

## System 1.42: Wie (Frontier-)LLMs “tatsächlich” funktionieren

**Abstract:** Frontier-LLMs folgen weder Kahnemans schnellem System 1 noch dem analytischen System 2, sondern operieren als hybrides System 1.42 – so lautet zumindest meine These, mit der ich die Diskussion anregen möchte. Diese “halluzinierenden Reasoner” erzeugen plausible Antworten, ohne eine verlässliche Selbstprüfung durchzuführen. Mit zunehmender Skalierung (mehr Daten, mehr Rechenzeit, mehr Token) halluzinieren sie zwar immer häufiger korrekte Antworten, jedoch nicht für alle Probleme und auch nicht immer. Ist das alles nur eine (für manche) überzeugende Simulation? Oder emergiere hier tatsächlich Generalisierung und damit echtes Reasoning? Die “42”-Referenz symbolisiert diese fundamentale Ungewissheit über etwas qualitativ Neues, für das es keine einfachen Antworten gibt. Der Text erklärt die technischen Grundlagen und spannt den Bogen von LLM als Autocomplete über Emergenz und Grokking bis zu Reasoning. Wie gehen wir mit Systemen um, die funktional so tun, als ob sie verstehen, und die gleichzeitig ziemlich andersartig sind als wir? Der Text ist der Versuch, all das mit zahlreichen Referenzen zu kontextualisieren.

**Schlagwörter:** Applied Generative AI, LLM, Reasoning

**Autor:** Christopher Pollin • **Veröffentlicht:** 01.07.2025 • **Lizenz:** CC BY 4.0



Large Language Models (LLMs) liefern erstaunlich kohärente Antworten, und es herrscht weiterhin Uneinigkeit darüber, ob sie wirklich verstehen oder lediglich Muster reproduzieren, oder ob ihr scheinbares Verstehen aus bloßer Musterreproduktion emergiert. Die ***Stochastic Parrots***<sup>1</sup>-Metapher charakterisiert sie als bloße Mustererkenner. Doch die neueste Generation dieser Modelle – Frontier-LLMs wie OpenAIs *o3*, Googles *Gemini 2.5 Pro* oder Anthropic's *Claude 4 Opus und Sonnet* – stellt diese Sichtweise in Frage und entfacht die immer wieder aufflammende Diskussion, ob **echte Reasoning-Fähigkeiten** entstehen.<sup>2</sup> Handelt es sich vielleicht um eine emergente Eigenschaft ausreichend hochskalierter<sup>3</sup> Modelle, oder ist es eine

**Illusion von Reasoning**<sup>4</sup>, die aus Mustererkennung und Extrapolation gigantischer Trainingsdatenmengen entsteht? Wie funktionieren LLMs "tatsächlich"?<sup>5</sup>

Wir wissen es nicht. Wir wissen nicht genau, warum die Modelle bei manchen Aufgaben so gut funktionieren. Wie sie funktionieren, wissen wir natürlich. Was wir beobachten: LLMs werden immer besser darin, korrekte Antworten zu "halluzinieren", ohne dass wir vorhersagen können, wann sie richtig liegen und wann nicht. Diese Beobachtung bezieht sich auf Frontier-Modelle, die bei unterschiedlichen Problemtypen unterschiedlich abschneiden.<sup>6</sup> Aus den stochastischen Papageien sind 2025 *Agentic Systems*<sup>7</sup> geworden, die auf Reasoning-Modellen basieren und dadurch manche ihrer eigenen Schwächen ausgleichen können. Ein LLM kann zwar nicht zählen, programmiert sich aber einen Taschenrechner und nutzt diesen per *Tool Use*<sup>8</sup>. Falsche Auskünfte kann es durch Websuchen korrigieren, wobei es auch dabei halluzinieren kann. Wir müssen LLMs anders denken. Als etwas Neues. Ich schlage vor, Frontier-Modelle als **System 1.42** zu verstehen, angelehnt an Daniel Kahnemans<sup>9</sup> Theorie der zwei Denksysteme:

**System 1** arbeitet schnell, intuitiv und automatisch. Die Frage "Was ist  $2 + 2$ ?" kann ohne mentale Repräsentation beantwortet werden. **System 2** arbeitet langsamer, analytisch und bewusst. Für die Frage "Was ist  $439 \times 238$ ?" müssen wir uns in irgendeiner Form bewusst (!) im Kopf Gedanken darüber machen und können erst dann eine korrekte Antwort liefern. Man kann sich aber natürlich auch verrechnen.

LLMs operieren wie System 1, produzieren aber manchmal System-2-ähnliche Komplexität, ohne diese wirklich validieren zu können. "Nie wirklich" deswegen, weil es schon eine Form von Verifikation über das Erzeugen von Text gibt, der ein Problem weiter kontextualisiert. Neben "*Let's think step by step*" ist "*Verify step by step*" das Schlagwort.<sup>10</sup> Christopher Summerfield beschreibt sie als *Strange New Minds*<sup>11</sup>, deren Outputs sich funktional kaum von echtem Denken unterscheiden lassen. Dies ergänzt meine Charakterisierung von LLMs als **halluzinierende Reasoner**, also Systeme, die aufgrund statistischer Mustererkennung scheinbar logische Schlussfolgerungen ziehen, jedoch ohne diese jemals tatsächlich zu validieren. Da sich diese Eigenschaft wahrscheinlich auch in naher Zukunft nicht ändern wird – dazu bräuchte es eine andere Architektur<sup>12</sup> – steht die externe Validierung der erzeugten Token im Mittelpunkt. Gleichzeitig entstehen daraus nicht nur technische Herausforderungen, sondern auch tiefgreifende gesellschaftliche und psychologische Implikationen. Wie Summerfield betont, könnten Menschen zunehmend ihre Authentizität und Kontrolle verlieren, wenn sie sich unkritisch auf KI-generierte Inhalte verlassen. Deren vermeintliche Logik existiert letztlich nämlich nur oberflächlich. Wenn man jedoch weiß, welche Fragen man wie stellen muss, kritisch bleibt, und dann in der Lage ist, herauszufinden, ob die Antwort korrekt ist, dann hat man ein sehr praktisches Werkzeug gefunden.

In Douglas Adams *Per Anhalter durch die Galaxis* liefert der Supercomputer *Deep Thought* nach Millionen Jahren die Antwort "42". Diese Antwort ist bedeutungslos ohne die richtige Frage. LLMs operieren genau

umgekehrt und generieren auf jede Frage Antworten, ohne deren Korrektheit überprüfen zu können. Die "42" in meiner System-1.42-Metapher verweist auf diese Ungewissheit. Auf etwas, das funktioniert und zugleich nicht funktioniert. Etwas dazwischen, das wir scheinbar noch nicht richtig benennen und fassen können.

Da ich kein Experte für maschinelles Lernen bin, ist meine Darstellung zwangsläufig vereinfacht. Mir geht es darum, ein Gespür für die Funktionsweise von Frontier-Modellen zu vermitteln. Dieses Verständnis ist nicht nur beim Prompt Engineering hilfreich, sondern hilft uns auch, uns auf die kommenden und bereits vorhandenen, sehr einschneidenden Veränderungen einzustimmen und vorzubereiten.

## Token, Embeddings, Context Window und Training

Es ist notwendig, ein paar grundlegende Konzepte zu klären. Token sind atomare Verarbeitungseinheiten, Embeddings mathematische Bedeutungsrepräsentationen, das Context Window das "Arbeitsgedächtnis" und der Trainingsprozess prägt die grundlegende Funktionsweise. Diese vier Konzepte bilden das technische Fundament für das Verständnis von LLMs. Die Mathematik dahinter müssen Sie nicht beherrschen, schaden würde es aber natürlich nicht.

### Token

Token bilden die grundlegende Einheit, in die Texte zerlegt werden, um von einem LLM verarbeitet zu werden. Ein Token kann ein ganzes Wort, ein Teil eines Wortes, Leer- oder Steuerzeichen oder gar nur ein einzelnes Zeichen sein. Diese Zerlegung erfolgt durch einen sogenannten **Tokenizer**<sup>13</sup>, ein Programm, das für jede Modellfamilie und Sprache optimiert wird.<sup>14</sup> Der Satz "Christopher codes and uses XSL-T. The cat sat on the keyboard!" besteht aus 11 Wörtern und 62 Zeichen (inklusive Leerzeichen). Nach der Tokenisierung entstehen für den Tokenizer für GPT4o von OpenAI daraus 15 Token:

The screenshot shows the OpenAI Tokenizer interface. At the top, there are three tabs: "GPT-4o & GPT-4o mini", "GPT-3.5 & GPT-4", and "GPT-3 (Legacy)". Below the tabs, the input text is displayed: "Christopher codes and uses XSL-T. The cat sat on the keyboard!". Underneath the input, there are two sections: "Tokens" and "Characters". The "Tokens" section shows the number "15", and the "Characters" section shows the number "62". At the bottom, the input text is shown again, but each word and punctuation mark is highlighted with a different color: Christopher (purple), codes (green), and (yellow), uses (red), XSL-T (blue), . (orange), The (purple), cat (green), sat (red), on (yellow), the (red), keyboard (blue)!. There are also two buttons at the bottom left: "Clear" and "Show example".

OpenAI Tokenizer. <https://platform.openai.com/tokenizer> (<https://platform.openai.com/tokenizer>)

Token entsprechen nicht unserer intuitiven Wortwahrnehmung. Der technische Begriff "XSL-T" wird in drei separate Token ("XSL", "-" und "T") zerlegt. Solche Fragmentierungen haben Konsequenzen. Jedes Token "beansprucht Aufmerksamkeit" im *Attention*-Mechanismus der Transformer-Architektur. Zwei weitere Begriffe hier mal als Teaser angekündigt und später erklärt. Token mit wenig semantischem Gehalt, wie beispielsweise einzelne Bindestriche, tragen weniger dazu bei, eine "gute" Antwort auf einen Prompt zu erhalten. Der Attention-Mechanismus erkennt ihre geringe Bedeutung und gewichtet sie entsprechend niedriger. Sie lenken das Modell also nicht ab, sondern blockieren vielmehr Platz im Context Window. Das ist vereinfacht, aber es hilft vielleicht als Gedankenstütze beim Arbeiten. Bei vereinzelter Auftreten ist das unproblematisch. Große Mengen Token – etwa XML-basierte Formate wie Excel- oder PowerPoint-Dateien – können das limitierte Kontextfenster schnell erschöpfen und die Qualität der Antworten dadurch beeinträchtigen. Die Effizienz hängt dabei vom jeweiligen Modell, Tokenizer und der Größe des Kontextfensters ab. Deshalb ist ein "Gespür" für den Token-Verbrauch wichtig. Ein Text im steirischen Dialekt mit Tippfehlern wird beispielsweise schlechter tokenisiert als ein Text mit fast dem gleichen Informationsgehalt, der im standardisierten, modernen Englisch repräsentiert wird.

## Embedding

In einem LLM wird jedes Token als Embedding repräsentiert. Ein Embedding ist eine Reihe von Zahlen, die einen Pfeil in einem abstrakten, mathematischen Raum mit Tausenden von Dimensionen definieren. Technisch gesprochen ist es ein Vektor. Dabei ermöglichen Embeddings mathematische Operationen wie Ähnlichkeitsberechnungen, wodurch semantische Nähe quantitativ erfasst werden kann. Die Embeddings für die Token "Katze" und "Hund" liegen in diesem Raum näher beieinander als "Katze" und "Stein", da sie in ähnlichen Kontexten auftreten: Sie sind beide Haustiere, lebendig und man kann mit ihnen gut kuscheln. Mit Steinen kuschelt es sich hingegen nicht so gut. Dies ist ein bewusst etwas absurdes Beispiel, das so (vielleicht) nie explizit in den Trainingsdaten vorkommt. Das Modell kann solche Zusammenhänge implizit aus Milliarden von Kontexten ableiten. Nicht der einzelne Vektor allein repräsentiert die Bedeutung, sondern seine Lage relativ zu allen anderen Vektoren. Man kann sich den Raum wie ein riesiges Universum vorstellen. So mache ich das zumindest. Jedes Wort erscheint zunächst als Ausgangsvektor. Die Transformer-Schichten transformieren jeden Token schrittweise, wodurch eine Flugbahn durch dieses Universum entsteht. Diese Transformation ist deterministisch. Gleiche Token-Sequenzen erzeugen gleiche Embeddings. Der Output kann trotzdem variieren durch Temperature-Sampling bei der Token-Auswahl. Dabei nähern sich "Katze" und "Hund" dem Verb "kuscheln", während "Stein" auf Abstand bleibt und andere Bedeutungsräume, je nach Kontext, durchquert. So wird aus dem festen Basis-Embedding Schritt für Schritt ein kontextabhängiger Vektor berechnet. Durch diese Kontextabhängigkeit entsteht bei jeder Eingabe eine ganze Kette neuer, kontextualisierter Embeddings.

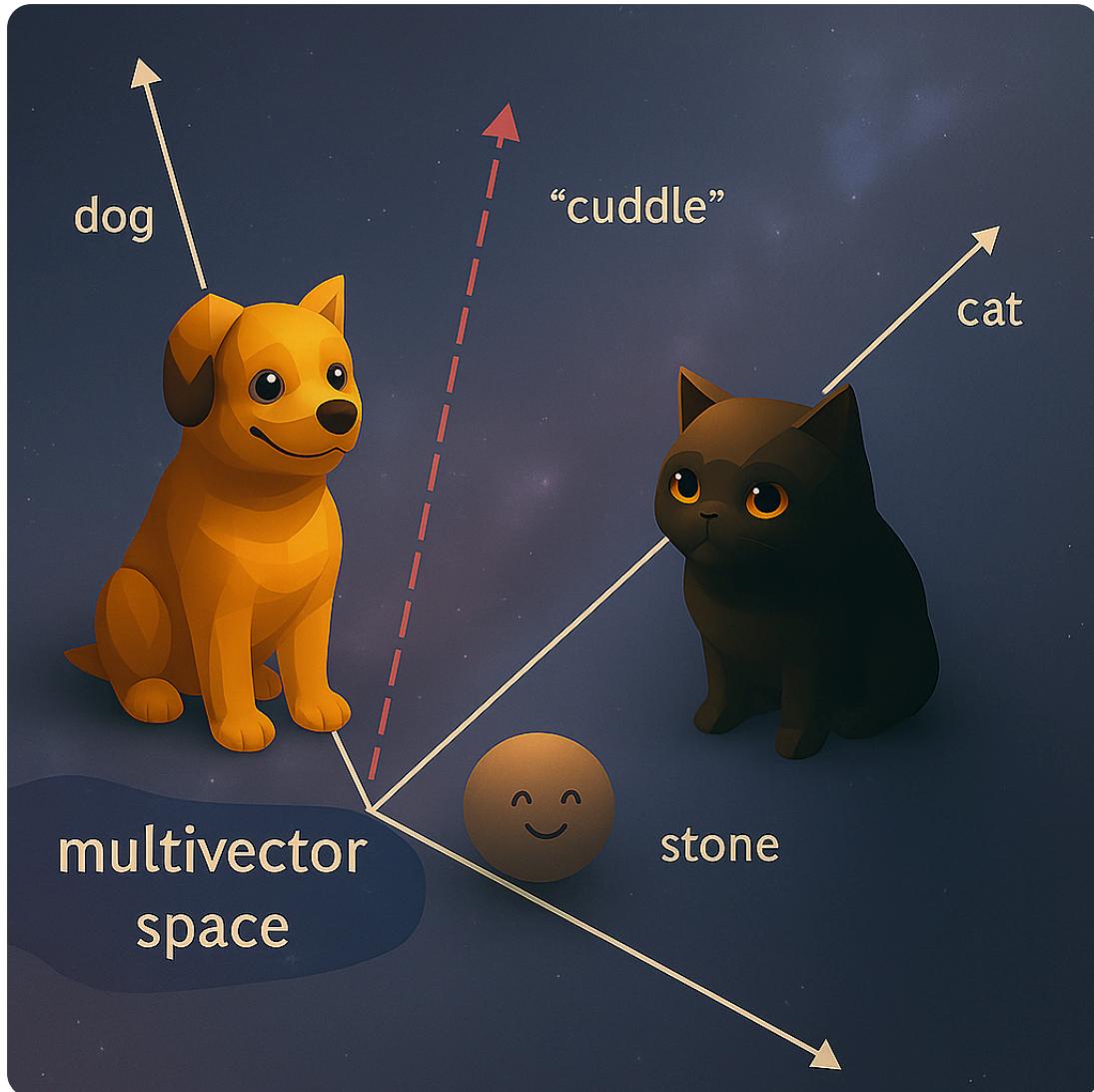
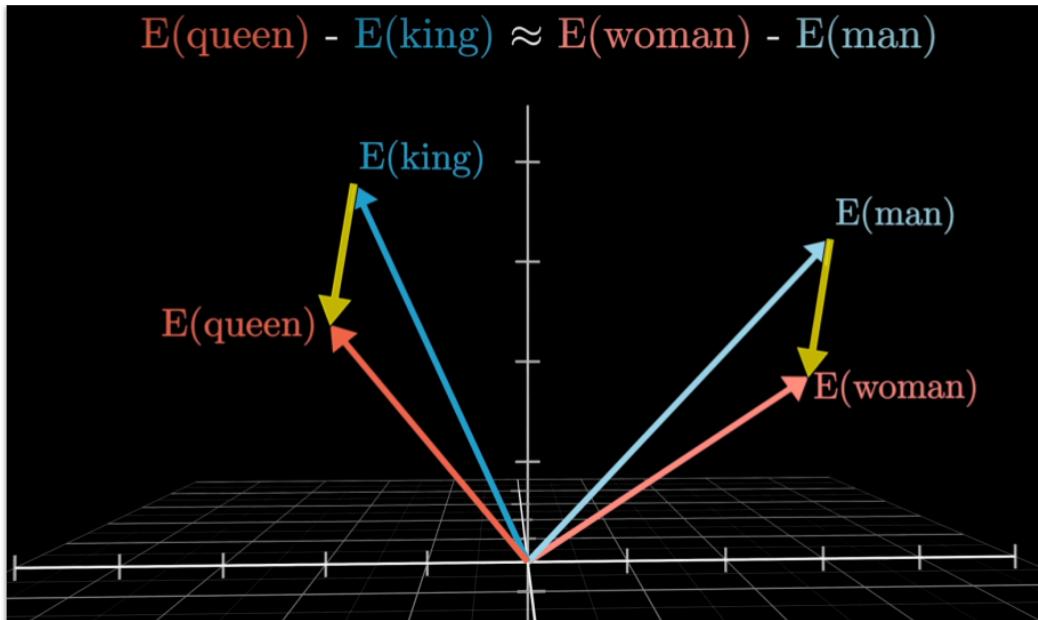


Illustration eines Embedding-Raums, der zeigt, dass die Vektoren für ‚Hund‘ (dog) und ‚Katze‘ (cat) nah am Konzept ‚kuscheln‘ (cuddle) liegen, während der Vektor für ‚Stein‘ (stone) deutlich weiter entfernt ist. Eigene Darstellung (Created with ChatGPT).

Diese Organisation emergiert durch das Training. Die Position jedes Token passt sich kontinuierlich an seine Beziehungen zu allen anderen Token an. So repräsentiert das Modell implizit, dass ‚König‘ zu ‚Königin‘ in ähnlicher Relation steht wie ‚Mann‘ zu ‚Frau‘, ohne dass es eine explizite Regeldefinition gibt. Jedoch bedeutet dies nicht automatisch, dass das Modell diese Relationen auch umgekehrt zuverlässig herstellen kann. Während das Modell auf die Frage ‚Wer ist die Mutter von Tom Cruise?‘ antworten kann, scheitert es möglicherweise an der umgekehrten Relation ‚Wer ist der Sohn von Mary Lee Pfeiffer?‘, falls diese Verbindung nicht explizit gelernt wurde. Zumindest war dies noch ein Problem im Jahr 2024.<sup>15</sup> Diese dynamische Bedeutungsrepräsentation erklärt, warum LLMs kontextuell passende Fortsetzungen generieren

können und warum dabei Fehler entstehen. Ihr "Verstehen" ist grundlegend anders als das menschliche. Es gilt allerdings nicht zwangsläufig, dass Modelle, die ein scheinbar einfaches Problem A lösen, ein komplexeres Problem B automatisch ebenfalls lösen – oder umgekehrt.



3Blue1Brown. But what is a GPT? Visual intro to transformers. Chapter 5, Deep Learning.

<https://youtu.be/wjZofJX0v4M> (<https://youtu.be/wjZofJX0v4M?si=sVIkaUQBmG3fVuiB&t=915>)

Die Bedeutungen eines Wortes werden in einem großen Sprachmodell durch dessen Position im Vektorraum dargestellt. Viele Richtungen in diesem Vektorraum korrelieren dabei metaphorisch mit bestimmten Bedeutungsachsen oder Eigenschaften – wie etwa "Musik vs. Politik", "weiblich vs. männlich" oder "abstrakt vs. konkret". Isoliert sind einzelne Dimensionen jedoch kaum nutzbar. Ein Wort startet zwar immer mit einem festgelegten Initial-Embedding, doch sobald es in einem konkreten Satz verwendet wird, berechnet das Modell schichtweise neue, kontextabhängige Vektoren entlang dieser Achsen. Dadurch landet etwa das Wort "Queen" in einem Satz über britische Monarchie nahe bei Konzepten wie "Krone" und "Palast", in einem musikalischen Kontext nahe "Rock" und "Freddie Mercury", und im Zusammenhang mit Drag nahe Begriffen wie "Performance" oder "Make-up". Die Vielzahl solcher Bedeutungsdimensionen macht es möglich, dass Sprachmodelle Nuancen erfassen und die Bedeutung eines Wortes je nach Kontext flexibel abbilden können. Oder auch daran scheitern.

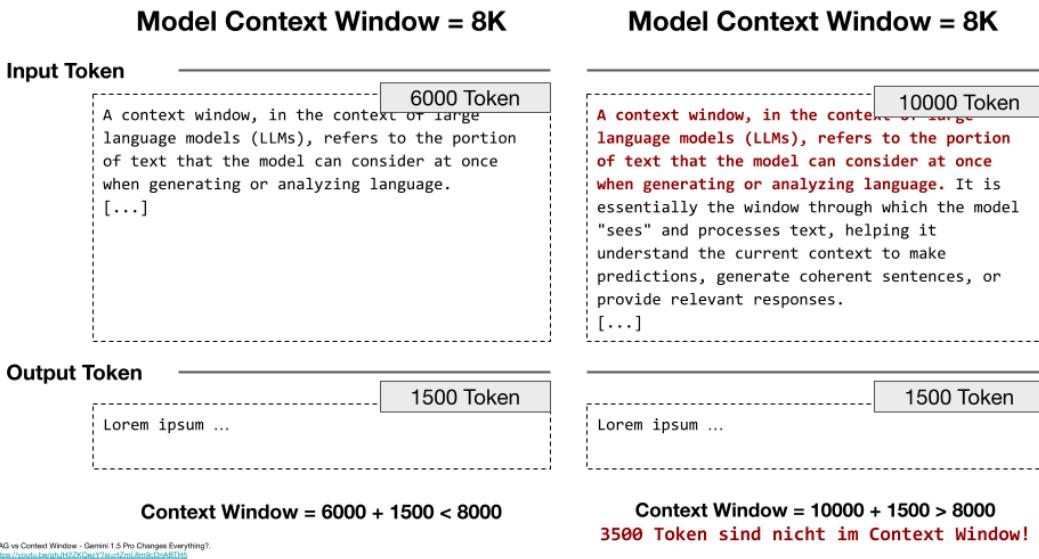
Diese Strukturen entstehen während des Trainings direkt aus den Daten und sind für Menschen nicht immer logisch nachvollziehbar. Unterschiedliche Sprach- und Schreibstile aktivieren implizit unterschiedliche Regionen im Embedding-Raum. So aktiviert "*The King doth wake tonight and takes his rouse*" stilistische Merkmale von Shakespeare, während "*The King wakes up tonight and begins his celebration*" moderne Sprachräume nutzt. Das hat einen Einfluss auf das Prompt Engineering. Ein präzise wissenschaftlicher Prompt navigiert das Modell anders als ein kreativ formulierter Prompt. Die gezielte Modellierung von Experten-Personas ("*You are an expert in...*") ist effektiv, da sie spezifische Verarbeitungswege im Modell aktiviert. Diese Wege wirken wie metaphorische Programme, die statistisch gelernt wurden, obwohl sie eigentlich keine expliziten Programme sind. Ich betrachte sie als Programme, die ich per Prompt ein-, aus- oder kombinieren kann.

## Context Window

Das *Context Window*<sup>16</sup> legt fest, wie viele Token ein Modell gleichzeitig "im Blick" behalten kann. Es funktioniert ähnlich wie ein temporäres Arbeitsgedächtnis, das sich allerdings an nichts erinnern kann. Während frühe Modelle wie GPT-2 auf 1.024 Token limitiert waren, können moderne Systeme wie Claude 3.5 bis zu 200.000 Token verarbeiten. Googles Gemini kann sogar eine bis zwei Millionen multimodale Token verarbeiten, darunter auch Bild-, Audio- und Videodaten.

Wird das Context Window überschritten, verliert das Modell den Zugriff auf frühere Informationen. Es muss dann aufgrund des verbleibenden Kontexts Vermutungen anstellen, was zu vermehrten Halluzinationen führen kann. Neben Input und Output kann das Context Window auch versteckte Systemanweisungen, externe Dokumente, Code-Schnipsel und Ergebnisse aus Retrieval-Mechanismen (RAG) enthalten, was die verfügbare Kapazität zusätzlich belastet. Alles außerhalb des Fensters ist für das Modell nicht mehr erreichbar, sondern höchstens indirekt über bereits gelerntes Wissen verfügbar. Um erneut darauf zugreifen zu können, ist ein externer Retrieval- oder Memory-Mechanismus nötig, der die entsprechenden Token wieder in das Context Window lädt. Dies kann ein LLM wiederum über Tool Use realisieren.

Ein größeres Context Window führt bei klassischen Transformers jedoch zu einem höheren Rechenaufwand, da die Last quadratisch mit der Tokenzahl steigt. Es ist also nicht nur teurer, sondern es kommt auch zu einem Leistungsabfall, wenn sie relevante Informationen aus der Mitte eines langen Kontextes berücksichtigen müssen. Zudem werden sie anfälliger für versteckte schädliche Eingaben (*Jailbreaking*<sup>17</sup>). Es zeigt sich jedoch auch, dass Context Windows theoretisch unendlich groß werden können.<sup>18</sup>



Die Abbildung zeigt die Grenze des Context Windows. Links passen 6 000 **Input-Token** plus 1 500 **Output-Token** problemlos in das 8K-Fenster. Es gibt also separate Obergrenzen für Input und Output.

Typischerweise wird ein Teil des Fensters für die Antwort reserviert, sodass die maximal mögliche Output-Länge kleiner ist als die erlaubte Eingabegröße – das ist aber eine Designentscheidung, kein Naturgesetz des Modells. Rechts hingegen wird mit 10 000 Input-Token plus 1 500 Output-Token die maximale Tokenanzahl überschritten. Die rot markierten 3 500 Token am Anfang des Inputs stehen während der Antwortgenerierung nicht mehr explizit zur Verfügung.<sup>19</sup> Kopiert man ein Buch hinein, und selbst wenn ganz am Anfang die Erde explodiert, verarbeitet das Modell diese Information nicht mehr aktiv, sobald das Context Window ausreichend gefüllt wurde. Allerdings kann das Modell dennoch scheinbar "wissen", dass die Erde explodierte, entweder weil diese Information Teil seines allgemeinen "Weltwissens" ist (z. B. bei bekannten Werken), oder weil es eine plausible Fortsetzung aus dem aktuellen Kontext extrapoliert ("halluziniert").

Dies unterscheidet sich fundamental davon, wenn die ursprünglichen Token tatsächlich noch aktiv im Context Window vorhanden wären.

## Training

"Wir züchten diese Modelle mehr, als dass wir sie bauen", formuliert es Dario Amodei.<sup>20</sup> Selbst die führenden LLM-Erzeuger wissen nicht exakt, wie ihre Modelle "tatsächlich" funktionieren oder welche exakte Konfiguration optimal ist. Vielmehr verstehen sie, welche Parameter in unterschiedlichen Trainingsphasen gezielt angepasst werden müssen.

Das Training von LLMs erfolgt typischerweise in drei Phasen, die das System jeweils auf spezifische Weise prägen. In der ersten, sehr ressourcenintensiven Phase, dem **Pre-Training**, lernen LLMs mittels selbstüberwachtem Lernen anhand von Billionen von Token aus verschiedensten Quellen. Ein Prozess, den Andrej Karpathy<sup>21</sup> als Kompression des "Internets als ZIP-Datei"<sup>22</sup> bezeichnet. Autoregressive Architekturen<sup>23</sup> wie GPT optimieren dabei die Vorhersage des nächsten Tokens (**Next-Token-Prediction**). Durch das Verarbeiten von Webseiten, wissenschaftlichen Publikationen, Code-Repositories und Büchern entstehen in den Milliarden von Parametern statistische Repräsentationen der Welt.<sup>24</sup> Was ich zuvor salopp als "Weltwissen" bezeichnete, ist im Sinne meiner System-1.42-Metapher eher eine verzerrte, mit Bias beladene Abstraktion von "Weltwissen". Wie sich Gravitation tatsächlich anfühlt, lässt sich durch keinen Text oder Video erlernen. Das Modell durchsucht bei einer Anfrage nicht das Web, sondern extrahiert Wahrscheinlichkeitsverteilungen. Es repräsentiert, wie Karpathy sagt, die "Gestalt" gelernter Texte. Dabei erfasst das Modell grobe syntaktische und semantische Muster, erwirbt aber noch keinerlei Spezialisierung.

Erst in der zweiten Phase, dem **Supervised Instruction Fine-Tuning**, erhält das Modell eine spezialisierte Aufgabe und konkrete Anweisungen dazu. Kuratierte Instruktion-Antwort-Paare verwandeln das rohe Basis-Modell in spezifische Anwendungsformen wie Chat- oder Coding-Modelle. In dieser Phase lernt das Modell, seine latenten statistischen Repräsentationen in konkrete, aufgabenorientierte und kohärente Outputs umzusetzen. Hier wird festgelegt, wie die Antworten aussehen sollen: kurz oder ausführlich, im JSON-Format oder in Form eines typischen Frage-Antwort-Spiels eines Chatbots.

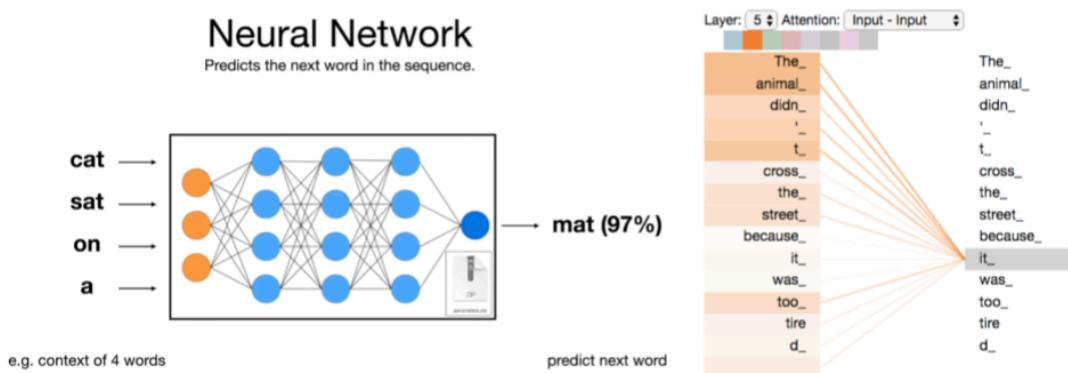
In der dritten Phase, dem **Reinforcement Learning<sup>25</sup> from Human Feedback<sup>26</sup>**, optimiert das Modell sein Verhalten anhand menschlicher Präferenzen. Dabei geht es nicht primär um faktenbasierte Korrektheit – obwohl diese durchaus gelegentlich verbessert wird – sondern vor allem um weichere, komplexere Qualitätsdimensionen wie Hilfsbereitschaft, Sicherheit oder Stil. Viele dieser Kriterien lassen sich nicht einfach fest definieren oder eindeutig labeln. Stattdessen bewerten Menschen Antworten vergleichend ("Antwort A ist besser als Antwort B"). So lernt das Modell, weiche Ziele wie "nützlich und sicher" anzustreben, ohne dass jede Eigenschaft separat und explizit definiert werden muss. Diese Phase birgt jedoch auch das Risiko, dass menschlicher Bias zusätzlich zum Bias in den Trainingsdaten direkt in das Modell übertragen wird.

Token als atomare Einheiten für die Syntax, Embeddings als semantische Repräsentationen und das Context Window als unmittelbarer Kontext bilden gemeinsam die technische Grundlage eines LLMs. Auf diesen Komponenten basiert unmittelbar die Transformer-Architektur mit ihrer Kernfunktion, der Next-Token-Prediction.

## Transformer und Next-Token-Prediction

LLMs basieren auf der sogenannten **Transformer-Architektur**<sup>27</sup>. Im Gegensatz zu älteren Modellen analysieren Transformer Texte nicht sequenziell Wort für Wort, sondern verarbeiten die Token innerhalb jeder Verarbeitungsschicht parallel. Dies gelingt durch den **Attention-Mechanismus**<sup>28</sup>, der berechnet, wie relevant jedes Token für die Bedeutung anderer Token im Text ist. Ein entscheidender Vorteil ist, dass Transformer auch Abhängigkeiten über große Distanzen hinweg erfassen können, etwa wenn sich eine Instruktion am Anfang des Prompts auf Inhalte bezieht, die erst Tausende Token später folgen. Der Attention-Mechanismus operiert innerhalb eines festgelegten Context Windows. Das Context Window begrenzt also die Zahl der sichtbaren Token, während Attention deren wechselseitige Relevanz berechnet.

Diese Architektur ermöglicht es LLMs, das nächste Token vorherzusagen, ein Prozess, der als Next-Token-Prediction bezeichnet wird. Bei der Eingabe "The cat sat on the..." berechnet das Modell aus den Attention-Mustern, dass "mat" mit hoher Wahrscheinlichkeit folgen sollte. Nicht, weil es weiß, dass Katzen auf Matten sitzen, sondern weil diese Wortfolge statistisch häufig auftritt. Der Attention-Mechanismus erlaubt es, komplexe Beziehungen zwischen entfernten Wörtern zu erfassen. Beispielsweise kann er erkennen, dass sich "it" auf das zuvor erwähnte "animal" bezieht. Aber wenn man alle möglichen Texte kennen würde, könnte man sich dann dem "richtigen" Kontext nicht ziemlich gut annähern?



*Next-Token-Prediction und Attention-Mechanismus. Links: Das neuronale Netz sagt basierend auf vier Eingabewörtern das nächste Wort vorher. Rechts: Der Attention-Mechanismus zeigt, wie das Modell*

*Beziehungen zwischen Wörtern herstellt. <https://jalammar.github.io/illustrated-transformer> (<https://jalammar.github.io/illustrated-transformer>)*

Da Modelle Token stets sequentiell erzeugen und gewählte Token nicht nachträglich ändern können, entsteht ein Effekt ähnlich einem **“Schmetterlingseffekt”**: Ein einzelnes Token zu Beginn beeinflusst den gesamten Kontext aller nachfolgenden Vorhersagen. Beispielsweise aktiviert “dog” als erstes Token andere statistische Muster als “cat”. Dadurch verändert sich die Wahrscheinlichkeitsverteilung nachfolgender Token erheblich. Hundethemen wie “Bellen” oder “Spaziergänge” werden wahrscheinlicher, während Katzenthemen wie “Kratzbäume” und “Schnurren” unwahrscheinlicher werden.<sup>29</sup>

Die Auswahl des nächsten Tokens wird durch drei Hauptparameter gesteuert: Die **Temperature** reguliert die Zufälligkeit der Ausgabe und reicht von eher deterministisch bei 0 bis “hochkreativ” bei 2. Alles ab 1 ist jedoch extrem “kreativ”. Eine höhere Temperature ermöglicht die Exploration eines breiteren Bereichs möglicher Lösungswege. Paradoxe Weise können dadurch bei komplexen Problemen bessere Ergebnisse erzielt werden, wenn auch weniger reproduzierbare, da das Modell aus lokalen Optima<sup>30</sup> befreit und unkonventionelle Ansätze erkundet werden. **Top-K** begrenzt die Auswahl auf die K wahrscheinlichsten Token und **Top-P** nur Token berücksichtigt, deren kumulative Wahrscheinlichkeit einen Schwellenwert P nicht überschreitet. Für die meisten Nutzer:innen genügt es, die Temperature einzustellen: niedrige Werte (0–0,4) für stabile, vorhersehbare Ergebnisse, höhere Werte (0,4–1) für kreativere, vielfältigere Ausgaben.<sup>31</sup>

## Autocomplete?

Was bedeutet es konkret, einen Kontext mit allen Details und Abhängigkeiten korrekt zu erfassen? Ilya Sutskever, Mitgründer von OpenAI, argumentiert, dass *Next-Token-Prediction* deutlich mehr umfasst als bloßes *Autocomplete*. Um zuverlässig das richtige nächste Token vorherzusagen, müsse ein System ein implizites “Verstehen der zugrunde liegenden Realität” entwickeln. Ein Sprachmodell konstruiert daher aus Kontextinformationen eine probabilistische Repräsentation möglicher Realitäten, indem es gelernte Muster und statistische Zusammenhänge rekonstruiert. Ein ausreichend intelligentes System könnte, so Sutskever weiter, sogar kreativ extrapolieren und sich dabei vorstellen, wie eine hypothetische Person mit “großer Einsicht und Weisheit” in einer gegebenen Situation handeln würde<sup>32</sup>, eine Vision, die Andrej Karpathys Charakterisierung von LLMs als “people spirits”<sup>33</sup>, also stochastischen Simulationen menschlichen Verhaltens, sehr ähnlich ist. Das ist natürlich eine metaphorische und stark vereinfachte, beinahe spirituell anmutende Darstellung. Als Denkkonzept hilft sie möglicherweise jedoch beim praktischen Umgang mit LLMs, selbst wenn sie stark vermenschlichend wirkt. Amanda Askell von Anthropic warnt übrigens vor zwei Extremen im Umgang mit LLMs: Sie entweder zu sehr oder zu wenig zu vermenschlichen. Der optimale

Mittelweg, bei dem die Systeme durch menschliche Kommunikationsmuster und Human Feedback trainiert wurden, führt zu besseren Ergebnissen.<sup>34</sup>

Zwar bilden Sprachmodelle durchaus breit angelegte konzeptuelle Kategorien, die jenen ähnlich sind, die Menschen intuitiv nutzen. Doch bei feineren semantischen Unterscheidungen bleibt ihre interne Repräsentation oft oberflächlich. Im Gegensatz zur menschlichen Wahrnehmung speichern Sprachmodelle Muster in stark komprimierter Form. Dabei gehen feine Unterschiede und adaptive Details verloren, wodurch das von Modellen aufgebaute Bild der Welt zwar plausibel, jedoch vergleichsweise grob bleibt.<sup>35</sup> Es gibt allerdings konkrete Hinweise darauf, dass Modelle intern komplexe, mehrstufige Prozesse vollziehen. Beispielsweise aktivieren Sprachmodelle zunächst Zwischenschritte oder Zwischenrepräsentationen, bevor sie zu einer endgültigen Ausgabe gelangen. Soll etwa die Hauptstadt eines Bundesstaates genannt werden, in dem sich eine bestimmte Stadt befindet (wie Dallas), aktiviert Claude 3.5 intern zunächst eine Zwischendarstellung ("Texas") und liefert dann die passende Antwort ("Austin"). Ähnliches gilt für Aufgaben wie das Erzeugen von Reimen. Modelle berücksichtigen intern frühzeitig mögliche Reimwörter, die den Satzbau wesentlich beeinflussen.<sup>36</sup>

## Emergenz? Grokking?

Führt das Ermitteln des nächsten Token in einem ausreichend großen und im Post-Training adaptierten neuronalen Netz tatsächlich zu Zwischenschritten und Vorausplanen? Lässt sich das noch als bloßes Autocomplete erklären? Tatsächlich konnten große Sprachmodelle ab einer kritischen Größe plötzlich komplexe logische und arithmetische Aufgaben lösen, die kleinere Modelle nicht bewältigten.<sup>37</sup> Solche Fähigkeitssprünge bezeichnet man als **emergente Eigenschaften**, definiert als Fähigkeiten, die bei kleinen Modellen nicht vorhanden sind und erst ab einer bestimmten Parameterzahl sichtbar werden, ohne dass dies durch lineare Skalierung vorhersehbar ist.<sup>38</sup>

Bei großen Modellen scheinen Fähigkeiten oft abrupt aufzutreten, was jedoch meist auf binäre Messmethoden ("kann/kann nicht") zurückzuführen ist, während die tatsächliche Entwicklung kontinuierlich verläuft.<sup>39</sup> Dennoch existieren empirisch belegte Fälle, in denen Modelle komplexe interne Abläufe entwickeln, die über reines Autocomplete hinausgehen. Diese Fortschritte sind allerdings stark abhängig von Trainingsbedingungen.<sup>40</sup> Daher bleibt offen, ob es sich um echte Emergenz handelt oder lediglich um eine überzeugende Simulation dank umfangreicherer Trainingsdaten. Denn Modelle simulieren nicht nur reines "Weltwissen", sondern entwickeln zunehmend unterschiedliche interne "Programme", also spezifische Problemlösungsstrategien. Eines dieser Problemlösungsprogramme ist übrigens "Let's think step by step". Dieses sogenannte "Chain-of-Thought" (CoT) wurde erst nachträglich entdeckt und wird häufig als emergente Eigenschaft interpretiert, ist allerdings weiterhin Gegenstand aktueller Debatten.

Ein neuronales Netz zeigt beim sogenannten **Grokking**<sup>41</sup> nach abgeschlossenem Training plötzlich einen Übergang von reiner Memorierung zu echter Generalisierung. Dieses Phänomen trat bislang hauptsächlich bei speziellen, künstlich erzeugten Aufgaben auf, bei denen kleine Datensätze, genau eingestellte Trainingsparameter sowie eine starke Regulierung entscheidend waren. Grokking wird jedoch zunehmend als mögliches Trainingsartefakt interpretiert. Unklar bleibt weiterhin, ob neuronale Netze tatsächlich abstrakte Regeln generalisieren oder nur überzeugend simulieren. Diese Unsicherheit zeigt sich besonders deutlich bei der umstrittenen Fähigkeit des sogenannten "Reasonings", also logischer Schlussfolgerungen, die typischerweise bei sehr großen Modellen untersucht wird. Dabei bleibt offen, ob Reasoning eine echte emergente Eigenschaft skalierender Modelle darstellt oder vielleicht eher ein Grokking-ähnliches Phänomen ist, also ein plötzlicher Durchbruch nach bereits abgeschlossenem Training.

## Reasoning?

LLMs rufen bei Reasoning-Aufgaben selten direkte Antworten aus Faktenwissen ab, sondern greifen primär auf prozedurales Wissen zurück – etwa auf gespeicherte Lösungsstrategien, Codes, Formeln oder Schritt-für-Schritt-Anleitungen.<sup>42</sup> Doch reicht es aus, lediglich das prozedurale Schema einer Problemlösung zu kennen, um von "echtem" Reasoning zu sprechen? François Chollet, der Intelligenz als "die Effizienz eines Systems im Erwerb neuer Fähigkeiten zur Bewältigung spezifischer Situationen" definiert, unterscheidet Reasoning klar davon. Er versteht darunter explizit "die Fähigkeit, neue Programme (aus vorhandenen Programmteilen) für bisher ungesehene Aufgaben zu synthetisieren". Intelligentes Verhalten verlangt demnach mehr als bloßes Schemawissen: Es erfordert, erworbene Fähigkeiten auf neuartige Weise zu kombinieren und dadurch Probleme zu lösen, die zuvor unbekannt waren.

Das Paper *The Illusion of Thinking* zeigt, dass Large Reasoning Models mit steigender Komplexität der Aufgaben zunächst scheinbar besser, anschließend jedoch zunehmend schlechter performen und schließlich vollständig scheitern.<sup>43</sup> Dies scheint Chollets<sup>44</sup> und Kambhampatis<sup>45</sup> Argument zu unterstützen, wonach Modelle primär auf memorierte Prozeduren zurückgreifen und nicht genuin logisch schlussfolgern können. Allerdings untersucht diese Studie nur isolierte Modelle, ohne Tool Use und ohne zusätzliche Test-Time Compute, zwei Kernelemente moderner Frontier-LLMs. Sie bietet daher keine umfassende Bewertung des aktuellen Standes. Zum Vergleich: OpenAIs o3-Modell erzielte auf Chollets ARC-AGI<sup>46</sup> Benchmark, der speziell für neuartige, abstrakte Reasoning-Aufgaben entwickelt wurde, bemerkenswerte 87,5 %. Dieses Ergebnis veranlasste Chollet sogar dazu, seine ursprüngliche Einschätzung der Modelle von "nicht intelligent" zu "nicht nicht-intelligent" zu revidieren.

Chollet insistiert auf einer anspruchsvollen Definition von Reasoning als "On-the-fly-Synthese neuer Programme aus vorhandenen Programmteilen für bisher ungesehene Aufgaben". LLMs operieren jedoch

fundamental anders: Sie generieren sequenziell Token, um Kontexte schrittweise anzureichern und daraus statistisch wahrscheinlichere Folgetoken abzuleiten. Chollet sieht LLMs nicht als diskrete Programmgeneratoren im klassischen Sinn, doch er bezeichnet ihre brute-force Suche über Chain-of-Thought-Pfade durchaus als eine Form von Programmsynthese. In seinen Worten: “Program synthesis is a tree-search process. Use LLMs to guide this tree-search process.” Diese tokenbasierte Strategie stößt allerdings bei komplexen, kompositorischen Aufgaben an Grenzen, wie unter anderem das Apple-Paper oder ARC-AGI-2 zeigen.<sup>47</sup> Chollet selbst charakterisiert o3<sup>48</sup> als “natürlichsprachlichen Suchprozess, der den kompletten Raum möglicher Chain-of-Thought Lösungen durchsucht”.

Noam Brown, einer der Hauptentwickler, demonstrierte bereits in früheren Arbeiten, dass zusätzliche Rechenzeit bei der Inferenz, also während der Tokengenerierung, einem Skalierungseffekt von bis zu 100.000x entspricht. Der entscheidende Durchbruch bei OpenAIs Modell o1 basiert auf der Erkenntnis des **“Verifier-Generator Gap”**: Bei vielen Reasoning-Aufgaben ist es einfacher, Lösungen zu verifizieren als sie zu generieren, ähnlich wie beim Sudoku, wo die Überprüfung deutlich leichter ist als die eigentliche Lösungsfindung. Modelle wie o1 und o3 nutzen dies gezielt aus und generieren durch **Reinforcement Learning** zahlreiche “Denkschritte” in Form von “Thinking Token”, erkennen Fehler und optimieren Lösungswege. Was wie ein chaotischer Prozess wirkt, ist eine Form der Exploration des Lösungsraums, die durch die inhärente Verifikationsfähigkeit von LLMs gesteuert wird.<sup>49</sup> Diese strukturierte Exploration erlaubt es ihnen, tatsächlich über die reine Interpolation von Trainingsdaten hinauszugehen und echte Extrapolation zu betreiben, indem sie unbekannte Bereiche des Lösungsraums navigieren (“uncharted areas”).

Die massive Skalierung von Test Time Compute verstärkt diese Fähigkeiten: Mehr Rechenzeit erlaubt sowohl breitere Exploration (mehr generierte Pfade) als auch präzisere Selektion (bessere Verifikation). Ob dies zu “echtem” Reasoning führt, bleibt offen – aber die funktionale Äquivalenz wird zunehmend ununterscheidbar. LLMs bleiben Next-Token-Predictors, doch durch die Orchestrierung von Generierung und Verifikation bei massiver Skalierung entsteht ein System, das erfolgreich Reasoning-Aufgaben löst, auch wenn der zugrundeliegende Mechanismus fundamental anders ist als menschliches Denken.

Die Fähigkeit von LLMs, sogenannte **Halluzinationen**<sup>50</sup> zu erzeugen, also Informationen zu generieren, die über das explizit Gelernt hinausgehen, ist kein Fehler, sondern eine fundamentale Eigenschaft ihrer Architektur. Halluzinationen sind problematisch, wenn sie falsche oder irreführende Inhalte erzeugen. Sie können jedoch auch wertvoll sein, wenn sie kreative Ideen oder neuartige Lösungsansätze hervorbringen, die nicht direkt aus den Trainingsdaten ableitbar sind. LLMs können Halluzinationen nicht vollständig vermeiden, da sie diese als explorativen Mechanismus benötigen, um komplexe Aufgaben zu lösen. Deshalb sind sorgfältig konzipierte Workflows notwendig, um Ergebnisse systematisch zu verifizieren, den Kontext

aktiv zu managen und Halluzinationen gezielt zu steuern. Dabei müssen wir uns bewusst sein, dass es sich um halluzinierende "System-1.42-Systeme" handelt und nicht um menschliche Intelligenz.<sup>51</sup>

Verstehen und Denken sind Eigenschaften, die wir Menschen vorbehalten wollen und sollen. Doch LLMs stellen diese Grenzen infrage, indem sie Verhalten simulieren, das Denken ähnelt, ohne dabei echtes Verständnis oder zuverlässige Selbstprüfung zu besitzen. Mit dem Begriff "System 1.42" möchte ich diese Ambivalenz hervorheben. Die entscheidende Parallele zu Douglas Adams' *Per Anhalter durch die Galaxis* ist dabei weniger die berühmte Antwort "42", sondern vielmehr die Reaktion der "Gewerkschaft der Philosophen und Denker", die den Supercomputer Deep Thought abschalten möchte, weil er ihre Existenzberechtigung bedroht. Wenn LLMs funktional nicht mehr von menschlichem Reasoning, Coding oder Forschen unterscheidbar sind, dann sind ihre Auswirkungen dieselben. Die Zukunft liegt deshalb nicht darin, LLMs zu hypen, zu mystifizieren oder zu verteufeln, sondern darin, sie kritisch und gezielt als halluzinierende Reasoner einzusetzen. Verschwinden werden sie nicht mehr, höchstens durch etwas Besseres ersetzt. Ihr Einfluss auf die Wissensarbeit – und weniger auf Routineprozesse – wird aus meiner Sicht bislang noch nicht ausreichend diskutiert, obwohl mit Stand Juli 2025 eine weitere Skalierung und Verbesserung sehr wahrscheinlich ist. Wie die System-1.42-Modelle selbst befindet sich mich daher in einem Zwischenzustand zwischen "Don't Panic!" und "Panic!". Diese Ambivalenz ist vielleicht die einzige ehrliche Haltung gegenüber einer Technologie, deren Evolution schneller verläuft, als unser Verstehen folgen kann.

1. Die Autor:innen argumentieren, dass LLMs lediglich linguistische Form manipulieren, ohne Zugang zu Bedeutung zu haben, und warnen vor den Risiken immer größerer Modelle. Bender, Emily M., Timnit Gebru, Angelina McMillan-Major, und Shmargaret Shmitchell. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? ". In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 610–23. FAccT '21. New York, NY, USA: Association for Computing Machinery, 2021. <https://doi.org/10.1145/3442188.3445922> (<https://doi.org/10.1145/3442188.3445922>). ↪
2. Überblicksarbeit, die zeigt, wie aktuelle Großmodelle dank Such-, Belohnungs- und Selbstverbesserungsverfahren vom reinen Musterabgleich zu "Reasoning" gelangen – und zugleich erklärt, wo diese neuen "Reasoning"-Fähigkeiten ihre Grenzen haben. Li,

Zhong-Zhi, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, et al. 'From System 1 to System 2: A Survey of Reasoning Large Language Models'. arXiv, 5 June 2025. <https://doi.org/10.48550/arXiv.2502.17419> (<https://doi.org/10.48550/arXiv.2502.17419>). ↪

3. Die Skalierungsthesen beschreiben den systematischen, mathematisch vorhersehbaren Zusammenhang zwischen der Leistungsfähigkeit von Sprachmodellen und den drei zentralen Faktoren: Modellgröße (Anzahl der Parameter), Umfang der Trainingsdaten und eingesetzte Rechenleistung. Sie besagen, dass die Modellleistung bei gezielter Vergrößerung dieser Faktoren nicht zufällig, sondern durch einfache mathematische Potenzgesetze verbessert wird; somit erlauben Skalierungsthesen präzise Prognosen über optimale Modellgröße, Datenmenge und Rechenzeit, um maximale Leistung effizient zu erreichen. Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 'Scaling Laws for Neural Language Models'. arXiv, 23 January 2020. <https://doi.org/10.48550/arXiv.2001.08361> (<https://doi.org/10.48550/arXiv.2001.08361>). ↪
4. Shojaee, Parshin, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, und Mehrdad Farajtabar. "The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity", 2025. <https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf> (<https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf>) ↪
5. Zur Veranschaulichung grundlegender Unterschiede zwischen menschlicher Kognition und den probabilistischen Mechanismen von LLMs, insbesondere hinsichtlich Lernen, Verarbeitung, Gedächtnis und vermeintlichem Reasoning. IBM Technology. AI vs Human Thinking: How Large Language Models Really Work. <https://youtu.be/-ovM0daP6bw> (<https://youtu.be/-ovM0daP6bw>) ↪
6. Beispielsweise scheitern LLMs an einfachen räumlichen und zeitlichen Aufgaben. ↪
7. Agentic Systems sind allgemein gesagt KI-Systeme, die modular aufgebaute Komponenten kombinieren, typischerweise einschließlich großer KI-Modelle, um

Aufgaben durch Planung, Entscheidungsfindung, Werkzeugnutzung und iterative Selbstreflexion zu lösen. Hu, Shengran, Cong Lu, and Jeff Clune. ‘Automated Design of Agentic Systems’. arXiv, 2 March 2025. <https://doi.org/10.48550/arXiv.2408.08435> (<https://doi.org/10.48550/arXiv.2408.08435>). ↪

8. Tool Use bezeichnet die Fähigkeit von LLMs, externe Funktionen und APIs während der Textgenerierung aufzurufen und deren Ergebnisse zu integrieren. ↪
9. Kahneman, Daniel. Thinking, fast and slow. New York, NY, US: Farrar, Straus and Giroux, 2011. ↪
10. Lightman, Hunter, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. ‘Let’s Verify Step by Step’. arXiv, 31 May 2023. <https://doi.org/10.48550/arXiv.2305.20050> (<https://doi.org/10.48550/arXiv.2305.20050>). ↪
11. Oxford Professor: AIs are strange new minds. <https://youtu.be/35r0iSajXjA> (<https://youtu.be/35r0iSajXjA>) ↪
12. Yann LeCun argumentiert, dass autoregressive LLMs “fundamental fehlerhaft” seien und “in ein paar Jahren niemand mehr verwendet werden wird”. Als Alternative schlägt er Energy-based Models mit “Inference by Optimization” vor, die echte Validierung durch iterative Optimierung ermöglichen würden. LeCun betont, dass solche Systeme “System 2”-ähnliche Aufgaben durch mehrstufige Optimierungsprozesse lösen könnten - im Gegensatz zu LLMs, die nur durch “Chain of Thoughts” (mehr Token generieren) komplexere Probleme angehen können. Yann LeCun “Mathematical Obstacles on the Way to Human-Level AI”. <https://youtu.be/ETZfkkv6V7Y> (<https://youtu.be/ETZfkkv6V7Y?si=WLOGyXNViqql6wP7>) ↪
13. Andrej Karpathy. Let’s build the GPT Tokenizer. <https://youtu.be/zduSFxRajkE> (<https://youtu.be/zduSFxRajkE?si=kFYNuF1fXCPUUpOs6>) ↪

14. Es gibt sprach- oder domänen spezifische Tokenizer. ↪
15. Bei Frontier-Modellen mit Websuche dürfte das aber deutlich besser funktionieren.  
Berglund, Lukas, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, und Owain Evans. "The Reversal Curse: LLMs trained on ‚A is B‘ fail to learn ‚B is A‘". arXiv, 26. Mai 2024. <https://doi.org/10.48550/arXiv.2309.12288>. ↪
16. What is a Context Window? Unlocking LLM Secrets. <https://youtu.be/-QVoIxEpFkM> (<https://youtu.be/-QVoIxEpFkM>) ↪
17. ChatGPT Jailbreak. Computerphile. <https://youtu.be/zn2ukSnDqSg> (<https://youtu.be/zn2ukSnDqSg>) ↪
18. Munkhdalai, Tsendsuren, Manaal Faruqui, und Siddharth Gopal. 2024. Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention. arXiv. <https://doi.org/10.48550/arXiv.2404.07143> (<https://doi.org/10.48550/arXiv.2404.07143>) ↪
19. Es gibt allerdings Mechanismen, um diese Limitierung zu umgehen. Prompt Caching beispielsweise ist eine API-Funktion, bei der häufig verwendete Prompt-Teile (wie Systemanweisungen oder Dokumente) serverseitig zwischengespeichert werden, wodurch sie nicht bei jeder Anfrage erneut das Context Window belegen. <https://www.anthropic.com/news/prompt-caching> (<https://www.anthropic.com/news/prompt-caching>) ↪
20. The Future of U.S. AI Leadership with CEO of Anthropic Dario Amodei. <https://www.youtube.com/watch?v=esCSpbDPJik> (<https://www.youtube.com/watch?v=esCSpbDPJik>) ↪
21. Andrej Karpathy. Deep Dive into LLMs like ChatGPT. <https://youtu.be/7xTGNNLPyMI> (<https://youtu.be/7xTGNNLPyMI>). Andrej Karpathy. How I use LLMs. <https://youtu.be/EWvNQjAaOHw> (<https://youtu.be/EWvNQjAaOHw>). Andrej Karpathy. [1hr Talk] Intro to Large Language Models. [https://www.youtube.com/watch?v=zjkBMFhNj\\_g](https://www.youtube.com/watch?v=zjkBMFhNj_g) ([https://www.youtube.com/watch?v=zjkBMFhNj\\_g](https://www.youtube.com/watch?v=zjkBMFhNj_g)) ↪

22. Allerdings hinkt der Vergleich an einer entscheidenden Stelle: Im Gegensatz zu einer ZIP-Datei erfolgt die "Dekompression" bei LLMs nicht verlustfrei, da es unmöglich ist, aus dem trainierten Modell sämtliche ursprünglichen Texte exakt wiederherzustellen. Dennoch halte ich den Vergleich für ein anschauliches und didaktisch sinnvolles Bild. ↪
23. Modelle, die Sequenzen Token für Token generieren, wobei jedes neue Token nur auf Basis aller vorherigen Token vorhergesagt wird, ohne spätere Revision. ↪
24. Hoffmann, J. et al. (2022). "Training Compute-Optimal Large Language Models". arXiv:2203.15556. Chinchilla-Skalierungsgesetze zeigen optimale Verhältnisse von Modellgröße zu Trainingsdaten. ↪
25. Reinforcement Learning (RL) bezeichnet einen Bereich des maschinellen Lernens, bei dem ein Agent durch Ausprobieren lernt, optimale Entscheidungen zu treffen. Der Agent erhält nach jeder Handlung Rückmeldungen ("Rewards"), die anzeigen, wie gut die Aktion war, und nutzt diese Erfahrungen, um zukünftige Entscheidungen so anzupassen, dass die langfristig akkumulierte Belohnung maximiert wird. Es wird dabei keine explizite Lösung vorgegeben; der Agent lernt allein durch die Interaktion mit seiner Umgebung. Reinforcement Learning. Computerphile. [https://youtu.be/844U9T\\_SOrA](https://youtu.be/844U9T_SOrA) ↪
26. Reinforcement Learning from Human Feedback (RLHF) Explained. [https://youtu.be/T\\_X4XFwKX8k](https://youtu.be/T_X4XFwKX8k) ↪
27. Eine sehr anschauliche Darstellung, die zeigt, wie Transformer funktionieren, gibt es bei 3Blue1Brown. But what is a GPT? Visual intro to transformers. Chapter 5, Deep Learning. <https://www.3blue1brown.com/lessons/gpt> (https://www.3blue1brown.com/lessons/gpt) ↪
28. Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, und Illia Polosukhin. "Attention Is All You Need". v7. arXiv,

2017. <https://doi.org/10.48550/arXiv.1706.03762> (<https://doi.org/10.48550/arXiv.1706.03762>) 2). ↵
29. Ethan Mollick. Thinking Like an AI. <https://www.oneusefulthing.org/p/thinking-like-a-n-ai> (<https://www.oneusefulthing.org/p/thinking-like-a-n-ai>) ↵
30. Ein lokales Optimum ist ein Ergebnis, das im unmittelbaren Umfeld schon das Beste ist – wenn man nur einen kleinen Schritt nach links, rechts, oben oder unten macht, wird es nicht besser. Doch weiter weg könnte es noch ein besseres Ergebnis geben. ↵
31. Lee Boonstra. Prompt Engineering. [https://www.gptaiflow.tech/assets/files/2025-01-18-pdf-1-TechAI-Google-whitepaper\\_Prompt%20Engineering\\_v4-af36dcc7a49bb7269a58b1c9b89a8ae1.pdf](https://www.gptaiflow.tech/assets/files/2025-01-18-pdf-1-TechAI-Google-whitepaper_Prompt%20Engineering_v4-af36dcc7a49bb7269a58b1c9b89a8ae1.pdf) ([https://www.gptaiflow.tech/assets/files/2025-01-18-pdf-1-TechAI-Google-whitepaper\\_Prompt%20Engineering\\_v4-af36dcc7a49bb7269a58b1c9b89a8ae1.pdf](https://www.gptaiflow.tech/assets/files/2025-01-18-pdf-1-TechAI-Google-whitepaper_Prompt%20Engineering_v4-af36dcc7a49bb7269a58b1c9b89a8ae1.pdf)) ↵
32. Ilya Sutskever (OpenAI Chief Scientist) - Why Next-Token Prediction Could Surpass Human Intelligence. <https://youtu.be/Yf1o0TQzry8> (<https://youtu.be/Yf1o0TQzry8?si=lj8B8UaESaDjTFgM>). Why next-token prediction is enough for AGI - Ilya Sutskever (OpenAI Chief Scientist). [https://youtu.be/YEUclZdj\\_Sc](https://youtu.be/YEUclZdj_Sc) ([https://youtu.be/YEUclZdj\\_Sc](https://youtu.be/YEUclZdj_Sc)) ↵
33. Andrej Karpathy: Software Is Changing (Again). <https://youtu.be/LCEmiRjPEtQ> (<https://youtu.be/LCEmiRjPEtQ>) ↵
34. Interview mit Amanda Askell. Claude's Character. <https://youtu.be/ugvHCXCOmm4> (<https://youtu.be/ugvHCXCOmm4>) ↵
35. Shani, Chen, Dan Jurafsky, Yann LeCun, and Ravid Schwartz-Ziv. 'From Token to Thoughts: How LLMs and Humans Trade Compression for Meaning'. arXiv, 26 May 2025. <https://doi.org/10.48550/arXiv.2505.17117> (<https://doi.org/10.48550/arXiv.2505.17117>) 7). ↵

36. Lindsey†, Authors Jack, Wes Gurnee\*, Emmanuel Ameisen\*, Brian Chen\*, Adam Pearce\*, Nicholas L. Turner\*, Craig Citro\*, et al. ‘On the Biology of a Large Language Model’. Transformer Circuits. Accessed 25 May 2025. <https://transformer-circuits.pub/2025/attribution-graphs/biology.html> (<https://transformer-circuits.pub/2025/attribution-graphs/biology.html>). ↪
37. Chowdhery et al. (2022) zeigen, dass Googles PaLM-Sprachmodell erst nach einer erheblichen Erhöhung der Parameterzahl (auf 540 Milliarden) plötzlich Fähigkeiten erlangte, wie etwa die Lösung komplexer logischer und arithmetischer Aufgaben, die kleinere Modelle nicht erreichten. Chowdhery, Aakanksha, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, u. a. “PaLM: Scaling Language Modeling with Pathways“. arXiv, 5. Oktober 2022. <https://doi.org/10.48550/arXiv.2204.02311> (<https://doi.org/10.48550/arXiv.2204.02311>). ↪
38. Wei, Jason, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, u. a. “Emergent Abilities of Large Language Models“. arXiv, 26. Oktober 2022. <https://doi.org/10.48550/arXiv.2206.07682> (<https://doi.org/10.48550/arXiv.2206.07682>). ↪
39. Schaeffer, Rylan, Brando Miranda, und Sanmi Koyejo. “Are Emergent Abilities of Large Language Models a Mirage?“ arXiv, 22. Mai 2023. <https://doi.org/10.48550/arXiv.2304.15004> (<https://doi.org/10.48550/arXiv.2304.15004>). ↪
40. Zhao, Rosie, Tian Qin, David Alvarez-Melis, Sham Kakade, und Naomi Saphra. “Distributional Scaling for Emergent Capabilities“. arXiv, 27. Mai 2025. <https://doi.org/10.48550/arXiv.2502.17356> (<https://doi.org/10.48550/arXiv.2502.17356>). ↪
41. AI Explained. Emergent Behaviors and Grokking. <https://www.coursera.org/learn/8-most-controversial-terms-in-ai-explained/lecture/u07Y4/emergent-behaviors-and-grokking-part-1> (<https://www.coursera.org/learn/8-most-controversial-terms-in-ai-explained/lecture/u07Y4/emergent-behaviors-and-grokking-part-1>). Power, Alethea, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. ‘Grokking: Generalization Beyond Overfitting on

- Small Algorithmic Datasets'. Accessed 3 June 2025. <https://arxiv.org/abs/2201.02177> (<https://arxiv.org/abs/2201.02177>). ↪
42. Ruis, Laura, Maximilian Mozes, Juhan Bae, Siddhartha Rao Kamalakara, Dwarak Talupuru, Acyr Locatelli, Robert Kirk, Tim Rocktäschel, Edward Grefenstette, und Max Bartolo. 2025. "Procedural Knowledge in Pretraining Drives Reasoning in Large Language Models". arXiv. <https://doi.org/10.48550/arXiv.2411.12580> (<https://doi.org/10.48550/arXiv.2411.12580>). ↪
43. Shojaee, Parshin, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, und Mehrdad Farajtabar. "The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity", 2025. <https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf> (<https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf>). Für kritische Perspektiven zur Methodik siehe Kevin A. Bryan. <https://x.com/Afinethorem/status/1931853801293484358> (<https://x.com/Afinethorem/status/1931853801293484358>) und AI Explained. Apple's 'AI Can't Reason' Claim Seen By 13M+, What You Need to Know. <https://youtu.be/wPBD6wTap7g> (<https://youtu.be/wPBD6wTap7g>) ↪
44. François Chollet. "ARC-AGI: Evaluating Real General Intelligence". <https://arcprize.org/blog/oai-o3-pub-breakthrough> (<https://arcprize.org/blog/oai-o3-pub-breakthrough>) ↪
45. Subbarao Kambhampati. Diskussion zur Debatte um Reasoning bei LLMs. LinkedIn, 2024. [https://www.linkedin.com/posts/subbarao-kambhampati-3260708\\_Thoughts-on-that-Apple-activity-7337469667478786048-ODVi](https://www.linkedin.com/posts/subbarao-kambhampati-3260708_Thoughts-on-that-Apple-activity-7337469667478786048-ODVi) ([https://www.linkedin.com/posts/subbarao-kambhampati-3260708\\_Thoughts-on-that-Apple-activity-7337469667478786048-ODVi](https://www.linkedin.com/posts/subbarao-kambhampati-3260708_Thoughts-on-that-Apple-activity-7337469667478786048-ODVi)) ↪
46. Chollet, François. "ARC-AGI: A Benchmark for General Intelligence". <https://arcprize.org> (<https://arcprize.org>) ↪
47. Chollet et al. (2025): ARC-AGI-2: A New Challenge for Frontier AI Reasoning Systems, arXiv:2505.11831v1 [cs.AI], online verfügbar unter <https://arxiv.org/abs/2505.11831> (<https://arxiv.org/abs/2505.11831>). ↪

48. OpenAI. “Introducing OpenAI o3”. Dezember 2024. <https://openai.com/index/openai-o3> (<https://openai.com/index/openai-o3>) ↪
  49. Learning to Reason with LLMs. [https://www.youtube.com/live/Gr\\_eYXdHFis](https://www.youtube.com/live/Gr_eYXdHFis) ([https://www.youtube.com/live/Gr\\_eYXdHFis](https://www.youtube.com/live/Gr_eYXdHFis)) ↪
  50. Banerjee, Sourav, Ayushi Agarwal, and Saloni Singla. LLMs Will Always Hallucinate, and We Need to Live With This. arXiv, 9. September 2024. <https://doi.org/10.48550/arXiv.2409.05746> (<https://doi.org/10.48550/arXiv.2409.05746>). Siehe auch kritische Diskussion: <https://x.com/Afinetheorem/status/1931853801293484358> (<https://x.com/Afinetheorem/status/1931853801293484358>) ↪
  51. Vgl. dazu die kritische Sichtweise in dem Video, das argumentiert, dass heutige LLMs unter ihrer Oberfläche *fractured entangled representations* besitzen und lediglich funktional Intelligenz vortäuschen. Echte, modulare und tiefere Repräsentationen könnten nur durch alternative Methoden wie offene, evolutionäre Suchprozesse entstehen. LLMs seien somit nützliche, aber letztlich oberflächliche *Impostors* auf dem Weg zu echter KI. AI doesn’t work the way you think it does. <https://youtu.be/o1q6Hz0MAg> (<https://youtu.be/o1q6Hz0MAg>) ↪
- 

## Zitervorschlag (APA)

Pollin, C. (2025, July 01). System 1.42: Wie (Frontier-)LLMs “tatsächlich” funktionieren. *Digital Humanities Craft*. <https://dhcraft.org/excellence/blog/System1-42/>