In: H. Homann, C. Rohbani, J.C. Will (Hg.). KI-Forum 2024: AI 4 Students – AI in Research – AI Showroom: 2–4. HsH Applied Academics, Hannover, 2024.

Wie sollen wir KI lehren?

Volker Ahlers
Hochschule Hannover, Fakultät IV – Wirtschaft und Informatik, Abt. Informatik
Data|H – Institute for Applied Data Science Hannover
volker.ahlers@hs-hannover.de

***Zusammenfassung*—Auf die Frage, wie KI gelehrt werden soll, gibt es keine allgemeingültige Antwort. In dieser Keynote werden stattdessen einige Denkanstöße und Anregungen zur Diskussion gegeben. Zentrales Ziel sollte sein, den Studierenden die Funktionsweise von KI-Verfahren zu vermitteln, damit sie diese sinnvoll einsetzen und deren Grenzen einschätzen können.**

***Index Terms*—Hochschuldidaktik, Curriculum, Informatikstudium, Künstliche Intelligenz, KI, AI, GenAI**

I. Einleitung

Spätestens seit der Veröffentlichung von ChatGPT im November 2022 ist Künstliche Intelligenz (KI) ein auch in der breiten Öffentlichkeit präsentes und teils kontrovers diskutiertes Thema. Im Bereich der Hochschullehre werden insbesondere Auswirkungen der allgemein verfügbaren Werkzeuge Generativer KI (GenAI) auf traditionelle Prüfungsformen wie Hausarbeiten betrachtet, aber auch Chancen durch die Möglichkeit, individuelle Lehrinhalte oder Übungsaufgaben zu generieren.

In den *Leitlinien zur Nutzung von generativen KI-Anwendungen in der Lehre an der HsH* unterscheiden die Autorinnen zwischen den Gesichtspunkten „Generative KI als Lehrgegenstand“ und „Generative KI als neue Anforderung an Handlungskompetenz“ [1, S. 11]. Im Folgenden soll es um den ersten Gesichtspunkt gehen: Wie kann und soll generative, aber auch klassische KI an Hochschulen gelehrt werden, in Studiengängen der Informatik oder mit starkem Informatikbezug, aber auch in eher informatikfernen Studiengängen?

II. Empfehlungen der Fachgesellschaften

Blickt man in die aktuellen *Empfehlungen für Bachelor- und Master-Programme im Studienfach Informatik an Hochschulen* der Gesellschaft für Informatik (GI) [2], so fällt auf, dass das Thema Künstliche Intelligenz in den als Kognitive Kompetenzen bezeichneten Kernfächern nicht erwähnt wird. Lediglich einer der Musterstudiengänge im Anhang enthält ein nicht näher beschriebenes profilbildendes Fach *Intelligente Systeme* im Umfang von 6 Leistungspunkten (Credits). Dies ist sicher der Tatsache geschuldet, dass die Empfehlungen zuletzt 2016 aktualisiert wurden, als der weitreichende Einsatz von Methoden des Deep Learning gerade erst begann. Es darf davon ausgegangen werden, dass das Thema KI in der nächsten Aktualisierung eine prominentere Rolle erhalten wird.

In den *Computer Science Curricula* der amerikanischen Association for Computing Machinery (ACM) [3] schlägt sich diese Entwicklung bereits nieder: Waren für das Thema *Intelligent Systems* in der Version 2013 noch 10 Stunden im *Tier 2* Curriculum (fortgeschrittene Themen) vorgesehen, empfiehlt die aktuelle Version 2023 für die entsprechende *Knowledge Area Artificial Intelligence* 12 Stunden im *Computer Science Core* Curriculum plus 18 Stunden im *Knowledge Area Core* Curriculum; in ersterem entspricht dies knapp 4,5 % der vorgesehenen 270 Stunden [3, S. 37]. Das ACM-Curriculum gibt auch detaillierte Vorschläge für Unterthemen und Learning Outcomes des Lehrgebiets Künstliche Intelligenz. Aufschlussreich ist darüber hinaus eine an die Informatiklehre angepasste Version der Bloomschen Taxonomie: Angestrebt werden sollte mindestens das Kompetenzniveau *Evaluate* mit zugehörigen Verben wie *analyze, classify, assess*, besser noch das Niveau *Develop* mit Verben wie *compose, create, design* [3, S. 31].

III. KI im Masterstudiengang Angewandte Informatik der Hochschule Hannover

Während es in den Bachelorstudiengängen *B. Sc. Angewandte Informatik* und *B. Sc. Mediendesigninformatik* der Hochschule Hannover derzeit noch keine dedizierten KI-Module gibt und sich Studierende für Praxisprojekte und Abschlussarbeiten mit KI-Bezug im Selbststudium einarbeiten müssen, existieren im Schwerpunkt *Data Science* des Masterstudiengangs *M. Sc. Angewandte Informatik* insbesondere die Module Machine Learning und Deep Learning [4].

Im Modul Machine Learning werden zunächst klassische Verfahren der Künstlichen Intelligenz und des Maschinellen Lernens behandelt, bevor Neuronale Netze eingeführt werden:

- Decision Trees
- Naïve Bayes
- Model Evaluation
- Logistic Regression: Gradient Descent
- Feedforward Neural Networks: Backpropagation
- Optimization
- Regularization: Dropout, Augmentation
- Clustering
- Data Preprocessing: Feature Engineering, Dimensionality Reduction
- Recommender Systems: Collaborative Filtering
- Anomaly Detection

Das primäre Ziel besteht darin, dass die Studierenden die Funktionsweise der zugehörigen Algorithmen sowie deren Voraussetzungen und Grenzen verstehen. In den Übungen zur Vorlesung wird Python als Programmiersprache genutzt; dabei werden Code-Gerüste in Form von Jupyter Notebooks zur Verfügung gestellt.

DOI: https://doi.org/10.25968/opus-3449

7. The next function executes one step in stochastic or mini-batch gradient descent.

```
    def train(self, X, Y, epsilon):
        # X has shape (M, 784)
        # Y has shape (M, 10)

        # number of examples in mini-batch
        M = len(X)  # or X.shape[0], number of rows

        # forward pass
        (Z1, A1, Z2, A2) = self.forward(X)

        # backward pass
        # TODO: Compute the following matrices:
        # dZ2 (using A2 and Y)
        # dZ1 (using dZ2, W2, and Z1)
        # dW2 (using A1 and dZ2)
        # dW1 (using X and dZ1)
        # db2 (using dZ2)
        # db1 (using dZ1)

        # SGD step
        self.W2 -= epsilon * dW2
        self.b2 -= epsilon * db2
        self.W1 -= epsilon * dW1
        self.b1 -= epsilon * db1
```

Look up the necessary formulas in the lecture notes or slides and implement the missing parts.

Hint: To implement the colsum function, use `np.sum()` or `np.mean()` with an appropriate `axis` parameter and `keepdims=True` as explained above.

Abbildung 1. Ausschnitt eines Übungsblatts des Moduls Machine Learning im Schwerpunkt *Data Science* des Masterstudiengangs *M.Sc. Angewandte Informatik*; in dem Code-Gerüst soll das in der Vorlesung behandelte Backpropagation-Lernverfahren in Python und NumPy implementiert werden (Autor der ursprünglichen Aufgabe: Felix Heine).

Abbildung 1 zeigt einen Ausschnitt eines zentralen Übungsblatts, in welchem die Studierenden die Aufgabe haben, ein Neuronales Netz samt Backpropagation nur mit Hilfe von Python und NumPy, also ohne High-Level-KI-Bibliotheken wie Keras, TensorFlow oder PyTorch zu implementieren und auf den MNIST-Datensatz [5] zur Erkennung handgeschriebener Ziffern anzuwenden. Auf einem normalen Rechner ist mit dem Ansatz eine Genauigkeit (Accuracy) von $97,5\,\%$ erreichbar. In der nachfolgenden Übung werden Keras und TensorFlow eingesetzt, um die Genauigkeit mit Hilfe von Regularisierungstechniken zu verbessern. Parallel dazu soll das Backpropagation-Lernverfahren für ein einfaches Netz mit wenigen Neuronen auf dem Papier nachgerechnet werden.

Voraussetzungen für einen solchen Lehransatz sind grundlegende Mathematikkenntnisse, insb. in den Bereichen Lineare Algebra (Matrizenmultiplikation), Analysis (partielle Ableitungen, Kettenregel) und Stochastik (bedingte Wahrscheinlichkeit, Satz von Bayes), sowie Programmiererfahrung. Letztere muss nicht notwendigerweise in Python vorliegen: Erfahrungsgemäß arbeiten sich unsere (Master-)Studierenden schnell in die Sprache ein, auch wenn sie vorher andere Programmiersprachen genutzt haben.

Im Folgemodul Deep Learning werden aktuelle Verfahren der künstlichen Intelligenz behandelt:

- Fundamentals of Machine Learning
- Feedforward Neural Networks
- Convolutional Neural Networks (CNN)
- CNN Architectures
- Computer Vision: Visualization Techniques, Neural Style Transfer
- Transfer Learning
- Object Detection: Semantic/Instance Segmentation
- Recurrent Neural Networks (RNN)
- Word Embeddings
- Machine Translation
- Transformer Architecture

Die Übungsblätter liegen hier vollständig in Form von Jupyter Notebooks vor, die interaktiv bearbeitet werden können.

IV. Large Language Models und Transformer

In den letzten Jahren haben Large Language Models (LLM) erstaunliche Leistungen auf dem Gebiet der Generativen KI (GenAI) gezeigt, zunächst in der Verarbeitung natürlicher Sprache und der maschinellen Übersetzung, spätestens seit der Veröffentlichung von ChatGPT aber vor allem in der Generierung von Texten (sowie Bildern, Musikstücken, …). Grundlage dieser Entwicklungen ist die 2017 vorgestellte Transformer-Architektur, die im Kern auf einem Multi-Head-Attention-Mechanismus zur Erkennung weitreichender Zusammenhänge in Sätzen und Texten basiert [6]. Diese Architektur hat sich als flexibler, effizienter trainierbar und besser parallelisierbar als frühere Ansätze wie Recurrent Neural Networks (RNN) und Long Short-Term Memory (LSTM) erwiesen.

In der Lehre in informatiknahen Studiengängen sollte vermittelt werden, dass sich auch komplexe Verfahren wie die Transformer-Architektur verstehen lassen. Am nachhaltigsten gelingt dies sicher, indem die Studierenden ein eigenes Transformer-Modell programmieren. So ist es mit überschaubarem Aufwand möglich, eine verkleinerte Version des GPT-1-Modells [7] in Python mittels Keras und Tensorflow zu implementieren und auf dem eigenen Notebook für Spezialaufgaben wie die Generierung von Rezepten und Weinrezensionen zu trainieren [8], [9].

Dabei wird schnell deutlich, dass der Attention-Mechanismus Zusammenhänge innerhalb von Sätzen allein aus einer großen Menge von Trainingsdaten lernt, ohne formale Regeln der Grammatik zu kennen. Auch Word Embeddings, also die Einbettung der häufigsten Wörter (genauer Token) der Trainingsdaten in einem hochdimensionalen Vektorraum, werden allein aus den Trainingsdaten ohne jedes Verständnis der Semantik (Wortbedeutung) erzeugt.

In informatikfernen Studiengängen bieten gerade Word Embeddings Möglichkeiten, die Funktionsweise von Large Language Models mit Hilfe von Visualisierungen zu veranschaulichen. So wird in geeigneten 2D-Projektionen der hochdimensionalen Embeddings deutlich, dass sinnverwandte Wörter nah benachbart zueinander eingebettet werden. Auch für die durch den Attention-Mechanismus ermittelten Zusammenhänge zwischen den Wörtern eines Satzes existieren Visualisierungsansätze [10].

Für Informatikstudierende hingegen ist es durchaus möglich, die Originalliteratur zur Transformer-Architektur und zum GPT-1-Modell zu lesen [6], [7]. Mit dem entsprechenden

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, None)	0
token_and_position_embedding (TokenAndPositionEmbedding)	(None, None, 256)	2,580,480
transformer_block (TransformerBlock)	[(None, None, 256), (None, 2, None, None)]	395,776
dense_2 (Dense)	(None, None, 10000)	2,570,000

Total params: 5,546,256 (21.16 MB)
Trainable params: 5,546,256 (21.16 MB)
Non-trainable params: 0 (0.00 B)

Abbildung 2. Ausgabe der `summary()`-Methode des selbst erstellten Keras-GPT-Modells.

Wissen lässt sich die Anzahl der Parameter unseres verkleinerten GPT-Modells anhand der in Abbildung 2 dargestellten Ausgabe der `summary()`-Methode des Keras-Modells nachvollziehen:

- Embedding (256-dimensional) mit Positional Encoding (Context Window der Länge 80 Token):

$$(\underbrace{10\,000}_{\text{Vokabular}} + \underbrace{80}_{\text{Positionen}}) \times \underbrace{256}_{\text{Embedding}} = 2\,580\,480$$

- Transformer Block mit zwei Attention Heads, zwei Dense Layers und Modelldimension 256:

$$\underbrace{2 \times 3 \times (256 \times 128 + 128) + (256^2 + 256)}_{\text{Attention Layer}} + \underbrace{2 \times (256^2 + 256)}_{\text{Dense Layers}} + \underbrace{2 \times (2 \times 256)}_{\text{Layer Normalizations}} = 395\,776$$

- Dense Layer (Output Size 10 000):

$$\underbrace{10\,000 \times 256}_{\text{Gewichtsmatrix}} + \underbrace{10\,000}_{\text{Biasvektor}} = 2\,570\,000$$

In Summe ergeben sich gut 5,5 Millionen trainierbare Parameter (Gewichtsmatrizen, Bias- und Embedding-Vektoren). Dies genügt, um aus der Vorgabe eines Herkunftslandes grammatikalisch weitestgehend korrekte und einigermaßen plausibel klingende englischsprachige Weinrezensionen zu generieren.

Warum ist unser GPT-Modell in seiner Leistung dennoch weit von dem entfernt, was wir von ChatGPT und ähnlichen Modellen kennen? Neben dem fehlenden Finetuning für verschiedene Aufgabenstellungen (der Chat-Funktion) ist dies vor allem eine Frage der Skalierung: Tabelle I vergleicht die Dimensionen verschiedener GPT-Modelle. So hat das GPT-3-Modell eine etwa 30.000-fache Anzahl trainierbarer Parameter und einen gut 7 Mio.-fachen Umfang an Trainingsdaten im Vergleich zu „unserem“ GPT-Modell (ab GPT-4 hat OpenAI keine entsprechenden technischen Details mehr veröffentlicht).

V. Zusammenfassung

Wie also sollen wir KI lehren? In erster Linie sollte vermittelt werden, dass KI-Verfahren keine Black Boxes sind, sondern auf – gar nicht so komplizierten – mathematischen und statistischen Prinzipien basierende Algorithmen sind. In informatiknahen Studiengängen sollten die Studierenden nicht nur den Einsatz von High-Level-KI-Bibliotheken lernen, sondern Algorithmen auch selbst programmieren, um deren Funktionsweisen und Grenzen zu verstehen. In informatikfernen Studiengängen können Visualisierungen zum Verständnis beitragen. In beiden Fällen sollten Chancen und Gefahren des Einsatzes von KI abgewogen sowie Methoden der erklärbaren KI behandelt werden.

Tabelle I
Hyperparameter von GPT-Modellen, nach [8].

Modell	Attention Blocks/Heads	Embedding Dimension	Anzahl Parameter	Training Data
unser GPT	1/2	256	5,5 Mio.	80 MB
GPT-1	12/12	768	120 Mio.	4,5 GB
GPT-2	48/48	1 600	1,5 Mrd.	40 GB
GPT-3	96/96	12 888	175 Mrd.	570 GB

Danksagung

Der Autor dankt Felix Heine und Adrian Pigors für die gemeinsame Lehre und viele anregende Diskussionen im Masterschwerpunkt Data Science. Ein Großteil der Inhalte des Moduls Machine Learning basiert auf Vorarbeiten von Felix Heine. Das Modul Deep Learning wurde von Felix Heine, Adrian Pigors und Volker Ahlers gemeinsam konzipiert.

Literatur

[1] C. Eube und T. Rasche, „Leitlinien zur Nutzung von generativen KI-Anwendungen in der Lehre an der HsH (Mai 2024, Version 1.0),“ Hochschule Hannover, 2024. https://zenodo.org/records/13747532

[2] O. Zukunft, „Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (Juli 2016),“ Gesellschaft für Informatik, 2016. https://dl.gi.de/items/0986c100-a3b9-47c8-8173-54c16d16c24e

[3] A. N. Kumar et al., „Computer Science Curricula 2023“, Association for Computing Machinery, 2024. https://doi.org/10.1145/3664191

[4] „Angewandte Informatik (MIN)“, Hochschule Hannover, Fakultät IV, 2024. https://f4.hs-hannover.de/studium/master-studiengaenge/angewandte-informatik-min

[5] Y. LeCun, C. Cortes, C. J. C. Burges, „The MNIST database of handwritten digits“. https://yann.lecun.com/exdb/mnist/

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser und I. Polosukhin, „Attention is all you need“, in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*, S. 6000–6010, Curran Associates, 2017. https://doi.org/10.48550/arXiv.1706.03762

[7] A. Radford, K. Narasimhan, T. Salimans und I. Sutskever, „Improving language understanding by generative pre-training,“ preprint, OpenAI, 2018. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[8] D. Foster, „Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play,“ 2. Aufl., O'Reilly Media, 2023. https://github.com/davidADSP/Generative_Deep_Learning_2nd_Edition

[9] A. Nandan, „Text generation with a miniature GPT,“ tutorial, Keras, 2020. https://keras.io/examples/generative/text_generation_with_miniature_gpt/

[10] J. Vig, „A multiscale visualization of attention in the transformer model,“ in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, S. 37–42, Association for Computational Linguistics, 2019. https://doi.org/10.18653/v1/P19-3007, https://github.com/jessevig/bertviz