Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bericht zum Proseminar
„Softwareentwicklung in der Wissenschaft“

Natural Language Processing

vorgelegt von

Anton Caesar

Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen

Studiengang: Informatik
Matrikelnummer: 7403376
Betreuer: Tobias Finn, Jakob Lüttgau

Hamburg, 31.08.2021

Inhaltsverzeichnis

1 Einleitung

Ziel der folgenden Arbeit ist es, zunächst ein Grundverständnis über das Thema Natural Language Processing zu vermitteln, um dann dessen Zusammenhang mit Machine Learning herzustellen. Dieser wird im Anschluss anhand eines ausgewählten Modells, des Transformers, sowie eines Praxisbeispiel verdeutlicht. Dabei sollen vor allem der technische Ablauf innerhalb des Modells und die heutigen Anwendungsmöglichkeiten dessen veranschaulicht werden.

Natural Language Processing ist ein Teilgebiet der Informatik, bei dem es um die strukturelle sowie inhaltliche Verarbeitung von menschlicher Sprache geht, welche als „natürlich" bezeichnet wird. Technisches Ziel ist dabei meist, Computersystemen die Fähigkeit zu geben, natürliche Sprache sowohl aufnehmen und verstehen als auch selbst in ihr kommunizieren zu können. In der direkten Anwendung wird dies z.B. zur Interaktion zwischen Mensch und Maschine sowie zur Analyse von Daten in natürlicher Sprache verwendet. Aufgrund der Eigenschaften und der Komplexität natürlicher Sprache überschneidet sich das Gebiet Natural Language Processing auch mit den Fachbereichen der Linguistik und, vor allem heutzutage, der Künstlichen Intelligenz sowie des Machine Learning. Die folgende Arbeit befasst sich dabei hauptsächlich mit letzterem, wobei der Begriff Natural Language Processing im Folgenden mit NLP abgekürzt wird.

1.1 Anwendung von NLP

Im Allgemeinen lassen sich die Anwendungsfälle von NLP meist in zwei Gruppen einteilen: Interaktion bzw. Kommunikation und Analyse. Erstgenanntes ist dabei bereits oft im heutigen Alltag zu finden. So nutzen Voice Assistants wie beispielsweise Siri, Alexa oder der Google Assistant NLP, um die in natürlicher Sprache erteilten Anweisungen zu verstehen und umzusetzen. Dabei geht es in erster Linie um die Vereinfachung bzw. Automatisierung bestimmter Aufgaben für den Nutzer, um diese für ihn zu erleichtern und ihm Arbeit abzunehmen. Auch an anderer Stelle findet sich NLP im heutigen Alltag, so z.B. in der Autokorrektur und -vervollständigung sowie in Übersetzer-Tools wie Google Translate, welche ebenfalls zur Unterstützung des Nutzers dienen.

Obwohl es auch bei diesen Beispielen teilweise um die Analyse von natürlicher Sprache geht, so dient diese nur zur Reaktion auf den Input des Nutzers. In Anwendungsbereichen, bei denen die Analyse von Sprache als Hauptziel gesetzt ist, eröffnen sich jedoch ganz andere Möglichkeiten. Hier können auf Basis von Machine Learning mittlerweile ganze Texte verarbeitet werden, um diese inhaltlich sowie strukturell zu verstehen, zu filtern und zu gruppieren. Auch solche Systeme werden im heutigen Alltag bereits an vielen Stellen eingesetzt, sind jedoch meist deutlich weniger präsent. Konkrete Anwendungsbeispiele wie Spamfilter oder Suchmaschinen nutzen so z.B. NLP, um ihre Ergebnisse zu verbessern, ohne direkt mit dem Nutzer zu interagieren. Auch die Automatisierung von Textzusammenfassung und Problemerkennung fällt in diesen Bereich.

Des Weiteren wird NLP in sozialen Medien und auch generell im Internet dazu genutzt, um ausgehend von den Eingaben und dem Verhalten des Nutzers verbesserte Werbeanzeigen und zielgruppenspezifisches Marketing zu ermöglichen [6]. Dabei kann in der Theorie jede textliche

Eingabe des Nutzers analysiert werden, um so Informationen über dessen Eigenschaften und Interessen zu sammeln.

1.2 Teilaufgaben von NLP

Obwohl Anwendungen, die NLP nutzen, oft nach außen hin wie ein geschlossenes System wirken, so ist es doch sinnvoll, einen kurzen Blick auf die mögliche Einteilung des Fachgebiets in spezifischere Teilaufgaben zu betrachten. Ebenso wie natürliche Sprache viele verschiedene Aspekte umfasst, kann die Verarbeitung dieser in differenzierten Schritten erfolgen, die oft auch im Einzelnen für Anwendungen genutzt werden können. Einige davon werden im Folgenden genannt [6]:

Part-of-Speech Tagging (POS Tagging)

Hierbei geht es um das Herauserkennen sowie die Zuordnung von Wörtern innerhalb eines Textes. Sowohl die Beachtung des Kontexts als auch die Bedeutung der Worte selbst können dabei eine Rolle spielen und sich auf die Korrektheit der Analyse auswirken.

Word sense disambiguation

Hier wird versucht, mit der Mehrdeutigkeit von Wörtern umzugehen und diese richtig zu interpretieren. Dabei ist ebenfalls der Kontext des Textes oder Satzes wichtig.

Named entity recognition

Diese Teilaufgabe befasst sich mit der Erkennung von Eigennamen und -begriffen, die oft nicht aus der Grammatik abgeleitet werden können, sowie mit deren Gruppierung.

Sentiment analysis

Die Sentiment analysis versucht, subjektive Aspekte des Textes zu erfassen, so z.B. vermittelte Emotionen und Standpunkte, aber auch rhetorische Mittel wie Sarkasmus.

1.3 Geschichte von NLP

Bevor in Punkt 2 der Zusammenhang mit Machine Learning verdeutlicht wird, liegt es nahe, noch einmal kurz die geschichtliche Entwicklung von NLP zu betrachten. Seine Anfänge findet der Fachbereich in den 1950er Jahren, als erstmals die Durchführung von Übersetzungen mithilfe des Computers versucht wurde. Im folgenden Jahrzehnt wurde mithilfe von statischen Analyseverfahren an der Simulation von intelligenten Kommunikationspartnern versucht. Eines der bekannteren Beispiele ist ELIZA, ein Computerprogramm aus der Mitte der 60er Jahre, welches einen Psychotherapeuten simulierte und mit dem Nutzer in natürlicher Sprache interagieren konnte. Die scheinbare Intelligenz des Programms und seiner Antworten hatte dabei jedoch noch nichts mit inhaltlicher Analyse zu tun. So ließ der Computer den Input des Nutzers lediglich durch eine Reihe von Filtern laufen, die bestimmte Textbausteine herauserkannten und auf Basis und eines vorprogrammierten Katalogs eine möglichst treffende Antwort generierten. Eine solche, statische Textverarbeitung blieb auch in den nächsten Jahrzehnten der hauptsächliche Ansatzpunkt.

Erst im Übergang zu den 90er Jahren ermöglichten der technische Fortschritt und die Entwicklung neuer Konzepte eine bedeutende Weiterentwicklung des Fachbereichs und es wurde erstmals Machine Learning genutzt [10]. Im Laufe der wachsenden Popularität des Internets in den 2000ern und des schnellen Anstiegs der Rechenleistung ergaben sich in den folgenden Jahren immer bessere Möglichkeiten des Trainings und der Datenbeschaffung. Viele der heute genutzten Methoden für NLP sind daher erst wenige Jahre alt.

2 NLP und Machine Learning

Bevor in diesem Teil der Fokus auf den Zusammenhang von NLP und Machine Learning gelegt wird, müssen noch einmal die Grundlagen von letzterem erklärt werden, um Verständnisprobleme auszuschließen. Beim Machine Learning werden in der heutigen Informatik meist Neuronale Netze genutzt, die ein zielgerichtetes Lernen aus Daten ermöglichen. Ein solches Netz besteht aus künstlichen Neuronen, welche über mehrere Layer verteilt und miteinander verknüpft sind. Dabei gibt es ein Input-Layer und ein Output-Layer, die dazwischenliegenden werden als Hidden-Layer bezeichnet. Durch die Verknüpfungen zwischen den Neuronen werden Werte im Netz durch die Layer verarbeitet, wobei jedes Neuron bestimmte Eigenschaften hat, welche die empfangenen Werte vor der Weitergabe beeinflussen können [9].

Ein Neuronales Netz wird mit passenden Trainingsdaten für eine oder mehrere Aufgaben trainiert. Hierbei wird durch eine Kostenfunktion, die die erzeugten Resultate mit den gewünschten vergleicht, die Güte der Vorhersagen bewertet. Basierend darauf werden dann die Gewichte innerhalb des Netzes angepasst. Üblicherweise wird dafür ein Gradient Descent Verfahren mit Backpropagation verwendet. Das bedeutet, dass der zur Anpassung verwendete Algorithmus rückwärts durch die Layer des Netzes läuft und dabei die Gewichte in jeder Schicht so verändert, dass die ermittelte Fehlergröße der nachfolgenden Neuronen verringert wird [1].

Interessant für das Verständnis der folgenden Abschnitte ist dabei auch, welche neuronalen Verknüpfungen innerhalb eines Netzes erlaubt sind. Zwei verschieden definierte Typen sind vereinfacht in Abbildung 1 dargestellt. Häufig zu finden ist das sogenannte Feedforward-Netz (links), in dem Neuronen nur ausgehende Kanten, also Verknüpfungen, zu jenen Neuronen des nächsten Layers haben dürfen. Damit wird verhindert, dass sich innerhalb des Netzes Schleifen bilden. Im Kontrast dazu steht das Recurrent Neural Network (rechts), in dem Kantenverbindungen auch auf die Neuronen selbst und zu vorherigen, „höheren“ Layern laufen dürfen. Durch so entstehende Schleifen ist es möglich, Werte und damit Informationen auch über längere Zeit im Netz zu halten [9].

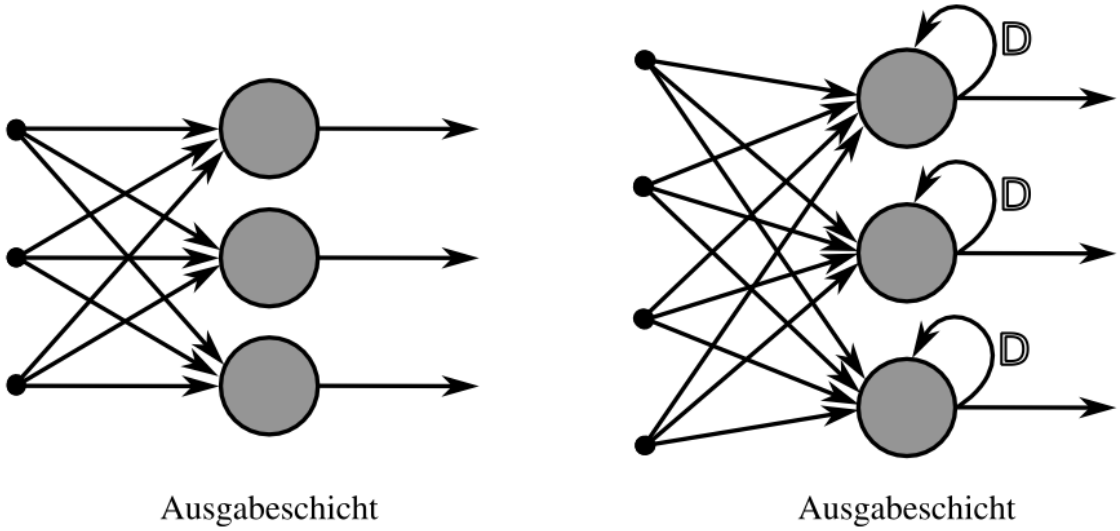

Abbildung 1: Darstellung eines Feedforward-Netzes (links) und eines Recurrent Neural Networks (rechts), Quellen: https://upload.wikimedia.org/wikipedia/commons/b/b4/SingleLayerNeuralNetwork_deutsch.png, https://upload.wikimedia.org/wikipedia/commons/9/9e/RecurrentLayerNeuralNetwork_deutsch.png

2.1 Preprocessing

Im Zusammenhang mit Natural Language Processing stellt sich die Frage, wie ein Neuronales Netz einen vorgegebenen Text als Eingabe nutzen kann. Die einzelnen Buchstaben oder Worte direkt in das Netz zu speisen ist dabei technisch schwierig. Deshalb wird ein Vorgang namens Preprocessing genutzt, dessen Aufgabe es ist, die ins Netz zu gebenden Daten so vorzubereiten, dass ihre Form als Input dienen kann.

Die Durchführung eines solchen Preprocessing unterscheidet sich je nach Anwendung, in einigen Fällen werden die Eingaben direkt verarbeitet, in anderen können vorher noch weitere Verfahren genutzt werden, um den Input zu vereinfachen. Zwei solcher Methoden sind Stemming und Lemmatization. Beim Erstgenannten werden die Worte der Eingabe auf ihre grammatikalische Grundform bzw. den Wortstamm zurückgeführt, beim Letzteren werden ähnliche oder inhaltlich gleiche Begriffe vereinheitlicht [4]. Beide Verfahren dienen dazu, die grammatikalische Komplexität des Inputs zu reduzieren, während der Inhalt weitestgehend gleichbleibt.

Unabhängig von der Anwendung des Stemming muss der gegebene Text jedoch in eine für das Neuronale Netz geeignete Form überführt werden, bevor er diesem übergeben wird. Dazu nutzt man Prozesse, die als Tokenization und Vectorization bezeichnet werden. Bei der Tokenization wird der Text zunächst in seine einzelnen Bestandteile aufgeteilt, in den meisten Fällen die einzelnen Worte. Bei Sprachen wie Deutsch oder Englisch ist der Aufwand in diesem Schritt verhältnismäßig gering, bei Sprachen wie Chinesisch, in denen es keine Lückensetzung zwischen den Begriffen gibt, ist der Vorgang jedoch komplizierter.

Für die weitere Vorbereitung werden die einzelnen Textbestandteile, welche als Tokens bezeichnet werden, mithilfe der Vectorization in Vektoren umgewandelt. Dabei wird jedem einzelnen Token ein individueller Vektor zugeordnet, der es im System repräsentiert. Welcher Wert dies ist, wird dabei vom verwendeten Algorithmus bestimmt, der sich zwischen

verschiedenen Anwendungen unterscheiden kann. In den meisten Fällen werden bei der Überführung inhaltlich ähnlichen und zusammenhängenden Begriffen auch ähnliche Vektorrepräsentationen zugeordnet. Nach Durchführung des Preprocessing liegt der gegebene Text dem Neuronalen Netz in Form von Vektoren, vereinfacht also Zahlen, vor, und kann in dieses eingegeben werden.

2.2 Erforderliche Komplexität

Eine weitere wichtige Frage bei dem Entwurf von NLP-Systemen, die Machine Learning nutzen, ist die Frage nach deren erforderlicher Komplexität. Soll das System nur einfache Sätze übersetzen, wird eventuell nicht einmal ein Neuronales Netz benötigt, da der Input Wort für Wort verarbeitet und übersetzt werden kann. Die erforderliche Komplexität steigt jedoch rasch an, wenn z.B. für die Analyse Inhalt und Sinn des Textes erfasst werden sollen. Das folgende Beispiel zeigt dabei, wieso das System hierbei über eine Art „Gedächtnis“ verfügen muss.

Abbildung 2: Beispiel eines komplexeren Textabschnitts, für dessen Verständnis die Maschine ein Gedächtnis braucht, Quelle: Eigene Abbildung

Wie in Abbildung 2 zu erkennen ist, verweisen mehrere Worte des zweiten Satzes auf Begriffe, die nur im ersten Satz direkt genannt werden. Um den zweiten Satz trotzdem verstehen zu können, muss sich die Maschine sowohl an den Inhalt des ersten Satzes erinnern als auch die Begriffe korrekt zuordnen können.

Im Machine Learning gibt es verschiedene Konzepte, die ein solches, komplexes Textverständnis ermöglichen. Eines der in den letzten Jahren erfolgreichsten Modelle ist dabei der Transformer, der im nächsten Teil vorgestellt wird.

3 Der Transformer

In der bisherigen Erklärung des Gebrauchs von Machine Learning im Bereich von NLP ging es um einzelne Neuronale Netze, in denen Informationen eingegeben und verarbeitet werden können. Um die eben erwähnte, komplexere Textverarbeitung zu ermöglichen, verfügen heutige Systeme jedoch über einen deutlich komplexeren Aufbau, der meist mehrere Netze mit weiteren Komponenten dazwischen beinhaltet.

Dies ist auch beim Transformer-Modell der Fall, dessen Aufbau gleich veranschaulicht wird. Stark vereinfacht ist ein Transformer ein Anwendungskonzept des Machine Learning, welches

eine Reihe eingegebener Zeichen in eine Ausgabe übersetzt [11]. Im Kontext des Arbeitsthemas beschränken sich die folgenden Ausführungen auf den Bereich der Text- und damit Sprachverarbeitung, theoretisch ist das Modell jedoch nicht nur darauf beschränkt. Je nach Training können Transformer für Übersetzung, Analyse, Zusammenfassung und Generierung von Texten genutzt werden und damit für die meisten Anwendungsfälle natürlicher Sprache. Des Weiteren sind auch über die Textverarbeitung hinausreichende Kompetenzen möglich, ein konkretes Beispiel dazu wird in Teil 4 gegeben.

3.1 Aufbau

Zunächst wird der typische Aufbau eines Transformers erklärt. Wie aus Abbildung 3 hervorgeht, besteht ein klassischer Transformer aus 6 Encodern und 6 Decodern, die jeweils als Stack bezeichnet werden. Diese sind dabei hintereinandergeschaltet und arbeiten jeweils mit dem Output des vorherigen Kodierers, wobei alle Decoder als zusätzlichen Input den Output des letzten Encoders erhalten [2]. Ebenfalls zu sehen ist, dass der erste Encoder den originalen Input entgegennimmt, während der letzte Decoder das Endergebnis liefert.

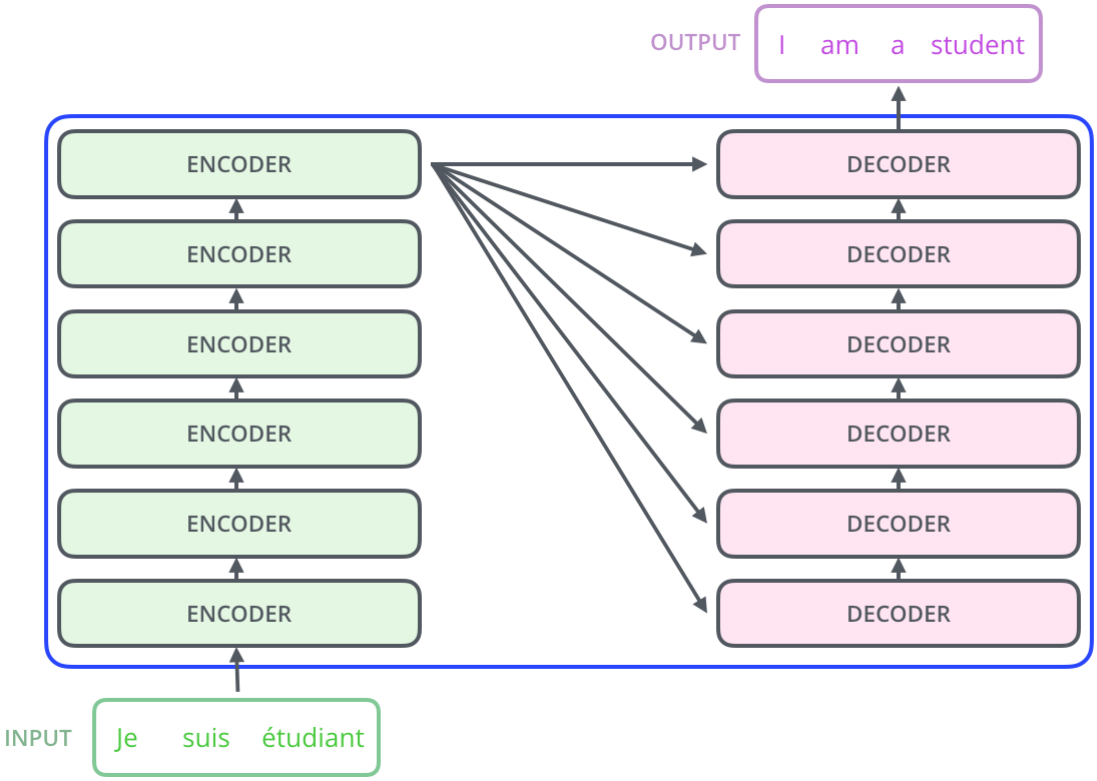

Abbildung 3: Übergeordneter Aufbau eines Transformers
Quelle: https://jalammar.github.io/images/t/The_transformer_encoder_decoder_stack.png

Verallgemeinert sieht der Ablauf innerhalb eines Transformers folgendermaßen aus: Ein Input, also in diesem Beispiel ein Satz oder Text, wird dem Transformer zur Eingabe vorgelegt. Dieser wurde dabei bereits durch Preprocessing in Vektoren überführt. Typisch für die Dimension solcher Vektoren ist die Zahl 512, d.h. dass jeder Vektor 512 Werte enthält.

3.2 Verarbeitung und Attention

Der vorbereitete Input wird in den Transformer gegeben. Dabei läuft er der Reihe nach durch alle 12 Kodierer, wobei er innerhalb eines jeden Kodierers verarbeitet und verändert wird. Alle Tokens des Inputs werden dabei parallel verarbeitet.

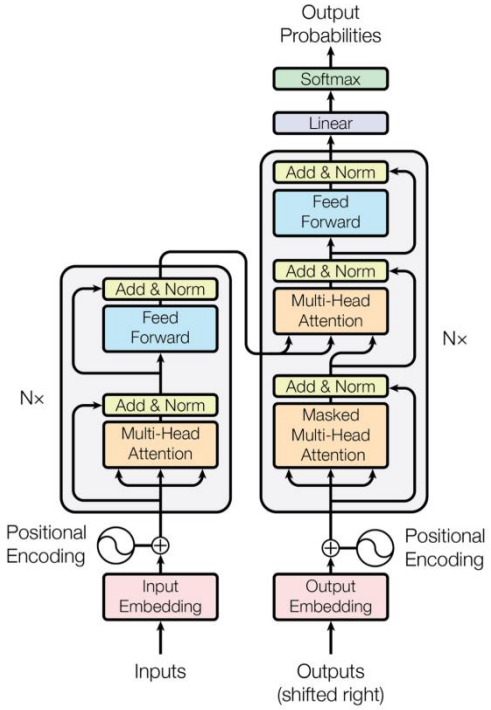

Abbildung 4: Aufbau von Encoder und Decoder, Quelle: https://arxiv.org/pdf/1706.03762.pdf

Abbildung 4 stellt den inneren Aufbau der Kodierer dar, wobei die Unterschiede zwischen Encoder und Decoder deutlich werden. Innerhalb eines Encoders läuft der Input zunächst durch ein sogenanntes Attention-Modul. In der Abbildung ist dieser Teil als „Multi-Head Attention" betitelt, womit die parallele Verarbeitung mehrerer Inputs gemeint ist. Präzise handelt es sich beim im Transformer verwendeten Attention-Modul um ein Self-Attention-Modul. Das dort angewendete Verfahren dient dazu, die Relevanz aller einzelnen Bestandteile des Inputs zu erkennen und deren Gewichtung entsprechend zu erhöhen oder zu senken. Ähnlich der menschlichen Aufmerksamkeit sollen auf diese Weise wichtige Informationen innerhalb des Inputs für die Verarbeitung hervorgehoben werden [5]. Das Attention-Modul ist ein zentraler Bestandteil des Transformers, jedoch auch in anderen Modellen zu finden. Im Folgenden wird erklärt, wie das Attention-Modul funktioniert.

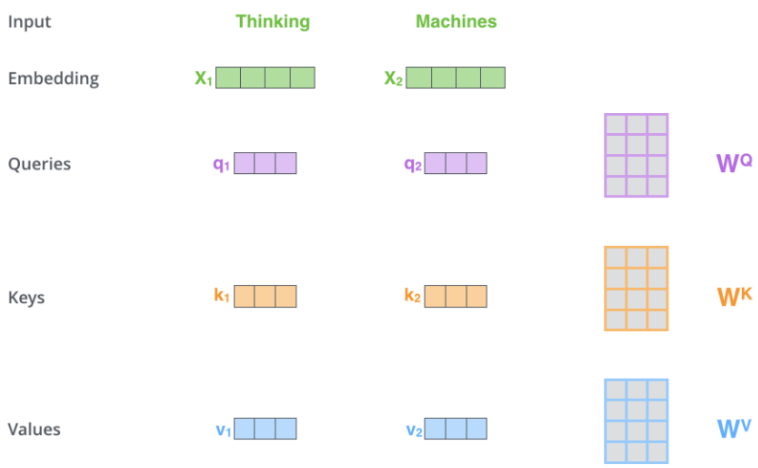

Abbildung 5: Erzeugung der Vektoren im Attention-Modul
Quelle: https://jalammar.github.io/images/t/transformer_self_attention_vectors.png

Wie in Abbildung 5 am Input-Beispiel „Thinking Machines“ dargestellt, werden zunächst für jedes Token im Input drei neue Vektoren errechnet. Diese bezeichnet man als Query (Abfragevektor), Key (Schlüsselvektor) und Value (Wertevektor). Dazu wird das Token jeweils mit einer Matrix multipliziert, deren Werte in der Trainingsphase erlernt wurden, um einen neuen Vektor zu erzeugen.

Die entstehenden Vektoren unterscheiden sich in ihrer Dimension vom Token-Vektor und weisen nur 64 Werte auf. Anhand dieser Vektoren wird eine Gewichtung für jedes einzelne Token errechnet, was in Abbildung 6 zu sehen ist. Dazu wird zunächst pro Token ein Score errechnet, der sich aus dem Skalarprodukt von Query und Key des Tokens ergibt. Dieser Vorgang wird für jedes Token mit den Keys aller anderen Tokens wiederholt, sodass für jedes Token n Scores errechnet werden, wobei n für die Gesamtzahl der Input-Tokens steht.

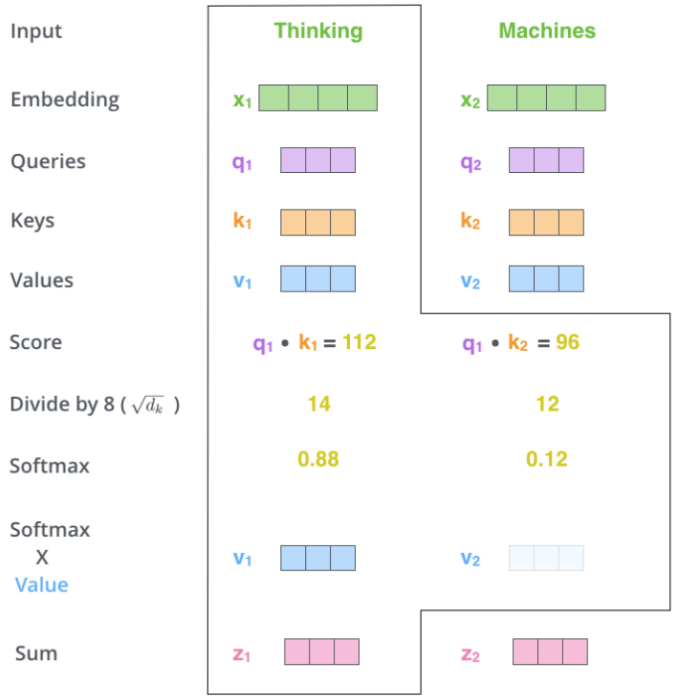

Abbildung 6: Berechnung der Gewichtungen für jedes Token
Quelle: https://jalammar.github.io/images/t/self-attention-output.png

Jeder dieser n Scores wird durch die Wurzel der Dimension des Key-Vektors, also $\sqrt{64} = 8$, geteilt, und mit einer Softmax-Funktion normalisiert, sodass die Summe aller Scores 1 ergibt. Die normalisierten Scores werden daraufhin separat mit dem Value-Vektor multipliziert. Im letzten Schritt werden alle auf diese Weise entstandenen Vektoren addiert und bilden die Ausgabewerte des Attention-Moduls für jedes Token [5].

Nach Durchlauf durch das Attention-Modul werden die Ausgabewerte mit denen addiert, die vor Anwendung des Attention-Mechanismus vorlagen, und im Anschluss normalisiert. Wie Abbildung 4 zeigt, laufen die Vektoren durch ein Feedforward-Netz, welches ebenfalls in der Trainingsphase trainiert wurde, bevor sie erneut mit den vorherigen Werten addiert, normalisiert und an den nächsten Kodierer weitergegeben werden.

Ebenfalls in Abbildung 4 zu sehen ist, dass die Decoder ähnlich wie die Encoder aufgebaut sind, als zentralen Unterschied jedoch über ein zusätzliches Attention-Modul in der Mitte verfügen. Hier wird ein weiteres Mal ein Attention-Mechanismus benutzt, wobei für die Berechnung von Key- und Value-Vektor der Output des letzten Encoders verwendet wird und lediglich der Query-Vektor aus dem des vorherigen Attention-Moduls.

Ansonsten unterscheidet sich der Ablauf innerhalb des Decoders nur geringfügig im ersten Attention-Modul. Dort werden bei der Berechnung der Scores nicht alle n Scores für jedes Token errechnet, sondern jene weggelassen, die mit den im Input an späterer Stelle stehenden Tokens entstehen würden. Dies wird als Masking bezeichnet und verhindert, dass Informationen, die erst nachfolgend im verarbeiteten Inhalt auftauchen, bereits Auswirkung auf die Entstehung vorheriger Output-Tokens haben [7]. Jedes Token wird somit nur durch Informationen der davor positionierten Tokens gebildet.

3.3 Resultate

Letztendlich liefert der Transformer einen finalen Output, der je nach Training aus dem Input errechnet wurde. Genau genommen liegt dieser Output nicht direkt in Textform vor, sondern besteht aus den ermittelten Auftrittswahrscheinlichkeiten aller Begriffe im vom Transformer umfassten Wortschatz. Um daraus ein konkretes Ergebnis zu erzeugen, werden für den Output an jeder Stelle die Worte ausgewählt, deren Wahrscheinlichkeit in der Vorhersage am höchsten ist.

Interessant ist, wie generell bei Neuronalen Netzwerken üblich, dass nicht direkt ersichtlich ist, wie der Transformer vom Input zum Output kommt. Während der Trainingsphase lernt das Modell eigenständig anhand der verwendeten Daten, wie die Aufmerksamkeit für den Input gesetzt werden muss und nach welcher Systematik möglichst treffende Antworten erzeugt werden können. Dabei werden die Parameter innerhalb des Modells basierend auf Kostenfunktion und Backpropagation angepasst, wie bereits in Abschnitt 2 erklärt.

Vergleicht man die Resultate, die ein erfolgreich trainierter Transformer liefert, mit denen von alternativen bzw. vorherigen Modellen wie CNNs und RNNs, zeigt sich ein eindeutiger Vorsprung in Leistung und Qualität. Bereits mit weniger Trainingsaufwand ist das Modell in der Lage, bessere Ergebnisse in den Bereichen der Übersetzung und Textgenerierung zu erzielen [2]. Im folgenden Abschnitt wird die Anwendung der Transformer-Methodik anhand eines konkreten Beispiels aufgezeigt.

4 GPT-3

GPT-3 ist ein Sprachverarbeitungsmodell, welches auf der Transformer-Mechanik basiert. Es wurde von OpenAI entwickelt und trainiert und ist die Abkürzung für Generative Pre-trained Transformer 3, also die dritte Generation des Modells. GPT-3 kann für viele Aufgaben im Bereich der Textverarbeitung genutzt werden, darunter Übersetzung, Zusammenfassung und Generierung, und ist das größte zur Zeit existierende Sprachmodell weltweit [8].

Während in vielen Anwendungsfällen außerhalb von NLP bereits Modelle und Netze mit einigen tausend, selten einigen hunderttausend Parametern ausreichen, werden im Bereich der Sprach Verarbeitung mittlerweile ganz andere Dimensionen erreicht. GPT-3 verfügt in seiner vollen Kapazität über etwa 175 Millarden Parameter und hat damit etwa 10 Mal mehr Parameter als Platz 2 in der Liste der Sprachmodelle. GPT-3 wurde mit mehreren 100 Milliarden Wörtern trainiert und ist in der Lage, vollständig autonom ganze Texte zu erfassen, die in den meisten Fällen strukturell Sinn ergeben und sogar inhaltlich korrekt sind. Auch kann das Modell in diversen Sprachen programmieren und gezielt Texte wie Drehbücher, Lieder und Dialoge verfassen [8].

Üblicherweise reicht die Kapazität eines Sprachmodells nicht, um z.B. alle eben genannten Aufgaben mit einem umfassenden Training durchführen zu können, sodass es für spezifische Aufgaben neu trainiert werden muss. Ein weiteres Merkmal von GPT-3 ist jedoch, dass es im Gegensatz zu anderen Modellen kaum solche Anpassungen, genannt Fine-Tuning, für spezifische Aufgaben benötigt.

Auch reichen die Kompetenzen des GPT-Modells teilweise weiter als nur für die Textverarbeitung. Ohne das dies beabsichtigt war, hatte sich bereits beim Vorgänger-Modell GPT-2 gezeigt, dass es die Fähigkeit zum Schachspielen besaß. Dazu musste dem Modell eine Liste mit bisherigen Zügen übergeben werden, aus dem dann der nächste Zug berechnet wurde. Teils konnte das Modell dabei sogar spezifische Taktiken anwenden, Züge scheinbar vorausplanen und falsche Züge in der übergebenen Liste erkennen [3].

5 Schlusswort

Ziel dieser Arbeit war es, einen allgemeinen Überblick zu Natural Language Processing und dessen Zusammenhang mit Machine Learning zu vermitteln, und dabei vor allem über die grundlegenden technischen Abläufe sowie heutige Möglichkeiten und Anwendungen zu informieren. Bereits in der heutigen Zeit spielt Natural Language Processing eine wichtigere Rolle, als es zunächst scheinen mag. Wie aus den genannten Beispielen hervorgeht, eröffnen sich weitreichende Möglichkeiten in der Kommunikation und Analyse der natürlichen Sprache, meist ohne die genaue Funktionsweise der Umsetzung zu kennen.

Vor allem in der heutigen, digitalen Gesellschaft entstehen damit sowohl Chancen als auch Gefahren, die in den nächsten Jahren sicher noch an Bedeutung gewinnen werden. Dabei zeigt schon jetzt das Beispiel GPT-3 eindrucksvoll, welches Potenzial der Bereich des Natural Language Processing bietet.

Literaturverzeichnis

[1] 3Blue1Brown, What is backpropagation really doing? | Chapter 3, Deep learning. 03.11.2017. https://www.youtube.com/watch?v=Ilg3gGewQ5U

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. 12.06.2017. https://arxiv.org/abs/1706.03762

[3] David Noever, Matthew Ciolino, Josh Kalin. The Chess Transformer. 02.08.2020. https://arxiv.org/ftp/arxiv/papers/2008/2008.04057.pdf

[4] edureka!, Natural Language Processing In 10 Minutes. 16.10.2018. https://www.youtube.com/watch?v=5ctbvkAMQO4&t=429s

[5] Giuliano Giacaglia, How Transformers Work. 11.03.2019. https://towardsdatascience.com/transformers-141e32e69591

[6] IBM, Natural Language Processing (NLP). 02.06.2020. https://www.ibm.com/cloud/learn/natural-language-processing

[7] Samuel Kierszbaum. Masking in Transformers´ self-attention mechanism. 27.01.2020. https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c

[8] Wikipedia, GPT-3. Abgerufen am 24.06.2021. https://en.wikipedia.org/wiki/GPT-3

[9] Wikipedia, Künstliches neuronales Netz. Abgerufen am 24.06.2021. https://de.wikipedia.org/wiki/Künstliches_neuronales_Netz

[10] Wikipedia, Natural language processing. Abgerufen am 31.08.2021. https://en.wikipedia.org/wiki/Natural_language_processing

[11] Wikipedia, Transformer. Abgerufen am 24.06.2020. https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)