M05a - LLM und Transformer

Stand: 03.2025

1 Large Language Models

LLMs funktionieren im Wesentlichen durch die Vorhersage des wahrscheinlichsten nächsten Wortes in einem gegebenen Text, basierend auf einer riesigen Menge an Trainingsdaten aus dem Internet. Dieser Prozess wird durch Billiarden von Parametern gesteuert, die während des Trainings optimiert werden. Ein entscheidender Aspekt ist die Architektur von **Transfomern**, die es LLMs erlaubt, den gesamten Text **parallel** zu verarbeiten, anstatt Wort für Wort, und dabei den Kontext durch einen Mechanismus namens **Self Attention** zu berücksichtigen. Das Ergebnis ist ein System, das erstaunlich flüssige und oft sinnvolle Texte generieren kann, obwohl das genaue *Warum* hinter den Vorhersagen aufgrund der Komplexität und Größe des Modells schwer zu verstehen ist.

2 Transfomer

ELI5-Version

Stell dir vor, du liest ein Buch, aber anstatt es Wort für Wort durchzugehen, kannst du gleichzeitig viele Seiten anschauen und genau erkennen, welche Wörter wichtig sind.

So funktioniert ein Transformer-Modell!

Es benutzt eine Technik namens **Self-Attention**, um nicht nur das aktuelle Wort, sondern auch alle anderen relevanten Wörter im Text zu berücksichtigen – egal, wo sie stehen. Das macht es viel besser darin, **Zusammenhänge zu verstehen**, als frühere KI-Modelle, die Texte nur schrittweise gelesen haben.

Dank dieser Technik sind Transformer-Modelle wie **GPT, BERT oder LLaMA** so gut in **Texterstellung, Übersetzungen und sogar Code-Generierung**.

Fazit

Zusammenfassend lässt sich sagen, dass der "Self-Attention"-Mechanismus ein entscheidendes Element von GPT-Modellen ist, das ihnen ein tiefgreifendes Verständnis von Sprache ermöglicht. Durch die Analyse der Beziehungen zwischen Wörtern in einem Satz können diese Modelle den Kontext von Wörtern präzise erfassen und so natürlichere und kohärentere Texte generieren.

Etwas detaillierter ...

Moderne Transformer-Modelle haben die Art und Weise revolutioniert, wie Maschinen Sprache verstehen und erzeugen. Während frühere Modelle wie Recurrent Neural Networks (RNNs) oder Long Short-Term Memory (LSTMs) Schwierigkeiten hatten, lange Texte zu erfassen, ermöglicht die Self-Attention-Mechanik der Transformer eine deutlich effizientere Verarbeitung von großer Textmenge. Diese Weiterentwicklung ist insbesondere für Anwendungen mit langen Dokumenten oder tiefgehender Kontextanalyse entscheidend.

1️⃣ Autoregressive Transformer-Modelle(Textvorhersage Schritt für Schritt)

- **Beispiele:** GPT, GPT-2, GPT-3, GPT-4, LLaMA, Mistral
- **Merkmal:** Die Modelle generieren Text, indem sie Token für Token vorherige Eingaben berücksichtigen (**kausale Self-Attention**). Sie können keine Informationen aus zukünftigen Wörtern im Satz nutzen.
- **Anwendung:** Kreative Textgenerierung, automatische Vervollständigung, interaktive Dialogsysteme.
- **Besonderheiten:** Da diese Modelle nur auf vorherige Tokens zugreifen, eignen sie sich besonders gut für realistische Textgenerierung und kreative Schreibaufgaben, haben aber Schwierigkeiten mit logischem Denken und langfristiger Kohärenz.

2️⃣ Bidirektionale Transformer-Modelle (Kontextbezogenes Sprachverständnis)

- **Beispiele:** BERT, RoBERTa, DeBERTa
- **Merkmal:** Die Modelle analysieren gleichzeitig den gesamten Satz und lernen, Lücken zu füllen (**Masked Language Modeling, MLM**). Dadurch können sie ein tieferes Sprachverständnis aufbauen als autoregressive Modelle.
- **Anwendung:** Textklassifikation, Informationsextraktion, Frage-Antwort-Systeme, Sentiment-Analyse.
- **Besonderheiten:** Da sie nicht für die Textgenerierung optimiert sind, eignen sie sich weniger für das kreative Schreiben, sondern eher für Analyseaufgaben und das Verstehen von Textzusammenhängen.

3️⃣ Seq2Seq (Encoder-Decoder) Transformer-Modelle (Transformation von Texten)

- **Beispiele:** T5, BART, UL2
- **Merkmal:** Bestehen aus zwei Teilen: Ein **Encoder**, der den Eingabetext verarbeitet, und ein **Decoder**, der einen neuen Text generiert. Diese Architektur ermöglicht es, eine Eingabe in eine andere Form zu übertragen.
- **Anwendung:** Maschinelle Übersetzung, Textzusammenfassung, Daten-zu-Text-Konvertierung, automatische Rechtschreibkorrektur.
- **Besonderheiten:** Diese Modelle können sowohl generieren als auch Texte umwandeln, wodurch sie sehr vielseitig einsetzbar sind. Insbesondere T5 wurde darauf optimiert, nahezu jede NLP-Aufgabe in eine "Text-zu-Text"-Problematik umzuwandeln

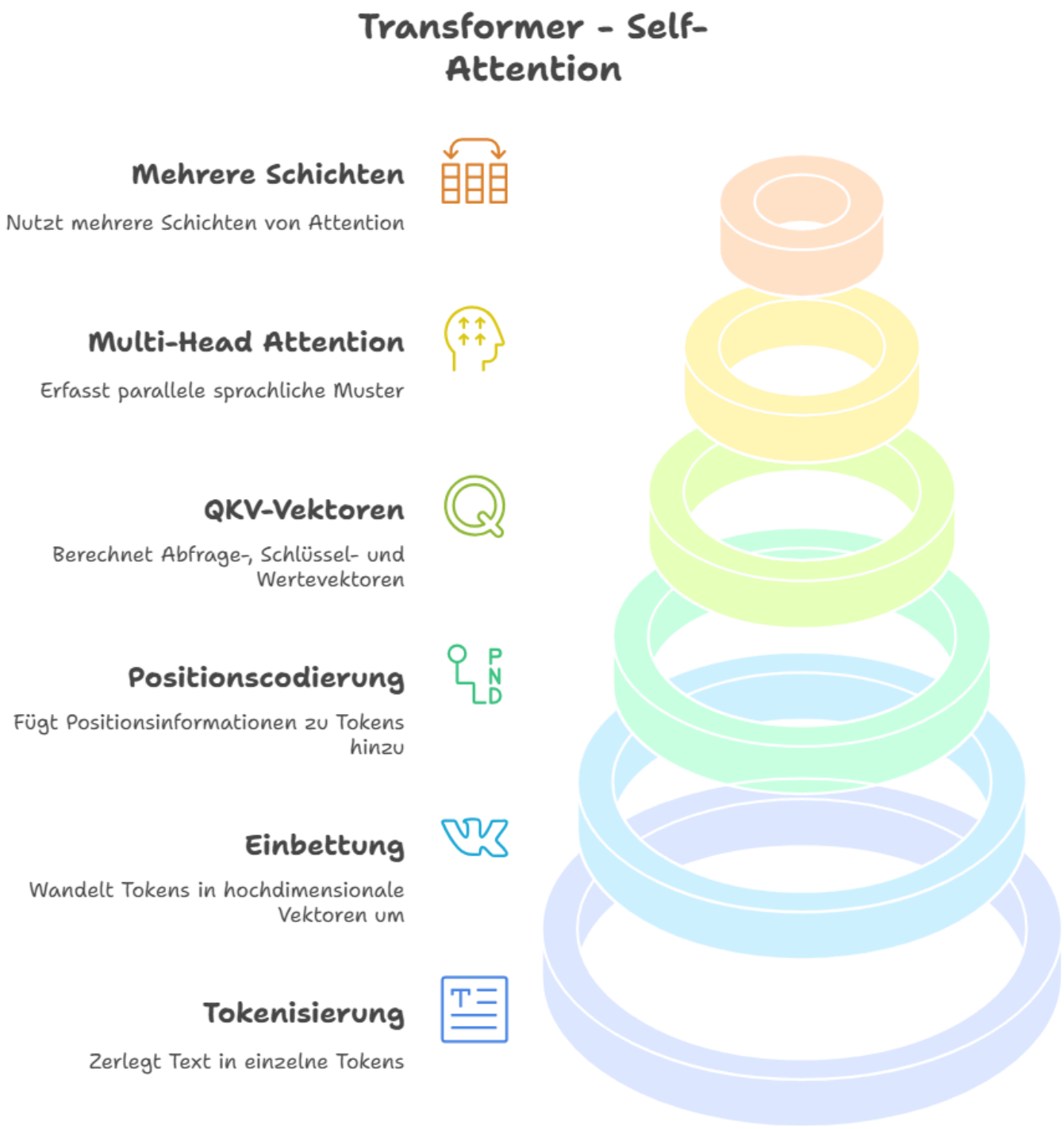
4️⃣ Diffusionsbasierte Text-Modelle (Alternative zur Transformer-Architektur)

- **Beispiele:** Experimentelle KI-Modelle wie **DIFFUSER**
- **Merkmal:** Statt Texte autoregressiv zu erzeugen, rekonstruieren diese Modelle Texte aus verrauschten Sequenzen. Sie basieren auf der Idee von Diffusionsmodellen, die bisher vor allem für Bilder genutzt wurden.
- **Anwendung:** Noch in der Forschung, potenziell für kreative Textgenerierung interessant. Diese Modelle könnten langfristig eine Alternative zu Transformer-Modellen bieten, wenn sie weiter optimiert werden.
- **Besonderheiten:** Der Ansatz ist vielversprechend, aber noch nicht weit verbreitet. Es gibt erste Versuche, Diffusionsmodelle für das Generieren von Texten einzusetzen, jedoch sind sie bislang nicht so effizient und präzise wie Transformer-basierte Modelle.

Fazit:

> **📝 Fazit**
>
> Transformer-Modelle lassen sich in **autoregessive, bidirektionale und Encoder-Decoder-Modelle** unterteilen, während Diffusionsmodelle einen experimentellen Ansatz für die Textgenerierung darstellen. Die Wahl des richtigen Modells hängt stark von der jeweiligen Anwendung ab. Während autoregressive Transformer sich besonders gut für das Generieren von Text eignen, werden bidirektionale Modelle vor allem für Analyseaufgaben genutzt. Encoder-Decoder-Modelle hingegen sind die erste Wahl, wenn es darum geht, einen Text in eine andere Form zu überführen, wie es beispielsweise bei der maschinellen Übersetzung der Fall ist. Diffusionsmodelle könnten in Zukunft eine alternative Technik bieten, sind aber derzeit noch nicht weit genug entwickelt.

3 Self-Attention



3.1 Tokenisierung und Einbettung

Bevor der Attention-Mechanismus angewendet werden kann, durchläuft ein Text zwei wichtige Vorverarbeitungsschritte:

1. **Tokenisierung**: Der Eingabetext wird in einzelne Tokens zerlegt
   - Unser Beispiel "Der Hund jagt die Katze" wird in 5 Tokens aufgeteilt
   - Je nach Tokenizer können Wörter auch in Subwort-Tokens zerlegt werden

2. **Token-Einbettung**: Jedes Token wird in einen hochdimensionalen Vektor umgewandelt
   - In BERT haben diese Vektoren typischerweise 768 Dimensionen
   - In GPT-Modellen können es 1024, 2048 oder mehr Dimensionen sein
   - Diese Einbettungen enthalten kontextlose Repräsentationen der Tokens
3. **Positionscodierung**: Da Transformer keine inhärente Reihenfolge kennen, werden Positionsinformationen hinzugefügt
   - Jedes Token erhält Informationen zu seiner Position im Satz
   - Dies wird zur Token-Einbettung addiert

Für unser Beispiel "Der Hund jagt die Katze" entsteht dadurch eine Matrix X mit der Dimension 5×768, wobei jede Zeile einen Token-Vektor repräsentiert, der sowohl semantische als auch Positionsinformationen enthält.

3.2 Query (Q), Key (K), Value (V) Vektoren

Für jedes Token werden basierend auf seiner Einbettung drei verschiedene Vektoren berechnet:

- **Query (Q)**: Sucht nach relevanten Informationen
- **Key (K)**: Dient als Schlüssel für die Suche
- **Value (V)**: Enthält die eigentlichen Informationen

Die Berechnung erfolgt durch lineare Transformationen:

```
Q = X * W_Q
K = X * W_K
V = X * W_V
```

Dabei ist:

- X: Die Eingabematrix mit Token-Embeddings
- W_Q, W_K, W_V: Trainierbare Gewichtungsmatrizen

Diese Gewichtungsmatrizen sind nicht zufällig, sondern werden in vortrainierten Modellen wie **BERT, GPT, T5 oder RoBERTa** bereits optimiert bereitgestellt. Diese Modelle wurden mit großen Textmengen trainiert und enthalten fertige Gewichtungen, die direkt genutzt oder für spezifische Anwendungsfälle feinjustiert (Fine-Tuning) werden können.

Mit Hilfe von Bibliotheken wie **Hugging Face Transformers** können diese Modelle geladen und direkt eingesetzt werden:

```
from transformers import AutoModel, AutoTokenizer

model_name = "bert-base-uncased"
model = AutoModel.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

Damit stehen optimierte Gewichtungsmatrizen zur Verfügung, die bereits umfangreiche Sprachmuster und Abhängigkeiten gelernt haben.

3.3 Beispiel: "Der Hund jagt die Katze"

1. **Eingabematrix X**: Bei 5 Tokens und einer 768-dimensionalen Einbettung ist X eine 5×768-Matrix.
   - Jede Zeile entspricht einem Token: ["Der", "Hund", "jagt", "die", "Katze"]
   - Jede Spalte enthält Werte des Token-Embeddings
2. **Berechnung der Q, K, V-Vektoren**:
   - Durch Multiplikation von X mit den Gewichtungsmatrizen W_Q, W_K, W_V
   - Wenn wir 12 Attention-Heads mit je 64 Dimensionen haben, teilen wir die 768 Dimensionen auf
3. **Multi-Head Attention**:
   - Die 768 Dimensionen werden in 12 Heads mit je 64 Dimensionen aufgeteilt
   - Reshaping der Matrizen: Von (Batch, 5, 768) zu (Batch, 12, 5, 64)
   - Für jeden Head wird separat die Attention berechnet

3.4 Attention-Berechnung (vereinfacht)

Für unser Beispiel "Der Hund jagt die Katze":

1. Das Token "jagt" (an Position 3) hat einen Q-Vektor
2. Dieser Q-Vektor wird mit den K-Vektoren aller Tokens verglichen
3. Der Vergleich mit dem K-Vektor von "Hund" ergibt einen hohen Ähnlichkeitswert
4. Das bedeutet: "jagt" sollte auf "Hund" aufmerksam werden
5. Der V-Vektor von "Hund" fließt stärker in die Ausgabe für "jagt" ein

3.5 Self-Attention und Skalierung

- Der dot-Produkt-Vergleich zwischen Q und K skaliert durch die Anzahl der Dimensionen
- Dies verhindert zu große Werte, die die Softmax-Funktion dominieren könnten
- Die finale Formel lautet:

```
Attention(Q, K, V) = Softmax(QK^T / sqrt(d_k)) V
```

wobei `d_k` die Dimension eines einzelnen Attention-Heads ist

3.6 Attention-Heads und ihre Bedeutung

- **Jeder Attention-Head lernt unterschiedliche sprachliche Konzepte**:
  - Ein Head könnte grammatikalische Beziehungen erkennen (z.B. "Hund" als Subjekt von "jagt")
  - Ein anderer Head könnte semantische Beziehungen erkennen (z.B. "Hund" und "Katze" als Tiere)
  - Weitere Heads könnten Positionen oder Redewendungen erkennen
- **Der Vorteil mehrerer Heads**:
  - Parallele Erfassung verschiedener sprachlicher Muster
  - Verbesserung der Sprachrepräsentation durch multiple Perspektiven

3.7 Mehrere Attention-Schichten

- Transformer-Modelle nutzen mehrere Schichten von Self-Attention
- Die Ausgabe einer Schicht dient als Eingabe für die nächste
- Dies ermöglicht tiefere Erfassung von Bedeutungszusammenhängen

In **BERT-Base** gibt es beispielsweise **12 Attention-Schichten**, jede mit **12 Attention-Heads**, während **GPT-3** je nach Modellgröße **96 oder mehr Schichten** mit mehreren Attention-Heads enthält.

> **Fazit**
>
> Der Attention-Mechanismus ist das Herzstück moderner Transformer-Modelle. Er ermöglicht eine effiziente Verarbeitung natürlicher Sprache, indem er flexibel zwischen Tokens die Beziehungen bestimmt. Durch die Kombination von Multi-Head Attention, Positionscodierung und mehreren Schichten können Modelle wie GPT oder BERT hochgradig kontextabhängige Repräsentationen lernen, die für zahlreiche NLP-Anwendungen genutzt werden.

4 Foundation Models

Foundation Models sind leistungsstarke KI-Modelle, die auf riesigen Datenmengen vortrainiert wurden und als Grundlage für verschiedene Anwendungen dienen. Sie nutzen Architekturen wie **Transformers** und sind in der Lage, durch **Transfer Learning** für spezifische Aufgaben feinjustiert zu werden.

Beispiele für solche Modelle sind **GPT (OpenAI), BERT (Google) oder LLaMA (Meta)**. Sie finden Anwendung in Bereichen wie **Texterstellung, Bilderkennung, Sprachverarbeitung**

und Codegenerierung.

Ein Schlüsselmerkmal ist ihre **Skalierbarkeit**, die es ermöglicht, sie für unterschiedlichste Domänen und Anwendungsfälle effizient anzupassen.

Merkmale von Foundation Models

Die Bewertung grundlegender Modelle ist entscheidend, um ihre Fähigkeiten, Grenzen und Eignung für bestimmte Aufgaben zu verstehen. Eine genaue Bewertung stellt sicher, dass die Modelle in realen Anwendungen sicher, zuverlässig und effektiv sind. Sie hilft auch bei der Identifizierung potenzieller Verzerrungen und Fehler, die ihre Leistung beeinträchtigen könnten.

Offene vs. geschlossene Modelle

Offene Modelle bedeuten, dass die trainierten Parameter eines Modells für die Öffentlichkeit zugänglich sind. Dadurch haben Wissenschaftler und Entwickler die Möglichkeit, die Mechanismen des Modells nachzuvollziehen, Forschungsarbeiten zu reproduzieren sowie Anpassungen oder Verbesserungen vorzunehmen.
Geschlossene Modelle hingegen sind Modelle, bei denen der Zugriff auf die Parameter eingeschränkt ist. Unternehmen setzen diese oft in kommerziellen Produkten oder Dienstleistungen ein, da die Veröffentlichung der Gewichte geschäftliche Interessen oder den Schutz der Nutzerdaten beeinträchtigen könnt

Parametergewichte

Die Anzahl der Parametergewichte eines Modells gibt Aufschluss über dessen Kapazität und Komplexität. Sie wird typischerweise in Millionen (M), Milliarden (B) oder Billionen (T) angegeben. Eine höhere Parameteranzahl erhöht grundsätzlich die Fähigkeit des Modells, komplexe Zusammenhänge und feine Unterschiede in den Daten zu erkennen. Allerdings ist dies kein eindeutiger Indikator für die Leistungsfähigkeit in jeder Anwendung, sondern eher ein Hinweis auf das allgemeine Potenzial des Modells.

Vorteile:

- **Größere Lernfähigkeit:** Modelle mit mehr Parametern können feinere und differenziertere Muster in Daten erfassen, was ihre Präzision und Wirksamkeit bei verschiedenen Aufgaben steigern kann.
- **Bessere Anpassungsfähigkeit:** Umfangreichere Modelle haben oft eine stärkere Fähigkeit zur Generalisierung und können unter geeigneten Trainingsbedingungen auch auf neue, unbekannte Daten besser reagieren.

Nachteile:

- **Hoher Rechenaufwand:** Eine größere Anzahl an Parametern erfordert mehr Rechenleistung für Training und Inferenz, was leistungsfähige Hardware und längere

Verarbeitungszeiten nötig macht.
- **Gefahr der Überanpassung:** Ohne angemessene Regularisierungstechniken besteht die Gefahr, dass das Modell sich zu stark an die Trainingsdaten anpasst und bei neuen, variierenden Datensätzen schlechter abschneidet.
- **Umweltbelastung:** Das Training umfangreicher Modelle verbraucht erheblich mehr Energie, was zu einem höheren CO_2-Ausstoß führt.

Kontextfenstergröße

Die Kontextfenstergröße eines Modells gibt an, wie viele Token (Wörter oder Wortbestandteile) es bei der Generierung oder Vorhersage von Text gleichzeitig verarbeiten kann. Diese Eigenschaft ist aus mehreren Gründen essenziell:

- **Erweiterter Kontext**: Ein größeres Kontextfenster erlaubt es dem Modell, mehr Informationen zu berücksichtigen, was zu kohärenteren und inhaltlich präziseren Ergebnissen führt. Dies ist besonders vorteilhaft für Aufgaben, die lange Texte oder komplexe Zusammenhänge erfordern.
- **Erfassung von Abhängigkeiten**: Ein Modell mit größerem Kontextfenster kann weiter zurückliegende Zusammenhänge im Text besser erfassen, was seine Leistungsfähigkeit bei Aufgaben wie Textzusammenfassungen, Frage-Antwort-Systemen oder interaktiven Dialogen erheblich steigert.

Ein fundiertes Verständnis dieser Faktoren hilft bei der Auswahl eines geeigneten Modells für spezifische Anwendungsfälle und ermöglicht eine gezielte Optimierung der Leistung.

Token

In Large Language Models (LLMs) wie GPT (Generative Pre-trained Transformer) stellen Token die grundlegenden Einheiten des verarbeiteten Textes dar. Ein **Token** kann ein vollständiges Wort, ein Wortbestandteil oder ein einzelnes Zeichen sein. Wie genau ein Token definiert wird, hängt vom verwendeten Tokenizer ab, der während des Modelltrainings zum Einsatz kam. So könnte das Wort *Hausboot* entweder als einzelnes Token betrachtet oder in die Bestandteile *Haus* und *boot* zerlegt werden - abhängig von der jeweiligen Tokenisierungsstrategie.

Die Nutzungskosten eines LLMs zur Textgenerierung werden in der Regel anhand der verarbeiteten Token berechnet. Dabei zählen sowohl die Token aus der Eingabe als auch diejenigen, die das Modell als Antwort generiert. Da die Verarbeitung jedes Tokens insbesondere bei sehr großen Modellen mit Milliarden von Parametern erhebliche Rechenressourcen erfordert, beeinflusst die Anzahl der Token direkt die Rechenkosten. Daher ist ein effizientes Management der Token-Nutzung essenziell, um Kosten zu optimieren.

Die **Kontextfenstergröße** eines LLM gibt an, wie viele Token aus einer Eingabe das Modell gleichzeitig berücksichtigen kann. Hat ein Modell beispielsweise ein Kontextfenster von

1.024 Token, kann es nur die letzten 1.024 Token eines Textes für die nächste Vorhersage verwenden. Ist der Eingabetext länger, kann das Modell auf frühere Abschnitte nicht mehr direkt zugreifen, was sich auf die Kohärenz und Relevanz der generierten Antworten auswirken kann.

Temperatur

Große Sprachmodelle

Name	Ersteller	Open/Closed	Input-Token	Output-Token	Anzahl Parameter
gpt-4o	OpenAI	Closed	128K	4K	200B
gpt-4o-mini	OpenAI	Closed	128K	4K	Unbekannt
Gemini 2.0 Ultra	Google	Closed	2M	4K	Unbekannt
Gemini 1.5 Pro	Google	Closed	2M	4K	Unbekannt
Gemini 1.5 Flash	Google	Closed	1M	4K	Unbekannt
Claude 3 Opus	Anthropic	Closed	200K	4K	Unbekannt
Claude 3 Sonnet	Anthropic	Closed	200K	4K	~400B
Claude 3 Haiku	Anthropic	Closed	200K	4K	Unbekannt
Llama 3.1 405B	Meta	Open	512K	4K	405B
Llama 3.1 70B	Meta	Open	256K	4K	70B
Llama 3.1 8B	Meta	Open	128K	4K	8B
Mistral 7B	Mistral.AI	Open	32K	4K	~7B

Auszug OpenAI Modelle: (Stand 02.2025)

Modell	Anwendungsbereiche
gpt-4o	Textgenerierung, Übersetzungen, Zusammenfassungen, breite Anwendungen in der natürlichen Sprachverarbeitung.
gpt-4o-audio-preview	Generierung und Verarbeitung von audio-basierten Inhalten, wie automatisierte Transkription, Spracherkennung und Musikkomposition.

Modell	Anwendungsbereiche
gpt-4o-realtime-preview	Echtzeitanwendungen wie interaktive Chatbots, Echtzeit-Übersetzungen, schnelle Antwortzeiten erfordernd.
gpt-4o-mini	Leichte Textgenerierung, geeignet für weniger rechenintensive Anwendungen, z.B. in Embedded-Systemen.
gpt-4o-mini-audio-preview	Einfache Spracherkennung, Soundeffekt-Generierung, weniger komplexe Audioverarbeitungsaufgaben.
gpt-4o-mini-realtime-preview	Echtzeitanwendungen mit schnellen Antwortzeiten, geeignet für mobile oder IoT-Geräte.
o1	Anspruchsvolle Berechnungen, komplexe Simulationen, detaillierte Datenanalysen, umfangreiche Sprachmodelle.
o3-mini	Kostengünstige NLP- und ML-Aufgaben, ideal für akademische Forschung oder Startups mit begrenztem Budget.
o1-mini	Mittelschwere Aufgaben in Machine Learning und Datenverarbeitung, weniger ressourcenintensive Alternative zu Vollversionen.