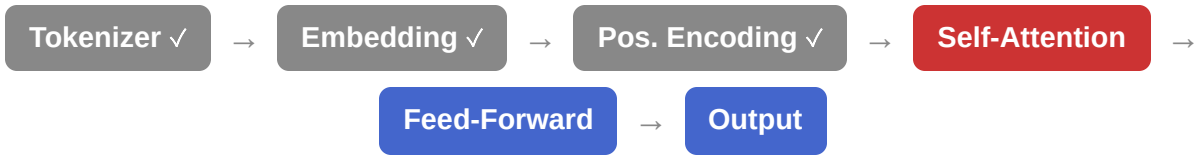
Schritt 4: Self-Attention — Das Herzstück



Erlebnispark: *Du bist das Token* ***„Katze"*** *(Position 1). Du stehst in einem Raum mit allen anderen Tokens. Du bekommst eine Brille aufgesetzt. Durch die Brille siehst du die anderen — manche leuchten hell, manche kaum. Du gehst zu denen die leuchten und nimmst etwas von ihnen mit.*

Wenn du den Raum verlässt, bist du nicht mehr nur „Katze". Du bist „Katze, die weiß, dass sie auf der Matte sitzt."

Was passiert in Self-Attention?

Jedes Wort stellt 3 Fragen — gleichzeitig, an alle anderen:

Vektor	Frage	Metapher	Formel
Query	„Was suche ich?"	Deine Brille — bestimmt wonach du schaust	$Q = Input \times W_Q$
Key	„Was biete ich an?"	Dein Namensschild — was andere an dir sehen	$K = Input \times W_K$
Value	„Was gebe ich weiter?"	Dein Paket — die Information die du teilst	$V = Input \times W_V$

W_Q, W_K, W_V sind Matrizen (Tabellen mit Zahlen). Sie werden beim Training gelernt. Für unser Papier-Modell legen wir sie fest.

Wichtig: Alle Tokens im Satz benutzen die **gleichen** W-Matrizen! Aber weil jedes Token einen anderen Input-Vektor hat, kommen verschiedene Q, K, V heraus.

Das ist die Brille aus dem Erlebnispark: **Gleiche Brille, verschiedene Augen.**

Unser Input (vom Positional Encoding)

Pos	Wort	d_1	d_2	d_3	d_4
0	**Die**	0.9	1.1	0.0	1.1
1	**Katze**	0.8	1.4	0.1	1.2
2	**sitzt**	0.9	-0.3	0.9	1.0
3	**auf**	0.6	-1.0	0.3	1.4
4	**der**	0.1	-0.6	0.0	1.1
5	**Matte**	-1.0	0.3	0.1	1.9

Werte gerundet auf 1 Dezimalstelle für leichteres Rechnen.

Die W-Matrizen: Drei verschiedene Brillen

Unsere W-Matrizen projizieren von **4D auf 2D**. Das heißt: Aus deinem 4-Zahlen-Vektor werden 2 Zahlen. Jede Matrix wählt andere Aspekte deines Vektors aus.

Die Multiplikation ist einfach: **Jede Ergebnis-Zahl = eine Zeile der W-Matrix × dein Vektor.**

W_Q (4×2) — „Was suchst du?"

1	0
0	1
0	0
0	0

Nimmt d_1 und d_2

W_K (4×2) — „Was bietest du?"

0	0
0	0
1	0
0	1

Nimmt d_3 und d_4

W_V (4×2) — „Was gibst du weiter?"

0	0
1	0
0	0
0	1

Nimmt d_2 und d_4

Warum so einfach? Unsere W-Matrizen bestehen nur aus Einsen und Nullen — sie wählen einfach bestimmte Dimensionen aus. Das macht die Multiplikation trivial: Du musst nur die richtigen Zahlen aus deinem Input-Vektor abschreiben.

In Wirklichkeit stehen dort gelernte Dezimalzahlen (z.B. 0.37, -0.82, ...) und die Multiplikation erzeugt ganz neue Kombinationen. Aber der Ablauf ist identisch.

Q, K, V für alle Tokens berechnen

Beispiel: Katze [0.8, 1.4, 0.1, 1.2]

$$Q_{Katze} = [0.8, 1.4, 0.1, 1.2] \times W_Q = [d_1, d_2] = [0.8, 1.4]$$

$$K_{Katze} = [0.8, 1.4, 0.1, 1.2] \times W_K = [d_3, d_4] = [0.1, 1.2]$$

$$V_{Katze} = [0.8, 1.4, 0.1, 1.2] \times W_V = [d_2, d_4] = [1.4, 1.2]$$

Alle Tokens:

Wort	Input [d_1, d_2, d_3, d_4]	Q [d_1, d_2]	K [d_3, d_4]	V [d_2, d_4]
Die	[0.9, 1.1, 0.0, 1.1]	[0.9, 1.1]	[0.0, 1.1]	[1.1, 1.1]
Katze	[0.8, 1.4, 0.1, 1.2]	[0.8, 1.4]	[0.1, 1.2]	[1.4, 1.2]
sitzt	[0.9, -0.3, 0.9, 1.0]	[0.9, -0.3]	[0.9, 1.0]	[-0.3, 1.0]
auf	[0.6, -1.0, 0.3, 1.4]	[0.6, -1.0]	[0.3, 1.4]	[-1.0, 1.4]
der	[0.1, -0.6, 0.0, 1.1]	[0.1, -0.6]	[0.0, 1.1]	[-0.6, 1.1]
Matte	[-1.0, 0.3, 0.1, 1.9]	[-1.0, 0.3]	[0.1, 1.9]	[0.3, 1.9]

Erlebnispark: *Jetzt hat jeder im Raum drei Dinge:*
— Eine **Brille** *(Query) — bestimmt wonach du suchst*
— Ein **Namensschild** *(Key) — was andere an dir sehen*
— Ein **Paket** *(Value) — was du weitergibst*

Beachte: Deine Brille (Q) zeigt andere Zahlen als dein Namensschild (K)! Du suchst etwas anderes als das, was du selbst anbietest.

Du bist „Katze" — Wen siehst du durch die Brille?

Erlebnispark: *Du setzt die Brille auf. Du schaust dich um. Deine Brille (Query) trifft auf die Namensschilder (Keys) der anderen. Je besser die zusammenpassen, desto heller leuchtet das Gegenüber.*

Score = $Q_{Katze} \cdot K_{anderer}$ (Skalarprodukt)
Das kennst du schon von der Embedding-Übung! Multipliziere die Zahlen an gleicher Position und addiere.

Q_{Katze} = **[0.8, 1.4]**

Score: Katze → Die

$$Q_{Katze} \cdot K_{Die} = [0.8, 1.4] \cdot [0.0, 1.1]$$
$$= (0.8 \times 0.0) + (1.4 \times 1.1)$$
$$= 0.00 + 1.54$$
$$= \mathbf{1.54}$$

Score: Katze → Katze (sich selbst!)

$$Q_{Katze} \cdot K_{Katze} = [0.8, 1.4] \cdot [0.1, 1.2]$$
$$= (0.8 \times 0.1) + (1.4 \times 1.2)$$
$$= 0.08 + 1.68$$
$$= \mathbf{1.76}$$

Score: Katze → sitzt

$$Q_{Katze} \cdot K_{sitzt} = [0.8, 1.4] \cdot [0.9, 1.0]$$
$$= (0.8 \times 0.9) + (1.4 \times 1.0)$$
$$= 0.72 + 1.40$$
$$= \mathbf{2.12}$$

Score: Katze → auf

$$Q_{Katze} \cdot K_{auf} = [0.8, 1.4] \cdot [0.3, 1.4]$$
$$= (0.8 \times 0.3) + (1.4 \times 1.4)$$
$$= 0.24 + 1.96$$
$$= \mathbf{2.20}$$

Score: Katze → der

$$Q_{Katze} \cdot K_{der} = [0.8, 1.4] \cdot [0.0, 1.1]$$
$$= (0.8 \times 0.0) + (1.4 \times 1.1)$$
$$= 0.00 + 1.54$$
$$= \mathbf{1.54}$$

Score: Katze → Matte

$$Q_{Katze} \cdot K_{Matte} = [0.8, 1.4] \cdot [0.1, 1.9]$$
$$= (0.8 \times 0.1) + (1.4 \times 1.9)$$
$$= 0.08 + 2.66$$
$$= \mathbf{2.74}$$

Alle Scores auf einen Blick:

Von Katze → zu	Die	Katze	sitzt	auf	der	Matte
Raw Score	1.54	1.76	2.12	2.20	1.54	**2.74**

Matte leuchtet am hellsten! (Score: 2.74)
Das macht Sinn: „Katze" hat am meisten mit „Matte" zu tun — die Katze sitzt auf der Matte. „Die" und „der" leuchten am schwächsten — Artikel sind für die Katze weniger relevant.

Skalierung: Warum durch √d teilen?

Bevor wir Softmax anwenden, teilen wir alle Scores durch $\sqrt{d_k}$, wobei d_k = die Dimension der Key-Vektoren (bei uns: 2).

√2 ≈ 1.41

Warum? Bei hohen Dimensionen werden Skalarprodukte sehr groß. Große Zahlen machen Softmax „extrem" — ein Token bekommt fast 100%, alle anderen fast 0%. Das Teilen durch √d hält die Werte in einem guten Bereich.

Bei unseren 2 Dimensionen ist der Effekt klein. Bei 64 Dimensionen (GPT-2) teilt man durch 8!

Skalierte Scores:

```
Die: 1.54 / 1.41 = 1.09
Katze: 1.76 / 1.41 = 1.25
sitzt: 2.12 / 1.41 = 1.50
auf: 2.20 / 1.41 = 1.56
der: 1.54 / 1.41 = 1.09
Matte: 2.74 / 1.41 = 1.94
```

Softmax — Das Leuchten auf 100% normieren

Erlebnispark: *Alle Tokens leuchten — aber unterschiedlich hell. Die Summe aller Auras muss* **100% ergeben**. *Softmax macht genau das: Es verwandelt die Scores in Prozente.*

Der Trick: Softmax benutzt $\mathbf{e^x}$ *(Euler-Zahl ≈ 2.718 hoch x). Dadurch werden große Scores* **überproportional größer** *— der Hellste leuchtet noch heller.*

Schritt 1: e^{score} für jeden berechnen

Taschenrechner: Taste e^x oder berechne 2.718^x

$e^{1.09}$ = **2.97**

$e^{1.25}$ = **3.49**

$e^{1.50}$ = **4.48**

$e^{1.56}$ = **4.76**

$e^{1.09}$ = **2.97**

$e^{1.94}$ = **6.96**

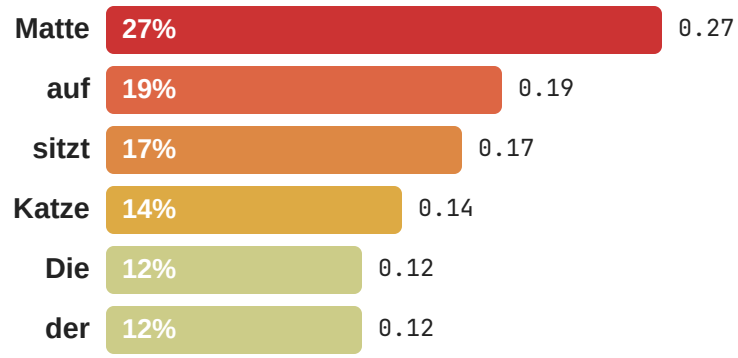
Schritt 2: Summe bilden

Summe = 2.97 + 3.49 + 4.48 + 4.76 + 2.97 + 6.96 = **25.63**

Schritt 3: Jedes durch die Summe teilen = Gewicht (Attention Weight)

Die: 2.97 / 25.63 = **0.12** (12%)
Katze: 3.49 / 25.63 = **0.14** (14%)
sitzt: 4.48 / 25.63 = **0.17** (17%)
auf: 4.76 / 25.63 = **0.19** (19%)
der: 2.97 / 25.63 = **0.12** (12%)
Matte: 6.96 / 25.63 = **0.27** (27%)

Das Leuchten — visualisiert:



Summe: 0.12 + 0.14 + 0.17 + 0.19 + 0.12 + 0.27 = **1.01** (≈ 100%, Rundungsfehler)

> **„Katze" achtet am meisten auf „Matte" (27%).**
> Danach auf „auf" (19%) und „sitzt" (17%). Zusammen nehmen „auf der Matte sitzt" über 60% der Aufmerksamkeit ein.
> Das macht Sinn: Diese Wörter beschreiben was die Katze tut und wo sie ist!

Value — „Hast du was für mich?"

Erlebnispark: *Du gehst zu jedem Token. Jeder gibt dir sein* **Paket** *(Value-Vektor). Aber du nimmst nicht von allen gleich viel — du nimmst* **proportional zum Leuchten**.

Von Matte (27%) nimmst du am meisten. Von „Die" und „der" (je 12%) am wenigsten.

Formel: Output = Σ (Gewicht$_i$ × V$_i$)

Das heißt: Multipliziere jedes Value-Paket mit seinem Gewicht und addiere alles zusammen.

Die Value-Vektoren (zur Erinnerung):

Token	V [v_1, v_2]	Gewicht	Gewicht × V
Die	[1.1, 1.1]	0.12	[0.12 × 1.1, 0.12 × 1.1] = [0.132, 0.132]
Katze	[1.4, 1.2]	0.14	[0.14 × 1.4, 0.14 × 1.2] = [0.196, 0.168]
sitzt	[-0.3, 1.0]	0.17	[0.17 × -0.3, 0.17 × 1.0] = [-0.051, 0.170]
auf	[-1.0, 1.4]	0.19	[0.19 × -1.0, 0.19 × 1.4] = [-0.190, 0.266]
der	[-0.6, 1.1]	0.12	[0.12 × -0.6, 0.12 × 1.1] = [-0.072, 0.132]
Matte	[0.3, 1.9]	**0.27**	**[0.27 × 0.3, 0.27 × 1.9] = [0.081, 0.513]**

Alles zusammenaddieren:

```
v₁:   0.132 + 0.196 + (-0.051) + (-0.190) + (-0.072) + 0.081 = 0.096
v₂:   0.132 + 0.168 + 0.170 + 0.266 + 0.132 + 0.513 = 1.381
```

$\text{Output}_{\text{Katze}}$ = [0.10, 1.38]

Was ist passiert?

Vorher (Input): Katze wusste nur etwas über sich selbst.
Nachher (Output): Katze hat Information von allen aufgesammelt — besonders von Matte.

Der hohe Wert in v_2 (1.38) kommt hauptsächlich von **Matte** (die 0.513 beigesteuert hat = 37% des Gesamtwerts). Die Katze hat das „Objekt-Wissen" der Matte absorbiert.

Erlebnispark: *Du verlässt den Raum. Du bist nicht mehr nur „Katze". Du bist jetzt* **[0.10, 1.38]** *— ein Vektor der weiß: „Ich bin eine Katze, und ich habe etwas mit einer Matte zu tun, und da ist eine Aktion (sitzen) im Spiel."*

Und das gleiche passiert **gleichzeitig für jedes Token***. Jeder geht durch den Raum, schaut durch seine Brille, sammelt Pakete ein. Am Ende hat jeder Token einen neuen Vektor, der den ganzen Satz-Kontext enthält.*

Übung: Berechne die Attention für „sitzt"

Jetzt bist du **„sitzt"**! Berechne deine Attention-Gewichte.
Q_{sitzt} = **[0.9, -0.3]**

Schritt 1: Scores berechnen ($Q_{sitzt} \cdot K_{jedes\ Token}$)

✏️ Selbst rechnen

Von sitzt → zu	K-Vektor	Rechnung	Score
Die	[0.0, 1.1]	(0.9 × 0.0) + (-0.3 × 1.1) =	
Katze	[0.1, 1.2]	(0.9 × 0.1) + (-0.3 × 1.2) =	
sitzt	[0.9, 1.0]	(0.9 × 0.9) + (-0.3 × 1.0) =	
auf	[0.3, 1.4]	(0.9 × 0.3) + (-0.3 × 1.4) =	
der	[0.0, 1.1]	(0.9 × 0.0) + (-0.3 × 1.1) =	
Matte	[0.1, 1.9]	(0.9 × 0.1) + (-0.3 × 1.9) =	

Schritt 2: Durch √2 = 1.41 teilen

✏️ Selbst rechnen

Skalierte Scores: ___, ___, ___, ___, ___, ___

Schritt 3: Softmax (e^{score}, Summe, teilen)

✏️ Selbst rechnen

e^{scores}: ___, ___, ___, ___, ___, ___

Summe: ___

Gewichte: ___, ___, ___, ___, ___, ___

Schritt 4: Gewichtete Summe der Values

✏️ Selbst rechnen

$\text{Output}_{\text{sitzt}}$ = [___, ___]

Tipp: „sitzt" hat eine negative Zahl in Q (-0.3). Dadurch werden Tokens mit hohen K_2-Werten *abgestraft*. Schau genau hin — wer leuchtet für „sitzt" am hellsten? Ergibt das sprachlich Sinn?

Das Gesamtbild: Die Attention-Matrix

> Wenn wir die Attention-Gewichte für **alle** Tokens berechnen, bekommen wir eine **6×6 Matrix**.
> Jede Zeile zeigt: Worauf achtet dieses Token?

↓ schaut auf →	Die	Katze	sitzt	auf	der	Matte
Die						
Katze	0.12	0.14	0.17	0.19	0.12	**0.27**
sitzt	*← Deine Übung von Seite 6!*					
auf						
der						
Matte						

Jede Zeile summiert sich zu 1.00 (= 100%)

> **Was die Matrix zeigt:**
> Jede Zeile ist eine „Aufmerksamkeits-Verteilung". Sie sagt: *Dieses Token holt X% seiner neuen Information von Token Y.*
>
> In einem trainierten Transformer zeigt diese Matrix faszinierende Muster: Subjekte achten auf ihre Verben, Adjektive auf ihre Nomen, Pronomen auf das Wort, das sie ersetzen.

Zusammenfassung: Der komplette Attention-Vorgang

Schritt	Was passiert	Formel	Bei uns
1. Projizieren	Jedes Token bekommt Q, K, V	$Q = X \cdot W_Q$ $K = X \cdot W_K$ $V = X \cdot W_V$	4D → 2D
2. Scores	Wie gut passt meine Frage zu deinem Namensschild?	$Score = Q \cdot K^T$	6×6 Matrix
3. Skalieren	Scores in guten Bereich bringen	$Score / \sqrt{d_k}$	÷ 1.41

4. Softmax	Leuchten auf 100% normieren	$e^{score} / \Sigma e^{scores}$	→ Gewichte
5. Gewichtete Summe	Pakete einsammeln, gewichtet	Output = Σ(Gewicht × V)	→ neuer 2D-Vektor

Tokenizer ✓ → **Embedding ✓** → **Pos. Encoding ✓** → **Self-Attention ✓** → **Feed-Forward** → **Output**

Erlebnispark — Zwischenstand: *Du hast den wichtigsten Raum durchlaufen. Jedes Token weiß jetzt nicht nur was es selbst bedeutet, sondern auch was die anderen Tokens im Satz für es bedeuten.*

Als Nächstes: *Multi-Head Attention (12 Räume gleichzeitig) und das Feed-Forward-Netzwerk (wo jedes Token für sich allein nochmal nachdenkt — ohne die anderen).*

Und dann? Dann machst du das Ganze noch 11 Mal. Willkommen im Transformer.