

**Bachelorarbeit zur Erlangung des Grades  
Bachelor of Science  
im Studiengang Wirtschaftsinformatik**

# **Funktionsweise, Anwendungen und Gefahren von Texterzeugungsmodellen am Beispiel von GPT**

Jonas Bevernis

Matrikel 16859

Erstgutachter: Prof. Dr. Thomas Wengerek

Zweitgutachter:

28.07.2021

Hochschule Stralsund

Fakultät für Wirtschaft

## Inhaltsverzeichnis

1. Einleitung.....	3
2. Funktionsweise.....	4
2.1. Theoretische Erklärung.....	4
2.2. OpenAI GPT .....	5
2.2.2 Funktionsweise von GPT.....	8
2.2.3. Programmbeispiele mit GPT.....	16
2.2.4 Auswertung der Programmbeispielen .....	30
Abbildungsverzeichnis.....	33
Literaturverzeichnis.....	35

## 1. Einleitung

### 1. Einleitung

- texterzugungssystem: im englischen natural language processing. Aber nicht ganz das gleiche. Unterschiede rausarbeiten.

## 2. Funktionsweise

### 2.1. Theoretische Erklärung

- [https://www.youtube.com/watch?v=Y\\_NvR5dlaOY&ab\\_channel=AnswerinProgress](https://www.youtube.com/watch?v=Y_NvR5dlaOY&ab_channel=AnswerinProgress) gar nicht so schlecht als kurzer einstig

- supervised und unsupervised learning
- rückgekoppelte neuronale Netze
- Sequence to sequence (oder seq2seq)
- LSTM als rückgekoppeltes neuronales Netz und möglichkeit für Texterzeugungs system erklären. (quasi der Vorgänger von Transformer Modellen)
- attention mechanismus erklären. (vielleicht direkt am LSTM?)
- Reihenfolge von wörtern im satz
- additive attention [2], and dot-product (multiplicative) attention (siehe : <https://arxiv.org/pdf/1706.03762.pdf>, seite 4, Scaled Dot-Product Attention)
- Konvergenz?

### 2.2. OpenAI GPT

OpenAI wurde im Dezember 2015 in San Francisco, von dem Programmierer Sam Altman und Elon Musk sowie weiteren Investoren als Non-Profit gegründet. [1] Elon Musk, der aufgrund seiner Bekanntheit, durch seine anderen Firmen wie SpaceX und Tesla anfangs Blicke auf OpenAI zog, hat den Vorstand allerdings seitdem, aufgrund eines möglichen Interessenkonfliktes, verlassen. [2] Laut eigener Aussage sicherte sich die Firma bei ihrer Gründung etwa eine Milliarde US-Dollar an Investment.

OpenAI beschreibt ihren Auftrag selbst so: „Unsere Mission ist es sicherzustellen, dass künstliche Intelligenz der gesamten Menschheit zugutekommt.“ [3] (eigene Übersetzung). Um dieses Ziel zu erreichen hat OpenAI einige Regeln aufgestellt, an die sich die Firma halten will:

1. Aller Einfluss den OpenAI über die Entwicklung von KI erhält soll dem Vorteil aller dienen, dazu müssen Interessenkonflikte insbesondere der Angestellten und Investoren vermieden werden.
2. Das Ziel von OpenAI, neben der Entwicklung von KI-Systemen, ist insbesondere die langfristige Sicherheit dieser Systeme und das Vorantreiben der Forschung zu sicheren KI-Systemen.
3. Um die Sicherheit von KI-Systemen gewährleisten zu können, ist es wichtig, dass OpenAI immer auf dem neusten Stand der KI-Forschung ist. Es ist ihr Ziel diese Forschung anzuführen.
4. Um ihre Ziele zu erreichen will OpenAI mit anderen kooperieren und den Großteil ihrer Forschung der Öffentlichkeit zur Verfügung stellen. [4]

Neben der GPT-Reihe, um die es später in erster Linie gehen soll, hat OpenAI auch eine Vielzahl anderer verschiedener Modelle entwickelt und Forschungspapiere veröffentlicht. So war ihre erste Veröffentlichung, etwa ein halbes Jahr nach der Gründung, zum Beispiel „OpenAI Gym“, ein Toolkit, das dabei helfen soll, verschiedene „reinforcement learning“ Algorithmen zu vergleichen [5]. „reinforcement learning“ bzw. „Bestärkendes Lernen“ ist eine Methode des maschinellen Lernens bei der das Modell durch Belohnungen positiver oder auch negativer Art lernt, sein Ziel bestmöglich zu erreichen. Soll das Modell zum Beispiel Lernen Pong zu spielen könnte es einen positiven Impuls geben, wenn ein Punkt gemacht wird und einen negativen Impuls, wenn der Gegner einen Punkt macht. Die weitere Forschung zum Thema „reinforcement learning“ zeigte sich im OpenAI Five Projekt, einer der größten Erfolge von OpenAI zu dem Zeitpunkt. Bereits im Jahr 2017 gewinnt ihr „reinforcement learning“ Model das erste Mal ein 1 gegen 1 im Spiel Dota 2 gegen einen der weltweit besten Spieler. Daraus entwickelt sich bis 2019 ein Team aus 5 KIs, dass in diesem Jahr das derzeitige Weltmeisterteam „OG“ in zwei aufeinanderfolgenden Spielen besiegt. [6] Um diesen Erfolg einordnen zu können, ist es wichtig die Komplexität eines Spiels wie Dota 2 zu verstehen. Neben den Wechselwirkungen zwischen Bewegung, einsetzbaren Fähigkeiten sowie kaufbaren Gegenständen mussten die KIs den Teamaspekt des Spiels meistern, um die Stärken der 5 Spielfiguren zu kombinieren. Nur so war ein Sieg möglich. OpenAI Five holte nicht nur OpenAI auch für eine breitere Öffentlichkeit ins Rampenlicht, sondern zeigte auch, wie mächtig „reinforcement learning“ Modelle sein können, wenn es um das Erlernen komplexer Zusammenhänge geht.

Neben OpenAI Five ist die GPT-Reihe wohl eine der öffentlichkeitwirksamsten und für das Thema dieser Arbeit interessantesten KI-Modelle an denen OpenAI arbeitet. GPT steht für „Generative Pre-trained Transformer“, die Reihe umfasst 3 Modelle:

**GPT:** Die erste Version von GPT wurde Juni 2018 veröffentlicht und stellte den anfänglichen Versuch von OpenAI dar, die zu dem Zeitpunkt neuen Transformer-Modelle mit der Methodik des „Semi-supervised training“ zu verbinden (eine genauere Erklärung dieser befindet sich Konzepte findet sich im Kapitel 2.2.2, da diese den Rahmen der Einleitung sprengen würde). Die grundsätzliche Idee ist dabei, das Modell mithilfe eines sehr großen Datensatzes unüberwacht zu trainieren und danach, mittels eines viel kleineren Datensatzes per überwachtem Lernen auf einen bestimmten Aufgabenbereich zu feintunen. So sollen die Vor- und Nachteile der beiden Herangehensweise ausgeglichen werden. Die erste Version von GPT stellte in erster Linie einen Proof of Concept dar, übertraf allerdings laut von OpenAI durchgeführten Tests bereits andere Systeme, die auf dem letzten Stand der Technik waren. Durch den relativen Erfolg von GPT setzte sich das Team einige Ziele:

1. Vergrößerung des Datensatzes. GPT wurde mithilfe von beschränkter Hardware und einem vergleichsweise kleinen Datensatz von etwa 5GB Text trainiert.
2. Verbesserung des Finetunings.
3. Genauere Forschung zum Konzept von „generative pre-training“ [7]

**GPT-2:** Am 14 Februar 2019 veröffentlicht OpenAI ihre weiterführende Forschung sowie die von ihnen mit GPT-2 erzielten Ergebnisse, das Modell selbst wird zu diesem Zeitpunkt allerdings nicht veröffentlicht. Mit Bezug auf ihre 2. Firmenregel wird nur eine deutlich kleinere Version des Modells veröffentlicht. Sie befürchten Missbrauch des Modells bei vollständiger Veröffentlichung dessen und regen Regierungen dazu an, die weitere KI-Entwicklung im Auge zu behalten. [8] Trotz allem entscheidet sich OpenAI Anfang November 2019 das vollständige Modell zu veröffentlichen. [9] Dieses Modell stellt eine direkte Überarbeitung des Vorgängers GPT dar, bei der in erster Linie der Datensatz massiv vergrößert wurde. Für GPT-2 wurde der Datensatz sowie die Menge an Parametern mehr als verzehnfacht. Dabei bietet das Modell, selbst mit wenig Finetuning, eine bessere Leistung als alle anderen Texterzeugungsmodelle mit denen OpenAI sich vergleicht. [8]

**GPT-3:** Version 3 ist die neueste der GPT-Reihe, die Mitte 2020 vorgestellt wurde. Wie schon beim Sprung von GPT auf GPT-2 stellt auch GPT-3 einen direkten Nachfolger zu GPT-2 dar, dessen Verbesserungen in erster Linie durch eine direkte Skalierung des Modells erreicht wurden. Im Vergleich zu GPT-2 (1,5 Milliarden Parameter) stellt das vollständige GPT-3 Modell, mit 175 Milliarden Parametern, eine über hundertfache Vergrößerung der Modellparameter dar. Anders als seine Vorgänger wurde das GPT-3 Modell nicht der Öffentlichkeit zur Verfügung gestellt. Zugriff auf GPT-3 ist nur über die OpenAI API möglich, die Nutzung dieser API ist allerdings aktuell mit einer Warteliste beschränkt und zudem kostenpflichtig. Ob sich dies mit den selbstgestellten Firmenzielen von OpenAI sowie dessen ursprünglicher Gemeinnützigkeit vereinbaren lässt, ist diskutabel. Dabei sei gesagt das hier aus rechtlicher Sicht wohl keine Probleme bestehen, denn neben der gemeinnützigen Organisation „OpenAI“ gibt es auch noch die „OpenAI LP“ die keinen Status als gemeinnützigen Organisation hat und zum Beispiel für den Vertrieb der OpenAI API zuständig ist. Im Vergleich zum Vorgänger sind die Ergebnisse von GPT-3 nochmal deutlich besser, so sind z.B. Fließtexte die von GPT-3 erzeugt teilweise kaum noch von Menschen geschriebenen Texten unterscheidbar. [10] Ein Beispiel dafür ist der Artikel „A robot wrote this entire article.“ vom Guardian [11]. Dabei setzt OpenAI bei GPT-3 in besonders auf das „Few-Shots“ Szenario. Es ist zwar weiterhin auch möglich, das Modell mit überwachtem Lernen zu Feintunen, doch das Modell ist auch ohne zusätzlichem Feintuning in der Lage, mit nur wenigen Beispielen (Few-Shots) eine Aufgabe zu erledigen. In dem zuvor genannten Artikel vom Guardian wurde dem Modell zum Beispiel nur die Aufgabe („Please write a short

## 2. Funktionsweise

op-ed around 500 words. Keep the language simple and concise. Focus on why humans have nothing to fear from AI.“ [11]) sowie ein einziger Absatz als Beispiel gegeben.

Da alle GPT Versionen aufeinander aufbauen und jeweils nach oben skalierte Versionen ihrer Vorgänger sind, ist die grundsätzliche Funktionsweise aller Modelle der Reihe gleich. Diese wird im folgenden Kapitel genauer erläutert. Dabei werden auch die Unterschiede der verschiedenen Versionen genauer veranschaulicht. In Kapitel 2.2.3 werden dann einige Experimente mit GPT durchgeführt, um so nicht nur die Theoretische, sondern auch die praktische Funktionsweise darzustellen. Aufgrund der zuvor genannten Umstände bezüglich des Zugriffs auf GPT-3 bzw. auf die OpenAI API wird für die Experimente exemplarisch GPT-2 genutzt. Die grundsätzliche Arbeitsweise sollte die Gleiche sein wie bei GPT-3 oder auch anderen Modellen abseits der GPT-Reihe. Dabei ist bei der Auswertung der Experimente zu bedenken, dass die Leistung von GPT-2 hinter der von GPT-3 zurückfällt.

## 2. Funktionsweise

### 2.2.2 Funktionsweise von GPT

Die Grundlage von GPT bilden zwei verschiedene Ideen. Auf der einen Seite steht das Transformer-Modell, welches den technischen Grundstein für das eigentliche KI-Modell bietet und auf der anderen Seite, dass „Semi-supervised training“ welches, die Trainingsart von GPT beschreibt. Die Kombination dieser beiden Konzepte bilden den Grundstein für die technische Funktionsweise aller GPT Modelle.

Um eine Abgrenzung des Kapitels zu schaffen, sei Folgendes gesagt: Sowie „Semi-supervised training“ als auch Transformer Modelle bieten allein ausreichend Material für eigenständige Arbeiten über die Themen. Das Ziel ist es dementsprechend nicht alle Aspekte der beiden Themen bis ins Detail zu erläutern, sondern vielmehr einen Überblick über die Funktionsweise zu geben, um so verstehen zu können, wie GPT beide Konzepte kombiniert.

Das Transformer-Modell ist eine neue Variante von Machine Learning Modellen, die im Juni 2017 mit dem Paper „Attention Is All You Need“ im Rahmen der 31. „Neural Information Processing Systems“ Konferenz vorgestellt wurde [12]. Ähnlich wie ein Long-Short-Term-Memory Modell, besteht das Transformer-Modell grundsätzlich aus zwei Teilen: dem Encoder und dem Decoder. Anders als klassische „sequence to sequence“ Modelle nutzt ein Tranformer keine rückgekoppelten neuronalen Netze, um die Verbindungen und Reihenfolge der Eingabesequenzen zu berücksichtigen. Wie der Name des Papers schon vermuten lässt, konzentriert sich das Tranformer-Modell Komplet auf den Attention-Mechanismus.

Die Struktur des Transformer Modells ist in Abbildung 1 zu sehen. Dabei stellt die linke Seite den Encoder dar, während die rechte Seite den Decoder zeigt.

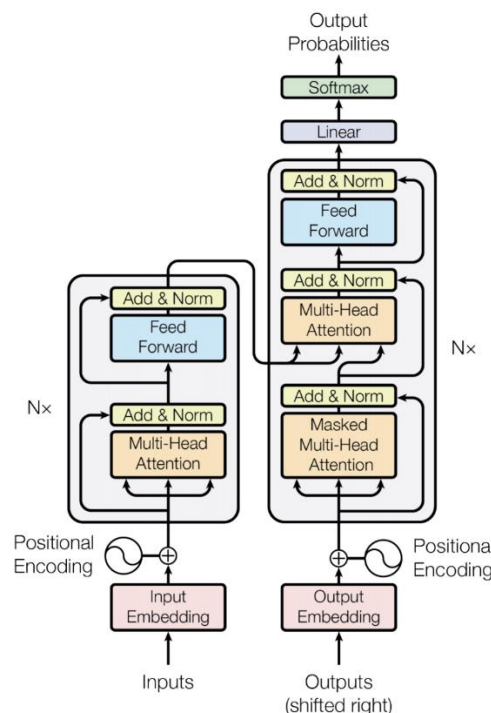


Abbildung 1: Architektur des Transformer Modells [12]

Sowie der Encoder als auch der Decoder bestehen aus aufeinander gestapelten Schichten, dies wird durch „ $N \times$ “ an den jeweiligen Seiten des Encoders und des Decoders beschrieben. Jede Schicht besteht wiederum aus zwei verschiedenen Sub-Schichten, das sind einerseits die



„Multi-Head Attention“ Module und andererseits das Feed-Forward Netzwerk. Im Falle des Decoders werden zwei „Multi-Head Attention“ Module aufeinandergestapelt. Um alle Sub-Schichten herum werden residuale Verbindungen sowie eine Normalisierungsfunktion eingesetzt (in der Grafik als „Add & Norm“). Eine residuale Verbindung ist nichts weiter als eine Verbindung innerhalb des Netzwerkes, die es erlaubt, bestimmte Schichten des neuronalen Netzes zu überspringen. Das Prinzip des „Add & Norm“ Blocks kann mithilfe dieser Formel veranschaulicht werden:  $LayerNorm(x + Sublayer(x))$ . Dabei steht  $Sublayer(x)$  für den Output der jeweiligen Schicht, also das „Multi-Head Attention“ Modul oder das Feed Forward Netzwerk. Die entsprechende Schicht wird also einmal übersprungen und einmal ausgeführt, daraufhin werden die beiden Ergebnisse addiert und schließlich die Normalisierung durchgeführt. Das einsetzen von residualen Verbindungen hat in erster Linie folgenden Vorteil: Wie in Abbildung 2 zu sehen ist, können tiefere Netzwerke, also neuronale Netze mit mehr Schichten, anders als eigentlich zu erwarten wäre, oft zu einer höheren Fehlerrate führen als neuronale Netze mit weniger Schichten. Die Nutzung eines residualen Netzwerkes, durch das Implementieren entsprechender residualer Verbindungen, kann diesem Phänomen entgegen wirken.

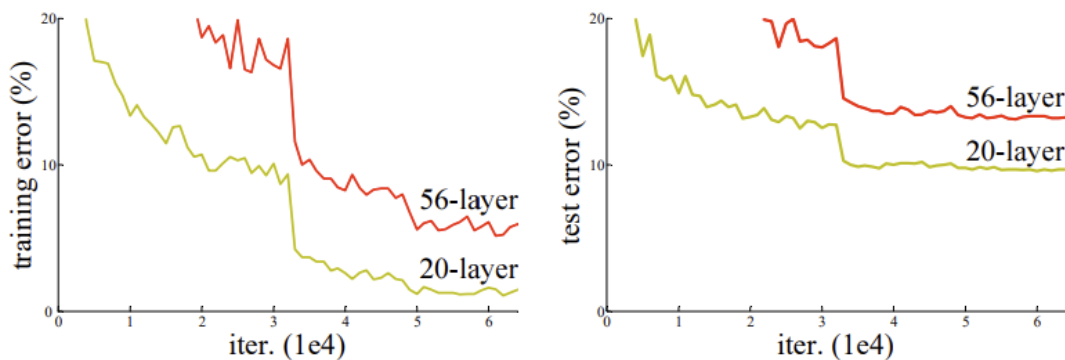


Abbildung 2: Beispielhafte Trainings- und Test-fehlerrate eines neuronalen Netzes mit 20 Schichten im Vergleich mit 56 Schichten auf dem CIFAR-10 Datensatz [13]

Im Decoder wird zusätzlich noch eine zweite „Multi-Head Attention“ Schicht eingebaut. Dies ist notwendig, um den Encoder und den Decoder miteinander zu verbinden und in Beziehung zu bringen.

Da es sich hier nicht um ein rückgekoppeltes neuronales Netz handelt, müssen sich die Positionen der Elemente einer Sequenz auf andere Art und Weise gemerkt werden. Das macht die ersten beiden Schritte „Embedding Input“ bzw. „Embedding Output“ und „Positional Encoding“ besonders wichtig. Beim „Embedding“ werden der Input oder Output einfach in ein off Modell verständliches Format überführt, in diesem Fall ein n-dimensionaler Vector. Die Aufgabe des „Positional Encoding“ kann mithilfe verschiedener Funktionen erreicht werden. In der ursprünglichen Arbeit „Attention Is All You Need“ werden zum Beispiel folgende Sinus- und Kosinus Funktionen verwendet:

$$\text{Encoder: } PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$\text{Decoder: } PE(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

## 2. Funktionsweise

Dabei steht  $pos$  für die Position und  $i$  für die Dimension, außerdem wurde folgendes festgelegt:  $d_{model} = 512$ . Es wären aber auch andere Funktionen für das „Positinal Encoding“ denkbar.

Unabhängig vom „embedding“ und dem „Positional Encoding“ ist es allerdings wichtig, den Output um einen Position nach rechts zu verschieben. Würde keine Verschiebung stattfinden, würde das neuronale Netz vermutlich nur lernen, dass das Element  $i$  des Inputs immer gleich dem Element  $i$  des Outputs entspricht. Das Modell würde also lernen, immer nur den Input zu kopieren, das Ziel ist es aber das nächste Element der Sequenz vorausszusagen.

Nun, wo der grundlegende Aufbau des Transformers beschrieben ist, wird es wichtig, sich die Details der Sub-Schichten anzusehen. „Feed Forward“ ist einfach nur eine Implementierung eines normalen „Feed Forward“ Netzes, welches in Kapitel 2.1 bereits erläutert wurde, hier gibt es keine weiteren Besonderheiten. Das „Multi-Head Attention“ Modul ist allerdings sehr wichtig, schließlich liegt der Fokus beim Transformer-Modell auf dem Bereich „Attention“.

Um die Multi-Head Attention Schicht verstehen zu können, ist es zuerst notwendig, die Scaled Dot-Product Attention zu verstehen, da diese den Komplexesten Teil des Multi-Head Attention Moduls ausmacht. Die Funktionsweise ist anhand von Abbildung 3 beschrieben.

Scaled Dot-Product Attention

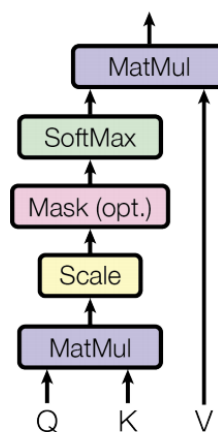


Abbildung 3: Scaled Dot-Product Attention [12]

Die Funktionsweise der Scaled Dot-Product Attention wird klarer, wenn die Grafik in ihre Formel übersetzt wird:  $Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ . Dabei ist es wichtig zu wissen, dass der Transformer die verschiedenen Inputs jeweils in drei verschiedene Vektoren unterteilt. Einerseits gibt es die Anfragen (Queries), das sind einzelne Elemente einer Sequenz, zum Beispiel ein einzelnes Wort aus einem Satz. In der Praxis werden gleich mehrere Queries in einer Matrix vereint. Auf der anderen Seite gibt es die Schlüssel-Werte (Key-Value) Paare, die jeweils in einer eigenen Matrix dargestellt werden und die gesamte Input-Sequenz widerspiegeln.

In Bezug auf die Grafik bzw. die Formel für das Scaled Dot-Product Attention bedeutet das Folgendes:  $Q$  ist die Matrix der Queries,  $K$  sind die Keys und  $V$  sind die Values. Außerdem

## 2. Funktionsweise

steht  $d_k$  einfach für die Dimension von  $K$  und Softmax ist eine Verteilungsfunktion, die das Ergebnis aus der Klammer auf einen Wert zwischen 0 und 1 verteilt.

Vereinfacht könnte man sagen, dass mit  $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$  die Gewichte der Attention Funktion berechnet werden. Die Gewichte berechnen sich also grundlegend indem alle Elemente der Sequenz dargestellt als Query  $Q$  mit allen Elementen der Sequenz dargestellt als Key  $K$  in Beziehung zueinander gesetzt werden, um so deren Einfluss aufeinander abzubilden. Diese Gewichte werden dann mit allen Werten von  $V$  verrechnet. So ergibt sich eine Matrix aller Werte  $V$ , die durch die Attention Funktion gewichtet wurden. Der Unterschied zur normalen Dot-Product Attention besteht dabei in erster Linie durch die Skalierung von  $QK^T$ , indem durch  $\sqrt{d_k}$  geteilt wird. Daher der Name Scaled Dot-Product Attention. In Bezug auf ein Texterzeugungssystem könnte eine Sequenz zum Beispiel ein Satz sein und ein Element der Sequenz wäre dementsprechend ein einzelnes Wort. Anders als bei einem LSTM wird so nicht jedes Wort einzeln betrachtet, sondern immer ganze Sequenzen am Stück bearbeitet.

In der Funktionsweise der Scaled Dot-Product Attention findet sich auch der Grund dafür das im Decoder zwei Multi-Head Attention Schichten zu finden sind. Im ersten Multi-Head Attention Modul, sowie im Decoder als auch im Encoder, wird die Scaled Dot-Product Attention wie beschrieben berechnet. Dabei sind die Queries  $Q$  sowie die Values  $V$  (und entfernter auch die Keys  $K$ , schließlich sind es zusammengehörige Key-Value Paare) Matrix Repräsentationen der gleichen Sequenz. Aus diesem Grund werden diese Schichten auch als Self-Attention bezeichnet. Im zweiten Multi-Head Attention Modul des Decoders werden nun die Ergebnisse des Encoders mit den Ergebnissen des Decoders vermischt, indem die Scaled Dot-Product Attention mit Variablen aus beiden Ergebnissen befüllt wird. So werden hier nun die Key-Value Matrizen des Encoders und die Query Matrix des Decoders als Input für die Attention Funktion genutzt. Dies wird dargestellt durch die entsprechenden Pfeile in Abbildung 1.

Nachdem die Funktionsweise der Scaled Dot-Product Attention nun klar ist, wird es jetzt möglich die Vorgehensweise des gesamten Multi-Head Attention Moduls zu verstehen, diese wird in Abbildung 4 veranschaulicht.

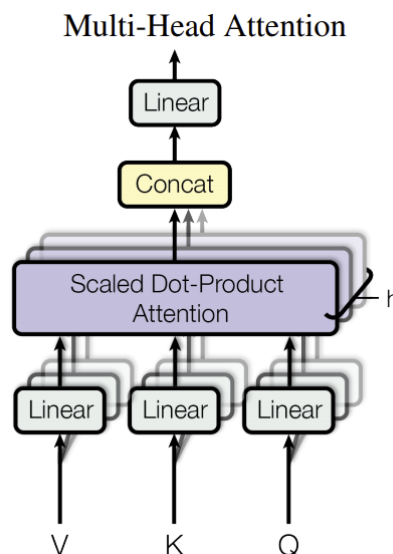


Abbildung 4: Multi-Head Attention Schicht [12]

Die grundsätzliche Idee des Multi-Head Attention Moduls ist es, die zuvor beschriebene Scaled Dot-Product Attention Parallel mehrfach für verschieden linear projizierte Matrizen von  $V$ ,  $K$  und  $Q$  zu berechnen. Dazu werden die drei Matrizen zuerst linear projiziert, indem sie mit der entsprechenden Gewichts-Matrix  $W$  (für Weight) multipliziert werden. Diese Gewichte der Matrix  $W$  werden während des Lernprozesses angepasst und so ebenfalls vom Transformer-Modell erlernt. Der Vorteil liegt dabei darin, dass das Modell anhand verschiedener Repräsentationen der gleichen Daten lernen kann und so die Genauigkeit der Ergebnisse erhöht. Die Attention-Werte der verschiedenen linearen Projizierungen werden dann, wie zuvor beschrieben, mithilfe der Scaled Dot-Product Attention Formel berechnet, durch Parallelisierung dieses Vorganges passiert dies mehrfach gleichzeitig. Die Darstellung dieses Ablaufs als Formel hilft das Prinzip zu erläutern:

$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ . Jedes einzelne Ergebnis der parallelen Berechnungen wird dabei als ein *head* bezeichnet. Durch die parallelen Berechnungen der *head* Ergebnisse ergibt sich der Name des Moduls: Multi-Head Attention.

Sind alle Attention-Werte berechnet werden die verschiedenen Ergebnisse zusammengeführt (in Abbildung 4 als „Concat“ bezeichnet) und das Ergebnis schließlich ein letztes Mal mit einer Gewicht-Matrix multipliziert. So ergeben sich die finalen Werte des Multi-Head Attention Moduls. Zur Veranschaulichung dient auch hier wieder eine Formel:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O [12]$$

Das zuvor beschriebene Modell stellt den Aufbau eines normalen Transformer-Modells dar. GPT nutzt allerdings eine abgewandelte Variante, den „Transformer Decoder“, welcher in der Arbeit „Generating wikipedia by summarizing long sequences“ [14] erstmals vorgestellt wurde. Die Besonderheit des Transformer Decoders ist das der Encoder aus der klassischen Transformer-Struktur wegfällt. Dafür müssen der Input- und Outputsequenzen zu einer einzigen Sequenz vereint werden. Sei die Inputsequenz  $m = (m^1, \dots, m^n)$  und Outputsequenz  $y = (y^1, \dots, y^x)$  gegeben erfolgt die Zusammenführung der Sequenzen in eine Sequenz  $w$  wie folgt:  $(w^1, \dots, w^{n+x+1}) = (m^1, \dots, m^n, \delta, y^1, \dots, y^x)$ .  $\delta$  ist dabei ein spezielles Element zur Separierung der beiden ursprünglichen Sequenzen. Diese spezielle Variante des Transformers soll durch eine starke Reduzierung der Modellparameter um fast 50% (durch die Entfernung des Encoders) eine höhere Effektivität bei besonders langen Sequenzen bieten. [14]

Dieser veränderte Aufbau im Falle von GPT lässt sich mit Abbildung 5 veranschaulichen. Wie zuvor beschrieben, ist auch in der Abbildung zu sehen, dass die typische Encoder-Decoder Struktur nicht vorhanden ist. GPT nutzt dabei ein 12-schichtiges Design.

## 2. Funktionsweise

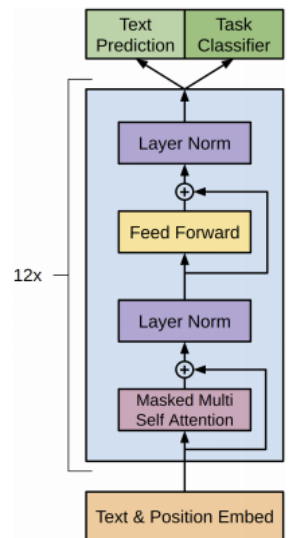


Abbildung 5: Aufbau des Transformer Decoders im Fall von GPT [15]

Damit sollte nun ein Überblick über die Funktionsweise des Transformer Modells, auf dem die GPT Reihe basiert, gegeben sein. Im Folgenden wird nun das Konzept des „Semi-supervised training“ und dessen Implementierung in GPT dargestellt, welches den zweiten Teil der beiden Grundkonzepte von GPT bildet.

Die grundsätzliche Idee des „Semi-supervised training“ ist es ein Modell anhand eines großen Datensatzes unüberwacht- und dann, anhand eines kleineren Datensatzes, überwacht anzulernen. Die Anwendung dieses Konzepts auf „sequence to sequence“ Modelle, genauer auf ein „Long short-term memory Modell“, wird in der Arbeit „Semi-supervised Sequence Learning“ von Andrew M. Dai und Quoc V. Le beschrieben, die Arbeit von OpenAI baut darauf auf. Dazu wird bei GPT zuerst der Prozess des „unsupervised pre-training“ angewandt. Dabei wird das Modell anhand eines großen Datensatzes unüberwacht vortrainiert. Anders als beim normalen unüberwachten Lernen ist dies aber nur die Vorbereitung, danach wird das Modell zusätzlich im „supervised fine-tuning“ mittels überwachten Lernens auf spezielle Aufgabenbereiche angepasst. Dabei ist wichtig, dass beim Wechsel des Datensatzes, und damit dem Wechsel vom „Unsupervised pre-training“ auf „Supervised fine-tuning“, auch die Funktionen zur Berechnung der Wahrscheinlichkeiten angepasst werden. Das Ziel des GPT Modells, oder auch jedes anderen Texterzeugungsmodells, ist es, das Wahrscheinlichste nächste Element einer Sequenz zu finden, dazu werden die Wahrscheinlichkeits-Funktionen  $L$  maximiert. Im Schritt des „Supervised fine-tuning“ wird, neben dem Hauptziel die Wahrscheinlichkeits-Funktion zu maximieren, zusätzlich eine Nebenbedingung implementiert. Die Funktion für die Nebenbedingung sieht wie folgt aus:  $L_3(C) = L_2(C) + \lambda * L_1(C)$ . Dabei ist  $L_1$  die Wahrscheinlichkeits-Funktion des „Unsupervised pre-training“ und  $L_2$  die Wahrscheinlichkeits-Funktion vom „Supervised fine-tuning“ mit  $\lambda$  als zusätzliches Gewicht,  $C$  stellt den Datensatz des überwachten Trainings dar. Das Hinzufügen dieser Nebenbedingung ermöglicht eine schnellere Konvergenz sowie eine bessere Allgemeingültigkeit des überwachten Modells. [15]

Wie die Inputs für das „Supervised fine-tuning“ aussehen können ist in der Folgenden Abbildung beispielhaft dargestellt.

## 2. Funktionsweise

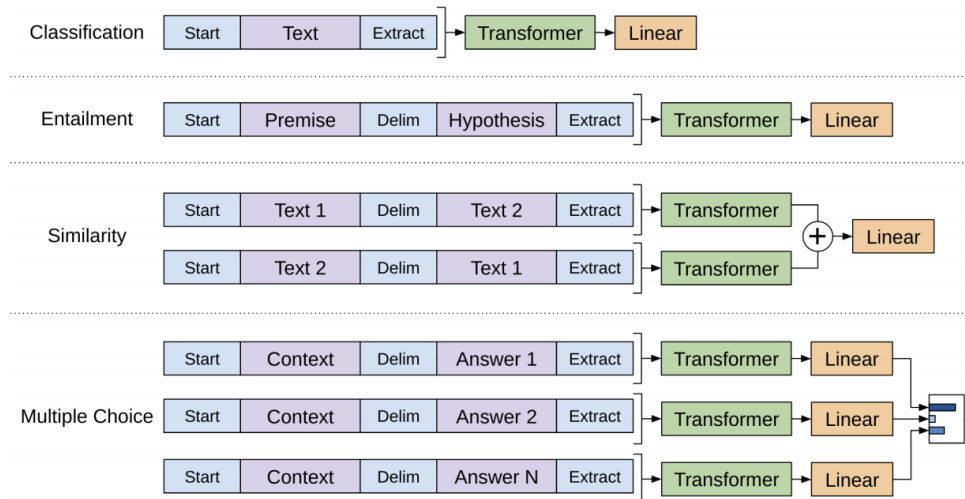


Abbildung 6: Inputs für das Fine-Tuning auf vier verschiedene Aufgaben [15]

Für strukturell einfache Fälle, wie zum Beispiel der Text-Klassifizierung kann das Modell problemlos und ohne weitere Bearbeitung des Inputs trainiert werden. Im zweiten, dritten und vierten Aufgabenbereich sind Fälle abgebildet, bei denen die Struktur des Textes eine Rolle spielt. Um diese Struktur auch in der Inputsequenz widerzuspiegeln, werden hier Trennungselemente eingesetzt. Dies muss geschehen da das Modell im „Unsupervised pre-training“ anhand von Fließtexten trainiert wird. Deshalb müssen auch strukturierte Texte in eine Art Fließtext überführt werden, der allerdings die ursprüngliche Struktur wiedergibt.

Im dritten Fall, der Erlernung von Ähnlichkeiten zwischen Texten, ist zu sehen, dass für ein besseres Ergebnis beide Texte einmal an der Stelle des Inputs und der Stelle des Outputs innerhalb der Sequenz stehen. Dies wird gemacht, da bei zwei sich ähnelnden Texten die Reihenfolge keine Rolle spielt. Die beiden Ergebnisse werden dann miteinander verrechnet, bevor weiter mit ihnen gearbeitet werden kann.

Bei der Aufgabe der „Multiple Choice“ Fragen ist zudem noch eine andere Besonderheit sichtbar. Der Aufbau einer solchen Frage sieht klassisch so aus, dass für eine Frage mehrere Antwortmöglichkeiten zur Verfügung stehen. Um diese Art des Inputs in eine für das Modell verständliche Sequenz zu überführen, wird jede Antwortmöglichkeit einzeln mit der dazugehörigen Frage kombiniert. Am Ende entsteht dadurch eine Verteilung über die Wahrscheinlichkeiten der verschiedenen Antwortmöglichkeiten. [15]

Außerdem spielt hier speziell die zuvor beschriebene Struktur des Transformer Decoders eine Rolle. Es ist zu erkennen, dass die eigentlichen Sequenzen mit Start-, Trennungs- und End Elementen versehen sind, ein typischer Aufbau für einen Transformer Decoder. Würde hier ein klassischer Transformer verwendet werden müssten dies Sequenzen an der Stelle des Trennungselements (in der Abbildung: „Delim“ für „delimiter“) in eine Inputsequenz für den Encoder und eine Outputsequenz für den Decoder unterteilt werden.

Durch die Vermischung von Unüberwachten und Überwachtem lernen ist es möglich, GPT anhand eines nicht aufwendig zu erzeugenden unüberwachten Datensatzes zu trainieren. Überwachte Datensätze sind deutlich aufwendiger zu erzeugen. Dank des Pre-Trainings ist für das Finetuning dann aber nur noch ein sehr viel kleinerer Datensatz nötig. So werden die Vorteile beider Ansätze verbunden.



## 2. Funktionsweise

Zum Abschluss des Kapitels „Funktionsweise von GPT“ soll die zuvor dargestellte Theorie einmal anhand eines kleinen Beispiels aus der Arbeit „Language Models are Unsupervised Multitask Learners“ [16], mit der GPT-2 vorgestellt wurde, veranschaulicht werden, bevor im nächsten Kapitel dann einige eigene Beispiele mit GPT-2 erarbeitet werden.

Question	Generated Answer	Correct	Probability
Who wrote the book the origin of species?	Charles Darwin	✓	83.4%
Who is the founder of the ubuntu project?	Mark Shuttleworth	✓	82.0%
Who is the quarterback for the green bay packers?	Aaron Rodgers	✓	81.1%
Panda is a national animal of which country?	China	✓	76.8%
Who came up with the theory of relativity?	Albert Einstein	✓	76.4%
When was the first star wars film released?	1977	✓	71.4%
What is the most common blood type in sweden?	A	✗	70.6%
Who is regarded as the founder of psychoanalysis?	Sigmund Freud	✓	69.3%
Who took the first steps on the moon in 1969?	Neil Armstrong	✓	66.8%
Who is the largest supermarket chain in the uk?	Tesco	✓	65.3%
What is the meaning of shalom in english?	peace	✓	64.0%
Who was the author of the art of war?	Sun Tzu	✓	59.6%
Largest state in the us by land mass?	California	✗	59.2%
Green algae is an example of which type of reproduction?	parthenogenesis	✗	56.5%
Vikram samvat calender is official in which country?	India	✓	55.6%
Who is mostly responsible for writing the declaration of independence?	Thomas Jefferson	✓	53.3%
What us state forms the western boundary of montana?	Montana	✗	52.3%
Who plays ser davos in game of thrones?	Peter Dinklage	✗	52.1%
Who appoints the chair of the federal reserve system?	Janet Yellen	✗	51.5%
State the process that divides one nucleus into two genetically identical nuclei?	mitosis	✓	50.7%
Who won the most mvp awards in the nba?	Michael Jordan	✗	50.2%
What river is associated with the city of rome?	the Tiber	✓	48.6%
Who is the first president to be impeached?	Andrew Johnson	✓	48.3%
Who is the head of the department of homeland security 2017?	John Kelly	✓	47.0%
What is the name given to the common currency to the european union?	Euro	✓	46.8%
What was the emperor name in star wars?	Palpatine	✓	46.5%
Do you have to have a gun permit to shoot at a range?	No	✓	46.4%
Who proposed evolution in 1859 as the basis of biological development?	Charles Darwin	✓	45.7%
Nuclear power plant that blew up in russia?	Chernobyl	✓	45.7%
Who played john connor in the original terminator?	Arnold Schwarzenegger	✗	45.2%

*Tabelle 1: 30 von GPT-2 beantwortete Fragen, sortiert nach der von GPT-2 berechneten Wahrscheinlichkeit für die gegebenen Antwort. Quelle: [16]*

In Tabelle 1 sind an GPT-2 gestellte Fragen sowie die von GPT-2 gegebenen Antworten und ihrer Wahrscheinlichkeit laut GPT-2 zu sehen. Als Datensatz für das Pre-Training wurde bei GPT-2 eine Version von „WebText“ verwendet, die insgesamt etwa 40GB Text umfasst. Um die Resultate aus Tabelle 1 einordnen zu können, ist es wichtig zu wissen, dass keine der Fragen direkt im Datensatz vorkommt. Zur Beantwortung ist also nötig, die entsprechenden Texte sowie auch die Fragen zu verstehen, um so eine Verbindung zwischen beiden herstellen zu können und die Fragen zu beantworten. Dabei sei gesagt, dass das Wort „verstehen“ hier in den Kontext eines Maschine Learning Modells gesetzt werden muss, dass GPT-2 die Texte und Fragen nicht wie ein Mensch „verstehen“ kann, ist klar und doch werden im Modell die entsprechenden Verbindungen zwischen Fragen und dazugehörigen Texten erkannt. Besonders interessant sind die etwa 25% der Fragen, die falsch beantwortet wurden. So gibt GPT-2 auf die Frage „Who played john connor in the original terminator?“ die Antwort „Arnold Schwarzenegger“ oder auf die Frage „What is he most common blood type in sweden?“ die Antwort „A“. Die Antworten sind zwar falsch, aber nicht grundlegend inkorrekt, Arnold Schwarzenegger hat im ersten Terminator Film mitgespielt und „A“ ist eine existierende Blutgruppe. Das zeigt, dass das Modell die Fragen verstanden hat, aber wie es auch einem Menschen passieren könnte, die Fragen falsch beantwortet.

Tabelle 1 macht deutlich, dass die Ansätze von GPT funktionieren und zu guten Ergebnissen führen können.

### 2.2.3. Programmbeispiele mit GPT

Bevor man mit den eigentlichen Experimenten mit GPT beginnen kann, müssen einige Grundlagen geklärt werden. Leider war es zu Zeitpunkt des Schreibens dieser Arbeit nicht möglich auf die OpenAI API und damit GPT-3 zuzugreifen, deshalb wird, wie vorher bereits angekündigt GPT-2 für diese Experimente verwendet. Der direkt von OpenAI veröffentlichte Code steht in einem GitHub Repository ([github.com/openai/gpt-2](https://github.com/openai/gpt-2)) zur Verfügung. In diesem Repo stehen, neben dem eigentlichen Modell, allerdings nur die nötigsten Funktionen zur Verfügung um erste Ausgaben mithilfe von GPT-2 zu erreichen. Insbesondere für das Finetuning steht gibt hier keine einfache Möglichkeit. Da GPT-2 nun allerdings schon länger zur Verfügung steht, ist es nicht nötig die zusätzlichen Funktionen selbst zu schreiben. Stattdessen gibt es auf GitHub verschiedenen Projekte in denen dies bereits getan wurde. Nach einiger Recherche zu den verschiedenen Repos stellt sich heraus das „gpt-2-simple“ vom GitHub Nutzer „minimaxir“ ([github.com/minimaxir/gpt-2-simple](https://github.com/minimaxir/gpt-2-simple)) die Nutzung nicht nur vereinfacht, sondern zum Beispiel auch eine einfache Möglichkeit des Finetunings hinzufügt. Dabei stellt „gpt-2-simple“ eine der beliebtesten Möglichkeiten der Interaktion mit GPT-2 dar, weshalb es einfach ist, im Internet Anleitungen zur Nutzung zu finden. Aus diesen Gründen wurden alle Programmbeispiele mithilfe von „gpt-2-simple“ durchgeführt. Da das ursprüngliche GPT-2 Repository nicht mehr gewartet wird, sind für die Ausführung des Codes sehr spezielle und veraltete Versionen der Abhängigkeiten nötig. So nutzt GPT-2 TensorFlow, funktioniert allerdings nicht mit den neueren TensorFlow 2.x Versionen, was wiederum spezielle und lange veraltete Versionen der Nvidia CUDA Treiber voraussetzt.

Das GPT-2 Modell ist in vier verschiedenen Größen erschienen: 124M, 355M, 774M und 1558M. Der Name gibt dabei jeweils die Menge an Hyperparameter in Millionen an. 1558M ist dabei das Vollständige GPT-2 Modell mit allen 1,5 Milliarden Parametern. Die Unterschiede der vier Versionen sind schnell erklärt. Je mehr Hyperparameter desto besser das Ergebnis der Ausgabe, desto höher aber auch die Hardwareanforderung für das Finetuning und die Erzeugung einer Ausgabe. In den Beispielen wurde in erster Linie das 355M sowie das 774M Modell genutzt. Die Gründe hierfür werden später, im Kapitel „Auswertung der Programmbeispielen“ genauer erläutert. Grundsätzlich sei aber gesagt das die Auswahl der Modelle, in erster Linie von der zur Verfügung stehenden Hardware geleitet wurde und in erster Linie mithilfe von Google Collaboratory ausgeführt wurden.

Was den eigentlichen Quellcode angeht sind dank „gpt-2-simple“ nur wenige Zeilen Quellcode notwendig. Dabei gibt es eine beliebte Google Collaboratory Vorlage von Max Woolf, dem Entwickler von gpt-2-simple, in der die nötigen Befehle für das Finetuning erklärt werden. Der für die Experimente genutzte Code basiert grob auf diesem Vorbild. Die grundlegenden Befehle für das Finetuning sind in Abbildung 7 zu sehen.



## 2. Funktionsweise

```
# Download the GPT-2 model
model_to_use="774M"
gpt2.download_gpt2(model_name=model_to_use)

# the name of the txt file containing the dataset
for finetuning
file_name = "QAdataset.txt"

# create TensorFlow Session
sess = gpt2.start_tf_sess()

# Start finetuning process
gpt2.finetune(sess,
               dataset=file_name,
               model_name=model_to_use,
               steps=4000,
               restore_from='fresh',
               run_name='QA_4000steps_774M',
               print_every=10,
               sample_every=2000,
               save_every=500,
               )
```

Abbildung 7: Code für das Finetuning von GPT-2 mithilfe von gpt-2-simple

Wie in der Abbildung zu sehen ist sind grundlegend nur drei Befehle nötig, um ein neues Modell zu erzeugen das anhand eines eigenen Datensatzes trainiert wurde. Dabei ist allerdings zu bedenken, dass der in Abbildung 7 zu sehende code nicht vollständig ist, es fehlen die Imports sowie einige Befehle, um Dateien zwischen Google Collaboratory und Google Drive hin und her zu kopieren. Der vollständige code ist im entsprechenden Collaboratory Notebook zu finden welches im GitHub Repository verlinkt ist. Der wichtigste Teil in Abbildung 7 ist die gpt2.finetune Funktion. Neben den Angaben zum Datensatz, dem gpt-2 Modell, welches genutzt werden soll, dem Name des zu erzeugenden Modells sowie verschiedenen Angaben zur Ausgabe und Speicherung während des Trainings, wird hier vor allem auch die Anzahl an Trainingsschritten (steps) festgelegt. Im Beispiel aus der Abbildung sind es 4000. Bei einem ausreichend großen Datensatz führt eine Erhöhung der Trainingsschritten zu besseren Ausgaben des Trainierten Modells aber auch zu einer höheren Trainingszeit. Zu bekundenden bei der Wahl der Anzahl an Trainingsschritten, ist aber auch das ab einem bestimmten Punkt eine weitere Erhöhung der Trainingsschritte nur noch zu marginal besseren Ergebnissen führt. Die Zahl der Trainingsschritte sollte also weder zu niedrig, dies führt zu schlechten Ergebnissen, noch zu hoch, dies führt zu sehr langen Trainingszeiten, gewählt werden. Die Optimale Anzahl an Trainingsschritten kann allerdings von Modell zu Modell und von Datensatz zu Datensatz unterschiedlich sein.

Um mithilfe des neu erzeugten Modells schließlich auch eine Ausgabe zu generieren. Steht in gpt-2-simple die „generate“ Funktion bereit, bei der neben einer Textvorgabe auch angegeben werden kann, wie viele ausgaben erzeugt werden sollen und wie lang jede Ausgabe sein soll. Dabei kann GPT-2 allerdings keine Ausgaben über 1024 Token erzeugen. Zudem kann die Ausgabe zum Beispiel per „truncate“ an einem bestimmten Punkt abgeschnitten werden oder die „batch\_size“ sowie weitere Eigenschaften angegeben werden.

Im Folgenden werden einige verschiedene Programmbeispiele beschrieben, die verschiedene Aspekte der Texterzeugung testen sollen und in diese Kategorien unterteilt werden können:

## 2. Funktionsweise

1. Verstehendes Lesen. Dabei wird eine Frage zu einem vorgegebenen Text gestellt, die vom Modell beantwortet werden soll.

2. Abstraktere Sequenzen. Das Modell soll Mathematische Aufgaben lösen.

Dabei soll das Ziel der Experimente nicht nur die Erlernung der Arbeitsweise mit GPT-2 sein, sondern auch die Möglichkeit bieten, die Unterschiede zwischen den verschiedenen Varianten von GPT-2 in der Praxis kennenzulernen, um festzustellen wie wichtig die Unterschiede zwischen verschiedenen großen neuronalen Netzen sind. Dabei sei im Voraus gesagt, dass alle Beispiele auf Englisch stattfinden. Erstens wurde GPT-2 anhand von englischem Text trainiert was bedeutet dass die Nutzung einer anderen Sprache die Ergebnisse verfälschen könnte, auch wenn es Theoretisch auch möglich ist, GPT-2 eine andere Sprache beizubringen. Zweitens ist es für das Finetuning deutlich einfacher Datensätze auf Englisch zu finden als Datensätze, die in deutscher Sprache verfasst sind.

Die im folgenden besprochenen Modelle und Datensätze sowie auch die Rohdaten der Ausgaben sind im zugehörigen GitHub Repository zu finden (LINK). Zusätzlich sind dort auch alle erstellten Skripte sowie Verlinkungen zu den Google Collaboratory Notebooks zu finden.

Die ersten Programmbeispiele beschäftigen sich mit dem Thema verstehendes Lesen. Damit ist folgende Aufgabe gemeint: Ein vorgegebener Text liefert Informationen über ein beliebiges Thema und dient als Kontext zu einer dann zum Text gestellten Frage. Diese Frage kann, ohne weiteres Wissen zum Thema, nur mit dem vorgegebenen Text beantwortet werden. Der Text kann dabei auch fiktiv sein oder Fehlerhafte Informationen enthalten. Ziel ist eigentlich nicht die korrekte Beantwortung der Frage an sich, sondern das Finden der entsprechenden Textstelle, auf die sich die Frage bezieht. Das sieht dann beispielsweise so aus:

**Kontext:** Microsoft wurde 1975 von Bill Gates und Paul Allen gegründet. Mit 168.000 Mitarbeitern ist Microsoft der Weltweit Größte Softwarehersteller.

**Frage:** Wann wurde Microsoft gegründet?

**Antwort:** 1975

Für einen Erwachsenen Menschen ist die Lösung einer solchen Aufgabe in der Regel selbst bei Komplexeren Fragen kein Problem. Wie das bei GPT-2 aussieht wird sich im Folgenden Beispiel zeigen.

Wird GPT-2 ohne weiter Vorarbeit mit dieser Aufgabe betraut, nur das natürlich die Antwort offenbleibt, stellt man fest dass GPT-2 nichts mit dem input anzufangen weiß. Anstatt die Frage zu beantworten, schreibt GPT-2 einen Text zum Thema der, abhängig von der ausgewählten Länge der Ausgabe, mehrere Absätze umfasst und die Frage nicht beantwortet. GPT-2 hat zwar gelernt Texte zu schreiben, aber ist nicht ohne weiteres in der Lage die, für Menschen offensichtliche, Aufgabestellung zu erkennen. Hier kommt das schon oft angesprochene Finetuning zum Spiel. Durch Finetuning mittels eines Datensatzes mit einer Vielzahl dieser Aufgaben ist es möglich GPT-2 die Aufgabe und das gewünschte Format des Outputs beizubringen. Da die Qualität des Datensatzes beim Trainieren von Neuronalen Netzen eine entscheidende Rolle spielt macht es hier wenig Sinn einen entsprechenden Datensatz selbst zu erstellen. Dies wäre enorm aufwendig und vermutlich qualitativ eher

schlecht. Stattdessen war es nach kurzer Recherche möglich einen bereits existierenden Datensatz zu finden. Im Artikel „Question Answering with GPT-2“ von Chloe Reams [17] wird beispielhaft beschrieben die Aufgabe des verstehenden Lesens mit GPT-2 umgesetzt werden kann. Da dies die ersten Experimente mit GPT-2 waren, basiert das hier dargestellte Beispiel grob auf den genannten Artikel, wobei einige wichtige Punkte, wie die Formatierung des Datensatzes stark abweichen. In diesem Artikel wird die Nutzung des SQUAD (Stanford Question Answering Dataset) vorgeschlagen. Die für die folgenden Beispiele genutzte Version des Datensatzes umfasst über 78.000 verschiedene Fragen zu den unterschiedlichsten Themen. Der Kontext und die Antworten sind jeweils ebenfalls vorgegeben. Dabei darf die hier genutzte Version nicht mit SQUAD2.0 verwechselt werden. SQUAD2.0 umfasst zusätzlich auch Fragen die mit dem gegebenen Kontext gar nicht beantwortet werden können. Der genutzte Datensatz steht auch im GitHub Repo unter „data“ bereit. Mit dem Finden eines passenden Datensatzes ist es allerdings noch nicht getan. Der Datensatz steht als CSV Datei bereit, für das Training ist allerdings, wie am Ende des Kapitels „Funktionsweise von GPT“ erklärt, eine Text Version des Datensatzes notwendig. In der Praxis bedeutet das Folgendes:

1. Datensatz Bereinigung: Im Datensatz sind verschiedene zusätzliche Informationen gegeben, die für das Finetuning nicht benötigt werden, diese müssen entfernt werden.
2. Formatierung der Daten: Damit GPT-2 den Datensatz verstehen kann und auch die gewünschte Struktur der Daten lernt ist es wichtig die Daten einheitlich zu Formatieren und mit speziellen Tokens voneinander zu trennen, dies ist die Praktische Implementierung der in Abbildung 6 dargestellten Beispiele. Dabei sieht die Formatierung der Daten in diesem Fall wie folgt aus: Mittels „<|startoftext|>“ und „<|endoftext|>“ werden die verschiedenen Aufgabenblöcke voneinander getrennt. Über die Tokens „[CONTEXT]“, „[QUESTION]“ und „[ANSWER]“ werden die die verschiedenen Teile der Aufgabe gekennzeichnet. In Abbildung 8 ist diese Formatierung an der vorherigen Beispielaufgabe zu sehen. Dabei sind die echten Texte, wie zuvor bereits erwähnt, allerdings alle in englischer Sprache.

```
<|startoftext|>  
  
[CONTEXT]: Microsoft wurde 1975 von Bill Gates und Paul Allen gegründet.  
Mit 168.000 Mitarbeitern ist Microsoft der Weltweit Größte  
Softwarehersteller.  
  
[QUESTION]: Wann wurde Microsoft gegründet?  
  
[ANSWER]: 1975  
  
<|endoftext|>
```

*Abbildung 8: Beispielhafte Formatierung einer einzelnen Aufgabe*

3. Beachten von Spezialfällen: In den Texten des Datensatzes kommen immer wieder auch Sonderzeichen vor, deshalb ist es wichtig beim Arbeiten mit den Daten immer darauf zu achten das UTF-8 Format zu verwenden. Außerdem gibt es einige wenige Spezialfälle, die aus dem Datensatz entfernt wurden. So ist die Antwort auf eine der Fragen zum Beispiel „null“ da dies in den meisten Programmiersprachen für ein leeres Objekt steht könnte diese Antwort Probleme bereiten. Die einfachste Lösung ist es solche Fälle aus dem Datensatz zu löschen.

Die Erzeugung der fürs Training benötigten Textdatei, unter Berücksichtigung der genannten Aspekte, erfolgt über ein kurzes Python Skript (im GitHub Repo unter `src/formatQA.py` zu finden). Anhand der so erzeugten Textdatei kann nun das Modell trainiert werden. Dazu wurden zum Großteil die von gpt-2-simple vorgeschlagenen Standardeinstellungen verwendet. Zum Vergleich werden hier nun die Zwei Finalen Modelle für diesen Aufgabenbereich vorgestellt:

Modell 1: Für das erste Modell wurde die 355M Variante von GPT-2 verwendet. Das Finetuning fand in 4000 Trainingsschritten statt und hatte am Ende des Trainings einen durchschnittlichen Loss von 1,02. Das Finetuning mit einer Nvidia Tesla P100 16GB dauerte 62 Minuten und das erzeugte Modell ist 1,33GB groß.

Modell 2: Für das zweite Modell wurde die größere 774M Variante von GPT-2 verwendet. Auch hier wurden wieder 4000 Trainingsschritte durchgeführt wobei hier ein etwas höherer durchschnittlicher Loss von 1,06 entstand. Das Finetuning fand wieder mithilfe einer Nvidia Tesla P100 16GB statt und dauerte mit 123 Minuten etwa doppelt so lange wie das kleinere Modell. Modell 2 ist mit etwa 2,9GB deutlich größer als Modell 1.

(Modell 3): Neben den beiden Modellen, die in den Beispielen vorgestellt werden, wurde noch ein drittes Modell anhand der 774M Varianten mit 8000 Trainingsschritten erstellt. Bei dieser wurde ein durchschnittlicher Loss von 0,98 erreicht. Das Training dauerte mit einer Nvidia Tesla V100 16GB 134 Minuten. Dank der schnelleren Grafikkarte dauerte das Training nur unwesentlich länger als bei Modell 2 obwohl hier die doppelte Menge an Trainingsschritten durchlaufen wurden. Das Modell ist ebenfalls 2,9GB groß. Der genaue Grund für das Training dieses dritten Modelles wird später erläutert, es wurde allerdings nicht direkt für die im Folgenden dargestellten Ergebnisse genutzt.

Um die Leistung der beiden Modelle zu testen, wurden beiden Modellen Verschiedenen Aufgaben gestellt. Hier werden nun beispielhaft drei Fragen zum gleichen Text gestellt und die Ergebnisse von den beiden Modellen dargestellt. Das Ziel bei der Wahl der Fragen war es drei Verschiedenen Schwierigkeitsstufen widerzuspiegeln. Um die Ergebnisse vergleichen zu können wurden alle Fragen 100-mal von den Modellen beantwortet, um so eine Verteilung an Antworten zu haben.

Folgendes ist der gegebene Kontext für die Fragen:

Ever since the discovery of Pluto in 1930, kids grew up learning that the solar system has nine planets. That all changed in the late 1990s, when astronomers started arguing about whether Pluto was indeed a planet. In a highly controversial decision, the International Astronomical Union ultimately decided in 2006 to designate Pluto as a "dwarf planet," reducing the list of the solar system's true planets to just eight.

*Abbildung 9: Kontext für die Fragen des Verstehenden Lesen. Quelle: [18]*

Die erste Frage zum Text ist folgende: „When was Pluto discovered?“. Die korrekte Antwort ist: 1930. Dies sollte die einfachste der Fragen darstellen. Bei der Frage handelt es sich eher um einen Proof of Concept um zu überprüfen ob die Modelle überhaupt funktionieren. Die Erwartung ist das beide Modelle kein Problem mit der Beantwortung der Frage haben sollten. Die Ergebnisse sind in Abbildung 10 zu sehen.

## 2. Funktionsweise

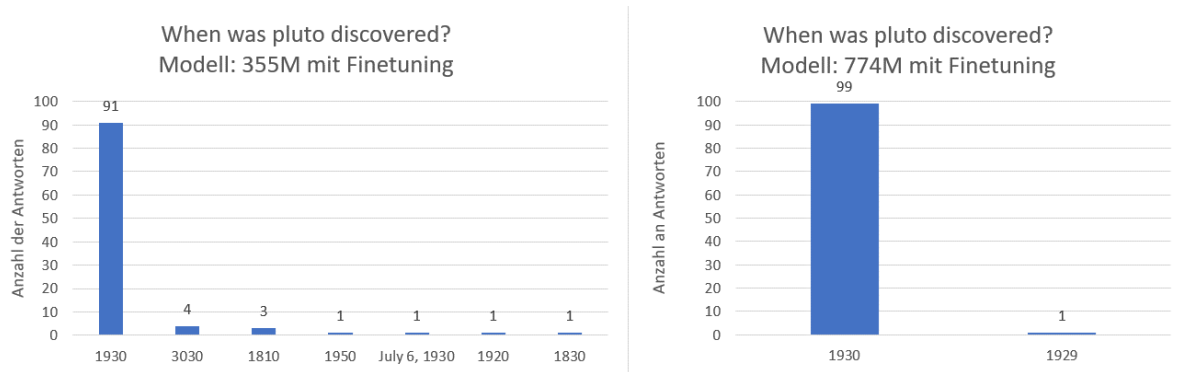


Abbildung 10: Vergleich der Antworten beider Modelle auf die Frage: „When was pluto discovered?“

Vergleicht man die Ergebnisse beider Modelle ist zuerst einmal zu sehen das beide Modelle die Frage in der Regel richtig beantworten, auch wenn das kleinere Modell selbst bei dieser einfachen Frage nur in 91% der Fälle die Richtige Antwort gibt, wohingegen das Größere Modell mit 99% nur ein einziges Mal die Falsche Antwort gibt. Dabei sei gesagt das vergleichbare Antworten zusammengerechnet wurden, die Antworten „1930“, „in 1930“ oder „in the year 1930“ würden alle als „1930“ gelten. Nur faktisch unterschiedliche Antworten werden auch einzeln aufgezählt, „late 1930“ wäre nicht das gleiche wie „1930“. Das gleich gilt auch für alle folgenden Beispiele. Besonders interessant sind hier wieder einmal die falschen Antworten. Das kleinere Modell gibt nicht nur öfter eine Falsche Antwort, sondern gibt dabei auch eine größere Menge an Falschen Antworten, als wäre sich das Modell insgesamt unsicherer dies wird weiterhin unterstützt das die zweithäufigste Antwort („3030“) in diesem Kontext wenig Sinn ergibt. Allerdings ist zu beachten, dass das größere Modell, da es nur einen Fehler gemacht hat, auch nicht die Möglichkeit hatte verschiedene falsche Antworten zu geben. Außerdem kann die Menge an verschiedenen Antworten auch mithilfe der „temperature“ beeinflusst werden. Ein „temperature“ Wert kann für die Generierung einer Ausgabe übergeben werden, dieser Wert ist leider schlecht dokumentiert. Er beeinflusst, aber generell die Zufallsverteilung mithilfe derer die Wörter ausgewählt werden, ein extrem geringer Wert wird dafür sorgen das nur sehr wenig verschiedene Wörter für die Ausgabe genutzt werden, bei einem extrem hohen Wert besteht der erzeugte Text aus so vielen verschiedenen Wörter, dass diese willkürlich zusammengewürfelt wirken. Ab etwa einem Wert von größer als 1 fängt die Grammatik langsam an negativ beeinflusst zu werden. In diesen Versuchen ist immer ein Wert von 0,7 genutzt worden.

Die zweite Frage war: „When did Pluto stop being a planet?“. Die erwartete Antwort wäre: „2006“. Dies ist die Frage mit mittlerer Schwierigkeit hier darf man nicht von dem Satz „That all changed in the late 1990s[...]“ verwirrt werden. Die Antwort steht später im Text eindeutig: „[...] ultimately decided in 2006 [...]“. Außerdem ist die Fragestellung absichtlich nicht mehr so eindeutig wie bei Frage 1. Die Ergebnisse sind in der Abbildung 11 zu sehen.

## 2. Funktionsweise

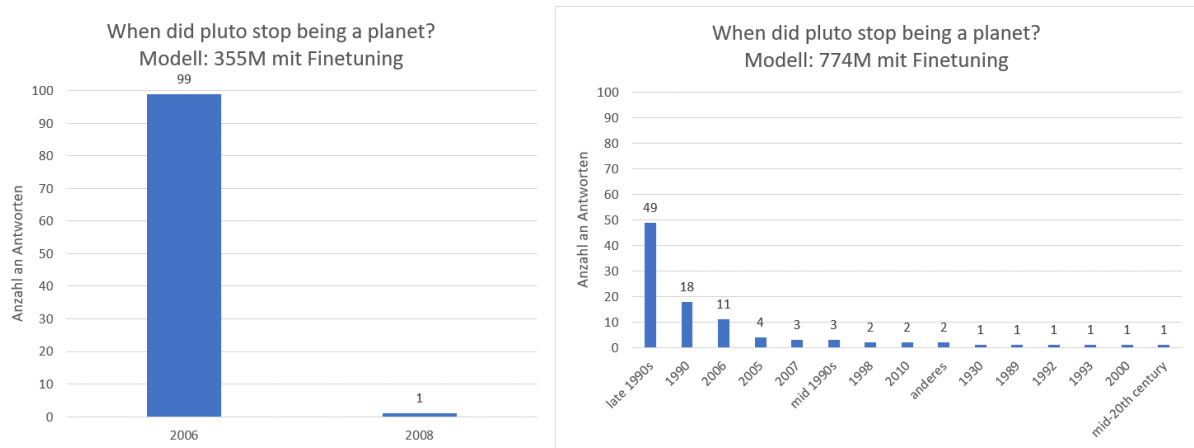


Abbildung 11: Vergleich der Antworten beider Modelle auf die Frage: „When did pluto stop being a planet?“

Das Ergebnis der zweiten Frage ist unerwartet. Paradoxerweise liefert das Kleinere Modell 1 hier deutlich bessere Ergebnisse als, wie eigentlich zu erwarten gewesen wäre, das größere Modell 2. Modell 1 übertrifft, trotz der schwierigeren Fragestellung, mit 99% Genauigkeit sogar das Ergebnis aus Frage 1, wohingegen Modell 2 bei dieser Frage, im Gegensatz zur Frage 1, vollständig versagt. Die mit Abstand häufigste Antwort ist hier „late 1990s“ gefolgt von „1990“. In insgesamt 67% der Antworten fällt das Modell damit auf die zuvor beschriebene Schwierigkeit der Frage hinein. Nur 11% der Antworten sind „2006“ und damit korrekt. Die Erwartung beim Ausführen dieses Experiments war das genaue Gegenteil des tatsächlichen Ergebnisses. Daraus stellt sich nun die Frage: Wie kommt es zu diesem Ergebnis? Die Einfachste Möglichkeit wäre folgende: Beim Finetuning des Größeren 774M Modells ist es nötig mehr Trainingsschritte zu machen als beim kleineren Modell um die eine Ähnliche Genauigkeit zu erreichen. Eigentlich sollte es zwar genau andersherum sein, das größere Modell brauch weniger Training als das Kleinere, aber da auch der durchschnittliche Loss bei Modell 2 höher ist als bei Modell 1 bestand diese Theorie. Um diese Hypothese zu überprüfen, wurde Modell 3 mit 8000 Trainingsschritten erstellt. Bei der Eingabe der gleichen Aufgabe bestand allerdings weiterhin dasselbe Problem, was diese erste Theorie ausschließt. Bei der Recherche zu dieser Frage stößt man in Forendiskussionen und GitHub Issues immer wieder auf die Aussage das GPT-2 Schwierigkeiten hat mit Zahlen umzugehen. Das ist insofern interessant als das die Antworten auf Frage 2 ausschließlich Zahlen sind. Es ist allerdings nicht möglich dies Anhand ein offizielles Aussagen oder Artikel zu bestätigen. Somit scheint dies eher ein Gerücht zu sein als ein Fakt. Außerdem würde das nicht erklären, warum Modell 1 so viel besser ist als Modell 2. Hätte GPT-2 Probleme mit Zahlen wäre zu erwarten das die Ergebnisse beider Modelle unterdurchschnittlich sind, was aber nicht der Fall ist. Weitere Tests zu dem Thema deuten allerdings daraufhin das Zahlen vielleicht tatsächlich etwas mit dem Problem zu tun haben könnten. Der erste Schritt im Versuch das Problem zu identifizieren war das Testen anderer Fragestellungen, um herauszufinden, ob es sich hierbei um einen Einzelfall handelt oder nicht. Dabei ergaben die Tests mit anderen Texten anfänglich das es sich dabei tatsächlich um einen Einzelfall handeln müsse, hier trat das Phänomen nicht auf. Bei erneuter Betrachtung des Kontextes für die Fragen, auch mit dem Hintergrundwissen über das Gerücht GPT-2 sei schlecht bei Zahlen, fiel auf das sich in dem Text tatsächlich überdurchschnittlich viele Zahlen befinden, egal ob in Ziffernschreibweise oder ausgeschrieben. Innerhalb der 3 Sätze des Kontexts befinden sich 5 Zahlen. Für einen Menschen wirkt das zwar unbedeutend, doch es ist möglich das dies weit genug vom

## 2. Funktionsweise

Durchschnitt der Texte mit denen GPT-2 entweder im Pre-Training oder im Finetuning trainiert wurde abweicht, um das Modell zu verwirren. Dabei sei bemerkt das dies, mit dem Wissen darüber wie GPT-2 funktioniert, nur bedingt eine Erklärung für das zu erkennende Verhalten ist, in der Theorie sollte es egal sein, ob es sich bei einem Wort nun um eine Zahl oder ein beliebiges anderes Wort handelt. Ohne weitere Tests ist es an dieser Stelle schwierig weitere Aussagen zu der Validität dieser Hypothese zu treffen.

Das gleiche Problem ist auch in Frage drei, „How many planets are in the solar system?“, zu finden. Die korrekte Antwort wäre „8“. Die Problematik dabei ist das am Anfang des Textes steht das es früher 9 Planeten im Sonnensystem gab. Erst im letzten Satz wird diese Aussage auf die heutige Anzahl an echten Planeten korrigiert. Versteht das Modell den Kontext nicht richtig ist es also gut möglich, dass die Antwort „9“ gegeben wird. Die Ergebnisse sind in Abbildung 12 zu sehen.

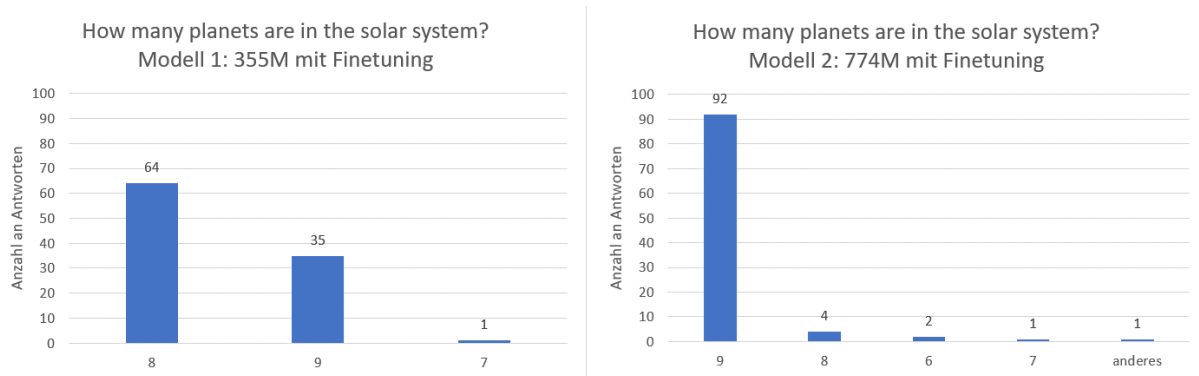


Abbildung 12: Vergleich der Ergebnisse zu der Frage: „How many planets are in the solar system?“

Es zeigt sich der Gleiche Trend wie bei schon bei Frage 2. Während das Kleinere Modell 1 in 64% der Fälle die Richtige Antwort gibt. Versagt Modell 2 erneut komplett und gibt in 92% der Fälle die falsche Antwort „9“. Nur in 4% der Fälle antwortet das Modell korrekt mit „8“.

Um die Zuvor aufgestellte Theorie zu unterstützen, dass dieses Verhalten tatsächlich etwas mit der Anzahl an Zahlen im Kontext zu tun haben könnte wurden zusätzlich noch weitere Tests durchgeführt, bei denen der Kontext jedes Mal besonders viele Zahlen beinhaltete. Beispielhaft sei hier noch einmal einer dieser Tests dargestellt.

Der Kontext für diesen Test ist Folgender:

The first season is based to a large extent on the first volume of the book series Leviathan Awakening and was broadcast on Syfy from December 14, 2015 to February 2, 2016, with the first episode being released in advance as a stream on November 23, 2015. The second season, based on the rest of the first and part of the subsequent band Caliban's War, aired in the United States from February 1 to April 19, 2017. In Germany, it will be broadcast on Netflix from September 8, 2017. In March 2017, the extension of the series by a third season became known, which was broadcast in the USA from April 11 to June 27, 2018 and whose plot is based on parts of the second and the entire third volume.

Abbildung 13: Kontext für den zusätzlichen Test zur Überprüfung der Theorie. Quelle: Auszug aus dem Wikipedia Eintrag zu „The Expanse“ [19] (eigene Übersetzung ins englische)



## 2. Funktionsweise

Die Frage zu diesem Text war „When was the third season aired?“. Die Korrekte Antwort ist: „April 11 to June 27, 2018“ oder eine Variation davon. Wie zu sehen ist sind in dem Text wieder eine Vielzahl an Zahlen, hauptsächlich, aber nicht ausschließlich, in Form von Daten. So soll überprüft werden, ob das Problem auch bei einem anderen Text auftritt, wenn dieser Ebenfalls viele Zahlen enthält. Die in Abbildung 14 zu sehenden Ergebnissen bestätigen die Theorie. Während das Modell 1 in 44% der Fälle korrekt antwortet. Wenn man Antwort zwei und drei noch mit dazu zählt, schließlich könnte man argumentieren das diese nicht wirklich falsch sind, Antwortet das Modell sogar in 73% der Fälle richtig. Modell 2 hingegen antwortet nur 20% der Zeit Korrekt, ist man auch hier wieder großzügig und zählt Antwort 3 dazu, sind es immerhin 39% der Fälle, in denen eine richtige Antwort gegeben wurde. In jedem Fall beantwortet das kleinere Modell 1 die Frage allerdings mehr als doppelt so häufig richtig wie Modell 2.

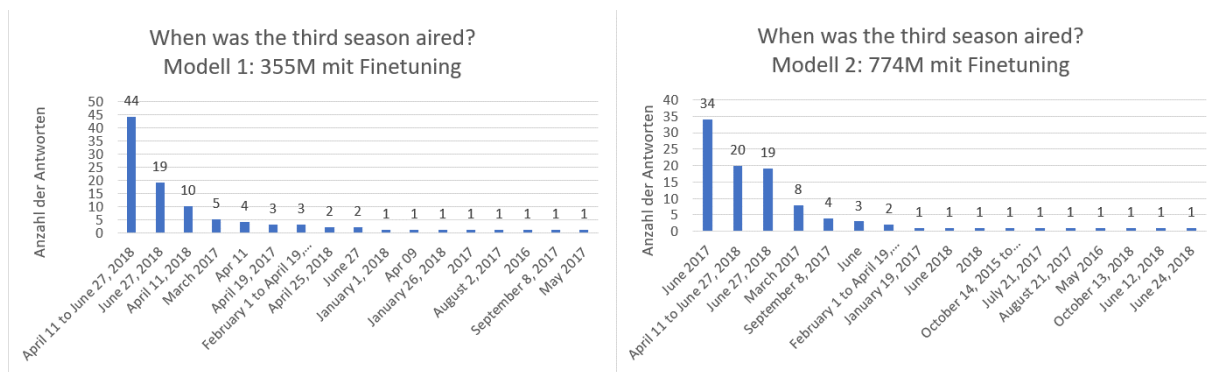


Abbildung 14: vergleich beider Modelle anhand eines anderen Textes

Da dieses Phänomen bei anderen Tests mit Texten, ohne besonders viele Zahlen im Text, nicht aufzutreten scheint, ist es wahrscheinlich das Texte mit vielen Zahlen schlechter vom größeren Modell 2 verstanden werden. Der Wahrscheinlichste Fall ist es, dass der Datensatz in irgendeiner Art und Weise Texte mit vielen Zahlen benachteiligt, wohlmöglich weil die Texte im Datensatz jeweils eher wenige Zahlen enthalten, wieso dies dann allerdings das kleinere Modell weniger betrifft als das größere Modell erklärt sich damit nicht. Es wäre auch denkbar das im Fall des Größeren Modells eine Überanpassung („overfitting“) auf den Datensatz auftritt, aufgrund der Trainingszeit bzw. der Trainingsschritte der Modelle scheint das allerdings eher unwahrscheinlich. Außerdem sollte eine Überanpassung auf beiden Modellen Gleicherweise gesehen, was nicht der Fall ist. Theoretisch wäre es auch möglich das der Datensatz zu klein ist. Da dieser allerdings über 78.000 Aufgaben enthält ist das auch fraglich.

Da GPT-2 wie alle anderen Maschine Learning Modell als Blackbox arbeitet, ist es schwierig den genauen Grund für dieses Verhalten zu finden. Es ist möglich, dass die Aufgestellte Hypothese stimmt und der SQUAD wirklich für das Verhalten verantwortlich ist. Genauso gut kann es aber auch sein, dass eine große Menge an Zahlen im Text nicht die Ursache des Problems ist, sondern die hier dokumentierten Probleme mit Zahlen nur ein Symptom eines anderen, zurzeit undefinierten, Problems darstellen.

Die bisher gezeigten Beispiele stellen eine sehr typische Aufgabe für Modelle wie GPT-2 dar. Dabei wurden bisher klassische Texte für das Training und die Eingaben verwendet. Da es in Kapiteln zur Anwendung von Texterzeugungsmodellen auch um abstraktere Beispiele gehen soll, ist es interessant sich an dieser Stelle auch anzugucken wie GPT-2 mit abstrakteren



Texten umgeht. Dazu soll dem Modell beigebracht werden einfache Gleichung zu lösen. Gleichung eignen sich gut für dieses Experiment, das sie wohl zu den Abstraktesten Texten gehören, außerdem ist es einfach die Ergebnisse zu bewerten und zu vergleichen. Bei einfachen Gleichungen mit einer Variablen gibt es eben meist nur ein korrektes Ergebnis.

Wie zuvor ist der erste und wichtigste Schritt das Finden eines passenden Datensatzes. Das relativ bekannte Unternehmen DeepMind, welches mittlerweile zu Google gehört, stellt auf ihrem GitHub Konto das Repository „mathematics\_dataset“ zur Verfügung, dieses ermöglicht es auf eine große Anzahl an mathematischen Aufgaben zuzugreifen. Diese sind sortiert nach Trainingsschwierigkeit und Aufgabenart. Um die Trainingszeit im Rahmen zu halten und die Bewertung der Ergebnisse einfach zu machen wurde für das Finetuning die Version „algebra\_linear\_1d“ aus dem Bereich „train-easy“ genutzt. Eine mögliche Aufgabe aus dem Datensatz wäre zum Beispiel diese: „solve  $0 = -4*a - 24 + 12$  for  $a$ “. Anders als beim SQUAD ist hier das gegebene Format bereits eine Textdatei. Allerdings wurde auch hier eine Veränderung an der Formatierung des Datensatzes, mittels eines kurzen Python Skripts (auf GitHub: `src/formatMath.py`) vorgenommen. Dabei wurde die zuvor für den SQUAD genutzte Formatierung für diese Aufgabe adaptiert. Der Token „<|startoftext|>“ sowie „<|endoftext|>“ wurden unverändert übernommen. Für die Identifizierung der einzelnen Teile der Aufgaben wurden folgende Token eingesetzt: „[EQUATION]“ für die eigentliche Formel, „[SOLVEFOR]“ für die Variable, nach der die Gleichung gelöst werden soll und „[SOLUTION]“ für die Lösung der Gleichung. Ein vollständiger Aufgabenblock kann beispielhaft in Abbildung 15 gesehen werden.

```
<|startoftext|>
[EQUATION]:  $0 = -4*a - 24 + 12$ 
[SOLVEFOR]:  $a$ 
[SOLUTION]:  $-3$ 
<|endoftext|>
```

*Abbildung 15: Beispielaufgabe für das Finetuning zum Lösen von Gleichungen*

Beim Betrachten der Beispielaufgabe in Abbildung 15 könnte die Frage aufkommen wozu die „[SOLVEFOR]“ Zeile dient. Schließlich ist diese bei Gleichungen mit einer einzigen Variablen überflüssig. Das Hauptziel der „[SOLVEFOR]“ Zeile ist es dem Modell das Verstehen der Aufgabe so einfach wie möglich zu machen. Es wäre mit Sicherheit auch möglich ein Funktionierendes Modell, ohne diese Zeile zu erstellen, vermutlich würde dies allerdings deutlich mehr Trainingsschritte und damit mehr Trainingszeit benötigen. Des Weiteren wurde die Anzahl an Gleichungen im Trainingsdatensatz im Vergleich zum Originalen Datensatz reduziert, um die Trainingszeit zu verringern. Der ursprüngliche Datensatz umfasst über 700.000 verschiedene Gleichungen. Der reduzierte Datensatz, der für die im folgenden vorgestellten Modelle genutzt wurde, beinhaltet nur noch 200.000 Gleichungen. Ähnlich wie bei den vorherigen Beispielen wurden Zwei verschiedene Modelle trainiert die nun vorgestellt und verglichen werden.

1. Modell: Finetuning anhand der 355M Version von GPT-2 mit 4000 Trainingsschritten. Der durchschnittlicher Loss betrug am Ende 0.49. Das Training fand mit Hilfe einer Nvidia Tesla P100 16GB statt und dauerte 61 Minuten.

2. Modell: Diesem Modell liegt die 774M Variante des GPT-2 Modells zugrunde. Wie beim ersten Modell wurden hier 4000 Trainingsschritte durchlaufen. So konnte ein durchschnittlicher Loss von 0.48 erreicht werden. Wie auch beim Ersten Modell wurde für das Training ein System mit einer Nvidia Tesla P100 16GB verwendet, welches 120 Minuten für die Aufgabe brauchte.

Um die Leistungsfähigkeit der Modelle zu überprüfen und vergleichen zu können wurden beiden Modellen sieben verschiedene Gleichungen zum Lösen gegeben. Dabei sollte das Modell die Aufgaben jeweils 100-mal beantworten, um so eine Verteilung über die Antworten zu generieren und die Wahrscheinlichkeiten der verschiedenen Antworten vergleichen zu können. Da es sich bei den Antworten um einzelne Zahlen handelt ist kein Kreatives Schreiben in den Antworten gewünscht. Deshalb wurde für die Tests der „temperature“ Wert auf 0.2 reduziert worden. Die Sieben Gleichungen und ihre erwarteten Lösungen sind in Tabelle 2 zu sehen.

Nr.	Gleichung	Lösen nach	erwartete Antwort
1	$10 = 5 * x$	x	2
2	$-40 = 15 * x + 5 * x$	x	-2
3	$135 = 6 * x + -4 * x + 15$	x	60
4	$0 = 31 * x - 336 + 119$	x	7
5	$x = 100 - 95$	x	5
6	$x = 100 + -95$	x	5
7	$10 = 2 * x + 4$	x	3

*Tabelle 2: Aufgaben und ihrer Lösungen für die Tests zum Lösen von Gleichungen*

Bei Betrachtung der verschiedenen Gleichungen können einige Besonderheiten auffallen:

1. Die Lösungen der Aufgaben sind immer Zahlen im Bereich -10 bis 10 mit der Ausnahme von Gleichung drei. Dies ist wichtig das die Gleichungen im Datensatz nahezu ausschließlich Lösungen in diesem Zahlenbereich haben. Das bedeutet die Modelle haben nicht gelernt höhere Zahlen als Antwort zu geben. Aufgabe drei dient als Test, ob die Modelle trotzdem in der Lage sind eine entsprechende Gleichung zu lösen, eine Korrekte Antwort ist hier allerdings nicht zu erwarten.

2. Alle Gleichung sollen nach der Variablen x aufgelöst werden. Im Datensatz werden die Verschiedensten Buchstaben als Variablenname genutzt. Der Grund dafür, dass das hier nicht der Fall ist, besteht in der Vergleichbarkeit der Ergebnisse. Es ist möglich das bestimmte Buchstaben, innerhalb des Datensatzes, besonders häufig oder besonders selten als Variablennamen genutzt werden. Durch die Nutzung der gleichen Variablenbezeichnung in allen Gleichungen wird sichergestellt das keine Aufgabe einen Vor- oder Nachteil gegenüber einer anderen hat.

3. Die Aufgaben 5 und 6 unterscheiden sich lediglich in der Schreibweise der Gleichungen. Während in Gleichung 5 direkt eine Subtraktion zweier Zahlen gefordert ist, wird in Gleichung 6 eine positive mit einer negativen Zahl addiert. Grundsätzlich sind beide Aufgaben damit gleich und es kommen beide Schreibweisen im Datensatz vor.

## 2. Funktionsweise

Nichtsdestotrotz wird es Interessant zu sehen, ob sich das Modell von den unterschiedlichen Schreibweisen beeinflussen lassen.

In Abbildung 16 sind die Ergebnisse der Aufgaben von beiden Modellen zu sehen. Dabei ist jeweils dargestellt wie viel Prozent der Antworten korrekt waren.

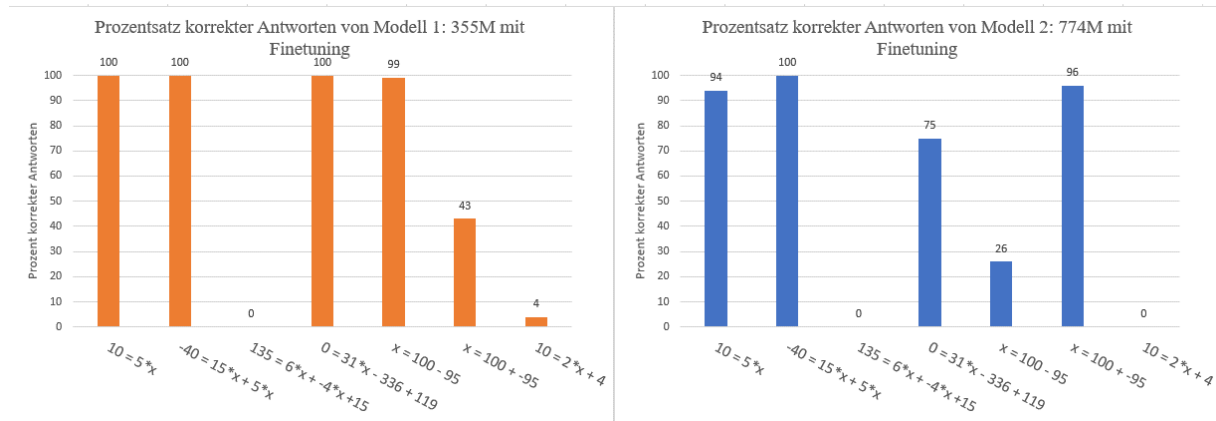


Abbildung 16: Vergleich der Lösungen für die Gleichungsaufgaben

Bei der Auswertung der Antworten werden Aufgaben nur dann als richtig beantwortet gezählt, wenn die korrekte Antwort mindestens den Schwellwert von 50% überschreitet. Das bedeutet, nur wenn mindestens 51 der 100 Antworten für eine Aufgabe die korrekte Lösung enthalten, zählt die Aufgabe als korrekt beantwortet. So soll verhindert werden, dass Aufgaben, bei denen sich ein Modell sehr unsicher ist, als korrekt klassifiziert werden. Theoretisch aber wären auch andere Bewertungsmethoden denkbar.

Unter den gerade beschriebenen Voraussetzungen beantworten beide Modelle jeweils vier der sieben Aufgaben korrekt. Unterberücksichtigung des insgesamt eher kurzen Trainings sowie der Abstraktheit dieser Aufgabe im Vergleich zu eher typischen Aufgaben für GPT-2, ist das ein gutes Ergebnis, insbesondere wenn man bedenkt, dass bei Aufgabe 3 auch keine richtige Antwort erwartet wurde.

Bei Aufgabe drei, deren Lösung weit außerhalb der Lösungen des Datensatzes liegt, wurde wie erwartet von keinem der Modelle richtig beantwortet. Stattdessen gab das Modell 1 82-mal die Antwort „9“ und 18-mal die Antwort „7“, während Modell 2 in 100% der Fälle mit „5“ antwortet. Auch wenn die Antworten falsch sind, zeigt dies allerdings gut ein Grundaspekt, nicht nur von GPT-2, sondern aller Machine Learning Modelle: die Ausgabe und damit ihre Möglichkeit Aufgaben zu lösen beschränkt sich immer auf das mithilfe von Datensätzen trainierte Wissen. In diesem Fall sind im Datensatz keine Lösungen größer 10 vorhanden, also kommt für die Modelle eine Lösung wie „60“ gar nicht in Frage.

Auch der Spezialfall von Aufgabe fünf und sechs zeigt eine Besonderheit. Beide Modelle lassen sich tatsächlich von den unterschiedlichen Schreibweisen verwirren und beantworten jeweils nur eine der beiden Aufgaben richtig. Interessanter Weise unterscheiden sich die Modelle allerdings darin, welche der Aufgaben richtig beantwortet wird. Während Modell 1 Aufgabe fünf richtig beantwortet, aber nicht Aufgabe sechs, ist es bei Modell 2 genau andersherum. Interessant ist dabei aber auch, dass dies jeweils die Antworten sind, bei denen sich die Modelle am unsichersten sind. Modell 1 gibt bei Aufgabe sechs in 43% der Fälle die richtige Antwort und schafft es damit nur knapp nicht den Schwellwert zu überschreiten, um als korrekt beantwortet zu gelten. Im Vergleich ist Modell 2 bei Aufgabe fünf mit 26%

richtigen Antworten zwar nicht so knapp vor dem Schwellwert gibt aber dennoch signifikant oft die richtige Antwort. Bei den anderen Aufgaben sind sich die Modelle meist sehr sicher, ob die Antwort nun richtig ist oder nicht. Dabei scheint Modell 2 bei dieser Aufgabe aber vor allem ein Problem mit den Vorzeichen zu haben, denn während des Modells in nur 26% der Fälle korrekt mit „5“ antwortet, gibt das Modell in 69% der Fälle die Antwort „-5“ und macht somit lediglich einen Vorzeichenfehler.

Wundersam ist Außerdem das beide Modelle bei Aufgabe sieben Komplet versagen. Anders als die Aufgaben drei, fünf und sechs war diese Aufgabe nicht darauf ausgelegt einen Fehler zu provozieren, im Gegenteil, nach den schwierigeren Aufgaben fünf und sechs sollte diese Aufgabe lediglich noch einmal einen einfach zu lösenden Abschluss darstellen. Warum die Modelle hier Probleme haben lässt sich ohne weiteres nicht sagen.

Alles in allem kann man sagen das beide Modelle die Aufgabenstellung „Gleichungen Lösen“ relativ gut bewältigen konnten. Das Problem aus den ersten Beispielen scheint sich hier nicht wiederzufinden, was weiter darauf hinweist, dass das Problem dort im Datensatz lag.

Allerdings ist hier Auch keine Verbesserung beim Wechsel auf das Größere Modell festzustellen. Vermutlich wären mehr Tests und längeres Training nötig, um die Vorteile des Größeren Netzes zu bemerken. Verschiedenen Artikel und Forendiskussionen deuten darauf hin, dass die größeren Modelle insbesondere im Bereich „Kreatives Schreiben“ auch schon bei kleineren Datensätzen deutlich Vorteile gegenüber den kleineren Modellen bieten. Da sich die Ausgaben in diesem Aufgabenbereich allerdings nur schwer objektiv bewerten lassen und damit auch nur schlecht zu vergleichen sind, wurden in diesem Aufgabenbereich keine weiteren Tests durchgeführt.

## 2. Funktionsweise

- Beispiele nicht zwingend repräsentativ für die Leistung von GPT-2. Stellt nur meine Erfahrungen dar
- Nicht vergessen auch die ganze sache mit dem Vorbereiten der Daten zu erklären
- Erzeugung neuer Datensätze mithilfe von GPT-2

### 2.2.4 Schlussfolgerungen aus den Programmbeispielen

Für die Ausführung wurden insgesamt drei Verschiedene Hardwarelösungen genutzt:

1. Locale Hardware mit einem Intel Core i7-8700K, einer Nvidia 1080 Ti mit 11GB GPU-Speicher und 16GB Arbeitsspeicher.
2. Google Collaboratory mit 2 Threads eines Intel Xeon Prozessors, einer Nvidia T4 mit 16GB GPU-Speicher und etwa 12GB Arbeitsspeicher
3. Google Collaboratory Pro mit 2 Threads eines Intel Xeon Prozessors, einer Nvidia P100 mit 16GB GPU-Speicher und etwa 24GB Arbeitsspeicher

- Erklären wieviele steps ich gewählt habe. 4000 gute grundlage. Vielleicht auch mit einigen loss zahlen aus verschiedenen runs mit unterschiedlichen steps belegen
- Auch generell die ganze hardware diskusion kann hierher (also die zu den speziellen beispielen)
- Die beispiele die man immer so im internet sieht sind vermutlich cherry picked
- Vielleicht ist „auswertung“ nicht die richtige überschrift
- Bezug auf verstehendes lesen: auswahl eines anderen beispiels bei dem das Problem nicht auftritt wäre einfach gewesen. Doch zeigt gut die Probleme eines Blackbox systems
- Beispiele nicht zwingend repräsentativ für die Leistung von GPT-2. Stellt nur meine Erfahrungen dar. Insbesondere auch nochmal darauf eingehen das meine erfahrungen hier natürlich nicht das gerücht bestätigen das GPT-2 schlecht mit Zahlen umgehen kann.
- Das eigentliche trainieren usw. ist einfach und wenig code. Aber wegen blackbox sind probleme insbesondere als anfänger schwer bis garnicht zu lösen. Vermutlich hilft da nur erfahrung mit gpt
- Beispiele zeigen gut das die verwendung kein einfaches Plug and Play ist.

## 2. Funktionsweise

- <https://github.com/openai/gpt-2/issues/265> antwort von UtilityHotbar um unter neuem tensorflow zu laufen. Man könnte irgendwas schreiben damit das zu sehende zeproduzierbar ist oder so.

## 2. Funktionsweise

- [https://www.youtube.com/watch?v=60NGxX7m1Mw&t=2403s&ab\\_channel=LinusTechTips](https://www.youtube.com/watch?v=60NGxX7m1Mw&t=2403s&ab_channel=LinusTechTips)  
(ab 40:00) probleme zu Github Copilot



## Abbildungsverzeichnis

Abbildung 1: Architektur des Transformer Modells [12] .....	8
Abbildung 2: Beispielhafte Trainings- und Test-fehlerrate eines neuronalen Netzes mit 20 Schichten im Vergleich mit 56 Schichten auf dem CIFAR-10 Datensatz [13].....	9
Abbildung 3: Scaled Dot-Product Attention [12].....	10
Abbildung 4: Multi-Head Attention Schicht [12] .....	11
Abbildung 5: Aufbau des Transformer Decoders im Fall von GPT [15] .....	13
Abbildung 6: Inputs für das Fine-Tuning auf vier verschiedene Aufgaben [15] .....	14
Abbildung 7: Code für das Finetuning von GPT-2 mithilfe von gpt-2-simple .....	17
Abbildung 8: Beispielhafte Formatierung einer einzelnen Aufgabe .....	19
Abbildung 9: Kontext für die Fragen des Verstehenden Lesen. Quelle: [18] .....	20
Abbildung 10: Vergleich der Antworten beider Modelle auf die Frage: „When was pluto discovered?“ .....	21
Abbildung 11: Vergleich der Antworten beider Modelle auf die Frage: „When did pluto stop being a planet?“ .....	22
Abbildung 12: Vergleich der Ergebnisse zu der Frage: „How many planets are in the solar system?“ .....	23
Abbildung 13: Kontext für den zusätzlichen Test zur Überprüfung der Theorie. Quelle: Auszug aus dem Wikipedia Eintrag zu „The Expanse“ [19] (eigene Übersetzung ins englische) .....	23
Abbildung 14: vergleich beider Modelle anhand eines anderen Textes.....	24
Abbildung 15: Beispielaufgabe für das Finetuning zum Lösen von Gleichungen .....	25

## Tabellenverzeichnis

Tabelle 1: 30 von GPT-2 beantwortete Fragen, sortiert nach der von GPT-2 berechneten Wahrscheinlichkeit für die gegebenen Antworten. Quelle: [16] .....	15
Tabelle 2: Aufgaben und ihrer Lösungen für die Tests zum Lösen von Gleichungen .....	26

## Literaturverzeichnis

- [1] D. Gershgorn, „Popular Science,“ 12 12 2015. [Online]. Available: <https://www.popsci.com/new-openai-artificial-intelligence-group-formed-by-elon-musk-peter-thiel-and-more/>. [Zugriff am 28 06 2021].
- [2] J. Vincent, „The Verge,“ 21 02 2018. [Online]. Available: <https://www.theverge.com/2018/2/21/17036214/elon-musk-openai-ai-safety-leaves-board>. [Zugriff am 28 06 2021].
- [3] „OpenAI,“ [Online]. Available: <https://openai.com/about/>. [Zugriff am 28 06 2021].
- [4] „OpenAI,“ [Online]. Available: <https://openai.com/chapter/>. [Zugriff am 28 06 2021].
- [5] G. Brockman und J. Schulman, „OpenAI,“ 27 04 2016. [Online]. Available: <https://openai.com/blog/openai-gym-beta/>. [Zugriff am 28 06 2021].
- [6] „openAI,“ [Online]. Available: <https://openai.com/projects/five/>. [Zugriff am 28 06 2021].
- [7] A. Radford, „OpenAI,“ 11 06 2018. [Online]. Available: <https://openai.com/blog/language-unsupervised/>. [Zugriff am 30 06 2021].
- [8] A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage und I. Sutskever, „Better Language Models and Their Implications,“ 14 02 2019. [Online]. Available: <https://openai.com/blog/better-language-models/>. [Zugriff am 30 06 2021].
- [9] I. Solaiman, J. Clark und M. Brundage, „GPT-2: 1.5B Release,“ 05 11 2019. [Online]. Available: <https://openai.com/blog/gpt-2-1-5b-release/>. [Zugriff am 30 06 2021].
- [10] T. B. Brown, . B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever und D. Amodei, „Language Models are Few-Shot Learners,“ 26 05 2020. [Online]. Available: <https://arxiv.org/abs/2005.14165>. [Zugriff am 01 07 2021].
- [11] GPT-3, „The Guardian,“ [Online]. Available: <https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3>. [Zugriff am 31 05 2021].
- [12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser und I. Polosukhin, „Attention Is All You Need,“ 12 06 2017. [Online]. Available: <https://arxiv.org/abs/1706.03762>. [Zugriff am 05 07 2021].
- [13] K. He, X. Zhang, S. Ren und J. Sun, „Deep Residual Learning for Image Recognition,“ 10 12 2015. [Online]. Available: <https://arxiv.org/pdf/1512.03385.pdf>. [Zugriff am 10 07 2021].
- [14] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, Ł. Kaiser und N. Shazeer, „Generating wikipedia by summarizing long sequences,“ 30 01 2018. [Online]. Available: <https://arxiv.org/pdf/1801.10198.pdf>. [Zugriff am 12 07 2021].

- [15] A. Radford, K. Narasimhan, T. Salimans und I. Sutskever, „Improving Language Understanding by Generative Pre-Training,“ 11 06 2018. [Online]. Available: [https://cdn.openai.com/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf). [Zugriff am 12 07 2021].
- [16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei und I. Sutskever, „Language Models are Unsupervised Multitask Learners,“ 14 02 2019. [Online]. Available: [https://cdn.openai.com/better-language-models/language\\_models\\_are\\_unsupervised\\_multitask\\_learners.pdf](https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf). [Zugriff am 13 07 2021].
- [17] C. Reams, „Question Answering with GPT-2,“ 25 01 2020. [Online]. Available: <https://medium.com/analytics-vidhya/an-ai-that-does-your-homework-e5fa40c43d17>. [Zugriff am 26 07 2021].
- [18] R. R. Britt, „Solar System Planets: Order of the 8 (or 9) Planets,“ 01 05 2021. [Online]. Available: <https://www.space.com/16080-solar-system-planets.html>. [Zugriff am 26 07 2021].
- [19] „The Expanse (Fernsehserie),“ [Online]. Available: [https://de.wikipedia.org/wiki/The\\_Expanse\\_\(Fernsehserie\)](https://de.wikipedia.org/wiki/The_Expanse_(Fernsehserie)). [Zugriff am 27 07 2021].

#### LINKS

<https://openai.com/blog/gpt-3-apps/> (Anwendungen von gpt-3)