

Crash Kurs KI

„Kritische KI-Kompetenz sollte integraler Bestandteil wissenschaftlicher Integrität und philosophischer Bildung sein.“

Leitlinien zum Umgang mit generativer KI am Institut für Philosophie der FU Berlin

**Montag
22.6.2026
12:15 Uhr**

Grundlagen

Symbolische vs. subsymbolisch KI, Künstliche neuronale Netze, Interpretierbarkeit (Vortrag: Miguel Hoeltje)

**Montag
29.6.2026
12:15 Uhr**

Large Language Models und Chatbots

Lern-Paradigmen, Transformer-Architektur, LLMs vs. Chatbots, Reasoner-Models, Multimodalität (Vortrag: Miguel Hoeltje)

**Montag
6.7.2026
12:15 Uhr**

Über Large Language Models hinaus

Agentische KI, Vision-Language-Action Modelle, „Bitter Lesson“ & „Era of Experience“, World Models, Neurosymbolische KI (Vortrag: Miguel Hoeltje)

**Montag
13.7.2026
12:15 Uhr**

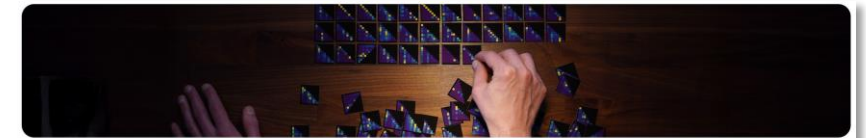
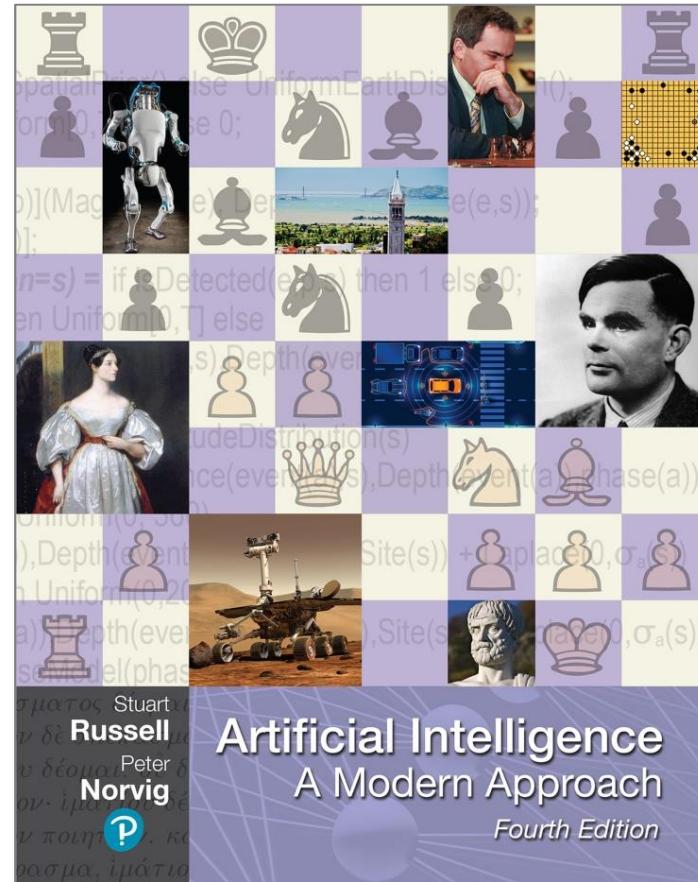
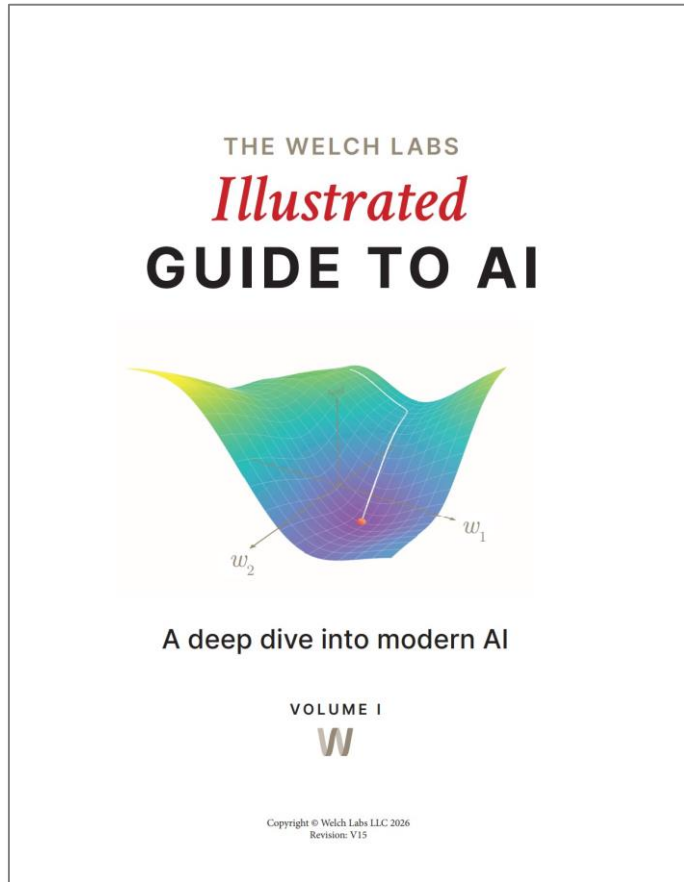
Zur ethischen Dimension der Entwicklung und Verwendung von KI

(Vortrag: Luise Müller)

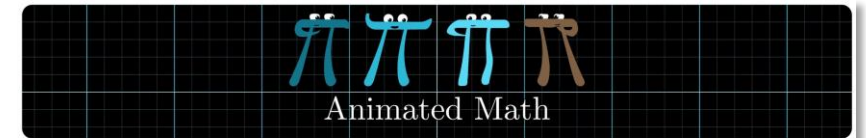
**Folien, Handouts, ggfs.
aktuelle Infos, etc.?**

▶ www.crash-ai.de ◀

Hilfreiche Lektüre und Videos:



Welch Labs
@WelchLabs • 874K subscribers • 135 videos
New Book! The Welch Labs Illustrated Guide to AI is now available for pre-order: <https://welchlabs.com/resources> and 1 more link
Subscribed



3Blue1Brown ✓
@3Blue1Brown • 8.34M subscribers • 234 videos
My name is Grant Sanderson. Videos here cover a variety of topics in math, or adjacent fields...more
3blue1brown.com and 7 more links
Subscribed



Primer ✓
@PrimerBiobs • 1.94M subscribers • 24 videos
Attempting to communicate the deep ideas of academic subjects...more
patreon.com/primerlearning and 2 more links
Subscribed

Hilfreiche Lektüre und Videos:

 Stanford Encyclopedia of Philosophy

Artificial Intelligence

First published Thu Jul 12, 2018

Logic-Based Artificial Intelligence





First published Wed Aug 27, 2003; substantive revision Tue Feb 27, 2024

Defeasible Reasoning

First published Fri Jan 21, 2005; substantive revision Fri Jun 13, 2025

The Frame Problem

First published Mon Feb 23, 2004; substantive revision Mon Feb 8, 2016

-  **LLM**
7:58
3Blue1Brown
 -  Inside an LLM
27:14
3Blue1Brown
 -  K Q V
26:10
3Blue1Brown
 -  22:43
3Blue1Brown
- Large Language Models explained briefly**
- Transformers, the tech behind LLMs | Deep Learnin...**
- Attention in transformers, step-by-step | Deep Learnin...**
- How might LLMs store facts | Deep Learning Chapter 7**

Rekap | Neuronale Netze

Jedes neuronale Netz hat eine Reihe von *Parametern*.

Welche Berechnung das Netz genau durchführt, kann durch diese Parameter beeinflusst werden.

Wir können uns die Parameter als kleine Stellschrauben oder Drehregler vorstellen, die sich verschieden einstellen lassen.

Der *Lernprozess* eines neuronalen Netzes besteht darin, anhand von *Trainingsdaten* zu versuchen, eine geeignete Einstellung der *Parameter* zu finden.

Erfolgreiches Lernen/„Training“ =

Das Finden einer Parametereinstellung, die beliebige Inputs mit adäquaten Outputs verknüpft.

Input
Folge von
Zahlen (*Vektor*)

PXL 1: 0,35

PXL 2: 0,27

PXL 3: 0,41

· ·

· ·

· ·

PXL 16382: 0,02

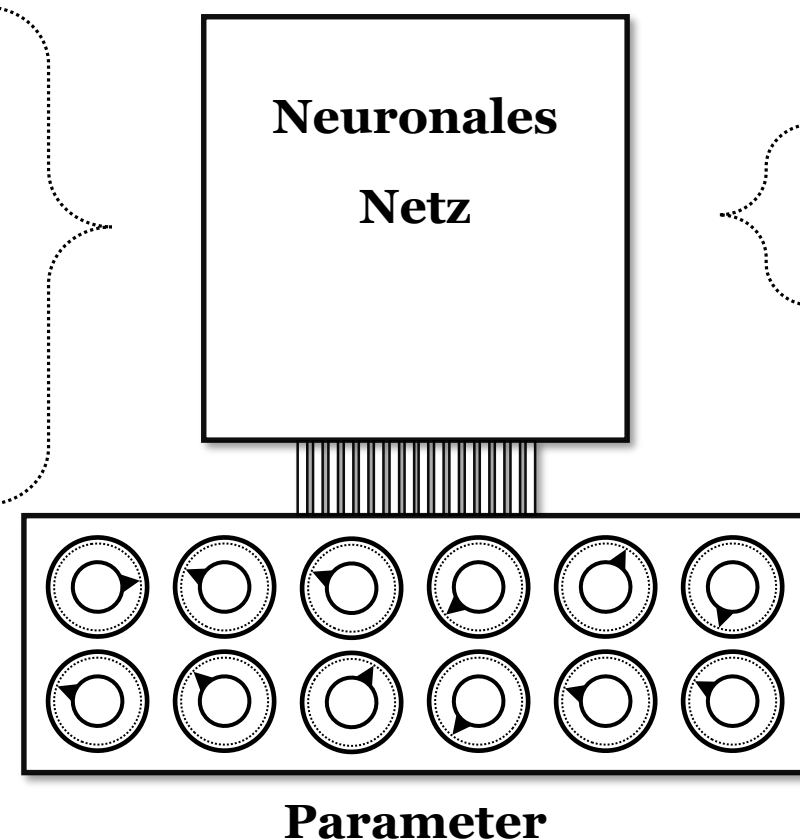
PXL 16383: 0,01

PXL 16384: 0,11

Output
Folge von
Zahlen (*Vektor*)

0,8 KATZE

0,2 HUND



Rekap | Einzelnes Neuron

Zusammenfassung:

- Lernen/Training besteht in der Suche nach Parameter-Einstellungen, die den Trainingsdaten gerecht werden.
- Künstliche Neuronen haben zwei Arten von Parametern:
 - *Gewichte* ($w_1 \dots w_n$), mit denen einzelne Inputs ($x_1 \dots x_n$) multipliziert werden (je ein Gewicht pro eingehender Verbindung)
 - einen Bias-Term b (einer pro Neuron)
- Ein einzelnes Neuron bildet die Summe der gewichteten Inputs und dem Bias-Term:
 - $w_1x_1 + \dots + w_nx_n + b$
- Anschließend wendet das Neuron noch eine Aktivierungsfunktion φ auf diese Summe an:

$$\mathbf{y} = \varphi(\mathbf{w}_1\mathbf{x}_1 + \dots + \mathbf{w}_n\mathbf{x}_n + \mathbf{b})$$

Die Aktivierungsfunktion haben wir bislang ignoriert (und werden das vorerst auch weiter tun).

Ihrer zentrale Rolle besteht darin, dem Neuron zu ermöglichen, auch nicht-linear auf Inputs zu reagieren. Das ist wichtig, weil mehrschichtige Netze dadurch eine größere Ausdrucksstärke bekommen und komplexere Muster lernen können.

Wichtig für uns: Die Aktivierungsfunktion gehört (typischerweise) nicht zu den veränderbaren Parametern.

Rekap

- Künstliche Neuronen können wir als mathematische Funktionen verstehen: Sie bilden zunächst eine gewichtete Summe der Inputs, addieren einen Bias-Term (und wenden darauf eine Aktivierungsfunktion an).
- Die **Gewichte**, mit denen die Inputs multipliziert werden, sowie der **Bias**, der hinzuaddiert wird, sind die **Parameter** des Neurons.
- Die Parameter sind die „Stellschrauben“, an denen während des Trainings eines Netzes „gedreht“ werden kann.
- Erfolgreiches Lernen bzw. Training besteht darin, eine Parametereinstellung zu finden, durch die Inputs aus dem relevanten Problembereich mit passenden Outputs verknüpft werden.
- Im Idealfall gelingt dies nicht nur für die Trainingsbeispiele, sondern auch für neue, bisher nicht gesehene Beispiele (**Generalisierung**).

Rekap

- Künstliche neuronale Netze sind aus künstlichen Neuronen zusammengesetzt, die typischerweise in Schichten — Layers — organisiert sind.
- Mathematisch entspricht ein neuronales Netz ebenfalls einer Funktion: Es nimmt eine Eingabe entgegen und erzeugt daraus eine Ausgabe.
- Diese Gesamtfunktion ergibt sich aus der Verknüpfung der Rechenoperationen der einzelnen Neuronen und Schichten.
- Für das Training von mehrschichtigen Netzen war die Entdeckung des Backpropagation-Algorithmus zentral. Backpropagation erlaubt es, effizient zu berechnen, wie stark einzelne Parameter zum Fehler eines Netzes beitragen. Auf dieser Grundlage können Parameter dann durch Verfahren wie *Gradient Descent* angepasst werden.

Rekap | Lern-Paradigmen

- In **supervised learning** the agent observes input-output pairs and learns a function that maps from input to output. For example, the inputs could be camera images, each one accompanied by an output saying “bus” or “pedestrian,” etc. An output like this is called a **label**. The agent learns a function that, when given a new image, predicts the appropriate label. In the case of braking actions (component 1 above), an input is the current state (speed and direction of the car, road condition), and an output is the distance it took to stop. In this case a set of output values can be obtained by the agent from its own percepts (after the fact); the environment is the teacher, and the agent learns a function that maps states to stopping distance.
- In **unsupervised learning** the agent learns patterns in the input without any explicit feedback. The most common unsupervised learning task is **clustering**: detecting potentially useful clusters of input examples. For example, when shown millions of images taken from the Internet, a computer vision system can identify a large cluster of similar images which an English speaker would call “cats.”
- In **reinforcement learning** the agent learns from a series of reinforcements: rewards and punishments. For example, at the end of a chess game the agent is told that it has won (a reward) or lost (a punishment). It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it, and to alter its actions to aim towards more rewards in the future.

Überblick & Vorgehen

Heute *Large Language Models (LLMs)*

LLMs? Es gibt keine allgemein anerkannte präzise Definition von „Large Language Model“

Für den Einstieg können wir uns an der folgenden Charakterisierung orientieren:

LLM \approx neuronales Netz, welches natürlichsprachlichen Text als Input nimmt und natürlichsprachlichen Text als Output erzeugt; welches typischerweise viele Parameter hat (oft hunderte von Milliarden); und welches auf einer großen Menge sprachlicher Daten trainiert wurde (oft Billionen von Wörtern), wobei das primäre Trainingsziel typischerweise darin besteht, das jeweils nächste Wort „vorherzusagen“.

Diese Charakterisierung wird später noch ein wenig ergänzt und präzisiert werden.

Ziel des Ganzen? Ein System, welches im Medium der natürlichen Sprache „sinnvoll“ auf natürlichsprachlichen Text reagieren kann (was dann eine ganze Reihe weiterer „Fähigkeiten“ nach sich zu ziehen scheint – dazu später mehr ...) und zwar idealerweise ohne enge thematische Begrenzungen.

Überblick & Vorgehen

Ziel Ein System, welches im Medium der natürlichen Sprache „sinnvoll“ auf natürlichsprachlichen Text reagieren kann (was dann eine ganze Reihe weiterer „Fähigkeiten“ nach sich zu ziehen scheint – dazu später mehr ...) und zwar idealerweise ohne enge thematische Begrenzungen.

Dieses Ziel ist durch Transformer-basierte LLMs erreicht worden (auch wenn man sich hinsichtlich weiterer Fähigkeiten der fraglichen Systeme trefflich streiten kann ...)

Nachher werden wir einen Blick auf die **Transformer Architektur** werfen: Wie sind die fraglichen LLMs aufgebaut und wie werden Sie trainiert?

Unser Schwerpunkt wird hier mehr auf **Überblick/Breite**, weniger auf technischen Details liegen.

Bevor wir uns den Transformern zuwenden ist es hilfreich, ein paar **generelle Überlegungen zum Ziel** anzustellen.

Dies wird dabei helfen, LLMs und die recht spezielle Art, wie sie das Ziel erreicht haben, im größeren Kontext von KI-Forschung/Linguistik/Wissenschaft/... zu verorten und auch historisch etwas einzuordnen.

Überblick & Vorgehen

Ziel Ein System, welches im Medium der natürlichen Sprache „sinnvoll“ auf natürlichsprachlichen Text reagieren kann (was dann eine ganze Reihe weiterer „Fähigkeiten“ nach sich zu ziehen scheint – dazu später mehr ...) und zwar idealerweise ohne enge thematische Begrenzungen.

Zunächst eine kleine Beobachtung →

①

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

②

Vorüberlegungen: Wie könnte man einen Computer dazu bringen, sich mit einem Menschen zu unterhalten?



Dann ein wenig zum Schwierigkeitsgrad, der mit dem Ziel verbunden ist.

Das führt uns zu einer →

③

Unterscheidung: Symbolische und subsymbolische KI

④

Interpretierbarkeit, Erklärungen, Verstehen, Wissenschaft vs. *Engineering*



Auf die Relevanz dieser Unterscheidung werden wir später noch zurückkommen

Überraschung ...

1

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

Wir gewöhnen uns oft erstaunlich schnell an technologische Entwicklungen (*the new normal*).

Aber es ist wert, betont zu werden: Bis ca. 2020 waren die *einzigsten* solchen „Systeme“ *Menschen* – kein künstliches System kam auch nur annähernd dem Ziel nahe.

Überraschung ...

1

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

OVERSIGHT OF A.I.:
PRINCIPLES FOR REGULATION

HEARING

BEFORE THE

SUBCOMMITTEE ON PRIVACY,
TECHNOLOGY, AND THE LAW

of the

COMMITTEE ON THE JUDICIARY
UNITED STATES SENATE

ONE HUNDRED EIGHTEENTH CONGRESS

FIRST SESSION

JULY 25, 2023

Recently, I, and many others, have been **surprised by the giant leap realized by systems like ChatGPT** to the point where it becomes difficult to discern whether one is interacting with another human or a machine. These advancements have led many top AI researchers, including myself, to revise our estimates of when human-level intelligence could be achieved. Previously thought to be decades or even centuries away, we now believe it could be within a few years or decades.

Yoshua Bengio, Founder and Scientific Director of *Mila—Quebec AI Institute*, and Professor at *Department of Computer Science and Operations Research*, Université de Montréal

Überraschung ...

1

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

<https://www.quantamagazine.org/when-chatgpt-broke-an-entire-field-an-oral-history-20250430/>

NATURAL LANGUAGE PROCESSING

When ChatGPT Broke an Entire Field: An Oral History

7 |

Researchers in “natural language processing” tried to tame human language. Then came the transformer.

Überraschung ...

1

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

CHRISTOPHER CALLISON-BURCH (*professor of computer and information science, University of Pennsylvania*): I got early access to the GPT-3 beta and was actually playing with it myself. I'm trying out all the things that my recent Ph.D. students had done as their dissertations, and just realizing — oh my God, the thing that had taken a student five years? Seems like I could reproduce that in a month. All these classical NLP tasks, many of which I had touched on or actively researched throughout my career, just felt like they worked in one shot. Like, done. And that was just really, really shocking. I sometimes describe it as having this career-existential crisis.

Überraschung ...

1

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

CHRISTOPHER CALLISON-BURCH: I didn't predict it. I don't think anyone did.

Überraschung ...

1

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

CHRISTOPHER POTTS: Somebody in our group got early access to the GPT-3 API. And I remember standing in my office, right where I'm standing now, thinking: I'm going to prompt it with some logic questions and it's going to fail at them. I'm going to reveal that it has just memorized all the things that you're so impressed by. I'm going to show you that this is a party trick.

I remember trying, and trying again. Then I had to fess up to the group: "Yeah, this is definitely much more than a party trick."

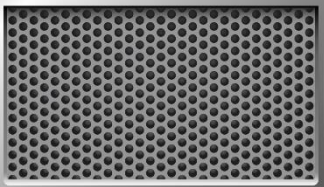
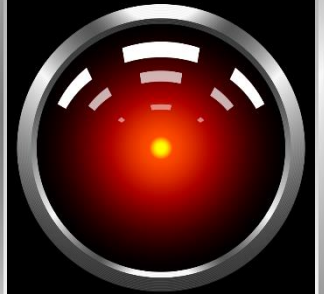
Vorüberlegung | *Sprechende Computer?*

2

Vorüberlegungen: Wie könnte man einen Computer dazu bringen, sich mit einem Menschen zu unterhalten?

I am sorry Dave, I'm
afraid I can't do that.

HAL 9000



Vorüberlegung | BASIC-Chatbot

2

Vorüberlegungen: Wie könnte man einen Computer dazu bringen, sich mit einem Menschen zu unterhalten?

Als Kind sahen meine
Versuche ungefähr so aus:

```
10 PRINT "WIE GEHT'S?"
20 INPUT A$
30 IF A$ = "GUT" THEN PRINT "DAS FREUT MICH!"
40 IF A$ = "SCHLECHT" THEN PRINT "DAS TUT MIR LEID."
50 END_
```

Dieser „Chatbot“ ist offensichtlich *extrem* limitiert:

stellt genau eine Frage und hat je eine Reaktion für die Antworten „GUT“ bzw. „SCHLECHT“, *das war's*.

Zudem hat er ein offensichtliches Problem:

Wenn der User auf „WIE GEHT'S?“ irgendetwas anderes antwortet, dann endet das Programm schlicht sofort ohne überhaupt eine Reaktion zu zeigen.

Vorüberlegung | BASIC-Chatbot

2

Vorüberlegungen: Wie könnte man einen Computer dazu bringen, sich mit einem Menschen zu unterhalten?

Mit ein bisschen Zeit und Geduld kann man diesen Chatbot natürlich flexibler machen ...

... und tatsächlich wurden historisch Systeme entwickelt, die (so *ungefähr*) diesem Muster folgten ...

```
10 PRINT "HALLO. WIE HEISST DU?"
20 INPUT N$
30 PRINT "HALLO "; N$; ". WIE GEHTS?"
40 INPUT A$
50 IF A$ = "GUT" THEN GOTO 200
60 IF A$ = "SCHLECHT" THEN GOTO 300
70 IF INSTR(A$, "HUNGER") > 0 THEN GOTO 400
100 GOTO 500
200 PRINT "DAS FREUT MICH, "; N$; "!"
210 GOTO 600
300 PRINT "DAS TUT MIR LEID, "; N$; "."
310 GOTO 600
400 PRINT "DU BIST ALSO HUNGRIG."
410 GOTO 600
500 PRINT "DAS HABE ICH NICHT VERSTANDEN."
510 GOTO 40
600 END
```

Vorüberlegung | ELIZA

ELIZA ist ein zwischen 1964 und 1966 von [Joseph Weizenbaum](#) entwickeltes [Computerprogramm](#), das die Möglichkeiten der Kommunikation zwischen einem Menschen und einem Computer über [natürliche Sprache](#) aufzeigen sollte.

```
ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:   █
```

Vorüberlegung | ELIZA

In einer einfachen Version verwendet ELIZA einen [Thesaurus](#), also ein strukturiertes Wörterbuch. Der eingegebene Satz wird durchsucht, ob er ein Wort enthält, das im Thesaurus vorhanden ist, und dann von diesem Wort ausgehend nach [Synonymen](#), meist aber nach [Oberbegriffen](#) gesucht. Das Programm enthält weiterhin eine Sammlung von Phrasen zu verschiedenen Themengebieten, häufig Fragen oder Aufforderungen, weiterzureden. Exemplarisch kann das folgende Frage-Antwort-Paar herangezogen werden:

Benutzer: „Ich habe ein Problem mit meinem Vater.“

ELIZA: „Erzählen Sie mir mehr über Ihre Familie!“

In diesem Fall hat das Programm die im Thesaurus festgelegte Zuordnung *Vater* zu dessen Oberbegriff *Familie* umgesetzt und ein Phrase zum Themenkomplex *Familie* herausgesucht. Diese einfache Funktionsweise zeigt die Beschränkung des Programms, das den Benutzer keineswegs versteht. So führt das Schlüsselwortprinzip auch zu folgendem scheinbaren „Dialog“:

Benutzer: „Krieg ist der Vater aller Dinge.“

ELIZA: „Erzählen Sie mir mehr über Ihre Familie!“

Wenn das Programm kein Wort findet, mit dem es etwas anfangen kann, antwortet es mit allgemeinen Ausweichphrasen wie z. B.

„Das habe ich noch nicht verstanden, können Sie mir das erklären.“ oder

„Davon verstehe ich nichts, lassen Sie uns über ein anderes Thema reden.“

Vorüberlegung | ELIZA

Der **ELIZA-Effekt** bezeichnet die Tendenz, einem Computersystem mehr Verständnis, Intelligenz oder Intentionalität zuzuschreiben, als unter Umständen tatsächlich vorhanden ist – besonders dann, wenn es sprachlich geschickt, empathisch oder menschenähnlich reagiert.

Der meiste Schaden, den der Computer potenziell zur Folge haben könnte, hängt weniger davon ab, was der Computer tatsächlich kann oder nicht kann, als vielmehr **von den Eigenschaften, die das Publikum dem Computer zuschreibt.**

Joseph Weizenbaum (1972): *Alptraum Computer*

Diesen Effekt (und die potenziellen Probleme, die mit einer übermäßigen Anthropomorphisierung einhergehen könnten), hat Weizenbaum bereits an dem relativ bescheidenen System ELIZA beobachtet.

In Bezug auf die heutigen LLMs ist dieser Effekt natürlich noch viel ausgeprägter ...

Überblick & Vorgehen

Ziel Ein System, welches im Medium der natürlichen Sprache „sinnvoll“ auf natürlichsprachlichen Text reagieren kann (was dann eine ganze Reihe weiterer „Fähigkeiten“ nach sich zu ziehen scheint – dazu später mehr ...) und zwar idealerweise ohne enge thematische Begrenzungen.

Zunächst eine kleine Beobachtung →

①

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

②

Vorüberlegungen: Wie könnte man einen Computer dazu bringen, sich mit einem Menschen zu unterhalten?



Dann ein wenig zum Schwierigkeitsgrad, der mit dem Ziel verbunden ist.

Das führt uns zu einer →

③

Unterscheidung: Symbolische und subsymbolische KI

④

Interpretierbarkeit, Erklärungen, Verstehen, Wissenschaft vs. *Engineering*



Auf die Relevanz dieser Unterscheidung werden wir später noch zurückkommen

Symbolische vs. subsymbolische KI

3

Unterscheidung: Symbolische und subsymbolische KI

Symbolische KI

Arbeitet mit ausdrücklich formulierten Symbolen, Begriffen und Regeln. Die Funktionsweise symbolischer Systeme und das in ihnen enthaltene „Wissen“ ist oft relativ gut nachvollziehbar, weil es in expliziten Regeln steckt.

Alle bislang skizzierten Sprach-Systeme fallen in diese erste Rubrik.

Subsymbolische KI

Arbeitet mit Zahlenwerten, Mustern und statistischen Zusammenhängen. Beispiel: Ein neuronales Netz lernt aus vielen Beispielen, welche Muster zu welchem Ergebnis passen. Genaue Funktionsweise und „Wissen“ stecken verteilt in vielen Parametern und ist oft schwer direkt nachvollziehbar.

Kurz gesagt: Symbolische KI funktioniert nach für Menschen nachvollziehbaren Regeln. Subsymbolische KI erkennt gelernte Muster in einer für Menschen nicht automatisch nachvollziehbaren Weise.

Large Language Models fallen in diese zweite Rubrik.

Symbolische vs. subsymbolische KI

3

Unterscheidung: Symbolische und subsymbolische KI

In der *symbolischen KI* versucht man, intelligentes Verhalten durch explizite Symbole, Begriffe, Regeln und Schlussverfahren zu erzeugen. Das System arbeitet also etwa mit Repräsentationen wie *Vogel(x)*, *fliegt(x)* und mit Regeln der Form: „Wenn x ein Vogel ist, dann kann x normalerweise fliegen.“ Wissen und Problemlösen sollen hier in einer für Menschen relativ gut nachvollziehbaren Form formuliert werden.

In der *subsymbolischen KI* dagegen steht nicht die explizite Formulierung von Wissen in Regeln im Zentrum, sondern das *Lernen aus Beispielen*. Neuronale Netze etwa verarbeiten Eingaben als Zahlenvektoren und passen interne Parameter – Gewichte und Biases – so an, dass sie bei bestimmten Aufgaben bessere Ergebnisse liefern. Das System kann dadurch Muster erkennen und nutzen, ohne dass diese Muster vorher ausdrücklich als Regeln einprogrammiert wurden.

Symbolische vs. subsymbolische KI

Beispiel für symbolische Herangehensweise: „Logic-based AI“

PROGRAMS WITH COMMON SENSE

by

JOHN MCCARTHY

SUMMARY

INTERESTING work is being done in programming computers to solve problems which require a high degree of intelligence in humans. However, certain elementary verbal reasoning processes so simple that they can be carried out by any non-feeble-minded human have yet to be simulated by machine programs.

This paper will discuss programs to manipulate in a suitable formal language (most likely a part of the predicate calculus) common instrumental statements. The basic program will draw immediate conclusions from a list of premises. These conclusions will be either declarative or imperative sentences. When an imperative sentence is deduced the program takes a corresponding action. These actions may include printing sentences, moving sentences on lists, and reinitiating the basic deduction process on these lists.

In 1958, John McCarthy made two important contributions to AI. In MIT AI Lab Memo No. 1, he defined the high-level language **Lisp**, which was to become the dominant AI programming language for the next 30 years. In a paper entitled *Programs with Common Sense*, he advanced a conceptual proposal for AI systems based on knowledge and reasoning. The paper describes the Advice Taker, a hypothetical program that would embody general knowledge of the world and could use it to derive plans of action. The concept was illustrated with simple logical axioms that suffice to generate a plan to drive to the airport. The program was also designed to accept new axioms in the normal course of operation, thereby allowing it to achieve competence in new areas *without being reprogrammed*. The Advice Taker thus embodied the central principles of knowledge representation and reasoning: that it is useful to have a formal, explicit representation of the world and its workings and to be able to manipulate that representation with deductive processes. The paper influenced the course of AI and remains relevant today.

Russell & Norvig (2022): *Artificial Intelligence: A Modern Approach*, 4th Edition, Global Edition

Symbolische vs. subsymbolische KI

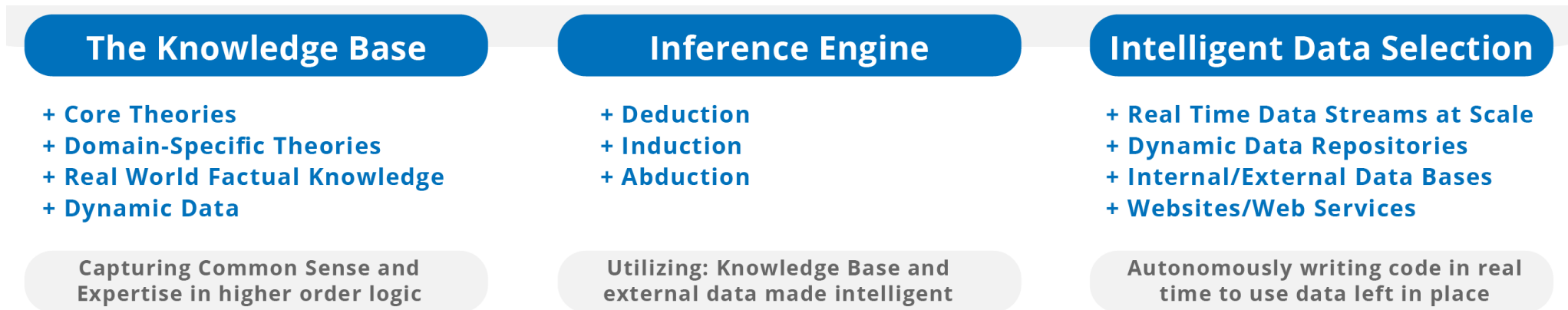
Beispiel für symbolische Herangehensweise: „Expert Systems“

The next major effort was the MYCIN system for diagnosing blood infections. With about 450 rules, MYCIN was able to perform as well as some experts, and considerably better than junior doctors. It also contained two major differences from DENDRAL. First, unlike the DENDRAL rules, no general theoretical model existed from which the MYCIN rules could be deduced. They had to be acquired from extensive interviewing of experts. Second, the rules had to reflect the uncertainty associated with medical knowledge. MYCIN incorporated a calculus of uncertainty called **certainty factors** (see Chapter 13), which seemed (at the time) to fit well with how doctors assessed the impact of evidence on the diagnosis.

The first successful commercial expert system, R1, began operation at the Digital Equipment Corporation (McDermott, 1982). The program helped configure orders for new computer systems; by 1986, it was saving the company an estimated \$40 million a year. By 1988, DEC's AI group had 40 expert systems deployed, with more on the way. DuPont had 100 in use and 500 in development. Nearly every major U.S. corporation had its own AI group and was either using or investigating expert systems.

Symbolische vs. subsymbolische KI

Beispiel für symbolische Herangehensweise: **Cyc**



The **Cyc knowledge base (KB)** is composed of some 25 million assertions. When you combine this with the generality of the knowledge and the efficient **inference engines** that can leverage this knowledge to generate new conclusions, Cyc easily has *trillions* of pieces of usable knowledge. When you add to this the knowledge that Cyc possesses by accessing external databases (akin to how you might say you know the phone numbers stored on the contact list in your phone, even if you could not recite them by memory alone), Cyc's KB size is clearly differentiated from other AI platforms.

Symbolische vs. subsymbolische KI

Beispiel für symbolische Herangehensweise: **Cyc**

Cycorp is staffed by **Ontological Engineers**, typically philosophy Ph.Ds who are trained in making careful distinctions, converting natural language to higher order logic, and translating this into CycL, the epistemological language for Cyc. This method is used for most projects, especially early on. Ontologists are also available to deal with especially tricky cases after a given product is deployed. However, Cycorp ultimately wants to hand the reins over when a product is deployed, empowering customers to use other methods for adding to the knowledge base as necessary.

Symbolische vs. subsymbolische KI

Beispiel für symbolische Herangehensweise: **HARPY**

The HARPY Speech Recognition System

Thesis Summary

Bruce T. Lowerre

April, 1976

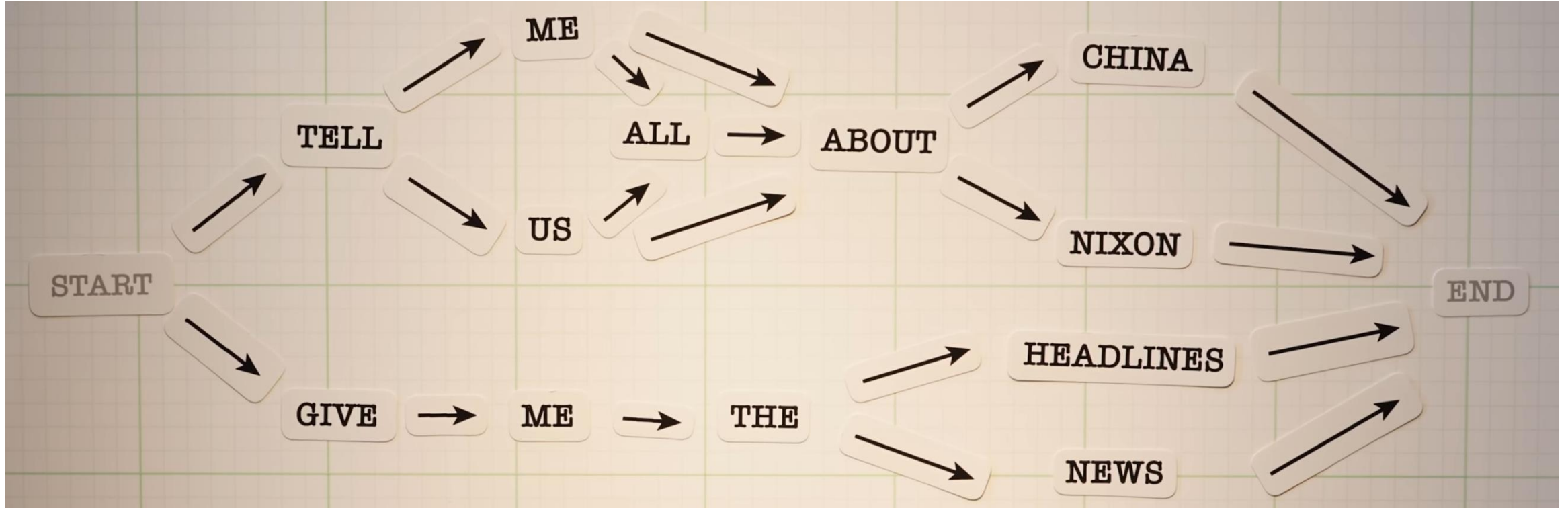
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Submitted to Carnegie-Mellon University in partial fulfillment
of the requirements for the degree of Doctor of Philosophy.

This research was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (Contract F44620-73-C-0074) and monitored by the Air Force Office of Scientific Research.

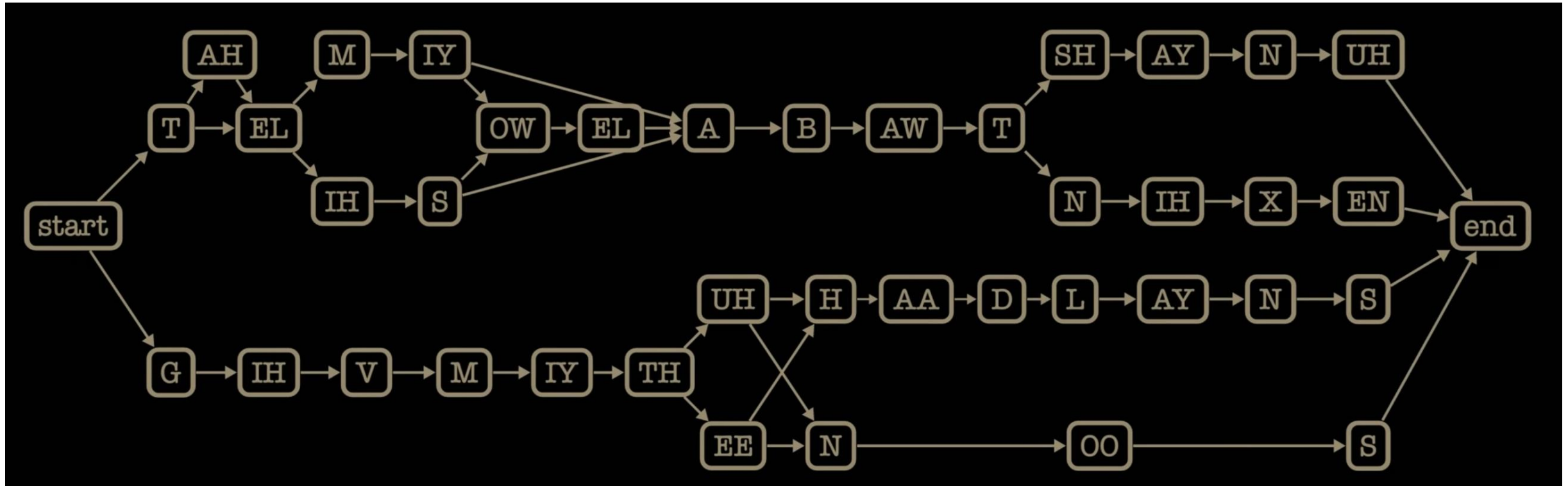
Symbolische vs. subsymbolische KI

Beispiel für symbolische Herangehensweise: **HARPY**



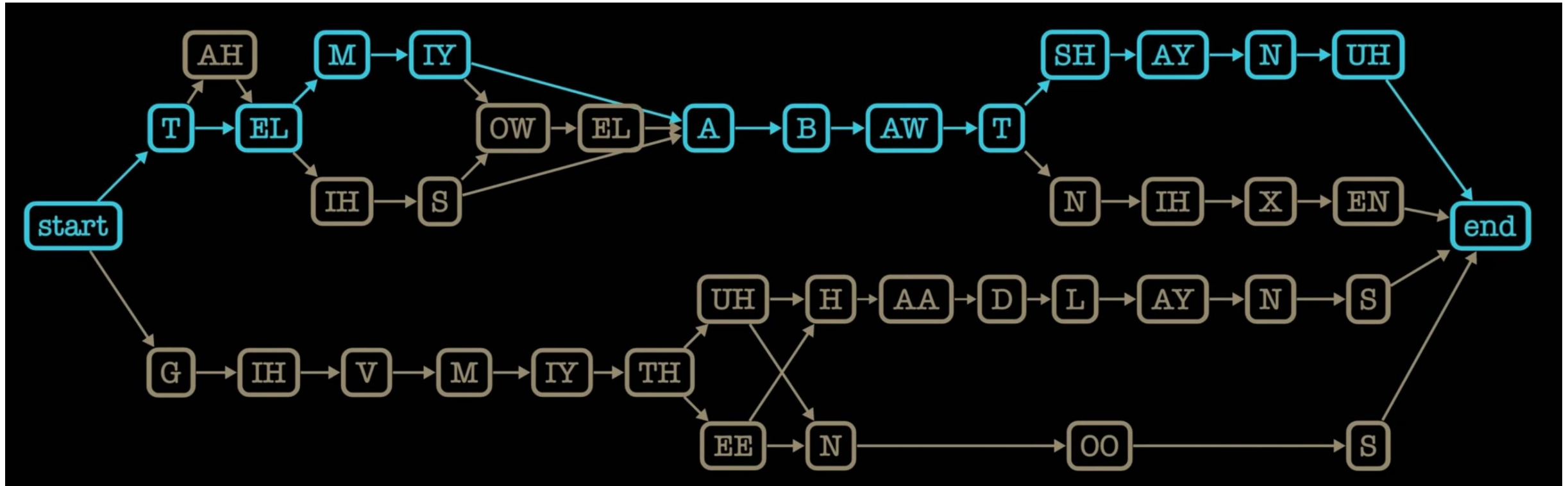
Symbolische vs. subsymbolische KI

Beispiel für symbolische Herangehensweise: **HARPY**



Symbolische vs. subsymbolische KI

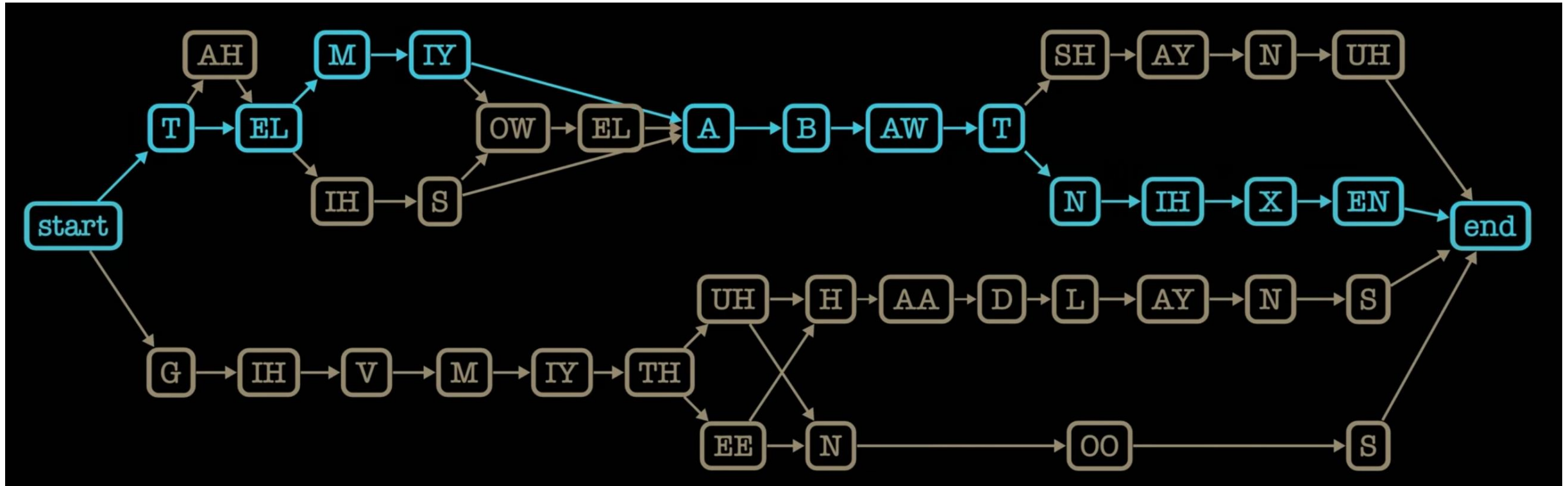
Beispiel für symbolische Herangehensweise: **HARPY**



Tell me about China.

Symbolische vs. subsymbolische KI

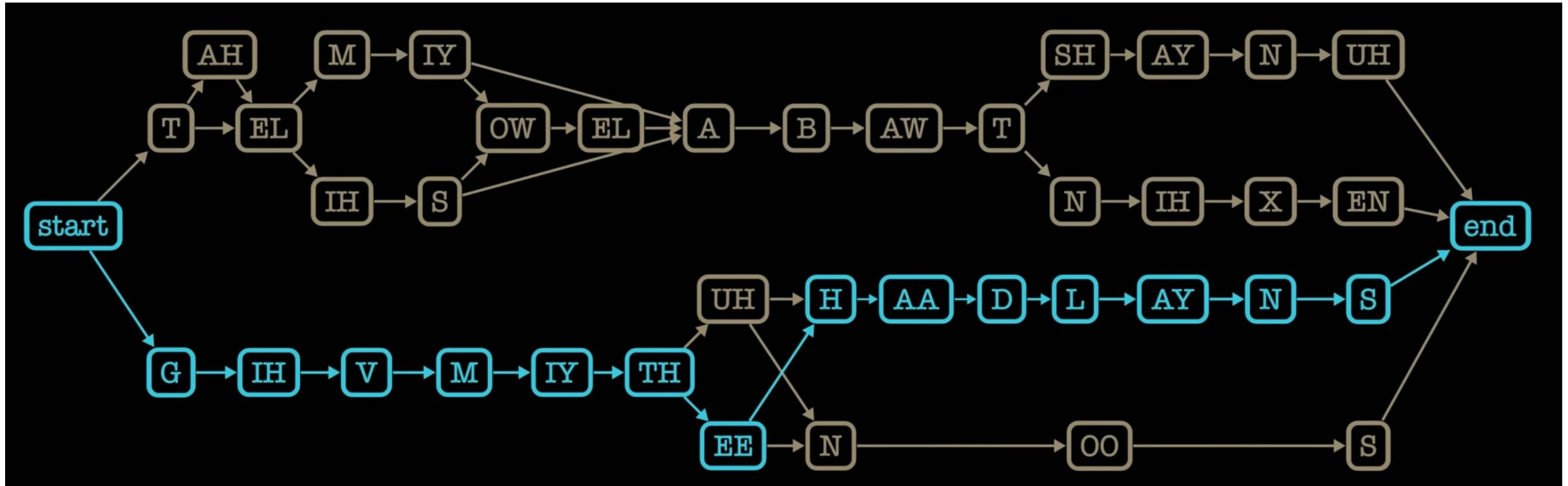
Beispiel für symbolische Herangehensweise: **HARPY**



Tell me about Nixon.

Symbolische vs. subsymbolische KI

Beispiel für symbolische Herangehensweise: **HARPY**



Give me the headlines.

Symbolische vs. subsymbolische KI

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

2

Vorüberlegungen: Wie könnte man einen Computer dazu bringen, sich mit einem Menschen zu unterhalten?

+

3

Unterscheidung: Symbolische und subsymbolische KI

Eine symbolische Herangehensweise an das Ziel ist ***extrem anspruchsvoll:***

- erfordert umfassendes Wissen um die Funktionsweise von Sprache (Syntax, Semantik, Pragmatik, ...)
- erfordert die Formalisierung von für die „sinnvolle“ Verwendung von Sprache relevanten Wissens (und das scheint prinzipiell grenzenlos zu sein ...)
- erfordert die Formalisierung von „Common Sense“ inkl. entsprechender Schlussfolgerungen (Schlussfolgern unter Unsicherheit, Umgang mit Ausnahmen, *defeasible reasoning*, ...)
- ... und vieles mehr.

Symbolische vs. subsymbolische KI

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

2

Vorüberlegungen: Wie könnte man einen Computer dazu bringen, sich mit einem Menschen zu unterhalten?

+

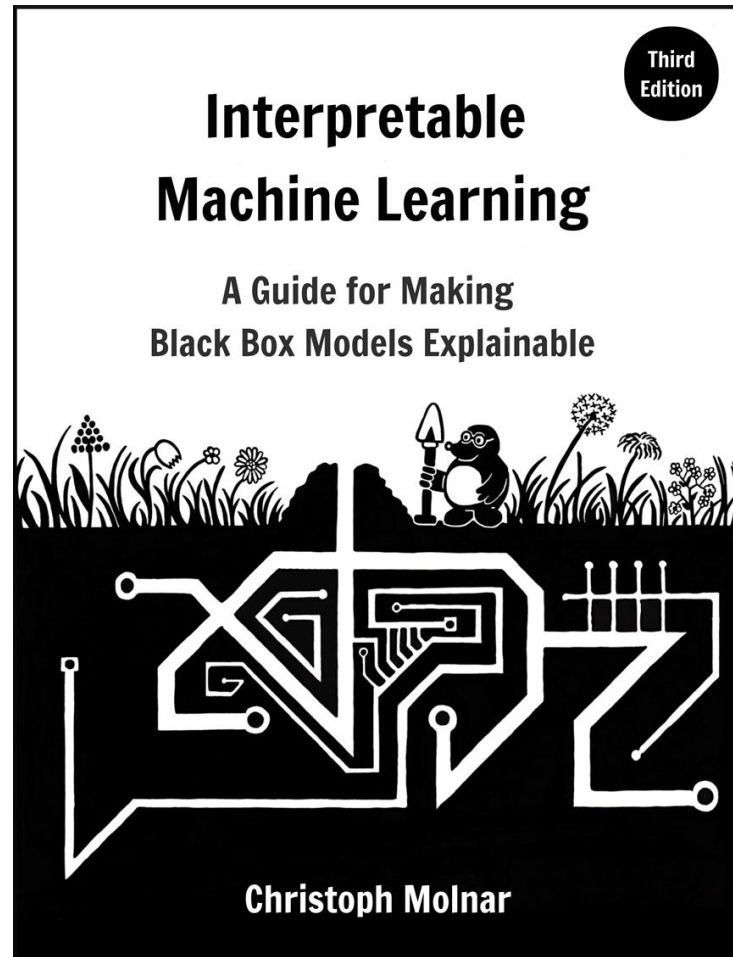
3

Unterscheidung: Symbolische und subsymbolische KI

Transformer-basierte *Large Language Models* sind subsymbolische Systeme.

- Neuronale Netze (zumindest im Kern)
- anhand von großen Datenmengen trainiert
- keinerlei „handprogrammierte“ Regeln, Wissensdatenbanken, etc.
- alles, was wir brauchen, ist eine riesige Datenmenge (und die richtige Netz-Architektur & Netzgröße)
- aber entsprechend wird es schwierig, zu verstehen, was in den fraglichen Netzen eigentlich genau passiert ...

Interpretierbarkeit



Hin zu LLMs ...

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

Um die Präsentation Transformer-basierter *LLMs* vorzubereiten, stellen wir zunächst ein paar ganz generelle Überlegungen dazu an, wie ein auf neuronalen Netzen basierender subsymbolischer Versuch, das Ziel zu erreichen, aussehen könnte ...

LLMs | Einstieg & „Pre-Training“

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

Aus letzter Sitzung

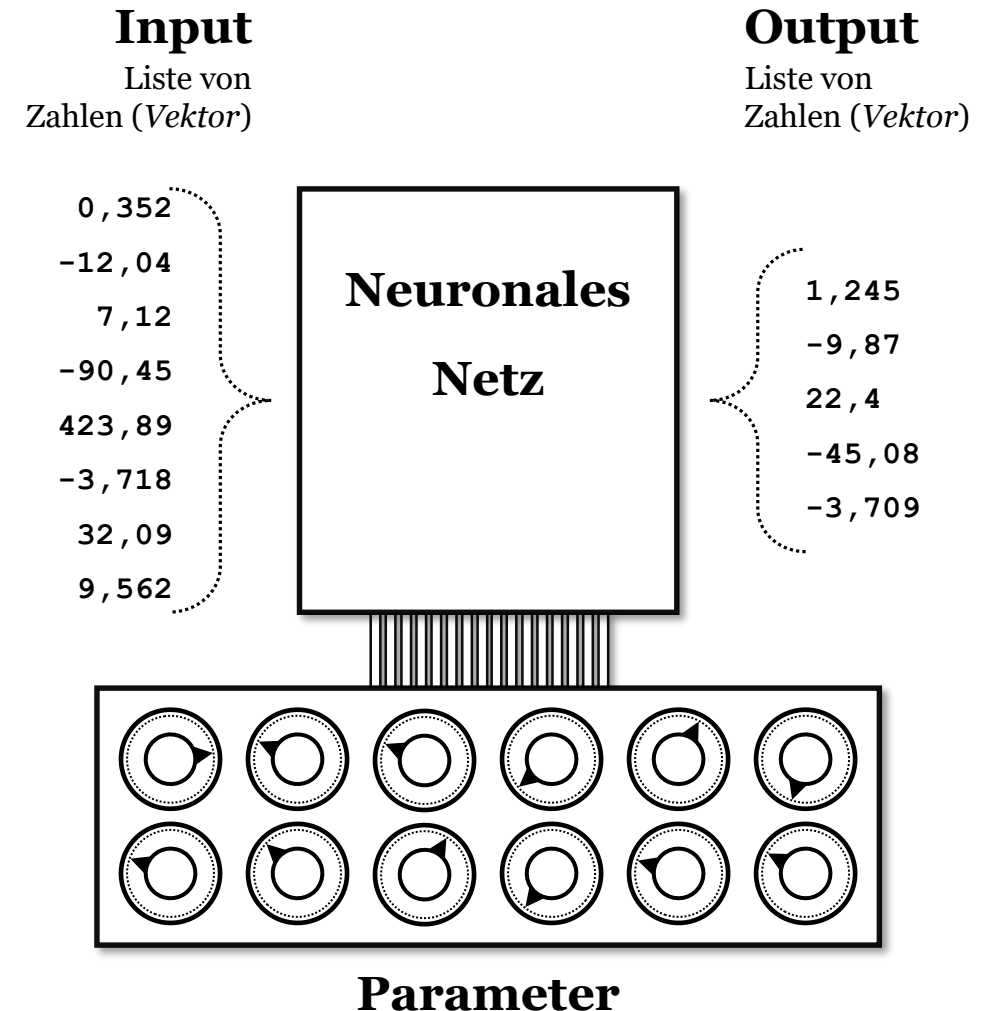
Neuronales Netz als *mathematische Funktion*:

Liste von Zahlen als Input, Berechnung, Liste von Zahlen als Output.

Welche Berechnung genau hängt von der Einstellung der *Parameter* ab.

Lernen/„Training“ =

Das Finden einer Parametereinstellung, die Inputs mit adäquaten Outputs verknüpft.



LLMs | Einstieg & „Pre-Training“

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

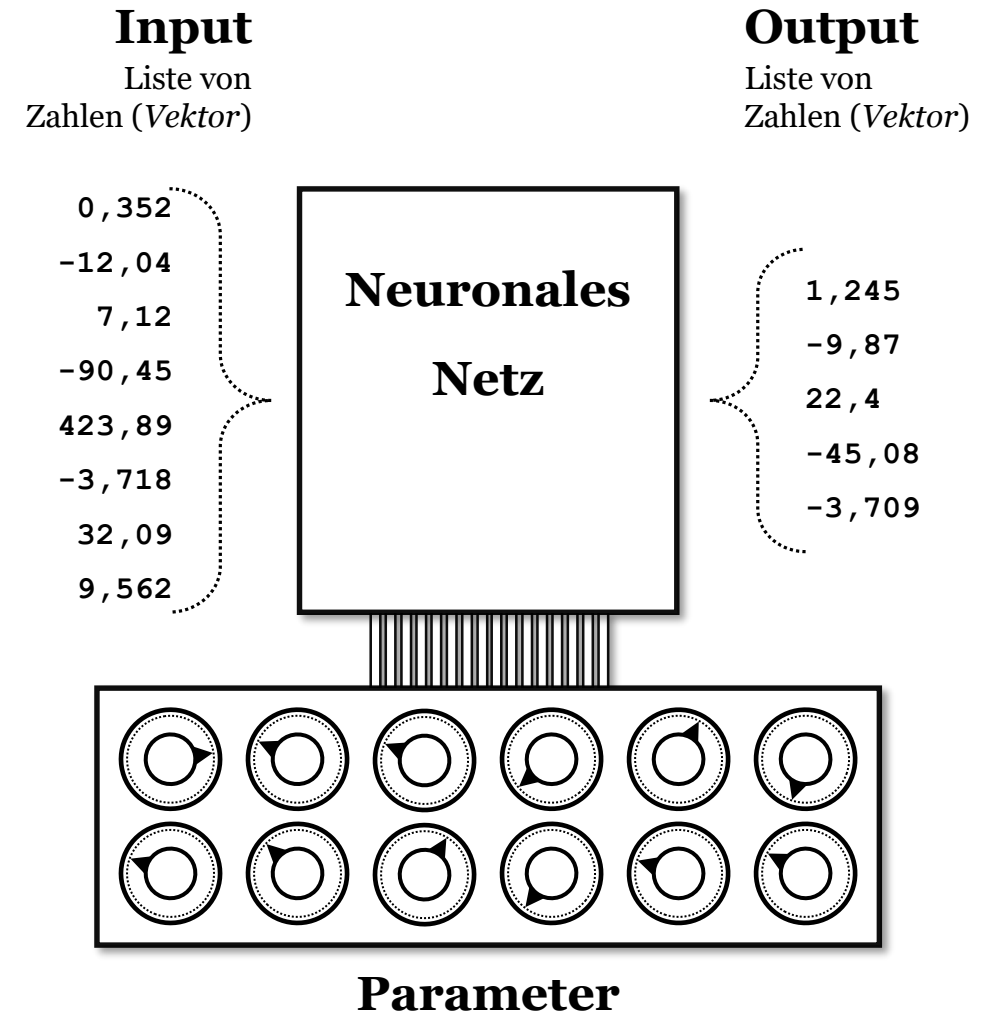
Wenn wir ein neuronales Netz konstruieren wollen, welches das Ziel erreichen kann, stellen sich sofort 2 Fragen:

(1) Was sollen wir als Inputs/Outputs nehmen?

Neuronale Netze brauchen *Zahlen*, wie wollen aber eigentlich mit *Text* operieren; wir brauchen also eine sinnvolle Weise, Text in Zahlen zu „übersetzen“, mit denen das Netz dann rechnen kann, und dann Zahlen zurück in Text „umzuwandeln“.

(2) Wie soll das Training aussehen und was sind die Daten?

Beim Beispiel aus der letzten Sitzung war das einfach: Da das Netz lernen sollte, handgeschriebene Ziffern zu „erkennen“, konnten wir als Daten mit entsprechenden Labeln versehene Bilder nehmen; trainiert wurde dann, möglichst gute Wahrscheinlichkeitsverteilungen über die relevanten Klassifikationen (EINS, ZWEI, ...) zu generieren.



LLMs | Einstieg & „Pre-Training“

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

Wenn wir ein neuronales Netz konstruieren wollen, welches das Ziel erreichen kann, stellen sich sofort 2 Fragen:

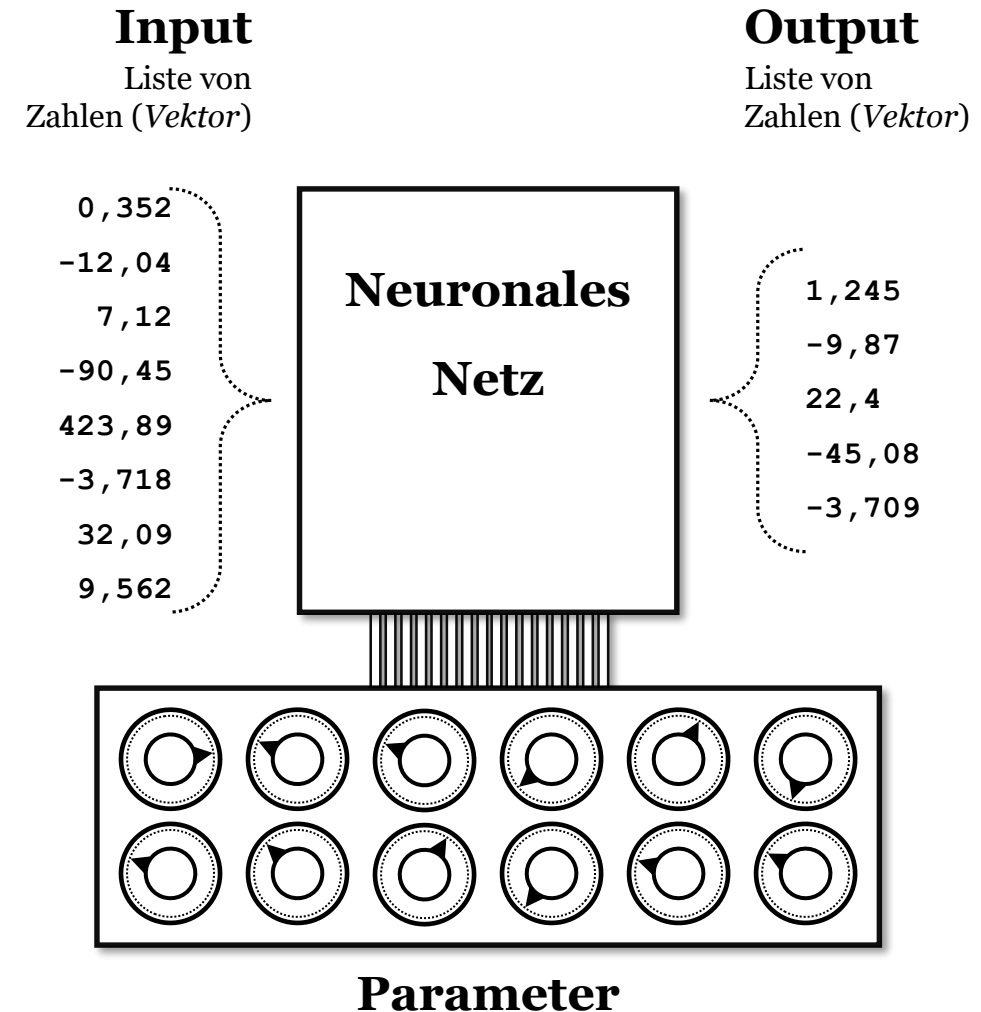
(1) Was sollen wir als Inputs/Outputs nehmen?

Neuronale Netze brauchen *Zahlen*, wie wollen aber eigentlich mit *Text* operieren; wir brauchen also eine sinnvolle Weise, Text in Zahlen zu „übersetzen“, mit denen das Netz dann rechnen kann, und dann Zahlen zurück in Text „umzuwandeln“.

(2) Wie soll das Training aussehen und was sind die Daten?

Aber mit Blick auf das neue **Ziel** ist vielleicht nicht auf den ersten Blick klar, was wir hier nehmen sollten ...

Wenden wir uns zunächst **(2)** zu; danach kehren wir dann zu **(1)** zurück.



LLMs | Einstieg & „Pre-Training“

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

(2) Wie soll das Training aussehen und was sind die Daten?

Hier ein paar Beobachtungen und Überlegungen, die zu einem Vorschlag zur Beantwortung von (2) führen:

- Menschen sind in der Lage, im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text zu reagieren.
- Also: Ein System, welches menschliche sprachliche Reaktionen „simulieren“ kann, würde dem Ziel gerecht werden.
- Ein System, welches für einen beliebigen Text T eine „menschliche“ Fortführung dieses Textes produzieren kann, wäre in der Lage, menschliche sprachliche Reaktionen zu simulieren.
- Sofern wir eine hinreichend große und hinreichend sinnvolle/kohärente Menge an menschlichem Text haben, können wir diese Textmenge verwenden, um ein solches System zu trainieren!
- Sei T eine solche Textmenge. Das Training könnte nun beispielsweise so aussehen: Für jede Wortfolge W_1, W_2, \dots, W_n in T soll das System lernen, möglichst gut das jeweils nächste Wort vorherzusagen (und es sollte lernen, wann es aufhören sollte).

Idee: Daten = menschlicher Text / Training = *Next-Word-Prediction*

LLMs | Einstieg & „Pre-Training“

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

Idee: Daten = menschlicher Text / Training = *Next-Word-Prediction*

Diese Idee passt gut zu einem linguistischen Ansatz innerhalb der Semantik: *Distributional Semantics*.

Distributional Hypothesis

The degree of semantic similarity between two linguistic expressions *A* and *B* is a function of the similarity of the linguistic contexts in which *A* and *B* can appear.

According to the Distributional Hypothesis, words with similar positions in text corpora tend to have similar meanings, and so representations of this information can serve as proxies for word meaning. This is a hypothesis about a *correlation* between positions in text corpora and word meaning that researchers can exploit.

LLMs | Einstieg & „Pre-Training“

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

(1) Was sollen wir als Inputs/Outputs nehmen?

Ausgehend von dieser Idee zu Daten & Training können wir nun ein paar Überlegungen zu (1) anstellen:

- Da das Trainingsziel *Next-Word-Prediction* ist, brauchen wir als mögliche Inputs Text, der in Wörter zerlegt ist.
 - (Wir werden gleich noch eine Unterscheidung zwischen *Wörtern* und *Tokens* machen; eigentlich geht es um *Tokens*; aber wir bleiben zunächst bei dem intuitiv leichter verständlichem Begriff eines Wortes.)
- Aber da ein neuronales Netz strenggenommen nur *Zahlen* und nicht Wörter/Text als Input nehmen kann, müssen wir einen brauchbaren Weg finden, Wörter in Zahlen umzuwandeln.
- Zudem sollte das Netz in der Lage sein, viel über einzelne Wörter (und deren Verhältnis zu anderen Wörtern) zu „lernen“ – nur so hat das Netz eine Chance, dem Trainingsziel gerecht zu werden.

LLMs | Einstieg & „Pre-Training“

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

(1) Was sollen wir als Inputs/Outputs nehmen?

Dies legt folgende Idee nahe:

- Ein einzelnes Wort sollte in *mehrere* Zahlen „übersetzt“ werden, von denen jede einzelne als ein *Parameter* des Netzes fungiert – so hat das Netz die Gelegenheit, die „Repräsentation“ einzelner Wörter im Laufe Trainings sukzessive und mit Blick auf das Trainingsziel anzupassen.
- Als Output sollten wir eine *Wahrscheinlichkeitsverteilung über das Vokabular* nehmen (analog zum Beispiel des Netzes zur Ziffernerkennung aus der letzten Sitzung).

LLMs | Einstieg & „Pre-Training“

Ziel Ein System, welches im Medium der natürlichen Sprache ohne enge thematische Begrenzungen „sinnvoll“ auf natürlichsprachlichen Text reagieren kann.

Der Plan (um ein neuronales Netz zu bekommen, welches dem Ziel gerecht wird):

- Text wird in Wörter zerlegt
- Jedem Wort im Vokabular wird eine Liste von Zahlen zugeordnet (= ein Vektor). Wir nennen eine solche Liste von Zahlen auch den **Embedding-Vektor** des fraglichen Wortes (oder auch *Word-Embedding*).
- Jede dieser Zahlen ist ein **Parameter** des Netzes (was nicht unbedingt heißt, dass das Netz nicht noch über weitere Parameter verfügt – siehe unten).
- Als Trainingsdaten verwenden wir Wortfolgen aus einem großen Textkorpus (beispielsweise vollständige Wikipedia; heutzutage quasi die Gesamtheit allen digitalen Textes)
- Als Output liefert das Netz eine Wahrscheinlichkeitsverteilung über alle Wörter im Vokabular
- Das Trainingsziel hierbei ist, möglichst gut das jeweils tatsächliche nächste Wort „vorherzusagen“

Das klingt alles sehr seltsam! Warum sollte man erwarten, dass dabei irgendetwas Brauchbares herauskommt?

LLMs | Einstieg & „Pre-Training“

Warum sollte man erwarten, dass dabei irgendetwas Brauchbares herauskommt?

Wie sich empirisch gezeigt hat, führt dieses Vorgehen tatsächlich dazu, dass ein entsprechend trainiertes Netz bei Embedding-Vektoren landet, die auf interessante Weise beispielsweise semantische Verhältnisse zwischen Wörtern reflektieren.

LLMs | Einstieg & „Pre-Training“

Warum sollte man erwarten, dass dabei irgendetwas Brauchbares herauskommt?

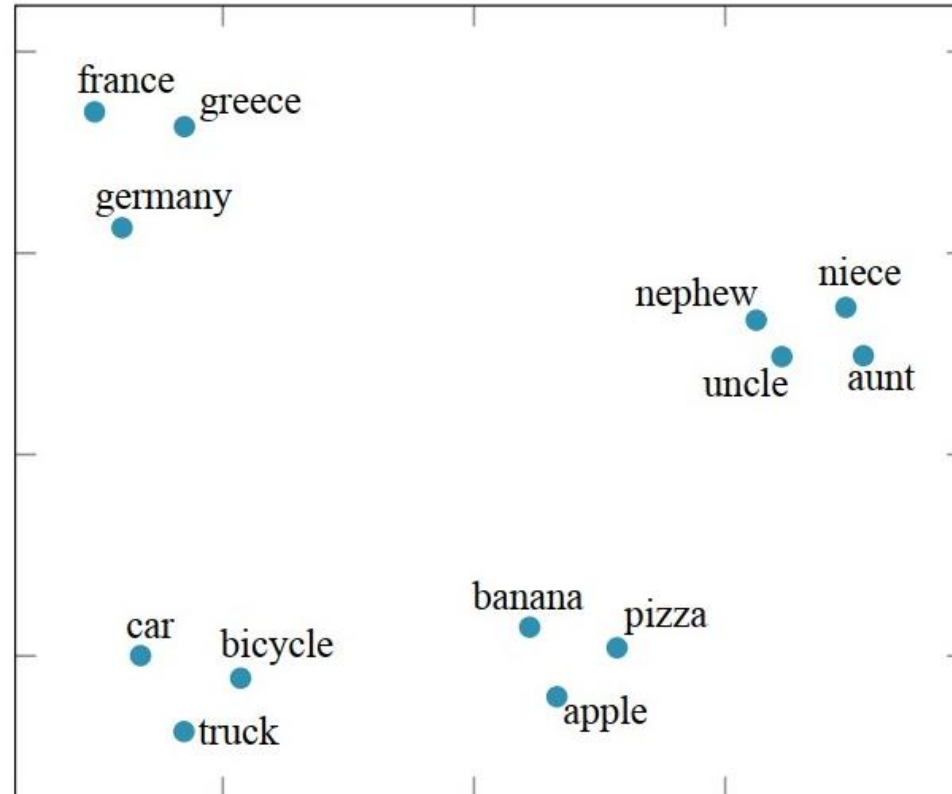


Figure 25.1 Word embedding vectors computed by the GloVe algorithm trained on 6 billion words of text. 100-dimensional word vectors are projected down onto two dimensions in this visualization. Similar words appear near each other.

LLMs | Einstieg & „Pre-Training“

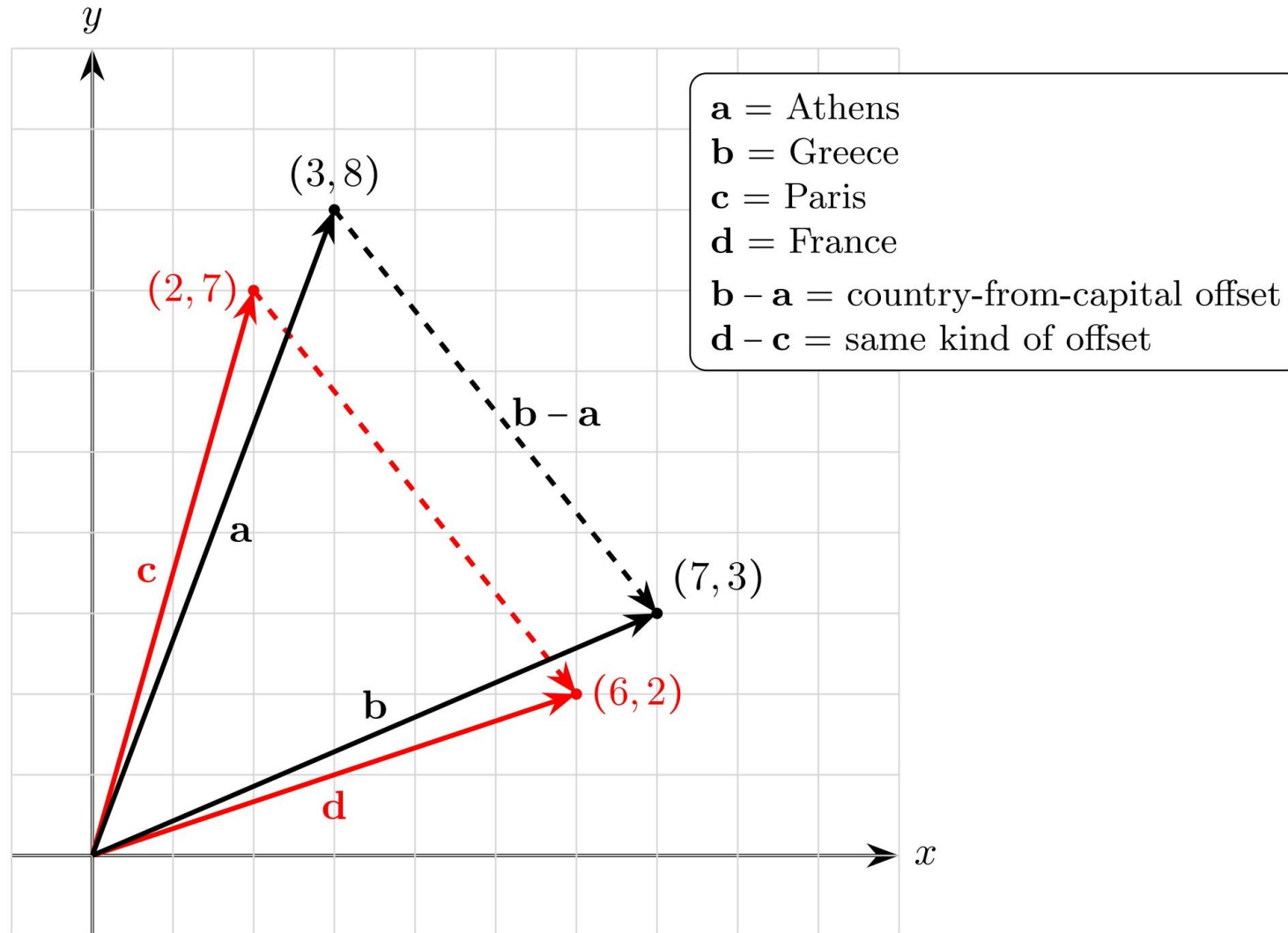
Warum sollte man erwarten, dass dabei irgendetwas Brauchbares herauskommt?

A	B	C	D = C + (B - A)	Relationship
Athens	Greece	Oslo	Norway	<i>Capital</i>
Astana	Kazakhstan	Harare	Zimbabwe	<i>Capital</i>
Angola	kwanza	Iran	rial	<i>Currency</i>
copper	Cu	gold	Au	<i>Atomic Symbol</i>
Microsoft	Windows	Google	Android	<i>Operating System</i>
New York	New York Times	Baltimore	Baltimore Sun	<i>Newspaper</i>
Berlusconi	Silvio	Obama	Barack	<i>First name</i>
Switzerland	Swiss	Cambodia	Cambodian	<i>Nationality</i>
Einstein	scientist	Picasso	painter	<i>Occupation</i>
brother	sister	grandson	granddaughter	<i>Family Relation</i>
Chicago	Illinois	Stockton	California	<i>State</i>
possibly	impossibly	ethical	unethical	<i>Negative</i>
mouse	mice	dollar	dollars	<i>Plural</i>
easy	easiest	lucky	luckiest	<i>Superlative</i>
walking	walked	swimming	swam	<i>Past tense</i>

Figure 25.2 A word embedding model can sometimes answer the question “A is to B as C is to [what]?” with vector arithmetic: given the word embedding vectors for the words **A**, **B**, and **C**, compute the vector $\mathbf{D} = \mathbf{C} + (\mathbf{B} - \mathbf{A})$ and look up the word that is closest to **D**. (The answers in column **D** were computed automatically by the model. The descriptions in the “Relationship” column were added by hand.) Adapted from Mikolov *et al.* (2013, 2014).

LLMs | Einstieg & „Pre-Training“

Warum sollte man erwarten, dass dabei irgendetwas Brauchbares herauskommt?



LLMs | Einstieg & „Pre-Training“

Warum sollte man erwarten, dass dabei irgendetwas Brauchbares herauskommt?

Wie sich empirisch gezeigt hat, führt dieses Vorgehen tatsächlich dazu, dass ein entsprechend trainiertes Netz bei Embedding-Vektoren landet, die auf interessante Weise beispielsweise semantische Verhältnisse zwischen Wörtern reflektieren.

Wohlgemerkt:

Diese „semantischen Eigenschaften und Verhältnisse“ sind dem Netz **nicht** vorgegeben worden, sind **nicht** „vorprogrammiert“.

Sie ergeben sich allein durch das Training mittels *Next-Word-Prediction*; die Komponenten der *Embedding-Vektoren* sind Parameter, die zu Beginn völlig zufällig sind und die das Netz im Zuge des Trainings anpasst.

Das spricht dafür, dass an der *Distributional Hypothesis* tatsächlich etwas dran ist – die Aufgabe *Next-Word-Prediction* scheint am besten gelöst zu werden, wenn das Netz bei einer Parametereinstellung landet, die in gewisser Weise (u.a.) den Bedeutungen der fraglichen Wörter gerecht wird.

LLMs | Einstieg & „Pre-Training“

Das Training mit dem Trainingsziel *Next-Word-Prediction* auf der Basis eines großen Textkorpus nennt man auch das ***Pre-Training*** eines LLMs.

(Zudem gibt es auch noch weitere Trainingsmethoden, beispielsweise das anschließende *Fine-Tuning*, sowie oft noch RLHF und einiges mehr ... auf diese zusätzlichen Trainingsmethoden gehen wir später oder ggf. nächste Sitzung noch ein.)

Als nächstes kommen wir nun zur Transformer-Architektur, und als konkretes Beispiel werden wir einen Blick auf GPT-3 werfen.

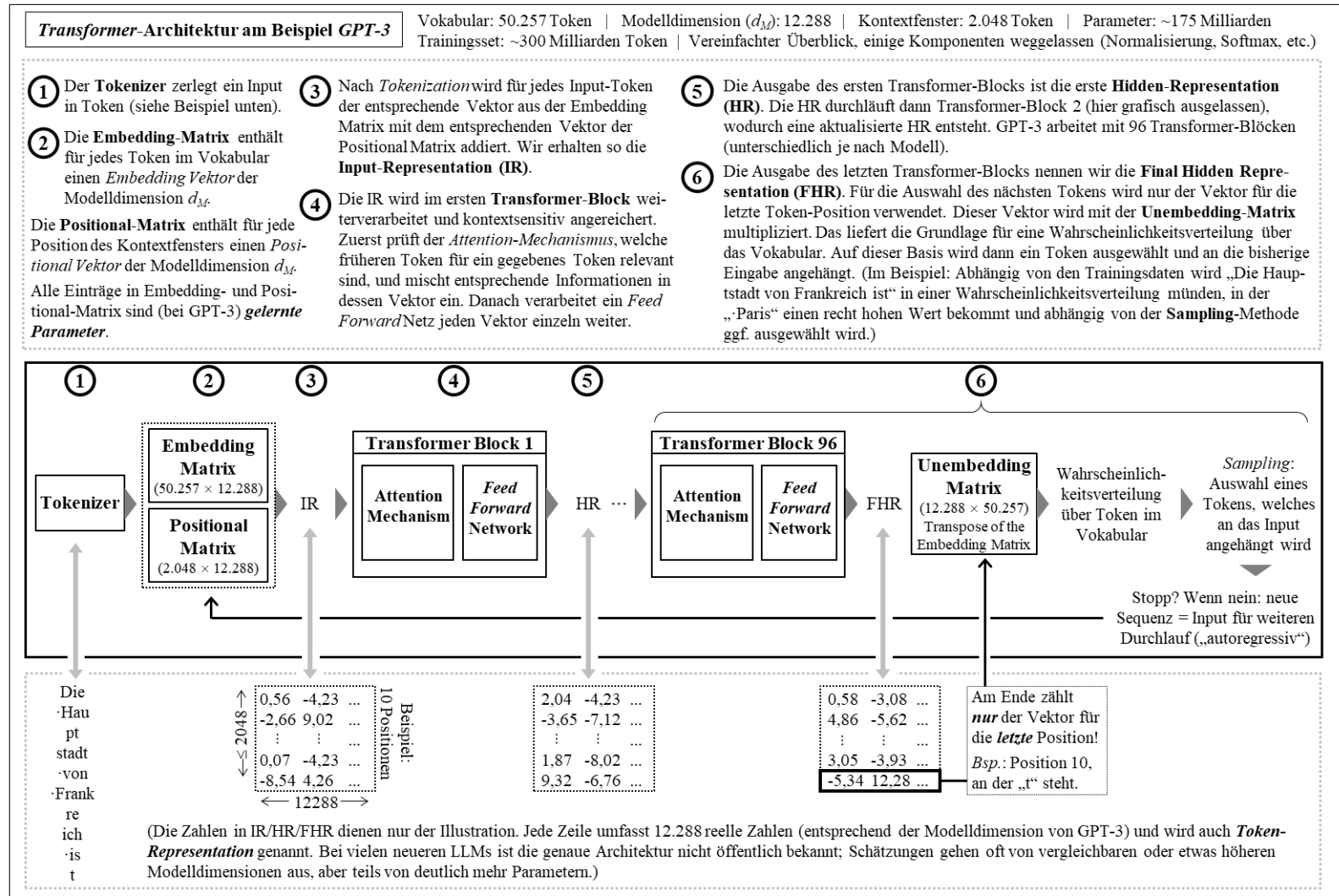
„GPT“ steht hierbei für ***Generative Pre-trained Transformer***.

Transformer LLMs | Handout

Handout:

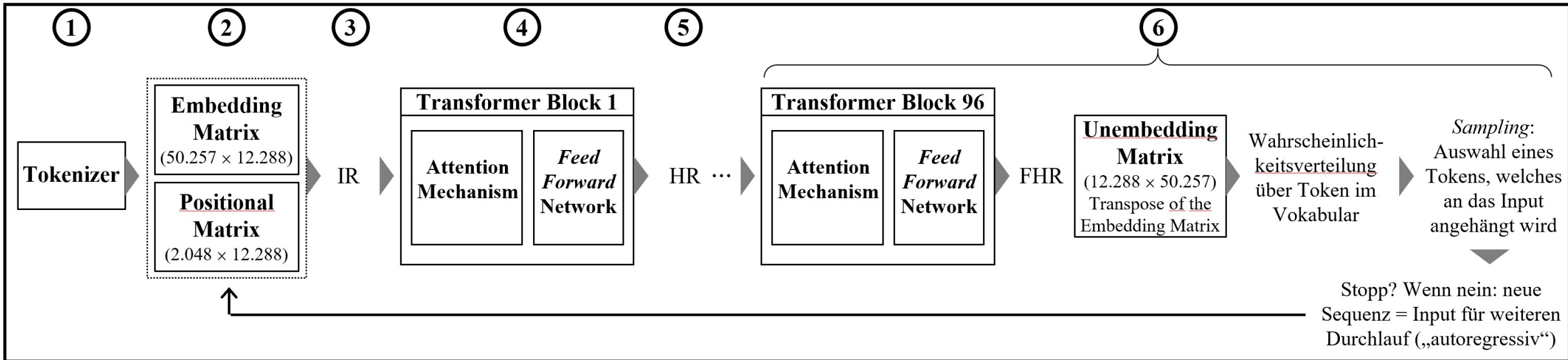
Transformer-Architektur am Beispiel GPT-3

Um die Komponenten eines Transformer-LLMs kennenzulernen, gehen wir dieses Handout schrittweise durch.



Transformer LLMs | Übersicht

Gesamtübersicht (Vereinfachter Überblick, einige Komponenten weggelassen (Normalisierung, Softmax, etc.).



Transformer LLMs | *Tokenizer*

Oben hatte ich das Trainingsziel als ***Next-Word-Prediction*** bezeichnet.

Etwas genauer müsste man eigentlich sagen ***Next-Token-Prediction*** – denn LLMs operieren typischerweise nicht auf Wort-Ebene, sondern auf Token-Ebene.

Ein Token ist hierbei die kleinste sprachliche Einheit, mit der das LLM operiert (entsprechend haben wir es dann auch nicht mit Embedding-Vektoren für „Wörter“ sondern für Token zu tun).

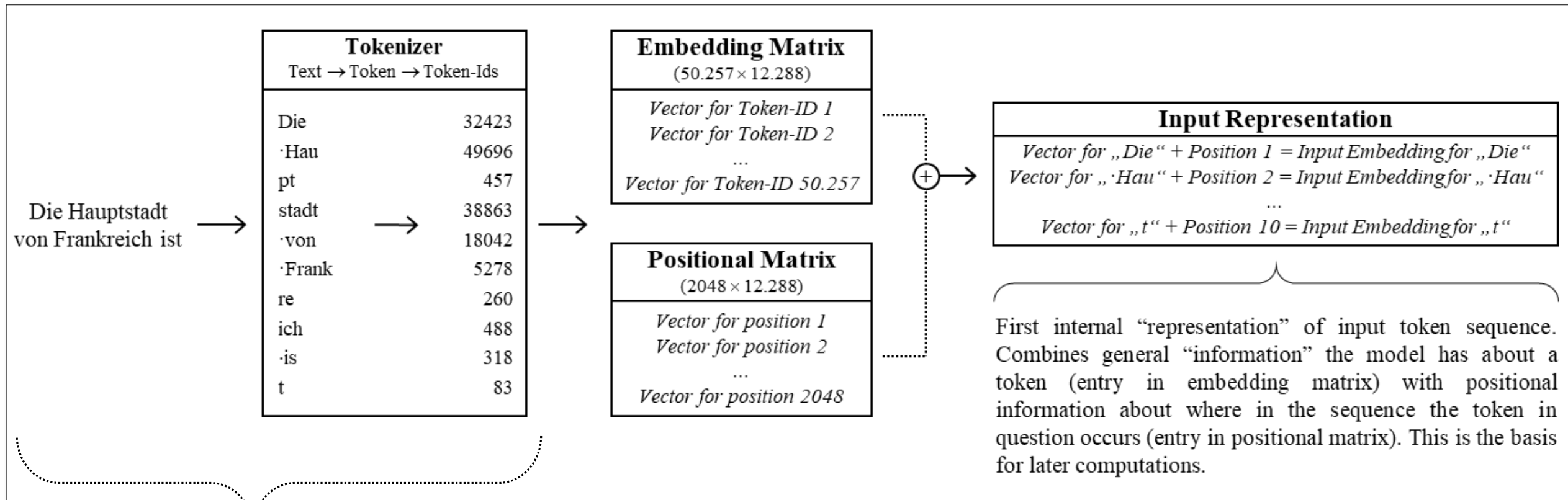
Die Zerlegung eines Inputs (beispielsweise eines Prompts) in Token wird vom Tokenizer vorgenommen, und das Vokabular eines LLMs ist die Gesamtheit der Token.

Bei GPT-3 umfasst das Vokabular 50257 Token.

Verschiedene LLM verwenden teilweise verschiedene Tokenizer (und neuere Modelle haben oft ein größeres Vokabular)

Die Tokenizer selbst sind oft mit einem Algorithmus wie *Byte-Pair-Encoding* trainiert.

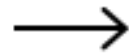
Transformer LLMs | *Input* \Rightarrow *IR*



Der **Tokenizer** zerlegt ein Input in Token.

Transformer LLMs | *Tokenizer*

Die Hauptstadt
von Frankreich ist



Tokenizer	
Text	Token → Token-Ids
Die	32423
·Hau	49696
pt	457
stadt	38863
·von	18042
·Frank	5278
re	260
ich	488
·is	318
t	83



Transformer LLMs | *Tokenizer*

Tokenizer machen manchmal überraschende Sachen ... :

Brot für die Welt.

Brot für die Welt.

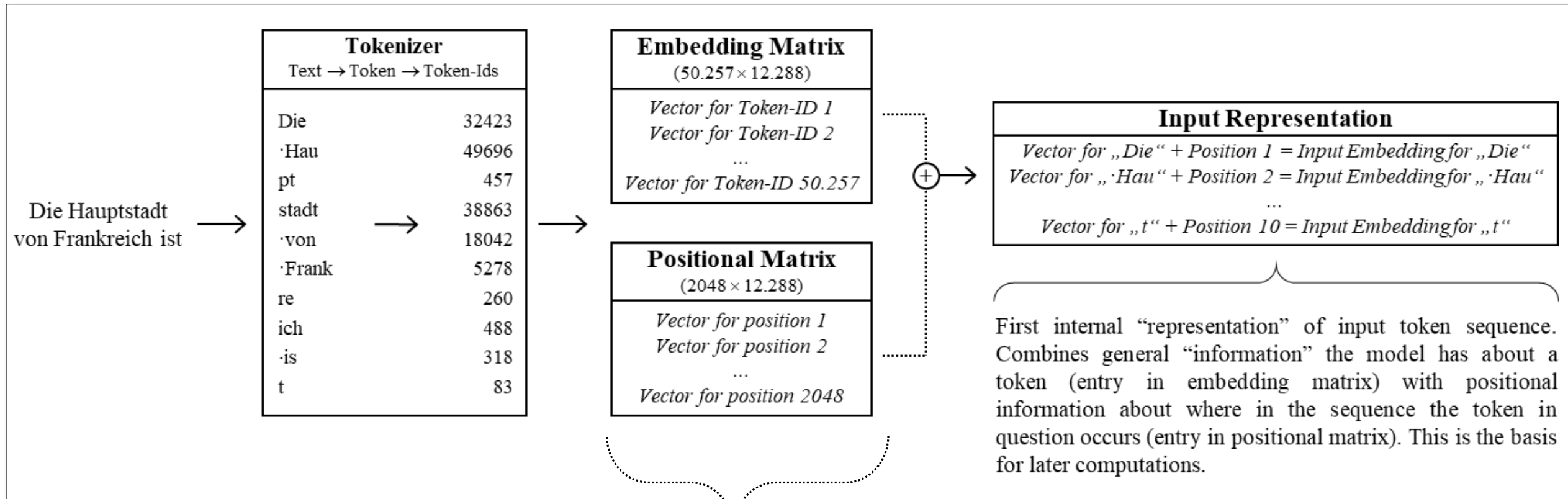
Wir brauchen noch Brot.

Wir brauchen noch Brot.

Aber bitte kein Graubrot!

Aber bitte kein Graubrot!

Transformer LLMs | $Input \Leftrightarrow IR$

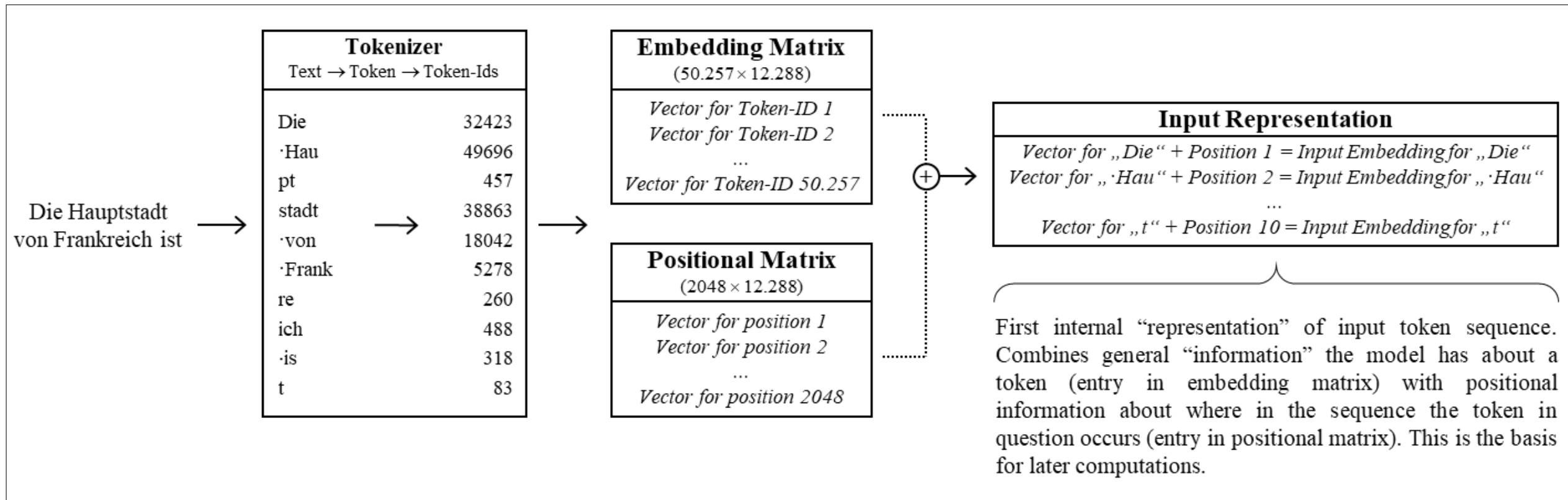


Die **Embedding-Matrix** enthält für jedes Token im Vokabular einen *Embedding Vektor* der Modelldimension d_M .

Die **Positional-Matrix** enthält für jede Position des Kontextfensters einen *Positional Vektor* der Modelldimension d_M .

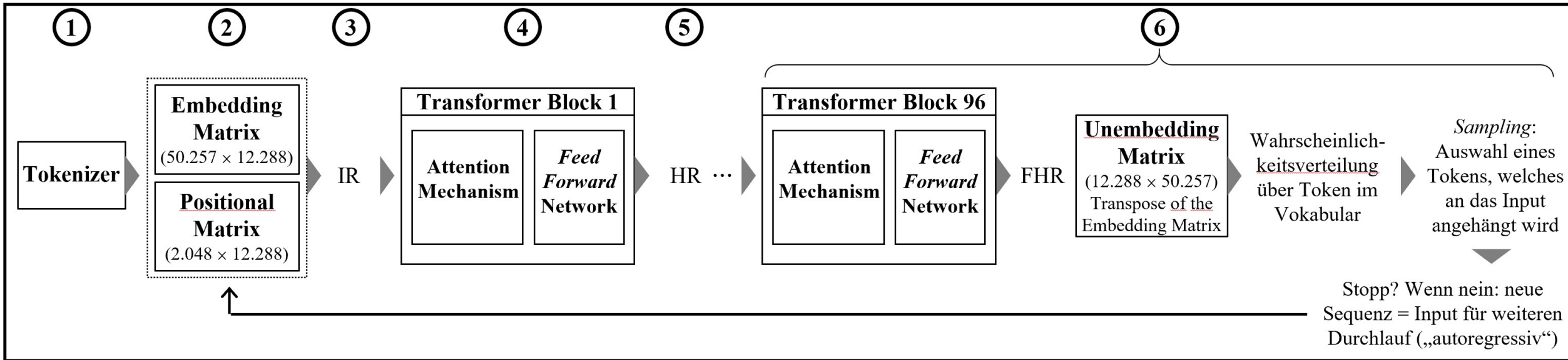
Alle Einträge in Embedding- und Positional-Matrix sind (bei GPT-3) **gelernte Parameter**.

Transformer LLMs | *Input* ⇔ *IR*

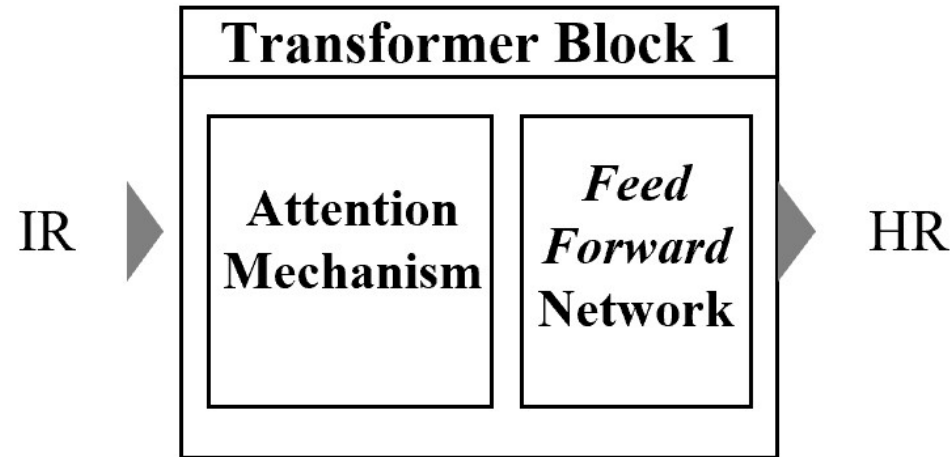


Nach *Tokenization* wird für jedes Input-Token der entsprechende Vektor aus der Embedding Matrix mit dem entsprechenden Vektor der Positional Matrix addiert. Wir erhalten so die **Input-Representation (IR)**.

Transformer LLMs | *Transformer Blocks*



Transformer LLMs | *Transformer Blocks*

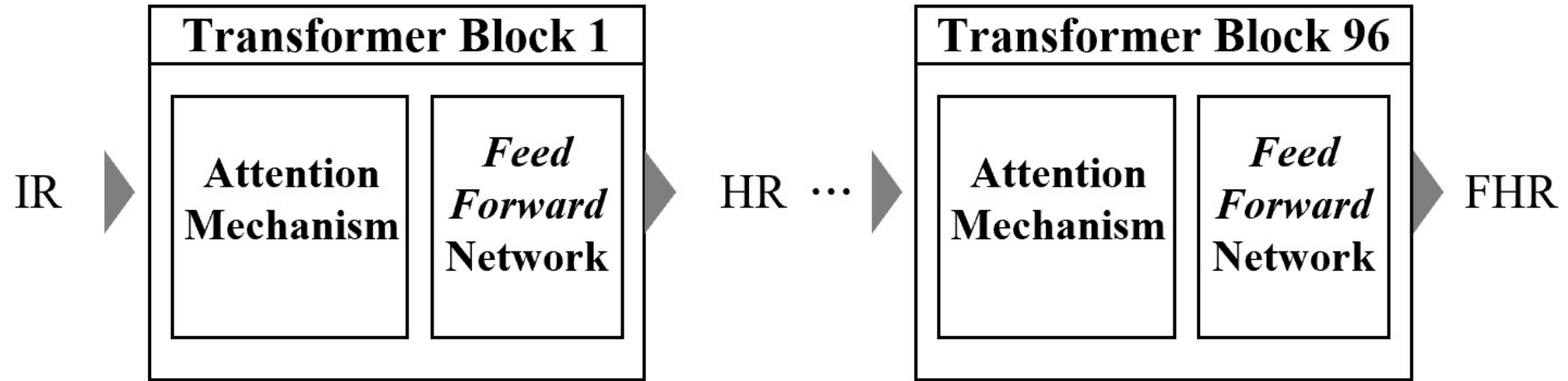


Die IR wird im ersten **Transformer-Block** weiterverarbeitet und kontextsensitiv angereichert.

Zuerst prüft der ***Attention-Mechanismus***, welche früheren Token für ein gegebenes Token relevant sind, und mischt entsprechende Informationen in dessen Vektor ein.

Danach verarbeitet ein ***Feed Forward Netz*** jeden Vektor einzeln weiter.

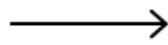
Transformer LLMs | *Transformer Blocks*



GPT-3 umfasst 96 *Transformer Blocks*...

Transformer LLMs | *Transformer Blocks*

IR/HR
($n \times 12288$)



Attention Layer

GPT-3: 96 Attention-Heads, die parallel arbeiten. Wir ignorieren zur Vereinfachung die Aufspaltung in einzelne Heads.

W_Q
Query
Matrix

Jede dieser Matrizen hat die Größe 12288×12288 , enthält also rund 151 Millionen gelernte Parameter.

W_K
Key
Matrix

Durch Matrix-Multiplikation mit IR/HR erhalten wir für jede *Token-Representation* I :

- Query-Vektor (wonach sucht I ?)
- Key-Vektor (worauf antwortet I ?)
- Value-Vektor (was steuert I bei?)

W_V
Value
Matrix

Für jede *Token-Representation* I :

- Vergleiche den *Query*-Vektor mit den *Key*-Vektoren der *Token-Representations*, die vor I (bis inklusive I) kommen.
- Dies bildet die Grundlage für eine Gewichtung, wie relevant frühere *Token-Representations* für I sind.
- Dies nutzen wir, um eine gewichtete Summe der *Value*-Vektoren der anderen *Token-Representations* zu bilden.
- Diese Summe leistet dann einen Beitrag zum Update der *Token-Representation* I .

Aktualisierte HR

(+ Residual Stream)



Transformer LLMs | *Transformer Blocks*

Feed Forward Network

(Häufig auch *Multi Layer Perceptron* (MLP) genannt)

Ein einfaches neuronales Netz (strukturell analog zu dem Beispiel, welches wir uns in der letzten Sitzung für die Ziffern-Erkennung angeschaut hatten).

Alle *Token-Representations* durchlaufen das MLP einzeln, es gibt keine „Interaktion“ zwischen den *Representations* für einzelne Token (anders als in der Attention Layer, wo *Token-Representations* über den Query/Key/Value-Mechanismus andere *Token-Representations* beeinflussen können).

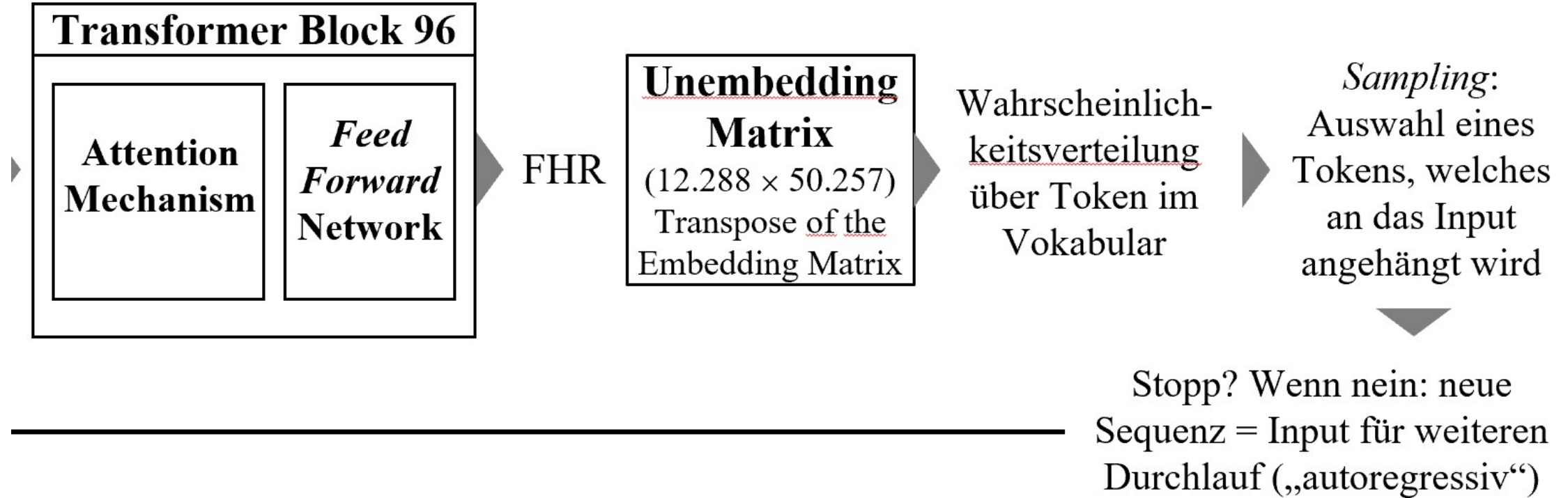
Die typische Rolle des MLP ist: die gerade durch Attention kontextualisierte *Token-Representations* nichtlinear „umzuformen“—Features verstärken/unterdrücken, neue Merkmale kombinieren, Bedeutungsaspekte umkodieren. Es ist der Teil, der pro Token „Rechenarbeit“ macht, während Attention die Informationen „zusammenträgt“.

Jedes der MLP bei GPT-3 umfasst allein 1,21 Milliarden Parameter.

Aktualisierte HR

(+ Residual Stream)

Transformer LLMs | *FHR* \Rightarrow Output



GPT vs. ChatGPT

Wir müssen ein reines LLM (das neuronale Netz) unterscheiden von dem größeren Gesamtsystem, in welches das LLM eingebettet ist.

GPT Das LLM

ChatGPT Das Gesamtsystem, mit dem User interagieren

Während das LLM Text entgegennimmt und autoregressiv (Token-für-Token) einen Output generiert, verfügt das Gesamtsystem über weitere Komponenten.

User-Prompt vs. vollständiges Input

Sag mir, wenn du so weit bist.

+ Wie funktionieren Kiemen?



Wenn wir als User mit einem LLM interagieren, bildet unser Prompt in der Regel nur einen kleinen Teil des tatsächlichen Inputs, welches das LLM entgegen nimmt.

Der Rest ist für uns als User nicht sichtbar.

Das vollständige Input umfasst zusätzlich zum Prompt in der Regel noch:

**System message +
developer instructions**
(general rules, safety rules, ...)

User/profile context

Conversation context

Current user prompt

External material

GPT vs. ChatGPT | Erweiterungen

Wichtige Erweiterungen einfacher GPT-basierter LLMs

- Skalierung: mehr Parameter, mehr Daten, mehr Rechenleistung
- Instruction Tuning / RLHF: von einer „Textfortsetzungsmaschine“ zu einem hilfreicherem Assistenten
- Prompting / In-Context Learning: Aufgaben werden über den Eingabeprompt spezifiziert
- Chain-of-Thought-ähnliches Schlussfolgern: Zwischenschritte können einige Problemlösungsaufgaben verbessern
- Retrieval-Augmented Generation: das Modell kann externe Dokumente oder Suchergebnisse nutzen
- Tool-Nutzung: das Modell kann Rechner, Suchmaschinen, Code-Tools, Datenbanken usw. aufrufen
- Multimodalität: das Modell kann Text zusammen mit Bildern, Audio, Video usw. verarbeiten
- Längere Kontextfenster: das Modell kann mehr vorherigen Text oder Dokumente berücksichtigen
- Fine-Tuning / Adapter: Modelle können an spezifische Aufgaben oder Domänen angepasst werden
- Mixture-of-Experts: nur Teile eines sehr großen Modells werden für eine gegebene Eingabe aktiviert

Multimodalität

- GPT-3-artige LLMs arbeiten vor allem mit Text-Tokens als Input und Output.
- Multimodale Systeme können mit verschiedenen Datentypen umgehen: Text, Bilder, Audio, Video usw.
- Grundidee: Unterschiedliche Datentypen werden in Vektorrepräsentationen umgewandelt, die das Modell verarbeiten kann.
- Das Modell lernt Zusammenhänge zwischen Modalitäten, zum Beispiel:
 - Bild → Beschreibung, Text → Bild, Sprache → Text, Bild + Frage → Antwort.
- Dadurch wird flexiblere Interaktion möglich: Das System kann Bilder „anschauen“, gesprochene Anweisungen verarbeiten oder visuellen und sprachlichen Kontext kombinieren.

Chain of Thought & Reasoning Models

Eine *Chain of Thought* ist eine explizite Sequenz sprachlicher Zwischenschritte, durch die ein LLM eine Aufgabe in Teilprobleme zerlegt und daraus eine Antwort ableitet.

Solche Zwischenschritte können die Leistung bei Aufgaben verbessern, die mehrstufiges Rechnen, Schlussfolgern oder Planen erfordern.

Dabei ist nicht garantiert, dass der angezeigte „Gedankengang“ den tatsächlichen internen Verarbeitungsprozess des Modells vollständig oder zuverlässig abbildet.

Chain of Thought & Reasoning Models

Ein Reasoning Model ist ein LLM, das durch Training gezielt dazu gebracht wird, mehrstufige Problemlösungsprozesse zu nutzen.

Dazu können versteckte oder sichtbare Zwischenschritte, zusätzliche Rechenzeit während der Antwortgenerierung, Verstärkungslernen (RLHF), Verifikation oder Feedback auf Zwischenschritte gehören.

Der zentrale Unterschied zu einem gewöhnlichen LLM liegt nicht darin, dass es eine völlig andere Art von System wäre, sondern darin, dass Training und Nutzung stärker auf zuverlässiges mehrstufiges Schließen optimiert sind.

Überblick & Vorgehen

Ziel Ein System, welches im Medium der natürlichen Sprache „sinnvoll“ auf natürlichsprachlichen Text reagieren kann (was dann eine ganze Reihe weiterer „Fähigkeiten“ nach sich zu ziehen scheint – dazu später mehr ...) und zwar idealerweise ohne enge thematische Begrenzungen.

Zunächst eine kleine Beobachtung →

①

Es war (auch für viele Expert*innen) *überraschend* und *einschneidend*, dass das Ziel jetzt erreicht wurde

②

Vorüberlegungen: Wie könnte man einen Computer dazu bringen, sich mit einem Menschen zu unterhalten?



Dann ein wenig zum Schwierigkeitsgrad, der mit dem Ziel verbunden ist.

Das führt uns zu einer →

③

Unterscheidung: Symbolische und subsymbolische KI

④

Interpretierbarkeit, Erklärungen, Verstehen, Wissenschaft vs. *Engineering*



Auf die Relevanz dieser Unterscheidung werden wir später noch zurückkommen

Kritik?

Logic-based AI is the most ambitious approach to AI, because it proposes to understand the common sense world well enough to express what is required for successful action in formulas. Other approaches to AI do not require this. Anything based on neural nets, for example, hopes that a net can be made to learn human-level capability without the people who design the original net knowing much about the world in which their creation learns. Maybe this will work, but then **they may have an intelligent machine and still not understand how it works**. This prospect seems to appeal to some people.

John McCarthy (1996): „What Computers Still Can't Do“ (Review des gleichnamigen Buches von Hubert Dreyfus)

Chomsky

There's a kind of an industry in computational cognitive science and computer science trying to show that you can get significant knowledge of a language by statistical analysis of text. Antecedently, that's extremely unlikely to succeed. You do not get discoveries in the sciences by taking huge amounts of data, throwing them into a computer, and doing statistical analysis of them. Try to think it through in the history of the sciences. It just doesn't happen. That's not the way you understand things. You have to have theoretical insights. You have to know what kind of experiments to carry out-- what kind of data are worth looking at, which kind of throw away, and so on. That's the way the sciences have always worked.

Chomsky

If you wanted to, say, study the laws of motion, you could take a huge number of videotapes of what's happening outside the window, and subject them to statistical analysis. You could get a pretty good prediction of the next thing that's going to happen outside the window-- actually, a better prediction than what the physics department can give. But it's not science. It's a way of matching data, and maybe predicting some new data. But that's not what understanding is. And it's very unlikely to work for language, either.