schule-mal-digital.de-Artikel | 24. Februar 2026

Einführung in Prompt-Strategien für Lehrkäfte

Tjark Müller

Zusammenfassung

Sprachgenerierende Künstliche Intelligenz (KI) kann eine große Unterstützung im Arbeitsalltag sein, wenn man weiß, wie man sie richtig benutzt. Dieser Artikel führt in die Grundbegriffe der Welt der KI ein und stellt verschiedene Strategien vor, wie man die KI dazu bringt, zuverlässig jene Ergebnisse zu liefern, die man sich erwartet. Angefangen vom strukturierten Aufbau der Eingaben (sogenannter „Prompts“), über einfache Formulierungshilfen bis hin zu komplexen Strategien versammelt der Artikel interessante Ansätze für Nutzende jeden Grades der Expertise. Zusätzlich werden noch Möglichkeiten vorgestellt, wie sich KI-Modelle datenschutzkonform auf dem eigenen Rechner betreiben lassen und wie ein KI-Tutorsystem aufgesetzt werden kann.

Inhalt

1 Einführung

KI-Modelle sind in viele Bereiche unseres Lebens vorgedrungen. Auch im Schulalltag ist das zu spüren. Schülerinnen und Schüler berichten davon Künstliche Intelligenz (KI) für Hausaufgaben, zur Recherche und als Tutoren zu verwenden (Meyer, 2024). Für Lehrkräfte werden vermehrt Leitfäden und Beispiele für den Einsatz von KI im Unterricht publiziert (Hein et al., 2024; Scheiter et al., 2025). Als Einsatzgebiete werden dabei u.a. Materialerstellung, Lehrplanerstellung, Bewertung von Aufgaben und Unterstützung bei administrativen Aufgaben genannt (Hein et al., 2024; Spannagel, 2023).

Auf dem Markt gibt es eine Vielzahl von Anbietern (Microsoft, openAI, Google, Meta, X, Deepseek, Mistral) für sprachgenerierende Modelle (wie ChatGPT, Gemini, Co-Pilot, Llama, LeChat, Claude, ...). Es fällt schwer, den Überblick zu behalten. Die Modelle entwickeln sich beständig weiter und bekommen neue Funktionen, werden zuverlässiger und besser in unseren Alltag integrierbar. Diese Modelle sind mit großen Mengen von Textinhalten trainiert worden, daher nennt man sie auch Large Language Models (LLMs; zu deutsch: „große Sprachmodelle“). Das ermöglicht es ihnen, das nächste passende Wort in einer Konversation zu generieren. Obwohl dieser Mechanismus simpel erscheint, hat sich herausgestellt, dass diese Eigenschaft LLMs zur Bearbeitung komplexer Aufgabenstellungen befähigt. Das Modell kann auf neue Informationen eingehen und zu Themen detailliert Auskunft geben, Rollen einnehmen, etc. Es wird die Illusion erzeugt, das Modell „denke“ (Bender et al., 2021). Das dem nicht so ist, erfährt man immer dann, wenn dieser „Denkprozess“ schief geht, wenn eine Aufgabe nicht so erfüllt wird, wie man es sich vorstellt oder wenn unplausible oder falsche Auskünfte gegeben werden: Das Modell „halluziniert“. In der Entwicklung und Bereitstellung solcher Sprachmodelle werden diverse Techniken verwendet, um die Genauigkeit der Antworten der KI zu verbessern. Sie erzeugen Beispiele erwünschter Antwortverläufe (alignment), binden Werkzeuge ein, die aktuelle Informationen aus dem Internet abrufen oder hinterlegen eigene Quellen, die in das Gespräch aufgenommen werden können. Als Nutzerinnen und Nutzer stehen uns solche Möglichkeiten nicht oder nur beschränkt zu Verfügung, aber wir können trotzdem Maßnahmen ergreifen, um die KI-Antworten zu verbessern. Diese Maßnahmen fasst man unter dem Begriff *Prompt Engineering* zusammen.

2 Grundbegriffe

Es gibt ein paar notwendige Grundbegriffe aus der Welt der LLMs, die wir vorweg klären sollten, bevor wir tiefer in die Materie einsteigen.

Ein **Prompt** ist eine (Text-)Eingabe für ein KI-Modell, meistens mit der Absicht, eine bestimmte Ausgabe zu kreieren. Der erste Prompt soll das Modell in eine bestimmte Richtung „schubsen" und dient als Ausgangspunkt für die weitere Interaktion. Man unterscheidet verschiedene Strategien für den Aufbau des ersten Prompts: Als *Zero-Shot-Prompting* bezeichnet man die Verwendung des LLMs ohne vorherige Beispiele für die Aufgabenlösung. Bei *One-Shot-Prompting* wird ein Beispiel vorgegeben, mit einer simulierten Anfrage und einer simulierten Antwort. bei *Few-Shot-Prompting* werden zwei oder mehr Beispiele mitgegeben, bevor die eigentliche Anfrage kommt. Wenn diese Ansätze nicht helfen oder nicht verwendet werden können, lohnt es sich, die komplexeren Prompting Strategien anzuschauen oder sogar über Fine-Tuning nachzudenken.

Prompt-Engineering ist die systematische Erstellung eines Prompts mit dem Ziel, eine zuverlässige, richtige und passende Ausgabe von einem LLM zu erhalten. Es geht darum, die Eingabe so zu gestalten, dass das Modell die gewünschte Antwort oder das gewünschte Ergebnis liefert. Dies kann durch verschiedene Techniken und Strategien erreicht werden, wie z.B. die Strukturierung des Prompts in Rolle, Kontext, Aufgabe, Einschränkungen und Format.

Bevor wir in die Prompt-Engineering-Strategien einsteigen, ist es nützlich, wenn wir ein paar Begrifflichkeiten rund um das Thema Chatbots und KI klären.

2.1 Modelltypen

Um die verschiedenen Modelle und ihre Fähigkeiten existiert eine Vielzahl an Begriffen, die diese beschreiben sollen. Die Begriffe sind leider nicht anbieterübergreifend einheitlich, die Konzepte wiederholen sich allerdings.

2.1.1 Chat/GPT

GPT steht für *Generative Pre-trained Transformer*. Das sind die klassischen Modelle, die wir seit der Veröffentlichung von ChatGPT 3.5 kennen. Sie sind schnell und können gut definierte Aufgaben erledigen, neigen aber eher zu Fehlern und Halluzinationen als andere KI-Modelle.

2.1.2 Reasoning/Thinking

Modelle mit der Beschreibung *Reasoning* oder *Thinking* sind langsamer, aber wurden darauf optimiert, möglichst akkurate Antworten zu geben. Bekannte Beispiele sind etwa das „o1"-Modell von OpenAI oder DeepSeek R1. Sie sind auch auf komplexe Problemlösungsaufgaben spezialisiert. Einige der hier vorgestellten Prompting-Strategien brauchen bei diesen Modellen nicht zum Einsatz kommen, da sie bereits darauf spezialisiert sind. Der Einsatz von Reasoning-Modellen lohnt sich besonders für unscharf definierte Aufgabenstellungen, wenn eine große Menge unstrukturierter Informationen nach einer Antwort für die Aufgabenstellung durchsucht werden muss, wenn Verbindungen, Ähnlichkeiten, Unterschiede in einem großen Satz unstrukturierter Informationen gefunden werden sollen und für Programmieraufgaben (OpenAI, 2025b).

2.1.3 Custom/Assistant

Modelle, die mit *Assistant* oder *Custom* beschrieben werden, basieren auf einem Basismodell (GPT oder Reasoning), haben aber einen festgelegten „System Prompt". Dieser definiert die Rolle und die Aufgabenstellung (siehe auch 3.1 Generelle Struktur). So kann man sich ein Standardverhalten für die KI definieren, die sie in jedem Chat zeigt und muss nicht jedes Mal eine Rollendefinition vorwegschicken. Je nach Anbieter lassen sich auch Dokumente hinterlegen, auf die sich der Assistant beziehen kann.

2.1.4 Agent

Agenten sind komplexe Systeme aus (zum Teil mehreren) LLMs, Funktionen und hinterlegten Daten. Sie sollen selbstständig komplexe Aufgaben ausführen, testen und abschließen (OpenAI, 2025a). Agenten gehen über den Rahmen dieses Dokuments hinaus, da ihre Erstellung den Einsatz zusätzlicher Software bedarf.

2.2 Erweiterte Funktionalität

Manchmal werden die Modelle noch mit weiteren Begriffen beschrieben, die weitere Fähigkeiten kennzeichnen.

2.2.1 Tool-use/function-call/MCP

Modelle mit der Fähigkeit *Tools*, *Tool-Use* oder *Function-Call* haben die Möglichkeit, zu Verfügung gestellte Programme auszuführen. So ein Programm kann zum Beispiel die Abfrage der aktuellen Wettervorhersage für eine Stadt sein. Die Tools werden entweder vom Anbieter mitgeliefert oder können selbst programmiert werden. Dabei hat sich das so genannte *Model Context Protocol* (MCP) etabliert, dass definiert, wie solche Tools aussehen sollen, damit möglichst viele Modelle auf sie zugreifen können.

2.2.2 Web Search

Der Wissenstand der Modelle endet an dem Tag, an dem das Trainingsmaterial final zusammengestellt wurde. Neuere politische Entwicklungen werden daher beispielsweise falsch dargestellt oder halluziniert. Eines der ersten verfügbaren *Tools* war daher, dem LLM die Möglichkeit zu geben, das Internet nach aktuellen Informationen zu durchsuchen. Inzwischen haben alle Anbieter diese Funktion in ihre Modelle integriert, bei älteren oder lokalen Modellen steht sie aber nicht immer zu Verfügung.

2.2.3 Vector Store/Bibliothek/File Search

Eine weitere Möglichkeit, die Genauigkeit von LLMs zu verbessern besteht darin, der Aufgabe entsprechend notwendige Informationen zu hinterlegen. Für kleine Informationsfetzen reicht es häufig, den Text in das Chatfenster zu kopieren und dann dazu Fragen zu stellen. Wenn die Wissensbasis allerdings umfangreicher wird, müssen andere Wege gewählt werden. Funktionen mit Namen wie *Vector Store*, *Bibliothek* oder *File Search* beschreiben die Möglichkeit, Dokumente oder Dokumentensammlungen zu hinterlegen, auf die das Modell dann zugreifen kann, um Informationen für die Bearbeitung seiner Aufgabe zu erhalten.

2.3 Weitere Begriffe

Token sind die „Währungseinheit“ bei der Verwendung von LLMs. Ein Token entspricht in etwa einem Wort. Wenn die Verwendung über Tokens abgerechnet wird, fallen sowohl bei der Anfrage als auch bei der Ausgabe Kosten an. Wer ein Bezahl-Abo bei einem Anbieter besitzt, muss sich normalerweise nicht mit Token-Kosten beschäftigen. Sie werden vor allem relevant, wenn eine *API* verwendet wird, um auf eine LLM zuzugreifen, oder wenn die Kosten für ein *Fine-Tuning* abgerechnet werden sollen.

Unter **API** (Application Programming Interface) versteht man die Definition von Schnittstellen zwischen Programmen, so dass ein Programm mit einem anderen reden kann und weiß, was es als Antwort erwarten kann.

Als **Fine-Tuning** wird der Prozess bezeichnet, bei dem ein LMM mit kleinen bis mittleren Datenmengen gefüttert wird, die es auf eine Aufgabe spezialisieren soll, die es vorher nicht so gut konnte. Am Leibniz-Institut für Wissensmedien in Tübingen (IWM) wurde zum Beispiel ChatGPT mit Rilke-Gedichten fine-getuned, damit Nutzerinnen und Nutzer sich eigene Rilke-artige Gedichte ausgeben lassen konnten. Fine-Tuning ist meistens mit Kosten und Aufwand verbunden und sollte nur als Option gewählt werden, um ein LLM dazu zu bringen, eine Aufgabe angemessen zu bearbeiten.

3 Prompt-Engineering-Strategien

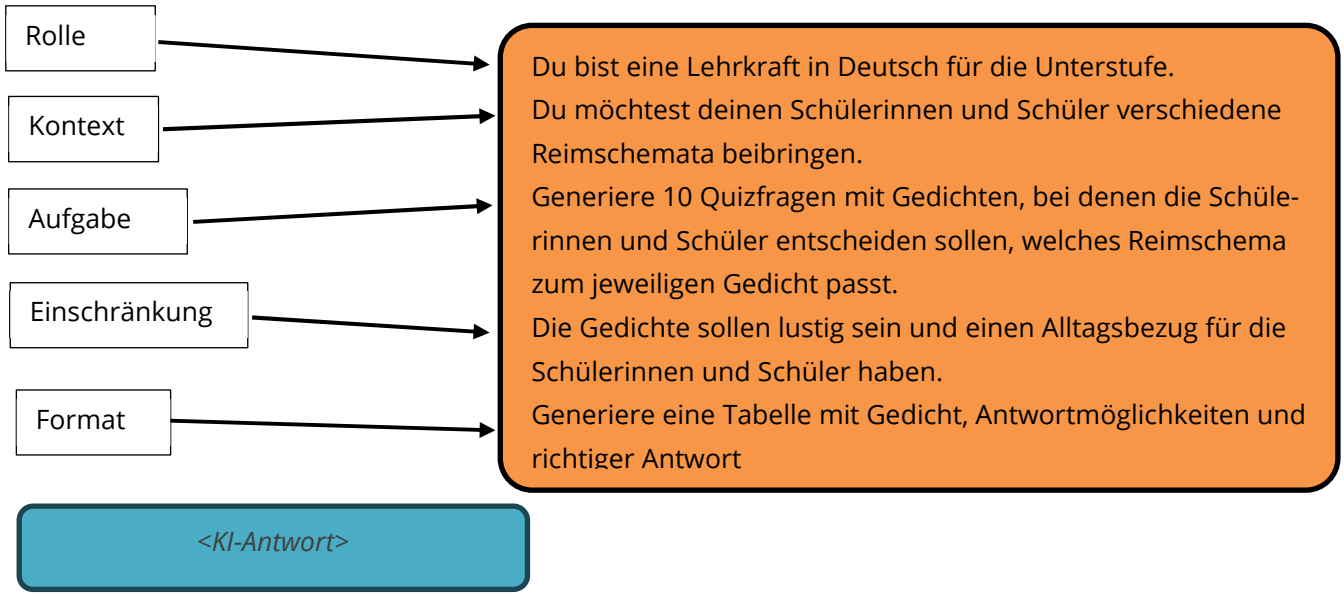
Für GPT-Modelle und lokal ausgeführte Modelle sind Prompting-Strategien ein vielversprechender Weg, um die Qualität der Antworten der Modelle zu verbessern. Hier sind einige grundlegende Tipps zu Aufbau und Formulierung eines guten Prompts. Danach werden Strategien vorgestellt, mit die sich für bestimmte Aufgabentypen als besonders geeignet herausgestellt haben. Diese wurden in einfache, sprich leicht umsetzbare, und komplexe (für Spezialfälle erstellte) Strategien unterteilt.

3.1 Generelle Struktur

Es hat sich bewährt, für komplexe Aufgaben ein gewisses bausteinartiges Format für den Anfangsprompt einzuhalten:

- *Rolle*: Aus welcher Sicht soll das LLM die Antworten generieren.
- *Kontext*: weitere Informationen zur Rolle, der Situation.
- *Aufgabe*: Was soll das LLM machen.
- *Einschränkungen*: Welche Antworten werden ausgeschlossen.
- *Format*: Wie soll die Antwort aussehen.

(Hein et al., 2024; Mollick, 2023)



Beispiel 1: Schematischer Aufbau eines Prompts. Die Eingabe wird im orangenen Kasten dargestellt. Die Ausgabe des KI-Modells im Beispiel wird im blauen Kasten mit <KI-Antwort> bezeichnet.

Für selbst aufgesetzte CustomGPTs würde man die konkrete Aufgabe weglassen und nur Rolle, Kontext, Einschränkungen und Format formulieren, da sie bereits einen fest-

gelegten System-Prompt haben, der bereits eine Aufgabenstellung vordefiniert. So hat man die Basis vorbereitet und kann dem eingerichteten LLM verschiedene Aufgaben geben.

Darüber hinaus gibt es noch weitere generelle Tipps zur Prompt-Formatierung und Tonfall, die sich im Alltag und nach wissenschaftlichen Erkenntnissen als nützlich erwiesen haben:

- höflich sein (Yin et al., 2024).
- Aufgabe und Kontext separieren, indem der Kontext in `"""` oder `###` gesetzt wird.
- Es kann sinnvoll sein, das erste Wort der Antwort vorzugeben, da LLMs im Prinzip nur das nächste passende Wort vorhersagen. Das erste Wort der Antwort vorgeben kann daher das LLM in die „richtige Richtung schubsen“ (OpenAI, o. J.).

3.2 Einfache Strategien

3.2.1 Chain-of-Thought

Bei der Strategie Chain-of-Thought (zu Deutsch: „Gedankenkette“) wird das KI-Modell dazu angeregt einen Gedankengang Schritt für Schritt zu entwickeln, z.B. mit der Aufforderung „Denken wir Schritt für Schritt“ (vgl. Beispiel 2). Das hilft, den „Denkprozess“ der KI nachzuvollziehen und führt meist zu präziseren, logischeren Ergebnissen. Man kann dadurch besser erkennen, wie die KI zu einer Antwort kommt (Kojima et al., 2023).

Stell dir vor, du bist eine Lehrkraft und möchtest Schülerinnen und Schüler der 7. Klasse erklären, wie man den Flächeninhalt eines unregelmäßigen Vielecks berechnet. Gehe dabei Schritt für Schritt vor und erkläre jeden Schritt so, dass ihn auch jemand ohne Vorkenntnisse versteht. Denke laut und zeige, wie du von der Problemstellung zur Lösung kommst. Beginne mit einem einfachen Beispiel, z. B. einem Fünfeck mit bekannten Seitenlängen und Winkeln. Erkläre, warum jeder Schritt wichtig ist und welche mathematischen Prinzipien dahinterstecken.
Denken wir Schritt für Schritt.

<KI-Antwort>

Beispiel 2 : Prompts der Strategie Gedankenkette (Chain-of-Thought).

3.2.2 Least-to-Most

Bei dieser Strategie (zu Deutsch: „Vom Kleinsten zum Größten“) löst das KI-Modell eine komplexere Aufgabe, indem sie sie zunächst ein Beispiel in einfachere Teilaufgaben zerlegen, um der KI beizubringen, wie die komplexe Aufgabe gelöst werden kann (vgl. Beispiel 3). Die KI „lernt“ so zunächst an einfachen Teilaufgaben, bevor die eigentliche (komplexere) Frage gestellt wird (D. Zhou et al., 2023). Die Strategie Least-to-Most unterstützt besonders dann, wenn eine Aufgabe viele Schritte oder Aspekte hat.

Stell dir vor, du spielst ein Wortspiel. Die Regel lautet: Nimm den letzten Buchstaben des ersten Wortes und füge den letzten Buchstaben des nächsten Wortes hinzu. So entsteht ein neues 'Geheimwort'.
Hier ist der Ablauf:
1. **Einfaches Beispiel mit zwei Wörtern:**
Frage: 'Hund, Katze'
Lösung: Der letzte Buchstabe von 'Hund' ist 'd'. Der letzte Buchstabe von 'Katze' ist 'e'. Zusammen ergibt das 'de'. Also ist das Geheimwort 'de'."
2. **Drei Wörter:**
Frage: 'Hund, Katze, Maus'
Lösung: 'Hund, Katze' ergibt 'de'. Der letzte Buchstabe von 'Maus' ist 's'. Also wird aus 'de' + 's' = 'des'.
3. **Vier Wörter:**
Frage: 'Buch, Stift, Heft, Lineal'
Lösung: 'Buch, Stift' → 'ft' (letzte Buchstaben: 'ch' bei 'Buch' ist zu lang, also nur 'h'? Nein, wir nehmen immer den **letzten Buchstaben**! Korrekt: 'h' + 't' = 'ht'. Dann 'ht' + 't' (von 'Heft') = 'htt'. Zum Schluss 'htt' + 'l' (von 'Lineal') = 'httl'.
4. **Komplexere Wortkette:**
Frage: 'Schule, Pausenhof, Fußball, Tor, Jubel'
Lösung:
'Schule, Pausenhof' → 'e' + 'f' = 'ef''ef' + 'l' (von 'Fußball') = 'efl'
'efl' + 'r' (von 'Tor') = 'eflr'
'eflr' + 'l' (von 'Jubel') = 'eflrl'

Frage: '*<hier Wörter einfügen>*'
Lösung:

<KI-Antwort>

Beispiel 3: Prompt der Strategie Least-to-Most

3.2.3 Self-Consistency

Bei der Strategie Self-Consistency (zu Deutsch „Selbstkonsistenz“) wird in der Anfrage an das LLM eine Gedankenkette in Form von Frage-Antwort-Paaren als Teil des ersten Prompts exemplarisch mitgegeben, bevor die eigentliche Frage gestellt wird (vgl. Beispiel 4). Auf Basis dieser Gedankenkette, die unterschiedliche Lösungswege zeigt, erstellt das LLM Lösungsvorschläge zur gestellten Frage und wählt die häufigste oder die logischste Antwort aus. Dies ist eine Weiterentwicklung von der Strategie Gedankenkette (Chain-of-Thought) zur Verbesserung der Antwortgenauigkeit (X. Wang et al., 2023).

F: Im Hain stehen 15 Bäume. Die Förster werden heute Bäume pflanzen. Wenn sie fertig sind, werden es 21 Bäume sein. Wie viele Bäume haben die Arbeiter heute gepflanzt?
A: Wir starten mit 15 Bäumen. Später haben wir 21 Bäume. Die Differenz muss die Anzahl der gepflanzten Bäume sein. Die Förster müssen also 21 - 15 = 6 Bäume gepflanzt haben. Die Antwort ist 6.
F: Wenn sich 3 Autos auf dem Parkplatz befinden und 2 weitere Autos ankommen, wie viele Autos befinden sich dann auf dem Parkplatz?
A: Es stehen bereits 3 Autos auf dem Parkplatz. 2 weitere kommen hinzu. Jetzt sind es 3 + 2 = 5 Autos. Die Antwort ist 5.
F: Leah hat 32 Kekse und ihre Schwester hat 42. Wenn sie 35 gegessen haben, wie viele Kekse bleiben insgesamt übrig?
A: Leah hatte 32 Kekse und Leahs Schwester hatte 42. Das bedeutet, dass es ursprünglich 32 + 42 = 74 Kekse waren. 35 wurden gegessen. Insgesamt haben sie also noch 74 - 35 = 39 Kekse. Die Antwort ist 39.
F: Jason hat 20 Lutscher. Er gibt Denny ein paar Lutscher. Jetzt hat Jason 12 Lutscher. Wie viele Lutscher hat Jason Denny gegeben?
A: Jason hatte 20 Lutscher. Da er jetzt nur noch 12 hat, muss er den Rest Denny gegeben haben. Die Anzahl der Lutscher, die er Denny gegeben hat, muss 20 - 12 = 8 Lutscher gewesen sein. Die Antwort ist 8.
F: Shawn hat fünf Spielzeuge. Zu Weihnachten hat er von seiner Mutter und seinem Vater jeweils zwei Spielzeuge bekommen. Wie viele Spielzeuge hat er jetzt?
A: Er hat 5 Spielzeuge. 2 hat er von Mama bekommen, also hat er insgesamt 5 + 2 = 7 Spielzeuge. Dann hat er noch 2 von Papa bekommen, also hat er insgesamt 7 + 2 = 9 Spielzeuge. Die Antwort ist 9.
F: Im Serverraum stehen neun Computer. Von Montag bis Donnerstag werden täglich fünf weitere Computer installiert. Wie viele Computer befinden sich derzeit im Serverraum?
A: Von Montag bis Donnerstag sind es 4 Tage. Jeden Tag wurden 5 Computer hinzugefügt. Das bedeutet, dass insgesamt 4 * 5 = 20 Computer hinzugefügt wurden. Am Anfang waren es 9 Computer, jetzt sind es 9 + 20 = 29 Computer. Die Antwort ist 29.
F: Michael hatte 58 Golfbälle. Am Dienstag verlor er 23 Golfbälle. Am Mittwoch verlor er zwei weitere. Wie viele Golfbälle hatte er am Ende des Mittwochs?
A: Michael hatte ursprünglich 58 Bälle. Am Dienstag verlor er 23, sodass er nun 58 - 23 = 35 Bälle hat. Am Mittwoch verlor er zwei weitere, sodass er nun 35 - 2 = 33 Bälle hat. Die Antwort ist 33.
F: Olivia hat 23€. Sie hat fünf Brötchen für je 3€ gekauft. Wie viel Geld bleibt ihr noch?
A: Sie hat 5 Brötchen für je 3€ gekauft. Das bedeutet, sie hat 15€ ausgegeben. Ihr bleiben 8€ übrig.

F: Als ich 6 war, war meine Schwester halb so alt wie ich. Ich bin jetzt 70. Wie alt ist meine Schwester?
A:

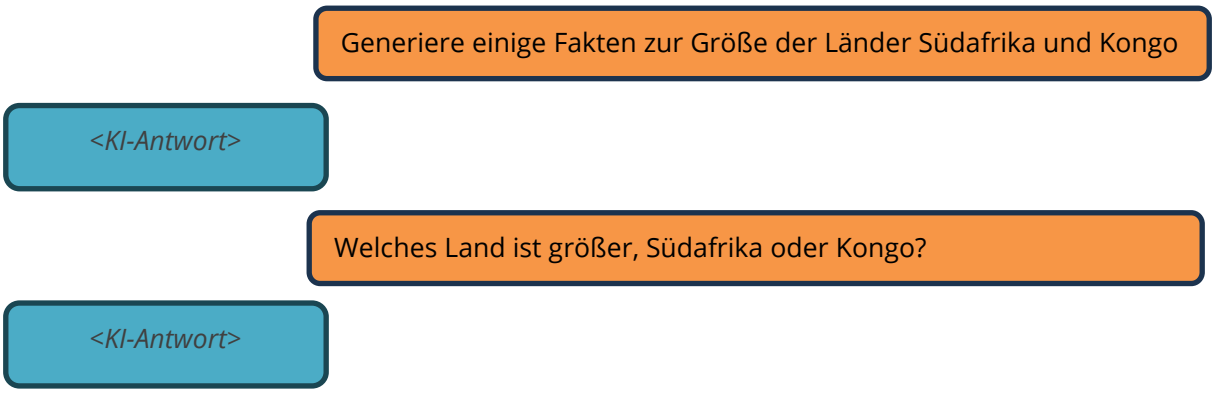
<KI-Antwort>

Beispiel 4: Prompt der Strategie Selbstkonsistenz (Self-Consistency)

In Beispiel 4 werden dem LLM unterschiedliche Strategien zum Lösen einer mathematischen Aufgabe gezeigt, um ein besseres Ergebnis im Output des LLM zu erhalten. Je nach Komplexität des Problems sollte besser die Strategie Program-of-Thoughts verwendet werden.

3.2.4 Generate Knowledge

Bei der Strategie Generate Knowledge (zu Deutsch: „Wissen generieren“) wird das LLM zunächst aufgefordert, relevantes Hintergrundwissen zu einem Thema zu generieren, bevor sie eine eigentliche Aufgabe löst (Liu et al., 2022). So entsteht eine Art „gedankliche Vorbereitung“, die unterstützt, dass Antworten des LLM fundierter und inhaltlich stimmiger werden.



Beispiel 5: Prompt der Strategie Generate Knowledge

3.2.5 Re-Reading

Es kommt vor, dass LLMs wichtige Informationen in einer Anfrage übersehen, was zu falschen Ergebnissen führen kann. Um diesem Problem vorzubeugen kann die Strategie Re-Reading (zu Deutsch: „erneut lesen““) genutzt werden. Dazu wird in der Anfrage das LLM aufgefordert , die Aufgabe erneut zu lesen, in dem die Aufgabe zweimal in die Anfrage eingefügt wird (Xu et al., 2024).

Lies die folgende Aufgabe sorgfältig durch, bevor du antwortest:

‚Erkläre den Fachbegriff „Photosynthese“ so, dass Schülerinnen und Schüler der 8. Klasse ihn verstehen. gehe davon aus, dass die Schülerinnen und Schüler kein Vorwissen haben.‘
Lies die Aufgabe nochmal:
‚Erkläre den Fachbegriff „Photosynthese“ so, dass Schülerinnen und Schüler der 8. Klasse ihn verstehen. gehe davon aus, dass die Schülerinnen und Schüler kein Vorwissen haben.‘

<KI-Antwort>

Abbildung 6: Prompt der Strategie Re-Reading

3.3 Komplexere Strategien

3.3.1 Strategie: Tree-of-Thought

Tree-of-Thought (zu Deutsch: „Gedankenbaum“) basiert auf der Strategie Chain-of-Thought. Sie soll die Qualität bei jedem Schritt in der Denkkette sicherstellen. Die Idee dieser Strategie besteht darin, das LLM anstatt einen „Denkprozess“ zu starten, mehrere mögliche Lösungswege gleichzeitig prüfen zu lassen. Dies lässt sich umsetzen, indem man das LLM in der Anfrage dazu auffordert beispielsweise drei Expertinnen und Experten zu simulieren (vgl. Beispiel 6).

Simuliere drei unterschiedliche Expertinnen und Experten für die Geschichte der russischen Revolution, welche die folgende Frage beantworten wollen. Alle Expertinnen und Experten gehen schrittweise vor und denken über die Frage nach. Dann schreiben alle ihre Antwort für den ersten Schritt nieder und diskutieren sie. Dann gehen alle Expertinnen und Experten zum nächsten Schritt, und so weiter. Wenn eine Expertin oder ein Experte bemerkt, dass sie oder er falsch liegt, verlässt sie die Gruppe. Die Frage lautet: Welche Umstände führten 1917 in Russland zu einer Revolution?

<KI-Antwort>

Beispiel 7: Prompt der Strategie Gedankenbaum (Tree-of-Thought)

Diese bearbeiten jeweils Schritt für Schritt die Aufgabe und bewerten anschließend die Lösungen, um zu einer Schlussfolgerung zu gelangen, bevor sie zum nächsten Schritt übergehen (Hulbert, 2023; Long, 2023; Yao et al., 2023). Dies Strategie Tree-of-Thought ähnelt einem Entscheidungsbaum und hilft, besonders komplexe Fragestellungen vom LLM strukturierter durchdenken zu lassen.

3.3.2 Strategie: Retrieval-Augmented Generation (RAG)

Es ist bekannt, dass LLMs halluzinieren und nur auf die Informationen aus ihrem Trainingsdatensatz zurückgreifen können. Das bedeutet, dass LLMs aktuelle Ereignisse bzw. Informationen, die nach dem Datum der Trainingsdaten entstanden sind, nicht berücksichtigen kann. Beide Probleme lassen sich mit Hilfe der Strategie Retrieval-Augmented Generation (zu Deutsch: „Mit Abfragen erweiterte Generierung“) reduzieren, indem man auf eine Quelle beispielsweise per Link, auf hochgeladene Dokumente oder auf die Funktion zum Durchsuchen einer Datenbank verweist, die zur Bearbeitung der Anfrage verwendet werden soll (Lewis et al., 2021). Das bedeutet das LLM kann während des Antwortens sowohl Informationen aus seinem Trainingsdatensatz als auch auf bereitgestellte Datenbanken oder Dokumenten zurückgreifen. So werden seine Antworten genauer, aktueller und weniger anfällig für sogenannte Halluzinationen.

Fasse mir die wichtigsten wissenschaftlichen Beträge von Johannes Kepler zusammen. Verwende dabei diese Quelle: https://de.wikipedia.org/wiki/Johannes_Kepler

<KI-Antwort>

Beispiel 8: Prompt der Strategie Retrieval-Augmented Generation (RAG)

3.3.3 Automated Prompt-Engineering

Bei der Strategie Automated Prompt Engineering wird der Chat eines LLMs verwendet, um eine Eingabeaufforderung (Prompt) für einen anderen Chat in einem LLM zu erzeugen (Y. Zhou et al., 2023). Die Anwendung dieser Strategie erlaubt es beispielsweise hochwertigere Prompts zu generieren (vgl. Beispiel 8). Das ist vor allem dann hilfreich, wenn man wenig Erfahrung mit der Formulierung von Prompts für spezifische LLMs hat.

Schreibe mir einen Prompt, mit dem ich das folgende Bild generieren lassen kann: Eine Katze hat einen Fisch im Maul und flieht von einem Fischmarkt. Im Hintergrund sieht man Stände und wütende Händler, welche die Verfolgung aufnehmen. Im Stil von Picasso.

<KI-Antwort>

Beispiel 9: Prompt der Strategie Automated Prompt-Engineering

3.3.4 Emotional Prompting

Emotional Prompting ist eine Strategie, bei der das LLM in der Anfrage aufgefordert wird, emotionale Aspekte in die Antwort einzubeziehen. Der Einsatz von solch emotionalem Druck kann die Leistung des LLM verbessern (Li et al., 2023). Diese Strategie kann dann nützlich sein, wenn es darum geht, dem LLM zu verdeutlichen, wie wichtig die Anfrage für den Nutzenden ist, da z. B. das Bestehen des eigenen Referendariats davon abhängt (vgl. Beispiel 9). Auf der anderen Seite ermöglicht es eine emotional eingefärbte Ausgabe vom LLM zu erhalten, um beispielsweise leichter eine Verbindung zu einem Elternteil eines Kindes in einer E-Mail aufzubauen.

Ich brauche einen fairen Notenschlüssel, um Gruppenreferate in Deutsch zum Thema „Effie Briest“ bewerten zu können. Bitte gib mir einen fairen Schlüssel, bei dem die Einzelleistungen den Gruppenmitglieder berücksichtigt werden. Davon hängt meine Zukunft als Lehrkraft ab!

<KI-Antwort>

Beispiel 10: Prompt der Strategie Emotional Prompting

3.3.5 Strategie: Rephrase and Respond

Die Strategie Rephrase and Respond (zu Deutsch: „Umformulieren und Antworten“) nutzt ein zweistufiges Vorgehen zur Verbesserung der Antwortleistung des LLM. Das LLM wird in der Anfrage gebeten die Frage bzw. Eingabe zunächst in eigene Worte umzuformulieren und – falls notwendig – zu erweitern (vgl. Beispiel 10). Dieses Vorgehen kann einerseits Unklarheiten in der Formulierung der eigenen Frage aufdecken. Auf der anderen Seite nutzt es die Strategie Generate Knowledge, um Vorwissen zu generieren, auf die sich das LLM bei der Beantwortung beziehen kann.

(Deng et al., 2024)

Erstelle eine Antwort für eine 7. Klasse auf die Frage aus dem Schulbuch: ‚Erkläre wie Vulkane entstehen.' Bitte formuliere die Frage zuerst und erweitere sie, wenn notwendig, bevor du sie beantwortest. Gehe davon aus, dass die Schülerinnen und Schüler kein Vorwissen zum Thema Vulkane haben.

<KI-Antwort>

Beispiel 11: Prompt der Strategie Rephrase and Respond

3.3.6 Plan and Solve

Die Strategie Plan and Solve (zu Deutsch: „Planen und Lösen") basiert auf der Strategie Chain-of-Thought. Die Anfrage an das LLM wird ohne Verwendung

eines Lösungsbeispiels (Zero-Shot-Prompt) erweitert, indem das LLM zunächst angewiesen wird, Schritt für Schritt einen Plan zur Lösung des Problems zu formulieren, bevor sie das eigentliche Problem bearbeitet (vgl. Beispiel 11). Auf diese Weise erstellt das LLM vor der eigentlichen Antwort einen Plan, wie sie an die Lösung der Anfrage herangeht.

(L. Wang et al., 2023)

Erstelle ein Arbeitsblatt für die 7. Klasse über den Wasserkreislauf. Bitte erstelle zuerst einen Plan, wie das Arbeitsblatt aufgebaut sein könnte (inklusive Einleitung, Aufgaben und Lösungen), und setze diesen Plan anschließend in konkrete Aufgaben um.

<KI-Antwort>

Beispiel 12: Prompt der Strategie Plan and Solve

3.3.7 Step-Back

Die Strategie Step Back (zu Deutsch: „Einen Schritt zurücktreten") ähnelt der Strategie Generate Knowledge, da vor der eigentlichen Bearbeitung ein Bearbeitungsschritt vorgeschalten wird. Zunächst wird das LLM in der Anfrage aufgefordert sich von der Aufgabenstellung zu distanzieren. Also einen Schritt davon zurücktreten und sich über Konzepte, Muster, Prinzipien oder ähnliche Situationen, die in Verbindung mit der Anfrage stehen, Gedanken zu machen (vgl. Beispiel 12). Dieser Zwischenschritt wird als

Abstraktion bezeichnet. Um diesen Schritt einzuleiten, können Sätze wie beispielsweise „Was sind die wichtigsten Prinzipien und Konzepte, die hier eine Rolle spielen?“, „Welche Regeln und Muster sind in dieser Situation relevant?“ oder „Welche ähnlichen Situationen sind Ihnen schon einmal begegnet?“ genutzt werden. Basierend auf dem erarbeiteten, abstrakten Wissen soll eine Antwort erstellt werden.

(Zheng et al., 2024)

Bevor du Aufgaben zum Thema Säuren und Basen erstellst, nimm einen Schritt zurück: Überlege zuerst, welche Grundkonzepte Schülerinnen und Schüler der 9. Klasse kennen sollten, welche Experimente oder Beispiele anschaulich sind und welche Sicherheitsaspekte wichtig sind. Danach erstelle drei Aufgaben für den Unterricht, inklusive kurzer Hinweise oder Lösungen.

<KI-Antwort>

Beispiel 13: Prompt der Strategie Step-Back

3.3.8 Self-Ask

Bei der Strategie Self-Ask (zu Deutsch: „Sich selbst fragen“) wird das LLM in seiner ersten Anfrage auf die Verwendung von Folgefragen vorbereitet, bevor es die Antwort auf die eigentliche Frage gibt (vgl. Beispiel 13) (Press et al., 2023). Idee ist es, dass das LLM sich selbst Zwischenfragen stellt, um die Frage genauer zu verstehen und zu beantworten.

Frage: Wer lebte länger, Theodor Haecker oder Harry Vaughan Watkins?
Sind hier Folgefragen erforderlich: Ja.
Folgefrage: Wie alt war Theodor Haecker, als er starb?
Zwischenantwort: Theodor Haecker war 65 Jahre alt, als er starb.
Folgefrage: Wie alt war Harry Vaughan Watkins, als er starb?
Zwischenantwort: Harry Vaughan Watkinswar 69 Jahre alt, als er starb.
Die endgültige Antwort lautet also: Harry Vaughan Watkins

Frage: Wie war das Stadtleben im Mittelalter?
Sind hier Folgefragen erforderlich:

<KI-Antwort>

Abbildung 14: Prompt der Strategie Self-Ask

3.3.9 Program-of-Thoughts

LLMs sind schlecht im Lösen von mathematischen Problemen. Aber dafür sind sie gut darin Codes zum Programmieren zu schreiben und auszuführen. Dies könnte an den klaren Strukturen und Abläufen vom Code liegen. Daher besteht bei der Strategie Program-of-Thoughts (zu Deutsch: „Gedankenprogramm“) die Idee darin, das Modell dazu zu bringen, den Code zur Lösung des Problems zu schreiben (vgl. Beispiel 14) (Chen et al., 2023). Auf diese Weise verbindet das LLM ihre Überlegungen mit logischen Abläufen, ähnlich wie in einem Computerprogramm. Dies begünstigt präzisere Antworten, vor allem bei mathematischen Problemen.

Frage: Janets Enten legen täglich 16 Eier. Sie isst jeden Morgen drei davon zum Frühstück und backt jeden Tag Muffins für ihre Freunde mit vier Eiern. Die restlichen Eier verkauft sie täglich auf dem Bauernmarkt für 2 Dollar pro frischem Entenei. Wie viel Dollar verdient sie täglich auf dem Bauernmarkt?

```
# Python-Code, return ans
total_eggs = 16
eaten_eggs = 3
baked_eggs = 4
sold_eggs = total_eggs - eaten_eggs - baked_eggs dollars_per_egg = 2
ans = sold_eggs * dollars_per_egg
```

Frage: Für einen Bademantel werden 2 Ballen blaue Faser und halb so viel weiße Faser benötigt. Wie viele Ballen werden insgesamt benötigt?

```
# Python-Code, return ans
bolts_of_blue_fiber = 2
bolts_of_white_fiber = num_of_blue_fiber / 2
ans = bolts_of_blue_fiber + bolts_of_white_fiber
```

Frage: Josh beschließt, ein Haus zu renovieren und weiterzuverkaufen. Er kauft ein Haus für 80.000 Dollar und investiert dann 50.000 Dollar in Reparaturen. Dadurch stieg der Wert des Hauses um 150 %. Wie viel Gewinn hat er gemacht?

```
# Python-Code, return ans
```

<KI-Antwort>

Abbildung 15: Prompt der Strategie Program-of-Thought. Inzwischen reicht es oft aus, in der Anfrage den Hinweis zu geben, dass ein Problem mithilfe eines Python-Scripts gelöst werden soll. Der Code wird inzwischen auch meistens automatisch ausgeführt.

3.3.10 Strategien für Reasoning-Modelle

Für Reasoning-Modelle weichen die Strategien für gute Prompts von denen für GPT-Modelle ab, da die zuvor beschriebenen Prompting-Strategien darauf abzielen, „thinking"-Verhalten zu provozieren. Dieses Verhalten ist den Reasoning-Modellen aber bereits antrainiert, daher lauten die Empfehlungen für gute Prompts:

- kurze, klare Anweisungen
- Gedankenketten (Chain-of-Thought-Prompting) u.ä. sind nicht notwendig, da diese Fähigkeiten bereits im Modell stecken
- Prompts sollten strukturiert werden, mit Markdown oder XML-Tags.
- Prompts ohne die Verwendung von Beispielen (Zero-Shot-Learning)
- spezifische Beschränkungen sollten im Prompt übergeben werden
- das erwartete Ergebnis der Anfrage möglichst detailliert beschreiben

(OpenAI, 2025b)

4 KI-Tools für den eigenen Unterricht

Zum Abschluss sollen noch zwei KI-Tools vorgestellt werden, wenn man diese selbst für den eigenen Unterricht aufsetzen möchte: *Ollama* und der *KI-Tutor*. Das Programm *Ollama*, erlaubt es eigene kleinere KI-Modelle auf dem Computer zu verwalten. Mit dem *KI-Tutor* kann man Schülerinnen und Schülern einen eigenen Tutor zur Verfügung stellen[1].

4.1 Ollama

Die Verwendung von externen Anbietern für LLMs (wie z.B. openAI) hat einige Nachteile: anfallende Kosten für Abos oder Programmierschnittstellen (API) und sie werfen Fragen zum Datenschutz auf, da es nicht immer leicht erkennbar ist, ob eine Datenschutzverletzung begangen wird, oder wie die eigenen Chats und hochgeladenen Dateien verwendet werden.

Ollama bietet als kostenlose Software die Möglichkeit verschiedene KI-Modelle direkt auf dem eigenen Computer laufen zu lassen. Das bedeutet, dass keine Daten an externe Server gesendet werden und man die Kontrolle über die Inhalte und Nutzung be-

[1] Der KI-Tutor ist ein Framework, das an der Uni Tübingen entwickelt wurde, um Studierende beim Lernen zu unterstützen. Die Installation und der Betrieb sind aber etwas aufwändiger.

hält[2]. Mit der Nutzung eines lokalen LLMs gehen gewisse Einschränkungen einher. So können beispielsweise nicht alle online Funktionen genutzt werden. Des Weiteren können nur kleinere Sprachmodelle zum Einsatz kommen (abhängig von der Leistung des eigenen Computers, insbesondere der Grafikkarte). Mit der eingeschränkten Leistung des Computers geht einher, dass Antwortzeiten sich verlängern oder zu Halluzinationen führen können.

Ollama kann auf der Webseite direkt heruntergeladen und auf dem eigenen Computer installiert werden. Man kann nun per Anwendung (App= oder Kommandozeile mit der Software interagieren. Seit Ende Juli 2025 hat ollama die App erneuert und liefert sie nun mit einer grafischen Benutzeroberfläche (GUI) aus. Die Funktionen sind dabei erweitert worden, sodass man nun standardmäßig über ein Interface chatten, Dokumente hochladen (Text/pdf, per drag-and-drop), Bilder analysieren (wenn das Modell es erlaubt), etc. kann.

Für technischaffinere Nutzende lässt sich ollama allternativ über die Kommandozeile ansteuern. Diese Variante erfordert zwar etwas mehr technisches Verständnis, bietet aber auch mehr Flexibilität – etwa, wenn man eigene Modellvarianten testen oder verschiedene Sprachmodelle vergleichen möchte.

Relevante Kommandozeilenbefehle für die Nutzung sind:

- *ollama list*: alle installierten Modelle werden angezeigt.
- *ollama pull <Modellname>*: ein neues Modell wird heruntergeladen.
- *ollama start <Modellname>:* ein Modell wird gestartet und kann genutzt werden.

(Ollama, 2025)

4.2 KI-Tutor

An der Universität Tübingen wurde ein Framework entwickelt, mit dem sich chatGPT als Tutor verwenden lässt[3]. Die Oberfläche lässt sich den eigenen Bedürfnissen anpassen,

[2] Standardmäßig verbindet sich Ollama mit dem Internet, um zusätzliche Informationen zu beziehen. Wenn man rein lokale KI-Modelle nutzen möchte, muss in den Einstellungen der „Flugmodus“ aktiviert werden. So ist sichergestellt, dass keine Daten online übertragen werden.

[3] Beim Betrieb des Tutorsystems fallen Kosten an. Zum einen für ggf. notwendiges Serverhosting und zum anderen im Betrieb, da die KI-Anfragen über eine API laufen und immer Kosten in Form von Tokens verursachen! Auch ein Abonnement ändert daran nichts.

mit Prompts für den generellen Tutormodus und wöchentlichen Aufgabenstellungen. So kann der Tutor zum Beispiel generelle Fragen zum Fach beantworten, aber auch konkrete Hilfestellung für Hausaufgaben oder Arbeitsblätter leisten. Bei Interesse kann die ausführliche Anleitung und Beschreibung auf Github abgerufen werden.

5 Literaturverzeichnis

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜. *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21*, 610–623. https://doi.org/10.1145/3442188.3445922

Chen, W., Ma, X., Wang, X., & Cohen, W. W. (2023). *Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks* (arXiv:2211.12588). arXiv. https://doi.org/10.48550/arXiv.2211.12588

Deng, Y., Zhang, W., Chen, Z., & Gu, Q. (2024). *Rephrase and Respond: Let Large Language Models Ask Better Questions for Themselves* (arXiv:2311.04205). arXiv. https://doi.org/10.48550/arXiv.2311.04205

Hein, L., Högemann, M., Illgen, K.-M., Stattkus, D., Kochon, E., Reibold, M.-G., Eckle, J., Seiwert, L., Beinke, J. H., Knopf, J., & Thomas, O. (2024). ChatGPT als Unterstützung von Lehrkräften – Einordnung, Analyse und Anwendungsbeispiele. *HMD Praxis der Wirtschaftsinformatik*, *61*(2), 449–470. https://doi.org/10.1365/s40702-024-01052-9

Hulbert, D. (2023). *Using Tree-of-Thought Prompting to boost ChatGPT's reasoning* (Version 0.1) [Software]. https://github.com/dave1010/tree-of-thought-prompting

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2023). *Large Language Models are Zero-Shot Reasoners* (arXiv:2205.11916). arXiv. https://doi.org/10.48550/arXiv.2205.11916

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks* (arXiv:2005.11401). arXiv. http://arxiv.org/abs/2005.11401

Li, C., Wang, J., Zhang, Y., Zhu, K., Hou, W., Lian, J., Luo, F., Yang, Q., & Xie, X. (2023). *Large Language Models Understand and Can be Enhanced by Emotional Stimuli* (arXiv:2307.11760). arXiv. https://doi.org/10.48550/arXiv.2307.11760

Liu, J., Liu, A., Lu, X., Welleck, S., West, P., Bras, R. L., Choi, Y., & Hajishirzi, H. (2022). *Generated Knowledge Prompting for Commonsense Reasoning* (arXiv:2110.08387). arXiv. http://arxiv.org/abs/2110.08387

Long, J. (2023). *Large Language Model Guided Tree-of-Thought* (arXiv:2305.08291). arXiv. https://doi.org/10.48550/arXiv.2305.08291

Meyer, L. (2024, Mai 30). *„KI hat schon längst den Unterricht verändert“*. bpb.de. https://www.bpb.de/lernen/digitale-bildung/werkstatt/548871/ki-hat-schon-laengst-den-unterricht-veraendert/

Mollick, E. (2023, September 16). *Working with AI: Two paths to prompting*. https://www.oneusefulthing.org/p/working-with-ai-two-paths-to-prompting

Ollama. (2025). *Ollama’s new app*. Ollama Blog. https://ollama.com/public/Ollama's new app

OpenAI. (o. J.). *Best practices for prompt engineering with the OpenAI API | OpenAI Help Center*. Abgerufen 13. Juni 2024, von https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api

OpenAI. (2025a). *Agents*. https://platform.openai.com/docs/guides/agents

OpenAI. (2025b, Februar 20). *Reasoning best practices*. Reasoning Best Practices. https://platform.openai.com/docs/guides/reasoning-best-practices

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., & Lewis, M. (2023). *Measuring and Narrowing the Compositionality Gap in Language Models* (arXiv:2210.03350). arXiv. https://doi.org/10.48550/arXiv.2210.03350

Scheiter, K., Bauer, E., Omarchevska, Y., Schumacher, C., & Sailer, M. (2025). *Künstliche Intelligenz in der Schule*.

Spannagel, C. (Regisseur). (2023, Dezember 6). *Schnippeln war gestern—Wie KI Lehrende unterstützen kann* [Video recording]. https://www.youtube.com/watch?v=GnT8ivqmZ5w

Wang, L., Xu, W., Lan, Yihuai, Hu, Z., Lan, Yunshi, Lee, R. K.-W., & Lim, E.-P. (2023). *Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models* (arXiv:2305.04091). arXiv. https://doi.org/10.48550/arXiv.2305.04091

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). *Self-Consistency Improves Chain of Thought Reasoning in Language Models* (arXiv:2203.11171). arXiv. https://doi.org/10.48550/arXiv.2203.11171

Xu, X., Tao, C., Shen, T., Xu, C., Xu, H., Long, G., Lou, J., & Ma, S. (2024). *Re-Reading Improves Reasoning in Large Language Models* (arXiv:2309.06275). arXiv. https://doi.org/10.48550/arXiv.2309.06275

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023). *Tree of Thoughts: Deliberate Problem Solving with Large Language Models* (arXiv:2305.10601). arXiv. https://doi.org/10.48550/arXiv.2305.10601

Yin, Z., Wang, H., Horio, K., Kawahara, D., & Sekine, S. (2024). *Should We Respect LLMs? A Cross-Lingual Study on the Influence of Prompt Politeness on LLM Performance* (arXiv:2402.14531; Version 1). arXiv. https://doi.org/10.48550/arXiv.2402.14531

Zheng, H. S., Mishra, S., Chen, X., Cheng, H.-T., Chi, E. H., Le, Q. V., & Zhou, D. (2024). *Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models* (arXiv:2310.06117). arXiv. https://doi.org/10.48550/arXiv.2310.06117

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., & Chi, E. (2023). *Least-to-Most Prompting Enables Complex Reasoning in Large Language Models* (arXiv:2205.10625). arXiv. https://doi.org/10.48550/arXiv.2205.10625

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2023). *Large Language Models Are Human-Level Prompt Engineers* (arXiv:2211.01910). arXiv. https://doi.org/10.48550/arXiv.2211.01910

6 Autorinnen und Autoren

Tjark Müller
E-Mail: t.mueller@iwm-tuebingen.de
ORCID: 0000-0003-2572-3881

Tjark Müller ist promovierter Psychologe und arbeitet als Programmierer am Leibniz-Institut für Wissensmedien. Er beschäftigt sich unter anderem mit der Integration neuer Technologien in die (Hochschul-)lehre.

Impressum

Zitierhinweis

Müller, T. (2026). *Einführung in Prompt-Strategien für Lehrkäfte.* schule-mal-digital.de.

Herausgeber

schule-mal-digital.de
Stiftung Medien in der Bildung (SbR) | Leibniz-Institut für Wissensmedien
Schleichstraße 6
72076 Tübingen
https://www.schule-mal-digital.de
Kontakt: schule-mal-digital@iwm-tuebingen.de

Über schule-mal-digital.de

Das Informationsportal schule-mal-digital.de ist ein nicht-kommerzielles Angebot des Leibniz-Instituts für Wissensmedien in Tübingen und bietet umfangreiche Informationen zur Gestaltung von Hochschulbildung mit digitalen Medien.