Prompt, Kontext oder Training?

Die drei Wege, ein KI-Modell an DEINE Aufgabe anzupassen — mit Entscheidungs-Assistent, ehrlichem Aufwands-Vergleich und der LoRA-Grafik, die zeigt, wie wenig „Training“ manchmal ist.

Kapitel online

Aktuell · Kapitel 37 von 50 · ca. 10 Min. · 300 XP

Einfach gesagt: Meist reicht eine bessere Anweisung oder mitgegebenes Wissen — nur wenn du das Verhalten selbst dauerhaft ändern willst, lohnt echtes Nachtrainieren.

„Wir müssen unser eigenes Modell trainieren!“ — dieser Satz fällt in Firmen ständig. Und meistens ist er falsch. Es gibt **drei Wege**, ein Sprachmodell an deine Aufgabe anzupassen, und sie unterscheiden sich um Größenordnungen in Aufwand und Kosten. Nach diesem Kapitel weißt du, welcher Weg wann dran ist.

Was du hier lernst

- Die **drei Wege**, ein Sprachmodell anzupassen: Prompt, RAG, Fine-Tuning
- Die **Faustregel**, die 90 % der Fälle entscheidet: Wissen → RAG, Verhalten → Fine-Tuning
- Was jeder Weg anfasst — den **Eingabe-Tensor** oder die **Gewichts-Tensoren**
- Warum **LoRA** mit eingefrorenen Riesen-Matrizen auf eine einzelne GPU passt
- Welche **drei Gegenfragen** du im nächsten „Wir trainieren selbst!“-Meeting parat hast

Für den Unterricht

Einstieg (2 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (6 Min): die interaktiven Module auf herrki.de/370_finetuning_vs_prompting ausprobieren.

Sichern mit Quiz (2 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

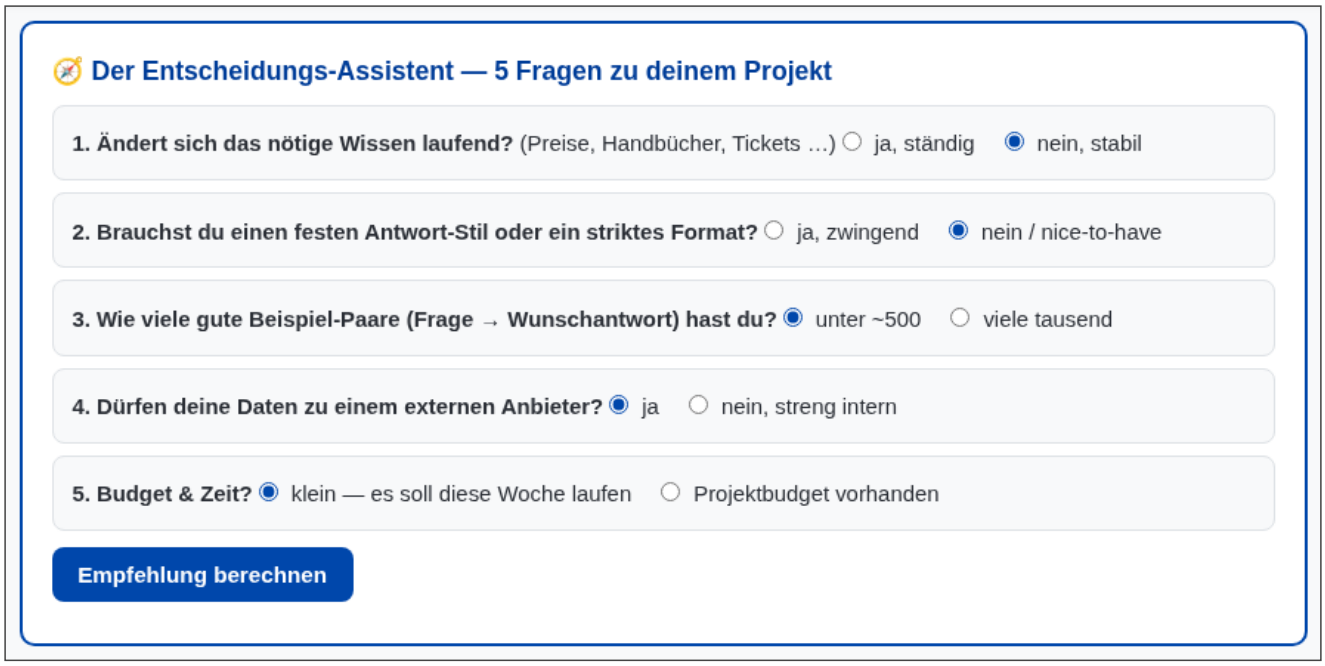
Die drei Wege — und was dabei WIRKLICH passiert

Weg	Was passiert technisch?	Aufwand	Typische Kosten
1. Besserer Prompt	Nur der *Eingabe-Tensor* ändert sich — Anweisungen, Beispiele, Rollen.	Minuten bis Stunden	≈ 0 €
2. Wissen per RAG	Gefundene Dokument-Absätze werden zur Laufzeit *an den Eingabe-Tensor angehängt* (Kapitel 330).	Stunden bis Tage	wenig (Embedding + Speicher)
3. Fine-Tuning / LoRA	Die *Gewichts-Tensoren* des Modells werden weitertrainiert — das Verhalten selbst ändert sich.	Tage + Datensatz + GPU	deutlich mehr (+ Pflege!)

Die Faustregel, die 90 % aller Fälle richtig entscheidet:
Wissen („Was steht in unseren Unterlagen?“) → **RAG**.
Verhalten, Ton, Format („Antworte IMMER wie unser Support, in unserem Schema“) → **Fine-Tuning**.
Alles andere → erst einmal **am Prompt arbeiten** — kostenlos und sofort.

Am Gerät ausprobieren: herrki.de/370_finetuning_vs_prompting#abschnitt-2

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

„Eigenes Modell trainieren“ — wie viel Training ist das wirklich? (LoRA)

Selbst Profis trainieren heute selten alle Gewichte neu. Der Standard heißt **LoRA** (Low-Rank Adaptation): Die riesigen Gewichts-Matrizen bleiben **eingefroren**, daneben werden winzige Zusatz-Matrizen gelernt, deren Wirkung addiert wird:

Am Gerät ausprobieren: herrki.de/370_finetuning_vs_prompting#abschnitt-3

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

```
// Statt die Riesen-Matrix W zu veraendern:
ausgabe = eingabe.matMul( W.add( A.matMul(B) ) )
// W bleibt eingefroren (Milliarden Werte) – gelernt werden nur A und B (Millionen Werte)
```

Bei einem 7-Milliarden-Parameter-Modell sind typischerweise nur **wenige Millionen** Werte trainierbar — Größenordnung **tausendfach weniger** (die genaue Zahl hängt von der LoRA-Konfiguration ab). Deshalb läuft so ein Training auf einer einzelnen GPU statt im Rechenzentrum.

Runter zum Tensor — die drei Wege in Zahlen

Nimm eine konkrete Frage an euren Support-Bot: *„Was kostet Artikel 4711?“* Als Tokens ist das ein kleiner Eingabe-Tensor. Schau, was jeder der drei Wege daran verändert — und was er **nicht** anfasst (die Token-Zahlen sind Beispielwerte, die Größenordnung stimmt):

```
// 1) BESSERER PROMPT — nur die Eingabe waechst
Eingabe vorher [8]: „Was kostet Artikel 4711 ?“
Eingabe nachher [46]: Rolle + Format-Regeln + Beispielantwort + dieselbe Frage
Gewichte: unveraendert — 0 Werte angefasst

// 2) RAG — derselbe Tensor, nur laenger
Eingabe [389]: 46 Prompt-Tokens + 343 Tokens aus der aktuellen Preisliste
Gewichte: unveraendert — 0 Werte angefasst

// 3) FINE-TUNING / LoRA — hier aendern sich zum ersten Mal die GEWICHTE
Eingabe [8]: unveraendert
Gewichte: verschoben — und zwar so:
```

Rechnen wir den dritten Weg nach. Die Gewichts-Matrix `W` ist hier **4×4** (zum Mitrechnen verkleinert — in echt z. B. 4096×4096 = 16.777.216 Werte). Daneben stehen die winzigen LoRA-Matrizen `A` und `B`; ihr Produkt ist die Verschiebung ΔW:

```
A [4×1] als Spalte: 0.50 −0.20 0.10 0.30
B [1×4] als Zeile: 0.40 0.00 −0.60 0.20

// ΔW = A·B — jeder Eintrag ist EIN Produkt aus einem A- und einem B-Wert:
ΔW Zeile 1, Spalte 1: 0.50 · 0.40 = 0.20
ΔW Zeile 1, Spalte 3: 0.50 · (−0.60) = −0.30
ΔW Zeile 2, Spalte 1: (−0.20) · 0.40 = −0.08

ΔW [4×4]: 0.20 0.00 −0.30 0.10
          −0.08 0.00 0.12 −0.04
           0.04 0.00 −0.06 0.02
           0.12 0.00 −0.18 0.06
```

Und jetzt der Punkt, auf den es ankommt: `W` selbst wird **nie überschrieben**. Beim Rechnen wird ΔW einfach daraufgelegt:

```
W Zeile 1: 0.31 −0.08 0.44 0.05 ← eingefroren, unantastbar
ΔW Zeile 1: 0.20 0.00 −0.30 0.10 ← das frisch Gelernte
W + ΔW Zeile 1: 0.51 −0.08 0.14 0.15 ← was das Modell benutzt

// Und wie viele Werte muss das Training ueberhaupt anfassen?
hier: A(4) + B(4) = 8 Werte statt W = 16 Werte
in echt: 4096·8 + 8·4096 = 65.536 Werte statt W = 16.777.216 Werte
                                                  → 256-mal weniger
```

Das ist die Rechnung für *eine* Matrix bei LoRA-Rang 8. Über ein ganzes Modell gerechnet — und nicht jede Matrix bekommt überhaupt einen Adapter — kommt die Größenordnung von

oben heraus. Genau deshalb reicht eine einzelne GPU.

> **Prompt** und **RAG** verändern **nur den Eingabe-Tensor** [8] → [389] — er wird umgebaut oder verlängert. Die Gewichte bleiben Bit für Bit dieselben; du kannst es jederzeit zurücknehmen.
>
> **Fine-Tuning** verschiebt die **Gewichts-Tensoren** [4096, 4096] per Gradient Descent (Kapitel 118) — bei LoRA über die kleinen Zusatz-Matrizen A und B. Das Verhalten selbst ändert sich, und zwar dauerhaft.

Zurück zur Anwendung

Wenn im nächsten Meeting „eigenes Modell!“ gerufen wird, hast du jetzt die drei Gegenfragen parat: *Ist es ein Wissens- oder ein Verhaltens-Problem? Wie viele Beispiel-Paare haben wir wirklich? Und haben wir den Prompt überhaupt ausgereizt?* In der Praxis ist die beste Lösung übrigens oft eine **Kombination**: sauberer Prompt + RAG — und Fine-Tuning erst, wenn beides nachweislich nicht reicht.

Zum Weiterlesen

- **LoRA im Original:** Edward Hu u. a., *„LoRA: Low-Rank Adaptation of Large Language Models“*, 2021 — arxiv.org/abs/2106.09685 — die eingefrorenen Riesen-Matrizen und die kleinen Zusatz-Matrizen A und B von oben
- **Noch eine Größenordnung billiger:** Tim Dettmers u. a., *„QLoRA“*, 2023 — arxiv.org/abs/2305.14314 — warum ein 65-Milliarden-Modell auf eine einzelne GPU passt
- **Warum der Prompt-Weg überhaupt funktioniert:** Tom Brown u. a., *„Language Models are Few-Shot Learners“* (GPT-3), 2020 — arxiv.org/abs/2005.14165 — hier wurde gezeigt, dass Beispiele im Prompt oft reichen, wo man früher trainiert hätte
- **Der RAG-Weg:** Patrick Lewis u. a., *„Retrieval-Augmented Generation“*, 2020 — arxiv.org/abs/2005.11401 — Wissen zur Laufzeit nachreichen statt Gewichte anfassen

Kurzer Check – mit Lösung

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

Frage 1: Eure Preisliste ändert sich wöchentlich — der Chatbot soll sie kennen. Welcher Weg?

- Fine-Tuning — jede Woche neu trainieren.
- **✓ RAG — die aktuelle Liste wird zur Laufzeit gefunden und in den Prompt gelegt.**
- Gar nicht möglich.

Wissen → RAG. Fine-Tuning „brennt" den Stand von heute in die Gewichte — nächste Woche ist er falsch, und niemand trainiert wöchentlich neu.

Frage 2: Was wird beim Fine-Tuning verändert — im Unterschied zu Prompt und RAG?

- **✓ Die Gewichts-Tensoren des Modells selbst; Prompt/RAG verändern nur die Eingabe.**
- Nur die Benutzeroberfläche.
- Die Trainingsdaten des ursprünglichen Herstellers.

Genau die Grenze zwischen „Eingabe anpassen" (billig, reversibel, sofort) und „Verhalten antrainieren" (teuer, mächtig, pflegebedürftig).

Frage 3: Warum ist LoRA so viel billiger als volles Nachtrainieren?

- Weil es das Modell verkleinert und Wissen löscht.
- Weil es ohne Trainingsdaten auskommt.
- **✓ Weil die Riesen-Matrizen eingefroren bleiben und nur winzige Zusatz-Matrizen gelernt werden.**

Milliarden Gewichte bleiben stehen, gelernt werden nur die kleinen A·B-Matrizen daneben — tausendfach weniger Werte, eine GPU statt Rechenzentrum.

Das nimmst du mit

- **Es sind drei Wege, nicht einer.** Prompt, RAG und Fine-Tuning unterscheiden sich um Größenordnungen in Aufwand und Kosten — „eigenes Modell“ ist fast nie der erste Schritt.
- **Wissen → RAG, Verhalten → Fine-Tuning.** Diese eine Faustregel entscheidet 90 % aller Fälle richtig.
- **Prompt und RAG fassen den Eingabe-Tensor an, Fine-Tuning die Gewichte.** Nur der dritte Weg verändert das Modell dauerhaft — und nur er kostet Training.
- **LoRA friert die Riesen-Matrizen ein.** Gelernt werden nur die kleinen Zusatz-Matrizen daneben: tausendfach weniger Werte, eine GPU statt Rechenzentrum.
- **Drei Gegenfragen fürs Meeting.** Wissens- oder Verhaltens-Problem? Wie viele Beispiel-Paare haben wir wirklich? Und ist der Prompt überhaupt ausgereizt?

Weiter geht’s: Alle drei Wege ändern, was das Modell **weiß**. Im nächsten Kapitel ändert sich, was es **tun darf**: „Agenten: KI, die Werkzeuge benutzt“.

So zitierst du dieses Kapitel

HerrKI: Prompt, Kontext oder Training?. herrki.de/370_finetuning_vs_prompting, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.