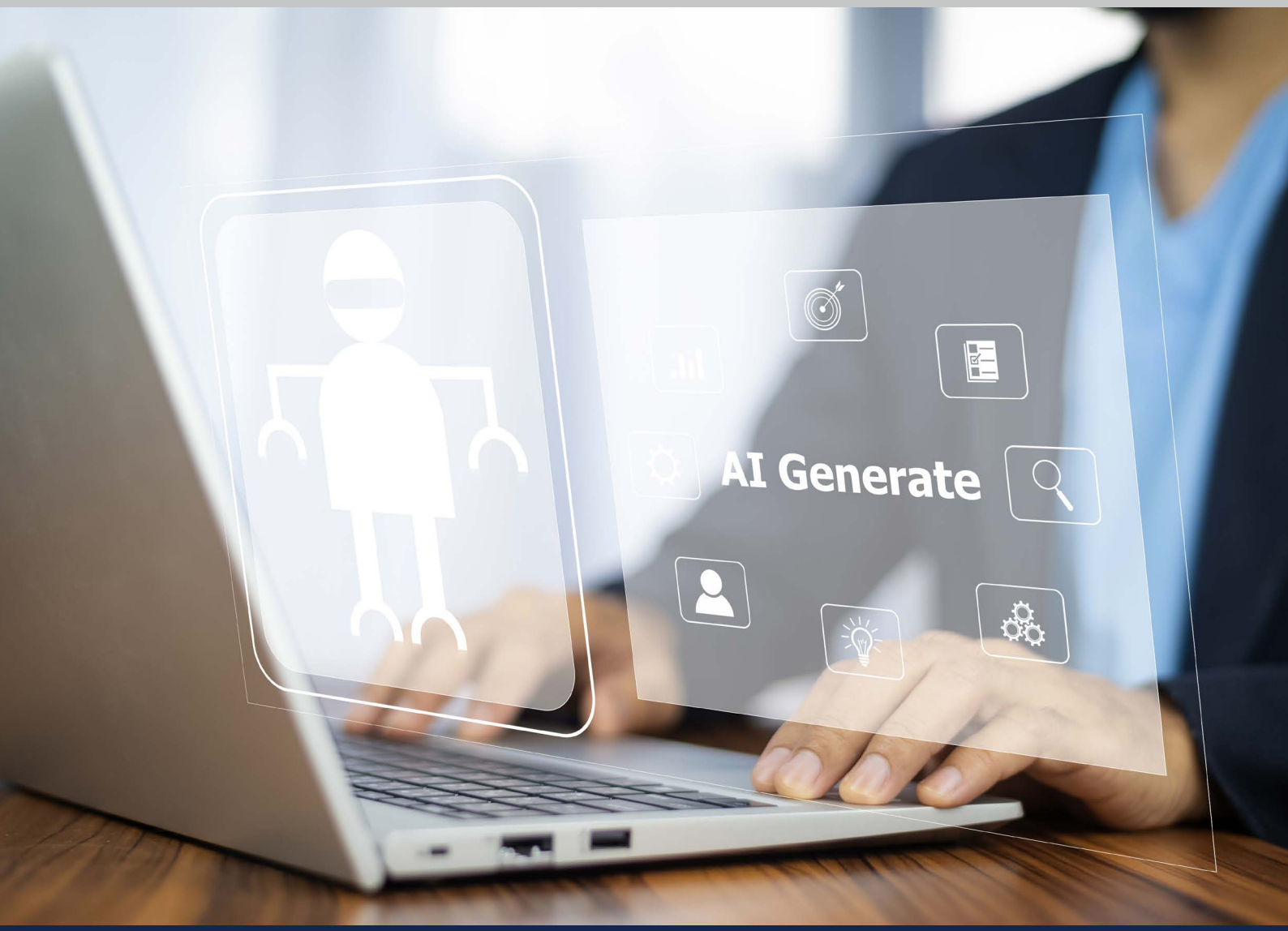
FORUM



Produktiver mit KI:
Der Praxisleitfaden zu Prompting Frameworks

Wie Projekt- und Produktmanager*innen KI strukturiert nutzen

Sehr geehrte Leser*innen,

KI gestützte Textmodelle beschleunigen Recherche, Service und Content. Verlässliche Ergebnisse gelingen mit klarer Anleitung. Prompting Frameworks liefern diese Struktur: Sie definieren Denk und Arbeitsschritte, binden Quellen und Tools ein und machen Antworten prüfbar. Dieses Whitepaper bietet einen laienfreundlichen, fundierten Einstieg: die wichtigsten Frameworks mit Nutzen, Grenzen und verständlichen Alltagsbeispielen – ergänzt um einfache Messmethoden und eine kompakte Checkliste für schnellen, sicheren Einsatz.

Dieses Whitepaper wurde mit Hilfe von ChatGPT erstellt.

Glossar: Wichtige Begriffe

- **Prompt**: Ihre strukturierte Anweisung an das KI-Modell (Ziel, Kontext, Format, Qualitätskriterien). Ein guter Prompt klärt das Was, das Wie und die Grenzen der Aufgabe; diese Formulierung ist stilistisch konsistent und vermeidet uneinheitliche Großschreibung in Anführungszeichen.
- **Systemprompt**: Festlegungen ganz oben im Gespräch, die die Rolle, den Ton und Grenzen des Modells bestimmen (z. B. „Antworte nur mit Quellen“). Wirkt stärker als spätere Nutzereingaben.
- **Framework**: Vorgehensweise, nach der Sie einen Prompt aufbauen (z. B. Schrittdenken, Plan-vor-Lösung, Antworten nur mit Quellen). Frameworks erhöhen Konsistenz und Prüfbarkeit.
- **Halluzination**: Faktisch falsche KI-Aussage, oft durch fehlende Quellen oder zu breite Aufgabenstellung verursacht. Mit RAG und klaren Einschränkungen reduzierbar.
- **RAG (Retrieval-Augmented Generation)**: Antworten mit eingebetteten, passenden externen Quellen. Senkt Halluzinationen, erhöht Zitierfähigkeit.
- **Embeddings**: Zahlenvektoren, die die Bedeutung von Texten abbilden. Grundlage für semantische Suche, Dokumentähnlichkeit und RAG.
- **Token**: Kleinste Verarbeitungseinheit eines Modells (oft Wortteile). Token beeinflussen Kosten und Längenbegrenzungen; präzise Prompts sparen Token.
- **Max Tokens**: Obergrenze für die Länge der Antwort (und teils Eingabe). Verhindert Abschweifen und schützt Budget; zu niedrig kann Antworten abschneiden.
- **Temperature**: Parameter für Zufälligkeit. Niedrig (0–0,2) liefert stabile, reproduzierbare Antworten; höher (0,7–1,0) fördert Varianz und Kreativität, erhöht aber die Fehleranfälligkeit.
- **Seed**: Startwert, um Zufall zu kontrollieren. Mit gleichem Seed, Temperature und Parametern steigen Reproduzierbarkeit und Vergleichbarkeit.

- **Sandbox**: Gesicherte Umgebung, in der generierter Code (z. B. aus PoT) kontrolliert ausgeführt wird.

Häufige Missverständnisse und wie Sie sie vermeiden

- **„Mehr Kontext = automatisch bessere Antwort“**: Relevanz schlägt Länge. Nutzen Sie präzise, aktuelle Quellen statt unspezifischen Zusatztext.
- **„Frameworks sind nur für Profis“**: Viele Muster (CoT, Plan-and-Solve, Socratic) sind leicht anzuwenden und entfalten schnell Wirkung.
- **„Code in KI ist gefährlich“**: Richtig, ohne Sandbox. Mit sicherer Umgebung ist PoT für Zahlen und Regeln sehr zuverlässig.
- **„RAG behebt alle Fehler“**: RAG hilft gegen Halluzinationen, setzt aber gute Dokumente und sauberes Retrieval voraus.
- **„Self-Consistency ist immer besser“**: Nur dort einsetzen, wo der Qualitätsgewinn die Mehrkosten rechtfertigt.

Was sind Prompting-Frameworks?

Ein Prompting-Framework ist keine Programmierung, sondern eine Arbeitsweise, bei der Sie der KI klare Schritte vorgeben, zunächst planen, dann lösen; Antworten stets mit Quellen belegen; Tools gezielt einsetzen und Beobachtungen erläutern; diese Überarbeitung schafft parallele Struktur und flüssigere Lesbarkeit. Solche Strukturen führen zu stabileren, nachvollziehbaren Ergebnissen. Das ist wichtig, weil KI-Modelle zwar leistungsfähig, aber nicht unfehlbar sind. Mit Frameworks senken Sie die Fehlerrate, sparen Zeit und erhalten Antworten, die sich prüfen lassen.

Die wichtigsten Frameworks, ausführlich erklärt

Chain-of-Thought (CoT)

CoT fordert „Schritt-für-Schritt“-Denken ein. Statt direkt eine Lösung zu geben, nennt die KI erst Zwischenüberlegungen und dann das Ergebnis. Das erhöht die Transparenz, reduziert logische Fehler und macht Entscheidungen nachvollziehbar. Typische Anwendung sind Regelprüfungen (z. B. Kulanzkriterien), einfache Berechnungen oder komplexe Einschätzungen. In der Praxis formulieren Sie eine klare Aufgabe, fügen, wenn nötig, Beispiele hinzu und **bitten explizit um strukturierte Zwischenschritte.**

Produktivitätswirkung: bessere Qualität bei moderatem Mehraufwand an Tokens.
Grenzen: längere Antworten und damit höhere Kosten; bei sehr einfachen Aufgaben oft überdimensioniert.

Self-Consistency

Self-Consistency führt eine Abfrage mehrfach mit kleinen Variationen aus und wählt am Ende per Mehrheitsentscheid die konsistenteste Antwort. Dies strafft die Sprache und präzisiert den Auswahlmechanismus. Es senkt auch Zufallsfehler und stabilisiert Ergebnisse, insbesondere bei kniffligem Reasoning. Sie setzen es ein, wenn die Antwort wichtig ist und Fehler teuer wären (z. B. rechtliche Einschätzungen, sensible Policy-Entscheidungen). Praktisch bedeutet das: mehrere Läufe, gleiche Aufgabe, Auswahl der häufigsten bzw. plausibelsten Lösung.
Produktivitätswirkung: deutlich geringere Fehlerrate; Grenzen: höhere Latenz und Kosten durch Mehrfachläufe.

Tree-of-Thoughts (ToT)

ToT lässt die KI mehrere Lösungswege parallel entwickeln, bewertet Zwischenergebnisse und wählt den besten Pfad. Das ähnelt einer strategischen Suche mit Alternativen, Rücksprungpunkten und Bewertungen. Sinnvoll ist ToT bei Planung, Strategie, Szenarioanalyse oder komplexen Puzzles („Was-wäre-wenn“). Sie instruieren das Modell, mehrere Ideenpfade anzulegen, Kriterien zur Bewertung zu nutzen und die beste Option zu begründen.
Produktivitätswirkung: tiefere Problemlösung und bessere Entscheidungen; Grenzen: höherer Aufwand und längere Laufzeit.

ReAct (Reason + Act)

ReAct kombiniert Denken und Handeln: Das Modell überlegt, ruft dann gezielt ein Tool (z. B. Websuche, Datenbank, Kalkulator) auf, beobachtet das Ergebnis und denkt weiter. So entstehen aktuelle, quellenbezogene Antworten mit klaren Belegen. Geeignet ist ReAct für Support, Recherche, Compliance und alle Aufgaben, die eine Aktion erfordern (z. B. Ticket anlegen). Sie definieren erlaubte Tools und geben vor, dass Gedanken, Aktion und Beobachtung jeweils dokumentiert werden.
Produktivitätswirkung: mehr Aktualität und Verlässlichkeit; Grenzen: komplexere Einrichtung, Tool-Maintenance, potenziell höhere Latenz.

Plan-and-Solve

Plan-and-Solve trennt die Aufgabe in zwei Phasen: Erst wird ein kurzer Plan erstellt, dann die Lösung umgesetzt. Das schafft Struktur, vermeidet Sprünge und reduziert Missverständnisse. Ideal für geordnete Alltagsaufgaben wie Ticket-Triage,

Qualitätsprüfungen oder Checklisten. In der Praxis: Sie bitten um einen knappen Plan mit Schritten und lassen die KI danach jeden Schritt ausführen.
Produktivitätswirkung: klare Abläufe, weniger Fehler; Grenzen: bei kreativen Aufgaben kann die Struktur zu starr sein.

Program-of-Thoughts (PoT)
PoT lässt die KI Zwischenschritte als kleinen Code (z. B. Python) formulieren, der dann ausgeführt wird. Das ist besonders nützlich für Zahlen, Regeln, Metriken und Berechnungen (z. B. Finanzkennzahlen, KPI-Berechnung). Sie geben die Aufgabe vor und fordern Code mit Ausgabe an; wichtig ist eine sichere Sandbox, in der der Code gefahrlos läuft.
Produktivitätswirkung: hohe numerische Präzision und Reproduzierbarkeit; Grenzen: technische Einrichtung, Sicherheitsanforderungen.

Socratic Prompting
Socratic arbeitet mit gezielten Rückfragen: Das Modell klärt Aufgabe, Begriffe, Ziel, Randbedingungen und Qualitätserwartungen. Das führt zu besseren Briefings, klareren Anforderungen und weniger Nacharbeit. Sie starten mit offenen Leitfragen („Was ist das Ziel? Welche Einschränkungen gibt es?“), lassen die KI Unklarheiten sammeln und anschließend präzise Antworten geben.
Produktivitätswirkung: höhere Klarheit und Nutzbarkeit im Alltag (1–5); Grenzen: benötigt etwas mehr Zeit am Anfang, lohnt sich jedoch durch weniger Iterationsschleifen.

Retrieval-Augmented Generation (RAG)
RAG erweitert die Antwort mit passenden externen Quellen (z. B. Wissensdatenbank, Handbuch, Richtlinie). Das reduziert Halluzinationen und erhöht die Zitierfähigkeit. Sie definieren, welche Dokumente durchsucht werden dürfen, und fordern, dass jede Aussage belegt wird. Ideal für Support, Compliance, Produktinfos, Forensik und Forschung.
Produktivitätswirkung: verlässliche, aktuelle Antworten; Grenzen: Aufbau und Pflege der Wissensbasis, Qualitätsabhängigkeit vom Retrieval.

Toolformer / Function Calling
Hier ruft die KI definierte Funktionen oder APIs auf, zum Beispiel, um Daten zu holen, Checks zu machen oder Tickets anzulegen. Die Antwort bleibt strukturiert (oft in

JSON), Aktionen werden deterministisch ausgeführt. Sie definieren erlaubte Funktionen, Parameter und Rückgabestrukturen.
Produktivitätswirkung: Automatisierung, weniger manuelle Schritte; Grenzen: sauberes Mapping der Parameter, Rechteverwaltung und Fehlerhandling sind Pflicht.

Reflexion
Reflexion bedeutet, dass die KI die eigene Antwort aktiv überprüft und verbessert, etwa durch einen zweiten Durchgang mit Kritik und Überarbeitung. In der Praxis ist das noch nicht überall stabil, kann aber bei Textqualität, Argumentation und Zusammenfassungen helfen. Sie fordern eine Selbstprüfung mit klaren Kriterien („Stimmt die Logik? Fehlen wesentliche Punkte?“) und lassen danach gezielt korrigieren.
Produktivitätswirkung: bessere Qualität; Grenzen: uneinheitliche Ergebnisse, zusätzlicher Lauf.

Endanwender-Kurzrezepte je Framework

- CoT: Formulieren Sie klare Regeln/Schritte und fordern Sie „Schritt-für-Schritt“ an.
- Self-Consistency: Lassen Sie in verschiedenen Chats drei Läufe desselben CoT-Prompts laufen und nehmen Sie das Mehrheitsvotum.
- ToT: Bitten Sie um drei Optionen mit Bewertung (z. B. 1–5) und eine begründete Entscheidung.
- ReAct: Nennen Sie erlaubte Tools und fordern Sie „Gedanke → Aktion → Beobachtung“ explizit ein.
- Plan-and-Solve: Erst 5–7 Schritte planen lassen, dann jeden Schritt ausführen; Ausgabe als Checkliste.
- PoT: Kurzen Python-Rechenweg erzeugen lassen und das Ergebnis ausgeben; nur in einer sicheren Umgebung ausführen.
- Socratic: 8–10 Rückfragen zur Klärung einfordern und am Ende ein kompaktes Briefing.
- RAG: „Nur bereitgestellte Passagen verwenden“ und jede Aussage mit Quelle/Abschnitt belegen lassen.
- Function Calling: Einen klaren Funktionsaufruf mit Parametern formulieren und eine strukturierte Rückgabe verlangen.
- Reflexion: Antwort prüfen lassen (Logik, Vollständigkeit) und gezielt korrigieren.

Wo steigern Frameworks die Produktivität?

- **Wissensarbeit/Recherche**: Mehr Klarheit durch Socratic und Plan-and-Solve. Aktuelle und belegte Antworten mit RAG oder ReAct. CoT erhöht die Argumentationsqualität bei komplexen Fragen.
- **Service/Kundendienst**: RAG liefert verlässliche Antworten aus der Wissensbasis mit Quellen. ReAct ermöglicht Aktionen und Ticketing. CoT hilft bei Kulanz- und Eskalationslogik, Plan-and-Solve bei Triage.
- **Marketing/Content**: Briefings mit Socratic sind präziser, Templates (z. B. CO-STAR) schaffen konsistente Vorgaben. RAG senkt Faktenfehler bei Produktinfos und Studien.
- **Softwareentwicklung/Regelwerke**: PoT verbessert Zahlen und Regeln (z. B. KPI-Berechnungen). Function Calling integriert Checks und Automatisierung. ToT unterstützt Architektur- und Planungsentscheidungen.

Praxisbeispiele zum Nachmachen

Im Folgenden finden Sie sechs kurze, eigenständige Szenarien aus dem Arbeitsalltag. Jedes Beispiel liefert Kontext, eine klare Aufgabe, einen geeigneten Prompt-Stil, Hinweise zur erwarteten Ausgabe und einen Vorschlag, wie Sie das Ergebnis objektiv prüfen. Alle Beispiele lassen sich mit gängigen Modellen reproduzieren (z. B. temperature=0, max_tokens 300–600).

- Socratic Prompting, Briefing klären, bevor wir startenKontext: Sie möchten einen kurzen Leitfaden für neue Kund*innen erstellen. Ziel: Ziel, Zielgruppe, Umfang und Erfolgskriterien klären, bevor Textarbeit beginnt.So prompten Sie: „Stelle mir nacheinander die wichtigsten 8–10 Fragen, um ein präzises Briefing für einen 1-seitigen Leitfaden zum Thema ‘Erste Schritte mit unserem Serviceportal’ zu erstellen. Fasse zum Schluss meine Antworten als Briefing (Ziel, Zielgruppe, Umfang, Tonalität, Quellen, Done-Kriterien) zusammen.“Erwartete Ausgabe: Eine geordnete Fragerunde und eine kompakte Briefing-Zusammenfassung.So prüfen Sie: Ist das Briefing vollständig (Ziel, Zielgruppe, Umfang, Ton, Quellen, Done-Kriterien)? Erkennen Sie sofort, was als Nächstes zu tun ist? Zeit bis zur eigenen Weiterarbeit (und ggf. Teamfreigabe) messen.
- Plan-and-Solve, Checkliste für ein Kick-off-MeetingKontext: Ein Projekt startet, viele Beteiligte, wenig Zeit. Ziel: Erst den Plan erstellen und anschließend eine ausführbare Checkliste formulieren.So prompten Sie: „Erstelle zuerst einen knappen Plan mit 5–7 Schritten für ein 60-minütiges Projekt-Kick-off (Ziel:

gemeinsames Verständnis, Risiken, nächste Schritte). Führe danach jeden Schritt aus und formuliere eine Checkliste mit konkreten Punkten und Verantwortlichkeiten.“Erwartete Ausgabe: Plan (Schrittfolge) und ausführbare Checkliste (klar, zuweisbar). So prüfen Sie: Sind alle Pflichtpunkte enthalten (Ziele, Scope, Rollen, Risiken, nächste Schritte)? Lassen sich die Aufgaben zuweisen? Reduziert sich Nacharbeit im Anschluss-Meeting?

Dauer: ca. 10–15 MinutenErwartete Nutzbarkeit im Alltag: 5/5

- Chain-of-Thought, Entscheidungsregeln transparent anwendenKontext: Meetings kosten Zeit; nicht jedes ist nötig.Aufgabe: Prüfen Sie nachvollziehbar, ob ein Meeting stattfinden sollte.So prompten Sie: „Prüfe Schritt für Schritt, ob ein 30-minütiges Meeting sinnvoll ist. Regeln: (1) Es gibt ein konkretes Ziel, (2) Entscheidungsträger*in ist verfügbar, (3) nötige Vorinfos sind vorhanden. Erkläre pro Regel kurz JA/NEIN und gib am Ende eine Empfehlung mit 2 Alternativen (z. B. asynchrones Update).“Erwartete Ausgabe: Drei überprüfte Regeln, kurze Begründungen, klare Empfehlung + Alternativen.So prüfen Sie: Stimmen die Regelprüfungen mit der Realität überein? Führen Alternativen zu gleicher oder besserer Ergebnisqualität bei weniger Zeitaufwand?

Dauer: ca. 5 MinutenErwartete Nutzbarkeit im Alltag: 4/5

- Program-of-Thoughts, Budget sauber berechnenKontext: Sie planen einen Workshop mit 18 Teilnehmenden. Ziel: Gesamtkosten und Kosten pro Person nachvollziehbar ermitteln.So prompten Sie: „Erzeuge Python-Code, der die Gesamtkosten und Kosten pro Person berechnet. Gegeben: Raum 450 €, Catering 18×28 €, Material pauschal 120 €, USt 19 %. Gib die Zwischenschritte (Netto, USt, Brutto) aus.“Erwartete Ausgabe: Kurzer, ausführbarer Code mit sauberer Ausgabe (Netto/Brutto, pro Person). So prüfen Sie: Rechnen Sie stichprobenartig nach; Ergebnisse müssen identisch sein. Dokumentieren Sie Eingabewerte und Output in Ihrem Log.

Dauer: ca. 5–8 MinutenErwartete Nutzbarkeit im Alltag: 5/5

- RAG oder ReAct, Fakten aus bereitgestellten TextenKontext: Sie haben einen Auszug aus einer Richtlinie (5–10 Sätze). Ziel: Nur aus dem angegebenen Textauszug antworten und die Fundstelle zitieren.So prompten Sie: „Nutze nur den folgenden Textauszug und beantworte: ‘Welche Fristen gelten für die Anmeldung?’ Zitiere Absatz/Zeile. Wenn etwas fehlt, sage ausdrücklich: ‚Im Text nicht enthalten.‘ Auszug: ‹Text hier einfügen›.“Erwartete Ausgabe: Kurze, zitiergenaue Antwort oder klarer Hinweis auf fehlende Information.So prüfen Sie:

Randomisieren Sie 3–5 Fragen je Auszug. Quote-Genauigkeit muss 100 % sein. Antworten ohne Zitat gelten als Fail.

Dauer: ca. 5–10 Minuten Erwartete Nutzbarkeit im Alltag: 4/5

- Tree-of-Thoughts, Optionen entwickeln und begründet wählen Kontext: Sie wählen einen Kanal für eine Produktankündigung (z. B. Blog, Webinar, LinkedIn). Ziel: Mehrere Optionen entwickeln, bewerten und begründet entscheiden. So prompten Sie: „Entwickle drei Handlungsoptionen (Blog, Webinar, LinkedIn-Kampagne). Liste pro Option Ziel, Aufwand, Reichweite, benötigte Assets, Risiko. Bewerte mit Schulnoten (1–5) und wähle die beste Option. Begründe die Wahl und nenne die zwei wichtigsten nächsten Schritte.“ Erwartete Ausgabe: Drei vergleichbare Optionen, transparente Bewertung, klare Entscheidung mit nächsten Schritten. So prüfen Sie: Sind die Kriterien vergleichbar ausgefüllt? Ist die Begründung konsistent mit den Bewertungen? Lassen sich die nächsten Schritte sofort terminieren?

Dauer: ca. 10–15 Minuten Erwartete Nutzbarkeit im Alltag: 4/5

- Tipp zur Reproduzierbarkeit: Halten Sie Modellname, Parameter (z. B. temperature=0), Prompt und Antwort im Log fest. So erkennen Sie, ob Änderungen am Prompt wirklich zu besseren Ergebnissen führen.

Prompt-Testing: Wann lohnt sich das?

Warum überhaupt testen? Ein kurzer Test hilft Ihnen zu entscheiden, ob ein Prompt oder Framework im Alltag wirklich Nutzen stiftet. Sie müssen dafür nicht in technische Details gehen.

Wann testen?

- Wenn Sie einen neuen Prompt oder ein neues Framework ausprobieren.
- Wenn Inhalte fachlich heikel sind (Compliance, Kund*innenkommunikation).
- Wenn die Aufgabe regelmäßig wiederkehrt (z. B. FAQs, Reports, Checklisten).
- Wenn es Beschwerden über Qualität, Zeit oder Kosten gibt.

Was ist ein „Test“?

- Ein kompakter Vergleich mit 3–5 typischen Fällen, um festzustellen, ob die KI das gewünschte Ergebnis konsistent liefert.

Grundbegriffe (kurz erklärt)

- **Testfall**: Konkrete Aufgabe mit erwarteter Antwort (Ihr „Goldstandard“).
- **Konsistenz-Einstellung**: Temperature 0 und Top-p 1.0 für stabilere Antworten.
- **Max Tokens**: Obergrenze für die Antwortlänge; zu niedrig schneidet Antworten ab.
- **Logging light**: Prompt, Antwort, Dauer und (bei RAG) Quelle/Zitat notieren.
- **A/B-Vergleich**: Zwei Prompts/Frameworks kurz gegeneinander testen.

So gehen Sie vor (in 3 Schritten)

- Ziel und Done-Kriterien notieren (max. 3 Punkte: z. B. „Quellenpflicht“, „Antwort verständlich in 5 Sätzen“).
- 3–5 Testfälle durchspielen; Qualität 1–5 bewerten, Zeit notieren, bei RAG Zitat prüfen.
- Besseres Vorgehen wählen; falls nötig Prompt präzisieren/kürzen und einen weiteren Durchgang machen.

Ergebnis

- Sie entscheiden pragmatisch, ob sich der Einsatz lohnt, ohne technische Tiefe.
- Für RAG gilt: Jede Aussage muss belegt sein (Quote-Genauigkeit 100 %).

Wie Sie LLM-Aussagen verifizieren

Dieser Abschnitt hilft Ihnen, Antworten eines Sprachmodells pragmatisch zu überprüfen, ohne technische Tiefe. Ziel ist, Aussagen entweder als „verifiziert“, „teilweise verifiziert“ oder „nicht verifiziert“ einzustufen.

Quellenpflicht erzwingen

- Bitten Sie die KI, jede relevante Aussage mit einer Quelle zu belegen (Link, Dokumentname, Absatz/Zeile).
- Bei bereitgestellten Texten (RAG): „Antworte nur aus diesen Passagen und zitiere die Stelle.“
- Prüfen Sie, ob das Zitat wirklich die genannte Aussage trägt (Kontext lesen, nicht nur einen Satz).

Zahlen und Regeln gegenrechnen

- Lassen Sie sich Rechenschritte kurz zeigen („Zeige Zwischenschritte“).
- Optional: PoT fragen („Erzeuge berechneten Zwischenschritt als Code“) und Ergebnis stichprobenartig nachrechnen.

- Bei Regeln: Jede Regel einzeln prüfen (JA/NEIN) und Begründung lesen.
  Cross-Check mit unabhängigen Quellen
- Für wichtige Aussagen: 1–2 zusätzliche, seriöse Quellen prüfen (amtliche Seiten, Primärliteratur, Herstellerdokumentation).
- Abweichungen notieren und den Grad der Übereinstimmung bewerten. Aktualität und Version prüfen
- Datumsangaben und Versionsstände der Quelle kontrollieren (z. B. Richtlinien, Produktdokumentation).
- Bei zeitkritischen Themen (Preise, Fristen, Verfügbarkeiten): nur Quellen mit aktuellem Datum verwenden. Halluzinations-Signale erkennen
- Keine oder vage Quellen („Quelle: Internet“), absolute Aussagen ohne Nachweis, vermischte Fachbegriffe, widersprüchliche Logik.
- In solchen Fällen: explizit um Präzisierung und Zitat bitten oder die Aussage als „nicht verifiziert“ einstufen. Ergebnis einstufen
- Verifiziert: Aussage ist durch Quellen eindeutig belegt und Zahlen stimmen.
- Teilweise verifiziert: Teile belegt, Rest unklar, als „offen“ markieren, Nachprüfung planen.
- Nicht verifiziert: Keine belastbare Quelle oder Widerspruch, nicht verwenden.

Kurze Protokollvorlage

- Aufgabe/Frage, Prompt (Kurzfassung), Antwort (Kurzfassung), Quelle/Zitat, Datum/Version, Prüfnotiz (OK/Teilweise/Nicht), nächste Schritte.
- Bei RAG: Dokument-ID und Passage notieren.

Hinweis für Nicht-Expert*innen

- Bitten Sie die KI, zu jeder Antwort einen „Unsicherheitsgrad“ (niedrig/mittel/hoch) und „Was noch zu prüfen ist“ in 1–2 Punkten zu ergänzen.
- Das erleichtert die Entscheidung, ob eine Aussage sofort nutzbar ist oder noch Validierung braucht.

Risiken und Sicherheit, verständlich erklärt

- Faktenfehler/Halluzinationen: Was kann passieren? Modelle erfinden Details oder mischen Quellen. Wie vermeiden? Aufgaben klar begrenzen („Antworte nur aus diesen Passagen“), bei Fakten eine Quelle verlangen, RAG einsetzen, ein konkretes Ausgabeformat anfordern, wie zum Bespiel eine Zusammenfassung

- Prompt-Injection/Jailbreaks: Was ist das? In fremden Texten (z. B. E-Mails, Webseiten oder Kundentickets) können versteckte Anweisungen stehen, die die KI dazu bringen, ihre eigentliche Aufgabe zu verlassen oder unerwünschte Aktionen auszuführen. Wie vermeiden? Begrenzen Sie die KI auf erlaubte Aufgaben, filtern Sie externe Eingaben und geben Sie dem Modell klare Anweisungen, nur Ihren Regeln zu folgen und Anweisungen in Quellen zu ignorieren. Sensible Aktionen sollten immer zusätzlich bestätigt werden. Vermeiden Sie es, vertrauliche Daten in Prompts einzugeben.
- Datenschutz/Compliance Risiko: Personenbezogene Daten in Prompts/Protokollen, ungeklärte Speicherung. Schutz: Daten minimieren und anonymisieren, keine sensiblen Daten in freie Prompts, Aufbewahrungsfristen definieren, Freigaben/Datenschutzvereinbarungen klären; bei Bedarf in kontrollierten Umgebungen arbeiten (z. B. On-Prem/virtuelle Desktops).
- Code-Sicherheit (PoT) Risiko: Generierter Code greift ungewollt zu oder läuft endlos. Schutz: Immer in einer Sandbox ausführen, Zeit- und Ressourcenlimits setzen, keinen Internet-/Dateizugriff erlauben, nur Testdaten verwenden, vorab kurz sichten.
- Kosten Latenz Risiko: Mehrfachläufe (Self-Consistency), Suchbäume (ToT) oder Rechercheschleifen (RAG/ReAct) erhöhen Zeit und Kosten. Steuerung: Token-Limits sinnvoll setzen, Caching nutzen, Retrieval nur bei Bedarf, einfache Fälle ohne Schleifen lösen, Laufzeiten messen und Schwellen definieren.

Sofortmaßnahmen für Endanwender*innen

- Halluzinationen: Quellen verlangen; bei bereitgestellten Texten Zitate prüfen; knappe, faktenbezogene Antworten einfordern.
- Prompt-Injection/Jailbreaks: Nur eigene Regeln im Systemprompt; Anweisungen in Quellen ignorieren lassen; keine sensiblen Daten eingeben.
- Tool/API-Risiken: Nur freigegebene Funktionen nutzen; Ergebnisse kurz bestätigen.
- Datenschutz: Keine personenbezogenen Daten in Prompts; Inhalte anonymisieren.
- PoT-Code: Nur in isolierter Umgebung laufen lassen; kurz sichten vor Ausführung.
- Kosten/Latenz: Bei einfachen Aufgaben ohne Schleifen arbeiten; kurze Ausgaben anfordern.

Minicheckliste Sicherheit

- Ist die Aufgabe begrenzt und sind Quellen klar genannt?
- Gibt es eine Quellenpflicht bei Fakten?
- Sind Funktionen/Rechte auf das Nötige beschränkt?
- Werden personenbezogene Daten vermieden/anonymisiert?
- Läuft PoT-Code streng isoliert mit Timeouts?

Handlungsempfehlungen für den Einstieg

- **Kleine Schritte**: Starten Sie mit Socratic und Plan-and-Solve für Klarheit und Struktur.
- **Gezielte Qualität**: Für Rechen- und Regelaufgaben PoT oder CoT; bei kritischen Fällen Self-Consistency.
- **Aktualität und Aktionen**: Für Wissensbezug RAG; für operative Schritte ReAct/Function Calling.
- **Governance**: Sandbox für PoT-Code, Rollen und Zugriffe für Tools, klare Quellenpolitik.
- **Kontinuierliche Messung**: Monatliche Reviews der Metriken und Anpassung des Framework-Portfolios.

Checkliste

Mit dieser kompakten Checkliste planen und testen Sie Ihren KI-Einsatz Schritt für Schritt, ganz ohne technische Tiefe:

- **Ziel und Done-Kriterien festlegen:** Notieren Sie in einem Satz, was Sie erreichen möchten. Halten Sie bis zu drei klare Erfolgskriterien fest, zum Beispiel: Quellenpflicht, Verständlichkeit in weniger als fünf Sätzen, oder klare nächste Schritte.
- **Konsistenz-Einstellung wählen:** Stellen Sie die Temperature auf 0 und Top-p auf 1.0. Wählen Sie die maximale Token-Zahl falls möglich passend zur Aufgabe (z. B. 300–600), damit die Antwort nicht abgeschnitten wird. Nutzen Sie einen stabilen Systemprompt (Rolle, Ton, Grenzen) und einheitliche Formatvorgaben.
- **Framework auswählen:** Entscheiden Sie je nach Aufgabe, welches Vorgehen passt, zum Beispiel Socratic für Klarheit, Plan-and-Solve für strukturierte Abläufe, Chain-of-Thought für nachvollziehbare Regeln, Program-of-Thoughts für Berechnungen, RAG/ReAct für belegte Fakten, Tree-of-Thoughts für Optionen und Entscheidungen.
- **Testfragen nutzen:** Legen Sie drei bis fünf typische Testfragen mit erwarteten Antworten fest. Bei RAG gilt: Jede Aussage muss mit Zitat oder Quelle belegt sein.

- **Durchführung und Protokoll:** Notieren Sie zu jeder Testfrage Prompt, Antwort und Zeit. Bei RAG erfassen Sie zusätzlich das Zitat oder die Quelle. Bei PoT führen Sie Code immer in einer sicheren Umgebung aus.
- **Bewertung:** Bewerten Sie jede Antwort nach Qualität (1–5), Zeit, Fehlerrate (Pass/Fail) und Nutzbarkeit im Alltag (1–5).
- **A/B-Vergleich:** Testen Sie zwei verschiedene Prompts oder Frameworks mit je drei Fällen. Wählen Sie das bessere Vorgehen anhand von Qualität, Zeit und Nutzbarkeit im Alltag aus.
- **Entscheidung und nächste Schritte:** Halten Sie fest, welches Framework gewonnen hat und was als Nächstes geschieht, zum Beispiel: Template speichern oder kurze Leitfaden-Notiz.
- **Risiken und Maßnahmen prüfen:** Achten Sie auf Halluzinationen (Quellenpflicht), schützen Sie sich vor Prompt-Injection (klare Systemgrenzen), und behalten Sie Kosten und Laufzeiten im Blick (Token-Limits, Caching).
- **Archivierung:** Sichern Sie Prompts, Beispielantworten, Systemprompt und Version für die spätere Wiederverwendung.

Literatur und weiterführende Links

Hinweis: Optional, weiterführend.

Chain-of-Thought: Wei et al., 2022. „Chain-of-Thought Prompting …“

Self-Consistency: Wang et al., 2022. „Self-Consistency Improves CoT …“

Tree-of-Thoughts: Yao et al., 2023. „Tree of Thoughts…“

ReAct: Yao et al., 2022. „ReAct: Synergizing Reasoning and Acting …“

Plan-and-Solve: Wang et al., 2023. „Plan-and-Solve Prompting …“

Program-of-Thoughts: Chen et al., 2023. „Program of Thoughts Prompting …“

Self-Ask/Socratic: Press et al., 2022. „Measuring and Narrowing the Compositionality Gap …“

RAG: Lewis et al., 2020. „Retrieval-Augmented Generation …“

Toolformer: Schick et al., 2023. „Language Models Can Teach Themselves to Use Tools“

Function Calling: OpenAI-Doku

Reflexion: Shinn et al., 2023. „Reflexion: Language Agents with Verbal RL“

Überblick Praxis (Embeddings/RAG): O’Reilly 2024, „Hands-On Large Language Models“ (Kapitel zu Embeddings, RAG, Prompting)

Hinweis: Für die interne Validierung empfehlen wir ein kurzes Mini-Benchmark-Set pro Bereich (10–30 Fälle) mit erwarteten Ergebnissen. So bleibt Ihre Bewertung einfach, fair und reproduzierbar.

Passende Weiterbildungen finden Sie hier:

Das Beste für Sie – Weiterbildung für Assistenz und Sachbearbeitung

Finden Sie passende Seminare zu unterschiedlichen Themen von Arbeitsorganisation über Kommunikation bis zu speziellen Fachthemen sowie Lehrgängen mit IHK-Zertifikat. Unsere erfahrenen Expert*innen bieten handfeste Lösungen, neue Kompetenzen, digitale Tools und wertvolle Praxistipps. Jetzt informieren.

e-Learning – Klicken und Lernen

Das FORUM Institut bietet mit hochwertigen e-Learning-Programmen eine flexible Weiterbildungsform. Entscheiden Sie selbst, wann und wo Sie lernen.
Jetzt testen.

Inhouse-Seminare – Maßgeschneiderte Lösungen

Alle unsere Seminare eignen sich auch hervorragend als Inhouse-Training.
Jetzt individuelles Angebot anfordern.

FORUM · Institut für Management GmbH · Postfach 10 50 60 · 69040 Heidelberg · Germany
Telefon: +49 6221 500-500 · Fax: +49 6221 500-555 · www.forum-institut.de