

Large Language Models selbst programmieren

Mit Python und PyTorch ein eigenes LLM entwickeln

» Hier geht's
direkt
zum Buch

DAS VORWORT

Über dieses Buch

Large Language Models selbst programmieren wurde geschrieben, um Ihnen zu helfen, Ihre eigenen GPT-ähnlichen großen Sprachmodelle (*Large Language Models*, LLMs) von Grund auf zu verstehen und zu erstellen. Es beginnt bei den grundlegenden Arbeiten mit Textdaten und der Codierung von Attention-Mechanismen (Aufmerksamkeitsmechanismen) und führt Sie dann durch die Implementierung eines vollständigen GPT-Modells von Grund auf. Anschließend geht es um den Vortrainingsmechanismus sowie das Feintuning bei spezifischen Aufgaben wie Textklassifizierung und dem Befolgen von Anweisungen. Wenn Sie sich bis zum Ende des Buchs durchgearbeitet haben, werden Sie über profunde Kenntnisse zur Funktionsweise von LLMs verfügen und in der Lage sein, Ihre eigenen Modelle zu erstellen. Obwohl derartige Modelle im Vergleich zu den großen Grundlagenmodellen deutlich kleiner sind, verwenden sie dieselben Konzepte und dienen als leistungsstarke Lehrmittel, um die Kernmechanismen und -techniken zu verstehen, die in modernen LLMs zum Einsatz kommen.

Wer das Buch lesen sollte

Large Language Models selbst programmieren richtet sich an Enthusiasten des Machine Learning, also Ingenieure, Forscherinnen, Studenten und Praktikerinnen, die ein tiefes Verständnis von der Funktionsweise von LLMs erlangen möchten und lernen wollen, ihre eigenen Modelle von Grund auf zu erstellen. Sowohl Einsteiger als auch erfahrene Entwicklerinnen werden in der Lage sein, ihre vorhandenen Fähigkeiten und Kenntnisse zu nutzen, um die Konzepte und Techniken zu verstehen, die für die Erstellung von LLMs relevant sind.

Von anderen Büchern hebt sich dieses Buch dadurch ab, dass es umfassend den gesamten Prozess der LLM-Erstellung abdeckt, von der Arbeit mit Datensätzen bis zur Implementierung der Modellarchitektur, dem Vortraining mit ungelabelten Daten und der Feinabstimmung oder Optimierung für spezifische Aufgaben. Zur Entstehungszeit dieses Buchs gab es keine andere Quelle, die einen so vollständigen und praxisnahen Ansatz zur Erstellung von LLMs von Grund auf bietet.

Um die Codebeispiele in diesem Buch zu verstehen, sollten Sie über solide Kenntnisse in der Python-Programmierung verfügen. Vorteilhaft kann es sein, wenn Sie mit Machine Learning, Deep Learning und künstlicher Intelligenz schon etwas vertraut sind, wobei aber ein umfassendes Hintergrundwissen in diesen Bereichen nicht erforderlich ist. LLMs sind ein einzigartiger Teilbereich der KI, sodass Sie auch dann, wenn Sie auf diesem Gebiet relativ neu sind, problemlos dem Buch folgen können.

Falls Sie bereits Erfahrung mit tiefen neuronalen Netzen (*Deep Neural Networks*) haben, sind Ihnen bestimmte Konzepte vielleicht schon vertraut, da LLMs auf diesen Architekturen aufbauen. Die Beherrschung von PyTorch ist jedoch keine Voraussetzung. Anhang A bietet eine kurze Einführung in PyTorch, die Sie mit den notwendigen Fähigkeiten ausstattet, um die Codebeispiele im Buch zu verstehen.

Als hilfreich können sich auch Kenntnisse in höherer Mathematik erweisen, insbesondere im Umgang mit Vektoren und Matrizen, wenn wir die Funktionsweise von LLMs erkunden. Darüber hinausgehendes mathematisches Wissen ist jedoch nicht erforderlich, um die im Buch vorgestellten Schlüsselkonzepte und Ideen zu verstehen.

Die wichtigste Voraussetzung ist eine solide Grundlage in der Python-Programmierung. Mit diesen Kenntnissen sind Sie gut gerüstet, um die faszinierende Welt der LLMs zu erkunden und die Konzepte sowie die Codebeispiele im Buch in eigenen Projekten umsetzen zu können.

Wie das Buch aufgebaut ist: ein Wegweiser

Dieses Buch ist so konzipiert, dass Sie es sequenziell lesen sollten, da jedes Kapitel auf den Konzepten und Techniken aufbaut, die in den vorangegangenen Kapiteln eingeführt wurden. Gegliedert ist das Buch in sieben Kapitel, die die wesentlichen Aspekte von LLMs und deren Implementierung behandeln.

Kapitel 1 bietet eine umfassende Einführung in die grundlegenden Konzepte von LLMs. Es erläutert die Transformer-Architektur, die die Basis für LLMs bildet, wie sie beispielsweise auf der ChatGPT-Plattform realisiert sind.

Kapitel 2 legt einen Plan für den Aufbau eines LLM von Grund auf fest. Es beschreibt den Ablauf davon, wie der Text für das LLM-Training vorbereitet wird. Dazu gehört die Aufteilung des Texts in Wort- und Teilworttokens, die Verwendung der Bytepaar-Codierung für eine fortgeschrittene Tokenisierung, die Auswahl der Stichproben von Trainingsbeispielen mit einem Schiebefensteransatz und das Konvertieren von Tokens in Vektoren, die in das LLM eingespeist werden.

Kapitel 3 konzentriert sich auf die Attention-Mechanismen (die Aufmerksamkeitsmechanismen), die in LLMs verwendet werden. Es stellt ein grundlegen-

des Framework für Self-Attention vor und geht dann zu einem erweiterten Self-Attention-Mechanismus über. Außerdem behandelt das Kapitel die Implementierung eines kausalen Attention-Moduls, das LLMs in die Lage versetzt, einzelne Tokens nacheinander zu erzeugen, zufällig ausgewählte Attention-Gewichte mit Dropout zu maskieren, um Überanpassung zu verringern, und mehrere kausale Attention-Module in einem Multi-Head-Attention-Modul übereinanderzustapeln.

Der Schwerpunkt von Kapitel 4 ist die Codierung eines GPT-artigen LLM, das sich trainieren lässt, um Klartext zu erzeugen. Es beschreibt Techniken wie die Normalisierung von Schichtaktivierungen, um das Training neuronaler Netze zu stabilisieren, das Hinzufügen von Shortcut-Verbindungen in Deep Neural Networks (tiefen neuronalen Netzen), um Modelle effektiver zu trainieren, das Implementieren von Transformer-Blöcken, um GPT-Modelle verschiedener Größen zu erzeugen, und die Berechnung der Parameteranzahl und des Speicherbedarfs von GPT-Modellen.

Kapitel 5 implementiert den Vortrainingsprozess von LLMs. Hier erfahren Sie, wie Sie die Verluste von Trainings- und Validierungsmengen berechnen, um die Qualität des LLM-generierten Texts zu bewerten, wie Sie eine Trainingsfunktion implementieren und das LLM vortrainieren und wie Sie die Modellgewichte speichern und wieder laden, um das Training eines LLM fortzusetzen sowie vortrainierte Gewichte von OpenAI zu laden.

Kapitel 6 stellt verschiedene Ansätze für das Feintuning von LLMs vor. Es beschreibt, wie Sie einen Datensatz für die Textklassifizierung vorbereiten, ein vortrainiertes LLM zum Feintuning modifizieren, ein LLM feintunen, um Spam-Nachrichten zu identifizieren, und die Genauigkeit eines feingetunten LLM-Klassifizierers bewerten.

Kapitel 7 untersucht den Prozess des Feintunings von LLMs per Anweisung, die Organisation von Anweisungsdaten in Trainingsstapeln, das Laden eines vortrainierten LLM und dessen Feinabstimmung, um menschliche Anweisungen zu befolgen, das Extrahieren von LLM-generierten Antworten auf Anweisungen zur Bewertung und die Bewertung eines per Anweisung feingetunten LLM.

Über den Code

Damit Sie die Codebeispiele in diesem Buch möglichst einfach nachvollziehen können, finden Sie sie auf der Manning-Website unter <https://www.manning.com/books/build-a-large-language-model-from-scratch> und im Jupyter-Notebook-Format auf GitHub unter <https://github.com/rasbt/LLMs-from-scratch>. Und machen Sie sich keine Sorgen, wenn Sie nicht weiterkommen – die Lösungen zu allen Codeübungen finden Sie in Anhang C.

Dieses Buch enthält viele Beispiele für Quellcode sowohl in nummerierten Listings als auch im laufenden Text. In beiden Fällen ist der Quellcode in Schreibmaschinen-schrift formatiert, um ihn von normalem Text zu unterscheiden.

In vielen Fällen ist der ursprüngliche Quellcode neu formatiert worden. Es sind Zeilenumbrüche hinzugekommen und geänderte Einrückungen, um die Codezeilen an den Platz auf einer Druckseite anzupassen. Außerdem wurden oftmals die Kommentare im Quellcode aus den Listings entfernt, wenn der Text ohnehin den Code beschreibt. Codeanmerkungen sind in vielen Listings zu finden, um wichtige Konzepte hervorzuheben.

Eines der Hauptziele dieses Buchs ist die Zugänglichkeit, sodass Codebeispiele sorgfältig so gestaltet wurden, dass sie sich auf einem normalen Laptop effizient ausführen lassen, ohne dass eine spezielle Hardware erforderlich ist. Wenn Sie aber auf eine GPU zugreifen können, geben Ihnen bestimmte Abschnitte hilfreiche Tipps dazu, wie Sie die Datensätze und Modelle skalieren, um diese zusätzliche Leistung zu nutzen.

Das gesamte Buch hindurch verwenden wir PyTorch als Bibliothek für Tensor-Operationen und Deep-Learning-Routinen, um LLMs von Grund auf zu implementieren. Sollte PyTorch für Sie neu sein, empfehle ich, mit Anhang A zu beginnen, der eine ausführliche Einführung bietet und Empfehlungen für die Einrichtung gibt.

Andere Onlinere Ressourcen

Interessieren Sie sich für die neuesten Trends in der KI- und LLM-Forschung?

- Besuchen Sie mein Blog unter <https://magazine.sebastianraschka.com>, in dem ich regelmäßig über die neueste KI-Forschung mit Schwerpunkt auf LLMs diskutiere.

Benötigen Sie Hilfe, um sich schneller mit Deep Learning und PyTorch vertraut zu machen?

- Ich biete mehrere kostenlose Kurse auf meiner Website unter <https://sebastianraschka.com/teaching> an. Nutzen Sie diese Ressourcen, um Ihren Einstieg in diese Gebiete anzukurbeln.

Suchen Sie nach Bonusmaterialien zum Buch?

- Im GitHub-Repository des Buchs unter <https://github.com/rasbt/LLMs-from-scratch> finden Sie zusätzliche Ressourcen und Beispiele, die Ihr Lernen ergänzen.

Danksagungen

Ein Buch zu schreiben ist ein beträchtliches Unterfangen, und ich möchte meiner Frau Liza meinen aufrichtigen Dank für ihre Geduld und Unterstützung während dieses Prozesses aussprechen. Ihre bedingungslose Liebe und ständige Ermutigung waren unverzichtbar.

Unglaublich dankbar bin ich Daniel Kleine, dessen unschätzbares Feedback zu den entstehenden Kapiteln und zum Code meine Erwartungen übertroffen hat. Mit seinem scharfen Blick für Details und seinen aufschlussreichen Vorschlägen haben Daniels Beiträge zweifellos dazu beigetragen, dass dieses Buch zu einem entspannten und unterhaltsamen Leseerlebnis wird.

Ich möchte auch den wunderbaren Mitarbeitern von Manning Publications danken, darunter Michael Stephens für die vielen produktiven Diskussionen, die dazu beigetragen haben, die Ausrichtung dieses Buchs zu bestimmen, und Dustin Archibald, dessen konstruktives Feedback und dessen Anleitung zur Einhaltung der Manning-Richtlinien entscheidend waren. Ich weiß auch eure Flexibilität zu schätzen, mit der ihr den einzigartigen Anforderungen dieses unkonventionellen Ansatzes Rechnung getragen habt. Ein besonderer Dank gilt Aleksandar Drago-savljević, Kari Lucke und Mike Beady für ihre Arbeit an den professionellen Lay-outs und Susan Honeywell und ihrem Team für die Präzisierung und den Feinschliff der Grafiken.

Robin Campbell und ihrem hervorragenden Marketingteam möchte ich für ihre unschätzbare Unterstützung während des gesamten Schreibprozesses von ganzem Herzen danken.

Schließlich möchte ich mich bei den Gutachtern bedanken: Anandaganesh Balakrishnan, Anto Aravinth, Ayush Bihani, Bassam Ismail, Benjamin Muskalla, Bruno Sonnino, Christian Prokopp, Daniel Kleine, David Curran, Dibyendu Roy Chowdhury, Gary Pass, Georg Sommer, Giovanni Alzetta, Guillermo Alcántara, Jonathan Reeves, Kunal Ghosh, Nicolas Modrzyk, Paul Silisteanu, Raul Ciote-scu, Scott Ling, Sriram Macharla, Sumit Pal, Vahid Mirjalili, Vaijanath Rao und Walter Reade für ihr gründliches Feedback zu den Entwürfen. Ihre scharfen Augen und die aufschlussreichen Kommentare haben wesentlich dazu beigetragen, die Qualität dieses Buchs zu verbessern.

Allen, die an dieser Reise mitgewirkt haben, bin ich aufrichtig dankbar. Ihre Unterstützung, ihr Fachwissen und ihr Engagement haben einen entscheidenden Beitrag dazu geleistet, dass dieses Buch zustande gekommen ist. Ich danke euch!