KI-gestützte Rechnungsdaten-Extraktion

Ein vollständiger Prompt-Engineering-Workflow: von PDF zu strukturierten Daten

Portfolio-Projekt · Prompt Engineering & Datenextraktion · Erstellt mit Claude / GPT-kompatiblen LLMs

Kurzfassung: Dieser Workflow wandelt unstrukturierten Rechnungstext automatisch in saubere, maschinenlesbare JSON-Daten um — Lieferant, Rechnungsnummer, Datum, Einzelposten, Beträge und Steuer. Das Herzstück ist ein sorgfältig entworfener Extraktions-Prompt, der zuverlässige Ergebnisse über viele Rechnungsformate hinweg liefert und mit fehlenden oder uneindeutigen Feldern robust umgeht.

1. Das Problem

Kleine und mittlere Unternehmen erhalten Rechnungen in dutzenden unterschiedlichen Layouts — als PDF, gescannt, per E-Mail. Die Daten manuell in die Buchhaltung zu übertragen ist langsam, teuer und fehleranfällig. Ein einzelner Mitarbeiter braucht oft 3–5 Minuten pro Rechnung; bei hunderten Rechnungen im Monat summiert sich das zu erheblichem Aufwand.

Ziel: Aus beliebigem Rechnungstext automatisch ein einheitliches, strukturiertes Datenobjekt erzeugen, das direkt in Buchhaltung, Datenbank oder Tabelle übernommen werden kann — ohne dass für jedes Layout neue Regeln programmiert werden müssen.

VORHER

- Manuelles Abtippen, 3–5 Min./Rechnung
- Tippfehler bei Beträgen und Steuersätzen
- Jedes Lieferanten-Layout muss einzeln verstanden werden

NACHHER

- Sekunden pro Rechnung, layoutunabhängig
- Konsistente Felder & Validierung
- Direkt weiterverarbeitbares JSON

2. Der Lösungsansatz

Der Workflow besteht aus drei Schritten. Schritt 1 und 3 sind technische Hülle (Text aus dem PDF holen, Ergebnis speichern); der eigentliche Wert — und das Kernstück dieses Projekts — liegt in Schritt 2, dem Extraktions-Prompt.

Schritt	Aufgabe	Werkzeug
1 · Eingabe	Rohtext der Rechnung bereitstellen (aus PDF extrahiert oder per OCR aus Scan)	PDF-Textextraktion / OCR
2 · Extraktion	Der LLM liest den Text und gibt strukturiertes JSON nach festem Schema zurück	**Engineered Prompt**
3 · Ausgabe	JSON validieren und in Tabelle / Datenbank / Buchhaltung übernehmen	JSON-Validierung

Designentscheidung: Der Prompt erzwingt eine feste JSON-Struktur und klare Regeln für fehlende Werte. Dadurch ist die Ausgabe vorhersehbar und lässt sich maschinell weiterverarbeiten — der entscheidende Unterschied zwischen einer netten Spielerei und einem produktiv nutzbaren Workflow.

3. Das Kernstück: Der Extraktions-Prompt

Dieser Prompt wird dem Modell als System-Anweisung gegeben, gefolgt vom Rechnungstext. Jede Regel adressiert ein konkretes Problem, das in echten Rechnungen auftritt.

```
Du bist ein präziser Datenextraktions-Assistent für Rechnungen.
Deine Aufgabe: Extrahiere die unten definierten Felder aus dem
bereitgestellten Rechnungstext und gib AUSSCHLIESSLICH ein gültiges
JSON-Objekt zurück – keine Erklärungen, kein Markdown, kein Fließtext.

REGELN:
1. Halte dich exakt an das vorgegebene Schema und die Feldnamen.
2. Wenn ein Wert nicht eindeutig im Text steht, setze ihn auf null.
   Rate oder erfinde NIEMALS Werte.
3. Beträge: gib reine Zahlen ohne Währungssymbol oder Tausender-
   trennzeichen aus (z. B. 1234.50). Verwende den Punkt als Dezimal-
   trennzeichen.
4. Daten: konvertiere immer in das Format JJJJ-MM-TT.
5. Die Währung wird separat im Feld "currency" als ISO-Code
   ausgegeben (z. B. "EUR", "USD"). Bei Unklarheit: null.
6. Erfasse jeden Einzelposten separat im Array "line_items".
7. Prüfe rechnerisch: subtotal + tax_amount sollte total ergeben.
   Stimmt es nicht, setze "math_check" auf false, sonst true.
8. Gib keine Felder aus, die nicht im Schema definiert sind.

SCHEMA:
{
  "vendor_name": string | null,
  "invoice_number": string | null,
  "invoice_date": "JJJJ-MM-TT" | null,
  "due_date": "JJJJ-MM-TT" | null,
  "currency": string | null,
  "line_items": [
    { "description": string, "quantity": number | null,
      "unit_price": number | null, "amount": number | null }
  ],
  "subtotal": number | null,
  "tax_rate": number | null,
  "tax_amount": number | null,
  "total": number | null,
  "math_check": boolean
}

RECHNUNGSTEXT:
"""
{hier wird der extrahierte Rechnungstext eingefügt}
"""
```

Warum dieser Prompt so gebaut ist

Regel	Welches reale Problem sie löst
Nur JSON, kein Fließtext	Ohne diese Regel kommentiert das Modell oft seine Antwort — das macht die Ausgabe maschinell unbrauchbar.
Fehlende Werte = null, nie raten	Verhindert „Halluzinationen". Eine erfundene Rechnungsnummer ist gefährlicher als ein leeres Feld.
Zahlen ohne Symbole, Punkt als Dezimaltrenner	Deutsche Rechnungen nutzen oft „1.234,50 €" — ohne Normierung scheitert jede spätere Berechnung.
Datum in JJJJ-MM-TT	Rechnungen mischen Formate (01.03.2026, March 1 2026). Ein einheitliches Format ist Pflicht für Datenbanken.
math_check	Eine eingebaute Selbstkontrolle, die Extraktionsfehler oder schlechte OCR sichtbar macht.

4. Worked Example

Ein realistischer Test mit einer typischen deutschen Rechnung. Links der Eingabetext (wie er aus dem PDF kommt), rechts das vom Workflow erzeugte Ergebnis.

EINGABE (RECHNUNGSTEXT)

```
Muster GmbH
Industriestraße 12, 44801 Bochum

RECHNUNG  Nr. 2026-0457
Datum: 03.03.2026        Fällig: 17.03.2026

Pos  Beschreibung              Menge   Einzelpreis   Betrag
1    Beratung (Stunde)          10      85,00 €      850,00 €
2    Lizenz Software Basic       1     120,00 €      120,00 €

Zwischensumme:                                      970,00 €
zzgl. 19% USt.:                                     184,30 €
Gesamtbetrag:                                     1.154,30 €
```

AUSGABE (VOM WORKFLOW ERZEUGTES JSON)


```
{
  "vendor_name": "Muster GmbH",
  "invoice_number": "2026-0457",
  "invoice_date": "2026-03-03",
  "due_date": "2026-03-17",
  "currency": "EUR",
  "line_items": [
    { "description": "Beratung (Stunde)", "quantity": 10, "unit_price": 85.00,
"amount": 850.00 },
    { "description": "Lizenz Software Basic", "quantity": 1, "unit_price": 120.00,
"amount": 120.00 }
```

```
  ],
  "subtotal": 970.00,
  "tax_rate": 19,
  "tax_amount": 184.30,
  "total": 1154.30,
  "math_check": true
}
```

Beachte: Die Beträge sind normiert (Punkt als Dezimaltrenner, kein €-Symbol), das Datum ist in ISO-Format umgewandelt, und `math_check` bestätigt, dass 970,00 + 184,30 = 1.154,30 aufgeht.

5. Umgang mit Sonderfällen

Echte Rechnungen sind selten sauber. Der Workflow ist auf die häufigsten Stolperfallen vorbereitet:

Sonderfall	Verhalten des Workflows
Fehlendes Fälligkeitsdatum	`due_date` wird `null` — kein Rateversuch.
Fremdwährung (z. B. $)	`currency` erkennt den ISO-Code; Beträge bleiben unverändert.
Rechnung ohne ausgewiesene Steuer	`tax_rate` und `tax_amount` werden `null` oder 0 laut Text.
OCR-Fehler bei Beträgen	`math_check` wird `false` → die Rechnung wird zur manuellen Prüfung markiert.
Mehrseitige Rechnung	Alle Posten werden zusammengeführt; Summen von der letzten Seite genommen.

Qualitätssicherung: Das Feld `math_check` ist bewusst eingebaut, damit fehlerhafte Extraktionen automatisch auffallen, statt unbemerkt in die Buchhaltung zu fließen. So lassen sich 95 % der Rechnungen automatisiert verarbeiten, während nur die markierten Fälle einen menschlichen Blick brauchen.

6. Eingesetzte Fähigkeiten

Prompt Engineering · Strukturierte Ausgaben (JSON) · Datenextraktion · Schema-Design · Edge-Case-Handling · Datennormalisierung · ChatGPT / Claude

7. Möglicher Ausbau

Dieser Prompt ist die Grundlage. In Folgeschritten lässt sich der Workflow erweitern um: automatische OCR-Anbindung für gescannte Rechnungen, direkten Export nach Excel oder in ein Buchhaltungssystem, sowie Batch-Verarbeitung ganzer Rechnungsordner. Der Prompt selbst bleibt dabei das stabile Herzstück — er ist bewusst so entworfen, dass er ohne Änderung mit verschiedenen Dokumenttypen funktioniert.