Transformer-Architektur am Beispiel *GPT-3*

Vokabular: 50.257 Token | Modelldimension (d_M): 12.288 | Kontextfenster: 2.048 Token | Parameter: ~175 Milliarden
Trainingsset: ~300 Milliarden Token | Vereinfachter Überblick, einige Komponenten weggelassen (Normalisierung, Softmax, etc.)

① Der **Tokenizer** zerlegt ein Input in Token (siehe Beispiel unten).

② Die **Embedding-Matrix** enthält für jedes Token im Vokabular einen *Embedding Vektor* der Modelldimension d_M.

Die **Positional-Matrix** enthält für jede Position des Kontextfensters einen *Positional Vektor* der Modelldimension d_M.

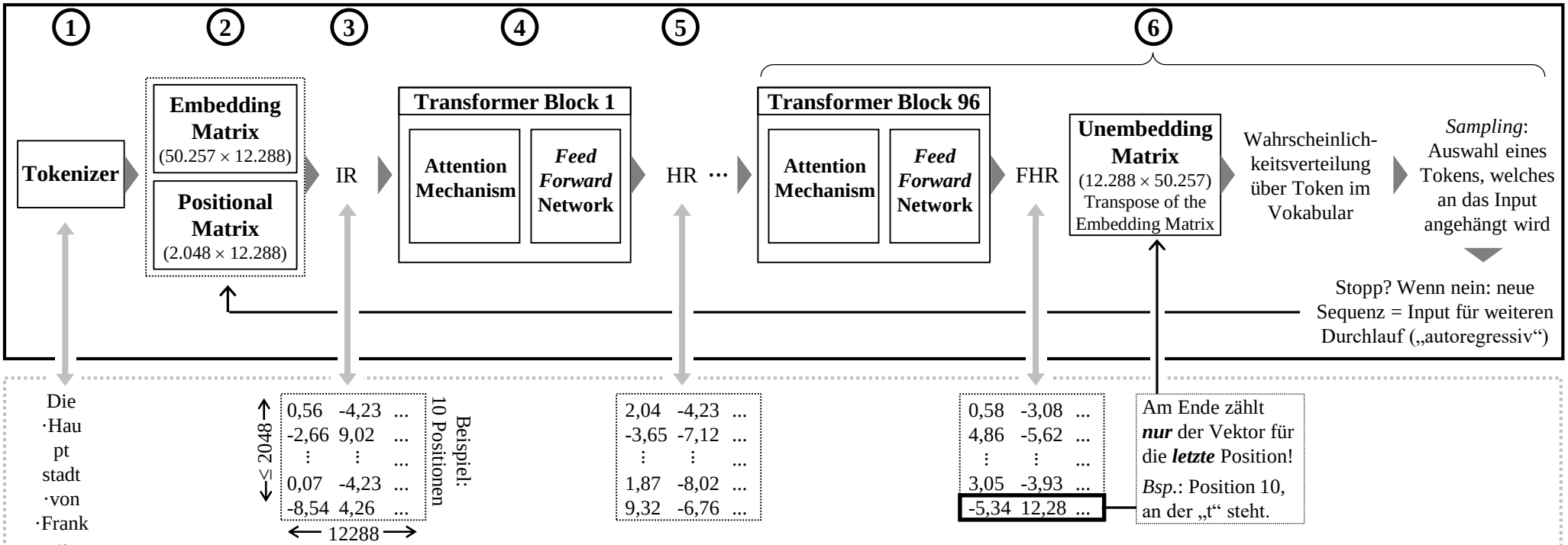
Alle Einträge in Embedding- und Positional-Matrix sind (bei GPT-3) ***gelernte Parameter***.

③ Nach *Tokenization* wird für jedes Input-Token der entsprechende Vektor aus der Embedding Matrix mit dem entsprechenden Vektor der Positional Matrix addiert. Wir erhalten so die **Input-Representation (IR)**.

④ Die IR wird im ersten **Transformer-Block** weiterverarbeitet und kontextsensitiv angereichert. Zuerst prüft der *Attention-Mechanismus*, welche früheren Token für ein gegebenes Token relevant sind, und mischt entsprechende Informationen in dessen Vektor ein. Danach verarbeitet ein *Feed Forward* Netz jeden Vektor einzeln weiter.

⑤ Die Ausgabe des ersten Transformer-Blocks ist die erste **Hidden-Representation (HR)**. Die HR durchläuft dann Transformer-Block 2 (hier grafisch ausgelassen), wodurch eine aktualisierte HR entsteht. GPT-3 arbeitet mit 96 Transformer-Blöcken (unterschiedlich je nach Modell).

⑥ Die Ausgabe des letzten Transformer-Blocks nennen wir die **Final Hidden Representation (FHR)**. Für die Auswahl des nächsten Tokens wird nur der Vektor für die letzte Token-Position verwendet. Dieser Vektor wird mit der **Unembedding-Matrix** multipliziert. Das liefert die Grundlage für eine Wahrscheinlichkeitsverteilung über das Vokabular. Auf dieser Basis wird dann ein Token ausgewählt und an die bisherige Eingabe angehängt. (Im Beispiel: Abhängig von den Trainingsdaten wird „Die Hauptstadt von Frankreich ist“ in einer Wahrscheinlichkeitsverteilung münden, in der „·Paris“ einen recht hohen Wert bekommt und abhängig von der **Sampling**-Methode ggf. ausgewählt wird.)



(Die Zahlen in IR/HR/FHR dienen nur der Illustration. Jede Zeile umfasst 12.288 reelle Zahlen (entsprechend der Modelldimension von GPT-3) und wird auch ***Token-Representation*** genannt. Bei vielen neueren LLMs ist die genaue Architektur nicht öffentlich bekannt; Schätzungen gehen oft von vergleichbaren oder etwas höheren Modelldimensionen aus, aber teils von deutlich mehr Parametern.)

GPT-3: Transformer-Block = Attention Layer + Feed Forward. GPT-3 umfasst 96 Transformer Blöcke. (Vereinfachte Darstellung.)

IR/HR
(*n* x 12288)

IR/HR umfasst für jedes Token der Input-Sequenz eine *Token-Representation* (einen Vektor der Länge 12288).

Die Aufgabe der Attention-Layer ist es, die einzelnen *Token-Representations* kontextuell anzureichern.

Hierzu ermöglicht die Attention-Layer durch den *Query-Key-Value*-Mechanismus, dass eine Token-Representation *I* verschieden aktualisiert wird, je nach dem, welche Token vor *I* kommen.

Grobes Beispiel: Eine *Token-Representation* für „book" kann verschieden aktualisiert werden, abhängig davon, welche Token vorher kommen:

My red book …

I will book ...

Attention Layer

GPT-3: 96 Attention-Heads, die parallel arbeiten. Wir ignorieren zur Vereinfachung die Aufsplittung in einzelne Heads.

W_Q Query Matrix

W_K Key Matrix

W_V Value Matrix

Jede dieser Matrizen hat die Größe 12288 x 12288, enthält also rund 151 Millionen gelernte Parameter.

Durch Matrix-Multiplikation mit IR/HR erhalten wir für jede *Token-Representation* I:

- Query-Vektor (wonach sucht I?)
- Key-Vektor (worauf antwortet I?)
- Value-Vektor (was steuert I bei?)

Für jede Token-Representation I:

- Vergleiche den *Query*-Vektor mit den *Key*-Vektoren der *Token-Representations*, die vor *I* (bis inklusive *I*) kommen.
- Dies bildet die Grundlage für eine Gewichtung, wie relevant frühere *Token-Representations* für *I* sind.
- Dies nutzen wir, um eine gewichtete Summe der *Value*-Vektoren der anderen *Token-Representations* zu bilden.
- Diese Summe leistet dann einen Beitrag zum Update der *Token-Representation I*.

Aktualisierte HR (+ Residual Stream)

Feed Forward Network

(Häufig auch *Multi Layer Perceptron* (MLP) genannt)

Ein einfaches neuronales Netz (strukturell analog zu dem Beispiel, welches wir uns in der letzten Sitzung für die Ziffern-Erkennung angeschaut hatten).

Alle *Token-Representations* durchlaufen das MLP einzeln, es gibt keine „Interaktion" zwischen den *Representations* für einzelne Token (anders als in der Attention Layer, wo *Token-Representations* über den Query/Key/Value-Mechanismus andere *Token-Representations* beeinflussen können).

Die typische Rolle des MLP ist: die gerade durch Attention kontextualisierte *Token-Representations* nichtlinear „umzuformen"—Features verstärken/unterdrücken, neue Merkmale kombinieren, Bedeutungsaspekte umkodieren. Es ist der Teil, der pro Token „Rechenarbeit" macht, während Attention die Informationen „zusammenträgt".

Jedes der MLP bei GPT-3 umfasst allein 1,21 Milliarden Parameter.

Aktualisierte HR (+ Residual Stream)