Transformer Decoder Architektur

1st Stefan Maier
Function and Algorithm
ZF Lifetec
Alfdorf, Deutschland
Stefan.Maier1@zf-lifetec.com

***Abstract*—Große Sprache Modelle (engl. Large Language Models) wie Chat GPT basieren auf der Transformer Architektur neuronaler Netze. Die Transformer Architektur besteht dabei aus einem Encoder- sowie Decoder-Element. In der Wissenschaft und Forschung existieren auch Architekturen welche nur einen Teil eines Transformers zur Lösung eines Problems verwenden. In dieser Arbeit wird genauer auf Transformer Architekturen eingegange, welche lediglich aus dem Decoder Element bestehen. Eine weit Verbreitete Decoder Architektur ist die von Chat GPT verwendete GPT (kurz: Generative Pretrained Transformer) Architektur. Es wird auf die Architektur und die weiter Entwicklung der Architektur eingegangen.**

***Index Terms*—component, formatting, style, styling, insert.**

I. Einführung

Die Verarbeitung von natürlicher Sprache (engl. Natural Language Processing. kurz NLP) ist ein wichtiger und großer Bestandteil der künstlichen Intelligenz Forschung. Die Forschungsbereiche decken dabei ein breites Spektrum an Aufgaben ab, wie z.B. Beantwortung von Fragen, Semantische Ähnlichkeit, Text Generierung, Dokumenten Klassifikation o.ä. Die ersten Fortschritte wurde im NLP Bereich bereits durch statistische Hidden Markov Modelle erzielt. Die Leistung solcher statistischen Modelle ist jedoch begrenzt und konnte die Komplexität der natürlichen Sprache nicht geeignet abbilden. Mit den Entwicklungen im Bereich des Deep-Learnings gelangen entscheidende Schritte in der NLP Forschung. Die Transformer Architektur, und das damit verbundene vortrainieren großer Sprachmodelle wie BERT, GPT,T5 oder RoBERTa haben große Fortschritte gebracht, die sich nicht nur auf die NLP Forschung beschränken sondern in der allgemeinen Gesellschaft und Wirtschaft Einzug halten.

A. Ziel der Arbeit

Das Ziel der Arbeit ist es die Decoder Architektur welche Häufig von generativen Sprachmodellen verwendet wird genauer zu beleuchten. Dabei wird die Decoder-only Architektur anhand des Generative Pretrained Transformers erklärt. Neben dem Generative Pretrained Transformer existieren ebenfalls weitere Modelle die kurz erläutert werden. Weiterhin soll die Arbeit die Anwendungsgebiete sowie Potenziale und Grenzen der Decoder Architektur aufzeigen. Im Ausblick wird auf aktuelle Forschungsfelder im Bereich der Decoder Architekturen eingegangen.

B. Aufbau und Struktur der Arbeit

Die Arbeit ist wie folgt strukturiert: In Kapitel 2 wird zunächst auf die Grundlagen in Form der Decoder Architektur als Teil der Transformer Architektur eingegangen. Dazu wird die historische Entwicklung, der Aufbau und der Trainingsprozess der Decoder Architektur erläutert und diese vom Encoder Teil der Transformer Architektur abgegrenzt. In Kapitel 3 wird auf unterschiedliche mögliche Anwendungsgebiete aus der Forschung und Entwicklung eingegangen. Aus diesen Erkenntnissen werden in Kapitel 4 die Potenziale und damit einhergehenden Grenzen solcher großen Sprachmodelle eingegangen. Kapitel 5 fasst die Ergebnisse zusammen und gibt einen Ausblick auf die aktuellen Forschungsbereiche.

II. Decoder Architektur

Im folgenden Abschnitt wird die Decoder Architektur am Beispiel des Generative Pretrained Transformers (GPT) erläutert. Dabei wird auf die historische Entwicklung, den Aufbau, den Trainingsprozess und die Abgrenzung zur Encoder Architektur eingegangen.

A. Historische Entwicklung der GPT-Decoder Architektur

Transformer bilden das Grundgerüst moderne großer Sprachmodelle. Diese wurden 2017 in [1] beschrieben. 2018 wurde die erste Version von GPT veröffentlicht, welche auf dem Decoder Part der Transformer Architektur aufbaut. Auf Basis der GPT Modell entstanden weitere Decoder-only Modelle welche die folgende Figure 1 gut veranschaulicht. Die Figure 1 zeigt die Entwicklung unterschiedlicher Transformer Modell bis zum letzten Jahr. Der Zweig der Decoder Only Architektur stellt den Entwicklungszweig reiner Decoder Architekturen dar . Mit GPT wurde das erste rein auf der Decoder Struktur basierende Neuronale Netzwerk entworfen. Daraus entsprangen viele unterschiedliche Decoder-Only Modelle von Google, Meta oder anderen Forschungseinrichtungen. Die aktuellsten Versionen von Llama, GPT-4 oder Bard stellen dabei bis heute die neusten Entwicklungen dar.

B. Aufbau der GPT-Decoder Architektur

Wie im vorherigen Abschnit beschrieben legt das GPT Modell den Grundstein für die Decoder-only Architektur. Dies wurde durch das Paper [3] beschrieben und basiert dabei auf den Ideen der in [1] beschrieben Transformer Architekturen. Die folgende Figure 2 illustriert dabei den Aufbau

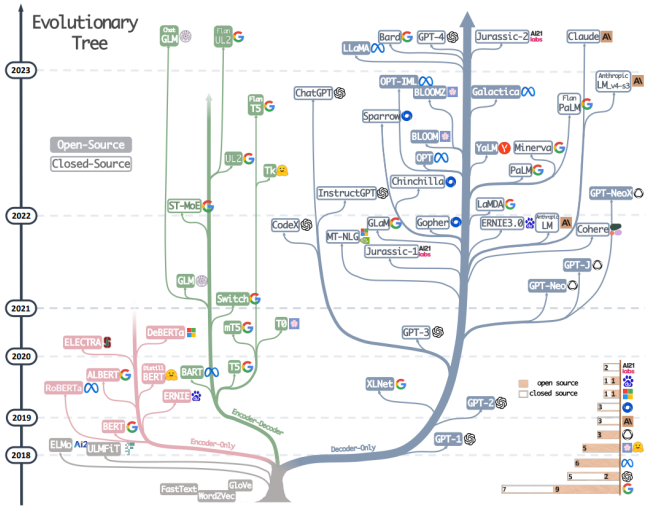

Fig. 1. Historische Entwicklung von Transformer Modellen aus [2]

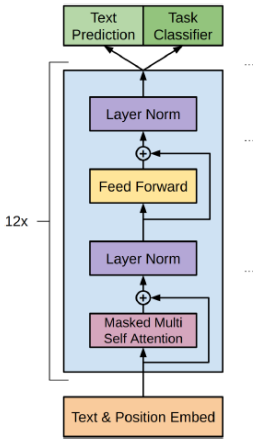

Fig. 2. Decoder Architektur nach [3]

einer Decoder Schicht des GPT Modells. Die Figure 2 zeigt dabei die Elemente einer Decoder Schicht. Die Elemente werden logisch zu einer Decoder Schicht (blau) zusammengefasst. Eine Decoder Schicht besteht dabei aus einem Masked Multi-Self Attention Layer, Feed Forward Netzwerken, sowie Layer Normalisierungs Schicht um Residuen Verbindungen der vorherigen Schicht hinzuzufügen. Die Residuen lösen das Problem der verschwindend Gradienten beheben [4]. Als Eingabe für die Decoder Schicht dienen dabei immer entsprechende Embeddings mit Positionskodierung, welche die Position der Embeddings in einem Satz bestimmen [4]. Der Kernbestandteil der Decoder Schicht ist dabei das Masked Multi-Self Attention Layer. Diese Schicht ist eine leichte Abwandlung des Mult Self Attention Layers, welches auch in Encoder Elementen eines Transformers verwendet wird. Nachfolgende Figure 3 zeigt den Aufbau eines Multi-Self Attention Layers. Als Eingabe werden die Embeddings der vorherigen Schicht verwendet und in Query, Key und Value aufgeteilt. Query ist dabei der der Text der verarbeitet wird, Key der

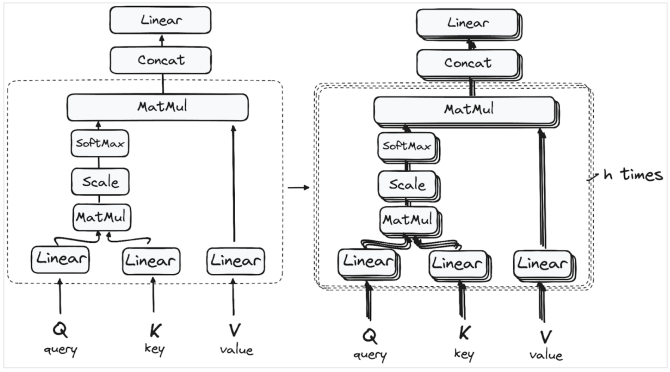

Fig. 3. Multi-Self Attention Layer nach [4]

Referenz Text gegen den verglichen wird und Value wird für die Ausgabeberechnung des Attention Elements verwendet. In Figure 3 wird der Attention Mechanismus (dargestellt links) h mal parallel berechnet (dargestellt rechts). Daher wird dieses Modellelement Multi Self Attention bezeichnet. Zur Berechnung der Ausgabe des Attention Layers werden die Attention Gewichte zunächst berechnet welche bestimmen, wie wichtig ein Wort für ein anderes Wort ist, was durch nachfolgende Figure 4 visualisiert ist. Die Attention Gewichte werden dabei durch die Matrix Multiplikation von Query und Key erzeugt. Query und Key durchlaufen dazu zunächst ein Lineares Layer dessen Gewichte erlernt werden müssen. Die

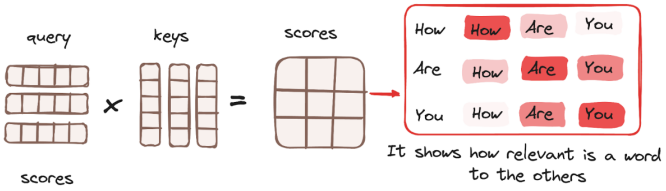

Fig. 4. Attention Mechanismus nach [4]

Färbung in der Figure 4 zeigt dabei die Wichtigkeit der Wörter zueinander. Die daraus berechnete Attention Gewichte werden mit dem Werten der Wörter multipliziert (zweites MatMult) und über ein Lineares Layer verfeinert, dies ergibt die Ausgabe des Attention Layers. Der wesentliche Unterschied zwischen dem Multi-Self Attention Layer und dem Masked Multi-Self Attention Layer ist dabei die Maskierung der Attention Gewichte. Die Maskierung sorgt dafür, dass ein Wort nur von vorherigen Wörtern abhängt. Die nachfolgende Figure 5 zeigt die Maskierung der Attention Gewichte mithilfe einer Maskierungsmatrix. Die resultierende Matrix enthält für alle noch unbekannten Tokens eine $-\infty$ und für alle bekannten Tokens den Wert der skalierten Attention Gewichte.

Die genaue Funktionsweise der einzelnen Schichten aus Figure 2 wird in [1] beschrieben. Abschließend wird in der GPT Architektur ein weiteres Layer je nach Aufgabe angehängt, welche in Figure 2 durch Text Vorhersage oder Aufgaben Klassifizierung repräsentiert wird. Dies kann beispielsweise ein Lineares Layer mit Softmax Ak-

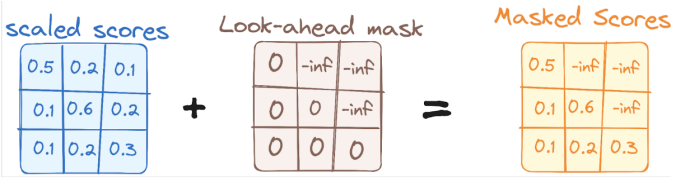

Fig. 5. Maskierung der Attention Gewichte nach [4]

tivierungsfunktion sein, welches die Wahrscheinlichkeiten für das nächste Wort aus dem Vokabular berechnet [4].

C. Abgrenzung der Decoder Architektur zur Encoder Architektur

Die vorgeschlagene Decoder Architektur durch [3] unterscheidet sich dabei von der Encoder Architektur durch die Verwendung von Masked Multi-Self Attention Layern. Die Encoder Architektur verwendet hingegen Multi-Self Attention Layer ohne Maskierung. Entscheidend ist ebenfalls das die Architektur aus [3] keine Einbeziehung der Encoder Schicht vorsieht und somit im Vergleich zur Transformer Architektur aus [1] keine weitere Multi-Self Attention Layer enthält. Die Decoder Architektur ist dabei auf die Generierung von Text ausgelegt und wird als autoregressives Modell betrachtet. Die Encoder Architektur hingegen ist auf die Verarbeitung von Texten ausgelegt und wird als diskriminatives Modell betrachtet.

D. Trainingsprozess der GPT-Decoder Architektur

Das entscheidende bei der in [3] beschrieben Decoder Architektur ist der Trainingsprozess der aus zwei Stufen besteht.

1) unüberwachtes Vortrainieren: Das unüberwachte Vortrainieren des Modells. Dabei ist die Zielfunktion folgende Likelyhoodfunktion:

$$\mathcal{L}_1(\mathcal{U}) = \sum_i logP(u_i|u_i - k...u_i - 1; \theta) \tag{1}$$

k ist dabei das Kontext Fenster. Somit wird die Wahrscheinlichkeit des nächsten Wortes auf Basis der vorherigen k-Tokens bestimmt. Das erste GPT Modell wurde dabei auf Tokens aus dem BookCorpus Datensatz trainiert. Wie bereits in Figure 2 beschrieben wurden 12 Decoder Schichten verwendet. Für die Attention Heads h wurden 12 Heads verwendet bei 768 Embedding Dimensionen. Für das Feed Forward Netzwerk wurde eine GELU (Gaussian Error Linear Unit) verwendet. Das Training wurde über 100 Epochen mit einer Batchgröße von 64 durchgeführt.

2) überwachtes Fine-Tuning: Der zweite Schritt ist das überwachte Fine-Tuning auf die gewünschte Zielaufgabe. Dabei handelt es sich um m-Eingabe Tokens, welche in eine Klasse aus $\mathcal{C}$ transformiert werden. Die Zielfunktion ist folgende Likelyhoodfunktion:

$$\mathcal{L}_2(\mathcal{C}) = \sum_{(x,y)} logP(y|x^1...x^m) \tag{2}$$

Dabei wird die Zielfunktion aus Equation 2 mit der Funktion aus Equation 1 zu Equation 3 kombiniert damit der Trainingsprozess besser konvergiert und generalisiert [3].

$$\mathcal{L}_3(\mathcal{C}) = \mathcal{L}_2(\mathcal{C}) + \lambda\mathcal{L}_1(\mathcal{C}) \tag{3}$$

Da das vortrainierte Modell wie in Equation 1 beschrieben auf die vorhersage der nächsten Worte optimiert ist, müssen für das Finetuning die Input Tokens entsprechend der nachfolgenden Figure 6, wie in [3] beschrieben, angepasst werden. Dabei werden strukturierte Daten in eine geordnete Reihen-

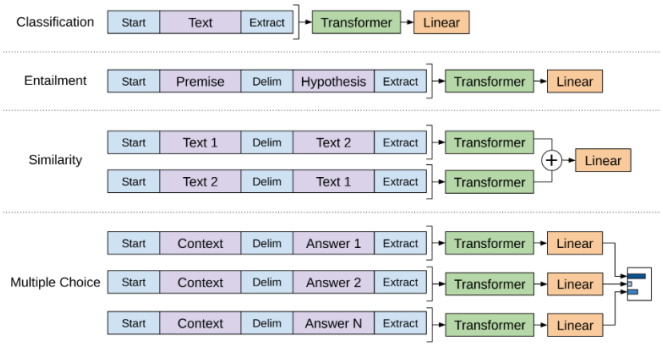

Fig. 6. Anpassung der Input Tokens nach [3]

folge gebracht, um die Eingabe für das Modell zu generieren. Figure 6 zeigt, dass für eine Multiple Choice Aufgabe zuerst ein zufällig ausgewähltes Start Token, der Kontext und anschließend die Antwort mit einem zufällig ausgewählten Wert für das Abschluss Token übergeben wird. Für diese spezifische Aufgabe werden anschließend drei Transformer parallel trainiert, um die korrekte Antwort vorherzusagen. Für das Finetuning wurden die meisten Hyperparameter aus dem Unüberwachten Training übernommen. Lediglich wurde ein Dropout vor der Klassifizierungsschicht hinzugefügt und die Lernrate wurde reduziert. Das Paper aus [3] zeigt dabei, dass 3 Epochen ausreichen um eine gewünschte Genauigkeit zu erzielen.

E. Weiterentwicklung der GPT-Decoder Architektur

Die erste GPT Version ist ein Meilenstein in der Entwicklung von Decoder-only Architekturen. Die Weiterentwicklung der GPT Architektur zeigt dabei die Entwicklung von GPT-2, GPT-3 und GPT-4.

1) GPT-2: GPT-2 verbesserte die Funktionsweise von großen Sprachmodellen, dass diese mehrere Aufgaben lösen können. Dabei war das Ziel ein Modell zu entwerfen, welches eine Warscheinlichkeit für eine Output Token Sequenz in Abhängigkeit der Input Tokens sowie der Aufgaben Spezifikation zu erlernen.

$$P(s_{n-k}, ..., s_n|s_1, ..., s_{n-k-1}, task) \tag{4}$$

Für das Training des GPT-2 Modell wird als Datensatz WebText verwendet welche aus Inhalten kuratierter Websiten besteht. Die Modell Architektur ist ähnlich zur ursprünglichen GPT Architektur, jedoch wurde die Anzahl des Kontextfensters von 512 auf 1024 erhöht. Anstatt 12 Decoder Schichten

wurden bei GPT-2 48 Schichten verwenden mit einer Embedding Dimension von 1600 was zu einer Parametermenge von ca. 1,5 Milliarden Parameter führt. GPT-2 zeigt zum ersten mal das ein großes Sprachmodell auf mehrere Aufgaben gleichzeitig implizit trainiert werden kann, sofern der Trainingsdatensatz divers genug ist [5].

2) GPT-3: Während der Fokus bei GPT-2 vorallem auf der Zero Shot Performance und das Unsupervised Mulittask Learning liegt, wurde das GPT-3 Modell im Fokus des Few Shot Learnings betrachtet. GPT-3 erweitert auch hier die Funktionalität von GPT-2 erneut und verwendet statt des Masked Multi-Self Attention Layer, Attention Konzepte aus [6] ähnlich der sogenannten Sparse Transformes. Diese ermöglichen es Längere Sequenzen zu verarbeiten und das Training tieferer Modelle zu ermöglichen. Das Paper aus [7] zeigt dass größere Modelle entscheidende Vorteile beim Meta Learning oder Few-Shot Learning haben. So besteht das größte GPT-3 Modell aus 175 Milliarden Parametern.

3) GPT-4: Das neueste Modell von OpenAI GPT-4 zeigt die Weiterentwicklung von GPT-3 und setzt zusätzlich zum Unsupervised Vortraining auf Finetuning über Menschen gesteuertes Feedback mittels Reinforcement Learning. Leider gibt es zum aktuellen Zeitpunkt keine weiteren Details zur Architektur von GPT-4 wie in [8] beschrieben. Das Paper in [8] zeigt jedoch anhand der verwendeten Trainingsressourcen, dass das GPT Modell entsprechend groß ist. Zudem wird beschrieben, dass eine Infrastruktur im Rahmen der GPT4 Entwicklung entwickelt wurde um verlässliche die Entwicklung des Modells vorher zu sagen und die Trainingsressourcen zu optimieren.

III. Anwendungsbereiche

In den unterschiedlichen Papern werden bei der Evaluation ebenfalls unterschiedliche NLP Aufgaben betrachtet die für eine Anwendung großer Sprachmodelle in Frage kommen. Es lassen sich 10 Anwendungsbereiche auf Basis von [9] identifizieren, die große Potenziale bieten.

A. Übersetzung

[10] zeigt, dass ChatGPT ein guter Übersetzer ist. Es zeigt sich jedoch auch, dass ChatGPT besser europäische Sprachen übersetzen kann als fernere Sprachen.

B. Erstellung von Inhalten

Das GPT-3 Paper [7] zeigte bereits, dass dieses Nachrichten Artikel generieren kann, welche für den Menschen als schwer unterscheidbar gelten. Daher eignenen sich große Sprachmodelle auch für die Erstellung diverser Inhalte. Beispielsweise können Inhalte erstellt werden, die der Ideenfindung oder Inspiration dienlich sein können. Beispiele können hierfür Blogs, Fragebögen oder auch beiträge in sozialen Medien sein [9].

C. Suchwerkzeug

Im Anwendungsbereich der Suchwerkzeuge eignet sich ein großes Sprachmodell, wie bereits das GPT-2 Paper [5] zeigt. Weiterhin kann das Modell für eine Schlüsselwort Forschung (engl. Keyword Research) verwendet werden um die wichtigsten Schlüsselwörter für ein Thema zu erhalten. Damit können SEO-freundliche Inhalte erstellt werden [9].

D. Virtuelle Assistenten und Kundenbetreuung

Als Einsatz von Chatbots zeigt auch Zalando in [11], dass große Sprachmodelle für die Kundenbetreuung eingesetzt werden können. Dabei können Kundenanfragen automatisiert beantwortet werden. Das Sprachmodell unterstützt dabei den potenziellen Kunden bei der Auswahl von Produkten für bestimmte Anlässe oder gibt Vorschläge auf basis der persönlichen Präferenzen und Vorlieben.

E. Erkennung und Verhinderung von Cyberangriffen

Große Sprachmodelle eignen sich unter anderem zur Erkennung von Cyberangriffen SentinalOne sowie Microsoft zeigen wie große Sprachmodelle Datensätze auf Muster hinsichtlich Cyberangriffen scannen um so frühzeitig Angriffe zu erkennen [12] [13]. Mit dem LLM SecPaLM zeigt Google, dass große Sprachmodelle verwendet werden können um das Verhalten von Skripten scannen zu können. Damit muss der Benutzer die Software nicht zwangsläufig in einer Sandbox ausführen um zu prüfen ob die Software schädlich ist [9].

F. Code-Entwicklung

Ein weiterer wichtiger Anwendungsfall ist die Unterstützung von großen Sprachmodellen bei der Code-Entwicklung. LLM's können dabei Code Abschnitte schreiben, Dokumentation generieren oder bei der Fehlersuche während des Programmierens unterstützen. [9]

G. Transkription

Im Paper von GPT-4 [8] wird auf die Multimodalität eingegangen und gezeigt wie gut große Sprachemodelle beispielsweise Bilder verarbeiten können. Diese Fähigkeit kann dazu genutzt werden um Transkritpionen anzufertigen [9].

H. Marktforschung

In der Marktforschung müssen viele Datensätze analysiert und ausgewertet werden. So kann ein großes Sprachmodell beispielsweise bei der Marktforschung unterstützen indem es Zusammenfassungen erstellt, Trends erkennt oder Marktlücken identifitiert [9].

I. Vertriebsautomatisierung

Große Sprachmodelle können auch im Vertriebswesen verwendete werden um Vertribsprozesse zu automatisieren. [9] zeigt, dass LLM's bei der Lead-Generierung, Pflege, Personalisierung, Qualifizierung, Bewertung sowie Lead Prognose unterstützen kann, indem es entsprechende Lead Datensätze auswertet.

J. Stimmungsanalyse

Ein weiterer wichtiger Anwendungsbereich ist in der Stimmungsanalyse. Große Sprachmodelle können dabei helfen die Stimmung von Texten zu analysieren und so beispielsweise die Stimmung von Kunden in Online Shops zu analysieren [9]. Mit der Stimmungsanalyse können Unterhnehmen die assozierten Wörter mit dem Unterhnehmen aus Kommentaren herausfiltern und erhalten ein Bild über die Marke des Unternehmens. Abhängig davon lässt sich die Marke entsprechend der Stimmungslage verbessern [9].

IV. Potenzial und Grenzen von Decoder Architekturen

Der vorherige Abschnitt zeigt das breite Anwendungsspektrum von Decoder Architekturen. Da sich Decoder Architekturen stetig weiter entwickeln werden sich die Anwendungsfelder ebenfalls erweitern. Nach [9] wird das Marktpotenzial generativer KI's bis 2032 auf bis zu 1,3 Billionen Dollar geschätzt. Neben den Potenzialen gibt es jedoch auch Grenzen die bei der Verwendung von Decoder Architekturen beachtet werden müssen. Es lässt sich dabei unterscheiden in Potenziale und Grenzen durch die Architketurwahl selbst und durch das konkrete Modell wie bspw. GPT. Für GPT-3 wurden die Poteziale und Grenzen in [5] und [8] genauer beschrieben.

A. Potenziale und Grenzen der Decoder Architektur

1) Potenziale: Nach [14] bieten dabei Transformer Architekturen mit Self Attention Mechanismus folgende Vorteile:

- Parallelisierung und Effizienz → Der Self Attention Mechanismus verarbeitet parallel die Eingabe Tokens.
- Langfristige Abhängigkeiten → Der Self Attention Mechanismus bietet im Vergleich zu Rekursiven Neuronalen Netzwerken die Möglichkeit lange Abhängigkeiten zu besser zu erlernen.
- Skalierbarkeit → Die Komplexität für den Attention Mechanismus is Linear und somit skaliert ein Transformer Modell besser als andere Netzwerke über die Token Länge.
- Transfer Learning mit Transformer → Wie auch die Paper von GPT zeigen, können Transformer Modelle ohne große Architekturanpassung auf unterschiedliche Anwendungsbereiche gefinetuned werden.
- Kontextuelle Embeddings → Die Verwendung von Kontextualisierten Embeddings macht es möglich das ein Wort mehrere Bedeutungen in Abhängigkeit des Kontextes haben kann.
- Globale Informationsverarbeitung → Durch die parallele Verarbeitung aller Tokens im Attention Mechanismus können globale Informationen verarbeitet werden. Im Vergleich zu RNN's welche die Daten sequenziell verarbeiten.

2) Grenzen: Die Grenzen nach [14] sind dabei:

- Attention Overhead bei langen Sequenzen → Für extreme Lange Sequenzen kann der Attention Mechanismus sehr Aufwendig zu berechnen sein.
- Fehlende Reihenfolge → Wird zwar durch das Positional Encoding im Eingangslayer erreicht, ist aber jedoch nicht so explizit wie bei RNN's.
- Übermäßige Parametrisierung → Große Sprachmodelle haben eine hohe Anzahl an Parametern. Damit ist das Training solcher Modelle wesentlich Aufwändinger als bei anderen Modellen.
- Unstrukturierte Eingabedaten → Transformer oder auch Decoder Architekturen wurden für strukturierte Daten entwickelt. Unstrukturierte Daten wie Bilder oder Audiodaten können nicht direkt verarbeitet werden. Dennoch zeigt GPT-4 die Multimodalität von LLM's [8].
- Feste Eingabelänge → Typischer weise besitzen Transformer Modelle eine feste Eingabelänge. Dennoch gibt es Transfromer Architekturen welche mit Variablen Eingabelängen umgehen können [15].

B. Potenziale und Grenzen von GPT

Im nachfolgenden werden die Potenziale und Grenzen von GPT nach [15] beleuchtet.

1) Potenziale: Folgen Potenziale bietet das GPT Modell:

- Verständnis des Kontexts → GPT Modelle können den Kontext eines Satzes besser verstehen und somit besser auf die Anfrage des Nutzers eingehen.
- Large-Scale Sprachmodell → Die GPT Mdelle sind auf Texten aus dem Internet vortrainiert und haben damit ein breite Wissensbasis.
- Fine-Tuning Prozess → Durch das Menschen gesteuerte Feedback Fine-tuning kann GPT bessere uns sichere Antworten generieren.
- Iterative Entwicklung → Durch ständige Forschung und Weiterentwicklung entsteht eine immer verbesserte Version von GPT.

C. Grenzen

Die Grenzen von GPT sind dabei:

- Mangelndes Weltwissen → GPT Modelle haben nur auf Daten und Informationen zugriff auf diese das Modell trainiert worden ist. Damit Fehlt dem GPT Echtzeit Informationen.
- Bias im Sprachmodell → GPT Modelle können durch die Trainingsdaten Biases aufweisen. Die Biases können durch entsprechendes Fine-Tuning reduziert aber nicht komplett eliminiert werden. In den Papern von GPT-3 wird genauer auf Biases eingegangen [7].
- Unangemessene oder unsichere Ausgaben → GPT produziert manchmal unangemessene oder unsichere Ausgaben.
- Fehlen von tiefem Verständnis → GPT Modelle haben kein tiefes Verständnis von Texten und können daher keine Schlussfolgerungen ziehen. Antworten durch GPT basieren auf Muster im Trainingsdatensatz und generiert dadurch unlogische oder flasche Antworten.
- Halluzination → GPT kann die Antworten faktisch nicht überprüfen und teniert zum Halluzinieren von Informationen.

V. Fazit

Im nachfolgenden Abschnitt wird zunächst die aktuellen Forschungsbereiche im Bereich der Decoder Architekturen beleuchtet. Abschließend wird ein Ausblick gegeben.

A. Aktuelle Forschungbereiche

Die Entwicklung von Decoder-only Architekturen zeigte in den letzten Jahren rasante Entwicklungen. Dennoch wird weiter an der Entwicklung von Transformer Modellen sowei Decoder Only Architekturen geforscht. Wie das Paper aus [16] zeigt, wird vor allem an der Effizienz der Transformer bzw. Decoder-only Architektur geforscht. Wie auch das GPT-3 Paper in [7] zeigt, sind fähige Modelle sehr tief und haben damit einen großen Parametersatz, was das Training inneffizient und kostenintesiv macht. Das Paper aus [16] zeigt dabei Architekturalternativen (ParallelGPT, LinearGPT, ConvGPT) auf welche eine ähnliche Performance zu den GPT Modellen aufweisen. Ein weiterer Forschungsbereich ist ob Decoder Only Modelle Turing Vollständig sind [17]. Das Paper zeigt, dass auch kleine Decoder Only Modelle Turing Vollständig sind. Damit können Decoder Only Modelle jede theoretisch berechenbare Funktion ausführen. Weitere relevante Forschungsgebiete sind die in den GPT Papern erwähnten Biases in Sprachmodellen.

B. Ausblick

GPT und die damit verbundenen Decoder Architekturen haben in den letzten Jahren große Fortschritte in der NLP Forschung gebracht. Die Entwicklung von GPT1 bis GPT-4 zeigt die Weiterentwicklung von Decoder-only Architekturen in den letzten Jahren. Die Anwendungsbereiche von Decoder Architekturen sind dabei vielfältig. Die aktuellen Forschungen zeigen, das vor allem die Modellgröße ein relevanter Faktor für die Güte von Modellen ist, daher gibt es berechtigte Bestrebungen an der Effizienz von Decoder Architekturen bzw. großen Sprachmodellen generell zu forschen. Ein wichtiger Aspekt bei der Verwendung von Sprachmodellen sind dabei immer der Bias der durch den vortrainings Datensatz entstehen kann. Die damit Verbundene Diskriminierung ist ein relevantes Forschungsgebiet welche in den nächsten Jahren weiter erforscht werden sollte. Es bleibt dennoch abzuwarten wie sich die Forschung im Bereich der Decoder Architekturen weiterentwickelt und welche neuen Anwendungsbereiche sich daraus ergeben.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Aug. 2023.

[2] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, B. Yin, and X. Hu, "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond."

[3] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training."

[4] "Wie Transformatoren funktionieren: Eine detaillierte Erkundung der Transformatorarchitektur," https://www.datacamp.com/tutorial/how-transformers-work.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners."

[6] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating Long Sequences with Sparse Transformers," Apr. 2019.

[7] 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, and D. Amodei, "Language Models are Few-Shot Learners," Jul. 2020.

[8] OpenAI, L. Ahmad, F. L. Aleman, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Ł. Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Ł. Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. d. A. B. Peres, M. Petrov, H. P. d. O. Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. J. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph, "GPT-4 Technical Report," Mar. 2024.

[9] T. Keary, "12 Praktische Large Language Model (LLM) Anwendungen," Sep. 2023.

[10] W. Jiao, W. Wang, J.-t. Huang, X. Wang, S. Shi, and Z. Tu, "Is ChatGPT A Good Translator? Yes With GPT-4 As The Engine," Nov. 2023.

[11] L. Wuttke, "Large Language Model Fallbeispiele," https://datasolut.com/large-language-model-fallbeispiele/, Sep. 2023.

[12] "Microsoft Copilot for Security | Microsoft Security," https://www.microsoft.com/en-us/security/business/ai-machine-learning/microsoft-copilot-security.

[13] "SentinelOne Unveils Revolutionary AI Platform for Cybersecurity - SentinelOne DE," https://de.sentinelone.com/press/sentinelone-unveils-revolutionary-ai-platform-for-cybersecurity/.

[14] A. Kulkarni, A. Shivananda, A. Kulkarni, and D. Gudivada, *Applied Generative AI for Beginners: Practical Knowledge on Diffusion Models, ChatGPT, and Other LLMs*. Berkeley, CA: Apress, 2023.

[15] ——, "The ChatGPT Architecture: An In-Depth Exploration of OpenAI's Conversational Language Model," in *Applied Generative AI for Beginners: Practical Knowledge on Diffusion Models, ChatGPT, and Other LLMs*, A. Kulkarni, A. Shivananda, A. Kulkarni, and D. Gudivada, Eds. Berkeley, CA: Apress, 2023, pp. 55–77.

[16] S. K. Suresh and S. P, "Towards smaller, faster decoder-only transformers: Architectural variants and their implications," Aug. 2024.
[17] J. Roberts, "How Powerful are Decoder-Only Transformer Neural Models?" Oct. 2024.