ähnliche Anpassungen vornehmen zu müssen.

Die vorherigen genannten neuronalen Sprachmodelle sind mit Daten in englischer Sprache vortrainiert [Dev+18; RN18; Raf+19]. Allerdings soll im Rahmen dieser Arbeit Textgenerierung mit deutscher Sprache umgesetzt werden. Es sind Modelle von Interesse, die mit deutschsprachigen Texten vortrainiert wurden.

Dabei kann zwischen mono- und multilingualen Sprachmodellen unterschieden werden. Monolinguale Sprachmodelle werden ausschließlich mit Textdaten einer Sprache trainiert, während multilinguale Sprachmodelle mit mehreren Sprachen vortrainiert werden.

Vortrainierte Sprachmodelle wurden auf der Webseite huggingface[1] gesucht, da dies eine der größten Bibliotheken für vortrainierte Sprachmodelle zum aktuellen Zeitpunkt (2022) ist und die *transformers* Bibliothek des Unternehmens Huggingface gleichzeitig auch für das Finetuning von vortrainierten Sprachmodellen verwendet werden kann [Wol+20].

Das Unternehmen deepset.ai[2] hat einige Sprachmodelle wie BERT für die deutsche Sprache vortrainiert[3], diese vortrainierten Modelle sind ebenfalls in der huggingface Bibliothek verfügbar [CSM20].

Ein weiteres Unternehmen das deutschsprachige Sprachmodelle veröffentlicht, ist das Münchener Digitalisierungs Zentrum (MDZ) an der Bayerischen

1 https://huggingface.co/models Letzter Aufruf 22.06.2022
2 https://www.deepset.ai/ Letzter Aufruf 22.06.2022
3 https://www.deepset.ai/models Letzter Aufruf 22.06.2022

Staatsbibliothek[4]. Zusätzlich zu BERT Modellen wurde auch ein GPT-2 Modell mit deutschsprachigen Daten vom MDZ trainiert und veröffentlicht.

Des Weiteren werden multilinguale Modelle von großen Firmen veröffentlicht, wie beispielsweise Google. Für T5 gibt es eine multilinguale Version, die mit Texten in 101 verschiedenen Sprachen vortrainiert wurde [Xue+20].

Zur Zeit dieser Arbeit (2022) werden in der Forschung Sprachmodelle mit hohen Parameterzahlen genutzt. Das liegt daran, dass die Leistung der Modelle mit höherer Parameteranzahl skaliert, das zeigte zum Beispiel das Sprachmodell GPT-3 (2020) mit 175 Milliarden trainierbaren Parametern [Bro+20]. Die erhöhte Parameteranzahl führt auch dazu, dass für bestimmte Aufgaben wenig Trainingsbeispiele nötig sind (engl. few-shot learning) [Bro+20]. Ein Modell aus dem Jahr 2022 ist PaLM, das vom Unternehmen Google erstellt wurde und 540 Milliarden trainierbare Parameter hat [Cho+22]. Im Vergleich dazu hat das mT5-base Modell eine Größe von 580 Millionen Parametern [Xue+20].

Ein Nachteil dieser großen Sprachmodelle ist die Verwendbarkeit, da viel Rechenleistung benötigt wird, um diese auszuführen (beispielsweise mehrere Grafikkarten). Große Sprachmodelle wie GPT-3 sind außerdem nur kostenpflichtig verfügbar[5].

2.2.5 *Performanzmessung bei neuronalen Sprachmodellen*

Beim Vortraining von neuronalen Sprachmodellen werden verschiedene Metriken genutzt, um die Qualität des trainierten Modelles zu bestimmen. Diese Metriken fungieren als Verlustfunktion, die im Laufe des Trainings optimiert wird durch das Anpassen der Parameter des neuronalen Netzwerkes [Huy19]:

KREUZENTROPIE Q ist eine Verteilung, die aus einem Beispieltext erlernt wird und P eine natürliche Sprache. Um die Nähe der zwei Verteilungen zu messen, wird Kreuzentropie genutzt, die wie folgt definiert ist: $H(P,Q) = H(P) + D_{KL}(P||Q)$. $D_{KL}(P||Q)$ ist dabei die Kullback-Leibler-Divergenz von Q zu P bzw. die relative Entropie von P mit Berücksichtigung zu Q. Da die empirische Entropie H(P) nicht optimierbar ist, wird die Kullback-Leibler-Divergenz minimiert.

PERPLEXITÄT Die Perplexität gibt an, wie gut ein Wahrscheinlichkeitsmodell eine Stichprobe vorhersagt. Je kleiner der Perplexitätwert desto besser das Modell. $PPL(P,Q) = 2^{H(P,Q)}$

Ein weiterer Weg, um vortrainierte Sprachmodelle zu evaluieren, ist das Finetuning für bestimmte Aufgaben. Dadurch kann die Transferleistung vom Vortraining zu einer bestimmten Aufgabe bestimmt werden. Ein Beispiel für dieses Vorgehen sind die Evaluierungen des T5 Sprachmodells [Raf+19].

4 https://github.com/dbmdz Letzer Aufruf 22.06.2022
5 https://openai.com/api/ Letzter Aufruf 25.06.2022

2.2.6 *Tokenisierung von Text für neuronale Sprachmodelle*

Ein Text muss in Tokens unterteilt werden, bevor er von einem neuronalen Sprachmodell verarbeitet werden kann. Die huggingface-Bibliothek stellt zu den meisten vortrainierten Sprachmodellen auch einen Tokenizer zur Verfügung [Wol+20]. Im Fall von T5 oder mT5 entsprechen Tokens Teilwörtern, die über den Algorithmus namens *Sentencepiece* aus einem Textkorpus erlernt werden [KR18; Raf+19; Xue+20]. Der Vorteil von Teilwörtern im Gegensatz zu der Verwendung von ganzen Wörtern ist, dass so ein kleineres Vokabular verwaltet werden kann und seltene Wörter aus erlernten Teilwörtern zusammengesetzt werden können [Hugb]. In der deutschen Sprache ist das beispielsweise vorteilhaft bei zusammengesetzten Wörtern. Für das Wort „Schmutzfangmattenservice“ erstellt der trainierte mT5 Tokenizer folgende Tokens „_S“, „chmutz“, „fang“, „matten“, „service“. Der Unterstrich des ersten Tokens steht bei Sentencepiece für ein Leerzeichen, wobei dieses im Beispiel bei der Zurückführung der Tokens zu einer Zeichenkette entfernt wird [KR18].

2.3 TEXTGENERIERUNG

Um Texte aus dem Web für Chatbots zu strukturieren, werden Möglichkeiten benötigt, aus diesen Texten Fragen zu generieren bzw. bereits vorhandene Fragen umzuformulieren. Da eine Intentionsklassifizierung abhängig von Trainingsbeispielen ist, die ein Chatbotentwickler hinterlegt, kann durch das Ergänzen dieser Trainingsbeispiele eine genauere Klassifizierung erreicht werden [Dam21; Den+22]. Mit dem Fortschritt von neuronalen Sprachmodellen, die durch Transferlernen für spezifische Aufgaben trainiert werden können, ist eine Textgenerierung möglich. In diesem Kapitel wird erläutert, welche Texte für ein Chatbottraining generiert werden können.

2.3.1 *Fragen*

Fragen sind eine Möglichkeit, um durch Kommunikation Informationen bzw. Antworten zu erhalten. In einem Chatbot-Szenario sind Fragen eine Möglichkeit für Nutzer ihre Anliegen zu kommunizieren. Es wird dabei in verschiedene Gruppen von Fragen unterschieden (nach [Bus08, S. 200]):

ENTSCHEIDUNGSFRAGEN (auch Ja-Nein-Fragen) Es kann nur mit Ja oder Nein geantwortet werden. Beispiel: „Hat das Essen geschmeckt?“

ALTERNATIVFRAGEN In der Frage werden verschiedene Optionen gegeben aus der eine Antwort gewählt werden kann. Beispiel: „War das Essen gut oder schlecht?“

ERGÄNZUNGSFRAGEN (auch W-Fragen) Diese Fragen nutzen ein Interrogativpronomen (Fragewort), um eine Frage einzuleiten. Beispiel: „Was ist dein Lieblingsgericht?“

ECHOFRAGEN Fragen, die vorherige Fragen aufgreifen, um diese zum Verständnis zu wiederholen. Beispiel: „Was am Essen fandest du gut?“

2.3.2 *Paraphrasierung*

Paraphrasieren bedeutet, dass zwei oder mehr Texte die gleiche Bedeutung (Semantik) haben [GVDCB13]. Dabei können Unterschiede in den verwendeten Wörtern (lexikalisch), Wortgruppen (Phrasen) und im Satzbau (Syntax) vorhanden sein [GVDCB13]. Dadurch, dass beim Training von Chatbots bzw. der Intentionsklassifizierung Textbeispiele genutzt werden, kann es Sinn ergeben, diese automatisiert zu paraphrasieren. Durch die automatisch generierten Paraphrasen kann die Intentionsklassifizierung in ihrer Genauigkeit verbessert werden (vgl. [Dam21; Den+22]).

2.3.3 *Automatische Evaluierung*

Das Generieren von Fragen aus einem Text sowie das Paraphrasieren von Fragen bzw. Äußerungen gehören zur Textgenerierung. Da im Rahmen dieser Arbeit Textgenerierung genutzt wird, um aus Texten aus dem Internet Fragen für einen Chatbot zu generieren, muss betrachtet werden, wie gut bestehende Lösungen Fragen aus einem Text generieren können. Automatische Metriken für die Textgenerierung bieten dabei eine schnelle Möglichkeit ein trainiertes Modell evaluieren.

Allerdings ist es abhängig von der Textgenerierungsaufgabe, wie gut die Metriken das Generieren von bestimmten Texten repräsentieren können. Für diese Arbeit werden die Metriken BLEU, METEOR, ROUGE und BERTScore genauer betrachtet, da sie in anderen Arbeiten der Fragengenerierung (Kapitel 3.2) verwendet werden und ein grober Anhaltspunkt sind, wie gut die Modelle funktionieren. Die Metriken wurden bereits für andere Textgenerierungsaufgaben wie Übersetzung oder Zusammenfassung evaluiert und korrelierten für diese Aufgaben mit den Ergebnissen menschlicher Gutachter (vgl. [Pap+02; Lin04; BL05; Zha+19]). Alle automatischen Metriken benötigen Daten mit Label, da die durch Menschen erstellte Label (auch Referenz genannt) mit der Generierung durch das Modell verglichen werden.

2.3.3.1 *BLEU-Score*

Der BLEU-Score (engl. Bi-Lingual Evaluation Understudy) wurde von Papineni u. a. 2002 veröffentlicht, um maschinelle Übersetzung automatisch zu evaluieren [Pap+02]. Ein Vorteil gegenüber menschlicher Evaluation ist, dass der BLEU-Score schnell berechnet werden kann und keine menschliche Arbeitskraft und somit auch weniger Zeit und Ressourcen benötigt [Pap+02]. Dieser Punkt ist für die Textgenerierungsaufgaben ebenfalls valide.

Das Ziel des BLEU-Scores ist es zu bestimmen, ob eine machinelle Übersetzung gleich oder nahe an einer Übersetzung durch einen Experten ist [Pap+02]. Der BLEU-Score ist angelehnt an die Metrik Präzision (siehe For-

mel 2.2), wobei bei BLEU N-Gramme eines Satzes (Referenz) mit den N-Grammen eines generierten Satzes (Hypothese) verglichen werden [Pap+02]. Übereinstimmende N-Gramme zählen als eine Überschneidung und mögliche Treffer sind gleich der Anzahl der N-Gramme des generierten Satzes [Pap+02].

Damit Tokens aus der Hypothese nicht öfter gezählt werden, als sie Vorkommen, wird *Clipping* verwendet. Aufgrund des *Clippings* können Wörter nicht öfter gezählt werden, als sie in der Referenz vorkommen [Pap+02].
Beispiel Fragengenerierung aus dem GermanQuAD-Testdaten [MRP21]:
Hypothese: „Wie viele Menschen leben auf Zypern?"
Referenz: „Wie viele Menschen leben im südlichen Teil von Zypern?"

Die Länge der Hypothese ist sieben Tokens, unter Berücksichtigung des Fragezeichens und der Annahme, dass das Fragezeichen von einem Tokenizer als eigener Token segmentiert wird. Die Tokens „Wie, viele, Menschen, leben, Zypern, ?" sind ebenfalls in der Referenz vorhanden. Das würde für den BLEU-1-Score (gleiche Unigramme) 6/7 (0.857) bedeuten. Für BLEU-2 lauten die gleichen Bigramme „Wie viele, viele Menschen, Menschen leben, Zypern ?" es gibt insgesamt sechs mögliche Bigramme in der Hypothese 4/6 (0.667). Die vorhandenen Trigramme sind „Wie viele Menschen, viele Menschen leben" von fünf möglichen Trigrammen das ergibt einen BLEU-3 Score von 2/5 (0.4). Schließlich ist folgendes 4-Gramm vorhanden „Wie viele Menschen leben" es gibt vier mögliche 4-Gramme, was in einem BLEU-4 Score von 1/4 (0.25) resultiert.

Zusätzlich wird der BLEU-Score für den gesamten Korpus aus dem geometrischen Mittel der verschiedenen N-Gramm Scores berechnet und mit einer sogenannten *brevity penalty* (BP) multipliziert [Pap+02]. Die BP wird verwendet, um kürzere Hypothesen im Score zu normalisieren (wie im Beispiel):

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{2.5}$$

Wobei r die Länge einer Referenz ist, die am nächsten von allen Referenzen an der Länge der Hypothese ist und c die Länge der Hypothese [Pap+02]. Für das Beispiel ergibt die BP:

$$BP = e^{(1-r/c)} = e^{(1-10/7)} = e^{-0.42857} = 0.6514 \tag{2.6}$$

Der BLEU-Score für den gesamten Korpus ergibt sich aus folgender Formel [Pap+02]:

$$\text{BLEU} = BP * \text{geom-mean}(\text{BLEU-1}, \text{BLEU-2}, \text{BLEU-3}, \text{BLEU-4}) \tag{2.7}$$

Für das Beispiel ergibt das folgenden BLEU-Score:

$$\begin{aligned} \text{BLEU} &= 0.6514 * \text{geom-mean}(0.857, 0.667, 0.4, 0.25) \\ \text{BLEU} &= 0.6514 * 0.4889 \\ \text{BLEU} &= 0.3185 \end{aligned} \tag{2.8}$$

2.3.3.2 *ROUGE-L*

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) ist eine weitere Metrik, die zur automatischen Evaluation von Textgenerierung genutzt werden kann [Lin04]. Die Metrik wurde ursprünglich dazu entwickelt, um die Qualität von Textzusammenfassungen automatisch zu erfassen [Lin04]. Lin stellt vier Varianten der Metrik vor, wobei nur die Variante ROUGE-L für diese Arbeit relevant ist [Lin04]:

ROUGE-L Die längste gemeinsame Teilsequenz an Tokens zwischen Hypothese und Referenz wird gemessen. Gegeben sind Zusammenfassungen X (Referenz) und Y (Hypothese) als Text und eine Funktion LCS (Longest Common Subsequence), welche die Länge der größten gemeinsamen Teilsequenz der beiden Zusammenfassungen zurückgibt.

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{2.9}$$

Wobei m die Länge der Zusammenfassung von X ist.

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{2.10}$$

Wobei n die Länge der Zusammenfassung von Y ist.

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}} \tag{2.11}$$

Wobei $\beta = P_{lcs}/R_{lcs}$. ROUGE-L ist demnach ein LCS basierter F-measure [Lin04].

Für die Fragengenerierung verwenden einige Arbeiten die Metrik ROUGE-L (siehe Kapitel 3.2).

2.3.3.3 *METEOR*

METEOR ist eine Metrik, die Unigramme zwischen Hyptohese und Referenz vergleicht [BL05]. METEOR wurde ebenfalls wie BLEU ursprünglich für maschinelle Übersetzung genutzt. Bei der Fragengenerierung verwenden ebenfalls einige Arbeiten diese Metrik (siehe Kapitel 3.2). Zusätzliche Features von METEOR sind das Umwandeln von Wörtern in ihre Stammform (engl. Stemming) [BL05; DL14]. In der neusten Version des METEOR Scores werden zudem für den Vergleich der Unigramme Paraphrasen-Tabellen genutzt,

welche auch für die deutsche Sprache verfügbar sind [DL14]. Der METEOR Score wird aus dem F_{mean} berechnet, wobei Präzision stärker gewichtet wird als Recall:

$$F_{mean} = \frac{10PR}{R + 9P} \tag{2.12}$$

Präzision (P) und Recall (R) wird dabei wie folgt berechnet [BL05]:

$$P = \frac{\text{Anzahl Unigramme in Hypothese und Referenz}}{\text{Anzahl Unigramme in Hypothese}} \tag{2.13}$$

$$R = \frac{\text{Anzahl Unigramme in Hypothese und Referenz}}{\text{Anzahl Unigramme in Referenz}} \tag{2.14}$$

Zusätzlich wird eine Strafe (*Penalty*) berechnet, die zusätzlich zum F_{mean} benötigt wird und wie folgt berechnet wird:

$$\text{Penalty} = 0.5 * \left(\frac{\text{Anzahl Blöcke}}{\text{Anzahl gefundener Unigramme}}\right) \tag{2.15}$$

Blöcke sind dabei die gruppierten Überschneidungen von Unigrammen zwischen Hypothese und Referenz [BL05]. Je weniger Blöcke es gibt d. h. je ähnlicher Referenz und Hypothese sind, desto kleiner ist die Strafe.

F_{mean} und Penalty werden zum METEOR Score berechnet:

$$\text{Score} = F_{\text{mean}} * (1 - \text{Penalty}) \tag{2.16}$$

In der aktuellen METEOR Universal Implementierung werden weitere Konstanten eingeführt, die je nach Sprache gesetzt werden können. Es gibt eine vorgegebene Konfiguration für die deutsche Sprache [DL14].

2.3.3.4 *BERTScore*

Der BERTScore ist eine weitere Metrik mit welcher generierte Texte verglichen werden können [Zha+19]. Im Gegensatz zu den bereits genannten Metriken werden beim BERTScore Ähnlichkeitswerte zwischen Tokens der Hypothesen und Referenzen berechnet [Zha+19]. Anstatt dass wie bei der BLEU Metrik N-Gramme verglichen werden, nutzt BERTScore Worteinbettungen der Tokens, um diese zu vergleichen [Zha+19]. Die Worteinbettungen werden durch ein vortrainiertes BERT Modell erstellt [Zha+19]. Das hat den Vorteil, dass semantisch ähnliche Wörter als Überschneidungen zwischen Hypothese und Referenz gewertet werden können [Zha+19]. Der BERTScore berechnet die Ähnlichkeit zwischen zwei Sätzen als Summe der Kosinusähnlichkeiten der Tokeneinbettungen [Zha+19]. Die Autoren des BERTScores behaupten, dass der BERTScore eine bessere Korrelation als BLEU und METEOR erreichen kann, beispielsweise für die maschinelle Übersetzung [Zha+19].

2.3.3.5 *Zusammenfassung automatischen Metriken*

Die vorgestellten automatischen Metriken und deren unterschiedliche Eigenschaften werden in Tabelle 2.4 zusammengefasst.

Metrik	Tokenisierungseinheit	Vergleichsebene	Zusätzliche Features
BLEU	Wort	N-Gramme	Strafe für zu kurze oder lange Hypothesen (engl. brevity penalty)
ROUGE-L	Wort	Längste gemeinsame Teilsequenz	Stammformbildung
METEOR	Wort	Unigramme	Stammformbildung, Überschneidung von Synonymen und Paraphrasen
BERTScore	Teilwörter	Kosinusähnlichkeit Tokeneinbettungen	Überschneidung ähnlicher Tokens durch Tokeneinbettungen

Tabelle 2.4: Vergleich der automatischen Metriken.

Die Metriken BLEU, METEOR und BERTScore wurden für die Aufgabe der maschinellen Übersetzung getestet und geben alle Korrelationen mit menschlichen Gutachtern an [Pap+02; BL05; Zha+19]. Bei der Metrik ROUGE wurde die Korrelation für die Aufgabe der Zusammenfassung mit Gutachtern gemessen [Lin04].

Zur Fragengenerierung werden die besprochenen Metriken in verwandten Arbeiten verwendet (siehe Kapitel 3.2). Ob die automatischen Metriken mit Gutachtern für die Aufgabe der Fragengenerierung oder Paraphrasierung korrelieren, ist nicht bekannt.

2.3.4 *Dekodierungsmethoden*

Zusätzlich zum Training von neuronalen Sprachmodellen ist bei der Generierung von Text auch die Methode der Dekodierung ein wichtiger Faktor für die Qualität der generierten Texte [Pla20]. Sprachmodelle bestimmen das nächste Wort aus einer Wahrscheinlichkeitsverteilung die von einer Eingabesequenz berechnet wird [Pla20]. Dabei gibt es letztendlich verschiedene Möglichkeiten nächste Wörter zu selektieren [Pla20].

Folgende Dekodierungsmethoden können mit dem huggingface Framework genutzt werden [Wol+20; Pla20]:

GREEDY SEARCH Bei *Greedy Search* wird als nächstes Wort das mit der höchsten Wahrscheinlichkeit gewählt. Ein Nachteil dieser Methode ist, dass das Modell sich wiederholen kann. In Abbildung 2.8 ist *Greedy Search* mit einem Beispiel dargestellt. Zu jedem Schritt der Generierung wird das wahrscheinlichste Wort der Wahrscheinlichkeitsverteilung gewählt (im Beispiel „nice“ und „woman“).

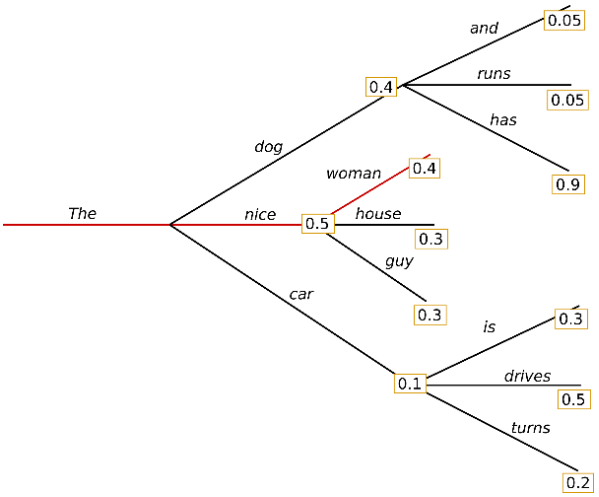

Abbildung 2.8: Darstellung von Greedy Search nach [Pla20].

BEAM SEARCH Um zu vermeiden, dass Wörter mit hoher Wahrscheinlichkeit, wie bei *Greedy Search* übersehen werden, wenn sie einem Wort mit niedriger Wahrscheinlichkeit folgen, kann *Beam Search* verwendet werden. Dabei wird ein Parameter festgelegt, der die Anzahl der *Beams* (Wortsequenzen) festlegt, die am Ende der Suche verglichen werden. Nach dem Vergleich wird die Wortsequenz mit der höchsten Wahrscheinlichkeit selektiert. In Abbildung 2.8 wird *Beam Search* mit einem Beispiel dargestellt.

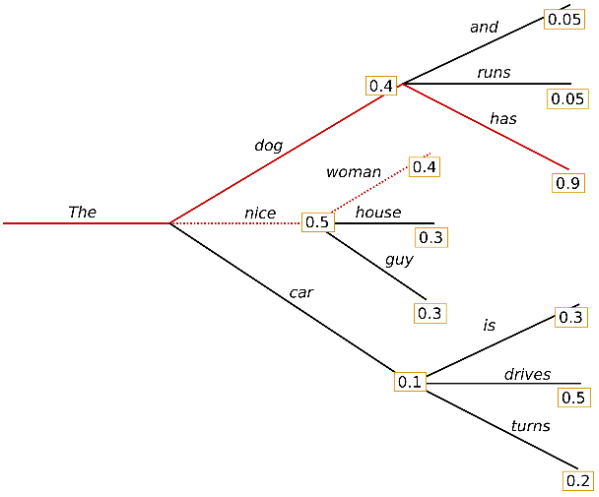

Abbildung 2.9: Darstellung von Beam Search nach [Pla20].

Im Beispiel werden zu jedem Generierungsschritt die zwei wahrscheinlichsten Wortsequenzen (Anzahl der *Beams* gleich zwei) gespeichert. Die gesamte Wahrscheinlichkeit der Wortsequenz „The dog has“ ist dabei größer $0.4 \cdot 0.9 = 0.36$ als die der zweiten gespeicherten Wortsequenz „The nice woman“ $0.5 \cdot 0.4 = 0.2$ [Pla20].

SAMPLING *Sampling* bedeutet, dass zufällig das nächste Wort ausgewählt wird, abhängig von der bedingten Wahrscheinlichkeitsverteilung. Da bei der zufälligen Auswahl eines nächsten Wortes unzusammenhängende Sätze resultieren können, kann die *Temperatur* der Softmax-Funktion angepasst werden, sodass Wörter mit geringer Wahrscheinlichkeit noch geringer werden und Wörter mit hoher Wahrscheinlichkeit noch wahrscheinlicher.

TOP-K SAMPLING In *Top-K Sampling* nach [FLD18] werden die K wahrscheinlichsten nächsten Wörter gefiltert und die Wahrscheinlichkeiten für diese K Wörter umverteilt. Dadurch kann zwischen den besten K Wörter ausgewählt werden.

TOP-P (NUCLEUS) SAMPLING Ein Problem des Top-K Sampling ist, dass unter den Top-K Wörtern auch sehr unwahrscheinliche Wörter vorhanden sein können, die ggf. ausgewählt werden. Dieses Problem kann durch Top-P (nucleus) Sampling vermieden werden, indem anstatt einer Zahl K nun die kumulative Wahrscheinlichkeit angegeben wird, aus welcher ausgewählt werden soll [Hol+19].

Im Folgenden werden weitere Methoden beschrieben, um Problemen der Textgenerierung vorzubeugen [Pla20]:

WIEDERHOLUNG VON N-GRAMMEN Da auch bei Verwendung der genannten Methoden die Wiederholung von N-Grammen auftreten kann, kann eine N-Gramm-Strafe (engl. n-gram penalty) eingeführt werden, die verhindert, dass bereits vorhandene N-Gramme wieder ausgewählt werden.

WIEDERHOLUNGEN VON WÖRTERN Ähnlich zur N-Gramm-Strafe wird eine Strafe für Wörter eingeführt, die bereits im generierten Text vorhanden sind.

MINIMALE LÄNGE Damit der generierte Text nicht zu früh beendet wird, kann eine minimale Länge definiert werden, die ohne einen Ende-der-Sequenz-Token erreicht werden muss.

2.4 CHATBOTS

Ein Chatbot ist ein Programm, das mit einem Nutzer sprach- oder textbasiert kommunizieren kann. Ein Nutzer kann zum Beispiel Fragen an einen Chatbot stellen und der Chatbot kann diese Frage verarbeiten und im Idealfall eine passende Antwort zurückliefern.

In diesem Kapitel wird beschrieben, wie Chatbots kategorisiert werden können und welche Funktionsweisen sie haben. Außerdem werden Vorteile sowie Nachteile von Chatbots beschrieben.

2.4.1 *Terminologie*

Aktuelle Chatbotsysteme nutzen Methoden aus folgenden Bereichen: KI, ML, NLP, Information Extraction (IE) und NLU.

Ein Nutzer kann mit dem Chatbotsystem durch ein Chatprogramm kommunizieren. Eine Nutzereingabe versucht das Chatbotsystem per NLU zu verarbeiten [AM20]. Das bedeutet, dass aus einer Eingabe **Intentionen (engl. intents)** und **Entitäten (engl. entities)** extrahiert werden [NC17; AM20], [Vaj+20, Kapitel 6].

Eine **Intention** ist der Zweck, den ein Nutzer mit seiner Eingabe ausdrücken möchte [Vaj+20, Kapitel 6], [AM20]. Für die Frage „Wie ist mein Kontostand?“ wäre die Intention eine Auskunft über den Kontostand des fragenden Nutzers zu erhalten. Die Intention ist demnach ein Mapping zwischen Nutzereingabe und einer Aktion, die vom Chatbot aufgrund der Eingabe ausgeführt wird [CR18; AM20]. Das Finden der Intentionen ist eine Klassifizierungsaufgabe, deshalb werden dafür Modelle trainiert, die anhand von wenigen Beispielen eine Intention (Klasse) erkennen [Vaj+20, Dialog Examples with Code Walkthrough]. Das Trainieren dieser Modelle ist überwacht, deshalb werden Intentionen durch einen Chatbotentwickler definiert [Vaj+20, Kapitel 6].

Entitäten sind zusätzliche Parameter, die im Text erwähnt werden. Diese Entitäten können mit der IE Aufgabe NER (zu deutsch etwa Erkennung benannter Entitäten) aus dem Text extrahiert werden (siehe Kapitel 2.1.2) [Vaj+20, Kapitel 5], [Vaj+20, Kapitel 6]. Ein Beispiel für eine Entität ist ein Datum wie der 01.01.2022 oder ein Ort wie Darmstadt. Dabei ist zu erwähnen, dass es eine lange Liste an möglichen Klassen für benannte Entitäten gibt, wie beispielsweise Organisationen, Personen usw. und diese Liste abhängig von der Domäne ist, in der die NER erfolgt [Vaj+20, Kapitel 5].

Die Qualität der IE von Intentionen und Entitäten entscheidet über den Erfolg eines Chatbots und damit auch über die Nutzerzufriedenheit.

Zusätzlich zu Intentionen und Entitäten muss ein Chatbotsystem den **Kontext** einer Konversation verwalten, um in späteren Punkten der Unterhaltung diesen Kontext wiederverwenden zu können [NC17; AM20], [Vaj+20, Kapitel 6].

2.4.2 *Kategorien*

Nimavat und Champaneria [NC17] sowie Adamopoulou und Moussiades [AM20] und Vajjala u. a. [Vaj+20, Kapitel 6] beschreiben verschiedene Kategorisierungsmöglichkeiten für Chatbots, die im Folgenden vorgestellt werden.

Ein Chatbot stellt Informationen in einer bestimmten Domäne zur Verfügung. Es wird zwischen einer offenen oder einer geschlossenen Domäne unterschieden:

OFFENE DOMÄNE (engl. open domain) Der Chatbot kann sich über allgemeine Themen unterhalten.

GESCHLOSSENE DOMÄNE (engl. closed domain) Der Chatbot kann nur Unterhaltungen in einer bestimmten Domäne führen.

Chatbots können zusätzlich nach dem zu erreichenden Ziel kategorisiert werden:

INFORMATIV (engl. informative) Nutzer können Informationen vom Chatbot abfragen. Ein Beispiel hierfür ist ein Chatbot der Frequently Asked Questions (FAQ) beantworten kann. Ein FAQ Chatbot enthält Frage-Antwort-Paare und versucht das Anliegen eines Nutzers festzustellen und die passende vorgefertigte Antwort zu finden.

AUFGABENBASIERT (engl. task-based) Nutzer können bestimmte Aufgaben an den Chatbot delegieren wie das Buchen eines Fluges. Der Chatbot muss ggf. Abflugdatum, Uhrzeit und weitere Parameter vom Nutzer erfahren, damit ein Flug erfolgreich gebucht werden kann. Der Dialogablauf eines aufgabenbasierten Chatbots ist vorher von einem Entwickler definiert, um ein bestimmtes Ziel mit der Konversation zu erreichen.

GESPRÄCHIG (engl. conversational) Nutzer können eine Unterhaltung mit dem Chatbot führen. Das Ziel eines gesprächigen Chatbots ist es, auf jede mögliche Frage oder Äußerung des Nutzers bestmöglich zu reagieren.

Eine weitere Kategorisierungsmöglichkeit ist die Methode zur Eingabeverarbeitung und Antwortgenerierung. Hierbei ist zu beachten, dass die Kategorisierungen aufeinander aufbauen:

REGELBASIERT (engl. rule-based) Eine Eingabe wird nach definierten Regeln verarbeitet und die vorher festgelegte Antwort mit der besten Übereinstimmung zur Eingabe wird zurückgegeben.

ABRUFBASIERT (engl. retrieval-based) Eine Eingabe wird zusätzlich in weiteren Ressourcen (Datenbanken, APIs) abgefragt und schließlich wie beim regelbasierten Ansatz wird die am besten übereinstimmende Antwort zurückgegeben.

GENERATIV (engl. generative) Eine Komponente generiert mithilfe von einem Modell eine Antwort aus den gefundenen Ressourcen.

Zudem kann ein Chatbot mit **menschlicher Hilfe** (engl. human-aid) betrieben werden, wenn der Chatbot für eine Frage keine Antwort findet und ein Mitarbeiter für einen solchen Fall eine Antwort formulieren kann [KBH18], [AM20]. Nachteile von menschlicher Hilfe im Chatbot sind, dass diese nicht so schnell wie Computer agieren kann und nicht immer verfügbar ist [KBH18].

2.4.3 *Architektur*

Abbildung 2.10 zeigt eine generelle Architektur für Chatbots nach Adamopoulou und Moussiades [AM20]. Andere Autoren stellen ähnliche Architekturen bzw. Pipelines für ein Dialogsystem bzw. Chatbot vor [NC17], [KBH18], [Vaj+20, Kapitel 6].

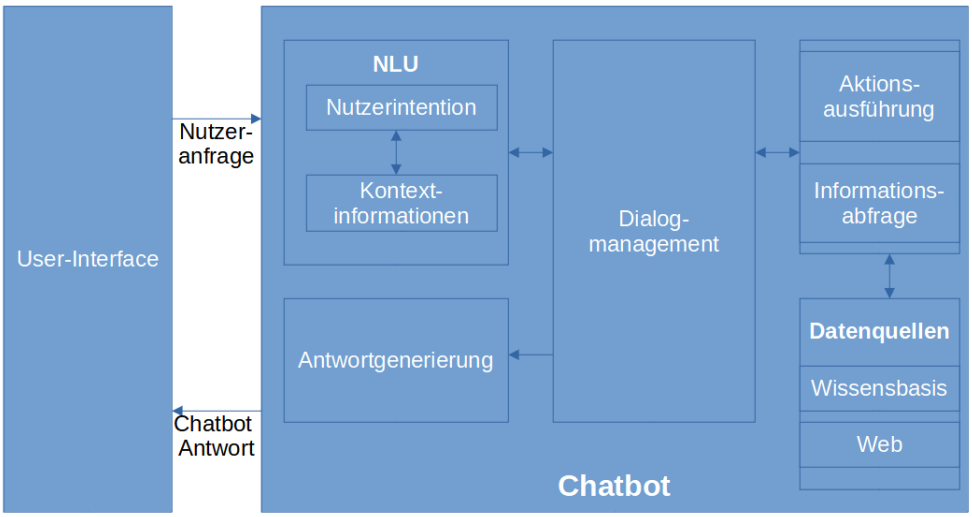

Abbildung 2.10: Chatbotarchitektur nach Adamopoulou und Moussiades [AM20].

Ein Nutzer interagiert mit einem Chatbot über ein User-Interface (UI) [AM20]. Das UI kann ein Chatprogramm sein, wie Whatsapp[6], es sind aber auch Systeme möglich die gesprochene Sprache verarbeiten können wie Amazon's Alexa[7].

Die Nutzeranfrage gelangt zum Chatbot und wird als erstes von einer NLU-Komponente verarbeitet [AM20]. Dabei wird versucht die Intention des Nutzers festzustellen und weitere Kontextinformationen aus der Nachricht zu extrahieren, wie benannte Entitäten [AM20]. In der NLU-Komponente können zudem weitere Kontextinformationen extrahiert werden, Nimavat und Champaneria nennen zusätzlich Mehrdeutigkeitsüberprüfung (engl. ambiguity detection), Stimmungsanalyse (engl. sentiment analysis) und Synonymerkennung (engl. synonym detection) als weitere Beispiele [NC17]. Mit den neu gewonnen Informationen kann der Chatbot das nächste Handeln bestimmen [AM20]. Falls eine Nachricht des Nutzers nicht eindeutig war, kann der Chatbot den Nutzer nach mehr Informationen fragen [AM20].

Falls die Nachricht verständlich für den Chatbot war, kann als nächstes eine Aktion ausgeführt werden [AM20]. Informationen können von verschiedenen Datenquellen wie Wissensdatenbanken, Suchmaschinen oder Application Programming Interfaces (APIs) erhalten werden [AM20]. Nachdem eine Aktion ausgeführt wurde oder Informationen abgefragt wurden, kann

6 https://www.whatsapp.com/ Letzter Aufruf 03.01.2022
7 https://developer.amazon.com/en-US/alexa/alexa-skills-kit/asr Letzter Aufruf 05.01.2022

in der Antwortgenerierungs-Komponente eine passende Antwort gefunden oder generiert werden und diese dem Nutzer geantwortet werden [AM20].

Eine Dialog-Komponente muss während der Unterhaltung den Kontext verwalten. Der Kontext besteht aus der Intention des Nutzers und den identifizierten bzw. fehlenden Entitäten und weiteren Kontextinformationen [AM20].

2.4.4 *Chatbotentwicklung*

In der Softwareentwicklung gibt es verschiedene Vorgehensweisen. Der *Software development life cycle (SDLC)* zeigt beispielsweise, welche Schritte in welcher Reihenfolge nötig sind, um eine Software zu entwickeln. In der Arbeit von Sen, Patel und Sharma werden relevante Phasen der Softwareentwicklung genannt [SPS21]:

- Anforderungsanalyse
- Design
- Implementierung
- Testen
- Wartung

Da ein Chatbot im weiteren Sinne eine Software ist, verläuft die Entwicklung größtenteils ähnlich. Beim Design und der Implementierung des Chatbots können Chatbot-Plattformen helfen. Diese bieten im Gegensatz eine höhere Abstraktionsstufe und ein Chatbot kann dort ohne Programmcode implementiert werden (vgl. [Vaj+20, PizzaStop Chatbot]).

Es kann beispielsweise ein Chatbot entwickelt werden, mit der Anforderung, über eine Konversation die Essensbestellung eines Kunden entgegenzunehmen [Vaj+20, PizzaStop Chatbot]. Die Plattform Dialogflow von Google[8] wird dabei verwendet.

Im Beispiel werden Intentionen wie *PizzaBestellen* erstellt, diese werden mit Beispielsätzen wie „Ich möchte eine Pizza bestellen“ hinterlegt. Durch das Beispiel wird ein Modell zur Intentionserkennung auf der Plattform trainiert [Vaj+20, Building our Dialogflow agent]. Zusätzlich zu Intentionen werden auch domänenspezifische Entitäten benötigt, im Beispiel des Pizza Chatbots sind das verschiedene Beläge, sowie die Größe der Pizza [Vaj+20, Building our Dialogflow agent].

Wie im Beispiel zu sehen ist das Design des Chatbots (In Form von Softwarekomponenten) durch die Plattform vorgegeben, Punkte wie Intentionen mit Beispielsätzen, Entitäten, Antworten usw. müssen von einem Entwickler aus den Anforderungen erstellt werden. Die Implementierungsphase besteht aus dem Erstellen von Intentionen und dazugehörigen Beispielsätzen, Entitäten und von Konversationen zwischen einem Kunden und dem Chatbot.

8 https://cloud.google.com/dialogflow Letzter Aufruf 06.01.2022

In der Testphase kann der Chatbot in einem Demo-Chatfenster getestet werden [Vaj+20, Testing our agent]. Es kann getestet werden, welche Intentionen und Entitäten der Chatbot erkennt und bei welchen der Chatbot Probleme hat [Vaj+20, Testing our agent]. Probleme bei der Erkennung von Intentionen und Entitäten deuten darauf hin, dass ein Modell nicht ausreichend trainiert wurde und deshalb weitere Konfiguration bzw. mehr Trainingsdaten nötig sind.

2.5 INFORMATIONSEXTRAKTION AUS DEM WEB

Informationen im Web liegen in Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript (JS) Dateien oder APIs vor, die mit dem Hypertext Transfer Protocol (HTTP) von Webservern abgefragt werden können und von Webbrowsern (Clients) grafisch dargestellt werden können [NB21, S. 98]. Wissen über die genannten Webtechnologien wird Im Folgenden vorausgesetzt. Um Informationen automatisiert zu extrahieren, können Programme erstellt werden, die Webseiten durchlaufen, sogenannte Webcrawler [NB21, S. 98]. Die Informationen einer Webseite können anschließend in ein Format umgewandelt werden, dass für nachfolgende Aufgaben genutzt werden kann [NB21, S. 98]. Klassische Formate sind Comma Separated Values (CSV) oder JavaScript Object Notation (JSON) [Jsob]. Allerdings ist das Persistieren der Daten in verschiedenen Datenbanksystemen genauso möglich [NB21, S. 98]. Es gibt verschiedene Methoden Informationen aus dem Web zu extrahieren, Nigam und Biswas nennen vier Methoden, die nun erläutert werden [NB21, S. 98 - 102]:

DOM TREE PARSING Beim Parsing des Document Object Model (DOM) [Dom] wird die HTML- und CSS-Datei in eine Baumstruktur eingelesen. Das DOM ist ein Standard für den Baum und dessen Funktionen [Dom]. Aus dieser Baumstruktur können die Inhalte über HTML-Elemente durch verschiedene Methoden extrahiert werden:

XPATH Adressierung des Elements durch den eXtensible Markup Language (XML)-Pfad

CSS Adressierung des Elements durch das CSS-Styling

Nigam und Biswas nennen als Vorteil dieser Methode, dass viele Programmbibliotheken dieses Vorgehen unterstützen. Ein Nachteil ist, dass das Aufbauen des DOM-Baums nicht performant ist im Vergleich zu anderen Methoden. Außerdem können regelmäßige Änderungen der HTML-Datei dieses Vorgehen weniger zuverlässig machen und deshalb benötigen Programme ständige Wartung, wenn sie diese Methode nutzen.

Ein weiteres Problem bei dieser Methode ist die Verwendung von JS zum clientseitigen Rendering des Inhalts [Jav]. Das liegt daran, dass bei der Abfrage vom Server nur statische Informationen abgefragt werden, in der Regel sind das HTML- und CSS-Dateien. Um JS zu be-

rücksichtigen, muss die Webseite in einem Browser ausgeführt werden [Jav].

STRING MATCHING Um das Performanzproblem des DOM-Parsings zu umgehen, können zeiteffizientere String-Matching-Methoden genutzt werden, anstatt den DOM-Baum zu parsen. Die anderen Nachteile des DOM-Parsing bleiben bei dieser Methode bestehen.

SEMANTIC FRAMEWORK Das Semantic-Web kann als eine Erweiterung des Web angesehen werden. Dabei werden zusätzlich strukturierte Daten (Metadaten) zur Verfügung gestellt. Es werden sogenannte Ontologien genutzt, wie beispielsweise schema.org[9], um HTML-Dateien zusätzlich zu annotieren. Ein Beispiel für ein solches Format ist JSON for Linked Data (JSON-LD) [Jsoa]. Das Semantic-Web ist ein Ansatz direkt strukturierte Daten zu erhalten, falls Semantic-Web-Technologien bei einer Webseite genutzt werden.

ML BASIERTES WEBSCRAPING ML-Methoden werden genutzt, um Inhalte einer Webseite zu extrahieren. Dabei werden visuelle Eigenschaften der HTML- und CSS-Datei (HTML-Tag, Textart, Textgröße usw.) verarbeitet oder Methoden aus dem Gebiet der Computer Vision genutzt.

In der Literatur wird zwischen Webcrawler und Webscraper unterschieden, wobei ein Webcrawler zusätzlich zum Extrahieren von Informationen von Webseiten Links zu weiteren Unterseiten folgen kann [NB21, S. 102], [BB18, S. 157]. Ein bekanntes Beispiel für ein Webcrawler ist der Googlebot, der Webseiten für die Suchmaschine Google indexiert [Gooe].

Ein wichtiges Konzept, um die Interaktion von Webcrawlern mit einer Webseite zu definieren, ist die sogenannte robots.txt Datei [NB21, S. 102], [BB18, S. 185]. Mit der robots.txt, die im Root-Verzeichnis der Webseite vorliegen kann, können bestimmte Teile einer Seite oder bestimmte Webcrawler vom Crawling-Prozess ausgeschlossen werden, wenn ein Webcrawler diese Datei berücksichtigt (nicht zwingend notwendig) [NB21, S. 102], [BB18, S. 185].

9 https://schema.org Letzter Zugriff 24.01.2022

3 STAND DER TECHNIK

In diesem Kapitel wird der Stand der Technik der verschiedenen relevanten Teilsysteme für diese Arbeit erläutert. Zuerst wird untersucht, welche Möglichkeiten es gibt, um Textdaten von Webseiten zu erhalten. Darauf folgend werden Möglichkeiten zur Fragengenerierung beschrieben. Zum Schluss werden Möglichkeiten zur Paraphrasierung behandelt. Alle relevanten Arbeiten werden im Bezug auf die Datenerweiterung für Chatbots betrachtet.

Zur Auswahl relevanter Arbeiten und Softwareprodukte wurden verschiedene Literaturdatenbanken für wissenschaftliche Arbeiten der Informatik mit Schlagwörtern durchsucht. Außerdem wurde die Webseite Github[1] durchsucht, da dort die meisten Open-Source-Lösungen erwartet werden. Folgende Literaturdatenbanken wurden durchsucht (Letzter Aufruf 29.06.2022):

1. https://ieeexplore.ieee.org/Xplore/home.jsp
2. https://dl.acm.org/
3. https://link.springer.com/
4. https://aclanthology.org/
5. https://www.semanticscholar.org/ (Metasuchmaschine)

Als Zeitraum der Literaturveröffentlichung wurden die letzten drei Jahre (ab 2019) festgelegt, um möglichst aktuelle Arbeiten und Ansätze zu finden. Ausgehend von diesen Arbeiten wurden auch referenzierte Arbeiten berücksichtigt, die vor dem festgelegten Zeitraum liegen, wenn diese relevant erschienen.

Die Schlagwörter, mit welchen nach relevanten Arbeiten gesucht wurde, werden in jedem Kapitel erwähnt. Das Gleiche gilt dafür, wenn darüber hinaus Quellen verwendet wurden, beispielsweise für kommerzielle Produkte im Chatbotbereich, die eines der drei genannten Teilprobleme bereits lösen.

3.1 STRUKTURIERUNG VON WEBTEXTEN FÜR CHATBOTS

Wie in Kapitel 2.5 beschrieben, werden zur automatischen Informationsextraktion von Webseiten mit größeren Informationsmengen Webcrawler benötigt. Im Rahmen dieser Arbeit sind relevante Informationen der Text einer Webseite bzw. Frage-Antwort-Paare von FAQ-Webseiten. Bilder-, Video- und Audiodaten, sowie andere Datenarten werden nicht berücksichtigt.

In den genannten Literaturdatenbanken (Kapitel 3) wurden die Begriffe „Web scraping“, „Web crawling“, „Web text extraction“, „Web information

1 https://github.com Letzer Aufruf 09.05.2022

extraction“ und „FAQ extraction“ gesucht und Arbeiten ausgewählt, die als relevant für die Aufgabe der Textextraktion für ein Chatbotsystem betrachtet wurden.

Zuerst werden Softwareprodukte oder Arbeiten zur generischen Textextraktion von Webseiten betrachtet. Das dient dem Zweck, Textblöcke zu extrahieren. Diese Textblöcke können anschließend als Antworten verwendet werden oder zur Generierung von Beispielen in Form von Fragen für einen Chatbot. Ein Textblock bildet dann mit einer oder mehreren generierten Fragen ein Frage-Antwort-Paar.

Zusätzlich zum Extrahieren von Textblöcken werden Ansätze betrachtet, die Frage-Antwort-Paare von FAQ-Webseiten extrahieren können, da Frage-Antwort-Paare bei den meisten Chatbotplattform direkt, d.h. ohne zusätzliche Textgenerierung eingebunden werden können.

Der Text einer Webseite ist in verschiedenen HTML-Tags vorhanden und somit semi-strukturiert. Beispielsweise steht das HTML-Tag p für einen Absatz (engl. paragraph) [BB18]. Der Text einer Webseite kann bei einer Extraktion demnach anhand der HTML-Tags segmentiert werden.

Bibliotheken, die ein solches Vorgehen umsetzen, sind JusText [Jus] oder Trafilatura [Bar21]. Beide Programmibliotheken sind in der Python Programmiersprache verfügbar und eignen sich deshalb im Rahmen des SMEBT Projektes zur Verwendung. JusText verarbeitet Webseiten nach Heuristiken, die in der Arbeit von Pomikálek vorgestellt werden [Pom11, S. 29].

Mit JusText wird Text anhand ausgewählter HTML-Tags der Webseite in verschiedene Textblöcke unterteilt [Pom11, S. 29]. Diese Textblöcke werden nach Heuristiken klassifiziert, dabei wird zwischen inhaltlich guten und schlechten Textblöcken unterschieden [Pom11, S. 29]:

1. Kurze Textblöcke, die einen Link enthalten, sind meist ohne Inhalt
2. Textblöcke, die viele Links enthalten, sind meist ohne Inhalt
3. Lange Textblöcke, die Stoppwörter enthalten, sind meist mit Inhalt, wohingegen lange Textblöcke ohne Stoppwörter im Text ohne Inhalt sind
4. Textblöcke mit und ohne Inhalt bilden Cluster, wobei die umliegenden Textblöcke ggf. zur gleichen Klasse zugeordnet werden können

Einen ähnlichen Ansatz zur generischen Textextraktion von Webseiten verfolgt die Bibliothek Trafilatura [Bar21]. Dabei werden durch XPath-Ausdrücke Textblöcke anhand deren HTML-Elemente oder HTML-Attribute ausgewählt oder verworfen [Bar21, S. 125]. Die ausgewählten Textblöcke werden aus dem HTML-Baum anhand von Textlänge, Linkdichte usw. extrahiert. Falls es Probleme bei der Textextraktion gibt, werden weitere Algorithmen wie JusText genutzt und der längste Text wird zurückgegeben [Bar21, S. 126].

Ein weiterer Ansatz wird in der Arbeit von Katongo, Litt und Jackson in Form einer Software namens *Wildcard* vorgestellt, die es Nutzern ermöglicht

durch UI Interaktion mit einer Webseite einen Webscraper zu erstellen, ohne Programmcode [KLJ21]. Die Daten werden direkt in einer Tabelle dargestellt, die der Nutzer manipulieren kann [KLJ21]. Das Vorgehen, Webseiten durch ein UI bereitzustellen und Extraktionsregeln aus den Nutzereingaben abzuleiten, wird auch *Wrapper Induction* genannt [KWD97; KLJ21]. Wrapper bedeutet hier, dass ein Programm für eine Webseite erstellt wird, um aus semi-strukturiertem HTML strukturierte Daten zu erhalten. Es hat also die gleiche Funktionalität wie ein Webcrawler, nur der Erstellungsprozess unterscheidet sich [KWD97; KLJ21].

Die Idee der *Wrapper Induction* existiert seit 1997 [KWD97] und ist eine Lösung zur Informationsextraktion für Endnutzer ohne Programmierkenntnisse. Allerdings ist unklar wie gut *Wrapper Induction* bei der Nutzbarkeit für Endnutzer abschneidet und wie aufwändig die Implementierung einer solchen Software ist.

Ein Chatbot hat als Datengrundlage die in Kapitel 2.4 genannten Konzepte: Intentionen mit Trainingsbeispielen (Fragen oder Aussagen), Entitäten und Konversationen. FAQ-Webseiten bieten eine passende Datengrundlage, da jede Frage zu einer Intention zugeordnet werden kann und Antworten in Absatzform gegeben sind. Deshalb können Antworten der FAQ-Seite als Antworten des Chatbots übernommen werden.

In der Arbeit von Lin u. a. wird der Ansatz FreeDOM vorgestellt [Lin+20]. Mit FreeDOM soll es möglich sein durch wenige annotierte Webseiten strukturierte Daten zu erhalten [Lin+20]. Die Autoren nutzen dazu den DOM-Baum und verarbeiten diesen mit einem neuralen Netzwerk, um strukturierte Daten von ähnlichen Seiten zu extrahieren [Lin+20]. Zur Evaluation wird der *Structured Web Data Extraction* Datensatz genutzt [Hao+11]. Lin u. a. erreichen hohe F1-Werte (>80 %) für die Umwandlung der Webseiten in strukturierte Daten mit ihrer Methode [Lin+20].

Der FreeDOM Ansatz ist für die strukturierte Datenextraktion interessant, da nur wenig Trainingsdaten benötigt werden. Allerdings ist das Modell dieser Arbeit nicht veröffentlicht und müsste mit den Informationen aus der Arbeit nachimplementiert werden, was vermutlich arbeitsaufwändig ist. Trotzdem ist dieser Ansatz hinsichtlich der Extraktion von Frage-Antwort-Paaren bzw. der Extraktion von Textblöcken, die durch einen Chatbot abgefragt werden können, interessant.

In der Arbeit von Jijkoun und Rijke werden FAQ-Webseiten genutzt, um Frage-Antwort-Paare zu extrahieren [JR05]. Dabei wurde jeweils eine heuristische und eine ML-Methode verwendet [JR05]. Bei der heuristischen Methode wird die HTML-Struktur verwendet, um den enthaltenen Text in Blöcke zu unterteilen [JR05, Kap. 4.1]. Fragen werden daran erkannt, dass sie unter 200 Zeichen enthalten, keinen Link und ein Fragezeichen oder ein großgeschriebenes Fragewort enthalten [JR05, Kap. 4.1]. Die ML-Methode nutzt zusätzlich zu den Heuristiken die Annahme, dass Fragen das gleiche Styling haben und dieses Styling auf der Webseite konsistent ist [JR05, Kap. 4.1].

Beispielsweise können die Fragen Überschriften sein [JR05, Kap. 4.1]. Der K-Nearest-Neighbors Klassifikationsalgorithmus wird genutzt, um Fragen und Antworten zu erkennen [JR05, Kap. 4.1]. Der Vergleich der Methoden zeigt, dass bei beiden Methoden in einer Stichprobe von 109 Webseiten eine Präzision und Recall von über 92 % erreicht wurde [JR05, Kap. 4.2]. Die heuristische Methode hat einen höheren Präzisionswert (94 %), während die ML-Methode einen besseren Recallwert von 94 % vorweist.

Die Arbeit von Jijkoun und Rijke zeigt, dass eine strukturierte Extraktion von Frage-Antwort-Paaren gut möglich ist, entweder durch Heuristiken oder ML-Methoden. Allerdings ist die Qualität der Evaluation schwierig zu beurteilen, da der Evaluationskorpus und der Programmcode nicht zur Verfügung steht und das Vorgehen deshalb nicht komplett nachvollziehbar ist.

In der Arbeit von Dallmeier wird Dokument-Layout-Analyse genutzt, um Daten von Webseiten bzw. genauer Internetforen zu extrahieren. Der Autor schlägt vor, Methoden der Computer Vision zu nutzen, um Daten zu strukturieren [Dal21]. Bei der Computer Vision werden visuelle Informationen z. B. Screenshots der Webseite zur Verarbeitung genutzt [Dal21]. Bei diesem Ansatz werden Textblöcke anhand ihrer visuellen Darstellung annotiert in Kategorien wie Inhalt oder Titel [Dal21]. Dallmeier nennt als Vorteil, dass diese Methode unabhängig von HTML und CSS ist [Dal21]. Als Nachteil wird die Performanz genannt, da HTML-Dateien schneller verarbeitet werden können als Screenshots von Webseiten zu erstellen und diese zu verarbeiten [Dal21]. Außerdem ist das Erstellen von Trainingsdaten ein weiterer Nachteil, da es arbeitsaufwändig ist [Dal21].

Dallmeier nennt in seiner Arbeit ein Vorgehen, das auch bei anderen Arbeiten bzw. Softwareansätzen verwendet wird, die Nutzung von Computer Vision. Dieser Ansatz scheint vielversprechend im Hinblick auf die Qualität der Textextraktion, da Texte zusätzlich nach ihrer Bedeutung in einem Dokument annotiert werden und so unpassende Texte gefiltert werden können.

Microsoft bietet in Verbindung mit dem Chatbotsystem *QnAMaker* eine Frage-Antwort-Extraktion an, um Inhalt zu erkennen und in Fragen und Antworten zu unterteilen [Agr+20]. Der Extraktionsprozess ist für einzelne Dokumente nutzbar, wie eine Webseite oder ein PDF-Dokument [Agr+20]. Der Ansatz, um diese Dokumente in eine Frage-Antwort-Paar Form umzuwandeln, wird umschrieben, allerdings sind die verwendeten Algorithmen nur im Ansatz umschrieben und der Programmcode ist proprietär [Agr+20, Kap. 2.3]. *QnAMaker* nutzt den *Azure Search Index*, um Frage-Antwort-Paare zu speichern und die Antworten für gestellte Fragen zu finden [Agr+20]. Eine Nutzeranfrage wird an die *QnAMaker*-Bot-API gesendet und anschließend vorverarbeitet [Agr+20]. Als Vorverarbeitungsschritte werden Spracherkennung, Lemmatisierung, Rechtschreibprüfung, Worttrennung und Stoppwortentfernung genannt, um eine Nutzeranfrage zu normalisieren [Agr+20]. Die verarbeitete Nutzeranfrage wird an den *Azure Search Index* gesendet [Agr+20]. Es werden weniger als hundert Dokumente aus der Abfrage für

ein erneutes Ranking der Dokumente verwendet [Agr+20]. Zum erneuten Ranking werden Eigenschaften der Dokumente von WordNet [Mil95], TF-IDF und Convolutional Deep Structured Semantic Models (CDSSM) genutzt [Agr+20].

Die Frage-Antwort-Extraktion des *QnAMaker* von Webseiten bzw. anderen Dokumenten wird in der Arbeit von Agrawal u. a. kurz beschreiben, allerdings findet in der Arbeit keine Evaluation des Extraktionsprozesses statt, weshalb die Qualität dieses Vorgehens nicht einschätzbar ist.

Das Unternehmen IBM bietet den *Corpus Conversion Service*, welcher in erster Linie für Dateien im Portable Document Format (PDF) entwickelt wurde, aber auch für einzelne Webseiten verwendet werden kann [Sta+18]. Das Vorgehen bei der Informationsextraktion ist dabei in fünf Schritte eingeteilt [Sta+18]:

1. Parsen des Dokuments in ein internes Format
2. Annotation des Dokuments mit semantischen Labels durch den Nutzer
3. Training eines ML-Modells aus dem Dokument und den Annotationen des Nutzers
4. Anwendung des ML-Modells für weitere Dokumente
5. Umwandlung der Dokumente in ein strukturiertes Format

Falls bereits ein passendes vortrainiertes Modell vorhanden ist, müssen nur die Schritte 1, 4 und 5 ausgeführt werden [Sta+18]. Der erste Schritt beim Parsen des Dokuments ist das Unterteilen dieses in minimal umgebende Rechtecke, die Textzeilen oder Textblöcke enthalten [Sta+18]. Jeder identifizierte Textblock wird von einem Nutzer oder einem ML-Modell gelabelt, mögliche Label sind Überschrift, Autoren, Haupttext usw. [Sta+18]. Das Annotieren ist für den Nutzer durch ein proprietäres Tool möglich [Sta+18]. Es werden zwei verschiedene Arten von ML-Modellen verwendet. Einerseits werden Modelle verwendet die unabhängig vom Layout des Dokuments Objekte wie Tabellen, Bilder usw. erkennen können, andererseits können für ein Dokumentlayout spezifische Modelle erstellt werden [Sta+18]. Zur allgemeinen Objekterkennung in Dokumenten wird die YOLOv2-Architektur genutzt [RF16; Sta+18]. Für das Erlernen von spezifischen Dokumentlayouts wird der ML-Algorithmus Random Forest verwendet [Bre01; Sta+18]. Mithilfe dieser beiden Techniken erreichen Staar u. a. ca. 99 % für Präzision und Recall bei der Klassifizierung von Textblöcken in 5000 PDF-Dateien [Sta+18]. Im Hinblick auf das Strukturieren von Informationen für einen Chatbot können Textblöcke als Fragen oder Antworten klassifiziert werden.

Die Arbeit von Staar u. a. zeigt, dass die Nutzung von Computer Vision und Objekterkennung gute Ergebnisse für PDF-Dateien erzielen. Eine Evaluation für Webseiten gibt es in der Arbeit nicht.

Der *Corpus Conversion Service* kann unter dem Namen *Smart Document Understanding* in Verbindung mit der IBM Chatbot Plattform *Watson Assistant* genutzt werden [Sma]. Dazu muss ein sogenannter *Search Skill* angelegt werden, der extrahierte Dokumente in einer Datenbank durchsuchbar macht [Sea]. Die Wissensbasis der Dokumente ist der *Watson Discovery* Service, der Suchanfragen in natürlicher Sprache verarbeiten kann [Sea]. Zusätzlich zu dieser Möglichkeit bietet *Watson Discovery* beispielsweise auch einen Webcrawler an, welcher Text oder FAQs von Webseiten extrahieren kann [Tim]. Implementierungsdetails über den Webcrawler und dessen Frage-Antwort-Extraktion sind nicht veröffentlicht.

Das Unternehmen Google bietet als Teil ihrer Chatbotplattform *Dialogflow* auch eine Frage-Antwort-Extraktion von Webseiten an, um eine Wissensbasis für einen Chatbot aufzubauen [Goof].

In der Arbeit von Rashid, Jamour und Hristidis wird ein Framework zum Extrahieren von FAQs aus dem Web vorgeschlagen [RJH21]. Zur Frage-Antwort-Extraktion wird ein Algorithmus verwendet, der Fragen und Antworten anhand häufig genutzter HTML-Tags extrahiert [RJH21, S. 1520]. Der Algorithmus konnte auf 71 von 100 getesteten Webseiten Frage-Antwort-Paare erfolgreich extrahieren [RJH21, S. 1523]. Die Autoren nutzen außerdem noch weitere Schritte, um Frage-Antwort-Paare mit hoher Qualität zu erhalten [RJH21, S. 1519-1521]:

1. FAQ-Webseiten-Erkennung
2. Unterscheidung in generelle oder spezifische Fragen
3. Unterscheidung in eigenständige oder unvollständige Fragen
4. Erkennung von doppelten Fragen

Die Arbeit von Rashid, Jamour und Hristidis zeigt, dass das Extrahieren von strukturierten Frage-Antwort-Paaren aus dem Web in mehreren Schritten in einer Pipeline ausgeführt werden kann. Die Autoren haben allerdings für die Frage-Antwort-Extraktion keinen Datensatz veröffentlicht, für andere Aufgaben sind Datensätze veröffentlicht [RJH21].

Zusammenfassend kann beobachtet werden, dass Produkte wie Google Dialogflow, IBM Watson und Microsoft QnAMaker Frage-Antwort-Extraktion von Webseiten für ihre Chatbotplattformen anbieten [Tim; Agr+20; Goof]. Bei IBM Watson und Microsoft QnAMaker werden die Webseiten als Dokumente in Datenbanken gespeichert mittels Softwarelösungen wie Microsoft Azure Search Index oder IBM Watson Discovery [Sea; Agr+20].

Nach der Unterteilung von Webcrawling Methoden von Nigam und Biswas sind die am häufigsten genutzten Methoden ML-Methoden, die entweder die DOM-Baumstruktur [Lin+20; JR05] nutzen oder ML-Methoden die Computer Vision verwenden [Agr+20; Sta+18; Sma; Dal21; NB21]. Die Arbeiten und Produkte, die ML-Methoden nutzen und in diesem Kapitel betrachtet wurden, sind proprietär bzw. ohne Programmcode veröffentlicht.

Für die Extraktion von inhaltlich relevantem Text oder Frage-Antwort-Paaren werden zudem Heuristiken genutzt (DOM-Baum Parsing). Ein Vorteil dieser Methoden ist, dass die Extraktion vollautomatisierbar ist [RJH21; JR05; Jus; Pom11; Bar21]. Im Vergleich zu ML-Methoden müssen keine Datensätze zum Trainieren von Modellen erstellt werden.

Außerdem gibt es Methoden für Endnutzer ohne Programmcode Webcrawler zu erstellen, die sogenannte *Wrapper Induction*, die auch zur DOM-Baum Parsing Methode zugeordnet werden kann [KWD97; KLJ21].

Die Nachfolgende Tabelle 3.1 fasst die verschiedenen Ansätze zusammen und bewertet sie hinsichtlich der Wiederverwendbarkeit für das SMEBT Projekt zur Extraktion von Informationen aus dem Web für Chatbots:

Zusammenfassung	Webcrawling Kategorie	Software Verfügbarkeit	Referenzen	Verwendbarkeit
Generische Textextraktion von Webseiten	DOM Tree Parsing (Heuristiken)	quelloffen	[Jus; Pom11]	Verwendbar in Verbindung mit einem Webcrawler.
Generische Textextraktion von Webseiten	DOM Tree Parsing (Heuristiken)	quelloffen	[Bar21]	Verwendbar in Verbindung mit einem Webcrawler.
Wrapper Induction durch Nutzereingaben zum Web Scraping	DOM Tree Parsing (Heuristiken)	quelloffen	[KLJ21]	Webbrowser-Plugin vorhanden
Mögliche Frage-Antwort-Extraktion durch Annotieren von Webseiten	ML-basiert (DOM)	Nur Methode	[Lin+20]	Keine Implementierung vorhanden
Frage-Antwort-Extraktion von Webseiten und Beantwortung von Nutzerfragen mithilfe der extrahierten Frage-Antwort-Paare	DOM Tree Parsing (Heuristiken und ML-basiert)	Methode wird beschrieben	[JR05]	Keine Implementierung vorhanden
Strukturierte Datenextraktion von Webforen mithilfe von Computer Vision	ML-basiert	Methode beschrieben	[Dal21]	Keine Implementierung vorhanden
QnAMaker bietet eine generische Frage-Antwort-Extraktion für Dokumente wie Webseiten	Algorithmen und ML-basiert (vermutlich Computer Vision basiert, keine Details veröffentlicht)	proprietär	[Agr+20]	Implementierung proprietär
Generische Textextraktion von Dokumenten (auch Webseiten), Unterteilung der Texte in Kategorien wie Überschrift, Haupttext usw. anhand visueller Darstellung	ML-basiert (Computer Vision)	proprietär	[Sta+18]	Implementierung proprietär
Frage-Antwort-Extraktion mithilfe eines Webcrawlers	Unbekannt	proprietär	[Sma; Sea; Tim]	Implementierung proprietär
Frage-Antwort-Extraktion für Webseiten	Unbekannt	proprietär	[Goof]	Implementierung proprietär
FAQ Extraktions Framework mit Erkennung von FAQ-Webseiten, Frage-Antwort-Extraktion und weiteren Komponenten	DOM Tree Parsing (Heuristiken)	Methode wird beschrieben, keine Software	[RJH21]	Implementierung proprietär

Tabelle 3.1: Zusammenfassung der State-Of-The-Art Arbeiten zur Webinformationsextraktion.

3.2 FRAGENGENERIERUNG

Bei der Fragengenerierung geht es darum aus einer Eingabe relevante Fragen zu erzeugen. Im Fall dieser Arbeit wird ausschließlich Text in natürlicher Sprache als Eingabe betrachtet. Andere Möglichkeiten für Eingaben wären beispielsweise Datenbanken oder Bilder [Zha+21, S. 2].

Als Suchbegriffe wurden „Question generation“ und „Question generation german“ in den angegebenen Literaturdatenbanken verwendet. Es konnten neuronale und regelbasierte Ansätze gefunden werden, allerdings nutzen die meisten gefundenen Ansätze der letzten Jahre neuronale Methoden.

Die Ansätze unterscheiden sich auch bei der Verwendung der sprachlichen Einheit, es werden entweder einzelne deskriptive Sätze oder Absätze als Eingabe verwendet.

Die Arbeit von Zhang u. a. [Zha+21] gibt einen Überblick über das Forschungsfeld der Fragengenerierung. In dieser Arbeit werden potentielle Ansätze verfolgt, die reproduziert werden können zum Training eines eigenen neuronalen Modells in deutscher Sprache.

Eine Aufgabe, die in Verbindung mit der Fragengenerierung steht, ist die Fragenbeantwortung (engl. Question Answering, kurz QA) [Zha+21, S. 3]. Dabei wird versucht, aus einem Text und einer Frage über den Text eine passende Antwortpassage zu erhalten [Zha+21, S. 3]. Einige Ansätze kehren diesen Ansatz zur Fragengenerierung um, indem ein Text mit einer markierten Antwort zu einer Frage umgewandelt wird (engl. answer-aware question generation) [Zha+21, S. 3].

Zhang u. a. stellen bei der Aufgabe der Fragengenerierung folgende Taxonomie fest (Abbildung 3.1) [Zha+21, S. 5]: Die Taxonomie zeigt mögliche

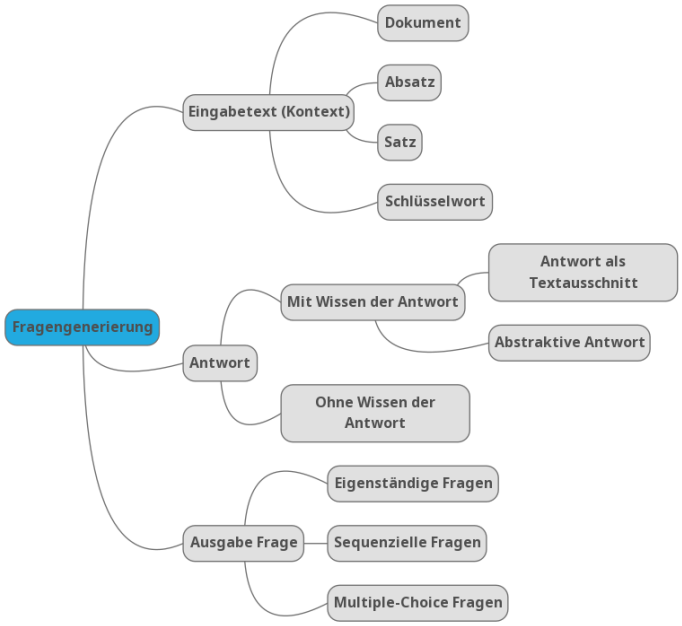

Abbildung 3.1: Taxonomie der Fragengenerierung nach Zhang u. a. [Zha+21, S. 5].

sprachliche Einheiten für Eingabetexte (auch Kontexte genannt), mögliche Arten zur Berücksichtigung gegebener Antworten und mögliche Arten von generierten Fragen [Zha+21, S. 5].

Bezüglich der Ansätze zur Fragengenerierung wird zwischen regelbasierten und neuronalen Methoden unterschieden. Ein Beispiel für eine regelbasierte Methode in englischer Sprache ist die Arbeit von Heilman und Smith. Die Autoren nutzen manuell erstellte Regeln, um deklarative Sätze in Fragen umzuwandeln [HS10]. Diese Regeln ändern unter anderem den Satzbau, z. B. durch Subjekt-Hilfsverb-Umkehrung, um eine Frage zu erzeugen [HS10]. Die Autoren nutzen zudem ein statistisches Modell, um Fragen zu klassifizieren in akzeptabel und nicht-akzeptabel [HS10]. Heilman und Smith werten die Performanz für die Fragengenerierung mithilfe von drei Korpussen aus. Die Testdatensätze die mit diesen Korpussen erstellt wurden haben zusammen eine Größe von 428 Fragen [HS10]. Die Fragen wurden von 14 englischsprachigen Studenten ausgewertet und 27.3 % der Fragen wurden als akzeptabel bewertet nach den Kriterien die in der Arbeit festgelegt wurden [HS10]. Nach dem Sortieren der Fragen durch das erstellte statistische Modell geben die Autoren außerdem eine Akzeptanz von 52.3 % für die Top 20% der Fragen an [HS10].

Für die deutsche Sprache gibt es ebenfalls einen regelbasierten Ansatz zur Fragengenerierung, in der Arbeit von Kolditz. Ähnlich dem Vorgehen von Heilman und Smith nutzt Kolditz manuell erstellte Regeln, um Sätze zu Fragen umzuwandeln [Kol15]. Kolditz nennt außerdem Probleme bei der Fragengenerierung, die speziell in der deutschen Sprache auftreten [Kol15, S. 19 - 21]. Kolditz evaluiert seinen Ansatz mit 150 generierten Fragen, die nach einer Skala evaluiert werden, dabei wurde festgestellt, dass 44 % der Fragen den höchsten Score erreichen, der bedeutet, dass eine Frage interessant und nicht zu komplex ist [Kol15].

Die regelbasierten Ansätze von Heilman und Smith und Kolditz nutzen unter anderem NLP Methoden wie POS-Tagging, Lemmatisierung usw., um aus Sätzen Fragen zu generieren [HS10; Kol15]. Im Gegensatz dazu stehen neuronale Ansätze, die im wesentlichen ohne die genannten NLP Methoden Fragen generieren können.

Aufbauend auf der Arbeit von Kolditz erstellen De Kuthy u. a. einen Datensatz mit generierten Fragen für einen Satz und eine Antwort [DK+20]. Die Autoren nutzen ein RNN mit LSTM-Zellen als Sequenz-Zu-Sequenz-Modell [DK+20]. Als Eingabe werden Satz und Antwort genutzt und beide Texte mit POS-Tags erweitert [DK+20]. Die Autoren trainierten verschiedene Modelle mit einem Trainingsdatensatz von jeweils 150.000 und 400.000 Beispielen [DK+20]. Dabei wurde der regelbasierte Ansatz mit dem Modell, das mit 400.000 Beispielen trainiert wurde, in einer manuellen Auswertung von 500 Fragen übertroffen mit einer Anzahl von 52 % grammatikalisch korrekter und beantwortbarer Fragen [DK+20]. Der generierte Datensatz von De Kuthy u. a. wurde nicht veröffentlicht.

Fragengenerierung für die englische Sprache nutzt in den letzten Jahren vortrainierte Modelle die eine Transformer-Architektur verwenden. Dabei

wird Transferlernen genutzt, also das Nachtrainieren (engl. Finetuning) bereits vortrainierter Sprachmodelle, für eine spezifische Aufgabe.

In der Arbeit von Lopez u. a. wird das GPT-2 Sprachmodell [Rad+19] genutzt für ein Finetuning für die Aufgabe der Fragegenerierung und den Datensatz *Stanford Question Answering Dataset (SQuAD)* [Raj+16]. Der SQuAD-Datensatz wird genauer in Kapitel 3.2 beschrieben. Die Absätze und Fragen werden zum Finetuning genutzt [Lop+20]. Lopez u. a. nutzen verschiedene Formate, um den Trainingsdatensatz für die Fragengenerierung zu transformieren: Es werden drei verschiedene Trennzeichen genutzt, um den Kontext von Fragen zu trennen (Bezeichnung: String) [Lop+20]:

- ARTIFICIAL: [SEP]
- NATURAL-QUESTION: Question:
- NATURAL-NUMBER: 1.

Dazu werden die vorhanden Fragen zu einem Kontext in zwei verschiedenen Formaten angehängt, wie in Abbildung 3.2 zu sehen [Lop+20]. Entweder werden alle Fragen mit Trennzeichen an den Kontext angehängt oder es wird pro Frage ein Kontext-Frage-Paar erstellt.

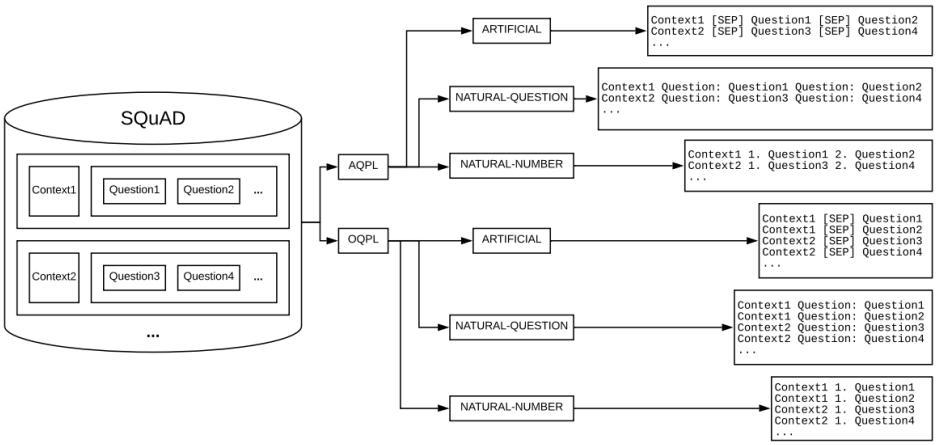

Abbildung 3.2: Datentransformation des SQuAD Formats [Raj+16] zur Fragengenerierung von GPT-2 nach Lopez u. a. [Lop+20]

Es wird außerdem betrachtet, wie sich die Markierung der Antwort in einem Kontext (engl. answer-awareness) auf die Performanz des Modells auswirkt in den Metriken BLEU-1 bis BLEU-4, Meteor und ROUGE-L [Lop+20]. Allerdings zeigt das Experiment von Lopez u. a. das dieses Vorgehen die Performanz der genannten Metriken nicht verbessert [Lop+20].

In der Arbeit von Chan und Fan wird ebenfalls ein vortrainiertes Transformer-basiertes Sprachmodell verwendet (BERT), um Fragen zu generieren [CF19]. Die Autoren nutzen verschiedene Aufteilungen des SQuAD-Datensatzes [Raj+16] zum Finetuning [CF19]. Außerdem wird das Modell angepasst und ein weiteres Ausgabelayer angehängt [CF19]. Es werden drei verschiedene

Ansätze mit dem vortrainierten BERT-Modell getestet, die sich anhand der Eingabesequenz unterscheiden [CF19]. In allen drei Ansätzen wird die Markierung der Antwortsequenz genutzt [CF19]. Der erste Ansatz BERT-QG nutzt Kontext und Antwort als Eingabe, um die Frage zu generieren [CF19]. Da dieses Vorgehen allerdings in schlechten Werten für automatisch gemessene Metriken BLEU-1 bis BLEU-4 sowie METEOR und ROGUE-L resultiert, wurden zwei rekurrente Vorgehen genutzt BERT-SQG und BERT-HLSQG [CF19]. BERT-SQG nutzt eine sequentielle Fragengenerierung, wobei die Frage Wort für Wort generiert wird und die Generierung gestoppt wird, wenn das BERT-Modell einen speziellen Token ([SEP]) vorhersagt [CF19].

Der Ansatz BERT-HLSQG ändert wiederum die Formatierung der Trainingsdaten, wobei die Antwort nun im Kontext mit speziellen Token ([HL]) markiert wird, anstatt zusätzlich nach dem Kontext [CF19]. Dieses Vorgehen führt zu minimalen Verbesserungen bei den automatisch gemessenen Metriken [CF19].

SQG	HLSQG
The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] [MASK]	The Super Bowl 50 was played at [HL]Santa Clara, California.[HL] [SEP] [MASK]

Tabelle 3.2: Unterschied BERT-SQG und BERT-HLSQG.

In der Arbeit von Klein und Nabi wird GTP-2 [Rad+19] in Verbindung mit BERT [Dev+18] genutzt, um Fragen in englischer Sprache zu generieren, mithilfe von markierten Antworten [KN19]. Die Autoren nutzen den SQuAD-Datensatz [Raj+16], um das GPT-2 und BERT Modell zu trainieren [KN19]. Der Trainingsprozess wird dadurch abgewandelt, dass ein BERT Modell für die Aufgabe der Fragenbeantwortung zusätzlich verwendet wird, um die Trainingsergebnisse der Fragengenerierung zu verbessern [KN19].

Die Arbeit von Alberti u. a. nutzt ebenfalls Fragengenerierung, um Frage-Antwort-Datensätze zu erstellen. Dafür werden Modelle für die Aufgabe der Fragegenerierung und Fragenbeantwortung verwendet [Alb+19]. Als Sprachmodell werden vortrainierte BERT Modelle [Dev+18] genutzt und als Datensätze SQuAD2 [RJL18] und Natural Questions (NQ) [Kwi+19] [Alb+19]. Der Ansatz zur Fragengenerierung nutzt ebenfalls die Markierung der Antwort im Kontext zur Generierung einer Frage [Alb+19].

Um FAQs aus einer Wissensbasis abzufragen, nutzen Mass u. a. ebenfalls ein GPT-2 Modell, um Fragen aus einer Antwort zu generieren [Mas+20]. Die Autoren nutzen diese Technik, um Fragen zu paraphrasieren, indem sie dem Modell eine Antwort als Eingabe übergeben [Mas+20, S. 809]. Genauere Angaben zum genutzten Datensatz und Vorgehen des Modells fehlen allerdings, da der Fokus auf dem Abfragen von Frage-Antwort-Paaren durch zwei BERT Modelle liegt [Mas+20, S. 809].

MixQG ist eine Arbeit von den Autoren Murakhovs'ka u. a., die sich ebenfalls mit der neuronalen Fragengenerierung für die englische Sprache beschäftigen [Mur+21]. In der Arbeit werden insgesamt neun Frage-Antwort-

Datensätze genutzt, um Modelle zu trainieren [Mur+21]. Dabei werden die Datensätze in verschiedene Antworttypen unterteilt [Mur+21]:

EXTRAKTIV Die Antwort ist eine Teilzeichenfolge aus dem Kontext

ABSTRAKT Die Antwort ist in freier Form geschrieben und nicht notwendigerweise im Kontext

JA-NEIN Die Antwort ist entweder mit ja oder nein zu beantwortet und abhängig von Frage und Kontext

MULTIPLE-CHOICE Es sind mehrere Antworten gegeben, aus welchen eine Antwort ausgewählt wird abhängig von einer gegebenen Frage. Die Antwort muss nicht im Kontext vorhanden sein

Es wird unter anderem der SQuAD-Datensatz genutzt, der nach der Antwortklassifizierung von Murakhovs'ka u. a. extraktive Antworten enthält [Raj+16; Mur+21]. Des Weiteren werden die vortrainierten Sprachmodelle T5 und BART genutzt, um Fragen zu generieren [Mur+21]. Die Autoren evaluieren ihre Ergebnisse mit den automatischen Metriken BLEU, ROUGE, METEOR und BERTScore [Mur+21]. Der Programmcode dieser Arbeit wurde auf Github veröffentlicht[2].

In der Arbeit von Akyon u. a. wird ein multilinguales T5 (mT5) Transformer-Modell trainiert, um die Aufgaben der Fragengenerierung mit Wissen der Antwort, Fragenbeantwortung und Antwortextraktion zu bewältigen [Aky+21]. Durch das Trainieren von mehreren Aufgaben ist auch eine Fragengenerierung ohne Wissen der Antwort möglich [Aky+21]. Im Gegensatz zu vorherigen Arbeiten, die diese Aufgaben für die englische Sprache umsetzen, wird in dieser Arbeit für die türkische Sprache mithilfe des Frage-Antwort-Datensatzes THQuAD, ein Modell trainiert [Aky+21; Soy+21]. Der Programmcode dieser Arbeit wurde ebenfalls auf Github veröffentlicht[3].

In der Arbeit von Grover u. a. wird ein T5-Modell mit dem SQuAD [Raj+16] Datensatz trainiert, dabei ist der Ansatz ähnlich zu dem von Lopez u. a. die Fragen nur aus einem Absatz zu generieren, ohne eine Antwort zu markieren [Gro+21]. Das Vorgehen von Grover u. a. ist auf Github verfügbar, allerdings von einem anderen Autor [Pat20].

Tabelle 3.3 fasst die relevanten Arbeiten zusammen.

2 https://github.com/salesforce/QGen/tree/main/MixQG Letzter Aufruf 22.06.2022
3 https://github.com/obss/turkish-question-generation Letzter Aufruf 22.06.2022

Zusammenfassung	Sprache	Modell	Fragen-generierungs-methode	Datensatz	Methode	Software Verfügbarkeit	Wiss. Arbeit	Evaluation
Deklarativsatz in Frage umwandeln	Englisch	-	Syntax Transformation	-	regelbasiert	Nur Methode	[HS10]	Manuell
Deklarativsatz in Frage umwandeln	Deutsch	-	Syntax Transformation	-	regelbasiert	Nur Methode	[Kol15]	Manuell
Fragengenerierung	Englisch	GPT-2	Mit und ohne Antwort	SQuAD	neuronal	Nur Methode	[Lop+20]	BLEU 1-4, ROUGE-L, METEOR
Fragengenerierung mit BERT basiertem Modell	Englisch	BERT	Mit Antwort	SQuAD	neuronal	Nur Methode	[CF19]	BLEU 1-4, ROUGE-L, METEOR
Fragengenerierung und Beantwortung	Englisch	GPT-2, BERT	Mit Antwort	SQuAD	neuronal / ML	Nur Methode	[KN19]	BLEU 1-4, ROUGE
Frage-Antwort-Paar Generierung mit BERT, um Fragenbeantwortung zu verbessern	Englisch	BERT	Mit Antwort	SQuAD, Natural Questions	neuronal	Nur Methode	[Alb+19]	Nur QA wird evaluiert
Fragengenerierung um FAQ-Suche zu verbessern	Englisch	GPT-2	Ohne Antwort	-	neuronal	Nur Methode	[Mas+20]	Nur QA wird evaluiert
Fragengenerierung und Beantwortung türkischer Texte mit Text-zu-Text Transformer	Türkisch	mT5	Mit und ohne Antwort	TQuADv1, TQuADv2, XQuAD	neuronal	Github	[Aky+21]	BLEU 1-2, ROUGE-L
Fragengenerierung mit verschiedenen Antworttypen	Englisch	T5, BART	Mit Antwort	9 verschiedene inklusive SQuAD	neuronal	Github	[Mur+21]	BLEU, ROUGE-1, -2, -L, METEOR, BERTScore
Fragengenerierung ohne Antwort	Englisch	T5	Ohne Antwort	SQuAD	neuronal	Nur Methode	[Gro+21]	Beispiele
Fragengenerierung	Englisch	T5	Mit und ohne Antwort	SQuAD	neuronal	Github	[Pat20]	BLEU-4, ROUGE-L, METEOR

Tabelle 3.3: Zusammenfassung verwandte Arbeiten zur Fragengenerierung.

Im Folgenden werden Datensätze beschrieben, die zur Fragengenerierung aus einem Text verwendet werden können. Dabei sind alle genannten Datensätze in erster Linie für die Aufgabe der Fragenbeantwortung bzw. des Leseverständnisses (engl. reading comprehension) erstellt worden. Das liegt daran, dass viel Forschung in diesem Bereich getätigt wird und die Performanz von Sprachmodellen unter anderem auch bei dieser Aufgabe gemessen wird (Ein Beispiel ist SQuAD als Benchmark für das T5 Sprachmodell [Raf+19, S. 7]).

Die Datensätze können in ein Format umgewandelt werden, das sich zum Training eines vortrainierten Sprachmodells zur Fragengenerierung eignet.

Zuerst wird der englischsprachige SQuAD-Datensatz beschrieben, der von bereits existierenden Fragengenerierungsmodellen in englischer Sprache verwendet wird. Der SQuAD-Datensatz ist außerdem schematisches Vorbild für die nachfolgenden Frage-Antwort-Datensätze in deutscher Sprache.

SQUAD Die erste Version (1.0 bzw. 1.1) des Stanford Question Answering Datensatzes (SQuAD) wurde im Oktober 2016 veröffentlicht und ist ein oft genutzter Datensatz zum Trainieren von Modellen für die Aufgabe der Fragenbeantwortung in englischer Sprache [Raj+16]. Der Datensatz wurde von Crowdworkern erstellt und enthält 107.785 Frage-Antwort-Paare für 536 Wikipedia-Artikel [Raj+16].

Eine neuere SQuAD Version (2.0) aus dem Jahre 2018 beinhaltet zusätzlich ca. 50.000 Fragen, die nicht durch eine gegebene Textpassage beantwortet werden können [RJL18]. Diese Fragen können allerdings nicht für eine Fragengenerierung verwendet werden, weil angenommen wird, dass ein Modell immer Bezug auf einen geeigneten Textabschnitt in der Passage nehmen muss, um eine Frage zu generieren.

Die SQuAD Daten sind im JSON-Format verfügbar und sind in folgendem Schema vorhanden:

```
{
    "data": [{
        "title": "Titel",
        "paragraphs": [{
            "context": "Text",
            "qas": [{
                "question": "Frage",
                "id": 1,
                "answers": [{
                    "text": "Text",
                    "answer_start": 1,
                }]
            }]
        }]
    }]
}
```

Listing 1: SQuAD Datenschema

XQUAD Der XQuAD-Datensatz ist eine Teilmenge der SQuAD v1.1 Testdaten mit 240 Absätzen und 1.190 Frage-Antwort-Paaren in zehn Sprachen, unter anderem auch Deutsch [ARY19]. Die Frage-Antwort-Paare wurden von professionellen Übersetzern angefertigt [ARY19].

GERMANQUAD Der Datensatz GermanQuAD wurde im April 2021 von der Firma deepset GmbH veröffentlicht [MRP21]. Der Datensatz besteht aus 13.722 Frage-Antwort-Paaren die im Format des SQuAD-Datensatzes (siehe Listing 1) vorliegen [MRP21]. Der Datensatz wurde von Crowdworkern mit Fragen und Antworten annotiert und besteht aus Absätzen von deutschsprachigen Wikipedia-Artikeln, die das Pendant zu Wikipedia-Artikeln im SQuAD-Datensatz sind [MRP21].

MLQA Der MLQA Datensatz ist ein multilingualer Evaluierungsdatensatz für die Aufgabe der Fragenbeantwortung [Lew+19]. Für die deutsche Sprache beinhaltet der Datensatz ca. 5.000 Frage-Antwort-Paare [Lew+19]. Die Autoren des Datensatzes haben außerdem den SQuAD-Datensatz maschinell Übersetzt[4] und diesen veröffentlicht, unter anderem in die deutsche Sprache [Lew+19].

Die Datensätze GermanQuAD, XQuAD und MLQA wurden von Menschen erstellt oder übersetzt. Zusätzlich wurde bei der Arbeit des MLQA Datensatzes ein maschinell Übersetzter SQuAD-Datensatz veröffentlicht, welcher genutzt werden kann [Lew+19]. Allerdings ist die Qualität des Da-

4 https://github.com/facebookresearch/MLQA#translate-train-and-translate-test-data Letzter Aufruf 20.06.2022

tensatzes im Gegensatz zu den anderen genannten Datensätzen vermutlich schlechter durch die maschinelle Übersetzung. Die Anzahl und Verteilung der jeweiligen Frage-Antwort-Paare wird in Tabelle 3.4 genauer aufgezeigt.

Datensatz	Aufteilung	Training	Test	Evaluation	Gesamt	Referenz
SQuAD	Training, Text	87.599 (89 %)	10.570 (11 %)	-	**98.169**	[Raj+16]
GermanQuAD	Training, Test	11.518 (84 %)	2.204 (16 %)	-	**13.722**	[MRP21]
XQuAD	Evaluation	-	-	1.190	**1.190**	[ARY19]
MLQA	Evaluation	-	512	4.517	**5.029**	[Lew+19]
Maschinell übersetztes SQuAD (MLQA)	Training, Test	80.069 (89 %)	9.927 (11 %)	-	**89.996**	[Lew+19]

Tabelle 3.4: Übersicht der Datensätze zur Fragengenerierung.

3.3 PARAPHRASIERUNG

Paraphrasierung bzw. die Generierung von Paraphrasen ist eine weitere Möglichkeit Trainingsdaten zur Intentionsklassifizierung zu erweitern. Ähnlich zur Fragengenerierung existieren Ansätze, die vortrainierte Sprachmodelle für diese Textgenerierungsaufgabe trainieren. Solche Ansätze sind hauptsächlich auf den Plattformen Github [Git] und Huggingface [Wol+20] auffindbar mit dem Suchbegriff „Paraphrase“. Zu den beliebtesten Projekten für die Paraphrasierung, gemessen an Sternen (Lesezeichen von Nutzern) des Projektes, gehören *Parrot Paraphraser* [Dam21], *Paraphrase Generator with T5* [Ali20], *Paraphrase any question with T5* [Gol20]. Die Ansätze sind alle mit neuronalen Modellen umgesetzt, wobei [Ali20; Gol20; Dam21] ein vortrainiertes T5-Modell nutzen [Raf+19].

In der Arbeit von Berro u. a. wird eine Pipeline zur automatischen Erweiterung von Chatbot Trainingsdaten vorgestellt [Ber+21]. Die Pipeline ist in zwei Schritten organisiert: Zuerst werden Paraphrasen generiert, anschließend werden mögliche Kandidaten für Paraphrasen automatisch gefiltert mithilfe von weiteren trainierten Modellen wie dem Universal Sentence Encoder (USE) [Chi+19; Ber+21]. Die Autoren nutzen in ihrer Pipeline verschiedene Techniken, um Paraphrasen zu erstellen [Ber+21]:

SCHWACH ÜBERWACHT (ENGL. WEAK SUPERVISION) Diese Techniken beschreiben Berro u. a. regelbasierte Techniken, bei welchen einzelne Wörter (Verben und Nomen) mithilfe eines Synonym-Wörterbuchs wie Wordnet [Mil95] ausgetauscht werden.

PIVOTÜBERSETZUNG Bei dieser Methode werden Sätze in eine andere Sprache übersetzt (Pivotsprache) und wieder zurückübersetzt, um Paraphrasierungen zu erhalten. Dieser Vorgang wird in anderen Arbeiten auch *Zurückübersetzung* genannt [Cha20]. Es kann eine oder mehrere Fremdsprachen als Pivotsprache genutzt werden für diese Methode.

Die Autoren schlagen auch das Nutzen von verschiedenen Übersetzungsdiensten wie DeepL[5] vor oder neuronale Übersetzungsmodellen von Huggingface [Wol+20].

TRANSFORMER MODELL Als dritte Methode schlagen die Autoren die Verwendung des T5-Modells [Raf+19] vor, zur Generierung von Paraphrasen. Dabei werden Datensätze mit Satzpaaren genutzt. In der Arbeit werden zu einem Finetuning Quora Fragenpaare [Quo] und ParaNMT [WG18] verwendet. Als Vorbild zum Training wurde die Software von [Gol20] verwendet.

In einem Blogeintrag von Chaudhary werden ähnlich zu den Methoden der Pipeline von Berro u. a. Datenerweiterungsmethoden vorgeschlagen [Cha20]. Chaudhary nennt unter anderem folgende Methoden für die Paraphrasengenerierung [Cha20]:

LEXIKALISCHER AUSTAUSCH VON WÖRTERN Dieser Ansatz ähnelt der schwach überwachten Methode von Berro u. a., wobei eine Ersetzung von Wörtern durch einen Thesaurus oder Worteinbettungen möglich ist [Cha20].

ZURÜCKÜBERSETZUNG (ENGL. BACK TRANSLATION) Dieser Ansatz ist der gleiche wie bei der Pivotübersetzung in [Ber+21].

REGELBASIERTE TRANSFORMATION EINES SATZES Chaudhary schlägt bei diesem Ansatz vor, angelehnt an definierte Regeln, Sätze zu paraphrasieren. Dabei kann z. B. ein Satz mithilfe eines Dependency Parses von einer aktiven Formulierung in eine passive Formulierung umgewandelt werden, ohne dabei die Bedeutung zu verlieren. Ein solches Vorgehen wird in der Arbeit von Coulombe umgesetzt [Cou18].

GENERATIVE METHODEN Dieser Ansatz ähnelt ebenfalls dem T5 Ansatz von Berro u. a., wobei über T5 hinaus weitere vortrainierte Sprachmodelle verwendet werden können, wie beispielsweise GPT.

In der Arbeit von Thompson und Post wird ebenfalls neurale Übersetzung genutzt, um Paraphrasen zu generieren [TP20]. Ein multilinguales neuronales Modell wird genutzt, welches mehrere Sprachen übersetzen kann und dadurch zur Pivotübersetzung genutzt werden kann. Dieser Ansatz ist ähnlich zu [Ber+21], aber nur mit einem einzigen Modell. Das Modell der Arbeit wurde auf Github veröffentlicht[6] und es gibt bereits eine Implementierung, welche für das Chatbotsystem Rasa[7] veröffentlicht wurde. Es wird außerdem eine Methode genutzt, um zu verhindern, dass die Eingabe ähnlich der Ausgabe ist, indem gleiche N-Gramme zwischen Ausgabe und Eingabe schlechter Bewertet werden beim Dekodieren bzw. Generieren von Paraphrasen [TP20].

5 https://www.deepl.com/translator Letzter Aufruf 20.06.2022
6 https://github.com/thompsonb/prism Letzter Aufruf 20.06.2022
7 https://github.com/RasaHQ/paraphraser Letzter Aufruf 20.06.2022

In der Arbeit von Deng u. a. wurden verschiedene Ansätze betrachtet, um Trainingsdaten zu erweitern [Den+22], wobei ähnlich zu [Ber+21] und [Cha20] Austausch durch Wörterbücher oder Worteinbettungen (Word2Vec [Mik+13]) verwendet wurden [Den+22, S. 191 - 192]. Außerdem nutzten die Autoren auch *Parrot* [Dam21], um neue Trainingsdaten zu erzeugen [Den+22, S. 192]. Deng u. a. kommen zum Schluss, dass die Verwendung von *Parrot* am effektivsten sei für das Training besserer Klassifizierungen, da es ganze Sätze paraphrasiert, anstatt nur Wörter auszutauschen [Den+22, S. 192].

Tabelle 3.5 zeigt zusammenfassend eine Übersicht über mögliche Methoden zur Paraphrasierung geben, die in den genannten Arbeiten vorgestellt werden.

Zusammenfassung	Sprache	Modell	Methode(n)	Datensatz / Datensätze	Quelle	Wiss. Arbeit	Evaluation
Parrot Paraphraser	Englisch	T5	Textgenerierung	MSR Paraphrase, Google PAWS, ParaNMT, Quora Fragenpaare	Github	[Dam21]	Intentklassifizierung wird evaluiert mit generierten Paraphrasen
Paraphrase Generator	Englisch	T5	Textgenerierung	Google PAWS	Github	[Ali20]	-
Paraphrase Generator	Englisch	T5	Textgenerierung	Quora Fragenpaare	Github	[Gol20]	-
Pipeline zur Paraphrasierung	Englisch	T5	Textgenerierung, Pivotübersetzung, Wortersetzung durch Synonyme	Quora Fragenpaare, ParaNMT	Github	[Ber+21]	-
Methoden zur Trainingsdatenerweiterung für NLP	Englisch	-	Textgenerierung, Pivotübersetzung, Wortersetzung durch Synonyme oder Worteinbettungen, Syntax Transformation	-	-	[Cha20]	-
Paraphrasierung mit neuronalen Übersetzungsmodellen	Multilingual	Eigenes Transformer-Modell	Pivotübersetzung	-	Github	[TP20]	BLEU, manuelle Evaluation (grammatikalisch, semantische Ähnlichkeit)
Datenerweiterung für NLP	Englisch	-	Wortersetzung durch Synonyme oder Worteinbettungen, Textgenerierung (mit Parrot)	-	-	[Den+22]	-

Tabelle 3.5: Zusammenfassung verwandte Arbeiten zur Paraphrasierung.

In Tabelle 3.6 werden relevante Datensätze zusammengefasst, die Paraphrasen in Satzgröße als Beispiele beinhalten. Die meisten Datensätze sind für die binäre Klassifizierungsaufgabe der Identifizierung von Paraphrasen erstellt worden. Das bedeutet, die Datensätze enthalten Paraphrasen, die keine sind. Diese Paraphrasen können anhand des Labels gefiltert werden, sodass nur die korrekten Paraphrasen zum Training der Textgenerierung genutzt werden können. Dieses Vorgehen reduziert die Anzahl der Beispiele. In der nachfolgenden Tabelle sind die Größen der Datensätze zur Paraphrasierungs aufgelistet, die Beispiele sind dabei nicht gefiltert, wie später im Training der Textgenerierung.

Datensatz	Aufteilung	Training	Test	Evaluation	Gesamt	Referenz
PAWS (Englisch)	Training, Test, Evaluierung	91.786	8.677	8.000	**108.463** (mit Label)	[ZBH19]
PAWS-X (Deutsch)	Training, Test, Evaluierung	49.401	1.932	1.967	**53.300** (mit Label)	[Yan+19]
Quora Fragepaare (Englisch)	Training	über 400.000	-	-	**400.000**	[Quo]
ParaNMT (Englisch)	-	über 50 Mio.	-	-	**50 Mio.**	[WG18]
Microsoft Research Paraphrase Corpus (Englisch)	Training, Test	4.076	1.725	-	**5.801**	[DB05]

Tabelle 3.6: Übersicht der Datensätze mit Paraphrasen in Satzgröße.

Die Datensätze ParaNMT und PAWS wurden mithilfe der Methode der Zurückübersetzung erstellt [WG18; ZBH19].

Der einzige vergleichbare Datensatz in deutscher Sprache (PAWS-X) ist eine neuronal übersetzte Version der Trainingsdaten von PAWS [Yan+19]. Die Test- und Evaluierungsdaten sind professionell Übersetzte Versionen des Datensatzes [Yan+19].

4 METHODEN

In diesem Kapitel werden die getätigten Experimente vorgestellt, um festzustellen, ob Trainingsdaten für Chatbots mit Textgenerierungsmethoden sinnvoll erweitert oder erstellt werden können.

Anfangs wird beschrieben, wie Webtexte für eine manuelle Evaluation extrahiert werden, unter Berücksichtigung der gefundenen Möglichkeiten (Kapitel 3.1).

Danach werden Ansätze ausgewählt, die in den relevanten Arbeiten verwendet werden (siehe Kapitel 3.2 und 3.3). Dabei wird darauf geachtet, dass State-Of-The-Art Methoden der Textgenerierung verwendet werden.

4.1 EXTRAKTION VON WEBTEXTEN

Für die Fragengenerierung werden Texte in Absatzgröße benötigt. Um diese durch Webseiten zu erhalten, wurden in Kapitel 3.1 verschiedene Möglichkeiten gezeigt. Im Rahmen dieser Arbeit wird eine vollautomatische Lösung genutzt. Der Grund dafür ist, dass die Lösungen die ein Annotieren der Webseite erfordern ausschließlich proprietär sind. Außerdem ist unklar, ob durch das Verwenden der proprietären Lösungen bessere Texte in Absatzgröße extrahiert werden können.

Bezüglich der Extraktion von Frage-Antwort-Paaren wurde in Kapitel 3.1 festgestellt, dass Ansätze entweder proprietär [Agr+20; Tim; Goof] oder methodisch beschrieben sind [JR05; RJH21].

Da Frage-Antwort-Paare direkt als Trainingsdaten in Chatbots verwendet werden können, werden diese Ansätze ebenfalls nicht beim Experiment berücksichtigt. Die Frage-Antwort-Paare können jedoch auch zur Textgenerierung bzw. Trainingsdatenergänzung verwendet werden. Aus Fragen können semantisch ähnliche Fragen erzeugt werden (Paraphrasierung) und aus Antworten neue Fragen generiert werden (Fragengenerierung).

Dadurch, dass die meisten Ansätze aufgrund ihrer Verwendbarkeit entfallen werden die quelloffenen Bibliotheken Trafilatura und JusText im Experiment getestet.

In Abbildung 4.1 werden Aufgaben und Artefakte des Webcrawlings in einem Business Process Model Notation (BPMN) Diagramm[1] dargestellt.

1 Eine Erklärung der verwendeten Symbole ist unter folgender URL zu finden https://camunda.com/de/bpmn/bpmn-2-0-symbol-reference/ Letzter Aufruf 22.06.2022

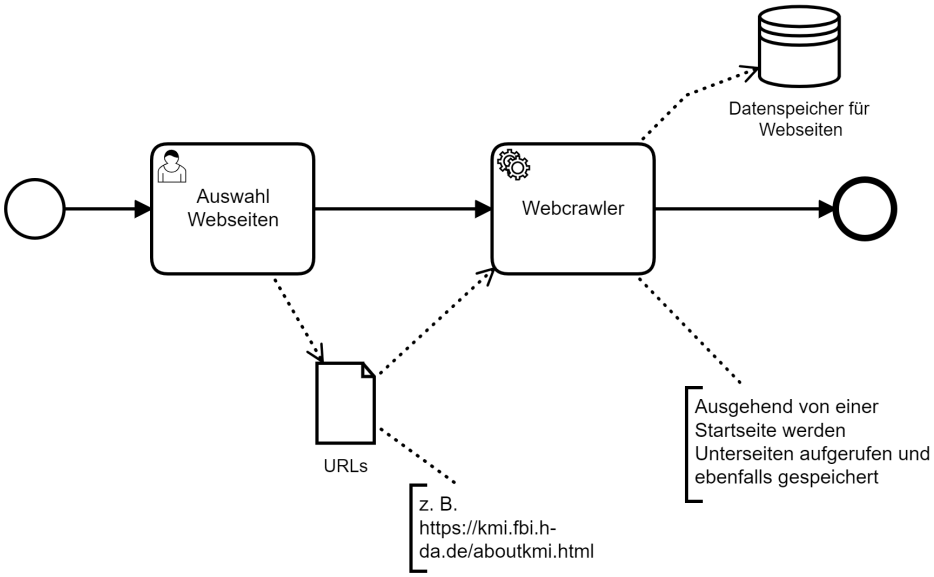

Abbildung 4.1: Textextraktion von Webseiten mit einem Webcrawler.

4.2 FRAGENGENERIERUNG

Bei der Fragengenerierung konnten in den relevanten Arbeiten regelbasierte und neuronale Ansätze identifiziert werden (Kapitel 3.2). Aktuelle neuronale Ansätze nutzen Sprachmodelle, die eine Transformer-Architektur verwenden. Außerdem nutzen diese als Eingabe einen Text in Absatzgröße.

Im Rahmen dieser Arbeit wird überprüft, ob die Fragengenerierung auch in der deutschen Sprache mit veröffentlichten Trainingsdaten möglich ist.

Dazu ist vor allem der veröffentlichte GermanQuAD [MRP21] Datensatz zu erwähnen, da dieser im gleichen Format wie der oft verwendete englische Datensatz SQuAD [Raj+16] ist und deshalb ebenfalls für die Fragengenerierung verwendet werden kann. Ein Unterschied zwischen den Datensätzen ist dabei die Beispielanzahl. Der GermanQuAD-Datensatz enthält ca 13.000 Frage-Antwort-Paare, während der SQuAD-Datensatz über 100.000 Paare verfügt [MRP21; Raj+16] (siehe Tabelle 3.4).

Des Weiteren sind vortrainierte Sprachmodelle die mit deutschsprachigen Daten trainiert wurden zu betrachten, da von diesen Sprachmodellen durch das Transferlernen profitiert werden kann und so ein potentieller Datenmangel kompensiert werden kann.

Die generierten Fragen werden automatisch evaluiert durch Metriken, die in anderen Arbeiten verwendet wurden (siehe Kapitel 2.3.3 und Kapitel 3.2). Diese Metriken geben eine Auskunft darüber, wie viel Wortüberschneidungen es zwischen den vorhergesagten Fragen und den Label Fragen gibt. Darüber hinaus ist eine manuelle Evaluierung sinnvoll, um die Qualität der Fragen im Bezug auf die Verwendung in einem Szenario der Chatbotentwicklung beurteilen zu können und die Forschungsfrage zu beantworten.

Es werden verschiedene Ansätze getestet, um die Aufgabe der Fragengenerierung umzusetzen. Dabei stammen die Ansätze von relevanten Arbeiten zur Fragengenerierung (Kapitel 3.2).

Zuerst wird die Fragengenerierung ohne Wissen der Antwort getestet, die als Eingabe einen Absatz nutzt und als Ausgabe eine Sequenz von Fragen generiert (wie in [Pat20; Gro+21; Lop+20]). Dadurch, dass bei der Fragengenerierung ohne Wissen der Antwort keine Textstelle als Antwort markiert wird, muss das Modell die relevante Textstelle selbst bestimmen.

Zusätzlich wird die Fragengenerierung mit Wissen der Antwort trainiert. Die markierte Antwort ist in den Datensätzen vorgegeben, in einer echten Anwendung muss allerdings die Antwort entweder manuell oder automatisch ausgewählt werden. Zum Auswählen der Antwort aus einem Text in Absatz oder Satzgröße gibt es verschiedene Möglichkeiten. Beispielsweise kann ein Modell ausschließlich für die Antwortextraktion trainiert werden (vgl. [Aky+21; Pat20]). Kolditz oder Patil schlagen zudem das Nutzen von Satzgliedern wie Nominalphrasen als potentielle Antwort vor [Kol15, Kap. 4.1], [Pat20]. Im Fall dieses Experiments wird als Antwort eine benannte Entität verwendet, dieser Ansatz wurde auch in der Arbeit von Patil vorgeschlagen [Pat20].

4.3 PARAPHRASIERUNG

Wenn Fragen oder Äußerungen bereits vorhanden sind, bietet die Paraphrasierung dieser Trainingsbeispiele einen einfachen Weg, um weitere Trainingsdaten zu erhalten.

Ähnlich der Fragengenerierung gibt es regelbasierte Ansätze, wie das Austauschen von Verben und Nomen durch Synonyme und neuronale Ansätze, die Sprachmodelle mit Transformer-Architektur für diese Aufgabe trainieren oder maschinelle Übersetzung nutzen (Kapitel 3.3).

Ein Datensatz zur Paraphrasenerkennung in deutscher Sprache ist der PAWS-X-Datensatz [Yan+19], dieser kann für ein Finetuning verwendet werden. Der deutschsprachige Anteil von PAWS-X enthält ca. 25.000 Satzpaare [Yan+19].

Zur Paraphrasierung kann das gleiche Sprachmodell wie bei der Fragengenerierung verwendet werden.

Auf eine Evaluierung des Austauschs der Synonyme wird verzichtet, da diese Methode vor allem von den verwendeten Wörterbüchern abhängt, aus welchen Synonyme bezogen werden. Für die deutsche Sprache gibt es dafür das quelloffene Wörterbuch OpenThesaurus [Ope], aber auch lizensierte Synonym-Wörterbücher wie GermaNet [HF97]. Dieser Ansatz verspricht allerdings nur lexikalische Änderungen der Texte und keine Umstellung der Worte in einem Satz.

4.4 MANUELLE EVALUIERUNG

Um die Ergebnisse genauer im Bezug auf die Verwendbarkeit für Trainingsdaten von Chatbots einzuordnen, wird eine manuelle Evaluierung der Experimente benötigt. Hierzu wird im nachfolgenden ein Fragebogen vorgestellt, der eine Stichprobe von generierten Fragen bewertet. Dabei sollen folgende Fragen durch die manuelle Evaluation beantwortet werden:

1. Ist die Frage grammatikalisch korrekt?
2. Ist die Frage durch den gegebenen Textabschnitt beantwortbar?
3. Kann die Frage für einen Chatbot genutzt werden?

Bezüglich Letzterem, ob sich eine Frage für einen Chatbot eignet, werden folgende Punkte berücksichtigt:

- Ist die Frage aus Sicht des Nutzers formuliert? (z. B. 1. Person Singular „Kann ich . . . “)
- Werden potentiell relevante Informationen für die Zielgruppe der Webseite abgefragt? (z. B. Termine, Kontaktpersonen, nützliche Informationen usw.)

Das Gleiche gilt für die Generierung von Paraphrasen, dabei werden folgende Punkte beachtet:

1. Ist der generierte Satz keine exakte Kopie?
2. Ist die Paraphrase grammatikalisch korrekt?
3. Bleibt die Bedeutung erhalten?
4. Werden andere Wörter verwendet? (lexikalische Änderungen)
5. Wird der Satzbau verändert? (syntaktische Änderungen)

4.5 ZUSAMMENFASSUNG DER METHODIK

Um die Forschungsfrage zu beantworten, ob Textgenerierung in Form von Fragen- und Paraphrasengenerierung genutzt werden kann, werden Experimente getätigt, welche durch automatische Metriken und manuelle Evaluationen beurteilt werden.

Dabei werden zwei Prozesse durchlaufen, um die Forschungsfrage zu beantworten. Zum einen werden zwei Transformer-Modelle zur Textgenerierung trainiert mit verschiedenen Datensätzen, die im Rahmen der Recherche gefunden wurden. Zum anderen wird ein Webcrawler entwickelt, der Texte in Absatzgröße von Webseiten extrahiert. Diese Absätze werden als Antworten verwendet, um daraus Fragen zu generieren. Die generierten Fragen werden anschließend mit dem Absatz als Frage-Antwort-Paar zu Trainingsdaten für einen Chatbot zusammengefasst.

Der Prozess des Finetunings eines Sprachmodells wird in Abbildung 4.2 durch ein BPMN Diagramm dargestellt. Im ersten Schritt muss ein passendes vortrainiertes Sprachmodell ausgewählt werden und passende Datensätze für das zu lösende Textgenerierungsproblem. Wenn ein Sprachmodell ausgewählt wurde und Trainings-, Test-, und Evaluierungsdaten gewählt sind, wird ein Modell für die Textgenerierungsaufgabe trainiert. Das trainierte Modell kann durch eine automatische Evaluation ausgewertet werden mit Textgenerierungsmetriken wie BLEU.

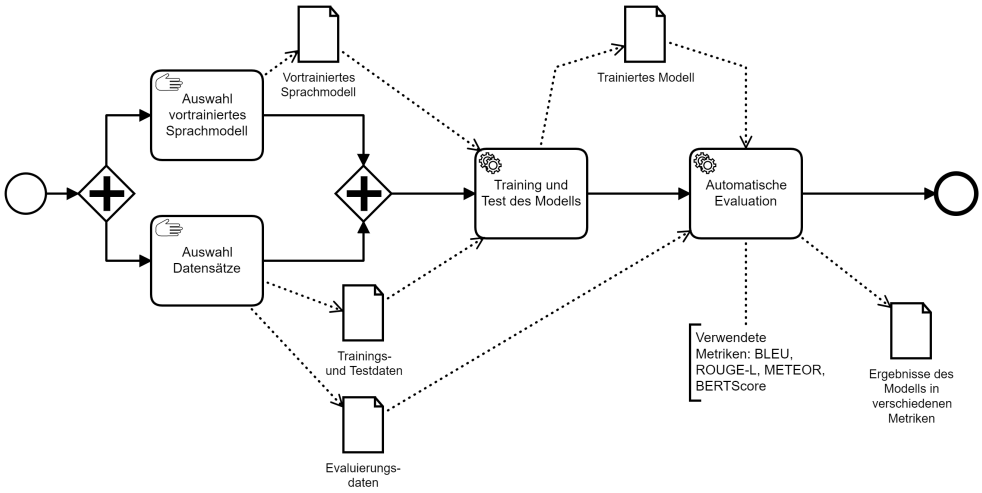

Abbildung 4.2: Finetuning eines Sprachmodells.

Die Abbildung 4.3 gibt einen Überblick über das gesamte Experiment in Form eines BPMN Diagramms.

Das Experiment startet mit dem Finetuning eines vortrainierten Sprachmodells für die Fragengenerierung und Paraphrasengenerierung in deutscher Sprache. Des Weiteren werden Webtexte von ausgewählten Webseiten in einer Datenbank gespeichert. Wenn Webtexte in der Datenbank vorliegen und ein trainiertes Modell zur Fragengenerierung und Paraphrasierung vorhanden ist, werden aus den Webtexten Fragen generiert. Die generierten Fragen werden schließlich mit ihren korrespondierenden Eingaben ausgewertet und es findet eine manuelle Auswahl der passenden Frage-Antwort-Paare statt. Im Rahmen dieser Arbeit werden die generierten Fragen aus Webtexten zusätzlich über einen Fragebogen evaluiert. In einem Anwendungsszenario der Chatbotentwicklung würden unpassende Frage-Antwort-Paare verworfen oder genutzt werden, abhängig von den Anwendungsfällen. Sind passende Frage-Antwort-Paare bestimmt, werden zusätzlich Paraphrasen der Frage durch eine Paraphrasengenerierung erzeugt. Dieses Vorgehen ermöglicht schließlich die Erstellung von Trainingsdaten für einen Chatbot durch eine Webseite. Die finalen Trainingsdaten für den Chatbot bestehen aus Antworten und mehreren unterschiedlichen Fragesätzen zu je einer Antwort. Alle Fragesätze zu einer Antwort ergeben schließlich eine Intention.

Eine Limitierung dieses Ansatzes ist, dass nur Fragesätze automatisch generiert werden können. Andere Satzarten sind möglich für eine Intenti-

onsklassifizierung, allerdings im Rahmen eines Frage-Antwort-Chatbots unwahrscheinlich.

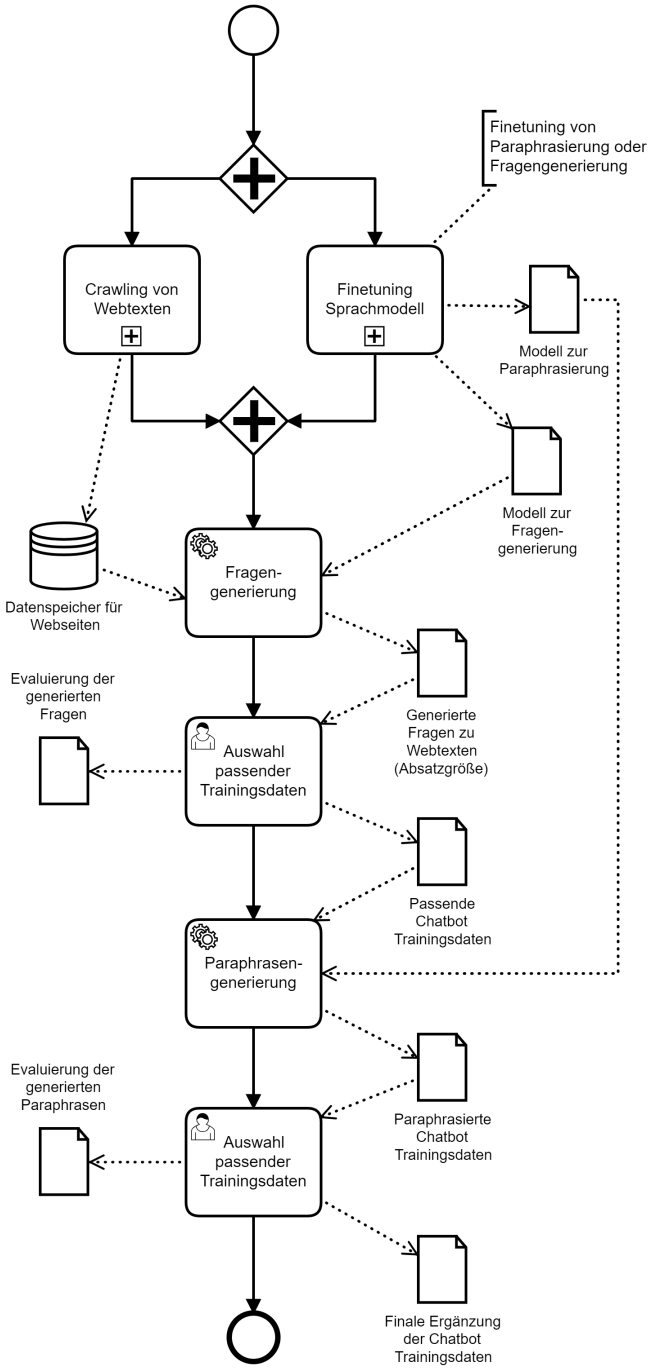

Abbildung 4.3: Aufbau des Experiments. Teilprozesse werden in den Abbildungen 4.1 und 4.2 dargestellt.

5 EXPERIMENT

Im Folgenden Kapitel werden die getätigten Experimente beschrieben, die zur Datenerweiterung für Trainingsdaten von Chatbots im Rahmen dieser Arbeit getätigt wurden.

Zuerst werden Webseiten beschrieben, von welchen Texte mithilfe eines Webcrawlers extrahiert werden.

Anschließend werden Datensätze in deutscher Sprache beschrieben, die zum Training von neuronalen Modellen mit Transformer-Architektur verwendet werden.

Danach wird die Auswahl und das Training von Modellen beschrieben, die zur Textgenerierung und damit zur Bewältigung der Aufgaben dieser Arbeit genutzt werden.

Schließlich wird die Konfiguration und die verwendeten Implementierungen der Metriken beschrieben.

Der Programmcode aller Schritte ist der Arbeit beigelegt oder auf Github[1] verfügbar. In den Kapiteln werden jeweils die Dateien bzw. Verzeichnisse beschrieben, die Programmcode für den jeweiligen Schritt enthalten.

5.1 WEBCRAWLING ZUR MANUELLEN EVALUATION

Um Textinformationen von Webseiten zu extrahieren, wurde ein Webcrawler mit dem *scrapy*[2] Framework entwickelt, der ausgehend von einer Startadresse weitere Unterseiten öffnet und herunterlädt. Die Webseiten werden dabei in ihrer HTML-Struktur heruntergeladen. Außerdem wird ein JavaScript Renderingservice namens *Splash*[3] genutzt, um dynamische Inhalte (JavaScript) zu rendern, falls diese beim Initialisieren der Seite erstellt werden.

Moderne Webseiten nutzen JavaScript-Frameworks, die Inhalt erst nach dem Rendern der Webseite anzeigen (vgl. Kapitel 2.5). Durch dieses Vorgehen können Daten berücksichtigt werden, die erst mit der Ausführung des JavaScript-Codes in das DOM geladen werden. Eine Limitierung dieses Ansatzes ist, dass keine Informationen berücksichtigt werden können, die Nutzerinteraktion mit der Webseite erfordern.

Die beiden quelloffenen Bibliotheken Trafilatura und JusText wurden für die Textextraktion getestet [Bar21; Jus]. Zum Test[4] wurden Webseiten verwendet, aus denen im Experiment Fragen generiert werden sollen. Mit der Bibliothek JusText konnten dabei passende Textblöcke (z. B. keine Werbetex-

1 https://github.com/TiloMichel/textgen-for-chatbot-training-german Letzter Aufruf 22.06.2022

2 https://scrapy.org/ Letzter Aufruf 11.04.2022

3 https://github.com/scrapinghub/splash Letzter Aufruf 11.04.2022

4 Der Programmcode des manuellen Tests befindet sich im Ordner *o_webcrawler* als Jupyter Notebook mit dem Namen *webpage_to_text_test.ipynb*

te) für eine Fragengenerierung extrahiert werden. Die Bibliothek Trafilatura ignorierte beim Test Textblöcke, die relevant waren.

Eine Limitierung dieses Ansatzes ist, dass keine allgemeine Aussage getroffen werden kann, wie gut die Textextraktion auf anderen Webseiten funktioniert. Das liegt daran, dass der HTML-Code von Webseiten im allgemeinen sehr verschieden ist und von den verwendeten Technologien bei der Erstellung abhängig.

Die Textextraktion ist ein zentraler Schritt, um eine Webseite in strukturierte Daten für einen Chatbot umzuwandeln. Das bedeutet, je besser informative Texte von Webseiten automatisiert extrahiert werden können, desto besser können in den darauffolgenden Schritten Fragen generiert werden. Im Rahmen dieser Arbeit wird angenommen, dass alle extrahierten Texte informativ sind.

In einer realen Anwendungen müssen die extrahierten Texte zusätzlich gefiltert werden, nach den vorgegebenen Anwendungsfällen für einen informativen Chatbot. Bei der Filterung der Texte geht es darum, welche Informationen der Webseite potentiell von einem Nutzer abgefragt werden könnten. Es ist unklar, ob dieses Problem automatisierbar ist.

Folgende Webseiten werden für eine Fragengenerierung aus Absätzen verwendet:

KOMMUNIKATIONS UND MEDIENINFORMATIK (KMI) [5] Die Webseite der KMI der Hochschule Darmstadt bietet Studiumsinteressierten Informationen über das Kommunikations und Medieninformatik Studium in Form von Text und Videos, sowie weiterführende Links zu Webseiten an der Hochschule Darmstadt, sowie einen Chatbot namens KIMI

ROBERT-KOCH-INSTITUT (RKI) [6] Das RKI bietet auf seiner Webseite Informationen rundum Gesundheit, Krankheiten, Forschung und Geschichte des RKIs. Besonders im Hinblick auf die COVID-19 Pandemie veröffentlicht das RKI aktuelle Informationen auf ihrer Webseite.

DEUTSCHE GESELLSCHAFT FÜR ERNÄHRUNG (DGE) [7] Die DGE bietet auf ihrer Webseite Informationen zu Veranstaltungen und Forschung zum Thema Ernährung.

Bei der Extraktion der Textblöcke stellte sich heraus, dass unpassende Textblöcke extrahiert wurden in Form von Überschriften oder Fragen. Aus diesem Grund wurde mithilfe des Dependency Parsers von spaCy überprüft, ob der Textblock mindestens einen Satz mit einem Subjekt, Prädikat bzw. Verb oder Objekt enthält [Hon+20]. Textblöcke, für die diese Überprüfung negativ war, wurden vor der Fragengenerierung herausgefiltert. Des Weiteren wurden Fragen herausgefiltert. Diese sind erkennbar an Fragezeichen, am Satzende oder an Interrogativpronomen.

5 https://kmi.fbi.h-da.de/ Letzter Aufruf 12.04.2022
6 https://www.rki.de/DE/Home/homepage_node.html Letzter Aufruf 12.04.2022
7 https://www.dge.de/ Letzter Aufruf 12.04.2022

Die Implementierung beider Filter wurde im Skript zur Fragengenerierung umgesetzt (4_inference/question_generator.py) unter den Methodennamen *__is_question* und *__contains_at_least_one_sentence* der QuestionGenerator Klasse.

5.2 AUSWAHL EINES SPRACHMODELLS

Vortrainierte Sprachmodelle sind in verschiedenen Sprachen verfügbar. Eine der größten Sammlungen vortrainierter Sprachmodelle unterhält das Unternehmen huggingface, besonders für Modelle mit Transformer-Architektur für NLP-Aufgaben [Wol+20]. In den verwandten Arbeiten (Kapitel 3.2) werden vor allem Sequenz-zu-Sequenz Modelle wie T5 (Encoder-Decoder) verwendet, aber auch Encoder (BERT) oder Decoder Modelle wie GPT-2. Modelle die in englischer Sprache für die Aufgabe der Fragengenerierung oder Paraphrasierung auf huggingface verfügbar sind, nutzen vor allem die T5 Architektur [Pat20; Dam21; Gol20].

Es existieren zwar vortrainierte BERT und GPT-2 Modelle auf huggingface für die deutsche Sprache, allerdings benötigen diese zusätzliche Anpassungen, um Text-zu-Text-Generierung zu ermöglichen (siehe Kapitel 2.2.4).

Die Wahl eines Modells fällt deshalb auf das multilinguale T5-Modell, da dieses ohne größere Anpassungen durch das Text-zu-Text Framework trainiert werden kann und auch über Modelle mit höherer Parameteranzahl verfügt [Raf+19; Xue+20].

Außerdem wurde das multilinguale T5-Modell im Vortraining mit 101 Sprachen trainiert, auch mit deutschsprachigen Daten vortrainiert (ca. 3.05 % der Daten des mT5 Trainingskorpus) [Xue+20, Tabelle 6].

5.3 FRAGENGENERIERUNG

In diesem Kapitel werden alle nötigen Schritte zum Training der Fragengenerierung für ein Modell beschrieben. Zuerst wird beispielhaft der GermanQuAD-Datensatz analysiert. Anschließend wird die Datensatzanpassung beschrieben. Danach wird genauer auf das Training eingegangen und welche Experimente getätigt wurden.

5.3.1 *Auswahl und Exploration der Datensätze*

Um das vortrainierte Sprachmodell für die Fragengenerierung zu trainieren werden Datensätze in deutscher Sprache benötigt. In Kapitel 3.2 wurden vorhandene Datensätze bereits eingeführt. Alle Datensätze werden in verschiedenen Kombinationen zur Fragengenerierung ohne Wissen der Antwort genutzt. Im Folgenden wird der GermanQuAD-Datensatz analysiert, um die Verteilung von Interrogativpronomen und Fragearten festzustellen (siehe Kapitel 2.3.1).

Der GermanQuAD-Datensatz enthält vor allem Fragesätze, die als Ergänzungsfragen klassifiziert werden können (siehe Kapitel 2.3.1). Ja-Nein- oder

Alternativfragen werden in dem Datensatz kaum berücksichtigt. Die Autoren des
GermanQuAD-Datensatzes geben in ihrer Arbeit eine relative Verteilung von Fragewörtern im Test Datensatz an, die in Abbildung 5.1 dargestellt wird (Die Summe der relativen Anzahl im Paper [MRP21] ergeben 100.1 Prozent, weshalb die Daten im Diagramm leicht abweichen) [MRP21, Tabelle 2]:

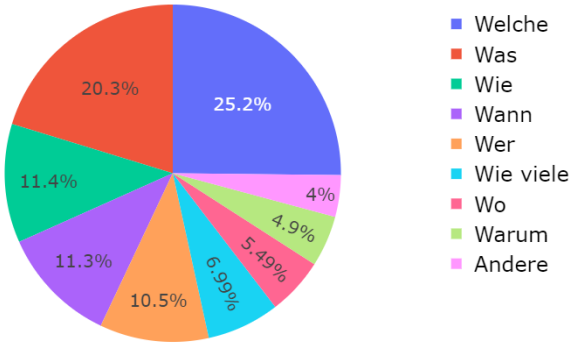

Abbildung 5.1: Verteilung von Fragewörtern im GermanQuAD Testdatensatz nach Möller, Risch und Pietsch [MRP21].

Da GermanQuAD zum Training eines Modells verwendet werden soll, wird der Datensatz im Hinblick auf die vorhandenen Fragen genauer analysiert. Dazu werden die Fragen mithilfe eines POS-Taggers untersucht. Der POS-Tagger ist in einer bereits trainierten spaCy Pipeline vorhanden [Hon+20]. Als Tagset wird das Stuttgart-Tübigen-Tagset (STTS)[8] genutzt [Stt]. Für Ergänzungsfragen sind die Interrogativpronomen von Interesse. Das STTS hat für diese Art von Wörtern drei POS-Tags: attribuierendes Interrogativpronomen (PWAT), substituierendes Interrogativpronomen (PWS) und adervbiales Interrogativpronomen (PWAV) [Stt]. Alle Fragen werden durchsucht und falls eines der drei möglichen Interrogativpronomenarten vorhanden ist, wird das Fragewort und die Interrogativpronomenart dokumentiert.

Der Trainingsdatensatz von GermanQuAD enthält 11.518 Fragen, einige Fragen enthalten mehr als ein Interrogativpronomen Beispiel: „Von wo nach wo wandern Zugvögel?“, weshalb die Anzahl der Interrogativpronomen auf 11.540 steigt [MRP21]. Es wurden bei der Analyse insgesamt 29 verschiedene Interrogativpronomen gefunden, wobei die fünf häufigsten in Tabelle 5.1 dargestellt werden und die Anzahl der Fragen, in welchen keine Interrogativprnomen erkannt wurden, am Ende aufgeführt werden.

8 https://homepage.ruhr-uni-bochum.de/stephen.berman/Korpuslinguistik/Tagsets-STTS.html Letzter Aufruf 23.06.2022

Fragewort	Absolute Häufigkeit (von 11.540)	Relative Häufigkeit (in %)
Wie	2281	19.77
Welche	1934	16.76
Was	1822	15.79
Wann	1292	11.2
Wer	590	5.11
Unbekannt	132	1.14

Tabelle 5.1: Verteilung der Interrogativpronomen in den GermanQuAD Trainingsdaten.

Die fünf häufigsten Interrogativpronomen haben zusammen einen relativen Anteil von 68,63 %.

Außerdem ist der Großteil der Fragen Ergänzungsfragen (ca. 98 %). Mithilfe der spaCy Pipeline wurden die Fragen überprüft, ob sie mit einem Fragezeichen enden und eine der drei POS-Tags für Interrogativpronomen zugehörig sind.

98.6 % (11.362 Fragen) enthalten ein Interrogativpronomen und enden mit einem Fragezeichen. 1.4 % (156) der Fragen fehlt ein Fragewort oder ein Fragezeichen am Ende der Frage.

Von diesen Fragen haben 28 kein Fragezeichen am Ende des Satzes und 132 konnten durch den POS-Tagger nicht einem der Interrogativpronomenarten zugeordnet werden. Die Anzahl 156 kommt zustande, da bei einigen Fragen beide Kriterien fehlen und nicht doppelt gezählt wird.

Schließlich können die Arten der Interrogativpronomen unterteilt werden. Von 11.540 Fragewörtern gehören 5.049 (43.8 %) zum PWAV, 3.800 (32.9 %) zum PWAT und 2.559 (22.2 %) zum PWS STTS-Tag. Für 132 (1.14 %) Fragen konnte kein Interrogativpronomen gefunden werden.

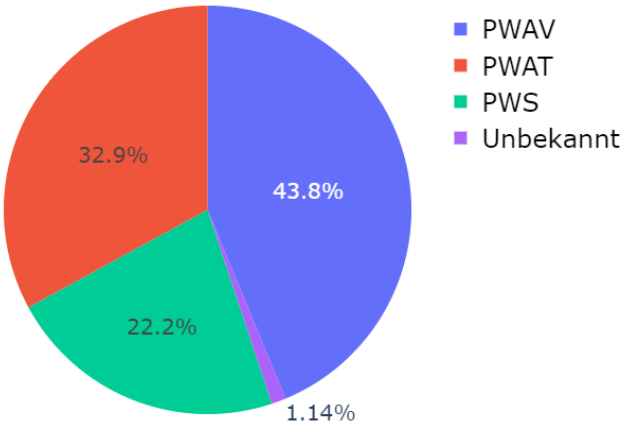

Abbildung 5.2: Verteilung von Fragewörter nach STTS-Tags im GermanQuAD Trainingsdatensatz.

Der Testdatensatz enthält 2.204 Frage-Antwort-Paare und 28 verschiedene Fragewörter [MRP21].

Fragewort	Absolute Häufigkeit (von 2217)	Relative Häufigkeit (in %)
Was	448	20.2
Wie	398	17.95
Welche	252	11.37
Wann	249	11.2
Wer	207	9.34
Unbekannt	45	2.02

Tabelle 5.2: Verteilung der Interrogativpronomen in den GermanQuAD Testdaten.

Die fünf häufigsten Interrogativpronomen haben zusammen relativen Anteil von 70.04 % an allen Fragewörtern der Testdaten.

Der Testdatensatz enthält 2.146 (97,4 %) Fragen nach den oben genannten Kriterien (Fragezeichen und Fragewort). 58 (2.6 %) Fragen enthalten eine der beiden oder beide Kriterien nicht.

Die Arten der Interrogativpronomen ergeben im Testdatensatz 930 (41.9 %) PWAV, 686 (30.9 %) PWS und 556 (25.1 %) zum PWAT STTS-Tag. Für 45 (2.03 %) Fragen konnte kein Interrogativpronomen gefunden werden.

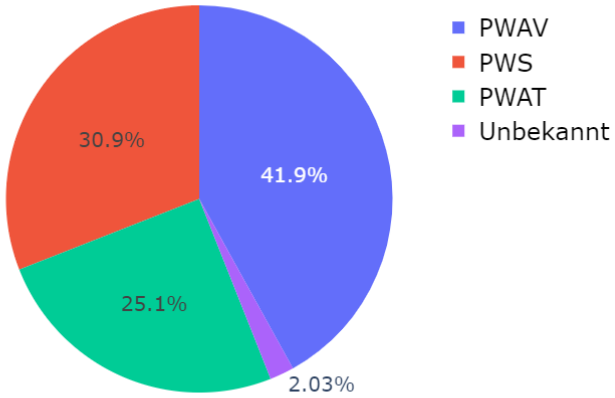

Abbildung 5.3: Verteilung von Fragewörtern nach STTS-Tags im GermanQuAD Testdatensatz.

Durch die Analyse der Daten konnte festgestellt werden, dass der Großteil der Fragen in Trainings- und Testdaten Ergänzungsfragen sind, da sie Interrogativpronomen enthalten.

Des Weiteren kann festgestellt werden, dass wenige Fragen ohne Fragezeichen enden. Außerdem konnten in wenigen Fragen keine Interrogativpronomen erkannt werden, das liegt unter anderem daran, dass diese Fragen Ja-Nein-Fragen oder Alternativfragen sind bsp. „Ist USB mit Atari MINT benutzbar?“ oder der POS-Tagger die Fragewörter nicht erkennen konnte.

Schließlich ist zu vermuten, dass ein Modell, das mit diesen Daten trainiert wird, vermutlich größtenteils Ergänzungsfragen generieren wird, da der Datensatz zum großen Teil aus dieser Frageart besteht.

Die Auswertung des GermanQuAD-Datensatzes wurde mit einem Jupyter Notebook umgesetzt, dass auch für die anderen deutschsprachigen Datensätze im SQuAD Format genutzt werden kann. Es befindet sich im Ordner *1_data_preparation_and_exploration* unter dem Namen *question_generation_dataset_statistics.ipynb*.

5.3.2 *Datenvorbereitung*

Ein Datensatz zur Fragengenerierung muss zuerst für das Training vorverarbeitet werden.

Zur Verarbeitung der Trainingsdaten werden die Bibliotheken datasets [Lho+21] von huggingface und pandas [tea20] genutzt.

Bei der Fragengenerierung ohne Wissen der Antwort werden aus dem ursprünglichen Datensatz (siehe Datenformat von SQuAD Listing 1) die Absätze und Fragen genutzt. Bei der Fragengenerierung mit Wissen der Antwort werden zudem die Antworten genutzt und im Text markiert mit speziellen Tokens („<hl>“, wobei hl für „highlight“ steht). Tabelle 5.3 zeigt Beispiele nach dem Formatieren der Daten.

Art der Fragengenerierung	Eingabe	Ausgabe
Ohne Wissen der Antwort	Richard Stallman gründet das GNU-Projekt mit dem Ziel, ein freies Betriebssystem zu erschaffen.</s>	Was war Stallmanns Ziel? <sep> Wer gründete das GNU-Projekt? </s>
Mit Wissen der Antwort	Richard Stallman gründet das GNU-Projekt mit dem Ziel, <hl>ein freies Betriebssystem zu erschaffen<hl>.</s>	Was war Stallmanns Ziel? </s>

Tabelle 5.3: Beispiele der Trainingsdaten zur Fragengenerierung.

Damit das Modell trainiert werden kann, müssen die Beispiele in Eingabe und Ausgabe unterteilt werden und durch einen Tokenizer verarbeitet werden. Die Tokens werden in Ids der Worteinbettungen des Modellvokabulars umgewandelt (siehe Kapitel 2.2.6).

Im ersten Schritt wird Eingabe und Ausgabe definiert, im Beispiel der Fragengenerierung ist die Eingabe ein Absatz und die Ausgabe verschiedene Fragen zu diesem. Es wird eine Unterteilung von drei Datensätzen genutzt: Trainings-, Test- und Evaluierungsdaten.

Trainingsdaten werden genutzt, um das vortrainierte Modell für die Aufgabe der Fragengenerierung zu trainieren (Finetuning).

Die Testdaten werden genutzt, um das Modell während des Trainings mit Werten der Verlustfunktion zu evaluieren. Dieses Vorgehen hilft, um Over- oder Underfitting zu vermeiden und festzustellen, ob das Model auf den Testdaten generalisiert.

Die Evaluierungsdaten werden schließlich genutzt, um das fertig trainierte Modell zu evaluieren, mit Metriken die für die jeweilige Aufgabe passend sind.

Wenn die drei Datensätze festgelegt sind, müssen die Beispiele durch den Tokenizer vorverarbeitet werden.

Außerdem benötigt das Modell Eingaben mit gleicher Tokenlänge, weshalb zu lange Eingaben abgeschnitten werden und zu kurze Eingaben aufgefüllt werden mit einem speziellen Auffülltoken (engl. padding token). Als Eingabelänge wurden 1024 Tokens festgelegt, da diese Länge auch im mT5 Votraining verwendet wurde [Xue+20], das erlaubt Eingaben großer Absätze. Als Ausgabelänge wurden 128 Tokens festgelegt, das entspricht ca. fünf Fragesätzen.

Um das Abschneiden von Eingaben zu vermeiden, wird außerdem überprüft, ob Eingaben oder Ausgaben die vorher festgelegte Tokenlänge überschreiten. Falls ja, wurden diese Beispiele aus dem Datensatz entfernt. Dieses Vorgehen ist nötig, da unvollständige Eingaben oder Ausgaben potentiell wichtige Informationen für das Training enthalten können, die sonst verloren gehen könnten.

Bei der Verwendung des Modells nach dem Training ist das Überprüfen der Tokenlänge nicht nötig, da das mT5-Modell relative Positionseinbettungen benutzt [Raf+19]. Das bedeutet, es können auch längere Texte an das Modell übergeben werden oder das Modell kann längere Texte generieren, als im Training erlernt wurden [T5m].

Beim Training werden allerdings feste Tokenlängen für Eingabe und Ausgabe benötigt, da die Hardware bei höheren Tokenlängen an ihre Grenzen stößt, weil größere Eingabesequenzen bei der Transformer-Architektur speicherineffizient sind [T5m].

Des Weiteren können bei T5 Prefixe für die Eingabe angegeben werden, damit klar ist, welche Aufgabe das Modell ausführen soll, da ein Modell auch für mehrere Aufgaben gleichzeitig trainiert werden kann. Beispielsweise kann der Prefix „generate question:“ verwendet werden, um zu zeigen, dass eine Frage generiert werden soll.

Ein weiterer Punkt ist das Markieren des Endes einer Sequenz durch den Ende-der-Sequenz-Token im Falle von T5 ist dieser „</s>“. Das hilft dem Modell das Ende der Sequenzen beim Training zu erkennen.

Um die Tokenlänge bei der Erstellung des Datensatzes zu berücksichtigen, wurden zwei Jupyter Notebooks erstellt. Die Überprüfung der Verteilung der Eingabe- und Ausgabelängen (in Tokens) ist im Ordner *1_data_preparation_and_exploration* als Jupyter Notebook hinterlegt unter dem Namen *check_dataset_examples_token_lengths.ipynb*. Abbildung 5.4 zeigt die Tokenlänge der Absätze im GermanQuAD Trainingsdatensatz. Die Erstellung der Datensätze wurde mit dem Jupyter Notebook *create_qg_dataset.ipynb* im gleichen Verzeichnis umgesetzt.

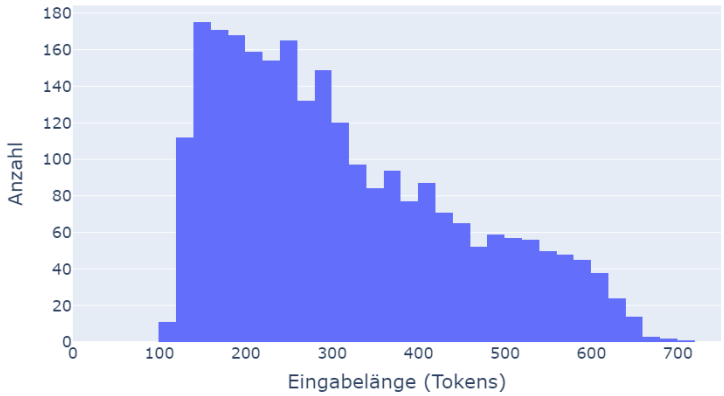

Abbildung 5.4: Verteilung der Tokenlänge von Absätzen (N = 2540) nach Anwendung des mT5 Tokenizers auf den GermanQuAD Trainingsdaten.

5.3.3 *Training der Fragengenerierung*

Zum Training bietet die Bibliothek *transformers* der Firma huggingface in Python Schnittstellen an, die für ein Training genutzt werden können [Wol+20]. Im Wesentlichen muss dort das vortrainierte Modell inklusive Tokenizer ge-

laden werden und die Hyperparameter angepasst werden für die jeweilige Aufgabe. Mit dem verarbeiteten Datensatz kann schließlich ein Modell trainiert werden.

Da die Methoden nach dem Vorbild von Arbeiten aus Kapitel 3.2 genutzt werden, wurden initial die Hyperparameter anderer Autoren wiederverwendet. Die folgenden Hyperparameter sind dabei von Bedeutung beim Training eines Modells (siehe huggingface TrainerArguments [Tra; Hugc]):

BATCHGRÖSSE PRO GERÄT Anzahl der Beispiele die parallel innerhalb eines Gerätes (z. B. GPU) augeführt werden.

GRADIENTAKKUMULIERUNGSSCHRITTE Anstatt die Beispiele nur verteilt zu trainieren, können die Gradienten für mehrere Schritte gesammelt werden, bevor das Netzwerk optimiert wird. Das ist notwendig, da das Training der Transformermodelle speicherintensiv ist und bessere Ergebnisse mit größerer Batchgröße erreicht werden können. Ein Nachteil dieses Ansatzes ist, dass das Training verlangsamt wird.

BATCHGRÖSSE Das Produkt aus Batchgröße pro Gerät, Gradientakkumulierungsschritten und Anzahl der verwendeten Geräte. Die Batchgröße bedeutet die Anzahl der Beispiele aus dem Datensatz, für welche die Verlustfunktion berechnet werden, bevor das Netzwerk durch einen Optimierungsalgorithmus angepasst wird.

EPOCHEN Die Anzahl der Durchläufe durch alle Trainingsbeispiele, um die Gewichte im Netzwerk zu optimieren.

OPTIMIERUNGSALGORITHMUS Ein Optimierungsalgorithmus mit welchem die Gewichte angepasst werden, um das Ergebnis der Verlustfunktion zu optimieren. Die Bibliothek huggingface stellt für das Finetuning von Transformer-Modellen die Algorithmen ADAMW [LH17] und AdaFactor [SS18] zur Verfügung [Opt].

LERNRATE Die Rate mit der die Gewichte im neuronalen Netz angepasst werden. Die Lernrate ist ein Parameter des Optimierungsalgorithmus.

SEED Seedwert für das Training zum Zweck der Reproduzierbarkeit

Wie in Kapitel 5.2 angesprochen, wird das multilinguale T5-Modell verwendet, um ein Training durchzuführen. Das Basismodell wurde ausgewählt, da es mit den verfügbaren Ressourcen in einem Google Colaboratory[9] Umfeld trainiert werden kann.

Das Trainieren von Modellen ist mit dem Nutzen gleicher Datensätze und dem verwenden gleicher Hyperparameter reproduzierbar. Diese sind im beigelegten Repository als JSON-Dateien hinterlegt im Ordner *2_training/configs*. Das Trainingsskript *run_text2text.py* im Verzeichnis *2_training* kann zum

9 Google Colaboratory ist eine Cloudlösung von Google, um Python Code im Browser auszuführen. Außerdem kann dadurch Hardware, die für das Trainieren von Modellen essentiell ist genutzt werden, wie beispielsweise Grafikkarten. https://colab.research.google.com/ Letzer Aufruf 15.06.2022

Training verwendet werden und ist eine angepasste Version des Trainingsskriptes von Murakhovs'ka u. a. [Mur+21]. Ein Beispiel für die Ausführung des Trainingsskripts wird im Jupyter Notebook *mT5_training_text2text.ipynb* gezeigt.

Es werden verschiedene Strategien zum Training genutzt. Die Experimente sind nach dem Schema M-Nummer benannt, die im Folgenden beschrieben werden:

M-1 FINETUNING GERMANQUAD Initial wird das mT5-Modell mit dem GermanQuAD-Datensatz [MRP21] trainiert. Als Trainings- und Testdaten wurden die GermanQuAD Daten in ursprünglicher Aufteilung verwendet (siehe Tabelle 3.4).

M-2 FINETUNING SQUAD Das mT5-Modell wird mit dem SQuAD-Datensatz trainiert und es wird versucht, Fragen in deutscher Sprache zu generieren. Im Paper des mT5-Modells wird dieser Ansatz als *cross-lingual-zero-shot-transfer* beschrieben und konnte bei der Fragenbeantwortung für die deutsche Sprache verwendet werden (vgl. [Xue+20, Tabelle 10]). Die Hyperparameter des bereits trainierten Modells von Patil wurden getestet, um das Modell in der multilingualen T5 Variante zu trainieren [Pat20].

M-3 FINETUNING ÜBERSETZTES SQUAD Die maschinell übersetzte Version von SQuAD aus der MLQA Arbeit [Lew+19] wurde zum Training verwendet. Das dient dem Zweck, nachvollziehen zu können, wie gut die maschinell übersetzten Daten bei einer Evaluierung abschneiden, im Vergleich zu den Daten, die durch Crowdworker entstanden sind (GermanQuAD und SQuAD).

M-4 TRANSFER VON SQUAD MODELL Ein mT5-Modell wurde im ersten Schritt mit den SQuAD Daten trainiert (M-2). Das trainierte SQuAD Modell wurde im nächsten Schritt für ein weiteres Finetuning für GermanQuAD verwendet. Dieser Ansatz wurde bei der Evaluation des GermanQuAD-Datensatzes bei einem anderen Transformer-Modell verwendet für die verwandte Aufgabe der Fragenbeantwortung und wird *Warm-Start* genannt [MRP21, Kapitel 4.2].

M-5 TRANSFER VON ÜBERSETZTEM SQUAD MODELL Der Ansatz von M-4 wurde umgesetzt, nur mit dem übersetzten SQuAD-Datensatz anstatt dem englischsprachigen SQuAD (M-3).

M-6 KOMBINATION GERMANQUAD, SQUAD Bei diesem Ansatz wurden die Datensätze SQuAD und GermanQuAD kombiniert und das Modell mit diesen trainiert.

M-7 KOMBINATION GERMANQUAD, ÜBERSETZTES SQUAD Der gleiche Ansatz wie bei Modell M-6 wurde mit der übersetzten SQuAD Version trainiert.

M-8 MULTITASK FRAGENGENERIERUNG In der Arbeit von Akyon u. a. wird ein Multitask-Modell vorgestellt für die Aufgaben der Fragengenerierung mit Wissen der Antwort, Fragenbeantwortung und Antwortextraktion [Aky+21]. Es wurde versucht, das Modell mit dem GermanQuAD-Datensatz zu erstellen. Die Kombination aus Antwortextraktion und Fragengenerierung mit Wissen der Antwort ist dabei ähnlich zur Fragengenerierung ohne Wissen der Antwort.

M-9 PIPELINE MIT ÜBERSETZUNGSMODELLEN Die Pipeline wird in Kapitel 5.3.4 beschrieben.

Beim Modell von Akyon u. a. wird in einem Multitask-Modell zusätzlich die Aufgabe der Extraktion von Antworten aus einem Absatz trainiert [Aky+21]. In dieser Arbeit ist beim Testen aufgefallen, dass ein trainiertes Modell die Antwortextraktion nicht nach dem Training mit dem GermanQuAD-Datensatz umsetzen konnte. Deshalb wird auf eine Auswertung des Experiments M-8 verzichtet [Aky+21].

Die Modelle werden mit automatischen Metriken verglichen, da eine manuelle Auswertung aller Modelle im Rahmen dieser Arbeit zu aufwändig ist. Der Ansatz mit den höchsten automatisch gemessenen Werten in den meisten Metriken wird schließlich mit Webtexten manuell ausgewertet.

Batchgröße pro Gerät und Gradientakkumulierungsschritte wurden beim Training so gewählt, dass der Arbeitsspeicher der Grafikkarte (Video RAM) voll ausgelastet wurde. Eine Batchgröße von 64 Beispielen pro Batch wurde bei jedem Modell für die Fragengenerierung ohne Wissen der Antwort verwendet.

Zusätzlich wurde ein Modell für die Generierung unter Wissen der Antwort trainiert:

M-10 FRAGENGENERIERUNG MIT WISSEN DER ANTWORT GERMANQUAD Nach dem Vorbild der Arbeiten, welche die Antwort im Kontext markieren, wurde ebenfalls ein Modell mit GermanQuAD trainiert. Dadurch, dass bei diesem Ansatz mehr Trainingsbeispiele (Pro Absatz wird eine Frage generiert anstatt mehrere Fragen) vorhanden sind, wurde auf eine Kombination mit anderen Datensätzen verzichtet.

5.3.4 *Pipeline mit Übersetzungsmodellen*

Da die öffentlich verfügbaren Modelle zur Fragengenerierung bereits in englischer Sprache vorhanden sind, bietet sich eine Kombination von Übersetzungsmodellen mit einem Modell zur Fragengenerierung für englische Sprache an.

Innerhalb des SMEBT-Projektes wird dazu eine Pipeline aus verschiedenen Transformer-Modellen genutzt:

1. Übersetzung Deutsch zu Englisch mit einem Übersetzungmodell[10]

10 https://huggingface.co/Helsinki-NLP/opus-mt-de-en Letzter Aufruf 15.06.2022

2. Fragengenerierung mit dem Fragengenerierungsmodell[11] ohne Wissen der Antwort von Patil [Pat20]

3. Übersetzung Englisch zu Deutsch mit dem T5-base Modell[12] [Raf+19]

Die Pipeline wird beim Experiment komplett durchlaufen, um deutschsprachige Fragen zu generieren.

5.3.5 *Automatische Evaluation*

Die Experimente werden zuerst mit automatischen Metriken aus Kapitel 2.3.3 verglichen. Als Vergleich der generierten Texte für die Fragengenerierung ohne Wissen der Antwort zwischen Modelhypothese und Goldreferenz werden BLEU-1 bis BLEU-4, ROUGE-L und METEOR verwendet, wie in verwandten Arbeiten [Lop+20]. Beim Vergleich der generierten Texte für die Fragengenerierung mit Wissen der Antwort werden zusätzlich die Metriken BLEU und BERTScore erhoben, da diese in verwandten Arbeiten verwendet wurden (siehe Kapitel 3.2). Tabelle 5.4 zeigt die verwendeten Implementierung für die verschiedenen Metriken.

Metrik	Vergleichsebene	Wertebereich	Implementierung	Referenz
BLEU	N-Gramme	0-100	SacreBLEU	[Pos18]
ROUGE-L	Längste gemeinsame Teilsequenz	0-1	ROUGE-L huggingface datasets (Google Research Github Repository)	[Huga; Good]
METEOR 1.5	Unigramme	0-1	Implementierung der Autoren	[BL05; DL14]
BertScore	Kontexteinbettungen der Tokens	0-1	Implementierung der Autoren	[Zha+19]

Tabelle 5.4: Implementierungen der verwendeten Metriken.

Als Evaluationsdatensatz wird der englischsprachige und deutschsprachige Anteil von XQuAD verwendet. XQuAD bietet sich an, da der Datemsatz eine Teilmenge des SQuAD Testdatensatzes ist, welcher professionell in die deutsche Sprache übersetzt wurde.

Als Alternative wurde der MLQA Datensatz betrachtet. Dieser scheint aber für die Fragengenerierung ohne Wissen der Antwort weniger geeignet, da im Vergleich zu SQuAD bzw. XQuAD weniger Fragen pro Absatz vorhanden sind. MLQA besitzt durchschnittlich ca. eine Frage pro Absatz (4509 Absätze und 5029 Frage-Antwort-Paare) [Lew+19], während XQuAD ca. fünf Fragen pro Absatz besitzt (240 Absätze und 1190 Frage-Antwort-Paare) [ARY19]. GermanQuAD hat eine ähnliche Anzahl von Fragen pro Absatz wie SQuAD und XQuAD (ca. fünf Fragen pro Absatz).

11 https://huggingface.co/valhalla/t5-base-e2e-qg Letzter Aufruf 15.06.2022
12 https://huggingface.co/t5-base Letzter Aufruf 15.06.2022

Die Modellausgabe wird vor dem berechnen der Metriken nachbearbeitet, indem die künstlichen Trennzeichen (<sep>) aus dem generierten Text entfernt werden bei der Fragengenerierung ohne Wissen der Antwort. Ein Beispiel der Verwendung des Trennzeichens ist in Tabelle 5.3 zu finden.

Ein weiterer Punkt, der bei der Generierung der Texte relevant ist, sind die Dekodierungsparameter, da diese die Ergebnisse der Evaluierung maßgeblich beeinflussen. Die Dekodierungsparameter sind für die zwei Arten der Fragengenerierung jeweils gleich für alle Modelle.

Die Modelle wurden mit einem Jupyter Notebook evaluiert. Das Notebook befindet sich im Ordner *3_evaluation* unter dem Namen *multiple_model_evaluation.ipynb*. Im Notebook sind die jeweiligen Dekodierungsparameter am Anfang des Notebooks angegeben, sie basieren auf der Arbeit von Patil [Pat20]. Die Übersetzungspipeline wurde in einem seperaten Notebook evaluiert, welches im gleichen Ordner vorliegt (*translation_pipeline_evaluation.ipynb*). Es werden die gleichen Implementierungen und Konfigurationen für die Metriken in beiden Notebooks genutzt.

Die Konfigurationen der Metriken befindet sich ebenfalls in besagten Jupyter Notebooks. Es wurde darauf geachtet dass die Konfiguration passend zur ausgewerteten Sprache ist (Englisch oder Deutsch).

Für die SacreBLEU Implementierung wird die Option *lowercase* aktiviert, damit mehr Überschneidungen erkannt werden können.

Bei der ROUGE-L Metrik werden Umlaute angepasst, da der Tokenizer Wörter mit Umlauten ansonsten aufteilt. Außerdem wird die Stammformbildung durch einen Stemmer für die deutsche Sprache deaktiviert, da ein spezifischer Stemmingalgorithmus für die englische Sprache verwendet wird.

Für METEOR und BERTScore wird die verwendete Sprache als Parameter übergeben, um passende Konfigurationen zu nutzen.

Beim BERTScore wird zusätzlich die Option „Baseline Rescaling“ verwendet, damit Werte zwischen 0 und 1 liegen, anstatt im Wertebereich der Kosinusähnlichkeit (-1 bis 1) [Zha+19].

Alle Ergebnisse werden mit weiteren Teilergebnissen im Ordner *3_evaluation/results* zur Verfügung gestellt.

5.3.6 *Manuelle Evaluation*

Die manuelle Evaluation wird mit dem Ansatz der Experimente zur Fragengenerierung ohne Antwort umgesetzt, der die höchsten automatischen Metriken auf dem XQuAD-Datensatz erzielt. Dazu liegt im Verzeichnis *4_inference* das Skript *question_generator.py* mit welchem Fragen generiert werden können, wenn von einer Webseite Textblöcke heruntergeladen wurden mit dem Webcrawler in Ordner *0_webcrawler*. Außerdem muss ein vortrainiertes Modell angegeben werden oder die Pipeline mit den Übersetzungsmodellen.

Das Skript produziert eine Tabellenkalkulationsblatt (Excel-Format) mit den generierten Fragen und den dazugehörigen Textblöcken. Außerdem können die Fragen des Fragebogens (Kapitel 4.4) in jeweils einer Spalte durch den Gutachter binär klassifiziert werden.

Unterstützend wurden die Fragen auf Rechtschreib- oder Grammatikfehler überprüft, mit der quelloffenen Bibliothek namens *Languagetool*[13]. Nach gleichem Vorgehen können auch Fragen mit Wissen der Antwort generiert werden.

5.4 TRAINING ZUR GENERIERUNG VON PARAPHRASEN

Zur Paraphrasengenerierung wird ebenfalls ein mT5-Modell trainiert. Das Training verläuft dabei gleich zu dem der Fragengenerierung aufgrund des Text-zu-Text Frameworks des mT5-Modells. Lediglich Hyperparameter und Datensatz müssen angepasst werden. Als Datensatz wird der deutschsprachige Anteil des PAWS-X-Datensatzes verwendet, der ursprünglich für die Aufgabe der Paraphrasenidentifizierung entwickelt wurde. Dabei enthält PAWS-X-Satzpaare, die binär Klassifiziert wurden, je nachdem ob ein Satzpaar eine Paraphrase ist [Yan+19]. Satzpaare, die eine Paraphrase bilden, werden zum Training des Modells genutzt. Dieser Ansatz folgt den Beispielen des T5-Modell-Trainings in englischer Sprache, allerdings haben Datensätze im Englischen im Vergleich mehr Beispiele (siehe Kapitel 3.3).

Die Konfiguration zum Training kann im Ordner *2_training/configs/mt5-paraphraser.json* gefunden werden. Die Eingabe- und Ausgabelänge wurde an den Datensatz angepasst, da hauptsächlich Satzpaare vorhanden sind und diese kleiner sind als Absätze.

Ein weiterer Ansatz, um Paraphrasen zu generieren ist das Nutzen von Modellen zur maschinellen Übersetzung als sogenannte Zurückübersetzung (siehe Kapitel 3.3). Dieser Ansatz wird im Vergleich zum Ansatz mit einem trainierten Modell ebenfalls getestet.

Um die Ähnlichkeit der generierten Sätze mit der Eingabe zu vergleichen, wird zusätzlich ein multilinguales Universal Sentence Encoder Modell (kurz USE) genutzt [Chi+19].

Die Ähnlichkeit der Vektoren wird dabei mit der Kosinusähnlichkeit (siehe Kapitel 2.2.2) der aus dem USE Modell erhaltenen Vektoren von Eingabe- und Ausgabesatz berechnet. Es wird eine Implementierung der spaCy-Bibliothek genutzt, um den Vergleich zu automatisieren [Hon+20; Men20].

5.5 ZUSAMMENFASSUNG DER EXPERIMENTE

In den Experimenten wurden verschiedene Textgenerierungsansätze getestet für die Aufgabe der Trainingsdatenergänzung von Chatbots. Der Fokus wurde auf die Fragengenerierung ohne Wissen der Antwort gelegt, für diese Art von Textgenerierung wurden acht verschiedene Modelle trainiert, mit verschiedenen Datensatzkombinationen.

Die Hyperparameter wurden dabei teilweise angepasst, alle Anpassungen sind im Programmcode (*2_training/configs*) hinterlegt. Die Fragengenerierungsmodelle ohne Wissen der Antwort werden mit einer Übersetzung-

13 https://github.com/languagetool-org/languagetool Letzter Aufruf 25.06.2022

pipeline verglichen, die ein bereits existierendes Modell nutzt und Übersetzungsmodelle, um Fragen zu generieren.

Des Weiteren wurde ein Modell für die Fragengenerierung mit Wissen der Antwort trainiert. Der Nachteil bei der Anwendung dieses Modells ist, dass eine Strategie zur Antwortselektion ausgewählt werden muss.

Tabelle 5.5 zeigt die jeweiligen Experimente und die verwendeten Datensätze zur Fragengenerierung.

Modell	Trainingsdaten	Testdaten
M-1	GermanQuAD Training	GermanQuAD Test
M-2	SQuAD Training	SQuAD Test
M-3	Übersetztes SQuAD Training (MLQA)	Übersetztes SQuAD Test (MLQA)
M-4	GermanQuAD Training	GermanQuAD Test
M-5	GermanQuAD Training	GermanQuAD Test
M-6	SQuAD Training, GermanQuAD Training	SQuAD Test, GermanQuAD Test
M-7	Übersetztes SQuAD Training (MLQA), GermanQuAD Training	Übersetztes SQuAD Test (MLQA), GermanQuAD Test
M-8	GermanQuAD Training	GermanQuAD Test
M-10	GermanQuAD Training	GermanQuAD Test

Tabelle 5.5: Aufteilung der Trainings- und Testdaten zum Modelltraining. Die Datensätze werden in Tabelle 3.4 beschrieben.

Die Evaluierung der Fragengenerierung findet mit automatischen Metriken (Kapitel 2.3.3) und manueller Evaluation statt (Kapitel 4.4).

Schließlich wurde ein Modell für die Paraphrasengenerierung trainiert mit dem deutschsprachigen Anteil von PAWS-X [Yan+19]. Das trainierte Modell wird mit Modellen zur Übersetzung verglichen, die eine Zurückübersetzung von Deutsch zu Englisch zu Deutsch ausführen.

Die Paraphrasierung wird manuell evaluiert (siehe Kapitel 4.4).

6 ERGEBNISSE

In diesem Kapitel werden die Ergebnisse der Experimente vorgestellt. Es wird in automatische Metriken und manuelle Evaluation unterteilt. Die manuelle Evaluation wurden im Rahmen dieser Arbeit vom Autor durchgeführt und wird in Form mehrerer Tabellen als Excel-Datei zur Verfügung gestellt. Die Zahlen der manuellen Auswertung geben einen ersten Überblick darüber, wie die Textgenerierungsmodelle abschneiden. Um eine detailliertere Auskunft über die Performanz der Systeme zu bekommen, sind allerdings mehr Gutachter und Beispiele nötig. Eine detaiiertere Auskunft soll in zukünftigen Arbeiten umgesetzt werden. Die Dateien sind im Repository im Ordner *4_inference/results* hinterlegt.

6.1 FRAGENGENERIERUNG OHNE WISSEN DER ANTWORT

Da das Fragengenerierungsmodell ohne Wissen der Antwort von Patil in der Pipeline verwendet wird und ein Transfer ohne weiteres Training (engl. *zero-shot-cross-lingual-transfer*) getestet wird, wurde initial Versucht, ein mT5-Modell zu trainieren, das ähnliche automatische Metriken wie das Modell von Patil erreicht. Dabei wurde ein mT5-Modell mit dem gleichen Vorgehen wie in [Pat20] trainiert[1] und die Lernrate und Epochenanzahl abgeändert. Als Modell für Experiment M-2 wurde ein mT5-Modell mit einer Lernrate von 7e-4 trainiert für 10 Epochen. Mit den Gewichten dieses Modells wurde das Modell in Experiment M-4 trainiert.

Nachträglich wurden zwei weitere mT5-Modelle mit dem SQuAD-Datensatz trainiert. Die Spalte *Hyperparameteränderung* gibt an, welche Änderung vorgenommen wurden. Standardmäßig wurde eine Lernrate (LR) von 1e-4 (0.0001) verwendet und für 20 Epochen trainiert. Dabei wurde eine Batchsize von 64 verwendet und jeweils die base Versionen von T5 oder mT5 genutzt.

1 Die Hyperparameter wurden unter folgendem Link gefunden https://wandb.ai/psuraj/question-generation/runs/1d8gz248/overview Letzter Aufruf 23.06.2022

Modell	Hyper-parameter-änderung	BLEU-Score	ROUGE-L (f-measure)	METEOR-score	BERT Score (f-measure)
mT5 (M-2)	LR 7e-4, 10 Epochen	7.17	0.2865	0.1034	0.2957
mT5	-	7.22	0.2884	0.1063	0.3063
mT5	LR 5e-4, 10 Epochen	7.44	0.2880	0.1061	0.2987
T5 [Pat20]	-	**13.59**	**0.3233**	**0.1413**	**0.3581**

Tabelle 6.1: Ergebnis automatischen Evaluierung der Modelle mit englischsprachigem Anteil des XQuAD-Datensatzes für die Fragengenerierung ohne Wissen der Antwort.

Keines der trainierten mT5-Modelle konnte das Modell von Patil in den Metriken erreichen. Es fällt auf, dass die Ergebnisse der trainierten mT5-Modelle nahe beieinander liegen. Ein Training mit T5 funktioniert dennoch besser als mit der multilingualen T5-Version für englischsprachige Daten.

Im nächsten Schritt wurden die Modelle der Fragengenerierung für die deutsche Sprache evaluiert. Zur Generierung der Fragen wurden die gleichen Dekodierungsparameter wie bei der englischsprachigen Evaluation verwendet. Die Dekodierungsparameter sind im Jupyter Notebook *multiple_model_evaluation.ipynb* hinterlegt.

Folgende Ergebnisse wurden bei der automatischen Evaluierung gemessen:

Modell	BLEU-Score	ROUGE-L (f-measure)	METEOR-score	BERT Score
M-1	1.73	0.1711	0.0835	0.3319
M-2	0.83	0.1016	0.0348	0.2244
M-3	*4.18*	0.1986	*0.1132*	0.3347
M-4	1.98	0.1807	0.0912	0.3388
M-5	2.29	0.1899	0.0942	0.3408
M-6	2.10	0.1828	0.0909	*0.3449*
M-7	3.81	*0.2014*	0.1119	0.3379
M-9	**7.5**	**0.2185**	**0.1447**	**0.3481**

Tabelle 6.2: Ergebnis automatischen Evaluierung der Modelle mit deutschsprachigen Anteil des XQuAD-Datensatzes für die Fragengenerierung ohne Wissen der Antwort. Textstilanpassungen für die Ergebnisse: **Bestes Ergebnis**, *Bestes Ergebnis trainierter Modelle*.

Die Werte der automatischen Metriken (Tabelle 6.2) zeigen nah beieinanderliegende Ergebnisse in den meisten Metriken, wobei die Fragengenerierung mithilfe des englischsprachigen Modells und Modellen zur Übersetzung in allen Metriken am besten abschneidet.

Die anderen Ansätze sind in ROUGE-L, METEOR und BERT-Score ähnlich von den gemessenen Werten. Bei Experiment M-2 wurde ausschließlich mit dem englischsprachigen Datensatz SQuAD trainiert. Dabei wurde versucht, ein Transfer ohne deutschsprachige Daten umzusetzen (engl. *zero-shot-cross-lingual-transfer*). Allerdings wurden bei diesem Experiment in allen Metriken die niedrigsten Werte gemessen. Es sind deutliche Unterschiede in den automatischen Metriken zu sehen im Vergleich zu den anderen Modellen, die mit deutschsprachigen Daten trainiert wurden (M-1, M-3, M-4, M-5, M-6, M-7, M-9).

Von den trainierten mT5-Modellen schneidet das Experiment M-3 am besten in BLEU- und METEOR-Werten ab, mit der maschinell übersetzten Version von SQuAD. Die Kombination des englischsprachigen SQuAD-Datensatzes und GermanQuAD bringt ebenfalls nur eine leichte Verbesserung in der Evaluation, der BERTScore ist bei diesem Experiment (M-6) allerdings am höchsten von den trainierten Modellen. Außerdem konnte das Modell M-6 ähnliche Werte zu den Modellen erzielen, die nur mit deutschsprachigen Daten trainiert wurden. Schließlich ist die Kombination vom übersetzten SQuAD-Datensatz und dem GermanQuAD-Datensatz in der ROUGE-L Metrik von allen trainierten Modellen am höchsten (M-7).

Dadurch, dass die Pipeline die besten Werte in der automatischen Evaluation erzielen konnte, wurde diese manuell ausgewertet.

Im Folgenden werden die Ergebnisse der manuellen Evaluation der Fragengenerierung dargestellt. Das Vorgehen dazu wurde in Kapitel 4.4 beschrieben. Die selben Dekodierungsparameter zur Generierung der Texte für die automatischen Evaluation wurden auch beim Generieren der Fragen für die manuelle Evaluierung verwendet.

Seite (Fragen) / Score	KMI (48)	DGE (50)	DGE FAQ (50)	RKI (50)	RKI FAQ (50)	*Alle*
Grammatikalisch falsch oder keine Frage	8	**14**	10	7	3	*42*
Grammatikalisch korrekt	7	10	**13**	3	11	*44*
Beantwortbar	23	24	26	**40**	35	*148*
Mögliche Chatbotfrage	**10**	2	1	0	1	*14*

Tabelle 6.3: Ergebnis der manuellen Evaluierung für Fragengenerierung ohne Wissen der Antwort.

Die Auswertung von 248 Fragen zeigt, dass 42 Fragen (ca. 16.94 %) als grammatikalisch falsch eingestuft wurden oder als keine Frage. 44 Fragen (17.74 %) wurden als grammatikalisch korrekt, aber nicht beantwortbar eingestuft. Der Großteil der Fragen 148 (59.68 %) wurde als grammatikalisch richtig und beantwortbar eingestuft. Allerdings wurden nur 14 (5.64 %) der Fragen als passend für einen Chatbot eingestuft.

6.2 FRAGENGENERIERUNG MIT WISSEN DER ANTWORT

Als eine weitere Möglichkeit der Fragengenerierung wurde die Generierung unter Wissen der Antwort getestet. Dabei wurde ein Modell mit dem GermanQuAD-Datensatz trainiert. Ein Vorteil dieses Ansatzes ist, dass das Mapping von Absatz und markierter Antwort immer nur eine resultierende Frage hat und deshalb die Trainingsdaten effektiver genutzt werden können, da die gegebenen Antwortpassagen genutzt werden können und mehr Beispiele zum Training vorhanden sind. Die Trainingsbeispiele sind bei diesem Ansatz gleich der Anzahl der Frage-Antwort-Paare. Im Gegensatz dazu ist bei der Fragengenerierung ohne Wissen der Antwort die Beispielanzahl gleich der einzelnen Absätze. Durch das Markieren der Antwort ergibt sich ein neues Problem, das gelöst werden muss.

Im Fall dieses Experiments werden im Absatz alle benannten Entitäten markiert und als Eingabe für das Modell verwendet. Dazu wurde ein deutschsprachiges NER-Modell genutzt aus dem flair-Framework [Akb+19] in der Version *large* [SA20].

Das trainierte Fragengenerierungs Modell wurde ebenfalls auf den deutschsprachigen XQuAD-Daten evaluiert. Dabei wurden die Dekodierungsparameter gesetzt nach dem Vorbild der Arbeit von [Pat20]. Die maximale Länge der Sequenz ist bei dieser Art von Fragengenerierung reduziert, da nur ein Fragesatz erzeugt wird. Andere Optionen, welche in der Fragengenerierung ohne Wissen der Antwort genutzt werden, fallen aus diesem Grund weg.

Für dieses Experiment wurden folgende automatische Metriken gemessen: BLEU: 8.93, ROUGE-L: 0.2657, METEOR: 0.1491, BERTScore: 0.4698.

Bei der Evaluation konnten im Vergleich zur Fragengenerierung ohne Wissen der Antwort bessere Werte in den automatischen Metriken erzielt werden. Das liegt unter anderem daran, dass bei diesem Ansatz die Antwort aus dem Datensatz genutzt wird, um Fragen zu generieren. Zusätzlich zum Absatz als Eingabetext wird die Antwort zur Frage im Text markiert und daraus eine Frage generiert.

Zur Auswertung in Verbindung mit Webtexten wurde zum einen versucht, nur den Text von Entitäten zu markieren, wie beispielsweise *KMI*, zum anderen wurde der Satz markiert, in welchem sich die Entität befindet. Dadurch, dass mehrere Entitäten in einem Satz vorhanden sein können, ergab der zweite Ansatz weniger Fragen. Es wurden die gleichen Bewertungskriterien wie bei der Fragengenerierung ohne Wissen der Antwort verwendet.

Schließlich wurde nur die KMI Seite für diese Evaluation verwendet, da diese die besten Ergebnisse im Bezug auf potentielle Chatbotfragen liefern konnte. Tabelle 6.4 zeigt die Ergebnisse der manuellen Auswertung.

Seite (Fragen) / Score	KMI Entität Text markiert (50)	KMI Satz mit Entität markiert (36)
Grammatikalisch falsch oder keine Frage	5	4
Grammatikalisch korrekt	26	22
Beantwortbar	19	7
Mögliche Chatbotfrage	0	3

Tabelle 6.4: Ergebnis Fragengenerierung mit Wissen der Antwort der manuellen Evaluierung

Mit diesem Ansatz konnte keine Verbesserung hinsichtlich der möglichen Chatbotfragen erzielt werden. Die Ergebnisse der Fragengenerierung ohne Wissen der Antwort der KMI Seite können mit diesen Daten verglichen werden, da ungefähr eine gleichgroße Stichprobe an Fragen vorhanden ist. Im Vergleich zur Fragengenerierung ohne Wissen der Antwort auf der KMI Seite, wurde beim Markieren der Antwort etwas seltener grammatikalisch falsche oder keine Fragen gefunden. Dafür sind im Vergleich die generierten Fragen bei der Fragengenerierung ohne Wissen der Antwort öfter beantwortbar als bei der Fragengenerierung mit Wissen der Antwort. Das Markieren des Satzes mit einer Entität schneidet bei dieser Messung am schlechtesten im Punkt Beantwortbarkeit ab. Die möglichen Chatbotfragen der Fragengenerierung ohne Wissen der Antwort konnten nicht übertroffen werden bzw. konnten beim Ansatz der Markierung der Entität im Text keine möglichen Chatbotfragen generiert werden.l

Schließlich fiel außerdem eine Verzerrung im Datensatz bei der Markierung der Texte der Entitäten auf, 36 der 50 generierten Fragen starteten mit „Wie heißt“. Das deutet darauf hin, dass kurze Antworten oft in „Wie heißt“ Fragen resultieren im GermanQuAD-Datensatz.

6.3 PARAPHRASIERUNG

Um die Paraphrasierungsansätze zu vergleichen, wurde ein mT5-Modell mit dem PAWS-X-Datensatz bzw. dem deutschsprachigen Anteil des Datensatzes trainiert [Yan+19]. Es wurden fünf Fragen aus der Fragengenerierung übernommen, die als potentielle Chatbotfrage eingestuft wurden. Pro Frage wurden fünf weitere Möglichkeiten mit dem trainierten Modell erzeugt. Die Zahl fünf wurde gewählt, da für das Chatbotsystem IBM Watson mindestens fünf Beispiele pro Intention empfohlen werden[2] [Ibm]. Dabei wurde nach den Bewertungskriterien aus Kapitel 4.4 evaluiert.

Die Dekodierungsparameter zum Generieren der Paraphrasen werden im Jupyter Notebook *paraphrase_generator.py* im Ordner *4_inference* angegeben. Die Anzahl der Beams, sowie die Anzahl der Rückgabesequenzen ist dabei

2 https://cloud.ibm.com/docs/assistant?topic=assistant-intents Letzter Aufruf

fünf, damit die fünf wahrscheinlichsten Paraphrasen pro Eingabe zurückgegeben werden können.

Tabelle 6.5 zeigt die Einordnung der 25 Paraphrasen.

Keine exakte Kopie	Grammatikalisch korrekt	Ähnliche Bedeutung	Lexikalische Änderung	Syntaktische Änderung
24	10	6	4	2

Tabelle 6.5: Ergebnis der manuellen Auswertung der Paraphrasierung mit mT5-Modell.

Bei der manuellen Auswertung fällt zudem auf, dass bei 15 der 25 Fragen nur Zeichen am Satzende verändert wurden (z. B. Punkt anstatt Fragezeichen) und bei 7 von 25 Fragen das Fragewort durch einen Artikel, ein Personalpronomen oder eine Subjunktion ausgetauscht wurde (z. B. Das anstatt Was).

Die durchschnittliche Ähnlichkeit der Eingabe- und Ausgabesätze ist bei diesem Ansatz 0.8308, identische Sätze (Wert von 1) wurden nicht berücksichtigt beim Ausrechnen des Durchschnitts.

Als zweite Option wurde die Methode der Zurückübersetzung getestet, wie sie in [Ber+21; FEQ19; Cha20; TP20] vorgeschlagen bzw. verwendet wird. Dabei wurden die Modelle der Pipeline verwendet, die eine Zurückübersetzung von Deutsch zu Englisch zu Deutsch umsetzt (Ansatz M-9, siehe Kapitel 5.3.4). Dabei wurden die gleichen fünf Beispiele für eine Paraphrasierung genutzt und fünf Paraphrasen pro Beispiel verwendet mit einer Beamgröße von fünf. Tabelle 6.6 zeigt die Ergebnisse der Zurückübersetzung der 25 Paraphrasen.

Keine exakte Kopie	Grammatikalisch korrekt	Ähnliche Bedeutung	Lexikalische Änderung	Syntaktische Änderung
23	20	17	18	3

Tabelle 6.6: Ergebnis der manuellen Auswertung der Paraphrasierung mit Zurückübersetzung.

Im Vergleich zur Methode mit einem trainierten mT5-Modell auf dem deutschsprachigen Anteil von PAWS-X konnte durch die Zurückübersetzung etwas bessere Ergebnisse erzielt werden, wobei die Anzahl der Paraphrasen mit Änderungen am Satzbau ähnlich gering sind.

Die berechnete durchschnittliche Ähnlichkeit durch das USE-Modell ist bei diesem Ansatz etwas höher als beim ersten Ansatz mit 0.8868.

Ein Problem bei der Zurückübersetzung ist, dass für Abkürzungen, wie KMI (Kommunikation und Medieninformatik), unpassende Artikel verwendet werden: „Wann findet die Einführung des KMI ins erste Semester statt?“ der richtige Artikel wäre „der“: „Wann findet die Einführung der KMI ins erste Semester statt?“.

Ein weiteres Problem ist, dass Änderungen die Semantik des Satzes verändern können: „Was muss man machen, um Programmieren zu lernen?“ wird nach einer Zurückübersetzung zu „Was musst du tun, um zu programmieren?“. Diese Änderung verändert die Semantik des Satzes, da es nicht mehr um das Lernen der Programmierung geht, sondern um das Programmieren.

7 DISKUSSION

In diesem Kapitel werden die getesteten Ansätze mit verwandten Arbeiten verglichen. Die Ergebnisse dieser Arbeit sind unter anderem die automatische sowie manuelle Auswertung der Fragengenerierung und die manuelle Auswertung der Paraphrasierung. Es wird versucht, diese Ergebnisse in Relation zu verwandten Arbeiten zu setzen (siehe Kapitel 3.2 und 3.3).

Des Weiteren werden die meisten verwandten Arbeiten im englischsprachigen Raum umgesetzt, dadurch ist ein Vergleich ungenauer, da die Ergebnisse dieser Arbeit in erster Linie in der deutschen Sprache vorhanden sind.

7.1 VERGLEICH DER FRAGENGENERIERUNG

In Kapitel 3.2 wurden verwandte Ansätze zur Fragengenerierung vorgestellt.

Zuerst findet ein Vergleich der manuellen Evaluationen von verwandten Arbeiten mit dieser Arbeit statt. Regelbasierte Ansätze wie der von Heilman und Smith in englischer Sprache oder der Ansatz von Kolditz in deutscher Sprache nutzen Sätze als Eingabe, um daraus Fragen zu generieren [HS10; Kol15]. Beide Ansätze berichten, dass ca. die Hälfte der generierten Fragen als akzeptabel eingeordnet werden konnten (Kolditz 44 %, Heilman und Smith 52 % nach einer Sortierungfunktion) [HS10; Kol15].

Im Rahmen dieser Arbeit wurde die Generierung von Fragen ausschließlich auf Absatzebene getestet, deshalb ist ein direkter Vergleich mit diesen beiden Arbeiten ungenau. Außerdem wurden verschiedene Bewertungsbögen für eine manuelle Evaluation verwendet, die allerdings in den Punkten grammatikalische Korrektheit und Beantwortbarkeit der generierten Frage Überschneidungen haben.

In dieser Arbeit wurden ca. 60 % der Fragen als grammatikalisch korrekt und beantwortbar durch die Eingabe bewertet, bei Verwendung der Fragengenerierung ohne Wissen der Antwort. Allerdings wurden nur ca. 5.6 % der Fragen als potentiell nutzbar für einen Chatbot bewertet.

Bei der Fragengenerierung unter Wissen der Antwort war der Anteil von grammatikalisch korrekten Fragen, die beantwortbar waren, deutlich schlechter mit 38% beim Markieren des Textes einer benannten Entität und 20% bei der Markierung eines Satzes mit benannten Entitäten. Der Ansatz des Markierens von benannten Entitäten als Antwort war ausschlaggebend für diese Ergebnisse, eine andere Form der automatischen Antwortselektierung aus einem Satz oder Absatz wurde nicht getestet.

Ein genauer Vergleich zwischen der Fragengenerierung ohne Wissen der Antwort ist nicht möglich, da eine Generierung auf Satzebene nötig wäre, sowie die Berücksichtigung anderer Faktoren, die in den Arbeiten von Heilman und Smith und Kolditz genannt werden [HS10; Kol15].

Um einen genaueren Vergleich zu erhalten, müsste überprüft werden, ob der Ansatz mit einem transformerbasierten Sprachmodell für die Fragengenerierung auf Satzebene bessere Performanz erreicht im Vergleich zu den regelbasierten Systemen. Ein Hinderniss für diese Überprüfung ist allerdings, dass kein Datensatz für das Training von Satz zu Frage in deutscher Sprache verfügbar ist zum aktuellen Zeitpunkt.

In der Arbeit von De Kuthy u. a. wurde ein ähnliches Vorgehen umgesetzt, allerdings mit einem RNN-basierten neuronalen Netz [DK+20]. Dabei nutzten die Autoren die Arbeit von Kolditz, um einen Datensatz mit Satz, Frage und Antwort Beispielen zu erstellen [DK+20]. De Kuthy u. a. berichten bei ihrer manuellen Evaluation eine Verbesserung des regelbasierten Ansatzes von Kolditz mithilfe des beschriebenen neuronalen Ansatzes [DK+20]. Außerdem nutzen De Kuthy u. a. die Metrik BLEU, um die Ähnlichkeit der generierten Fragen zu vergleichen [DK+20]. Dabei erhalten sie deutlich bessere Werte. Das beste Modell erreicht auf einem Testdatensatz ein BLEU-Wert von 84.24 [DK+20]. Im Vergleich dazu waren die BLEU-Werte in dieser Arbeit im einstelligen Bereich für die automatische Evaluierung. Allerdings ist bei dem Vergleich der BLEU-Werte der Unterschied zwischen Satz- und Absatzeingabe zu berücksichtigen. Die Autoren nutzen ebenfalls die SacreBLEU-Implementierung, welche auch in dieser Arbeit verwendet wurde [DK+20].

Die Datensätze der Arbeit von De Kuthy u. a. konnten auf Anfrage leider nicht für diese Arbeit genutzt werden, weshalb ein genauer Vergleich nicht möglich war, bzw. ein Betrachten dieses Ansatzes.

Durch die Ergebnisse der vorher genannten Arbeiten wäre es allerdings zu vermuten, dass eine Fragegenerierung auf Satzebene besser funktioniert als die genutzte Fragengenerierung ohne Wissen der Antwort aus einem Absatz [HS10; Kol15; DK+20]. Eine Betrachtung zur Erstellung von Chatbotdaten wäre dabei ebenfalls interessant.

In diesem Teil wird die Fragengenerierung ohne Wissen der Antwort mit verwandten Arbeiten verglichen. In der Arbeit von Lopez u. a. wurden die Teilergebnisse der Metrik BLEU (BLEU-1 bis BLEU-4) berichtet, sowie Ergebnisse der Metriken METEOR und ROUGE-L [Lop+20]. Der Ansatz *Alle Fragen pro Zeile* wurde in ähnlicher Form in dieser Arbeit mit dem multilingualen T5-Modell verwendet. Es werden pro Metrik jeweils etwas höhere Werte berichtet, wobei zu beachten ist, dass Lopez andere Implementierungen der Metriken nutzt, als die, die in dieser Arbeit verwendet wurden und die Fragen bei Lopez u. a. in englischer Sprache generiert werden [Lop+20].

Die höchsten Werte für BLEU-1 bis BLEU-4 konnten mit dem trainierten Modell M-6 erzielt werden, weshalb dieses zusätzlich betrachtet wird. Tabelle 7.1 gibt eine Übersicht der Metriken der Fragengenerierung ohne Wissen der Antwort von dieser und verwandten Arbeiten.

Modell / Metriken	Evaluationsdatensatz	B1	B2	B3	B4	R-L	M
Pipeline	XQuAD (deutsch)	41.17	14.84	7.06	3.78	0.2187	0.1447
M-6	XQuAD (deutsch)	51.53	19.87	9.3	5.04	0.1829	0.0909
Patil [Pat20]	XQuAD (englisch)	52.84	24.95	13.92	**8.25**	0.3233	0.1413
Lopez u. a. [Lop+20]	SQuAD Test	**54.83**	**30.13**	**15.72**	7.31	**0.4388**	**0.2053**

Tabelle 7.1: Vergleich Fragengenerierung ohne Wissen der Antwort und Experimente dieser Arbeit. Abkürzungen: B = BLEU, R = ROUGE, M = METEOR.

Das Fragengenerierungsmodell ohne Wissen der Antwort von Patil wurde automatisch evaluiert mit dem englischsprachigen Anteil von XQuAD, da in der Arbeit von Patil keine automatischen Metriken für dieses Modell erhoben wurden [Pat20].

Beim Vergleich zwischen Lopez u. a. und den gemessenen Werten dieser Arbeit ist zu beachten, dass die Nutzung verschiedener Implementierungen der Metriken die verschiedenen Sprachen sowie verschiedene Evaluierungsdatensätze den Vergleich erschweren.

Es fällt auf, dass die BLEU-1 bis BLEU-3-Werte bei Lopez u. a. etwas besser sind als die besten Werte von Modellen dieser Arbeit und dem Modell von Patil. Das Modell von Patil schneidet von allen Modellen am besten im BLEU-4-Wert ab und übertrifft die trainierten Modelle dieser Arbeit mit höheren Werten in den Metriken BLEU-1 bis BLEU-4 und ROUGE-L. Lediglich der METEOR-Wert der Pipeline ist etwas besser als das Modell von Patil.

Die Pipeline ist im Vergleich zu M-6 etwas besser in den Metriken ROUGE-L und METEOR.

Die unterschiedlichen Werte hängen zum einen mit den unterschiedlichen Evaluationsdaten zusammen. Lopez u. a. nutzt den SQuAD-Testdatensatz, in dieser Arbeit wird der XQuAD-Datensatz genutzt, je nach Sprache des Modells.

Zum anderen können die unterschiedlichen Implementierungen der Metriken einen großen Anteil am Ergebnis haben. Des Weiteren ist unklar, ob in der Arbeit von Lopez u. a. die generierten Texte vor dem Ausrechnen der Metriken verändert wurden [Lop+20].

In dieser Arbeit werden vor dem Ausrechnen der Metriken die künstlichen Trennzeichen aus dem generierten Text entfernt (siehe Kapitel 5.3.5).

Unter der Annahme, dass die verschiedenen Implementierungen der Metriken gleiche bis ähnliche Ergebnisse liefern, schneidet das Modell von Lopez u. a. am besten ab in diesem Vergleich. Das Modell wurde nicht veröffentlicht, weshalb ein genauerer Vergleich der Modelle nicht möglich ist.

Im nachfolgenden Abschnitt werden die automatischen Metriken der Fragengenerierungsansätze mit Wissen der Antwort in englischer Sprache mit den Ergebnissen dieser Arbeit verglichen. Dabei wurden die Metriken für

das Experiment M-10 mithilfe des deutschsprachigen Anteils von XQuAD erhoben.

Der jeweilge Evaluationsdatensatz wird in der Tabelle 7.2 ebenfalls erwähnt, wobei darauf geachtet wurde, das die genutzten Evaluationsdaten möglichst ähnlich zu XQuAD sind (Absätze mit mehreren Fragen und extraktiven Antworten).

Modell / Metriken	Evaluationsdatensatz	B	B1	B2	B3	B4	R-L	M	BERT
HL QG Modell (M-10)	XQuAD (deutsch)	8.93	37.53	13.21	7.27	4.33	0.2657	0.1491	**0.4698**
Chan und Fan [CF19]	SQuAD Test	-	**51.54**	**36.45**	**27.96**	**22.17**	**0.4968**	0.248	-
Klein und Nabi [KN19]	SQuAD Test	-	31.36	19.5	12.41	7.84	0.3451	-	-
Lopez u. a. [Lop+20]	SQuAD Test	-	36.07	18.83	10.95	6.4	0.398	-	-
Akyon u. a. [Aky+21, Bestes Modell mT5-large]	XQuAD (türkisch)	-	29.3	21.9	-	-	0.375	-	-
Murakhovs'ka u. a. [Mur+21]	QAConv (englisch)	**16.65**	-	-	-	-	0.3762	**0.3993**	0.4533
Patil [Pat20]	SQuAD Test	-	-	-	-	21.32	0.436	0.2709	-

Tabelle 7.2: Vergleich der Metriken für Fragengenerierung mit Wissen der Antwort und dem Experiment dieser Arbeit. Abkürzungen: B = BLEU, R = ROUGE, M = METEOR,
BERT = BERTScore.

Wie zu sehen, schneidet das Modell von Murakhovs'ka u. a. für die Fragengenerierung mit Wissen der Antwort am besten ab in den Metriken BLEU und METEOR. Lediglich der BERTScore ist etwas besser bei Experiment M-10, das liegt vermutlich daran, dass für die deutsche Sprache ein anderes Modell verwendet wird, um Kontexteinbettungen zu erstellen [Zha+19]. Die Arbeit von Chan und Fan berichtet die höchsten Werte in den Metriken BLEU-1 bis BLEU-4, sowie der ROUGE-L.

Es fällt außerdem auf, dass die Metriken BLEU-2 bis BLEU-4, ROUGE-L und METEOR-Werte des Experiments M-10 im Vergleich zu anderen Modellen schlechter abschneiden.

Wie auch bei der Fragengenerierung ohne Wissen der Antwort ist der direkte Vergleich der Modelle hier wieder ungenau, da verschiedene Implementierungen der Metriken genutzt werden und Datensätze für unterschiedliche Sprachen verwendet werden. Dennoch gibt der Vergleich in Tabelle 7.2 einen groben Überblick darüber, welche Werte für die jeweiligen Metriken bei der Fragengenerierung erreicht werden können.

Abschließend ist zur Fragengenerierung zu sagen, dass die automatischen Metriken bestenfalls eine Orientierungsmöglichkeit sind, um trainierte Modelle miteinander zu vergleichen. Der Vergleich verschwimmt allerdings,

wenn verschiedene Implementierungen der Metriken verwendet werden oder die Evaluation in verschiedenen Sprachen stattfindet. Weitere Faktoren sind auch die vortrainierten Sprachmodelle und deren Größe (in trainierbaren Parametern), die zum Transferlernen verwendet werden.

Die bessere Alternative zur automatisierten Evaluation scheint eine manuelle Evaluation zu sein, beispielsweise mithilfe eines Bewertungsbogens. Durch die manuelle Evaluation kann auch der Anwendungsbereich der jeweiligen Textgenerierung berücksichtigt werden. Außerdem können grammatikalische Korrektheit sowie die Beantwortbarkeit der Fragen aus dem gegebenen Absatz berücksichtigt werden.

7.2 VERGLEICH MIT ANSÄTZEN ZUR PARAPHRASIERUNG

Der Vergleich der Paraphrasengenerierung ist schwieriger, da hier keine automatischen Metriken etabliert sind. Lediglich in Thompson und Post wird BLEU verwendet, um Eingaben und Ausgaben der Paraphrasierung zu vergleichen mit manuellen Auswertungen [TP20]. Anstatt automatische Metriken zu verwenden, werden Eigenschaften des Textes beurteilt, ähnlich zur manuellen Evaluierung der Fragengenerierung. Zwei wichtige Eigenschaften bei der Beurteilung von Paraphrasen scheinen die grammatikalische Korrektheit und die Erhaltung der Semantik zu sein, da diese in verwandten Arbeiten genutzt bzw. erwähnt wurden [TP20; Dam21]. Weitere Eigenschaften sind die lexikalischen Änderung d. h. das Verwenden anderer Wörter wie Synonyme oder syntaktische Änderungen, also das Umstellen eines Satzes [TP20; Dam21]. Bei der manuellen Evaluierung von zwei Paraphrasierungsansätzen in Kapitel 6.3 dieser Arbeit wurden diese vier Eigenschaften beachtet. In vielen der gefundenen Arbeiten findet leider keine Evaluierung der Paraphrasen statt, weshalb hierfür auf die Ergebnisse von Kapitel 6.3 verwiesen wird. Die Arbeit von Thompson und Post wurde nicht getestet. Diese wäre aber ebenfalls für eine manuelle Auswertung interessant, da dort für mehrere Sprachen Paraphrasen generiert werden können mithilfe der Zurückübersetzung [TP20]. Des Weiteren könnten die englischsprachigen Transformermodelle zur Paraphrasierung ebenfalls mit einer Übersetzungspipeline genutzt werden. Eine manuelle Auswertung dieses Ansatzes wäre ebenfalls relevant für zukünftige Arbeiten.

8 SCHLUSSFOLGERUNG

Dieses Kapitel fasst die Ergebnisse der getesteten Ansätze zur Wissengenerierung für Chatbots zusammen. Anschließend werden weitere offene Forschungsfragen in einem Ausblick beschrieben.

8.1 FAZIT

Mithilfe der Experimente dieser Arbeit sollte beantwortet werden, inwiefern Chatbotwissen aus Webseiten erstellt werden kann und durch Textgenerierung erweitert werden kann, um so das Verständnis von natürlicher Sprache von Chatbots automatisiert zu verbessern.

Dabei wurden zwei Ansätze betrachtet, um Trainingsdaten für eine Intentionsklassifizierung von Chatbots zu erhalten.

Der erste Ansatz betrachtete die Fragengenerierung in Kombination mit einem Webcrawling von Texten drei verschiedener Webseiten. Durch die Webtexte sollten Trainingsbeispiele als Frage-Antwort-Paare generiert werden. Die Antwort ist dabei ein Webtext in Absatzgröße und die Frage wurde mithilfe eines Modells generiert.

Ein zweiter Ansatz nutzte die Generierung von Paraphrasen, um bereits bestehende Trainingsdaten für Chatbots zu erweitern und so eine bessere Erkennung von Nutzerintentionen zu erzielen.

Bei der Fragengenerierung wurden zudem automatische Metriken gemessen, die in relevanten Arbeiten erhoben wurden (Kapitel 3.2). Der Vergleich mit den automatischen Metriken aus relevanten Arbeiten hat gezeigt, dass Arbeiten für die englischsprachige Fragengenerierung oft bessere Werte für diese Metriken erzielen konnten (Kapitel 7).

Allerdings ist ein Problem dieses Vergleiches, dass verschiedene Implementierungen von den automatischen Metriken und unterschiedliche Datensätze zur Evaluierung verwendet werden. Auch im Hinblick auf die verwendeten Sprachen ist der Vergleich ungenau.

Des Weiteren vergleichen automatische Metriken die Überschneidungen von Tokens bzw. Wörtern. Die Überschneidung von Wörtern hat allerdings nicht unbedingt eine Aussagekraft darüber, ob eine passende Frage generiert wurde.

Um weitere Informationen zu den generierten Fragen zu erhalten, wurde zusätzlich eine manuelle Evaluation getätigt. Diese zeigte, dass viele Fragen grammatikalisch korrekt und beantwortbar durch den gegebenen Absatz sind. Die meisten generierten Fragen wurden als nicht verwendbar zum Chatbottraining eingestuft. Das liegt daran, dass die Fragen beispielsweise nicht zum Thema der Seite passten oder zu spezifisch gestellt wurden und damit unrealistisch als mögliche Nutzereingabe sind.

Es wurden zwei Fragengenerierungsansätze getestet: Erstens die Fragengenerierung ohne Wissen der Antwort und zweitens die Generierung von Fragen mit Wissen der Antwort.

Die Ergebnisse der manuellen Evaluierung der Fragengenerierung ohne Wissen der Antwort waren dabei höher, wobei zu erwähnen ist, dass der Ansatz zur Fragengenerierung mit Wissen der Antwort nur in Verbindung mit der Markierung benannter Entitäten getestet wurde. Ein anderer Ansatz der automatischen Antwortselektierung aus einem Absatz könnte besser abschneiden.

Bei der Generierung von Paraphrasen konnten verschiedene Methoden gefunden werden, wobei zwei in dieser Arbeit manuell ausgewertet wurden. Die Generierung ist dabei, gleich wie die Fragengenerierung, ebenfalls umsetzbar mit tranformerbasierten Modellen wie mT5 (siehe Kapitel 3.3). Allerdings unterscheiden sich die Datensätze vor allem in der Anzahl von Beispielen zwischen der englischen und deutschen Sprache, weshalb ein Modell mit dem deutschsprachigen Anteil des Datensatzes PAWS-X schlechte Ergebnisse bei einer manuellen Evaluierung erzielte. Eine Alternative zum Training eigener Modelle ist das Verwenden von Modellen für die neuronale maschinelle Übersetzung. Dabei konnten annehmbare Paraphrasierungen in deutscher Sprache erstellt werden, wobei diese meist lexikalische Änderungen hatten.

8.2 AUSBLICK

Aus den Ergebnissen dieser Arbeit ergeben sich neue Fragen speziell zur Generierung von Fragen oder Paraphrasen für die Erweiterung oder Erstellung von Trainingsdaten für Chatbots.

In dieser Arbeit wurde die Fragengenerierung von Textblöcken für die deutsche Sprache getestet und evaluiert, allerdings ist unklar, inwiefern die verfügbaren englischsprachigen Modelle passende Fragen für englischsprachige Chatbots erstellen können. Dabei wäre die Frage, ob englischsprachige Modelle bessere Werte in der manuellen Auswertung erreichen können als Modelle dieser Arbeit. Dadurch könnte auch beantwortet werden, wie viel der Datenmangel zwischen den Sprachen Deutsch und Englisch ausmacht.

Des Weiteren ist ungeklärt, inwiefern der SQuAD-Datensatz für eine Fragengenerierung für Chatbots geeignet ist, da dieser speziell Fragen für das Leseverständnis enthält. Der SQuAD-Datensatz ist aufgrund seiner Qualität und Anzahl an Frage-Antwort-Paaren omnipräsent in den Arbeiten zur Fragengenerierung der letzten Jahre (vgl. Kapitel 3.2), allerdings sind Anwendungsfelder zur Generierung der Fragen durch verwandte Arbeiten nicht beleuchtet.

Zusätzlich könnte erforscht werden, was passende Fragen für einen informativen Chatbot ausmacht im Vergleich zu Leseverständnis-Fragen.

In Experiment M-10 wurde ein Modell mit Wissen der Antwort trainiert. Diese Art von Fragengenerierung schneidet besser ab in den automatisch

gemessenen Metriken. Das liegt daran, dass die Antwort in Datensätzen wie SQuAD ebenfalls als Teilzeichenkette vorhanden ist und bei einer automatischen Evaluation genutzt werden kann. Allerdings muss eine potentielle Antwort bei konkreten Anwendungen ebenfalls automatisch markiert werden, um eine ähnliche Umsetzung wie bei der Fragengenerierung ohne Wissen der Antwort nutzen zu können. In dieser Arbeit wurde die Markierung von benannten Entitäten als Antwort im Text verwendet. Allerdings hat dieser Ansatz bei einer manuellen Auswertung schlechter abgeschnitten im Bezug auf potentielle Chatbotfragen im Vergleich zur Fragengenerierung ohne Wissen der Antwort. Um das Potential der Fragengenerierung unter Wissen der Antwort besser in Anwendungen nutzen zu können, müssten weitere Möglichkeiten der Antwortmarkierung getestet werden.

Bezüglich der Paraphrasengenerierung könnten englischsprachige mit deutschsprachigen Datensätzen kombiniert werden und so ein multilinguales Modell trainiert werden. Dadurch könnte ein besseres Paraphrasierungsmodell für die deutsche Sprache trainiert werden, da die vorhandenen deutschsprachigen Datensätze nach der manuellen Evaluation anscheinend nicht genug Trainingsbeispiele haben.

Schließlich sind große neuronale Sprachmodelle eine weitere Möglichkeit die Performanz zu verbessern, allerdings haben diese hohe Hardware Anforderungen.

LITERATUR

[AM20] Eleni Adamopoulou und Lefteris Moussiades. "An Overview of Chatbot Technology". In: *Artificial Intelligence Applications and Innovations*. Hrsg. von Ilias Maglogiannis, Lazaros Iliadis und Elias Pimenidis. Cham: Springer International Publishing, 2020, S. 373–383. ISBN: 978-3-030-49186-4.

[Add19] Automatic Addison. *Artificial Feedforward Neural Network With Backpropagation From Scratch*. Letzter Aufruf 11.01.2022. 2019. URL: https://automaticaddison.com/artificial-feedforward-neural-network-with-backpropagation-from-scratch/.

[Agr+20] Parag Agrawal u. a. "QnAMaker: Data to Bot in 2 Minutes". In: *CoRR* abs/2003.08553 (2020). arXiv: 2003.08553. URL: https://arxiv.org/abs/2003.08553.

[Akb+19] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter und Roland Vollgraf. "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, Juni 2019, S. 54–59. DOI: 10.18653/v1/N19-4010. URL: https://aclanthology.org/N19-4010.

[Aky+21] Fatih Cagatay Akyon, Devrim Cavusoglu, Cemil Cengiz, Sinan Onur Altinuc und Alptekin Temizel. "Automated question generation and question answering from Turkish texts using text-to-text transformers". In: *CoRR* abs/2111.06476 (2021). arXiv: 2111.06476. URL: https://arxiv.org/abs/2111.06476.

[Alb+19] Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin und Michael Collins. "Synthetic QA Corpora Generation with Roundtrip Consistency". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Juli 2019, S. 6168–6173. DOI: 10.18653/v1/P19-1620. URL: https://aclanthology.org/P19-1620.

[Ali20] Sai Vamsi Alisetti. *Paraphrase Generator with T5*. Letzter Aufruf 23.05.2022. 2020. URL: https://github.com/Vamsi995/Paraphrase-Generator.

[ARY19] Mikel Artetxe, Sebastian Ruder und Dani Yogatama. "On the Cross-lingual Transferability of Monolingual Representations". In: *CoRR* abs/1910.11856 (2019). arXiv: 1910.11856. URL: http://arxiv.org/abs/1910.11856.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho und Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

[BL05] Satanjeev Banerjee und Alon Lavie. "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments". In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, Juni 2005, S. 65–72. URL: https://aclanthology.org/W05-0909.

[Bar21] Adrien Barbaresi. "Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Aug. 2021, S. 122–131. DOI: 10.18653/v1/2021.acl-demo.15. URL: https://aclanthology.org/2021.acl-demo.15.

[Ber+21] Auday Berro, Mohammad-Ali Yaghub Zade Fard, Marcos Baez, Boualem Benatallah und Khalid Benabdeslem. "An Extensible and Reusable Pipeline for Automated Utterance Paraphrases". In: *Proc. VLDB Endow.* 14.12 (2021), S. 2839–2842. ISSN: 2150-8097. DOI: 10.14778/3476311.3476358. URL: https://doi.org/10.14778/3476311.3476358.

[Boj+16] Piotr Bojanowski, Edouard Grave, Armand Joulin und Tomas Mikolov. "Enriching Word Vectors with Subword Information". In: abs/1607.04606 (2016). URL: http://arxiv.org/abs/1607.04606.

[Bre01] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (2001), S. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[BB18] Seppe van den Broucke und Bart Baesens. *Practical Web Scraping for Data Science - Best Practices and Examples with Python*. New York: Apress, 2018. ISBN: 978-1-484-23582-9.

[Bro+20] Tom B. Brown u. a. *Language Models are Few-Shot Learners*. 2020. DOI: 10.48550/ARXIV.2005.14165. URL: https://arxiv.org/abs/2005.14165.

[Bus08] Hadumod Bussmann. *Lexikon der Sprachwissenschaft*. Alfred Kröner Verlag, Nov. 2008. 819 S. ISBN: 978-3-520-45291-7. URL: https://www.ebook.de/de/product/22999092/lexikon_der_sprachwissenschaft.html.

[CR18] Massimo Canonico und Luigi De Russis. "A Comparison and Critique of Natural Language Understanding Tools". In: Letzer Aufruf 29.06.2022. 2018. URL: https://iris.polito.it/retrieve/handle/11583/2694629/188100/cloud_computing_2018_6_20_20057.pdf.

[CSM20] Branden Chan, Stefan Schweter und Timo Möller. "German's Next Language Model". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dez. 2020, S. 6788–6796. DOI: 10.18653/v1/2020.coling-main.598. URL: https://aclanthology.org/2020.coling-main.598.

[CF19] Ying-Hong Chan und Yao-Chung Fan. "A Recurrent BERT-based Model for Question Generation". In: *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, S. 154–162. DOI: 10.18653/v1/D19-5821. URL: https://aclanthology.org/D19-5821.

[Cha20] Amit Chaudhary. *A Visual Survey of Data Augmentation in NLP*. Letzter Aufruf 25.05.2022. 2020. URL: https://amitness.com/2020/05/data-augmentation-for-nlp.

[Chi+19] Muthuraman Chidambaram, Yinfei Yang, Daniel Cer, Steve Yuan, Yun-Hsuan Sung, Brian Strope und Ray Kurzweil. *Learning Cross-Lingual Sentence Representations via a Multi-task Dual-Encoder Model*. 2019. arXiv: 1810.12836 [cs.CL].

[Cho+22] Aakanksha Chowdhery u. a. *PaLM: Scaling Language Modeling with Pathways*. 2022. DOI: 10.48550/ARXIV.2204.02311. URL: https://arxiv.org/abs/2204.02311.

[Chr05] Chrislb. *File:ArtificialNeuronModel deutsch.png*. Letzter Aufruf 11.01.2022. 2005. URL: https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_deutsch.png.

[Gooa] *Classification: Accuracy*. online. Letzer Aufruf 17.05.2022. URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy.

[Goob] *Classification: Precision and Recall*. online. Letzer Aufruf 17.05.2022. URL: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.

[Cou18] Claude Coulombe. "Text Data Augmentation Made Simple By Leveraging NLP Cloud APIs". In: *CoRR* abs/1812.04718 (2018). arXiv: 1812.04718. URL: http://arxiv.org/abs/1812.04718.

[Dom] *DOM - Document Object Model Standard*. Letzter Aufruf 27.01.2022. URL: https://dom.spec.whatwg.org/.

[Dal21] Eric C. Dallmeier. "Computer Vision-based Web Scraping for Internet Forums". In: *2021 7th International Conference on Optimization and Applications (ICOA)*. 2021, S. 1–5. DOI: 10.1109/ICOA51614.2021.9442634.

[Dam21] Prithiviraj Damodaran. *Parrot: Paraphrase generation for NLU.* Version v1.0. Letzter Aufruf 23.05.2022. 2021. URL: https://github.com/PrithivirajDamodaran/Parrot_Paraphraser.

[DK+20] Kordula De Kuthy, Madeeswaran Kannan, Haemanth Santhi Ponnusamy und Detmar Meurers. "Towards automatically generating Questions under Discussion to link information and discourse structure". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dez. 2020, S. 5786–5798. DOI: 10.18653/v1/2020.coling-main.509. URL: https://aclanthology.org/2020.coling-main.509.

[Den+22] Yongkai Deng, Chenyu Fan, Sutong Jin und Hongxi Zhang. "Different Data Enhancement Methods applied on Imbalanced Data". In: *2022 14th International Conference on Computer Research and Development (ICCRD)*. 2022, S. 190–196. DOI: 10.1109/ICCRD54409.2022.9730352.

[DL14] Michael Denkowski und Alon Lavie. "Meteor Universal: Language Specific Translation Evaluation for Any Target Language". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, Juni 2014, S. 376–380. DOI: 10.3115/v1/W14-3348. URL: https://aclanthology.org/W14-3348.

[Dev+18] Jacob Devlin, Ming-Wei Chang, Kenton Lee und Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. URL: http://arxiv.org/abs/1810.04805.

[Gooc] *Dialogflow*. Letzter Aufruf 30.01.2022. URL: https://cloud.google.com/dialogflow.

[DB05] William B. Dolan und Chris Brockett. "Automatically Constructing a Corpus of Sentential Paraphrases". In: *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*. 2005. URL: https://aclanthology.org/I05-5002.

[FLD18] Angela Fan, Mike Lewis und Yann N. Dauphin. "Hierarchical Neural Story Generation". In: *CoRR* abs/1805.04833 (2018). arXiv: 1805.04833. URL: http://arxiv.org/abs/1805.04833.

[FEQ19] Christian Federmann, Oussama Elachqar und Chris Quirk. "Multilingual Whispers: Generating Paraphrases with Translation". In: *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, S. 17–26. DOI:

10.18653/v1/D19-5503. URL: https://aclanthology.org/D19-5503.

[GVDCB13] Juri Ganitkevitch, Benjamin Van Durme und Chris Callison-Burch. "PPDB: The Paraphrase Database". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, Juni 2013, S. 758–764. URL: https://aclanthology.org/N13-1092.

[Git] *Github*. Letzter Aufruf 23.05.2022. URL: https://github.com/.

[Gol20] Ramsri Goutham Golla. *Paraphrase any question with T5*. Letzter Aufruf 23.05.2022. 2020. URL: https://github.com/ramsrigouthamg/Paraphrase-any-question-with-T5-Text-To-Text-Transfer-Transformer-.

[Good] *Google Research ROUGE Implementierung*. online. Letzer Aufruf 17.05.2022. URL: https://github.com/google-research/google-research/tree/master/rouge.

[Gooe] *Googlebot*. Letzter Aufruf 27.01.2022. URL: https://developers.google.com/search/docs/advanced/crawling/googlebot.

[Gro+21] Khushnuma Grover, Katinder Kaur, Kartikey Tiwari, Rupali und Parteek Kumar. "Deep Learning Based Question Generation Using T5 Transformer". In: *Advanced Computing*. Hrsg. von Deepak Garg, Kit Wong, Jagannathan Sarangapani und Suneet Kumar Gupta. Singapore: Springer Singapore, 2021, S. 243–255. ISBN: 978-981-16-0401-0.

[HF97] Birgit Hamp und Helmut Feldweg. "GermaNet - a Lexical-Semantic Net for German". In: *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*. 1997. URL: https://aclanthology.org/W97-0802.

[Hao+11] Qiang Hao, Rui Cai, Yanwei Pang und Lei Zhang. "From One Tree to a Forest: A Unified Solution for Structured Web Data Extraction". In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. Beijing, China: Association for Computing Machinery, 2011, S. 775–784. ISBN: 9781450307574. DOI: 10.1145/2009916.2010020. URL: https://doi.org/10.1145/2009916.2010020.

[HS10] Michael Heilman und Noah A. Smith. "Good Question! Statistical Ranking for Question Generation". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. HLT '10. Los Angeles, California: Association for Computational Linguistics, 2010, S. 609–617. ISBN: 1932432655.

[Hol+19] Ari Holtzman, Jan Buys, Maxwell Forbes und Yejin Choi. "The Curious Case of Neural Text Degeneration". In: *CoRR* 1904.09751 (2019). arXiv: 1904.09751. URL: http://arxiv.org/abs/1904.09751.

[Hon+20] Matthew Honnibal, Ines Montani, Sofie Van Landeghem und Adriane Boyd. "spaCy: Industrial-strength Natural Language Processing in Python". In: (2020). DOI: 10.5281/zenodo.1212303. URL: https://github.com/explosion/spaCy.

[Huga] *Huggingface Datasets ROUGE*. online. Letzer Aufruf 17.05.2022. URL: https://github.com/huggingface/datasets/tree/master/metrics/rouge.

[Huy19] Chip Huyen. *Evaluation Metrics for Language Modeling*. https://thegradient.pub/understanding-evaluation-metrics-for-language-models/. Letzer Aufruf 29.06.2022. 2019.

[Jsoa] *JSON for Linking Data*. Letzter Aufruf 24.01.2022. URL: https://json-ld.org/.

[Jsob] *JavaScript Object Notation*. Letzter Aufruf 24.01.2022. URL: https://www.json.org/json-en.html.

[JR05] Valentin Jijkoun und Maarten de Rijke. "Retrieving Answers from Frequently Asked Questions Pages on the Web". In: *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. CIKM '05. Bremen, Germany: Association for Computing Machinery, 2005, S. 76–83. ISBN: 1595931406. DOI: 10.1145/1099554.1099571. URL: https://doi.org/10.1145/1099554.1099571.

[KLJ21] Kapaya Katongo, Geoffrey Litt und Daniel Jackson. "Towards End-User Web Scraping for Customization". In: *Companion Proceedings of the 5th International Conference on the Art, Science, and Engineering of Programming*. Programming '21. Cambridge, United Kingdom: Association for Computing Machinery, 2021, S. 49–59. ISBN: 9781450389860. DOI: 10.1145/3464432.3464437. URL: https://doi.org/10.1145/3464432.3464437.

[KN19] Tassilo Klein und Moin Nabi. "Learning to Answer by Learning to Ask: Getting the Best of GPT-2 and BERT Worlds". In: *CoRR* abs/1911.02365 (2019). arXiv: 1911.02365. URL: http://arxiv.org/abs/1911.02365.

[Kol15] Tobias Kolditz. "Generating questions for german text". Erhalten nach E-Mail Anfrage von Herr Kolditz. Master. University of Tübingen, 2015.

[KBH18] Pavel Kucherbaev, Alessandro Bozzon und Geert-Jan Houben. "Human-Aided Bots". In: *IEEE Internet Computing* 22.6 (2018), S. 36–43. DOI: 10.1109/MIC.2018.252095348.

[KR18] Taku Kudo und John Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, S. 66–71. DOI: 10.18653/v1/D18-2012. URL: https://aclanthology.org/D18-2012.

[KWD97] Nicholas Kushmerick, Daniel S. Weld und Robert B. Doorenbos. "Wrapper Induction for Information Extraction". In: *IJCAI*. 1997.

[Kwi+19] Tom Kwiatkowski u. a. "Natural Questions: a Benchmark for Question Answering Research". In: *Transactions of the Association of Computational Linguistics* (2019).

[Lew+19] Patrick S. H. Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel und Holger Schwenk. "MLQA: Evaluating Cross-lingual Extractive Question Answering". In: *CoRR* abs/1910.07475 (2019). arXiv: 1910.07475. URL: http://arxiv.org/abs/1910.07475.

[Lho+21] Quentin Lhoest u. a. "Datasets: A Community Library for Natural Language Processing". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Nov. 2021, S. 175–184. URL: https://aclanthology.org/2021.emnlp-demo.21.

[Lin+20] Bill Yuchen Lin, Ying Sheng, Nguyen Vo und Sandeep Tata. "FreeDOM: A Transferable Neural Architecture for Structured Information Extraction on Web Documents". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2020, S. 1092–1102. ISBN: 9781450379984. URL: https://doi.org/10.1145/3394486.3403153.

[Lin04] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Juli 2004, S. 74–81. URL: https://aclanthology.org/W04-1013.

[Spa] *Linguistic Features*. Letzter Aufruf 29.01.2022. URL: https://spacy.io/usage/linguistic-features.

[Lop+20] Luis Enrico Lopez, Diane Kathryn Cruz, Jan Christian Blaise Cruz und Charibeth Cheng. "Simplifying Paragraph-level Question Generation via Transformer Language Models". In: *CoRR* abs/2005.01107 (2020). arXiv: 2005.01107. URL: https://arxiv.org/abs/2005.01107.

[LH17] Ilya Loshchilov und Frank Hutter. "Fixing Weight Decay Regularization in Adam". In: *CoRR* 1711.05101 (2017). arXiv: 1711.05101. URL: http://arxiv.org/abs/1711.05101.

[Goof] *Manage knowledge bases: Supported content*. Letzter Aufruf 17.02.2022. URL: https://cloud.google.com/dialogflow/es/docs/how/knowledge-bases#supported-content.

[Mas+20] Yosi Mass, Boaz Carmeli, Haggai Roitman und David Konopnicki. "Unsupervised FAQ Retrieval with Question Generation and BERT". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Juli 2020, S. 807–812. DOI: 10.18653/v1/2020.acl-main.74. URL: https://aclanthology.org/2020.acl-main.74.

[MP43] Warren S McCulloch und Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), S. 115–133.

[Men20] Martino Mensio. *Spacy - Universal Sentence Encoder*. 2020. URL: https://github.com/MartinoMensio/spacy-universal-sentence-encoder.

[Mik+13] Tomas Mikolov, Kai Chen, Greg Corrado und Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781 (2013). URL: http://arxiv.org/abs/1301.3781.

[Mil95] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (1995), S. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: https://doi.org/10.1145/219717.219748.

[MRP21] Timo Möller, Julian Risch und Malte Pietsch. "GermanQuAD and GermanDPR: Improving Non-English Question Answering and Passage Retrieval". In: *CoRR* abs/2104.12741 (2021). arXiv: 2104.12741. URL: https://arxiv.org/abs/2104.12741.

[Stt] *Morphosyntactic tagsets for German*. Letzter Aufruf 29.06.2022. URL: https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/germantagsets/.

[Mur+21] Lidiya Murakhovs'ka, Chien-Sheng Wu, Tong Niu, Wenhao Liu und Caiming Xiong. "MixQG: Neural Question Generation with Mixed Answer Types". In: *CoRR* abs/2110.08175 (2021). arXiv: 2110.08175. URL: https://arxiv.org/abs/2110.08175.

[NB21] Harshit Nigam und Prantik Biswas. "From Web Scraping to Web Crawling". In: *Applications of Artificial Intelligence and Machine Learning*. Hrsg. von Ankur Choudhary, Arun Prakash Agrawal, Rajasvaran Logeswaran und Bhuvan Unhelkar. Sin-

gapore: Springer Singapore, 2021, S. 97–112. ISBN: 978-981-16-3067-5.

[NC17] Ketakee Nimavat und Tushar Champaneria. "Chatbots: An overview. Types, Architecture, Tools and Future Possibilities". In: Okt. 2017.

[Ope] "OpenThesaurus: ein offenes deutsches Wortnetz". In: (2005). Letzter Aufruf 07.06.2022. URL: http://www.danielnaber.de/publications/gldv-openthesaurus.pdf.

[Pap+02] Kishore Papineni, Salim Roukos, Todd Ward und Wei-Jing Zhu. "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, S. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

[Pat20] Suraj Patil. *Question Generation using transformers*. online. Version 1.0.0. Letzter Aufruf 10.05.2022. 2020. URL: https://github.com/patil-suraj/question_generation.

[PSM14] Jeffrey Pennington, Richard Socher und Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Okt. 2014, S. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[Phi] Michael Phi. *Illustrated Guide to Transformers- Step by Step Explanation*. Letzter Aufruf 18.01.2022. URL: https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0.

[Pla20] Patrick von Platen. *How to generate text: using different decoding methods for language generation with Transformers*. Letzter Aufruf 26.05.2022. 2020. URL: https://huggingface.co/blog/how-to-generate.

[Pom11] Jan Pomikálek. "Removing Boilerplate and Duplicate Content from Web Corpora". Letzter Aufruf 09.02.2022. Dissertation. Masarykova univerzita, Fakultät für Informatik, Brünn, 2011. URL: https://is.muni.cz/th/o6om2/.

[Pos18] Matt Post. "A Call for Clarity in Reporting BLEU Scores". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, Okt. 2018, S. 186–191. DOI: 10.18653/v1/W18-6319. URL: https://aclanthology.org/W18-6319.

[Quo] *Quora Question Pairs Dataset*. Letzter Aufruf 25.05.2022. URL: https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs.

[RN18] Alec Radford und Karthik Narasimhan. “Improving Language Understanding by Generative Pre-Training”. In: 2018.

[Rad+19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei und Ilya Sutskever. “Language Models are Unsupervised Multitask Learners”. In: 2019.

[Raf+19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li und Peter J. Liu. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *CoRR* abs/1910.10683 (2019). arXiv: 1910.10683. URL: http://arxiv.org/abs/1910.10683.

[RJL18] Pranav Rajpurkar, Robin Jia und Percy Liang. “Know What You Don’t Know: Unanswerable Questions for SQuAD”. In: *CoRR* abs/1806.03822 (2018). arXiv: 1806.03822. URL: http://arxiv.org/abs/1806.03822.

[Raj+16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev und Percy Liang. “SQuAD: 100.000+ Questions for Machine Comprehension of Text”. In: *CoRR* abs/1606.05250 (2016). arXiv: 1606.05250. URL: http://arxiv.org/abs/1606.05250.

[RJH21] Muhammad Shihab Rashid, Fuad Jamour und Vagelis Hristidis. “QuAX: Mining the Web for High-Utility FAQ”. In: *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2021, S. 1518–1527. ISBN: 9781450384469. URL: https://doi.org/10.1145/3459637.3482289.

[RF16] Joseph Redmon und Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *CoRR* abs/1612.08242 (2016). arXiv: 1612.08242. URL: http://arxiv.org/abs/1612.08242.

[RN21] Stuart Russell und Peter Norvig. *Artificial Intelligence - A Modern Approach*. 4. Aufl. Pearson Education Limited., 2021. ISBN: 9781292153964.

[SA20] Stefan Schweter und Alan Akbik. “FLERT: Document-Level Features for Named Entity Recognition”. In: *CoRR* 2011.06993 (2020). arXiv: 2011.06993. URL: https://arxiv.org/abs/2011.06993.

[SPS21] Shilpa Sen, Mayank Patel und Ajay Kumar Sharma. “Software Development Life Cycle Performance Analysis”. In: *Emerging Trends in Data Driven Computing and Communications*. Hrsg. von Rajeev Mathur, C. P. Gupta, Vaibhav Katewa, Dharm Singh Jat und Neha Yadav. Singapore: Springer Singapore, 2021, S. 311–319. ISBN: 978-981-16-3915-9.

[SS18] Noam Shazeer und Mitchell Stern. "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost". In: *CoRR* 1804.04235 (2018). arXiv: 1804.04235. URL: http://arxiv.org/abs/1804.04235.

[Soy+21] Fatih Soygazi, Okan Çiftçi, Uğurcan Kök und Soner Cengiz. "THQuAD: Turkish Historic Question Answering Dataset for Reading Comprehension". In: *2021 6th International Conference on Computer Science and Engineering (UBMK)*. 2021, S. 215–220. DOI: 10.1109/UBMK52708.2021.9559013.

[Sta+18] Peter W. J. Staar, Michele Dolfi, Christoph Auer und Costas Bekas. "Corpus Conversion Service: A Machine Learning Platform to Ingest Documents at Scale". In: *CoRR* abs/1806.02284 (2018). URL: http://arxiv.org/abs/1806.02284.

[Hugb] *Summary of the tokenizers*. Letzter Aufruf 22.06.2022. URL: https://huggingface.co/docs/transformers/tokenizer_summary.

[T5m] *T5 Model : What is maximum sequence length that can be used with pretrained T5 (3b model) checkpoint?* Letzter Aufruf 22.06.2022. 2020. URL: https://github.com/huggingface/transformers/issues/5204.

[TP20] Brian Thompson und Matt Post. "Paraphrase Generation as Zero-Shot Multilingual Translation: Disentangling Semantic Similarity from Lexical and Syntactic Diversity". In: *Proceedings of the Fifth Conference on Machine Translation*. Online: Association for Computational Linguistics, Nov. 2020, S. 561–570. URL: https://aclanthology.org/2020.wmt-1.67.

[Tim] Tim Lai. *FAQ Extraction via Watson Assistant Search Skill*. Letzter Aufruf 15.02.2022. URL: https://community.ibm.com/community/user/watsonapps/blogs/tim-lai1/2020/12/02/faq-extraction-via-watson-assistant-search-skill.

[Jav] *Understand the JavaScript SEO basics*. Letzter Aufruf 25.01.2022. URL: https://developers.google.com/search/docs/advanced/javascript/dynamic-rendering?hl=de/.

[Uni] *Universal POS tags*. Letzter Aufruf 29.01.2022. URL: https://universaldependencies.org/u/pos/.

[Sea] *Use a search skill to embed existing help content*. Letzter Aufruf 15.02.2022. URL: https://cloud.ibm.com/docs/assistant?topic=assistant-skill-search-add.

[Sma] *Using Smart Document Understanding*. Letzter Aufruf 15.02.2022. URL: https://cloud.ibm.com/docs/discovery-data?topic=discovery-data-configuring-fields.

[Vaj+20] Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta und Harshit Surana. *Practical Natural Language Processing - A Comprehensive Guide to Building Real-World NLP Systems*. Sebastopol: O'Reilly Media, Inc., 2020. ISBN: 978-1-492-05400-9.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser und Illia Polosukhin. "Attention is All you Need". In: 30 (2017). Hrsg. von I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan und R. Garnett.

[Ibm] *Watson Assistant: Intelligenter virtueller Agent*. Letzter Aufruf 30.01.2022. URL: https://www.ibm.com/de-de/products/watson-assistant.

[WG18] John Wieting und Kevin Gimpel. "ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Juli 2018, S. 451–462. DOI: 10.18653/v1/P18-1042. URL: https://aclanthology.org/P18-1042.

[Wol+20] Thomas Wolf u. a. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Okt. 2020, S. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://aclanthology.org/2020.emnlp-demos.6.

[Xue+20] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua und Colin Raffel. *mT5: A massively multilingual pre-trained text-to-text transformer*. 2020. DOI: 10.48550/ARXIV.2010.11934. URL: https://arxiv.org/abs/2010.11934.

[Yan+19] Yinfei Yang, Yuan Zhang, Chris Tar und Jason Baldridge. "PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification". In: *CoRR* abs/1908.11828 (2019). arXiv: 1908.11828. URL: http://arxiv.org/abs/1908.11828.

[Zha+21] Ruqing Zhang, Jiafeng Guo, Lu Chen, Yixing Fan und Xueqi Cheng. "A Review on Question Generation from Natural Language Text". In: *ACM Trans. Inf. Syst.* 40.1 (2021). ISSN: 1046-8188. DOI: 10.1145/3468889. URL: https://doi.org/10.1145/3468889.

[Zha+19] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger und Yoav Artzi. "BERTScore: Evaluating Text Generation with BERT". In: *CoRR* abs/1904.09675 (2019). arXiv: 1904.09675. URL: http://arxiv.org/abs/1904.09675.

[ZBH19] Yuan Zhang, Jason Baldridge und Luheng He. "PAWS: Paraphrase Adversaries from Word Scrambling". In: *CoRR* abs/1904.01130 (2019). arXiv: 1904.01130. URL: http://arxiv.org/abs/1904.01130.

[Sci] *f1 score*. online. Letzter Aufruf 17.05.2022. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[Opt] *huggingface - Optimizer*. Letzter Aufruf 16.05.2022. URL: https://huggingface.co/docs/transformers/v4.19.0/en/main_classes/optimizer_schedules.

[Hugc] *huggingface - Performance*. Letzter Aufruf 16.05.2022. URL: https://huggingface.co/docs/transformers/performance.

[Tra] *huggingface - TrainingArguments*. Letzter Aufruf 16.05.2022. URL: https://huggingface.co/docs/transformers/main_classes/trainer#transformers.TrainingArguments.

[Jus] *jusText*. online. Letzter Aufruf 09.02.2022. URL: https://github.com/miso-belica/jusText.

[tea20] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.