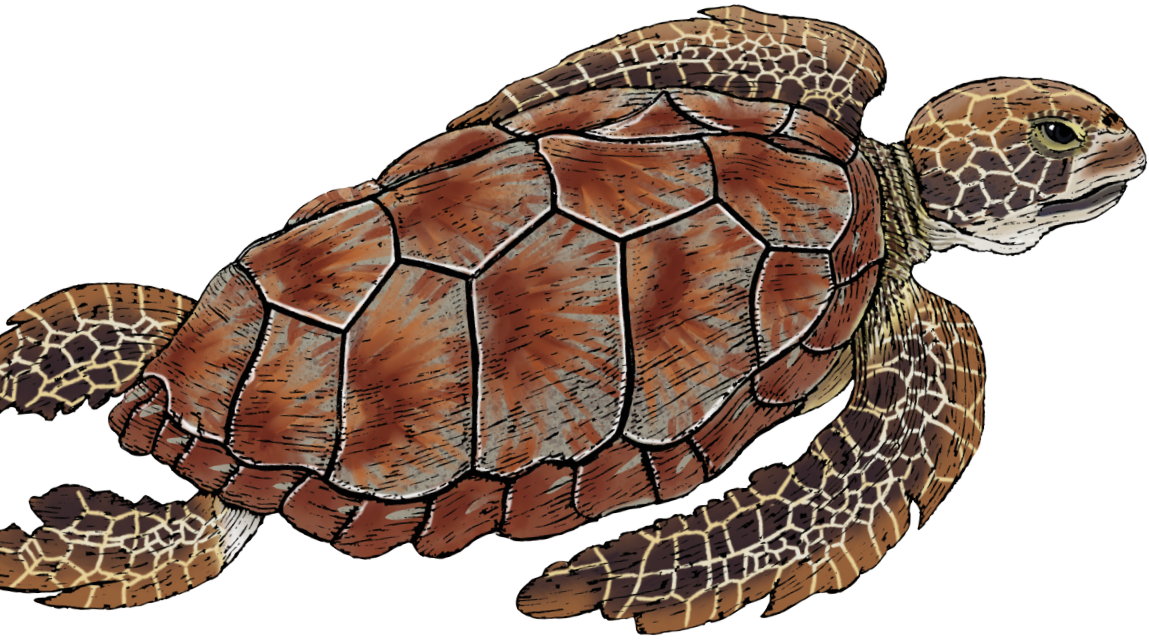
O'REILLY®

Deutsche Ausgabe

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

Stellen Sie sich eine Welt vor, in der Sie mit Computern so schnell kommunizieren können wie mit Ihren Freundinnen und Freunden. Wie würde die aussehen? Was für Anwendungen könnten Sie erschaffen? Dies ist die Welt, die OpenAI mit seinen GPT-Modellen mitgestaltet, indem es menschenähnliche Konversationsfunktionen auf unsere Geräte bringt. Als neueste Errungenschaften der künstlichen Intelligenz sind GPT-4 und andere GPT-Modelle *Large Language Models* (LLMs), die mit massiven Datenmengen trainiert werden, sodass sie mit sehr großer Genauigkeit Texte erkennen und wie von Menschen erstellt generieren können.

Die Möglichkeiten dieser KI-Modelle gehen über einfache Sprachassistenten weit hinaus. Dank der Modelle von OpenAI können Sie in der Entwicklung nun auf die Fähigkeiten der Verarbeitung natürlicher Sprache (*Natural Language Processing*, NLP) zurückgreifen, um Anwendungen zu schaffen, die unsere Bedürfnisse auf eine Weise verstehen, die bisher wie Science Fiction klang. Von innovativen Kundensupportsystemen, die lernen und sich anpassen, bis hin zu personalisierten Lernwerkzeugen, die den jeweiligen Lernstil der Schüler und Studentinnen verstehen, eröffnen GPT-4 und ChatGPT eine ganze neue Welt von Möglichkeiten.

Aber was *sind* GPT-4 und ChatGPT? Ziel dieses Kapitels ist es, sich die Grundlagen, Ursprünge und zentralen Fähigkeiten dieser KI-Modelle anzuschauen. Wenn Sie die Grundlagen der 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-4 und ChatGPT beeinflusst haben. Wir versuchen, Sprachmodelle und NLP, die Rolle von Transformer-Architekturen und die Tokenisierungs- und Vorhersageprozesse in GPT-Modellen umfassend zu erklären.

Die Grundlagen von Sprachmodellen und NLP

Als LLMs sind GPT-4 und ChatGPT 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 GPT-4 und ChatGPT befassen, ist es wichtig, sich NLP 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 Spracherkennung und bei NLP.

GPT-4 und ChatGPT basieren auf einem bestimmten Typ von Deep-Learning-Algorithmen, den sogenannten *Transformern*. Transformer sind wie Lesemaschinen. Sie schauen sich die unterschiedlichen Teile eines Satzes oder eines Textblocks an, um dessen Kontext zu verstehen und eine kohärente Antwort zu erzeugen. Sie können zudem die Reihenfolge von Wörtern und deren Kontext in einem Satz verstehen. Damit sind sie bei Aufgaben wie der Übersetzung natürlicher Sprachen, dem Beantworten von Fragen (Question Answering, QA) und dem Erzeugen von Text ausgesprochen effektiv. Abbildung 1-1 zeigt die Beziehungen zwischen diesen Begriffen.

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 Themenklassifi-

zierung, 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.

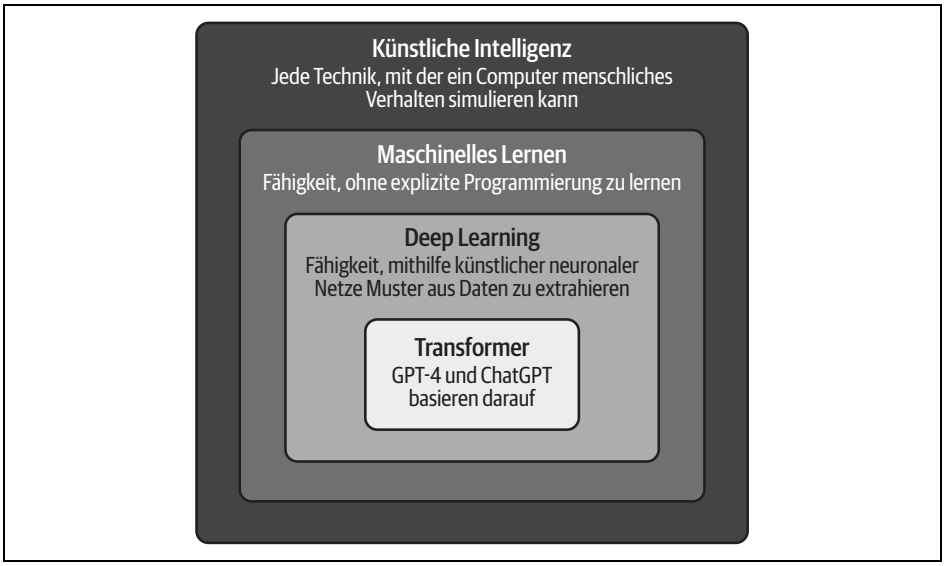


Abbildung 1-1: Technologien von AI bis zu Transformern

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. Die modernen Sprachmodelle, die in den letzten Monaten veröffentlicht wurden, 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). Die GPT-4- und ChatGPT-Modelle sind moderne LLMs, die besonders bei der Textgenerierung brillieren.

Der Beginn der Entwicklung von LLMs reicht bereits viele 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 die NLP revolutioniert, vor allem weil Transformer eine der kritischsten Einschränkungen früherer NLP-Modelle (wie RNNs) effektiv aufgehoben haben – den Kampf mit langen Textsequenzen 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. 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 Cross-Attention- und Self-Attention-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 be-

leuchtet 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.

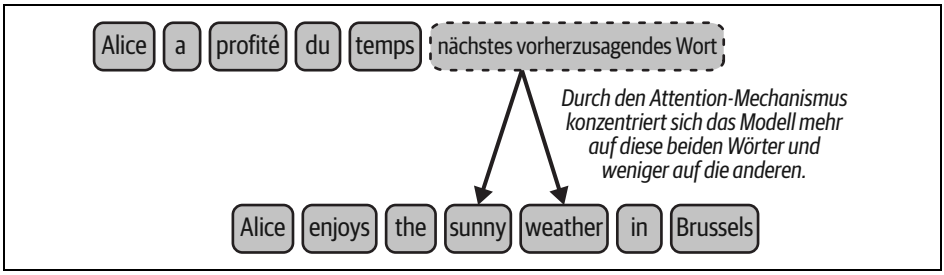


Abbildung 1-2: Cross-Attention nutzt den Attention-Mechanismus, um sich auf die wichtigen Teile des Eingabetexts (des englischen Satzes) zu konzentrieren und das nächste Wort des Ausgabetexts (des französischen Satzes) vorherzusagen.

Self-Attention bezieht sich auf die Fähigkeit eines Modells, sich auf unterschiedliche Teile des Eingabetexts zu fokussieren. Im Kontext der 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).

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.

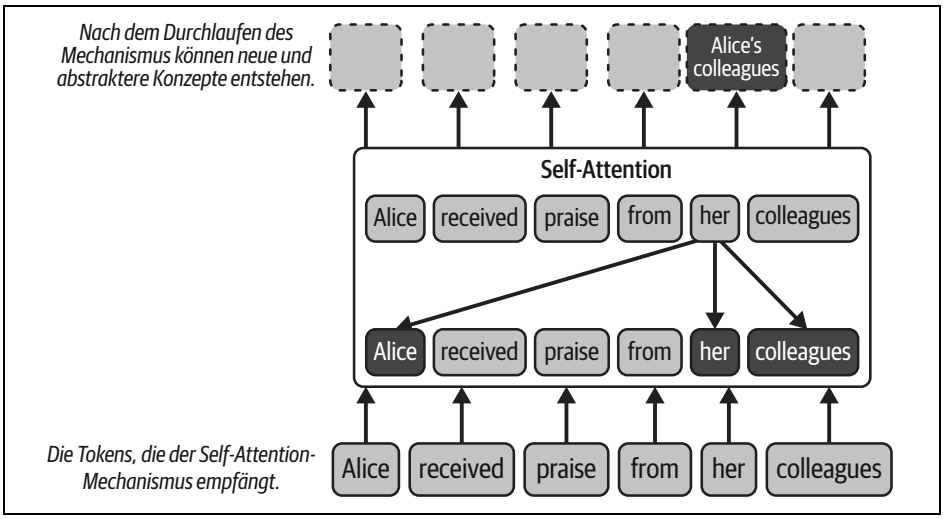


Abbildung 1-3: Self-Attention erlaubt das Entstehen des Konzepts »Alice's colleagues«.

Die Transformer-Architektur, im Jahr 2017 von Vaswani et al. von Google im Artikel »Attention Is All You Need« (*https://oreil.ly/jVZW1*) vorgestellt, wurde ursprünglich für Sequence-to-Sequence-Aufgaben wie das maschinelle Übersetzen entwickelt. 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 Pretrained 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 GPT 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, diese werden wir aber in diesem Buch nicht behandeln. Abbildung 1-4 zeigt die Entwicklung dieser verschiedenen Modelle.

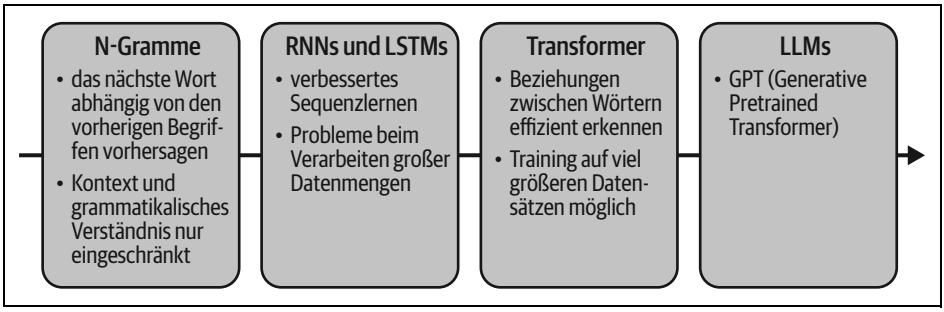


Abbildung 1-4: Die Entwicklung von NLP-Techniken von N-Grammen bis zum Entstehen von LLMs

Die Tokenisierungs- und Vorhersageschritte in GPT-Modellen entmystifizieren

In der GPT-Familie erhalten LLMs 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. Möglicherweises 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 GPT-4-Tokenizer steht aktuell nicht zur Verfügung, aber Sie können den GPT-3-Tokenizer (*https://platform.openai.com/tokenizer*) ausprobieren.

Als Faustregel kann man bei Tokens annehmen, dass 100 Tokens ungefähr 75 Wörtern in einem englischen Text entsprechen.

Dank des weiter oben vorgestellten 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 Kontext des Prompts am wahrscheinlichsten folgen werden. OpenAI bietet zwei Versionen von GPT-4, deren Kontextfenster 8.192 und 32.768 Tokens lang sind. 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.

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: »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.

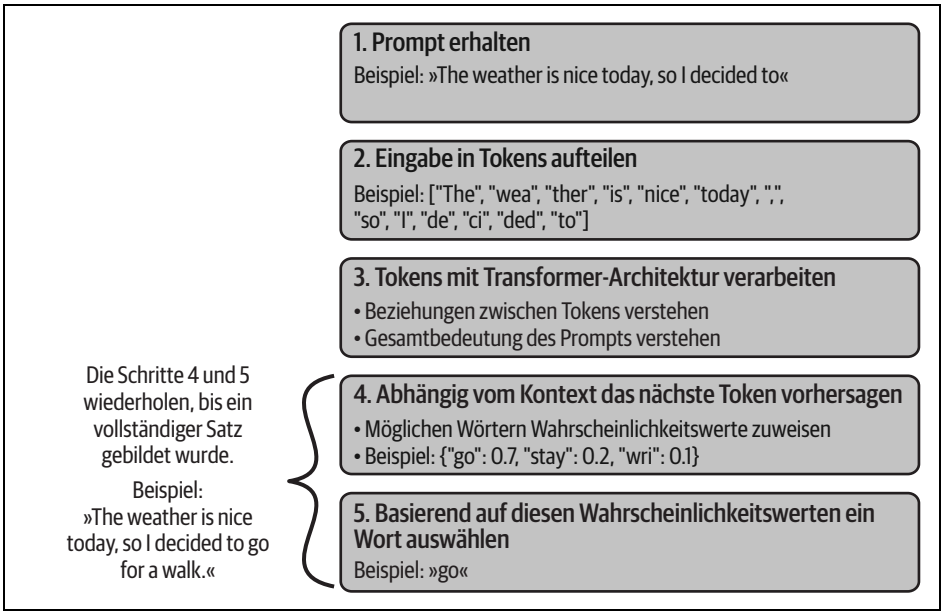


Abbildung 1-5: Der Vervollständigungsprozess läuft iterativ ab – Token für Token.

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

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

Inhalt

O'REILLY®

Anwendungen mit GPT-4 und ChatGPT entwickeln

Diese kompakte 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 ChatGPT und beschreiben, wie sie für eigene NLP-Aufgaben eingesetzt werden können. In nachvollziehbaren Schritten wird erläutert, wie Sie Applikationen zur Textgenerierung, Inhaltszusammenfassung oder für das Question Answering entwickeln.

Anschauliche Beispielprojekte und 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.

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 ChatGPT und deren Funktionsweise**
- **Die Integration dieser Modelle in Python-basierte Anwendungen für Aufgaben im Natural Language Processing**
- **Die praktischen Schritte, um mit den APIs von GPT-4 und ChatGPT und den entsprechenden Python-Bibliotheken Anwendungen zu entwickeln**
- **Spezifische Aspekte der Arbeit mit LLMs wie das API-Schlüsselmanagement, Datenschutz, Softwarearchitekturdesign oder die Gefahren durch Prompt Injection**
- **Fortgeschrittene GPT-Themen wie das Prompt Engineering, das Optimieren von Modellen, Plug-ins und der Einsatz des LangChain-Frameworks**

»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 »Generative AI« (Apress)

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 arbeitet als Softwarearchitektin und Data Engineer im Department R&D bei Worldline. Außerdem ist sie Developer Advocat und Tech-Speakerin.

9 783960 092414

www.dpunkt.de

Euro 32,90 (D)
ISBN 978-3-96009-241-4