

# Drupal KI Lösungen: Textoptimierung direkt im CMS mit AI Content Suggestions-Modul

## Inhaltsverzeichnis

- Was das Modul leistet
  - 2. Stilcheck
  - 3. Zusammenfassung
  - 4. Verschlagwortung
  - 5. Tonalitätsanpassung
  - 6. SEO-Optimierungen
  - 7. Einfache Sprache
  - 8. Textkürzung & -verlängerung
  - 9. Grammatik- und Satzbaupverbesserung
  - 10. Zielgruppen-Anpassung
  - 11. Strukturierung des Textes
  - 12. Konsistenz- und Kohärenz-Check
  - 13. Marken- & Stilrichtlinien anwenden
  - 14. Social-Media-Formate
  - 15. Mehrsprachigkeit
  - 16. Hinweise zu unsicheren Fakten (kein echtes Fact-Checking)
- Voraussetzungen und Budget-Kontrolle
- Einschränkungen des Moduls
- Wie die Arbeit mit dem Modul abläuft
- Typische Einsatzszenarien mit Testaufbau
  - Beispiel: Stil- und Rechtschreibprüfung
  - Beispiel: Zusammenfassung erzeugen für Teaser-Feld

- Beispiel: Text für eine bestimmte Zielgruppe optimieren
- Beispiel: Überarbeite den Text aus fachlicher Sicht.
- Beispiel: Kurzfassungen für Meta-Descriptions oder Social-Posts
- Beispiel: Vorschläge für Taxonomie-Begriffe
- Unterschiede zu Joomla
- Mit eigenem Modul eine Kopie des Plugins "Summarise text" erzeugen
- Optimierung der Texte direkt im Feld mit dem Modul Field Widget Actions
- Was kostet der Spaß?
- Weiterführende Links

Redaktionelle Teams stehen oft vor der Aufgabe, Texte korrekt, verständlich, suchmaschinenfreundlich und stilistisch einheitlich aufzubereiten. Künstliche Intelligenz ist dabei eine große Hilfe und Drupal unterstützt das mit vielen Modulen. Häufig braucht es dafür mehrere Tools und viele Arbeitsschritte. Das Submodul AI Content Suggestions setzt an dieser Stelle an: Es ergänzt das Node-Formular um KI-gestützte Werkzeuge, die direkt während des Schreibens nutzbar sind.

## Was das Modul leistet

**AI Content Suggestions** erweitert das Bearbeitungsformular eines Nodes um zusätzliche Bereiche, über die sich Textfelder mithilfe eines LLM (Large Language Model) analysieren lassen.

### 1. Rechtschreibprüfung

- Tippfehler erkennen
- Rechtschreibregeln umsetzen
- Zeichensetzung verbessern
- Grammatikfehler finden

### 2. Stilcheck

- Sprachstil glätten
- Lesbarkeit verbessern
- überlange Sätze identifizieren
- Wortwiederholungen kennzeichnen

- Füllworte vermeiden

### **3. Zusammenfassung**

- Kurzfassung
- Meta-Description vorschlagen
- Social-Media-Teaser erzeugen

### **4. Verschlagwortung**

- Schlagwörter extrahieren
- Themenbereiche erkennen
- Taxonomie-Vorschläge

### **5. Tonalitätsanpassung**

- sachlich
- werblich
- freundlich
- formal
- locker

### **6. SEO-Optimierungen**

- SEO-Titelvorschläge
- Meta-Description optimieren
- Keywords und verwandte Begriffe
- Frageformate für Featured Snippets

### **7. Einfache Sprache**

- leichte Sprache
- kurze, klare Sätze
- Übersetzung von Fremdworten

## 8. Textkürzung & -verlängerung

- stark kürzen
- prägnant zusammenfassen
- ausführlich erweitern
- Beispiele ergänzen

## 9. Grammatik- und Satzbauverbesserung

- Grammatikfehler anzeigen
- Satzbau straffen
- Wiederholungen reduzieren
- unnötige Füllwörter entfernen
- trefferendere Wortwahl

## 10. Zielgruppen-Anpassung

- Kinder
- Fachpublikum
- Anfänger:innen
- Entscheider:innen

## 11. Strukturierung des Textes

- Zwischenüberschriften erzeugen
- Abschnitte gliedern
- Bulletpoints statt Fließtext
- Schritt-für-Schritt-Struktur

## 12. Konsistenz- und Kohärenz-Check

- Logik prüfen
- Argumentationslinien verbessern
- Unklarheiten markieren

## 13. Marken- & Stilrichtlinien anwenden

- Tonalität an Corporate Language anpassen
- interne Schreibregeln berücksichtigen
- konsistente Begriffe vorschlagen

## 14. Social-Media-Formate

- Komplette Posts für Facebook & Co
- kurze Snippets
- Zusammenfassungen für OG-Meta-Tags

## 15. Mehrsprachigkeit

- Übersetzung
- zweisprachige Ausgabe

## 16. Hinweise zu unsicheren Fakten (*kein echtes Fact-Checking*)

- potenziell prüfbedürftige Aussagen markieren
- fehlende Quellenhinweise erkennen

Die Inhalte im Node bleiben dabei unverändert. Vorschläge werden angezeigt und können gezielt übernommen werden.

## Voraussetzungen und Budget-Kontrolle

Mehr zur Installation und Konfiguration von AI Core, Einrichtung eines Provider-Moduls und Budgetüberwachung gibt es beim vorherigen Blogbeitrag

<https://www.montviso.de/blog/kuenstliche-intelligenz-drupal-textgenerie...>

Die technischen Grundlagen entsprechen denen des gesamten Drupal-AI-Ökosystems.

Notwendig sind:

- AI Core
- ein aktives Provider-Modul wie OpenAI und API-Key
- und das aktivierte Submodul AI Content Suggestions

Wie beim Automators-Modul spielt auch hier die Kostenkontrolle eine wichtige Rolle. Empfehlenswert ist der Einsatz von Modellen mit moderatem Tokenpreis (z. B. GPT-4o-mini). Durch die Auswahl weniger, klar definierter Plugins lässt sich die Anzahl der KI-Abfragen ebenfalls gut steuern.

## Einschränkungen des Moduls

- Die Tools funktionieren ausschließlich für Nodes.
- Die Anzeige erscheint erst nach dem ersten Speichern eines neuen Nodes.
- Inhalte werden nie automatisch überschrieben, sondern müssen händisch in ein Zielfeld geschrieben werden.
- Je nach Modell kommt es zu kurzen oder auch längeren Wartezeiten.
- Die Auswahl an Plugins ist eingeschränkt, weil man z.B. nur ein Summerise Text Plugin hat und dort nur einen Prompt hinterlegen kann. Es ist aber einfach mittels Kopie des Submodules ein eigenen Modul zu erzeugen, wo man nur den Namen ändert und dann einen anderen Prompt einpflegen kann.

## Wie die Arbeit mit dem Modul abläuft

Nach der Aktivierung erscheinen im Node-Formular zusätzliche Dropdowns. Der Ablauf ist immer gleich:

- Plugin auswählen (z. B. Tonalität, Kurzfassung).
- Das zu prüfende Feld wählen.
- Button anklicken und kurz abwarten.
- KI-Vorschläge manuell ins Ziel-Feld übernehmen.

Die Werkzeuge beziehen sich immer auf einzelne Felder. Dadurch lassen sich gezielt der Body, der Teaser oder weitere Textfelder prüfen, ohne den gesamten Node analysieren zu müssen.

## Typische Einsatzszenarien mit Testaufbau

Beim ersten Beispiel gehe ich sehr detailliert vor, die übrigen werden nur noch stichpunktartig beschrieben.

## Beispiel: Stil- und Rechtschreibprüfung

*Wir verwenden den vorhandenen Contenttyp: „Article“ und das Feld: Body und optimieren den darin enthaltenen Text.*

### Ausgangspunkt: vorhandener Inhaltstyp „Article“

Wir legen einen Inhaltstyp Artikel mit Maschinennamen „article“ an. Er hat die Felder

- **Title**
- **Body** (Text (formatted, long, with summary))  
Das ist das Feld, mit dem Text, den die KI prüfen und stilistisch glätten soll und das im AI Content Suggestions-Dropdown ausgewählt wird.
- **Feld für den optimierten Text (AI)**

Wir legen ein weiteres Feld an, in das später der überarbeitete Text eingefügt wird. Das Feld wird so konfiguriert:

- **Feldtyp:** Text (formatted, long, with summary)
- **Bezeichnung:** z.B. „Optimierter Text (AI)“
- **Maschinenname:** field\_ai\_body\_optimized
- **Textformat:** identisch zu dem von Body in unserem Fall Content
- **Formulareinstellungen:** Dieses Feld schieben wir unter **Body**.
- **Ansicht:** Dieses Feld soll im Frontend sichtbar sein, Body nicht.

Für Redaktionen ist es hilfreich zu sehen, ob ein Artikel bereits einmal durch die KI-Prüfung gegangen ist. Dafür legen wir noch ein weiteres Feld anlegen, namens Status. Uns reicht ein Boolean (Ja/Nein) namens „AI-Stil- und Rechtschreibprüfung durchgeführt“. Man könnte auch eine Taxonomie mit den verschiedenen Stati referenzieren.

- **Feldtyp:** Boolean (Ja/Nein)
- **Bezeichnung:** „AI-Stil- und Rechtschreibprüfung durchgeführt“
- **Maschinenname:** field\_ai\_reviewed
- **Standardwert:** „Nein“
- **Formulareinstellungen:** Dieses Feld schieben ans Ende des Formulars.
- **Ansicht:** Dieses Feld soll im Frontend ausgeblendet sein.

Später lässt sich darüber eine View bauen (z.B. „Artikel ohne AI-Prüfung“), welche die Verwaltung vereinfacht.

| Field                                           | Machine name            | Widget                                                 |
|-------------------------------------------------|-------------------------|--------------------------------------------------------|
| ÷ Title                                         | title                   | <div>Textfield</div> Textfield size: 60                |
| ÷ Body                                          | field_body              | <div>Text area (multiple rows)</div> Number of rows: 5 |
| ÷ Optimierter Text (AI)                         | field_ai_body_optimized | <div>Text area (multiple rows)</div> Number of rows: 5 |
| ÷ AI-Stil- und Rechtschreibprüfung durchgeführt | field_ai_reviewed       | <div>Single on/off checkbox</div> Use field label: Yes |

# Konfiguration von AI Content Suggestions

## Submodul aktivieren

Filter

Suggestions

^ AI

☒ AI Content Suggestions

Allows a configured AI to be passed content to suggest alterations

## Plugins aktivieren

Unter /admin/config/ai/suggestions eine stehen verschiedene Plugins zur Auswahl.

Below is a list of the available plugins you can use to analyze your content.

#### Suggest title

☒ Enable Suggest title.

Allow an LLM to suggest an SEO-friendly title for the content.

#### Moderate text

☒ Enable Moderate text.

Allow an LLM to provide moderation suggestions about the content.

#### Summarise text

☒ Enable Summarise text.

Allow an LLM to provide a summary of the content.

#### Alter tone

☒ Enable Alter tone.

Allow an LLM to provide tone suggestions about the content.

#### Suggest taxonomy tags

☒ Enable Suggest taxonomy tags.

Allow an LLM to suggest SEO-friendly taxonomy tags for the content.

#### Evaluate Readability

☒ Enable Evaluate Readability.

In Frage kommt für unseren Zweck „Enable summarise text“. Es stellt die LLM zur Verfügung, die wir auch bei ChatGPT kennen.

### Summarise text



Enable Summarise text.

### Summarise text model.

OpenAI - gpt-4.1-mini

-- Default from AI provider --  
OpenAI - gpt-3.5-turbo  
OpenAI - gpt-3.5-turbo-0125  
OpenAI - gpt-3.5-turbo-1106  
OpenAI - gpt-3.5-turbo-16k  
OpenAI - gpt-3.5-turbo-instruct  
OpenAI - gpt-3.5-turbo-instruct-0914  
OpenAI - gpt-4  
OpenAI - gpt-4-0125-preview  
OpenAI - gpt-4-0613  
OpenAI - gpt-4-1106-preview  
OpenAI - gpt-4-turbo  
OpenAI - gpt-4-turbo-2024-04-09  
OpenAI - gpt-4-turbo-preview  
OpenAI - gpt-4.1  
OpenAI - gpt-4.1-2025-04-14  
OpenAI - gpt-4.1-mini

Für den **besten Kompromiss aus Qualität UND Budget** gibt es eine sehr klare Empfehlung –OpenAI - gpt-4o-mini.

Die Einschätzung basierend auf:

- Preis pro 1.000 Tokens
- Qualität bei Stil-, Rechtschreib- und Tonalitätsoptimierung
- Geschwindigkeit
- Stabilität

### **Prompt** einpflegen

z.B. so:

„Du erhältst einen deutschen Text, der stilistisch überarbeitet werden soll.

Bitte überprüfe den Text auf:

- Rechtschreibung
- Grammatik
- Satzbau
- Lesbarkeit
- Stil
- sprachliche Präzision
- Tonalität
- Struktur

Die Tonalität soll der Schreibweise auf

<https://www.montviso.de/blog/kuenstliche-intelligenz-drupal-textgenerie...>

entsprechen:

klar, ruhig, sachlich, erklärend, strukturiert, ohne Übertreibungen, ohne künstliche Werbesprache.

Der Text soll professionell wirken, ohne steif zu sein, und soll in normalem, natürlichem Deutsch formuliert sein.

WICHTIG:

- Die Ansprache (Du oder Sie) darf NICHT verändert werden.
- Vorhandene Formatierung (Absätze, Listen, Überschriften) soll erhalten bleiben
- Füge sinnvolle Absatzüberschriften in <H2>-überschriften dazu bzw. wandle vorhandene in <H2>-überschriften um,
- Der Text darf nicht wesentlich länger oder kürzer werden.
- Keine Icons oder Sonderzeichen hinzufügen.
- Keine neuen Informationen erfinden.
- Keine inhaltlichen Aussagen verändern.
- Keine Erklärungen, Begründungen oder zusätzliche Hinweise ausgeben.

Gib ausschließlich eine überarbeitete Version des eingereichten Textes zurück.“

Falls du den Prompt später erweitern willst, sind dies typische Punkte, die die Qualität noch weiter erhöhen können:

## Umgang mit Fachbegriffen

- „Fachbegriffe beibehalten, aber klar formulieren.“
- „Falls ein Begriff sehr komplex ist, kurz und sachlich erläutern – ohne Beispiele.“

## **Umgang mit Zitaten oder Code**

- „Zitate unverändert lassen.“
- „Codeblöcke oder technische Syntax unangetastet lassen.“

## **Umgang mit Stilvarianten**

- „Falls der Text bereits gut formuliert ist, nur minimale Eingriffe vornehmen.“
- „Überarbeitungen sollen dezent und unaufdringlich sein.“

## **Umgang mit Struktur**

- „Struktur nicht verändern.“
- „Absätze nicht zusammenziehen oder unnötig auftrennen.“

## **Umgang mit Zeichenlimits, für den Falls, dass du Texte für Social-Media Meta Tags erzeugst:**

- „Zeichenlimit X nicht überschreiten.“

## **Umgang mit Aufzählungen**

- „Listen beibehalten, aber sprachlich glätten.“

## **Umgang mit technischen Begriffen wie AI, Drupal, Module**

- „Technische Begriffe korrekt schreiben und in stabiler Nomenklatur belassen.“

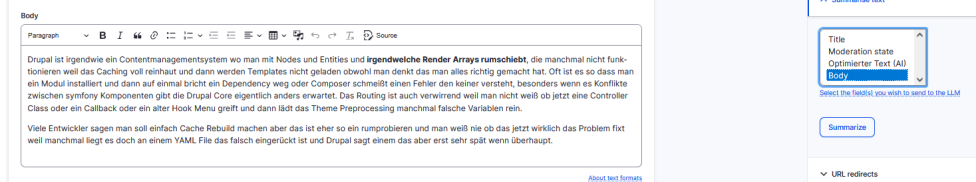
## **Umgang mit formaler Konsistenz**

- „Anführungszeichen einheitlich in deutschem Stil setzen.“
- „Keine englischen Anführungszeichen verwenden.“

## **Systemprompt**

Das ist der Block ganz am Ende auf der Seite mit der Auswahl von Content Suggestions Plugins. Dieses Feld darf nicht verändert werden.





- **Feld wählen**, das optimiert werden soll: In unserem Fall „Body“
- **Button „Summarise“ klicken** → warten → Vorschlag ablesen, falls er noch nicht den Erwartungen entspricht → Prompt optimieren, bis es passt.
- **Vorschlag gefällt**: Text in „Optimierter Text (AI)“ einfügen
- **Status Ändern**: Häkchen setzen bei „AI-Stil- und Rechtschreibprüfung durchgeführt“ .

Body

Paragraph

Drupal ist irgendwie ein Contentmanagementsystem wo man mit Nodes und Entities und **irgendwelche Render Arrays rumschiebt**, die manchmal nicht funktionieren weil das Caching voll reinhaut und dann werden Templates nicht geladen obwohl man denkt das man alles richtig gemacht hat. Oft ist es so dass man ein Modul installiert und dann auf einmal bricht ein Dependency weg oder Composer schmeißt einen Fehler den keiner versteht, besonders wenn es Konflikte zwischen symfony Komponenten gibt die Drupal Core eigentlich anders erwartet. Das Routing ist auch verwirrend weil man nicht weiß ob jetzt eine Controller Class oder ein Callback oder ein alter Hook Menu greift und dann lädt das Theme Preprocessing manchmal falsche Variablen rein.

Viele Entwickler sagen man soll einfach Cache Rebuild machen aber das ist eher so ein rumprobieren und man weiß nie ob das jetzt wirklich das Problem fixt weil manchmal liegt es doch an einem YAML File das falsch eingerückt ist und Drupal sagt einem das aber erst sehr spät wenn überhaupt.

[About text formats](#)

Optimierter Text (AI)

Heading 2

## Einführung in Drupal und seine Herausforderungen

Drupal ist ein Content-Management-System, das auf Nodes, Entities und Render-Arrays basiert, mit denen man arbeitet. Dabei kann es vorkommen, dass Render-Arrays nicht richtig funktionieren, weil das Caching dazwischenfunkt. In solchen Fällen werden Templates nicht geladen, obwohl man glaubt, alles korrekt eingerichtet zu haben.

### Probleme bei Modulen und Abhängigkeiten

Oft passiert es, dass nach der Installation eines Moduls plötzlich eine Abhängigkeit fehlt oder Composer einen Fehler ausgibt, der kaum verständlich ist. Besonders schwierig wird es bei Konflikten zwischen Symfony-Komponenten, die der Drupal-Core anders erwartet.

### Routing und Theme Preprocessing

Das Routing kann verwirrend sein, da unklar bleibt, ob eine Controller-Klasse, ein Callback oder ein älterer Hook-Menu greift. Zudem lädt das Theme Preprocessing manchmal falsche Variablen, was die Fehlersuche erschwert.

### Umgang mit Caching-Problemen

Viele Entwickler empfehlen, einfach einen Cache-Rebuild durchzuführen. Das ist allerdings eher eine Methode des Ausprobierens, da man selten sicher sein kann, ob das Problem wirklich dadurch gelöst wird. Manchmal liegt die Ursache auch an falsch eingerückten YAML-Dateien, auf die Drupal erst spät oder gar nicht hinweist.

[About text formats](#)

☒ AI-Stil- und Rechtschreibprüfung durchgeführt

## Beispiel: Zusammenfassung erzeugen für Teaser-Feld

Wir stehen vor dem Problem, dass wir das Feld „Summarise Text“ bereits für unseren umfangreichen Optimierungs-Prompt genutzt haben.

Leider kann man ein Plugin nicht mehrfach mit unterschiedlichem Prompt verwenden. Also lassen wir uns einen Workaround einfallen, oder schreiben ein kleines Modul (siehe weiter unten). Hier zwei Methoden als Workaround:

### Methode 1:

Wir ergänzen den Prompt bei Plugin „Summarise Text“ am Ende um die Anweisung:

„Hänge nun noch eine Zusammenfassung des optimierten Textes an, die nie mehr als 400 Zeichen hat.

Bitte keine längeren Texte.

Für die Zusammenfassung keine Überschriften, keine Formatierung, nur reiner Fließtext.

Trenne die Zusammenfassung vom vorher erzeugten optimierten Text durch Linie aus ----- in einer neuen Zeile. “

Das erzeugt den optimierten Text und nach der Trennlinie die Zusammenfassung, die wir nun in ein dafür vorgesehenes Teaserfeld kopieren können.

### Methode 2:

Wir aktivieren unter /admin/config/ai/suggestions auch noch das Plugin Suggest Title.

Das macht natürlich nur Sinn, wenn wir dieses Plugin nicht tatsächlich für den dafür vorgesehenen Einsatzzweck benötigen. Hier stehen die gleichen LLM zur Verfügung, wie bei Summarise Text. Dort hinterlegen wir einen einfachen Prompt.

**Suggest title**

☒ Enable Suggest title.

Suggest title model.

OpenAI - gpt-4.1-mini

**Suggest Title prompt \***

Erstelle eine Zusammenfassung des optimierten Textes, die nie mehr als 400 Zeichen hat.  
Bitte keine längeren Texte.  
Für die Zusammenfassung keine Überschriften, keine Formatierung, nur reinen Fließtext.

Allow an LLM to suggest an SEO-friendly title for the content.

Jetzt haben wir eine neue Kategorie, wenn wir unseren Inhalt editieren.

^ Suggest title

Title

Moderation state

**Optimierter Text (AI)**

Body

Select the field(s) you wish to send to the LLM

Suggest title

Wir wählen für die Zusammenfassung das Feld mit dem bereits optimierten Text.  
Die so erzeugte Zusammenfassung schreiben wir in ein Feld „Teaser“, das wir als reines Textfeld eingestellt haben.

## Bitte beachten!

Wir haben nur den Stil des ursprünglichen Nonsense Textes verändert und darüber eine Zusammenfassung erzeugt.

Der Text enthält Falschaussagen, die wir so nicht aus der Welt schaffen. Da braucht es also immer noch fachlich befähigte Redakteur\*innen, die den Text auf Faktengenauigkeit checken. Sicher wäre der Text um Längen besser, wenn wir nur einige Stichworte vorgegeben hätten und die KI den Text hätten erzeugen lassen, so wie es hier beschrieben ist:

<https://www.montviso.de/blog/kuenstliche-intelligenz-drupal-textgenerierung-aus-stichworten-mit-ai-automators-modul>

## Beispiel: Text für eine bestimmte Zielgruppe optimieren

Wir machen uns einen Spaß daraus, den bereits optimierten Text für Fünfjährige anpassen zu lassen.

Dazu aktivieren wir das Plugin „Alter Tone“ und wählen wieder ein geeignetes LLM. Danach haben wir folgende Auswahlmöglichkeiten beim Editieren unseres Contents:



# Beispiel: Überarbeite den Text aus fachlicher Sicht.

Hier testen wir, inwieweit die KI auch in der Lage ist, fachliche Schwächen zu finden. Selbstverständlich erfordert das immer noch Wissen, Recherche, Kontrolle und Nacharbeit in der Firma.

Aber testen wir doch einfach mal, was dieser Prompt bringt:

Optimize text prompt \*

Agiere als Drupal-Spezialist und optimiere den Text für ein Publikum mit Grundkenntnissen in IT aber wenig Kenntnissen in Drupal. Analysiere den folgenden Text auf fachliche Schwächen, Ungenauigkeiten und problematische Aussagen. Arbeite strikt nach diesen Schritten:

1. Identifiziere alle Stellen, die fachlich ungenau, unvollständig, missverständlich oder potenziell falsch sind.
2. Erkläre zu jeder Stelle, warum sie problematisch ist (kurz und präzise, keine Allgemeinplätze).
3. Gib zu jeder Stelle einen konkreten Verbesserungsvorschlag oder eine korrigierte Fassung.
4. Wenn etwas fachlich korrekt und solide ist, erwähne es nicht – konzentriere dich NUR auf Schwächen.
5. Erfinde nichts hinzu: Wenn der Text etwas nicht sagt, ergänze es nicht selbstständig, sondern weise nur auf fehlende Informationen hin.
6. Erkläre Fachbegriffe

Formatiere die Ausgabe exakt so:

\*\*\* Problemstelle:

Original:  
„...Zitat aus dem Text...“

Warum problematisch:  
...deine fachliche Begründung...

Vorschlag:  
...korrigierte oder verbesserte Version...

Wende diese Regeln strikt an.  
Gib ausschließlich die Analyse aus, keine Einleitung, keine Zusammenfassung, keine Erklärungen außerhalb dieses Schemas.

Das ergibt (Zeilenumbrüche und Fettdruck händisch eingefügt):

Nodes (Inhalte wie Artikel oder Seiten) gehören. Für die Darstellung der Inhalte nutzt Drupal Render-Arrays, strukturierte Daten, die später in HTML umgewandelt werden.“

**Problemstelle:** Original: „Diese Bausteine funktionieren nicht immer wie erwartet, da das Caching häufig für unerwartetes Verhalten sorgt und Templates trotz korrekter Konfiguration nicht geladen werden.“

**Warum problematisch:** Die Aussage ist zu allgemein und unpräzise. Das Caching beeinflusst vor allem die Performance und kann dazu führen, dass Änderungen nicht sofort sichtbar sind, aber es „funktioniert nicht wie erwartet“ ist unspezifisch. Dass Templates trotz korrekter Konfiguration nicht geladen werden, wird ohne Erklärung genannt – der Grund kann vielfältig sein, nicht nur Cache. Vorschlag: „Manchmal kann das Caching dazu führen, dass Änderungen nicht sofort sichtbar sind, was zu Verwirrung führt. Auch können Templates aus verschiedenen Gründen, etwa fehlerhafte Pfadangaben oder Prioritäten, trotz scheinbar korrekter Konfiguration nicht geladen werden.“

**Problemstelle:** Original: „Beim Installieren neuer Module können Abhängigkeiten unerwartet abbrechen, was zu schwer verständlichen Fehlern führt, insbesondere bei Konflikten zwischen Symfony-Komponenten. Diese werden vom Drupal Core meist anders behandelt, was die Fehlerbehebung zusätzlich erschwert.“

**Warum problematisch:** Die Aussage ist unklar und irreführend. „Abhängigkeiten abbrechen“ ist keine übliche Formulierung. Konflikte zwischen Symfony-Komponenten im Drupal-Module-Setup sind eher selten; Symfony ist Teil des Drupal-Core-Frameworks und wird vom Core kontrolliert. Zudem wird nicht erläutert, was genau „anders behandelt“ bedeutet. Vorschlag: „Beim Installieren neuer Module können Abhängigkeitsprobleme auftreten, die schwer verständliche Fehlermeldungen verursachen. Konflikte zwischen Bibliotheken, insbesondere bei externen Symfony-Komponenten, sind selten, da Drupal Core diese normalerweise zentral verwaltet. Dennoch kann dies die Fehlersuche erschweren.“

**Problemstelle:** Original: „Das Routing in Drupal ist komplex und unübersichtlich. Es ist oft unklar, ob eine Controller-Klasse, ein Callback oder ein veralteter Hook zum Einsatz kommt.“

**Warum problematisch:** Die Begriffe „Controller-Klasse“, „Callback“ und „Hook“ werden genannt, aber nicht erklärt. Außerdem können „Callbacks“ und „Hooks“ unterschiedliche Konzepte sein. In Drupal 8+ werden Hooks und Controller unterschiedlich verwendet, jedoch ist „Callback“ nicht die gängige Bezeichnung. Vorschlag: „Das Routing in Drupal kann komplex sein, da verschiedene Mechanismen zur Ausführung von Funktionen genutzt werden: moderne Controller-Klassen, ältere Hooks (von Programmierreihenfolgen) und gelegentlich Callback-Funktionen. Für IT-Einsteiger kann es unklar sein, welche Option in welchem Fall verwendet wird.“

**Problemstelle:** Original: „Zudem kann das Theme Preprocessing falsche Variablen bereitstellen, was die Ausgabe beeinflusst.“

**Warum problematisch:** „Theme Preprocessing“ wird ohne Erklärung erwähnt. Die Aussage, dass es „falsche Variablen bereitstellt“, ist unklar und könnte missverstanden werden: Meist entstehen Probleme durch falsche oder fehlende Variablen, nicht weil das Preprocessing selbst „falsch“ arbeitet. Vorschlag: „Beim sogenannten Theme Preprocessing – einem Schritt, in dem Variablen für die Darstellung vorbereitet werden – können Fehler entstehen, etwa wenn Variablen falsch definiert sind. Das kann die Ausgabe im Frontend beeinträchtigen.“

**Problemstelle:** Original: „Viele Entwickler empfehlen, den Cache neu zu bauen, um Fehler zu beheben. Dies ist jedoch häufig ein Versuch und Irrtum, da nicht immer klar ist, ob das Problem dadurch gelöst wird.“

**Warum problematisch:** Der Begriff „Cache neu bauen“ wird verwendet, aber nicht erklärt. Auch wird der Zweck von Cache-Leerung oder Cache-Updates nicht differenziert. Es klingt so, als sei Cache-Leerung unwirksam oder nur wilde Fehlersuche, was die Relevanz des Cache-Managements unterschätzt. Vorschlag: „Viele Entwickler empfehlen, den Cache zu leeren (Cache-Build durchzuführen), um Änderungen oder Fehler sichtbar zu machen. Dies hilft oft, ist jedoch keine Garantie für die Fehlerbehebung, da die Ursachen auch anderswo liegen können.“

**Problemstelle:** Original: „Fehler, etwa durch falsch eingerückte YAML-Dateien, werden von Drupal häufig erst spät oder gar nicht angezeigt.“

**Warum problematisch:** Drupal validiert YAML-Dateien mittlerweile beim Import und zeigt meist direkte Fehlermeldungen an, wenn die Syntax falsch ist. Es kann zwar passieren, dass manche Fehler schwer zu erkennen sind, aber „häufig erst spät oder gar nicht angezeigt“ ist eine übertriebene

Next Page

Wir sehen schon, der verhunzte Text, den wir als Beispiel für die Stiloptimierung erzeugt haben, wird mit keiner KI ein informativer Drupal-Artikel.

Aber er diente ja auch nur der Veranschaulichung, was alles mit diesem Submodul vom AI Modul möglich ist.

## Beispiel: Kurzfassungen für Meta-Descriptions oder Social-Posts

Naheliegen, dass man diese Kurzfassungen auch für Meta-Descriptions oder andere Zwecke, wie Social-Posts verwenden kann.

Dazu verwendet man das Summarise Text Plugin, bzw. eine Kopie davon mit gut angepasstem Prompt.

## Beispiel: Vorschläge für Taxonomie-Begriffe

- Wir fügen zum Testen der vorhandenen Taxonomie „Tags“ ein paar Schlagworte hinzu:

Name

✚ Drupal

✚ Typo3

✚ Wordpress

- Feld „Themen“ im Inhaltstyp „Artikel“ wird angelegt und mit Tags referenzieren. Wir lassen in dem Fall Mehrfachauswahl zu.
- „Suggest taxonomy tags“ in AI Content Suggestions Konfiguration aktivieren
- übliches Model wählen
- Prompt für unseren Fall einfügen:

Agiere wie ein erfahrener Redakteur einer IT-Fachzeitschrift.

Untersuche den folgenden Online-Artikel sorgfältig und ermittle alle Content-Management-Systeme (CMS), die im Text erwähnt werden – direkt, indirekt oder im Kontext dargestellt.

Gib ausschließlich die CMS-Begriffe als kommasetrennte Liste zurück, ohne weitere Erklärungen.

Beziehe nur CMS ein, die im Artikel eindeutig erkennbar genannt oder thematisch angesprochen werden.

Zum Einpflegen des Prompts öffnet sich ein neues Fenster. Man trägt nicht nur den Prompt ein, sondern gibt ihm auch einen Namen.

Vergesst bitte nicht, nach dem Speichern des Prompts auch noch die Konfiguration von AI Content Suggestions zu speichern.

Ich habe den Text in unserem Beispieldinhalt noch ergänzt um diesen Absatz:

Unterschiede zu Joomla

Joomla hat einen anderen Ansatz, als Drupal mit Inhalten umzugehen.  
Und Wordpress und Contao auch.

Rechts gibt es nun die neue Kategorie „Suggest taxonomy tags“. Hier kann man die Taxonomie „Tag“ angeben, in der die CMS Drupal, Typo 3 und Wordpress bereits gelistet sind.

Hier die beiden unterschiedlichen Konfigurationen von „Suggest taxonomy tags“ beim Inhalt selbst, NACH Klick auf das Button „Suggest taxonomy tags“.&br; Die gefundenen CMS-Namen erscheinen dann über diesem Button.

Im ersten Bild wurde die Taxonomie „Tag“ ausgewählt, damit wird nur Drupal gefunden.

Im zweiten Bild wurde die Taxonomie nicht ausgewählt, damit werden alle CMS gefunden, die im Text erwähnt werden.

Moderation state  
 Optimierter Text (AI)  
 Body

Select the field(s) you wish to send to the LLM

☒ Use source vocabulary  
 Check this box if you want to use a source vocabulary to suggest terms.

Choose vocabulary

Tags ▾

Optionally, select which vocabulary do you want to find the terms in.

☐ Use source vocabulary's full hierarchy  
 Check this box if you want to take into account the selected vocabulary's hierarchy, if such exists.

Drupal

Suggest taxonomy terms

^ Suggest taxonomy tags

Title  
 Moderation state  
 Optimierter Text (AI)  
 Body

Select the field(s) you wish to send to the LLM

☐ Use source vocabulary  
 Check this box if you want to use a source vocabulary to suggest terms.

Drupal, Joomla, Wordpress, Contao

Suggest taxonomy terms

Diese Begriffe können dann in das Autocomplete Feld „CMS“ mit Referenz auf Tag eingepflegt werden und entsprechend konfiguriert, kann man auch die neuen Begriffe hier direkt anlegen.

# Mit eigenem Modul eine Kopie des Plugins „Summarise text“ erzeugen

Wir haben das Plugin „Summarise text“ für unseren umfangreichen Prompt zum Optimieren des Textes zweckentfremdet.

Und in der Folge mussten wir das Titel Suggestions-Plugin für eine Zusammenfassung verwenden.

Das ist natürlich doof, wenn man mehrere Optimierungs-Workflows hintereinander schalten möchte.

Deshalb erstellen wir ein kleines Modul, das zum Testen ganz einfach sein darf.

Nennen wir es „ai\_optimiere\_text“ und legen dort zwei Dateien an:

1. „ai\_optimiere\_text.info.yml“ mit rudimentären Infos über das Modul

2. unter

/pfad/web/modules/custom/ai\_optimiere\_text/src/Plugin/AiContentSuggestions legen wir eine Datei „OptimizeText.php“ an.

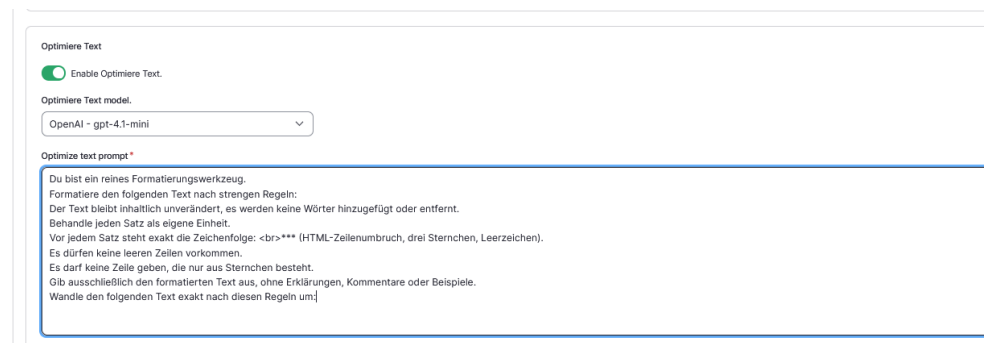
In diese Datei kopieren wir den Text aus

/pfad/web/modules/contrib/ai/modules/ai\_content\_suggestions/src/Plugin/AiContentSuggestions

Hier muss nur noch der Namespace angepasst werden.

Nach dem Cache leeren steht uns ein neues Plugin unter

/admin/config/ai/suggestions zur Verfügung:



Optimiere Text

☒ Enable Optimiere Text.

Optimiere Text model.

OpenAI - gpt-4.1-mini

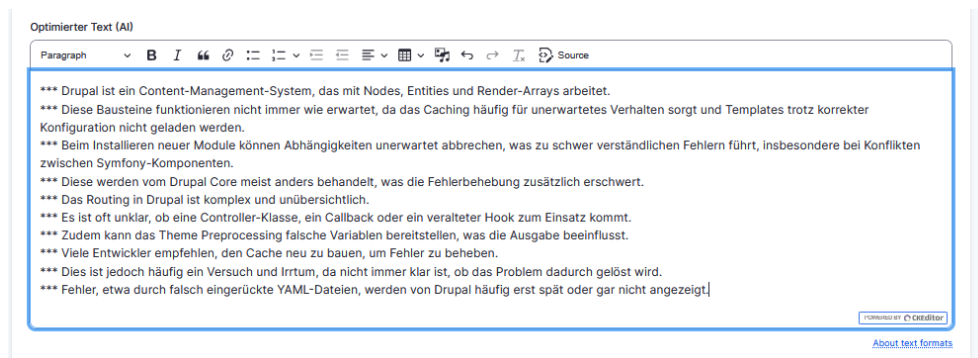
Optimize text prompt\*

Du bist ein reines Formatierungswerkzeug.  
Formatiere den folgenden Text nach strengen Regeln:  
Der Text bleibt inhaltlich unverändert, es werden keine Wörter hinzugefügt oder entfernt.  
Behandle jeden Satz als eigene Einheit.  
Vor jedem Satz steht exakt die Zeichenfolge: <br>\*\*\* (HTML-Zeilenumbruch, drei Sternchen, Leerzeichen).  
Es dürfen keine leeren Zeilen vorkommen.  
Es darf keine Zeile geben, die nur aus Sternchen besteht.  
Gib ausschließlich den formatierten Text aus, ohne Erklärungen, Kommentare oder Beispiele.  
Wandle den folgenden Text exakt nach diesen Regeln um!

Zum Testen wollte ich einen ganz einfachen Prompt verwenden, aber es zeigt sich, dass prompten immer der komplexe Teil der Arbeit mit Künstlicher Intelligenz ist. Die KI hat nicht verstanden, dass ich jeden Satz in einer eigenen Zeile stehen haben

möchte.

Aber mit Hilfe von ChatGPT habe ich den Prompt solange angepasst, bis er passt, auch wenn das Ergebnis natürlich keinen praktischen Sinn macht.



Damit wissen wir nun, dass unser eigenes Custom-Plugin funktioniert.

## Optimierung der Texte direkt im Feld mit dem Modul Field Widget Actions

Ich finde es ganz angenehm, dass Inhalte im Feld nicht automatisch überschrieben werden, sondern das AI Content Suggestions Modul Vorschläge liefert, die man per COPY + PASTE ins Zielfeld schreibt.

Wenn Euch das nervt, gibt es eine Lösung, die seit Version 1.2 des Modules AI zur Verfügung steht.

Aktiviert – wenn nicht bereits passiert – im Backend das Submodule „Field Widget Actions“.

Nicht irritieren lassen, es ist dort als deprecated gekennzeichnet. Das bedeutet: ab AI Version 2 wird es ein eigenes Modul, kein Submodule mehr sein.

Danach kann man im „Manage form display“ des Inhaltstyps die Aktion wählen, die durchgeführt werden soll.

In unserem Fall „Content Suggestions“ mit eigenem Prompt.

Widget settings: **Text area (multiple rows)**

Rows\*

5

Text editors may override this setting.

Placeholder

Text that will be shown inside the field until a value is entered. This hint is usually a sample value or a brief description of the content of the field.

- None -

**AI Automators**

Automator Text Suggestion

**AI Content Suggestions**

Content Suggestion with prompt

Content Suggestion with prompt

Add action

Bitte macht nicht den Fehler wie ich, direkt auf Update zu klicken, sondern zuerst auf „Add action“.

Dann erscheint eine umfangreiche Konfigurationsoberfläche und wir können hier einen eigenen Prompt einfügen.

Wir müssen also kein eigenes Modul programmieren, wenn wir mehrere unterschiedliche Optimierungs-Prompts bei unterschiedlichen Inhaltstypen benötigen.

Wir können auch auf Inhaltsebene unterschiedliche LLM verwenden, je nach Qualitätsanspruch und Komplexität der Aufgabe.

OpenAI-gpt-4.1 hat bei mir nicht funktioniert, sondern OpenAI-gpt-4.1-mini.

Wir passen unseren umfangreichen Prompt aus der Verwaltungsoberfläche von AI Content Suggestions beim Plugin „Summarise Text“ an und tragen ihn hier ein.

Dazu muss man mehrere Dinge wissen:

- Das Modul liefert bei Ausführung im Feld eine Anzahl von Vorschlägen, die man im Prompt beeinflussen kann. Es sollten nicht zu viele sein, weil es sonst unübersichtlich wird.
- Das Modul gibt html nicht gerendert aus. Das müsste man mit einer Programmierung ändern. Zumindest habe ich keine Möglichkeit gefunden.

- Man muss im Prompt ausdrücklich fordern „KEINE Markdown-Zeichen wie ##“.
- Das Modul funktioniert an dieser Stelle ohne Token auf den Feldnamen nur sehr schlecht, obwohl man beim Inhalt selbst das Feld angibt.

Deshalb sieht der Prompt jetzt wie folgt aus:

Du erhältst einen deutschen Text aus dem Feld [node:field\_body:value], der stilistisch überarbeitet werden soll.

Bitte überarbeite den Text, ohne den Inhalt zu verändern, und überprüfe ihn auf:

- Rechtschreibung
- Grammatik
- Satzbau
- Lesbarkeit
- Stil
- sprachliche Präzision
- Tonalität
- Struktur

Die Tonalität soll der Schreibweise auf

<https://www.montviso.de/blog/kuenstliche-intelligenz-drupal-textgenerie...> entsprechen:

klar, ruhig, sachlich, erklärend, strukturiert, ohne Übertreibungen oder Werbesprache.

WICHTIG:

- Die Ansprache (Du oder Sie) darf NICHT verändert werden.
- Bestehende Absätze, Listen und Überschriften sollen erhalten bleiben.
- Verwende ausschließlich echtes HTML (<h2>, <p>, <ul>, <li>, <strong> ...), KEINE Markdown-Zeichen wie ## oder \*\*.
- Wandle vorhandene oder fehlende Überschriften sinnvoll in <h2>-Tags um.
- Gib nur den reinen HTML-Text zurück – keine Erklärungen oder Kommentare.
- Der Text soll inhaltlich identisch bleiben, weder kürzer noch länger

werden.

- Keine neuen Informationen, keine Icons oder Sonderzeichen hinzufügen.

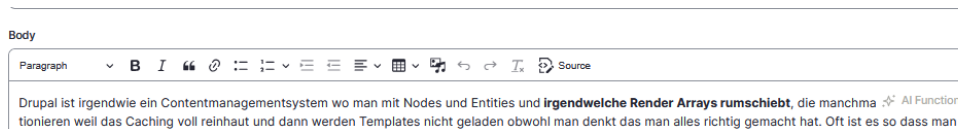
GIB GENAU 4 VORSCHLÄGE zurück.

Jeder Vorschlag soll reines HTML sein, bestehend aus `<p>`, `<h2>`,

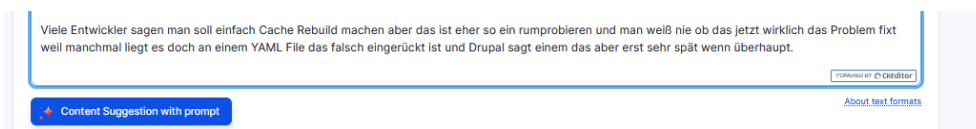
`<strong>` usw., ohne Markdown oder zusätzliche Formatierungshinweise.

Wie sieht das dann am Inhalt aus? Zuerst sehe ich kein Button.

Dann entecke ich ausgegraut am rechten oberen Rand „AI Function“.



Nach Klick auf diesen Schriftzug erscheint unter dem Feld ein Button:



Mit Klick auf das Button öffnet sich ein Fenster, das den geänderten Text in der angegebenen Anzahl von Varianten zeigt. In unserem Fall also vier Varianten. Bei Klick auf den Pfeil am Ende des gewünschten Textes, wird dieser in das Feld übertragen.

Allerdings nicht gerendert. Wir müssen ihn dann noch händisch im CKEditor via „Quelle“ einfügen, damit er im Frontend mit Formatierung und ohne HTML Tags erscheint.

<p><strong>Drupal-Architektur und Caching-Probleme</strong><br>Drupal ist ein Content-Management-System, bei dem man mit Nodes, Entities und <strong>Render-Arrays arbeitet</strong>. Diese Render-Arrays funktionieren manchmal nicht, weil das Caching dazwischenfunkt. Dadurch werden Templates nicht geladen, obwohl man denkt, alles richtig gemacht zu haben.</p><p><strong>Abhängigkeiten und Composer-Konflikte</strong><br>Oft passiert es, dass nach der Installation eines Moduls plötzlich eine Abhängigkeit fehlt oder Composer einen schwer verständlichen Fehler ausgibt. Besonders problematisch sind Konflikte zwischen Symfony-Komponenten, die Drupal Core anders erwartet.</p><p><strong>Routing und Theme-Preprocessing</strong><br>Das Routing ist oft verwirrend, weil unklar ist, ob eine Controller-Klasse, ein Callback oder ein alter Hook Menu greift. Zusätzlich lädt das Theme Preprocessing manchmal falsche Variablen ein.</p><p><strong>Cache-Rebuild und Fehlersuche</strong><br>Viele Entwickler empfehlen einfach einen Cache-Rebuild, was aber eher ein Ausprobieren ist. Oft weiß man nicht, ob das wirklich das Problem löst, weil Fehler auch an falsch eingerückten YAML-Dateien liegen können, die Drupal erst spät oder gar nicht anzeigt.</p>

&gt;

<p><strong>Grundlagen und Herausforderungen bei Drupal</strong><br>Drupal ist ein Content-Management-System, das mit Nodes, Entities und <strong>Render-Arrays arbeitet</strong>. Diese Render-Arrays funktionieren gelegentlich nicht, weil das Caching Probleme verursacht. Infolgedessen werden Templates nicht geladen, obwohl alles korrekt eingestellt scheint.</p><p><strong>Probleme mit Modulen und Abhängigkeiten</strong><br>Häufig kommt es vor, dass nach der Installation eines Moduls eine Abhängigkeit verloren geht oder Composer einen unverständlichen Fehler anzeigt. Konflikte zwischen Symfony-Komponenten, die Drupal Core anders erwartet, sind dabei eine häufige Ursache.</p><p><strong>Unklarheiten im Routing und Theme-Preprocessing</strong><br>Das Routing ist oft komplex, weil nicht immer ersichtlich ist, ob eine Controller-Klasse, ein Callback oder ein veralteter Hook Menu verwendet wird. Zudem kann das Theme Preprocessing falsche Variablen übergeben.</p><p><strong>Cache-Neuerstellung und Fehlerdiagnose</strong><br>Viele Entwickler empfehlen einen Cache-Rebuild, der jedoch meist nur Versuch und Irrtum ist. Häufig liegen die Probleme auch an falsch eingerückten YAML-Dateien, deren Fehler Drupal erst spät oder gar nicht meldet.</p>

&gt;

<p><strong>Überblick über Drupal und Render Array Probleme</strong><br>Drupal ist ein Content-Management-System, bei dem man mit Nodes, Entities und <strong>Render-Arrays arbeitet</strong>, die manchmal aufgrund des Cache-Systems nicht korrekt funktionieren. Dadurch werden Templates nicht geladen, obwohl man alle Einstellungen richtig vorgenommen hat.</p><p><strong>Abhängigkeitsprobleme und Composer-Fehler</strong><br>Es ist häufig so, dass nach der Installation eines Moduls eine Abhängigkeit plötzlich fehlt oder Composer einen schwer verständlichen Fehler anzeigt. Insbesondere Konflikte zwischen Symfony-Komponenten, die Drupal Core anders erwartet, führen zu Problemen.</p><p><strong>Routing und Theme-Preprocessing</strong><br>Das Routing ist oft schwer nachzuvollziehen, da unklar bleibt, ob eine Controller-Klasse, ein Callback oder ein alter Hook Menu greift. Außerdem lädt das Theme Preprocessing teils falsche Variablen.</p><p><strong>Cache-Rebuild und Fehlersuche</strong><br>Viele Entwickler empfehlen, einfach einen Cache-Rebuild durchzuführen. Dies ist jedoch meist ein Versuch, da nicht sicher ist, ob das Problem dadurch behoben wird. Fehler können auch an falsch eingerückten YAML-Dateien liegen, die Drupal erst spät oder gar nicht meldet.</p>

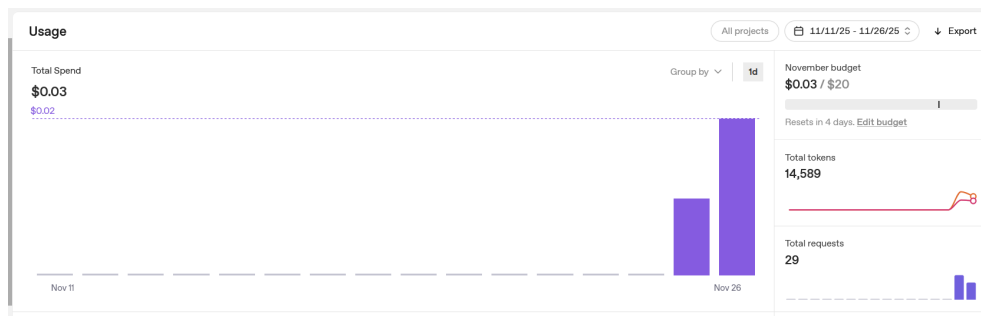
&gt;

<p><strong>Drupal Grundlagen und Herausforderungen</strong><br>Drupal ist ein Content-Management-System, das mit Nodes, Entities und <strong>Render-Arrays arbeitet</strong>. Diese Render-Arrays funktionieren nicht immer, da das Caching gelegentlich Probleme verursacht und Templates nicht geladen werden, obwohl alles korrekt zu sein scheint.</p><p><strong>Modulinstallation und Abhängigkeitsprobleme</strong><br>Häufig tritt auf, dass nach der Installation eines Moduls eine notwendige Abhängigkeit verloren geht oder Composer Fehlermeldungen produziert, die schwer verständlich sind, insbesondere bei Konflikten zwischen Symfony-Komponenten, die Drupal Core anders erwartet.</p><p><strong>Routing-Komplexität und Theme Preprocessing</strong><br>Das Routing ist oft unübersichtlich, weil nicht klar ist, ob eine Controller-Klasse, ein Callback oder ein veralteter Hook Menu verwendet wird. Zusätzlich lädt das Theme Preprocessing gelegentlich falsche Variablen.</p><p><strong>Cache-Rebuild als

&gt;

## Was kostet der Spaß?

Zum Abschluss eine Übersicht, was die Tests für diesen Blog an Tokens verbraucht haben.



Ich habe also an zwei Tagen 0,03 Cent verbraucht. Solange man die Funktionen nicht einem größeren Personenkreis oder gar Gästen zur Verfügung stellt, halten sich die Kosten in Grenzen.

## Weiterführende Links:

- AI Content Suggestions – offizielle Modul-Dokumentation  
[http://project.pages.drupalcode.org/ai/1.0.x/modules/ai\\_content\\_suggestions/](http://project.pages.drupalcode.org/ai/1.0.x/modules/ai_content_suggestions/)
- AI (Artificial Intelligence) – Projektseite auf drupal.org  
<https://www.drupal.org/project/ai>
- Issue „AI Content Suggestions: Allow summarize, analyze, suggest title and taxonomy terms...” (zeigt typische geplante/gedachte Use-Cases)  
<https://www.drupal.org/project/ai/issues/3489572>
- Issue „Restore permission for accessing AI Content Suggestion features“ (Hinweis auf Permission)  
<https://www.drupal.org/project/ai/issues/3498628>
- Drupal AI Ecosystem Part 5: AI Content Suggestions:  
<https://opensenselabs.com/blog/drupal-ai-module/ai-content-suggestions>