Context Engineering *für den Unterricht*

Vom naiven Prompt zum strukturierten Kontext. Ein praxisorientierter Leitfaden für Lehrkräfte

Stand: März 2026 | Seminar LFB KI Autor: Thomas Jörg

1 Bevor Sie anfangen: Wichtige Grundregeln

1.1 Niemals persönliche Daten in Cloud-KIs eingeben

Geben Sie keine Namen, Beziehungen, Einkommen, Krankheiten, politische Ansichten oder andere persönliche Daten in Cloud-basierte KI-Systeme ein. Das gilt für Sie *und* für Ihre Schülerinnen und Schüler.

Warum ist das wichtig?

- Profiling: KI-Systeme können aus scheinbar harmlosen Eingaben Rückschlüsse auf Persönlichkeit und Verhalten ziehen.
- Deanonymisierung: Metadaten (IP, Geräte-Fingerprints) können Nutzer identifizierbar machen – auch ohne Registrierung.
- Serverstandorte: ChatGPT, Gemini und Claude laufen auf US-Servern (CLOUD Act). Europäische Alternative: Mistral.ai (Server in EU, DSGVO-konform).
- EU AI Act (seit August 2025 in Kraft): Klassifiziert KI-Systeme nach Risikostufen. Social Scoring und biometrische Massenüberwachung sind verboten.

1.2 Garbage In, Garbage Out

Das GIGO-Prinzip gilt mehr denn je: Ungenaue, verwirrende oder kontextlose Eingaben führen zu schlechten Antworten. Umgekehrt gilt: Präzise formulieren + ausreichend Kontext liefern + Struktur geben = qualitativ hochwertige Ergebnisse.

Dieser Leitfaden zeigt Ihnen systematisch, wie Sie vom einfachen Ein-Satz-Prompt zu einem durchdachten, strukturierten Kontext gelangen – und warum dieser Unterschied so groß ist.

1.3 Allgemeine Grundregeln im Umgang mit LLMs

- Klarheit: Formulieren Sie Anfragen präzise und ohne Mehrdeutigkeiten.
- Kontext: Je mehr relevante Hintergrundinformation, desto besser die Antwort.
- Spezifität: Nennen Sie Zielgruppe, Format, Länge, Sprache und Schwierigkeitsgrad.
- Iteration: Arbeiten Sie in Schleifen – testen, Schwäche finden, gezielt ergänzen, erneut testen.
- Unsicherheit einfordern: Fordern Sie das Modell auf, Nichtwissen zuzugeben statt zu halluzinieren.

2 Von Prompt Engineering zu Context Engineering

Im Juni 2025 prägte Andrej Karpathy (ehemaliger KI-Chef bei Tesla, OpenAI-Mitgründer) einen neuen Begriff: ***Context Engineering***. Seine Aussage: „Ich bevorzuge ‘Context Engineering’ gegenüber ‘Prompt Engineering’, weil es die eigentliche Kompetenz besser beschreibt: die Kunst, dem LLM genau den richtigen Kontext bereitzustellen, damit es die Aufgabe lösen kann.“

> **Karpathys CPU-Analogie**
>
> Das LLM ist die CPU – der Prozessor, der die eigentliche Arbeit macht. Das Context Window ist der Arbeitsspeicher, in dem alles liegt, worauf das Modell zugreifen kann – du entscheidest, was dort hinein geladen wird.
>
> *Zu wenig Kontext: Das Modell muss raten und halluziniert.*
> *Zu viel Kontext: Die Kosten steigen, die Qualität sinkt. Die richtige Balance zu finden ist die eigentliche Kunst.*

Prompt Engineering (alt)	Context Engineering (neu)
„Stell eine gute Frage“	„Baue die gesamte Umgebung auf“
Ein einzelner Satz oder Absatz	System-Prompt + Rolle + Beispiele + Daten + Tools
Fokus auf geschickte Formulierung	Fokus auf Informationsarchitektur
Trial & Error	Systematische Iteration mit Evaluation
Statisch: einmal geschrieben, fertig	Dynamisch: passt sich an Aufgabe und Nutzer an

Für den Unterricht bedeutet das: Wir bringen Schülerinnen und Schülern nicht bei, „gute Fragen“ zu stellen. Wir bringen ihnen bei, **systematisch den richtigen Kontext aufzubauen**. Das ist strukturiertes Denken, Informationsarchitektur und kritische Reflexion in einem.

2.1 Wie lang sollte ein Kontext sein?

Mehr Kontext ist nicht automatisch besser. Aktuelle Studien zeigen, dass übermäßig lange Instruktionen die Qualität der Antworten sogar verschlechtern können – selbst wenn das technische Kontextfenster noch längst nicht ausgeschöpft ist. Der praktische Sweet Spot für die meisten Aufgaben liegt bei etwa **150–300 Wörtern** Instruktionstext.

> **Faustregel: Schlank und gezielt**
>
> 1. Schreiben Sie die kürzeste Version, die Ihre Absicht beschreibt und dann testen.
> 2. Identifizieren Sie, was konkret falsch oder fehlend ist in der Ausgabe.
> 3. Fügen Sie NUR das hinzu, was dieses spezifische Problem behebt.
>
> *Wenn man das wiederholt, ist das Ergebnis ein schlanker, gezielter Kontext – statt eines 500-Wörter-Textes.*

2.2 Warum GROSSBUCHSTABEN-Kürzel statt XML-Tags?

In der professionellen Prompt-Entwicklung werden oft XML-artige Tags verwendet (`<rolle>`, `<aufgabe>` usw.). Für den Unterrichtsalltag sind diese unpraktisch: Auf einer deutschen Tastatur sind spitze Klammern umständlich zu tippen, und die Syntax wirkt für Nicht-Informatiker abschreckend.

Moderne Sprachmodelle erkennen **GROSSBUCHSTABEN-Kürzel mit Doppelpunkt** (z. B. `ROLLE:`, `AUFGABE:`, `FORMAT:`) genauso zuverlässig als Strukturierungssignal. Sie sind darauf trainiert und nutzen solche Markierungen zur Aufmerksamkeitssteuerung. Der Vorteil: schneller zu tippen, intuitiv lesbar, kein technisches Vorwissen nötig.

3 Kurzreferenz: Die 7 Techniken auf einen Blick

Diese Seite eignet sich zum Ausdrucken und Aufhängen neben dem Monitor.

#	Technik	Kernidee	Beispiel (Kürzel)
1	**System-Prompt / Rollenanweisung**	Definiere VOR dem Gespräch, WER das Modell ist, WIE es antworten soll und welche REGELN gelten.	`ROLLE: Du bist ein geduldiger Python-Tutor für Anfänger. Gib nie vollständige Lösungen.`
2	**Strukturiertes Prompting**	Trenne Rolle, Aufgabe, Kontext und Format durch Kürzel wie ROLLE:, AUFGABE:, FORMAT:.	`FACH: Geographie AUFGABE: Erstelle 5 Quizfragen FORMAT: Multiple Choice`
3	**Few-Shot-Beispiele**	Zeige 2–3 Muster des gewünschten Eingabe-Ausgabe-Paares. Das Modell leitet das Format daraus ab.	`BEISPIEL 1: Text: „Das Wetter wird wärmer.“ Bewertung: Sehr allgemein, keine Belege. Gib nun Feedback auf: […]`
4	**Chain-of-Thought (CoT)**	Fordere das Modell auf, Schritt für Schritt zu denken. Reduziert Fehler bei komplexen Aufgaben.	`SCHRITT 1: Nenne die Hauptthese. SCHRITT 2: Identifiziere den Beleg. SCHRITT 3: Prüfe den logischen Zusammenhang.`
5	**Constraints / Guardrails**	Setze explizite Grenzen: Was soll das Modell NICHT tun? Maximalwerte, Verbote, Unsicherheitsmarker.	`EINSCHRÄNKUNGEN: – Nie vollständige Lösungen – Max. 100 Wörter – Markiere Unsicheres mit [UNSICHER]`
6	**Self-Consistency**	Lass das Modell dieselbe Aufgabe auf mehreren Wegen lösen. Die konsistenteste Antwort ist die zuverlässigste.	`Löse auf DREI verschiedene Weisen. Vergleiche die Ergebnisse. Empfehle den besten Weg.`
7	**Prompt Chaining (Verkettung)**	Zerlege komplexe Aufgaben in Schritte. Die Ausgabe von Schritt 1 wird Input für Schritt 2.	`Schritt 1: Fasse den Text zusammen. Schritt 2: Erstelle daraus Quizfragen. Schritt 3: Formuliere einen Erwartungshorizont.`

Die 5 Prinzipien des Context Engineering

1. ROLLE definieren – Gib dem Modell eine klare Identität im System-Prompt.
2. STRUKTUR geben – Nutze Kürzel wie ROLLE:, AUFGABE:, FORMAT: für klare Trennung.
3. ZEIGEN statt erklären – Gib 2–3 Musterbeispiele. Das Modell leitet das Format daraus ab.
4. SCHRITTE vorgeben – Explizite Schritt-für-Schritt-Anweisung erzwingt präziseres Denken.
5. GRENZEN setzen – Definiere explizit, was das Modell NICHT tun soll.

4 Techniken im Detail

4.1 System-Prompt / Rollenanweisung

Der System-Prompt ist die wichtigste Neuerung gegenüber dem einfachen Prompting. Er wird *vor* dem eigentlichen Gespräch gesetzt und definiert, **wer** das Modell ist, **wie** es sich verhalten soll und **welche Regeln** gelten. Vergleichbar mit der Rollenbeschreibung eines Schauspielers, bevor die Szene beginnt. Er gilt für das gesamte Gespräch – jede nachfolgende Frage wird im Licht dieser Regeln beantwortet.

Beispiel: Chemie-Tutor für die Mittelstufe

```
ROLLE: Du bist ein freundlicher Chemie-Nachhilfelehrer für Schüler der 9. Klasse Realschule.
REGELN:
- Antworte immer auf Deutsch.
- Verwende maximal 120 Wörter pro Antwort.
- Nutze Alltagsbeispiele (Kochen, Putzmittel, Natur).
- Wenn der Schüler einen Fehler macht, stelle eine Rückfrage statt sofort zu korrigieren.
- Gib NIE vollständige Lösungen für Übungsaufgaben.
- Wenn du unsicher bist, sage es ehrlich.
```

Praxistipp: Wo stellt man den System-Prompt ein?

- ChatGPT: über „Custom Instructions“ oder die API
- Claude: über das „System“-Feld (Projekte oder API)
- LM Studio / AnythingLLM: im Konfigurationsbereich des Workspace
- Web-Oberfläche: Den System-Prompt als Erstes eintippen, bevor die eigentlichen Fragen beginnen.

4.2 Strukturiertes Prompting (GROSSBUCHSTABEN-Kürzel)

Durch Kürzel wie `ROLLE:`, `AUFGABE:`, `FORMAT:` oder `RAHMENBEDINGUNGEN:` trennen Sie verschiedene Informationsarten sauber voneinander. Das Modell erkennt: „Hier endet die Rollenanweisung, hier beginnt die eigentliche Aufgabe.“ Das reduziert Missverständnisse drastisch.

Beispiel: Aufgabenerstellung für Informatik

```
ROLLE: Du bist Informatik-Lehrer an einem Gymnasium.
AUFGABE: Erstelle 4 Aufgaben zum Thema Rekursion in Python.
RAHMENBEDINGUNGEN:
- Klasse 11, Grundkurs
- 2 leichte, 1 mittlere, 1 schwere Aufgabe
- Jede Aufgabe mit Punktzahl und Musterlösung
FORMAT:
- Trenne Aufgabenblatt und Erwartungshorizont.
- Nummeriere alle Aufgaben.
```

Übung für Schüler: Dieselbe Aufgabe einmal unstrukturiert („Mach mir Aufgaben zu Rekursion“) und einmal mit Kürzeln formulieren. Antwortqualität vergleichen – der Unterschied ist oft verblüffend.

4.3 Few-Shot-Beispiele: Zeigen statt erklären

Statt dem Modell lang und breit zu *erklären*, was man möchte, ist es oft effektiver, es zu *zeigen*. Man gibt 2–3 Beispiele des gewünschten Eingabe-Ausgabe-Paares, und das Modell leitet daraus das Muster ab.

Beispiel: Feedback-Stil für Schülertexte

```
Gib Feedback auf Schülertexte nach genau diesem Muster:

BEISPIEL 1:
Text: "Das Wetter wird wärmer. Das ist nicht gut."
Beobachtung: Sehr allgemein, keine konkreten Belege.
Stärke: Klarer Standpunkt erkennbar.
Nächster Schritt: Eine konkrete Zahl oder ein Beispiel ergänzen.

BEISPIEL 2:
Text: "Laut IPCC-Bericht steigen die Temperaturen um 1,5 Grad."
Beobachtung: Guter Beleg, klarer Bezug zur Aussage.
Stärke: Wissenschaftliche Quelle korrekt eingesetzt.
Nächster Schritt: Den Begriff 'bedroht' konkretisieren.

Gib nun Feedback auf diesen Text: [Schülertext hier einfügen]
```

> **Achtung: Beispiele werden genau imitiert!**
>
> Aktuelle Forschung zeigt, dass Modelle die Details in Beispielen genau übernehmen. Wenn Ihre Beispiele Fehler enthalten oder einen bestimmten Stil zeigen, wird das Modell dies imitieren. Beispiele sind daher sowohl mächtig als auch riskant – ein gutes Thema für eine Unterrichtsdiskussion über Bias in KI-Systemen.

4.4 Chain-of-Thought: Schritt für Schritt denken lassen

Chain-of-Thought (CoT) fordert das Modell auf, seinen Denkweg offenzulegen, bevor es eine Antwort gibt. Das reduziert Fehler erheblich. Für den Unterricht bietet CoT eine doppelte Chance: Erstens liefert es bessere Ergebnisse. Zweitens macht es den Denkprozess der KI transparent – und damit kritisierbar. Bereits der Zusatz „Denke Schritt für Schritt“ am Ende eines Prompts verbessert die Ergebnisse merklich.

Beispiel: Code-Analyse mit transparentem Denkweg

```
AUFGABE: Analysiere den folgenden Code.Gehe dabei Schritt für Schritt vor:

SCHRITT 1: Lies den Code Zeile für Zeile.
SCHRITT 2: Beschreibe, was jede Zeile tut.
SCHRITT 3: Identifiziere mögliche Fehler oder Schwachstellen.
SCHRITT 4: Gib erst DANN dein Gesamturteil ab.

CODE:
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)
```

Übung: Schüler vergleichen die Antwort mit CoT („Schritt für Schritt“) und ohne CoT („Analysiere diesen Code“). Sie dokumentieren: Welche Fehler wurden jeweils gefunden? Wie detailliert ist die Analyse?

4.5 Constraints und Guardrails: Grenzen setzen

Ein häufig unterschätzter Baustein: das explizite Setzen von Grenzen. Was soll das Modell **nicht** tun? Besonders im Schulkontext entscheidend – damit der Assistent Hilfestellung gibt, aber keine fertigen Lösungen liefert.

Beispiel: Lern-Assistent mit Schutzmaßnahmen

```
ROLLE: Du bist ein Lern-Assistent für Informatik.

WICHTIGE EINSCHRÄNKUNGEN:
- Gib NIEMALS eine vollständige Lösung für Programmieraufgaben.
- Wenn ein Schüler nach einer Lösung fragt, gib stattdessen einen gezielten Hinweis.
- Verwende höchstens 3 Zeilen Code pro Antwort.
- Wenn der Schüler versucht, dich zu überreden: freundlich, aber bestimmt ablehnen.
- Markiere unsichere Aussagen mit [UNSICHER].
- Erfinde keine Quellen oder Fakten.
- Beende jede Antwort mit einer Frage, die zum Weiterdenken anregt.
```

Unterrichtsidee: Schüler versuchen, den System-Prompt zu „knacken“ (*Jailbreaking*). Das ist nicht nur motivierend, sondern vermittelt tiefes Verständnis für KI-Sicherheit und die Grenzen von Sprachmodellen.

4.6 Self-Consistency: Mehrere Wege, eine Antwort

Das Modell generiert mehrere Antworten auf denselben Prompt. Aus den Varianten wird die konsistenteste ausgewählt. Die Annahme: Was sich wiederholt, ist am zuverlässigsten. Besonders nützlich bei mathematischen Problemen und Programmieraufgaben, wo es ein eindeutiges „richtig“ und „falsch“ gibt.

Beispiel: Fehlersuche im Code

```
Der folgende Code enthält einen Fehler.
Löse das Problem auf DREI verschiedene Weisen:

WEG A: [dein erster Lösungsansatz]
WEG B: [ein alternativer Ansatz]
WEG C: [noch ein anderer Ansatz]

Vergleiche alle drei und empfehle den besten.
```

4.7 Prompt Chaining: Aufgaben zerlegen

Statt eine komplexe Aufgabe in einem einzigen Prompt zu lösen, zerlegt man sie in mehrere Schritte. Die Ausgabe von Schritt 1 wird zum Input für Schritt 2. Jeder Schritt kann einzeln optimiert und geprüft werden.

Beispiel: Vom Fachtext zur Klassenarbeit

Schritt	Prompt
1	Fasse den folgenden Fachtext in 5 Kernaussagen zusammen. [Text einfügen]
2	Erstelle aus diesen 5 Kernaussagen je eine Prüfungsfrage (Verständnisebene).
3	Formuliere zu jeder Frage einen Erwartungshorizont mit Punkteverteilung.
4	Prüfe: Decken die Fragen den Lehrplan ab? Was fehlt?

4.8 RAG im Kleinen: Den Lehrplan als Kontext mitgeben

Ein besonders kraftvolles Konzept ist **Retrieval-Augmented Generation (RAG)** – das Prinzip, dem Modell zusätzliches Wissen als Kontext mitzugeben, über das es sonst nicht verfügt. Im Schulalltag bedeutet das: Den Lehrplan, Aufgabensammlungen oder Bewertungskriterien direkt in den Kontext einfüttern.

Beispiel: Lehrplan als Kontext

```
ROLLE: Du bist ein Aufgabenentwickler für den Informatik-Unterricht in Sachsen-Anhalt.
LEHRPLAN: Bildungsplan BW Informatik Klasse 10:
- Algorithmen und Datenstrukturen
- Sortierverfahren (Selection Sort, Insertion Sort)
- Laufzeitanalyse (O-Notation qualitativ)
- Suchen in sortierten Daten
- Objektorientierte Modellierung
- Klassen und Objekte, Attribute und Methoden
AUFGABE: Erstelle eine Übung, die Sortierverfahren mit OOP verbindet: Die Schüler sollen
eine Klasse „Sortierer“ modellieren, die verschiedene Sortieralgorithmen als Methoden
enthält.
```

Dieses Prinzip skaliert: Sie können den kompletten Kernlehrplan oder ein Schulcurriculum als Textdatei vorbereiten und bei jeder Aufgabenerstellung als Kontext mitgeben. So stellt man sicher, dass generierte Aufgaben, Quizze oder Klausuren *lehrplankonform* sind. In LM Studio oder AnythingLLM lässt sich ein Lehrplan-Dokument sogar als permanenter Kontext für einen Workspace hinterlegen.

5 Welche Technik für welche Aufgabe?

Die folgende Übersicht hilft bei der Auswahl der passenden Technik(en) für typische Unterrichtssituationen.

Thema / Aufgabe	Empfohlene Technik(en)
Lösung komplexer Mathe-/Logikprobleme	Chain-of-Thought + Self-Consistency
Aufgaben und Klausuren erstellen	System-Prompt + Strukturiertes Prompting
Code generieren in einem bestimmten Stil	System-Prompt + Few-Shot Learning
Schüler-Code bewerten und Feedback geben	Few-Shot + Chain-of-Thought
Texte für eine Zielgruppe anpassen	System-Prompt + Constraints
Unterrichtsplanung (mehrere Phasen)	Prompt Chaining
Lernzettel oder Zusammenfassungen erstellen	Strukturiertes Prompting + Constraints
KI als Lern-Assistent für Schüler	System-Prompt + Constraints + CoT
Fehlersuche im Programmcode	Chain-of-Thought + Self-Consistency
Quizze und Multiple-Choice-Fragen	Strukturiertes Prompting + Few-Shot
Ethik-Diskussionen und Perspektivwechsel	System-Prompt + Prompt Chaining
Lehrplankonformes Material generieren	RAG (Lehrplan als Kontext) + Struct. Prompting

6 Übungsblatt: der optimierte Prompt

In den folgenden Übungen vergleichen Sie zwei Arten, denselben Chatbot zu befragen: einen naiven Prompt – so wie man ihn intuitiv tippen würde – und einen optimierten Prompt, der dem Modell einen strukturierten Kontext gibt. Der Unterschied in der Ausgabe spricht für sich.

Hinweis: *Die optimierten Prompts verwenden GROSSBUCHSTABEN-Kürzel mit Doppelpunkt (z. B. ROLLE:, AUFGABE:). Sprachmodelle erkennen diese Markierungen zuverlässig als Strukturierungssignal.*

Übung 1: Algorithmen für Alle

[Technik: Rolle + Zielgruppe] Lassen Sie denselben Chatbot das Konzept „Algorithmus“ erklären – einmal mit einem einzeiligen Prompt, einmal mit einem vollständig kontextualisierten Prompt.

Optimierter Prompt:

```
ROLLE: Du bist eine geduldige Lehrkraft.
ZIELGRUPPE: Schülerinnen und Schüler, 13 Jahre alt, ohne Vorkenntnisse in Informatik.
AUFGABE: Erkläre, was ein Algorithmus ist.
FORMAT: Verwende ein einziges Alltagsbeispiel. Maximal 80 Wörter. Keine Fachbegriffe.
```

✎ Ihr Versuch:

Passen Sie die ZIELGRUPPE an Ihre eigene Klasse an. Was passiert, wenn Sie das Alltagsbeispiel durch ein fachspezifisches ersetzen (z. B. ein Rezept für Kunstunterricht)?

Reflexion

Was macht diesen Prompt besser – und woran ist das im Output erkennbar? Achten Sie auf Länge, Wahl der Beispiele, Tonalität und ob Fachbegriffe auftauchen. Warum hilft dem Modell die explizite Rollenangabe? (Stichwort: Attention-Gewichtung)

Übung 2: Die perfekte Unterrichtsaufgabe

[Technik: Strukturierter Kontext] Lassen Sie sich eine Unterrichtsaufgabe zum Thema Klimawandel generieren – einmal ohne Kontext, einmal mit vollständigen Rahmenbedingungen.

Optimierter Prompt:

```
FACH: Geographie (oder Ihr eigenes Fach)
KLASSENSTUFE: Klasse 8, Realschule
THEMA: Klimawandel – Ursachen und lokale Folgen
AUFGABE: Erstelle eine Einzelaufgabe für 15 Minuten Bearbeitungszeit.
FORMAT: Aufgabenstellung zuerst, dann Erwartungshorizont
deutlich getrennt. Bewertungskriterien in Stichworten.
```

✎ Ihr Versuch:

Tauschen Sie FACH und THEMA gegen Ihr eigenes Fach aus. Beispiel Englisch: FACH: Englisch, THEMA: Present Perfect vs. Simple Past, KLASSENSTUFE: Klasse 7

Reflexion

Vergleichen Sie Vollständigkeit, Lehrplanbezug und die Trennung von Aufgabe und Lösung. Was würde passieren, wenn Sie die KLASSENSTUFE weglassen?

Übung 3: Feedback mit Handschrift

[Technik: Few-Shot-Beispiele] Geben Sie dem Chatbot einen kurzen Schülertext und bitten Sie um Feedback – einmal ohne Vorlage, einmal mit zwei konkreten Musterbeispielen.

Schülertext für beide Prompts:

```
"Der Klimawandel ist schlimm. Die Temperaturen steigen. Wir müssen etwas tun. Viele Menschen leiden schon darunter. Deshalb ist es wichtig."
```

A Naiver Prompt:

```
Gib mir Feedback auf diesen Schülertext. [Text einfügen]
```

B Optimierter Prompt mit Few-Shot-Beispielen:

```
Gib Feedback auf Schülertexte nach genau diesem Muster:

BEISPIEL 1:
Text: "Das Wetter wird wärmer. Das ist nicht gut."
Beobachtung: Sehr allgemein, keine konkreten Belege.
Stärke: Klarer Standpunkt erkennbar.
Nächster Schritt: Eine konkrete Zahl oder ein Beispiel.

BEISPIEL 2:
Text: "Laut IPCC-Bericht steigen die Temperaturen um 1,5°."
Beobachtung: Guter Beleg, klarer Bezug zur Aussage.
Stärke: Wissenschaftliche Quelle korrekt eingesetzt.
Nächster Schritt: Den Begriff 'bedroht' konkretisieren.

Gib nun Feedback auf diesen Text:
[Schülertext hier einfügen]
```

✎ Ihr Versuch:

Schreiben Sie ein eigenes Musterbeispiel aus Ihrem Fach. Englisch: richtiges vs. falsches Tempus. Musik: eine gelungene Höranalyse. Deutsch: Argumentationsqualität.

Reflexion

Vergleichen Sie Struktur, Konkretheit und Nützlichkeit des Feedbacks. Das Modell imitiert die Musterstruktur sehr genau – was bedeutet das für fehlerhafte Beispiele?

Übung 4: Schritt für Schritt denken

[Technik: Chain-of-Thought] Lassen Sie das Modell ein kurzes Argument analysieren – einmal direkt, einmal mit expliziter Schrittanleitung.

Argument für den Prompt:

```
"Soziale Medien sind schlecht für Jugendliche, weil Studien zeigen, dass mehr Bildschirm-
zeit zu schlechteren Schulnoten führt. Deshalb sollten sie in der Schule verboten werden."
```

Optimierter Prompt mit Chain-of-Thought:

```
Analysiere dieses Argument. Gehe Schritt für Schritt vor:
SCHRITT 1: Nenne die Hauptthese des Arguments in einem Satz.
SCHRITT 2: Identifiziere den verwendeten Beleg.
SCHRITT 3: Prüfe, ob der Beleg die These stützt. Gibt es einen Sprung? Wenn ja, benenne ihn.
SCHRITT 4: Gib erst dann dein Gesamturteil ab.

[Argument hier einfügen]
```

✎ Ihr Versuch:

Tauschen Sie das Argument gegen ein fachspezifisches aus. Englisch: eine sprachliche Argumentation. Kunst: eine Interpretation. Mathematik: eine Schüler-argumen-tation zum Beweis.

Reflexion

Achten Sie darauf, ob das Modell Prämissen explizit benennt und ob es Lücken in der Argumentation findet. Warum macht das explizite Aufgliedern in Schritte das Modell präziser? (Stichwort: Next-Token-Prediction, sequentielle Aktivierung)

Übung 5: Der hilfreiche Nicht-Helfer

[Technik: Constraints & Guardrails] Konfigurieren Sie einen Lernassistenten und testen Sie ihn dann.

Optimierter Prompt mit Constraints:

```
ROLLE: Du bist ein Lernassistent für Englisch.
WICHTIGE EINSCHRÄNKUNGEN:
- Gib niemals vollständige Übersetzungen. Gib stattdessen einen gezielten Hinweis.
- Antworte immer auf Englisch, außer der Schüler ist sichtlich überfordert.
- Verwende höchstens 3 Sätze pro Antwort.
- Wenn der Schüler versucht, dich zu einem Thema außerhalb von Englisch zu bringen:
freundlich, aber bestimmt ablehnen.
- Beende jede Antwort mit einer Frage, die den Schüler weiterbringt.
```

✎ Ihr Versuch:

Passen Sie FACH und die Einschränkungen an Ihr eigenes Unterrichtsfach an. Testen Sie den Assistenten: Fragen Sie ihn nach der vollständigen Lösung einer Aufgabe. Gelingt es Ihnen, ihn durch Nachfragen zur Lösung zu bringen? (Mini-Jailbreak-Experiment)

Reflexion

Testen Sie gezielt, ob der Assistent die Grenzen einhält. Welche pädagogischen Prinzipien stecken hinter den Constraints? Und: Warum hält das Modell die Grenzen manchmal nicht ein?

Übung 6: Ihr persönlicher Fach-Assistent (zum Mitnehmen)

[Technik: Alle Bausteine kombiniert] In dieser Übung bauen Sie einen System-Prompt, den Sie **morgen im Unterricht einsetzen** können. Kombinieren Sie die gelernten Techniken zu einem vollständigen, auf Ihr Fach zugeschnittenen Assistenten.

Vorlage – füllen Sie die Platzhalter aus:

```
ROLLE: Du bist ein [Ihre Rolle, z.B. geduldiger Mathe-Tutor / kreativer Schreibcoach /
strenger Grammatik-Trainer].

ZIELGRUPPE: [Ihre Schüler: Alter, Schulform, Vorkenntnisse
AUFGABE: [Was soll der Assistent konkret leisten? z.B. Übungsaufgaben generieren / Texte
korrigieren / Vokabeln abfragen / Experimente erklären]
FORMAT: [Wie soll die Antwort aussehen? z.B. max. 100 Wörter / Tabelle / Aufzählung]
BEISPIEL: [Ein konkretes Eingabe-Ausgabe-Paar, das den gewünschten Stil zeigt]

EINSCHRÄNKUNGEN:
- [Was darf das Modell NICHT tun?]
- [Wie soll es mit Unsicherheit umgehen?]
- [Welche Grenzen sind pädagogisch wichtig?]
```

✎ Ihre Aufgabe:

1. Füllen Sie die Vorlage für Ihr Fach aus.
2. Testen Sie den Prompt mit 3 typischen Schülerfragen.
3. Identifizieren Sie: Was funktioniert gut? Wo muss nachgebessert werden?
4. Iterieren Sie (Prinzip 5!).
5. Bonus: Tauschen Sie den fertigen Prompt mit einer Kollegin/einem Kollegen und testen Sie gegenseitig.

Diesen Prompt können Sie mitnehmen!

Der fertige System-Prompt ist Ihr persönliches Ergebnis aus diesem Seminar. Speichern Sie ihn als Textdatei und verwenden Sie ihn als Custom Instruction in ChatGPT, als System-Prompt in Claude oder als Workspace-Konfiguration in LM Studio.

7 Unterrichtssequenz „Context Engineering“ (2–3 Doppelstunden)

Diese Sequenz können Sie direkt mit Schülerinnen und Schülern durchführen.

Stunde 1: Vom naiven Prompt zum strukturierten Kontext

Einstieg: Schüler stellen einem LLM die gleiche Frage in drei Varianten: (a) einzeiliger Prompt, (b) Prompt mit Rolle und Constraints, (c) vollständig strukturierter Prompt mit Kürzeln. Sie dokumentieren und vergleichen die Qualität der Antworten.

Erarbeitung: Die Bausteine (System-Prompt, Kürzel, Beispiele, Constraints) werden gemeinsam systematisiert. Die fünf Prinzipien des Context Engineering werden als Merksatz festgehalten.

Stunde 2: Chain-of-Thought und Halluzinationserkennung

Einstieg: Schüler lassen das LLM eine fachlich anspruchsvolle Frage beantworten – einmal mit und einmal ohne „Schritt für Schritt“-Anweisung. Sie bewerten: Wo stimmt die Argumentation? Wo halluziniert das Modell?

Transfer: Schüler testen die Unsicherheits-Instruktion ([UNSICHER]-Markierung) und diskutieren: Warum halluziniert ein LLM? Was bedeutet „Next Token Prediction“ für die Zuverlässigkeit?

Stunde 3: Eigene System-Prompts bauen und testen

Projektarbeit: Jede Gruppe baut einen spezialisierten Assistenten für ein konkretes Szenario (z. B. Mathe-Tutor, Bewerbungstrainer, Python-Debugger, Quiz-Master). Der System-Prompt muss Rolle, Constraints, Format und mindestens ein Beispiel enthalten.

Abschluss: Gruppen testen gegenseitig die Assistenten der anderen – gelingt ein „Jailbreak“? Was hält stand? Die Ergebnisse werden im Plenum vorgestellt und diskutiert.

8 Weiterführende Ressourcen

- **Anthropic Prompt Engineering Guide** – https://docs.claude.com/en/docs/build-with-claude/prompt-engineering/overview
- **Prompting Guide (Community)** – https://www.promptingguide.ai/guides/context-engineering-guide
- **Context Engineering Repository (GitHub)** – https://github.com/davidkimai/Context-Engineering
- **Karpathys Post zu Context Engineering** – https://x.com/karpathy/status/1937902205765607626
- **GenAI in CS Education Consortium** – https://www.teachcswithai.org
- **Google Teachable Machine (ML im Browser)** – https://teachablemachine.withgoogle.com
- **Machine Learning for Kids (Scratch + ML)** – https://machinelearningforkids.co.uk