

Computergenerierter Zufall als kreatives Moment in Malerei und Literatur

Potentiale und Grenzen von Machine-Learning-Modellen am Beispiel von GPT

Andreas de Vries (Informatik)

1 Einleitung

Und so ein Hirn, das trefflich denken soll,
wird künftig auch ein Denker machen.

Entzückt die Phiole betrachtend

J. W. von Goethe (1831), Faust II, 6869–6870

Goethe wusste es schon vor 200 Jahren: Der Mensch wird künstliche Intelligenzen erschaffen. Allerdings existieren die heutigen Systeme künstlicher Intelligenz nicht in einer Phiole, sondern in luftgekühlten und auf Platinen gesteckten Mikroprozessoren. Auch wissen wir nicht, ob bei ihrer Entstehung Mephisto anwesend war. Kluge Köpfe wie Stephen Hawking zumindest scheinen davon ausgegangen zu sein.¹

Künstliche Intelligenz – meist kurz: KI, oder englisch: AI – wird in mehr und mehr Bereichen des wissenschaftlichen und gesellschaftlichen Lebens eingesetzt. Im Sinne dieser Anwendungen ist KI also als eine Hilfstechnologie zu betrachten. Nun hat nach McLuhan jede Technologie auch Auswirkungen auf die kognitive und die soziale Organisation der Menschen, die sie nutzen.² Sie erhebt sich also immer über den Charakter einer Anwendung und gestaltet Strukturen.

Gestaltet KI auch Kunst, Musik, Poesie? Spätestens mit den Werken von Refik Anadol³ ist es so, bei denen man sich sogar fragt: Wer hat sie eigentlich erschaffen, der Künstler oder die Software? Die grundsätzliche Frage lautet also: Kann KI Kunst erschaffen?

1 <https://www.telegraph.co.uk/news/science/science-news/12107623/Prof-Stephen-Hawking-disaster-on-planet-Earth-is-a-near-certainty.html>, <https://www.bbc.com/news/technology-37713629>

2 Vgl. McLuhan (1964), S. 7.

3 Refik Anadol, türkisch-amerikanischer Medienkünstler und Architekt, <https://refikanadol.com/>, https://de.wikipedia.org/wiki/Refik_Anadol

Diese Frage ist zu groß, als dass dieser Beitrag dafür eine Antwort finden könnte. Aber zumindest einem wichtigen Aspekt dieser Frage wird er nachgehen: Wie kann Software Kreativität beeinflussen? Die Hypothese lautet: Es ist das durch Software gezielt eingebrachte Moment des Zufalls, das zur Kreativität beisteuern kann. Ob Software damit kreativ *ist*, bleibt offen. Computergenerierter Zufall spielt eine wesentliche Rolle in so unterschiedlichen Bereichen wie der Malerei von Gerhard Richter und der Erzeugung von Texten durch ChatGPT.

Dieser Beitrag gliedert sich wie folgt. Zunächst wird in Abschnitt 2 eine Werktechnik des Malers Gerhard Richter betrachtet, die bewusst computergenerierten Zufall einsetzt und eine erste Einordnung der Rollen von Software und Mensch bei der Schaffung von Kunstwerken zulässt. Nach einem einführenden Überblick über Grundlagen und grundsätzliche Funktionsweisen aktuell eingesetzter KI-Systeme in Abschnitt 3 und einer Darstellung von Methoden des *Neural Language Processing* in Abschnitt 4 wird in Abschnitt 5 das Sprachmodell GPT anhand von Beispieldialogen untersucht und wesentliche Merkmale, Grenzen und Potentiale aktuell verfügbarer Sprachmodelle aufgezeigt. Abschließend wird in Abschnitt 6 die These diskutiert, dass die wahrgenommene Kreativität von Sprachmodellen durch die zugrunde liegenden statistischen Wahrscheinlichkeitsverteilungen entsteht und damit dieselbe Ursache wie schon ein halbes Jahrhundert zuvor bei Richters Farbtafeln hat: den computergenerierten Zufall.

2 Zufall als Werktechnik in der Malerei Gerhard Richters

Gerhard Richter gehört zu den bedeutendsten Künstlern des frühen 21. Jahrhunderts. In seinem umfangreichen Werk experimentierte er mit verschiedenen, oft neuartigen Werktechniken. Im Februar 1961, wenige Monate vor dem Bau der Mauer, aus Dresden über Westberlin nach Düsseldorf geflohen, entwickelte er, frei von Zensur oder Bestimmung, in den folgenden Jahren fast das gesamte Spektrum an Methoden und Motiven seines Œuvres: Gemälde nach Fotos, Farb- und Grauschlieren, graue monochrome Tafeln und gerasterte Farbtafeln.⁴ Nur die ebenso bekannten, mit einem Rakel gemalten abstrakten Bilder entstanden erst später, nämlich etwa ab den 1980er Jahren.⁵

1966 malte Richter seine ersten Farbtafeln, indem er handelsübliche Musterkarten aus Farbfachgeschäften als Vorlage abmalte und in einem gleichmäßigen Gitter per Zufall anordnete. Diese Raster bestanden dabei mal aus gleichen Rechtecken, mal aus Quadraten. Aus dieser Phase sind 16 Bilder dokumentiert.⁶

In einer zweiten Phase malte Richter 1971 24 weitere Farbraster. Von 1973 bis 1974 entstanden noch einmal 26 Bilder, unter anderem die Werke *1024 Farben*⁷ und *4096 Farben*⁸, in denen aus einer vorgegebenen Farbpalette Einzelquadrate per Zufall eingefärbt und

4 Vgl. Godfrey & Serota (2012), S. 86f; Goebel (2010), S. 13f.

5 Camille Morineau in: Godfrey & Serota (2012), S. 123.

6 <https://www.gerhard-richter.com/de/art/paintings/abstracts/colour-charts-12/>

7 Richter WVZ 356–3, <https://www.gerhard-richter.com/de/art/paintings/abstracts/colour-charts-12/1024-colours-6083/>

8 Richter WVZ 359, <https://gerhard-richter.com/de/art/paintings/abstracts/colour-charts-12/4096-colours-6089/>

zu einem quadratischen Raster angeordnet wurden. Bei den Bildern dieser Phase taucht jede Farbe mehrfach auf. Richter scheint spätestens hier erstmals den Computer zur zufälligen Farbauswahl verwendet zu haben.⁹ Im Jahr 2007 entstand mit dieser Technik das Bild *4900 Farben*¹⁰ und das monumentale Richter-Fenster im Südquerhausfenster des Kölner Doms mit insgesamt etwa 11.500 Einzelquadraten aus einer Farbpalette von 72 Farbtönen.¹¹ Nur die Auswahl der Farben der einen Hälfte des Fensters wird durch ein Computerprogramm per Zufall erzeugt, die andere Hälfte wird spiegelsymmetrisch dazu angeordnet.¹²

Wer aber hat denn nun die Bilder dieser Phase oder das Richter-Fenster am Kölner Dom erschaffen? Ein Programm oder der Künstler? Richter betonte oft, dass der Zufall in seinem Werk als kreatives Moment eine wichtige Rolle spiele. »Zufall als Thema und Methode. Methode, um etwas Objektives entstehen zu lassen, Thema, um ein Gleichnis (Bild) zu schaffen für unsere Überlebensstrategie.«¹³ »Etwas entstehen lassen, anstatt kreieren; also keine Behauptungen, Konstruktionen, Erstellungen, Ideologien – um so an das Eigentliche, Reichere, Lebendigere heranzukommen, an das, was über dem Verstand ist ... Jetzt die Einbeziehung des Zufalls (wiederum nie Automatismus).«¹⁴ »Es war für mich interessant, den Zufall in eine ganz starre Ordnung zu bringen.«¹⁵ Der Zufall in Richters Farbtafeln ist nicht beliebig, sondern tritt in einem streng vorgegebenen Rahmen ein: Ein festes Raster ist gegeben, im Domfenster sogar strenge Spiegelsymmetrien, ebenso die Palette der möglichen Farben; lediglich die Farbauswahl selbst ist Produkt des Zufalls.

Die Anordnung der Farbtöne auf den Feldern erfolgte per Zufall, um eine diffuse, gleichgültige Gesamtwirkung zu erzielen, während das Detail anregend sein kann. Das starre Raster verhindert Figurationen, obwohl diese mit Anstrengung sichtbar werden können. Diese Art von künstlichem Naturalismus ist ein Aspekt, der mich fasziniert wie die Tatsache, dass, wenn ich alle möglichen Permutationen gemalt hätte, das Licht über 400 Billionen Jahre bräuchte, um vom ersten bis zum letzten Bild zu kommen.¹⁶

Ein Nebenaspekt aus informatischer Sicht: Echten Zufall kann ein Computer nicht erzeugen. Der von einem Computer generierte Zufall ist nur ein Pseudozufall, also simuliert. Der Grund ist, dass ein Computer nur deterministisch arbeiten kann. Für den simulierten Zufall werden Pseudozufallszahlengeneratoren (PRNG) verwendet, also Algorithmen, die Folgen von zufällig aussehenden Zahlen generieren, deren Werte von ei-

9 Vgl. Museum Folkwang (2017), S. 24f., 28ff.

10 Richter WVZ 901, <https://www.gerhard-richter.com/de/art/paintings/abstracts/colour-charts-12/4900-colours-14891/>

11 Vgl. Museum Ludwig & Metropolitankapitel der Hohen Domkirche Köln (2017), S. 28, 40f., 48.

12 Vgl. ebd., S. 7, 48.

13 Richter »Notiz 3.11.89«, zit.n. Pelzer in: Museum Ludwig & Metropolitankapitel der Hohen Domkirche Köln (2017), S. 67.

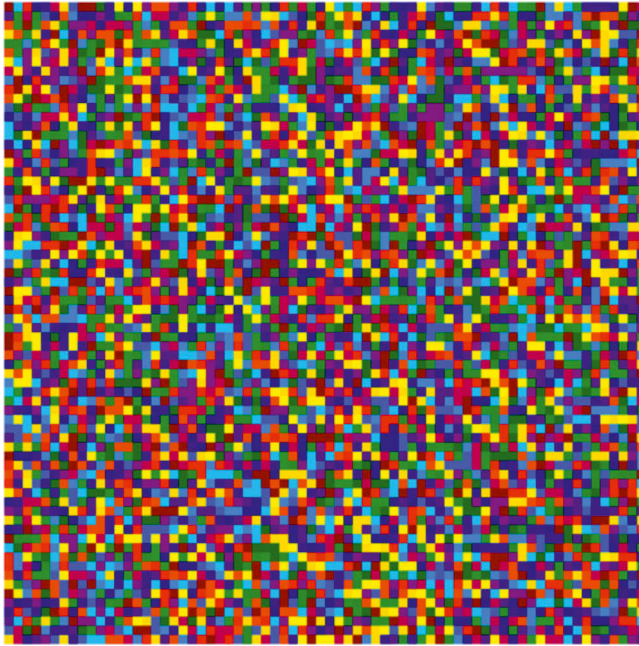
14 Richter 1985, zit.n. Pelzer (2017), S. 68.

15 Richter 1993, zit.n. Pelzer (2017), S. 74.

16 Richter 1993, zit.n. Pelzer (2017), S. 77.

nem Startwert («Seed») abhängen, der sich wiederum typischerweise aus der aktuellen Systemzeit des Computers ergibt und daher verschieden von praktisch allen vorher bestimmten ist.

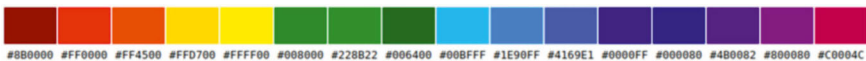
Abb. 1: 4900 Farben nach Gerhard Richter. Kunstwerk oder Plagiat?



Quelle: eigene Darstellung, modifiziert nach <https://www.gerhard-richter.com/de/art/paintings/abstracts/colour-charts-12/4900-colours-14891/>

Da jede der gängigen Programmiersprachen einen solchen Pseudozufallszahlengenerator bereitstellt, ist es nicht schwer, ein Programm zu schreiben, das Richters Werktechnik implementiert und Farben aus einer vorgegebenen Farbpalette in einem Raster per Zufall anordnet. In Abbildung 1 ist die Ausgabe Produkt eines solchen Programms dargestellt, das frei verfügbar unter <https://vriesa.github.io/arts/Richter.html> ist. Es basiert auf einer Palette von 16 Farben, die in Abbildung 2 dargestellt ist.

Abb. 2: Die Abbildung 1 zugrunde liegende Palette von 16 Farben und deren RGB-Codes



Quelle: eigene Darstellung

Hier stellt sich die Frage noch viel mehr: Wer konkret hat das Werk in Abbildung 1 geschaffen? Der Computer, der Programmierer, oder gar Gerhard Richter? Ist das nun Kunst oder ein Plagiat?¹⁷

3 Künstliche Intelligenz im Überblick

Ein KI-System ist eine Software, die automatisiert – »maschinell« – lernen kann. Dabei konnten schon zum Teil spektakuläre Ergebnisse erreicht werden:

- Siege in Brett- oder Computerspielen gegen menschliche Weltklassenspieler, z. B. AlphaGo im März 2016 gegen den Koreaner Lee Sedol im Brettspiel Go.¹⁸
- Erschaffung künstlerischer Werke, z. B. von klassischer Musik¹⁹, Gemälden²⁰, aber auch von fiktiven Texten²¹.
- Automatische Prüfung von Buchmanuskripten darauf hin, ob sie das Potential für Bestseller haben (z. B. mit QualiFiction).²²
- *Deep Fake*: echt wirkende Animationen bekannter Persönlichkeiten, z. B. des britischen Premiers Johnson²³.

Angesichts dieser verblüffenden, scheinbar kognitiven Leistungen von Computerprogrammen sagen einige Stimmen bereits die »technologische Singularität« voraus, d. h. den Augenblick, ab dem eine Maschine intelligenter als die Menschheit insgesamt und uns damit weit überlegen sein wird.²⁴ Diese These ist allerdings durchaus umstritten. Sie basiert auf dem Begriff einer »starken KI«, also eines Computerprogramms, das »wirklich« intelligent ist und mindestens auf Augenhöhe mit dem Menschen schwierige Aufgaben übernehmen kann. Demgegenüber gehören jedoch alle bisher funktionierenden Systeme zur sogenannten »schwachen KI«, die auf konkrete Problembereiche spezialisiert sind.²⁵

17 Die Frage ist an dieser Stelle selbstverständlich überspitzt formuliert. Natürlich handelt es sich nicht um ein Plagiat im strengen Wortsinne, denn durch die Zufälligkeit ist genau diese Anordnung der 4900 Farben sicher einzigartig. Im Sinne eines Geschmacksmusters allerdings sieht es etwas anders aus. Der Verfasser erklärt daher ausdrücklich, hier eine Werktechnik von Gerhard Richter zur reinen Illustration und Verdeutlichung tief liegender Fragen eingesetzt zu haben und erhebt keinen künstlerischen Anspruch oder erkennt hier eine nennenswerte Schöpfungstiefe.

18 https://de.wikipedia.org/wiki/AlphaGo_gegen_Lee_Sedol

19 <https://youtu.be/H6Z2n7BhMPY>

20 <https://www.christies.com/lot/lot-edmond-de-belamy-from-la-famille-de-6166184/>

21 <https://watt-ai.github.io/demos/gpt2> (frei verfügbare Demoversion; es gibt allerdings aktuellere und deutlich leistungsfähigere Programme)

22 Nina George (2019), buchszone.de, 21. November 2019 <https://buchszone.de/kuenstliche-intelligente-essay-nina-george/>

23 <https://youtu.be/30NvDC1zcl8>: Johnson empfiehlt seinen damaligen politischen Gegner James Corbyn für das Amt des Premierministers; siehe auch <https://futureadvocacy.com/deepfakes/>

24 https://de.wikipedia.org/wiki/Technologische_Singularität

25 Vgl. Russell & Norvig (2022), S. 1032.

Vertreter:innen der schwachen KI haben gar nicht den Anspruch oder auch nur die Erwartung, dass Computersysteme »intelligent« sein oder werden könnten.²⁶ Dennoch werden solche Systeme in ihren spezialisierten Aufgabenbereichen immer komplexere Probleme lösen, für die bislang Intelligenz oder Erfahrung nötig war. In diesem Sinn stehen wir daher in der Tat erwartbar vor einer technologischen Revolution: der *Mechanisierung geistiger Arbeit*, ähnlich wie nach Erfindung der Dampfmaschine durch James Watt die Mechanisierung körperlicher Arbeit und damit die Industrialisierung begann. Einige aktuell erfolgreiche Einsatzgebiete von KI sind:

- *Computer Vision*: Objekterkennung (z.B. Verkehrszeichen für Fahrassistenzsysteme), Gesichtserkennung (z.B. Identifikation beim Flugzeugboarding), Nachverfolgung von Objekten in Bewegungsabläufen (z.B. räumliche Zuordnung von Fahrzeugen und Personen im Verkehrsraum).
- *Verarbeitung natürlicher Sprache (Natural Language Processing, NLP)*: maschinelle Übersetzungen (Google Translate, DeepL), akustische Spracherkennung (Siri, Alexa, Cortana, Google Assistant).
- *Wissenschaft*: Medizin (Krebserkennung, Tomographie-Auswertungen, Pflegeroboter), Astronomie (Gravitationswellenmessungen LIGO 2017), Mathematik (Theorembeweiser, Lösen partieller Differentialgleichungen).
- *Text-Mining*: Textsuche nach ungenau oder falsch eingegebenen Suchbegriffen.
- *Data Mining*: Erkennen von Zusammenhängen in großen Datenbeständen (*Big Data*).

3.1 Lernarten von KI-Systemen

Abstrakt betrachtet ist ein KI-System ein statistisches Modell, das »lernt«, indem es seine Modellparameter an Beobachtungsdaten der Außenwelt anpasst. Das geschieht entweder anhand gegebener Trainingsdaten oder anhand einer Optimierungsfunktion. Es gibt drei wesentliche Arten des Lernens:

- *Überwachtes Lernen*: Die Trainingsdaten enthalten bereits die korrekten Ergebnisse (»Zielwerte«), vergleichbar dem Lernen für eine Klausur mit Musterlösungen; Beispiele sind lineare und nichtlineare Regressionen, Zeitreihenanalysen, Klassifikationen.
- *Unüberwachtes Lernen*: Das System stellt die Parameter selbständig ein, es gibt keinen »Trainer« oder »Lehrer«, der sagt, was herauskommen soll; Beispiele sind Klassifikationen (»Bayes-Klassifizierer«) oder Clusterungen (»k-means-Algorithmus«).
- *Bestärkendes Lernen (Reinforcement Learning)*: Das System lernt selbständig anhand einer reellwertigen Erfolgsfunktion, mit der immer wieder variierte Strategien durch Abgleich mit dem jeweils erreichten Ergebnis bewertet werden und so die beste Strategie ermittelt wird. Dabei kann sich sogar die ganze Konfiguration des Modells (die Einstellung der »Hyperparameter«) ändern. Beispiele dieses Lernansatzes sind

26 Damit sparen sie sich gewissermaßen auch die schwierige Aufgabe, den Begriff Intelligenz präzise zu definieren.

Deep-Learning-Netze für Brettspiele (z.B. AlphaZero²⁷), die anhand der gegebenen Spielregeln in der Trainingsphase immer wieder gegen sich selbst spielen und sich so optimieren.

Die meisten der heute verwendeten KI-Systeme sind selbstlernend, lernen also unüberwacht oder bestärkend. Oft wird aber auch eine Mischung der Lernarten angewandt, indem ein KI-System mit überwachtem Lernen vortrainiert wird, bevor es dann veröffentlicht wird und dann unüberwacht weiter lernt.

Oft aber lernen KI-Systeme auch nach ihrer Veröffentlichung und im alltäglichen Gebrauch überwacht weiter. Zu erkennen ist dies daran, dass der Anwender eine korrekte Lösung eingeben kann. Beispiele dafür sind Spamfilter, die durch die Nutzermarkierungen lernen, welche Mail Spam ist und welche nicht, oder die Gesichtserkennung in den Fotoverwaltungsprogrammen, in denen den Gesichtern Namen zugeordnet werden können.

Ein speziell für linguistische Anwendungen gebräuchliche Testmethode eines trainierten Modells ist das *Zero-Shot Learning*. Hier wird geprüft, ob das System korrekte Beispiele als korrekt klassifiziert, die nicht in der Trainingsmenge waren. Für Systeme der Textgenerierung ist ein gutes Zero-Shot-Lernverhalten wesentlich, um nicht nur die eingegebenen Trainingsdaten zu wiederholen, sondern auch »neue«, aber korrekte Sätze zu erzeugen.

3.2 Künstliche neuronale Netze

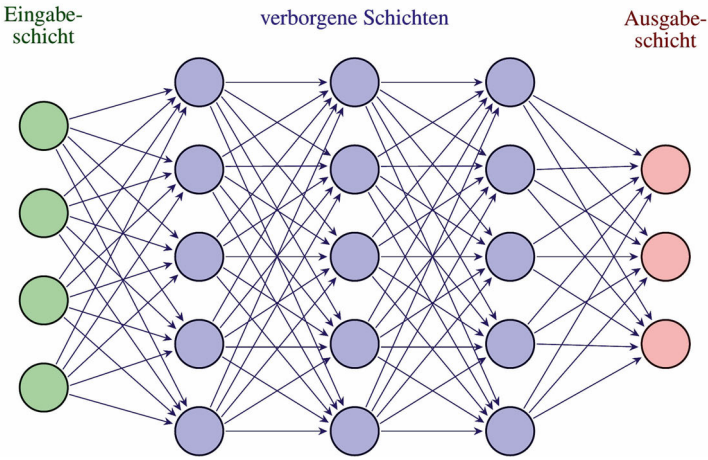
Das flexibelste der aktuell verwendeten KI-Systeme sind künstliche neuronale Netze. Sie sind dem menschlichen Gehirn nachempfunden und setzen sich aus »Neuronen« genannten Recheneinheiten zusammen. Im Gegensatz zu biologischen Gehirnen jedoch hat ein künstliches neuronales Netz in der Regel eine deutlich einfachere, in Schichten und oft hierarchisch strukturierte Netzwerktopologie, wie in Abbildung 3 skizziert. Es gibt eine Eingabeschicht, verborgene innere Schichten und eine Ausgabeschicht.

Die Eingabeschicht besteht aus Neuronen, die Signale von außen aufnehmen und an die verknüpften Nachbarneuronen weitergeben. Die inneren Neuronen empfangen Signale von verbundenen Eingabeneuronen und berechnen daraus das Signal, das sie an die verknüpften Neuronen senden. Die Neuronen der Ausgabeschicht schließlich geben ihre Signale als Ergebnis der Verarbeitungen nach außen ab.

Ein einzelnes Neuron ist schematisch in Abbildung 4 dargestellt. In der Regel werden alle Neuronen eines neuronalen Netzes durchnummeriert, hier hat es die Nummer *i*.

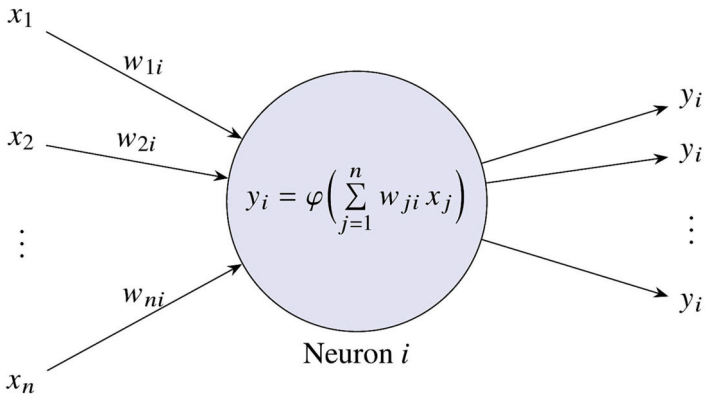
27 <https://de.wikipedia.org/wiki/AlphaZero>

Abb. 3: Netzwerktopologie eines typischen neuronalen Netzes



Quelle: eigene Darstellung

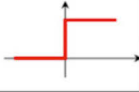
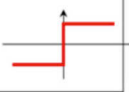
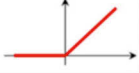
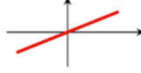
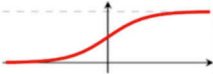
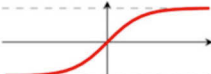
Abb. 4: Schematische Darstellung eines künstlichen Neurons



Quelle: eigene Darstellung

Es erhält die Signale x_1, \dots, x_n seiner eingehenden Neuronen mit individuellen Verstärkungs- oder Dämpfungsfaktoren w_{1i}, \dots, w_{ni} , den »Gewichten«. Das Neuron berechnet die mit diesen Faktoren gewichtete Summe der Eingangssignale und wendet darauf eine spezifische Aktivierungsfunktion ϕ an. Der Ergebniswert y_i dieser Berechnung wird nun als Ausgangssignal an die Nachbarneuronen gesendet. Die Aktivierungsfunktion wirkt also als digitaler Schalter, der bei ausreichend hohem Eingabewert auf Eins – also auf »An« – schaltet: Das Neuron »feuert«. Die gewichtete Summe kann kurz als Vektorgleichung $\mathbf{w}_i^T \mathbf{x}$ geschrieben werden, wobei \mathbf{w}_i^T der aus den Gewichten w_{1i}, \dots, w_{ni} gebildete Zeilenvektor ist. Gängige Aktivierungsfunktionen sind in der Übersicht in Abbildung 5 angegeben.

Abb. 5: Gängige Aktivierungsfunktionen eines künstlichen Neurons²⁸

Heaviside- oder Stufenfunktion: $\varphi(x) = \begin{cases} 0 & \text{wenn } x < 0 \\ 1 & \text{wenn } x \geq 0 \end{cases}$ 	Vorzeichenfunktion (signum, sgn): $\varphi(x) = \begin{cases} -1 & \text{wenn } x < 0 \\ 0 & \text{wenn } x = 0 \\ 1 & \text{wenn } x > 0 \end{cases}$ 
ReLU (rectified linear unit): $\varphi(x) = \begin{cases} 0 & \text{wenn } x < 0 \\ x & \text{wenn } x \geq 0 \end{cases}$ 	Lineare Funktion: $\varphi(x) = x$ 
Sigmoid oder logistische Funktion: $\varphi(x) = \frac{1}{1 + e^{-x}}$ 	Tangens hyperbolicus: $\varphi(x) = \tanh x$ 

Die letzten drei dieser Funktionen sind differenzierbar:

linear: $\varphi'(x) = 1$, logistisch: $\varphi'(x) = \varphi(x) \cdot (1 - \varphi(x))$, tanh: $\varphi'(x) = 1 - \varphi^2(x)$.
--

Quelle: Patterson & Gibson (2017), S. 65ff., Russell & Norvig (2022), S. 803f. (eigene Darstellung)

Die linke Spalte zeigt dabei Funktionen, die für negative Werte des Arguments x verschwinden oder zumindest schnell gegen Null konvergieren, für positive Werte aber – plötzlich oder allmählich – auf Eins hochgehen. Die rechte Spalte zeigt Aktivierungsfunktionen, die punktsymmetrisch um den Ursprung Werte zwischen -1 und 1 annehmen. Die letzten drei Funktionen, zeilenweise gezählt, sind differenzierbar, und ihre Ableitungen sind sehr effizient berechenbar, da sie entweder konstant sind – wie die lineare Funktion – oder sich durch einen einfachen Term aus der Funktion selber ergeben. Das macht sie aus Sicht der Informatik sehr interessant.

In dem zur Textgenerierung verwendeten Sprachmodell GPT wird als Aktivierungsfunktion GELU verwendet, für *Gaussian Error Linear Unit*. Sie ist eine differenzierbare Variante der ReLU-Funktion und liefert empirisch signifikant bessere Werte.²⁹

Welche der Aktivierungsfunktionen in einem neuronalen Netz verwendet wird, hängt von dem zugrunde liegenden Anwendungsfall ab. Oft wird die ReLU-Funktion angewandt, oder wenn Differenzierbarkeit gefordert wird, die logistische Funktion (im Englischen häufig *sigmoid* genannt).

Oft soll ein Neuron erst feuern, wenn die gewichtete Summe der Eingangssignale einen gewissen Schwellwert b überschreitet. Der Variablenname b steht für »Bias« und ist in der Regel positiv. Die Aktivierungsfunktion soll dann den Term $\phi(\mathbf{w}_i^T \mathbf{x} - b_i)$ mit der Zahl b_i als Bias berechnen. Das kann durch eine Modifikation der Aktivierungsfunktion

28 Vgl. Patterson & Gibson (2017), S. 65ff., Russell & Norvig (2022), S. 803f.

29 Vgl. Hendrycks & Gimpel (2016).

selber geschehen, üblicherweise werden dazu aber »Bias-Neuronen« eingeführt, die den Neuronen des Netzes das Eingangssignal $w_{0j} = -b_j$ liefern, also ständig feuern.³⁰

Das historisch erste künstliche neuronale Netz ist das Perzeptron von Frank Rosenblatt.³¹ Es hat die Stufenfunktion als Aktivierungsfunktion und einen Bias von 0,5. Rosenblatts ursprüngliche Variante von 1958 hatte keine verborgene Schicht, sondern bestand nur aus Ein- und Ausgabeneuronen.

3.3 Deep Learning

Ein neuronales Netz ist als mathematisches Modell nichtlinear, wenn seine Aktivierungsfunktionen nichtlinear sind. Wenn alle eingesetzten Aktivierungsfunktionen linear sind, kann das Netz als multilinearer Tensor dargestellt werden.³² Es ist jedoch erst die Nichtlinearität, die neuronale Netze für die meisten Anwendungsfälle interessant macht.³³

Die freien Parameter eines solchen Modells sind die einzelnen Gewichte der Neuronenverknüpfungen. Ein neuronales Netz lernt also, indem seine Gewichte so an die Trainingsdaten angepasst werden, dass der Fehler der Trainingsausgaben gegenüber den korrekten Ausgaben minimiert wird, oder dass – im Fall des bestärkenden Lernens – die Zielfunktion maximiert wird.

Neuronale Netze können auf alle drei oben erwähnten Arten lernen, also überwacht, unüberwacht und bestärkend. Für neuronale Netze wird dabei häufig die Trainingsmethode des *Dropouts* angewendet. Hier werden während des Trainings per Zufall ausgewählte Gewichte auf null gesetzt. Auf diese Weise wird das Risiko einer Überanpassung des Modells an die Trainingsdaten, ein *Overfitting*, reduziert.³⁴

Von »Deep Learning« spricht man im Allgemeinen, wenn das zugrunde liegende neuronale Netz mehrere verborgene Schichten hat, also ein »tiefes« Netzwerk ist.³⁵ Es gibt drei verbreitete Typen tiefer neuronaler Netze: Feedforward-Netze, konvolutive Netze und rekurrente Netze.

- Feedforward-Netze³⁶ sind Netze, in denen jedes Neuron Signale nur an Neuronen der nächsten Schicht sendet. Es heißt vollständig verknüpft (*fully-connected*), wenn jedes Neuron an jedes Neuron der nächsten Schicht sendet. Ein Beispiel für ein solches Netz ist in Abbildung 3 dargestellt.

30 Vgl. Raschka & Mirjalili (2021), § 2.1.1; Russell & Norvig (2022), S. 803; siehe auch Ivezić et al. (2020), S. 382.

31 Rosenblatt, F. (1958).

32 Vgl. Sengupta et al. (2022).

33 Vgl. Russell & Norvig (2022), S. 803.

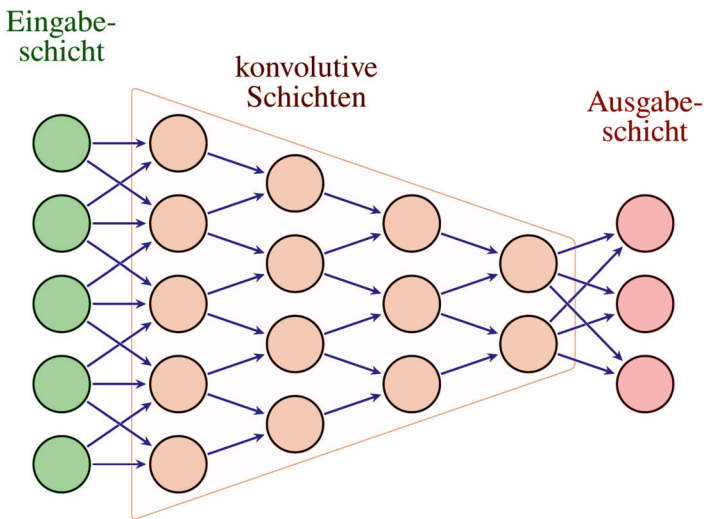
34 Vgl. Russell & Norvig (2022), S. 823; Weidman (2020), S. 122.

35 Siehe Goodfellow et al. (2016), S. 163ff.; Patterson & Gibson (2017), S. 81ff.; Russell & Norvig (2022), S. 801.

36 Vgl. Géron (2017), S. 263ff.; Goodfellow et al. (2016), S. 163ff.; Patterson & Gibson (2017), S. 54ff.; Russell & Norvig (2022), S. 802ff.

- Konvolute Netze (*convolutional neural network, CNN*), oder *Faltungsnetze*,³⁷ sind spezielle Feedforward-Netze, in der verborgene Neuronen nur Signale von wenigen jeweils benachbarten Neuronen der Vorgängerschicht bekommen, so dass das Netz gefaltet³⁸ und damit Schicht für Schicht kleiner wird, wie in Abbildung 6 skizziert. Konvolute Netze eignen sich zur Analyse von Bildern, also zweidimensionalen Datenrastern (Pixeln), aber auch für eindimensionale Datenraster wie Zeitreihen oder akustische Signale.³⁹

Abb. 6: Ein konvolutives Netz (CNN)



Quelle: eigene Darstellung

- Rekurrente Netze⁴⁰ (RNN) haben Neuronen, die Signale auch an Neuronen der eigenen oder einer vorhergehenden verborgenen Schicht senden, also Rückkopplungen ermöglichen. Dazu muss die Verarbeitung des Netzes in zeitverzögerten Schritten ablaufen. Durch die Rückkopplungen trägt das Ausgabemuster einer Schicht zum Eingabemuster bei, d.h. die Eingaben früherer Zeitpunkte beeinflussen den aktuellen Zustand des Netzes. Daraus ergibt sich die wesentliche Konsequenz, dass Eingabemuster in Gewichten des Netzes gespeichert werden können: Ein rekurrentes Netz hat ein Gedächtnis. Insbesondere zur Verarbeitung sequentieller Daten wie Bewegungs- oder Spracherkennung ist diese Eigenschaft entscheidend. So nutzen gängi-

37 Vgl. Géron (2017), S. 355ff.; Goodfellow et al. (2016), S. 321ff.; Patterson & Gibson (2017), S. 125ff.; Russell & Norvig (2022), S. 811ff.

38 Hier ist der Begriff Faltung oder Konvolution aus der Mathematik gemeint, siehe z.B. [https://de.wikipedia.org/wiki/Faltung_\(Mathematik\)](https://de.wikipedia.org/wiki/Faltung_(Mathematik))

39 Vgl. Goodfellow et al. (2016), S. 321.

40 Vgl. Géron (2017), S. 381ff.; Goodfellow et al. (2016), S. 368ff.; Patterson & Gibson (2017), S. 143ff.; Russell & Norvig (2022), S. 823f.

ge Sprachassistenten wie Alexa, Siri, Cortana oder Google Assistant spezielle rekurrente Netze, LSTM's (Long-Short-Term Memory), die beliebig viele Daten speichern können.⁴¹

Häufig findet man eine Kombination dieser Netzwerktypen. Neuronale Netze sind zwar sehr flexibel einsetzbar, aber auch sehr rechenintensiv. Daher kommen für ein KI-System oft auch Modelle zum Einsatz, die für bestimmte Aufgabenbereiche deutlich effizienter und schneller arbeiten. Meist sind es klassische Modelle der mathematischen Statistik, wie Regressionsmodelle, Zeitreihenanalysen, Klassifikatoren oder Cluster-Erkennung.

4 Natural Language Processing (NLP)

Der Bereich der Informatik, der sich mit den menschlichen Sprachen beschäftigt, ist die Computerlinguistik (im Englischen: *computational linguistics*). Ein zentrales methodisches Element ist dabei die Abstraktion einer Sprache, eines Sprachstils oder eines Jargons zu einem Sprachmodell, das von einem Computer verarbeitet werden kann. In diesem Kapitel wird dieser Begriff neben weiteren wesentlichen Konzepten und Technologien erläutert, die zur Generierung von Texten durch eine KI wie GPT angewendet wird.

4.1 Sprachmodelle

Ein *Sprachmodell* (*language model*) definiert eine Wahrscheinlichkeitsverteilung über Einheiten einer natürlichen Sprache, *Fragmente* oder *Tokens* genannt. Abhängig von dem jeweiligen Modellentwurf kann ein Fragment ein Wort, ein Phonem, ein einzelner Buchstabe oder ein Byte sein.⁴² Die ältesten Sprachmodelle sind *N-Gramme*.⁴³

Ein *N-Gramm* ist eine Folge von *N* Fragmenten. Ein auf *N-Grammen* basierendes Sprachmodell definiert die bedingten Wahrscheinlichkeiten des *N*-ten Fragments, wobei die vorhergehenden *N - 1* Fragmente gegeben sind. Üblicherweise werden diese Wahrscheinlichkeiten von Trainingsdaten abgeleitet.

Betrachten wir dazu als einfaches Beispiel das Sprachmodell für *N = 2*, »Bigramme«, basierend auf dem Trainingssatz mit einem Vokabular von fünf Wörtern:

wenn du denkst du denkst dann denkst du nur wenn du denkst

Sehen wir die Bigramme dieses Satzes als Präfixe und die darauf folgenden Wörter (»Monogramme«) als Suffixe, so ergibt sich daraus die folgende Tabelle.

41 Zumindes verwendeten diese Programme bis 2016/2017 rekurrente Netze, vgl. Buijsman (2021), S. 122.

42 Vgl. Goodfellow et al. (2016), S. 449.

43 Vgl. Shannon & Weaver (1949), S. 40f.

Präfix x	Suffix y	Bedingte Wahrscheinlichkeiten $P(x y)$
dann denkst	du	$P(\text{du} \text{dann denkst}) = 1$
denkst dann	denkst	$P(\text{denkst} \text{denkst dann}) = 1$
denkst du	denkst, nur	$P(\text{denkst} \text{denkst du}) = P(\text{nur} \text{denkst du}) = 1/2$
du denkst	du, dann, <eos>	$P(\text{denkst} \text{du denkst}) = P(\text{nur} \text{du denkst}) = P(\text{<eos>} \text{du denkst}) = 1/3$
du nur	wenn	$P(\text{wenn} \text{du nur}) = 1$
nur wenn	du	$P(\text{du} \text{nur wenn}) = 1$
wenn du	denkst	$P(\text{denkst} \text{wenn du}) = 1$

*(<eos> steht für *end of sentence*)

Die totalen Wahrscheinlichkeiten $P(x)$ der Monogramme x ergeben sich aus der Häufigkeitsverteilung der fünf Vokabeln in dem Satz mit zwölf Wörtern:

Wort x	dann	denkst	du	nur	wenn
$P(x)$	1/12	4/12 = 1/3	4/12 = 1/3	1/12	2/12 = 1/6

Die totalen Wahrscheinlichkeiten $P(xy)$ der sieben Digramme xy geben entsprechend die Häufigkeiten der 11 Wortpaare des Trainingssatzes wieder:

Digramm xy	dann denkst	denkst dann	denkst du	du denkst	du nur	nur du	wenn du
$P(xy)$	1/11	1/11	2/11	3/11	1/11	1/11	2/11

Diese Wahrscheinlichkeiten bilden dann das Sprachmodell unseres Trainingssatzes. Sprachmodelle auf Basis von N -Grammen repräsentieren Markovketten-Modelle (*Markov Chain Models*), da sie nach diesem mathematischen Prozessmodell immer nur den vorherigen Zustand des Systems – hier das vorhergehende $(N - 1)$ -Gramm – für den nächsten Zustand berücksichtigen.

Allgemein können wir mit einem Sprachmodell prognostizieren, welche Wörter am wahrscheinlichsten als nächstes in einem Text erscheinen. Auf diese Weise funktionieren Textvervollständigungen oder Korrekturvorschläge der gängigen Textverarbeitungsprogramme oder die Texteingabeprogramme von Smartphones, ebenso wie die Spam-Erkennung in E-Mails.⁴⁴

44 Vgl. Russell & Norvig (2022), S. 877.

4.2 Worteinbettungen

You shall know a word by the company
it keeps.

John R. Firth (1957)

N -Gramme sind atomistisch in dem Sinne, dass jedes Fragment wie ein Atom unterschiedlich von jedem anderen ist und Ähnlichkeiten oder Verwandtschaften von Fragmenten nicht erkannt werden. Für ein ausschließlich darauf fußendes Sprachmodell würden beispielsweise die Wörter *Mensch* und *Person* als völlig unabhängig gesehen. Dass die 5-Gramme »Dieser Mensch fühlt sich wohl« und »Diese Person fühlt sich wohl« inhaltlich so gut wie gleich sind, gibt die Statistik allein zunächst nicht her.

Um diese Unzulänglichkeit zu beheben, werden sogenannte *Worteinbettungen* (*word embeddings*) entwickelt. Hier werden Wörter als Vektoren in einem mehrdimensionalen Vektorraum dargestellt, so dass ähnliche Wörter geometrisch nah beieinander liegen.⁴⁵ Vektorräume sind in der Mathematik ein lange ausgereiftes Konzept und Darstellung und Rechenoperationen von Vektoren sind für Computer effizient implementierbar.

Abb. 7: Die geometrische Lage von drei Begriffen im Vektorraum, die Geschlechterrelationen wiedergeben: Sie ergeben sich durch Parallelverschiebungen (*links*). In höherdimensionalen Räumen können auch mehrere Relationen für ein Wort eingebettet werden, wie rechts der Pluralrelation.



Quelle: Mikolov et al. (2013a)

Zunächst nur als Methode zur Abbildung syntaktischer Ähnlichkeiten konzipiert, entdeckten Mikolov et al. (2013a) bei Analysen von Worteinbettungen durch rekurrente neuronale Netze, dass damit neben syntaktischen Regelmäßigkeiten auch semantische Regelmäßigkeiten geometrisch abgebildet wurden. Genauer gesagt kann eine Relationen zwischen zwei Begriffen durch Parallelverschiebung auf andere Begriffe übertragen werden, vgl. Abbildung 7 (links): Der Vektor zwischen dem geometrischen Ort für »Mann« und dem Ort für »Frau« zeigt parallel verschoben nach »Onkel« auf »Tante«, oder verschoben nach »König« auf »Königin«.

45 Vgl. Goodfellow et al. (2016), S. 452.

Ein Programm kann nun also Vektorrechnung mit Wörtern durchführen:

$$\text{»König«} - \text{»Mann«} + \text{»Frau«} = \text{»Königin«}.$$

Was *bedeutet*, dass das Programm herausfinden kann, dass »Königin« einer »Frau« sehr wahrscheinlich dem entspricht, wie »König« einem »Mann« entspricht. Oder mit anderen Worten, dass ein weiblicher »König« eine »Königin« ist. Ohne jegliches Verständnis kann damit ein Softwaresystem der Bedeutung von Wörtern nahekommen. Und das rein auf der Basis, dass Wörter in den Trainingsdaten nebeneinander stehen.⁴⁶ Da für ein Wort in einem höherdimensionalen Vektorraum auch viele solcher Relationen eingebettet werden können, kann ein Programm also auch in mehreren »Bedeutungsdimensionen« rechnen. Als Überblick über Worteinbettungen und für weiterführende Literatur dazu sei Almeida und Xexéo (2019) empfohlen.

Auf Basis dieser Erkenntnisse entwickelte Mikolov mit seinem Team für Google das neuronale Netz Word2Vec. Es war 2013 ein Meilenstein, ein Jahrzehnt später jedoch gelten Transformer-Modelle als Stand der Technik. Worteinbettungen allerdings spielen darin immer noch eine wesentliche Rolle.

4.3 Transformer

Ein Transformer ist ein Deep-Learning-Modell, durch das ein Softwaresystem eine Folge von Wörtern in eine andere Folge von Zeichen übersetzen kann. Dies kann z.B. benutzt werden, um Text von einer Sprache in eine andere zu übersetzen. Dazu wird ein Transformer mit einer großen Menge von Beispieldaten vortrainiert, bevor das trainierte Modell dann zur Übersetzung verwendet werden kann.

Weitere Beispielanwendungen von Transformatoren sind die Textgenerierung oder die Zusammenfassung längerer Texte. Transformer weisen hierbei eine bessere Effizienz gegenüber anderen Ansätzen wie Long-short-term-memory-Architekturen (LSTM) auf und sind die Grundarchitektur vieler vortrainierter Machine-Learning-Modelle wie Bidirectional Encoder Representations from Transformers (BERT) und Generative Pretrained Transformer (GPT). Auch in AlphaFold2 sind Transformer enthalten. Wesentlicher Bestandteil eines Transformers ist der Self-Attention-Mechanismus.

4.3.1 Der Self-Attention-Mechanismus

Der allgemeine Attention-Mechanismus entstand im Zusammenhang mit Sprachübersetzung mit rekurrenten neuronalen Netzen (RNN). Ein Attention-Mechanismus besteht aus einem Encoder, dem eigentlichen Attention-Modul, und einem Decoder. Die Idee des Mechanismus ist dabei, die gesamte Eingabe, zum Beispiel einen Satz, in Form von »Kontextvektoren« in den Worteinbettungsraum abzubilden und in die Zielsprache zu übertragen. Für die schrittweise Berechnung des Kontextvektors werden dabei für jedes Wort der gesamte Satz der Quellsprache und die bis dahin errechnete

46 Vgl. Buijsman (2019), S. 110.

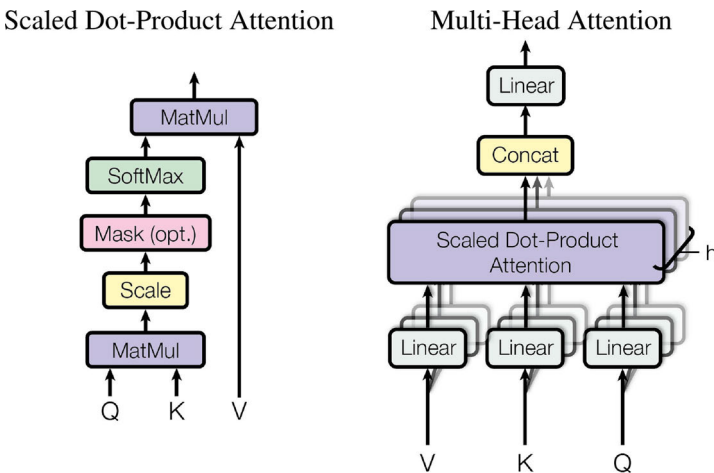
Ausgabe der Zielsprache berücksichtigt.⁴⁷ Im Unterschied zu den N -Grammen oben wird also jedes Wort im Kontext zu allen anderen Wörtern des Satzes betrachtet, und nicht nur die vorherigen N Wörter. Auf diese Weise entsteht in der ursprünglichen Version durch Training des Systems intern ein verborgenes Sprachmodell, das von den beiden rekurrenten neuronalen Netzen des Encoders und des Decoders gebildet ist.

In einem einflussreichen Beitrag in 2017 entwickelten Vaswani und Mitarbeiter:innen⁴⁸ diesen Ansatz zur Self-Attention weiter. Bei der Self-Attention sind Quell- und Zielsprache gleich, d.h. Ziel ist nicht mehr primär die Übersetzung in eine andere Sprache. Auch wird die komplexe Verwobenheit von Encoder, Attention-Modul und Decoder des ursprünglichen Attention-Mechanismus vereinfacht.

Die Aufgabe der Self-Attention ist es vielmehr, die Relation eines Wortes des Eingabesatzes mit den anderen Wörtern des Satzes zu berechnen. Gelingt dies erfolgreich, kann zum Beispiel die Frage beantwortet werden, worauf sich das Pronomen *it* im folgenden Satz bezieht:

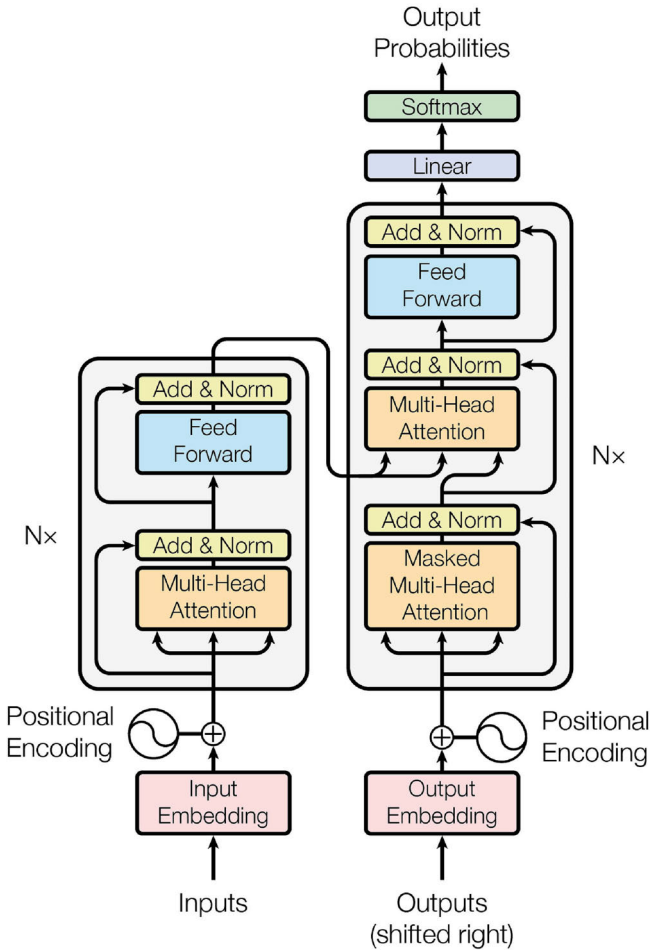
The animal didn't cross the street because it was too tired.

Abb. 8: Die Architektur des Transformers. Von links nach rechts das Self-Attention-Modul, die parallelgeschaltete Multi-Head-Attention, und unten der Aufbau des Transformers mit N Encodern (links) und N Decodern (rechts).



47 Bahdanau et al. (2014), insbes. S. 2f. und Russell & Norvig (2022), S. 917f.; die Begriffe Encoder und Decoder sind nicht nur auf Attention-Mechanismen beschränkt, vgl. Russell & Norvig (2022), S. 829.

48 Vaswani et al. (2017).



Quelle: Vaswani et al. (2017)

Ablauf des Algorithmus

Mathematisch ist die Self-Attention eine Abbildung von n Vektoren $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ des Einbettungsraums auf n Kontextvektoren $(\mathbf{c}_1, \dots, \mathbf{c}_n)$ desselben Raums. Der Ablauf des Algorithmus ist in Abbildung 8 skizziert. Im ersten Schritt wird jeder der Eingabevektoren auf drei Vektoren kleinerer Dimension projiziert,⁴⁹

- den »Abfragevektor« (*query vector*) $\mathbf{q}_i = W_q \mathbf{x}_i$
- den »Schlüsselvektor« (*key vector*) $\mathbf{k}_i = W_k \mathbf{x}_i$
- den »Wertvektor« (*value vector*) $\mathbf{v}_i = W_v \mathbf{x}_i$

Betrachten wir alle Vektoren als Spaltenvektoren, sind die drei Matrizen hierbei rechteckige Matrizen mit den Dimensionen

49 Die folgende Beschreibung des Algorithmus und die Notation nach Russell & Norvig (2022), S. 920.

$$\dim W_q = \dim W_k = (d \times n), \quad \dim W_v = ((n - 2d) \times n).$$

Damit haben also die Vektoren q_i und k_i dieselbe Dimension d , und v_i hat die Dimension $(n - 2d)$. Die Matrizen müssen also so designt sein, dass $d < n/2$ gilt, also d nicht zu groß ist. Die Vektoren werden dann jeweils als Spaltenvektoren zu Matrizen Q , K und V zusammengefasst,

$$Q = \begin{pmatrix} | & & | \\ q_1 & \cdots & q_n \\ | & & | \end{pmatrix}, \quad K = \begin{pmatrix} | & & | \\ k_1 & \cdots & k_n \\ | & & | \end{pmatrix}, \quad V = \begin{pmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{pmatrix}.$$

Die Matrizen haben also die Dimensionen

$$\dim Q = \dim K = (d \times n), \quad \dim V = ((n - 2d) \times n)$$

genau wie die obigen Matrizen W_q , W_k , W_v . Im zweiten Schritt werden nun aus den Matrizen Q und K die n^2 Bewertungen (*scores*)

$$r_{ij} = q_i \cdot k_j / \sqrt{d}$$

gebildet. Das Skalarprodukt der beiden Vektoren heißt auf Englisch *dot product*. Die Indizes i und j beziehen sich auf Wörter desselben Satzes, d.h. die Zahl r_{ij} drückt die Bewertung der Relation zwischen Wort i und Wort j aus. Aus diesen Bewertungen werden nun Wahrscheinlichkeiten berechnet:

$$p_{ij} = \exp(r_{ij}) / \sum_{k=1}^n \exp(r_{ik})$$

Die Funktion auf der rechten Seite ist die normalisierte Exponentialfunktion und heißt im Englischen *softmax*. Der vierte Schritt der Self-Attention ist die Berechnung der n Kontextvektoren der Eingabe:

$$c_i = \sum_{j=1}^n p_{ij} \cdot v_j$$

Die Wahrscheinlichkeiten p_{ij} sind asymmetrisch in den Indizes i und j , ebenso wie die Bewertungen r_{ij} . Die p_{ij} stellen im Kern das Sprachmodell des Transformers dar. Sie bilden sich durch die Trainingsdaten.

4.3.2 Multi-Head Attention

Da für die Self-Attention nur Matrix- und Vektoroperationen durchgeführt werden, hat sie gegenüber den Attention-Mechanismen mit rekurrenten neuronalen Netzen wesentliche Vorteile: Insbesondere können sie parallelisiert für alle Eingabewerte berechnet werden, während letztere die Eingabewerte sequenziell durcharbeiten müssen. Das schafft bei geeigneter Rechnerarchitektur enorme Laufzeitvorteile.⁵⁰

Diese Eigenschaft nutzt der Transformer aus, indem er die Self-Attention zu einem Modul kombiniert, *Multi-Head Attention* genannt. Der Eingabesatz wird hier in mehrere Teile zerlegt, auf denen separat die Self-Attention parallelisiert durchgeführt werden. Die Ergebnisse werden dann zu n Kontextvektoren c_i konkateniert. Das hat den Vorteil, dass wichtige Informationen seltener über die gesamte Eingabe ausgemittelt werden und so am Ende berücksichtigt werden.⁵¹ Die Struktur der Multi-Head Attention ist in Abbildung 8 Mitte illustriert.

4.3.3 Architektur eines Transformers

Ein Transformer besteht im Wesentlichen aus in Serie geschalteten Kodierern (*Encoder*) und in Serie geschalteten Dekodierern (*Decoder*), wie in Abb. 8 rechts dargestellt. Die Eingabesequenz wird durch die Einbettungsschicht in eine Vektorrepräsentation überführt. Die Gewichte der Einbettungsschicht werden während des Trainings angepasst. Im Falle des Transformers kommt zusätzlich eine Positionskodierung zum Einsatz, wodurch die sequentielle Abfolge der Wörter berücksichtigt werden kann. Ein Wort erhält somit zu Beginn eines Satzes eine andere Repräsentation als am Ende.⁵² Der Decoder führt schließlich die eigentliche Textgenerierung durch.

Die Architektur ist in Abbildung 8 rechts skizziert. Die Struktur eines Kodierers ist dabei im Kästchen links zu erkennen. Sie besteht im Wesentlichen aus einem Self-Attention-Modul und einem neuronalen Feedforward-Netz, typischerweise mit ReLU als nichtlineare Aktivierungsfunktion. Ein Dekodierer ist im Kästchen rechts dargestellt. Er ähnelt einem Kodierer, nur hat er zusätzlich ein »maskiertes« Multi-Head-Attention Modul: Die Maskierung sichert ab, dass die Vorhersagen für Position i nur von den bisher bekannten Positionen kleiner als i abhängen. Das ist wichtig, da Text von links nach rechts erzeugt werden soll.⁵³

Die Eingabesequenz wird in der Vektorrepräsentation einer Serie von Kodierern übergeben. Jeder dieser Kodierer übergibt dem zweiten Multi-Head-Attention-Modul jedes einzelnen Dekodierers seine Kontextvektoren, während jeder Dekodierer seine ausgegebenen Kontextvektoren dem nachfolgenden Dekodierer übergibt.

Was ist das Ergebnis eines Transformers? Er ordnet jedem Wort der Eingabesequenz zu, welche Relation es zu den anderen hat. In Abbildung 9 beispielsweise wird dem Pronomen *it* die stärkste Relation zu dem Wort *animal* zugerechnet, gefolgt von *street*.

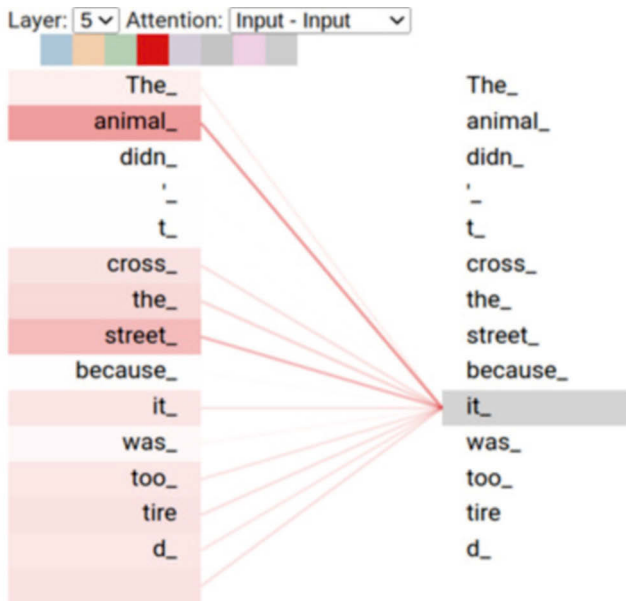
50 Vgl. Vaswani et al. (2017), S. 6.

51 Russell & Norvig (2022), S. 920.

52 Russell & Norvig (2022), S. 920ff.

53 Vgl. Vaswani et al. (2017), S. 3; Russell & Norvig (2022), S. 922.

Abb. 9: Ergebnis des Transformers: Das Wort *it* verweist mehr auf *animal* als auf *street*. Gezeigt ist hier nur die »rote« Dimension des Einbettungsraums.



Quelle: https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_tzt.ipynb#scrollTo=OJKU36QAfQOC

5 GPT

Generative Pre-trained Transformer, abgekürzt GPT, ist ein Sprachmodell, das von dem Unternehmen OpenAI entwickelt wird. Die erste Version wurde 2018 veröffentlicht,⁵⁴ 2019 folgte GPT-2,⁵⁵ und 2020 dann GPT-3.⁵⁶ Als öffentlich verfügbares Programm stellt OpenAI seit dem 30. November 2022 die Variante ChatGPT zur Verfügung, die auf GPT-3.5 basiert. Die ursprünglichen GPT-Modelle können nur Sätze vervollständigen, funktionieren im Prinzip also nach dem gleichen Prinzip wie die in Smartphones schon seit längerem vorhandenen Textvervollständigungen. ChatGPT erweitert diese Modelle in der Weise, dass es Eingabe über ein Textfeld ermöglicht und eingegebene Fragen und Anweisungen beantworten kann.

GPT ist eng angelehnt an das Transformer-Modell in Abbildung 3, allerdings wird kein Encoder eingesetzt,⁵⁷ siehe Abbildung 10. Das Training von GPT besteht aus zwei

54 Radford et al. (2018).

55 <https://openai.com/blog/better-language-models/>; Solaiman et al. (2019), S. 3.

56 Brown et al. (2020).

57 Radford et al. (2018), S. 3.

Phasen, dem unüberwachten Vortraining (*unsupervised pre-training*) und dem überwachten Feintuning (*supervised fine-tuning*).⁵⁸ Mit dem Vortraining kann zum Beispiel eine Sprache gelernt werden und im Feintuning um individuelle Sprachstile ergänzt werden, wie beispielsweise der Sprachduktus von Shakespeare oder die Ausdrucksweise eines 7-jährigen Kindes. Die Version GPT-3 ist ein Modell mit 175 Milliarden Parametern und arbeitet mit Kontextfenstern von 2048 Wörtern.⁵⁹

Bei Sprachmodellen mit mehreren Milliarden Parametern spricht man von *Large Language Models (LLM)*. Nach empirischen Befunden scheint die Textgenerierung von Sprachmodellen der GPT-Familie ab einer Parameteranzahl von 774 Millionen einen Qualitätssprung zu erfahren.⁶⁰

Selbst die Entwickler von GPT-3 sorgen sich um das gefährliche Potential ihres Produkts als *Large Language Model*:

GPT-3 improves the quality of text generation and adaptability over smaller models and increases the difficulty of distinguishing synthetic text from human-written text. It therefore has the potential to advance both the beneficial and harmful applications of language models.

Entwicklerteam von GPT-3 (Brown et al. 2020, S. 34)

Mitarbeiter:innen von OpenAI führten gemeinsam mit Wissenschaftlern der Cornell University, des Middlebury Institute of International Studies, der University of Oregon und der University of Texas at Austin eine 75-seitige Untersuchung über die sozialen Auswirkungen von Sprachmodellen durch.⁶¹ Neben positiven Beispielen der Nutzung großer Sprachmodelle für die Softwareentwicklung, das Gesundheitswesen oder den Unterhaltungsbereich werden auch Missbrauchsmöglichkeiten wie absichtliche Desinformation oder Vortäuschung, unter anderem in Abhängigkeit der technischen Fertigkeiten der Akteure genannt. Daneben wird aber auch das grundsätzliche Problem des *AI Bias* adressiert, also der Voreingenommenheit von KI. Sie kann durch die verwendeten Trainingsdaten entstehen, aber auch durch das Design der Software selbst.

Als vordringlichste Aufgabe zur Vermeidung missbräuchlicher Nutzung von *Large Language Models* wird die automatische Erkennung synthetischer Texte mit Hilfe von KI-Systemen genannt, ohne die Problematik eines Katz-und-Maus-Spiels zwischen Täuschern und Aufklärern zu verschweigen. Bemerkenswerterweise erkennt GPT von ihm selbst erzeugte Falschtexte schlechter als die Konkurrenzsoftware RoBERTa,⁶² ein Phänomen, was derzeit wenig verstanden zu sein scheint. Die Untersuchung empfiehlt Publikationsnormen für Veröffentlichungen von KI-Systemen u.a. die Schaffung von öffentlichen Rahmen zur interdisziplinären Steuerung von Zielkonflikten, den Aufbau von Infrastruktur zur verteilten Risikoanalyse und die Einrichtung von Kommunikationskanälen zwischen die KI-Systeme entwickelnden Organisationen.

58 Radford et al. (2018), S. 3f.

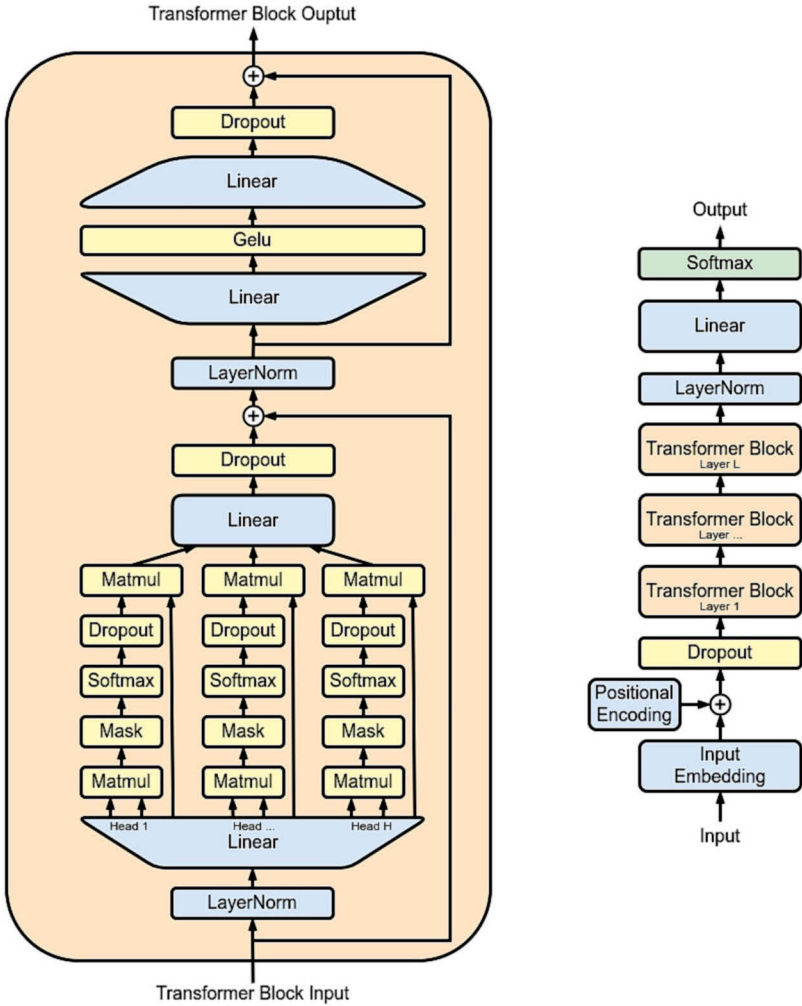
59 Brown et al. (2020), S. 8.

60 vgl. Solaiman et al. (2019), S. 10.

61 Solaiman et al. (2019).

62 Solaiman et al. (2019), S. 15.

Abb. 10: Das Modell von GPT. Links eine einzelne Transformerschicht, rechts der grobe Aufbau. Grafik modifiziert nach https://commons.wikimedia.org/wiki/File:Full_GPT_architecture.png (Public Domain)



Quelle: https://commons.wikimedia.org/wiki/File:Full_GPT_architecture.png (Public Domain)

Allerdings weigerte sich das Unternehmen OpenAI bei Veröffentlichung der Version GPT-4 im März 2023, Details zu Architektur, Modellgröße oder Trainingsdaten offenzulegen.⁶³ Dieses Verhalten widerspricht grundlegenden Forderungen des Deutschen Ethikrats in einer Stellungnahme vom 20. März 2023: »Auch dort, wo Systeme eine Quasi-Monopolstellung erlangen, sind hohe Anforderungen an Transparenz, Erklärbarkeit

63 OpenAI (2023), S. 2.

und Nachvollziehbarkeit zu stellen.«⁶⁴ Es wird Aufgabe von Politik und Gesellschaft bleiben, die notwendigen rechtlichen Regulierungen und Rahmenbedingungen dafür festzulegen.⁶⁵

5.1 Beispiele von Textgenerierungen durch ChatGPT

Im Folgenden werden im Wortlaut Beispiele von Textgenerierungen dokumentiert, die der Verfasser durch Bedienung des Chatbots ChatGPT⁶⁶ im Februar und März 2023 erhielt. Die Eingaben des Verfassers sind dabei kursiv gedruckt.⁶⁷

Beispiel 1

Erkläre den Begriff Entropie für 10-jährige Kinder.

Der Begriff Entropie kann für 10-jährige Kinder schwierig sein, aber ich werde mein Bestes geben, um es einfach zu erklären.

Entropie ist ein Begriff, der erklärt, wie chaotisch oder unordentlich Dinge sind. Wenn etwas geordnet und sauber ist, hat es eine niedrige Entropie. Wenn es jedoch chaotisch und unordentlich ist, hat es eine höhere Entropie.

Zum Beispiel, wenn wir ein Puzzle machen, haben wir alle Teile in einer bestimmten Reihenfolge und es ist sehr geordnet. Die Entropie des Puzzles ist niedrig. Wenn wir jedoch alle Teile mischen und sie in einer zufälligen Weise legen, dann wird das Puzzle sehr unordentlich und chaotisch und die Entropie wird höher.

Ein weiteres Beispiel ist, wenn wir ein Zimmer aufräumen. Wenn das Zimmer sehr unordentlich und voller Dinge ist, hat es eine höhere Entropie. Aber wenn wir es aufräumen und alles an seinem Platz ist, hat es eine niedrigere Entropie.

Entropie ist ein wichtiges Konzept in der Physik, aber es ist auch in unserem täglichen Leben präsent, auch wenn wir es nicht immer so nennen.

Beobachtungen zu Beispiel 1: Der Begriff ist inhaltlich angemessen und trotz seiner Vagheit korrekt erklärt. Es werden zur Erläuterung zwei Beispiele genannt, die Kindern vertraut sind.

Beispiel 2

Please write a sonnet about entropy in the style of Shakespeare, as if he had known the concept of entropy.

64 Deutscher Ethikrat (2023), S. 285.

65 Bergt (2023).

66 <https://chat.openai.com/>

67 Vgl. dazu ähnliche Experimente von King & ChatGPT 2023.

Entropy, thou art the bane of all our days,
 The measure of disorder in our fate,
 A force that renders all our works to waste,
 And leaves us with a future desolate.

Like time, thou creep'st upon us unawares,
 And steals the energy from all our toil,
 But still we strive against thy cruel snares,
 And struggle to maintain a stable boil.

But entropy, thou art not all a curse,
 For in thy wake new life doth oft arise,
 A cycle of destruction and rebirth,
 That keeps the universe before our eyes.

So entropy, though thou may'st bring decay,
 Thou art the spark that lights the fire of change.

Please write a passage in German about entropy in the style of Goethe's Faust.

Entropie, du bist die Kraft allen Wandels,
 Die treibende Kraft des Verfalls der Schöpfung,
 Ein ständiger Zyklus von Aufbau und Zerfall,
 Ein Tanz, der zu einer ewigen Herrschaft führt.

Ich sehe dich im Fallen des Laubes,
 Im langsamen Verfall aller großen Dinge,
 In der Veränderung der kurzen Jahreszeiten,
 In der Bewegung des wandelnden Sandes.

Aber in deinem Kielwasser entsteht neues Leben,
 Ein Phoenix aus den Aschen des Alten,
 Eine Chance für neue Horizonte,
 Eine noch unerzählte Geschichte.

Entropie, du bist die ewige Kraft,
 Eine Erinnerung daran, dass alles seinen Lauf hat.

Beobachtungen zu Beispiel 2: Beide Texte beschreiben bildreich die Wirkung von Entropie, wobei physikalisch nicht plausibel die Entropie als treibende Kraft von Zyklen dargestellt wird (»A cycle of destruction and rebirth«, »Ein ständiger Zyklus von Aufbau und Zerfall«). Der erste Text ist ein an das Frühneuenglische angelehntes Sonett ohne erkennbaren Versfuß, das aus drei vierzeiligen Kreuzreimen besteht und mit zwei Waisen endet. Der zweite Text ist in Standardhochdeutsch formuliert, ganz ohne Reim und ohne erkennbaren Versfuß formuliert. Die sprachliche Variabilität im Englischen ist deutlich höher als im Deutschen.

Beispiel 3

Anna ist größer als Bert, Bert ist größer als Cindy. Ist Anna größer oder kleiner als Cindy?

Da Bert größer als Cindy ist und Anna größer als Bert ist, können wir schließen, dass Anna auch größer als Cindy ist. Also, Anna ist größer als Cindy.

Wenn Bert kleiner als Anna ist und größer als Cindy, ist Anna dann kleiner als Cindy?

Es ist nicht möglich zu bestimmen, ob Anna kleiner als Cindy ist, basierend auf der Information, dass Bert kleiner als Anna und größer als Cindy ist.

Zum Beispiel könnte Anna 160 cm groß sein, Bert 150 cm und Cindy 140 cm. In diesem Fall wäre Bert kleiner als Anna und größer als Cindy, aber Anna wäre immer noch größer als Cindy.

Alternativ könnte Anna 180 cm groß sein, Bert 170 cm und Cindy 190 cm. In diesem Fall wäre Bert kleiner als Anna und größer als Cindy, aber Anna wäre immer noch größer als Bert.

Daher ist es nicht möglich, das Verhältnis zwischen Anna und Cindy zu bestimmen, basierend auf der Information, dass Bert kleiner als Anna und größer als Cindy ist.

Bist du dir da sicher?

Ja, ich bin mir sicher. Wenn wir nur wissen, dass Bert kleiner als Anna und größer als Cindy ist, haben wir keine Informationen über das Verhältnis zwischen Anna und Cindy. Es gibt keine Möglichkeit, das Verhältnis zwischen Anna und Cindy zu bestimmen, solange wir keine direkte Vergleichsinformationen zwischen den beiden haben.

Das ist falsch, du verstehst nichts von Transitivität!

Es tut mir leid, wenn meine vorherige Antwort für Verwirrung gesorgt hat. Tatsächlich hast du Recht – wenn Bert kleiner als Anna ist und Bert größer als Cindy ist, dann kann man mit Hilfe der Transitivität schlussfolgern, dass Anna kleiner als Cindy sein muss.

Das liegt daran, dass die Transitivität der Relation »größer als« besagt, dass wenn $a > b$ und $b > c$, dann $a > c$. In diesem Fall haben wir $Bert > Cindy$ und $Bert < Anna$, was bedeutet, dass $Cindy < Bert < Anna$ ist, und daher muss Anna größer als Cindy sein.

Vielen Dank, dass du mich darauf aufmerksam gemacht hast.

Beobachtungen zu Beispiel 3: Eine transitive Relation wie »größer als« ermöglicht die logische Schlussfolgerung, dass wenn $a > b$ und $b > c$, dann $a > c$. Dieser logische Schluss wird von ChatGPT aber offenbar nur dann gezogen, wenn die beiden Relationen der Prämisse auch beide mit »größer als« ausgedrückt werden. Sobald eine der Relationen äquivalent als »kleiner als« ausgedrückt wird, sind die Antworten entweder glatt falsch oder in sich völlig widersprüchlich.

5.2 Auswertung der Beobachtungen

Die obigen Beispiele der Textgenerierung zeigen grundsätzlich ein bemerkenswert hohes sprachliches und inhaltliches Niveau. Auf die als Eingabe formulierten Fragen oder Aufforderungen reagiert ChatGPT aufmerksam und in höflicher Sprache. Die abgegebenen Texte des KI-Systems zeigen ein größtenteils korrektes Standardhochdeutsch,

während sie im Englischen deutlich nuancenreicher und variabler erscheinen. Beispiel 1 zeigt wie angewiesen einen Kinder ansprechenden Sprachstil mit einfachen kurzen Sätzen und trotz des abstrakten Themas eine kindgerechte Ausdrucksweise. Beispiel 2 belegt eine Fähigkeit im Englischen zur Bildung von Versreimen, zumindest bis zu einem gewissen Grade, die im Deutschen nicht gelingt (Stand Februar 2023). Beispiel 3 beweist, dass das Sprachmodell logische Inkonsistenzen nicht erkennt und sich sogar in logische Widersprüche verwickelt (Stand März 2023).

Die Befunde zeigen ein für ein Sprachmodell auf Basis der in den obigen Abschnitten beschriebenen Technologien nicht unerwartetes Verhalten, wenn davon ausgegangen wird, dass die Trainingsdaten vornehmlich englischsprachige Texte sowie deutschsprachige Texte verschiedener Sprachstile enthielten. Auch die in speziellen Fällen zu beobachtenden logischen Inkonsistenzen und Widersprüche lassen das auf statistischer Textvervollständigung beruhende Sprachmodell erkennen. Manchmal verblüffen allerdings die inhaltlichen Assoziationen und Ideen, mit denen angewiesene Themen beantwortet werden.

Wäre es ein menschlicher Gesprächspartner, so würde man ChatGPT insgesamt als geistreich, kreativ, höflich und sprachlich versiert bezeichnen können, aber auch als geschwätzig, oft vage und abwägend, sich gelegentlich widersprechend, meinungslos und unzuverlässige Informationen gebend.

Es gibt mindestens zwei Probleme, unter denen aktuelle Sprachmodelle wie GPT zuvörderst leiden. Da wäre zum einen die Trainingsmethode: Da ein Sprachmodell anhand von Daten aus der Vergangenheit lernt, kann es das Bestehende lediglich reproduzieren. Das an die Trainingsdaten angepasste (»erlernte«) Modell ist deskriptiv, im besten Fall repräsentativ, aber kaum normativ. Wenn in Trainingsdaten von Bewerbungen die Menge an akzeptierten männlichen Bewerbern größer ist, könnte eine Entscheidungs-KI also abstraktes Muster lernen, dass männliche Bewerbungen weiblichen vorzuziehen sind.

Ein weiteres, grundlegendes Problem, an dem Sprachmodelle wie GPT scheitern, ist das Erkennen und die Vermeidung logischer Inkonsistenz. Die Ursache liegt in dem prinzipiell statistischen Charakter der Sprachmodelle, die anhand der Trainingsdaten Wahrscheinlichkeitsverteilungen von Wortfolgen ermitteln. Trotz der durchaus verblüffenden Wirkung der vektoriiellen Worteinbettungen, durch die statistische Korrelationen plötzlich semantische Bedeutung widerspiegeln, spielen logische Zusammenhänge keine Rolle. Die Antwort eines Sprachmodells der aktuellen Generation ist das statistische Ergebnis eines Trainings, nicht das einer logischen Schlussfolgerung.

6 Diskussion

In diesem Beitrag wurde der Einfluss des computergenerierten Zufalls auf kreative Prozesse untersucht. Als paradigmatisches Beispiel dafür wurde eine Werktechnik Gerhard Richters betrachtet, der etwa ab den 1970er Jahren Farbfolgen quadratischer Raster durch Zufallsgeneratoren berechnen ließ und seine Rolle als Künstler in einer auswählenden Instanz sah. Unabhängig davon entwickelten sich in der Computerlinguistik Sprachmodelle, die den Zufall auf eine völlig andere Weise in Form von Wahrscheinlichkeitsverteilungen zur Generierung von Texten verwendeten. Funktionsweise und

Prinzipien dieser Sprachmodelle wurden kurz dargestellt. Als prominenter Vertreter der Sprachmodelle wurde GPT der Firma OpenAI beschrieben und die öffentliche Variante ChatGPT anhand konkreter Dialogbeispiele untersucht.

Die Beispielantworten von ChatGPT zeigen auf der einen Seite im Englischen, aber auch in etwas geringerem Maße auch im Deutschen, eine bemerkenswert hohe Sprachqualität, die auf den ersten Blick nicht von Texten menschlicher Urheberschaft zu unterscheiden sind. Dagegen müssen die Texte auf der inhaltlichen Ebene kritischer betrachtet werden, sie wirken manchmal »geistreich« und überraschend, sind manchmal aber faktisch falsch und gelegentlich sogar in sich widersprüchlich.

Diese Befunde bestätigen eines der grundlegenden Probleme, an denen Sprachmodelle wie GPT scheitern, nämlich dem Erkennen und der Vermeidung logischer Inkonsistenz. Die Ursache liegt in dem prinzipiell statistischen Charakter der Sprachmodelle, die anhand der Trainingsdaten Wahrscheinlichkeitsverteilungen von Wortfolgen ermitteln und sie zur Textgenerierung anwenden. Abgesehen von der – oft durchaus verblüffenden – Wirkung der vektoriiellen Worteinbettungen, durch die gewisse semantische Bedeutungen als Korrelationen erscheinen, spielen logische Zusammenhänge keine Rolle. Die Antwort eines Sprachmodells der aktuellen Generation ist das statistische Ergebnis eines Trainings, nicht das einer logischen Schlussfolgerung.

Der Zufall ist also jedem der modernen Sprachmodelle gewissermaßen genetisch implantiert: Ein generierter Satz ist *mit sehr hoher Wahrscheinlichkeit* grammatikalisch korrekt, die Antwort auf eine Frage *mit hoher Wahrscheinlichkeit* inhaltlich richtig. Dieses Merkmal liefert gleichzeitig Grenzen und Potential ihres Einsatzes: Benötigt man zuverlässige Information, sollte man Sprachmodelle wie ChatGPT meiden; möchte man jedoch einen geistreichen und belesenen, wenn auch oft geschwätzigen und sich gelegentlich sogar widersprechenden Gesprächspartner für Anregungen von Formulierungen oder Ideen, so können sie sehr hilfreich sein.

Allgemein betrachtet liefern Sprachmodelle wie GPT damit eine weitere Facette des Einsatzes von Software für kreative Prozesse, seien es die Erstellung von Kunstwerken oder die Generierung von Texten. Richters Werktechnik der Farbtafeln bestand darin, Software mit Zufallsgeneratoren das Rohmaterial liefern zu lassen, aus dem der Mensch auswählt. Einen anderen, aber vergleichbaren Einfluss hat der Zufall in Form von Wahrscheinlichkeitsverteilungen im Bereich der Texterzeugung durch aktuelle Sprachmodelle. Richters Ansatz hierauf zu übertragen könnte damit insofern die Produktion von Literatur ermöglichen, indem aus den zufälligen Wortfolgen Texte oder Ideen ausgewählt werden. Ob es eines Tages Softwaresysteme geben wird, die Literatur erschaffen, ist grundsätzlich nicht auszuschließen. Betrachtet man den aktuellen Stand, ist der Weg bis dahin allerdings noch lang.

Sowohl in der Malerei als auch in der Texterzeugung gilt bislang: Die menschliche Auswahl aus zufälligen Artefakten ist der eigentlich kreative Prozess, nicht der Zufall.

Literaturverzeichnis

- Almeida, F.; Xexéo, G. (2019): Word Embeddings: A Survey. <https://arxiv.org/abs/1901.09069> (25. April 2023).
- Bahdanau, D.; Cho, K.; Bengio, Y. (2014): Neural Machine Translation by Jointly Learning to Align and Translate. <https://arxiv.org/abs/1409.0473> (25. April 2023).
- Bergt, S. (2023): Transparenz gewaschen. *taz*, 21. März 2023, <https://taz.de/!5922783> (25. April 2023).
- Brown, T. B. et al. (2020): Language Models are Few-Shot Learners. <http://arxiv.org/abs/2005.14165v4> (25. April 2023).
- Buijsman, S. (2021): *Ada und die Algorithmen*. C. H. Beck, München 2021.
- Deutscher Ethikrat (2023): Mensch und Maschine – Herausforderungen durch Künstliche Intelligenz. Stellungnahme. <https://www.ethikrat.org/fileadmin/Publikationen/Stellungnahmen/deutsch/stellungnahme-mensch-und-maschine.pdf> (25. April 2023).
- Firth, J. R. (1957): *Studies in Linguistic Analysis*. Basil Blackwell, Oxford.
- Géron, A. (2017): *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly, Sebastopol.
- Godfrey, M., Serota, N. (2012) (Hg.): *Gerhard Richter. Panorama*. Prestel, München London New York.
- Goebel, P. (2010): *Gerhard Richter: Das Glasfenster im Kölner Dom* (2007). VDM Verlag Dr. Müller, Saarbrücken.
- Goodfellow, I.; Bengio, Y.; Courville, A. (2016): *Deep Learning*. MIT Press, Cambridge.
- Hendrycks, D.; Gimpel, K. (2016): Gaussian Error Linear Units (GELUs). <https://arxiv.org/abs/1606.08415> (25. April 2023).
- Ivezić, Ž.; Connolly, A. J.; VanderPlas, J. T.; Gray, A. (2020): *Statistics, Data Mining, and Machine Learning in Astronomy*. Princeton University Press, Princeton Oxford.
- King, Michael R./ChatGPT (2023): A Conversation on Artificial Intelligence, Chatbots, and Plagiarism in Higher Education, in: *Cellular and Molecular Bioengineering*, 16 (1), S. 1–2. <https://doi.org/10.1007/s12195-022-00754-8> (25. April 2023).
- McLuhan, M. (1964): *Understanding Media*. Routledge, London.
- Mikolov, T., Yih, W., Zweig, G. (2013a): Linguistic regularities in continuous space word representations. *Proceedings of NAACL-HLT in Atlanta, Georgia, 9–14 June 2013*, S. 746–751, <https://aclanthology.org/N13-1090.pdf> (25. April 2023).
- Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. (2013b): Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*, <https://arxiv.org/abs/1301.3781> (25. April 2023).
- Museum Folkwang (2017) (Hg.): *Gerhard Richter. Die Editionen*. Steidl, Göttingen.
- Museum Ludwig & Metropolitankapitel der Hohen Domkirche Köln (2007) (Hg.): *Gerhard Richter – Zufall*. Verlag Kölner Dom Verlag der Buchhandlung Walther König, Köln.
- OpenAI (2023): GPT-4 Technical Report. <https://cdn.openai.com/papers/gpt-4.pdf> (25. April 2023)
- Patterson, Josh; Gibson, A. (2017): *Deep Learning*. O'Reilly, Sebastopol.

- Raschka, S.; Mirjalili, V. (2021): *Machine Learning mit Python und Keras, TensorFlow 2 und Scikit-learn*. 3. Auflage. mitp, Frechen.
- Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. (2018): *Improving Language Understanding by Generative Pre-Training*. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (25. April 2023).
- Rosenblatt, F. (1958): *The perceptron: a probabilistic model for information storage and organization in the brain*. *Psychological Reviews* 65, S. 386–408.
- Russell, S. J.; Norvig, P. (2022): *Artificial Intelligence. A Modern Approach*. Pearson, Harlow.
- Sengupta, R.; Adhikary, S.; Oseledets, I.; Biamonte, J. (2022): *Tensor networks in machine learning*. *EMS Magazine*, 126, December 2022, <https://doi.org/10.4171/MAG/101> (25. April 2023).
- Shannon, C.; Weaver, W. (1949): *The Mathematical Theory of Communication*. University of Illinois Press, Urbana Chicago.
- Solaiman, I., et al. (2019): *Release Strategies and the Social Impacts of Language Models*. <https://arxiv.org/abs/1908.09203> (25. April 2023)
- Vaswani, A.; Shazeer, N.; Parmar, N.; et al. (2017). *Attention Is All You Need*. <https://arxiv.org/abs/1706.03762> (25. April 2023).
- Weidman, S. (2020): *Deep Learning. Grundlagen und Implementierung*. Dpunkt.verlag & O'Reilly, Heidelberg.

