M05 - Transformer

Anwendung Generativer KI

Stand: 03.2025

1 Was ist ein Transformer?

Ein Transformer ist eine spezielle Art von neuralem Netzwerk, das seit 2017 die KI-Welt revolutioniert hat. Diese Architektur steckt hinter bekannten Textgeneratoren wie ChatGPT, Llama und Gemini. Transformer werden auch für Bilder, Audio und andere Anwendungen genutzt.

2 Textgenerierende Transformer

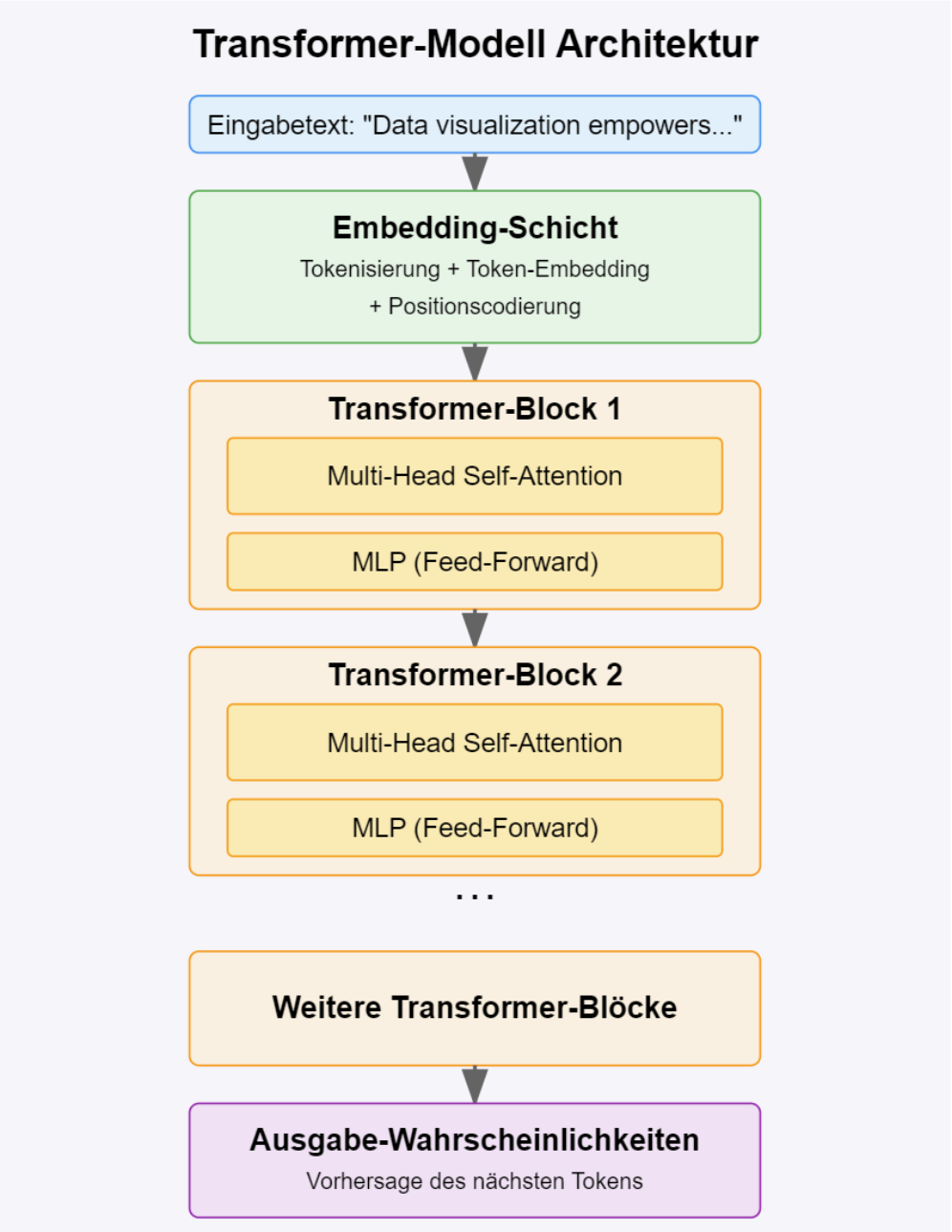
Sie arbeiten nach dem Prinzip "Was kommt als nächstes?": Wenn du einen Text eingibst, berechnet das Modell, welches Wort mit der höchsten Wahrscheinlichkeit folgen sollte. Der wichtigste Teil ist der **Self-Attention-Mechanismus**, der es dem Modell erlaubt, Beziehungen zwischen allen Wörtern in einem Text zu verstehen.

Ein bekanntes Beispiel ist GPT-2 (small) mit 124 Millionen Parametern - kein Riese nach heutigen Standards, aber ein gutes Modell zum Verständnis der Grundlagen.

3 Aufbau eines Transformer-Modells

Ein Transformer besteht aus drei Hauptteilen:

1. **Embedding-Schicht**: Wandelt Text in Zahlen um
2. **Transformer-Blöcke**: Verarbeiten diese Zahlen
3. **Ausgabe-Schicht**: Berechnet Wahrscheinlichkeiten für das nächste Wort



3.1 Embedding - Text in Zahlen

Bevor ein Transformer arbeiten kann, muss Text in Zahlen umgewandelt werden:

1. **Tokenisierung**: Der Text wird in Einzelteile (Token) zerlegt. Diese können ganze Wörter oder Wortteile sein.
2. **Token-Embedding**: Jedem Token wird ein Zahlenvektor zugewiesen (bei GPT-2 small sind das 768 Zahlen pro Token).
3. **Positionscodierung**: Dem Modell wird mitgeteilt, an welcher Position jedes Token steht.

Beispiel: "Data visualization empowers users to..." wird in Token zerlegt und jedes Token bekommt einen eigenen Zahlenvektor.

3.2 Transformer-Blöcke: Das Herzstück

Hier findet die eigentliche Verarbeitung statt. Jeder Block enthält:

- **Multi-Head Self-Attention**: Ermöglicht dem Modell, auf relevante Teile des Textes zu "achten"
- **MLP (Mehrschichtiges Perzeptron)**: Ein kleines neuronales Netzwerk zur Weiterverarbeitung

Mehrere **Transformer-Blöcke** werden hintereinandergeschaltet, damit das Modell die Informationen schrittweise immer besser verstehen kann.
Jeder Block verfeinert die Darstellung ein Stück weiter:

- Frühere Blöcke erkennen eher einfache Muster (z. B. Beziehungen zwischen einzelnen Wörtern),
- spätere Blöcke erkennen komplexere Zusammenhänge (z. B. die Bedeutung ganzer Sätze oder Abschnitte).

Durch diese gestufte Verarbeitung kann das Modell tiefere, genauere Einsichten in den Text gewinnen.

3.2.1 (Multi-Head) Self-Attention

Dies ist der Kern eines Transformers. Für jeden Token werden drei Arten von Vektoren berechnet:

- **Query (Q)**: "Was suche ich?" - ähnlich einer Suchanfrage
- **Key (K)**: "Wo könnte es sein?" - ähnlich Seitentiteln
- **Value (V)**: "Was ist der Inhalt?" - die eigentliche Information

Durch Berechnungen mit diesen Vektoren bestimmt das Modell, wie stark jedes Wort auf andere Wörter "achten" soll. GPT-Modelle verwenden dabei "maskierte Attention", was bedeutet, dass ein Wort nur auf vorangegangene Wörter achten darf, nicht auf zukünftige.

Die Attention wird auf mehrere "Köpfe" (Heads) aufgeteilt, damit das Modell unterschiedliche Beziehungen zwischen Wörtern lernen kann.

3.2.2 MLP (Mehrschichtiges Perzeptron):

Im Transformer verarbeitet das **MLP** die Informationen, die die Attention gesammelt hat, noch einmal weiter.
Es hilft dem Modell dabei, wichtige Zusammenhänge und Muster besser zu erkennen und die Darstellung der Daten zu verfeinern.
Das Ergebnis ist eine stärkere, "intelligentere" Repräsentation des Textes, bevor er an den nächsten Block weitergegeben wird.

3.3 Ausgabe-Wahrscheinlichkeiten

Am Ende berechnet das Modell Wahrscheinlichkeiten für jedes mögliche nächste Wort. Mit verschiedenen Parametern kann man die Textgenerierung steuern:

- **Temperatur**: Steuert, wie "kreativ" oder wie "sicher" die Antworten sind
  - Niedrige Temperatur: vorhersehbare, sichere Antworten
  - Hohe Temperatur: kreativere, überraschendere Antworten
- **Top-k-Sampling**: Beschränkt die Auswahl auf die k wahrscheinlichsten Wörter
- **Top-p-Sampling**: Wählt aus einer dynamischen Anzahl wahrscheinlicher Wörter aus

4 Python-Beispiel für einen einfachen Transformer

Hier ist ein einfaches Beispiel, wie man mit der Hugging Face Transformers-Bibliothek einen vortrainierten Transformer verwenden kann:

```
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer


def generate_text(prompt, max_length=20):
    """Generiert Text mit GPT-2 basierend auf einem Eingabeprompt"""

    # Modell und Tokenizer laden
    tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
    model = GPT2LMHeadModel.from_pretrained('gpt2')

    # Tokenizer konfigurieren und Text vorbereiten
    tokenizer.pad_token = tokenizer.eos_token
    inputs = tokenizer(prompt, return_tensors="pt", padding=True)

    # Text generieren
    outputs = model.generate(
        inputs["input_ids"],
        attention_mask=inputs["attention_mask"],
        max_length=max_length,
        temperature=0.7,          # Kreativitätsparameter
        top_k=10,                 # Top-k Sampling
        do_sample=True,           # Sampling aktivieren
        pad_token_id=tokenizer.eos_token_id
    )

    # Ergebnis dekodieren und zurückgeben
    return tokenizer.decode(outputs[0], skip_special_tokens=True)


# Beispiel ausführen
if __name__ == "__main__":
```

```
prompt = "Data visualization empowers users to"
print(generate_text(prompt))
```

Tip

Transformer Explainer: LLM Transformer Model Visually Explained