Schritt 1b: BPE — Byte Pair Encoding

Tokenizer (BPE) → Embedding → Pos. Encoding → Self-Attention → Feed-Forward → Output

> **Das Problem:** Word-Level hat zu viele Tokens im Vokabular. Character-Level hat zu viele Tokens pro Satz. **BPE findet den Mittelweg** — automatisch!

Der BPE-Algorithmus in 3 Sätzen

1. **Starte mit Buchstaben.** Jeder Buchstabe ist ein eigenes Token (= Character-Level).

2. **Zähle alle Nachbar-Paare.** Welche zwei Tokens stehen am häufigsten nebeneinander?

3. **Verschmelze das häufigste Paar.** Die zwei Tokens werden eins. Zurück zu Schritt 2. Wiederhole, bis dein Vokabular die gewünschte Größe hat.

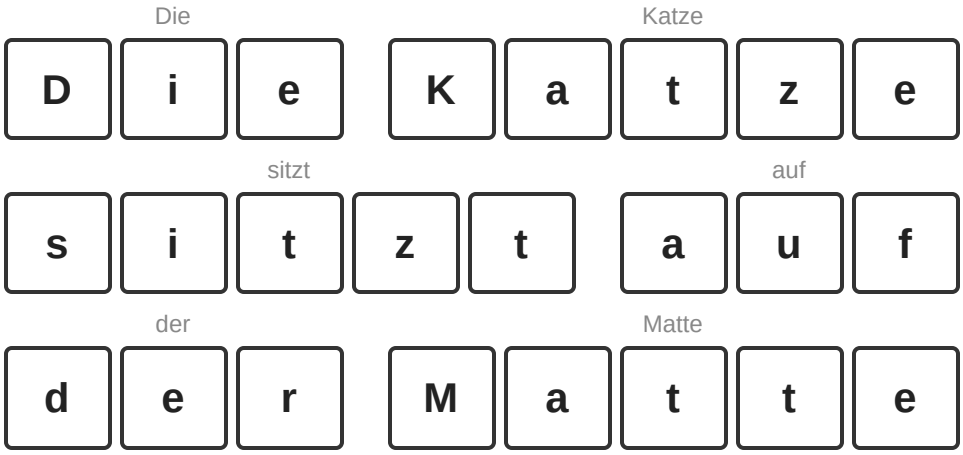
> **Wichtig:** BPE lernt die Merges aus einem großen Textkorpus (Millionen von Sätzen). Unser einzelner Satz ist nur ein Mini-Beispiel, um den Ablauf zu verstehen. In der Realität sind die Häufigkeits-Unterschiede viel deutlicher.

Unser Beispiel: „Die Katze sitzt auf der Matte"

Wir arbeiten **innerhalb der Wortgrenzen** — Paare werden nur innerhalb eines Wortes gezählt, nicht über Leerzeichen hinweg.

Runde 0 — Start: Jeder Buchstabe einzeln

Schneide jeden Buchstaben aus (wie Blatt 3 vom Tokenizer). Lege die Wörter gruppiert hin:



Gesamt: 25 Tokens (ohne Leerzeichen)

Alle Nachbar-Paare zählen:

Paar	Wo?	Anzahl
a → t	K**at**ze, M**at**te	**2** ✓
t → z	Ka**tz**e, si**tz**t	**2** ✓
D → i	Die	1
i → e	Die	1
K → a	Katze	1
z → e	Katze	1
s → i	sitzt	1
i → t	sitzt	1
z → t	sitzt	1
a → u	auf	1
u → f	auf	1
d → e	der	1

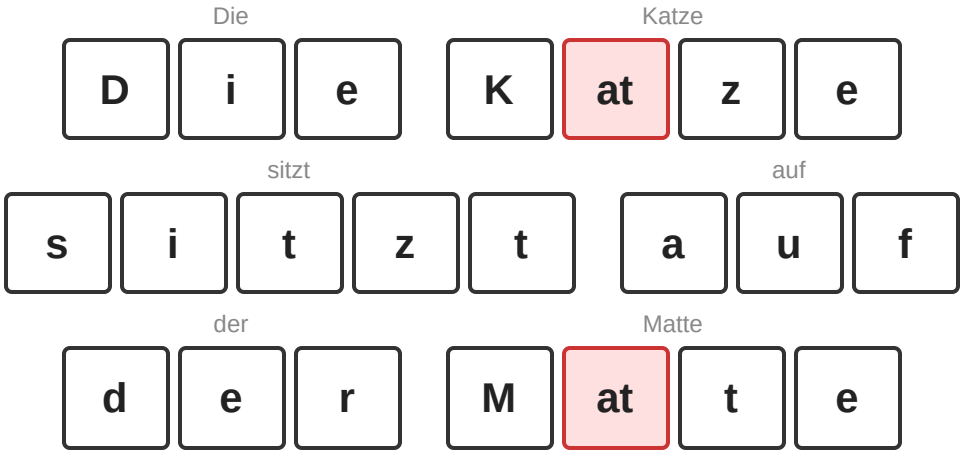
e → r	der	1
M → a	Matte	1
t → t	Matte	1
t → e	Matte	1

Gewinner: `a + t` und `t + z` — beide 2×. Bei Gleichstand nehmen wir das erste: `a + t`

Merge-Regel #1: **a + t → at**

Auf dem Tisch: Klebe überall, wo `a` direkt neben `t` liegt, die beiden Kärtchen zusammen zu einem neuen Kärtchen `at`.

Runde 1 — Nach dem Merge: a + t → at



Gesamt: 23 Tokens (2 weniger als vorher — die 2 Merges)

Neue Nachbar-Paare zählen:

> **Achtung:** `at` ist jetzt EIN Token! Die Paare ändern sich. In „Katze" ist es jetzt `K→at`, `at→z`, `z→e` (nicht mehr `a→t`, `t→z`).

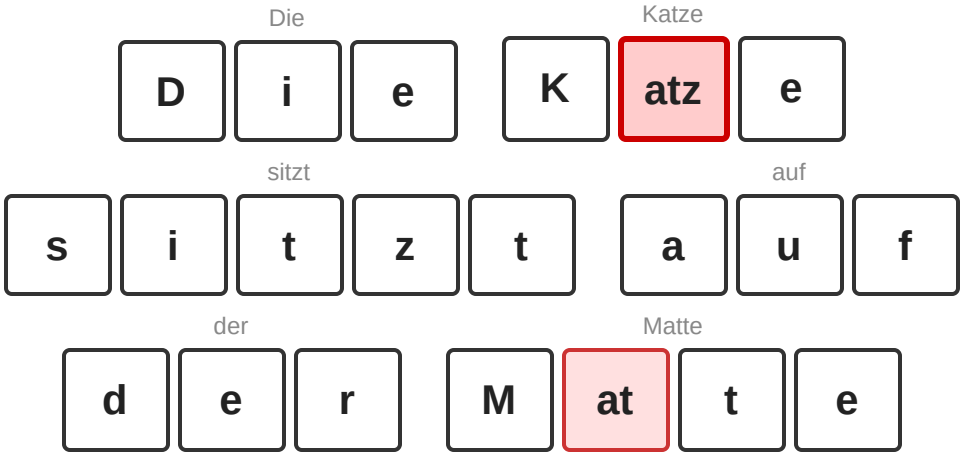
Paar	Wo?	Anzahl
D → i	Die	1
i → e	Die	1
K → at	Katze	1
at → z	Katze	1
z → e	Katze	1
s → i	sitzt	1
i → t	sitzt	1
t → z	sitzt	1
z → t	sitzt	1
a → u	auf	1
u → f	auf	1

d → e	der	1
e → r	der	1
M → at	Matte	1
at → t	Matte	1
t → e	Matte	1

> **Alles ist 1×!** Bei nur einem Satz gibt es nach dem ersten Merge keinen klaren Gewinner mehr. In der Realität trainiert BPE auf **Millionen von Sätzen** — da gibt es immer einen Gewinner.
>
> Für unser Beispiel wählen wir: `at + z → atz` (um „Katze" weiter zusammenzubauen).

Merge-Regel #2: **at + z → atz**

Runde 2 — Nach dem Merge: at + z → atz



Gesamt: 22 Tokens

> **Beachte:** In „Katze" wurde `at→atz`, aber in „Matte" nicht! Dort war es `at + t`, nicht `at + z`. BPE wendet nur die exakte Regel an.

Jetzt bist du dran! Zähle die Paare:

✏️ Hier selbst ausfüllen

Paar	Wo?	Anzahl

Welches Paar wählst du?

✏️ Hier selbst ausfüllen

Merge-Regel #3: _____ + _____ → _____

Runde 3 — Dein Merge von Runde 2 anwenden

Zeichne oder klebe die neuen Token-Kärtchen hier ein:

✏️ Hier selbst ausfüllen

Die: ___ ___ ___

Katze: ___ ___ ___

sitzt: ___ ___ ___ ___ ___

auf: ___ ___ ___

der: ___ ___ ___

Matte: ___ ___ ___ ___

Paare zählen:

✏️ Hier selbst ausfüllen

Paar	Wo?	Anzahl

Merge-Regel #4:

✏️ Hier selbst ausfüllen

_____ + _____ → _____

Das Ergebnis: Dein BPE-Vokabular

> Nach mehreren Runden hast du ein Vokabular, das zwischen Buchstaben und ganzen Wörtern liegt. Trag hier ein, welche Tokens du am Ende hast:

Merge-Regeln (in Reihenfolge):

#	Regel	Neues Token
1	a + t → at	at
2	at + z → atz	atz
3		
4		
5		
6		

So würde BPE „Die Katze sitzt auf der Matte" am Ende tokenisieren:

✏️ Hier selbst ausfüllen

Trag deine finalen Tokens hier ein: [___, ___, ___, ...]

Vergleich: Wie viele Tokens?

Methode	Tokens	Vokabular
Character-Level	25	14 Zeichen
BPE (dein Ergebnis)		
Word-Level	6	6 Wörter

Die wichtigste Erkenntnis

BPE baut Tokens von unten auf:

Buchstaben → häufige Paare → häufige Teilwörter → ganze Wörter

```
a → at → atz → atze → Katze
```

Je häufiger ein Wort im Trainingstext vorkommt, desto eher wird es zu einem einzigen Token. Seltene Wörter bleiben in Teilen — aber das ist OK, weil die Teile trotzdem bekannt sind.

Deshalb kann GPT auch Wörter verarbeiten, die es nie gesehen hat!

Nächstes Blatt: Die Embedding-Tabelle — jede Token-ID bekommt einen Zahlen-Vektor