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:

30.06.2021

Hochschule Stralsund

Fakultät für Wirtschaft

Inhaltsverzeichnis

1. Einleitung

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

2. Funktionsweise

2.1. Theoretische Erklärung

- supervised und unsuperviesd learning

- rückgekoppelte neuronale Netze

- Sequence to sequence (oder seq2seq)

- LSTM als rückgekoppeltes neuronales Netz und möglichkeit für Texerzeugungs system erklären. (quasi der Vorgänger von Transformer Modellen)

- attantion mechanismuss erklären. (vielleicht direkt am LSTM?)

- Reheinfolge 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 das Künstliche Intelligenz der gesamten Menschheit zugutekommt.“ [3] (eigene Übersetzung). Um dieses Ziel zu erreichen hat OpenAI einige Regeln aufgestellt, an die sie 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 Inverstoren 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 das 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 ein 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 das in diesem Jahr das derzeitige Weltmeister Team „OG“ in zwei aufeinander folgenden 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 diese 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 ersten Versuch von OpenAI dar die zu dem Zeitpunkt neuen Transformer Modelle mit der Methodik des „Semi-supervised training“ zu verbinden (eine genauere Erklärung dieser beiden Konzepte findet sich im Kapitel 2.2.2 da diese den Rahmen der Einleitung sprengen würde). Die Grundsätzliche Idee ist dabei ist es das Modell mithilfe eines sehr großen Datensatzes unüberwacht zu Tainieren 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 Letzen 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 Datensatzes 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 ihrer 2. Firmenregel wird nur eine deutlich Kleine Version des Modells veröffentlicht. Sie befürchten missbrauch des Models 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 andere Texterzeugungsmodele mit denen OpenAI sich vergleicht. [8]

GPT-3: Version 3 ist die neuste 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 möglich über die OpenAI API möglich die Nutzung dieser API ist allerdings aktuell mit einer Warteliste beschränkt und zudem kostenpflichtig. Ob sich dies mit den selbst gestellten Firmenzielen von OpenAI, sowie dessen ursprünglicher Gemeinnützigkeit vereinbaren lässt ist diskutable. Dies soll hier allerdings vorerst keine weitere Rolle spielen. Im Vergleich zum Vorgänger sind die Ergebnisse von GPT-3 nochmal besser, so sind z.B. Fließtexte die von GPT-3 erzeugt werden nicht auf den ersten Blick 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“ Scenario. 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 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 das die Leistung von GPT-2 hinter der von GPT-3 zurückfällt.

2.2.2 Funktionsweise von GPT

Die Grundlage von GPT bilden zwei Verschiedene Ideen. Auf der einen Seite steht das Transformer Modell welche den technischen Grundstein für das Eigentliche KI-Modell bietet und auf der anderen Seite das „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 sein folgendes gesagt: Sowie „Semi-supervised training“ als auch Transformer Modelle bieten alleine ausreichend Material für eigenständige Arbeiten über die Themen. Das Ziel ist es dementsprechend nicht alle Aspekte der beiden Themen bis ins kleinste 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 vorgestellte wurde [12]. Ähnlich wie ein Long-Short-Term-Memory Modell, besteht das Transformer Model 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 sie 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.

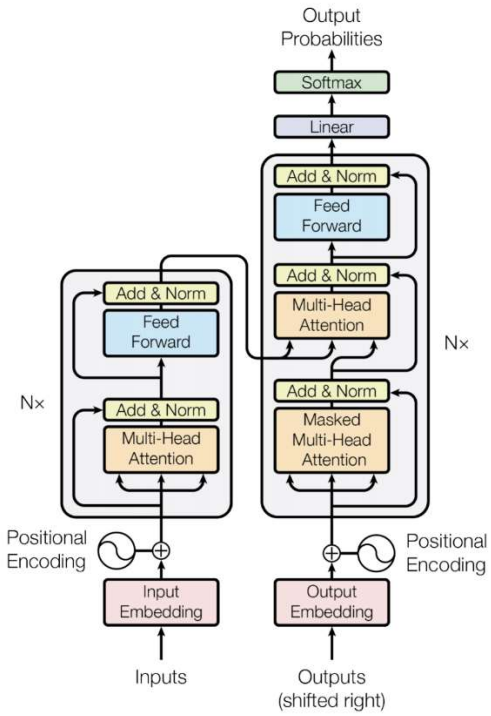

Abbildung 1: Architektur des Transformer Modells [12]

Sowie der Encoder als auch der Decoder bestehen aus aufeinander gestapelten Schichten, dies wird durch „Nx“ and 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 das Netzwerkes die es erlaubt bestimmte schichten des Neuronalen Netzes zu überspringen. Das Prinzip des „Add & Norm“ Blocks kann mithilfe dieser Formal 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 Risidualen 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.

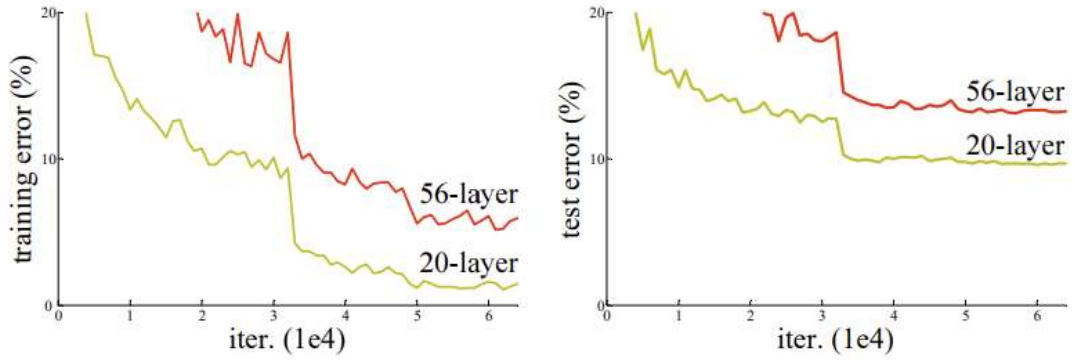

Abbildung 2: Beispielhafte Trainings- und Test fehlerrate eines Neuronalen Netzes mit 20 schichte im Vergleich mit 56 Schichten auf dem CIFAR-10 Datensatz [13]

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 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:

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

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

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 „embeding“ und dem „Positional Encoding“ ist es allerdings wichtig. Den Output um einen Position nach rechts zu verschieben. Würde keine Verschiebung stattfinden würde das Neuronal 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 vorauszusagen.

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 allerding 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 das diese den Komplexesten Teil des Multi-Head Attention Moduls ausmacht. Die Funktionsweise ist anhand von Abbildung 3 Beschrieben.

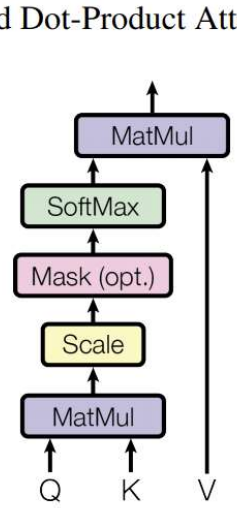

Abbildung 3: Scaled Dot-Product Attention [12]

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

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 das mit $softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$ 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 werden. 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.

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, ist es jetzt möglich die Vorgehensweise des gesamten Multi-Head Attention Moduls zu verstehen, diese wird in Abbildung 4 veranschaulicht.

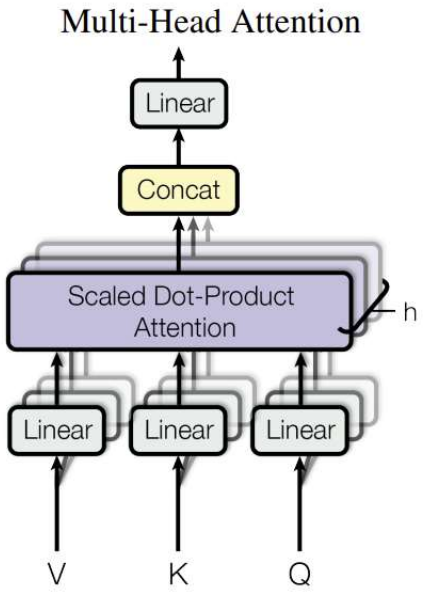

Abbildung 4: Multi-Head Attention Schicht

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 projizierte, 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 Gewichts-Matrix multipliziert. So ergeben sich die Finalen Werte des Multi-Head Attention Moduls. Zur Veranschaulichung dient auch hier wieder eine Formel:
$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^o$ [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 Output Sequenzen zu einer einzigen Sequenz vereint werden. Sein die Inputsequenz $m = (m^1, \ldots, m^n)$ und Outputsequenz $y = (y^1, \ldots, y^x)$ gegeben erfolgt die Zusammenführung der Sequenzen in eine Sequenz w wie folgt: $(w^1, \ldots, w^{n+x+1}) = (m^1, \ldots, m^n, \delta, y^1, \ldots, y^x)$. δ ist dabei ein Spezielles element zur seperierung der beiden uhrsprünglichen Sequenzen. Diese spezielle Variante des Transformers soll durch eine starke Reduzierung der Modell Parameter 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 das die Typischer Encoder-Decoder Struktur nicht vorhanden ist. GPT nutzt dabei ein 12-schichtiges Design.

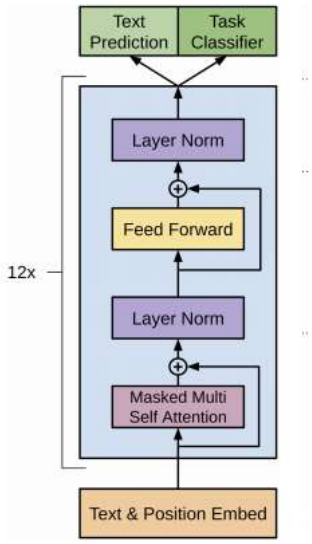

Abbildung 5: Aufbau des Transfomer Decoders im Fall von GPT [15]

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 Andew M. Dai und Quoc V. Le beschrieben, die Arbeit von OpenAI baut auf darauf auf. Dazu wird bei GPT zuerst der Prozess des „unsupervised pre-training“ angewandt. Dabei wird das Model 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 das 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 nächte 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: $\mathrm{L}_3(\mathrm{C}) = \mathrm{L}_2(\mathrm{C}) + \lambda * \mathrm{L}_1(\mathrm{C})$. Dabei ist L_1 die Wahrscheinlichkeits-Funktion des „Unsupervised pre-training“ und L_2 die Wahrscheinlichkeits-Funktion vom „Supervised fine-tuning“ mit λ 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.

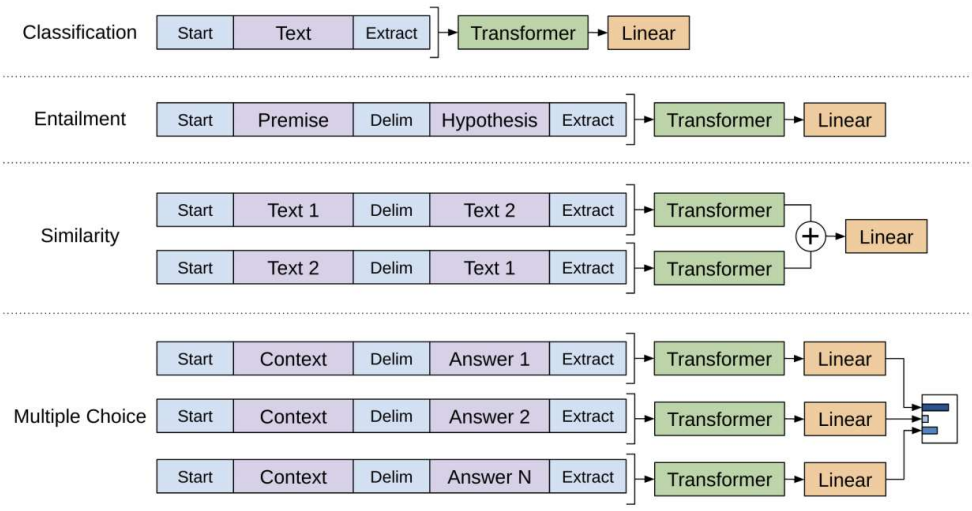

Abbildung 6:Inputs für das Fine-Tuning auf vier verschiedene Aufgaben [15]

Für strukturell einfache Fälle, wie zum Beispiel der Text Klassifizierung kann das Modell problemlos und ohne weiter 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 gesehen 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 das 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 Input 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 das 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 Fine-Tuning dann aber nur noch ein sehr viel kleinerer Datensatz nötig. So werden die Vorteile beider Ansätze verbunden.

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%

Abbildung 7: 30 von GPT-2 beantwortete Fragen, sortiert nach der von GPT-2 berechneten Wahrscheinlichkeit für die gegebenen Antworten [16]

In Abbildung 7 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 Abbildung 7 einordnen zu können ist es wichtig zu wissen das keine der Frage direkt im Datensatz vorkommt. Zur Beantwortung ist es also nötig die entsprechenden Texte sowie auch die Fragen zu verstehen, um so eine Verbindung zwischen beidem herstellen zu können und die Frage 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.

Abbildung 7 macht deutlich das die Ansätze von GPT funktionieren und zu durchaus guten Ergebnissen führen können.

- gpt nutzt spezielle form des Transformers, Transformer-Decoder, https://arxiv.org/pdf/1801.10198.pdf - T-D

https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf unsupervised pre-Training

- Erklärung der verschiedenen Konzepte.

- aufteilung der drei versionen um unterschiede darzustellen

- vielleicht auch Grafiken

- Vergleich der Leistung der modelle anhand der tests von OpenAI? (vielleicht auch in anderem kapitel?)

Abbildungsverzeichnis

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/charter/. [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].

LINKS

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