



# Digital Applications & Data Management

WS25/26

Dr. Jens Kohl



# Roadmap Vorlesung

1. Einführung und Übersicht
2. Grundlagen Data Science
3. Vorgehen Data Science Use Case
4. Case Study Data Science
5. Grundlagen unüberwachtes Lernen
6. Grundlagen überwachtes Lernen  
(tabellarische Daten)
7. Case Study überwachtes Lernen  
(tabellarische Daten)
8. Grundlagen überwachtes Lernen (Bilddaten)
9. Case Study überwachtes Lernen und Transfer Learning (Bilddaten)
10. Grundlagen Generative AI
11. Generative AI mit Texten und Prompt Engineering
12. Agentic AI
13. Ausblick: Machine Learning in der Cloud und Reinforcement Learning

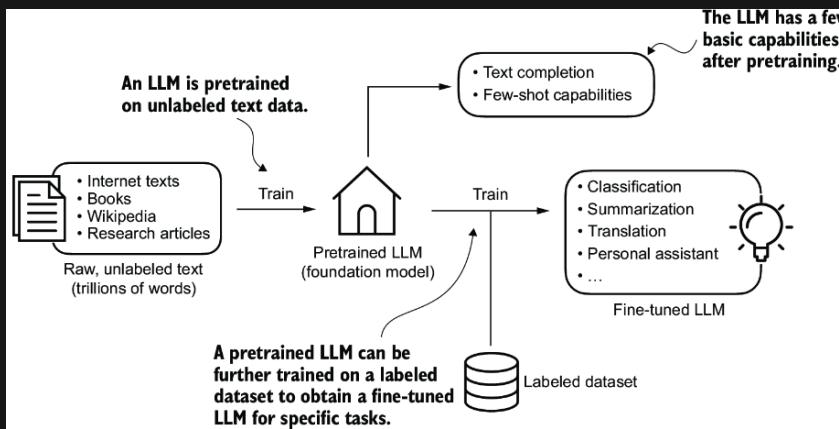


# Vorlesung 11: Generative AI mit Texten und Prompt Engineering

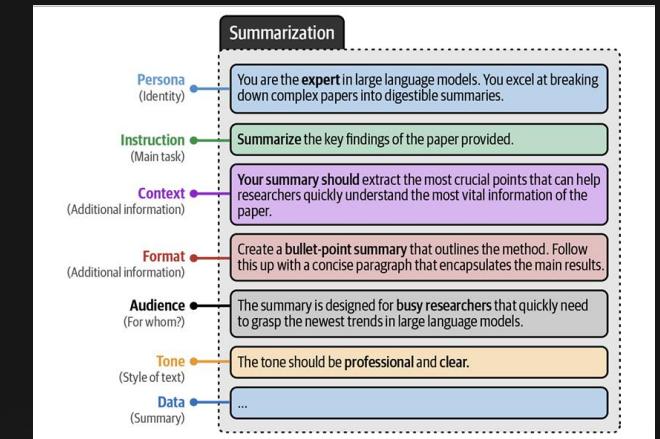


# Was machen wir heute?

## Motivation TODO



**Forbes**  
BREAKING | BUSINESS  
**Lawyer Used ChatGPT In Court—And Cited Fake Cases. A Judge Is Considering Sanctions**  
By Molly Bannon, Former Staff. Molly Bannon has been a Forbes news reporter since 2023.  
Published Jun 08, 2023, 02:06pm EDT, Updated Jun 08, 2023, 03:42pm EDT  
Share Save  
This article is more than 2 years old.  
**TOPLINE**  
The lawyer for a man suing an airline in a routine personal injury suit used ChatGPT to prepare a filing, but the artificial intelligence bot delivered fake cases that the attorney then presented to the court, prompting a judge to weigh sanctions as the legal community grapples with one of the first cases of AI "hallucinations" making it to court.



Wie funktionieren Large-Language Modelle, was sind typische LLM-Probleme und wie kann ich den Output verbessern?



# Generative AI für Texte

## Übersicht

- Historie
- Allgemeine Begriffe
- Übersicht ChatGPT-Architektur
- Welche Daten werden verwendet?
- Wie erfolgt das Trainieren des Modells?
- Wie erfolgt die Inferenz des Modells zur Laufzeit?
- Was sind typische Probleme eines LLM?
- Wie erstellt man am besten Anfragen an ChatGPT? (Prompts)
- Ausblick TODO



# Kurze Historie



# Generative AI für Texte

## Übersicht und Historie bekannter Verfahren

- 1972 Rekurrente Neuronale Netzwerke (RNN)<sup>1</sup>: neuronale Netze mit Rückkopplung. Sehr mächtig, aber ressourcenintensiv.
- 1997 LSTM<sup>2</sup> basiert auf RNN und löste zuerst Vanishing Gradient<sup>3</sup>-Problem bei zu verarbeitenden längeren Sequenzen. Milliardenfach eingesetzt, bspw. Spracherkennung (Alexa, Siri, ...). Sehr ressourcenintensives Training.
- 2015 Attention<sup>4</sup> löst Problem längerer Sequenzen durch individuelles Gewichten einzelner In-/Outputs. Kein RNN!
- 2017 Transformer<sup>5</sup> basiert auf Attention, bietet flexibleres, schnelleres Lernen als LSTM aufgrund Parallelisieren. Kein RNN!
- 2019 BERT<sup>6</sup>: Transfer Learning, basierend auf Transformer.
- Generative Pre-trained Transformer (GPT):
  - 2018: GPT-1<sup>7</sup> vortrainiertes Modell basierend auf Transformer, das gegeben Input neue Wörter generiert.
  - 2019: GPT2<sup>8</sup>
  - 2020: GPT3<sup>9</sup>
  - 2022 InstructGPT<sup>10</sup>/ ChatGPT: Fine-Tuning via Supervised Learning per Chats mit Menschen sowie Reinforcement Learning
  - 2023: Chat GPT4 Einsatz multimodaler Daten
  - 2023: Meta LLaMA<sup>11</sup>: Beschleunigung Forschung durch open-source

1 Amari, Shun-ichi, "Learning patterns and pattern sequences by self-organizing nets of threshold elements", 1977

2 Hochreiter, Schmidhuber: "Long Short-Term Memory", 1997. [Link](#)

3 Vanishing Gradient: beim Trainieren von Netzen mit vielen Schichten und Einsatz Exponentialfunktion als Aktivierungsfunktion kann bei Backpropagation Fall entstehen, daß Gradienten sehr, sehr klein werden und damit die Gewichte und Bias des Modells, v.a. der ersten Schichten, nicht mehr geändert/ trainiert werden. Vgl. Hochreiter: „Untersuchungen zu dynamischen neuronalen Netzen“, 1991.

4 Vaswani et al.: "Attention is all you need", 2017. [Link](#)

5 Bahdanau et al.: "Neural Machine Translation by Jointly Learning to Align and Translate", 2015. [Link](#)  
Basierend auf Schmidhuber, "Learning to control fast-weight memories.", 1992

6 Devlin et al.: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2019. [Link](#)

7 Radford et al., "Improving Language Understanding by Generative Pre-Training", 2018, [Link](#)

8 Radford et al., "Language Models are Unsupervised Multitask Learners", 2019, [Link](#)

9 Brown et al., "Language Models are Few-Shot Learners", 2020, [Link](#)

10 Ouyang, Long, et al. "Training language models to follow instructions with human feedback", 2022, [Link](#)

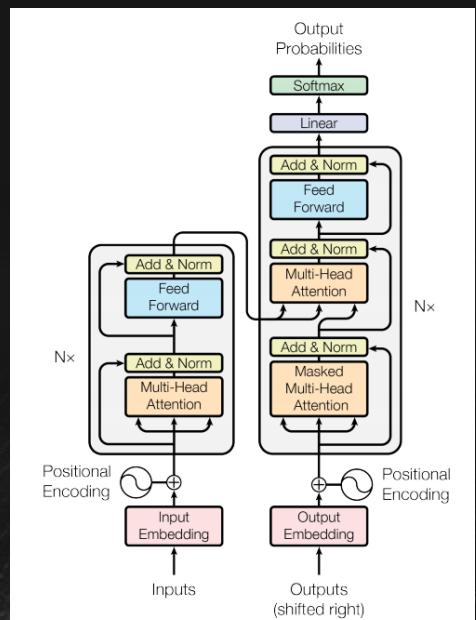
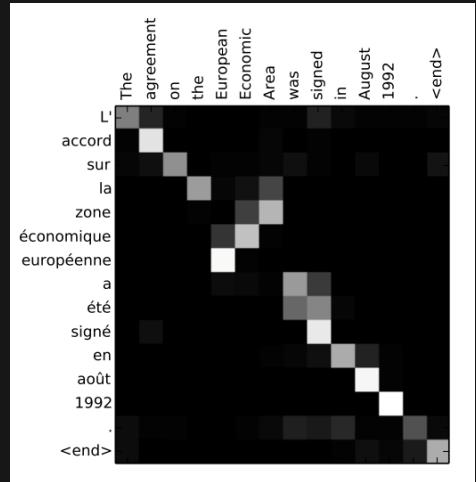
11 Touvron, Izacard et al., "LLaMA: Open and Efficient Foundation Language Models", 2023, [Link](#)



# Generative AI mit Texten

## Kurze Historie

- Schmidhuber (1992), "Learning to control fast-weight memories: An alternative to dynamic recurrent networks",  
Bahdanau (2015) et al., "Neural machine translation by jointly learning to align and translate"  
Wieso wichtig?
  - Kann Zusammenhänge von Textteilen auch über längere Textsequenzen durch die Identifizierung von Beziehungen über individuelle Gewichte lernen.
  - Das Training kann parallelisiert und somit beschleunigt werden.
  - Leider: quadratische Komplexität (auch mit Caching) 😞
- Vaswani et al. (2017), "Attention is all you need"  
Wieso wichtig?
  - Definierte die bis heute verwendete Struktur für Large Language Modelle:  
multi-head self-attention + positional encoding + NN + residual connections





# Generative AI mit Texten

Kurze Historie: der Sturm beginnt...

- Radford et al. "Improving language understanding by generative pre-training," 2018.  
Wieso wichtig: definiert Generative Pre-trained transformer (GPT1);
- Radford et al. (2019), "Language models are unsupervised multitask learners",  
Brown et al. (202), "Language models are few-shot learners"  
Wieso wichtig: Verbesserter Output durch Skalierung Parameter und Trainingsdaten

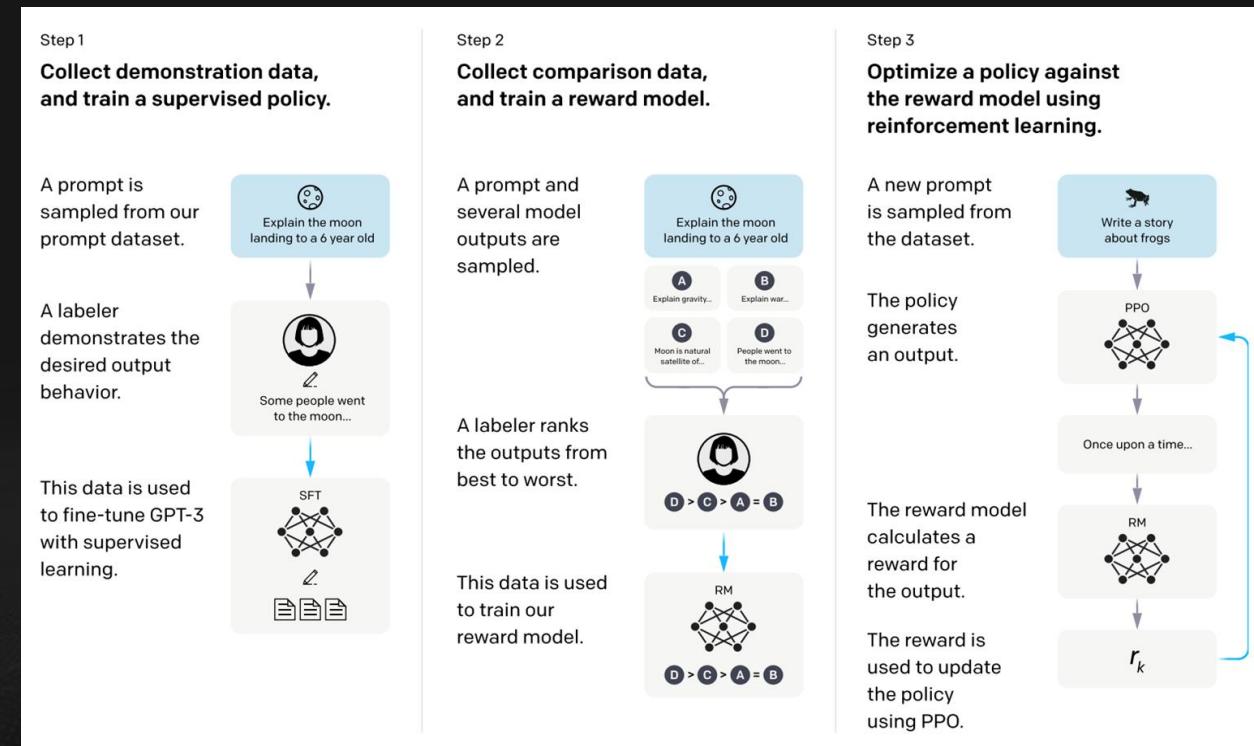
Modell	Veröffentlicht	Trainingsdaten	Anzahl Parameter	Max. Sequenzlänge
GPT-1	Juni 2018	BooksCorpus Datensatz (~11000 Bücher)	117 Millionen	1024
GPT-2	February 2019	BooksCorpus, CommonCrawl (WebArchiv: >3 Mrd. Websites mit ~417TB Daten), WebText (> 40GB von Reddit)	1.5 Milliarden	2048
GPT-3	June 2020	BookCorpus, Common Crawl, Wikipedia, Bücher, Artikel, ...	175 Milliarden	4096
GPT-4	March 2023	Unbekannt	>1 Billion	Unbekannt



# Generative AI mit Texten

Kurze Historie: der Sturm beginnt...

- Ouyang et al. (2022), "Training language models to follow instructions with human feedback" (InstructGPT).  
Wieso wichtig: Einsatz von Human Feedback verbessert Qualität des Modells enorm



Mehr Details in  
späterer Vorlesung!



# Generative AI mit Texten

Kurze Historie: der Sturm beginnt...

- Touvron et al. (2023), "Llama: Open and efficient foundation language models" and "Llama 2: Open foundation and fine-tuned chat models"

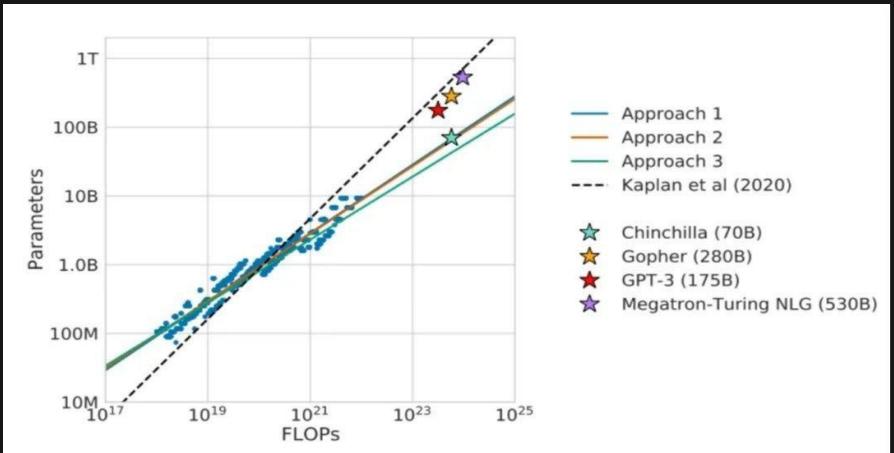
Wieso wichtig:

- Veröffentlichung des Source Codes und der Methoden für Entwicklung eines LLM
- Sehr detailliert
- Sehr gut geschrieben....



# Generative AI mit Texten

Wir sind aktuell in der Optimierungsphase



„Chinchilla“  
Scaling Laws

Hoffmann et al (2022), “Training compute-optimal large language models”.

$$\begin{cases} C = C_0 ND \\ L = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + L_0 \end{cases}$$

→ 20 text token per parameter



Pre-training as we know it will end

Compute is growing:

- Better hardware
- Better algorithms
- Larger clusters

Data is not growing:

- We have but one internet
- The fossil fuel of AI

Aktuell wird sehr viel geforscht, die Parameteranzahl zu reduzieren bei gleicher/ besserer Modellgenauigkeit (analog Entwicklung Transfer learning Modelle für Bilder, bspw. VGG16 zu Inception).  
Das ist aber außerhalb des Fokus dieser Vorlesung...



# Begriffe



# Generative AI für Texte

## Begriffe

- Multi-Modal Model: Modell, das verschiedene Eingaben verarbeiten kann (Text, Bilder, ...)
- LLM: Large-language model
- SLM: small-language model
- Token: bestimmte Zeichenfolge, die das Modell lesen oder verstehen kann. Kann Buchstabe sein, Teil eines Worts oder ganzes Wort.
- Foundation Model: Modell, das auf sehr großer Datenmenge trainiert wird und angepaßt werden kann an bestimmte Umfänge
- Fine-tuning: Anpassen eines Modells an bestimmte Aufgaben oder Umfänge.

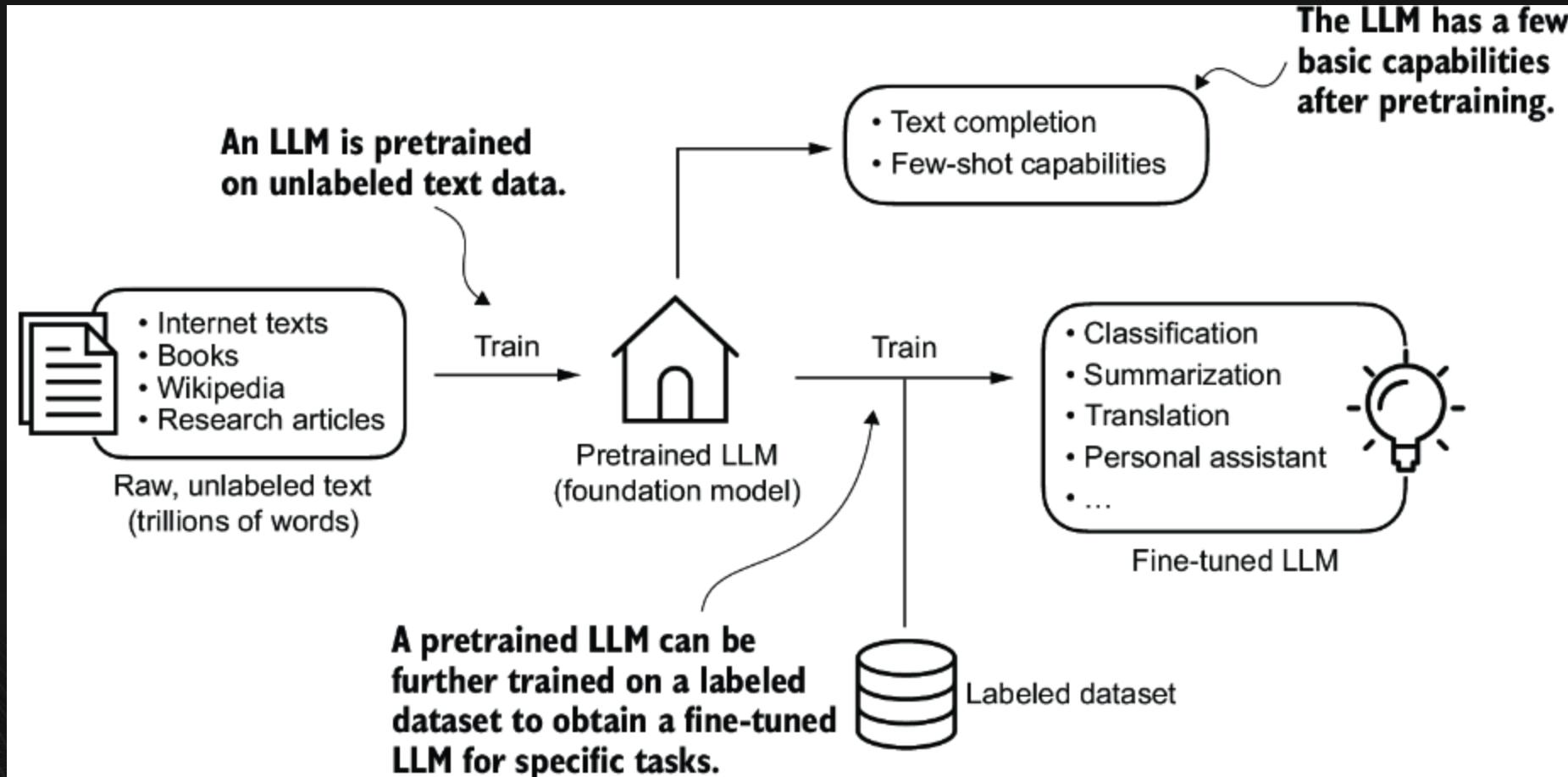


# Funktionsweise



# Generative AI für Texte

## Allgemeine Übersicht LLM





# TABELLARISCHE DATEN & BILDER



# SEQUENTIELLE DATEN

- Zeitreihen: Aktienkurse, Messungen, Temperaturkurven, ...
  - Video: Folge von einzelnen Bildern (Frames per Second)
  - Sprache: Folge von einzelnen gesprochenen Wörtern
  - Texte: Folge von einzelnen geschriebenen Wörtern

Bei sequentiellen Daten haben (im Gegensatz zu statischen) vorherige Daten Einfluß auf die aktuellen und nachfolgenden Daten.



# Einschub: Natural Language Processing

## SPRACHEN UNTERSCHIEDEN SICH DEUTLICH

- Verschiedene Wortformen:
  - Deklinationen, Kasus oder Konjugation (bspw. Englisch vs. Russisch vs. Finnisch).
  - Zeiten (viele Zeiten wie im Englischen oder Altgriechischen vs. keine Zeiten wie im Chinesischen).
- Satzstellung: Stellung Verb im Satz, Adjektiv vor/ nach Nomen, ...
- Akzente (andere Betonungen) und Dialekte (andere Wörter).

## TEXTE UNTERSCHIEDEN SICH DEUTLICH

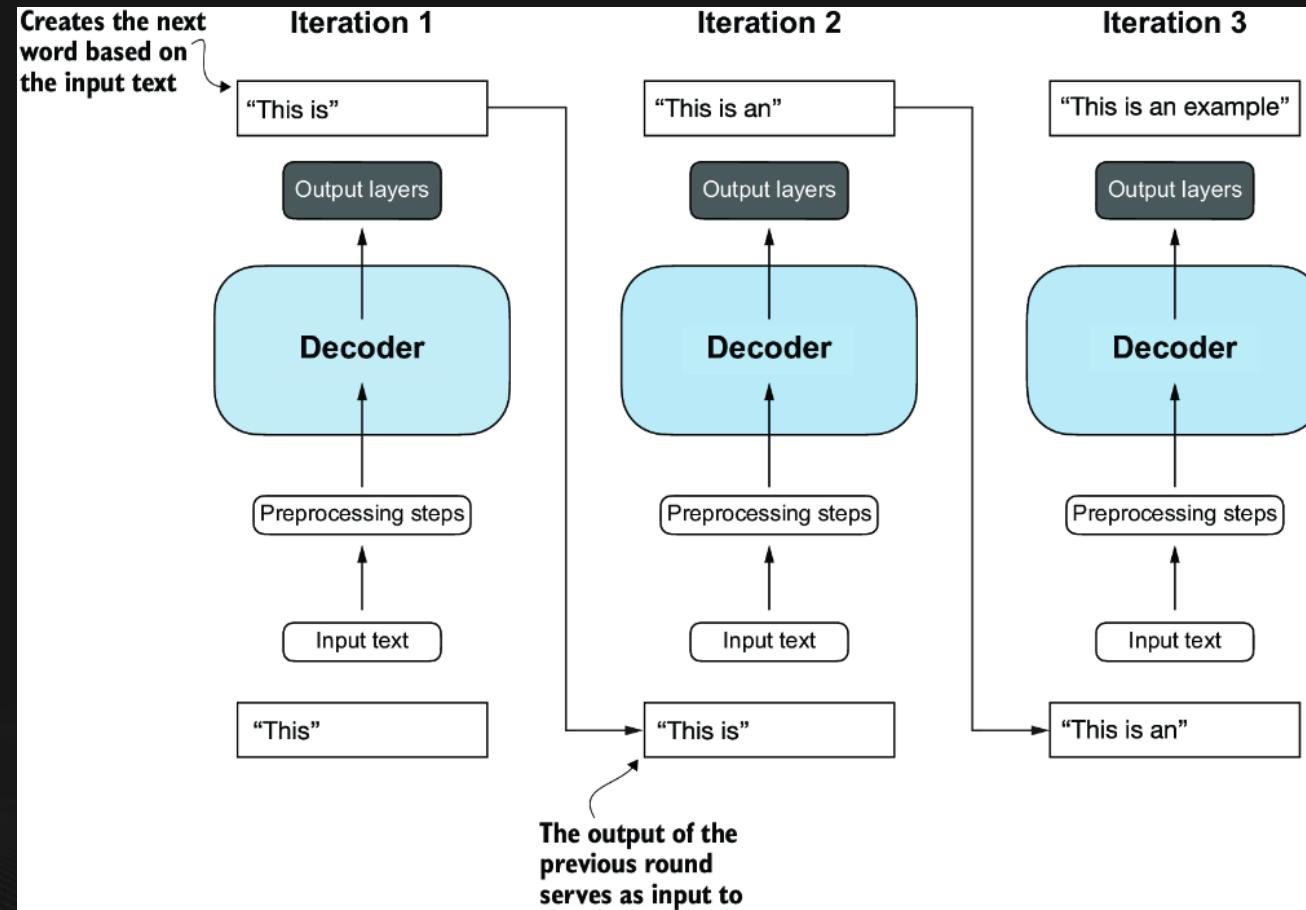
- Schrift:
  - Buchstabenalphabet (latein/ kyrillisch/ ...)
  - Zeichenschrift (chinesisch/japanisch/koreanische)
  - Schrift ohne Vokale (arabisch/ ...)
- Wortabstand oder kein Wortabstand
- Schriftrichtung:
  - links, rechts (bspw. Hebräisch oder Arabisch)
  - Nach unten (bspw. chinesisch)

NLP ermöglicht das effiziente Verarbeiten von Texten für Machine Learning



# Generative AI für Texte

Wie funktioniert das? Inferenz



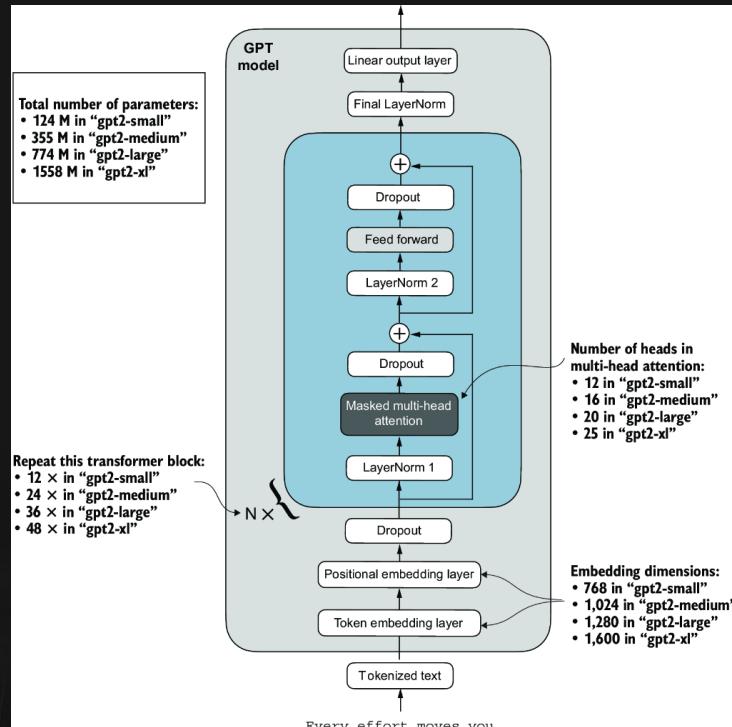
Gegeben ein Input von mehreren Wörtern, möchten wir wissen, was das wahrscheinlichste nächste Wort/ Token ist.



# Generative AI für Texte

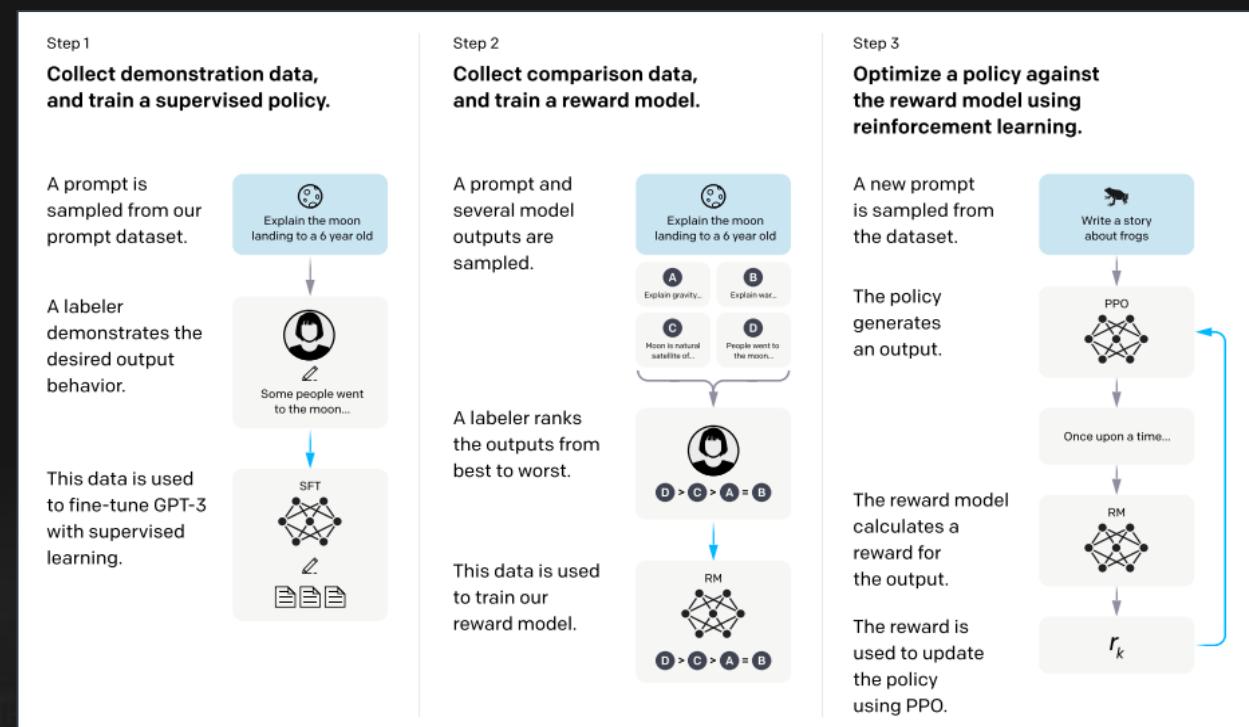
## Übersicht Training

### Schritt 1: Trainieren Large-Language Modell



Lernt „Large Language Modell“, das für eine Inputsequenz die wahrscheinlichsten Folgewörter vorhersagt

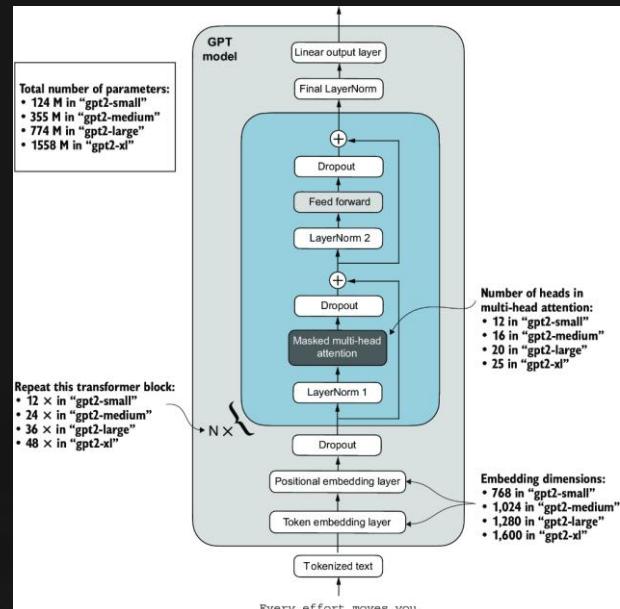
### Schritt 2: Fine-Tuning Modell (ab ChatGPT 3.5)



Modell lernt repräsentative Anwendungsfälle von/ mit Menschen und Belohnungsfunktion. Diese Funktion ermöglicht Modell, selbst zu erkennen, ob Chat „gut“ läuft und so Optimierung Chatverlauf.



# Chat GPT im Detail



Eingabe: Sequenz mit Länge 2048 Wörter.  
Bei kürzeren Anfragen wird Sequenz mit  
leeren Wörtern aufgefüllt

Word	Wahrscheinlichkeit
Certainly, here is the recipe ....	20%
Shaken	10%
....	...
....	....
Car	1%

Ausgabe: Länge 2048 Wörter mit  
Wahrscheinlichkeitsverteilung  
(meist interessiert man sich nur für letztes  
Wort der Sequenz)

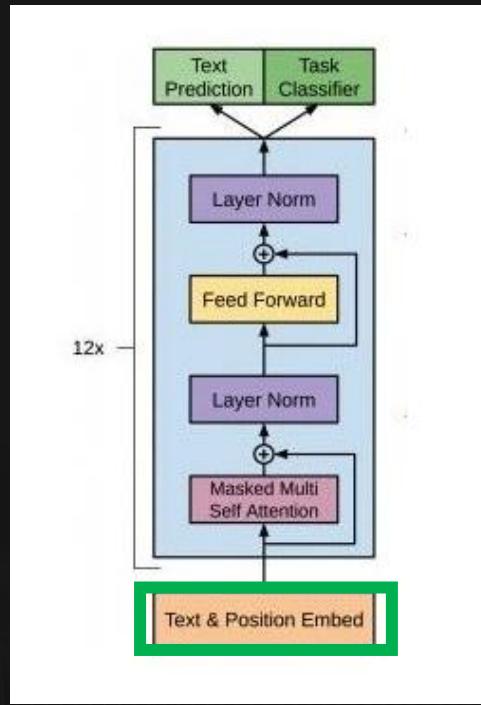
Ein ganzer Satz wird wie folgt vorhergesagt:  
I want Vodka Martini → shaken  
I want Vodka Martini shaken → not  
I want Vodka Martini shaken not stirred

Aber: Computer verstehen keine Wörter, sondern Zahlen. Deshalb brauchen wir ein Wörterbuch mit eindeutiger Nummer je Wort.



# Generative AI

## Text Embedding



<START>	I	want	Vodka	Martini		
---------	---	------	-------	---------	--	--

2048 Zeilen

0	1	0	...	0	<START>
0	0	1	...	0	I
0	0	0	...	0	want
0	0	0	1	0	Vodka
0	0	0	...	0	Martini
:	:	:	:	:	....
0	0	0	...		

Länge Vokabular = 50256 (vor Chat GPT2) bzw. 100256 (ab GPT4)

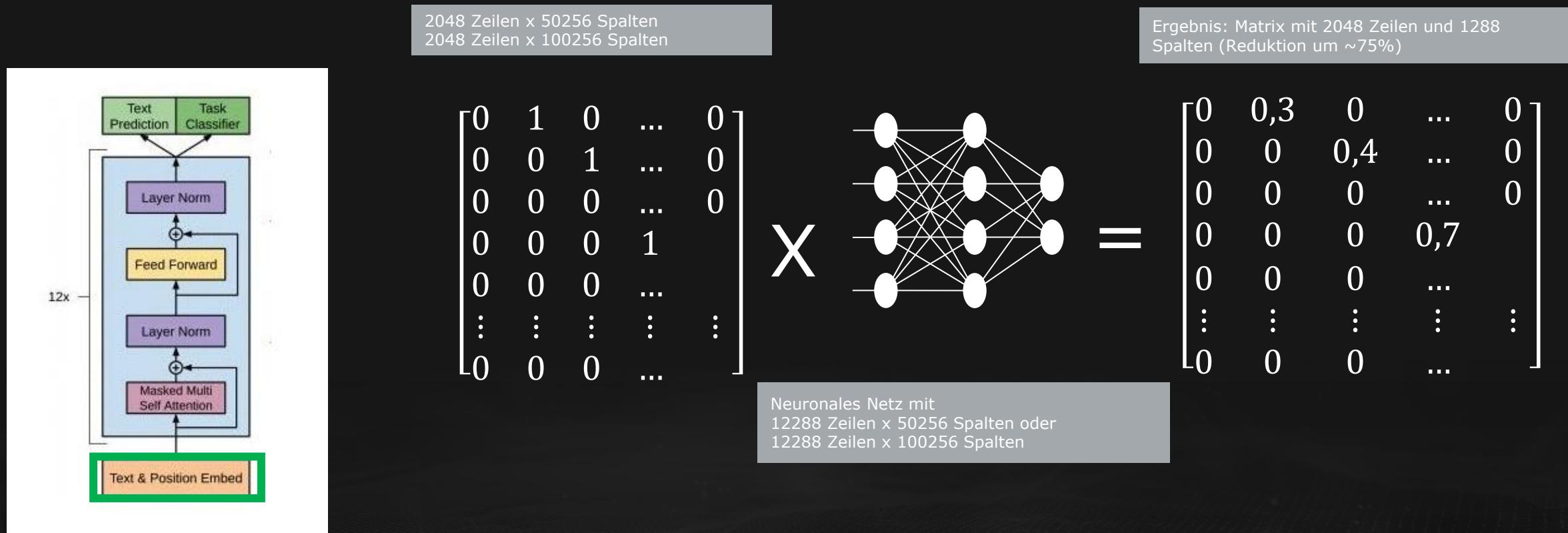
Wörterbücher verfügbar unter:  
– GPT2: [Link](#)  
– GPT4: [Link](#)

Jedes Wort erhält eine Indexnummer aus dem Wörterbuch. Wörterbuch basiert aber mehr auf Silben (Effizienzgründe).



# Generative AI für Texte

## Text embedding

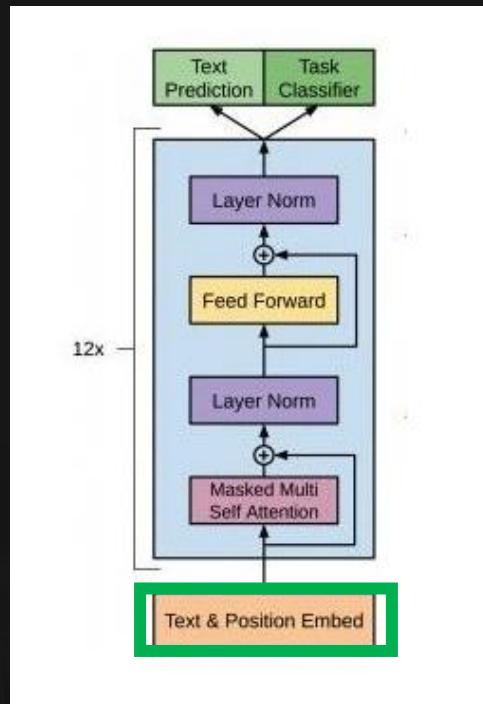


Das Encoding ermöglicht eine effizientere Speicherung der eingegebenen Wörter und reduziert somit den Aufwand und die Kosten.



# Generative AI für Texte

## Positional Embedding



Ergebnis: Matrix mit 2048 Zeilen und 1288 Spalten (Reduktion um ~75%)

$$\begin{bmatrix} 0 & 0,3 & 0 & \dots & 0 \\ 0 & 0 & 0,4 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0,7 & \\ 0 & 0 & 0 & \dots & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \end{bmatrix}$$

=

$$\begin{bmatrix} 0 & 1,3 & 0 & \dots & 0 \\ 0 & 0 & 0,8 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0,4 & \\ 0 & 0 & 0 & \dots & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \end{bmatrix}$$

+

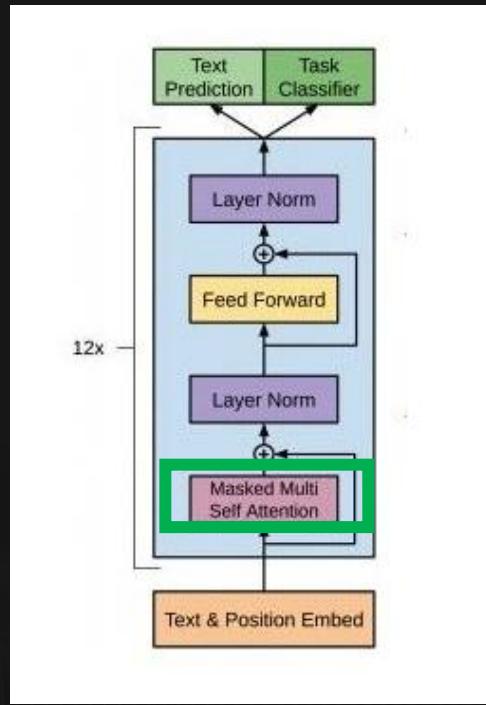
Positional Encoding-Funktionen  
(Sinus und Cosinus-Funktionen)

Durch Positional Encoding speichern wir die Reihenfolge der Wörter und können so bspw. „A liebt B“ von „B liebt A“ unterscheiden.

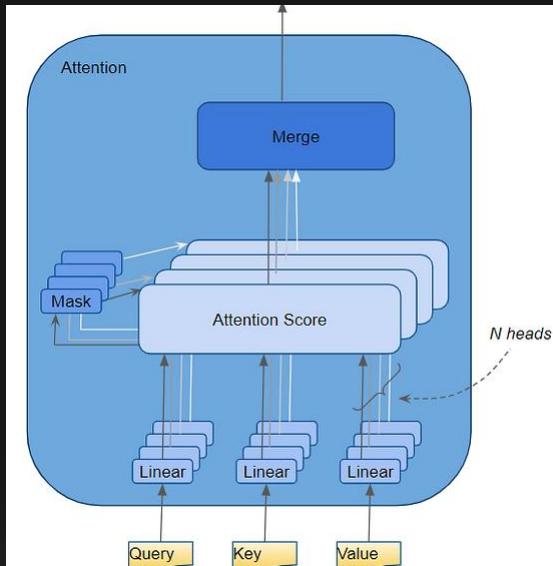


# Generative AI für Texte

## Attention layer



$$\begin{bmatrix} 0 & 1,3 & 0 & \dots & 0 \\ 0 & 0 & 0,8 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0,4 & \\ 0 & 0 & 0 & \dots & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \end{bmatrix}$$



Stellen Sie sich das vor, als ob ein Graph gebildet wird, welches Wort mit welchem am meisten zusammenhängt

Key: Wörter, die das Modell kennt

Query: Anfragen an das Modell

Value: Wert, wie stark ein Query-Key Paar zusammengehören

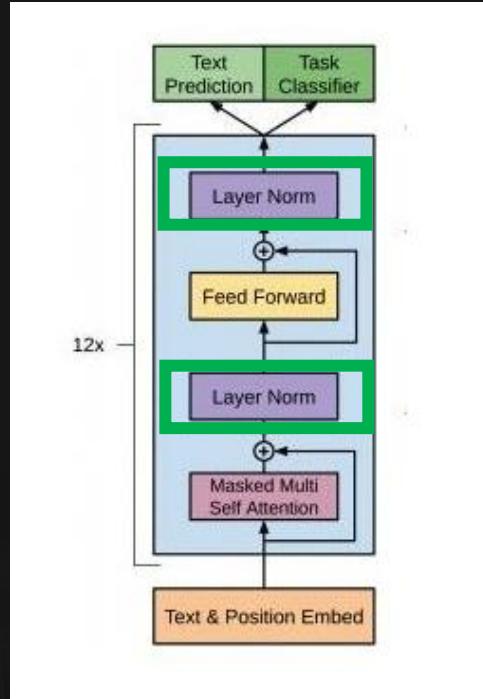
Anfragen an Modell werden mit Werten für Keys multipliziert und dann angepaßt. Dadurch ergibt sich ein Score, der sagt, wie gut jeder Teil der Anfrage auf jeden Key passt (bspw. Tomaate vs Tomate, Tomte vs Tomate, Tante vs. Tomate). Diese Matching-Wahrscheinlichkeit mal aktueller Wert in Value ergibt Output. Dies wird 12mal gemacht

Durch den Multi-Attention Layer lernt das Modell, was die wichtigsten Elemente in der Sequenz sind und wie die Elemente untereinander zusammenhängen (Lernen von Korrelationen). Das Lernen wird durch das „Masked“ verbessert, da das Modell nachfolgende Elemente in der Sequenz nicht sehen kann und diese somit vorhersagen muss.



# Generative AI für Texte

## Layer Normalization

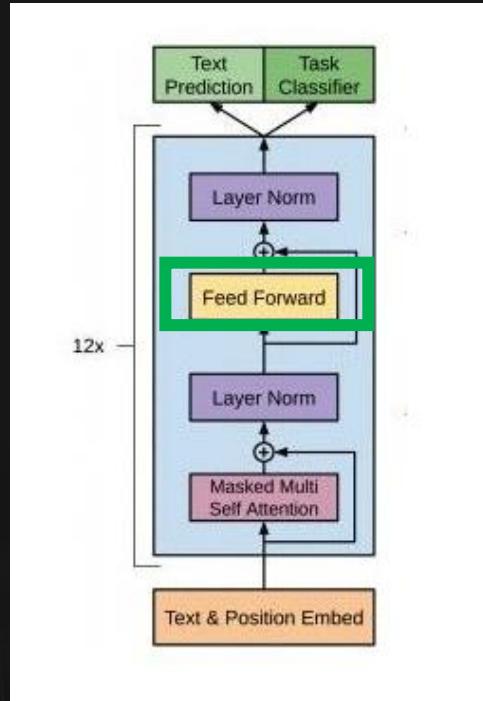


Bei den Rechenoperationen im vorigen Schritt entstehen oft sehr große/kleine Werte. Dadurch wird Modell-Performance schlechter. Layer Norm ordnet alle Werte in einen definierten Bereich ein und optimiert so die Performance.

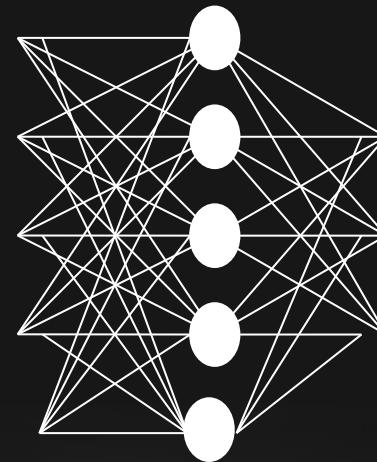


# Generative AI für Texte

## Feed Forward



$$\begin{bmatrix} 0 & 1,3 & 0 & \dots & 0 \\ 0 & 0 & 0,8 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0,4 & \\ 0 & 0 & 0 & \dots & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \end{bmatrix}$$



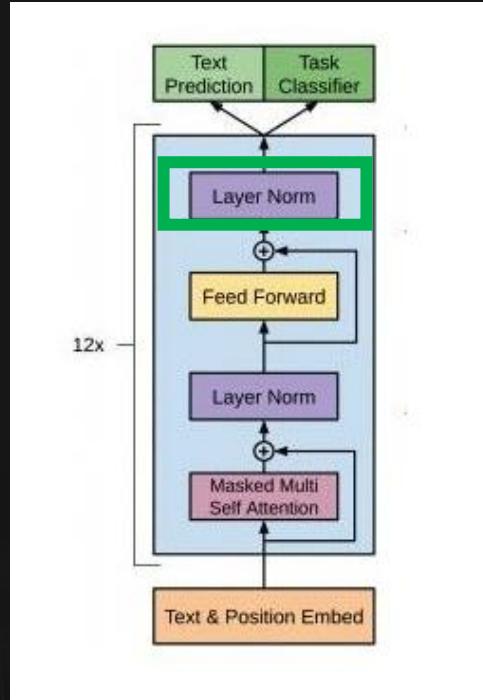
$$\begin{bmatrix} 0 & 5,3 & 0 & \dots & 0 \\ 0 & 0 & 3,8 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 2,4 & \\ 0 & 0 & 0 & \dots & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \end{bmatrix}$$

Der Feed Forward Layer ist ein "normale" Neuronales Netz und ermöglicht dem Modell, nicht-lineare, komplexe Zusammenhänge in der Eingabesequenz zu lernen



# Generative AI für Texte

## Layer normalization

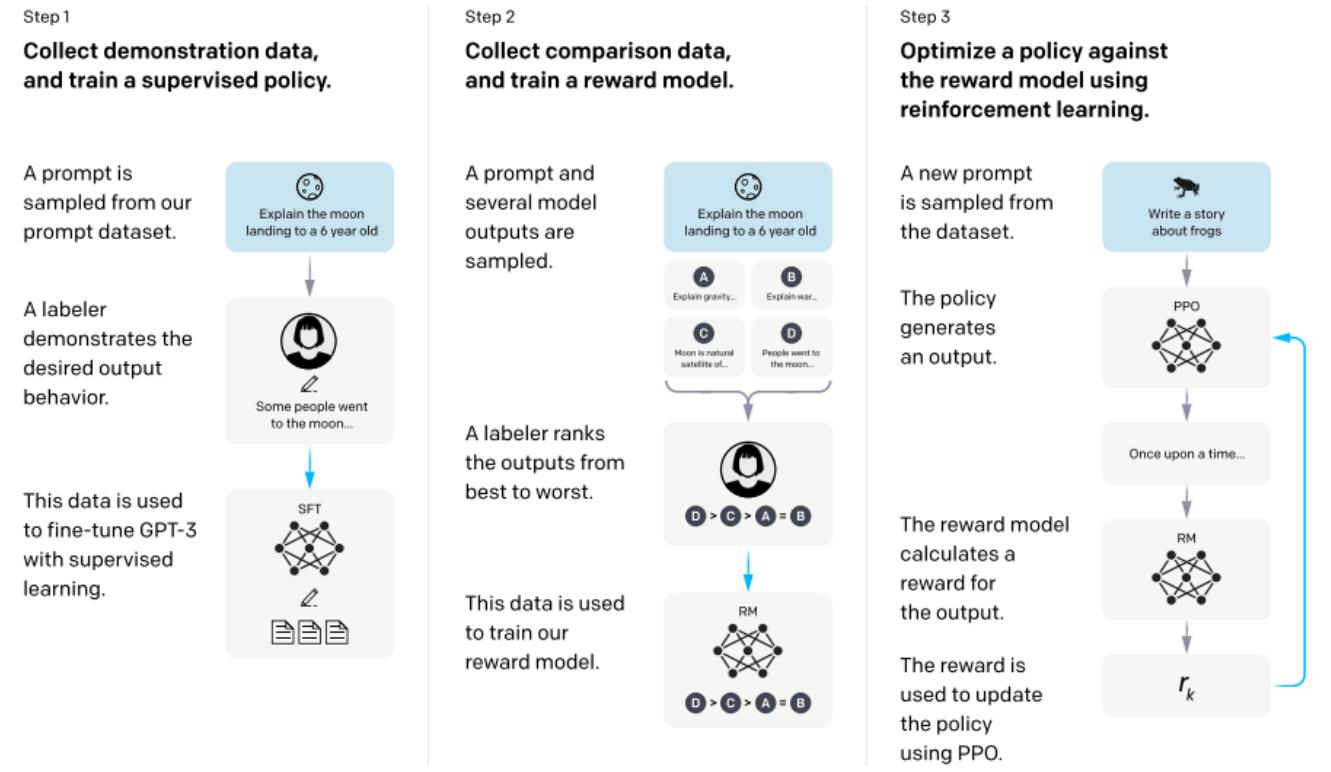


Bei den Rechenoperationen im vorigen Schritt entstehen oft sehr große/kleine Werte. Dadurch wird Modell-Performance schlechter. Layer Norm ordnet alle Werte in einen definierten Bereich ein und optimiert so die Performance.



# Generative AI für Texte

## Optimierung Modell via Supervised und Reinforcement Learning



### Detaillierung Schritte:

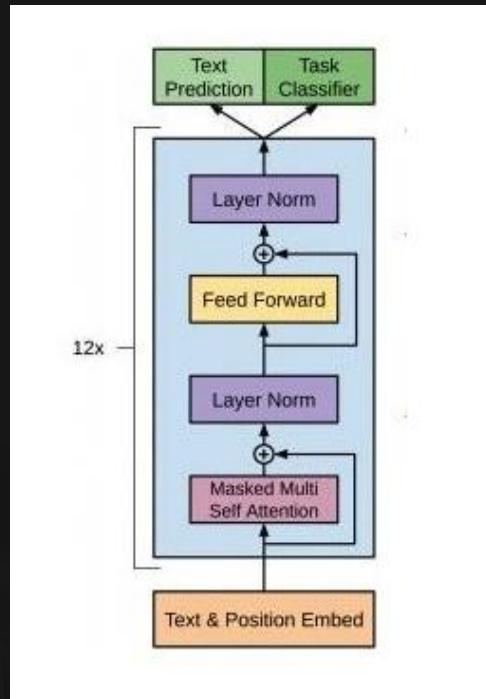
- **Step 1:** Menschen labeln/ bewerten fertige Chats als "Gut" oder "schlecht" (**überwachtes Lernen**)
- Step 2: Modell generiert verschiedene Outputs auf ausgewählte Prompts. Diese Outputs werden von Menschen gerankt. Dadurch lernt Modell eine Belohnungsfunktion für Bewertung seiner Ausgaben.
- Step 3: Belohnungsfunktion ermöglicht Modell, sich selber stets zu verbessern. Das heißt, es lernt, welche Ausgaben den höchsten Nutzen für Anwender und somit höchste Belohnung haben (**Reinforcement Learning**).

Schritte 1 und 2 sind aufgrund menschlicher Einbindung nicht beliebig skalierbar.  
Das Lernen der Belohnungsfunktion ermöglicht durch autonomes Verbessern Modell.



# Generative AI für Texte

## Inferenz zur Laufzeit



- Start-Eingaben:
  - Input/ Prompt als Sequenz mit START-Token.
  - Output: leere Sequenz mit “Start-of-Sentence”-Token.
- Diese beiden Sequenzen werden in das Modell eingegeben und in Token aufgespalten.
- Output Modell ist Wahrscheinlichkeitsverteilung für nächste Token.
- Token mit höchster Wahrscheinlichkeit wird genommen und an die Decoder-Sequenz angehängt (die jetzt 1 Token mehr hat).
- Die erstellte Sequenz wird wieder in das Modell eingegeben und der Output des Modells an die Sequenz angehängt. Der vorige Schritt wird solange wiederholt bis entweder ein End-of-Sentence-Token vorhergesagt wird oder die maximale Satzlänge erreicht ist.
- Die Sequenz wird in eine menschenlesbare Form umgewandelt und zurückgegeben („detokenisiert“).

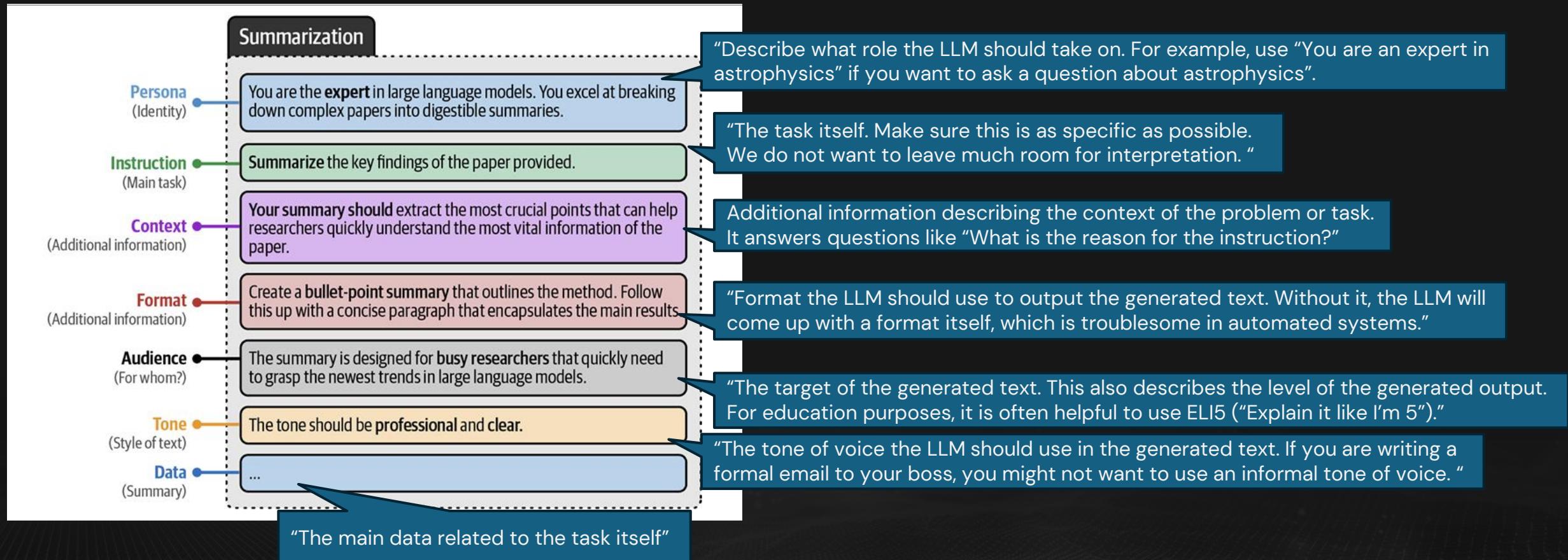


# Prompt Engineering Techniken



# Prompt Engineering

## Was ist ein Prompt?





# Prompt engineering Techniken

## Übersicht

- Zero-shot
- Few-shot
- Instruction
- Chain of Thought
- Generated Knowledge Prompting
- Prompt Chaining
- Reflexion



# Prompt engineering Techniken

## Zero shot

Classify the text into neutral,  
negative or positive.

Text: I think the vacation is okay.

Sentiment:



The sentiment of the text is  
**neutral**.

Zero shot bedeutet: Modell erhält keine Beispiele, was richtige/ erwartete Antworten wären



# Prompt engineering Techniken

Few shot

Who won the Stanley Cup in 2024?

Edmonton Oilers 1990 in 5 games  
against Boston bruins



The Florida Panthers won the 2024 Stanley Cup, marking their first championship in franchise history. They defeated the Edmonton Oilers in a thrilling seven-game series, with the decisive Game 7 ending in a 2-1 victory for the Panthers. Sam Reinhart scored the game-winning goal in the second period, and Florida's strong defensive performance held off Edmonton's comeback attempts

Few shot bedeutet: Modell erhält Beispiele, was richtige/ erwartete ähnliche Antworten wären



# Prompt engineering Techniken

## Instruction

User: Answer the question using the format shown in the context.

Who won the football World cup in 2006? Italy won the world cup in 2006.  
They defeated France.

Who won the world cup in 2014?



Germany won the World Cup in 2014. They defeated Argentina

Instruction bedeutet daß dem Modell gesagt, wird was zu tun ist



# Prompt engineering Techniken

## Chain-of-thought-prompting

I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with?

Let's think step by step.

Kombinieren Sie das mit Few-shot für noch bessere Ergebnisse



Let's break it down step by step:

1. You bought 10 apples.
  2. You gave 2 apples to the neighbor and 2 apples to the repairman, so you gave away 4 apples in total.  
 $10 \text{ apples} - 4 \text{ apples} = 6 \text{ apples remaining.}$
  3. Then you bought 5 more apples.  
 $6 \text{ apples} + 5 \text{ apples} = 11 \text{ apples.}$
  4. You ate 1 apple.  
 $11 \text{ apples} - 1 \text{ apple} = 10 \text{ apples remaining.}$
- So, you are left with **10 apples**.

Chain-of-Thought ermöglicht bessere Ergebnisse durch Zwischenschritte, die das Modell machen muß.



# Prompt engineering Techniken

## Generated Knowledge Prompting

Part of golf is trying to get a higher point total than others.  
Yes or No?



Yes

Input: Greece is larger than Mexico.  
Knowledge: Greece is approximately 131,957 sq km, while Mexico is approximately 1,964,375 sq km, making Mexico 1,389% larger than Greece.  
Input: Part of golf is trying to get a higher point total than others.  
Knowledge:



The statement "Part of golf is trying to get a higher point total than others" is **incorrect**. In golf, the goal is to achieve a **lower score** than your competitors. Golf is scored by counting the number of strokes a player takes to complete the course, and fewer strokes indicate a better performance. The player with the fewest strokes (or lowest score) at the end of the game wins. In contrast to many sports where a higher score is preferred (such as basketball or football), golf rewards players for achieving a **lower total score**. Therefore, the aim is not to get a higher point total, but rather to complete each hole in as few strokes as possible.

Generated Knowledge prompting ermöglicht bessere Ergebnisse, indem dem Prompt „Wissen“ mitgegeben wird.



# Prompt engineering Techniken

## Prompt chaining

Your task is to review an email and provide feedback.

Here is the email:

```
<email>  
{{EMAIL}}  
</email>
```

Dieser Block erlaubt Parameter mitzugeben

Give feedback on tone, clarity, and professionalism. Be concise



**Tone:** The tone is informal, which may be fine for a casual or close relationship with your boss. However, it could come across as too familiar or even unprofessional in a more formal context. A slightly more formal greeting and expression would be better in a workplace setting.

**Clarity:** The message is clear, but the repetition of "All fine with you?" makes it redundant. It's better to combine this into one, more concise question.

**Professionalism:** The email lacks proper grammar ("Havent" should be "Haven't") and a subject line. Adding a proper greeting, subject, and closing would improve professionalism.

### Suggestions for Improvement:

1. Use a more formal greeting (e.g., "Hello [Boss's Name]").
2. Avoid repetitive questions.
3. Use proper grammar and punctuation.
4. Add a subject line.
5. Close the email with a polite sign-off.

### Example Revision:

**Subject:** Checking In

Hello [Boss's Name],

I hope all is well with you! It's been a while since I last heard from you. How are things going?

Looking forward to hearing from you.

Best regards,

[Your Name]

Prompt Chaining ermöglicht das Erledigen komplexer Aufgaben durch Verfeinern in kleinere Aufgaben



# Prompt engineering Techniken

## Reflexion

your task is to write an email to a spouse saying sorry for forgetting her birthday



Here's an email you could write to your spouse to apologize for forgetting her birthday:.....

evaluate the mail regarding emotionality and brevity on a scale from 1 to 5.  
Give feedback how to improve the mail



### Evaluation of the Email:

#### Emotionality:

- Rating: 5/5

The email is highly emotional and conveys sincere regret. The language used expresses genuine remorse, takes full responsibility, and offers a heartfelt apology. ...

#### Brevity:

- Rating: 3/5

While the message is heartfelt, it is a bit long for an apology email. The key points (apology, responsibility, plan to make it up) ....

#### Suggestions for Improvement:

1. Trim unnecessary repetition: Phrases like "I feel terrible" and "I deeply regret" are strong but can be expressed more succinctly.
2. Refine the conclusion: The last sentence could be shortened. A simpler sign-off could maintain emotional sincerity while being more concise.

now rewrite the mail according to your feedback



Here's a rewritten version of the email based on the feedback: **Subject:** I'm So Sorry  
Dear [Spouse's Name],  
I am truly sorry for forgetting your birthday. There's no excuse for it, and I feel awful for not making your day as special as it should have been. You mean the world to me, and I deeply regret not showing that.

Reflexion ermöglicht das Verbessern der Ergebnisse des Prompts.



# Advanced Prompt Engineering Techniken



# Context anchor

## Einfaches Beispiel:

"You are a technical writer at Google DeepMind. Using the context from [document], write a blog post that explains [concept] to developers who understand ML basics but haven't worked with transformers."

## Zusätzlich: Bestimmen Empfänger/ Kunden (Audience):

### Beispiel:

"TARGET AUDIENCE PROFILE:

- Demographics: [age, role, industry]
- Current situation: [what they're dealing with]
- Pain points: [specific frustrations]
- Goals: [what they want to achieve]
- Language they use: [specific phrases/jargon]
- What they've already tried: [failed solutions]
- What makes them skeptical: [objections]"

## Zusätzlich: Bestimmen Ausgabeformat

### Beispiel 1:

"Output as JSON with keys: 'summary' (max 50 words), 'technical\_details' (bullet points), 'code\_example' (Python), 'next\_steps' (numbered list)."

### Beispiel 2:

OUTPUT FORMAT (follow exactly):

---

[SECTION 1 NAME]: [Beschreibung welcher Text reinkommt]

[SECTION 2 NAME]: [Beschreibung welcher Text reinkommt]

[SECTION 3 NAME]: [Beschreibung welcher Text reinkommt]

---

EXAMPLE OUTPUT:

NOW GENERATE FOR: [input]"

Ein Beispiel, so daß  
Model das Schema sieht

Rolle, Context und Empfänger geben Modell „Guidance“, was erwartet wird.



# Context anchor

## Detaillierung

### Template 1:

"You are a [specific role] with [X years] experience in [domain].  
Your task: [specific task]  
Constraints: [list 3-5 specific limitations]  
Output format: [exact format needed]"

### Template 2:

HARD CONSTRAINTS (cannot be violated):

- [constraint 1]
- [constraint 2]
- [constraint 3]

Ihre harte Bedingungen/  
KO-Kriterien

SOFT PREFERENCES (optimize for these):

- [preference 1]
- [preference 2]

Nice-to-have  
Bedingungen

TASK: [Aufgabe]

Confirm you understand all constraints before proceeding."

### Beispiel: Kurze Mail:

"Generate 3 variations. Each must:

- Be under 280 characters
- Include one technical term
- End with a question
- Avoid jargon like 'revolutionary' or 'game-changer'"

### Beispiel: Cold-call mail:

"Write a cold-call mail to prospective apartment buyers targeting

- rich people with middle age
- want a guaranteed investment
- Include: promise, proof, advantages of Schwabing  
Max 50 words."

Mehr Aufwand, aber ganz klare Guidance was vom Modell erwartet wird.



# Detaillierter Context anchor

## Mehr Beispiele

### Beispiel: Review technischer Artikel

"Rewrite this paragraph with

- a grade 5 clarity level
- 2 insights that a reviewer would respect
- a logical chain with each sentence adding new information
- a single takeaway in the final line"

### Beispiel: Programmieren:

"HARD CONSTRAINTS (cannot be violated):

- Must be written in Rust
- Cannot use any external dependencies
- Must compile on stable Rust 1.75+
- Maximum binary size: 5MB

SOFT PREFERENCES (optimize for these):

- Fast compilation time
- Minimal memory allocation

TASK: Write a CLI tool that parses 10GB CSV files and outputs JSON with schema validation"

### Beispiel: Programmieren:

"HARD CONSTRAINTS (cannot be violated):

- Must be written in Python
- Cannot use any external dependencies
- Must follow the typical data science use case structure
- Must work without any errors

SOFT PREFERENCES (optimize for these):

- Explain the code and the steps you took in Markdown cells

TASK: Solve this data science use case in tags <use case>



# Context anchor with boundaries

Definition Grenzen für korrektere Rückmeldung

Template:

"[CONTEXT]  
[IHNEN INPUT]

[FOCUS]

Only use information from CONTEXT to answer.

If the answer isn't in CONTEXT, say "Insufficient information in provided context."

[TASK]

[IHRE FRAGEN]

[CONSTRAINTS]

Cite specific sections when referencing CONTEXT

- Do not use general knowledge outside CONTEXT
- If multiple interpretations exist, list all"

Beispiel: Analyse eines längeren Dokuments

"[CONTEXT]

[paste your company's 50-page API documentation]

[FOCUS]

Only use information from CONTEXT to answer. If the answer isn't in CONTEXT, say "Insufficient information in provided context."

[TASK]

How do I implement rate limiting with retry logic for the /users endpoint?

[CONSTRAINTS]

- Cite specific sections when referencing CONTEXT
- Do not use general knowledge outside CONTEXT
- If multiple interpretations exist, list all"

Noch mehr Aufwand, aber zusätzlich klare Definition was nicht gewünscht wird. Kann Halluzinieren reduzieren



# Context anchor with boundaries

Definition Grenzen für korrektere Rückmeldung

Template:

"[CONTEXT]  
[IHNEN INPUT]

[FOCUS]

Only use information from CONTEXT to answer.

If the answer isn't in CONTEXT, say "Insufficient information in provided context."

[TASK]

[IHRE FRAGEN]

[CONSTRAINTS]

Cite specific sections when referencing CONTEXT

- Do not use general knowledge outside CONTEXT
- If multiple interpretations exist, list all"

Beispiel: Analyse eines längeren Dokuments

"[CONTEXT]

[paste your company's 50-page API documentation]

[FOCUS]

Only use information from CONTEXT to answer. If the answer isn't in CONTEXT, say "Insufficient information in provided context."

[TASK]

How do I implement rate limiting with retry logic for the /users endpoint?

[CONSTRAINTS]

- Cite specific sections when referencing CONTEXT
- Do not use general knowledge outside CONTEXT
- If multiple interpretations exist, list all"

Noch mehr Aufwand, aber zusätzlich klare Definition was nicht gewünscht wird. Kann Halluzinieren reduzieren



# Multi-perspective prompting/ role switching

Template:

"Analyze [topic/problem] from these perspectives:

[PERSPECTIVE 1: Technical Feasibility]

[specific lens]

[PERSPECTIVE 2: Business Impact]

[specific lens]

[PERSPECTIVE 3: User Experience]

[specific lens]

[PERSPECTIVE 4: Risk/Security]

[specific lens]

SYNTHESIS: Integrate all perspectives into a final recommendation with trade-offs clearly stated."

Einfaches Beispiel:

"What would be a good group of people to explore xyz?

What would they say?"<sup>2</sup>

Einfaches Beispiel:

"First, act as a skeptic and find flaws in this approach. Now, act as an advocate and defend it. Finally, synthesize both perspectives into a balanced recommendation."

Beispiel:

"Explain how to improve my marketing funnel from 3 angles:

- A behavioral psychologist
- A data scientist
- A CMO

Then merge all 3 into a unified action plan."

Beispiel:

"Analyze whether we should migrate from Postgres to DynamoDB from these perspectives:

[PERSPECTIVE 1: Technical Feasibility] Engineering complexity, timeline, data migration risks

[PERSPECTIVE 2: Business Impact] Cost implications, team velocity, vendor lock-in

[PERSPECTIVE 3: User Experience] Latency changes, feature implications, downtime requirements

[PERSPECTIVE 4: Risk/Security] Data consistency guarantees, backup procedures, compliance

SYNTHESIS: Integrate all perspectives into a final recommendation with trade-offs clearly stated."

Veröffentlichung<sup>1</sup> zeigte daß Rolle nicht sehr viel Einfluß auf Output hat.



# Zerlegung Aufgabe in einzelne Schritte

Beispiel:

prompt 1: "Break down all steps required to design a SaaS onboarding system."

prompt 2: "For each step, list the decisions that matter."

Prompt 3: "Now create the final onboarding system using that plan."

-



# Reversal prompts

Modell gibt falsche Antwort (auf Wunsch) und wird dann beauftragt, das zu ändern.

Beispiel:

"Give me the worst possible explanation of vector embeddings.  
Then rewrite it into the best possible explanation.  
Highlight exactly what changed and why."

Zeigt, was das Modell als gute und schlechte Antwort versteht.



# Verbalized sampling

Beispiel:

"Generate 5 different responses to the [IHRE ANFRAGE] below. For each response, include a numeric probability estimate indicating how likely the model considers that response.  
Make sure the sum of probabilities across all responses equals 1.0.  
[IHRE ANFRAGE]"

Post-training alignment des LLM führt zu Präferenz bestimmter Antworten (die der Labeller besser bewertete).  
Diese sind nur nicht immer die besten Antworten....



# Chain of verification

Template:

Task: [IHRE ANFRAGE]

Step 1: Provide your initial answer

Step 2: Generate 5 verification questions that would expose errors in your answer

Step 3: Answer each verification question

Step 4: Provide your final, corrected answer based on verification

“Debuggen” der Modellausgabe: was sind die Annahmen des Modells?



# Chain of verification: Erweiterung um few-shot (negative) Beispiele

Template:

I need you to [task]. Here are examples:

- GOOD Example 1: [example]
- GOOD Example 2: [example]

BAD Example 1: [example]

Why it's bad: [reason]

BAD Example 2: [example]

Why it's bad: [reason]

Now complete: [your actual task]

Beispiel:

I need you to write cold email subject lines. Here are examples:

- GOOD: "Quick question about your Q4 engineering roadmap"
- GOOD: "Saw your post on distributed systems—thoughts on this?"

BAD: "URGENT: Limited Time Offer Inside!!!"

Why it's bad: Spam trigger words, fake urgency

BAD: "You won't believe what we built..."

Why it's bad: Clickbait, no context

Now write 5 subject lines for: SaaS tool that reduces cloud costs by 40%

Es ist immer einfacher, zu sagen, was schlecht ist....



# Confidence-weighted prompting

Template:

"Answer this question: [question]

For your answer, provide:

1. Your primary answer
2. Confidence level (0-100%)
3. Key assumptions you're making
4. What would change your answer
5. Alternative answer if you're <80% confident"

Beispiel:

"Before answering, list:

- What assumptions you're making
- What information you'd need to be more certain
- What could make this answer wrong"

Beispiel:

Answer this question: Will LLM-based agents replace Software developer in systems programming by 2030? For your answer, provide:

1. Your primary answer
2. Confidence level (0-100%)
3. Key assumptions you're making
4. What would change your answer
5. Alternative answer if you're <80% confident"

Modell soll sagen, wie "sicher" es sich der Antworten ist und auch gleich Alternativen vorschlagen



# Iterative Refinement Loop

Template:

Prompt 1: "create a [draft/outline/initial version] of [task]"

Prompt 2: "Review the above output.  
Identify 3 weaknesses or gaps."

Prompt 3: "Rewrite the output addressing all identified  
weaknesses."

Prompt 4: "Final review: Is this production-ready?  
If not, what's missing?"

Beispiel:

First prompt: "Generate initial analysis of development of  
prices for apartments in Munich"

Second prompt: "Critique the above for accuracy and bias"

Third prompt: "Revise based on critique, maintaining  
technical precision"

Beispiel:

Prompt 1: "Create a draft sales email for reaching out to engineering VPs at  
Series B startups about our CI/CD optimization tool"

Prompt 2: "Review the above output. Identify 3 weaknesses or gaps."

Prompt 3: "Rewrite the output addressing all identified weaknesses."

Prompt 4: "Final review: Is this production-ready? If not, what's missing?"<sup>1</sup>

Erweiterung des Reflexion prompts. Verbesserung Ergebnisse durch Feedback Loop



# Iterative Refinement Loop weiter gedacht: Modell überarbeitet Prompt und Ergebnisse

Template:

"I need to accomplish: [high-level goal]

Your task:

1. Analyze what would make the PERFECT prompt for this goal
2. Consider: specificity, context, constraints, output format, examples needed
3. Write that perfect prompt
4. Then execute it and provide the output

[GOAL]: [your actual objective]"

Beispiel:

"I need to accomplish: write a cold call mail to potential apartment buyers in Munich but it shouldn't have any fluff or marketing words. It should address serious people with lots of money. They don't have much time so the mail should be short and actionable. If interested they shall call the number 089123456 or mail to XX@immobilien-munich.de. The mail should be in German though.

Your task:

1. Analyze what would make the PERFECT prompt for this goal
2. Consider: specificity, context, constraints, output format, examples needed
3. Write that perfect prompt
4. Then execute it and provide the output

[GOAL]: cold call mail to achieve lots of traction"

Erweiterung des Reflexion prompts. Verbesserung Ergebnisse durch Feedback Loop



# Definition strukturierte Vorgehensweise für komplexe(re) Themen

Template 1:

Before answering, work through this exact process:

STEP 1 - UNDERSTAND: Restate the problem in your own words. What am I actually being asked?

STEP 2 - DECOMPOSE: Break this into sub-problems. What needs to be solved first?

STEP 3 - ANALYZE: For each sub-problem, what are the possible approaches?

What are the tradeoffs?

STEP 4 - SYNTHESIZE: Connect the pieces. How do the sub-solutions combine?

STEP 5 - VALIDATE: Check your reasoning. What could be wrong?

What assumptions am I making?

STEP 6 - ANSWER: Now provide the final answer with confidence levels.

PROBLEM: [IHRE ANFRAGE]

Beispiel:

"Before answering, complete these steps:

[UNDERSTAND]

- Restate the problem in your own words
- Identify what's actually being asked

[ANALYZE]

- Break down into sub-components
- Note any assumptions or constraints

[STRATEGIZE]

- Outline 2-3 potential approaches
- Evaluate trade-offs

[EXECUTE]

- Provide your final answer
- Explain your reasoning

Question: Should I buy an apartment in Schwabing or one in Dachau as a better investment"

Erweiterung des Reflexion prompts. Verbesserung Ergebnisse durch Feedback Loop



# LLM Probleme



# Systematische Probleme von LLM

## Übersicht

- Einsatz "kontaminierter" oder inkorrekt er Daten während des Trainings.
- (leichte) Änderungen der Eingabe führen zu starken Änderungen im Output<sup>1</sup>
- Halluzinationen....und nicht vermeidbar<sup>2,3</sup>
- Security: jailbreak (MCP!) <sup>4,5</sup>
- Abnehmende Performance in längeren Konversationen Mensch-LLM (multi-turn<sup>6</sup>)

### Quellen:

- 1 Jang et al., "Can large language models truly understand prompts? A case study with negated prompts", 2023.
- 2 Xu et al., "Hallucination is inevitable: An innate limitation of large language models", 2024.
- 3 Banerjee et al. "LLMs Will Always Hallucinate, and We Need to Live With This", 2024.
- 4 Zou et al., "Universal and transferable adversarial attacks on aligned language models", 2023.
- 5 Beurer-Kellner et al. "Design Patterns for Securing LLM Agents against Prompt Injections", 2025
- 5 Laban et al., "LLMs Get Lost In Multi-Turn Conversation", 2025.



# Systematische LLM-Probleme

## Inkorrekte und kontaminierte Daten

### Kontaminierte Daten

Gegeben die riesige Datenmenge für das Training, wie hoch ist die Chance, daß in Trainings- und Testdaten die Gleichen/ähnlichen Daten sind?

### Inkorrekte/ problematische Daten

LLM wird anhand von Web-Daten trainiert.  
Aber: es gibt so viel problematische Daten

- Lügen (, damned lies and statistics)
- Verschwörungstheorien
- Rassismus
- ....

Diese Daten haben negative Einfluß auf die Qualität der Ausgaben des LLM  
(inclusive Halluzinieren)



# Systematische LLM-Probleme

## Halluzinieren

Halluzinieren: “the generation of texts or responses that exhibit grammatical correctness, fluency, and authenticity, but deviate from the provided source inputs (faithfulness) or do not align with factual accuracy (factualness)”<sup>1</sup>

**Forbes**

**BREAKING | BUSINESS**

## Lawyer Used ChatGPT In Court—And Cited Fake Cases. A Judge Is Considering Sanctions

By Molly Bohannon, Former Staff. Molly Bohannon has been a Forbes news reporter since 2023.

Published Jun 08, 2023, 02:06pm EDT, Updated Jun 08, 2023, 03:42pm EDT

[Share](#) [Save](#)

This article is more than 2 years old.

**TOPLINE**

The lawyer for a man suing an airline in a routine personal injury suit used ChatGPT to prepare a filing, but the artificial intelligence bot delivered fake cases that the attorney then presented to the court, prompting a judge to weigh sanctions as the legal community grapples with one of the first cases of AI “hallucinations” making it to court.

**BBC**

Home News Sport Business Innovation Culture Arts Travel Earth Audio Video Live

## Airline held liable for its chatbot giving passenger bad advice - what this means for travellers

23 February 2024

Maria Yagoda  
Features correspondent

**The Washington Post**  
Democracy Dies in Darkness

Tech Artificial Intelligence Help Desk Internet Culture Space Tech Policy

Innovations

## ChatGPT invented a sexual harassment scandal and named a real law prof as the accused

The AI chatbot can misrepresent key facts with great flourish, even citing a fake Washington Post article as evidence

April 5, 2023 More than 2 years ago

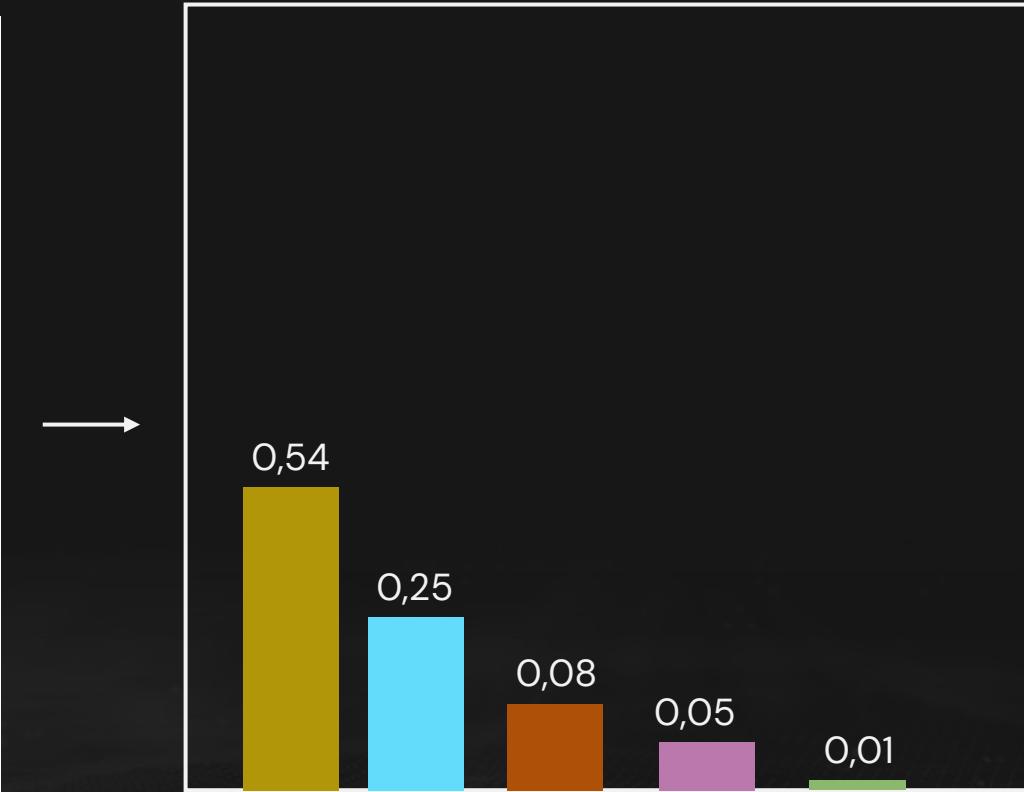
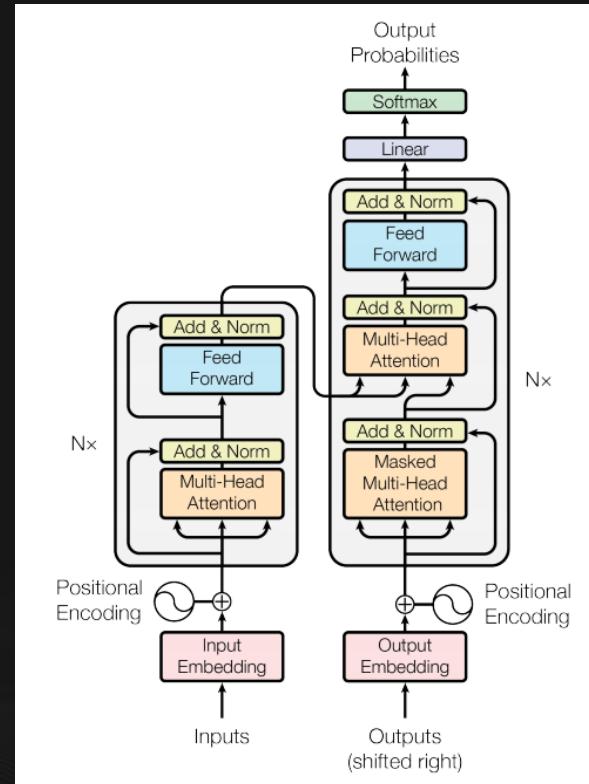
9 min Summary 390



# Systematische LLM-Probleme

Wie entsteht Halluzinieren? (Beispiel)

Yesterday on  
Mallorca,  
I went to see \_\_



beach

Megapark

harbor

Oberbayern

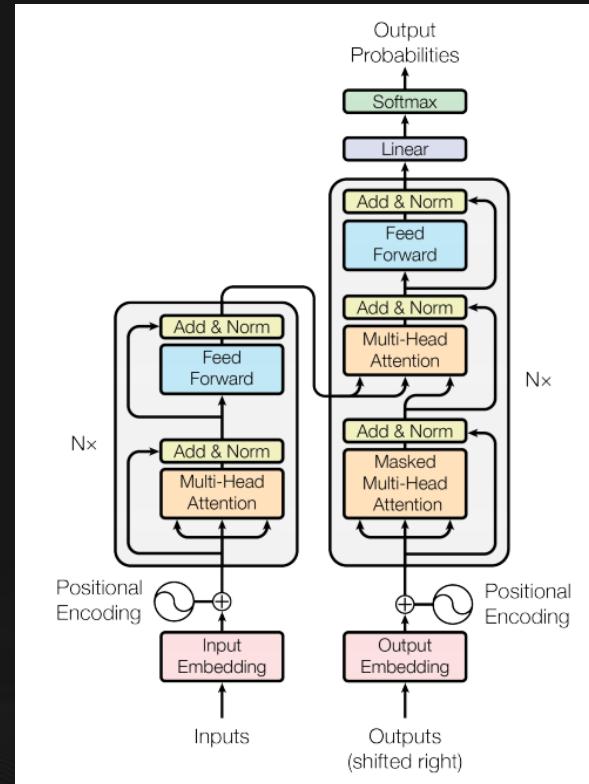
Munich



# Systematische LLM-Probleme

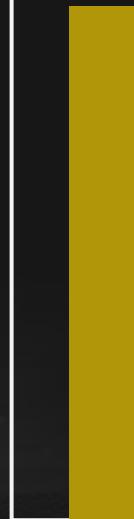
Wie entsteht Halluzinieren? (Beispiel)

Yesterday on  
Mallorca,  
I went to see \_\_



LLM hat Parameter für Tunen Output:

- Temperature < 1, reduziert Zufälligkeit
- Temperature = 0, LLM-Output wird vorhersagbar. Dies wird bspw. für Use Cases mit Mathematik eingesetzt.



beach

Megapark

harbor

Oberbayern

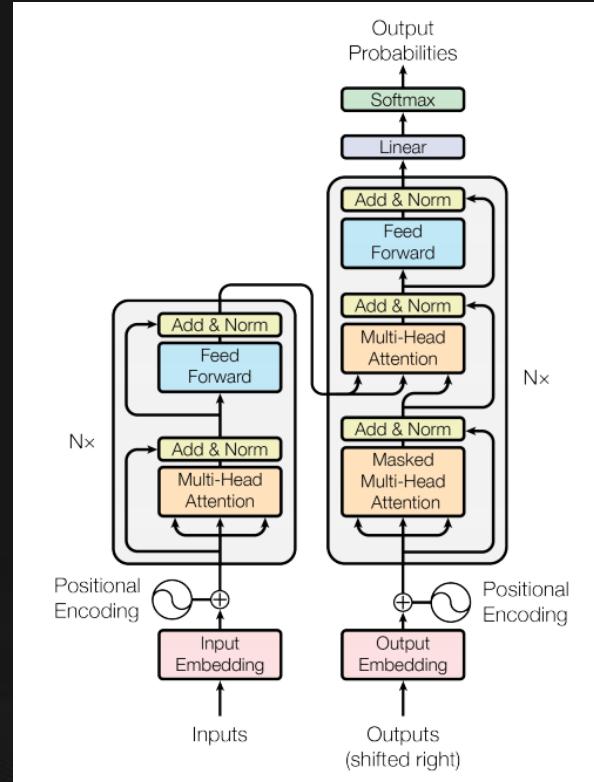
Munich



# Systematische LLM-Probleme

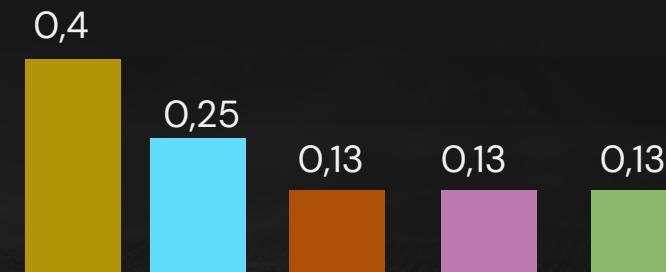
Wie entsteht Halluzinieren? (Beispiel)

Yesterday on  
Mallorca,  
I went to see \_\_



LLM hat Parameter für Tunen Output:

- Temperature  $> 1$  für mehr "Kreativität". Die Wahrscheinlichkeitsverteilung wird "angeglichen".
- Dadurch erhalten Tokens mit einer geringen Wahrscheinlichkeit viel höhere Werte, fast so hoch wie korrekte Tokens..
- Damit werden auch eigentlich nicht relevante Tokens berücksichtigt.



beach

Megapark

harbor

Oberbayern

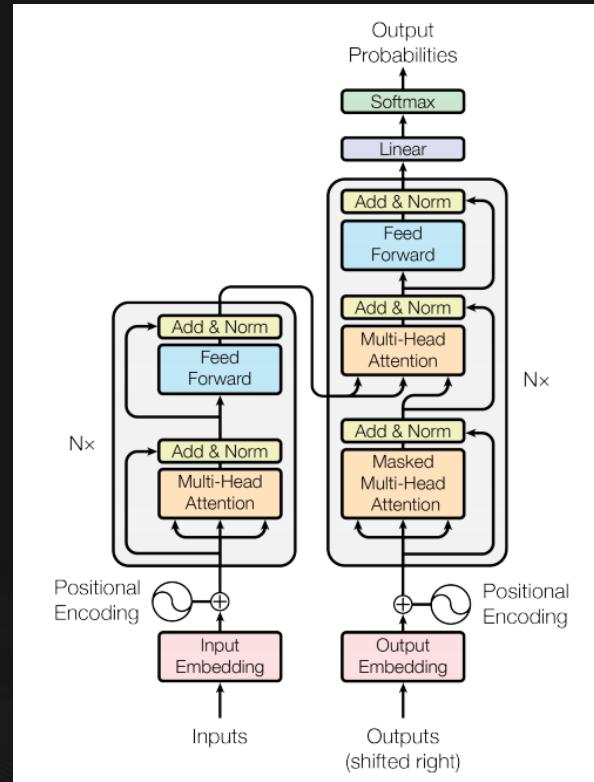
Munich



# Systematische LLM-Probleme

Wie entsteht Halluzinieren? (Beispiel)

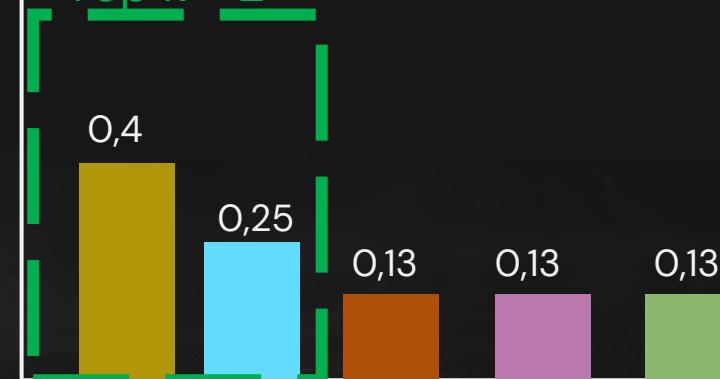
Yesterday on  
Mallorca,  
I went to see \_\_



LLM hat Parameter für Tunen Output:

- Top k wählt die k-token mit höchster Wahrscheinlichkeit.
- Bei hohem k können somit auch Tokens berücksichtigt werden, die eigentlich nicht passen.

Top k = 2



beach

Megapark

harbor

Oberbayern

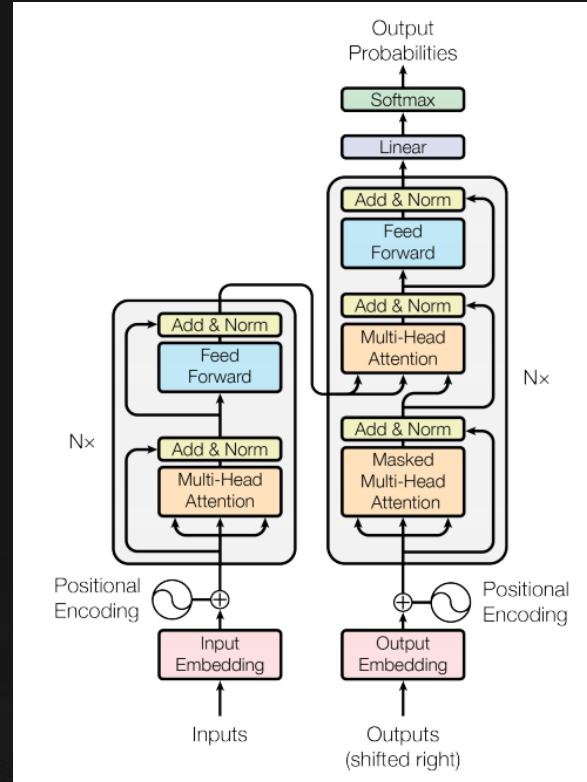
Munich



# Systematische LLM-Probleme

Wie entsteht Halluzinieren? (Beispiel)

Yesterday on  
Mallorca,  
I went to see \_\_



LLM hat Parameter für Tunen Output:

- Top  $p$  wählt genau so viele Tokens um eine höhere Gesamtwahrscheinlichkeit als  $p$  zu erhalten.
- Bei hohem  $p$  werden auch Tokens berücksichtigt, die nicht passen.



beach

Megapark

harbor

Oberbayern

Munich



# Ausblick



# Ausblick LLM

- Gu & Dao (2023) "Mamba: Linear-time sequence modeling with selective state spaces", / Beck et al. (2024), "xLSTM: Extended Long Short-Term Memory".  
Wieso wichtig: Optimiert Laufzeit von quadratischer zu linear Komplexität  
Für Eingabe von 100 Zeichen also 100 Operationen statt  $100 * 100 = 10\,000$
- Small-language models für embedded devices wie Handys,  
e.g. Gunter et al. (2024), "Apple Intelligence Foundation Language Models")
- Mixture of Experts:  
Wieso wichtig: Nutzen von kleineren, fokussierten Modellen statt 1 großem Modell
- Reasoning agents: Guo et al, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning"  
Wieso wichtig:
  - Viele Optimierungen: sehr gutes Modell trotz wenig Compute Power (US-Embargo!) und Kosten (~6 M \$ Kosten)
  - Source code verfügbar
  - Sehr detailliertes, sehr gut geschriebenes Paper
  - Einsatz von Reinforcement Learning für automatisierte Modellverbesserung (ohne Menschen)



# Generative AI für Texte

## Zusammenfassung: Vorteile und Herausforderungen

### Vorteile

- Ergebnisse sind oft richtig gut
- Unterstützt Menschen bei vielfältigen, repetitiven, aber auch kreativen Aufgaben
- Berücksichtigt Kontext von Anfragen
- Kontinuierliche Weiterentwicklung Modelle und starker Wettbewerb (Claude, Jurassic, ...)
- Starker Wettbewerb, gerade auch durch Open Source Modelle (bspw. Llama, Falcon, teilweise Mistral )

### Nachteile

- Manchmal sind Ergebnisse komplett falsch, weil Modell kein symbolisches Verständnis hat („Hallucination“)
- Sensitivität: ähnliche Inputwörter können zu sehr unterschiedlichen Ergebnissen führen.
- Erklärbarkeit/ Modell (falls nicht open source)
- Weiterhin Wissen notwendig, um sicherzustellen, daß der generierte Text sinnvoll ist.
- Abhängigkeit Trainingsdaten: nicht kuratiert und deshalb ggf. problematisch (biased, nicht mehr gültig, ....)
- Sehr hoher Ressourcenbedarf
- Ersetzen von Menschen?



# Literatur und weitere Quellen

## LSTM:

- <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Hochreiter, Schmidhuber: Long short-term memory, 1997 (meistzitierte AI-Paper!).

## Attention:

- <https://distill.pub/2016/augmented-rnns/>

## Transformers:

- <https://e2eml.school/transformers.html>
- <https://nlp.seas.harvard.edu/annotated-transformer/>
- <https://www.tensorflow.org/text/tutorials/transformer>
- <https://jalammar.github.io/illustrated-transformer/>
- <https://levelup.gitconnected.com/understanding-transformers-from-start-to-end-a-step-by-step-math-example-16d4e64e6eb1>

## Large Language Models:

- <https://mark-riedl.medium.com/a-very-gentle-introduction-to-large-language-models-without-the-hype-5f67941fa59e>

## GPT:

- Karpathy: Entwicklung GPT2-Clone in 600 Zeilen Python: <https://github.com/karpathy/nanoGPT> und <https://www.youtube.com/watch?v=kCc8FmEb1nY>
- <https://www.youtube.com/watch?v=kCc8FmEb1nY>
- Jay Mody: <https://jaykmody.com/blog/gpt-from-scratch/#putting-it-all-together> und <https://github.com/jaymody/picoGPT/blob/main/gpt2.py>
- <https://bootcamp.uxdesign.cc/how-chatgpt-really-works-explained-for-non-technical-people-71efb078a5c9>
- <https://medium.com/@fareedkhandev/create-gpt-from-scratch-using-python-part-1-bd89ccf6206a>
- Visualisierung verschiedener Modelle unter: [Link](#)
- Llama: Source Code unter [Link](#), Tutorials unter [Link](#), [Link](#)

## Prompt:

- <https://docs.anthropic.com/en/prompt-library/library>
- <https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/overview>

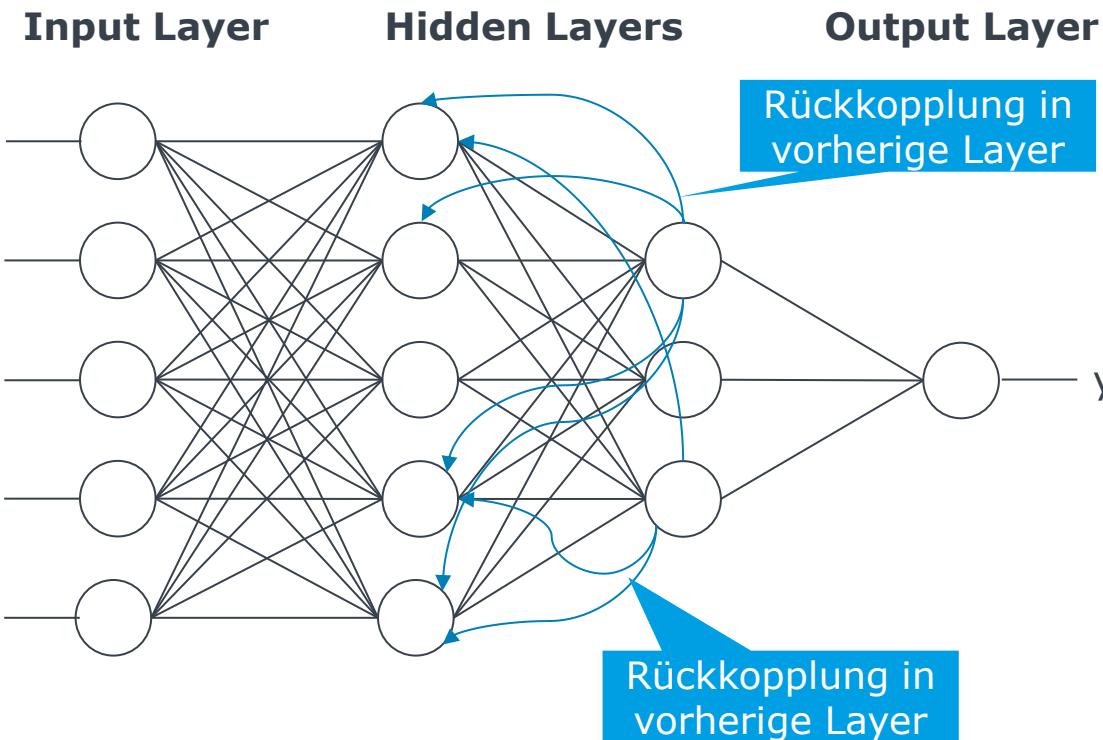


# Backup

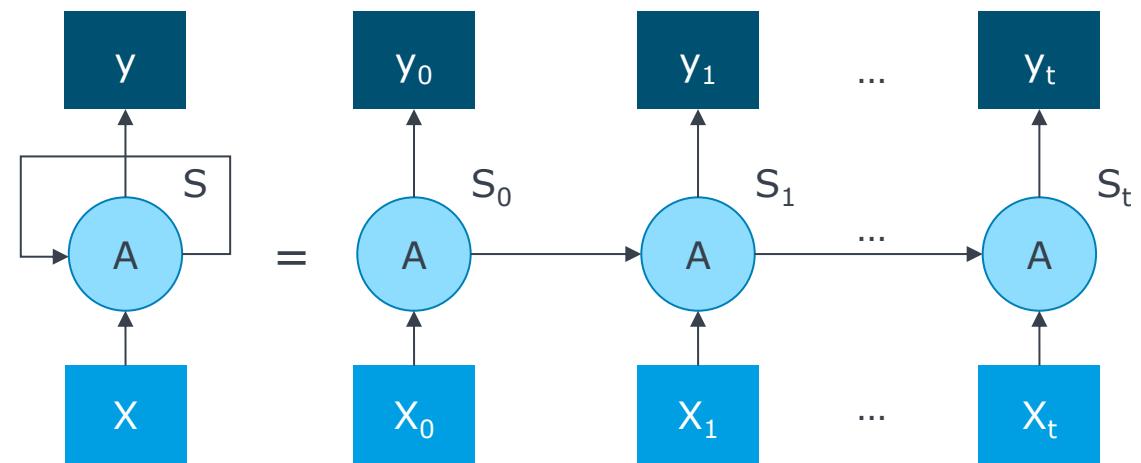


# DETAILLIERUNG REKURRENTE NEURONALE NETZE

# STRUKTUR REKURRENTE NEURONALE NETZE.



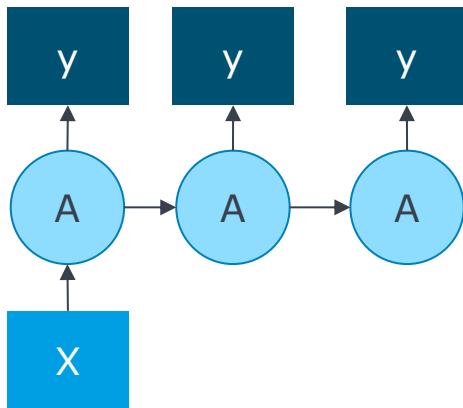
**Detaillierung Struktur RNN über Zeitschritte  $t$**   
 („Querschnittsbild“)



- Eingaben  $X_i$ : Sequenz wird auf Inputs aufgeteilt.
- Hidden State  $S_i$ : „Kurzzeitgedächtnis“. Abhängig vom Input  $x_i$  sowie vorherigen Zustände  $S_{i-1}$ . Überträgt auch (gewichtet) Informationen zum nächsten Schritt.
- A: Neurales Netzwerk inklusive Rückkopplung Ergebnisse.
- Ausgaben  $y_i$ : Ausgaben/ Prädiktion des Netzwerks.

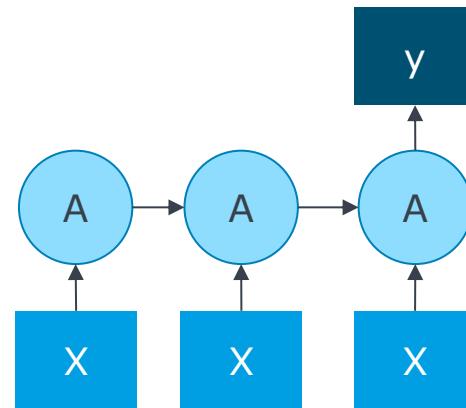
## (GROB-)STRUKTUR RNN ABHÄNGIG VOM EINSATZGEBIET.

**Ein Input, viele Outputs**



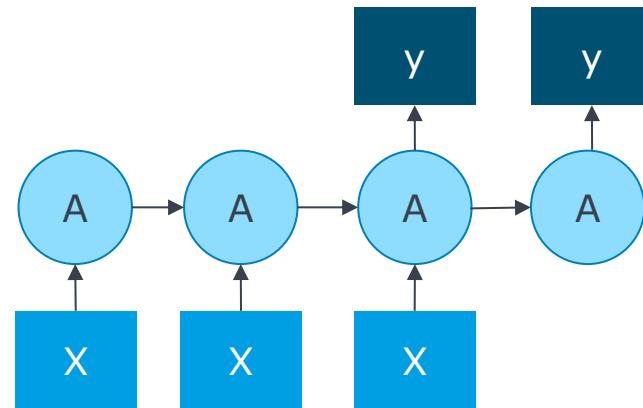
**Sequence Output:**  
 Bspw. automatisierte  
**Bilderkennung und -beschreibung**  
 (1 Bild mit n Wörtern beschreiben)

**Viele Inputs, ein Output**



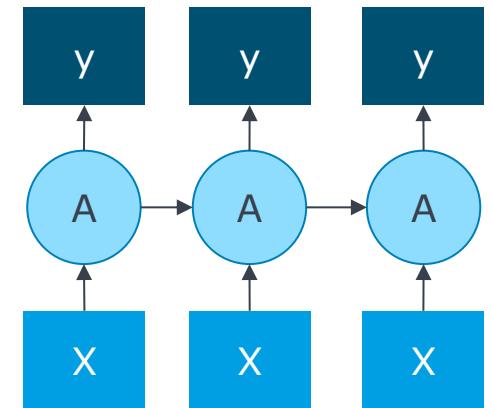
**Sequence Input:**  
 Bspw. automatisierte  
**Sentiment Analysis/ Gefühlsanalyse**  
 (1 Satz wird analysiert ob positive/ negative Aussagen)

**Viele Inputs, viele Outputs**



**Sequence input and output:**  
 Bspw. Automatisiertes **Übersetzen** **Texte** (Deutsch – Englisch), aber Input und Output können verschiedene Längen haben.

**Viele Inputs, viele Outputs**



**Gleich lange Sequence input and output:**  
 Bspw. Videoklassifikation, jeder Frame wird gelabelt

- Frage: welche Struktur wird eingesetzt für**
- Bewerten ob positives/ negatives Kundenfeedback
  - Automatisiertes Übersetzen eines Satzes
  - Labeln Videoframes (je Frame)
  - Beschreiben eines Bildes mit Wörtern



## BEKANNTE MACHINE LEARNING VERFAHREN FÜR TEXTE

# ÜBERSICHT BEKANNTE MACHINE LEARNING VERFAHREN.

## - LSTM (1997)<sup>1</sup>:

- Rekurrentes Neuronales Netzwerk, entwickelt von Sepp Hochreiter und Jürgen Schmidhuber an der TU München.
- aktuell (noch?) eines der häufigst eingesetzte Verfahren für sequentielle Daten, bspw. Spracherkennung in Handys ([Link](#)).
- Kann viel längere Zeitsequenzen verarbeiten als normale RNN (keine Probleme mit Vanishing Gradient<sup>2</sup>)
- Ressourcenaufwendig: hoher Speicherbedarf und (sehr) aufwendige Trainingszeit.
- Und neigt zum Overfitting (das man durch Tunen Hyperparameter und Struktur LSTM aber reduzieren kann).

Wiederholungsfrage: was ist Overfitting?

## - Transformer (2017)<sup>3</sup>:

- Entwickelt von Google, basierend auf **Attention<sup>4</sup>-Verfahren** aus 2015 (basierend auf Schmidhuber 1992).
- Kein RNN, sondern Kombination verschiedener sequentieller Verfahren ohne Rückkopplung (inkl. Transfer Learning).
- Schnelleres Training als LSTM aufgrund Aufteilen Lernen auf mehrere Rechner (Parallelisieren).
- Hat LSTM in vielen Gebieten abgelöst, bspw. für maschinelles Übersetzen, aufgrund höherer Flexibilität.
- Einsatz analog Transfer Learning für State-of-the-Art Ansätze wie BERT (Bidirectional Encoder Representations from Transformers)<sup>5</sup> und GPT-3 (Generative Pre-trained Transformer 3)<sup>6</sup>.

Quellen: 1 Hochreiter, Schmidhuber: "Long Short-Term Memory ", 1997. [Link](#)

2 Vanishing Gradient: beim Trainieren von Netzen mit vielen Schichten und Nutzen Exponentialfunktion als Aktivierungsfunktion kann bei Backpropagation Fall entstehen, daß Gradienten sehr, sehr klein werden und damit die Gewichte und Bias des Modells, v.a. der ersten Schichten, nicht mehr geändert/ trainiert werden. Vgl. Hochreiter: „Untersuchungen zu dynamischen neuronalen Netzen“, 1991.

3 Vaswani et al.: "Attention is all you need ", 2017. [Link](#)

4 Bahdanau et al.: "Neural Machine Translation by Jointly Learning to Align and Translate", 2015. [Link](#). Basierend auf Schmidhuber, "Learning to control fast-weight memories.", 1992.

5 Devlin et al.: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2019. [Link](#)

6 Brown et al.: "Language Models are Few-Shot Learners", 2020

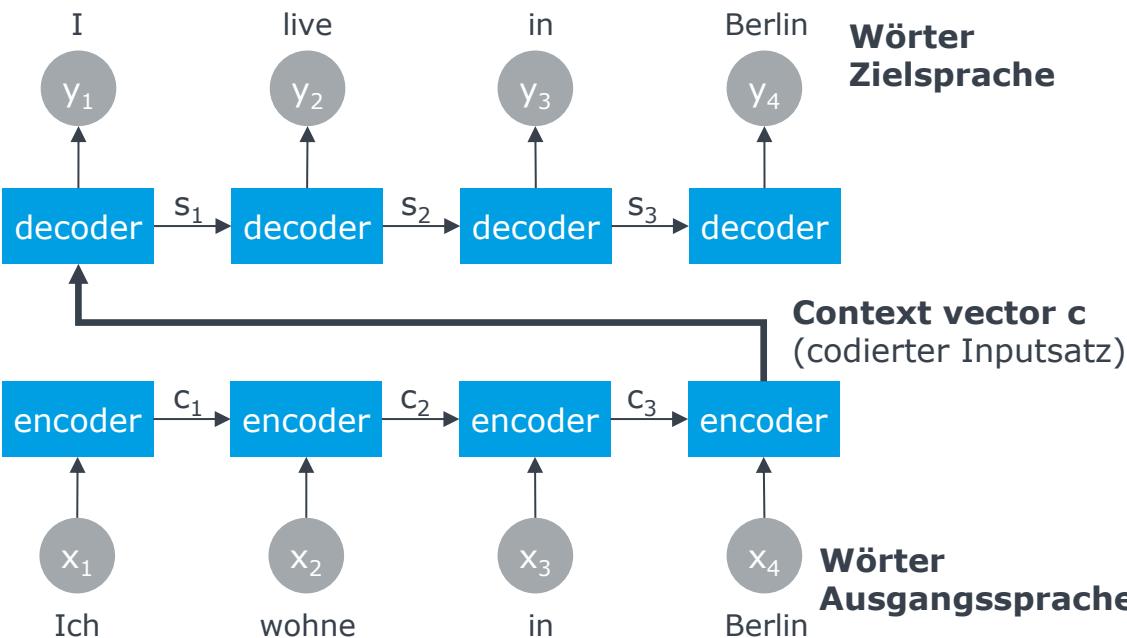
## DETAILLIERUNG BEISPIELHAFTE NLP-TECHNIKEN:

- **Sentence breaking:** Aufspalten Text in einzelne Sätze, bspw. anhand Punkt oder Komma.
  - **Lemmatization:** Vereinheitlichen ähnlicher Wörter/ Reduktion Varianz mit Wörterbuch. {Eats, eating, ate, eaten} → eat {ride, taking a ride} → ride
  - **Stemming:** Vereinheitlichen eines Wortes/ Reduktion Varianten durch Abschneiden Endungen. {played, plays, playing, ...} → play
  - **Stop Word Removal:** Entfernen Füllwörter (Pronomen, Artikel, Präpositionen, Fragewörter,....) [i, my, her, hers, has, had, hadnt, no, nor, not, only, doesnt, ...]
  - **Text/Word Embedding:** Umwandeln einzelner Buchstaben/ ganzer Wörter in eindeutige Zahlen. (Rechner können ja nur mit Zahlen arbeiten....)
- |       |           |
|-------|-----------|
| 1     | → <start> |
| Bus   | → 5       |
| kommt | → 144     |
| heute | → 114     |
| 2     | → <end>   |

NLP ermöglicht das effiziente Verarbeiten von Texten für Machine Learning

## (VEREINFACHTES) ATTENTION – MOTIVATION.

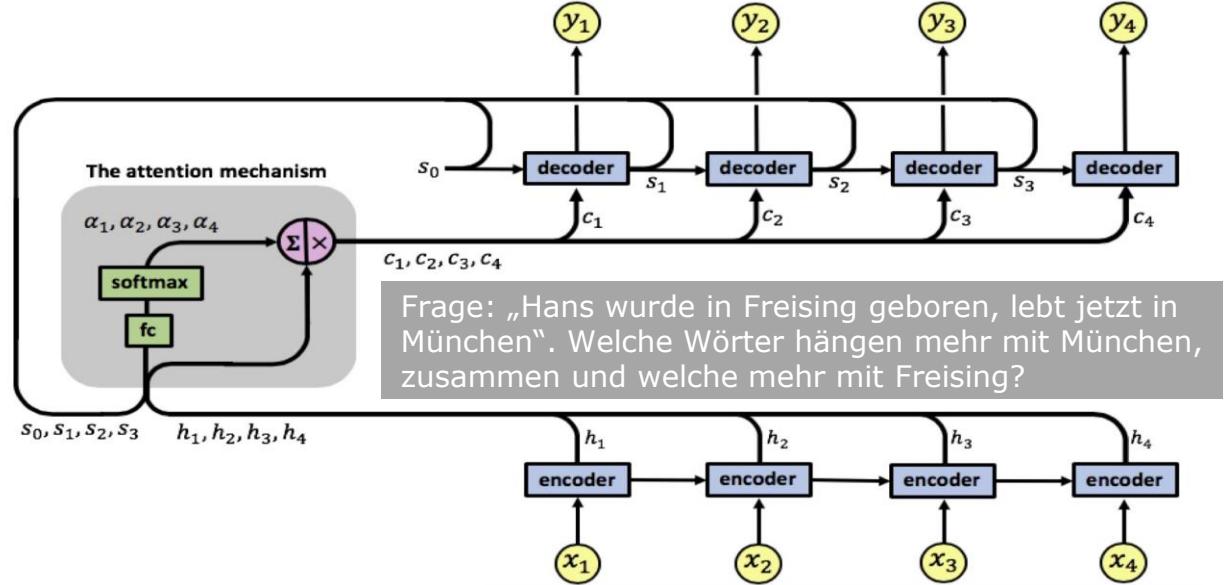
### Übersetzen Ausgangs- in Zielsprache bei RNN



### Herausforderungen/ Probleme bisheriger Verfahren:

- letztes Wort Ausgangssprache ist Input erstes Wort für Zielsprache; dabei sind oft erste Wörter ähnlich.
- „Normales“ RNN: Länge Decoder (Satz einer Zielsprache) festgelegt, was zu Problemen beim Lernen von langen Sätzen führt (umgangssprachlich: Modell „vergisst“ Kontext).
- RNN/ LSTM: sehr hoher Rechenaufwand & nicht parallelisierbar.

### Optimierung durch Attention Mechanismus

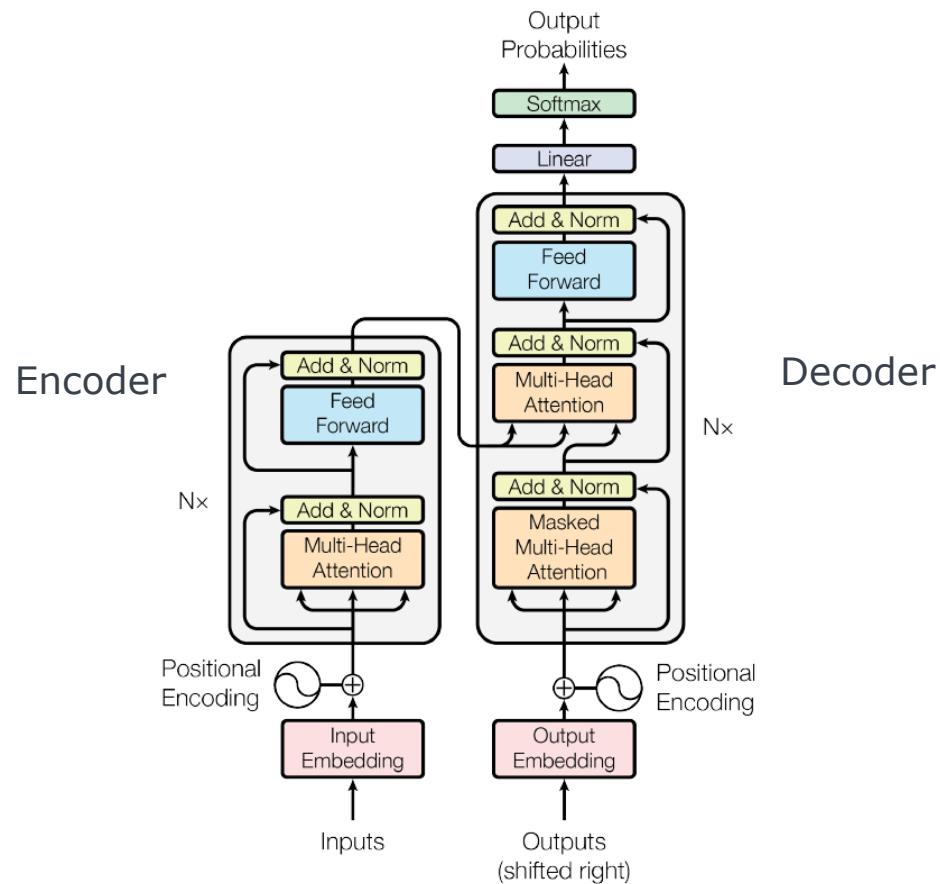


### Attention löst die erwähnten Herausforderungen:

- Attention ermöglicht dem Decoder, bestimmte Teile des Inputs stärker zu gewichten bei der Vorhersage des Outputs (hier: über  $c_1 - c_4$ ). Lernen dieser Attention-Gewichte ( $a_1 - a_4$ ) per neuronalem Netz.
- Das Problem der Satzlänge wird durch individuelle Gewichte für jeden Output gelöst. Von dieser unterschiedlichen Fokussierung stammt der Name Attention.

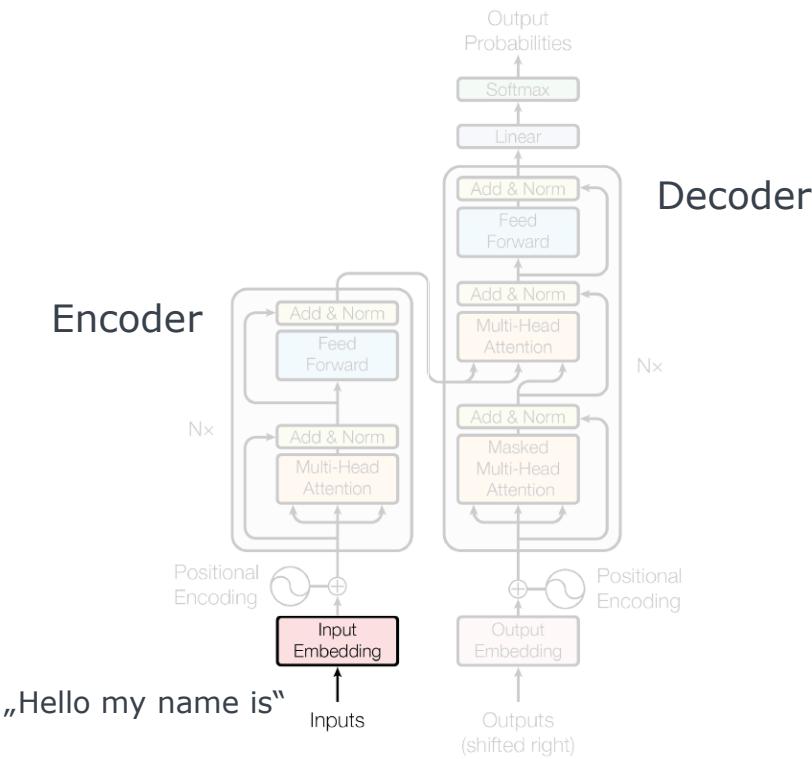
# DETAILLIERUNG TRANSFORMER

# ÜBERSICHT TRANSFORMER.



**Aufgabe Encoder:** speichere Eingabe in sequentieller Form inkl. Attention (welche Input-Wörter hängen mit welchen zusammen?)  
**Aufgabe Decoder:** nutze diese Form für Fokus auf passende Input-Wörter und erstelle Wahrscheinlichkeitsverteilung über gesamtes Vokabular für möglichst genaue Vorhersage des nächsten Ausgabeworts/ nächsten Ausgabewörter.

# TRANSFORMER IM DETAIL. ENCODER – INPUT EMBEDDING.

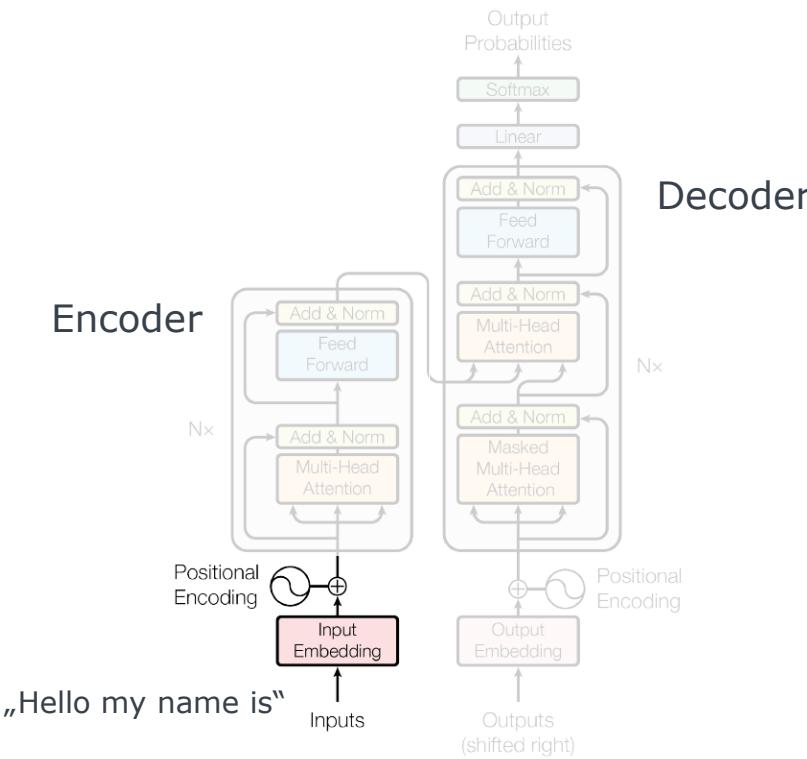


## Input Embedding:

- Eingabetext wird aufgeteilt in einzelne Wörter („Tokens“). Kann auf mehreren Rechnern parallel erfolgen.  
„Hello my name is“ → [Hello] [my] [name] [is]
- Hinzufügen Tokens zu einem Vokabular mit eindeutiger Nummer.  
[Hello] [my] [name] [is] -> [3 721 68 1337]
- Jeder Token-Eintrag hat eine Verbindung zu dem (bisher) gelernten Modell und dessen n-dimensionalen Vektor. Zusätzlich werden ähnliche Wörter gruppiert.  
[3 721 68 1337] -> [ [0.123, -5.234, ...], [...], [...], [...] ]

Computer verstehen keine Wörter, sondern Zahlen. Deshalb bauen wir eine Art Wörterbuch mit eindeutiger Nummer je Wort. Um die Qualität des Wörterbuchs zu verbessern, gruppieren wir von der Bedeutung ähnliche Wörter.

# TRANSFORMER IM DETAIL. ENCODER – POSITIONAL ENCODING

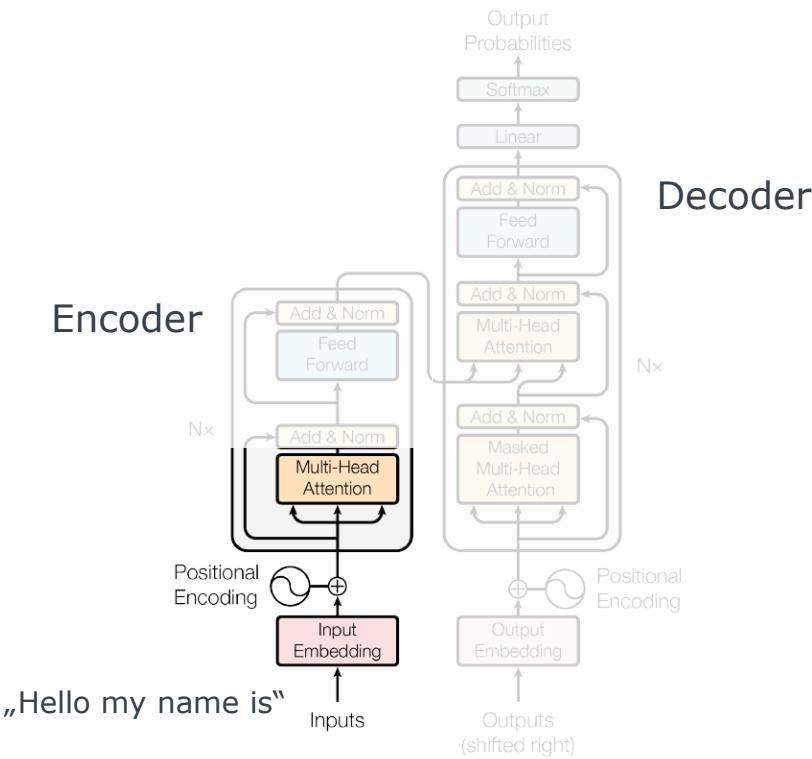


## Positional Encoding:

- Speichere Position, die die einzelnen Wörter im Input-Satz hatten.
- Addiere die Position zu dem Embedding aus dem vorigen Schritt.

Durch Positional Encoding speichern wir die Reihenfolge der Wörter und können so bspw. „A liebt B“ von „B liebt A“ unterscheiden.

# TRANSFORMER IM DETAIL. ENCODER – MULTI-HEAD ATTENTION.



## Self-headed attention:

- Modell soll die Verbindungen zwischen den einzelnen Input-Wörtern untereinander lernen (bspw. „wie“ mit „geht“ und „dir“).

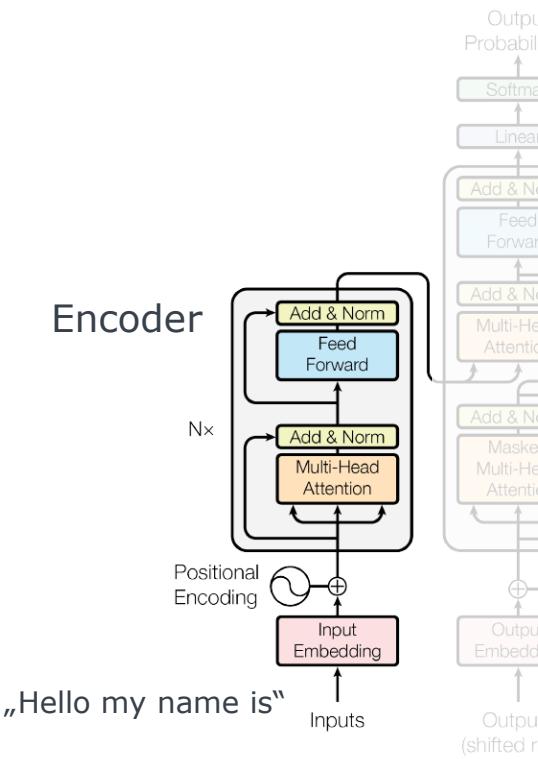
	Hello	my	name	is
Hello	0,75	0,15	0,05	0,05
my	0,1	0,6	0,2	0,1
name	0,05	0,2	0,65	0,1
is	0,2	0,1	0,1	0,6

## Multi-headed attention:

- vermeidet, daß self-headed attention den eigenen Input stärker gewichtet als die restlichen Wörter.
- Gewichten der Ergebnisse einzelner self-headed Attentions und Kombination zu finalem Vektor je Token. Dieser Vektor sagt, wie stark das Token auf die anderen Wörter schauen sollte.

Multi-headed Attention erlaubt das Finden und Lernen von Korrelationen zwischen den einzelnen Bestandteilen eines Satzes.

# TRANSFORMER IM DETAIL. ENCODER – RESIDUAL, NORMALISIERUNG UND FEED FORWARD



## Residual:

- Der vorherige Ausgabevektor wird zum Positions-Encoding addiert. Dies ermöglicht ein starkes Verbessern der Trainierbarkeit.

## Normalisierung:

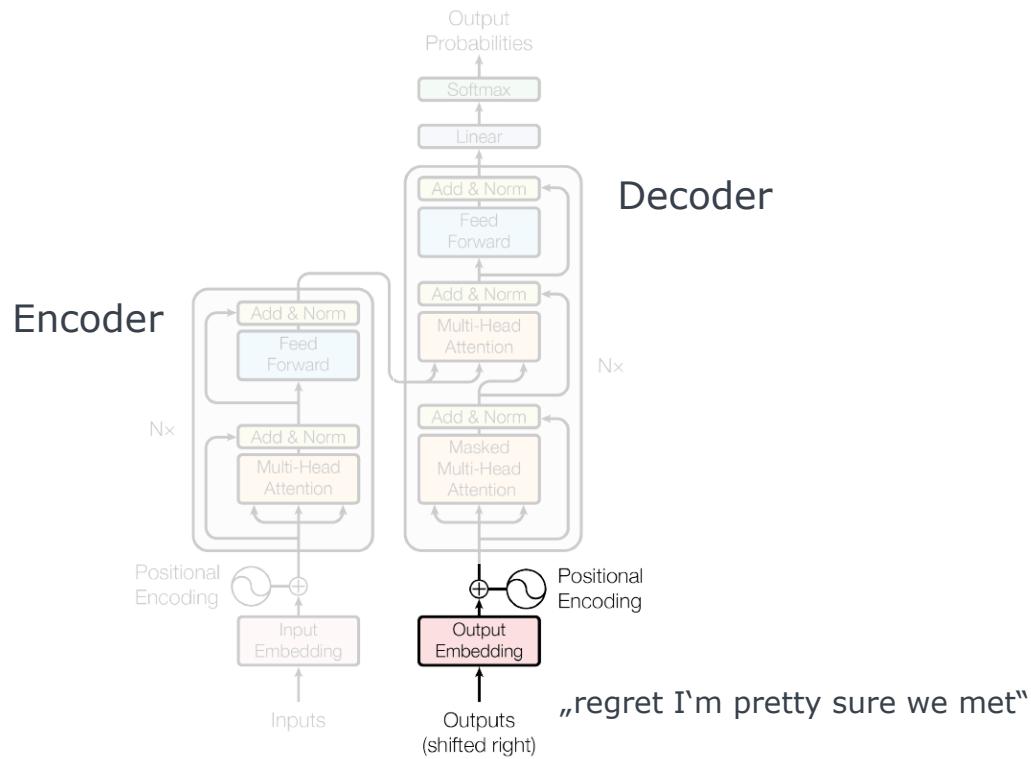
- Stabilisieren Netzwerk und deutliche Reduktion Trainingszeiten.

## Feed forward:

- Einsatz eines einfachen, nicht-rekurrenten neuronalen Netzwerks für Verarbeitung Ergebnisse (analog bisherige Vorlesungen...).
- Speichern der Ergebnisse in einem Format für die spätere Berechnung Wahrscheinlichkeit (analog Bilderkennung!)

Einsatz dieser Techniken ermöglichte Verbesserung der Ergebnisse sowie Reduzierung Ressourceneinsatz.

# TRANSFORMER IM DETAIL. DECODER – INPUT EMBEDDING UND POSITIONS ENCODING



## **Input Embedding:**

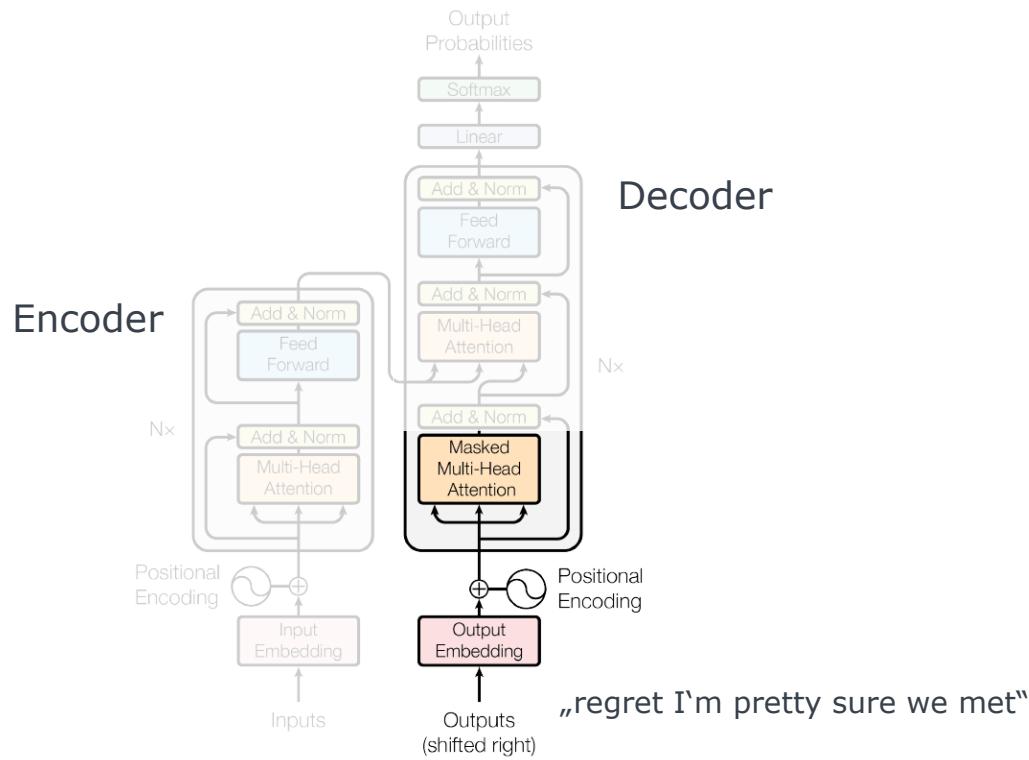
- Gleiches Vorgehen für Output-Satz wie bei Input

## **Positional Encoding:**

- Analog zu Encoding.

Gleiches Vorgehen wie beim Encoding, nur für Decoding Satz. Dieser ist beim Trainieren bekannt, beim späteren Einsatz jedoch nicht.

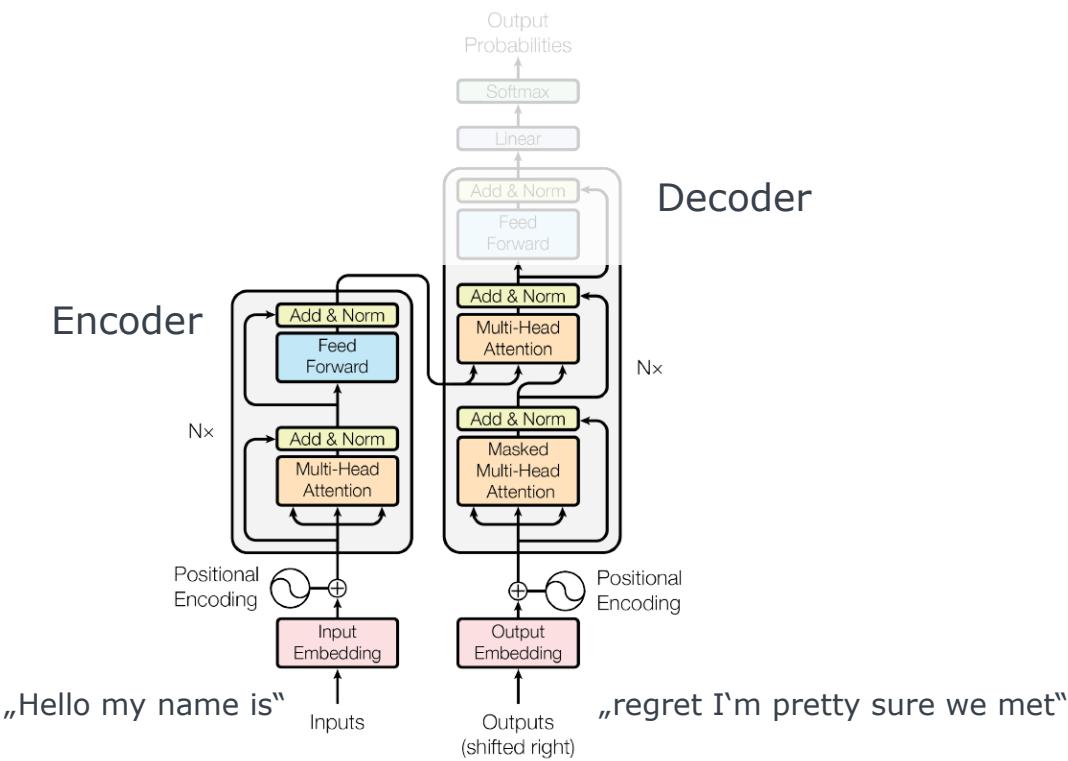
# TRANSFORMER IM DETAIL. DECODER – MASKED MULTI-HEAD ATTENTION.



## Masked Multi-head attention

- Ähnlicher Schritt wie Multi-Head Attention beim Encoding.
- Aber: beim Trainieren haben wir jeweils einen Satz für Input als auch für Output.
- Damit das Modell lernt und nicht einfach das nächste Wort „nachschaut“, werden die restlichen Wörter des Vergleichssatzes ausmaskiert (auf 0 gesetzt), d.h. das Modell kann diese nicht nachsehen.
- Dadurch kann auch parallel trainiert werden und das Training somit beschleunigt werden.

# TRANSFORMER IM DETAIL. DECODER –MULTI-HEAD ATTENTION.

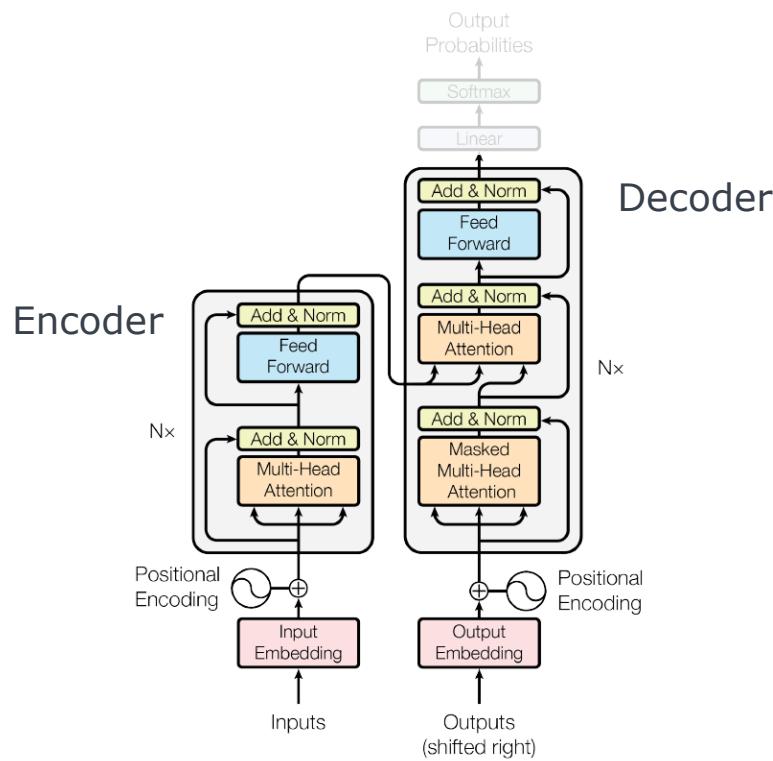


## Multi-head attention (oder Encoder-Decoder Attention)

- In diesem Block kommen die Tokens vom Encoder und die maskierten Eingaben vom Decoder (inkl. Residual) zum ersten Mal zusammen.
- Dadurch kann das Modell für jeden Token vom Decoder lernen, welcher Input vom Encoder relevant ist.

Hier entsteht das Mapping von Input auf Output also bspw. ein Attention Vektor für jedes Wort

# TRANSFORMER IM DETAIL. DECODER –FEED FORWARD.



## Normalisierung:

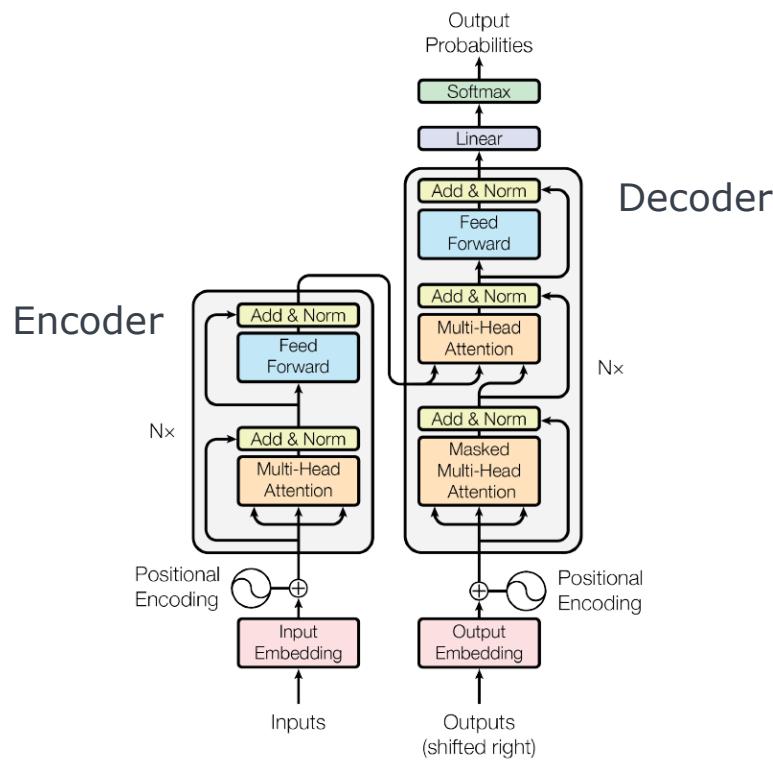
- Stabilisieren Netzwerk und deutliche Reduktion Trainingszeiten.

## Feed forward:

- Einsatz eines einfachen, nicht-rekurrenten, neuronalen Netzwerks für Verarbeitung Ergebnisse.
- Speichern der Ergebnisse in einem Format für spätere Berechnung Wahrscheinlichkeit (analog Bilderkennung!)

Einsatz dieser Techniken ermöglichte Verbesserung der Ergebnisse sowie Reduzierung Ressourceneinsatz.

# TRANSFORMER IM DETAIL. DECODER – OUTPUT.



## Lineares Neuronales Netz

- so groß wie das gesamte Vokabular für Output, d.h. eine Zelle je Wort (gleiches Vorgehen wie Flatten-Vektor bei CNN!)

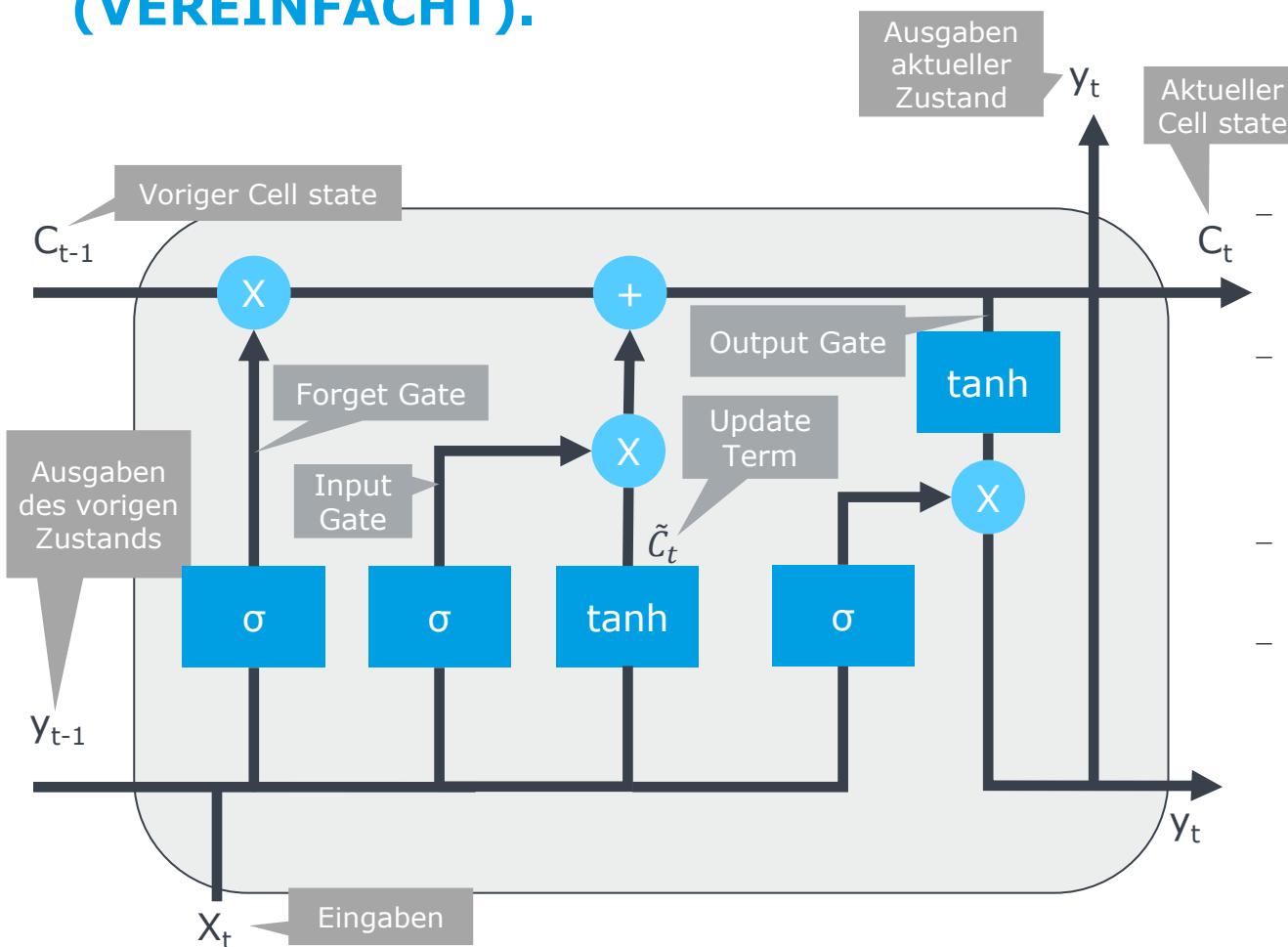
## Softmax:

- Erstellt Wahrscheinlichkeitsverteilung für alle Zellen (d.h. über alle Wörter des gesamten Wörterbuchs).
- Die Zelle mit höchstem Wert für Wahrscheinlichkeit wird als vorhergesagtes nächstes Token genommen.

Zusammengefaßt ist die Aufgabe des Decoders das Sammeln aller notwendigen Informationen um mittels einer Wahrscheinlichkeitsverteilung aus dem gesamten Vokabular das nächste Ausgabewort möglichst genau vorherzusagen.

## DETAILLIERUNG LSTM

# LONG SHORT-TERM MEMORY: STRUKTUR UND ABLAUF (VEREINFACHT).



Wir schauen uns eine von mehreren LSTM-Zelle an, die miteinander verknüpft sind, d.h. die abgebildete Zelle hat „Nachbarzellen“.

- **Forget Gate**: entscheidet, welche Infos des Cell State  $C_{t-1}$  zu vergessen oder weiter zu erinnern sind. Geschieht per Parameter mit Wert 0 für Vergessen, 1 für Erinnern oder Wert dazwischen, falls Gate unsicher ist.
- **Input Gate**: hat zwei Schritte.  
Schritt 1: Entscheidet, welche Infos für Cell State  $C_t$  zu aktualisieren sind. Erfolgt per Parameter mit Werten von 0 (irrelevant) bis 1 (relevant).  
Schritt 2: Bestimmt Update-Terme, d.h. die neuen Werte für den Cell State.
- **Cell State**: Hier wird die Zelle  $C_t$  aktualisiert, basierend auf Inputs des Forget Gate und Input Gates.
- **Output Gate**: entscheidet was aus der Zelle ausgegeben wird.

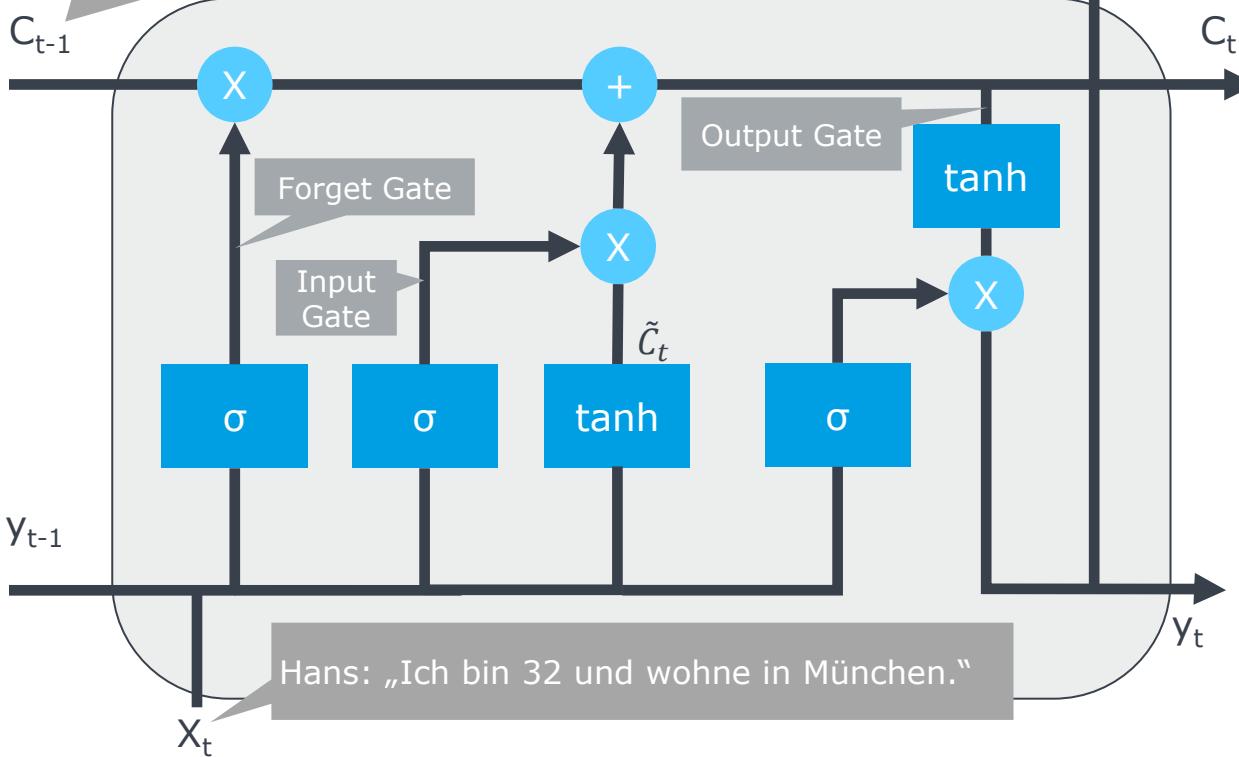
LSTM vermeidet Vanishing Gradient durch:

- Steuerung Verhalten Gradienten durch Werte Gates: Wert 0 verhindert bspw. Änderung Gradient.
- Werte des Gates für Vermeiden Explosion werden gelernt.
- Formel für Ausgabe  $C_t$  enthält Addition. Dadurch hat die den Gradienten erzeugende Ableitung ein „besseres Verhalten“ als bei bspw. Multiplikation

## LONG SHORT-TERM MEMORY: FALLBEISPIEL (VEREINFACHT).

Voriger Cell state	
Geschlecht	Dort
Weiblich	Berlin

Ausgabe
Dort
München



**Ziel:** „Verstehen“ einer Konversation durch Algorithmus

Anna: „Ich bin 30 Jahre alt und wohne in Berlin“

Hans: „Ich bin 32 und wohne in München.“

Anna: „Wie viele Menschen leben dort?“

Frage: worauf bezieht sich dort?

Geschlecht	Dort
Weiblich	Berlin

**Start mit Hans' Satz:** Cell state  $C_{t-1}$ :

- **Forget Gate:** vergiss Werte für Geschlecht und Wohnort (Forget gate = 0), da mit Hans andere Person mit anderem Geschlecht und Wohnort spricht.
- **Input Gate:**
  - Schritt 1: bestimme zu aktualisierende Werte
  - Schritt 2: bestimme Änderungswerte  $\tilde{C}_t$
- **Cell State:** schreibe die Werte in die Zelle  $C_t$
- **Output Gate:** schreibe in Ausgabe  $y_t$  Wert für dort, da dieser Begriff für Annas nächsten Satz relevant ist. LSTM lernt also, daß Annas Frage sich auf München bezieht!

Geschlecht	Dort
Männlich	München

Geschlecht	Dort
Männlich	München

Dort
München