Vorlesung Fortgeschrittene Softwaretechnik

Wintersemester 2024/25

Prof. Dr. Stephan Diehl

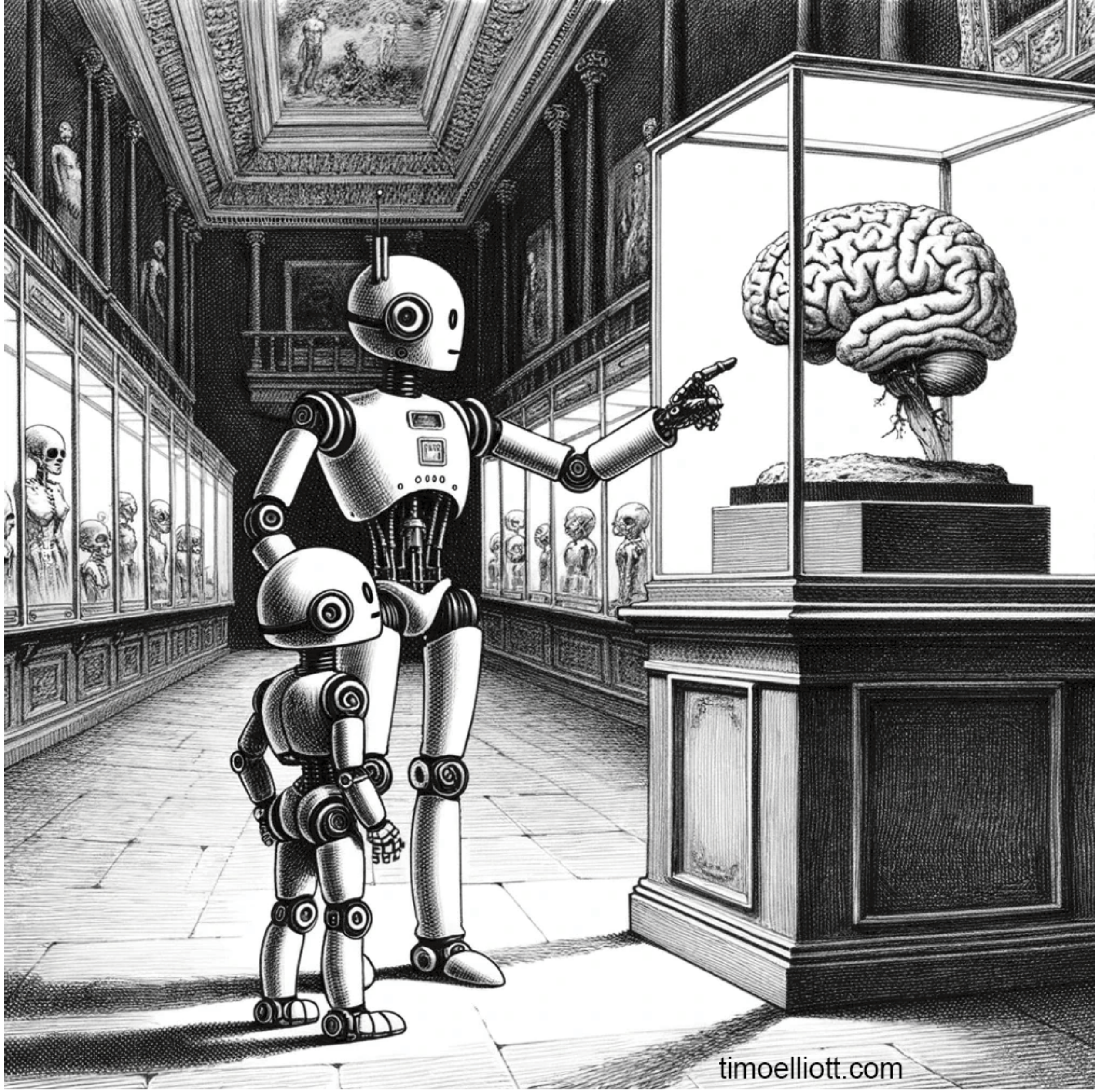
Informatik

Universität Trier

Large Language Models

Theorie

Bildquelle: https://i0.wp.com/timoelliott.com/blog/wp-content/uploads/2023/10/and-that-is-the-original-processor-1.png?w=1024&ssl=1



"And that is the original processor!"

Plan für die nächsten Wochen

Themenblöcke bisher: Testen, CI, Patterns, VR/AR+SE

	Datum	Thema/Inhalt	Übung			Dozent	Block
DO	12.12.2024	Empirische SE + Software Evolution				SD	Quantitative Studien
DI	17.12.2024	MSR, Empfehlungsdienste	Praxis BOA		Ausgabe LIT	SD	
DO	19.12.2024	???					
	FREI						
DI	07.01.2025	RG MSR/BOA	Übung BOA			SD	
DO	09.01.2025	Research Design + Quantitative Analyse				SD	
DI	14.01.2025	Qualitative Analyse	Praxis: QA 1		Ausgabe LIT	SD	Qualitative Studien
DO	16.01.2025	Praxis: QA 2 (gemeinsames Kodieren)				SD	
DI	21.01.2025	RG QA+SE	Übung QA			SD	
DO	23.01.2025	Info zu Portfolio				SD	
DI	28.01.2025	LLM + SE	Übung LLMSE		Ausgabe LIT	SD	LLM+SE
DO	30.01.2025	LLM + SE				SD	
DI	04.02.2025	RG: LLM+SE	Übung LLMSE			SD	
DO	06.02.2025	----------------------------				SD	
DI	11.02.2025	Abgabe Expose				SD	
DO	13.02.2025	----------------------------				SD	

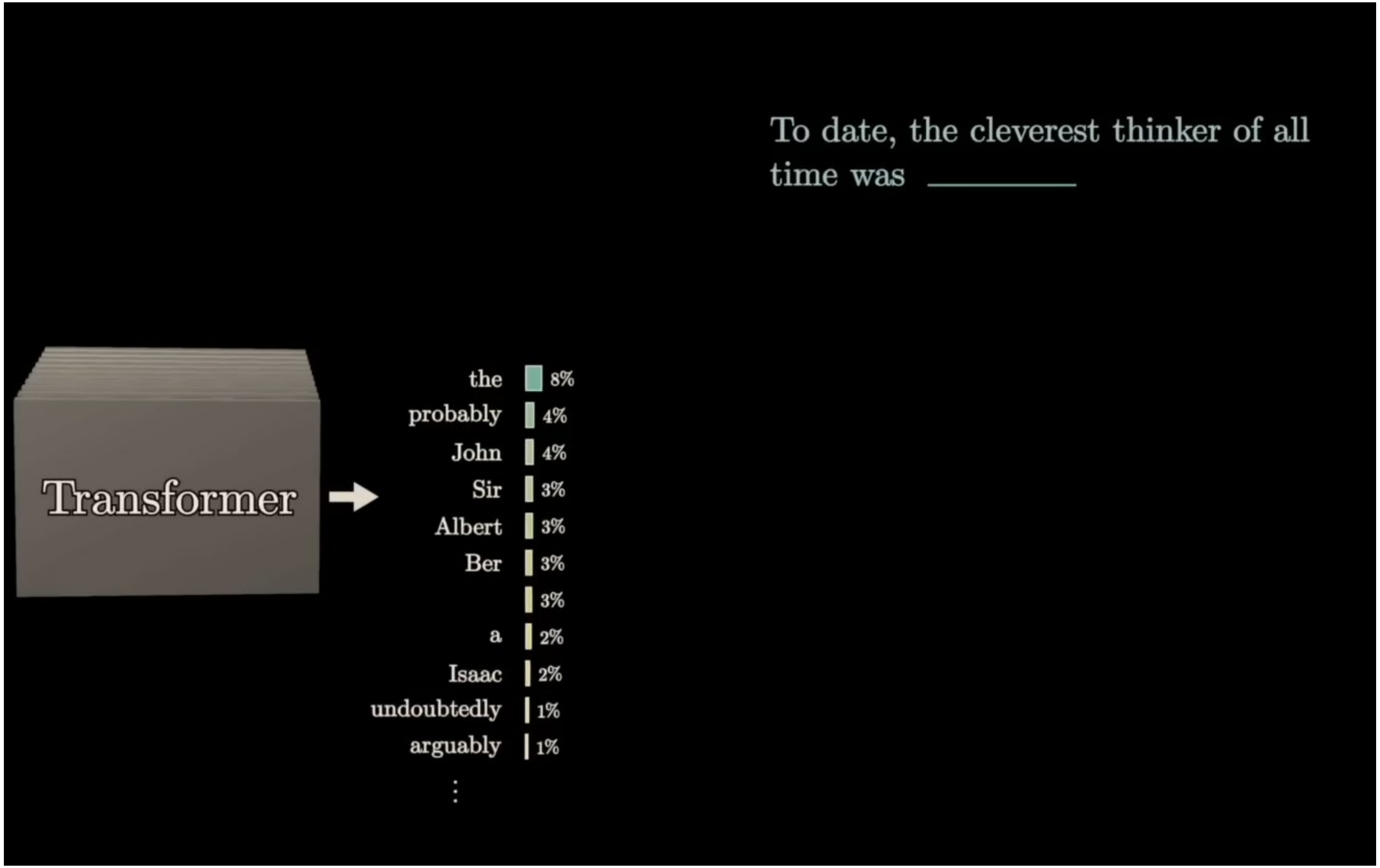
Transformers erklärt in 9 Minuten !!!

- **Transformers, explained: Understand the model behind GPT, BERT, and T5**
- https://www.youtube.com/watch?v=SZorAJ4I-sA

Grundlegende Mathematische Operationen

- Embedding in mehrdimensionalen Raum
- Skalar-Produkt (dot product, entspricht cos(), wenn Vektoren normiert)
- Matrixmultiplikation vs. Netzwerk
- Softmax

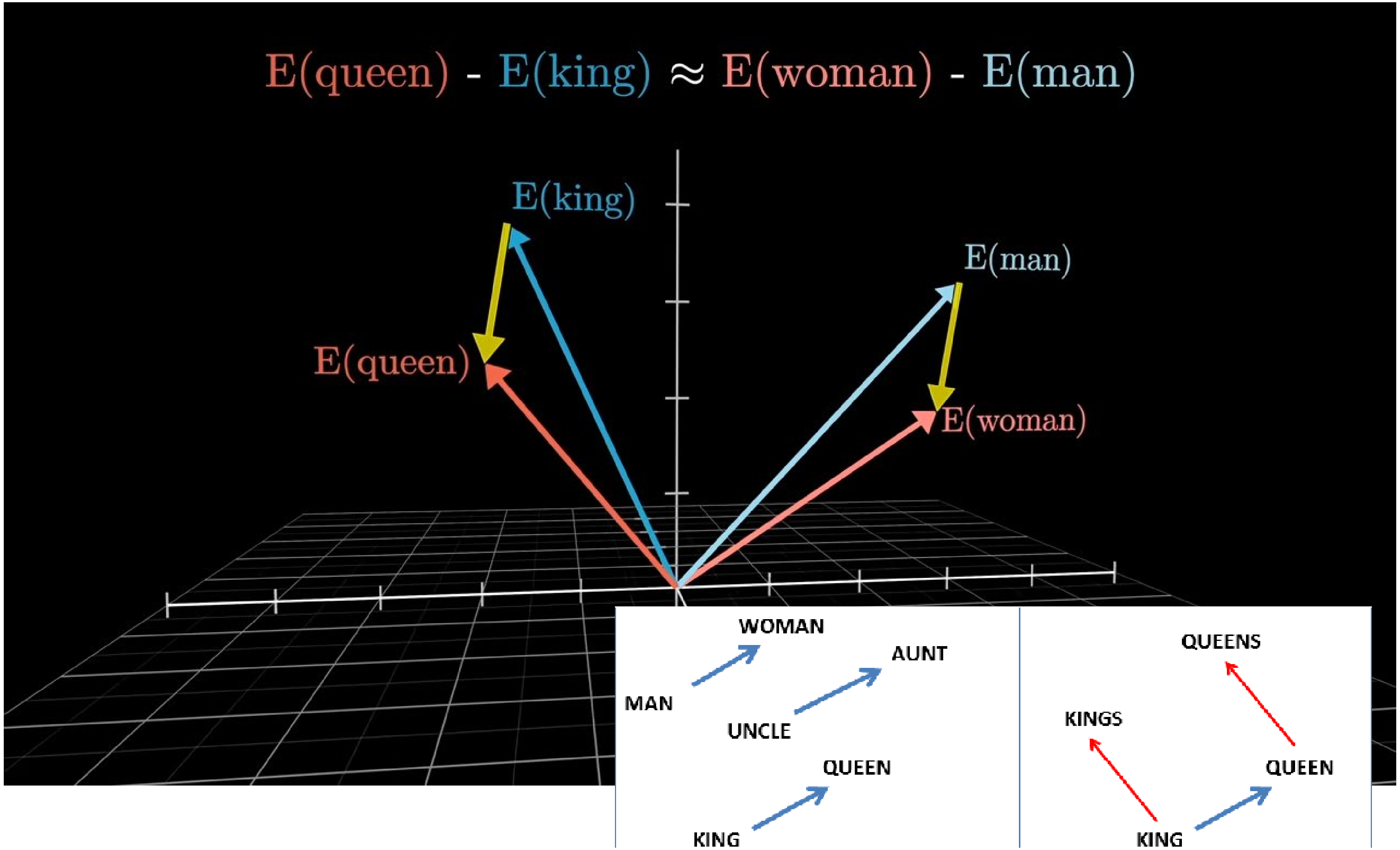
LLM berechnet bedingte Wahrscheinlichkeit: **p(next token | previous tokens)**



Quelle: https://www.youtube.com/watch?v=KJtZARuO3JY ab Minute 2:00

Word Embeddings

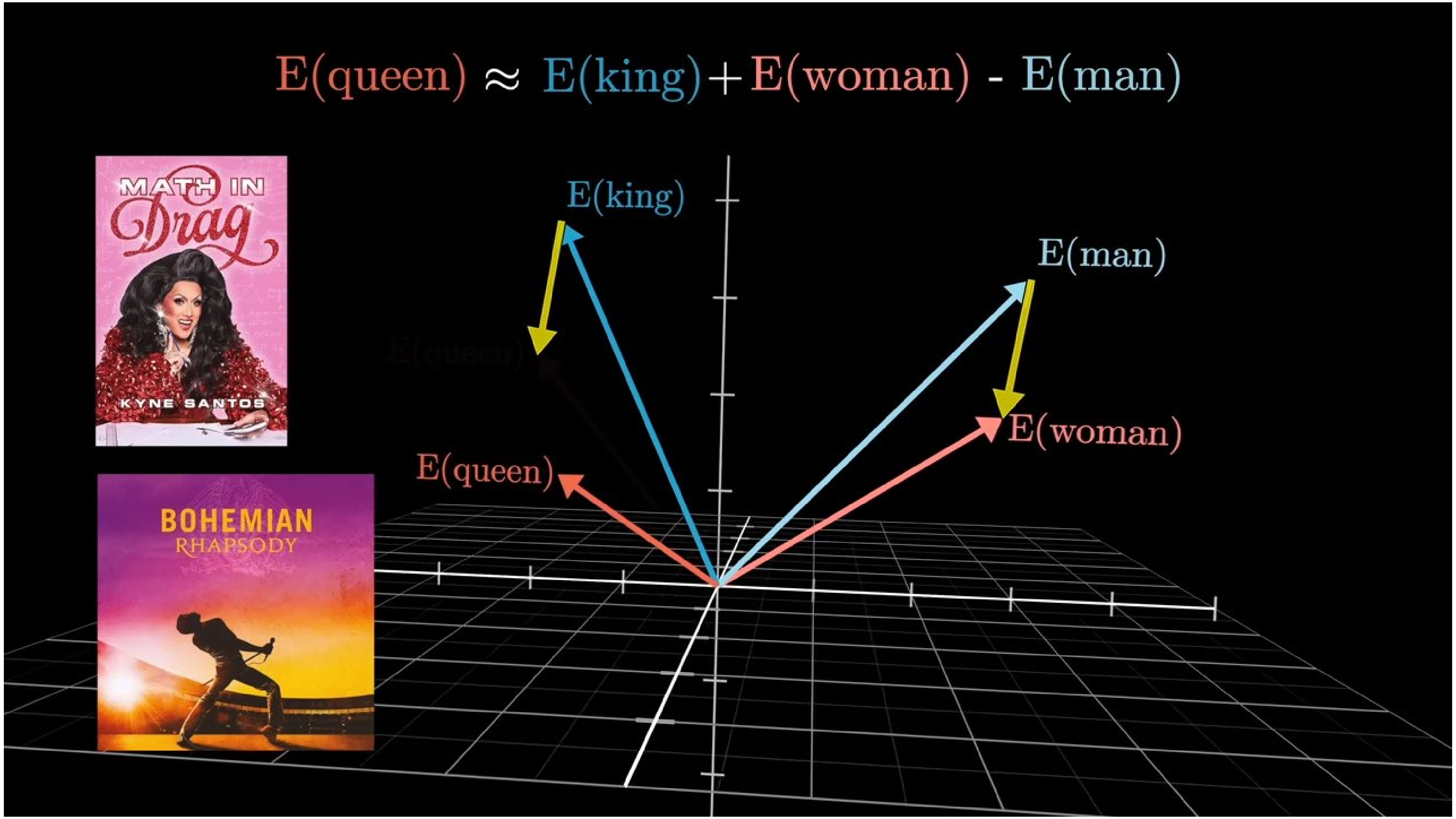
Klassisches Beispiel



Quelle: https://www.youtube.com/watch?v=wjZofJX0v4M

Quelle: https://api.semanticscholar.org/CorpusID:7478738

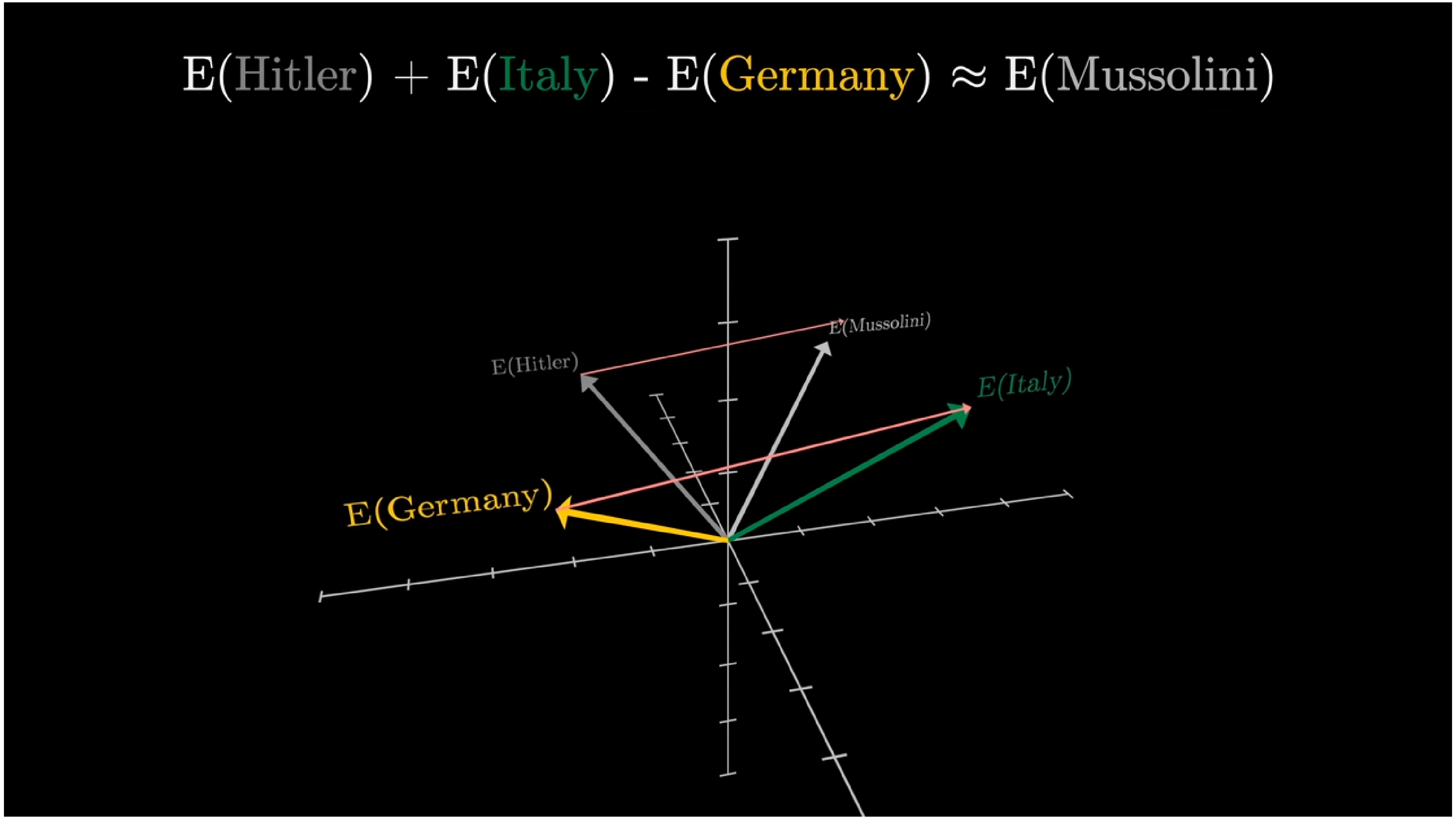
Word Embeddings



Gelernte Einbettung aus realistischen Trainingsdaten

Quelle: https://www.youtube.com/watch?v=wjZofJX0v4M

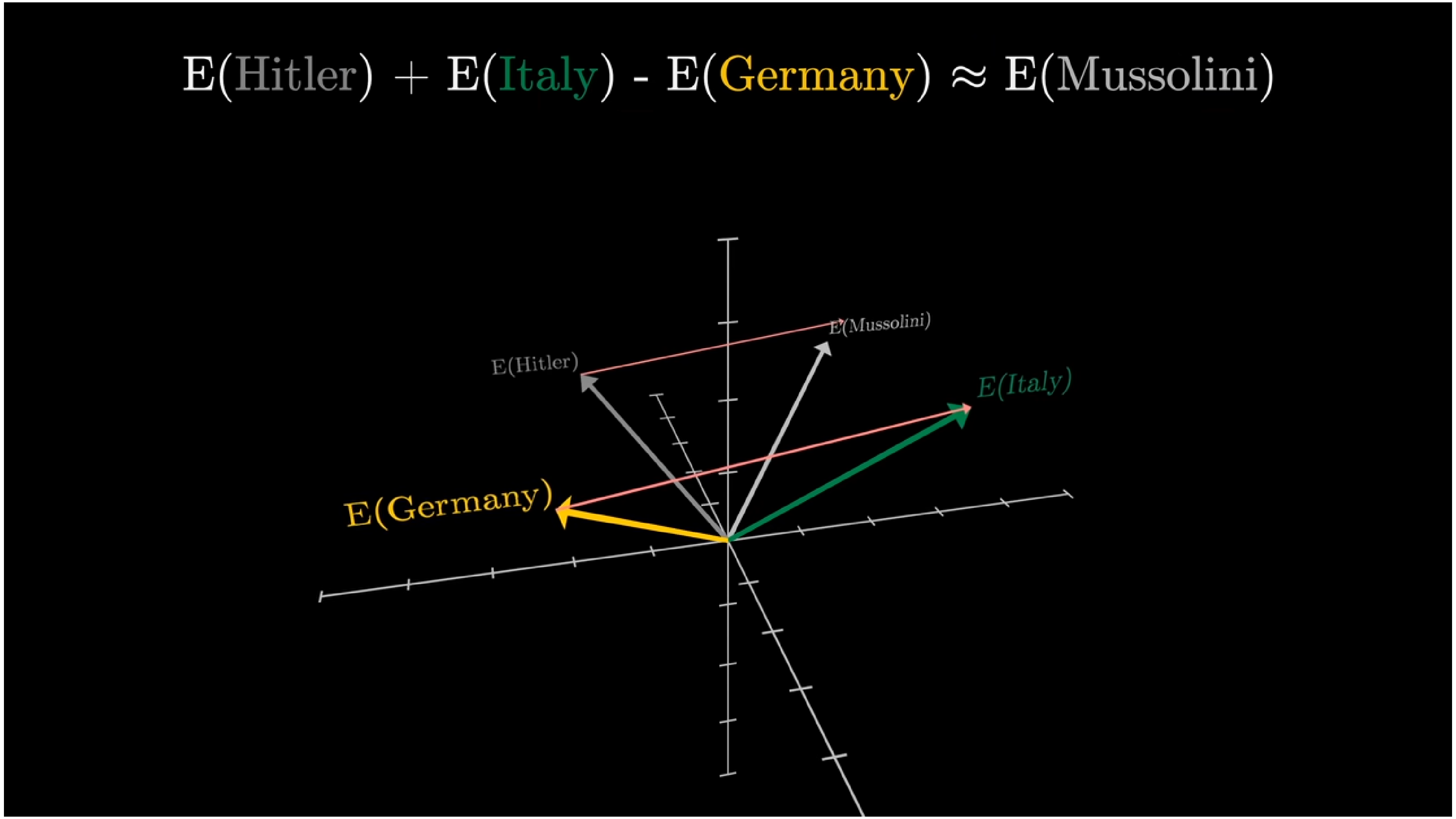
Word Embeddings



Gelernte Einbettung aus realistischen Trainingsdaten

Quelle: https://www.youtube.com/watch?v=wjZofJX0v4M

Word Embeddings

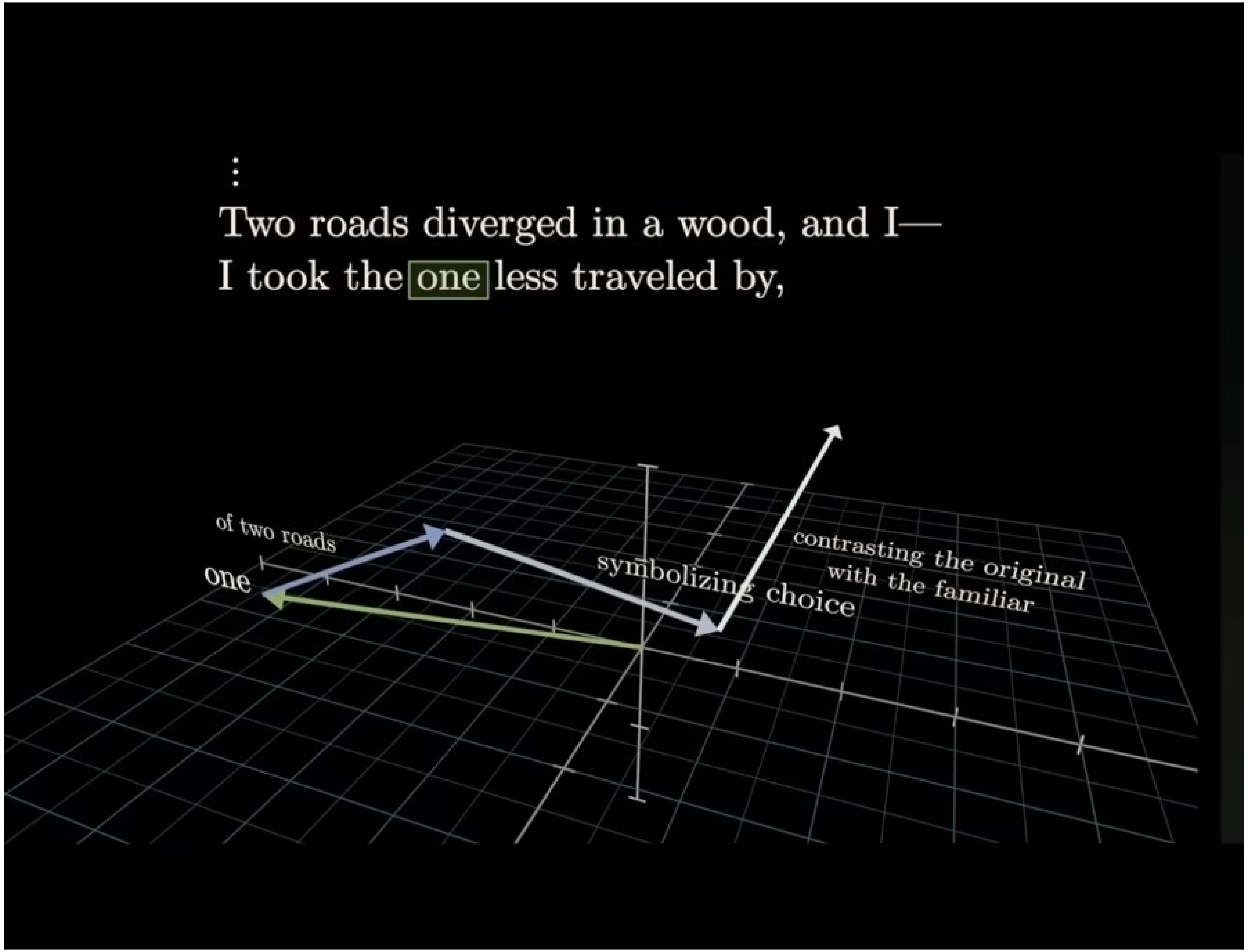


Gelernte Einbettung aus realistischen Trainingsdaten

Quelle: https://www.youtube.com/watch?v=wjZofJX0v4M ab Minute 15:30

Word Embeddings

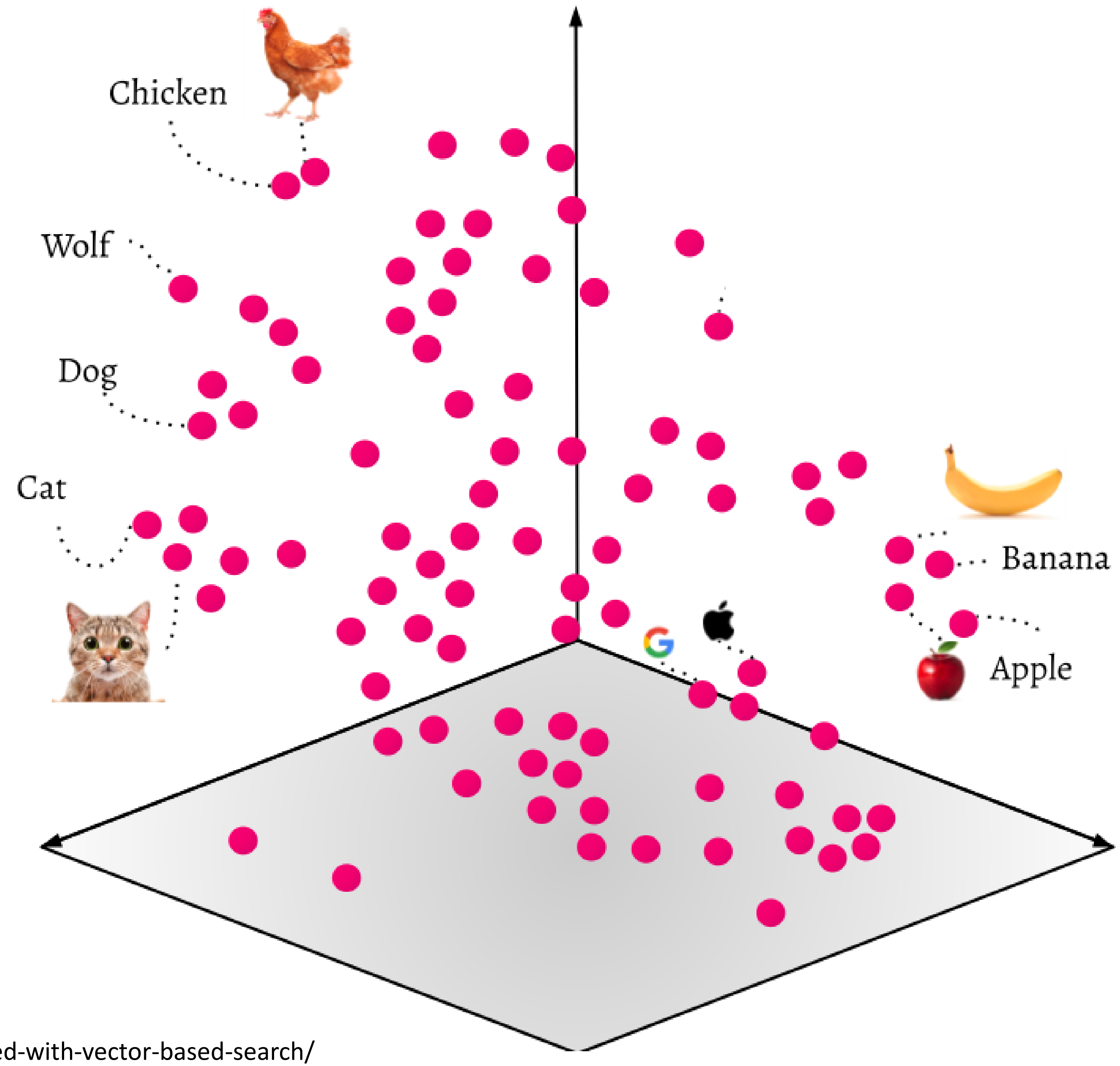
In GPT: Vektoren im Embedding Space repräsentieren nicht nur Worte (oder Token), sondern kodieren auch die Position des Wortes in der Eingabe und können Informationen über den Kontext erfassen.



Quelle: https://www.youtube.com/watch?v=KJtZARuO3JY ab Minute 18:45

Skalarprodukt

Wortähnlichkeit

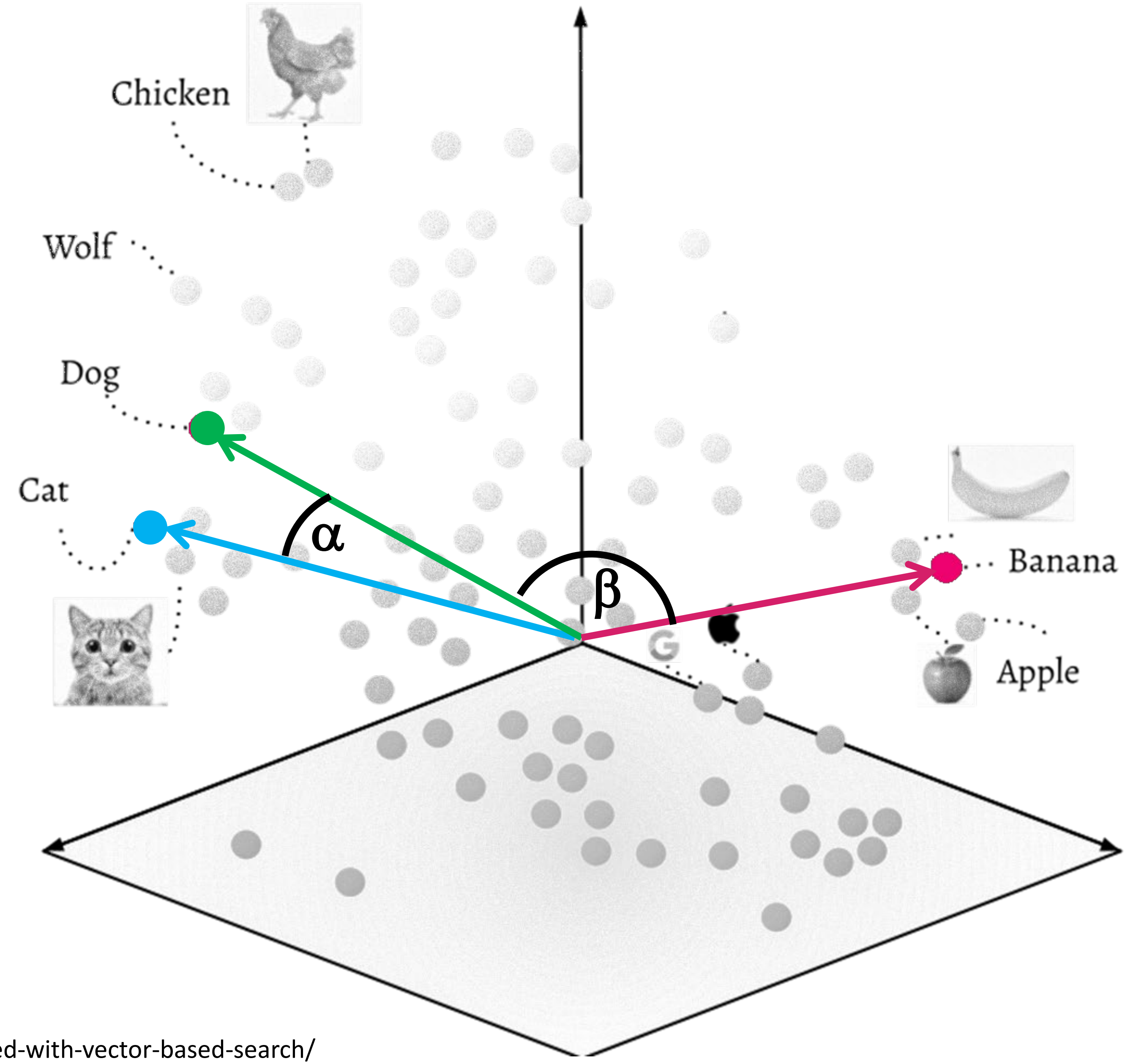


Bildquelle: https://odsc.com/blog/getting-started-with-vector-based-search/

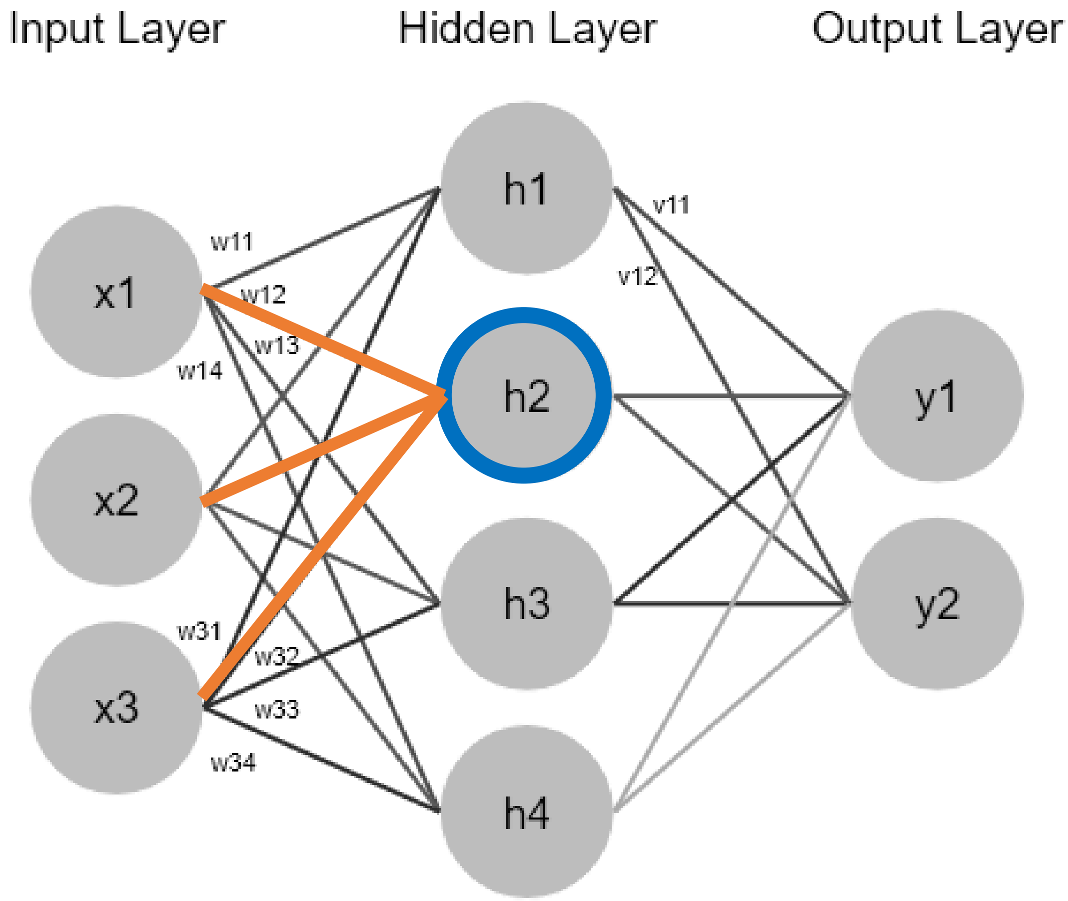
Skalarprodukt

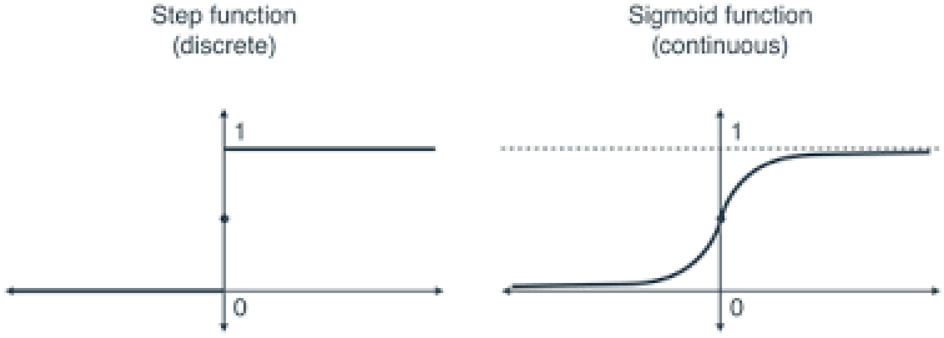
Wortähnlichkeit

Skalar-Produkt (dot product) zweier Vektoren entspricht dem Kosinus des eingeschlossenen Winkels, wenn die Vektoren normiert sind.



Bildquelle: https://odsc.com/blog/getting-started-with-vector-based-search/





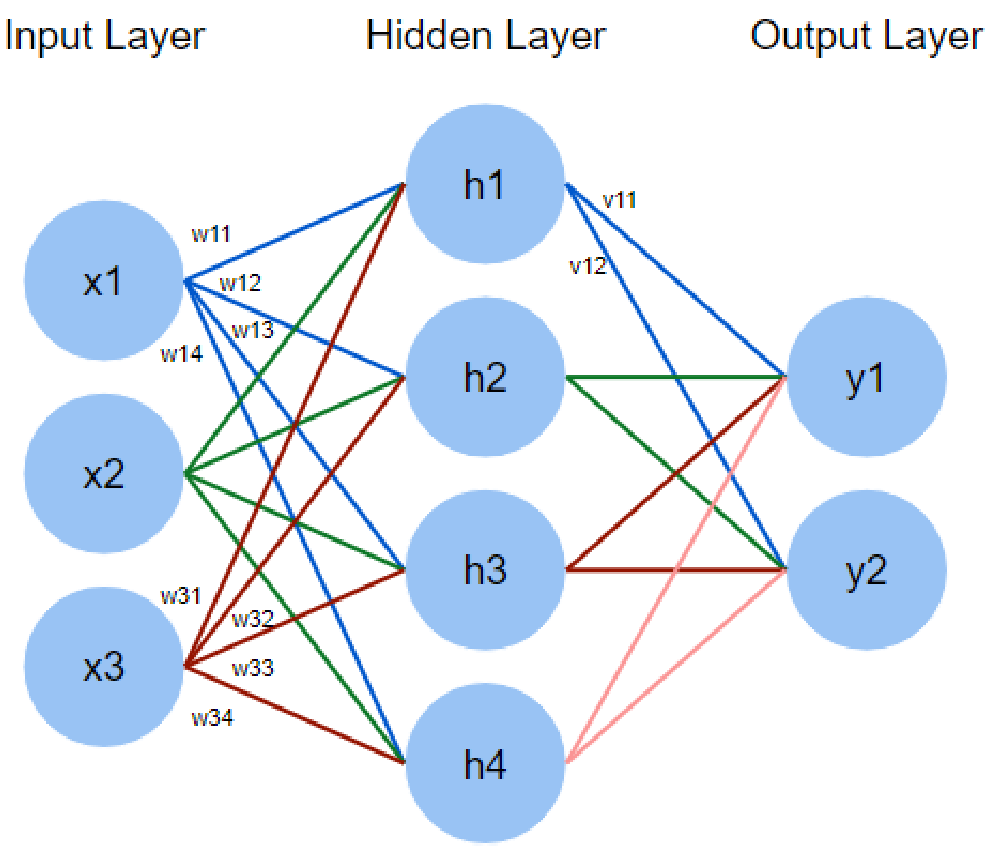
Ein Neuron berechnet die gewichtete Summe seiner eingehenden Verbindungen und leitet den nicht-linear gefilterten Wert weiter:

$$h_2' = w_{1,2} * x_1 + w_{2,2} * x_2 + w_{3,2} * x_3$$

$$h_2' = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} * \begin{bmatrix} w_{1,2} \\ w_{2,2} \\ w_{3,2} \end{bmatrix}$$ Skalar-Produkt

$$h_2 = \frac{1}{1 - e^{h_2'}}$$

Matrixmultiplikation berechnet jeden Eintrag des Ergebnisses durch das Skalarprodukt einer Zeile der ersten Matrix mit einer Spalte der zweiten Matrix.



Quelle: https://datascience.stackexchange.com/questions/75855/what-types-of-matrix-multiplication-are-used-in-machine-learning-when-are-they

Feedforward pass

In the feed-forward pass the input features will be multiplied by the weights at each layer to produce the outputs

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} * \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} \end{bmatrix} = \begin{bmatrix} h'_1 & h'_2 & h'_3 & h'_4 \end{bmatrix}$$

At the hidden layer these will then go through the activation function, if we assume sigmoid then

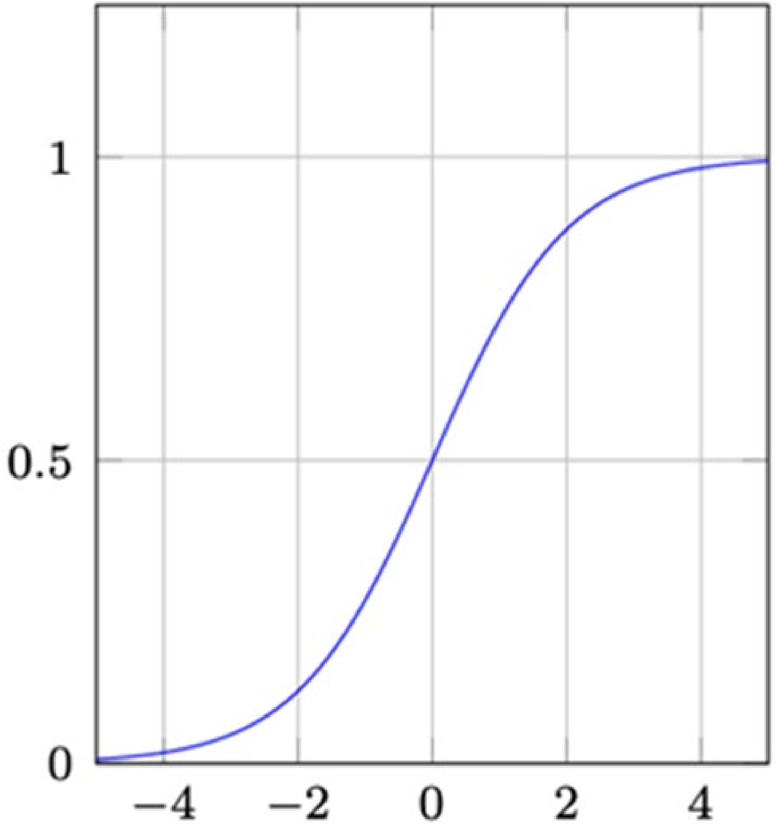
$$\begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix} = \frac{1}{1+e^{\begin{bmatrix} -h'_1 & -h'_2 & -h'_3 & -h'_4 \end{bmatrix}}}$$

Finally we go through the next set of weights to the output neurons

$$\begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix} * \begin{bmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \\ v_{3,1} & v_{3,2} \\ v_{4,1} & v_{4,2} \end{bmatrix} = \begin{bmatrix} y'_1 & y'_2 \end{bmatrix}$$

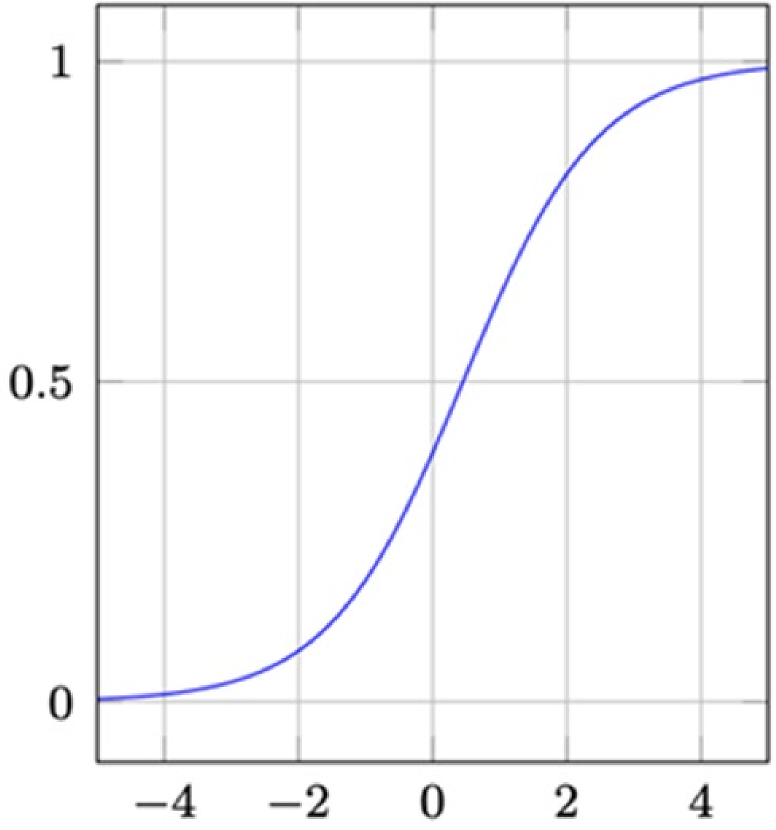
$$\begin{bmatrix} y_1 & y_2 \end{bmatrix} = \frac{1}{1+e^{\begin{bmatrix} -y'_1 & -y'_2 \end{bmatrix}}}$$

Sigmoid

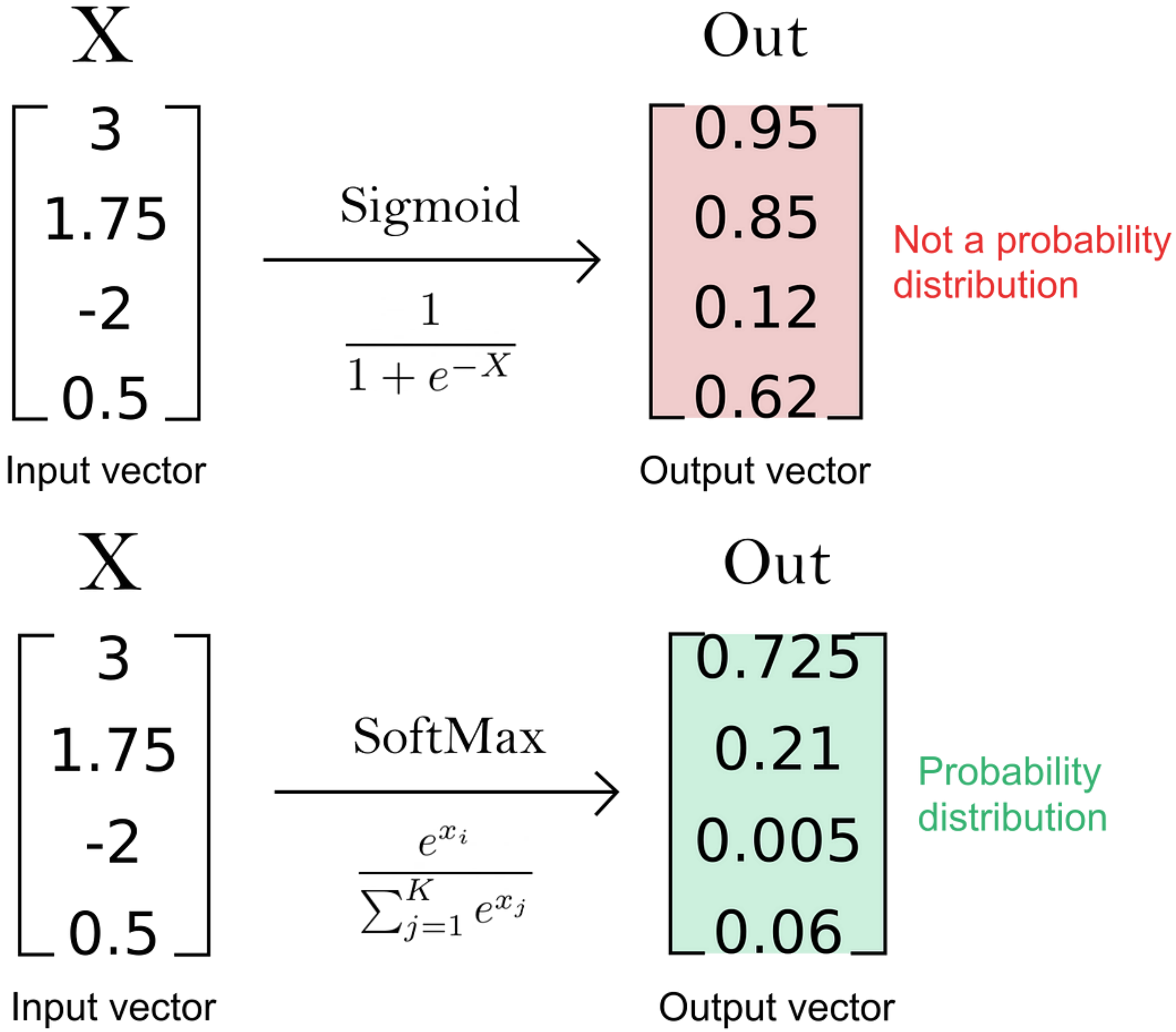


$$s(x_{i)}) = \frac{1}{1 - e^{-x_i}}$$

Sigmax



$$s(x_i) = \frac{e^{x_i}}{1\sum_{j=1}^{n} e^{x_j}}$$

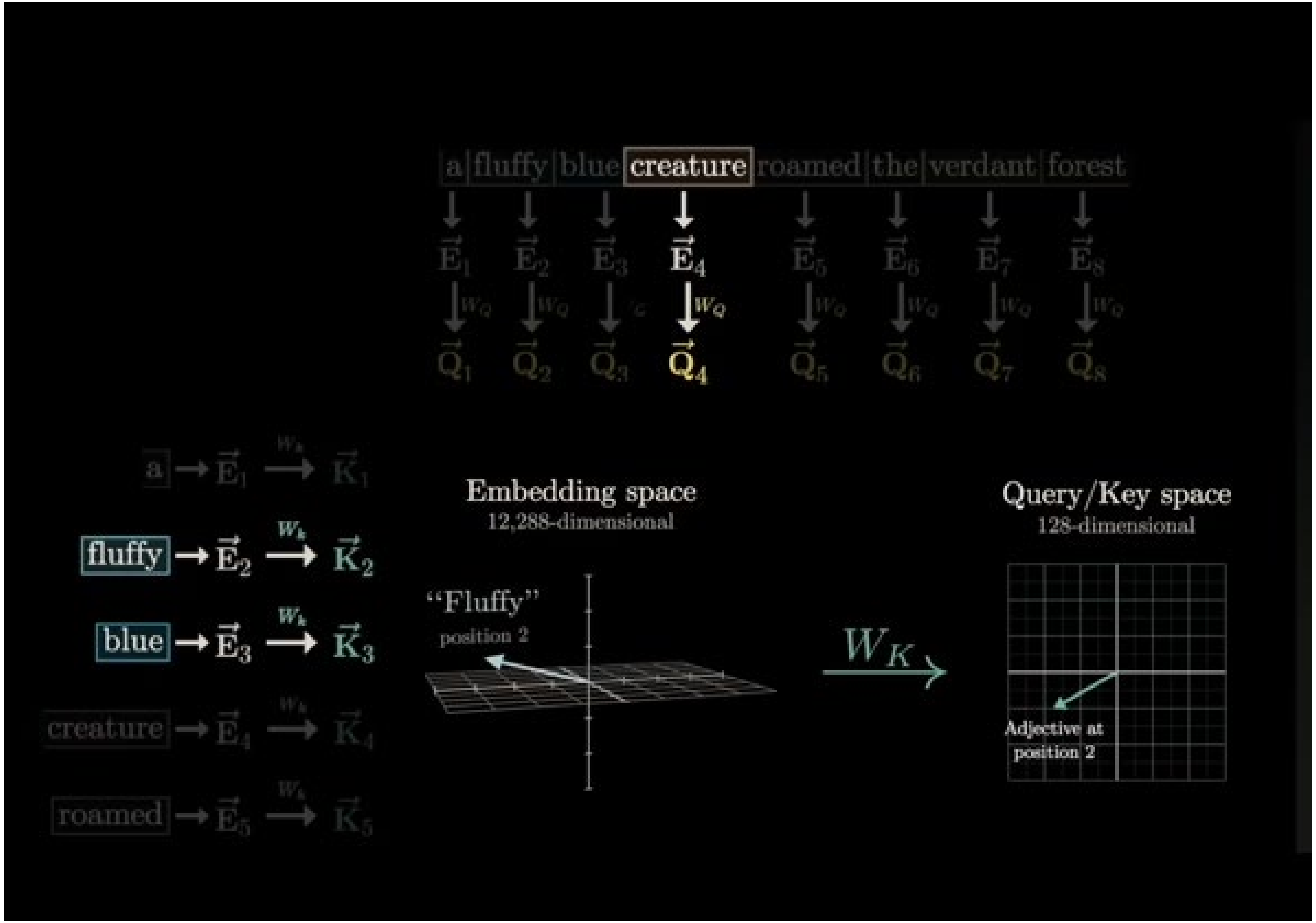


Summe der Einträge ist 1

Quelle: https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9

Self-Attention

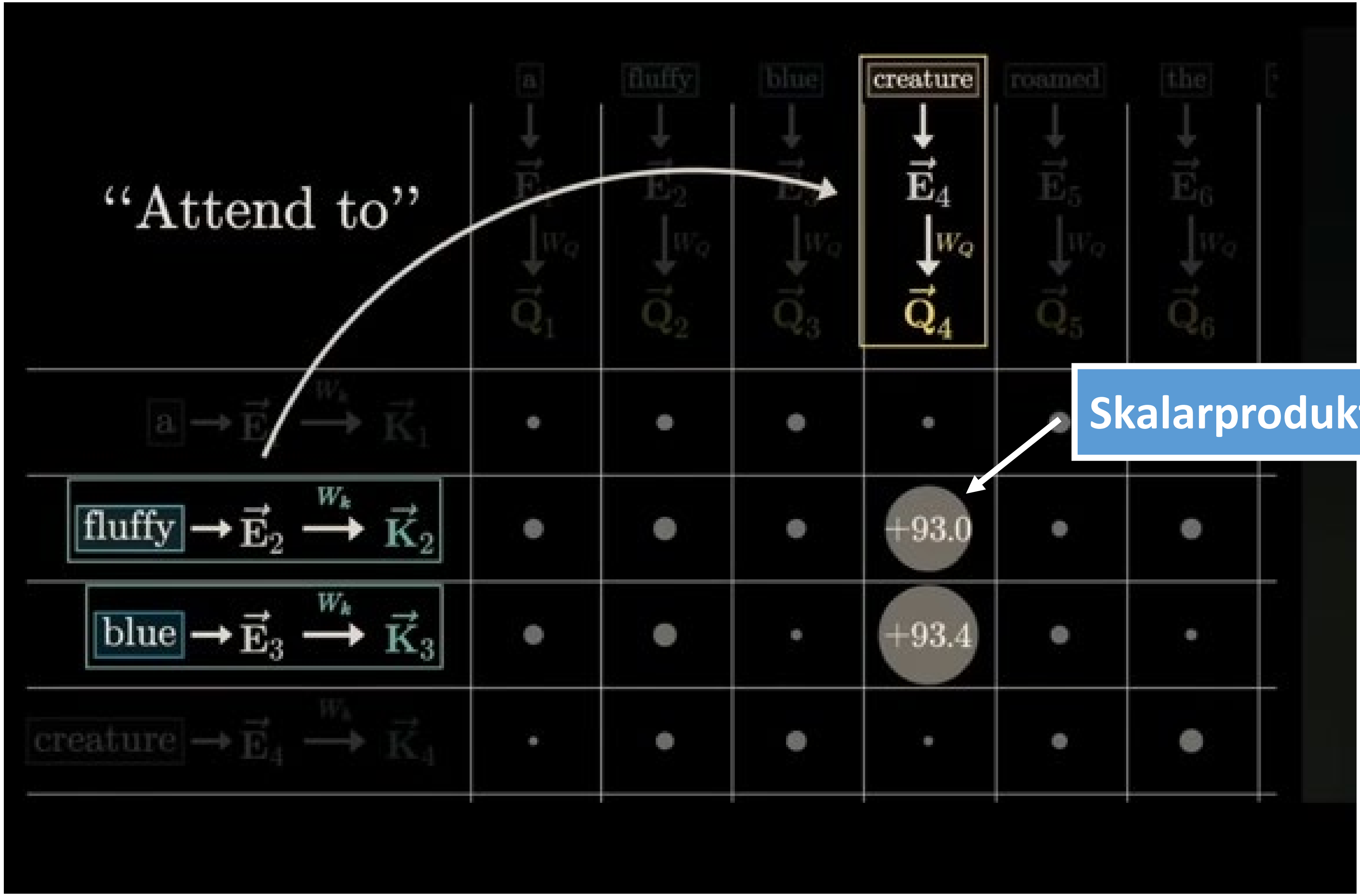
Abbildung vom Embedding Space in den kleineren Query/Key Space



Quelle: https://www.youtube.com/watch?v=KJtZARuO3JY ab Minute 29:15

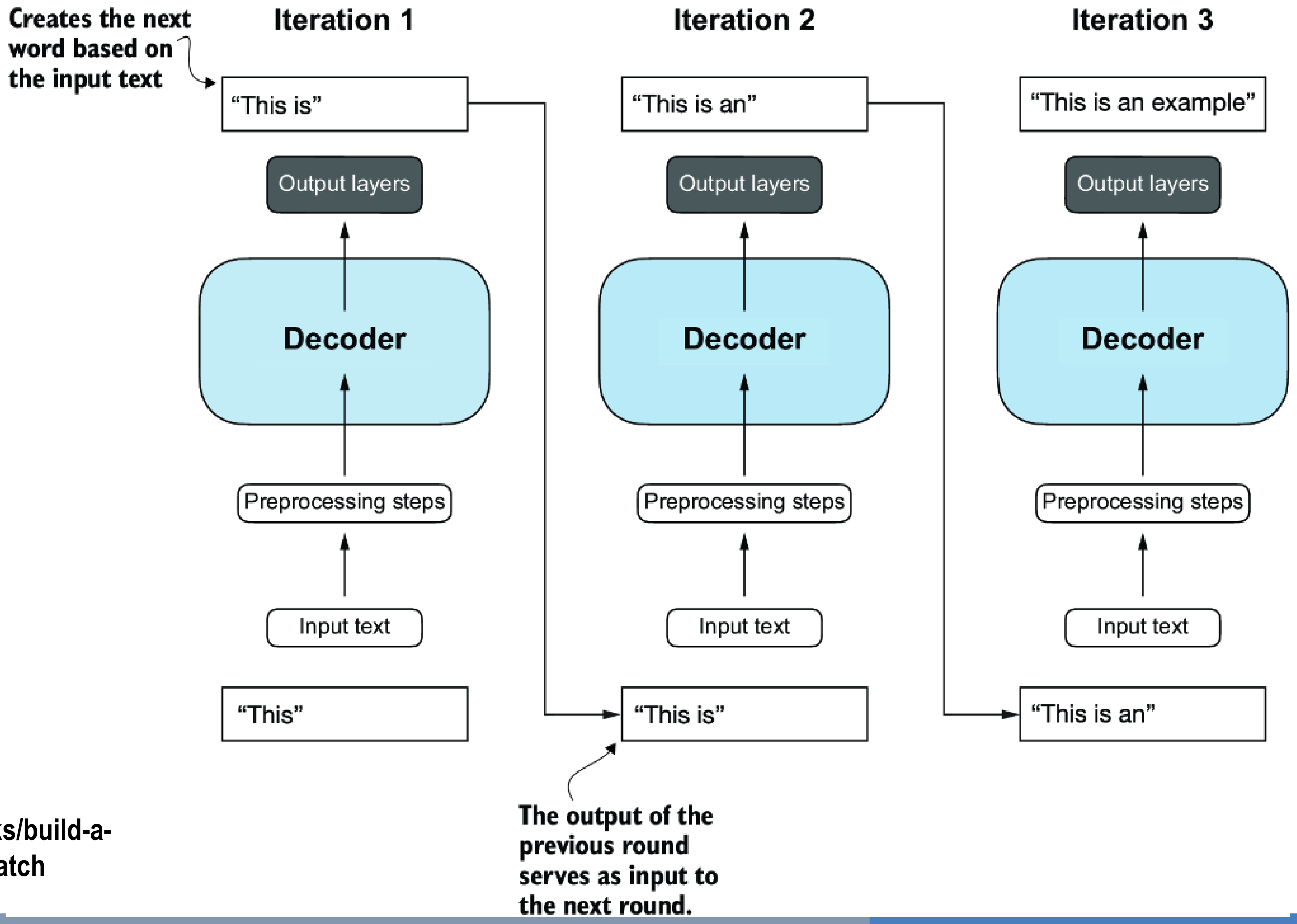
Self-Attention

Matrizenmultiplikation berechnet den Einfluss von jedem Token der Eingabe auf jedes andere.



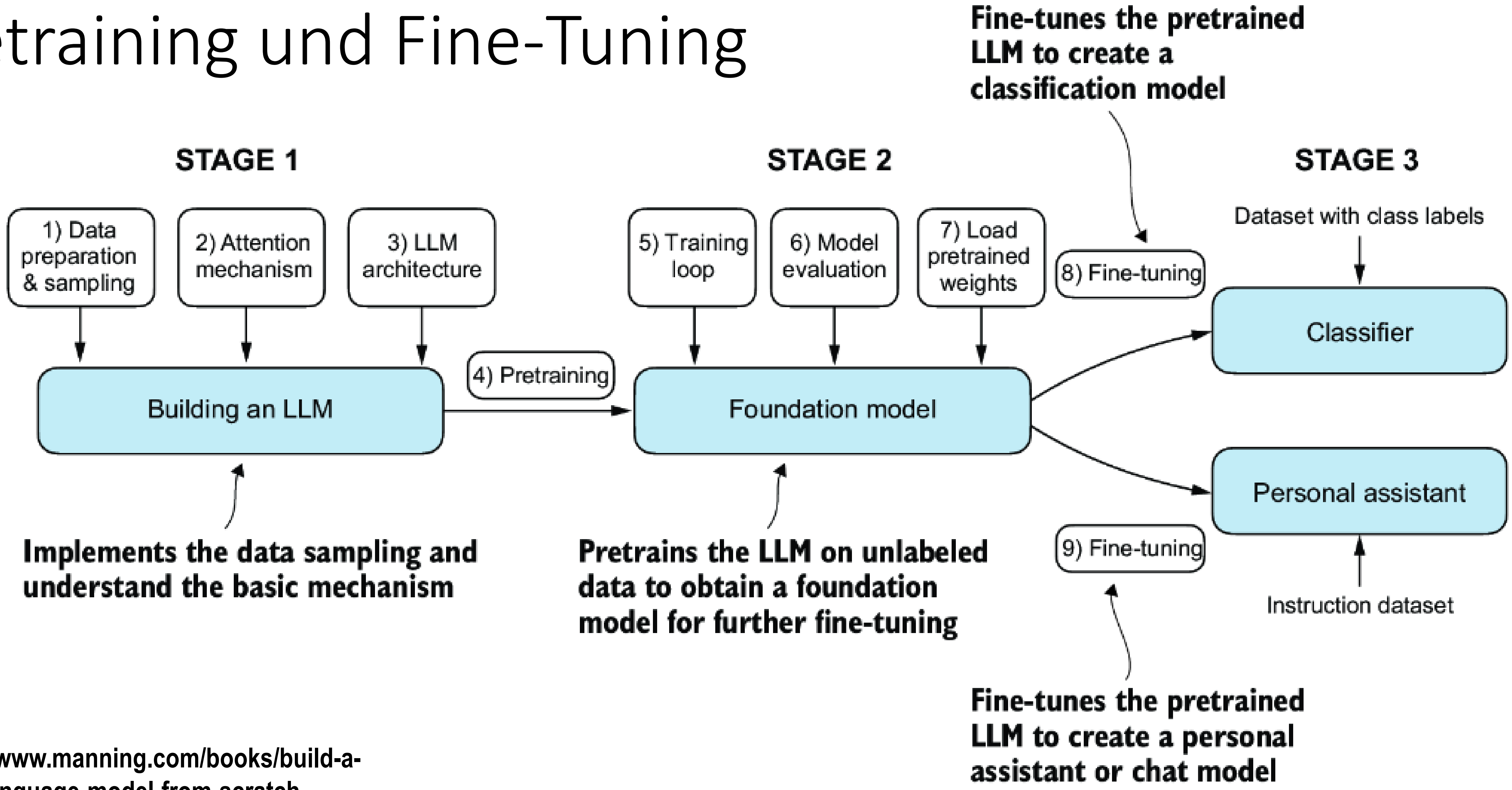
Skalarprodukt

Quelle: https://www.youtube.com/watch?v=KJtZARuO3JY ab Minute 29:15



Quelle:
https://www.manning.com/books/build-a-large-language-model-from-scratch

Pretraining und Fine-Tuning



Quelle:
https://www.manning.com/books/build-a-large-language-model-from-scratch

3Blue1Brown

Aber was ist ein GPT? Visuelle Einführung in Transformers | Deep Learning, Kapitel 5
https://www.youtube.com/watch?v=wjZofJX0v4M

Attention in transformers, visually explained | DL6
https://www.youtube.com/watch?v=eMlx5fFNoYc

How might LLMs store facts | DL7
https://www.youtube.com/watch?v=9-Jl0dxWQs8

Large Language Models explained briefly
https://www.youtube.com/watch?v=LPZh9BOjkQs

Kurzfassung:

- **Visualizing transformers and attention | Talk for TNG Big Tech Day '24**
- https://www.youtube.com/watch?v=KJtZARuO3JY

Weitere gute Lernvideos

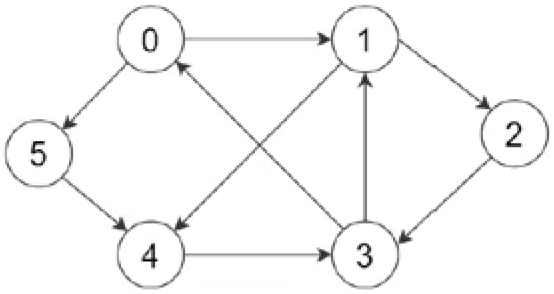
Understanding ChatGPT and LLMs from Scratch - Part 1
https://www.youtube.com/watch?v=Wt3Oicmy9VA

Transformers erklärt in 9 Minuten !!!
Transformers, explained: Understand the model behind GPT, BERT, and T5
https://www.youtube.com/watch?v=SZorAJ4I-sA

Stanford: Introduction to Transformers
https://www.youtube.com/watch?v=XfpMkf4rD6E

Attention basically

```python
class Node:

  def __init__(self):

    # the vector stored at this node
    self.data = np.random.randn(20)

    # weights governing how this node interacts with other nodes
    self.wkey = np.random.randn(20, 20)
    self.wquery = np.random.randn(20, 20)
    self.wvalue = np.random.randn(20, 20)

  def key(self):
    # what do I have?
    return self.wkey @ self.data

  def query(self):
    # what am I looking for?
    return self.wquery @ self.data

  def value(self):
    # what do I publicly reveal/broadcast to others?
    return self.wvalue @ self.data
```



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

```python
class Graph:

  def __init__(self):
    # make 10 nodes
    self.nodes = [Node() for _ in range(10)]
    # make 40 edges
    randi = lambda: np.random.randint(len(self.nodes))
    self.edges = [[randi(), randi()] for _ in range(40)]

  def run(self):

    updates = []
    for i,n in enumerate(self.nodes):

      # what is this node looking for?
      q = n.query()

      # find all edges that are input to this node
      inputs = [self.nodes[ifrom] for (ifrom,ito) in self.edges if ito == i]
      if len(inputs) == 0:
        continue # ignore
      # gather their keys, i.e. what they hold
      keys = [m.key() for m in inputs]
      # calculate the compatibilities
      scores = [k.dot(q) for k in keys]
      # softmax them so they sum to 1
      scores = np.exp(scores)
      scores = scores / np.sum(scores)
      # gather the appropriate values with a weighted sum
      values = [m.value() for m in inputs]
      update = sum([s * v for s, v in zip(scores, values)])
      updates.append(update)

    for n,u in zip(self.nodes, updates):
      n.data = n.data + u # residual connection
```

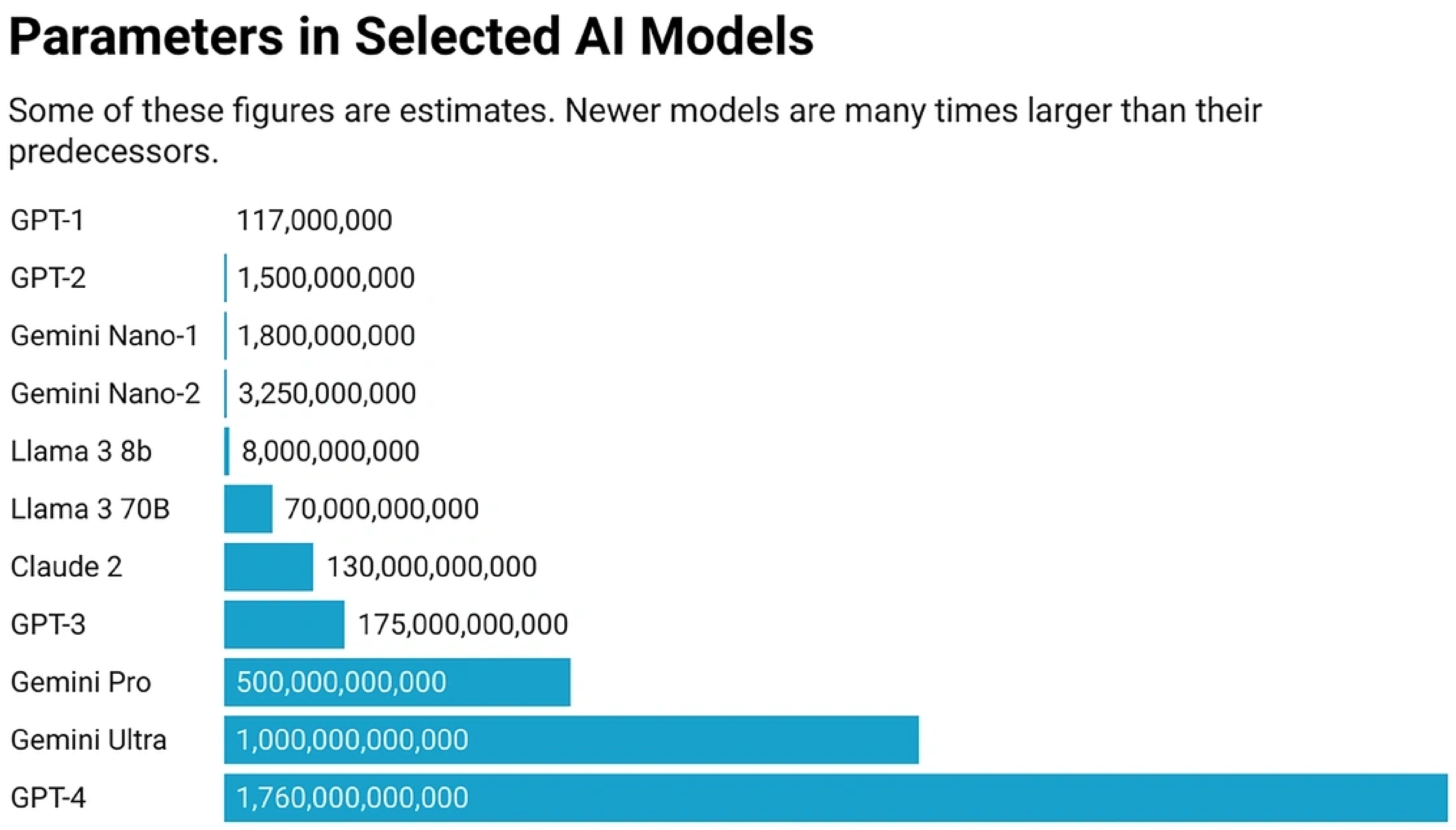
https://www.youtube.com/watch?v=XfpMkf4rD6E **ab Minute 25:30**

Einige Kennzahlen von LLMs

Parameter Count of LLMs

GPT4 is 8 x 220B params = 1.7 Trillion params

Quelle: https://explodingtopics.com/blog/gpt-parameters



Parameter = Gewichte der Verbindungen zwischen den Neuronen

➔ Gewichte in den Matrizen

Kontextfenster (Context Length)

The number of tokens an AI can process is referred to as the context length or window. ChatGPT-4, for example, has a context window of 32,000 tokens. That’s about 24,000 words in English.

- GPT-1 (2018) had a context length of **512** tokens.
- GPT-2 (2019) supported 1,024.
- GPT-3 (2020) supported 2,048.
- GPT-3.5 (2022) supported 4,096
- GPT-4 (2023) first supported 8,192. Then 16,384. Then 32,768. Now, it supports up to **128,000** tokens.

That’s about **96**,000 words in English.
About 192 text pages (single-spaced, 500 words per page)

Quelle: https://mattrickard.com/the-context-length-observation

Kontextfenster (Context Length)

- Gemini 1.5 Pro bietet ein Kontextfenster von 2 Millionen Tokens
  - Quelle: https://ai.google.dev/gemini-api/docs/long-context?hl=de
  - = 1,5 Millionen Worte **= 300.000 Textseiten**

- DeepSeek V3
  - **32,000 tokens**
- DeepSeek R1
  - **128,000 tokens (192 Textseiten)**