Karlsruher Institut für Technologie

Konfidenzberechnungen für Roboterbefehle aus natürlichsprachigen Äußerungen

Bachelorarbeit von

Lukas Hilgert

an der Fakultät für Informatik
Institut für Anthropomatik und Robotik (IAR)

Erstgutachter:	Prof. Dr. Alexander Waibel
Zweitgutachter:	Prof. Dr. Tamim Asfour
Betreuender Mitarbeiter:	M.Sc. Stefan Constantin

01. Oktober 2021 – 31. Januar 2022

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

Karlsruhe, 31.01.2022

. . . . . .

(Lukas Hilgert)

Zusammenfassung

Wenn ein Roboter von einem Benutzer Befehle in natürlicher Sprache entgegennimmt, sollte sichergestellt sein, dass die Befehle richtig verstanden wurden und nicht falsche Aktionen ausgeführt werden. Unter Verwendung einer der neusten Arbeiten auf dem Gebiet der sprachübergreifenden Qualitätsabschätzung für maschinelle Übersetzung auf Wortebene mit dem Modell TransQuest [21] wird in dieser Arbeit ein System aus einer Komponente für Natural Language Understanding (NLU) und darauffolgender Qualitätsabschätzung durch TransQuest vorgestellt, das für einen Roboter im Kontext von Aktivitäten in der Küche bei Unsicherheiten Rückfragen an den Benutzer stellen kann. In dieser Kombination kann ca. die Hälfte aller Fehler entdeckt werden, wobei gleichzeitig nur selten unnötige Rückfragen ausgelöst werden. Vereint man die Qualitätsabschätzung von TransQuest mit der durch A-posteriori-Wahrscheinlichkeiten, können noch einmal mehr Fehler gefunden werden.

Inhaltsverzeichnis

Zusammenfassung **i**

1. Einführung **1**
1.1. Motivation . . . 1
1.2. Zielsetzung . . . 1
1.3. Struktur . . . 2

2. Grundlagen **3**
2.1. Perzeptron . . . 3
2.2. Mehrlagiges Perzeptron (MLP) . . . 3
2.3. Backpropagation . . . 4
2.4. Fortgeschrittene Formen von neuronalen Netzen . . . 5
2.5. Transformer . . . 5

3. Verwandte Arbeiten **9**
3.1. Übersetzung . . . 9
3.2. Konfidenzabschätzung / Qualitätsabschätzung . . . 9

4. Modelle und Methoden **11**
4.1. Übersetzung . . . 12
4.2. Qualitätsabschätzung . . . 12
4.2.1. TransQuest . . . 12

5. Evaluation **15**
5.1. Datensatz . . . 15
5.2. Training . . . 18
5.3. Ergebnisse . . . 20

6. Fazit **23**
6.1. Zukünftige Arbeiten . . . 23

Literatur **25**

A. Anhang **29**
A.1. Modelle . . . 29
A.2. Datensatz . . . 30
A.2.1. Bekannte Nomen / Objekte . . . 30
A.2.2. Beispiele . . . 32

A.3. Ergebnisse 33
A.3.1. T5 33
A.3.2. TransQuest 34

Abbildungsverzeichnis

2.1. Architektur des Transformer-Models [30] 6
2.2. Aufbau von Multi-Head Attention [30] 7
2.3. Aufbau von Scaled Dot-Product Attention [30] 7

4.1. Architektur von TransQuest auf Wortebene [21] 14

Tabellenverzeichnis

5.1. Größe der einzelnen Datensätze vor Anreicherung 15
5.2. Korrekte Übersetzung 16
5.3. Fehlertyp: Fehlendes Wort 16
5.4. Fehlertyp: Ersetztes Wort 17
5.5. Fehlertyp: Zufälliges Wort 17
5.6. Fehlertyp: Nomen nicht abgebildet 17
5.7. Fehlertyp: Fehlendes Adverb 17
5.8. Mögliche Ergebnisse bei binärer Klassifikation 18
5.9. Evaluation von TransQuest (20 %, Validierungsfehler) 20
5.10. Evaluation von TransQuest (20 %, F1-Wert) 20
5.11. Evaluation von TransQuest (50 %, F1-Wert) 21
5.12. Qualitätsabschätzung für T5: TransQuest 21
5.13. Qualitätsabschätzung für T5: A-posteriori-Wahrscheinlichkeiten 21
5.14. Qualitätsabschätzung für T5: TransQuest, A-posteriori-Wahrscheinlichkeiten 21

A.1. Verlauf des Trainings von T5 33
A.2. Verlauf des Trainings von TransQuest 34

1. Einführung

1.1. Motivation

Auch nach vielen Fortschritten im Bereich der Verarbeitung natürlicher Sprache, der auch gerade von der Veröffentlichung des Transformer-Modells [30] profitiert hat, sind die Ergebnisse noch lange nicht perfekt. Da ein System zu einer Eingabe immer eine bestimmte Ausgabe erzeugen wird, obwohl auch die Wahrscheinlichkeit für eine andere Ausgabe hoch war, ist es daher umso wichtiger, dass ein System die Konfidenz / Qualität für seine Ausgabe abschätzen kann. Das gilt auch, wenn das System immer mehrere mögliche Ausgaben z. B. mittels Beam Search gleichzeitig erzeugt, weil hier mittels der Konfidenz die beste Möglichkeit auswählen werden könnte. Wenn ein Roboter einen Befehl ausführen soll, was in dieser Arbeit als Anwendungsfall auftritt, wäre es besser, wenn das System abschätzen kann, ob der Befehl sicher verstanden wurde, damit keine falsche Aktion ausgeführt wird. Auch bei Übersetzung z. B. kann für ein Wort die Konfidenz gering sein und daraufhin vom Benutzer eventuell mehr Informationen erfragt werden. Ein anderer Fall wäre bei automatischer Spracherkennung oder bei Sprachsynthese aus Text, wenn ein englisches Wort in einem deutschen Text vorkommt und dann fälschlicherweise auf ein deutsches Wort abgebildet wird.

1.2. Zielsetzung

In dieser Arbeit soll es darum gehen, ein System für Natural Language Understanding (NLU) mit einer Textschnittstelle für Befehle in englischer Sprache für einen Roboter im Kontext von Tätigkeiten in der Küche zu entwickeln, wobei auch andere Kontexte für den Roboter denkbar wären. Dazu werden die Befehle in eine Art „Robotersprache“ gebracht. Das System soll dabei die Konfidenz für die einzelnen Teile des Befehls (Aktion, beteiligte Objekte) abschätzen können. Dabei sollen die A-posteriori-Wahrscheinlichkeiten des Modells und vor allem andere fortgeschrittene Methoden zur Qualitätsabschätzung zum Einsatz kommen.

Auf der Konfidenz basierend soll das System Rückfragen stellen können. Wenn z. B. ein Objekt falsch geschrieben ist, könnte nochmal gefragt werden, welches Objekt gemeint ist. In einem anderen Fall könnte das genannte aber unbekannte Objekt einem anderen bekannten Objekt ähnlich sein, woraufhin das System dahingehend fragt. Da zu jedem Befehl neben den Objekten auch eine Aktion gehört, könnte auch diese falsch verstanden werden und mittels einer Rückfrage geklärt werden, welche Aktion ausgeführt werden soll.

1.3. Struktur

Nach dieser Einführung wird ein Überblick über die Grundlagen (Kapitel 2) und die Entwicklung von künstlichen neuronalen Netzen geben. Insbesondere wird dort auf das Transformer-Modell eingegangen, welches in dieser Arbeit starke Verwendung findet. Danach wird auf verwandte Arbeiten (Kapitel 3) zu den Themen Übersetzung und insbesondere Qualitätsabschätzung eingegangen. Im Abschnitt zu den Modellen werden die konkreten Transformer-Modelle beschrieben (Kapitel 4), die in dieser Arbeit verwendet werden, und wie sie genau zum Einsatz kommen. Anschließend wird der verwendete Datensatz, dessen Erweiterung für diese Arbeit, das Training der Modelle und die Evaluation (Kapitel 5) der Modelle beschrieben. Im Anhang (Anhang A) sind noch einige Materialien zu den Modellen, Ausschnitte zum Datensatz und Ergebnisse des Trainings zu finden.

2. Grundlagen

2.1. Perzeptron

Das Perzeptron ist eine Methode zur Klassifikation im Maschinellen Lernen. Aus mehreren Eingaben wird eine einzige Ausgabe berechnet. Hierzu wird zunächst die gewichtete Summe aller Eingaben y berechnet, wobei w das zugehörige Gewicht ist. Zusätzlich kann eine weitere Eingabe („Bias“) mit dem konstanten Wert 1 und einem Gewicht existieren.

$$x = \sum_i y_i w_i \tag{2.1}$$

Die gewichtete Summe wird nun noch mit einer nicht-linearen Funktion („Aktivierungsfunktion“) skaliert.

$$y = \frac{1}{1 + e^{-x}} \tag{2.2}$$

So ist die Ausgabe wie beim biologischen Vorbild erst dann entsprechend hoch, wenn die Eingaben einen gewissen Schwellwert überschreiten.

2.2. Mehrlagiges Perzeptron (MLP)

Mehrlagige Perzeptronen sind eine Art von künstlichen mehrschichtigen neuronalen Netzen. In ihrer Grundstruktur sind sie gerichtete Graphen und bestehen damit aus zwei Bestandteilen: Knoten und Kanten. Dabei sind Knoten die Perzeptronen und die Kanten die Ein- bzw. Ausgaben der Perzeptronen. Für Ein- und Ausgabe existieren jeweils eine Schicht, dazwischen können sich beliebig viele „verdeckte“ Schichten befinden. Jede dieser Schichten besteht aus einer Gewissen Anzahl von Knoten bzw. Perzeptronen.

Von jedem Perzeptron existieren ausgehende Kanten (Ausgabe des Perzeptrons) zu einer Schicht darunter, wenn man die Eingabeschicht als „oben“ ansieht, wobei die Ausgabeschicht die unterste ist. Dabei können Kanten auch Schichten überspringen. Jeder dieser Kanten ist wie bei den Eingaben beim einzelnen Perzeptron ein Gewicht zugeordnet.

Die Einträge des Eingabevektors setzen die Zustände der Eingabeschicht. Um nun die Ausgaben der Perzeptronen in den anderen Schichten bis hin zur Ausgabeschicht zu berechnen, geht man folgendermaßen vor: Zunächst berechnet man nacheinander für jede Schicht von jedem Perzeptron die Ausgabe und gibt diese dann als Eingaben an die verbundenen Perzeptronen der unteren Schicht weiter.

2.3. Backpropagation

Das Ziel des Trainings von neuronalen Netzen ist, die richtigen Gewichte für jede Kante zu finden, damit das Netz für jeden Eingabevektor den richtigen bzw. bestmöglichen Ausgabevektor erzeugt. Für eine endliche Menge an Ein- und Ausgabevektoren kann man auf die folgende Art [24], zu der es Alternativen gibt, den gesamten Fehler berechnen, wobei in dieser Formel y die tatsächlichen Zustände der Ausgabeknoten und d die gewollten Zustände sind. Weiter ist c der Index über die Ein- / Ausgabepaare und j über die Ausgabeknoten.

$$E = \frac{1}{2} \sum_{c} \sum_{j} (y_{j,c} - d_{j,c})^2 \tag{2.3}$$

Im Gegensatz zur Berechnung der Ausgabe eines Netzes, die „vorwärts“ abläuft, „propagiert“ man den Fehler rückwärts durch das Netz, um diesen Fehler per Gradientenabstieg zu minimieren [24]. Zunächst berechnet man die partielle Ableitung für jeden Ausgabeknoten:

$$\partial E / \partial y_j = y_j - d_j \tag{2.4}$$

Nun wendet man die Kettenregeln an, um herauszufinden, wie sich eine Änderung der gesamten Eingabe für einen Ausgabeknoten auf den Gesamtfehler auswirkt. Die gesamte Eingabe hängt wiederum von den vorherigen Knoten und den Gewichten ab.

$$\partial E / \partial x_j = \partial E / \partial y_j \cdot dy_j / dx_j = \partial E / \partial y_j \cdot y_j(1 - y_j) \tag{2.5}$$

Jetzt gilt es herauszufinden, wie sich die Gewichte auswirken (Gewicht der Kante von Knoten i zu Knoten j).

$$\partial E / \partial w_{ji} = \partial E / \partial x_j \cdot \partial x_j / \partial w_{ji} = \partial E / \partial x_j \cdot y_i \tag{2.6}$$

Um die Ableitungen für die vorletzte Schicht zu berechnen, betrachtet man alle Verbindungen ausgehend von i.

$$\partial E / \partial y_i = \sum_{j} \partial E / \partial x_j \cdot w_{ji} \tag{2.7}$$

Dies kann nun für jede folgende Schicht getan werden, wodurch man die Ableitungen für alle Gewichte findet. Für die einfachste Variante von Gradientenabstieg kann man nun für alle Ein- / Ausgabepaare die Ableitungen summieren und dann die Gewichte proportional anpassen.

$$\Delta w = -\varepsilon \partial E / \partial w \tag{2.8}$$

ε ist die Lernrate: ein konstanter Skalierungsfaktor, der festlegt, wie stark die Gewichte verändert werden. Alternativ kann man die Lernrate mit jedem Lernschritt angepasst werden.

$$\Delta w(t) = -\varepsilon \partial E / \partial w(t) + \alpha \Delta w(t + 1) \tag{2.9}$$

Hier wird t mit jedem Schritt um 1 erhöht und α ist ein Faktor zwischen 0 und 1 für den exponentiellen Zerfall, wie stark sich die aktuelle Änderung auswirkt.

2.4. Fortgeschrittene Formen von neuronalen Netzen

Diese strukturell recht simplen Netzwerke wie MLPs sind jedoch für komplexe Aufgaben nicht ausreichend, daher entwickelte man fortgeschrittenere Formen von neuronalen Netzen. Dabei wird ebenfalls Backpropagation (siehe Abschnitt 2.3) zum Training verwendet [8, 12] - allerdings in jeweils teils stark angepasster Form.

TDNN und CNN Das Time-Delay Neural Network (TDNN) wurde 1989 veröffentlicht [31], welches für die Erkennung von Phonemen entwickelt wurde. Dabei sollte die Erkennung insbesondere unabhängig von zeitlichen Verschiebungen der Phonemen sein. Die namensgebenden Verzögerungen sind den Eingaben der Perzeptronen vorgeschaltet, damit teilweise vergangene Eingaben in deren Ausgabe einfließen. Die tieferen Schichten erhalten jeweils nur Eingaben aus einer Auswahl bzw. einem Fenster der Eingabeschicht bzw. der höheren verdeckte Schichten. Die einzelnen Einheiten in einer Schicht sind in Rechteckform organisiert - pro Reihe werden die Gewichte geteilt.

Auch bei den im selben Jahr erschienen Convolutional Neural Networks (CNN) wird „Weight Sharing“ verwendet [12, 13]. Wie bei TDNNs erhalten hier tiefere Schichten nur Teile der Eingabe. Bei Netzwerken wie LeNet z. B. erhält eine Einheit jeweils nur einen Ausschnitt eines Bildes [13], dabei gibt es Überlappungen zwischen den Ausschnitten. Am Ende stehen jedoch vollständig verbundene („fully connected“) Schichten.

RNN Recurrent Neural Networks (RNNs) besitzen Verbindungen, die im Netzwerk „rückwärts“ gehen - also mit demselben Knoten oder einem aus einer vorherigen Schicht verbunden sind. So werden Eingaben der kürzeren Vergangenheit als Aktivierungen kurz gespeichert, was insbesondere in der Sprachverarbeitung von Vorteil ist [8]. Besonders hervorzuheben sind Long Short-Term Memory-Netzwerke (LSTM), die in speziellen Speicherzellen zusätzliche eine Art Langzeitgedächtnis haben [8].

Sequence-to-Sequence und Attention Für Sequence-to-Sequence-Modelle werden vorwiegend komplexe RNNs oder CNNs oder das Transformer-Modell [30] (siehe Abschnitt 2.5) benutzt, die aus einem Encoder- und Decoder-Teil bestehen [1, 30]. Dabei wird aus einer Sequenz von Eingabesymbolen zunächst durch den Encoder ein Vektor aus kontinuierlichen Repräsentationen berechnet. Daraus und aus der Sequenz der vorherigen Ausgaben generiert der Decoder ein Symbol pro Schritt. Die Leistung kann mit dem „Attention“-Mechanismus gesteigert werden [1]. Hierzu wird ein Teil des Netzwerks darauf trainiert, festzustellen, welcher Teil der Eingabe relevant für den nächsten Teil der Ausgabe ist.

2.5. Transformer

Das 2017 veröffentlichte Transformer-Modell kommt ausschließlich mit Attention und ohne Convolution oder Recurrence aus [30], lässt sich schneller und mit höherer Parallelität trainieren und erzielt darüber hinaus bessere Ergebnisse als vorherige Modelle.

Architektur Das Transformer-Modell ist ebenfalls in Encoder und Decoder aufgeteilt (Abbildung 2.1). Wie bei Sequence-to-Sequence-Modellen werden die Eingaben bzw. die schon berechneten Ausgaben zunächst in einen Vektor mit fester Größe konvertiert. Da

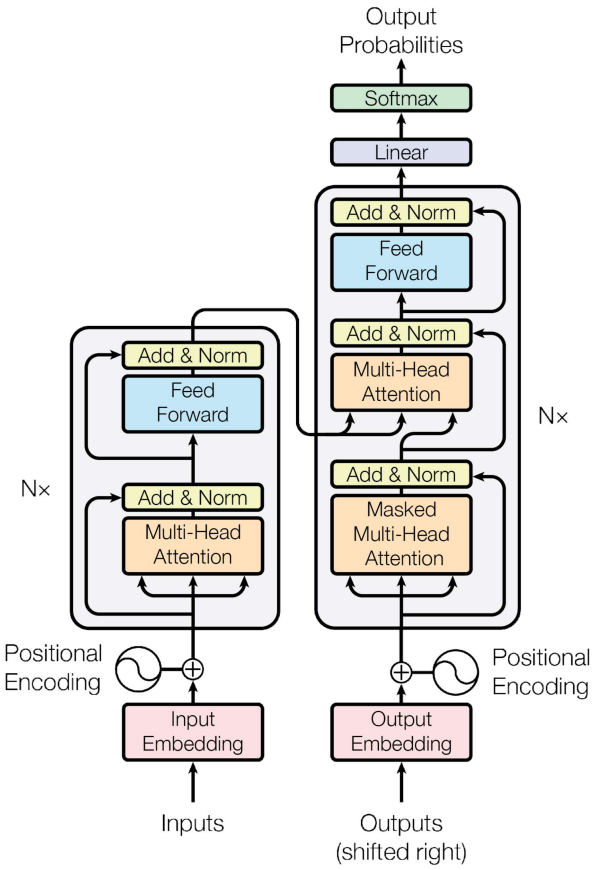

Abbildung 2.1.: Architektur des Transformer-Models [30]

dieses Modell weder Convolution noch Recurrence aufweist, wird noch „Positional Encoding“ addiert, welches dazu dient, Information über die Position der Eingabe zu erhalten. In seiner Ursprungsform [30] bestehen Encoder- und Decoder-Teil aus sechs gleichen aufeinanderfolgenden Einheiten, die auch ähnlich aufgebaut sind. In diesen Einheiten gibt es zwei Arten von Untereinheiten: Klassische „Feed-Forward“-Netzwerke, die also keine rückwärtigen Verbindungen aufweisen, und „Multi-Head Attention“-Mechanismen. Die Eingabe jeder Untereinheit wird mit deren Ausgabe addiert und die Summe normiert. Im Encoder erhält zunächst eine Multi-Head Attention-Untereinheit die vorverarbeitete Eingabe, danach folgt Feed Forward. Der Decoder erhält die schon existierende vorverarbeitete Ausgabe, welche zuerst in eine „Masked“ Multi-Head Attention-Untereinheit kommt, die sicherstellt, dass die Ausgabe des Netzwerkes nur den vorherigen und nicht zukünftigen Ausgaben abhängt. Danach folgt Multi-Head Attention, welche ebenfalls die Ausgabe des Encoders miteinbezieht. Als letzte Untereinheit kommt wieder ein Feed Forward-Netzwerk. Zum Schluss folgen noch lineare Transformationen und die Softmax-Funktion (Gleichung 2.12), welche die Wahrscheinlichkeiten für die nächste Ausgabe berechnet.

Attention Allgemein kann eine Attention-Funktion als Abbildung einer Anfrage („Query“) und einem Satz aus Schlüssel-Wert-Paaren auf eine Ausgabe, welche mit der gewichte-

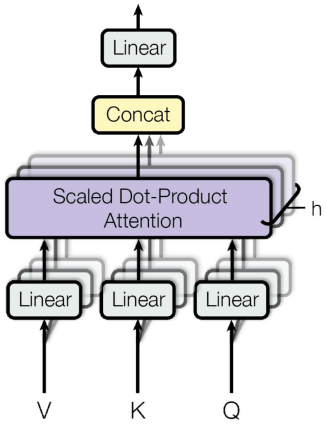

Abbildung 2.2.: Aufbau von Multi-Head Attention [30]

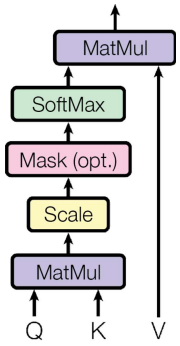

Abbildung 2.3.: Aufbau von Scaled Dot-Product Attention [30]

ten Summe der Werte berechnet wird, gesehen werden. Dabei sind die zu jedem Wert gehörenden Gewichte das Ergebnis einer Kompatibilitätsfunktion der Anfrage mit dem entsprechenden Schlüssel. Der Aufbau einer Multi-Head Attention-Untereinheit Abbildung 2.2 ist wie folgt: Die Eingabe wird aufgeteilt in Wert, Schlüssel und Query h mal in Attention-Schichten / -Köpfen parallel von einer „Scaled Dot-Product Attention“ verarbeitet, hierbei sind die Gewichte in den linearen Abbildungen unterschiedlich. Zum Schluss werden die Ergebnisse konkateniert und mit einer Gewichtsmatrix W^O multipliziert.

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{2.10}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.11}$$

In der Scaled Dot-Product Attention (Abbildung 2.3) wird das Skalarprodukt der Anfragen mit sämtlichen Schlüsseln berechnet, es mit $\sqrt{d_k}$ skaliert und dann mit der Softmax-Funktion die Gewichte für die Wert-Vektoren bestimmt. Optional findet die Maskierung für Masked Multi-Head Attention vor Softmax statt.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.12}$$

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \text{ für } 1, \ldots, K \tag{2.13}$$

3. Verwandte Arbeiten

Da sich diese Arbeit primär mit der Qualitätsabschätzung einer Übersetzung beschäftigt, wird in diesem Abschnitt nur kurz auf Arbeiten zur Übersetzung eingegangen, um dann einige Veröffentlichungen zur Qualitätsabschätzung der jüngeren Vergangenheit vorzustellen.

3.1. Übersetzung

Als Modell zur Übersetzung wird T5 (Raffel u. a. [20]) benutzt, welches als Sequence-to-Sequence-Modell aus dem Encoder und Decoder [27] des originalen Transformers (Vaswani u. a. [30]) (siehe Abschnitt 2.5) besteht. Der Name T5 steht für „Text-to-Text Transfer Transformer“ und das Modell ist neben Übersetzung in verschiedene Sprachen zu einigen anderen Text-zu-Text-Aufgaben wie Zusammenfassen, Fragebeantworten aber auch Klassifikation fähig.

3.2. Konfidenzabschätzung / Qualitätsabschätzung

Bei einigen Aufgaben, bei denen maschinelle Übersetzung oder Natural Language Understanding (NLU) eingesetzt wird, ist es sinnvoll, die Konfidenz des System / die Qualität der Ausgabe abschätzen zu können, um eventuelle Fehler durch Nachfragen oder eine andere Form der menschlichen Nachbearbeitung korrigieren zu können, ohne dass eine Referenzübersetzung vorliegt [23]. Hierbei gibt es verschiedene Granularitäten: Wort-, Satz- und Dokumentenebene.

Als einfachste Methode würden sich die A-posteriori-Wahrscheinlichkeiten anbieten, die das Modell „von selbst“ liefert, allerdings wurden bessere Methoden entwickelt. Wie auch zur Übersetzung selbst kann zur Qualitätsabschätzung maschinelles Lernen eingesetzt werden wie z. B. neuronale Netze, welche einen ähnlichen Weg in der aufsteigenden Komplexität der Architektur zurückgelegt haben wie jene für die Übersetzung selbst.

So wurden zunächst einfache Autoencoder verwendet (Niehues und Pham [15]), die auch mit den A-posteriori-Wahrscheinlichkeiten kombiniert wurden. Fortgeschrittenere Modelle wie deepQuest (Ive, Blain und Specia [9] und OpenKiwi (Kepler u. a. [11]) nutzen komplexe RNNs (siehe Absatz 2.4). Eine der neusten Arbeiten, welche zunächst nur für Qualitätsabschätzung auf Satzebene (Ranasinghe, Orasan und Mitkov [23] bzw. Ranasinghe, Orasan und Mitkov [22]) und dann auch auf Wortebene (Ranasinghe, Orasan und Mitkov [21]) veröffentlicht wurde, verwendet Transformer (Vaswani u. a. [30]) (siehe Abschnitt 2.5).

Ein weiteres wichtiges Gebiet für Qualitätsabschätzung ist die automatische Spracherkennung. Hier wurden verschiedene Methoden zur besseren Qualitätsabschätzung entwickelt wie einfache Algorithmen für maschinelles Lernen (Schaaf und Kemp [25]), die

Analyse von alternativen Ausgaben des Systems (Kemp und Schaaf [10]) und die Fusion mit anderen Modalitäten wie Bilddaten zur Berechnung der Konfidenz (Große, Holzapfel und Waibel [7]).

4. Modelle und Methoden

Das System soll nun in der Lage sein, natürlichsprachige Äußerungen auf Englisch in eine Form zu bringen, welche so einfach zu verarbeiten ist, dass der gewünschte Befehl ausgeführt werden kann. Da diese Übersetzung potenziell nicht korrekt sein kann, ist es sinnvoll, bei Unsicherheiten dem Benutzer Rückfragen stellen zu können. Man braucht nun also ein Modul zur Übersetzung (Abschnitt 4.1) und ein weiteres, dass dann die Qualität dieser Übersetzung abschätzt (Abschnitt 4.2), woraufhin regelbasiert spezifische Rückfragen gestellt werden können.

Sowohl das Modell für die Übersetzung als auch die Qualitätsabschätzung wurden von „Hugging Face“ [29, 32] basierend auf den Originalveröffentlichungen implementiert. Hugging Face bietet eine Sammlung von vortrainierten Transformern für Text- (für mehr als 100 Sprachen), Bild- und Audioinformationen und eine API für Download, Fine-tuning und Teilen von Modellen an. Fine-tuning beschreibt das Training eines vortrainierten Modells auf einen eigenen, spezifischen Datensatz. Durch dieses Vortraining kann Trainingszeit gespart und außerdem die Leistung gesteigert werden, falls der eigene zum Fine-tuning benutzte Datensatz eher klein ist. Hugging Face Transformer stellen dabei den Stand der Technik dar [29, 32] und sind für die Maschinelles Lernen-Bibliotheken Jax, PyTorch und TensorFlow erhältlich. In dieser Arbeit wird Hugging Face in Version 4.8.1 [1] mit PyTorch verwendet.

Bevor ein Text von einem Modell verarbeitet werden kann, muss er mithilfe eines Tokenizers in Token zerlegt werden [17]. Dazu gibt es verschiedene Ansätze [28]: Man kann den Text in einzelne Zeichen, Teilwörter oder ganze Wörter zerteilen, wobei ersteres den Nachteil hat, dass es für ein Modell deutlich schwieriger ist, eine sinnvolle, kontextfreie Repräsentation für ein einzelnes Zeichen zu finden [28]. Allerdings ist es für den Fall, dass in der Eingabe unbekannte (Teil-) Wörter vorkommen, sinnvoll, auch für einzelne Zeichen einen zugeordneten Token zu haben, um nicht jedes unbekannte (Teil-) Wort als unbekannten Token interpretieren zu müssen. Ganze Wörter als Token sind ebenfalls unpraktisch, weil dadurch das Vokabular aufgrund der großen Zahl möglicher Wörter zu groß wird und dies die Trainingsdauer negativ beeinflusst [28]. Bei der Auswahl der Teilwörter wird durch den Algorithmus darauf geachtet, dass häufig vorkommende Wörter nicht unterteilt werden. Jeder Token wird nun durch eine Zahl namens „Input ID“ repräsentiert [6], welche dann letztendlich als Eingabe für das eigentliche Modell verwendet wird. Zusätzlich gibt der Tokenizer eine „Attention Mask“ zurück, die dem Modell Hinweise gibt, welche Token relevant sind. Dies kommt im Fall von „Padding“ zur Anwendung, wenn eine Eingabe kürzer als die vom Modell festgelegte Länge ist und am Ende des Satzes mit leeren Token aufgefüllt wird. Tokenizer können außerdem die Ausgabe von Modellen, die

[1] `https://github.com/huggingface/transformers/releases/tag/v4.8.1`

Input IDs generieren (wie T5 (siehe Abschnitt 4.1)), decodieren und so menschenlesbaren Text erzeugen.

4.1. Übersetzung

Zur Übersetzung der natürlichsprachigen Befehle in eine Robotersprache wird T5 [20] (siehe Abschnitt 3.1) verwendet. Dabei werden die Befehle in das Verb (die auszuführende Aktion) samt optionalem Adverb und die Nomen (die beteiligten Objekte) zerlegt. Alle Wörter in der Robotersprache sind durch Leerzeichen getrennt. Hierzu werden alle Objekte auf einige dem Roboter bekannte Objekte (Unterabschnitt A.2.1) abgebildet.

Beispiel: please put the milk into the refrigerator → put-into milk fridge

4.2. Qualitätsabschätzung

Die Verwendung der A-posteriori-Wahrscheinlichkeiten zur Qualitätsabschätzung ist nicht die beste Methode, wie sich in vorherigen Arbeiten bereits gezeigt hat ([9, 11, 15, 21, 23]). Besonders hervorzuheben ist, dass gerade fehlende Wörter schlecht erkannt werden können, weil sich die Wahrscheinlichkeiten nur auf tatsächlich generierte Wörter beziehen. Wenn z. B. die Übersetzung aus zwei Wörtern bestehen soll, die tatsächliche vom System gelieferte Übersetzung aber nur ein Wort enthalten würde, wäre die A-posteriori-Wahrscheinlichkeit für dieses eine Wort wahrscheinlich sehr hoch, weil es korrekt übersetzt wurde, aber die Übersetzung insgesamt nicht vollständig ist.

Von den verschiedenen Ebenen von Qualitätsabschätzung war für diese Arbeit nur die Wortebene relevant, da innerhalb eines Befehls bzw. einen Satzes die Teile des Befehls bzw. die Wörter bestimmt werden sollen, die schlecht verstanden wurden. Wenn stattdessen die Qualität auf Satzebene abschätzt werden würde, könnte das System keine spezifischen Fragen stellen.

4.2.1. TransQuest

Wie alle in diesem Abschnitt genannten Modelle entspricht das von TransQuest verwendete XLM-RoBERTa (XLM-R) [2] von seiner Architektur dem Encoder [27] des originalen Transformers [30] (siehe Abschnitt 2.5). Diese Modelle werden auch „Autoencoding Models“ genannt.

XLM Zum einen basiert XLM-R auf dem sprachübergreifenden Sprachmodell XLM [3], welches während des Trainings auf verschiedene Aufgaben wie Causal Language Modeling (CLM), Masked Language Modeling (MLM) und Translation Language Modeling (TLM) trainiert wurde. Causal Language Modeling (CLM) beschreibt das Vorhersagen des nächsten Wortes, wenn ein Satz gegeben ist. Bei Masked Language Modeling (MLM) wird ein Teil der Token der Eingabe „maskiert“ / verdeckt und das Modell muss diese Token anhand des Kontexts erraten [5]. Translation Language Modeling (TLM) ist im Gegensatz zu den beiden vorherigen Aufgaben überwachtes Lernen und eine Variante von MLM, bei der sowohl in der Quellsprache als auch in der Übersetzung Wörter maskiert sind, wodurch das

Modell sowohl den Kontext der einen als auch der anderen Sprache verwenden kann. Das Vokabular wird zwischen den Aufgaben geteilt. Die Eingaben werden wie schon bei BERT [5] mit einem speziellen Separatortoken getrennt [3]. Beispiele für diese verschiedenen Aufgaben sind im Anhang (Abschnitt A.1) zu finden.

Durch das Training über verschiedene Sprachen hinweg konnten besseren Initialisierungen für Satzencoder für sprachübergreifende Klassifikation und für (un-) überwachte maschinelle Übersetzungssysteme erreicht werden. Außerdem profitierten unüberwachte sprachübergreifende Worteinbettungen („Word Embeddings“) und Sprachmodelle für Sprachen mit wenig verfügbaren Daten.

RoBERTa XLM-R ist außerdem von RoBERTa [14] inspiriert, welches gezeigt hat, dass man bei Modellen wie BERT [5] durch Anpassen der Hyperparameter und der Größe des Trainingsdatensatzes das Vortrainieren deutlich verbessern kann. Die bei MLM verwendete Maske war bei BERT statisch, sie wurde einmal während der Vorverarbeitung der Daten erstellt. Es zeigt sich, dass eine dynamische Maske, die vor der Eingabe jeder Sequenz neu generiert wird, die Leistung des Modells leicht steigert. Darüber hinaus wurde festgestellt, dass mehr Daten pro Trainingsschritt (bei insgesamt weniger Trainingsschritten) das Modell verbessern. Außerdem wird Byte-Pair Encoding (BPE) [26] auf Byte-level [19] verwendet. Dies ist eine Form des Teilwort-Tokenizers, der zunächst den Text in Wörter unterteilt, deren Häufigkeit bestimmt, die Wörter wiederum in einzelne Zeichen unterteilt und dann die häufigsten Symbolpaare als Teilwörter nimmt [28]. Bei Byte-level BPE benutzt man wiederum Bytes anstelle von Unicode-Zeichen als Basis der Teilwörter. Dadurch kam es allerdings nicht zu einer reinen Verbesserung der Ergebnisse des Modells, es wurde aber von den Autoren als vorteilhaft angesehen [14]. Zum Schluss wurden diese Erkenntnisse in ein Modell vereint, welches dann auch noch mit mehr Daten und länger als BERT trainiert wurde. All diese Faktoren zusammen verbesserten die Leistung deutlich.

XLM-R Im Gegensatz zu XLM mit 15 Sprachen ist XLM-RoBERTa (XLM-R) deutlich hoch skaliert und auf 100 Sprachen trainiert [2], wobei ausschließlich MLM als Aufgabe zum Einsatz kam. XLM-R nutzt außerdem die Erkenntnisse von RoBERTa und verbessert damit die Leistung von XLM bei MLM erheblich. Dadurch übersteigt es die Leistung von mBERT, eine Version von BERT für mehrere Sprachen. Die sprachübergreifende Eigenschaft von XLM-R schlägt zudem monosprachliche Modelle. XLM-R ist für Aufgaben wie Named Entity Recognition (NER), Fragebeantworten und Klassifikation einsetzbar. In NER muss jedem einzelnen Token der Eingabe eine Klasse zugewiesen werden. Zudem ist es gut für sprachübergreifende Worteinbettungen geeignet. Diese Anwendung ist für TransQuest relevant, da TransQuest für die Qualitätsabschätzung von Übersetzungen entwickelt wurde und zusätzlich eines der Ziele von TransQuest war, sprachübergreifend zu sein [23]. So muss nicht für jedes Sprachpaar neu trainiert werden und darüber hinaus profitieren Sprachen mit wenig verfügbaren Daten davon.

TransQuest Die Funktionsweise von TransQuest auf Wortebene (Abbildung 4.1) ist folgendermaßen [21]: In den Zieltext werden zwischen jeden Token (hier Wörter, da der Satz an Leerzeichen aufgespalten wird) eine Lücke (`<GAP>`) eingefügt. Der Sinn dieser Lücken ist, dass so fehlende Token in der Übersetzung markiert werden können. So könnte z. B. das erste Wort fehlen, wodurch hier eine Lücke wäre und somit das erste Tag für den

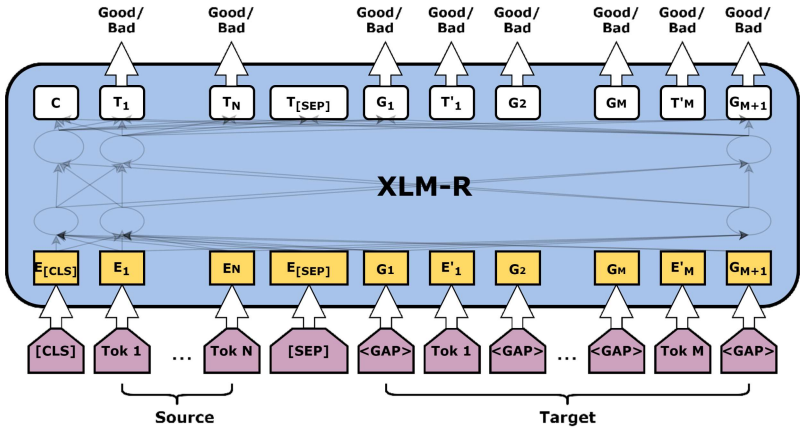

Abbildung 4.1.: Architektur von TransQuest auf Wortebene [21]

Zieltext negativ wäre. Dann werden Quelle und Ziel mit dem Seperatortoken `[SEP]` zwischen beiden konkateniert und von XLM-R verarbeitet, um Worteinbettungen zu erhalten. Vor jede Eingabe wird der spezielle Token `[CLS]` eingefügt wie auch schon bei BERT [5]. Zum Schluss folgt eine einfache lineare Schicht, die jedem Token entweder ein „`OK`“- oder „`BAD`“-Tag zuordnet. Weil diese Schicht recht simpel ist, reduziert dies die Trainingsdauer im Vergleich zu anderen Lösungen [23]. Dann hat man eine Qualitätsabschätzung für die Übersetzung und zugleich für die Quelle. Dort werden jene Wörter mit `BAD` markiert, welche Fehler in der Übersetzung verursachen. Das ist aber für diese Arbeit wenig relevant, da die Rückfragen des Systems an den Nutzer anhand der fehlenden / falschen Wörter in der Übersetzung geschehen sollen.

Rückfragen Wenn das Gesamtsystem interaktiv benutzt wird, soll das Ergebnis dieser Qualitätsabschätzung als Basis für simple regelbasierte, spezifische Rückfragen dienen. Wenn nur der erste Tag, der auf die erste Lücke vor dem ersten Wort in der Übersetzung verweist, `BAD` ist, wird dies als fehlende Aktion interpretiert und eine Rückfrage gestellt, welche Aktion mit dem erkannten Objekt ausgeführt werden soll. Das gleiche gilt für den Fall, dass der zweite Tag - also der Tag für das erste Wort in Robotersprache - negativ ist. So ähnlich wird für die Objekte verfahren. Falls Lücken oder Wörter im hinteren Bereich (alles hinter den ersten beiden Tags) der Übersetzung als falsch eingestuft sind, wird der Benutzer gefragt, welche Objekte bei dieser Aktion zum Einsatz kommen. Dabei werden auch die erkannten Objekte aufgelistet. Wenn nun zu viele Fehler vorkommen, bittet das System die Frage umzuformulieren. Beispiele von Übersetzungen, folgender Qualitätsabschätzung und eventuellen Rückfragen sind im Anhang zu finden (Unterabschnitt A.2.2).

5. Evaluation

Im Folgenden wird der verwendete Datensatz vorgestellt und wie er für diese Arbeit angepasst bzw. erweitert wurde. Danach wird auf das Training der Modelle und die dabei zum Einsatz kommenden Metriken eingegangen. Zum Schluss folgt eine Beschreibung der Ergebnisse der hier im Fokus liegenden Qualitätsabschätzung und letztendlich des Gesamtsystems.

5.1. Datensatz

Der Datensatz, der sowohl zum Trainieren der Übersetzung als auch der Qualitätsabschätzung und der abschließenden Evaluation sowohl der einzelnen Module als auch des Gesamtsystems dient, basiert auf dem „EPIC KITCHENS-100 Dataset“ [1] [4]. Dieser besteht aus audio-visuellen Aufnahmen aus der 1. Person von Aktivitäten in der Küche. Er enthält diese als Befehle in roher Textform und daraus extrahiert die Verben / Aktionen, die ausgeführt werden, und Nomen / Objekte, die dabei beteiligt sind. Die Sätze beginnen nahezu immer mit dem Verb außer in Fällen wie „still grating carrot“, darauf folgen die Nomen und Adverbien etc. Es wird immer nur eine Aktion beschrieben, die aber mehrere Objekte betreffen kann. Ein Beispiel mit zwei Objekten wäre, wenn mit einem Löffel in einer Schüssel umgerührt werden soll.

Der Datensatz teilt sich in Trainings- und Validierungsteil auf. Für diese Arbeit wurde der Trainingsteil aufgeteilt in zwei Mengen, die während des Trainingsvorgangs für Training und Validierung verwendet werden (Tabelle 5.1). Der Validierungsteil wird als Testdatensatz benutzt, welcher zum Schluss in der abschließenden Evaluation zum Einsatz kommt. Zusätzlich existieren Verb- und Nomenklassen (Unterabschnitt A.2.1), die eine Menge an Verben / Nomen auf einen Vertreter abbilden. Für Nomen gibt es 300 Vertreter, für Verben 97.

Datensatz	Anzahl Sätze
Training (original)	67217
Training (80 % des originalen Training)	53774
Validierung (20 % des originalen Training)	13443
Test (original: Validierung)	9669

Tabelle 5.1.: Größe der einzelnen Datensätze vor Anreicherung

[1] `https://github.com/epic-kitchens/epic-kitchens-100-annotations`, Commit 8feb43e6a8744593785b96a59631251967ee44ba

Weil die Befehle im originalen Datensatz in der Quellsprache recht kurz sind und unnatürlich klingen, werden sie für diese Arbeit um Höflichkeitsformen wie „please“ und „could / would you“ erweitert. An der Übersetzung ändert sich nichts. Dadurch existiert jeder Satz mindestens doppelt - einmal in einfacher Form und einmal in längerer Form.

Die Übersetzung läuft folgendermaßen: Aus dem englischen Befehl soll eine Ausgabe in Robotersprache erzeugt werden, die eine Folge aus dem Verb (Aktion) und den Nomen (Objekte) ist. Dabei sollen alle Nomen auf ihren dem System bekannten Vertreter abgebildet werden. Eine korrekte Übersetzung besteht nun aus mindestens zwei Wörtern, die durch Leerzeichen getrennt sind: Eine Aktion, die immer an erster Stelle steht, und danach alle Objekte in logischer Folge passend zur Aktion. Wenn z. B. die Aktion lautet, ein Objekt in ein Behältnis zu stellen, erscheint das betreffende zu bewegende Objekt immer an erster Stelle nach der Aktion.

Der originale Datensatz enthält nur korrekte Übersetzungen. Damit TransQuest trainiert werden kann, werden allerdings auch fehlerhafte Übersetzungen benötigt und dazu entsprechende Tags für beide Sprachen. Deshalb wird für jeden Satz eine Kopie mit gleicher Eingabe aber falscher Ausgabe generiert. In den falschen Übersetzungen werden die nicht korrekt übersetzten Wörter bzw. die Lücken bei fehlenden Wörtern entsprechend getaggt. Für TransQuest existieren also vier Datenspalten: Quelltext in Englisch, Tags für den Quelltext, Zieltext in Robotersprache und Tags für den Zieltext. Folgende Arten von fehlerhaften Übersetzungen, welche sich in den Übersetzungen von T5 häufig zeigten, werden generiert:

Befehl: „would you put cheese in the refrigerator“

Übersetzung		put-in		cheese		fridge	
Tags	OK	OK	OK	OK	OK	OK	OK

Tabelle 5.2.: Korrekte Übersetzung

Ein Wort aus der korrekte Übersetzung fehlt (Tabelle 5.3). Dies kann das Verb am Anfang oder eines der Nomen danach sein. Hierfür wird die Lücke getaggt, an deren Stelle das Wort stehen sollte.

Übersetzung		put-in		fridge	
Tags	OK	OK	BAD	OK	OK

Tabelle 5.3.: Fehlertyp: Fehlendes Wort

Ein Wort wird durch ein anderes ersetzt (Tabelle 5.4), dieses Wort steht dann zweimal in der Übersetzung. Es wird nur das als falsch markiert, welches an der falschen Stelle steht.

Übersetzung		put-in		fridge		fridge	
Tags	OK	OK	OK	BAD	OK	OK	OK

Tabelle 5.4.: Fehlertyp: Ersetztes Wort

Damit nicht einfach wahllos Wörter aus dem Originaltext übernommen werden, enthalten manche falsche Übersetzungen ein zufälliges Wort aus dem Original an zufälliger (falscher) Stelle (Tabelle 5.5).

Übersetzung		put-in		would		cheese		fridge	
Tags	OK	OK	OK	BAD	OK	OK	OK	OK	OK

Tabelle 5.5.: Fehlertyp: Zufälliges Wort

Alle Nomen sollen auf ihren Vertreter abgebildet werden. Bei korrumpierten Übersetzungen wird dies nicht getan (Tabelle 5.6) und das nicht abgebildete Nomen als falsch getaggt.

Übersetzung		put-in		cheese		refrigerator	
Tags	OK	OK	OK	OK	OK	BAD	OK

Tabelle 5.6.: Fehlertyp: Nomen nicht abgebildet

Viele Verben haben ein Adverb, von welchem sie mit einem Bindestrich getrennt sind. In der fehlerhaften Übersetzung fehlt das Adverb samt Bindestrich (Tabelle 5.7) und das ganze Verb wird als fehlerhaft markiert.

Übersetzung		put		cheese		fridge	
Tags	OK	BAD	OK	OK	OK	OK	OK

Tabelle 5.7.: Fehlertyp: Fehlendes Adverb

Eine der Fehlerarten wird nun für eine falsche Übersetzung zufällig ausgewählt. Dabei ist die Wahrscheinlichkeit gleichverteilt. TransQuest erhält nun jeweils eine korrekte und eine falsche Version in Robotersprache desselben Befehls auf Englisch mit entsprechenden Tags. Somit wäre der Datensatz für TransQuest doppelt so groß wie der für das Training von T5. Allerdings deuten Beispieldatensätze von TransQuest und die Ergebnisse während des Trainings mit diesem Datensatz daraufhin, dass weniger Daten ausreichen, um in akzeptabler Trainingszeit ausreichend gute Ergebnisse zu erzielen. Daher verwendete TransQuest zunächst nur 40 % der Daten (bzw. 20 % der originalen Sätze vor der Datenanreicherung) wie T5. Allerdings zeigte sich später, dass 50 % der originalen Daten die Ergebnisse noch einmal verbessern und dass dies auch noch in vertretbarer Zeit geschieht.

Für die Evaluation von Übersetzung und Qualitätsabschätzung zusammen wird aber der volle (Test-) Datensatz für die Übersetzung verwendet.

5.2. Training

Beide Modelle des Systems verwenden Metriken, um ihre Qualität nach jedem Trainingsschritt bewerten zu können, damit nur der beste Zustand des Modells gespeichert wird, um später benutzt zu werden.

T5 Die Qualität von T5 wird während des Trainings mithilfe von BLEU (bilingual evaluation understudy) [16] berechnet. Dies ist ein sprachunabhängiger Algorithmus zur automatischen Evaluation von maschineller Übersetzung, welcher versucht, möglichst nah an menschlicher Evaluation zu sein. Dafür wird jeder übersetzte Satz mit Referenzübersetzungen verglichen und dann die dabei berechneten Werte über alle Sätze gemittelt. Für jeden Satz wird die Anzahl der Wörter berechnet, die in den Referenzübersetzungen vorkommen. Dabei werden zu häufiges Vorkommen einzelner Wörter und stark abweichende Satzlängen bestraft. Am Ende ergibt dies einen BLEU-Wert zwischen 0 (schlecht) und 1 (gut, sehr ähnlich zu Referenz). Alternativ werden die Werte wie im Fall dieser Arbeit als Prozentwerte angegeben.

Zum Training wird ein Projekt verwendet, das auf GitHub veröffentlicht wurde [2]. Für die Aufgabe in dieser Arbeit durchläuft T5 während des Trainings 16 Epochen. Obwohl der Fehler pro Epoche immer weiter abnahm, verbesserte sich der BLEU-Wert im Laufe der Entwicklung nach 10 bis 12 Durchläufen nicht mehr. Deswegen wurde 16 gewählt, um sicher zu gehen, dass der beste Wert erreicht wird, ohne die Trainingsdauer zu stark zu verlängern. Wie sich zeigt (Unterabschnitt A.3.1), springt der Wert schnell auf über 90 % und verbessert sich dann nur noch leicht auf über 95 %. Während der Entwicklung zeigten sich ähnliche Ergebnisse. Da T5 ein mächtiges Modell zur Verarbeitung natürlicher Sprache ist und diese Übersetzungsaufgabe mit diesen Daten relativ simpel ist, ist ein fast perfekter BLEU-Wert nicht überraschend.

TransQuest Der Code von TransQuest ist ebenfalls auf GitHub erhältlich [3]. Für TransQuest werden neben Trainings- und Validierungsfehler einige Metriken für binäre Klassifikation angeben, da ein Wort bzw. eine Lücken nur richtig oder falsch getaggt werden kann. Allgemein gibt es bei der Bewertung binärer Klassifikation, wie sie auch ähnlich im späteren Abschnitt (Abschnitt 5.3) verwendet wird, vier Ergebnisse:

	Modell positiv	Modell negativ
Tatsächlich positiv	Richtig positiv (RP)	Falsch negativ (FN)
Tatsächlich negativ	Falsch positiv (FP)	Richtig negativ (RN)

Tabelle 5.8.: Vier mögliche Ergebnisse bei binärer Klassifikation, falsch positiv heißt auch „falscher Alarm“)

[2] `https://github.com/keleog/finetune_huggingface_t5` , Commit 461d9079eb731b3580f862a9142a237695d259e1
[3] `https://github.com/tharindudr/transQuest`, Commit fd9714050ea73201119f1c70b060b3faee86c133

TransQuest verwendet drei davon abgeleitete Werte:

1. **Genauigkeit (g)** ist das Verhältnis von RP zu allen vom Modell vorhergesagten Positiven, also wie viele der als korrekt abgeschätzten Übersetzungen tatsächlich korrekt sind.
2. **Sensitivität (s)** ist das Verhältnis von RP zu allen tatsächlich Positiven, also wie viele der korrekten Übersetzungen als korrekt abgeschätzt werden.
3. **F1-Wert** wird nun aus Genauigkeit und Sensitivität berechnet mittels $F_1 = 2\frac{gs}{g+s}$.

Die Beispielkonfigurationen für das Training von TransQuest verwenden den Validierungsfehler, um zu entscheiden, wann das Training abgebrochen wird und welche Version des Modells am Ende gespeichert wird. Dieser Wert scheint auch gut mit den anderen Metriken zu korrelieren, wie man bei Schritt 8100 sehen kann (Unterabschnitt A.3.2). Auch die anderen Metriken sind deutlich über 0,8 oder sogar 0,9. Jedoch liefert der F1-Wert als Auswahlkriterium ein Modell mit leicht verbesserter Leistung.

A-posteriori-Wahrscheinlichkeiten Wenn man mit T5 mit einem Text als Eingabe einen neuen Text generiert, was in diesem Fall die Übersetzung von Englisch in die Robotersprache ist, kann man für jeden generierten Token alle Werte erhalten, die das Modell für jeden im Vokabular enthaltenen Token berechnet hat. Mittels Softmax (Gleichung 2.12) kann man diese Werte in Wahrscheinlichkeitswerte zwischen 0,0 und 1,0 umrechnen. Der höchste Wert von allen ist der für den tatsächlich generierten Token. Diese A-posteriori-Wahrscheinlichkeit stellt dar, wie sicher sich das Modell bei der Generierung dieses Tokens bezogen auf alle anderen war. Da hier nur die Wahrscheinlichkeit für ganze Wörter relevant ist, wird der Durchschnitt aller Wahrscheinlichkeiten pro Token pro Wort als Wahrscheinlichkeit für das ganze Wort verwendet.

Um jetzt mittels dieser A-posteriori-Wahrscheinlichkeiten die Qualität der Übersetzung abschätzen zu können, muss man einen Grenzwert finden, ab dem ein Wort mit `OK` oder `BAD` getaggt wird, um eine Rückfrage zu stellen bzw. die Ergebnisse mit TransQuest vergleichen zu können. Dafür wird eine Art Training verwendet, bei der zunächst jeder Wert zwischen 0,0 und 1,0 im Abstand 0,1 als Grenzwert zum Einsatz kommt. Der mit den wenigsten falsch getaggten Übersetzungen wird als neuer Startpunkt genutzt und der Suchbereich um diesen Wert und die Schrittweite immer weiter eingeschränkt. Hierbei wird der Validierungsdatensatz verwendet, da dieser auch während des Trainings der Modelle dazu dient, deren Leistung mit der gewählten / trainierten Parametern zu bewerten.

Um die Qualitätsabschätzungen von TransQuest und die durch die A-posteriori- Wahrscheinlichkeiten zu kombinieren, muss bei unterschiedlich getaggten Wörtern entschieden werden, welche der beiden Qualitätsabschätzungen man schwerer gewichtet. Da TransQuest die besseren Ergebnisse liefert, sollte diese Abschätzung als sicherer betrachtet werden. Deshalb bietet es sich an, für die Wahrscheinlichkeiten zwei verschiedene Grenzwerte zu benutzen. Wenn TransQuest für ein Wort ein `OK`- bzw. ein `BAD`-Tag gibt, muss die A-posteriori-Wahrscheinlichkeit also sehr gering bzw. sehr hoch sein, um TransQuest zu „überstimmen“. Für Lücken muss keine Entscheidung getroffen werden, da sich A-posteriori-Wahrscheinlichkeiten nur auf die generierten Wörter beziehen.

5.3. Ergebnisse

Bei der Qualitätsabschätzung können vier bzw. fünf Arten von Ergebnissen auftreten:

1. Übersetzung korrekt, Qualitätsabschätzung bestätigt das
2. Übersetzung korrekt, Qualitätsabschätzung erkennt es nicht
3. Übersetzung falsch, Qualitätsabschätzung bestätigt das
4. Übersetzung falsch, Qualitätsabschätzung erkennt es nicht
5. Übersetzung falsch, Qualitätsabschätzung erkennt es an der falschen Stelle

Am interessantesten ist der dritte Punkt, dies ist das Ziel dieser Arbeit, damit man im besten Fall am Ende keine falschen Übersetzungen mehr hat, da alle durch Rückfragen korrigiert wurden. Dementsprechend gilt es, das Auftreten des vierten Falls zu reduzieren. Außerdem sollten nicht zu oft Rückfragen gestellt werden, obwohl die Übersetzung eigentlich korrekt war (Fall zwei). Hier muss die richtige Balance zwischen dem Finden von Fehlern und der Vermeidung unnötiger Rückfragen gefunden werden.

TransQuest Zunächst kann man TransQuest einfach mit den generierten Testdaten evaluieren, wobei sich sehr gute Ergebnisse zeigen (Tabelle 5.9, Tabelle 5.10). Im Zuge der Entwicklung der Algorithmen für die Generierung der fehlerhaften Übersetzungen waren die Werte ebenfalls in diesem Bereich. Richtige Übersetzungen werden fast immer korrekt erkannt. Wenn die Übersetzung hingegen fehlerhaft ist, wird dies öfter gar nicht erkannt, in den meisten Fällen ist die Qualitätsabschätzung aber vollständig korrekt. Dass die Qualitätsabschätzung zwar einen Fehler erkennt, aber die Tags nicht alle an der richtigen Stelle sind, kommt sehr selten vor. Das Modell, das anhand des F1-Wertes ausgewählt wurde, erkennt etwas mehr falsche Übersetzungen, markiert jedoch auch leicht mehr korrekte Übersetzungen als falsch. Die A-posteriori-Wahrscheinlichkeiten kann man hier nicht verwenden, da diese vom Übersetzungsmodell geliefert werden, welches hier noch fehlt.

Qualitätsabschätzung erkennt richtige Übersetzung korrekt	0,5683
Qualitätsabschätzung erkennt richtige Übersetzung nicht	0,0083
Qualitätsabschätzung erkennt falsche Übersetzung vollständig korrekt	0,3630
Qualitätsabschätzung erkennt falsche Übersetzung teilweise korrekt	0,0107
Qualitätsabschätzung erkennt falsche Übersetzung nicht	0,0497

Tabelle 5.9.: Evaluation von TransQuest (20 % Daten, bester Validierungsfehler) auf Testdaten

Qualitätsabschätzung erkennt richtige Übersetzung korrekt	0,5662
Qualitätsabschätzung erkennt richtige Übersetzung nicht	0,0103
Qualitätsabschätzung erkennt falsche Übersetzung vollständig korrekt	0,3773
Qualitätsabschätzung erkennt falsche Übersetzung teilweise korrekt	0,0113
Qualitätsabschätzung erkennt falsche Übersetzung nicht	0,0349

Tabelle 5.10.: Evaluation von TransQuest (20 % Daten, bester F1-Wert) auf Testdaten

Nachdem der F1-Wert als Auswahlkriterium feststand, wurde das Modell auch nochmal mit mehr (auf Basis von 50 % statt 20 % der Daten für das Training der Übersetzung) Daten trainiert und wieder evaluiert. Dabei werden etwas weniger richtige Übersetzungen als falsch getaggt, obwohl etwas weniger richtige Übersetzungen in dieser Version der generierten Daten sind, und korrekt nur teilweise richtig getaggt. Dafür steigen die verpassten Fehler wieder leicht.

Qualitätsabschätzung erkennt richtige Übersetzung korrekt	0,5673
Qualitätsabschätzung erkennt richtige Übersetzung nicht	0,0057
Qualitätsabschätzung erkennt falsche Übersetzung vollständig korrekt	0,3811
Qualitätsabschätzung erkennt falsche Übersetzung teilweise korrekt	0,0099
Qualitätsabschätzung erkennt falsche Übersetzung nicht	0,0359

Tabelle 5.11.: Evaluation von TransQuest (50 % Daten, bester F1-Wert) auf Testdaten

T5 und TransQuest Wenn man nun den Testdatensatz von T5 übersetzen lässt und dann von TransQuest die Qualität dieser Übersetzung abschätzen lässt, kann man dies mit den A-posteriori-Wahrscheinlichkeiten zur Qualitätsabschätzung vergleichen und darüber hinaus beide Ansätze kombinieren.

	Übersetzung korrekt	Übersetzung falsch
Qualitätsabschätzung korrekt	0,9068	0,0420
Qualitätsabschätzung falsch	0,0108	0,0405

Tabelle 5.12.: TransQuest als Qualitätsabschätzung für T5

	Übersetzung korrekt	Übersetzung falsch
Qualitätsabschätzung korrekt	0,8945	0,0347
Qualitätsabschätzung falsch	0,0230	0,0478

Tabelle 5.13.: A-posteriori-Wahrscheinlichkeiten als Qualitätsabschätzung für T5

	Übersetzung korrekt	Übersetzung falsch
Qualitätsabschätzung korrekt	0,8968	0,0509
Qualitätsabschätzung falsch	0,0207	0,0316

Tabelle 5.14.: TransQuest und A-posteriori-Wahrscheinlichkeiten als Qualitätsabschätzung für T5

Man kann sehen, dass TransQuest (Tabelle 5.12) fast die Hälfte der falschen Übersetzungen erkennt und dabei bei nur etwa 1 % der korrekten Übersetzungen fälschlicherweise

einen Fehler anzeigt. Mit den A-posteriori-Wahrscheinlichkeiten (Tabelle 5.13) hingegen werden mehr als 2 % der richtigen Übersetzungen als fehlerhaft markiert. Auch die Anzahl der erkannten falschen Übersetzungen sind etwa 20 % weniger. Wenn man nun beide Ansätze kombiniert (Tabelle 5.14), hat man immer noch etwas mehr Rückfragen als beim alleinigen Einsatz von TransQuest. Auf der anderen Seiten werden jedoch auch nochmal 20 % mehr Fehler entdeckt.

6. Fazit

Das Ziel in dieser Arbeit war es, ein Modell für sprachübergreifende Qualitätsabschätzung für maschinelle Übersetzung auf die Aufgabe von Natural Language Understanding (NLU) für Roboterbefehle zu übertragen. Hierzu wurden die Befehle in eine einfache Robotersprache gebracht, worauf die Qualitätsabschätzung angewendet werden konnte.

Zunächst einmal sieht man, dass die Übersetzungen in eine Robotersprache einfach genug sind, dass weniger als 10 % der Übersetzungen falsch sind. Von diesen falschen Übersetzungen kann durch die Qualitätsabschätzung mittels TransQuest etwas die Hälfte erkannt werden. Unnötige Rückfragen treten viermal weniger auf als sinnvolle. Dieses Ziel der Fehlerfindung konnte also erreicht werden.

Die Fehlererkennung durch A-posteriori-Wahrscheinlichkeiten ist hingegen, wie zu erwarten war, deutlich schlechter. Gerade die Menge unnötiger Rückfragen durch das System stiegen dadurch. Durch eine Kombination beider Methoden können jedoch mehr Fehler gefunden werden.

6.1. Zukünftige Arbeiten

Hinsichtlich der Generierung der Daten für TransQuest würden sich eventuell noch andere Möglichkeiten bieten: Zum einen könnte man die fehlerhaften Übersetzungen aus Trainings- und Validierungsdaten von T5 auf die tatsächlich enthaltenen Fehler analysieren, diese taggen und diese Daten zusätzlich zu den „künstlich“ generierten zum Training verwenden. Außerdem könnte man die Verteilung der Fehlerarten ermitteln und entsprechend die Generierung anpassen.

Die Qualitätsabschätzung durch TransQuest ist im Gegensatz zu den A-posteriori-Wahrscheinlichkeiten nicht kontinuierlich - ein Wort wird entweder als `OK` oder `BAD` getaggt. Allerdings sollte man aus TransQuest wie auch aus T5 zuvor die A-posteriori-Wahrscheinlichkeiten extrahieren können, wodurch man bei der Kombination von beiden jeweils unterschiedliche Schwellwerte setzen und darüber hinaus beide mit verschiedener Gewichtung in eine gemeinsame Qualitätsabschätzung einfließen lassen könnte.

Bei der Erzeugung der Befehle in Robotersprache durch T5 passieren recht wenige Fehler, etwa 10 % sind nicht ganz korrekt. Um das System in Kombination genaueren Tests zu unterziehen, müsste man einen Datensatz verwenden, bei dem deutlich mehr Fehler auftreten.

Obwohl das System nun einen großen Teil der fehlerhaften Übersetzungen erkennt und selten bei korrekten fälschlicherweise Rückfragen stellt, zeigen sich doch Grenzen: So kann z. B. nicht bestimmt werden, ob ein Befehl semantisch sinnvoll ist. Dass der Roboter den ganzen Kühlschrank zum Benutzer bringt, ist zwar ein syntaktisch valider Befehl, sollte aber zumindest eine Rückfrage auslösen, ob dies wirklich die gewollte Aktion ist.

Auch das Wegwerfen eines Küchengeräts sollte nicht kommentarlos ausgeführt werden. Hier wäre ein höheres Wissen über Tätigkeiten in der Küche - eine Art „Weltwissen“ - sinnvoll, womit die Sinnhaftigkeit der Befehle hinterfragt werden könnten. Dazu könnten Modelle wie GPT [18] und dessen Nachfolger hilfreich sein.

Literatur

[1] Dzmitry Bahdanau, Kyunghyun Cho und Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[2] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer und Veselin Stoyanov. „Unsupervised Cross-lingual Representation Learning at Scale“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Juli 2020, S. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747.

[3] Alexis CONNEAU und Guillaume Lample. „Cross-lingual Language Model Pretraining“. In: *Advances in Neural Information Processing Systems*. Hrsg. von H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox und R. Garnett. Bd. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf.

[4] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price und Michael Wray. „Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100“. In: *International Journal of Computer Vision* 130.1 (Jan. 2022), S. 33–55. ISSN: 1573-1405. DOI: 10.1007/s11263-021-01531-2. URL: https://doi.org/10.1007/s11263-021-01531-2.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee und Kristina Toutanova. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Juni 2019, S. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[6] *Glossary*. Hugging Face. 2022. URL: https://huggingface.co/docs/transformers/glossary (besucht am 20. 01. 2022).

[7] Philipp W.L. Große, Hartwig Holzapfel und Alex Waibel. „Confidence Based Multimodal Fusion for Person Identification“. In: *Proceedings of the 16th ACM International Conference on Multimedia*. MM '08. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2008, S. 885–888. ISBN: 9781605583037. DOI: 10.1145/1459359.1459513. URL: https://doi.org/10.1145/1459359.1459513.

[8] Sepp Hochreiter und Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Computation* 9.8 (Nov. 1997), S. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[9] Julia Ive, F. Blain und Lucia Specia. „deepQuest: A Framework for Neural-based Quality Estimation“. In: *COLING*. 2018.

[10] Thomas Kemp und Thomas Schaaf. „Estimating confidence using word lattices“. In: *Fifth European Conference on Speech Communication and Technology*. Citeseer. 1997.

[11] Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera und André F. T. Martins. „OpenKiwi: An Open Source Framework for Quality Estimation“. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Florence, Italy: Association for Computational Linguistics, Juli 2019, S. 117–122. DOI: 10.18653/v1/P19-3020. URL: https://aclanthology.org/P19-3020.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard und L. D. Jackel. „Backpropagation Applied to Handwritten Zip Code Recognition“. In: *Neural Computation* 1.4 (Dez. 1989), S. 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541. eprint: https://direct.mit.edu/neco/article-pdf/1/4/541/811941/neco.1989.1.4.541.pdf. URL: https://doi.org/10.1162/neco.1989.1.4.541.

[13] Y. Lecun, L. Bottou, Y. Bengio und P. Haffner. „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324. DOI: 10.1109/5.726791.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer und Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[15] Jan Niehues und Ngoc-Quan Pham. „Modeling Confidence in Sequence-to-Sequence Models“. In: *Proceedings of the 12th International Conference on Natural Language Generation*. Tokyo, Japan: Association for Computational Linguistics, Okt. 2019, S. 575–583. DOI: 10.18653/v1/W19-8671. URL: https://aclanthology.org/W19-8671.

[16] 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 of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Juli 2002, S. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040.

[17] *Preprocessing*. Hugging Face. 2022. URL: https://huggingface.co/docs/transformers/preprocessing (besucht am 20. 01. 2022).

[18] Alec Radford, Karthik Narasimhan, Tim Salimans und Ilya Sutskever. „Improving language understanding by generative pre-training“. In: (2018). URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

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

[20] 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: *Journal of Machine Learning Research* 21.140 (2020), S. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[21] Tharindu Ranasinghe, Constantin Orasan und Ruslan Mitkov. „An Exploratory Analysis of Multilingual Word-Level Quality Estimation with Cross-Lingual Transformers“. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, Aug. 2021, S. 434–440. DOI: 10.18653/v1/2021.acl-short.55. URL: https://aclanthology.org/2021.acl-short.55.

[22] Tharindu Ranasinghe, Constantin Orasan und Ruslan Mitkov. „TransQuest at WMT2020: Sentence-Level Direct Assessment“. In: *Proceedings of the Fifth Conference on Machine Translation*. Online: Association for Computational Linguistics, Nov. 2020, S. 1049–1055. URL: https://www.aclweb.org/anthology/2020.wmt-1.122.

[23] Tharindu Ranasinghe, Constantin Orasan und Ruslan Mitkov. „TransQuest: Translation Quality Estimation with Cross-lingual Transformers“. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dez. 2020, S. 5070–5081. DOI: 10.18653/v1/2020.coling-main.445. URL: https://aclanthology.org/2020.coling-main.445.

[24] D. Rumelhart, G. Hinton und R. Williams. „Learning representations by back-propagating errors“. In: *Nature* 323 (1986), S. 533–536. DOI: 10.1038/323533a0.

[25] T. Schaaf und T. Kemp. „Confidence measures for spontaneous speech recognition“. In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Bd. 2. 1997, 875–878 vol.2. DOI: 10.1109/ICASSP.1997.596075.

[26] Rico Sennrich, Barry Haddow und Alexandra Birch. „Neural Machine Translation of Rare Words with Subword Units“. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, S. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162.

[27] *Summary of the models*. Hugging Face. 2022. URL: https://huggingface.co/docs/transformers/model_summary (besucht am 06. 01. 2022).

[28] *Summary of the tokenizers*. Hugging Face. 2022. URL: https://huggingface.co/docs/transformers/tokenizer_summary (besucht am 20. 01. 2022).

[29] *Transformers*. Hugging Face. 2022. URL: https://huggingface.co/docs/transformers/index (besucht am 16. 01. 2022).

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser und Illia Polosukhin. „Attention is All You Need“. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, S. 6000–6010. ISBN: 9781510860964.

[31] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano und K.J. Lang. „Phoneme recognition using time-delay neural networks“. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (1989), S. 328–339. DOI: 10.1109/29.21701.

[32] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest und Alexander M. Rush. „Transformers: State-of-the-Art Natural Language Processing“. In: Association for Computational Linguistics, Okt. 2020, S. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

A. Anhang

A.1. Modelle

Beispiele für die verschiedenen Aufgaben, die während des Trainings von z. B. XLM benutzt wurden:

Causal Language Modeling (CLM)
Eingabe:
„Ich habe meine Bachelorarbeit auf Deutsch“
Als Ausgabe sollte dann so etwas wie „geschrieben“ oder „verfasst“ folgen.

Masked Language Modeling (MLM)
Eingaben:

- „Ich habe meine **MASKE** auf Deutsch verfasst.“
- „Ich **MASKE** meine Bachelorarbeit auf Deutsch verfasst.“
- „Ich habe meine Bachelorarbeit auf **MASKE** verfasst.“

Einer der Tokens (hier ein Wort) im Satz ist maskiert und das Model muss dann mit dem Kontext den passenden Token bestimmen.

Translation Language Modeling (TLM)
Eingaben:

- „Ich habe meine **MASKE** auf Deutsch verfasst.“ ↔ „I wrote my bachelor thesis in **MASKE**.“
- „Ich **MASKE** meine Bachelorarbeit auf Deutsch verfasst.“ ↔ „I wrote my **MASKE** thesis in German.“
- „Ich habe meine Bachelorarbeit auf **MASKE** verfasst.“ ↔ „I **MASKE** my bachelor thesis in German..“

Hier sind immer Satzpaare in verschiedenen Sprache gegeben, wodurch das Modell die Kontexte von beiden Texten benutzen kann, um den maskierten Token zu ersetzen.

A.2. Datensatz

A.2.1. Bekannte Nomen / Objekte

ID	Objekte	ID	Objekte	ID	Objekte	ID	Objekte
0	tap	35	tray	70	dishwasher	105	lemon
1	spoon	36	bin	71	mixture	106	juicer
2	plate	37	pepper	72	cucumber	107	wrap
3	cupboard	38	salt	73	clothes	108	scale
4	knife	39	colander	74	peach	109	rest
5	pan	40	jar	75	flour	110	rack:drying
6	lid	41	carrot	76	courgette	111	alarm
7	bowl	42	top	77	filter	112	salmon
8	drawer	43	tomato	78	butter	113	freezer
9	sponge	44	kettle	79	scissors	114	light
10	glass	45	pasta	80	chopstick	115	spreads
11	hand	46	oven	81	tofu	116	squash
12	fridge	47	sauce	82	blender	117	leek
13	cup	48	skin	83	olive	118	cap
14	fork	49	paper	84	mat	119	fish
15	bottle	50	maker:coffee	85	spice	120	lettuce
16	onion	51	garlic	86	sausage	121	curry
17	cloth	52	towel	87	peeler:potato	122	seed
18	board:chopping	53	egg	88	napkin	123	foil
19	bag	54	rubbish	89	cover	124	machine:washing
20	spatula	55	rice	90	microwave	125	corn
21	container	56	mushroom	91	pizza	126	soup
22	liquid:washing	57	chicken	92	button	127	oatmeal
23	box	58	cutlery	93	towel:kitchen	128	onion:spring
24	hob	59	coffee	94	vegetable	129	clip
25	dough	60	glove	95	stock	130	lighter
26	package	61	can	96	grater	131	ginger
27	water	62	leaf	97	ladle	132	tea
28	meat	63	sink	98	yoghurt	133	nut
29	pot	64	milk	99	cereal	134	vinegar
30	potato	65	heat	100	wrap:plastic	135	holder
31	oil	66	jug	101	broccoli	136	pin:rolling
32	cheese	67	aubergine	102	sugar	137	pie
33	bread	68	salad	103	brush	138	powder
34	food	69	chilli	104	biscuit	139	burger

ID	Objekte	ID	Objekte	ID	Objekte	ID	Objekte
140	book	180	cellar:salt	220	slicer	260	stick:crab
141	shell:egg	181	hummus	221	control:remote	261	ring:onion
142	tongs	182	chair	222	label	262	pestle
143	cream	183	juice	223	celery	263	window
144	pork	184	pancake	224	cabbage	264	gin
145	oregano	185	bean:green	225	hoover	265	bar
146	banana	186	toaster	226	breadstick	266	mint
147	processor:food	187	apple	227	roll	267	heater
148	paste	188	chocolate	228	cocktail	268	grass:lemon
149	recipe	189	ice	229	crisp	269	rubber
150	liquid	190	knob	230	ladder	270	gherkin
151	choi:pak	191	handle	231	beer	271	breadcrumb
152	cooker:slow	192	wine	232	pan:dust	272	watch
153	plug	193	pea	233	battery	273	melon
154	utensil	194	pith	234	powder:washing	274	cinnamon
155	noodle	195	yeast	235	backpack	275	popcorn
156	salami	196	coconut	236	cumin	276	dumpling
157	kitchen	197	fishcakes	237	cutter:pizza	277	rosemary
158	teapot	198	spinach	238	air	278	power
159	floor	199	apron	239	pear	279	syrup
160	tuna	200	raisin	240	quorn	280	candle
161	lime	201	basil	241	funnel	281	pineapple
162	omelette	202	grape	242	wall	282	sheets
163	bacon	203	kale	243	strawberry	283	soda
164	sandwich	204	wire	244	almond	284	raspberry
165	phone	205	asparagus	245	tv	285	airer
166	thermometer	206	paprika	246	scotch:egg	286	balloon
167	orange	207	mango	247	shelf	287	turkey
168	basket	208	caper	248	straw	288	computer
169	parsley	209	drink	249	stand	289	key
170	spinner:salad	210	stalk	250	machine:sous:vide	290	pillow
171	tablet	211	turmeric	251	masher	291	pen
172	presser	212	whetstone	252	guard:hand	292	face
173	coriander	213	kiwi	253	shrimp	293	plum
174	opener:bottle	214	bean	254	fruit	294	whiskey
175	cake	215	thyme	255	artichoke	295	door:kitchen
176	avocado	216	finger:lady	256	cork	296	tape
177	lentil	217	beef	257	cherry	297	camera
178	blueberry	218	whisk	258	sprout	298	cd
179	fan:extractor	219	blackberry	259	mat:sushi	299	extract:vanilla

A.2.2. Beispiele

Abkürzungen: Übers. = Übersetzung, `G` = `GAP`

Eingabe: „please open the refrigerator“

Übers.	<G>	open	<G>	fridge	<G>
Tags	OK	OK	OK	OK	OK

Wie man sieht, wurde die korrekte Aktion und das korrekte Objekt, welches dabei auf seinen Vertreter abgebildet wurde, erkannt.

Eingabe: „spray degreaser“

Übers.	<G>	liquid:washing	<G>
Tags	BAD	OK	OK

Ein knapper Befehl, wie er im originalen Datensatz häufig vorkommt. Hier wird die fehlende Aktion in der Übersetzung erkannt und dann die passende Rückfrage „What do you want to do with liquid:washing?“ gestellt.

Eingabe: „pour sauce over salad“

Übers.	<G>	pour	<G>	sauce	<G>	salad	<G>
Tags	OK	BAD	OK	OK	OK	OK	OK

Das Adverb der Aktion fehlt in der Übersetzung, woraufhin die Aktion nochmals erfragt wird mit „What do you want to do with sauce and salad?“.

Eingabe: „mix egg with butter and sugar“

Übers.	<G>	mix-with	<G>	egg	<G>	butter	<G>	sugar	<G>	sugar	<G>
Tags	OK	OK	OK	OK	OK	OK	OK	BAD	OK	OK	OK

Ein Objekt kommt in der Übersetzung doppelt vor, woraufhin nur eines der beiden getaggt wird und dann gefragt wird: „You want to mix-with what and egg and butter and sugar?“. Hier müsste man dann mit „Nichts“ antworten können.

Eingabe: „turn on extractor fan“

Übers.	<G>	fan:extractor	<G>
Tags	OK	BAD	OK

Obwohl das Objekt eigentlich nicht falsch ist, wird es hier aufgrund des fehlenden Verbs als falsch markiert. Da die Übersetzung nur ein Wort enthält und dieses auch noch vermeintlich falsch ist, wird unspezifisch auf einen Fehler hingewiesen mit „Sorry, I didn’t get that! Could you rephrase that?“.

A.3. Ergebnisse

A.3.1. T5

Epoche	Durchschnittlicher Fehler	BLEU
1	0.0014021805322344 29	90.52
2	0.000917323041902461	90.80
3	0.000389014740231572	92.27
4	0.000295299949685780	58.25
5	0.000237962215205115	67.95
6	0.000197701537117162	49.13
7	0.000175574346781258	86.62
8	0.000100977589862135	94.29
9	0.000069971487919167	94.92
10	0.000060497353059744	94.39
11	0.000057441340696067	95.08
12	0.000051308514761061	94.94
13	0.000048683940089730	94.82
14	0.000043816920769313	95.02
15	0.000046696227093488	94.68
16	0.000040761255375411	94.72

Tabelle A.1.: Verlauf des Trainings von T5

A.3.2. TransQuest

Schritt	Genauigkeit	Sensitivität	F1-Wert	Trainingsf.	Validierungsf.
300	0.000000	0.000000	0.000000	0.096945	0.293849
600	0.000000	0.000000	0.000000	1.265043	0.344664
900	1.000000	0.001864	0.003721	0.000311	0.251082
1200	0.834259	0.239947	0.372699	1.617638	0.229007
1500	0.761426	0.665513	0.710246	0.000059	0.178809
1800	0.862210	0.673236	0.756094	0.199313	0.140458
2100	0.915323	0.785885	0.845680	0.000107	0.104749
2400	0.928833	0.809854	0.865272	0.000270	0.129526
2700	0.882855	0.826897	0.853960	0.000096	0.127625
3000	0.975204	0.733156	0.837033	0.349799	0.118225
3300	0.971188	0.646338	0.776143	0.000710	0.141600
3600	0.901325	0.887883	0.894553	0.000028	0.099257
3900	0.930341	0.850067	0.888394	0.000031	0.104694
4200	0.941469	0.856724	0.897100	1.971122	0.096700
4500	0.926277	0.859920	0.891866	0.000969	0.090628
4800	0.855764	0.850067	0.852906	0.891123	0.102885
5100	0.957238	0.727297	0.826574	0.000023	0.169221
5400	0.891994	0.884154	0.888057	0.000113	0.097247
5700	0.836817	0.831691	0.834246	0.044242	0.101287
6000	0.922916	0.822636	0.869896	0.000862	0.079826
6300	0.945929	0.801332	0.867647	0.000154	0.113984
6600	0.915772	0.862850	0.888523	0.001725	0.081569
6900	0.927804	0.828229	0.875193	0.000110	0.107662
7200	0.926572	0.819973	0.870020	0.000109	0.107913
7500	0.981203	0.695073	0.813718	0.088691	0.153016
7800	0.866576	0.845806	0.856065	0.404168	0.098235
8100	0.910166	0.860719	0.884752	0.000746	0.074152
8400	0.962483	0.573901	0.719052	0.005581	0.170898
8700	0.914795	0.792011	0.848987	0.040727	0.119108
9000	0.656713	0.837550	0.736189	0.002020	0.227991
9300	0.970707	0.829561	0.894601	0.000165	0.083693
9600	0.913807	0.900666	0.907189	0.000245	0.083474
9900	0.867254	0.812517	0.838994	0.000385	0.164731
10200	0.803124	0.903862	0.850520	0.000670	0.112002
10500	0.947299	0.761119	0.844064	0.002077	0.115971
10800	0.965599	0.754993	0.847407	0.002673	0.093681
11100	0.988281	0.741145	0.847055	0.000625	0.097027
11400	0.917834	0.785353	0.846441	0.001209	0.113346

Tabelle A.2.: Verlauf des Trainings von TransQuest