

# Eigene Sprachmodelle

**Nutzung und Feintuning**

**Christian Winkler, datanizing GmbH  
KI Navigator, Workshop**

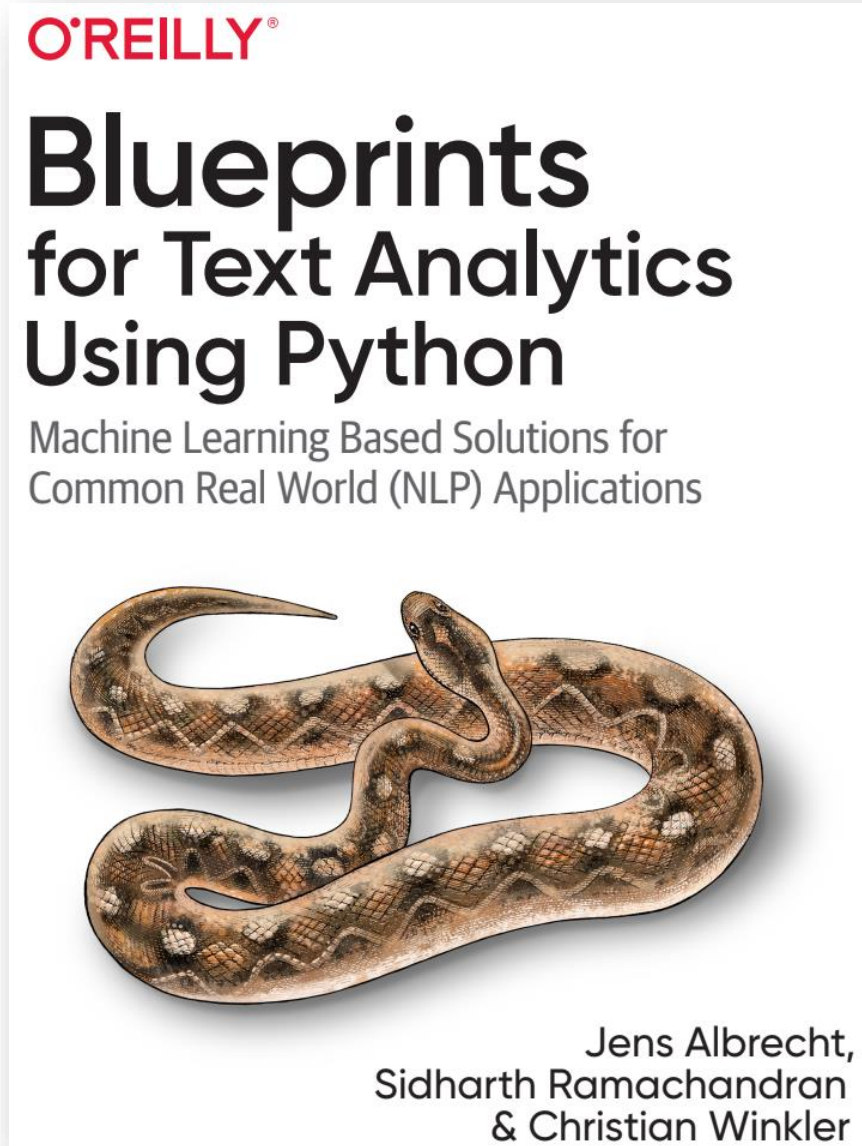
<https://github.com/datanizing/ki-navigator>



# Vorstellung und Ziele



Christian Winkler



[christian.winkler@datanizing.com](mailto:christian.winkler@datanizing.com)

[christian.winkler@th-nuernberg.de](mailto:christian.winkler@th-nuernberg.de)

# Agenda

1. Einführung und Motivation
2. Sprachmodelle: BERT
3. Nutzung von BERT: Semantische Ähnlichkeit
4. Feintuning von BERT
5. Generative Modelle
6. Modelle quantisieren
7. Feintuning von LLMs
8. Frontends für LLMs
9. Zusammenfassung und Ausblick
10. Feedback



# **Einführung und Motivation**

## Bisher bei der Textanalyse...

### Vektorisierung mit Wörtern als Features

- Reihenfolge nicht beachtet
- Zusammenhänge nicht betrachtet
- Semantik geht verloren

### Vektorisierung mit N-Grammen als Features

- Reihenfolge beachtet
- Zusammenhänge in Form von Tupeln
- Abstraktion in Form von Semantik fehlt

Worte jeweils  
einzelne  
Entitäten

Kontext  
entscheidet über  
Semantik!

# Wiederholung – die Document-Term-Matrix

## Textdaten müssen vektorisiert werden

- Tokenisieren und Stoppworte eliminieren
- Vokabular bestimmen
- Wörter in Dokumenten zählen

## Optimierung

- TF/IDF

## Unzulänglichkeiten

- Reihenfolge
- Semantik

	looking	cheap	flight	where	should	stay	thanks	answer	nearest	train	station	car	airport
Looking <b>for</b> cheap flight?	1	1	1	0	0	0	0	0	0	0	0	0	0
Where should <b>I</b> stay?	0	0	0	1	1	1	0	0	0	0	0	0	0
Thanks <b>for your</b> answer	0	0	0	0	0	0	1	1	0	0	0	0	0
Nearest train station	0	0	0	0	0	0	0	0	1	1	1	0	0
Looking <b>for a</b> car	1	0	0	0	0	0	0	0	0	0	0	1	0
Train <b>to</b> airport	0	0	0	0	0	0	0	0	0	1	0	0	1

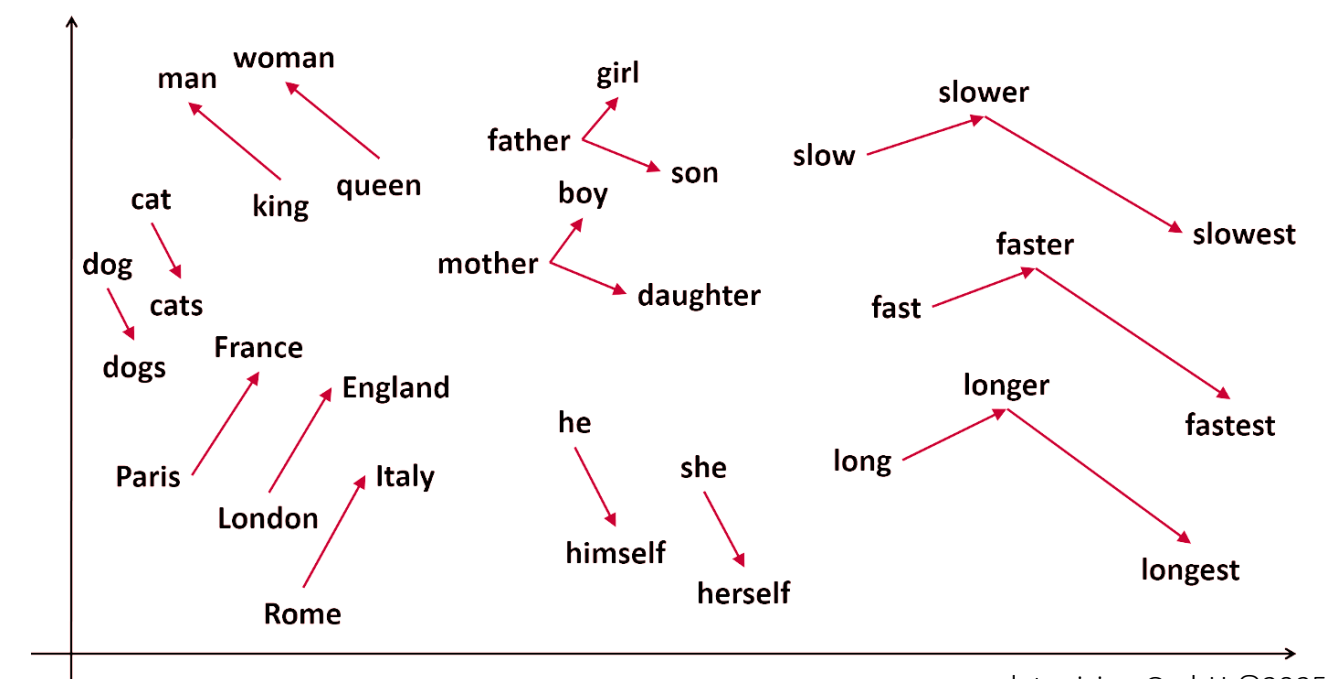
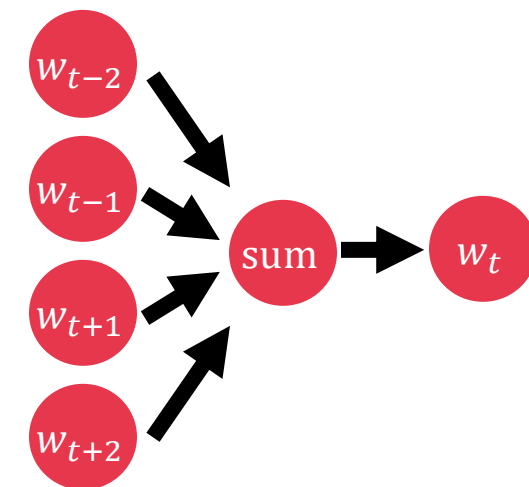
# Verbesserung: Wortmodelle wie word2vec

"You shall know a word by the company it keeps."

Beispiel: Was ist "tezgüino"? Was ähnelt "tezgüino"?

A bottle of \_\_\_\_\_ is on the table.  
Everybody likes \_\_\_\_\_.  
Don't have \_\_\_\_\_ before you drive.  
We make \_\_\_\_\_ out of corn.

EISENSTEIN, JACOB: *Natural Language Processing*. Georgia Tech, 2018. Ch. 14







# Sprachmodelle: BERT

# Grundfunktionsweise BERT: Transformer-Modell mit Attention

BERT basiert auf einem sog. Transformer-Modell

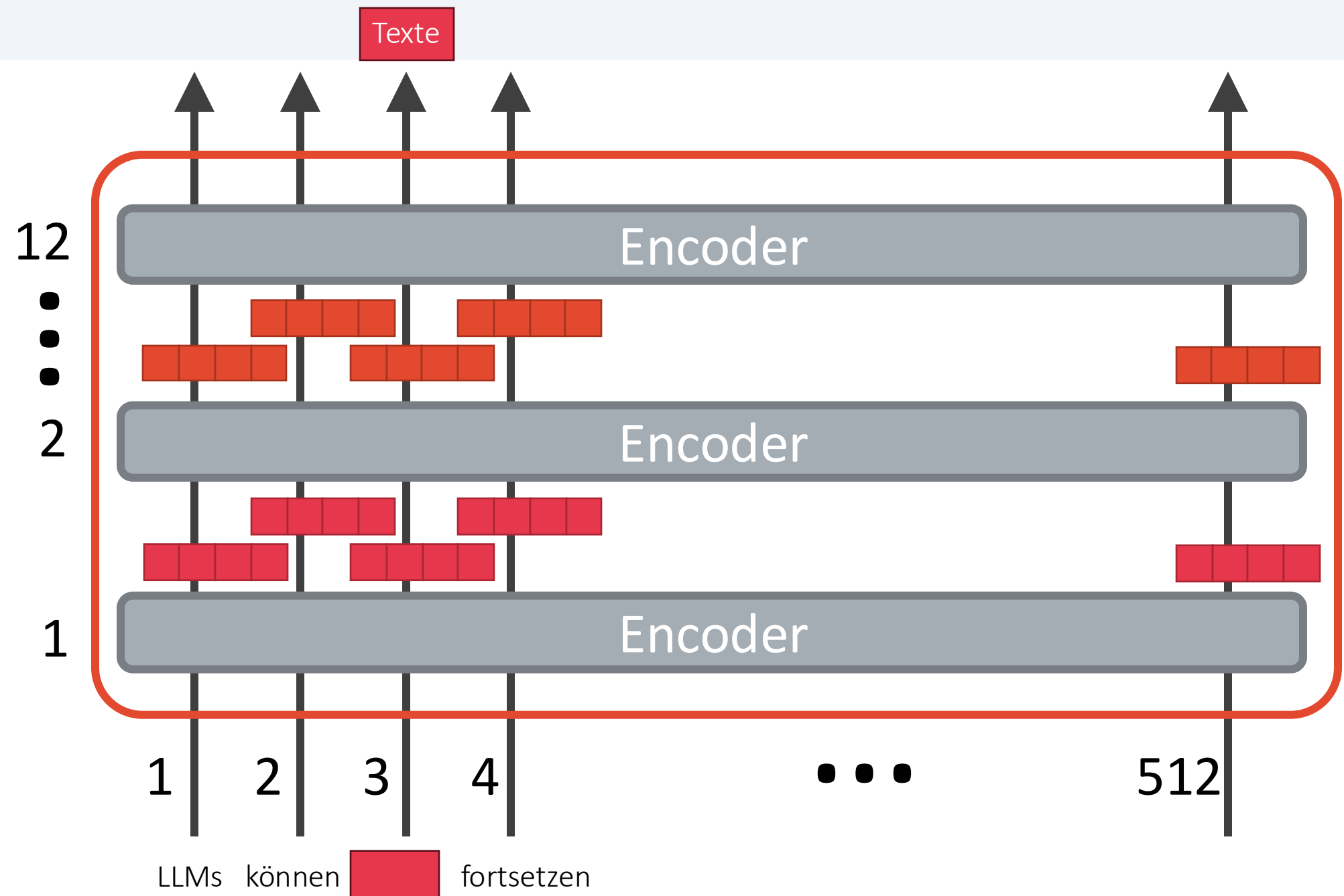
- Vorher verwendet für Übersetzungen
- Sog seq2seq-Verfahren

BERT nutzt nur den Encoder

- Encoder wird kaskadiert
- Unterschiedliche Layer modellieren die Kontextualisierung

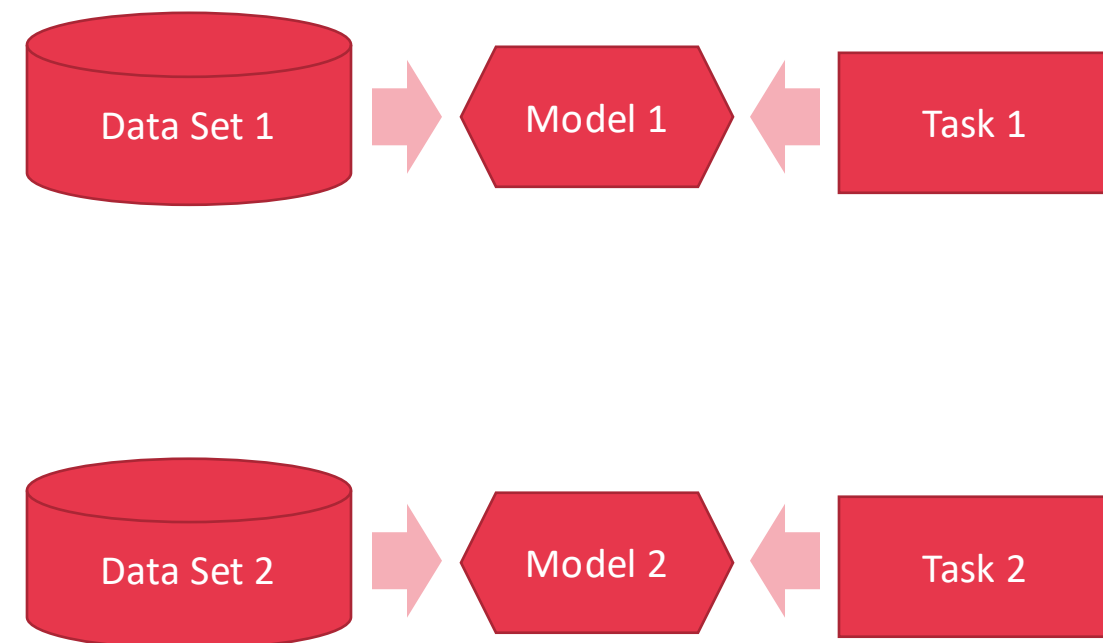
BERT ist ein sehr komplexes Modell

- Base: 12 Layer, 12 Attention Heads, 110 Millionen Parameter
- Large: 24 Layer, 16 Attention Heads, 340 Millionen Parameter
- Sehr, sehr aufwändiges Training  
➔ Transfer Learning



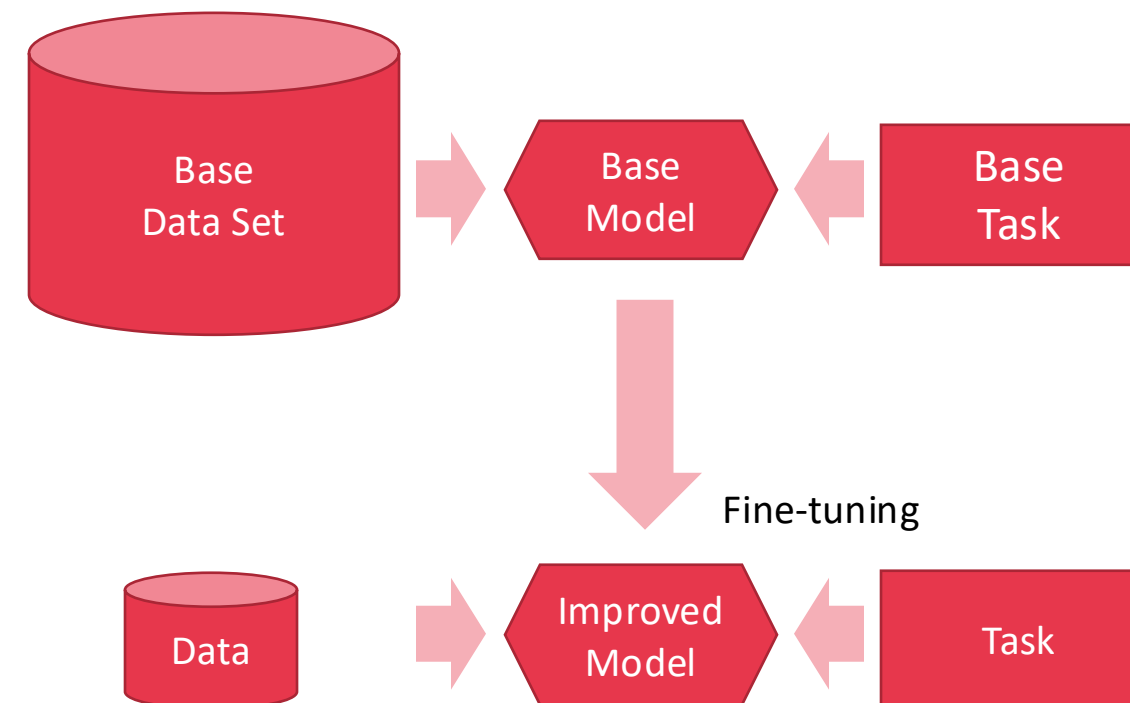
# Funktionsweise von Transfer Learning

## Klassisches ML



Ein Modell wird für genau eine Aufgabe bei Null beginnend überwacht trainiert. Es werden immer viele Trainingsdaten benötigt.

## Transfer Learning



Ein Basis-Modell, dass auf einem großen Datensatz (unüberwacht!) trainiert wurde, wird mit wenig Daten auf eine spezifische Aufgabe angepasst.

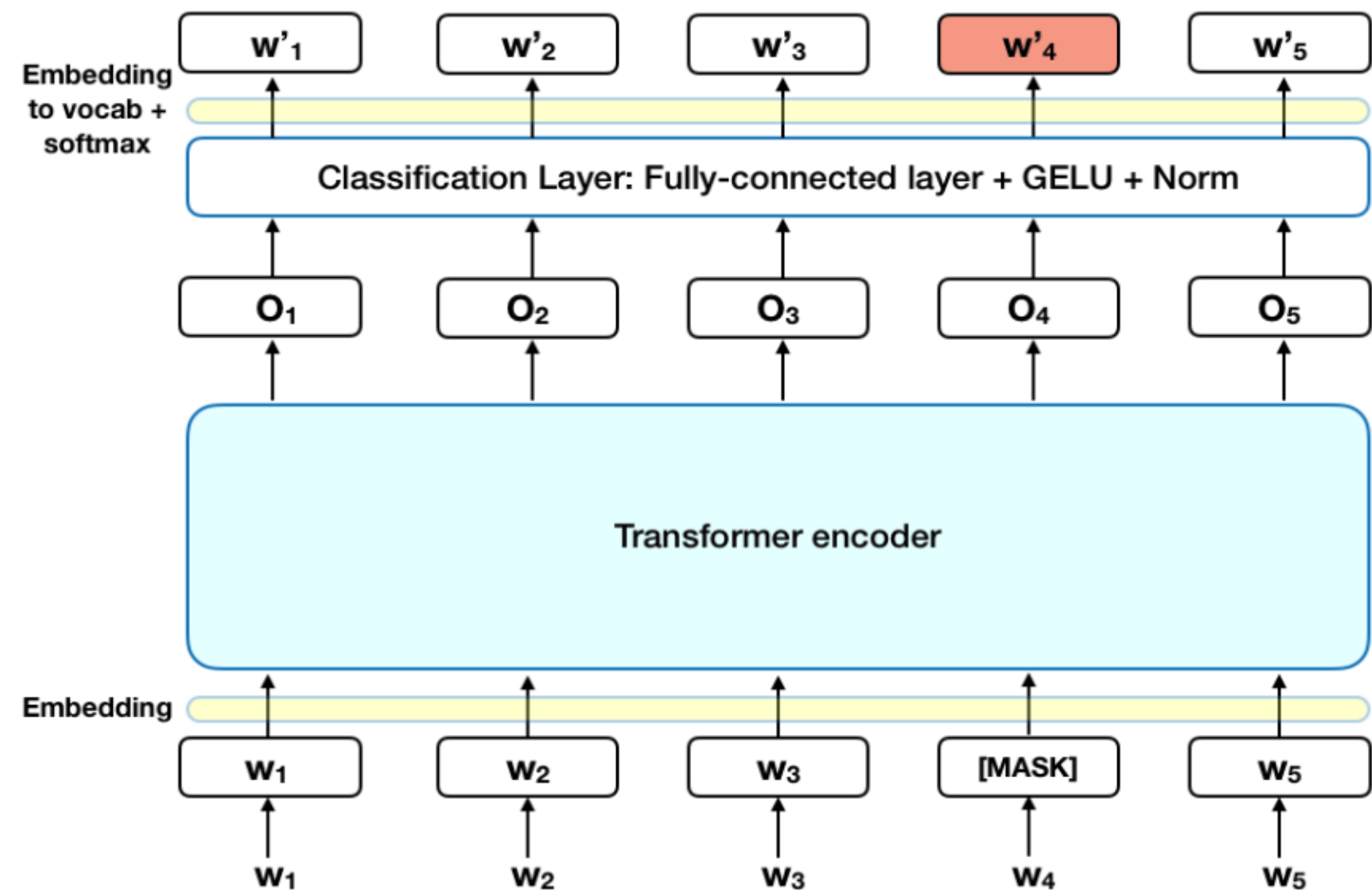
# Training auf Vorhersage "versteckter" Worte (Maskierung)

Ich **hebe Geld** bei der **Bank** ab.

Auf der **Bank** **hebe** ich **Geld** ab.

Problem: Sprache ist vorwärts und rückwärts kontextualisiert

- Mögliche Lösung durch direktionales Modell
- Bei BERT durch Transformer und Masking



Quelle: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>



# Was sind *eigene Sprachmodelle*?

## Kein Cloud-Zwang

- Ablauffähig auf lokalen Computern („on premise“)
- Parameter liegen vor
- GPU ist oft hilfreich, aber keinesfalls Zwang

## Vorteile: Freiheit und mehr Möglichkeiten

- Datenschutz und Datensicherheit
- Experimente möglich
- Anpassung an eigene Bedürfnisse
- Finetuning



# **Nutzung von BERT: Maskierung und Klassifikation**





# **Anwendung von BERT: Semantische Ähnlichkeit**



# Ist BERT dafür ausreichend?

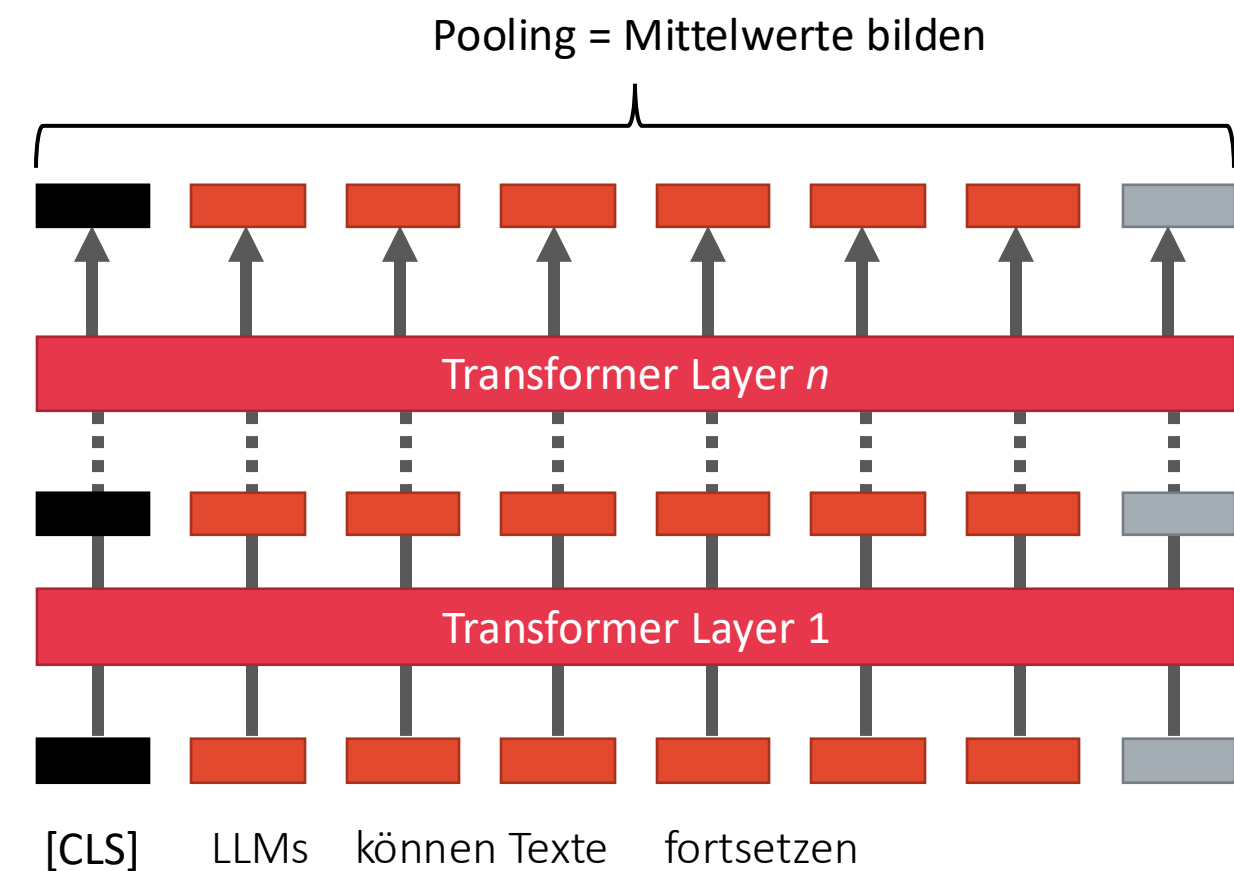
Fast, aber noch nicht ganz

Modell ist darauf angepasst, fehlende Wörter zu erraten

Für Ähnlichkeit wird ein etwas modifiziertes Modell benötigt

Zum Glück ist das bereits verfügbar

Modell heißt SBERT (<https://sbert.net>) und kommt sogar aus Deutschland



# Ablauf

Daten laden

Satz-Fragmente berechnen

Embedding für jeden Satz berechnen

Embeddings speichern

Sätze speichern

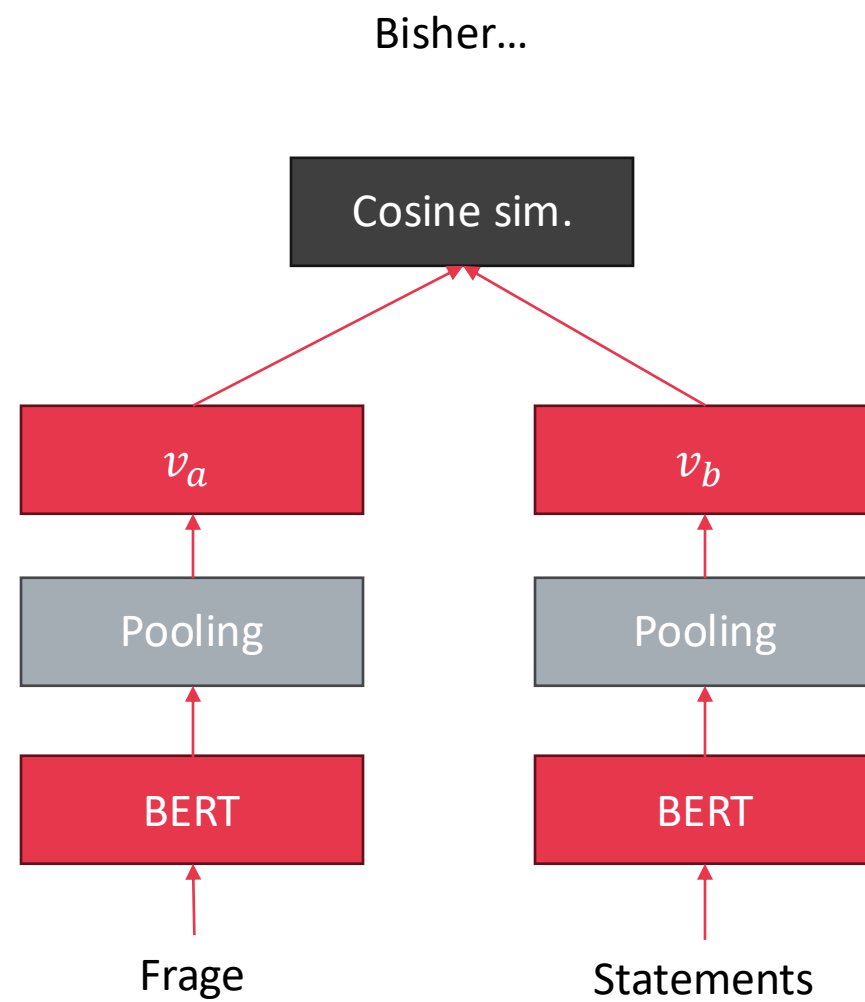
Search-Embedding berechnen

Ähnlichste Sätze finden

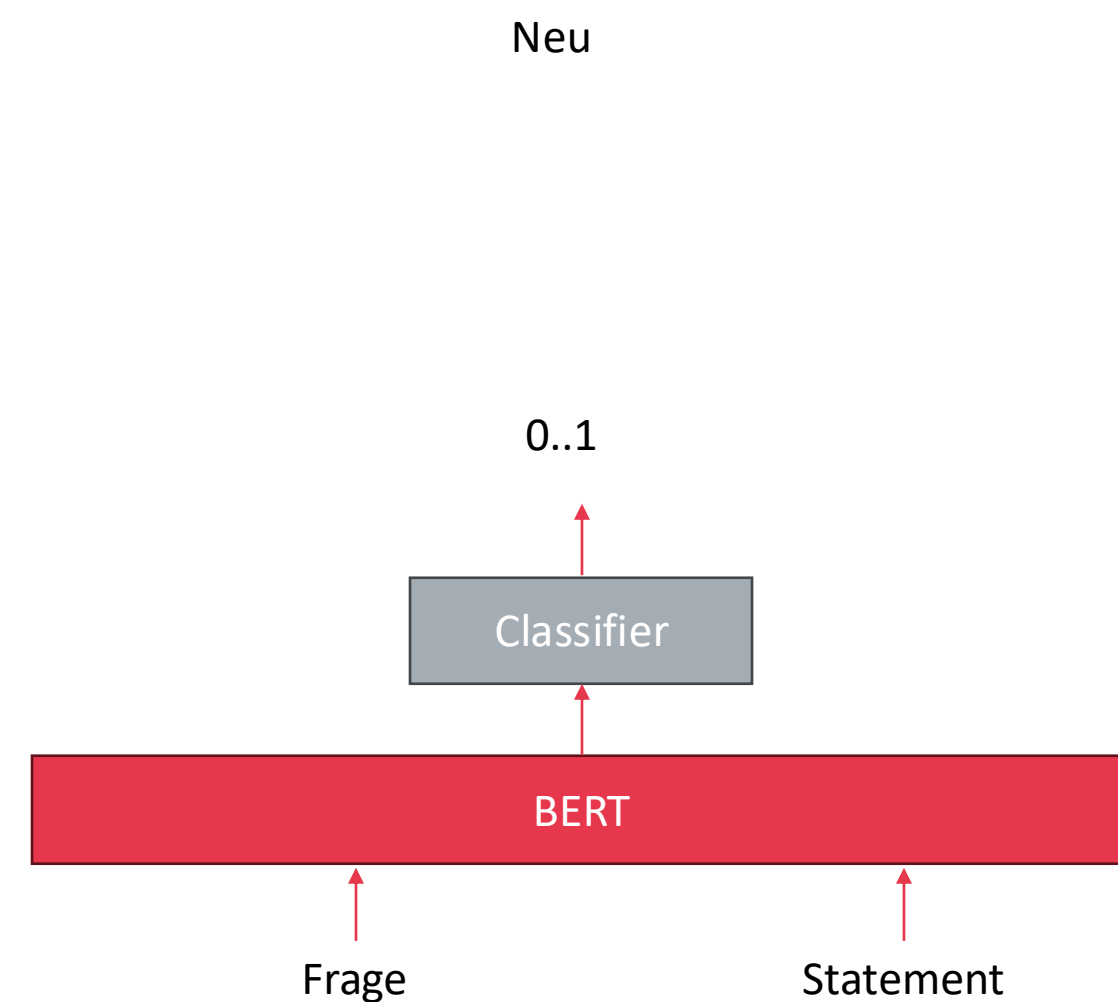
Nach Ähnlichkeit sortieren



# Optimierung durch Cross-Encoder



**Bi-encoder**



**Cross-encoder**



# Nutzung zum Information Retrieval

## Beste Ergebnisse für Ähnlichkeiten von Sätzen

- Codierung des Satzes (z.B. der Frage) notwendig **inkl. gleichzeitiger Codierung** der Referenz (des Ergebnisses)
- Sehr viele Codierungen
- Sehr gut, aber sehr langsam

## Grundidee der Sentence Transformers

- Unabhängig Codierung
- Nutzung des Ähnlichkeitsmaßes

## Weitere Optimierung durch Cross-Encoder

- Vorher Verkleinerung der Ergebnismenge
- Wartezeit verkürzen

Tausende Aussagen

→ Tausende Codierungen für jede Suche notwendig (!)

Tausende Codierungen einmal notwendig

→ Für jede Suche eine weitere Codierung

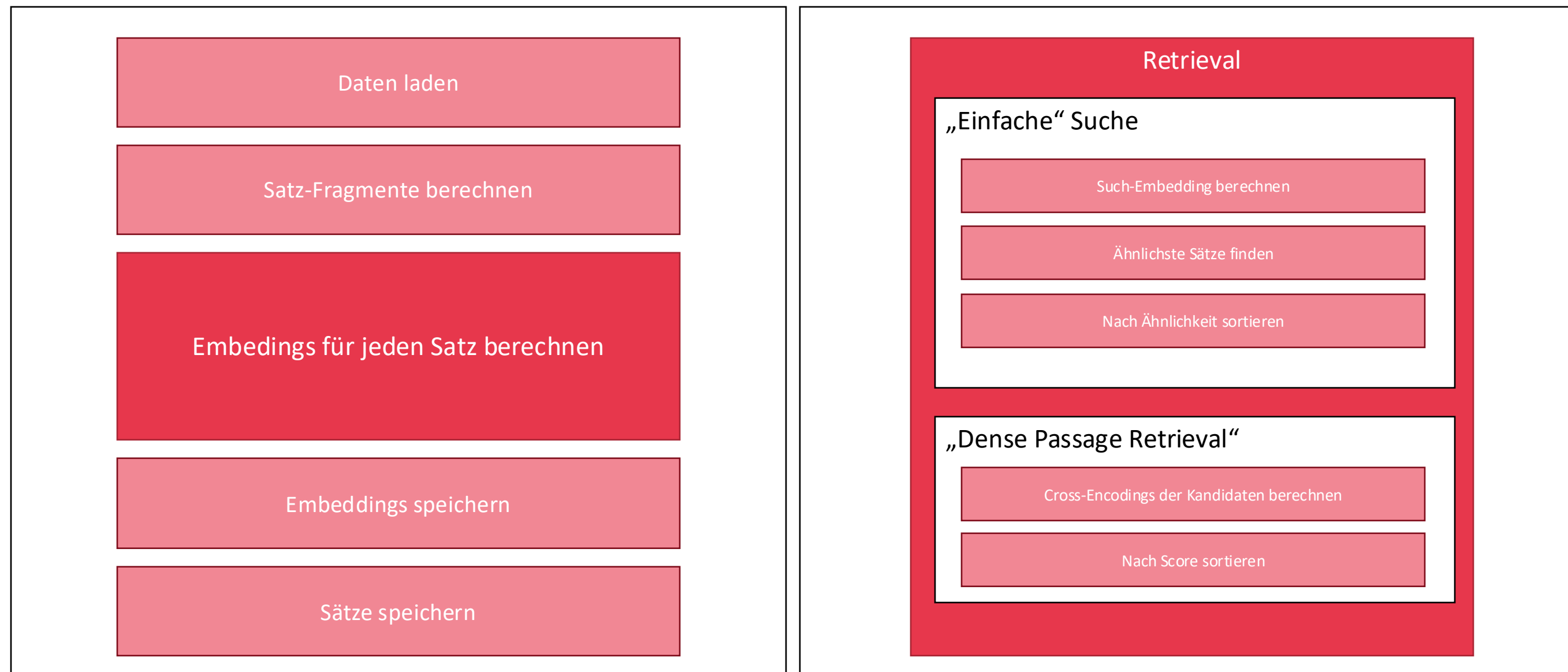
Tausende Codierungen einmal notwendig

→ Für jede Suche eine weitere Codierung

→ Für den Filter weitere 200 Codierungen

→ Deutlich verbesserte Ergebnisse

# Verbesserung der bisherigen Lösung







# Feintuning von BERT



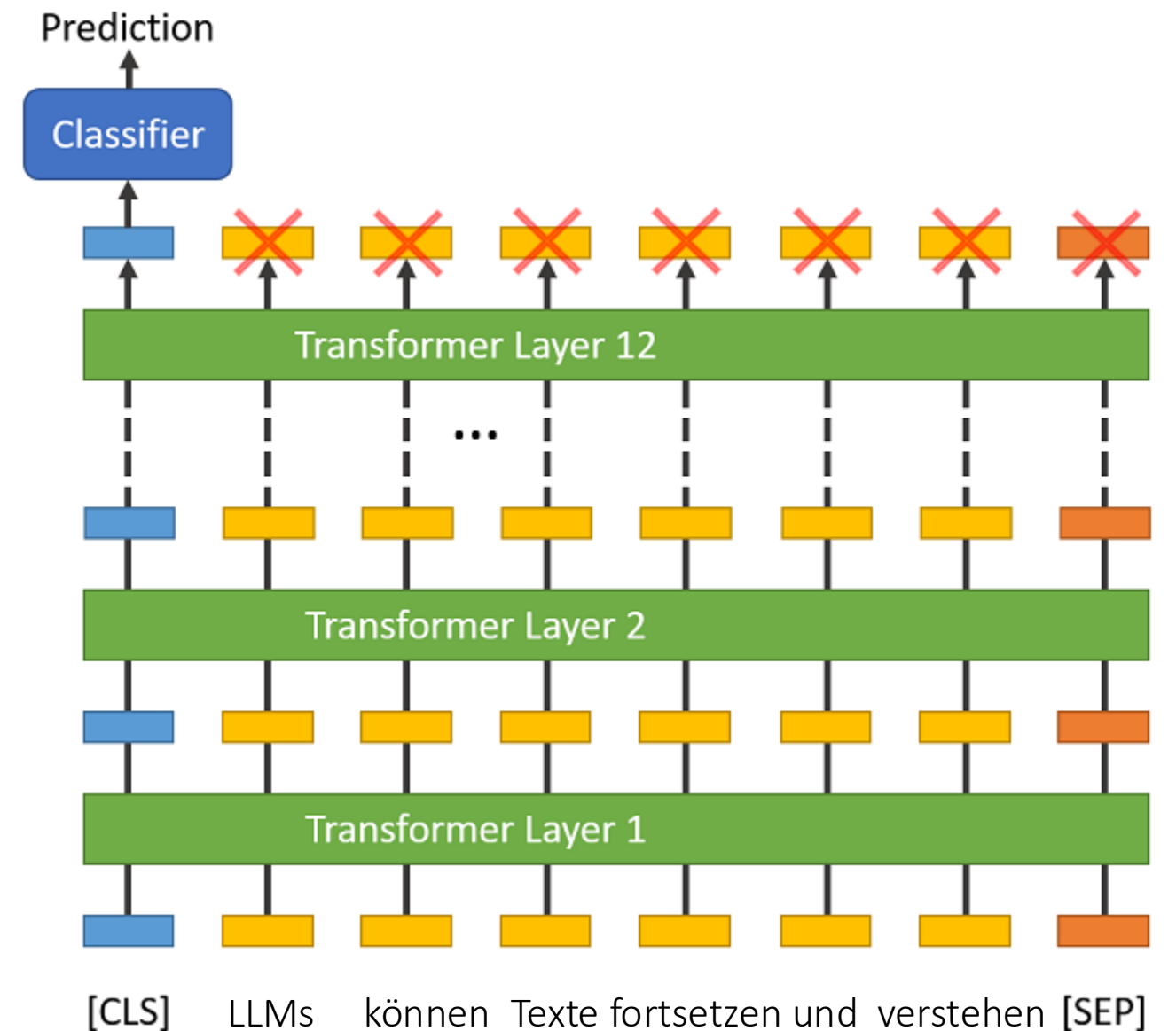
# Finetuning für Klassifikationsaufgaben

Grober Ablauf des Trainingsprozesses:

- Embedding der gesamten Sätze berechnen
- Berechnete Embeddings sind hoch kontextualisiert
- Sehr viel (alle?) Information steckt daher im kontextualisierten Embedding des ersten Tokens (der gar nicht zum Satz gehört)
- Embedding-Vektor von [CLS] wird durch Classifier verarbeitet
- Iteration mit Anpassung der Gewichte im letzten Layer

Klassifikationsprozess:

- Kontextualisierung des gesamten Satzes
- Klassifikation nur mit (kontextualisiertem) [CLS]



Quelle: <https://mccormickml.com/2019/07/22/BERT-fine-tuning/>

Mai 2020

Classic 7-Tage-News Archiv

2020 Mai

Sonntag, 31.05.2020

- 19:06 **Apple Airplay: Musik von iPhone und Co. im Netzwerk streamen**  
techstage
- 18:06 **Die Highlights bei Netflix, Disney+ und Amazon Prime Video im Juni 2020**  
100
- 17:32 **Vor 20 Jahren: Die Absage der CeBIT Home 2000 ebnet den Weg zur Games Convention**  
2
- 17:18 **"Willkommen zurück im Weltraum": Raumfahrer aus USA an ISS angekommen**  
115
- 16:40 **Youtuber zeigt Akten in altem Krankenhaus – Polizei ermittelt**  
699
- 16:00 **Apple schließt kritische Lücke in Anmeldedienst "Sign in with Apple"**  
48
- 14:24 **Slots: Rettungspaket für Lufthansa nimmt wichtige Hürde**  
115
- 11:52 **"Albtraum-Szenario": ACLU verklagt Clearview wegen illegaler Gesichtserkennung**  
92
- 11:36 **Konjunkturpaket: Scheuer meldet Investitionsbedarf für Verkehr und 5G an**  
54
- 08:05 **Missing Link: Vom Siegeszug des Webprotokolls – und resultierenden Problemen**  
146
- 04:34 **USA feiern SpaceX-Start als "Tribut an die Führerschaft Trumps"**  
350
- 00:09 **Was war. Was wird. Von Dekreten und technischen Diktaten.**  
4W 69

# Unser Use Case

## Heise Newsticker

“Instanz” für Tech- und Computer-News im deutschsprachigen Raum

"Willk

115

YouTu

699

Apple

48

## Optimierung

Welche Artikel werden besonders gern kommentiert?

Höhere Reichweite

Mehr Umsatz

# Ablauf: Finetuning von BERT

Pytorch vorbereiten (CPU/GPU)

Daten einlesen

Tokenisierung

[CLS], [SEP]

Input IDs

Attention  
Masks

Training/Validation-Split

Modell laden

Training in Epochen

Trainingsschritt

Vorhersage Trainingsdaten

Berechnung Loss

Rückwärts-Auswertung

Gewichte anpassen (AdamW)

Validierungsschritt

Vorhersage Validierungsdaten

Berechnung Accuracy und Loss



# Finetuning eigener Embedding-Modelle

## Warum sollte man das machen?

- (Bessere) Abbildung domänenspezifischen Vokabulars
- Bessere Abbildung von Ähnlichkeiten

## Wie geht das?

- Erzeugung von ähnlichen (oder weniger ähnlichen) Textfragmenten
- Annotation mit Score
- Finetuning mit BERT oder SBERT

## Ist das nicht extrem aufwändig?

- Text lässt sich generativ erzeugen, Verbesserungen relativ schnell erkennbar
- Mit GPU keine langen Wartezeiten





# Finetuning eigener Cross-Encoder

## Warum sollte man das machen?

- (Noch) bessere Abbildung von Ähnlichkeiten
- Besseres Ranking

## Wie geht das?

- Nutzung der gleichen Daten wie beim Finetuning der Embeddings
- Finetuning erfolgt ebenso mit BERT oder SBERT

## Ist das nicht extrem aufwändig?

- Vorteil: Ähnlichkeitsdaten schon vorhanden
- Mit GPU leicht zu bewältigen



# Generative Modelle

# Grundfunktionsweise: Transformer-Modell mit Attention

GPT basiert auf einem sog. Transformer-Modell

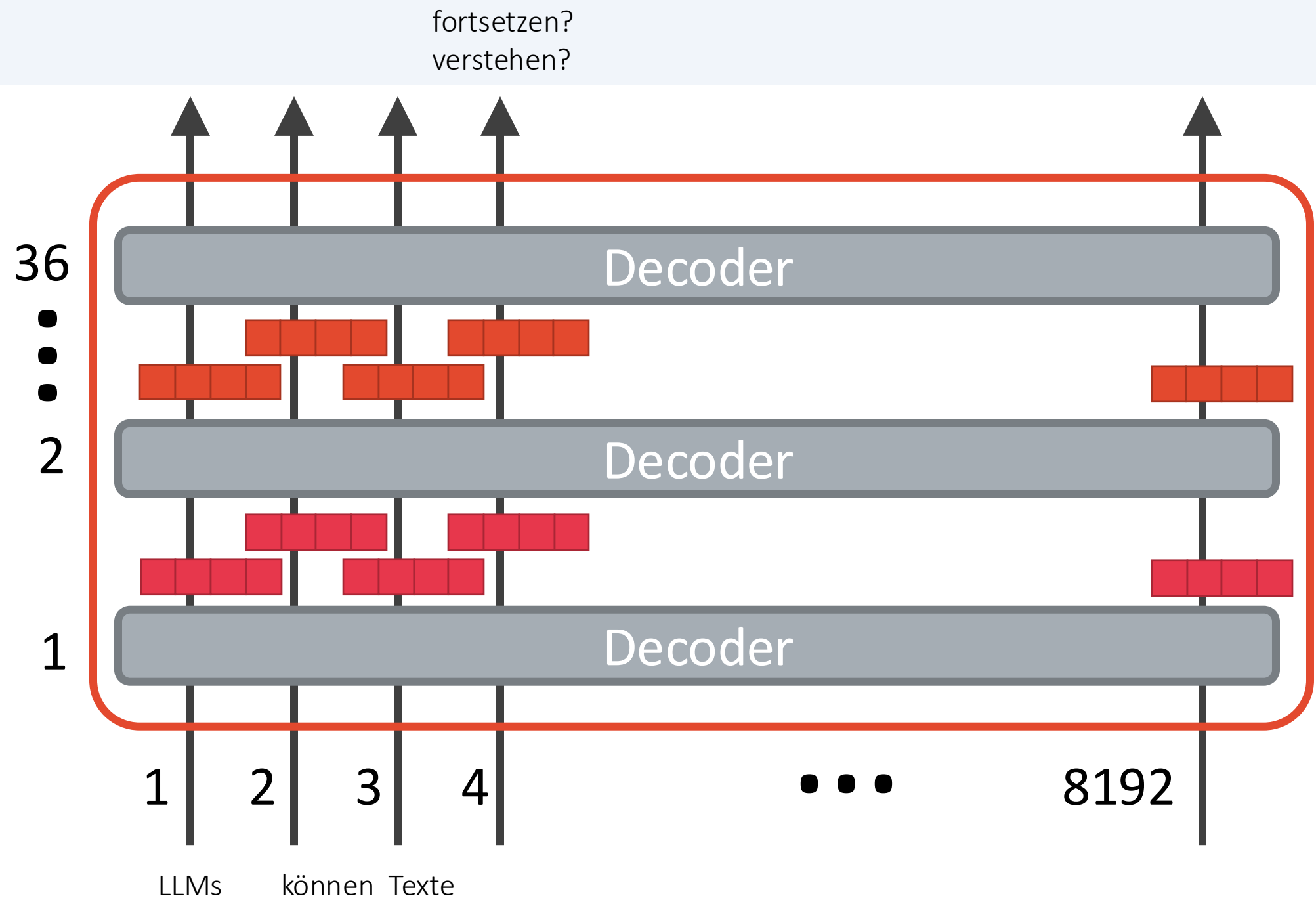
- Vorher verwendet für Übersetzungen
- Sog seq2seq-Verfahren (Encoder/Decoder)

GPT nutzt nur den Decoder

- Decoder wird kaskadiert
- Unterschiedliche Layer modellieren die Kontextualisierung (nur nach vorne)
- Training auf Vorhersage des nächsten Wortes (autoregressiv)

GPT hat sehr viele Parameter

- Ständiges Wachstum
- Training noch viel aufwändiger als bei BERT
- Kann nur mit extremem Hardware-Aufwand bewältigt werden



Wachst

# GPT-4

# Wachstum der Sprachmodelle

## Unendliches Wachstum?

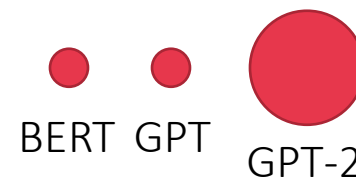
- Rechenkapazität wächst immer weiter und schneller
- Trainingsmenge (Text) in nahezu beliebiger Menge verfügbar (aber vielleicht irgendwann auch ausgeschöpft)
- Geld offenbar gar kein Problem

## Aufgaben werden immer schwieriger

- Von Sentiment-Analyse zu wissenschaftlichen Arbeiten

## Aber: wenig strukturell neue Ideen

- Dünn besetzte Modelle (sparse)
- Distillation (besonders bei BERT, jetzt auch bei GPT)



GPT-3

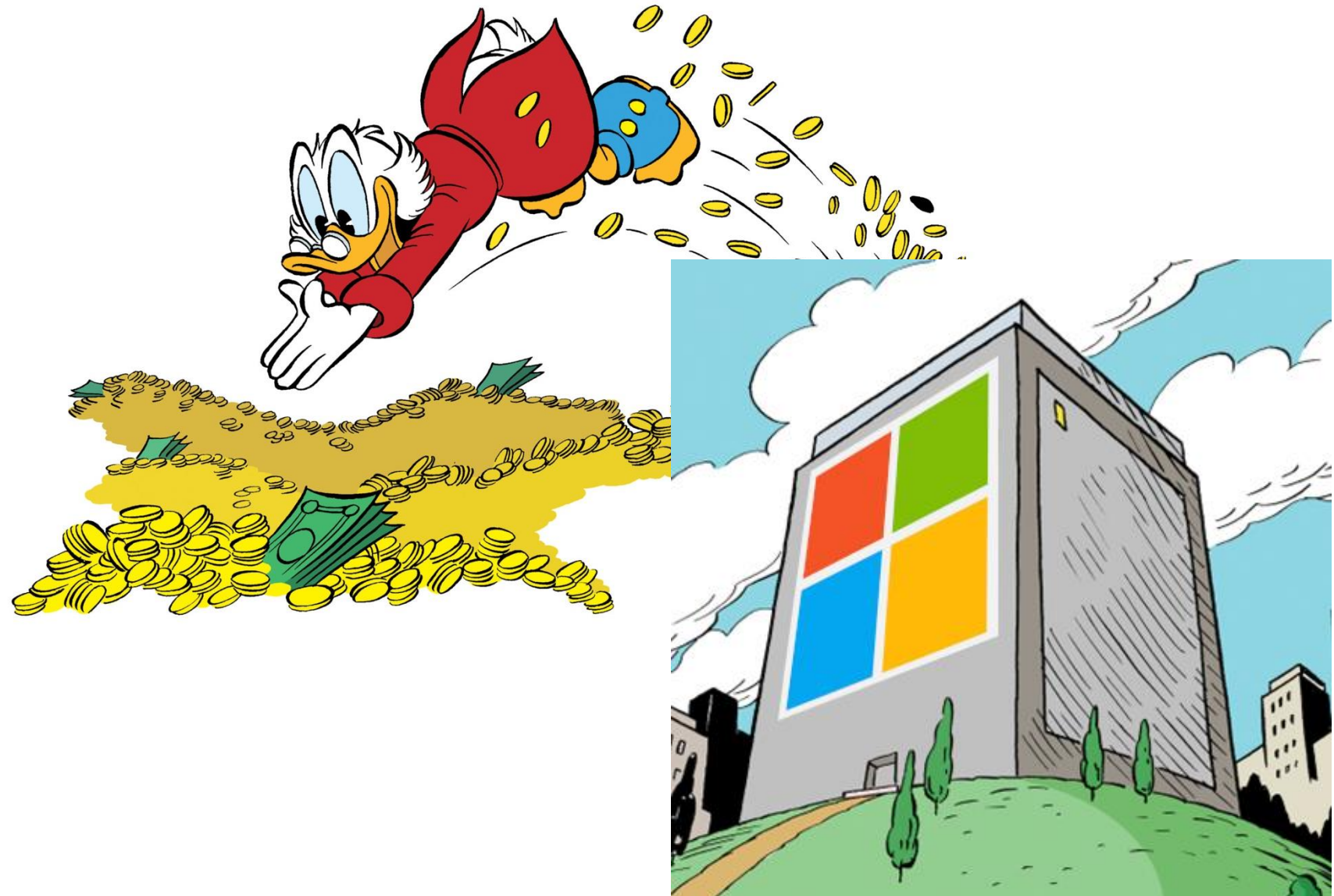
# Welche Rechenleistung wird für ChatGPT benötigt und was kostet das?

## Training:

- Sehr, sehr aufwändig
- Billionen von Token
- Kosten für GPT-4 angeblich über 100 Millionen Dollar<sup>1</sup>

## Betrieb:

- Auch dafür werden Grafikkarten benötigt
- Bei ChatGPT wohl mehr als 700.000 Dollar pro Tag<sup>2</sup>  
(alter Wert, heute mehrere 10 Millionen Dollar pro Tag)
- Überwachung durch Menschen (mit minimalem Lohn!) notwendig<sup>3</sup>



<sup>1</sup> <https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/>

<sup>2</sup> <https://www.businessinsider.com/how-much-chatgpt-costs-openai-to-run-estimate-report-2023-4>

<sup>3</sup> <https://www.nbcnews.com/tech/innovation/openai-chatgpt-ai-jobs-contractors-talk-shadow-workforce-powers-rcna81892>



# Fragen beantworten durch Instruction Following

## Basis-Training

- LLMs können Wörter vorhersagen
- Sehr gut zur Fortsetzung von Texten
- Enorm aufwändiges Basis-Training

## Beantworten von Fragen?

- Finetuning
- Super-Struktur hinzufügen (Templates)
- Oftmals kombiniert mit Reinforcement Learning (RLHF mit Reward-Funktion)
- Bestimmte Antworten werden bevorzugt

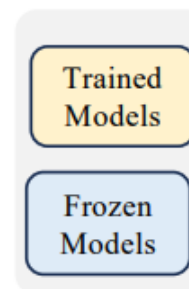
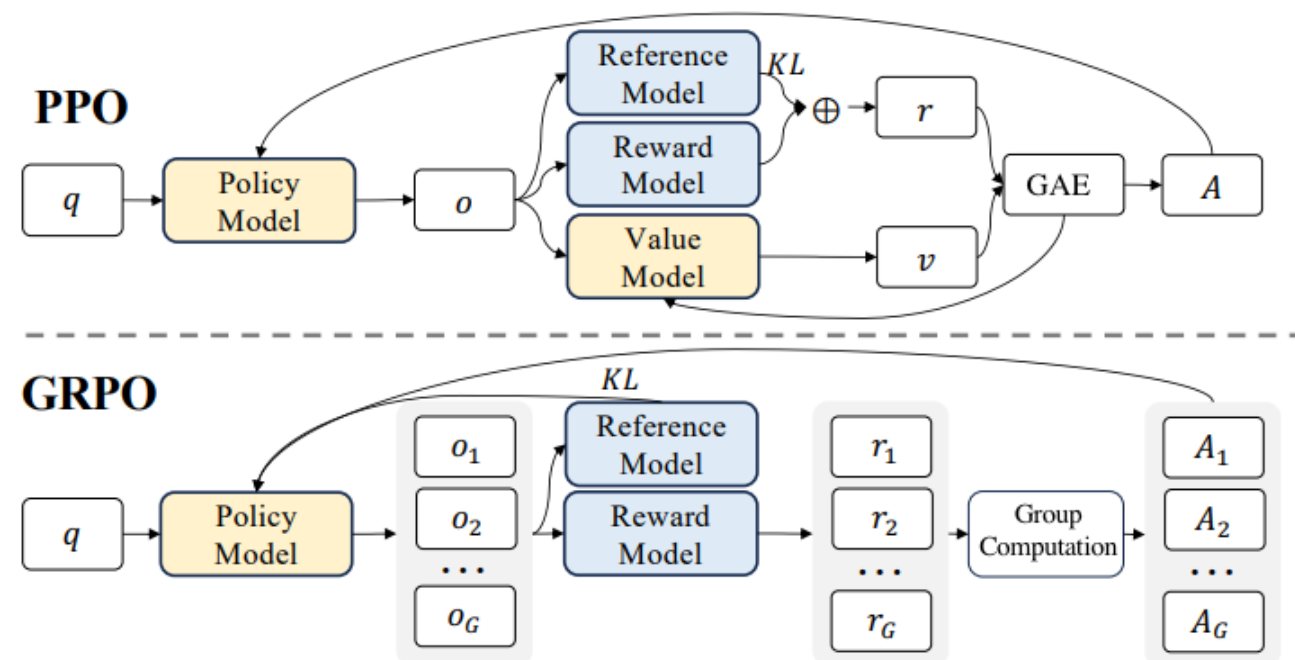
```
</system/>
You are a helpful travel assistant.</end/>
</user/>
I am going to Paris, what should I see?</end/>
</assistant/>
Paris, the capital of France, is known for its stunning
architecture, art museums, historical landmarks, and romantic
atmosphere. Here are some of the top attractions to see in
Paris:\n\n1. The Eiffel Tower: The iconic Eiffel Tower is one
of the most recognizable landmarks in the world and offers
breathtaking views of the city.\n2. The Louvre Museum: The
Louvre is one of the world's largest and most famous museums,
housing an impressive collection of art and artifacts,
including the Mona Lisa.\n3. Notre-Dame Cathedral: This
beautiful cathedral is one of the most famous landmarks in
Paris and is known for its Gothic architecture and stunning
stained glass windows.\n\nThese are just a few of the many
attractions that Paris has to offer. With so much to see and
do, it's no wonder that Paris is one of the most popular
tourist destinations in the world."</end/>
```

# Innovationen bei generativen Modellen durch DeepSeek

## Reinforcement Learning

- Schon vorher verwendet (RLHF beim SFT).
- Jetzt schon in einem früheren Stadium (vor SFT, R1 Zero)

## Group Relative Policy Optimization (GRPO)



## Multi-Head Latent Attention und Multi-Token Prediction

## Low-Level Optimierungen (PTX) wegen Hardware-Einschränkungen

- Exporte nicht erlaubt, daher nur wenige GPUs vorhanden.
- Training wird effizienter und günstiger.
- Inferenz kann viel günstiger angeboten werden (\$ pro Token).

## Gewichte als FP8

- Spart 50% VRAM und verdoppelt (damit) Inferenz-Geschwindigkeit.
- Trainingsprozess wird (einmalig) komplizierter.

## Architektur: Mixture of Experts mit Auxiliary-Loss-Free Load Balancing





# Modelle quantisieren

# Präzision von Fließkommazahlen

## Bisher häufig verwendet

- Double precision: 64 bit → 8 bytes
- Single precision: 32 bit → 4 bytes

## Besonders häufig bei LLMs

- 16 bit → 2 bytes (bfloat16)
- 8 bit bleiben für Exponent, aber nur 7 bit für Mantisse
- Präzision wird zugunsten des Wertebereichs geopfert

## Für ein 70B Modell braucht man 140 GB RAM

- Kann man das weiter reduzieren?
- Präzision auf 8 bit reduzieren (int8) mit **bitsandbytes**
- Funktioniert, aber muss jedesmal berechnet werden (Speicher!)
- LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, 2022

Type	Bits			
	Sign	Exponent	Significand	Total
<u>Half (IEEE 754-2008)</u>	1	5	10	16
<u>Single</u>	1	8	23	32
<u>Double</u>	1	11	52	64
<u>x86 extended precision</u>	1	15	64	80
<u>Quad</u>	1	15	112	128

Source: [https://en.wikipedia.org/wiki/Floating-point\\_arithmetic](https://en.wikipedia.org/wiki/Floating-point_arithmetic)

# Platz sparen mit GPTQ

## Kann man den RAM-Bedarf weiter reduzieren?

- Ja, durch weitere Reduzierung der Genauigkeit (z.B. zu int4)
- Funktioniert sehr gut für die *Evaluation* (aber nicht für das Training)
- Veröffentlichung in 2023:  
[GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers](#)
- Algorithmus ist viel schneller als alles Andere zuvor

## Existierende Implementierungen zur Modell-Evaluation

- In Hugging Face Transformer integriert:  
<https://huggingface.co/blog/gptq-integration>
- Auch als separate Software nutzbar: [llama.cpp](#)

## Bessere Quantisierung

- Idee: Dynamisch für unterschiedliche Layer
- AWQ: Activation-aware quantization
  - Viele Modelle bereits verfügbar
  - Braucht viel VRAM (Layer umschreiben)
- ExLlamaV2: mittlere Anzahl von Bits
  - Quantisierung muss oft selbst durchgeführt werden
  - Benötigt Datenset zum Abgleich
  - Sehr schnelle Inferenz mit geringem RAM-Bedarf
- llama.cpp: Ausführung auf CPU (gguf-Format)

## Fertige Software für Inferenz

- TGI
- vLLM
- SGLang
- tabbyAPI







# Feintuning von LLMs

# Werkzeugkasten

## Grundsätzliche Problematik

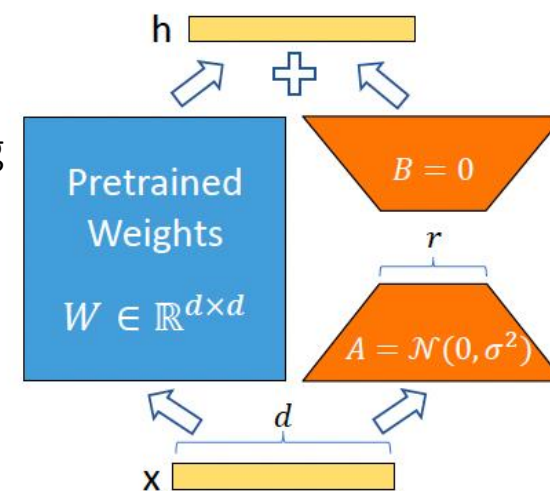
- Modelle haben sehr (zu) viele Parameter
- Training, aber auch Finetuning sehr aufwändig
- Ziel: Anzahl der Parameter reduzieren

## LoRA: Low Rank Adaptation

- Reduktion der trainierbaren Parameter
- “Einfrieren” der Originalgewichte
- Darstellung der (veränderten) Gewichte durch ein Produkt zweier niedrigdimensionaler Matrizen
- Sehr effizient und oft besser als “vollständiges” Feintuning

## PEFT – “Parameter Efficient Tuning”

- Bibliothek von Hugging Face
- Praktisch überall verwendet
- Ermöglicht Feintuning auf “Consumer Hardware” (A100)

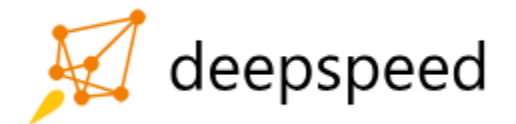


## QLoRA

- Feintuning eines bereits quantisierten Modells
- Weitere Optimierung mit LoRA
- Deutlich verringerter Speicherbedarf

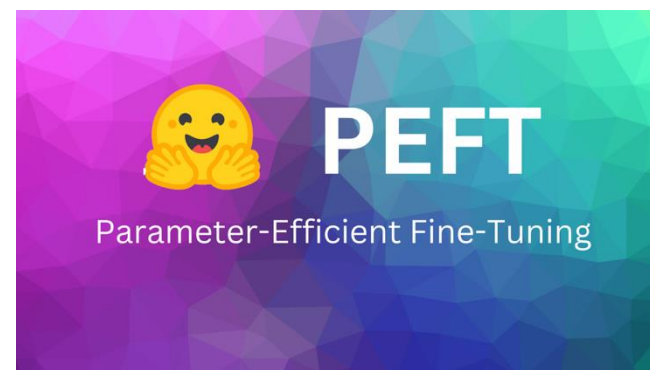
## Deep Speed

- Optimierung für Instruction Following (Reinforcement Learning with Human Feedback, RLHF)
- Sehr hohe Beschleunigung in der Lernphase



## Accelerate

- Verteilte Ausführung



# Modelle, Datensätze und Trainingssoftware

## Beispiele mit kleinen Modellen

- Qwen3-0.6B
- Llama-3.2-1B
- SmolLM von Hugging Face
- Phi-3.5

## Datensätze

- Häufig Instruction Following
- Aber nicht nur, auch Fortsetzung
- Kommt auf den Anwendungsfall an
- Unterschied ist nur in der Konstruktion der Texte zum Training

## Umgebung

- Nicht immer ganz einfach mit Docker
- Spezielle Anpassungen für CUDA notwendig

## Selbst schreiben

- Sehr viele Optionen, teilweise auch modellspezifisch
- Gibt die besten Einblicke

### unsloth

- Besonders effizient, unterstützt QLoRA
- Jupyter-Notebooks verfügbar

### axolotl

- Viele Modelle unterstützt und wird ständig erweitert
- Notebook oder CLI
- Nicht ganz einfach zu installieren (Enironment verwenden!)



# Ist eigenes Finetuning generativer Modelle sinnvoll?

## Training ist aufwändig

- Trainingsdaten werden benötigt
- Braucht GPUs
- Braucht Rechenzeit

## Woher kommen die Trainingsdaten?

- Unternehmensdaten (alte Dokumente)
- Offene Daten (wie z.B. Publikationen)

## Qualitätsindikatoren

- Definieren
- Sonst „Blindflug“

## Alternative

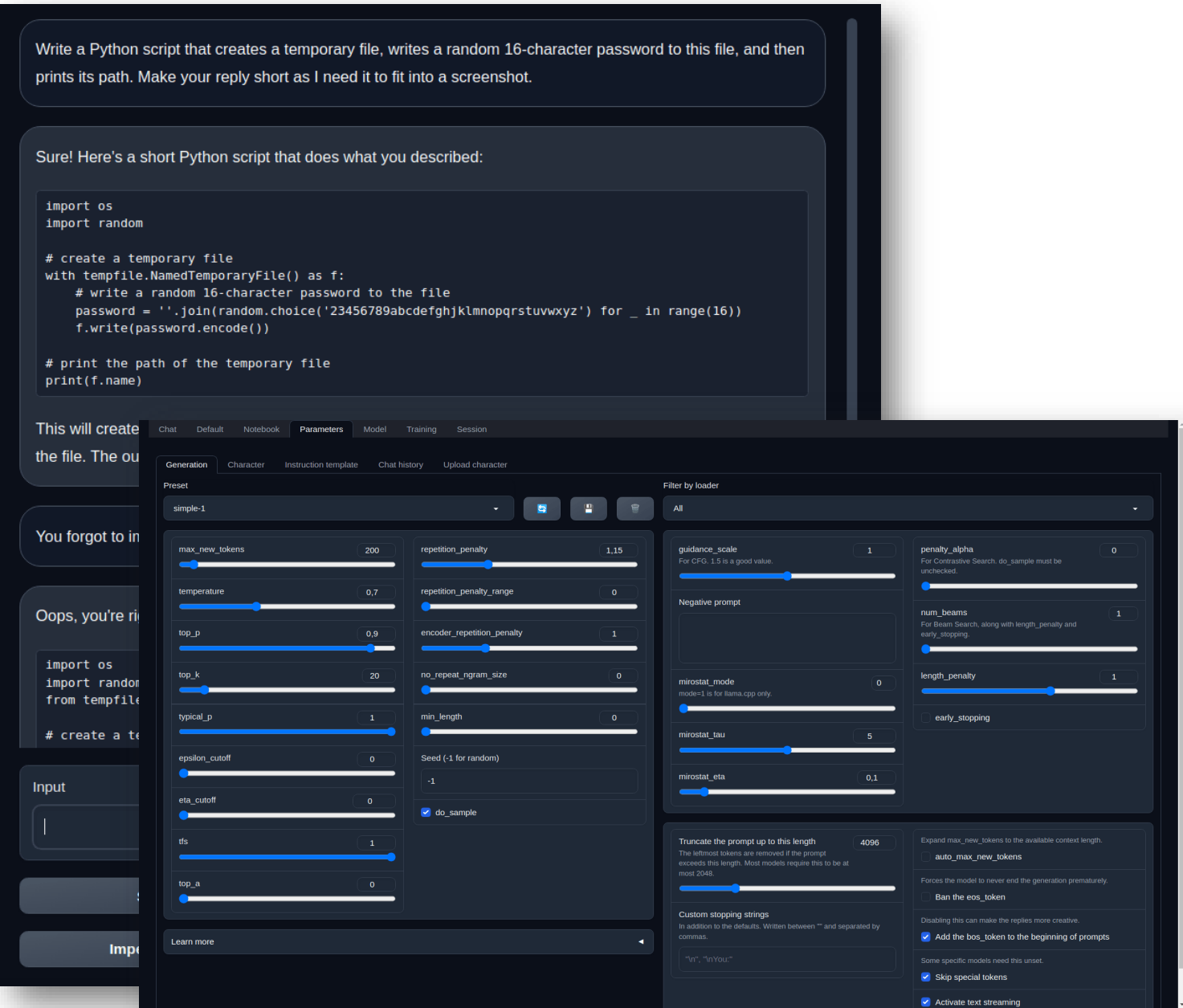
- Nutzung existierender Modelle und LoRas
- Richtige Auswahl der Basismodelle





# Frontends für LLMs

# Die Qual der Wahl...



## llama.cpp

**User:** Explain Heise Zeitschriftenverlag

**Llama:** Heise Zeitschriftenverlag is a German publishing company that specializes in computer science and technology magazines. The company was founded in 1975 by Klaus Heise and his brother Hans-Peter Heise. Today, it is one of the largest publishers of technical literature in Germany, with over 200 publications covering topics such as software development, hardware engineering, network administration, and more. Some of their well-known titles include Computer Bild, c't, and Chip.

Say something...

Send Stop Reset

157ms per token, 6.35 tokens per second  
Powered by [llama.cpp](#) and [ggml.ai](#).

# Systematischer Überblick

	llama.cpp	koboldcpp	Text generation web UI	ollama	Open WebUI	LM Studio
Frontend	CLI, simple web	sophisticated web	Sophisticated web	CLI	sophisticated web	Own
Backend	llama.cpp	llama.cpp	various	llama.cpp	ollama, OpenAI API	llama.cpp, mlx
API	Python, OpenAI	OpenAI	OpenAI	Ollama, OpenAI	own	OpenAI
Models	unmanaged	unmanaged	unmanaged	managed	ollama, OpenAI API	managed

Notable mention: mistral.rs (CLI), Silly Tavern (Roleplay), tabbyAPI (Exllamav2)



# **Zusammenfassung und Ausblick**



# Neue und vielversprechende Technologien

## Reasoning

- „Denkende“ Modelle
- Exponieren den Antwort-Prozess

## GRPO

- Bereits bei DeepSeek erwähnt
- Standard für Reasoning-Modelle

## Attention Interleaving

- Berechnung und insbesondere KV-Cache ist aufwändig
- Abwechselnd Window und Global Attention
- Aktuell in der Forschung

## Kontext-Kompression

- Alternative zur Einsparung KV-Cache
- Fortschritte durch DeepSeek-OCR (bzw. Architektur)

## Mixture of Experts

- Viele neue Modelle nutzen diese Architektur
- Weniger Berechnungen pro Inferenz nötig

## Quantisierung

- Inferenzgeschwindigkeit ist limitiert durch RAM-Bandbreite
- Reduktion der Genauigkeit
- Bis vier Bit fast keine Auswirkungen (MXFP4)

# Embedding und Reranking-Modellen: Basis für RAG

## Information Retrieval mit Embedding Modellen

- Berechnung der Vektoren
- Semantisches Retrieval (Vektor-Datenbank)
- Optimiert für Fragen/Antworten/Sprachen
- Immer größere Kontexte möglich

## Reranking Modelle

- Verbessern das Matching
- Erlaubt Verwendung verschiedener Modelle
- Sehr großer Hebel, aber langsam

Rank (Bo...	Model	Zero-shot	Memory U...	Number of P...	Embedding D...	Max Tokens
1	<a href="#">llama-embed-nemotron-8b</a>	99%	28629	7B	4096	32768
2	<a href="#">gemini-embedding-001</a>	99%	Unknown	Unknown	3072	2048
3	<a href="#">Qwen3-Embedding-8B</a>	99%	28866	7B	4096	32768
4	<a href="#">Qwen3-Embedding-4B</a>	99%	15341	4B	2560	32768
5	<a href="#">Qwen3-Embedding-0.6B</a>	99%	2272	595M	1024	32768
6	<a href="#">gte-Qwen2-7B-instruct</a>	⚠ NA	29040	7B	3584	32768
7	<a href="#">Linq-Embed-Mistral</a>	99%	13563	7B	4096	32768
8	<a href="#">multilingual-e5-large-instruct</a>	99%	1068	560M	1024	514
9	<a href="#">embeddinggemma-300m</a>	99%	578	307M	768	2048
10	<a href="#">SFR-Embedding-Mistral</a>	96%	13563	7B	4096	32768

<https://huggingface.co/spaces/mteb/leaderboard>



# Leistungsfähige offene generative Modelle

## DeepSeek

- Nach nur neun Monaten ein Klassiker
- Sehr leistungsfähig und sehr groß



## Kimi K2

- Nutzt die DeepSeek-Architektur
- Noch größer



## Qwen3

- Sehr viele Modelle in unterschiedlichsten Größen inkl. VL
- Neue Architektur mit Qwen3-Next
- Vision Language und Image Editing



## GLM 4.6

- Sehr gutes, moderat großes Modell
- Warten auf GLM 4.6 Air (4.5 Air war sehr gut)



## Ring/Ling

- Eher unbekannte Modelle, sehr gute Benchmarks
- Sehr groß (1T Parameter)



## LiquidAI

- Neue, effiziente Architektur
- Stefan Kühn weiß mehr



## OpenGPT



# Boom bei OCR-Modellen

## OlmOCR

- Offenes Modell mit offenen Trainingsdaten



## Paddle-OCR

- Offenes Modell von Baidu (ERNIE)
- Formeln, Tabellen, Diagramme



## DeepSeek-OCR

- Neue Architektur mit Vision Tokens
- Formeln, Tabelle, Diagramme, Strukturformeln



## Docling

- Nutzt Granite-Modelle von IBM



## Nanonets-OCR2

- Semantisches Tagging



## MinerU

- Pipelines und Workflows



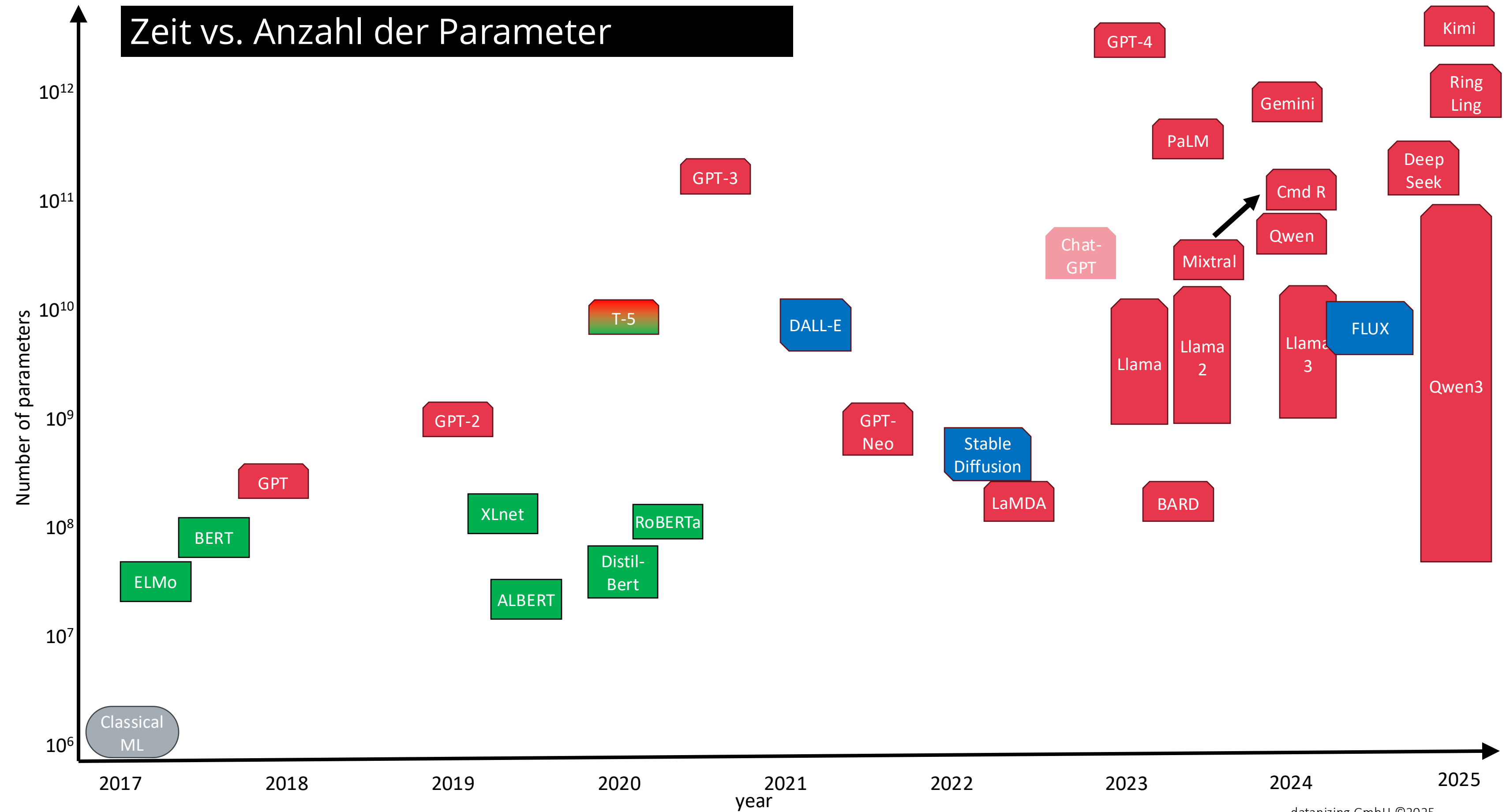
## Dots.ocr

- Formeln

dots.ocr



# Zeit vs. Anzahl der Parameter



# Aktuelle Entwicklungen

## RAG

- Retrieval Augmented Generation

## Spezialmodelle

- CodeLlama, Llemma, geometische Modelle etc.

## Performance von Open Weight-Modellen

- Mistral, Mixtral, Qwen, DeepSeek, Kimi K2

## Neue Methoden

- HQQ (Half quadratic quantization)
- Groq, Cerebras: Extrem schnelle Inferenz
- GRPO und Andere DeepSeek-Innovationen



Quelle: [https://www.linkedin.com/posts/maxime-labonne\\_arena-elo-graph-updated-with-new-models-activity-7187062633735368705-u2jB](https://www.linkedin.com/posts/maxime-labonne_arena-elo-graph-updated-with-new-models-activity-7187062633735368705-u2jB)



# Feedback