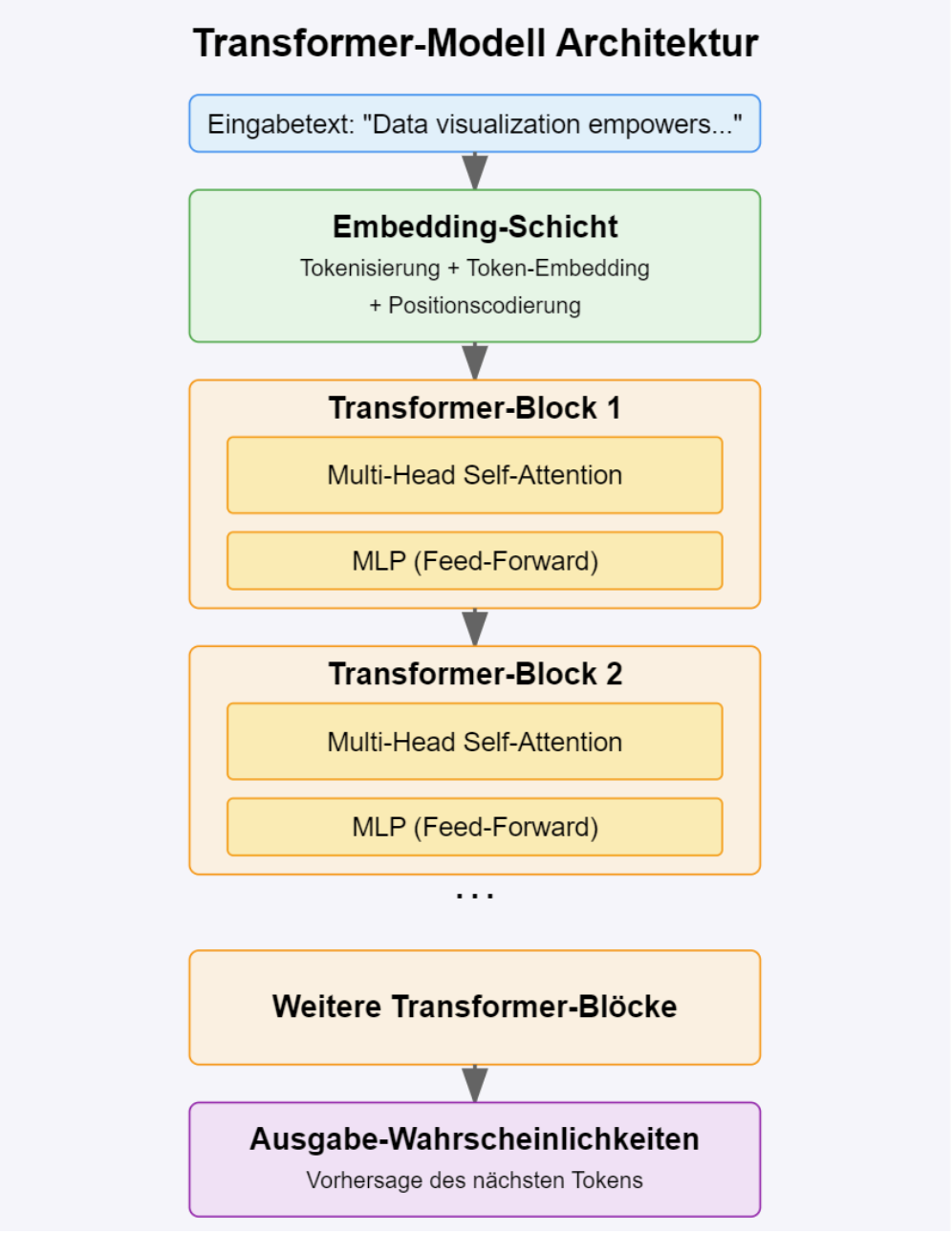
M05b - Transformer-Explainer

Stand: 03.2025

[Transformer Explainer: LLM Transformer Model Visually Explained]



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 umwandeln

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

3.3 (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.

3.4 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 Zusätzliche wichtige Elemente

Für ein stabiles Training braucht ein Transformer noch:

- **Layer-Normalisierung**: Hält die Zahlen im Modell in einem sinnvollen Bereich
- **Dropout**: Schaltet zufällig Teile des Netzwerks ab, damit es nicht auswendig lernt
- **Residual-Verbindungen**: Ermöglichen ein besseres Training tiefer Netzwerke

Hier siehst du eine vereinfachte Darstellung der Transformer-Architektur. Diese Visualisierung zeigt den Fluss eines Textes durch das Modell - von der Eingabe über die Embedding-Schicht durch mehrere Transformer-Blöcke bis zur endgültigen Vorhersage des nächsten Wortes.

5 Python-Beispiel für einen einfachen Transformer

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

```
# Einfaches Beispiel für die Verwendung eines Transformer-Modells
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer

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

# Eingabetext
eingabetext = "Data visualization empowers users to"

# Text in Token umwandeln
input_ids = tokenizer.encode(eingabetext, return_tensors='pt')

# Parameter für die Textgenerierung
temperatur = 0.7  # Kreativität (höher = kreativer)
top_k = 50        # Nur die 50 wahrscheinlichsten Wörter betrachten
max_laenge = 30  # Maximale Textlänge

# Text generieren
ausgabe = model.generate(
    input_ids,
    max_length=max_laenge,
    temperature=temperatur,
    top_k=top_k,
    do_sample=True
)

# Generiertes Ergebnis dekodieren und anzeigen
generierter_text = tokenizer.decode(ausgabe[0], skip_special_tokens=True)
print(generierter_text)
```