Worauf schaut die KI gerade? (Attention)

Kapitel online

Jedes Wort fragt alle anderen: „Wie wichtig bist du für mich?“ — Erlebe an der doppeldeutigen „Bank“, wie Attention Bedeutung aus dem Kontext baut. Der Kernmechanismus aller modernen KI-Modelle.

Aktuell · Kapitel 30 von 50 · ca. 15 Min. · 250 XP

Einfach gesagt: Tippst du im Satz auf „Bank“, siehst du, wie jedes andere Wort mitentscheidet, ob sie Ufer oder Geldinstitut meint.

Ein Wort wie **„Bank“** hat allein keine feste Bedeutung: Geldinstitut? Sitzgelegenheit? In „Latent Space“ war jedes Wort EIN Punkt — aber welcher Punkt ist richtig, hängt vom **Satz drumherum** ab. Der Mechanismus, der das löst, heißt **Attention** — und er ist der Grund, warum moderne Sprachmodelle überhaupt funktionieren.

Was du hier lernst

- Warum **„Bank“** erst durch den Satz drumherum eine Bedeutung bekommt
- Was **Query, Key und Value** sind — der Karteikasten hinter der Formel
- Wie aus den **Attention-Gewichten** eine ganze Tabelle wird: alle Wörter x alle Wörter
- Warum ein Transformer **viele Köpfe** hat: Bedeutung, Grammatik, Nachbarn — parallel
- Dass der ganze Mechanismus aus **drei Zeilen** besteht: matMul, Softmax, matMul

Für den Unterricht

Einstieg (3 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (9 Min): die interaktiven Module auf herrki.de/320_attention ausprobieren.

Sichern mit Quiz (3 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Die Idee in einem Satz

Jedes Wort fragt alle anderen Wörter: „Wie wichtig bist du für MICH?“ — und mischt sich aus den Antworten eine neue, vom Kontext abhängige Bedeutung zusammen.

Drei Zutaten pro Wort, ganz ohne Formelgewitter — stell dir einen Karteikasten vor:

Query (Suchanfrage): „Wonach suche ich gerade?“ — *Bank* fragt z. B.: „Gibt es hier Hinweise, welche Bank ich bin?“

Key (Karteireiter): „Was biete ich an?“ — *Fluss* meldet: „Ich bin Natur/Wasser!“

Value (Karteninhalt): die eigentliche Information, die weitergegeben wird, wenn Suchanfrage und Reiter zusammenpassen.

Passt eine Query gut zu einem Key, gibt es ein **hohes Attention-Gewicht** — und der zugehörige Value fließt stark in die neue Bedeutung ein.

Am Gerät ausprobieren: herrki.de/320_attention#abschnitt-2

Direkt zum Modul

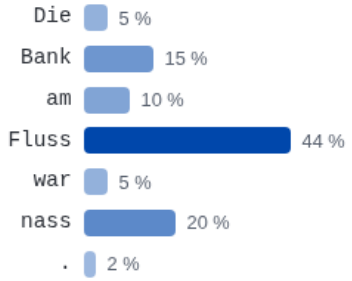
🖌️ Tippe ein Wort an — und sieh, worauf es „schaut“

Satz: „Die Bank am Fluss war nass.“ | „Die Bank hat mein Konto gesperrt.“

Kopf: Kopf 1: Bedeutung | Kopf 2: Grammatik | Kopf 3: Nachbarn

Die | Bank | am | Fluss | war | nass | .

„Bank“ fragt: Wie wichtig seid ihr für mich?




```
Kopf „Bedeutung“: Die „Bank“ schaut vor allem auf das Wort, das klärt, WELCHE Bank sie ist.
Stärkster Blick von „Bank“: → „Fluss“ (44 %).
```

Die ganze Matrix auf einen Blick

Attention berechnet diese Gewichte für **alle Wörter gleichzeitig** — eine ganze Tabelle (Zeile = wer schaut, Spalte = worauf; jede Zeile summiert sich zu 100 %):

schaut ↓ auf →	Die	Bank	am	Fluss	war	nass	.
Die	15	37	7	22	7	7	4
Bank	5	15	10	44	5	20	2
am	7	21	14	41	7	7	3
Fluss	6	36	12	18	6	18	3
war	7	15	7	15	15	37	4
nass	5	26	5	31	15	15	3
.	8	15	8	15	15	15	23

Interaktives Modul – am Gerät ausprobieren

Die ganze Matrix auf einen Blick

Attention berechnet diese Gewichte für **alle Wörter gleichzeitig** — eine ganze Tabelle (Zeile = wer schaut, Spalte = worauf; jede Zeile summiert sich zu 100 %):

Warum MEHRERE Köpfe?

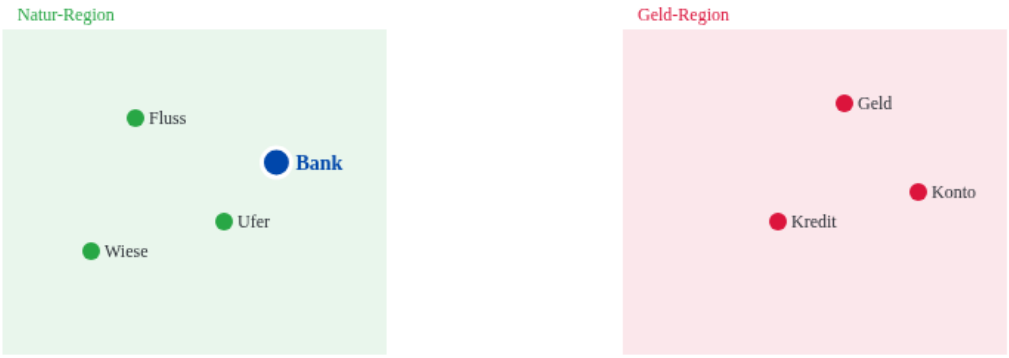
Du hast es oben schon umgeschaltet: Ein Kopf achtet auf **Bedeutung** (Bank↔Fluss bzw. Bank↔Konto), einer auf **Grammatik** (Artikel↔Substantiv, Hilfsverb↔Partizip), einer schlicht auf **Nachbarwörter**. Echte Modelle haben Dutzende solcher Köpfe pro Schicht — jeder lernt im Training selbst, worauf er achtet. „Der Transformer schaut auf alles gleichzeitig“ heißt genau das: **alle Wörter × alle Wörter × viele Blickwinkel, parallel.**

Am Gerät ausprobieren: herrki.de/320_attention#abschnitt-3

Direkt zum Modul

🗺️ Und so wandert die Bedeutung im Latent Space

Derselbe Wortpunkt „Bank“, zwei Sätze: Schalte oben den Satz um und sieh zu, wie der Kontext den Punkt zwischen der Geld-Region und der Natur-Region verschiebt (Karte wie in Kapitel 310):



Interaktives Modul – am Gerät ausprobieren

Runter zum Tensor

Alles, was du eben angetippt hast, ist Tensor-Rechnung (Kapitel 060). Rechnen wir eine Zeile der Heatmap von Hand nach: Was fragt die „Bank“ im Satz „Die Bank am Fluss war nass.“ — und wer antwortet am lautesten?

```
// Zum Mitrechnen auf 3 Zahlen verkleinert – echt sind es 64 bis 128 pro Kopf.
// V (der weitergegebene Inhalt) sieht hier der Einfachheit halber aus wie K.
                     Natur/Wasser Geld Satzbau
Q „Bank“ (sucht) 1.5 1.2 0.4
K „Fluss“ (bietet) 1.8 0.0 0.3
K „nass“ (bietet) 0.9 0.0 0.4
K „Die“ (bietet) 0.0 0.0 0.6
```

```
Punkte = Q · K, Achse für Achse:
  Bank·Fluss = 1.5×1.8 + 1.2×0.0 + 0.4×0.3 = 2.82
  Bank·nass = 1.5×0.9 + 1.2×0.0 + 0.4×0.4 = 1.51
  Bank·Die = 1.5×0.0 + 1.2×0.0 + 0.4×0.6 = 0.24

Softmax macht daraus Prozente (Summe 100 %):
  e^2.82 = 16.8 e^1.51 = 4.5 e^0.24 = 1.3 Summe 22.6
  Fluss 16.8/22.6 = 74 % nass 20 % Die 6 %

Neue Bedeutung = gewichteter Mix der Values:
  Natur-Achse: 0.74×1.8 + 0.20×0.9 + 0.06×0.0 = 1.51
  → „Bank“ wandert Richtung Natur – genau die Karte oben.
```

Und jetzt der zweite Satz: Statt „Fluss“ steht dort „Konto“ mit `K = [0.0, 1.9, 0.2]`. Dieselbe Frage, dieselben Gewichte — nur die Nachbarn sind andere: `1.5×0.0 + 1.2×1.9 + 0.4×0.2 = 2.36`. Die Geld-Achse gewinnt, und dieselbe „Bank“ bekommt eine andere Bedeutung. So heißt das im Code, für alle sieben Wörter gleichzeitig:

```
// Q, K, V entstehen aus den Wort-Embeddings (Kapitel 040/310) per Matrix-Multiplikation.
punkte = Q.matMul( K.transpose() ) // [7,64]x[64,7] -> [7,7]: unsere 2.82, 1.51, 0.24 ...
gewichte = softmax( punkte ) // jede Zeile -> Prozente, Summe 100 % (deine Heatmap!)
neu = gewichte.matMul( V ) // [7,7]x[7,64] -> [7,64]: der Mix von oben
```

Drei Zeilen. Millionenfach parallel. Das ist der komplette Trick.

Zurück zur Anwendung

Wenn ChatGPT in einem langen Gespräch weiß, dass sich „sie“ auf deine Katze aus Absatz 1 bezieht — das ist Attention. Wenn ein Übersetzer „bank“ mal mit Ufer, mal mit Geldhaus übersetzt — Attention. Und in „Erst denken, dann reden“ siehst du als Nächstes, was passiert, wenn ein Modell mit diesem Mechanismus Schritt für Schritt *nachdenkt*.

Zum Weiterlesen

- **Das Paper, das alles veränderte:** Ashish Vaswani u. a., *„Attention Is All You Need“*, 2017 — arxiv.org/abs/1706.03762 — die drei Zeilen von oben im Original, samt Multi-Head und Positions-Kodierung
- **Die Vorgeschichte:** Dzmitry Bahdanau u. a., *„Neural Machine Translation by Jointly Learning to Align and Translate“*, 2014 — arxiv.org/abs/1409.0473 — hier taucht Attention zum ersten Mal auf, noch als Zusatz zu einem älteren Netz
- **Bild für Bild erklärt:** Jay Alammar, *„The Illustrated Transformer“* — jalammar.github.io (jalammar.github.io/illustrated-transformer) — die freundlichste Bebilderung von Query, Key und Value im Netz
- **Zum Mitprogrammieren:** Harvard NLP, *„The Annotated Transformer“* — nlp.seas.harvard.edu (nlp.seas.harvard.edu/annotated-transformer) — das komplette Paper als lauffähiger Code, Absatz neben Zeile

Kurzer Check – mit Lösung

Lösungen für Lehrkräfte – für Lernende die Online-Version nutzen.

Frage 1: Was berechnet Attention für ein Wort?

- Seine Silbenzahl im Vergleich zu allen anderen Wörtern.
- **✓ Gewichte, wie wichtig jedes andere Wort für seine Bedeutung ist — und mischt daraus eine neue, kontextabhängige Bedeutung.**
- Die alphabetische Reihenfolge des Satzes.

„Wie wichtig bist du für mich?“ an alle — dann gewichteter Mix. Deshalb bekommt „Bank“ neben „Fluss“ eine andere Bedeutung als neben „Konto“.

Frage 2: Wofür stehen Query, Key und Value?

- **✓ Suchanfrage („wonach suche ich?“), Karteireiter („was biete ich an?“) und Inhalt (was weitergegeben wird).**
- Drei verschiedene Sprachmodelle, die abstimmen.
- Frage, Antwort und Bewertung des Nutzers.

Passt Query zu Key → hohes Gewicht → der Value fließt stark ein. Alle drei entstehen per Matrix-Multiplikation aus den Embeddings.

Frage 3: Warum hat ein Transformer VIELE Attention-Köpfe?

- Als Reserve, falls ein Kopf ausfällt.
- Damit er mehrere Nutzer gleichzeitig bedienen kann.
- **✓ Jeder Kopf lernt einen anderen Blickwinkel — Bedeutung, Grammatik, Position … — und alle arbeiten parallel.**

Genau das hast du umgeschaltet: Ein Kopf allein wäre ein einziger Blickwinkel. Viele Köpfe = viele gleichzeitige Beziehungs-Analysen desselben Satzes.

Das nimmst du mit

- **Bedeutung entsteht im Satz.** „Bank“ ist allein mehrdeutig — erst die Nachbarwörter entscheiden, welcher Punkt im Latent Space gemeint ist.
- **Query, Key, Value sind ein Karteikasten.** Wonach suche ich, was biete ich an, was gebe ich weiter — alle drei entstehen per Matrix-Multiplikation aus den Embeddings.
- **Aus den Gewichten wird eine ganze Tabelle.** Jedes Wort bewertet jedes andere: alle mal alle, in einem Rutsch.
- **Viele Köpfe, viele Blickwinkel.** Bedeutung, Grammatik, Nachbarschaft — jeder Kopf schaut anders auf denselben Satz, alle gleichzeitig.
- **Drei Zeilen sind der ganze Trick.** matMul, Softmax, matMul — millionenfach parallel, und genau das macht moderne Sprachmodelle möglich.

Weiter geht’s: Attention misst, welche Wörter zueinander passen — dieselbe Ähnlichkeitsrechnung macht aus einem Haufen Firmendokumente eine Suchmaschine, die Bedeutung statt Buchstaben findet: „Semantische Suche & RAG“.

So zitierst du dieses Kapitel

HerrKI: Worauf schaut die KI gerade? (Attention). herrki.de/320_attention, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.