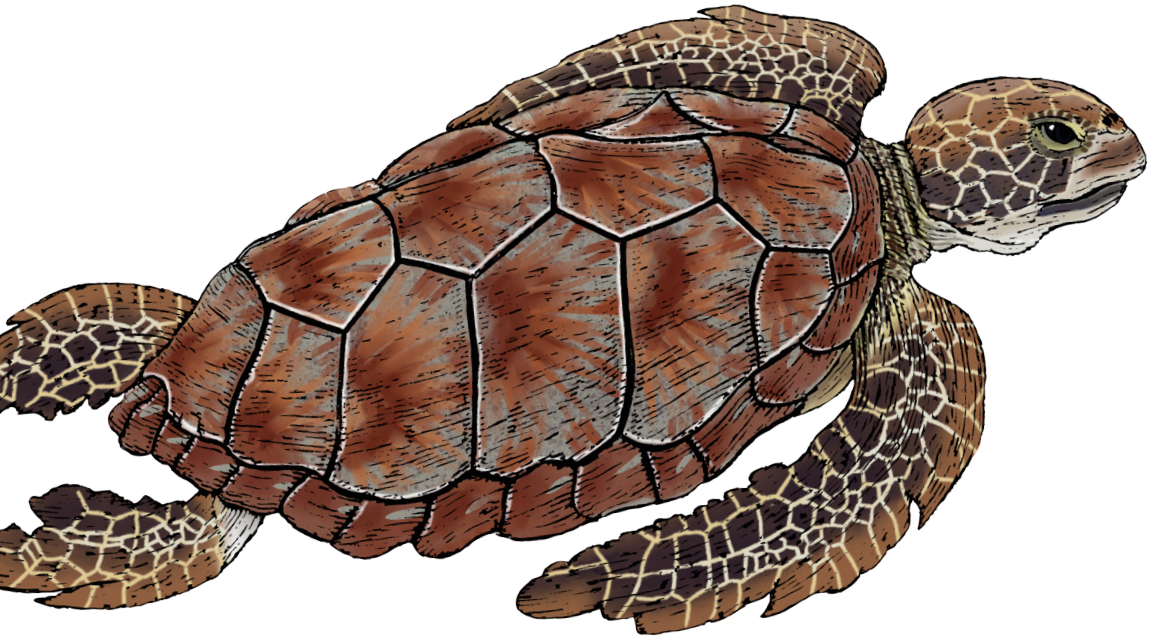
O'REILLY®

2. Auflage

Anwendungen mit GPT-4 und ChatGPT entwickeln

Intelligente Chatbots, Content-Generatoren und mehr erstellen



Olivier Caelen &
Marie-Alice Blete

Übersetzung von Thomas Demmig

KAPITEL 1

Grundlagen von GPT-4 und ChatGPT

Die Möglichkeit, auf künstliche Intelligenz zurückzugreifen, lag für Entwicklerinnen und Entwickler nie näher. *Large Language Models* (LLMs) wie GPT-4 oder GPT-3.5 Turbo haben mit ChatGPT gezeigt, was sie können. Jetzt finden wir uns in einem Sturm des Fortschritts wieder, dessen Geschwindigkeit in der Welt der Software so noch nie vorgekommen ist. OpenAI hat diese technologischen Innovationen direkt einsetzbar gemacht – welche transformativen Anwendungen werden Sie daraus bauen, nachdem die Tools dazu nun zur Verfügung stehen?

Die Möglichkeiten dieser KI-Modelle gehen über Chatbots weit hinaus. Dank der LLMs können Sie in der Entwicklung nun auf die Fähigkeiten der Verarbeitung natürlicher Sprache (*Natural Language Processing*, NLP) zurückgreifen, um Anwendungen zu schaffen, die die Bedürfnisse der User auf eine Weise verstehen, die bisher wie Science-Fiction klang. Durch die neuen bildlichen Fähigkeiten von GPT-4 ist es zudem nun möglich, Software zu bauen, die Text anhand von Bildern interpretieren und generieren kann. Von innovativen Kundensupportsystemen, die lernen und sich anpassen, bis hin zu personalisierten Lernwerkzeugen, die den jeweiligen Lernstil der Schüler und Studentinnen verstehen, eröffnen Sprachmodelle von GPT eine ganze neue Welt von Möglichkeiten.

Aber was *sind* GPT-Modelle? Ziel dieses Kapitels ist es, sich deren Grundlagen, Ursprünge und zentralen Fähigkeiten anzuschauen. Wenn Sie die Grundlagen dieser KI-Modelle verstehen, sind Sie beim Bauen der nächsten Generation von LLM-gestützten Anwendungen schon einen ganzen Schritt weiter.

Einführung in Large Language Models

Dieser Abschnitt erklärt die grundlegenden Bausteine, die die Entwicklung von GPT-Modellen beeinflusst haben. Wir versuchen, Sprachmodelle und NLP, die

Rolle von Transformer-Architekturen und die Tokenisierungs- und Vorhersageprozesse in diesen Modellen umfassend zu erklären. Die Reise ist bei der Verarbeitung von Texten aber noch nicht zu Ende. Mit GPT-4 Vision werden die Möglichkeiten von LLMs über den Umgang mit Texten hinaus auf *multimodale* Eingaben erweitert. GPT-4 ist damit nicht nur in der Textinterpretation gut, sondern auch beim Auswerten von Bildern.

Die Grundlagen von Sprachmodellen und NLP

Als LLMs sind GPT-Modelle die neueste Art von Modellen im Feld der Verarbeitung natürlicher Sprache, die wiederum eine Untermenge des maschinellen Lernens (*Machine Learning*, ML) und der künstlichen Intelligenz (*Artificial Intelligence*, AI) ist. Bevor wir uns genauer mit den GPT-Modellen befassen, ist es wichtig, sich NLP (*Natural Language Processing*) und die zugehörigen Bereiche anzuschauen.

Es gibt unterschiedliche Definitionen von AI, aber eine davon – mehr oder weniger der Konsens in diesem Gebiet – sagt, dass AI das Entwickeln von Computersystemen ist, die Aufgaben erledigen können, für die im Allgemeinen menschliche Intelligenz erforderlich ist. Diese Definition vereint viele Algorithmen unter dem Dach der KI. Denken Sie beispielsweise an das Vorhersagen von Verkehrsmengen in GPS-Anwendungen oder die regelbasierten Systeme, die in strategischen Videospielen zum Einsatz kommen. In diesen Beispielen scheint der Rechner von außen betrachtet Intelligenz zu erfordern, um die Aufgaben erledigen zu können.

ML ist eine Untermenge von AI. In ML müssen wir nicht versuchen, die Entscheidungsregeln des AI-Systems direkt zu implementieren. Stattdessen versuchen wir, Algorithmen zu entwickeln, die dem System erlauben, selbstständig aus Beispielen zu lernen. Seit den 1950er-Jahren – als die ML-Forschung begann – wurden in der wissenschaftlichen Literatur viele ML-Algorithmen vorgeschlagen.

Darunter haben sich insbesondere die Deep-Learning-Algorithmen einen Namen gemacht. *Deep Learning* ist ein Zweig des ML, der sich auf Algorithmen konzentriert, die durch die Struktur des Gehirns inspiriert wurden. Diese Algorithmen werden künstliche neuronale Netze (*Artificial Neural Networks*) genannt. Sie können sehr große Datenmengen verarbeiten und funktionieren erstaunlich gut bei Aufgaben wie der Bild- oder der Spracherkennung und bei NLP.

Die GPT-Modelle basieren auf einem bestimmten Typ von Deep-Learning-Algorithmen, den sogenannten *Transformern*, die in dem Artikel »Attention Is All You Need« von Vaswani et al. von Google im Jahr 2017 eingeführt wurden (*https://oreil.ly/jVZW1*). Transformer sind vergleichbar mit Lesemaschinen. Sie nutzen einen Attention-Mechanismus, um die unterschiedlichen Teile des Texts zu priorisieren, um so ein tieferes Verständnis des Kontexts zu erhalten und kohärente Antworten zu erzeugen. So können sie die Bedeutung von Wörtern in Sätzen erfassen, was die Performance beim Übersetzen natürlicher Sprachen, dem Beantworten von Fragen (*Question Answering*, QA) und dem Erzeugen von Text verbessert. Abbildung 1-1 verdeutlicht diese zentralen Konzepte und ihre Rolle beim Verbessern der Fähigkeiten von Transformer-Modellen für die diversen Sprachaufgaben.

NLP ist ein Unterbereich von AI, der sich darauf fokussiert, mit Computern natürliche, menschliche Sprache zu verarbeiten, zu interpretieren und zu generieren. Moderne NLP-Lösungen basieren auf ML-Algorithmen. Das Ziel von NLP ist, Computer Texte aus natürlicher Sprache verarbeiten zu lassen. Dieses Ziel umfasst ein breites Aufgabenspektrum:

Textklassifikation
: Eingabetext wird in vordefinierte Gruppen kategorisiert. Dazu gehören beispielsweise die Sentimentanalyse und die Themenklassifizierung. Firmen können mit der Sentimentanalyse die Meinung von Kundinnen und Kunden zu ihren Angeboten ermitteln. Das Filtern von E-Mails ist ein Beispiel für die Themenklassifizierung, bei der eine E-Mail in Kategorien wie »Privat«, »Soziale Netze«, »Werbung« und »Spam« unterteilt werden kann.

Automatisches Übersetzen
: Text aus einer Sprache wird automatisch in eine andere übersetzt. Dazu kann auch das Übersetzen von Code aus einer Programmiersprache in eine andere gehören, zum Beispiel von Python nach C++.

Question Answering – Beantworten von Fragen
: Das Beantworten von Fragen basiert auf einem gegebenen Text. So kann beispielsweise ein Onlinekundenserviceportal ein NLP-Modell nutzen, um FAQs über ein Produkt zu beantworten, oder Lernsoftware verwendet NLP, um Antworten auf die Fragen von Studenten oder Schülerinnen zum aktuellen Thema zu geben.

Textgenerierung
: Es wird ein kohärenter und relevanter Ausgabetext generiert, der auf einem gegebenen Eingabetext – dem Prompt – basiert.

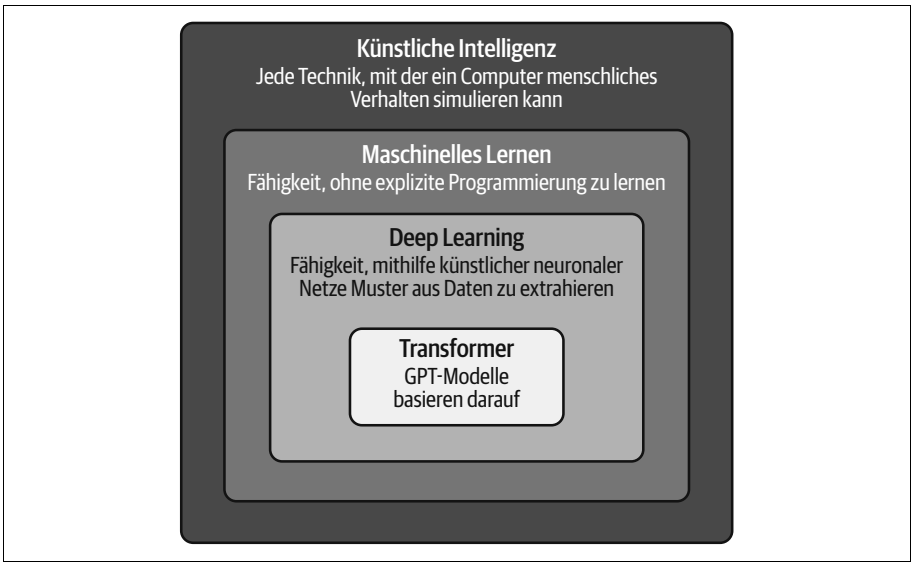


Abbildung 1-1: Technologien von KI bis zu Transformern

Wie schon erwähnt, sind LLMs ML-Modelle, die versuchen, unter anderem Aufgaben im Bereich der Textgenerierung zu lösen. Sie ermöglichen Computern, menschliche Sprache zu verarbeiten, zu interpretieren und zu generieren, was zu einer effektiveren Kommunikation zwischen Mensch und Maschine führt. Dazu analysieren LLMs riesige Textmengen beziehungsweise *trainieren* damit und erlernen dabei Muster und Beziehungen zwischen Wörtern in Sätzen. Für diesen Lernprozess können diverse Datenquellen zum Einsatz kommen. Es können Texte aus Wikipedia, Reddit, aus Archiven mit Tausenden von Büchern oder sogar das Archiv des Internets selbst genutzt werden. Erhält ein LLM einen Eingabetext, kann es Vorhersagen über die Wörter machen, die am wahrscheinlichsten folgen werden, und damit sinnvolle Antworten liefern. Das LLM besitzt sehr viele interne Parameter, und beim Lernen sucht der Algorithmus, der das LLM aufbaut, nach den optimalen Parametern, mit denen das Modell die bestmöglichen Vorhersagen zu den nächsten Wörtern treffen kann. Die modernen Sprachmodelle, wie zum Beispiel die aktuellen von GPT, sind so groß und wurden mit so vielen Texten trainiert, dass sie nun direkt die meisten NLP-Aufgaben ausführen können, wie zum Beispiel die Textklassifikation, maschinelles Übersetzen oder das Beantworten von Fragen (*Question Answering*, QA).

OpenAI hat unterschiedliche Sprachmodelle vorgestellt. Aktuell sind die neuesten und leistungsfähigsten die der GPT-4-Serie. Neben der Möglichkeit der Textverarbeitung steht GPT-4 Vision für eine deutliche Weiterentwicklung als multimodales Modell, mit dem nicht nur Text, sondern auch Bilder als Eingabe möglich sind. LLMs können Bilder mithilfe einer spezialisierten Transformer-Architektur namens *Vision Transformer* (ViT) verarbeiten. Das ganz aktuelle Modell GPT-4o geht sogar noch weiter – es kann Text, Bilder und Audio verarbeiten und generieren.

Der Beginn der Entwicklung von LLMs reicht bis in die 1990er-Jahre zurück. Sie begann mit einfachen Sprachmodellen wie *N-Grammen*, die versucht haben, abhängig vom vorherigen Wort das nächste Wort in einem Satz vorherzusagen. N-Gramm-Modelle haben dazu die *Häufigkeit* genutzt. Das vorhergesagte nächste Wort ist das am häufigsten vorkommende Wort, das im zum Training verwendeten N-Gramm-Modell auf das vorherige Wort folgt. Dieser Ansatz war zwar ein guter Ausgangspunkt, aber da sich die N-Gramm-Modelle im Verstehen des Kontexts und der Grammatik ziemlich schwertaten, waren die erzeugten Texte inkonsistent.

Um die Leistungsfähigkeit von N-Gramm-Modellen zu verbessern, kamen ausgefeiltere Lernalgorithmen zum Einsatz, unter anderem rekurrente neuronale Netze (*Recurrent Neural Networks*, RNNs) und *Long-Short-Term-Memory*-Netzwerke (LSTM). Diese Modelle konnten längere Sequenzen lernen und den Kontext besser als N-Gramme analysieren, aber sie brauchten immer noch Hilfe beim effizienten Verarbeiten großer Datenmengen. Diese Art von rekurrenten Modellen war lange Zeit die effizienteste und daher diejenige, die zum Beispiel bei der automatischen maschinellen Übersetzung zum Einsatz kam.

Die Transformer-Architektur und ihre Rolle in LLMs

Die Transformer-Architektur hat NLP revolutioniert, vor allem weil Transformer eine der kritischsten Einschränkungen früherer NLP-Modelle (wie RNNs) effektiv aufgehoben haben – den Kampf erster Modelle mit langen Eingabetextsequenzen und das Bewahren des Kontexts über diese langen Sequenzen. Mit anderen Worten: RNNs haben dazu tendiert, in längeren Sequenzen den Kontext zu vergessen (das berüchtigte »katastrophale Vergessen«), Transformer hingegen haben die Möglichkeit, diesen Kontext effektiv beizubehalten und zu codieren.

Die zentrale Säule dieser Revolution ist der *Attention-Mechanismus* – eine einfache, aber sehr leistungsfähige Idee. Statt alle Wörter in einer Textsequenz als gleich wichtig zu betrachten, zollt das Modell bei jedem seiner Schritte vor allem den relevantesten Begriffen Aufmerksamkeit. Dieser Mechanismus ermöglicht direkte Verbindungen zwischen entfernten Elementen im Text. So kann beispielsweise das letzte Wort ohne Beschränkungen mit dem ersten Wort verbunden werden, was eine signifikante Einschränkung älterer Modelle wie RNNs war. Cross-Attention und Self-Attention sind zwei Architekturblöcke, die auf diesem Attention-Mechanismus basieren, und sie kommen oft in LLMs zum Einsatz. Die Transformer-Architektur nutzt diese Blöcke ausgesprochen häufig.

Cross-Attention hilft dem Modell dabei, die Relevanz der unterschiedlichen Teile des Eingabetexts zu bestimmen, um das nächste Wort im Ausgabetext exakt vorherzusagen. Sie ist wie ein Scheinwerfer, der Wörter oder Phrasen im Eingabetext beleuchtet und die relevanten Informationen hervorhebt, die erforderlich sind, um die nächste Wortvorhersage zu treffen, und der die weniger wichtigen Details ignoriert.

Um das zu verdeutlichen, wollen wir ein Beispiel mit einer einfachen Übersetzung eines Satzes durchgehen. Stellen Sie sich vor, wir hätten den englischen Satz »Alice enjoyed the sunny weather in Brussels« als Eingabe, der in den französischen Satz »Alice a profité du temps ensoleillé à Bruxelles« übersetzt werden soll. In diesem Beispiel wollen wir uns darauf fokussieren, das französische Wort *ensoleillé* zu generieren, das *sunny* bedeutet. Für diese Vorhersage würde die Cross-Attention mehr Gewicht auf die englischen Wörter *sunny* und *weather* legen, da beide für die Bedeutung von *ensoleillé* relevant sind. Durch das Konzentrieren auf diese beiden Wörter hilft Cross-Attention dem Modell dabei, eine genaue Übersetzung für diesen Teil des Satzes zu erzeugen. Abbildung 1-2 illustriert dieses Beispiel.

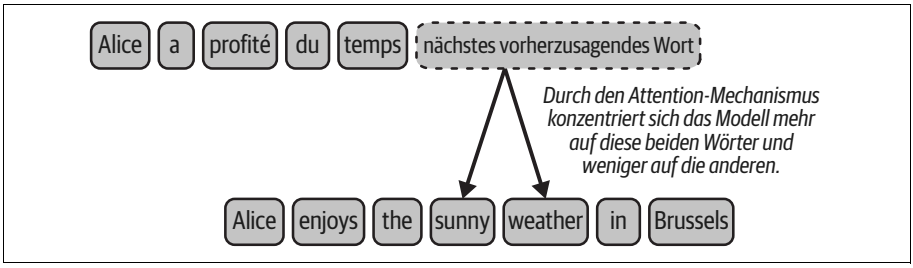


Abbildung 1-2: Cross-Attention nutzt den Attention-Mechanismus, um sich auf die wichtigen Teile des Eingabetexts (des englischen Satzes) zu konzentrieren und das nächste Wort des Ausgabetexts (des französischen Satzes) vorherzusagen.

Self-Attention bezieht sich auf die Fähigkeit eines Modells, sich auf unterschiedliche Teile des Eingabetexts zu fokussieren. Im Kontext von NLP kann das Modell die Wichtigkeit jedes Worts in einem Satz in Bezug auf die anderen Wörter bewerten. Dadurch kann es die Beziehungen zwischen den Wörtern besser verstehen, und das Modell kann aus mehreren Wörtern im Eingabetext neue *Konzepte* bauen.

Schauen Sie sich als spezifischeres Beispiel folgenden Satz an: »Alice received praise from her colleagues.« Nehmen wir an, dass das Modell versucht, in diesem Satz die Bedeutung des Worts *her* zu verstehen. Der Self-Attention-Mechanismus weist den Wörtern unterschiedliche Gewichte zu und hebt dabei die Wörter hervor, die in diesem Kontext für *her* relevant sind. Im Beispiel würde die Self-Attention mehr Gewicht auf die Wörter *Alice* und *colleagues* legen. Sie hilft dem Modell dabei, aus diesen Wörtern neue Konzepte zu bauen. In diesem Beispiel wäre eines der Konzepte, das so entstehen könnte, möglicherweise »Alice's colleagues« (siehe Abbildung 1-3).

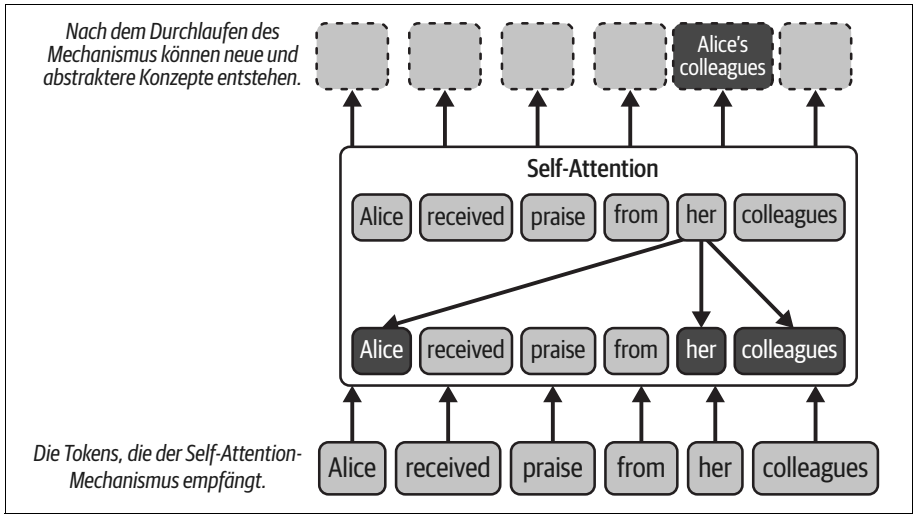


Abbildung 1-3: Self-Attention erlaubt das Entstehen des Konzepts »Alice's colleagues«.

Anders als die rekurrente Architektur haben Transformer zudem den Vorteil, leicht *parallelisierbar* zu sein. Das heißt, die Transformer-Architektur kann mehrere Teile des Eingabetexts simultan statt sequenziell verarbeiten. Dies ermöglicht eine schnellere Berechnung und ein schnelleres Training, weil unterschiedliche Teile des Modells parallel abgearbeitet werden können, ohne dass darauf gewartet werden muss, dass vorherige Schritte fertig sind. Bei der rekurrenten Architektur ist hingegen eine sequenzielle Verarbeitung notwendig. Die

Möglichkeit der parallelen Verarbeitung bei Transformer-Modellen passt sehr gut zur Architektur von Grafikkarten (*Graphics Processing Units*, GPUs), die dazu designt sind, viele Berechnungen gleichzeitig durchzuführen. Daher sind GPUs aufgrund ihrer hochparallelen, leistungsfähigen Berechnungsmöglichkeiten ideal zum Trainieren und Ausführen der Transformer-Modelle. Dieser Vorteil erlaubte den Data Scientists, Modelle mit viel größeren Datensätzen zu trainieren, was den Weg für die Entwicklung von LLMs geebnet hat.

Die Transformer-Architektur ist ein Sequence-to-Sequence-Modell, das ursprünglich für Sequence-to-Sequence-Aufgaben wie das maschinelle Übersetzen entwickelt wurde. Ein Standard-Transformer besteht aus zwei Hauptkomponenten: einem Encoder und einem Decoder, die beide stark auf Attention-Mechanismen aufbauen. Die Aufgabe des Encoders ist es, den Eingabetext zu verarbeiten, wichtige Merkmale zu identifizieren und eine sinnvolle Repräsentation dieses Texts zu generieren – ein *Embedding*. Der Decoder nutzt dann dieses Embedding, um eine Ausgabe zu erzeugen, wie zum Beispiel eine Übersetzung oder eine Zusammenfassung. Diese Ausgabe interpretiert letztendlich die codierten Informationen.

Generative vortrainierte Transformer (Generative Pre-Trained Transformers), im Allgemeinen als *GPT* bekannt, sind eine Familie von Modellen, die auf der Transformer-Architektur basieren und dabei spezifisch den Decoder-Teil der ursprünglichen Architektur nutzen. In der GPT-Architektur ist der Encoder nicht vorhanden, daher ist auch keine Cross-Attention erforderlich, um die Embeddings zu integrieren, die von einem Encoder erstellt wurden. GPT baut nur auf dem Self-Attention-Mechanismus im Decoder auf, um kontextabhängige Repräsentationen und Vorhersagen zu erzeugen. Andere bekannte Modelle, wie zum Beispiel BERT (*Bidirectional Encoder Representations from Transformers*), basieren auf dem Encoder-Teil, wir werden sie in diesem Buch aber nicht behandeln. Abbildung 1-4 zeigt die Entwicklung dieser verschiedenen Modelle.

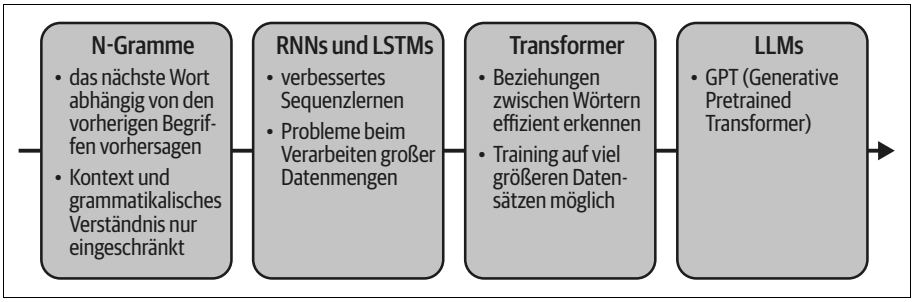


Abbildung 1-4: Die Entwicklung von NLP-Techniken von N-Grammen bis zum Entstehen von LLMs

Die Tokenisierungs- und Vorhersageschritte in GPT-Modellen entmystifizieren

LLMs erhalten als Eingabe einen Prompt und generieren als Reaktion darauf einen Text. Dieser Prozess wird als *Textvervollständigung* bezeichnet. Der Prompt könnte beispielsweise lauten: »*The weather is nice today, so I decided to*«. Das Modell könnte dann vielleicht »*go for a walk*« ausgeben. Möglicherweise fragen Sie sich, wie das LLM-Modell diesen Ausgabetext aus dem Eingabe-Prompt erzeugt. Wie Sie sehen werden, ist das vor allem eine Frage der Wahrscheinlichkeiten.

Wird ein Prompt an ein LLM geschickt, teilt dieses die Eingabe zunächst in kleinere Elemente auf – in sogenannte *Tokens*. Diese Tokens stehen für einzelne Wörter, Wortteile oder Leer- und Satzzeichen. Der obige Prompt könnte beispielsweise unterteilt werden in `["The", "wea", "ther", "is", "nice", "today", ",", "so", "I", "de", "ci", "ded", "to"]`. Jedes Sprachmodell bringt seinen eigenen Tokenizer mit. Der Tokenizer von GPT-3.5 und GPT-4 steht online auf der OpenAI-Plattform (*https://oreil.ly/hbKT7*) zu Testzwecken zur Verfügung.

Als Faustregel kann man bei Tokens annehmen, dass 100 Tokens ungefähr 75 Wörtern in einem englischen Text entsprechen. Bei anderen Sprachen mag das anders aussehen, und die Anzahl an Tokens kann für die gleiche Zahl an Wörtern größer sein.

Dank des Attention-Prinzips und der Transformer-Architektur verarbeitet das LLM diese Tokens und kann die Beziehung zwischen ihnen sowie die Gesamtbedeutung des Prompts interpretieren. Die Transformer-Architektur erlaubt einem Modell, die entscheidenden Informationen und den Kontext im Text effizient zu erfassen.

Um einen neuen Satz zu erstellen, sagt das LLM die Tokens voraus, die basierend auf dem vom User angegebenen Eingabekontext des Prompts am wahrscheinlichsten folgen werden. OpenAI bietet viele Versionen von GPT-4 an. Zuerst hatten Sie nur die Wahl zwischen einem Eingabekontextfenster mit maximal 8.192 oder 32.768 Tokens. Anfang 2024 wurden die neuesten Modelle GPT-4 Turbo und GPT-4o von OpenAI veröffentlicht, die ein größeres Eingabekontextfenster von 128.000 Tokens aufweisen – das entspricht fast 300 Seiten englischen Texts. Anders als die vorherigen rekurrenten Modelle, die Probleme mit langen Eingabesequenzen hatten, kann das moderne LLM durch die

Transformer-Architektur und den Attention-Mechanismus den Kontext im Ganzen berücksichtigen. Abhängig von diesem Kontext weist das Modell jedem potenziell folgenden Token einen Wahrscheinlichkeitswert zu. Das Token mit der höchsten Wahrscheinlichkeit wird dann als Nächstes in der Sequenz gewählt. In unserem Beispiel könnte auf »The weather is nice today, so I decided to« als nächstes bestes Token »go« folgen.

Wie Sie im nächsten Kapitel sehen werden, ist es mithilfe eines *Temperatur*-Parameters möglich, nicht einfach das nächste Token mit der höchsten Wahrscheinlichkeit zu nutzen, sondern es dem Modell zu erlauben, das nächste Token aus einer Reihe von Tokens mit der höchsten Wahrscheinlichkeit auszuwählen. Das sorgt für mehr Variabilität und Kreativität in der Antwort des Modells.

Dieser Prozess wird anschließend wiederholt, aber nun wird der Kontext zu »The weather is nice today, so I decided to go« – also mit dem zuvor vorhergesagten Token »go« am Ende des ursprünglichen Prompts. Das zweite Token, das das Modell möglicherweise vorhersagt, könnte »for« sein. Dieser Prozess wird so lange wiederholt, bis ein vollständiger Satz geformt ist: »The weather is nice today, so I decided to go for a walk«. Der Prozess baut auf der Fähigkeit des LLM auf, das wahrscheinlichste nächste Wort aus den riesigen Textdaten gelernt zu haben. Abbildung 1-5 zeigt diesen Prozess.

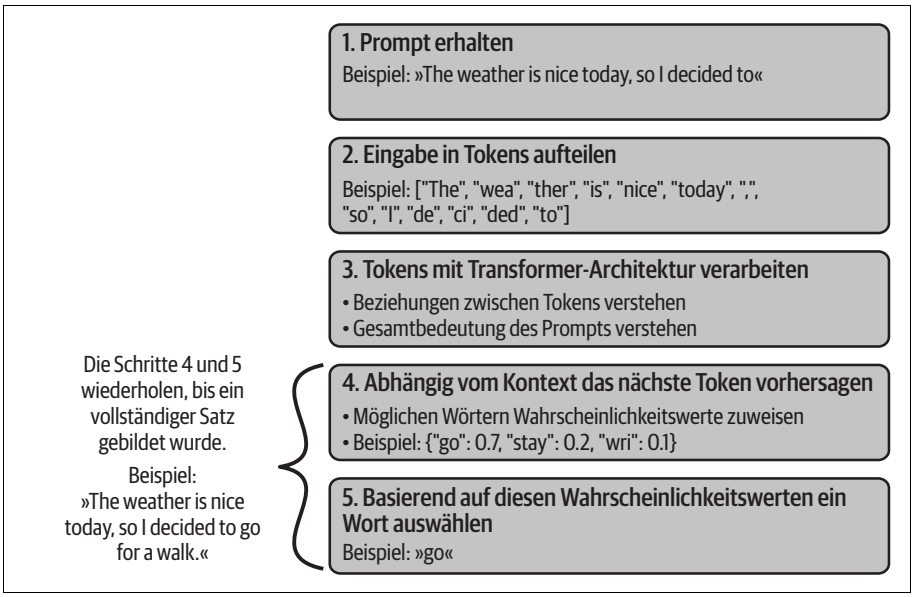


Abbildung 1-5: Der Vervollständigungsprozess läuft iterativ ab – Token für Token.

Einbinden von Vision in ein LLM

Mit GPT-4 Vision wird die GPT-4-Serie um multimodale Fertigkeiten ergänzt, sodass sie jetzt auch mit mehr als nur mit Text umgehen kann. Die spezifischen Mechanismen sorgen dafür, dass dieses Feature proprietär bleibt und nicht so einfach eingesehen werden kann. Es lassen sich aber Schlüsse aus Open-Source-LLMs ziehen, die visuelle Daten integrieren, sodass wir zumindest ein grundlegendes Verständnis der potenziellen Methoden erlangen können, mit denen GPT-4 um solch eine multimodale Funktionalität erweitert wird. In diesem Abschnitt sollen die Prozesse vorgestellt werden, die sich in diesen Open-Source-Gegenstücken beobachten lassen, damit Sie eine Idee davon bekommen, wie die Bild-Text-Integration in GPT-4 möglicherweise umgesetzt ist.

Convolutional Neural Networks (CNNs) waren lange Zeit State-of-the-Art bei Aufgaben zur Bildverarbeitung. Sie sind sehr gut beim Klassifizieren von Bildern und Erkennen von Objekten. Das erreichen sie, indem sie Filterschichten nutzen, die über ein Eingabebild geschoben werden. Diese Filter können die räumlichen Beziehungen zwischen den Pixeln des Bilds aufdecken, und sie helfen den CNNs dabei, Muster zu erkennen – von einfachen Kanten in den ersten Schichten bis hin zu komplexen Formen und Objekten in den tieferen Schichten.

Aber ähnlich wie die Einführung der Transformer-Architekturen im Jahr 2017 NLP revolutionierte und RNNs ablöste, wurden im Jahr 2020 neue Modelle für die Bildverarbeitung vorgeschlagen, die auf Transformer-Architekturen basieren. Seitdem wird die seit Langem bestehende Dominanz von CNNs in der Bildverarbeitung infrage gestellt. Im Jahr 2021 zeigte ein Artikel in Google mit dem Titel »An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale« von Dosovitskiy et al. (*https://oreil.ly/ijPSk*), dass ein reines Transformer-Modell namens *Vision Transformer* (ViT) für viele Bildklassifikationsaufgaben eine bessere Leistung liefern kann.

Sie fragen sich vielleicht, wie der Transformer die Bilddaten verarbeitet. Aus der Ferne betrachtet, ist das Vorgehen dem bei Text ziemlich ähnlich. Wird ein Text-Prompt an ein LLM geschickt, teilt es den Text erst in kleinere Zeichenabschnitte auf – die Tokens – und verarbeitet diese dann, um das nächste Token vorherzusagen. Bei Bildern teilt der ViT das Bild erst in kleine, gleich große Kacheln. In Abbildung 1-6 sehen Sie ein Beispiel dafür.

Diese Bildkacheln werden dann mit den Text-Tokens in einer einheitlichen Eingabesequenz zusammengeführt. Ohne nun allzu sehr in die technischen Details zu gehen: Wenn ein LLM Textdaten verarbeitet, wird jedes Token zunächst in einen hochdimensionalen *Vektor* umgewandelt. Diese Mapping-

Funktion zwischen den Tokens und den hochdimensionalen Vektoren wird während des Lernprozesses des LLM berechnet. Eine Mapping-Funktion zwischen den Kacheln und dem gleichen hochdimensionalen Raum wird ebenfalls während des Lernprozesses ermittelt. So finden sich danach sowohl die Tokens wie auch die Kacheln im gleichen hochdimensionalen Raum. Die kombinierte Sequenz aus Text und Bild kann dann von der Transformer-Architektur verarbeitet werden, um das nächste Token vorherzusagen. Durch die Kombination aus Bildkacheln und Text-Tokens im gleichen hochdimensionalen Darstellungsraum kann das Modell Self-Attention-Mechanismen für beide Modalitäten nutzen und Antworten erzeugen, die Text- und Bildinformationen berücksichtigen. Bei der Python-Entwicklung kann die Fähigkeit, Bilder zu verarbeiten, die Interaktion der User mit Ihrer KI-Anwendung deutlich besser machen, zum Beispiel über intuitivere Chatbots oder Lerntools, die Inhalte aus Bildern verstehen und erläutern können.

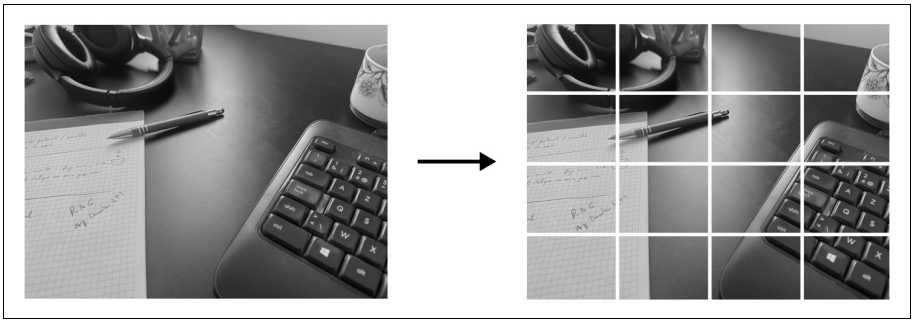


Abbildung 1-6: Ein Bild wird in gleich große Kacheln unterteilt, bevor es an den Transformer übergeben wird.

Eine kurze Geschichte des GPT: von GPT-1 bis GPT-4

In diesem Abschnitt werden wir die Entwicklung der GPT-Modelle von OpenAI von GPT-1 bis GPT-4 betrachten.

GPT-1

Mitte 2018 – nur ein Jahr nach der Erfindung der Transformer-Architektur – hat OpenAI den Artikel »Improving Language Understanding by Generative Pre-Training« (*https://oreil.ly/Yakwa*) von Radford et al. veröffentlicht, in dem die Firma den *Generative Pre-Trained Transformer* vorstellt – auch bekannt als GPT-1.

Inhalt

Vorwort **11**

1 Grundlagen von GPT-4 und ChatGPT **15**

Einführung in Large Language Models 15

Die Grundlagen von Sprachmodellen und NLP 16

Die Transformer-Architektur und ihre Rolle in LLMs 19

Die Tokenisierungs- und Vorhersageschritte in GPT-Modellen entmystifizieren 23

Einbinden von Vision in ein LLM 25

Eine kurze Geschichte des GPT: von GPT-1 bis GPT-4 26

GPT-1 26

GPT-2 28

GPT-3 28

Von GPT-3 zu InstructGPT 29

GPT-3.5, ChatGPT und Codex 32

GPT-4 33

Die Entwicklung der KI hin zur Multimodalität 38

Anwendungsfälle für LLMs und Beispielprodukte 39

Be My Eyes 39

Morgan Stanley 40

Khan Academy 41

Duolingo 41

Yabble 42

Waymark 43

Inworld AI 43

Vorsicht, KI-Halluzinationen: Einschränkungen und Überlegungen 44
Das Potenzial von GPT durch fortgeschrittene Features ausreizen . . . 47
Zusammenfassung 50

2 Die APIs von OpenAI 53
Grundlegende Konzepte 54
In der OpenAI-API verfügbare Modelle 55
GPT Base 56
InstructGPT (Legacy) 56
GPT-3.5 56
GPT-4 57
GPT-Modelle mit dem OpenAI Playground ausprobieren 58
Einstieg in die OpenAI-Python-Bibliothek 63
OpenAI-Zugriff und API-Schlüssel 63
»Hallo Welt«-Beispiel 66
Chat-Completion-Modelle einsetzen 68
Eingabeoptionen für den Chat-Completion-Endpoint 69
Mit der Temperatur und top_p experimentieren 74
Ausgabeformat für den Chat-Completion-Endpoint 77
Vision 78
Eine JSON-Ausgabe erzwingen 82
Andere Modelle zur Textvervollständigung verwenden 86
Eingabeoptionen für den Textvervollständigungs-Endpoint 87
Ausgabeformat für den Textvervollständigungs-Endpoint 89
Überlegungen zu Kosten und Datenschutz 89
Preise und Token-Beschränkungen 89
Sicherheit und Privatsphäre: Achtung! 90
Andere OpenAI-APIs und Funktionen 91
Embeddings 91
Moderation-Modell 94
Text-to-Speech 98
Speech-to-Text 100
Images-API 104
Zusammenfassung (und Spickzettel) 115

3 Apps mit GPT-4 und ChatGPT bauen 117
Überblick über die Anwendungsentwicklung 117
API-Schlüsselmanagement 118
Sicherheit und Datenschutz 120
Prinzipien zum Design der Softwarearchitektur 121
LLM-Fähigkeiten in Ihre Projekte integrieren 122
Gesprächsfähigkeiten 122
Sprachverarbeitungsfähigkeiten 123
Mensch-Maschine-Interaktion 125
Fähigkeiten kombinieren 126
Beispielprojekte 127
Projekt 1: Einen News-Generator bauen – Sprachverarbeitung 127
Projekt 2: YouTube-Videos zusammenfassen – Sprachverarbeitung 130
Projekt 3: Einen Experten für Zelda BOTW erschaffen – Sprachverarbeitung und Unterhaltung 135
Projekt 4: Persönlicher Assistent – Mensch-Maschine-Schnittstelle 142
Projekt 5: Dokumente organisieren – Sprachverarbeitung 150
Projekt 6: Sentimentanalyse – Sprachverarbeitung 152
Kostenmanagement 160
Angriffspunkte in LLM-gestützten Apps 162
Eingaben und Ausgaben analysieren 164
Die Unvermeidbarkeit der Prompt Injection 165
Mit einer externen API arbeiten 165
Umgang mit Fehlern und unerwarteten Verzögerungen 165
Rate Limits 167
Responsivität und User Experience verbessern 168
Zusammenfassung 172

4 Fortgeschrittene Integrationsstrategien mit OpenAI 173
Prompt Engineering 173
Effektive Prompts mit Rollen, Kontexten und Aufgaben designen 175
Schritt für Schritt denken 182

Few-Shot Learning implementieren 184
Iteratives Verbessern mit User-Feedback 187
Die Effektivität von Prompts verbessern 193
Optimieren 196
Der Einstieg 197
Mit der OpenAI-API optimieren 201
Optimieren über die Weboberfläche von OpenAI 205
Optimieren für Anwendungen 207
Synthetische Daten für eine E-Mail-Marketing-Kampagne generieren und optimieren 210
Die Kosten des Optimierens 219
RAG – Retrieval-Augmented Generation 219
Naive RAG 220
Fortgeschrittene RAG 221
Grenzen der RAG 227
Zwischen Strategien wählen 228
Evaluierungen 231
Von einer Standardanwendung zu einer LLM-gestützten Anwendung 232
Prompt Sensitivity 232
Nichtdeterminismus 233
Halluzinationen 234
Zusammenfassung 236

5 LLMs durch Frameworks, Plug-ins und mehr erweitern 239
Das LangChain-Framework 239
LangChain-Bibliotheken 241
Dynamische Prompts 242
Agenten und Tools 244
Memory 248
Embeddings 250
Das LlamaIndex-Framework 254
Demonstration: RAG in zehn Zeilen Code 254
Prinzipien von LlamaIndex 255
Anpassung 257

GPT-4-Plug-ins 258
Überblick 260
Die API 261
Das Plug-in-Manifest 263
Die OpenAPI-Spezifikation 264
Beschreibungen 266
GPTs 266
Die Assistant API 272
Eine Assistant API erstellen 274
Eine Unterhaltung mit Ihrer Assistant API managen 275
Aufruf von Funktionen 280
Die Assistenten auf der Webplattform von OpenAI 284
Zusammenfassung 287

6 Das große Ganze 289
Wichtige Erkenntnisse 289
Alle Elemente zusammenbringen: der Assistenten-Use-Case 291
Schritt 1: Die Idee 291
Schritt 2: Die Anforderungen definieren 292
Schritt 3: Einen Prototyp bauen 293
Schritt 4: Verbessern, Iterieren 294
Schritt 5: Die Lösung robust machen 295
Erkenntnisse 296

A Glossar der wichtigsten Begriffe 299

B Tools, Bibliotheken und Frameworks 307

Index 311

O'REILLY®

Anwendungen mit GPT-4 und ChatGPT entwickeln

Diese Einführung zeigt Python-Entwicklerinnen und -Entwicklern, wie sie Anwendungen mit Large Language Models erstellen. Olivier Caelen und Marie-Alice Blete erklären die wichtigsten Features von GPT-4 und GPT-3.5 und beschreiben, wie sie für eigene NLP-Aufgaben eingesetzt werden können. In nachvollziehbaren Schritten wird erläutert, wie Sie mithilfe der OpenAI-Python-Bibliothek Applikationen zur Textgenerierung, für das Question Answering oder für intelligente Assistenten entwickeln.

Anschauliche Beispiele sowie klare und detaillierte Erklärungen unterstützen Sie dabei, die Konzepte zu verstehen und sie auf Ihre Projekte anzuwenden. Die Codebeispiele sind in einem GitHub-Repository verfügbar. Zudem enthält das Buch ein Glossar mit den wichtigsten Begriffen. Für diese 2. Auflage wurde das Buch aktualisiert und deutlich erweitert, es berücksichtigt die neuesten Entwicklungen in der KI-Technologie wie beispielsweise GPT-4o und Multimodalität.

Sind Sie bereit, das Potenzial von Large Language Models in Ihren Anwendungen zu nutzen? Dann ist dieses Buch ein Muss für Sie. Es behandelt:

- **Grundlagen und Stärken von GPT-4- und GPT-3.5-Modellen und deren Funktionsweise**
- **Die Integration dieser Modelle in Python-basierte Anwendungen für Aufgaben im Natural Language Processing**
- **Beispielprojekte, die den Einsatz der OpenAI-API für Aufgaben wie Texterstellung, Inhaltszusammenfassung, Dokumentenklassifikation oder Sentimentanalyse demonstrieren**
- **Fortgeschrittene LLM-Themen wie das Prompt Engineering, das Optimieren von Modellen, RAG, Plug-ins, LangChain, LlamaIndex und KI-Assistenten**

»Verbindet nahtlos Theorie und praktische Übungen und macht die Komplexität von GPT-4 und ChatGPT zugänglich.«

—Lucas Soares
ML Engineer, Biometrid

»Mit praktischen Beispielen und Schritt-für-Schritt-Anleitungen ebnet das Autorenteam den Weg zur innovativen App-Entwicklung.«

— Tom Taulli
Autor des Buchs
»Programmieren mit KI« (O'Reilly)

Olivier Caelen ist ML Researcher beim Paytech-Pionier Worldline. Er hat einen Doktortitel in Machine Learning und unterrichtet auch ML-Kurse an der Université libre de Bruxelles.

Marie-Alice Blete ist AI Engineer. Sie teilt ihr Wissen gerne mit Data-Science-Kolleginnen und -Kollegen und hat Freude daran, sich als Tech-Speakerin in die Community einzubringen.

9 783960 092575

www.dpunkt.de

Euro 39,90 (D)
ISBN 978-3-96009-257-5

Gedruckt in Deutschland
Mineralölfreie Druckfarben
Zertifiziertes Papier