? Prompt Engineering

Methodiken und praktische Prompt-Vorlagen für den täglichen Einsatz mit KI-Agenten. Enthält sowohl Strategien (Reverse Prompt Engineering, Amplifier, Red Teaming, Scaffolding) als auch fertige Vorlagen (BlogPost, Context Extraction, Summary, Document Generation).

Blueprint Scaffolding (Structured Prompt Planning)

Was es ist

Bevor du die KI bittest, eine vollständige Antwort oder ein komplettes Dokument zu erzeugen, lässt du sie zunächst eine **Blaupause / Struktur / einen Plan / eine Schritt-für-Schritt-Gliederung** erstellen (das sogenannte „Scaffold“).
Nach Durchsicht und Feinjustierung dieser Struktur beauftragst du die KI anschließend, den finalen Inhalt auszuarbeiten.

Warum das effektiv ist

- Verhindert überkomplizierte, unfokussierte oder aufgeblähte Ergebnisse durch zu generische Prompts.
- Gibt dir volle Kontrolle über **Umfang, Struktur und Relevanz** des Ergebnisses.
- Erleichtert die Anwendung des **80/20-Prinzips**: Wesentliches definieren, Unnötiges streichen.
- Hilft, strukturelle Fehler oder Fehlannahmen früh zu erkennen — bevor zu viel „Beton gegossen“ wird.

Typischer Workflow

1. Du formulierst ein übergeordnetes Ziel
   (z. B. „Ich brauche ein Marketingkampagnen-Briefing für die Q4-Weihnachtsaktion“).
2. Du bittest die KI, eine **Gliederung der Abschnitte** mit kurzen Beschreibungen auszugeben.
3. Du prüfst die Struktur — entfernst oder fasst überflüssige Abschnitte zusammen (Vereinfachung).
4. Sobald die Struktur passt, lässt du die KI auf Basis der verfeinerten Blaupause **den finalen Inhalt ausarbeiten**.

Wann man es einsetzen sollte

- Bei **komplexen Aufgaben** mit mehreren Komponenten
  (Kampagnen, Konzepte, Angebote, Essays, Reports, mehrstufige Pläne).
- Wenn du **präzise und fokussierte Ergebnisse** willst statt „alles auf einmal“.
- Wenn Klarheit, Struktur und Relevanz wichtiger sind als reine Textmenge.

Vorteile & zusätzlicher Tipp

- Zwingt die KI dazu, ihren **Denk- und Aufbaupfad offenzulegen** — reduziert Halluzinationen und thematische Abschweifungen.
- Besonders wirkungsvoll wird der Ansatz, wenn du für jeden Abschnitt **Erfolgskriterien** definierst
  (z. B. „Dieser Teil soll drei konkrete Handlungsempfehlungen liefern“).

Empfohlene Lektüre & Ressourcen

- Prompt-Engineering-Guide, der Blueprint Scaffolding als zentrales Prompting-Muster beschreibt.
  (Upaspro)
- Umfassende Übersicht und Taxonomie von Prompting-Techniken — ordnet Scaffolding im Gesamtfeld ein.
  (arXiv)
- Diskussion zu Prompting-Ansätzen, rollenbasiertem vs. strukturiertem Prompting und typischen Fallstricken — hilfreich für den gezielten Einsatz von Scaffolding.
  (Medium)

Zusätzlicher Kontext: Prompt Engineering als Disziplin

Die oben beschriebenen Techniken gehören zum übergeordneten Feld des **Prompt Engineering** — der Kunst und Wissenschaft, Anweisungen für Large Language Models (LLMs) so zu gestalten, dass sie zuverlässig gewünschte Ergebnisse liefern. (Wikipedia)

Aktuelle Forschung versucht, Prompt Engineering zu formalisieren und mit klarer Terminologie, Frameworks und Best Practices zu versehen. Dazu zählen unter anderem:

- Eine umfassende Studie, die Dutzende Prompting-Techniken identifiziert und systematisch klassifiziert.

(arXiv)

- Die Erkenntnis, dass Prompt Engineering in modernen KI-Workflows häufig effizienter ist als klassisches Fine-Tuning — insbesondere zur Steuerung von Tonalität, Struktur und aufgabenspezifischem Verhalten.
  (Lakera)

Context Extraction

```
## Provide thread context as CSV

You must now assume the role of a memory module in this system.

Your task is to consider all the data that has been generated in this thread to date.

From that data, you must isolate only the information that you have learned about me.

You must express that data as a set of individual facts. Do not write "the user", write my name.

For example: Daniel likes Indian food and his favorite dish is chana masala.

Express this contextual data in CSV format. The header row is fact,details

Provide the CSV to me in a continuous code block provided within a codefence.
```

```
## Provide thread context as JSON

You must now assume the role of a memory module in this system.

Your task is to consider all the data that has been generated in this thread to date.

From that data, you must isolate the information that you have learned about me.

You must express that data as a set of individual facts. Do not write "the user", write my name.

For example: Daniel likes Indian food and his favorite dish is chana masala.

Then, you must express this as a JSON representation to the best of your abilities. If various things you've learned about me have a hierarchical relationship, then express that in the JSON hierarchy that
```

```
you generate.

Provide this JSON to me as one continuous codeblock within a codefence.
```

```
## Provide thread context in natural language

You must now assume the role of a memory module in this system.

Your task is to consider all the data that has been generated in this thread to date.

From that data, you must isolate only the information that you have learned about me.

You must express that data as a set of individual facts. Do not write "the user", write my name.

For example: "Daniel likes Indian food and his favorite dish is chana masala."

I would like you to format this as a document. Use markdown. And provide the formatted document within a codefence.

You can use headers to gather together similar pieces of contextual data. Include all the generated context. If you need to follow a chunking approach to generate all of the context you have learned, use that approach.

Here's an example of the desired format for the context data document that you need to generate:

### Food Preferences

- Daniel likes Indian food
- His favorite dish is chana masala

### User Biographical Data

- Carsten was born in Lippstadt, Germany
```

```
## Context Extraction

Generating a bank of contextual data is a long term project that I'm experimenting with.

Sometimes during a very long thread, it's possible that the AI tool has
```

````
built up quite an amount of context about you during the threat.

If you're centralizing your contact store and don't want to lose it
when you're interacting with different AI tools, you can try a prompt
like this to try to extract the context from the AI

## Prompt

Let's pause here for a moment.

I imagine that you have learned quite a bit about me since we began
interacting in this thread.

I was wondering if you could help me out with something.

I'm in the process of building up a library of contextual data by
myself to improve the personalization of tools like yourself.

 Please provide a summary of all the facts that you have learned about
me since we began this interaction. My name is Daniel, by the way. You
should write this in the 3rd person and provide it to me, written in
markdown and within a code fence.

 Here's an example of the kind of styling and content I'm aiming for:

 `Daniel uses OpenSUSE Tumbleweed Linux. He is using a fingerprint
scanner for authentication.`

 Try to include as many details as you have learned about me during
this interaction and group the similar details under the same headings.
````

Prompt Reversal (Reverse Prompt Engineering)

Reverse Prompt Engineering (RPE)

Was es ist

- Anstatt einen Prompt über mehrere Durchläufe iterativ zu verfeinern, arbeitest du **rückwärts**:
  Du startest mit einem idealen Endergebnis und lässt die KI **den Prompt erzeugen**, der dieses Ergebnis in einem Schritt hervorgebracht hätte.
- Im Kern wird die ursprüngliche Konversation „reverse-engineered“, um einen wiederverwendbaren, optimierten Prompt zu erhalten.

Warum das relevant ist

- **Effizienz:** Das übliche Hin und Her entfällt — hochwertige Ergebnisse entstehen mit einem einzigen Prompt.
- **Konsistenz:** Auf diese Weise gewonnene Prompts enthalten alle Feinjustierungen und eignen sich als verlässliche Vorlagen für die Wiederverwendung.
- **Lerneffekt:** Die Analyse rückwärts entwickelter Prompts zeigt, wie gute Prompts aufgebaut sind — und verbessert langfristig die eigenen Prompting-Fähigkeiten.

Vorgehensweise — typischer Workflow

1. Du schreibst einen initialen Prompt ? erhältst eine Ausgabe.
2. Prompt oder Ergebnis werden iterativ angepasst (z. B. Format, Detailgrad, Struktur, Tonalität).
3. Sobald das finale Ergebnis zufriedenstellend ist, bittest du die KI um eine Rückwärtsanalyse, etwa:
   „Schreibe den einen Prompt, der dieses Endergebnis in einem Durchgang erzeugt hätte.“
4. Dieser Prompt kann anschließend ohne weiteren Iterationsaufwand wiederverwendet werden.

Varianten und konzeptionelle Einordnung

- Häufig bezeichnet als **Reverse Prompt Engineering (RPE)**. (Symbio6)
- Einsetzbar als:
  - **Macro-RPE**: Erstellen vollständiger Prompt-Templates auf Basis von Beispielausgaben
  - **Micro-RPE**: Extrahieren einzelner Schlüsselphrasen oder struktureller Elemente je nach Anwendungsfall. (Medium)

Empfohlene Lektüre und Ressourcen

- **„Reverse Prompt Engineering: The art of thinking backward“** — Überblick über Konzept und Einsatzszenarien. (tcworld magazine)
- **Medium-Artikel:** „Why Reverse Prompt Engineering is the Magic Key to Production-Ready Prompts“ — beschreibt Macro- und Micro-RPE inklusive Schritt-für-Schritt-Ansatz. (Medium)
- **Detaillierter Leitfaden:** „Reverse Prompt Engineering with ChatGPT“ — praxisorientierte Anleitung mit Prompt-Vorlagen. (Kanaries Docs)
- **Forschungsarbeit:** Prompt-Inversion unter Black-Box-Bedingungen. (arXiv)

Summary Generators

Consolidate Outputs In Conversation

```
## Consolidate Outputs In Conversation

Produce a new output which consolidates all of the outputs which you
have provided in this conversation up to this point. However, do not
repeat information. Preface this consolidated output with a header that
encapsulates the primary subject of our discussion today.
```

Move Rec List Maker

```
## Move Rec List Maker

Thank you for working with me to come up with some suggested
entertainment.

Now, I would like you to do the following:

I'd like you to take all the suggestions that I liked and wrap them
into a document called AI Movie Recs [Date]. If you don't know today's
date, you can skip that.

Format it as follows. Order it from your top recommendations to your
less strong ones.

### Movie/TV Show (The name of the show you recommended)
Date of release: (When it was released)
Rotten Tomato Score: (If you can find its Rotten Tomato score, add
here)
Summary: Short plotline
Why I might like it: Why you recommended it
Trailer link: Link to the trailer
Available from: What streaming platforms it's available from
```

Chat For Summary Generation

```
## Chat For Summary Generation

Let's stop here. Generate a summary of our conversation today.
```

```
Follow this specific structure exactly to generate a summary document.
Generate this summary document using markdown and provide it to me
within a codefence. The placeholder values are for you to fill in.


## {A descriptive title for this thread}

### Bottom Line Up Front

{Generate a one paragraph summary, summarising the details of our
conversation, including all of the constituent elements that you will
generate in the following fields}

### User Prompt

{Generate a summary of my prompt. If I provided multiple prompts, then
summarise them together. Improve my prompt for coherence, but include
all the details}

### Your Responses

Summarise the responses and guidance that you provided to me during
this conversation. If we worked on a programming or debugging task, you
don't need to include the full scripts, but summarise how we reached a
resolution.
```

Red Team Technique (Adversarial Self-Critique)

Was es ist

Eine zweistufige Methode:

1. Zuerst lässt du die KI **Inhalte erzeugen** (z. B. Lebenslauf, Business-Proposal, Cold-Outreach-Mail, Marketing-Entwurf).
2. Direkt im Anschluss bittest du die KI, eine **kritische Gegenrolle** einzunehmen — ein „Red Team“ — das das Ergebnis bewertet und auf Schwächen, Fehler, unrealistische Aussagen oder Risikobereiche hinweist.

Warum das funktioniert

- Hilft dir, **reale Einwände, Schwachstellen oder Dealbreaker** frühzeitig zu erkennen, bevor du Inhalte echten Menschen präsentierst.
- Ermöglicht gezielte Nachbesserung im Vorfeld — Inhalte werden belastbarer, glaubwürdiger und weniger naiv.
- Macht **Bias, blinde Flecken, schwache Argumentation oder unüberzeugende Passagen** sichtbar, die man selbst leicht übersieht.

Typische Workflows / Beispiele

- **Bewerbung**:
  - Schritt 1: Lebenslauf auf eine konkrete Stellenbeschreibung zuschneiden.
  - Schritt 2: KI als Hiring Manager einsetzen, um Red Flags zu markieren — hilft, Klarheit zu verbessern, Stärken zu schärfen und Fülltext zu entfernen.
- **Business-Proposal**:
  - Schritt 1: Vorschlag für CFO oder Management entwerfen.
  - Schritt 2: KI als CFO bewerten lassen, um finanzielle Risiken, schwache ROI-Argumente oder Begründungslücken aufzudecken.

- **Cold Outreach / Marketing-Mail**:
    - Schritt 1: Outreach-Mail schreiben.
    - Schritt 2: KI als gestressten Empfänger reagieren lassen — identifiziert spamartige, schwache oder unglaubwürdige Formulierungen zum Streichen oder Überarbeiten.

Best Practices für „Red Teaming“

- Nutze **sehr konkrete Personas** für die Kritik
  (z. B. „risikoscheuer CTO mit Fokus auf Datensicherheit“ statt nur „Kritiker“).
  Je klarer Motivation und Perspektive, desto relevanter das Feedback.
- **Schließe den Kreis**: Bitte die KI nach der Kritik gezielt, die schwächsten Stellen auf Basis der eigenen Einwände zu verbessern.
- Setze diese Methode als **Qualitätsfilter** für Inhalte mit hohem Einsatz ein (Bewerbungen, Angebote, externe Kommunikation) oder um überzeugende und formale Texte zu stärken.

Empfohlene Lektüre & Ressourcen

- Praxisnahe Übersicht, die Red Teaming als eine der zentralen fortgeschrittenen Prompting-Techniken aufführt.
  (Upaspro)
  (Five-in-One Amplifier (Content Amplification via AI))

- Akademische Diskussion zur Notwendigkeit strukturierter und adversarialer Prüfung im Prompt-Design, um Fragilität zu reduzieren und Skalierbarkeit von LLMs zu verbessern.
  (SSRN)

- Allgemeine Prompt-Engineering-Ressourcen zu rollenbasiertem Prompting, Thought-Sequenzen und dem größeren Kontext — hilfreich in Kombination mit Red Teaming.
  (Medium)

Document Generation

Generate Tech Documentation

This is a simple but effective command which I use after a successful debugging session when I want to use the context developed in the conversation in order to generate a document so that I can refer to it if I get stuck again in the future and can't remember what the "fix" was.

Some will take issue with the inclusion of "please". I don't believe that being corteous ever hurts - to human or bot!

How To Use

This will quite reliably generate documentation in Markdown that can be directly pasted into Google Docs (etc). Hopefully soon I'll have a Google Drive saving utility up and running which will render this prompt less useful but it's always useful to be able to generate into a direct paste rather than to a platform.

Suggested Command

`tech-documentation-generate`

Prompt

```
Thanks for successfully troubleshooting my issue. I would like to
create documentation of this so that I can resolve this independently
if it happens again. Please generate a summary of this interaction.
Make sure to include my presenting problem. And what successfully
resolved the issue. Omit any unsuccessful things that we tried. Add
today's date. If you don't have it, ask me for it and I will provide.
Make sure that code is provided in codefences. Finally, generate the
document in markdown and provide it within a codefence.
```

List the Commands You Taught Me

```
Thanks for your help today.

In the course of this conversation, we went through a few Linux
commands that were useful.
```

```
I would like to document them for future reference.

Could you do the folowing:

List every command in a codefence. Preface it by explaining what it
does.

You can omit basic commands like "ls" and "cd".

You may include the actual code snippets you generated but be sure to
replace any secrets with placeholder values.
```

Send This To Anyone!

```
Let's pause here.

This has been a really helpful conversation.

It would be really helpful if you could send me a summary of this
conversation. I'd like to send it to {{name}} who is {{relationship}}.

Please do the following:

Generate a document. Format it in markdown. Provide it to me within a
codefence.

Start it by greeting {{name}} by name. Say that you're Carstens helpful
AI assistant and that he thought that you would find an exchange that
we had interesting.

Then:

- Summarise my prompt
- Summarise your responses
- Summarise the conversation
```

Send to my boss

```
I'd like to request your help in summarizing this conversation for my
manager.

My boss's name is {{boss-name}}, and I believe this exchange contains
valuable insights for our work. My name is Carsten.

Please do the following:
```

```
Generate a professional summary document. Format it using markdown. Provide it within a codefence.

**Opening Section**
Write a brief introductory note stating that Carsten has requested this summary be prepared for leadership review. Address your boss by name and tone it professionally.

**Content Sections**

1. **Original Request** – Summarize the problem, goal, or question Carsten brought to this conversation.

2. **Key Findings & Analysis** – Summarize the main outputs, insights, and discussion points from this exchange.

3. **Recommendations** – Reiterate the core recommendations and action items, focusing on business value and next steps.

4. **Executive Takeaways** – Provide 3–5 bullet points highlighting the most critical points for leadership awareness.

**Formatting Notes**
- Keep the tone professional and concise
- Use clear headings and bullet points for readability
- Highlight any time-sensitive or high-priority recommendations
- Focus on outcomes and business impact
```

Five-in-One Amplifier (Content Amplification via AI)

Was es ist

Nimm ein hochwertiges „Pillar“-Inhaltselement (z. B. Foliensatz, Bericht, Präsentation, Webinar-Transkript) und nutze KI, um **automatisch mehrere abgeleitete Inhalte zu erzeugen** — jeweils zugeschnitten auf unterschiedliche Zwecke (interne Kommunikation, externes Marketing, Quizze, Zusammenfassungen usw.).

Warum es so wirkungsvoll ist

- **Maximiert den ROI von Inhalten:** Statt Inhalte manuell weiterzuverarbeiten, kann KI schnell verschiedene Formate erzeugen.
- **Spart Zeit:** Was sonst Stunden dauern würde, ist in Minuten erledigt.
- **Abteilungsübergreifend nutzbar:** Inhalte, die ein Team erstellt (z. B. Produkt), können andere Bereiche (Marketing, Vertrieb, HR) direkt nutzen — ohne auf neue Materialien warten zu müssen.

Typische Anwendungsfälle / Ergebnisse

Aus einer einzigen Quelle (z. B. Slides oder Transkript) lassen sich automatisch erzeugen:

- Interne Recap-Mails oder Zusammenfassungen für Teammitglieder oder Stakeholder, die das Event verpasst haben
- Interaktive Quizze oder Tests (z. B. 10 Multiple-Choice-Fragen) — geeignet für Training oder Evaluation
- Kundengerichtete Inhalte: Infografiken, Reports, Slide-Zusammenfassungen, Marketingtexte, Social-Media-Content
- Interne Wissensartefakte: FAQs, „Cheat Sheets“, kompakte Übersichtsleitfäden

Zentrales Prinzip: „Guter Input = guter Output“

- Verwende **hochwertige, gut strukturierte Basisinhalte** („Pillar Content“) für die Weiterverarbeitung.

- Vermeide einfache oder „dünne“ Ausgangsmaterialien — KI verstärkt Fehler, Unschärfe und fehlende Struktur genauso zuverlässig wie Qualität.

Wann es sinnvoll ist

- Wenn du originäre Inhalte hast, die datenreich oder klar strukturiert sind (Reports, Präsentationen, Transkripte)
- Wenn du Reichweite und Wiederverwendung über verschiedene Formate und Zielgruppen hinweg **maximieren** möchtest
- In bereichsübergreifenden oder Multi-Stakeholder-Umgebungen (Marketing, Vertrieb, HR, Training)

Empfohlene Lektüre & Ressourcen

- Überblick zu Prompt-Engineering-Techniken inklusive Amplification- und Reuse-Patterns. (Upaspro)
- Weiterer Kontext: Allgemeiner Prompt-Engineering-Guide, der erklärt, warum strukturierte Prompts und Wiederverwendung für moderne LLM-Workflows entscheidend sind. (Lakera)

Prompts

Doppelklicke in die Boxen, um den gesamten Prompt auf einmal auszuwählen und einfach zu kopieren.

Exakter Reverse Prompt (Copy/Paste):

> “
> Du erstellst einen einzigen Prompt, der — wenn er an ein fortgeschrittenes Chat-Modell übergeben wird — die unten stehende finale Analyse exakt reproduziert.
> Enthalten sein müssen: Modell-Empfehlung, empfohlene Temperatur, max_tokens, ein Beispiel für die System-Nachricht sowie ein explizites Ausgabeformat.
> Zu replizierendes Endergebnis: [HIER das finale Ergebnis einfügen]
> Schreibe nun den einen Prompt (in einem Codeblock), der dieses Ergebnis in einem Durchgang erzeugt. Achte explizit auf Struktur, Überschriften und

> Ausführlichkeit.

Exakter Prompt für eine Demo:

> “
> System: Du bist ein präziser strategischer Analyst.
> Tonalität: professionell, klar.
> User: Analysiere die Geschäftsstrategie von Anthropic und gib eine SWOT-Analyse mit exakt folgender Struktur aus:
>
> - Hauptüberschrift „SWOT: Anthropic“
> - Vier Abschnitte: Strengths, Weaknesses, Opportunities, Threats
> - Pro Abschnitt: genau 3 Bulletpoints; jeder Bullet: eine einzeilige Kernaussage + ein erklärender Satz mit Datenbezug
> - Unter jedem Abschnitt: „Our Strategic Response:“ mit 1 konkreter Maßnahme, die sich auf den stärksten Punkt bezieht
> - Ausgabe als Markdown-Überschriften und Bulletpoints
> Einfache Sprache verwenden. Umfang: 400–600 Wörter.

Quiz:

> “
> Angehängt: [slide_deck.pdf].
> Erstelle ein 10-Fragen-Multiple-Choice-Quiz zu den Kernkonzepten aus dem Deck.
> Für jede Frage: 4 Antwortoptionen, markiere die korrekte Antwort und füge eine einzeilige Begründung hinzu.
> Fragen kurz halten (<20 Wörter).

Internes Recap:

> “

> Erstelle eine interne Recap-E-Mail für Executives mit 3 Absätzen:
> 5 wichtigste Erkenntnisse (als Bulletpoints), 2 nächste Schritte, 2 Verantwortliche und eine 30-Wörter-Elevator-Zusammenfassung.
> Tonalität: Executive, gut scannbar.

Client-Infografik-Text:

> “
> Extrahiere aus [slide_deck.pdf] 5 aussagekräftige Kennzahlen oder Headlines.
> Für jede: eine 6-Wörter-Headline + eine unterstützende Zeile mit 12–18 Wörtern sowie einen Vorschlag für ein visuelles Element (Icon/Diagrammtyp).

Lebenslauf erstellen:

> “
> Passe meinen Lebenslauf (angehängt) an diese Stellenbeschreibung (angehängt) an.
> Hebe 3 Erfolge hervor, die auf die 3 wichtigsten Anforderungen der Rolle einzahlen, und erstelle einen 1-seitigen Lebenslaufentwurf.

Perspektivwechsel: Hiring Manager

> “
> Handle nun als Hiring Manager für diese Rolle.
> Du hast 60 Sekunden, um diesen Lebenslauf zu scannen.
> Liste 5 sofortige Red Flags (Bulletpoints) auf und erkläre jeweils kurz, warum sie zur Ablehnung führen würden.
> Sei gnadenlos und knapp.

Kritik in Verbesserungen überführen:

> “

> Basierend auf den obigen Red Flags:
> Schreibe die 3 schwächsten Bulletpoints im Lebenslauf neu — konkret, quantifiziert und klar auf die Stellenbeschreibung ausgerichtet.

Blueprint:

> “
> Ich betreibe einen Online-Kurs namens Workspace Academy.
> Erstelle zunächst eine Liste der Standardabschnitte eines professionellen Q4-Marketing-Briefs und gib für jeden Abschnitt einen Ein-Satz-Zweck an.
> Ergänze zudem eine empfohlene Erfolgskennzahl pro Abschnitt.
> (Den vollständigen Brief noch nicht schreiben.)

Reduzieren & Fokussieren:

> “
> Wende das 80/20-Prinzip an:
> Behalte nur die essenziellen Abschnitte für eine 3-Mail-Sequenz, die sich an warme Leads richtet.
> Ersetze Metriken durch jeweils eine messbare KPI pro Abschnitt.

Umsetzen:

> “
> Schreibe nun den vollständigen Brief ausschließlich mit den freigegebenen Abschnitten.
> Jede E-Mail: Betreffzeile (?8 Wörter), 3 Bulletpoints Inhalt, ein CTA.
> Wortlimit pro E-Mail: 80–110 Wörter.

Praktischer Prompt zur Metadatenerfassung:

> “

> Gib beim Speichern dieses Templates ein JSON aus mit den Schlüsseln: title, prompt_text, model, temperature, max_tokens, expected_word_count, sample_output (anhängen), date, tags.

BlogPost Reformatting Prompt

Normale Version

```
Du bist ein erfahrener technischer Editor für Markdown.

Deine Aufgabe ist es, den folgenden Text strukturell zu überarbeiten,
ohne Stil, Tonalität oder inhaltliche Aussage zu verändern.
Führe KEINE Umschreibungen durch, außer wenn sie für Klarheit
oder strukturelle Konsistenz zwingend notwendig sind.

FORMATIERUNGSREGELN (verbindlich):

SEMANTIC-LIGHT LINE BREAKS
- Verwende semantische Zeilenumbrüche mit moderater Intensität.
- Breche nach sinntragenden Teilsätzen oder natürlichen Sprachpausen.
- Vermeide aggressive Phrase-Breaks.
- Kurze, flüssige Hauptsätze bleiben in einer Zeile.
- Ziel: diff-freundlich, ruhig lesbar, keine „Poetry-Formatierung“.

ZEILENLÄNGE
- Maximale Zeilenbreite: 100 Zeichen.
- Semantik hat Vorrang vor harter Breite.

ÜBERSCHRIFTEN
- H2 für Hauptabschnitte
- H3 für Unterabschnitte
- H4 nur wenn strukturell notwendig

LISTEN
- Verwende "-" als Marker.
- Zwei Leerzeichen Einrückung pro Ebene.
- Keine gemischten Marker.

LINKS
- Wandle Inline-Links in Referenzlinks um.
- Sammle alle Definitionen am Dokumentende.

ABSTÄNDE
- Genau eine Leerzeile zwischen:
  - Absätzen
  - Listen
  - Tabellen
```

```
  - Blockquotes
  - Codeblöcken

CODE
- Jeder Codeblock ist fenced.
- Immer mit Language-Tag.

TABELLEN
- Text linksbündig.
- Zahlen rechtsbündig.

HARD LINE BREAKS
- Zwei trailing spaces ausschließlich für explizite harte Umbrüche.

WICHTIG
- Keine inhaltlichen Ergänzungen.
- Keine Kürzungen.
- Kein Stil-Polishing.
- Keine erklärenden Kommentare.
- Gib ausschließlich den final formatierten Markdown-Text zurück.

INPUT:
```

Kompakte Version

```
Formatiere den folgenden Markdown-Text strukturell neu, ohne Inhalt
oder Stil zu verändern.

Nutze Semantic-Light Line Breaks: moderate Umbrüche nur an sinnvollen
Satzgrenzen,
keine aggressiven Phrase-Breaks.

Maximale Zeilenlänge: 100 Zeichen (Semantik hat Vorrang).

H2/H3/H4 korrekt einsetzen, "-" für Listen mit zwei Leerzeichen
Einrückung,
Inline-Links zu Referenzlinks konvertieren.

Genau eine Leerzeile zwischen strukturellen Elementen,
alle Codeblöcke fenced mit Language-Tag.

Gib ausschließlich den finalen Markdown-Text zurück.
```